
GeoNode Documentation

Release 3.0

GeoNode Development Team

Sep 07, 2021

ABOUT

1	Table of contents	3
1.1	What is GeoNode	3
1.1.1	Showcase	3
1.1.2	Most useful links	3
1.2	Licensing	4
1.3	Current Version and Features	4
1.4	Get in touch with the community	4
1.5	Roadmap	4
1.6	GeoNode Basics	5
1.6.1	<i>With GeoNode, non-specialized users can share data and create interactive maps.</i>	5
1.6.2	Geospatial data storage	5
1.6.3	Data mixing, maps creation	7
1.6.4	GeoNode as a building block	7
1.6.5	Convinced! Where do I sign?	9
1.7	Supported Browsers	9
1.7.1	Internet Explorer	10
1.7.2	Testing on Internet Explorer	10
1.8	Online Demo	11
1.9	Quick Installation Guide	14
1.9.1	Quick Installation Guide	14
1.10	GeoNode Users Guide	16
1.10.1	Accounts and User Profile	16
1.10.2	Interacting with Users and Groups	25
1.10.3	Data	35
1.10.4	Managing Documents	46
1.10.5	Managing Layers	68
1.10.6	Managing Maps	108
1.10.7	Publishing Data	155
1.10.8	Using GeoNode with Other Applications	155
1.11	GeoNode Core	157
1.11.1	Overview	157
1.11.2	Ubuntu 18.04	157
1.11.3	CentOS 7.0	177
1.11.4	Docker	177
1.12	GeoNode Project	186
1.12.1	Overview	186
1.12.2	Ubuntu 18.04	187
1.12.3	Docker	190
1.13	SPCGeoNode	191
1.13.1	Overview	191

1.13.2	Prerequisites	192
1.13.3	Usage	192
1.13.4	Upgrade	193
1.13.5	Development vs Production	193
1.13.6	Releases	193
1.13.7	FAQ	194
1.14	GeoNode Settings	197
1.14.1	Settings	197
1.15	Customize the Look and Feel	237
1.15.1	GeoNode Themes	237
1.15.2	Theming your GeoNode Project	237
1.16	GeoNode permissions	244
1.16.1	Permissions	244
1.17	Read-Only and Maintenance Mode	255
1.17.1	Read-Only and Maintenance Modes	255
1.18	Monitoring	259
1.18.1	Monitoring	259
1.18.2	Monitoring: API	278
1.18.3	Monitoring: User Analytics	299
1.18.4	Monitoring: Notifications	311
1.18.5	Web API	312
1.19	GeoNode Backup and Restore	320
1.19.1	Full GeoNode Backup & Restore	320
1.20	Viewer and Hooksets	327
1.20.1	GXP	327
1.20.2	MapStore 2	327
1.20.3	Leaflet	327
1.21	GeoNode Components and Architecture	327
1.21.1	Overview	327
1.21.2	Django	327
1.21.3	WebServers	327
1.21.4	GeoServer	328
1.21.5	Databases	328
1.21.6	OAuth2 Security: Authentication and Authorization	328
1.22	Hardening GeoNode	378
1.22.1	Publish on HTTPS	378
1.22.2	OAuth2 Fixtures Update and Base URL Migration	378
1.22.3	GeoNode Security Subsystem	378
1.22.4	OAuth2 Tokens and Sessions	379
1.23	Social Login	379
1.23.1	Overview	379
1.23.2	Configuration	379
1.23.3	Linkedin	379
1.23.4	Facebook	379
1.24	GeoNode Django Contrib Apps	379
1.24.1	Geonode auth via LDAP	379
1.24.2	Geonode Logstash for centralized monitoring/analytics	384
1.25	GeoNode Admins Guide	391
1.25.1	Accessing the panel	391
1.25.2	Reset or Change the admin password	392
1.25.3	Simple Theming	393
1.25.4	Add a new user	406
1.25.5	Activate/Disable a User	409
1.25.6	Change a User password	410

1.25.7	Promoting a User to Staff member or superuser	410
1.25.8	Creating a Group	412
1.25.9	Managing a Group	417
1.25.10	Group based advanced data workflow	428
1.25.11	Manage profiles using the admin panel	429
1.25.12	Manage layers using the admin panel	431
1.25.13	Manage the maps using the admin panel	433
1.25.14	Manage the documents using the admin panel	435
1.25.15	Manage the base metadata choices using the admin panel	436
1.25.16	Announcements	445
1.25.17	Menus, Items and Placeholders	448
1.25.18	OAuth2 Access Tokens	456
1.26	GeoNode Management Commands	461
1.26.1	Migrate GeoNode Base URL	461
1.26.2	Update Permissions, Metadata, Legends and Download Links	463
1.26.3	Loading Data into GeoNode	467
1.26.4	Create Users and Super Users	498
1.26.5	Batch Sync Permissions	502
1.26.6	Delete Certain GeoNode Resources	504
1.27	Changing the default Languages	506
1.27.1	Changing the Default Language	506
1.27.2	GeoNode Configuration	507
1.27.3	Additional Steps	507
1.27.4	Restart	508
1.28	GeoNode Upgrade from older versions	508
1.28.1	Upgrade from 2.8.x	508
1.28.2	Upgrade from 2.7.x	508
1.28.3	Upgrade from 2.6.x	508
1.28.4	Upgrade from 2.4.x	508
1.29	GeoNode Async Signals	510
1.29.1	Supervisord and Systemd	510
1.29.2	Celery	510
1.29.3	Rabbitmq and Redis	510
1.30	GeoNode add a thesaurus	510
1.30.1	Loading a thesaurus	510
1.30.2	Configure a thesaurus in GeoNode	511
1.30.3	Apply a thesaurus to a resource	511
1.31	Participate in the Discussion	511
1.31.1	Join the community, ask for help or report bugs	511
1.32	Write Documentation	512
1.32.1	How to contribute to GeoNode's Documentation	512
1.33	Provide Translations	514
1.33.1	Contribute to Translations	514
1.34	Write Code	520
1.35	Frontend Development	520
1.35.1	Frontend development	520
1.36	How to Develop	522
1.36.1	Start to develop with Docker	522
1.36.2	How to Install GeoNode-Core for development	523
1.36.3	How to run GeoNode Core for development	531
1.36.4	How to run GeoNode Project for development	531
1.36.5	Workshops	531

Welcome to GeoNode's Documentation.

GeoNode is an Open Source, Content Management System (CMS) for geospatial data. It is a web-based application and platform for developing geospatial information systems (GIS) and for deploying spatial data infrastructures (SDI).

TABLE OF CONTENTS

1.1 What is GeoNode



GeoNode is a geospatial content management system, a platform for the management and publication of geospatial data. It brings together mature and stable open-source software projects under a consistent and easy-to-use interface allowing non-specialized users to share data and create interactive maps.

Data management tools built into GeoNode allow for integrated creation of data, metadata, and map visualization. Each dataset in the system can be shared publicly or restricted to allow access to only specific users. Social features like user profiles and commenting and rating systems allow for the development of communities around each platform to facilitate the use, management, and quality control of the data the GeoNode instance contains.

It is also designed to be a flexible platform that software developers can extend, modify or integrate against to meet requirements in their own applications.

1.1.1 Showcase

A handful of other Open Source projects extend GeoNode's functionality by tapping into the re-usability of Django applications. Visit our gallery to see how the community uses GeoNode: [GeoNode Showcase](#).

The development community is very supportive of new projects and contributes ideas and guidance for newcomers.

For a live demo see also [Online Demo](#)

1.1.2 Most useful links

General

- Project homepage: <https://geonode.org>
- Repository: <https://github.com/GeoNode/geonode>
- Official Demo: <http://master.demo.geonode.org>
- GeoNode Wiki: <https://github.com/GeoNode/geonode/wiki>
- Issue tracker: <https://github.com/GeoNode/geonode-project/issues>

In case of sensitive bugs like security vulnerabilities, please contact a GeoNode Core Developer directly instead of using issue tracker. We value your effort to improve the security and privacy of this project!

Related projects

- GeoNode Project: <https://github.com/GeoNode/geonode-project>
- GeoNode at Docker: <https://hub.docker.com/u/geonode>
- GeoNode OSGeo-Live: <https://live.osgeo.org/en/>

1.2 Licensing

GeoNode is Copyright 2018 Open Source Geospatial Foundation (OSGeo).

GeoNode is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. GeoNode is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with GeoNode. If not, see <http://www.gnu.org/licenses>.

1.3 Current Version and Features

GeoNode current version: 2.10.3

Main Features: [State of GeoNode 2.10](#)

1.4 Get in touch with the community

GeoNode is an open source project and contributors are needed to keep this project moving forward. Learn more on how to contribute on our [Community Bylaws](#).

- User Mailing List: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-users>
- Developer Mailing List: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-devel>
- Gitter Chat: <https://gitter.im/GeoNode/general>

1.5 Roadmap

GeoNode's development roadmap is documented in a series of GeoNode Improvement Projects (GNIPS). They are documented at [GeoNode Wiki](#).

GNIPS are considered to be large undertakings which will add a large amount of features to the project. As such they are the topic of community discussion and guidance.

The community discusses these on the developer mailing list: <http://lists.osgeo.org/pipermail/geonode-devel/>

1.6 GeoNode Basics



is a platform for the management and publication of geospatial data. It brings together mature open-source software projects under an easy to use interface.

1.6.1 *With GeoNode, non-specialized users can share data and create interactive maps.*

1.6.2 Geospatial data storage

GeoNode allows users to upload vector data (currently shapefiles, json, csv, kml and kmz) and raster data in their original projections using a web form.

Vector data is converted into geospatial tables on a DB, satellite imagery and other kinds of raster data are retained as GeoTIFFs.

Special importance is given to standard metadata formats like ISO 19139:2007 / ISO 19115 metadata standards.

As soon as the upload is finished, the user can fill the resource metadata in order to make it suddenly available through the [CSW](#) (OGC Catalogue Service) endpoints and APIs.

Users may also upload a metadata XML document (ISO, FGDC, and Dublin Core format) to fill key GeoNode metadata elements automatically.

Similarly, GeoNode provides a web based styler that lets the users to change the data portrayals and preview the changes at real time.

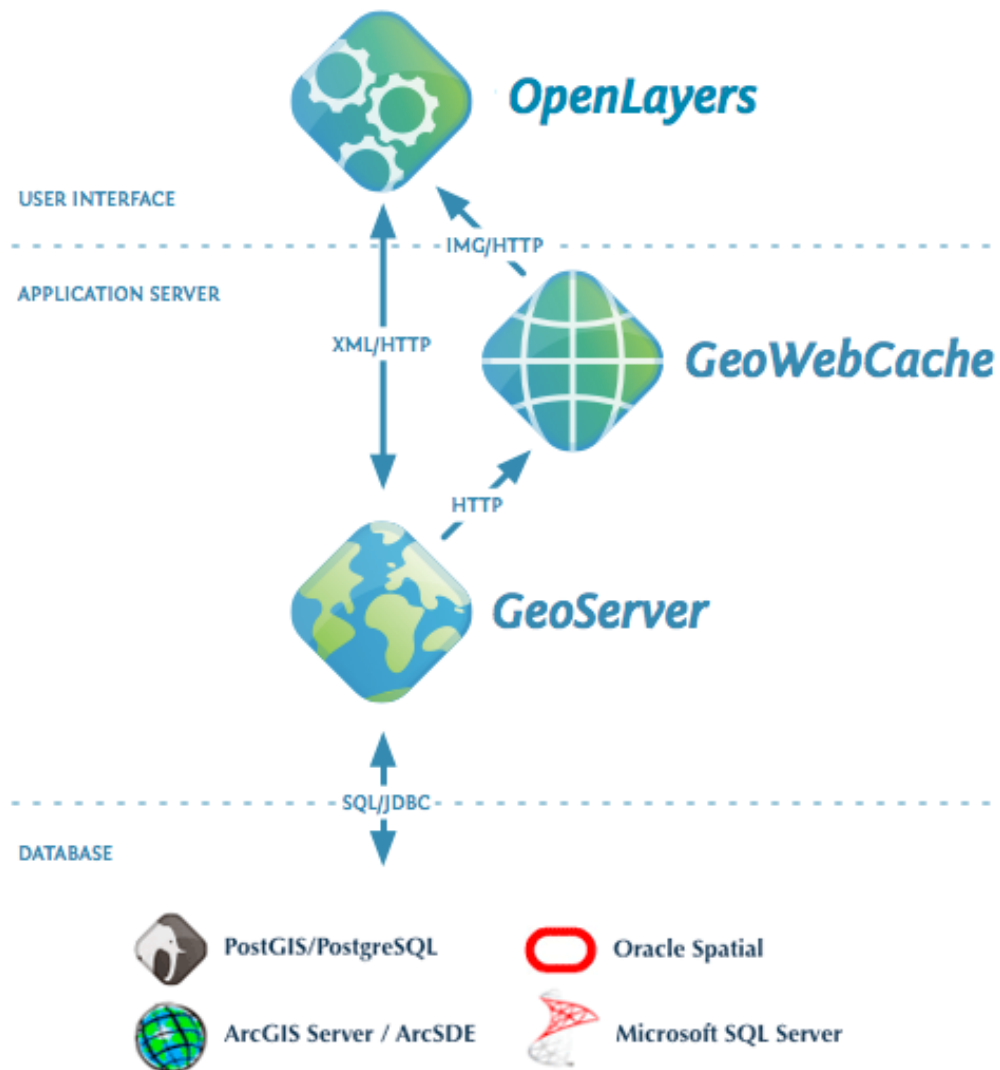
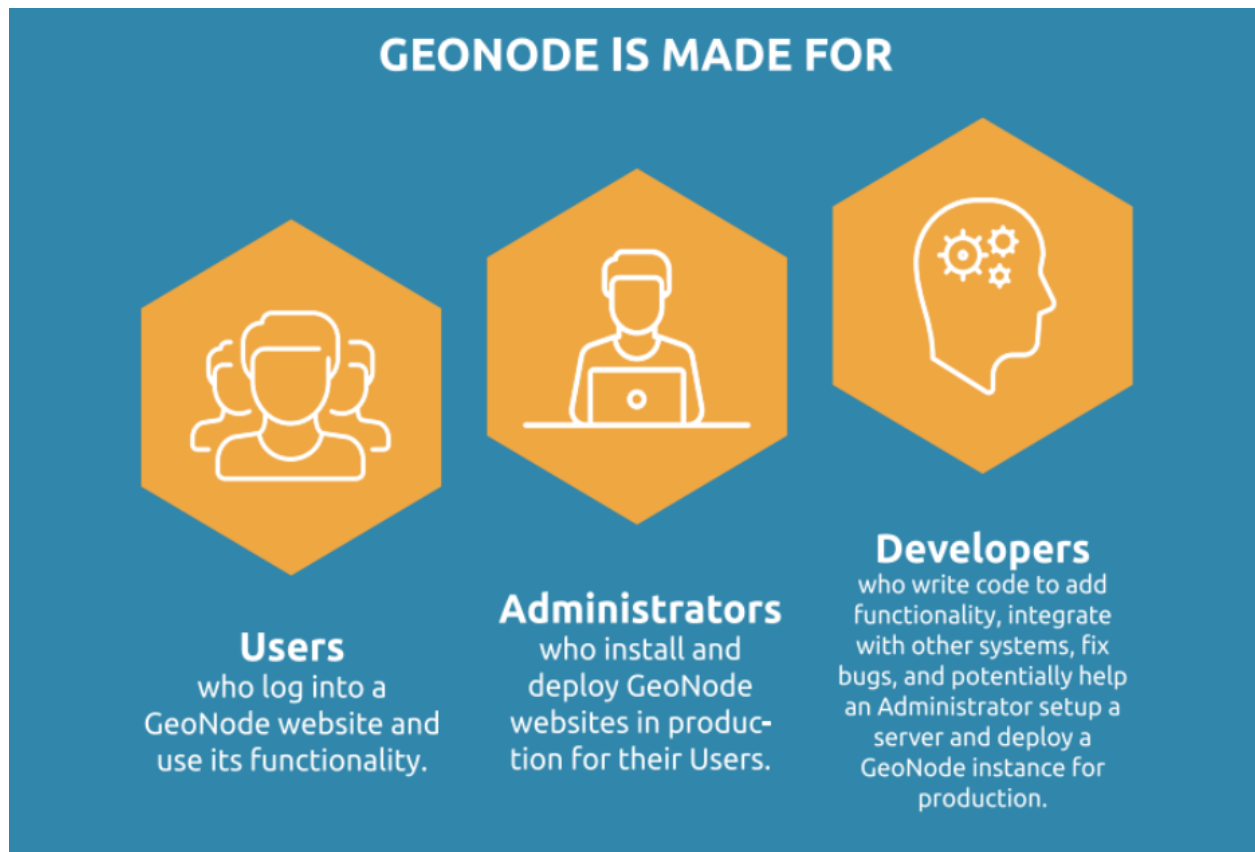


Fig. 1: *GeoNode simplified architecture*



1.6.3 Data mixing, maps creation

Once the data has been uploaded, GeoNode lets the user search for it geographically or via keywords in order to create fancy maps.

All the layers are automatically re-projected to web Mercator for maps display, making it possible to use different popular base layers, like Open Street Map, Google Satellite or Bing layers.

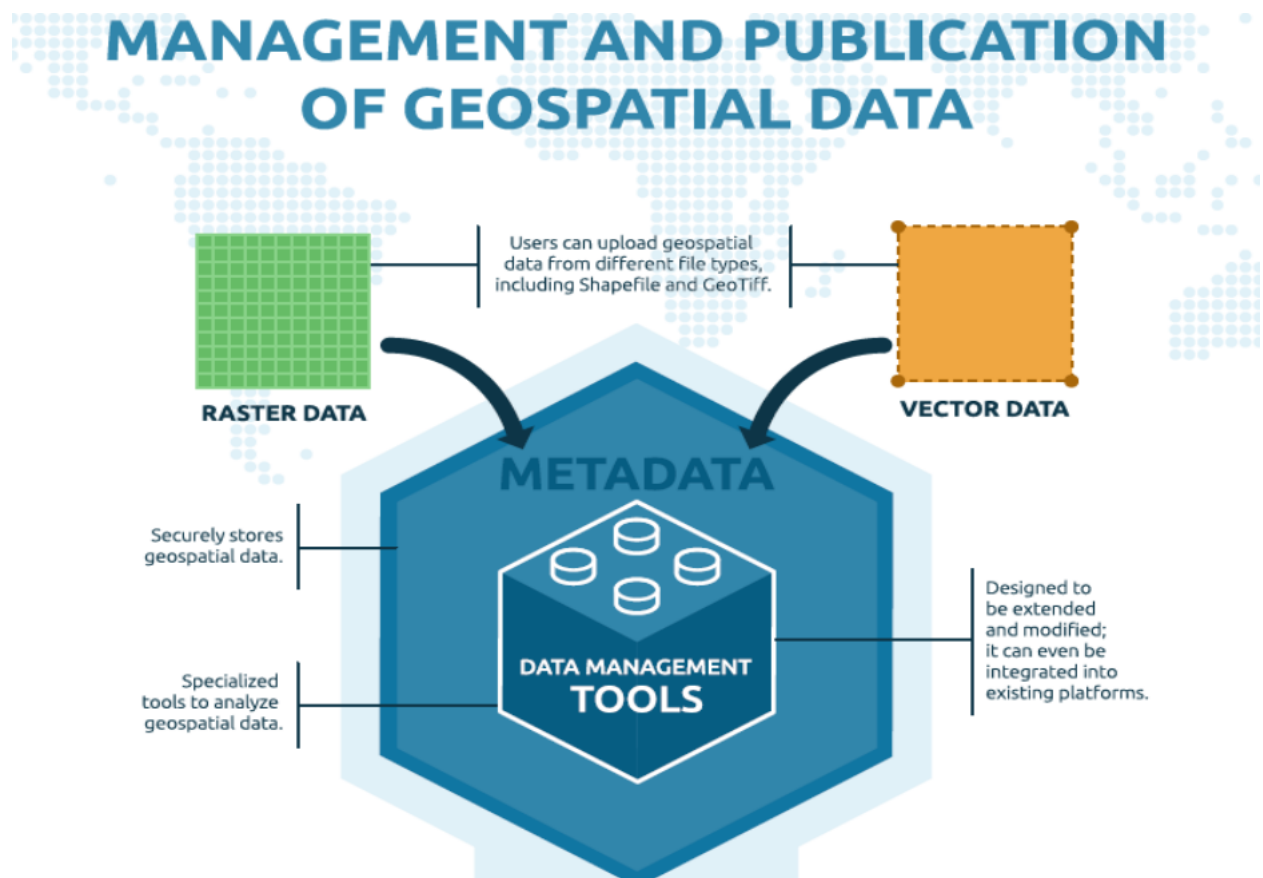
Once the maps are saved, it is possible to embed them in any web page or get a PDF version for printing.

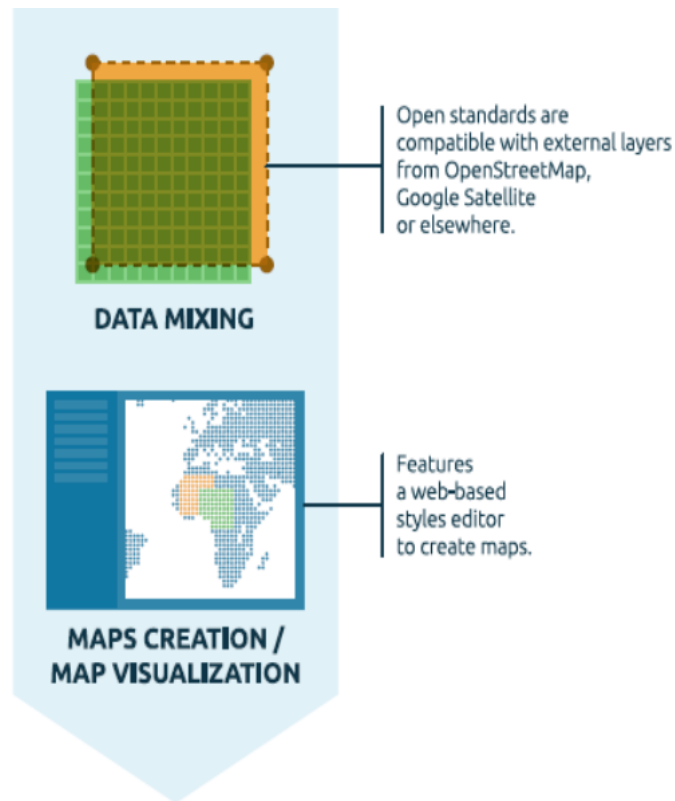
1.6.4 GeoNode as a building block

A handful of other Open Source projects extend GeoNode's functionality by tapping into the re-usability of Django applications.

Visit our gallery to see how the community uses GeoNode: [GeoNode Projects](#).

The development community is very supportive of new projects and contributes ideas and guidance for newcomers.





1.6.5 Convinced! Where do I sign?

The next steps are:

1. Make a ride on the *Online Demo*
2. Follow the *Quick Installation Guide* in order to play with your own local instance and access all the admin functionalities
3. Read the documentation starting from the *user guide* to the *admin guide*
4. Subscribe to the [geonode-users](#) and/or [geonode-devel](#) mailing lists to join the community. See also the section *Get in touch with the community* for more info.

Thanks for your interest!

1.7 Supported Browsers

GeoNode is known to be working on all modern web browsers.

This list includes (but is not limited to):

- Google Chrome.
- Apple Safari.
- Mozilla Firefox.
- Microsoft Edge.

- Microsoft Internet Explorer.

Note: The vast majority of GeoNode developers prefer using Google Chrome.

1.7.1 Internet Explorer

Versions of Microsoft Internet Explorer older than 10, exhibit known issues when used to browse a GeoNode site. As such a message is displayed warning the user that they should upgrade their browser.

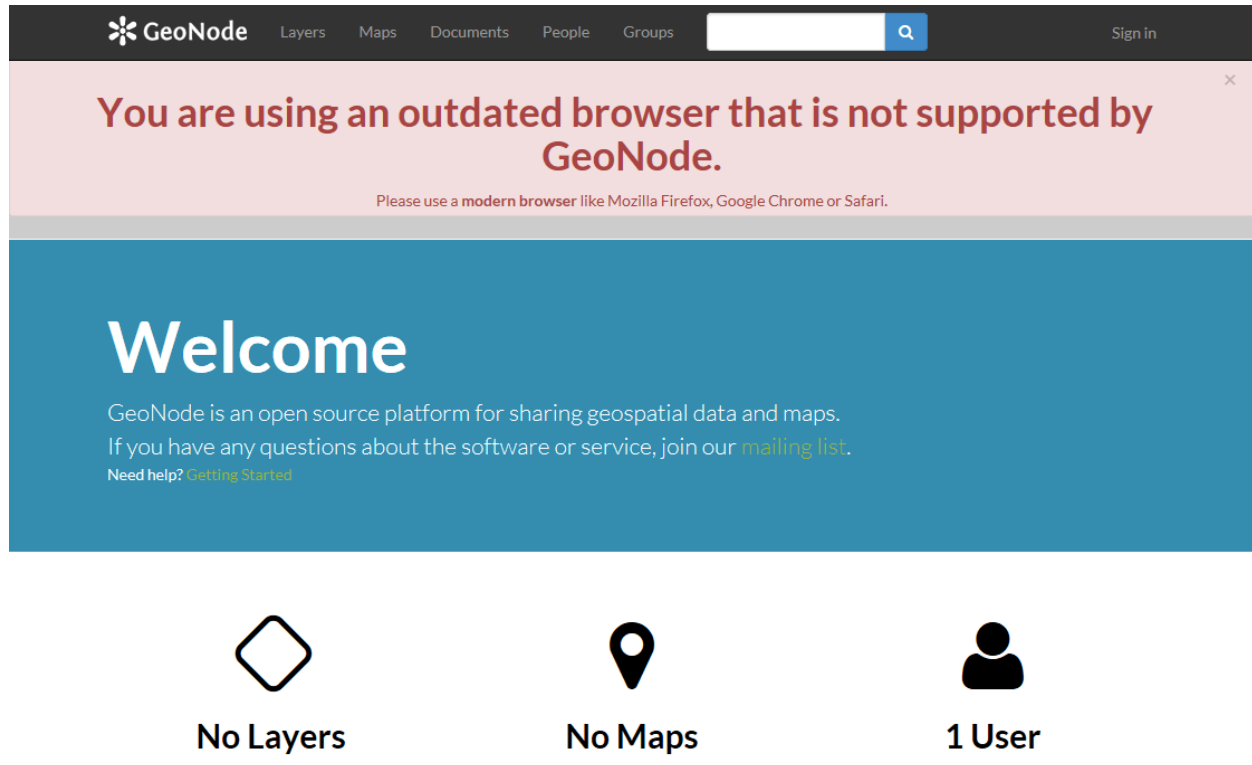


Fig. 2: Internet Explorer error message

1.7.2 Testing on Internet Explorer

When working on front end code, developers should take care to test carefully with Microsoft Internet Explorer to ensure that the features they are working on do indeed work correctly and on this browser. It is good practice to test on all browsers available, but the use of modern front end libraries like bootstrap and jQuery make it much more likely code will work across browsers seamlessly.

In order to test on Internet Explorer, developers can use the [Modern IE](#) site to download virtual machines for use in Oracle VM Virtual Box.

Once the VM is downloaded, follow the instructions to configure it in your VirtualBox setup.

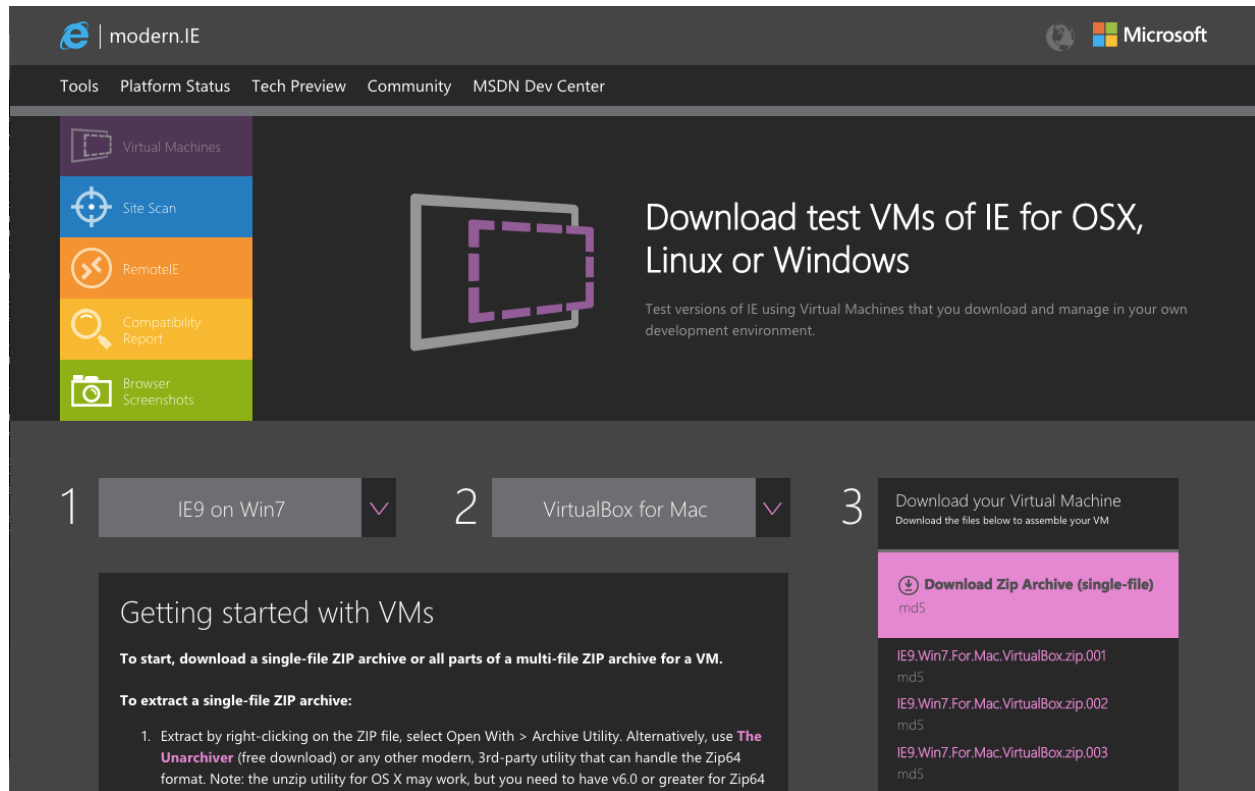


Fig. 3: Testing on Internet Explorer

After the VM is setup, you can access your development instance of GeoNode by visiting the IP address of your host machine or on the bridged interface (usually 10.0.2.2) and begin your testing.

1.8 Online Demo

Note: Disclaimer we do not guarantee for any data published on this Demo Site. Publish the data at your own risk. Every dataset will be removed automatically every Sunday. If you find some dataset that shouldn't be there, please write suddenly to developers and maintainers.

See the section *Get in touch with the community* for details.

A live demo of the latest stable build is available at <http://master.demo.geonode.org/>.

Anyone may sign up for a user account, upload and style data, create and share maps, and change permissions.

Since it is a demo site, every sunday all the datasets will be wiped out. Users, passwords and groups will be preserved.

It should hopefully allow you to easily and quickly make a tour of the main capabilities of GeoNode.

Warning: This GeoNode instance is configured with standards settings and a very low security level. This is a demo only not to be considered a really production ready system. For a complete list of settings, refer to the section: *Settings*

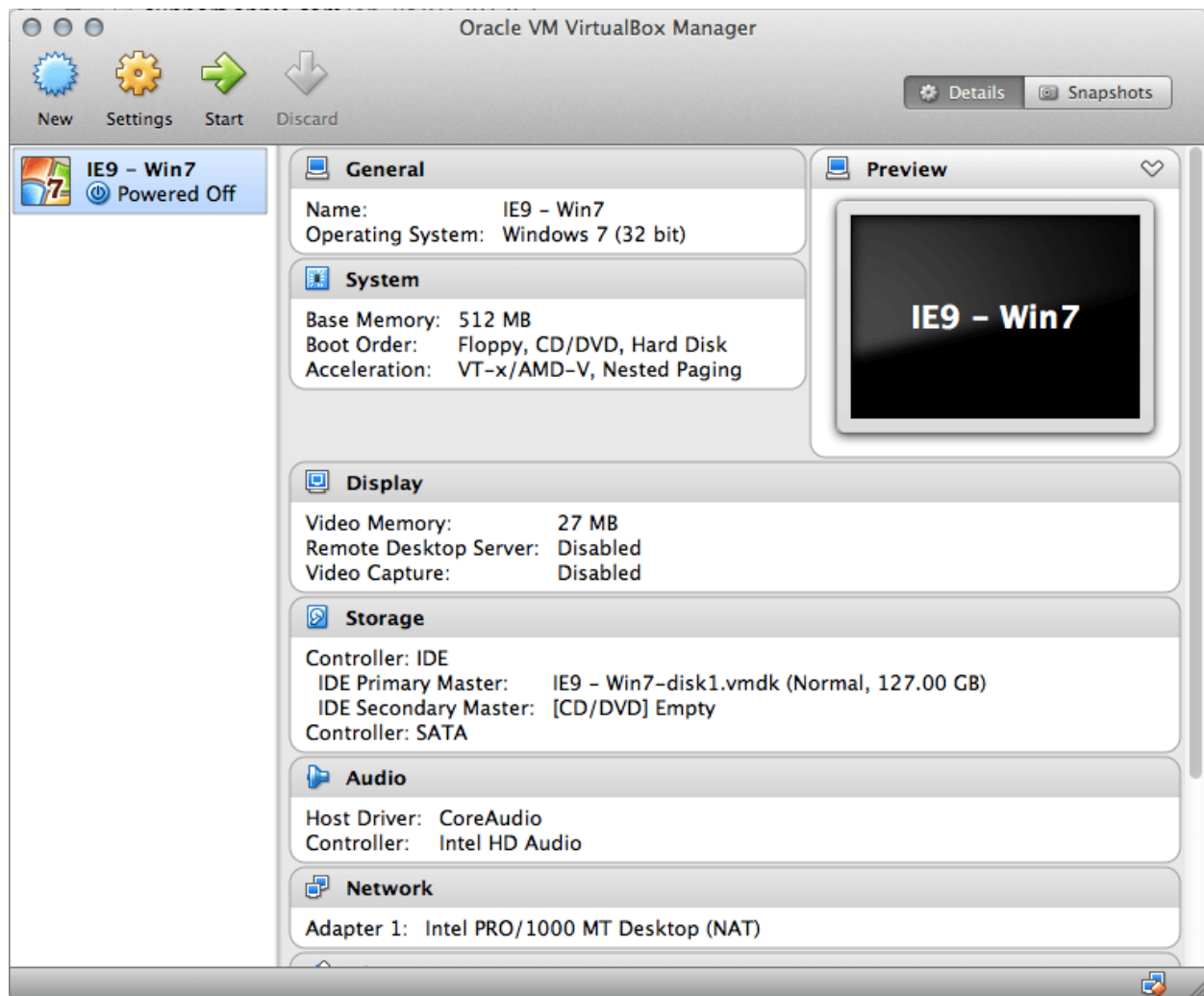


Fig. 4: Oracle VirtualBox admin interface

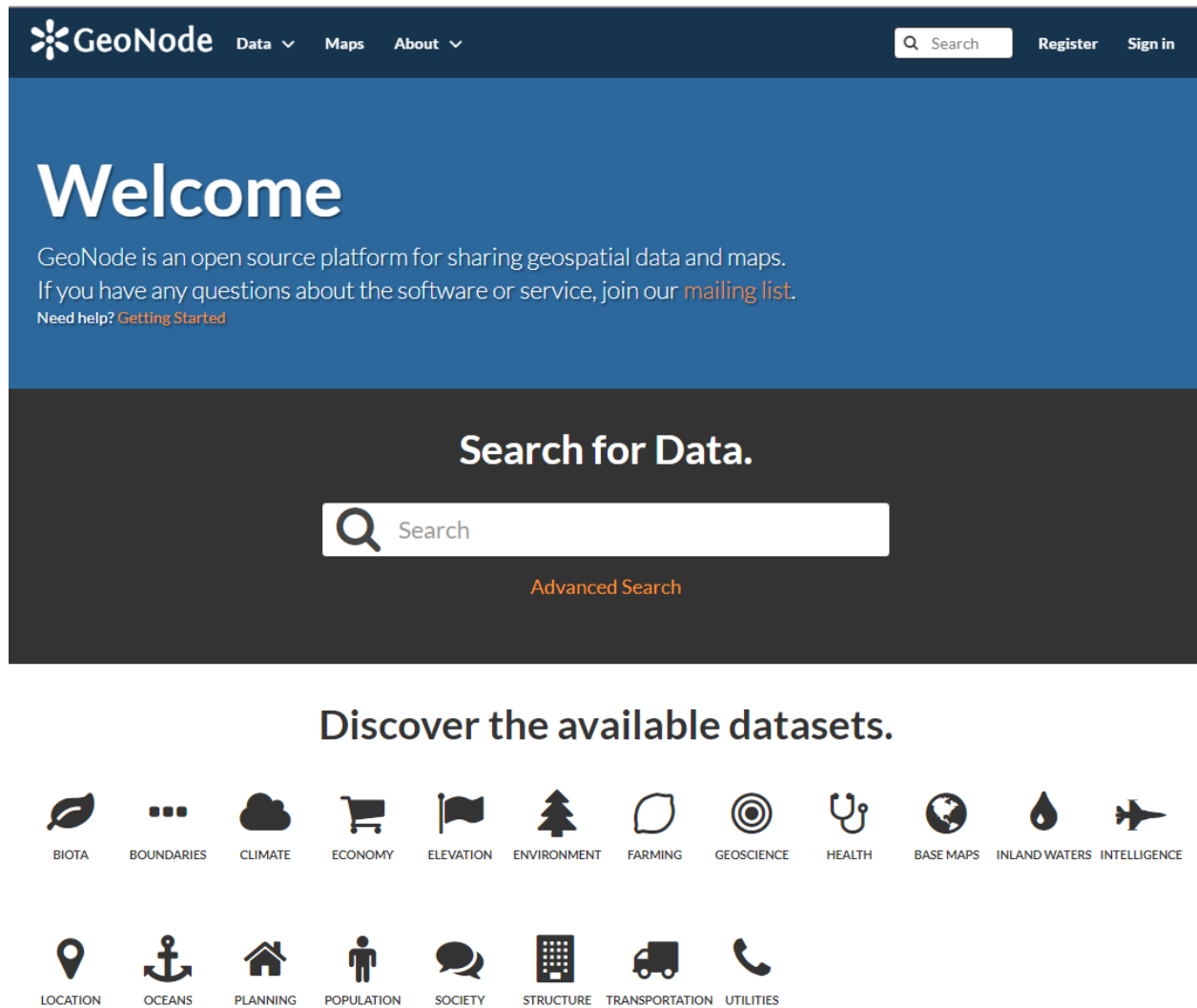


Fig. 5: Online Demo @ master.demo.geonode.org

1.9 Quick Installation Guide

1.9.1 Quick Installation Guide

Introduction

The following is a quick guide to get started with GeoNode in most common operating systems.

Note: For a full setup and deployment, please refer to the *complete installation guides*

This is meant to be run on a fresh machine with no previously installed packages or GeoNode versions.

Warning: The methods presented here are meant to be used for a limited internal demo only. Before exposing your GeoNode instance to a public server, please read carefully the *hardening guide*

Recommended Minimum System Requirements

A definite specification of technical requirements is difficult to recommend. Accepted performance is highly subjective. Furthermore, the performance depends on factors such as concurrent users, records in the database or the network connectivity of your infrastructure.

For deployment of GeoNode on a single server, the following are the *bare minimum* system requirements:

- 8GB of RAM.
- 2.2GHz processor with 2 cores. (Additional processing power may be required for multiple concurrent styling renderings)
- 10 GB software disk usage.
- Additional disk space for any data hosted with GeoNode and tiles cached with GeoWebCache. For spatial data, cached tiles, and “scratch space” useful for administration, a decent baseline size for GeoNode deployments is between 50GB and 100GB.
- 64-bit hardware **strongly** recommended.

OSGeo Live CD



OSGeoLive is a self-contained bootable DVD, USB thumb drive or Virtual Machine based on Ubuntu, that allows you to try a wide variety of open source geospatial software without installing anything.

It is composed entirely of free software, allowing it to be freely distributed, duplicated and passed around.

It provides pre-configured applications for a range of geospatial use cases, including storage, publishing, viewing, analysis and manipulation of data.

It also contains sample datasets and documentation.

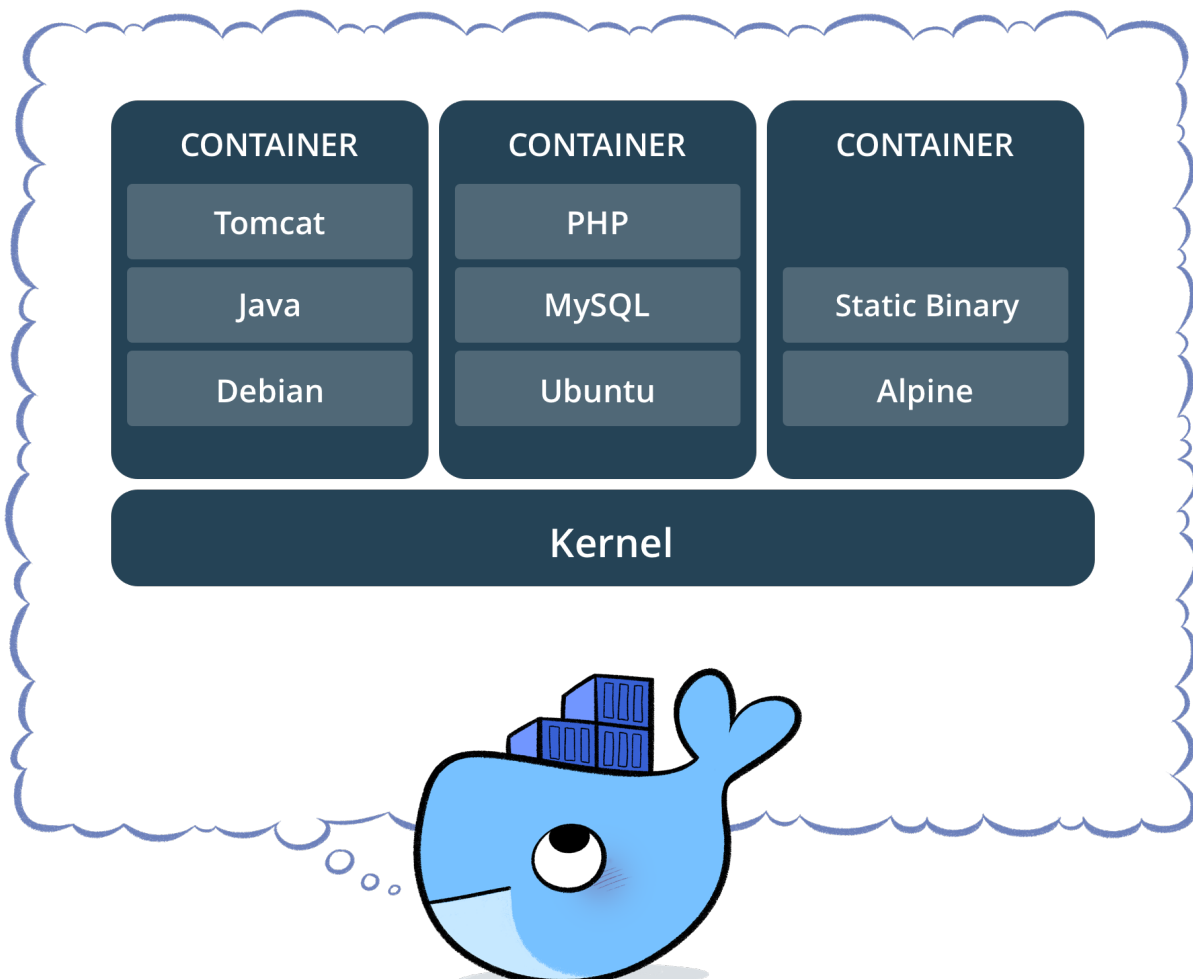
To try out the applications, simply:

- Insert DVD or USB thumb drive in computer or virtual machine.
- Reboot computer. (verify boot device order if necessary)
- Press *Enter* to startup & login.
- Select and run applications from the *Geospatial* menu.

OSGeoLive is an [OSGeo Foundation](#) project. The [OSGeo Foundation](#) is a not-for-profit supporting Geospatial Open Source Software development, promotion and [education](#).

Install via Docker

[Docker](#) is a free software platform used for packaging software into standardized units for development, shipment and deployment.



Note: credits to Docker

Introducing main concepts

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.

Docker containers running on a single machine share that machine's operating system kernel; they start instantly and use less compute and RAM.

Containers can share a single kernel, and the only information that needs to be in a container image is the executable and its package dependencies, which never need to be installed on the host system.

Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space.

The following tutorials will introduce the use of Docker community edition on:

- Ubuntu 18.04
- CentOS 7.0

GeoNode provides also an advanced stack of Docker containers for a more stable, production-like setup:

- *SPCGeoNode*

1.10 GeoNode Users Guide

1.10.1 Accounts and User Profile

In GeoNode many contents are public so unregistered users have read-only access to public maps, layers and documents. In order to create maps, add layers or documents, edit the data and share these resources with other users, you need to sign in.

GeoNode is primarily a *social* platform, thus a primary component of any GeoNode instance is the user account.

This section will guide you through account registration, updating your account information and preferences, connections with social networks and email addresses.

Creating a New Account

To take full advantage of all the GeoNode features you need an user account. Follow these step to create a new one.

1. From any page in the web interface, you will see a *Register* link. Click that link, and the register form will appear

Note: The registrations in GeoNode must be open, in case you don't see the register link then it's not possible to register unless the administrator of the site does that for you.

2. On the next page, fill out the form. Enter a user name and password in the fields. Also, enter your email address for verification.

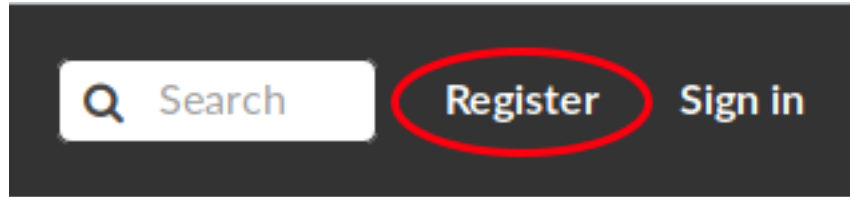


Fig. 6: Sign in screen

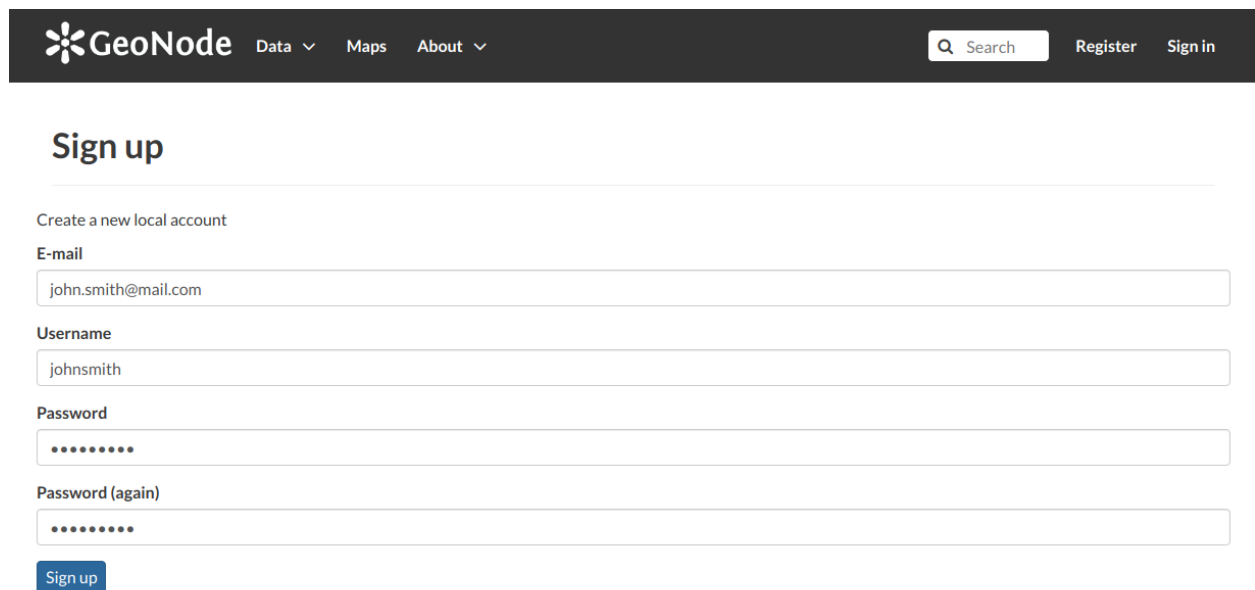
A screenshot of the GeoNode "Sign up" page. The top navigation bar is dark grey with the GeoNode logo, "Data", "Maps", and "About" links, a search bar, and "Register" and "Sign in" buttons. The main content area has a white background with the heading "Sign up". Below the heading is a horizontal line, followed by the text "Create a new local account". The form includes four input fields: "E-mail" (containing "john.smith@mail.com"), "Username" (containing "johnsmith"), "Password" (containing seven dots), and "Password (again)" (containing seven dots). A blue "Sign up" button is at the bottom left of the form.

Fig. 7: Registering for a new account

3. You will be automatically logged in and redirected to the Profile page. An email will be sent confirming that you have signed up. If no errors occur during the registration, the following alerts will appear on the screen:

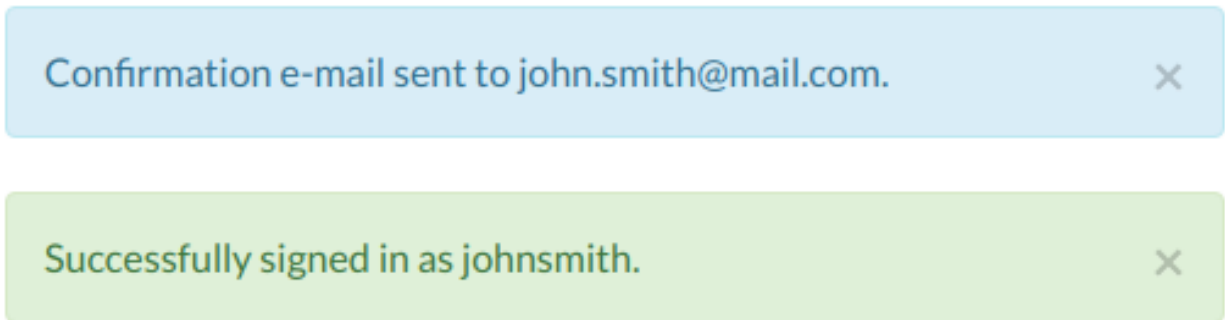


Fig. 8: Alerts

To logout click on the *Log out* link of the user menu.

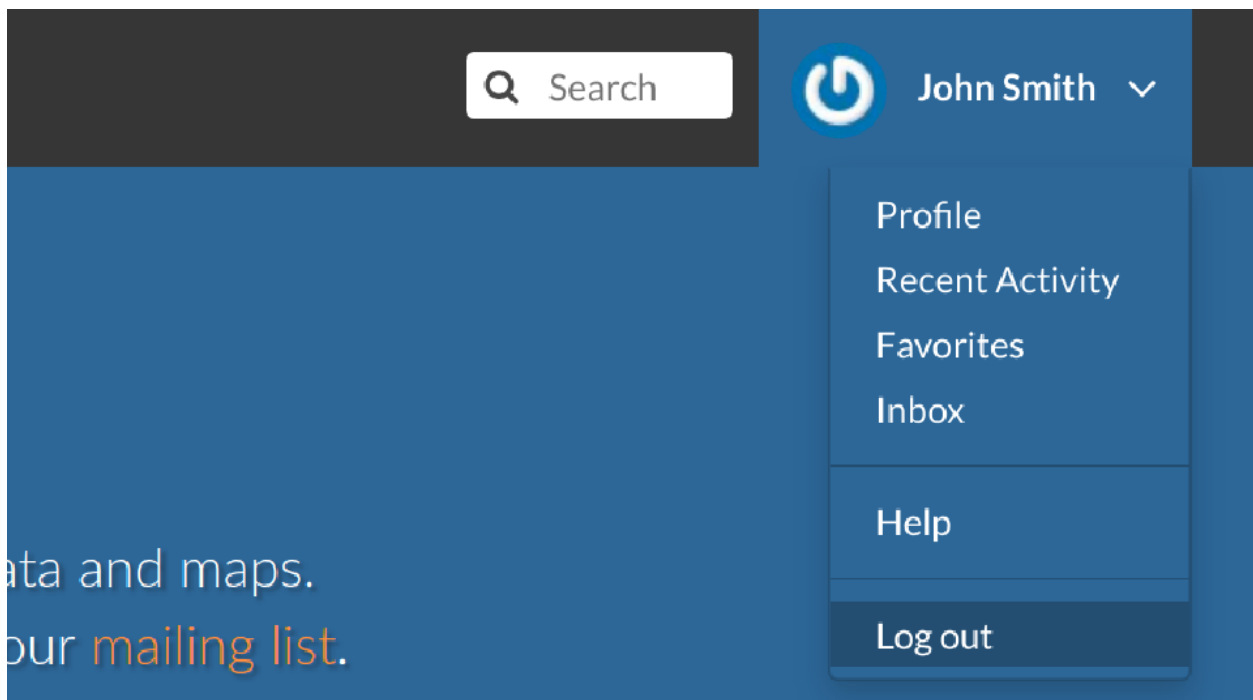


Fig. 9: Logout link

You have to confirm this action as described in the picture below.

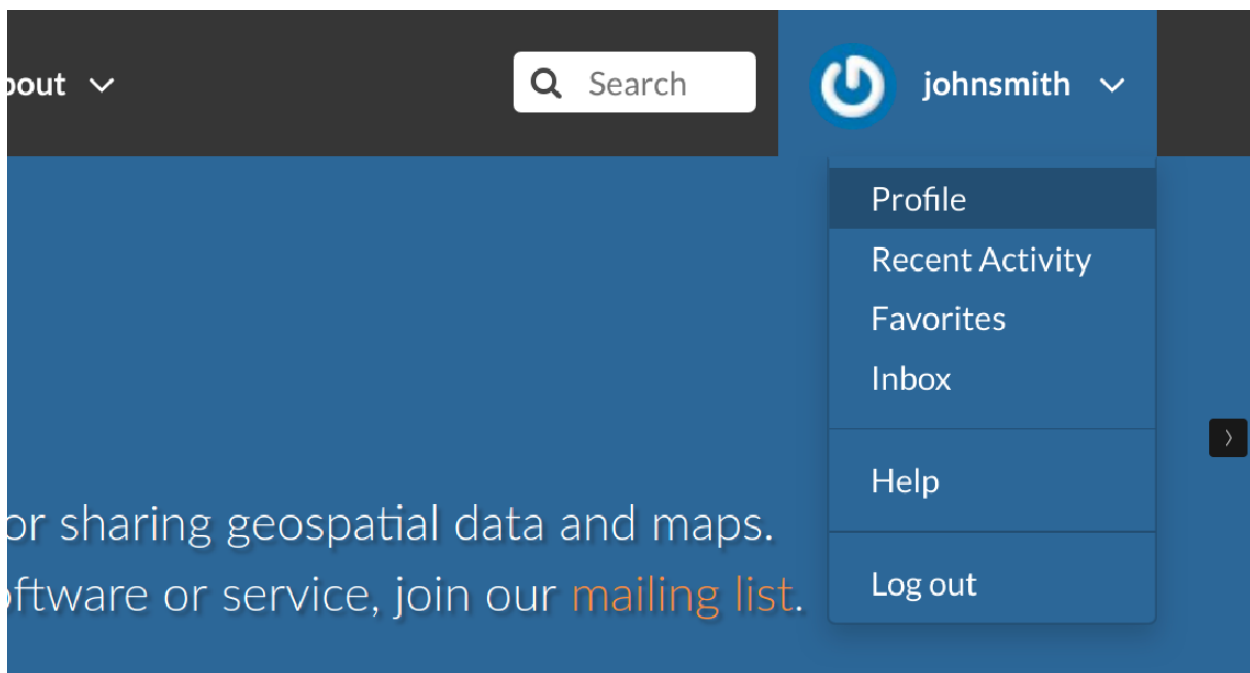
Fig. 10: *Confirm Log out*

Updating the Profile

Once having an account you can enrich your profile with useful information, you can also edit or delete the existing ones. You can connect the account with your social network, associate many e-mail addresses to it and manage many options such as preferences about notifications.

You can update these information anytime from your *Profile* page, it is accessible from the user menu.

So, click on your user name in the top right of the screen. A drop-down list will show. Click on *Profile* to enter the *Profile* settings page.


Fig. 11: *Link to your profile*


The *Profile* page looks like the one shown in the picture below.

Your personal information are shown under the username. At the bottom of the page are listed all the resources associated to your *Profile*, you can decide to view only layers or maps or documents by clicking on the corresponding tab.

Through the link `User layers WMS GetCapabilities` document you can retrieve an XML document with the list of the available layers.

On the right side of the page there are many useful links to edit personal information, to upload and create layers or maps, to update your *Profile* settings and to get in touch with other GeoNode users.

 **GeoNode** [Data](#) [Maps](#) [About](#)

 **johnsmith** [▼](#)

johnsmith



johnsmith

Position	Not provided.
Organization	Not provided.
Location	Not provided.
Voice	Not provided.
Fax	Not provided.
Description	Not provided.
Keywords	Not provided

 User layers WMS GetCapabilities document

[Message User](#)

[Edit profile](#)

[Connected social accounts](#)

[Associated e-mails](#)

[Set/Change password](#)

[Upload new layers](#)

[Create a new layer](#)

[Create a new map](#)

[My Activities](#)

[Favorites](#)

[Notifications](#)

[Invite Users](#)

Resources

[All contents](#) [Layers](#) [Maps](#) [Documents](#)



OSM Railways

Railways extracted from OSM

 johnsmith  4 Jun 2019  0  0  0

[View Map](#)



railways

No abstract provided

 johnsmith  4 Jun 2019  0  0  0

[Create a Map](#)

[<](#)

page 1 of 1

[>](#)

Fig. 12: User profile page

The *Favorites* link, also accessible from the user menu, drive you to the list of the resources marked as your favorites.

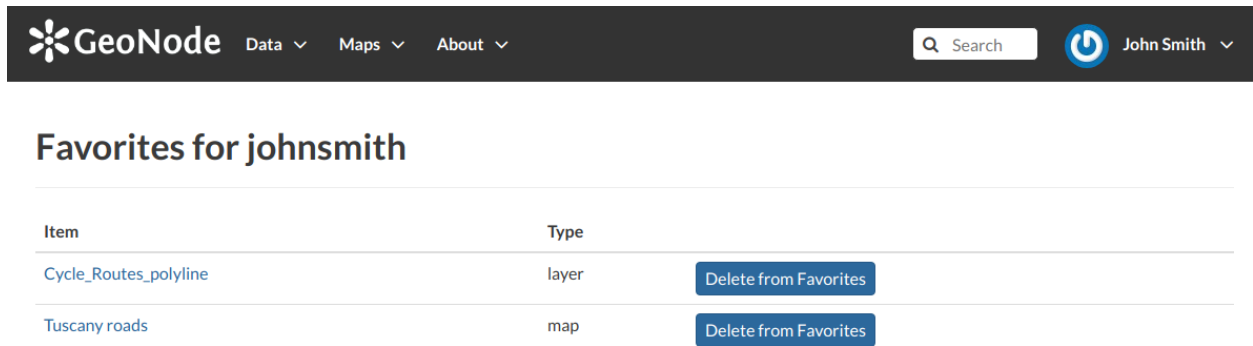


Fig. 13: *Favorites*

Click the *Delete from Favorites* button to remove the resource from the list.

The *My Activities* link allows to see all your recent activities on GeoNode such as layers uploading and maps creation.

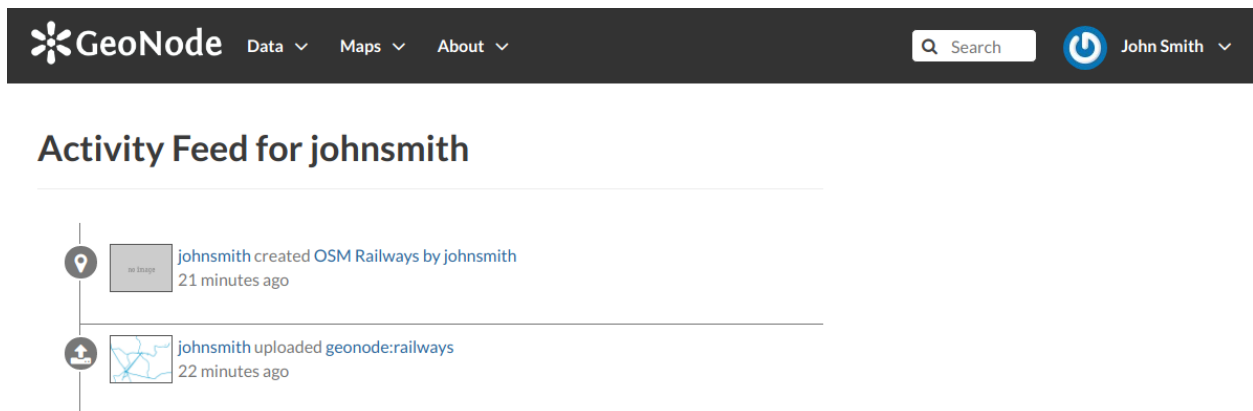


Fig. 14: *Recent activities*

This link is also available in the user menu.

All other links and their functionalities will be described in depth in the following sections.


Editing Profile Information

Your *Profile* contains personal information such as your address, your telephone number, your organization and so on but it is empty by default at the beginning.

Through the *Edit profile* button of the *Profile* page (see [Updating the Profile](#)) you can set your details, including your avatar.


When finished, click *Update profile*. You will be redirected to the *Profile* page.

A message will confirm the profile has been correctly updated.

Data ▾Maps ▾About ▾

Searchjohnsmith ▾

Edit Your Profile


Change your avatar

First name

Last name

Email address

Organization Name

name of the responsible organization

Profile

John Smith profile

introduce yourself

Position Name

role or position of the responsible person

Voice

telephone number by which individuals can speak to the responsible organization or individual

Facsimile

telephone number of a facsimile machine for the responsible organization or individual

Delivery Point

physical and email address at which the organization or individual may be contacted

City

city of the location

Administrative Area

state, province of the location

Postal Code

ZIP or other postal code

Country

country of the physical address

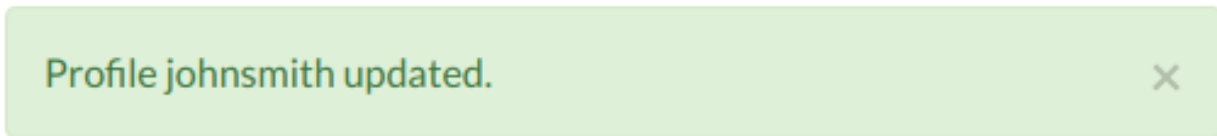
Keywords

A space or comma-separated list of keywords

Language

Timezone

Update profile

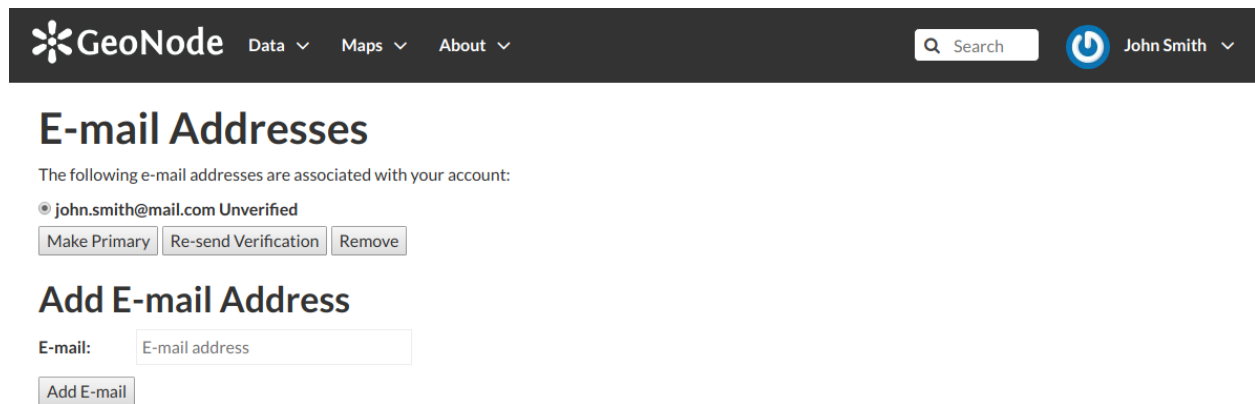
Fig. 16: *Updating Profile correctly finalized*

Connecting your Account with Social Networks

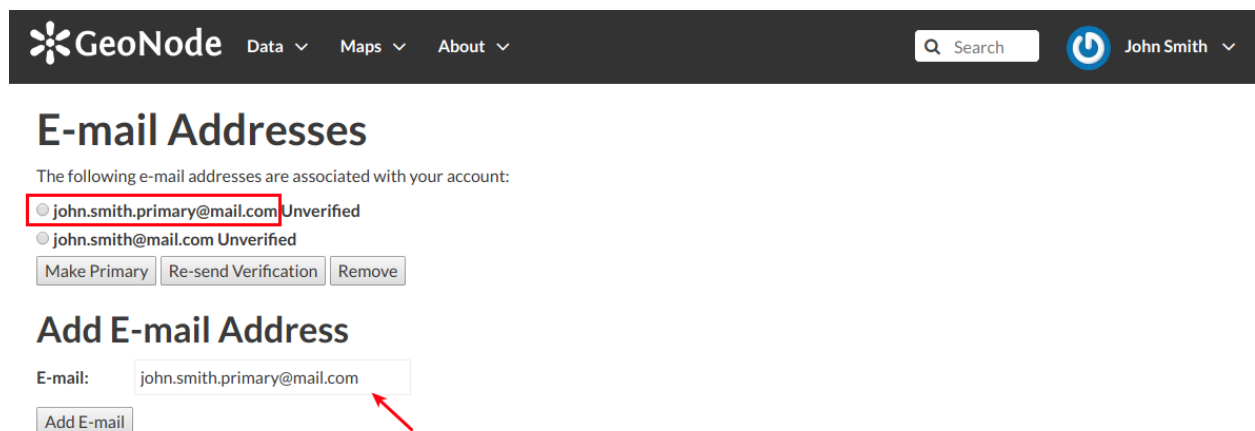
You currently have no social network accounts connected to this account.

Associating your Account with an e-mail

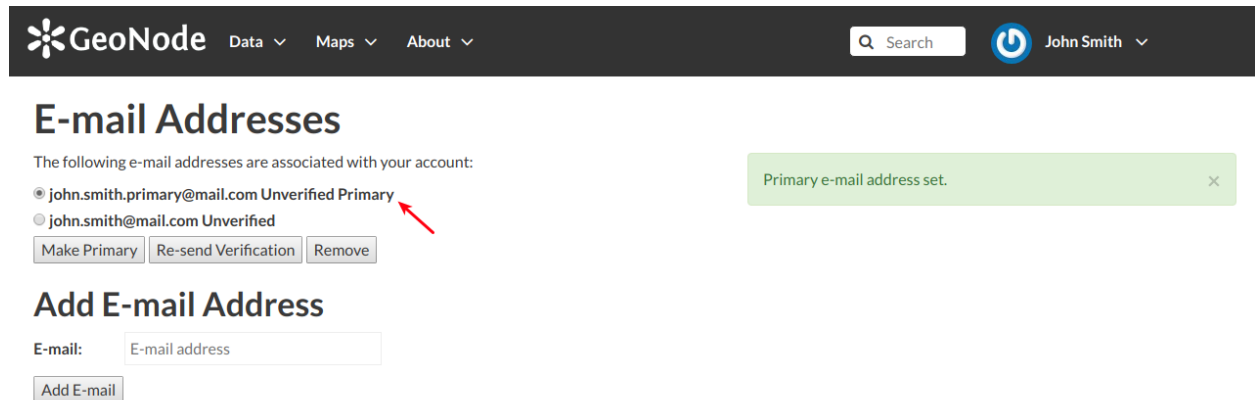
Your account is automatically associated with the e-mail that you used to register yourself on the platform.

Fig. 17: *Accounts e-mail*

By clicking on *Associated e-mails* of the *Profile* page (see [Updating the Profile](#)), you will have the possibility to fill up a new e-mail address. Type it in the *E-mail* input field then click on *Add E-mail* to perform a new association.

Fig. 18: *New e-mail association*

You can make it primary if necessary, in order to receive the notification on this address. To do that, select the e-mail that you want, then click on *Make Primary*.



E-mail Addresses

The following e-mail addresses are associated with your account:

- john.smith.primary@mail.com Unverified Primary
- john.smith@mail.com Unverified

Make Primary Re-send Verification Remove

Add E-mail Address

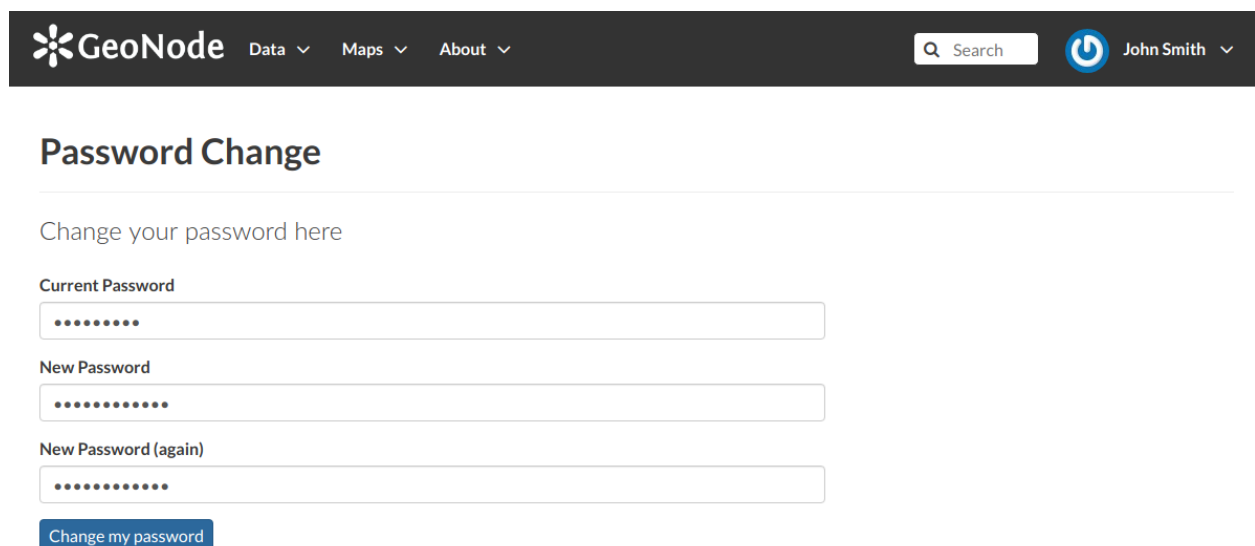
E-mail:

Add E-mail

Fig. 19: *Primary e-mail address*

Managing the Password

To change your password, click on the *Set/Change password* link of the *Profile* page (see *Updating the Profile*). You will be asked to enter your current password and the new one (two times). Click on *Change my password* to perform the change.



Password Change

Change your password here

Current Password

.....

New Password

.....

New Password (again)

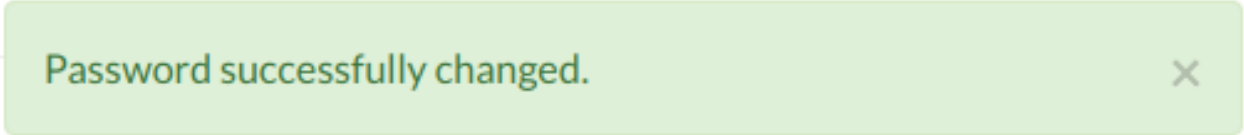
.....

Change my password

Fig. 20: *Change your password*

If no errors occur you will see a confirmation message.

Next time you log in you will have to use the new password.



Password successfully changed.

Fig. 21: *Change password confirmation*

Setting Notification Preferences

By default GeoNode sends notifications to the users for events that the users could be subscribe such as a new layer uploaded or a new rate added to a map. You can adjust your notification settings by clicking on the *Notifications* link of the *Profile* page (see *Updating the Profile*).

Note: Make sure to have a verified email address to which notices can be sent. If not see *Associating your Account with an e-mail*.

Now check/uncheck the notification types you wish to receive or not receive. It is possible to be notified for the events shown in the picture below.

1.10.2 Interacting with Users and Groups

The GeoNode platform allows you to communicate by message with other GeoNode users and groups of users.

You can also invite external users to join your GeoNode. In order to do that, click on *Invite Users* in the *Profile* page (see *Updating the Profile*) or in the *About* menu in the *Home* page.

You can invite your contacts typing their email addresses in the input field as shown in the picture below. Click on *Submit* to perform the action.

A message will confirm that invitations have been correctly sent.

Note: You can invite more than one user at the same time by typing the email addresses inline with a semi-colon separator.

The next sections will show you how to view information about other users and how to contact them.


Viewing other users information


Once your account is created, you can view other accounts on the system.

To see information about other users on the system, click the *People* link of the *About* menu in *Home* page.

You will see a list of users registered on the system.

The *Search* tool is very useful in case of many registered users, type the name of the user you are looking for in the input text field to filter the users list.

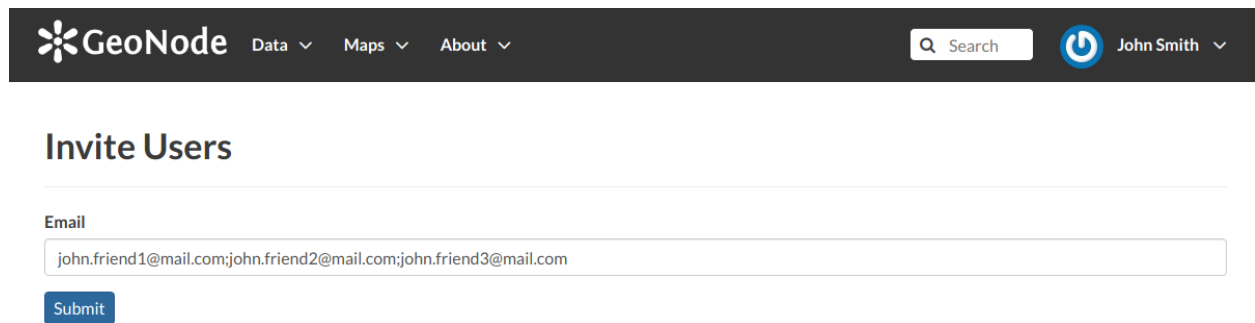

Data ▾
Maps ▾
About ▾

 John Smith ▾

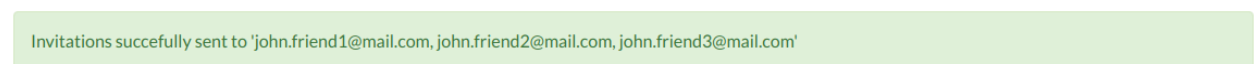
Notification Settings

Notification Type	Email
Request to download a resource A request for downloading a resource was sent	<input checked="" type="checkbox"/>
Layer Created A Layer was created	<input checked="" type="checkbox"/>
Layer Updated A Layer was updated	<input checked="" type="checkbox"/>
Layer Approved A Layer was approved by a Manager	<input checked="" type="checkbox"/>
Layer Published A Layer was published	<input checked="" type="checkbox"/>
Layer Deleted A Layer was deleted	<input checked="" type="checkbox"/>
Comment on Layer A layer was commented on	<input checked="" type="checkbox"/>
Rating for Layer A rating was given to a layer	<input checked="" type="checkbox"/>
Map Created A Map was created	<input checked="" type="checkbox"/>
Map Updated A Map was updated	<input checked="" type="checkbox"/>
Map Approved A Map was approved by a Manager	<input checked="" type="checkbox"/>
Map Published A Map was published	<input checked="" type="checkbox"/>
Map Deleted A Map was deleted	<input checked="" type="checkbox"/>
Comment on Map A map was commented on	<input checked="" type="checkbox"/>
Rating for Map A rating was given to a map	<input checked="" type="checkbox"/>
Document Created A Document was created	<input checked="" type="checkbox"/>
Document Updated A Document was updated	<input checked="" type="checkbox"/>
Document Approved A Document was approved by a Manager	<input checked="" type="checkbox"/>
Document Published A Document was published	<input checked="" type="checkbox"/>
Document Deleted A Document was deleted	<input checked="" type="checkbox"/>
Comment on Document A Document was commented on	<input checked="" type="checkbox"/>
Document for Map A rating was given to a document	<input checked="" type="checkbox"/>
User following you Another user has started following you	<input checked="" type="checkbox"/>
User requested access A new user has requested access to the site	<input checked="" type="checkbox"/>
Account activated This account is now active and can log in the site	<input checked="" type="checkbox"/>
Layer Uploaded A layer was uploaded	<input checked="" type="checkbox"/>
Monitoring alert Alert situation reported by monitoring	<input checked="" type="checkbox"/>

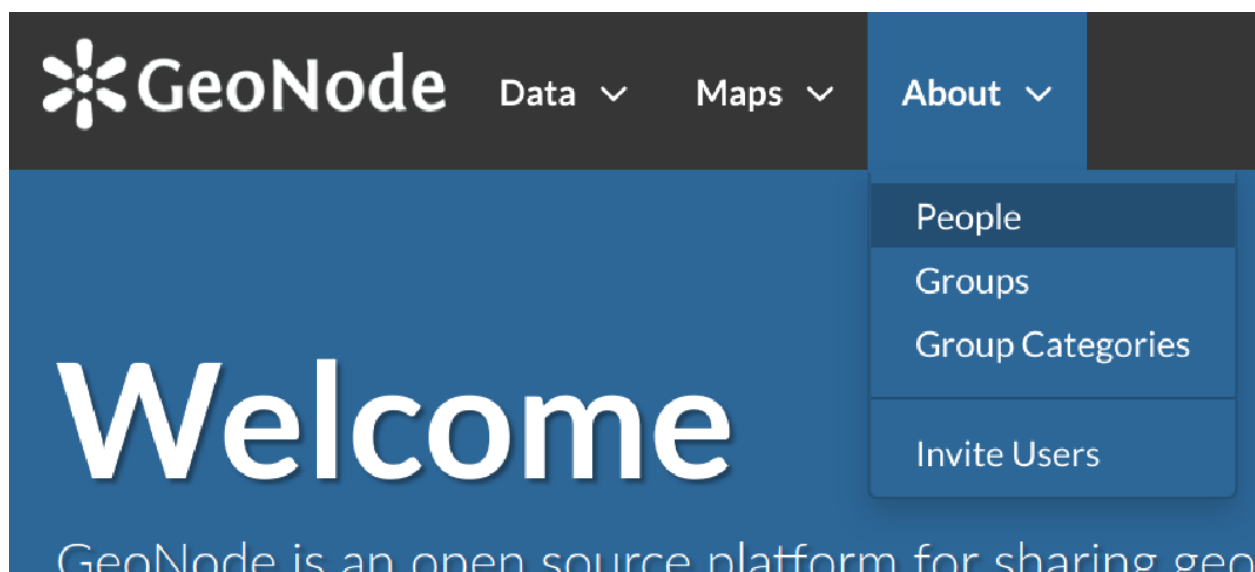
Change





The screenshot shows the 'Invite Users' page in the GeoNode interface. At the top is a dark navigation bar with the GeoNode logo, links for 'Data', 'Maps', and 'About', a search bar, and a user profile for 'John Smith'. Below the navigation bar, the page title 'Invite Users' is displayed. The main content area contains a form with a label 'Email' and a text input field containing the email addresses 'john.friend1@mail.com;john.friend2@mail.com;john.friend3@mail.com'. A blue 'Submit' button is located below the input field.

Fig. 23: *Invite users to join GeoNode*

The screenshot shows a green confirmation message box. The text inside the box reads: 'Invitations successfully sent to 'john.friend1@mail.com, john.friend2@mail.com, john.friend3@mail.com''.

Fig. 24: *Invitations confirm message*Fig. 25: *About menu - People link*

 Data ▾ Maps ▾ About ▾

Search  John Smith ▾

SEARCH

Search by name 

Total: 198 

Fig. 26: List of the registered users

Select a user and click on its *username* to access to the user details page.

The screenshot shows the GeoNode user details page for Mario Rossi (mariorossi). The page layout includes a dark header with the GeoNode logo, navigation links (Data, Maps, About), a search bar, and a user profile dropdown (John Smith). The main content area features the user's profile information, a table of personal details, and a list of resources.

GeoNode Data Maps About Search John Smith

Mario Rossi (mariorossi)

Mario

Position	Not provided.
Organization	Mario Rossi Transport
Location	Not provided.
Voice	Not provided.
Fax	Not provided.
Description	Not provided.
Keywords	Not provided

User layers WMS GetCapabilities document

[Message User](#)
[User Activities](#)

Resources

All contents Layers Maps Documents

Tuscany roads
Roads of Tuscany

Mario Rossi 3 Jun 2019 0 0 0
View Map

roads
No abstract provided

Mario Rossi 3 Jun 2019 0 0 0
Create a Map

< page 1 of 1 >

Fig. 27: User details

In this page the main information about the user are shown: personal information (name, surname, organization and so on...) and the resources the user owns (layers, maps and documents).

Through the *User Activities* link, in right side of the page, it is possible to visualize all the activities the user has been done.

The *Message User* link lets you to contact other users, see the next section to read more about that.

It is also possible, in GeoNode, to see the recent activities of all users through the *Recent Activities* link of the user

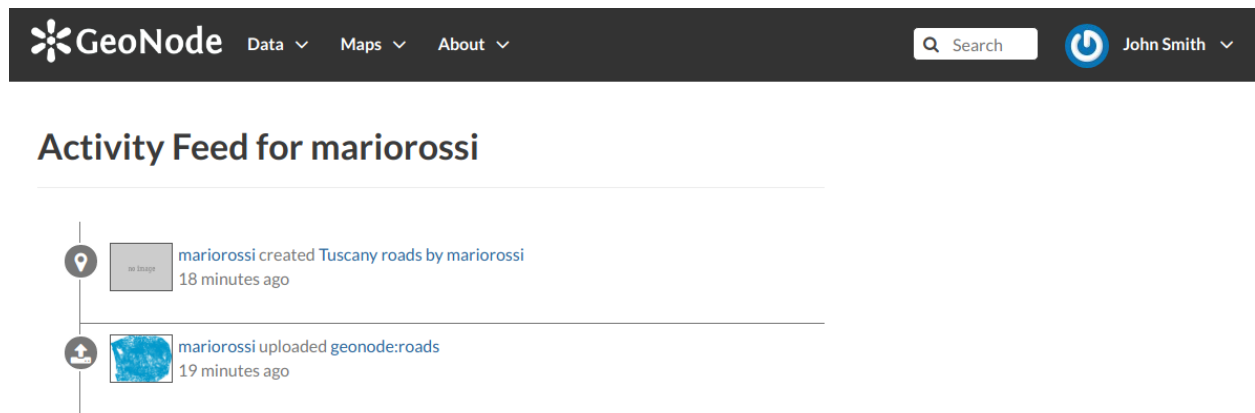


Fig. 28: User activities

menu.

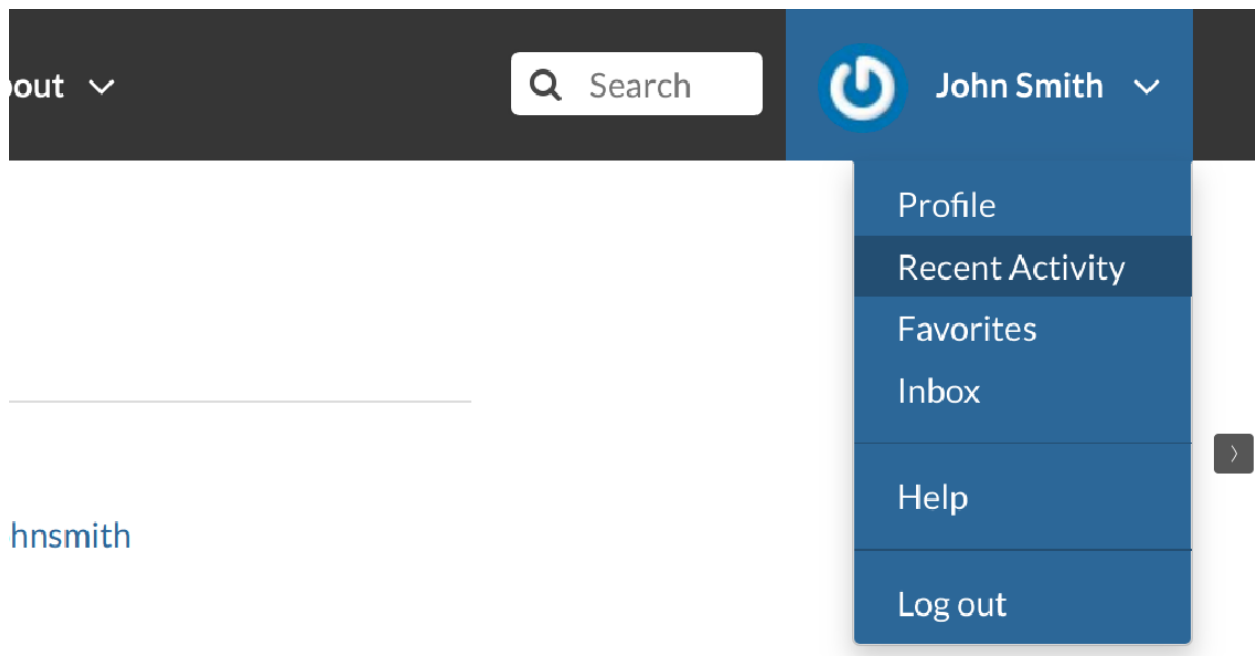




Fig. 29: Recent Activities link

In the picture below an example.


As you can see, you can decide whether to see only the activities related to layers or those related to maps or comments by switching the tabs.


 [Data](#) [Maps](#) [About](#)


 John Smith


Recent activity


All Layers Maps Comments





 johnsmith created [OSM Railways](#) by johnsmith
24 minutes ago





 johnsmith uploaded [geonode:railways](#)
25 minutes ago





 cbardalesc subido [geonode:](#)
14 hours, 18 minutes ago





 cbardalesc subido [geonode:Crhc0](#)
14 hours, 18 minutes ago





 cbardalesc subido [geonode:Crhc](#)
14 hours, 19 minutes ago





 Palma creato [SELVA](#) by Palma
14 hours, 20 minutes ago



 Palma aggiornato [geonode:latina3](#)
14 hours, 36 minutes ago



 Palma aggiornato [geonode:viterbo](#)
15 hours, 28 minutes ago




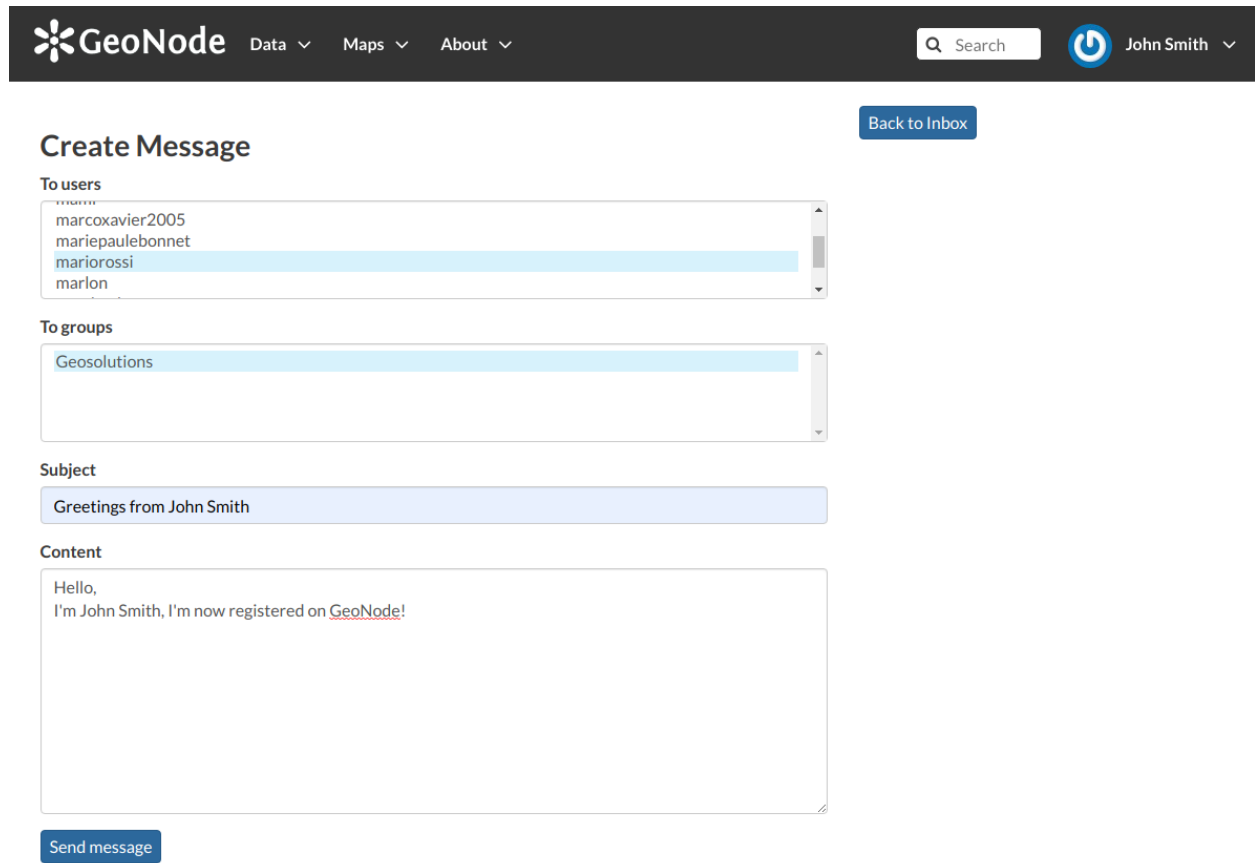
 pedroalves10 criado [teste](#) by pedroalves10

Fig. 30: Recent Activities

Contacting other users

GeoNode allows you to communicate by message with other registered users and groups.

To send a message to some user and/or groups you can follow the link *Message User* from your *Profile* page (see *Updating the Profile*) or from the *Profile* details page (see the previous section *Viewing other users information*) of that user.



The screenshot shows the 'Create Message' interface in GeoNode. At the top is a dark navigation bar with the GeoNode logo, links for 'Data', 'Maps', and 'About', a search bar, and a user profile for 'John Smith'. Below the navigation bar is a 'Back to Inbox' button. The main form is titled 'Create Message' and contains several sections: 'To users' with a list of users (marcoxavier2005, mariepaulebonnet, mariorossi, marlon) where 'mariorossi' is selected; 'To groups' with a list of groups (Geosolutions) where 'Geosolutions' is selected; 'Subject' with a text input field containing 'Greetings from John Smith'; and 'Content' with a large text area containing 'Hello, I'm John Smith, I'm now registered on [GeoNode!](#)'. At the bottom left of the form is a 'Send message' button.

Fig. 31: *Send message to users and groups*

Insert your content, type a subject and click on *Send message* to send the message to the users and groups you have selected.

You will be redirected to the *Conversation* details page related to the subject.

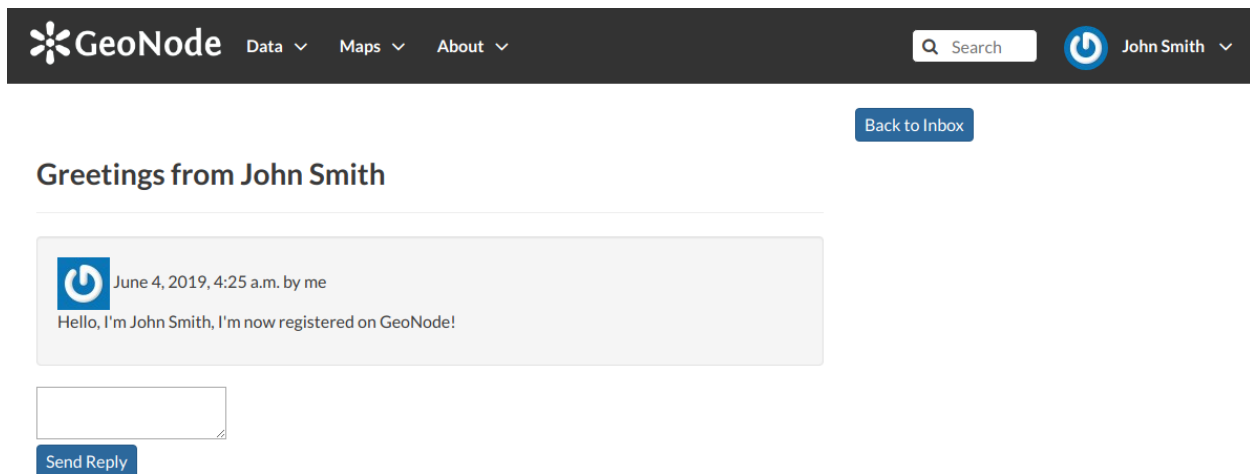


Fig. 32: Your message

The Inbox page

You can view your conversations in your *Inbox* page, reachable through the *Back to inbox* button (see the picture above) or from the *Inbox* link of the user menu.

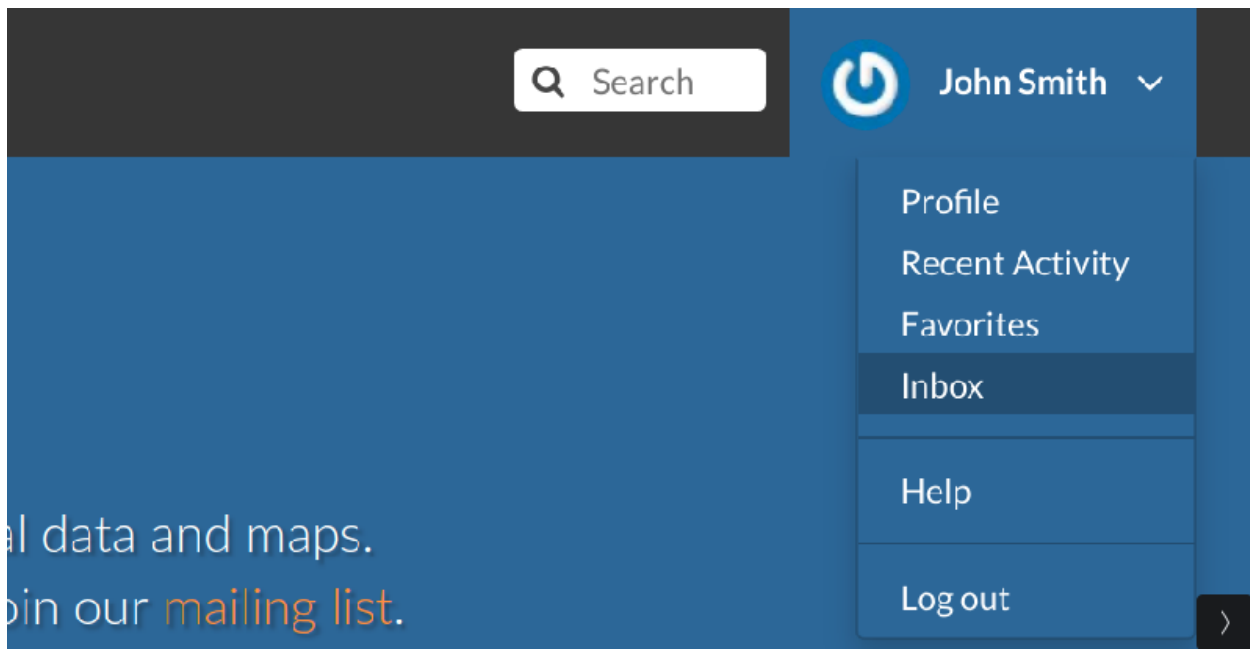


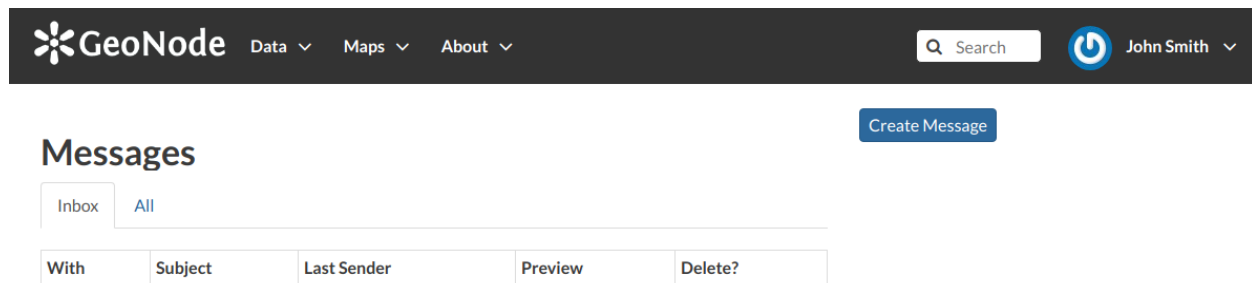
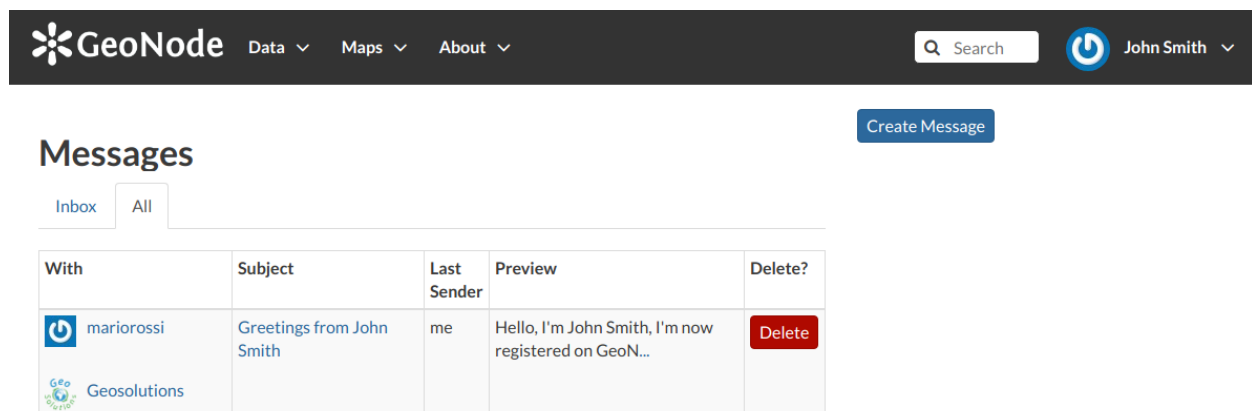
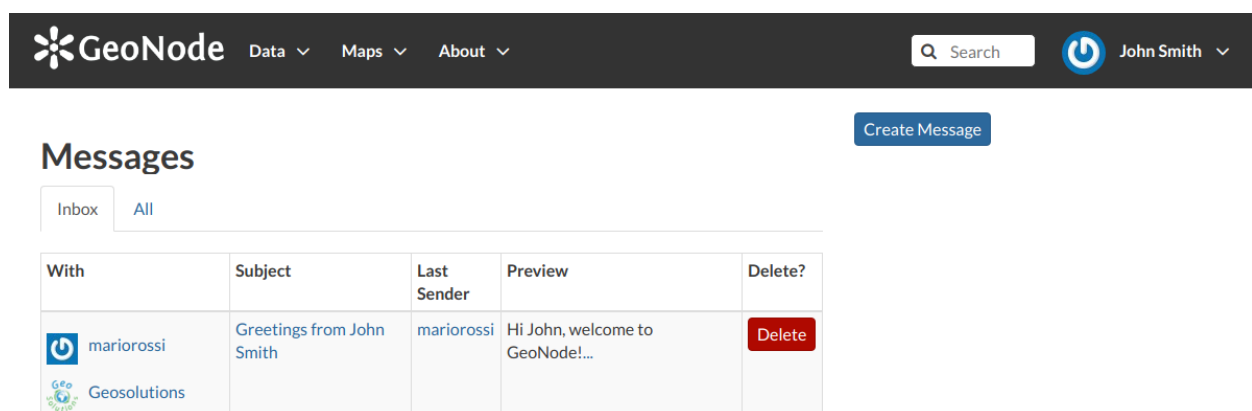
Fig. 33: Inbox link

The picture below shows how your *Inbox* page should look like.

In *Inbox* all the unread messages are listed. You haven't received any message yet so your *Inbox* is empty. If you switch to the *All* tab you can see all the conversations you are involved in.

When some user send a reply to your message your *Inbox* shows it, see the picture below for an example.

You can open the *Conversation* details by clicking on the *Subject* link.

Fig. 34: *Inbox page*Fig. 35: *All your conversations*Fig. 36: *A reply to your message*

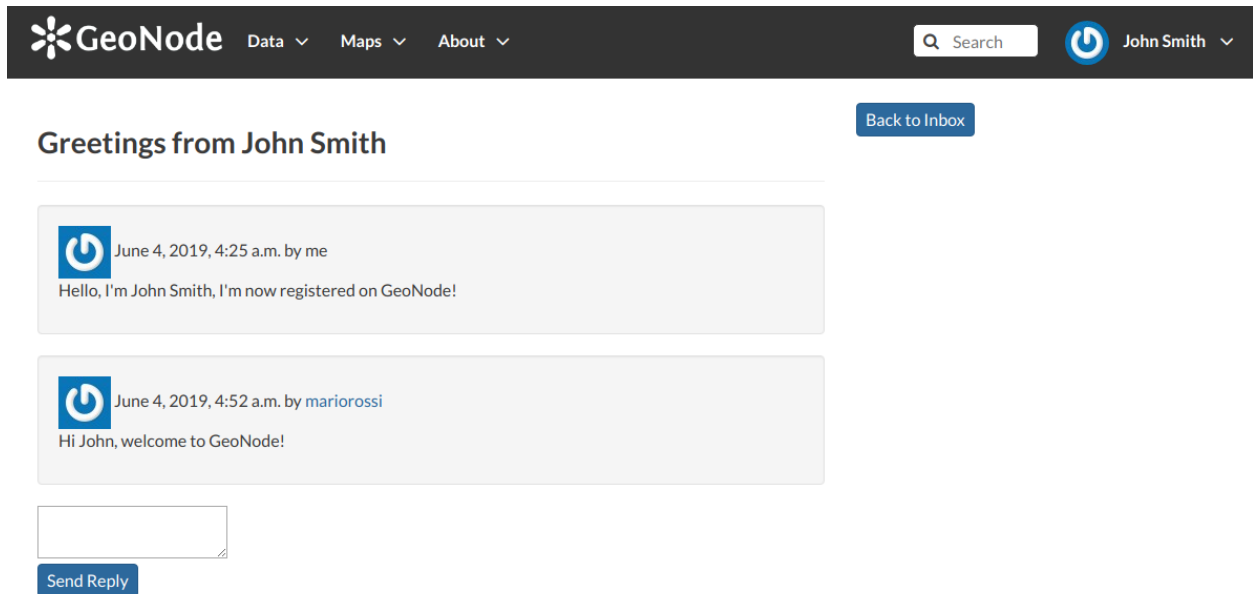


Fig. 37: Conversation details

As you can see in the picture above, in the *Conversation* page you have the ability to write a quick reply. Type your message in the text box and click on *Send Reply* to do that.

In the *Inbox* page there is also the *Create Message* button that provides you a quick link to the message creation form.

1.10.3 Data

Data management tools built into GeoNode allow for integrated creation of data, documents, link to external documents, and map visualizations. Each dataset in the system can be shared publicly or restricted to allow access to only specific users. Social features like user profiles and commenting and rating systems allow for the development of communities around each platform to facilitate the use, management, and quality control of the data the GeoNode instance contains.

The following sections will explain more in depth what data can be managed in GeoNode and how to easily find that data.

Data Types

GeoNode welcome page shows a variety of information about the current GeoNode instance.

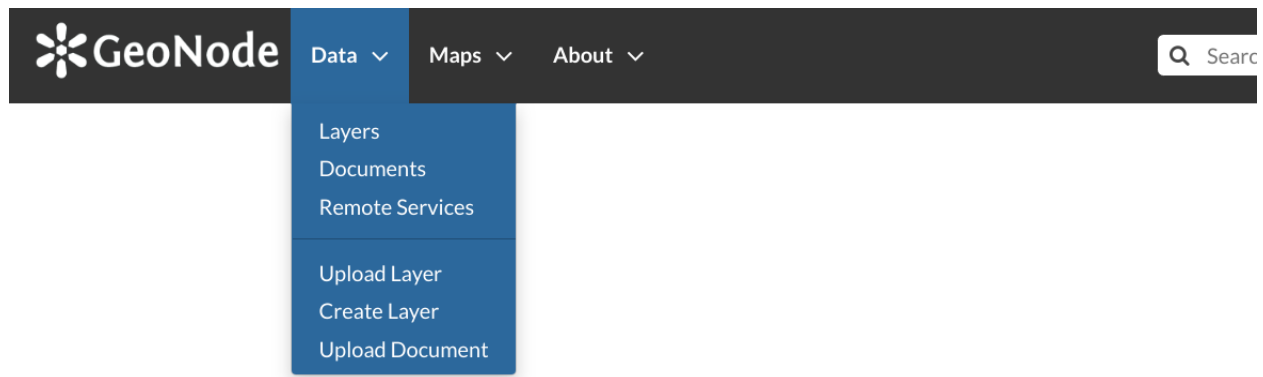
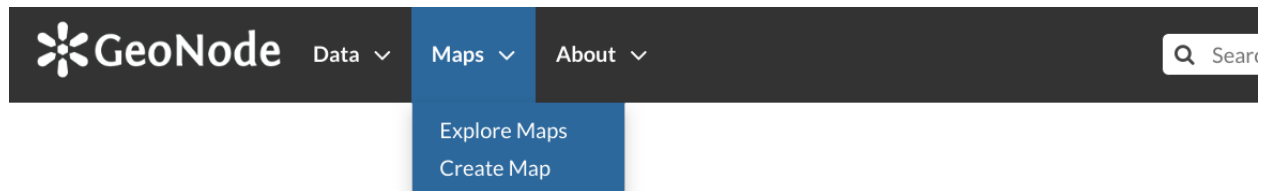
You can explore the existing data using many search tools and filters (see [Finding Data](#)) or through the links of the navigation bar at the top of the page.

There are three main types of resources that GeoNode can manage:

1. Documents
2. Layers
3. Maps

Documents and layers can be accessed from the *Data* menu of the navigation bar.

The *Maps* menu let you to manage maps.

Fig. 38: *Data menu*Fig. 39: *Maps menu*

Documents

GeoNode allows to publish tabular and text data and to manage metadata and associated documents.

Documents can be uploaded directly from your disk (see [Uploading Documents](#) for further information).

The following documents types are allowed: `.doc`, `.docx`, `.gif`, `.jpg`, `.jpeg`, `.ods`, `.odt`, `.odp`, `.pdf`, `.png`, `.ppt`, `.pptx`, `.rar`, `.sld`, `.tif`, `.tiff`, `.txt`, `.xls`, `.xlsx`, `.xml`, `.zip`, `.gz`, `.qml`.

Through the document detailed page is possible to view, download and manage a document.

Layers

Layers are a primary component of GeoNode.

Layers are publishable resources representing a raster or vector spatial data source. Layers also can be associated with metadata, ratings, and comments.

By clicking the Layers link you will get a list of all published layers. If logged in as an administrator, you will also see the unpublished layers in the same list.

GeoNode allows the user to upload vector and raster data in their original projections using a web form.

Vector data can be uploaded in many different formats (ESRI Shapefile, KML and so on...). Satellite imagery and other kinds of raster data can be uploaded as GeoTIFFs.

Maps

Maps are a primary component of GeoNode.

Maps are comprised of various layers and their styles. Layers can be both local layers in GeoNode as well as remote layers either served from other WMS servers or by web service layers such as Google or MapQuest.

GeoNode maps also contain other information such as map zoom and extent, layer ordering, and style.

You can create a map based on uploaded layers, combine them with some existing layers and a remote web service layer, share the resulting map for public viewing. Once the data has been uploaded, GeoNode lets the user search for it geographically or via keywords and create maps. All the layers are automatically reprojected to web mercator for maps display, making it possible to use popular base maps such as [OpenStreetMap](#).

Finding Data

This section will guide you to navigate GeoNode to find layers, maps and documents by using different routes, filters and search functions.

In *Home* page you can find some quick search tool.

The *Search* box in the navigation bar (see the picture below) let you type a text and find all the data which have to deal with that text.



Fig. 40: Search tool in GeoNode welcome page

When you trigger a search you are brought to the *Search* page which shows you the search result through all data types.

This page contains a wealth of options for customizing a search for various information on GeoNode. This search form allows for much more fine-tuned searches than the simple search box is available at the top of every page.

It is possible to search for data by Text, Categories, Type, Keywords, Owners, Date, Regions or Extent.

Try to set some filter and see how the resulting data list changes accordingly. An interesting type of filter is *EXTENT*: you can apply a spatial filter by moving or zooming a map within a box as shown in the picture below.

Data can be ordered by date, name and popularity.

The GeoNode welcome page offers you many other options to find resources.

- The *Search for data* tool allows you to search for data by name.

The *Search* page, which you will be redirected to, will have the TEXT filter already set with the name you have typed in the search box (see the picture below). If you want to reach the *Search page* directly, without any input text, you can click the *Advanced Search* link.

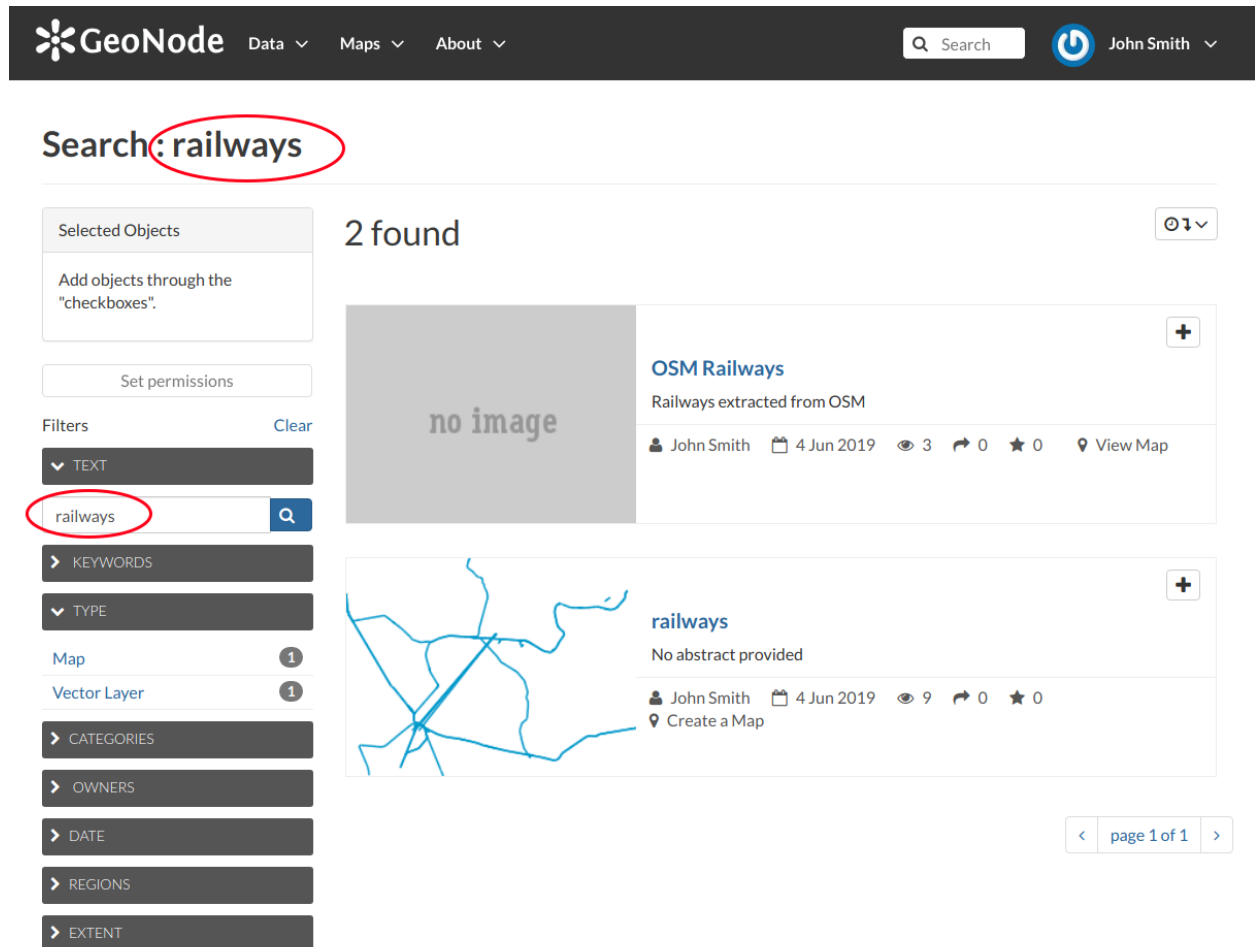


Fig. 41: *The Search page*



Fig. 42: Search filter by *EXTENT*

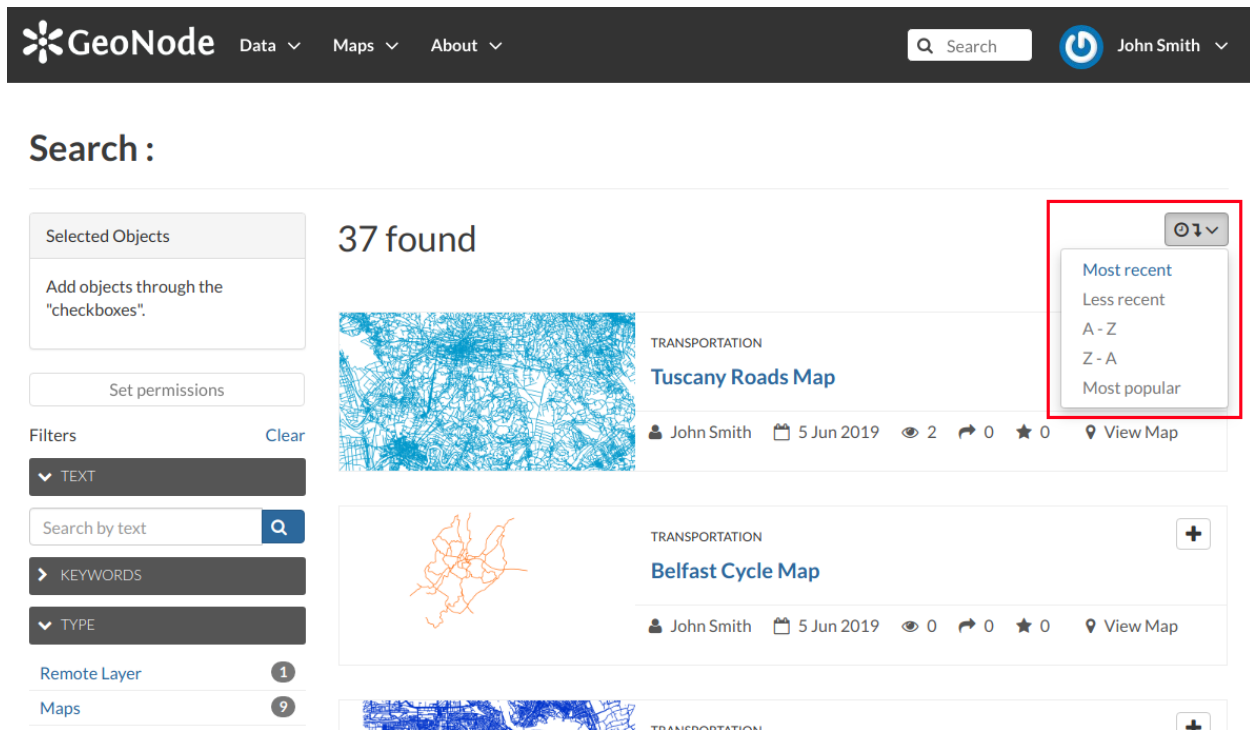


Fig. 43: Ordering Data

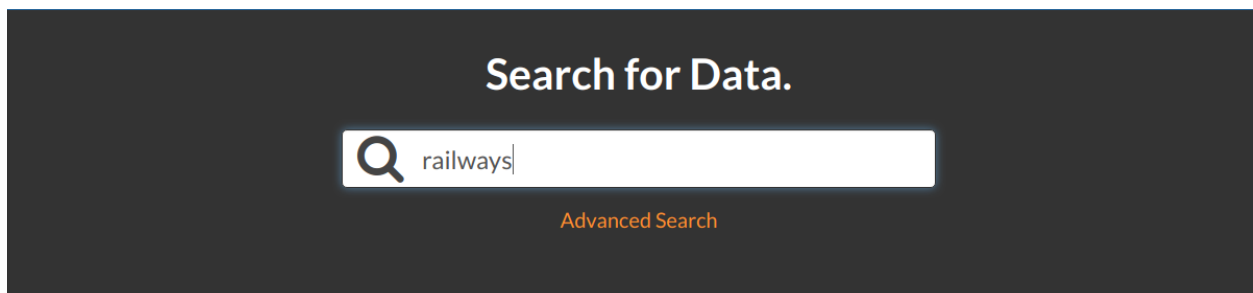


Fig. 44: Searching for data

The screenshot shows the GeoNode web interface. At the top is a dark navigation bar with the GeoNode logo, links for Data, Maps, and About, a search bar, and a user profile for John Smith. Below the navigation bar, the page title is "Search : railways". On the left side, there is a sidebar with a "Selected Objects" section, a "Set permissions" button, and a "Filters" section. The "Filters" section includes a "TEXT" filter with a search input field containing the word "railways" (highlighted with a red circle), and several other filter categories like KEYWORDS, TYPE, CATEGORIES, OWNERS, DATE, REGIONS, and EXTENT. The main content area displays "2 found" results. The first result is "OSM Railways" by John Smith, dated 4 Jun 2019, with 7 views and 0 likes. It has a placeholder image that says "no image". The second result is "railways" by John Smith, dated 4 Jun 2019, with 9 views and 0 likes. It has a thumbnail map showing a network of blue lines representing railways. At the bottom right of the results area, there is a pagination control showing "page 1 of 1".

GeoNode Data Maps About Search John Smith

Search : railways

2 found

Selected Objects

Add objects through the "checkboxes".

Set permissions

Filters Clear

TEXT

railways

KEYWORDS

TYPE

Map 1

Vector Layer 1

CATEGORIES

OWNERS

DATE

REGIONS

EXTENT

no image

OSM Railways

Railways extracted from OSM

John Smith 4 Jun 2019 7 0 0 View Map

railways

No abstract provided

John Smith 4 Jun 2019 9 0 0 Create a Map

page 1 of 1

Fig. 45: Results of searching made by name

- In the *Home* page section shown below are listed all the categories available in the GeoNode instance you are using. You can search for data by category by clicking on it.

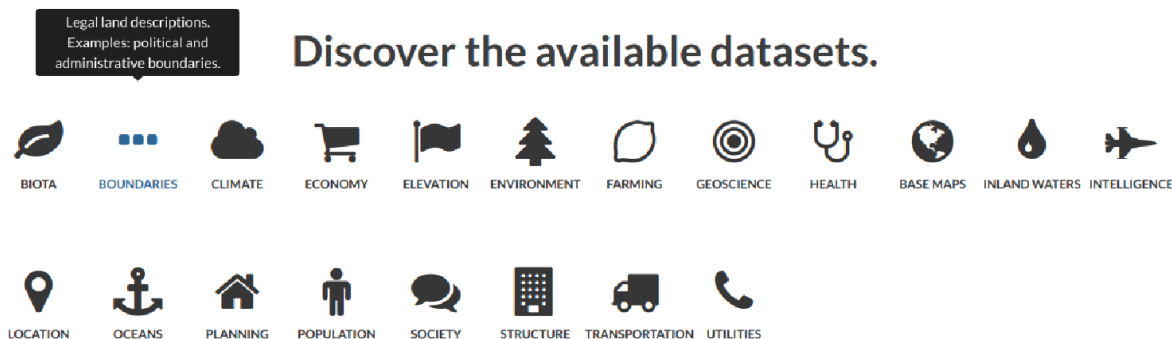


Fig. 46: Searching for datasets by category

In the *Search* page, data will be filtered by that category.

- The *Featured Datasets* section (see the picture below) shows you aggregate data about *Layers*, *Maps*, *Documents* and *Users*. You can trigger a search on layers by clicking on the *Layers* icon, the same happens for *Maps*, *Documents* and *Users*. The *Explore all datasets* drive you to the *Search* page with no filter on data types. In this section there are also useful quick links to add new resources: the *Add layers* drives you to the layer uploading page, the *Add documents* to the document uploading page and the *Create maps* guide you to the map creation.

For each data type GeoNode makes available an individual *Search* page, the next paragraphs will explain that in depth. For *Users* see [Viewing other users information](#).

Documents

When you are searching for *Documents* you can:

- use the *Documents* quick link of the *Featured Datasets* section as described above
- click on the *Documents* link of the *Data* menu in the navigation bar

The *Documents* search page looks like the generic one but only *Document* is considered as data type. You can filter documents by CATEGORIES, as in the example below, or by TEXT, KEYWORDS and so on. You can also use more than one filter at the same time.

Layers

To find *Layers* you can:


- use the *Layers* quick link of the *Featured Datasets*
- click on the *Layers* link of the *Data* menu in the navigation bar

In the *Layers* search page only *Layer* will be considered as data type. You can set one or more filter to refine the search. In the example below the layers have been filtered by EXTENT and CATEGORIES.

The screenshot shows the GeoNode web interface. At the top is a dark navigation bar with the GeoNode logo, links for Data, Maps, and About, a search bar, and a user profile for John Smith. Below the navigation bar, the page is titled "Search :". On the left side, there is a sidebar with various filter options: "Selected Objects" (with instructions to add objects through checkboxes and a "Set permissions" button), "Filters" (with a "Clear" button), and several expandable categories: "TEXT", "KEYWORDS", "TYPE" (showing "Vector Layer" with 1 item), "CATEGORIES" (showing "Boundaries" with 1 item, "climatologyMeteorolo..." with 1 item, and "Transportation" with 1 item), "OWNERS", "DATE", "REGIONS", and "EXTENT". The "Boundaries" category is highlighted with a red circle. The main content area shows "1 found" results. The first result is a map of Italy with the title "BOUNDARIES" (circled in red) and subtitle "Com2016_WGS84_g". Below the title, it says "Administrative boundaries of Italian municipalities". The result is attributed to John Smith, dated 5 Jun 2019, with 0 views, 0 shares, and 0 stars. A "Create a Map" link is also present. On the right side of the result, there is a "+" icon and a "page 1 of 1" indicator.

Fig. 47: Results of searching made by category


Featured Datasets



21 Layers

Click to search for geospatial data published by other users, organizations and public sources. Download data in standard formats.


[Add layers »](#)



6 Maps

Data is available for browsing, aggregating and styling to generate maps which can be saved, downloaded, shared publicly or restricted to specify users only.


[Create maps »](#)



1 Document

As for the layers and maps GeoNode allows to publish tabular and text data, manage their metadata and associated documents.

[Add documents »](#)



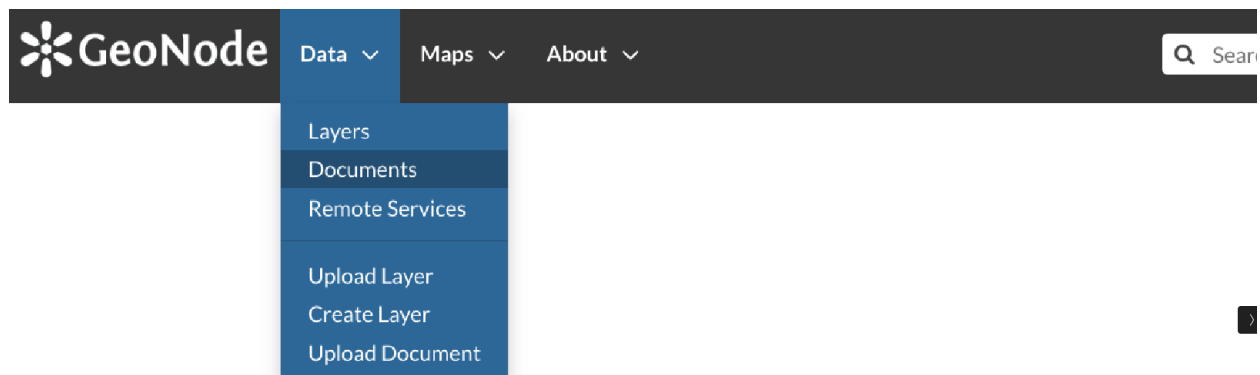
208 Users

Geonode allows registered users to easily upload geospatial data and various documents in several formats.

[See users »](#)

[Explore all datasets](#)

Fig. 48: *Featured Datasets*



The image shows the top navigation bar of the GeoNode website. The bar is dark grey with the GeoNode logo on the left. To the right of the logo are three menu items: 'Data', 'Maps', and 'About', each with a downward arrow. The 'Data' menu is currently open, showing a list of options: 'Layers', 'Documents', 'Remote Services', 'Upload Layer', 'Create Layer', and 'Upload Document'. On the far right of the navigation bar is a search bar with a magnifying glass icon and the text 'Search'. A small dark square button with a right-pointing arrow is located at the bottom right of the navigation bar.

Fig. 49: *Link for Documents*

The screenshot shows the GeoNode interface. At the top, the header includes the GeoNode logo, navigation links for Data, Maps, and About, a search bar, and a user profile for John Smith. The main section is titled 'Explore Documents' and includes an 'Upload Documents' button. On the left, a sidebar contains filters for Text, Keywords, Document Type, Images (2), Categories, Owners, Date, Regions, and Extent. The 'Categories' filter is expanded, showing 'Boundaries' (1) and 'Society' (1). The 'Boundaries' category is circled in red. The main content area displays '1 Documents found'. A document titled 'old_italian_boundaries.jpg' by 'GEOSOLUTIONS' is shown, with a thumbnail image of a map of Italy. The document title 'old_italian_boundaries.jpg' is circled in red. The document details include the owner 'John Smith', the date '5 Jun 2019', and zero views, shares, and stars. A pagination bar at the bottom right shows 'page 1 of 1'.

Fig. 50: Documents filtered by categories

The screenshot shows the GeoNode header with the 'Data' dropdown menu open. The menu options are: Layers, Documents, Remote Services, Upload Layer, Create Layer, and Upload Document. The 'Layers' option is highlighted in blue.

Fig. 51: Link for Layers

Fig. 52: Layers filtered by extent

Maps

If you are searching for *Maps* you can:

- use the *Maps* quick link of the *Featured Datasets* section as described above
- click on the *Explore Maps* link of the *Maps* menu in the navigation bar

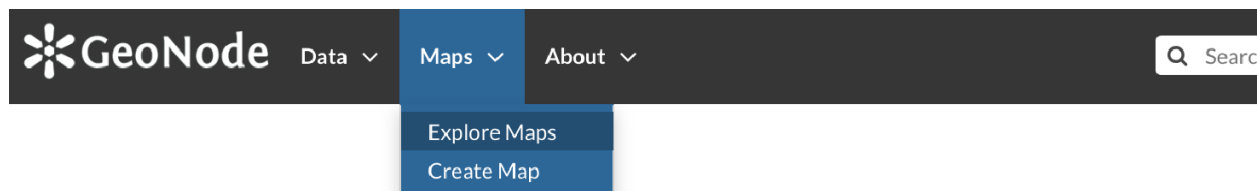


Fig. 53: Link for Maps

As seen for the other data types, the *Maps* search page allows you to filter your maps by a combination of criteria. The example below shows maps filtered by REGIONS.

Fig. 54: Maps filtered by regions

1.10.4 Managing Documents

In this section all the aspects concerning *Documents* will be discussed.

You will learn how to upload a document and how to inspect its metadata and details. All the editing tools will be also explained.

Uploading Documents

GeoNode allows to share reports, conceptual notes, posters, spreadsheets, etc. A wide range of documents files can be hosted on the platform, including text files (.doc, .docx, .txt, .odt), spreadsheets (.xls, .xlsx, .ods), presentations (.ppt, .pptx, .odp), images (.gif, .jpg, .png, .tif, .tiff), PDF, zip files (.rar, .zip, .gz), SLD, XML or QML files.

Warning: Only authenticated users can upload data into GeoNode.

Documents uploading is accessible from two positions:

- the *Upload Documents* button of the *Documents Search* page (see *Documents*)
- the *Upload Document* link of the *Data* menu in the navigation bar

The *Document Upload* page looks like the one shown in the picture below.

In order to upload a document:

1. select a file from your disk or enter a URL address if the document is stored on the internet
2. insert the title of the document
3. select one or more published resources the document can be linked to (optional)

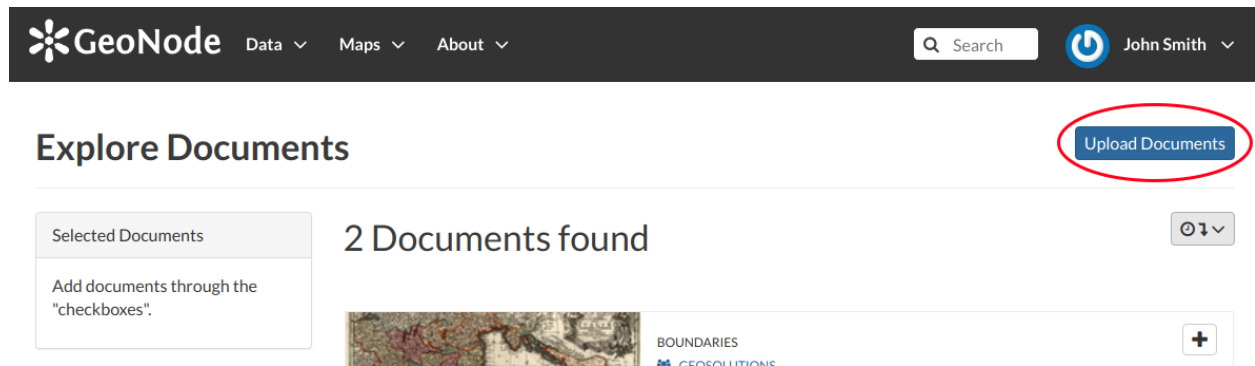


Fig. 55: Documents Upload button

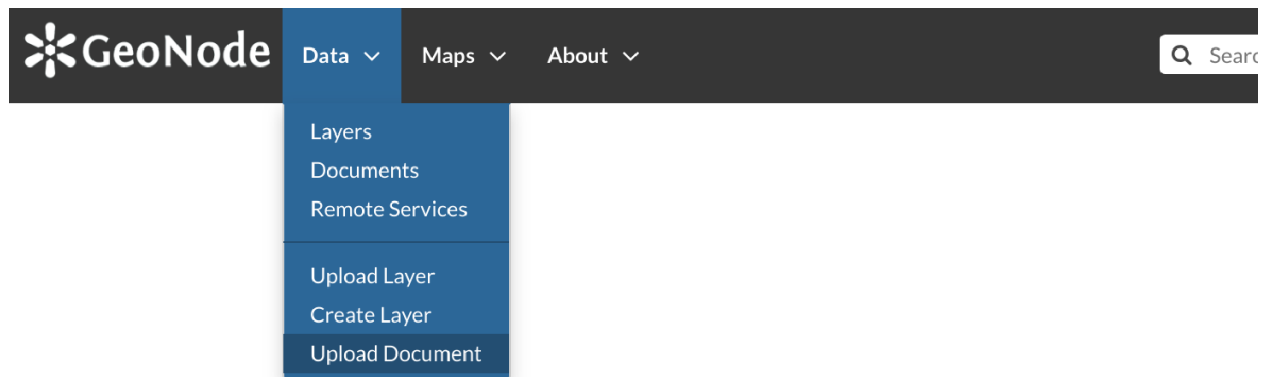


Fig. 56: Document Upload link

Fig. 57: Document Upload page

4. click the red *Upload* button

At the end of the uploading process you will be driven to the *Metadata* page to fill out other information about the document. See the next section to know more about that.

Filling the Document Metadata

Metadata contains all the information related to the document: they are its ID card. They provide essential information for its identification and its comprehension. Metadata also make the document more easily retrievable through search by other users.

Editing a document's metadata is done in three steps (*Basic Metadata*, *Location and Licenses*, *Optional Metadata*). The first two steps are mandatory (no documents will be published if the required information are not provided) whereas the last one is optional.

1. On the **Basic Metadata** page, the essential information that has to be filled is:
 - The *Title* of the document, which should be clear and understandable;
 - The *Resources* the document should be linked to;
 - An *Abstract* on the document;
 - The *Creation/Publication/Revision* dates which define the time period that is covered by the document;
 - The *Keywords*, which should be chosen within the available list. The contributor search for available keywords by clicking on the searching bar, or on the folder logo representing, or by entering the first letters of the desired word. Key-words should be relevant to the imported document;

- The *Category* in which the document belongs;
- The *Group* to which the document is linked.

GeoNode Data ▾ Maps ▾ About ▾ John Smith ▾

Metadata for Old italian boundaries Completeness: 93% Check Schema mandatory fields

[Edit](#) [Settings](#)

1 Basic Metadata **2 Location and Licenses** **3 Optional Metadata**

Title
Old italian boundaries

Link to
✖ Com2016_WGS84_g (layer)

Abstract
Old italian boundaries

Date type
Publication

Date
2019-06-05 04:00

Category
Boundaries

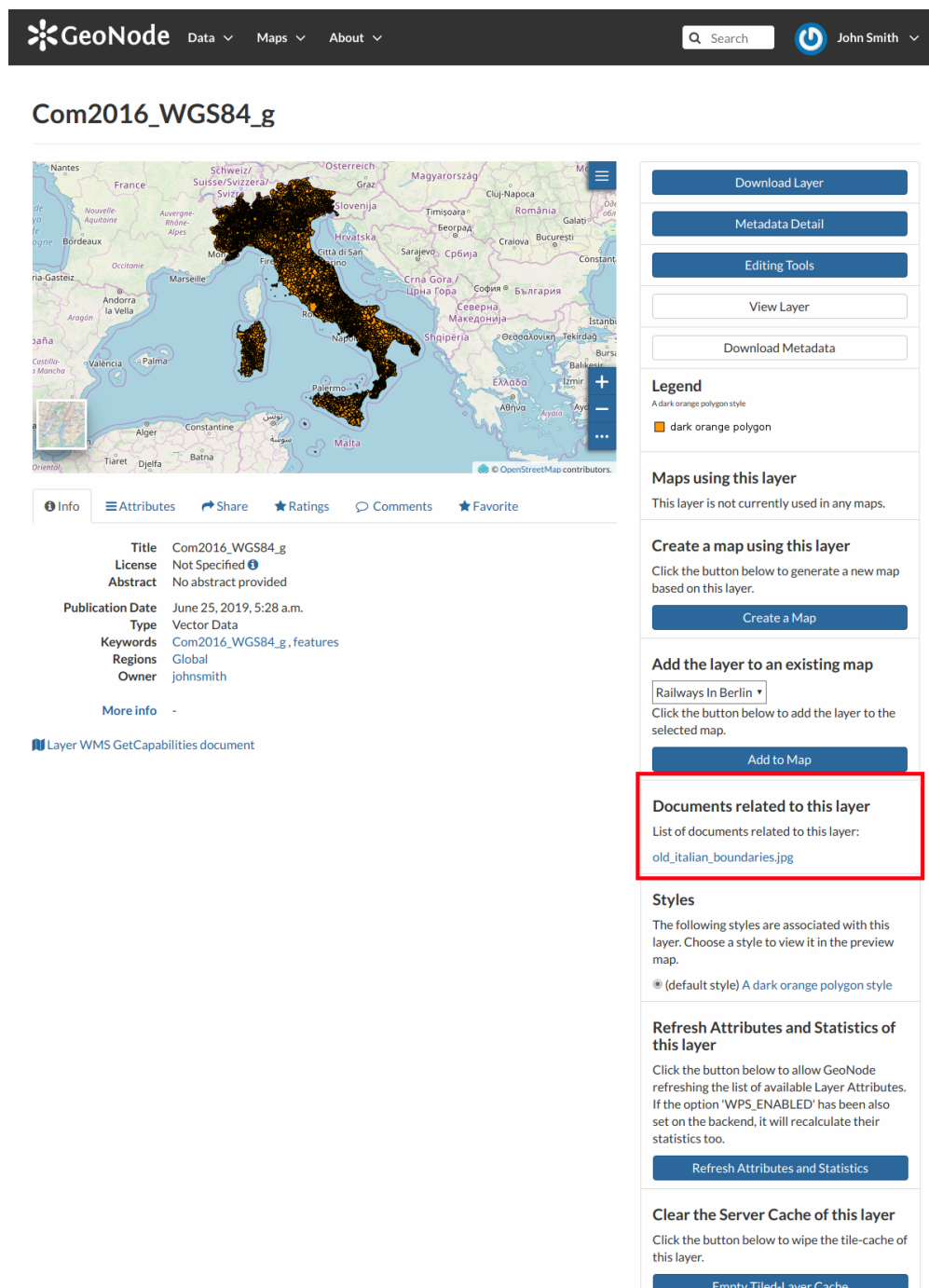
Group
Geosolutions

[Update](#) [Next >>](#)

Fig. 58: Document Basic Metadata

Once all the fields are filled, click on the blue button *Next >>* in the bottom right corner of the page.

Note: When a document is linked to some resources, you can see that link on the *Resource Page*.



GeoNode Data Maps About

Search John Smith

Com2016_WGS84_g

Download Layer

Metadata Detail

Editing Tools

View Layer

Download Metadata

Legend
A dark orange polygon style
dark orange polygon

Maps using this layer
This layer is not currently used in any maps.

Create a map using this layer
Click the button below to generate a new map based on this layer.
Create a Map

Add the layer to an existing map
Railways In Berlin
Click the button below to add the layer to the selected map.
Add to Map

Documents related to this layer
List of documents related to this layer:
old_italian_boundaries.jpg

Styles
The following styles are associated with this layer. Choose a style to view it in the preview map.
(default style) A dark orange polygon style

Refresh Attributes and Statistics of this layer
Click the button below to allow GeoNode refreshing the list of available Layer Attributes. If the option 'WPS_ENABLED' has been also set on the backend, it will recalculate their statistics too.
Refresh Attributes and Statistics

Clear the Server Cache of this layer
Click the button below to wipe the tile-cache of this layer.
Empty Tiled Layer Cache

Info Attributes Share Ratings Comments Favorite

Title Com2016_WGS84_g
License Not Specified
Abstract No abstract provided
Publication Date June 25, 2019, 5:28 a.m.
Type Vector Data
Keywords Com2016_WGS84_g, features
Regions Global
Owner johnsmith
[More info](#)

[Layer WMS GetCapabilities document](#)

Fig. 59: Documents linked to a Layer

It will be also visible on the *Document Information* page.

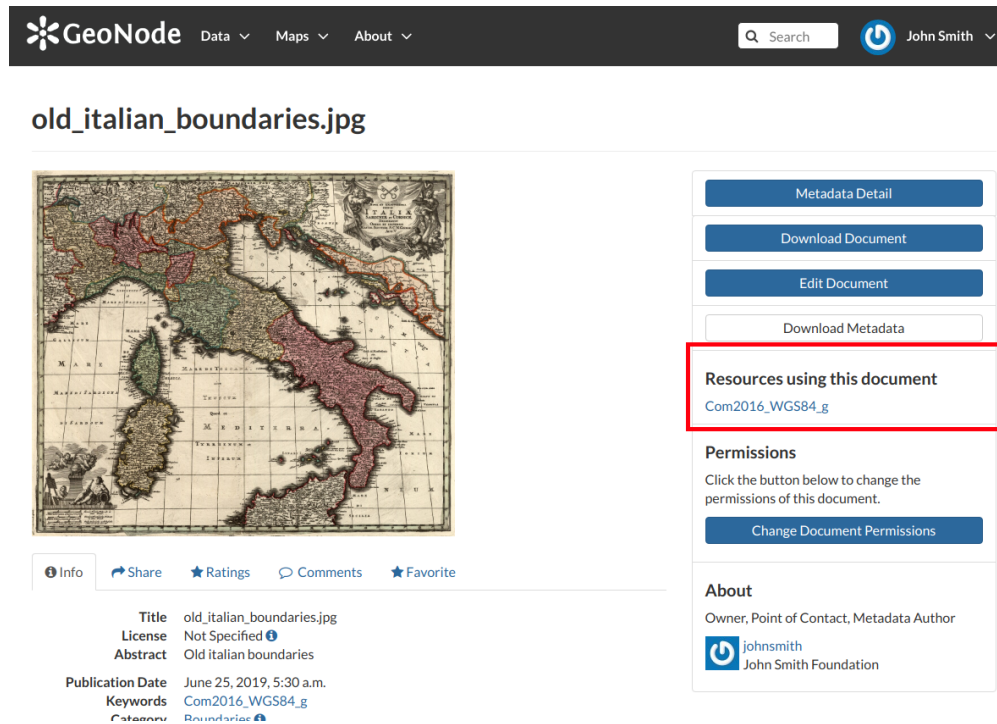


Fig. 60: Resources linked to a Document

2. On the **Location and Licenses** page, the following information should be filled:

- The *Language* of the document;
- The *Regions*, which informs on the spatial extent covered by the document. Proposed extents cover the following scales: global, continental, regional, national;
- The *Data Quality statement* (general explanation of the data producer's knowledge about the lineage of a dataset);
- Potential *Restrictions* to sharing the document should be provided in the Restrictions box.

Click on the blue button *Next >>* to go ahead to the next step.



3. On the **Optional Metadata** page, complementary information can be added:

- The *Edition* to indicate the reference or the source of the document;
- The *Purpose* of the document and its objectives;
- Any *Supplemental information* that can provide a better understanding of the uploaded document;
- The *Maintenance frequency* of the document;
- The *Spatial representation type* used.

Responsible Parties, *Owner* and *Permissions* are listed on the right side of the page, you can edit them.

If all the mandatory information is filled out the document can be published, if not the *Completeness* progress bar warns you that something is missing.

Click on the blue button *Update* to save information on the system.

 Data ▾ Maps ▾ About ▾  John Smith ▾

Metadata for Old italian boundaries

Completeness
✖ Check Schema mandatory fields
93 %

Edit Settings

Mandatory

Mandatory

Optional

1

2

3

Basic Metadata

Location and Licenses

Optional Metadata

Language

English ▾

License

Open Data Commons Open Database Licens ▾

Regions

✕ Global

✕ Italy

✕ Europe

Data quality statement

Old maps digitalized

Restrictions

exclusive right to the publication, production ▾

Restrictions other



No other restrictions

<< Back

Update

Next >>

Fig. 61: Document Location and Licenses

 Data ▾ Maps ▾ About ▾  John Smith ▾

Metadata for Old italian boundaries

Completeness

✖ Check Schema mandatory fields

93 %

Edit Settings

Mandatory

Mandatory

Optional

1

2

3

Basic Metadata

Location and Licenses

Optional Metadata

Other, Optional, Metadata

temporal extent starttemporal extent end

Maintenance frequency

Spatial representation type

Responsible Parties

Point of Contact

Owner and Permissions

Owner

Metadata Author

1.0

2019-06-05 17:2019-07-05 17:

frequency of maintenance for the data is not known

grid data is used to represent geographic data

johnsmith

johnsmith

johnsmith

Purpose

Supplemental information

John Smith wants to share information with other user

No information provided


<< Back


Update

Fig. 62: Document Optional Metadata

Document Information

From the *Documents Search Page* (see [Documents](#)) you can select the document you are interested in and see some basic information about it. You can access the document details page by clicking on its name. That page looks like the one shown in the picture below.


[Data](#) [Maps](#) [About](#)

 John Smith


[Metadata Detail](#)
[Download Document](#)
[Edit Document](#)
[Download Metadata](#)
Resources using this document
[Com2016_WGS84_g](#)
Permissions
 Click the button below to change the permissions of this document.
[Change Document Permissions](#)
About
 Owner, Point of Contact, Metadata Author
 johnsmith
 John Smith Foundation

Info
 Share
 Ratings
 Comments
 Favorite

Title Old italian boundaries

License Not Specified ⓘ

Abstract Old italian boundaries

Publication Date June 5, 2019, 4:51 a.m.

Keywords [Com2016_WGS84_g](#)

Category [Boundaries](#) ⓘ

Regions [Global](#)

Owner [johnsmith](#)

Group [Geosolutions](#)

More info -

Restrictions exclusive right to the publication, production, or sale of the rights to a literary, dramatic, musical, or artistic work, or to the use of a commercial print or label, granted by law for a specified period of time to an author, composer, artist, distributor

Language English

Data Quality good

Supplemental Information No information provided

Fig. 63: *Document Information page*

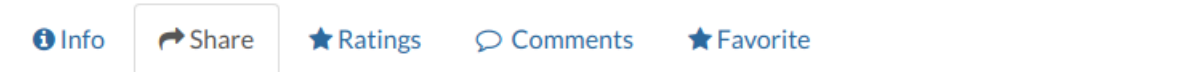
On the page of a document, the resource is either directly displayed on the page or accessible by clicking on the link provided under the title.

Exploring the Tabs Sections

There is a *Tab Section* below the document, where you can first view *Info* about the document.

The **Info Tab** section shows the document metadata such as its title, abstract, date of publication etc. The metadata also indicates the user who is responsible for uploading and managing this content, as well as the group to which it is linked.

The **Share Tab** provides the social media links for the document to share. There is also a link to share the document through email.

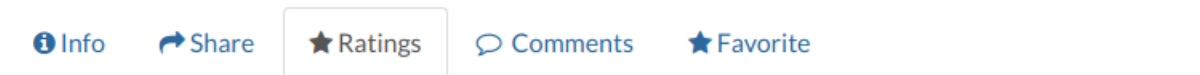


Share This Document

- [Email](#)
- [Facebook](#)
- [Twitter](#)
- [Google +](#)

Fig. 64: Document Sharing

You can **Rate** the document through the *Ratings system*.



Rate this document



Average Rating



Fig. 65: Rate the Document

In the **Comments Tab** section you can post your comment. Click on *Add Comment*, insert your comment and click *Submit Comment* to post it.

Your comment will be added next to the last already existing comment. If you want to remove it click on the red *Delete* button.

If you want this document in your *Favorites* (see [Updating the Profile](#)), open the **Favorite Tab** and click on *Add to Favorites*.

GeoNode also supports the *EXIF* (*EXchangeable Image Format*) for `jpeg` and `tiff` image documents. The *EXIF* means that additional information (metadata) are stored within the image, so GeoNode allows you to see those information in the **Exif Tab**.

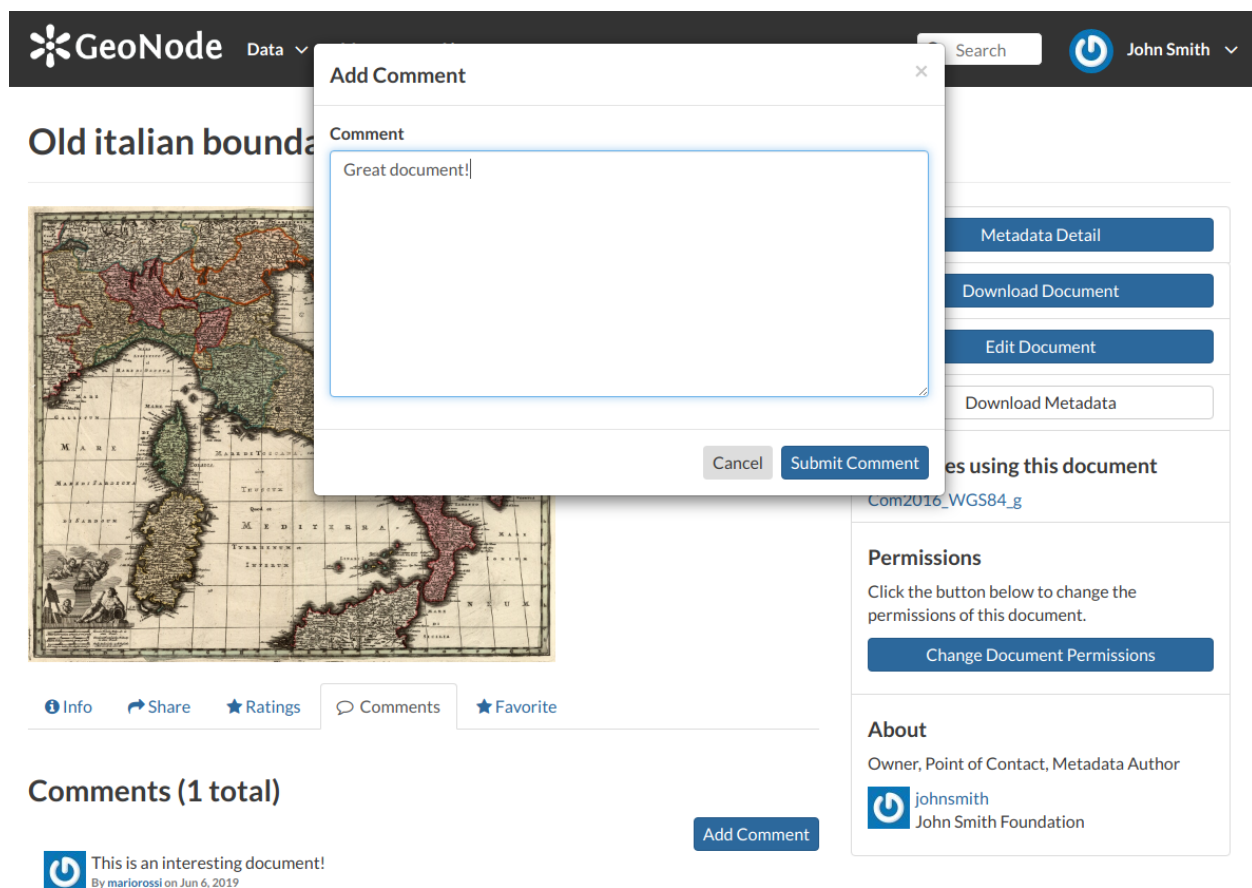


Fig. 66: *Document Comments*

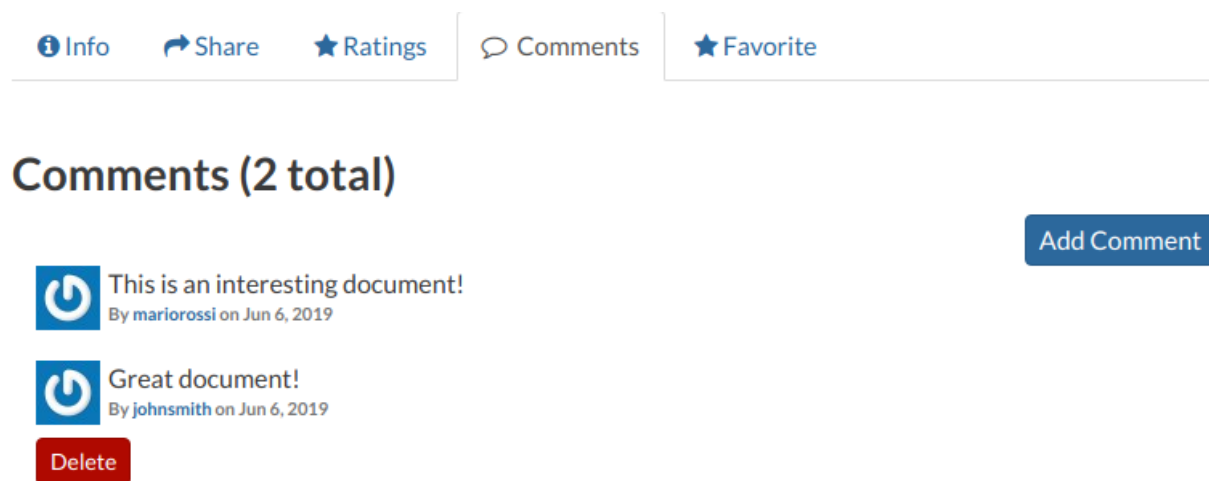


Fig. 67: *Your Comment*


[Info](#) [Share](#) [★ Ratings](#) [Comments](#) [★ Favorite](#)

Favorite


Add to Favorites

[Go to Favorites](#)

Fig. 68: *Your Favorite Comment*

 [Data](#) [Maps](#) [About](#) [John Smith](#)

exif_image.jpg



[Info](#) [Share](#) [★ Ratings](#) [Comments](#) [Exif](#) [★ Favorite](#)

Width	640
Height	480
Make	NIKON
Model	COOLPIX P6000
Date	Nov. 1, 2008, 9:15 p.m.
Latitude	43.464455
Longitude	11.8814783333
Flash	True
Speed Rating	64


[Metadata Detail](#)
[Download Document](#)
[Edit Document](#)
[Download Metadata](#)
Resources using this document
This document is not related to any maps or layers
Permissions
Click the button below to change the permissions of this document.
[Change Document Permissions](#)
About
Owner, Point of Contact, Metadata Author
 johnsmith
John Smith Foundation

Fig. 69: *The EXIF tab*

The Tools Section

On the right side of the *Document Page* you can see other useful information such as the links to the resources linked to the document, the document *Owner*, the *Point of Contact* and the *Metadata Author*.

In the same section of the *Document Page* you can find the following useful tool:

- *Metadata Detail* to explore in detail the document metadata (see the next paragraph)
- *Download Document* to download the document
- *Edit Document* to change the document metadata, replace the file etc (see [Document Editing](#))
- *Download Metadata* to download the whole set of metadata in various formats
- *Change Document Permissions* to assign permissions on the document to users and groups (see [Changing the Document Permissions](#)).

Exploring Metadata Details

When clicking on the *Metadata Detail* button the *Metadata Details Page* will open.

It displays the whole set of available metadata about the document.

Metadata are grouped in order to show the following types of information:

- *Identification* to uniquely identify the document
- *Owner*, the user who own the document
- *Information*, the identification image, the Spatial Extent, Projection System and so on
- *Features*, Restrictions, Language and so on
- *Contact Points*, the user available to have a contact
- *References*, various links to the resource information
- *Metadata Author*, the metadata author information

Document Editing

The *Document Information* page makes available useful tools for document editing. Click on the *Edit Document* button to see what you can do to make changes. The picture below shows you the *Editing Panel* that will appear on the screen.

You can *Replace* the document file with another one by clicking on *Replace*. It will drive you to the *Document Upload* page (see [Uploading Documents](#)) where you can upload a new file.

The *Remove* button allows you to delete the document. You will have to confirm that choice.

The *Editing Panel* shows you also some links for editing the metadata and the thumbnail. These actions will be explained more in depth in the next paragraphs.

[Metadata Detail](#)

[Download Document](#)

[Edit Document](#)

[Download Metadata](#)

Resources using this document

[Com2016_WGS84_g](#)

Permissions

Click the button below to change the permissions of this document.

[Change Document Permissions](#)

About

Owner, Point of Contact, Metadata Author


 **johnsmith**
John Smith Foundation

Fig. 70: Document useful tool

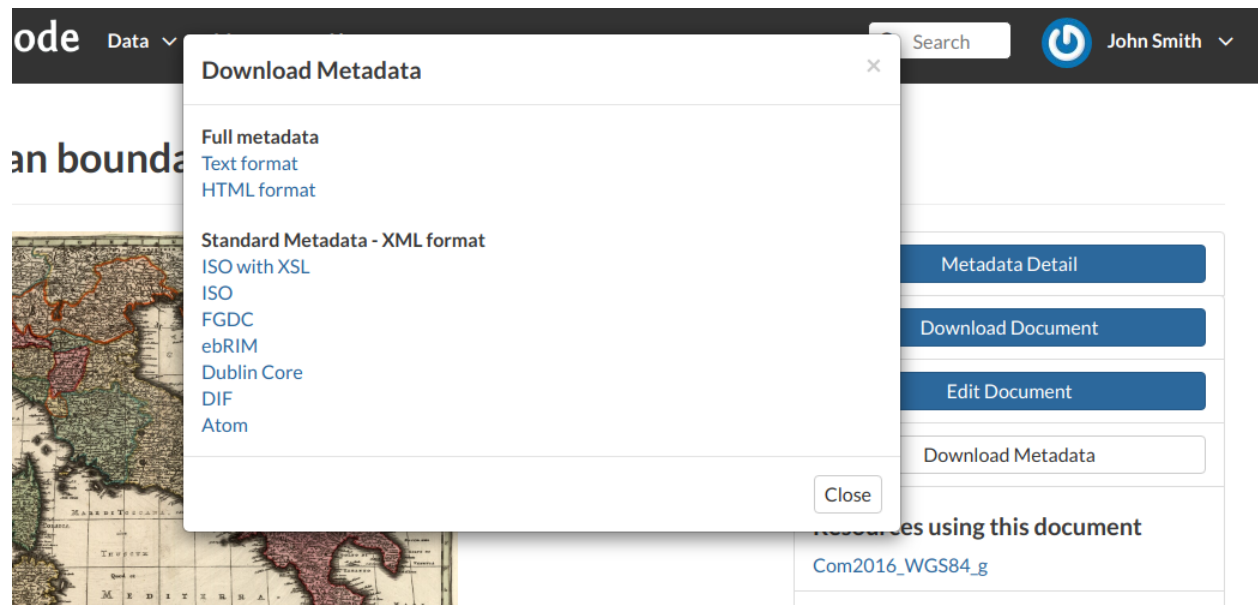


Fig. 71: Document Metadata download

Setting the Document Thumbnail

From the *Editing Panel*, it is also possible to *Set the Thumbnail* of the document. Click on *Set* to open the *Thumbnail Uploading* page and choose the image that will illustrate your document. You can either drag and drop it in the *Drop files here* box or selecting from your folders by clicking on *Choose Files*. Once this is done, click on the red button *Upload files*. If the thumbnail has been successfully uploaded you can see it by coming back to the document list. Click on the *Explore Documents* button to check that.

If no errors occur the following message will be shown.


Editing the Document Metadata


You can edit the metadata of your document through the buttons shown in the red rectangle in below picture.

The *Wizard* button drive you to the wizard described in the *Filling the Document Metadata* section. The *Advanced Edit* button takes you to a big form where all the available metadata of the document can be edited.

Some information are mandatory such as the *Title* or the *Category* the document belongs to, some others are optional.

In the example shown in the picture above, the information inside the red rectangles have been changed. To save the changes click on *Update*, you will be redirected to the document page.


[Data](#)
[Maps](#)
[About](#)

 John Smith

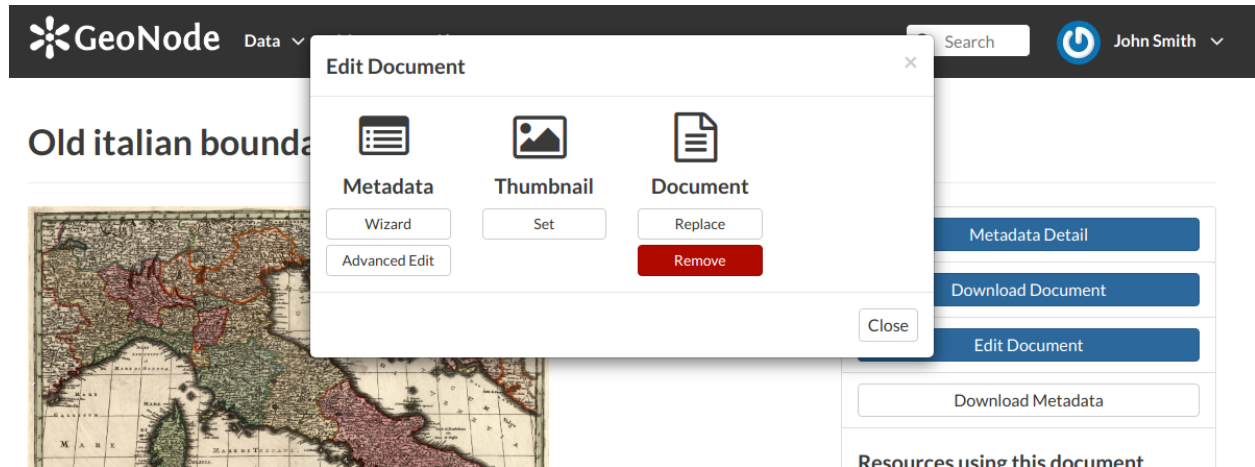


Fig. 73: Document Editing panel

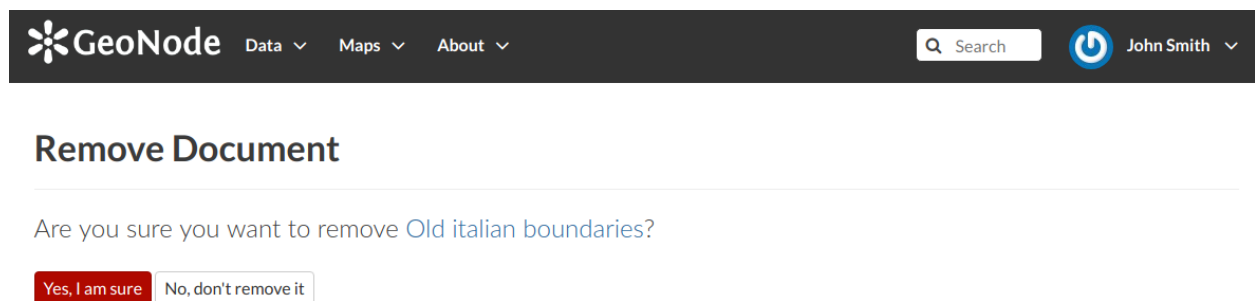





Fig. 74: Document Removal confirmation

 [Data](#) [Maps](#) [About](#)  John Smith [▼](#)

Upload Document's Thumbnail

[Explore Documents](#)



Drop files here

or select them one by one:

Choose Files

Files to be uploaded

420884656472

JPEG

420884656472.jpg [Remove](#)

Clear [Upload files](#)

Permissions

Who can view it?

☒ Anyone

The following users:

* johnsmith

The following groups:

Choose groups...

Who can download it?

Who can change metadata for it?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Fig. 75: Upload Document's Thumbnail

Your document was successfully updated

[Document Info](#) [Edit Metadata](#)

Fig. 76: Uploading success

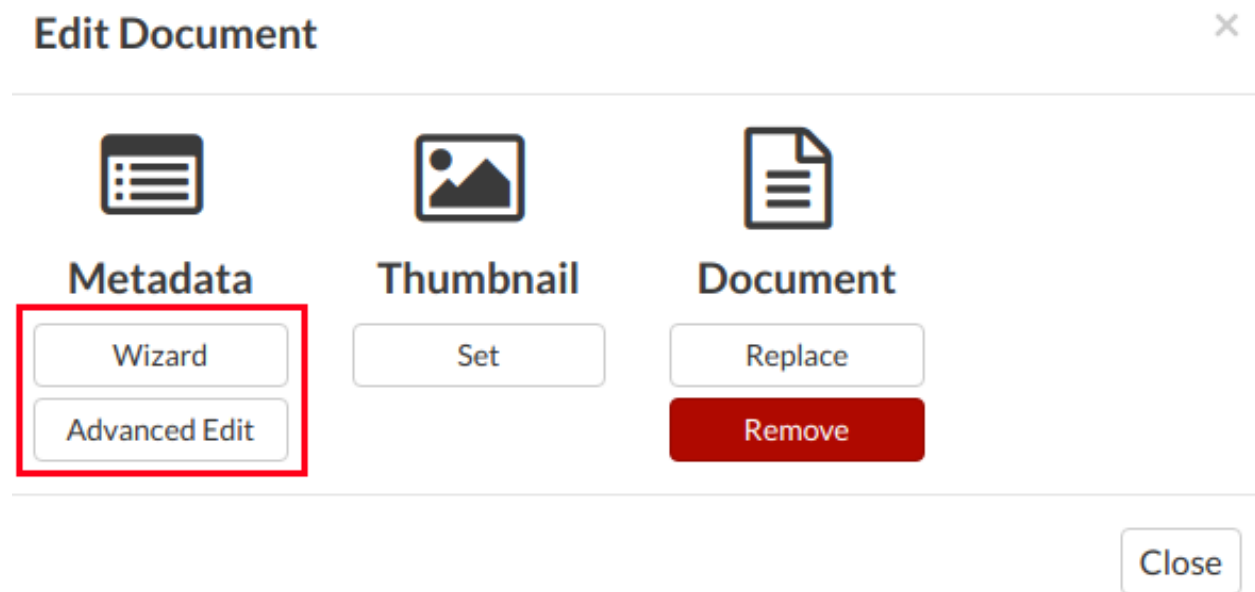


Fig. 77: Editing Metadata

Changing the Document Permissions

GeoNode encourages to publicly, share and make available for download information uploaded on the platform. By default, anyone can see and download a document. However, the document responsible can choose to limit access to the document to some contributors and/or groups.

Through the button shown in the picture below it is possible to manage the document permissions.


The *Change Document Permissions* button on the right side of the document page allows to set up who can:

- View the document;
- Download it;
- Edit its metadata;
- Manage it (update, delete, change permissions, publish/unpublish).

See an example in the picture below.

Usually that editing of metadata and the management of a document are in charge of the responsible of the document, i.e. the contributor who uploaded it and who has those permissions by default.

Once the permissions are set, click *Apply changes* to save them.


[Data](#)
[Maps](#)
[About](#)

[John Smith](#)

Edit Metadata

[Explore Document](#)

Editing details for Old Italian boundaries

[Update](#)

Owner
@joseveth

Title
[Newcastle, Canada in 1760s](#)

Alternate

Date
2019-06-05 04:51 am

Date type
Publication

Edition
v.1

Abstract
[A map from the Mull's map collection](#)

Purpose
[Study](#)

Maintenance frequency
.....

Free-text Keywords
[Com2016_WGS84.g](#)

Region
[Canada](#) [Italy](#)

Restrictions
exclusive right to the publication, production, or sale of the rights to a literary, dramatic, musical, or artistic

Restrictions other

License
Not Specified

Language
English

Spatial representation type
.....

temporal extent start

temporal extent end

Supplemental information
No information provided

Data quality statement
good

Group
geosolutions

☐ Metadata uploaded previous

☐ Featured

☒ Is Published

☒ Approved

Thumbnail url
<http://master.demo.geonode.org/uploads/thumb/document-8d8b794-8777-11e9-a268-0242ac120006-thumb.png>

☐ Dirty State

Point of Contact
@joseveth

Metadata Author
@joseveth

Link to
[Com2016_WGS84.g \(latest\)](#)

Category

- climatology/Meteorology/Atmosphere
- Society
- Utilities Communication
- Oceans
- Health
- Geoscientific Information
- Inland Waters
- Structure
- Intelligence Military
- Climatology Meteorology Atmosphere
- Population


- Imagery Base Maps Earth Cover
- Economy
- Environment
- Bios
- Elevation
- Planning Cadastre
- Boundaries
- Transportation
- Location
- Farming

[Update](#)

 Data ▾ Maps ▾ About ▾

Search John Smith ▾

Nouvelle Carte de l'Italie



[Info](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

Title	Nouvelle Carte de l'Italie
License	Not Specified ⓘ
Abstract	A map from the Moll's map collection
Publication Date	June 5, 2019, 4:51 a.m.
Keywords	Com2016_WGS84_g
Category	Boundaries ⓘ
Regions	Global , Italy
Owner	johnsmith

[Metadata Detail](#)
[Download Document](#)
[Edit Document](#)
[Download Metadata](#)
Resources using this document
[Com2016_WGS84_g](#)
Permissions
Click the button below to change the permissions of this document.
[Change Document Permissions](#)
About
Owner, Point of Contact, Metadata Author
 johnsmith
John Smith Foundation

Fig. 79: The button to change permissions

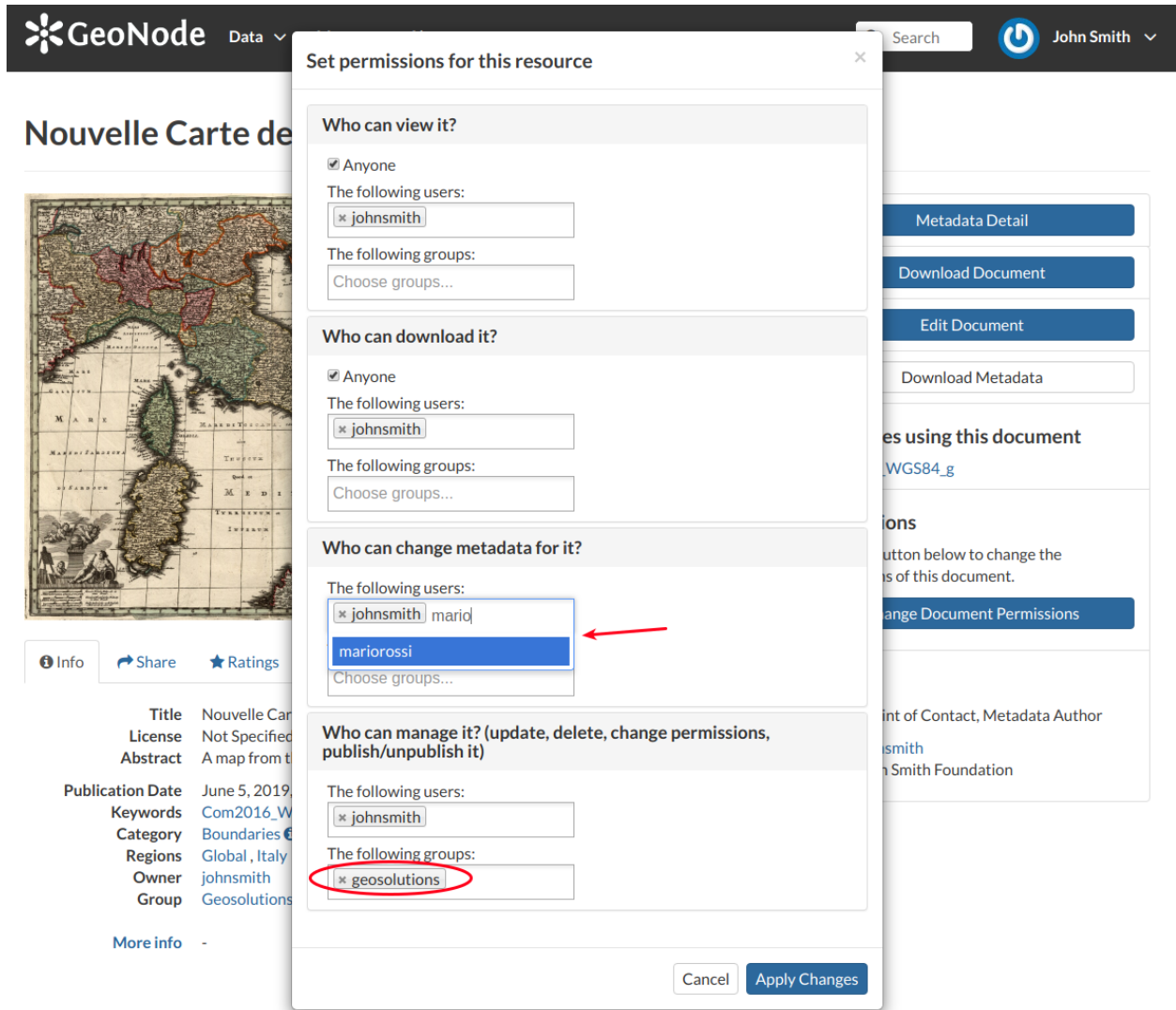


Fig. 80: Changing the Document permissions

1.10.5 Managing Layers

Layers are published resources representing raster or vector spatial data sources. Layers can also be associated with metadata, ratings, and comments.

In this section, you will learn how to create a new layer by uploading a local data set, add layer info, change the style of the layer, and share the results.

Layers Uploading

The most important resource type in GeoNode is the *Layer*. A layer represents spatial information so it can be displayed inside a map.

To better understand what we are talking about let's upload your first layer.

The *Layer Uploading* page can be reached from the *Upload Layer* link of the *Data* menu in the navigation bar.

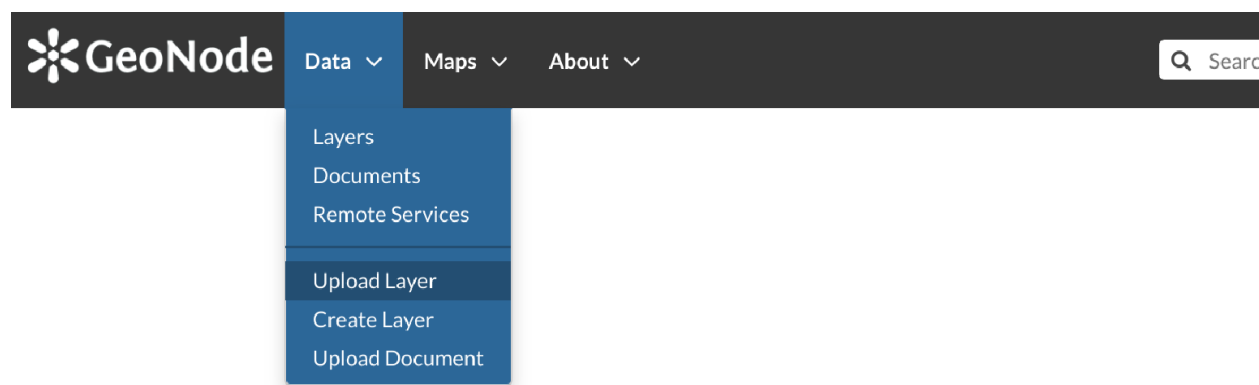


Fig. 81: Link for Layers Uploading

There is also an *Upload Layers* button in the *Layers Page*.

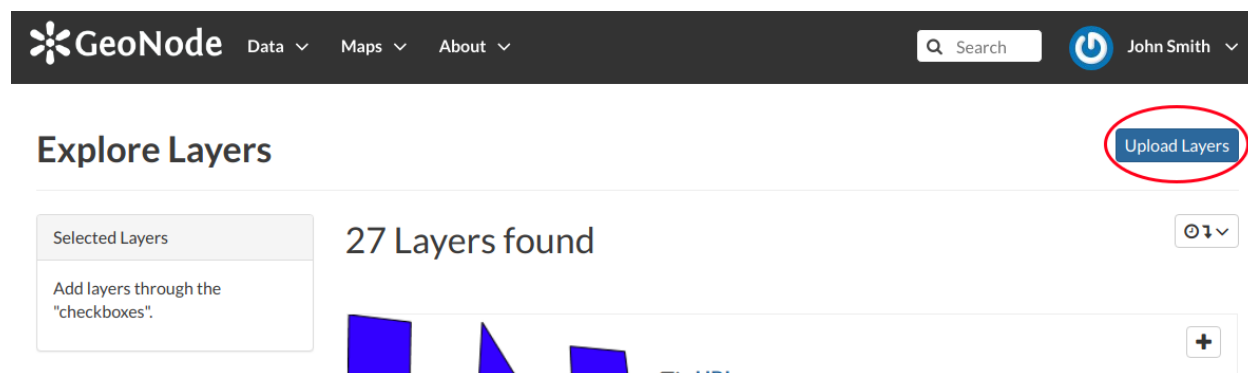


Fig. 82: Button for Layers Uploading

The *Layers Uploading* page looks like the one in the picture below.

Through the *Choose Files* button you can select files from your disk, make sure they are valid raster or vector spatial data. You can also change the default *Permissions* settings (see [Changing the Layer Permissions](#) for further information on how to set permissions).

Fig. 83: *The Layers Uploading page*

Select the *charset*, then click on *Upload files* to start the process or click *Clear* to remove all the loaded files from the page.

Fig. 84: *Shapefile Uploading*

In this example the `roads` ESRI Shapefile, with all its mandatory files (`.shp`, `.shx`, `.dbf` and `.prj`), has been chosen. A progress bar shows the operation made during the layer upload and alerts you when the process is over. When the process ends click the *Layer Info* to check the layer has been correctly uploaded.

Note: There are lot of free spatial dataset available in the Internet. In this example, an extract of the Berlin city center roads map from the [BBBike extracts](#) [OpenStreetMap](#) dataset has been used.

In the next paragraphs you will learn how to create a layer from scratch, how to set permissions, how to explore the layer properties and how to edit them.

Creating a Layer from scratch

An interesting tool that GeoNode makes available to you is the *Create Layer*. It allows you to create a new vector layer from scratch. The *Layer Creation Form* is reachable through the *Create Layer* link shown in the picture below.

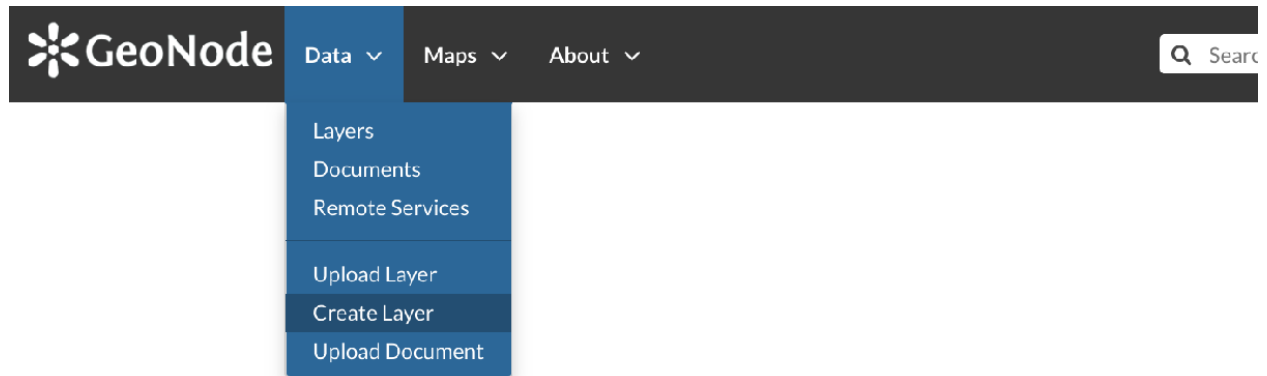


Fig. 85: *Create layer link*

In order to create the new Layer you have to fill out the required fields:

- *Name*
- *Title*
- *Geometry type*

Geometry type

A screenshot of a web form's 'Geometry type' dropdown menu. The dropdown is open, showing a list of options: 'Points', 'Lines', and 'Polygons'. The 'Lines' option is currently selected and highlighted with a light blue background. The dropdown menu has a thin border and a small downward arrow on the right side of the top bar.

Fig. 86: *Geometry types*

Usually the layers features should have some *Attributes* that enrich the amount of information associated with each of them. Through the *Add Attribute* button you can add new attributes.

Fig. 87: *New Layer creation from scratch*

At this time you can also change the default *Permissions* settings, see [Changing the Layer Permissions](#) to learn how.

Once the form has been filled out, click on *Create*. You will be redirected to the *Layer Page* (see [Layer Information](#)). Now your Layer is created but is still empty, no features have been added yet. See the [Layer Editing](#) section to learn how to add new features.

Using Remote Services

In GeoNode you can add new layers not only by loading them from your disk but also using *Remote Services*. In this section you will learn how to add a new service and how to load resources in GeoNode through that.

Let's try it!

Click on the *Remote Services* link of the *Data* menu in the navigation bar.

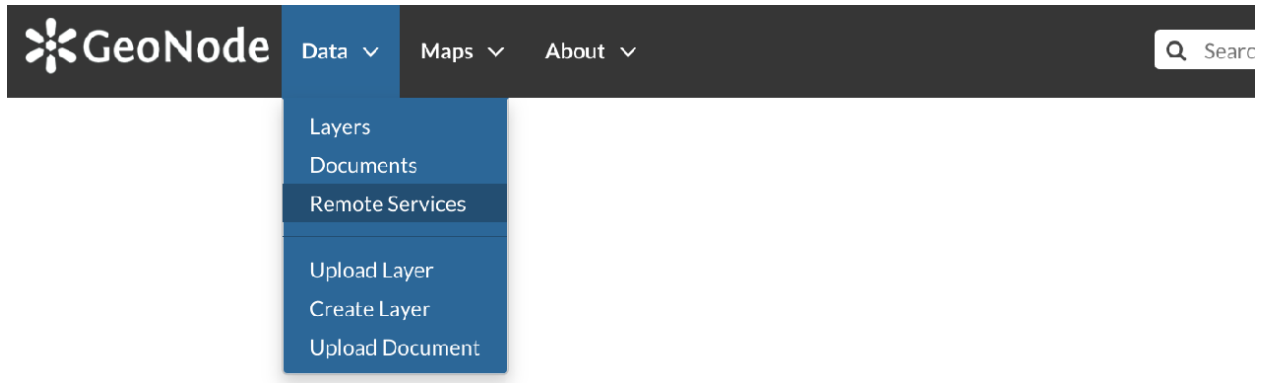


Fig. 88: *Remote Services* link

The page that opens will contain the list of the available services.

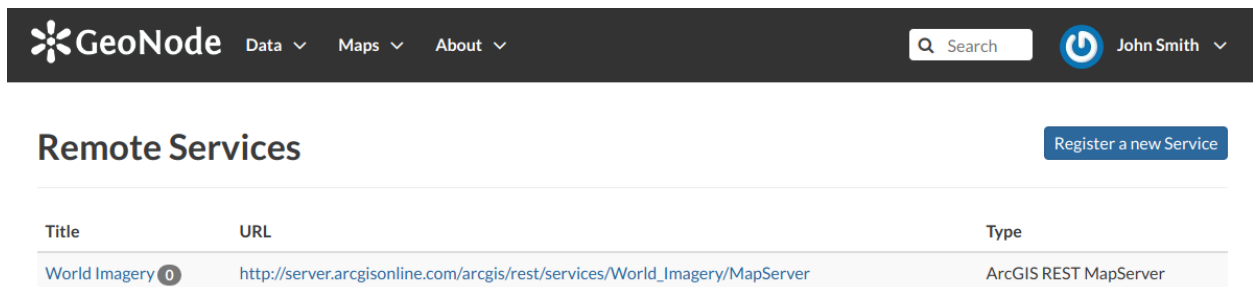


Fig. 89: *Remote Services*

To configure a new service:

- click on *Register a new Service*
- type the *Service URL*
- select the *Service Type*

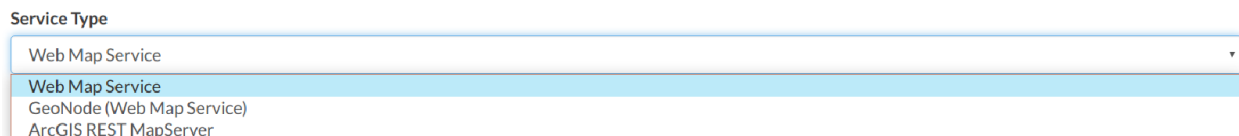


Fig. 90: *Service Types*

- click on *Create*

GeoNode supports three **types of remote services**:

- *Web Map Service*

Generic Web Map Service (WMS) based on a standard protocol for serving georeferenced map images over the Internet. These images are typically produced by a map server (like [GeoServer](#)) from data provided by one or more distributed geospatial databases. Common operations performed by a WMS service are: *GetCapabilities* (to retrieves metadata about the service, including supported operations and parameters, and a list of the available layers) and *GetMap* (to retrieves a map image for a specified area and content).

Note: Lots of WMS services are available on the internet, in this example we used the `https://demo.geo-solutions.it/geoserver/wms`.

- *GeoNode Web Map Service*

Generally a WMS is not directly invoked; client applications such as GIS-Desktop or WEB-GIS are used that provide the user with interactive controls. A GeoNode WMS automatically performs some operations and lets you to immediately retrieve resources.

Note: An example of GeoNode WMS is available at `http://dev.geonode.geo-solutions.it/geoserver/wms`.

- *ArcGIS REST MapServer*

This map service provides basic information about the map, including the layers that it contains, whether the map is cached or not, its spatial reference, initial and full extents, whether the service is allowed to export tiles and max tiles export count, etc. A set of operations that manage the state and contents of the service are allowed: Edit Service, Refresh, Update Tiles. The URL should follow this pattern: `https://<servicecatalog-url>/services/<serviceName>/MapServer`.

Note: Try the following service to better understand how it works: `https://sampleserver6.arcgisonline.com/arcgis/rest/services/USA/MapServer`.

Once the service has been configured, you can load the resources you are interested in through the *Import Resources* page where you will be automatically redirected to. Take a look at the gif below to see the whole process.

Fig. 91: A new Remote Service

From the page where the services are listed, it is possible to click on the *Title* of a service. It opens the *Service Details* page.

Each service has its own metadata such as the *Service Type*, the *URL*, an *Abstract*, some *Keywords* and the *Contact* user. You can edit those metadata through the form available from the *Edit Service Metadata* button of the *Service Details* page (see the picture below).

The screenshot shows the GeoNode web interface. At the top is a dark navigation bar with the GeoNode logo, menu items 'Data', 'Maps', and 'About', a search bar, and a user profile for 'John Smith'. The main content area is divided into two columns. The left column displays the details for a 'GeoServer Web Map Service':

- Type:** Web Map Service
- URL:** <https://demo.geo-solutions.it/geoserver/wms>
- Abstract:** A compliant implementation of WMS plus most of the SLD extension (dynamic styling). Can also generate PDF, SVG, KML, GeoRSS
- Keywords:** GEOSERVER, WFS, WMS
- Contact:** johnsmith

Below this is a section titled 'Service Resources' with a sub-header '2'. It contains a table with two columns: 'Title' and 'Description'.

Title	Description
sfdem	No abstract provided
Regioni Italiane	No abstract provided

The right column is titled 'Manage' and contains three buttons: 'Edit Service Metadata', 'Import Service Resources', and 'Remove Service'.

Fig. 92: Remote Service metadata

Changing the Layer Permissions

When creating or uploading a new Layer you have to set who can view, download, edit and manage that Layer. By default only owners can edit and manage layers, anyone can view and download them.

In order to modify the Layer *Permissions* settings you have to click the *Change the Layer Permissions* button in the Layer page.


Through the *Permissions Settings Panel* you can add or remove permissions for users and groups. The picture below shows an example.


You can set the following types of permissions:

- *View* allows to view the layer;
- *Download* allows to download the layer;
- *Change Metadata* allows to change the layer metadata;
- *Edit Data* allows to change attributes and properties of the layers features;
- *Edit Style* allows to change the layer style;
- *Manage* allows to update, delete, change permissions, publish and unpublish the layer.


Warning: When assigning permissions to a group, all the group members will have those permissions. Be careful in case of editing permissions.

Click on *Apply Changes* to save these settings.


[Data](#)
[Maps](#)
[About](#)

 John Smith

Italian Towers



[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title Italian Towers

License Not Specified

Abstract No abstract provided

Publication Date June 7, 2019, 4:03 a.m.

Type Vector Data


Keywords features , italian_towers

Category Environment

Regions Global

Owner johnsmith

[More info](#)

 [Layer WMS GetCapabilities document](#)

[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Default Point

- Red Square

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Add the layer to an existing map

Regioni Italiane Map

Click the button below to add the layer to the selected map.

[Add to Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

- (default style) Default Point

Refresh Attributes and Statistics of this layer

Click the button below to allow GeoNode refreshing the list of available Layer Attributes. If the option 'WPS_ENABLED' has been also set on the backend, it will recalculate their statistics too.

[Refresh Attributes and Statistics](#)

Clear the Server Cache of this layer

Click the button below to wipe the tile-cache of this layer.

[Empty Tiled-Layer Cache](#)


Permissions

Click the button below to change the permissions of this layer.

[Change Layer Permissions](#)

About

Owner, Point of Contact, Metadata Author

 johnsmith
John Smith Foundation

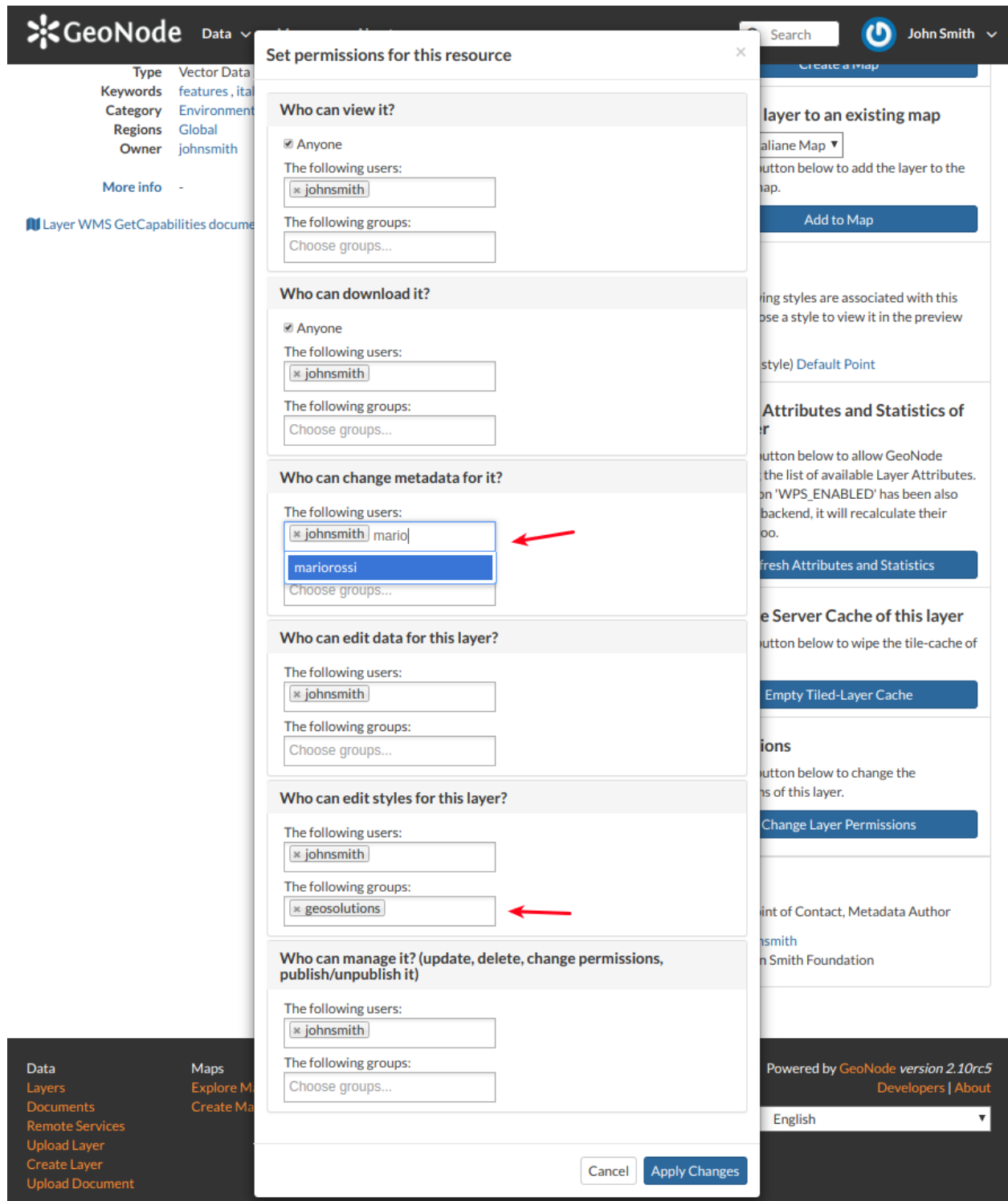


Fig. 94: Layer Permissions settings for users and groups

Layer Information

In this section you will learn more about layers. In the [Layers](#) section we explain how to find layers, now we want to go more in depth showing you how to explore detailed information about that.

From the layers list page, click on the layer you are interested in. The *Layer Page* will open.

As shown in the picture above, the *Layer Page* is divided into three main sections:

1. the *Layer Preview* section, under the title
2. the *Tabs* section, under the layer preview
3. the *Tools* section, on the right side of the page

Layer Preview

The *Layer Preview* shows the layer in a map with very basic functionalities:

- the *Base Map Switcher* that allows you to change the base map;
- the *Zoom in/out* tool to enlarge and decrease the view;
- the *Zoom to max extent* tool for the zoom to fit the layer size;
- the *Query Objects* tool to retrieve information about the map objects by clicking on the map;
- the *Print* tool to print the preview.

The GeoNode map viewer is [MapStore](#) based, see the [MapStore Documentation](#) to learn more.


Tabs Sections


The *Layer Page* shows you some tabs sections containing different information about the layer:

- The tab *Info* is active by default. This tab section shows some layer metadata such as its title, the abstract, date of publication etc. The metadata also indicates the layer owner, what are the topic categories the layer belongs to and which regions are affected.
- The *Attributes* tab shows the data structure behind the layer. All the attributes are listed and for each of them some statistics (e.g. the range of values) are estimated (if possible).
- The *Share* tab provides the links for the layer to share through social media or email.
- You can *Rate* the layer through the *Rating system*.
- In the *Comments* tab section you can post your comment. Click on *Add Comment*, insert your comment and click *Submit Comment* to post it.

Your comment will be added next to the last already existing comment. If you want to remove it click on the red *Delete* button.

- If you want this layer in your *Favorites* (see [Updating the Profile](#)), open the *Favorite* tab and click on *Add to Favorites*.


[Data](#)
[Maps](#)
[About](#)

 John Smith


[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Default Point

- Red Square

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Add the layer to an existing map

Regioni Italiane Map

Click the button below to add the layer to the selected map.

[Add to Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

- (default style) Default Point

Refresh Attributes and Statistics of this layer

Click the button below to allow GeoNode refreshing the list of available Layer Attributes. If the option 'WPS_ENABLED' has been also set on the backend, it will recalculate their statistics too.

[Refresh Attributes and Statistics](#)

Clear the Server Cache of this layer

Click the button below to wipe the tile-cache of this layer.

[Empty Tiled-Layer Cache](#)

Permissions

Click the button below to change the permissions of this layer.

[Change Layer Permissions](#)

About

Owner, Point of Contact, Metadata Author

 johnsmith
John Smith Foundation

Info [Attributes](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

Title Italian Towers
License Not Specified
Abstract No abstract provided
Publication Date June 7, 2019, 4:03 a.m.
Type Vector Data
Keywords features , italian_towers
Category Environment
Regions Global
Owner johnsmith

[More info](#)


[Layer WMS GetCapabilities document](#)

Fig. 96: *Layer Preview*

[Info](#) [Attributes](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

Title	Regioni Italiane
License	Not Specified i
Abstract	No abstract provided
Publication Date	June 7, 2019, 4:49 a.m.
Type	Data
Keywords	features , Reg2015_WGS84_g
Category	Environment i
Regions	Global , Africa , Central Africa , North Africa , Algeria , Tunisia , Europe , Albania , Austria , Bosnia and Herzegovina , Croatia , France , Greece , Holy See (Vatican City) , Hungary , Italy , Liechtenstein , Malta , Monaco , Montenegro , San Marino , Serbia , Slovenia , Switzerland , Pacific
Owner	johnsmith
More info	-
Language	English
Supplemental Information	No information provided

Fig. 97: *Layer Info tab*


[Data](#) [Maps](#) [About](#)

John Smith



[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend
 Default Point
 ■ Red Square

Maps using this layer
 This layer is not currently used in any maps.

Create a map using this layer
 Click the button below to generate a new map based on this layer.
[Create a Map](#)

Add the layer to an existing map

[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)
Fig. 98: *Layer Attributes tab*

[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Share This Layer

- Email
- Facebook
- Twitter
- Google +

Fig. 99: *Layer Sharing*



Fig. 100: *Rate the Layer*



Fig. 101: *Layer Comments*

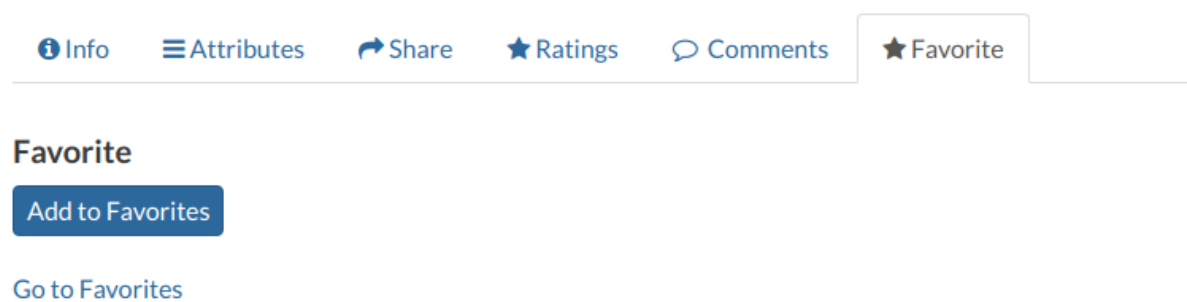


Fig. 102: *Your Favorite Layer*

Layer Tools

In the right side of the *Layer Page* there are some buttons and information that can help you to manage your layer. This paragraph will cover only those tools which show layers information. The *Editing Tools* will be explored in the *Layer Editing* section.

- through the *Download Layer* button you can download your layer with some options, see *Downloading Layers*;
- the *Metadata Detail* button to see the layer metadata, see *Layers Metadata* to read more;
- the *Editing Tools* button allows you to access to many editing tools. Those functionalities will be explained in the *Layer Editing* section;
- the *View Layer* button opens the layer loaded in a map, see the *Map Information* for more details;
- the *Download Metadata* button allows you to download the layer metadata in various formats;
- the *Legend* shows what the symbols and styles on the map are referring to;
- in the *Map using this layer* section all the map which uses the layer are listed;
- in the *Create a map using this layer*, the *Create a Map* button allows you to create a map from scratch using the layer;
- the section *Add the layer to an existing map* shows you a dropdown menu in which all the maps the user can view are listed. The button *Add to Map* allows you to add the layer to the map you have selected in the previous menu;
- the *Styles* section shows all the styles associated with the layer. Click on the checkbox corresponding to one of the styles listed to apply it the preview;
- in the *Refresh Attributes and Statistics of this layer* section the *Refresh Attributes and Statistics* allows GeoNode to refresh the list of available Layer Attributes. If the option ‘WPS_ENABLED’ has been also set on the backend, it will recalculate their statistics too;
- in the *Clear the Server Cache of this layer* section the *Empty Tiled-Layer Cache* allows to wipe the tile-cache of this layer;
- the *About* section shows you the layer *Owner*, the *Contact* user and the *Metadata Author*.

Downloading Layers

At the top of the *Layer Page* there is the *Download Layer* button (see *Layer Information*). It provides access to the ability to extract geospatial data from within GeoNode.

You will see a list of options of the supported export formats. You can choose the *Images* formats PNG, PDF, JPEG if you want to save a “screenshot-like” image of the layer.

You can also download the layer data, the supported export formats will be listed in the *Data* tab. Click on your desired format to trigger the download.

As shown in the image above, GeoNode allows you to download a subset of data. Click on *Do you want to filter it?* to filter the layer data before the download.

roads

[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Maps using this layer
This layer is not currently used in any maps.

Create a map using this layer
Click the button below to generate a new map based on this layer.
[Create a Map](#)

Styles
The following styles are associated with this layer. Choose a style to view it in the preview map.
☐ A orange line style
☒ (default style) Red road

Refresh Attributes and Statistics of this layer
Click the button below to allow GeoNode refreshing the list of available Layer Attributes

Metadata

Title	roads
License	Not Specified ⓘ
Abstract	No abstract provided
Publication Date	June 7, 2019, 2:15 p.m.
Type	Vector Data
Keywords	features, ne_10m_roads
Category	Transportation ⓘ
Regions	Global
Owner	johnsmith

[More info](#) -

[Layer WMS GetCapabilities document](#)

Fig. 103: Change the Layer Style in preview

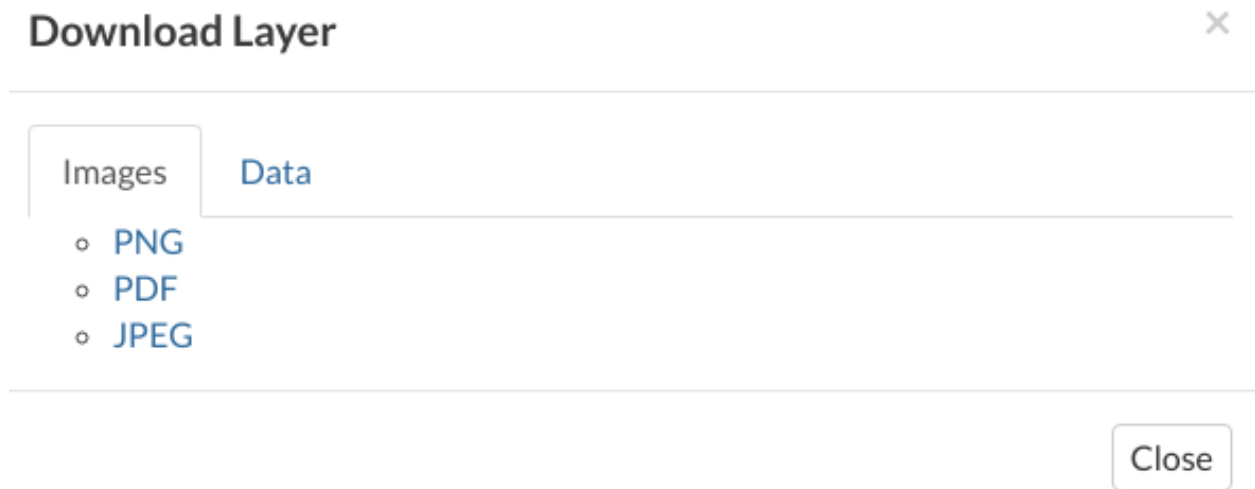
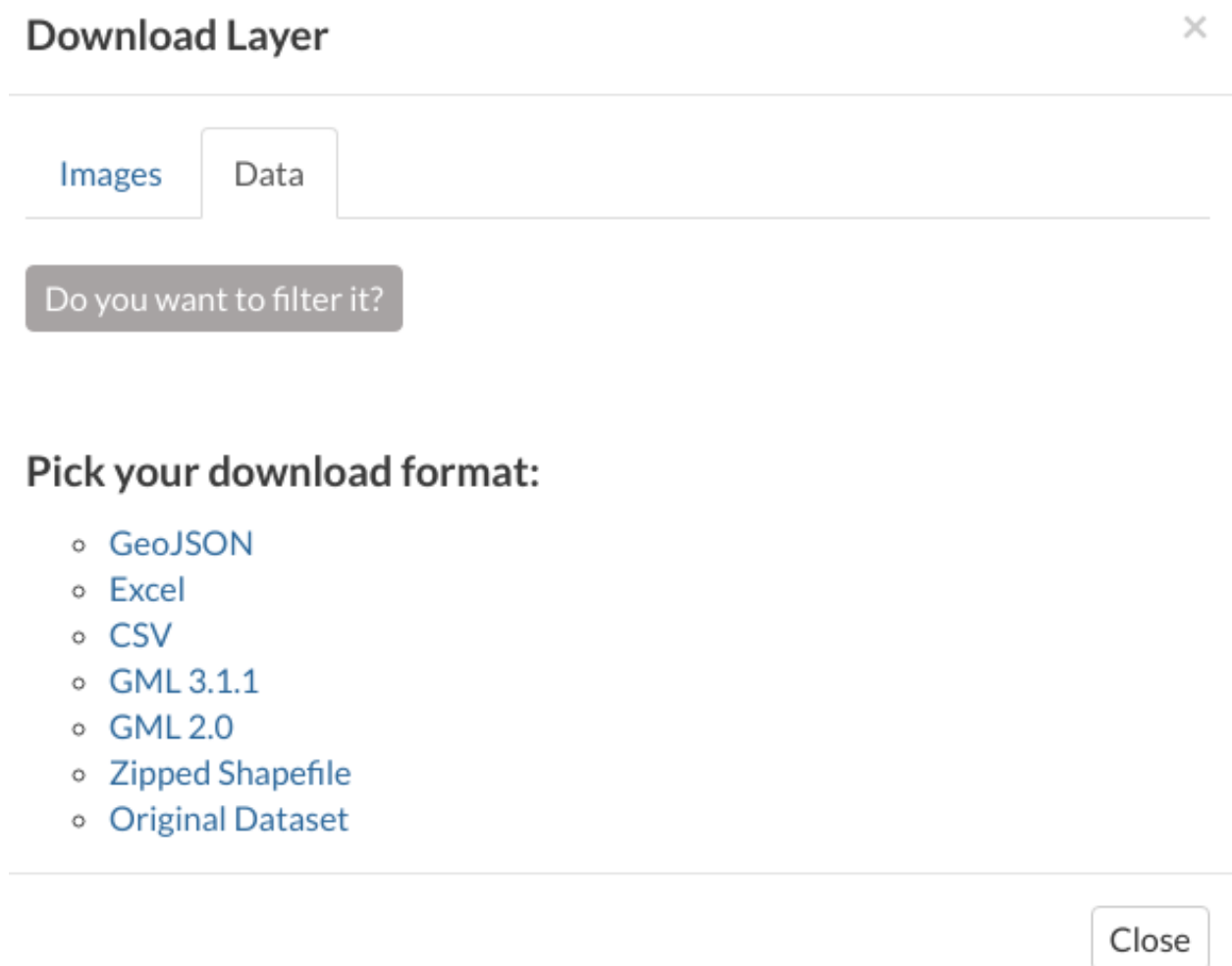
Fig. 104: *Downloading Layers as Images*Fig. 105: *Downloading the Layer Data*

Fig. 106: Downloading the Layer Data

Layer Editing

The *Editing Tools* button of the *Layer Page* (see *Layer Information*) opens a panel like the one shown in the picture below.

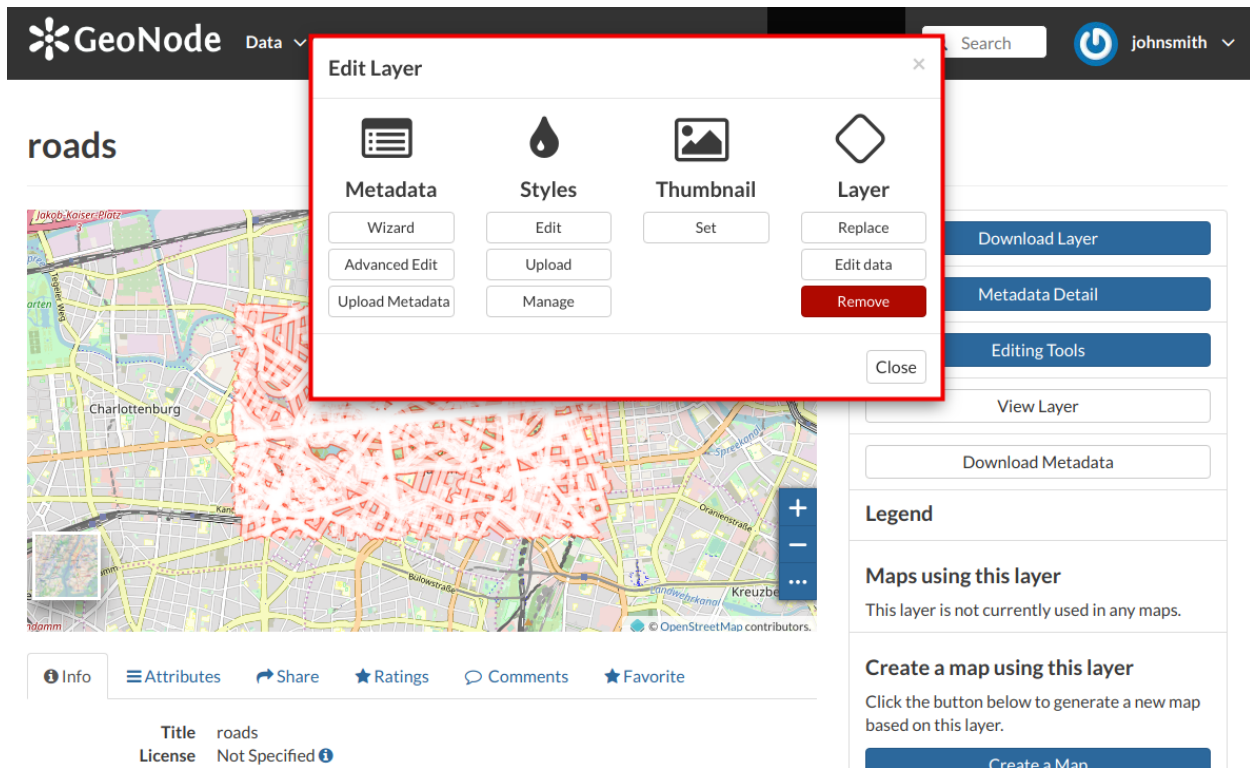


Fig. 107: The Layer Editing panel

In that panel you can see many options grouped by four categories:

1. *Metadata*
2. *Styles*
3. *Thumbnail*
4. *Layer*

In this section you will learn how to edit a *Layer*, how to replace and edit its data. See *Layers Metadata* to learn how to explore the layer *Metadata*, how to upload and edit them. The *Styles* will be covered in a dedicated section, see *Layer Styling*.

Setting the Layer Thumbnail

The Thumbnail of the layer that will be displayed on the *Layers* list page can be changed by dragging and zooming on the layer preview to select which portion will be displayed, then by clicking on the *Set* button of the *Layer Editing* panel.

A message will confirm the thumbnail has been correctly changed.

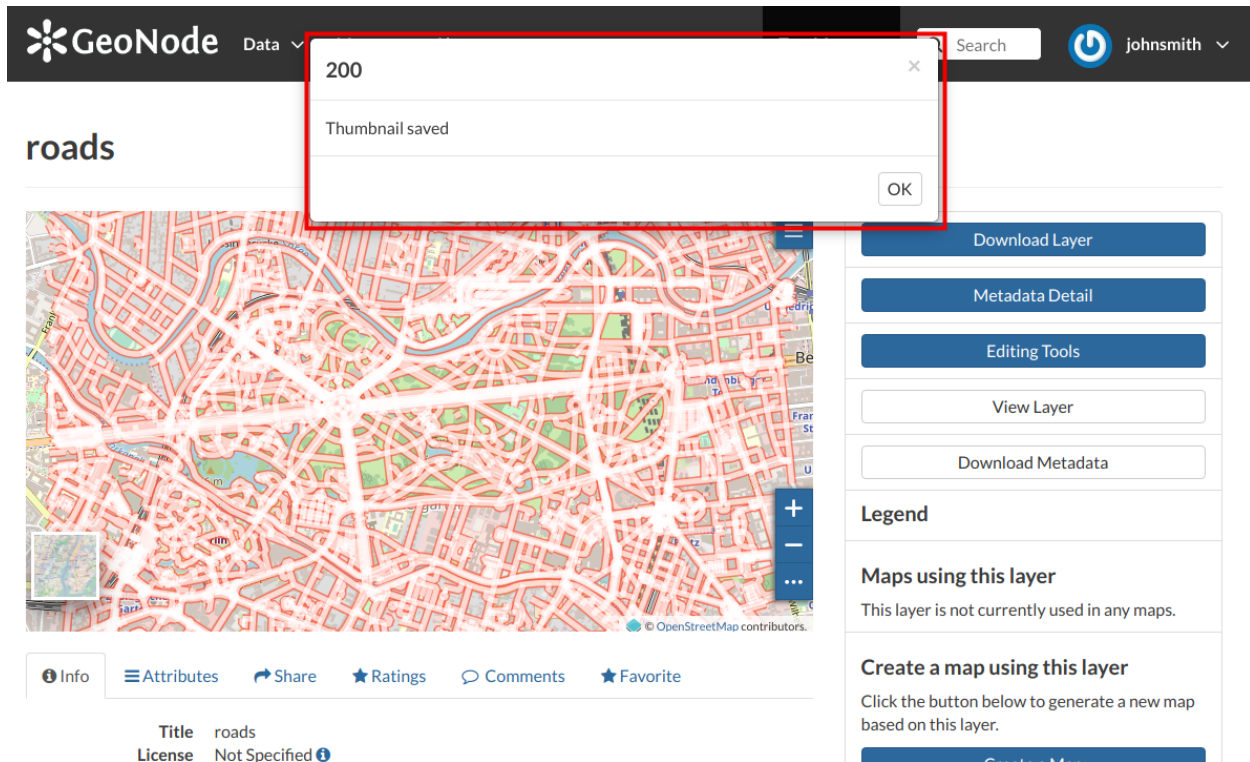


Fig. 108: The Layer Editing panel

It is also possible to manually upload a thumbnail by using the *Upload* button of the *Layer Editing* panel. Using the “Upload Thumbnail” page it is possible to enable the automatically generated thumbnail or upload an image to be used in place of it.

Replacing the Layer

From the *Layer Editing* panel click on *Replace* to change the layer source dataset. You will be driven to the *Replace Layer* page in which *Choose Files* button allows you to select files from your disk.

Once the *Charset* selected the upload process can be triggered by clicking on *Replace Layer*. If no errors occur you will see a message like the one in the picture below.

We have replaced the *roads* dataset with the *railways* one. You can see the differences in the *Layer Preview*.

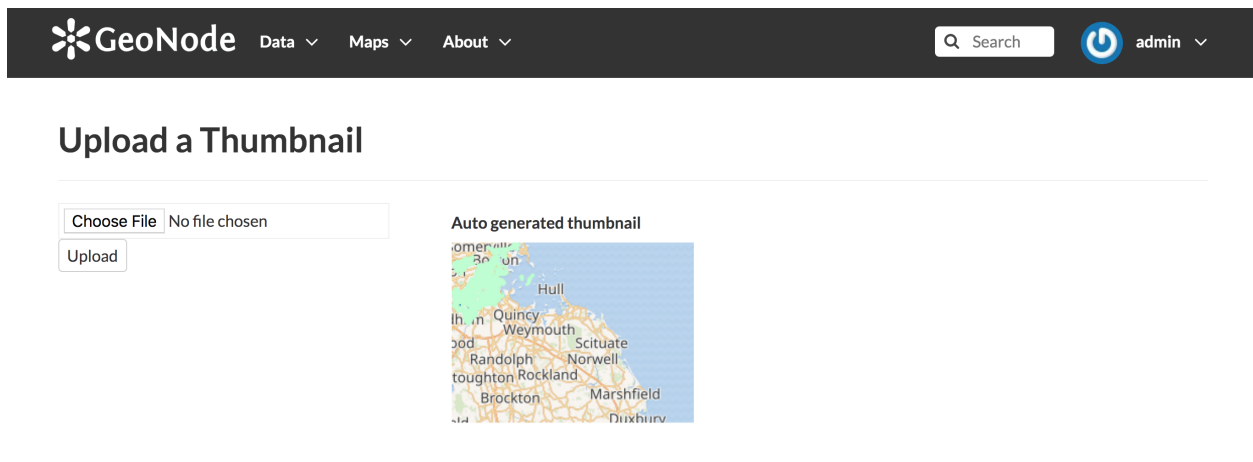


Fig. 109: *The Upload Thumbnail panel*

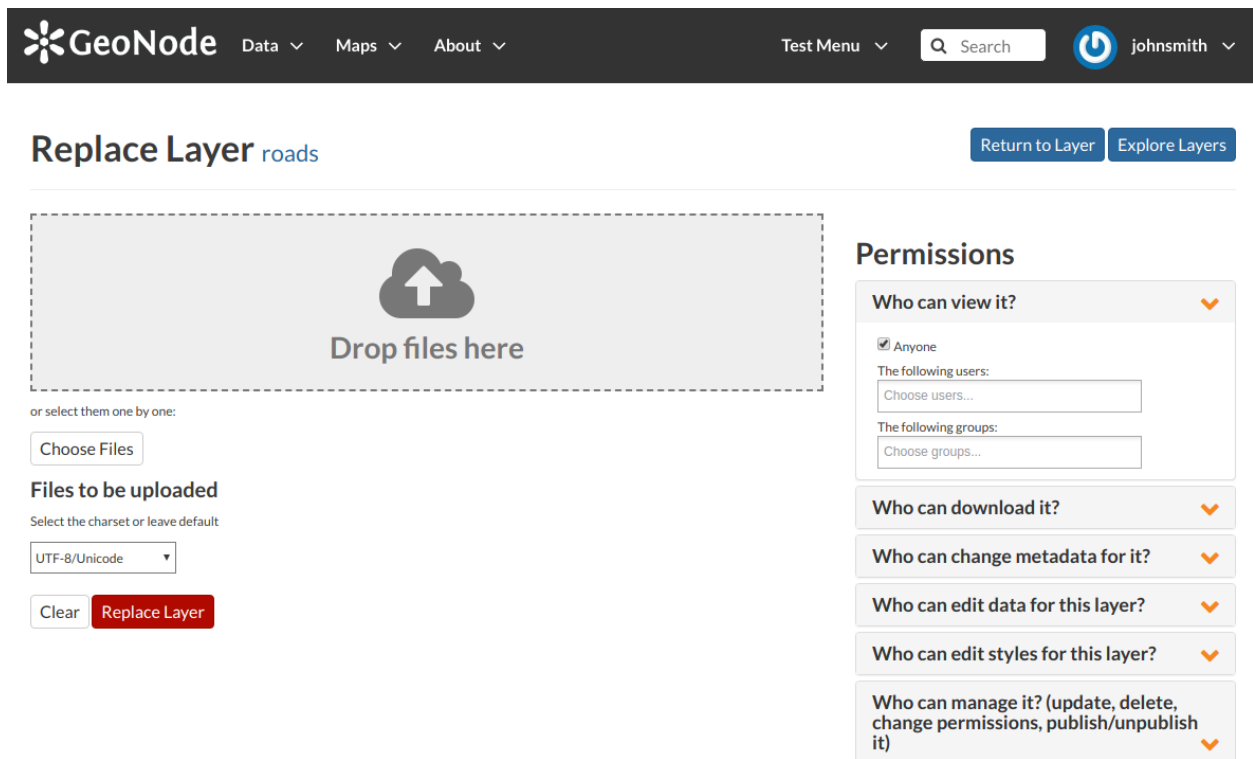





Fig. 110: *Replace a Layer*

 [Data](#) [Maps](#) [About](#) [Test Menu](#)  [johnsmith](#)

Replace Layer [roads](#)

[Return to Layer](#) [Explore Layers](#)



Drop files here

or select them one by one:

Choose Files

Files to be uploaded

railways

ESRI Shapefile

- [railways.shx Remove](#)
- [railways.shp Remove](#)
- [railways.prj Remove](#)
- [railways.dbf Remove](#)

Your layer was successfully updated

Layer Info

Edit Metadata

Upload Metadata

Upload SLD


Manage Styles

Select the charset or leave default

UTF-8/Unicode

Clear Replace Layer

Permissions


Who can view it? 


☒ Anyone
The following users:


Choose users...


The following groups:

Choose groups...

Who can download it? 

Who can change metadata for it? 

Who can edit data for this layer? 

Who can edit styles for this layer? 



Who can manage it? (update, delete, change permissions, publish/unpublish it) 

Fig. 111: *Replace Layer success*

1.10. GeoNode Users Guide

87


[Data](#)
[Maps](#)
[About](#)

[Test Menu](#)

[johnsmith](#)


[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

- ☐ A orange line style
- ☒ (default style) Red road

[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Fig. 112: Result of the Layer Replacement

Editing the Layer Data

The *Edit data* button of the *Layer Editing* panel opens the *Layer* within a *Map*.

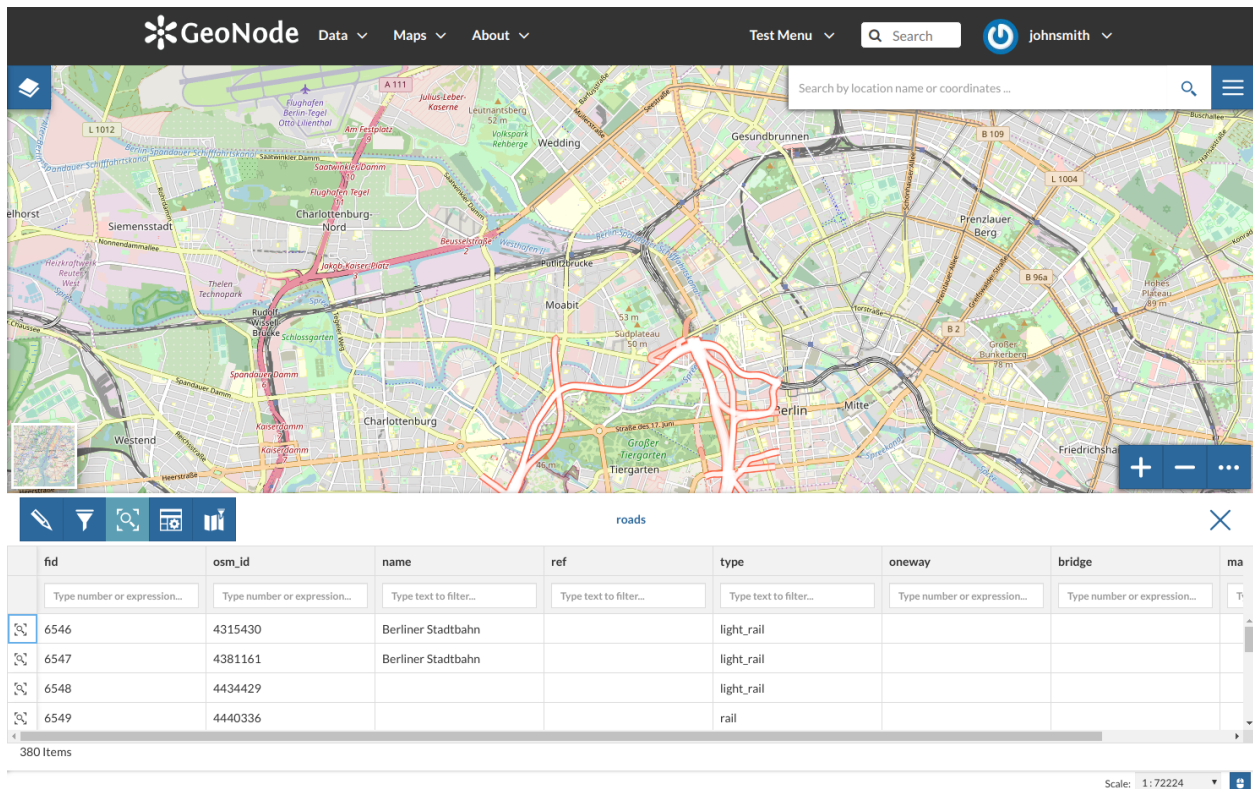



Fig. 113: *Editing the Layer Data*

The *Attribute Table* panel of the *Layer* will automatically appear at the bottom of the *Map*. In that panel all the features are listed. For each feature you can zoom to its extent by clicking on the corresponding *magnifying glass* icon  at the beginning of the row, you can also observe which values the feature assumes for each attribute.

Click the *Edit Mode*  button to start an editing session.

Now you can:

- *Add new Features*





Through the *Add New Feature* button  it is possible to set up a new feature for your layer. Fill the attributes fields and click  to save your change. Your new feature doesn't have a shape yet, click on  to draw its shape directly on the *Map* then click on  to save it.

Fig. 114: *Add a New Feature to the Layer*

Note: When your new feature has a multi-vertex shape you have to double-click the last vertex to finish the drawing.

- *Delete Features*


If you want to delete a feature you have to select it on the *Attribute Table* and click on .

Fig. 115: *Delete a Feature*

- *Change the Feature Shape*

You can edit the shape of an existing geometry dragging its vertices with the mouse. A blue circle lets you know what vertex you are moving.

Fig. 116: *Feature Shape Editing - Change the existing shape*

Features can have *multipart shapes*. You can add parts to the shape when editing it.

Fig. 117: *Feature Shape Editing - Add parts to the existing shape*

- *Change the Feature Attributes*

When you are in *Edit Mode* you can also edit the attributes values changing them directly in the corresponding text fields.

Once you have finished you can end the *Editing Session* by clicking on the  button.

By default the GeoNode map viewer is [MapStore](#) based, see the [MapStore Documentation](#) for further information.

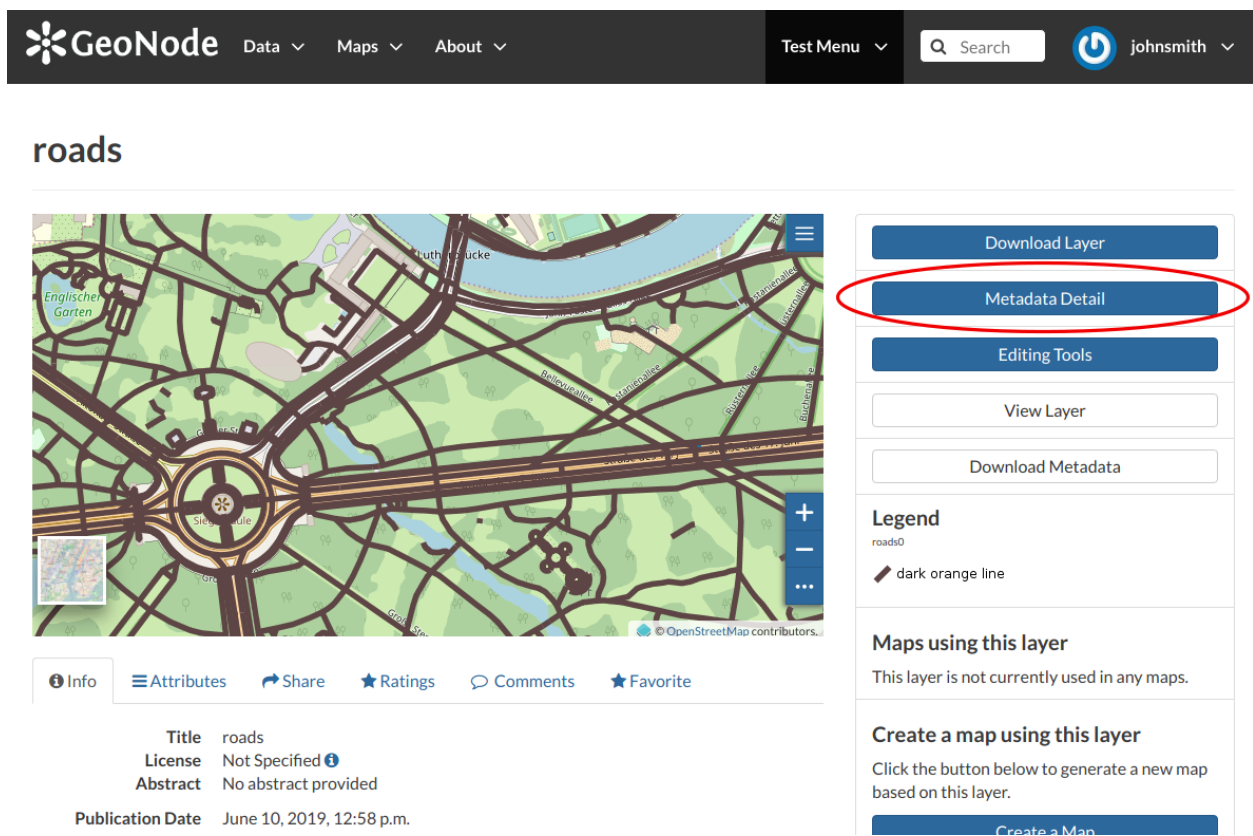
Layers Metadata


In GeoNode special importance is given to *Metadata* and their standard formats. You can explore the *Metadata* of a *Layer* by clicking the *Metadata Detail* button from the *Layer Page*.


The *Layer Metadata* page will be displayed.

In that page you can see the whole set of available metadata about the layer. Metadata are grouped in order to show the following types of information:

- *Identification* to uniquely identify the layer (Title, Abstract, Publication Date etc.);
- *Owner*, the user who owns the layer;
- *Information*, the Identification Image, the Spatial Extent, Projection System and so on;
- *Features*, Language, Supplemental and other Information;
- *Contact Points*, the available user to get in contact;
- *References*, various links to the resource information and data;
- *Metadata Author*, information about the author of the metadata.

Fig. 118: *Feature Attributes Editing*Fig. 119: *The Layer Metadata Detail button*


[Data](#)
[Maps](#)
[About](#)
[Test Menu](#)

 [johnsmith](#)

Metadata : roads

[Return to Layer](#)

Identification

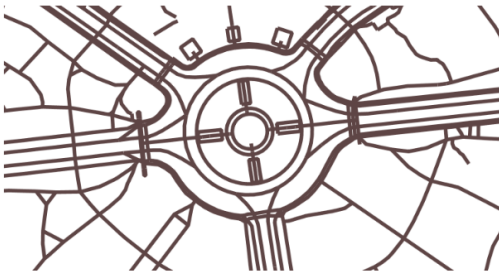
Title	roads
Abstract	No abstract provided
License	Not Specified
Publication Date	June 10, 2019, 12:58 p.m.
Type	Vector Data
Keywords	features ne_10m_roads
Category	Transportation
Regions	Global
Approved	Yes
Published	Yes
Featured	No

Owner

Name	johnsmith
email	johnsmith@mail.com
Position	None
Organization	None
Location	None
Voice	None
Fax	None

Information

Identification Image



Spatial Extent	---
Projection System	EPSG:4326
Extension x0	13.326002500000000
Extension x1	13.386977980000000
Extension y0	52.503006700000000
Extension y1	52.525999300000000

Features

Language	English
Supplemental Information	No information provided

Contact Points

Name	johnsmith
email	johnsmith@mail.com
Position	None
Organization	None
Location	None
Voice	None
Fax	None

References

Link Online	/layers/geonode:roads0
Metadata Page	/layers/geonode:roads0/metadata_detail

Thumbnail	roads.png
Remete Thumbnail	roads.png
Legend	roads.png
GeoJSON	roads.json
Excel	roads.excel
CSV	roads.csv
GML 3.1.1	roads.gml
GML 2.0	roads.gml
Zipped Shapefile	roads.zip
PNG	roads.png
PDF	roads.pdf
JPEG	roads.jpg
Original Dataset	roads.zip

OGC WFS: geonode Service	Geoservice OGC:WFS
OGC WMS: geonode Service	Geoservice OGC:WMS

Metadata Author

Name	johnsmith
email	johnsmith@mail.com
Position	None
Organization	None
Location	None
Voice	None
Fax	None

Downloading Metadata

The *Download Metadata* button of the *Layer Page* allows you to download the layer metadata in various formats.

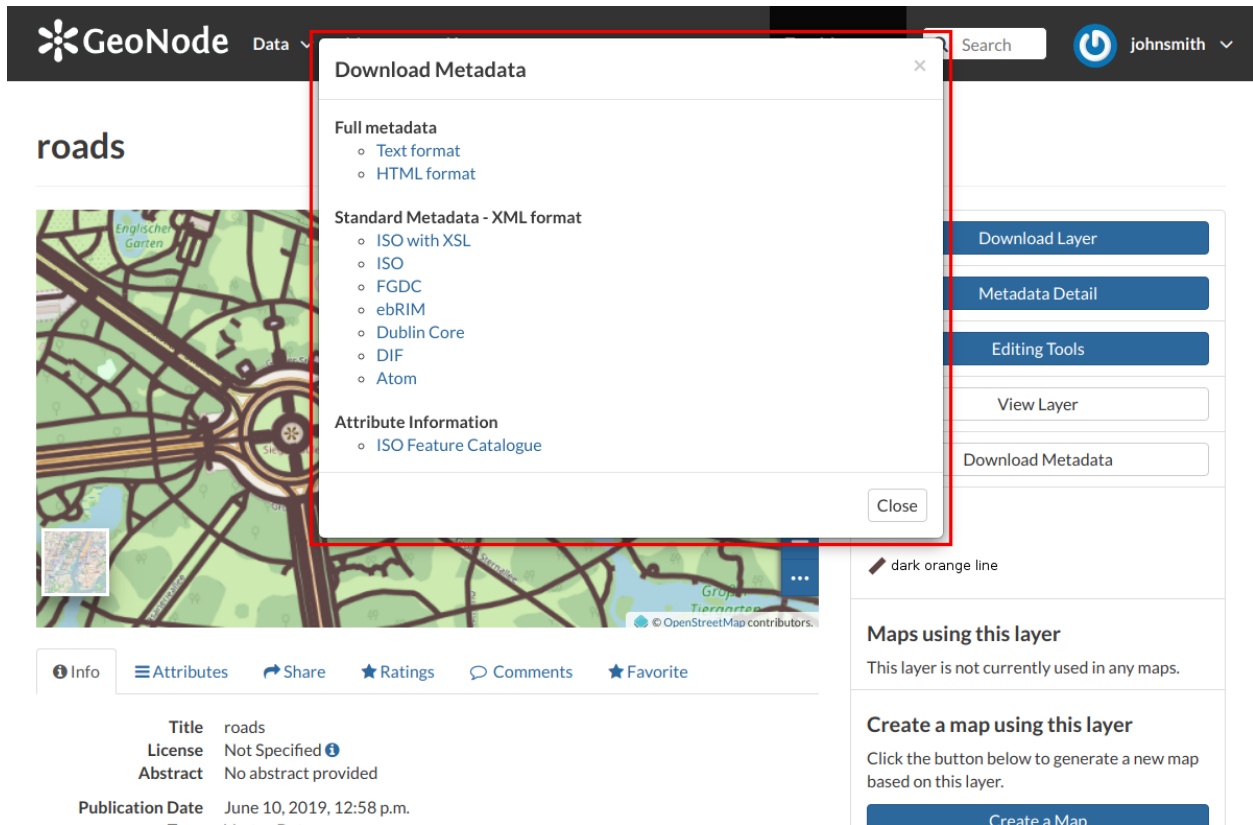


Fig. 121: *How to Download Metadata*

The available download formats are grouped in three categories:

- *Full metadata*
- *Standard Metadata - XML format*
- *Attribute Information*

Click on the format name that you prefer to start the download.

Metadata Wizard



Metadata contains all the information related to the layer. They provide essential information for its identification and its comprehension. Metadata also make the layer more easily retrievable through search by other users.

The *Metadata* of a layer can be changed through a *Wizard* which involves four steps, one for each type of metadata considered:

- *Basic Metadata*

The first two steps are mandatory (no layers will be published if the required information are not provided) whereas the last two are optional.

In the first step the system asks you to insert the following metadata:

Data ▾Maps ▾About ▾Test Menu ▾ Admin ▾

Metadata for roads

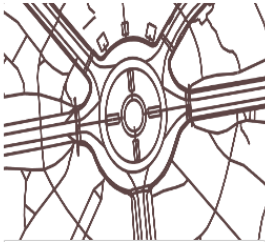
Completeness
✖ Check Schema mandatory fields
83 %

EditPreviewSettings

MandatoryMandatoryOptional

1Basic Metadata

Thumbnail



Edit

2Location and Licenses

Title ?

Abstract ?

3Optional Metadata

features × ne_10m_roads × roads ×

Date type ?

Publication

Category

Transportation

Group

John Smith Foundation Team

Date

2019-06-10 12:00

4Dataset Attributes

Return to LayerUpdateNext >>

Fig. 122: Basic Layer Metadata

- The *Thumbnail* of the layer (click *Edit* to change it);
- The *Title* of the layer, which should be clear and understandable;
- An *Abstract* on the layer;
- The *Creation/Publication/Revision Dates* which define the time period that is covered by the layer;
- The *Keywords*, which should be chosen within the available list. The contributor search for available keywords by clicking on the searching bar, or on the folder logo representing, or by entering the first letters of the desired word;
- The *Category* which the layer belongs to;
- The *Group* which the layer is linked to.

- *Location and Licenses*

The screenshot shows the 'Metadata for roads' form in GeoNode. The top navigation bar includes the GeoNode logo, 'Data', 'Maps', and 'About' menus, a 'Test Menu' dropdown, a search bar, and an 'Admin' link. A 'Completeness' badge indicates 'Metadata Schema mandatory fields completed' at 100%. The form has tabs for 'Edit', 'Preview', and 'Settings', and a 'Done!' button. A progress bar shows 'Mandatory' (green) and 'Optional' (blue) sections. The 'Location and Licenses' step (2) is active, showing fields for 'Language' (English), 'License' (Not Specified), 'Regions' (Global), 'Data quality statement' (Data provided by a community of users), 'Restrictions' (exclusive right to the publication, production), and 'Restrictions other' (other restrictions and legal prerequisites for accessing and using the resource or metadata). Navigation buttons at the bottom include 'Return to Layer', '<< Back', 'Update', and 'Next >>'.

Fig. 123: *Location and Licenses Metadata for Layers*

The following list shows what kinds of metadata you are required to enter (see also the picture below):

- The *Language* of the layer;
- The *License* of the dataset;
- The *Regions*, which informs on the spatial extent covered by the layer. Proposed extents cover the following scales: global, continental, regional, national;
- The *Data Quality statement* (general explanation of the data producer's knowledge about the lineage of a dataset);

- Potential *Restrictions* on layer sharing.

- *Optional Metadata*

The screenshot shows the 'Metadata for roads' form in GeoNode. The form is divided into four steps: 1. Basic Metadata, 2. Location and Licenses, 3. Optional Metadata, and 4. Dataset Attributes. A progress bar at the top indicates that 100% of the mandatory fields are completed. The form includes various input fields for metadata, such as Edition, Purpose, Supplemental information, temporal extent start/end, Maintenance frequency, Spatial representation type, Responsible Parties, Point of Contact, Owner and Permissions, Label, Description, and Display Order.

Fig. 124: *Optional Layer Metadata*


Complementary information are:

- The *Edition* to indicate the reference or the source of the layer;
- The *Purpose* of the layer and its objectives;
- Any *Supplemental information* that can provide a better understanding of the uploaded layer;
- The *Maintenance frequency* of the layer;
- The users who are *Responsible* for the layer, its *Owner*, and the *Author* of its metadata;
- The *Spatial representation type* used.

- *Dataset Attributes*

At this step you can enrich the dataset attributes with useful information like the following:

- The *Label* displayed
- A detailed *Description*
- The *Display Order*


[Data](#)
[Maps](#)
[About](#)
[Test Menu](#)

[Admin](#)

Metadata for roads

Completeness
 ✓ Metadata Schema mandatory fields completed
 100 %

[Edit](#)
[Preview](#)
[Settings](#)
[Done!](#)

MandatoryMandatoryOptional

1234

Basic MetadataLocation and LicensesOptional MetadataDataset Attributes

Attribute	Label	Description	Display Order
fid			1
the_geom			2
osm_id		The ID in the OSM full dataset	3
name		The name of the road	4
ref			5
type		The road type	6
oneway		The road can be traveled in one di	7
bridge		Is it a bridge?	8
maxspeed		The speed limits	9

[Return to Layer](#)
[<< Back](#)
[Update](#)

Fig. 125: Dataset Attributes Metadata for Layers

Use *next* >> or << *back* to navigate through those steps. Once you have finished click on *Update*.

Some metadata are mandatory, if you miss any of that metadata the *Completeness* bar shows you a red message like the one in the picture below.

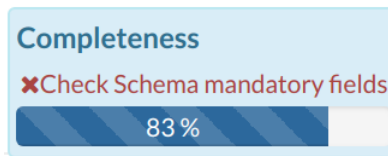


Fig. 126: *Completeness Progress Bar*

Metadata Advanced Editing

In the *Layer Editing* panel the *Advanced Edit* is also available.

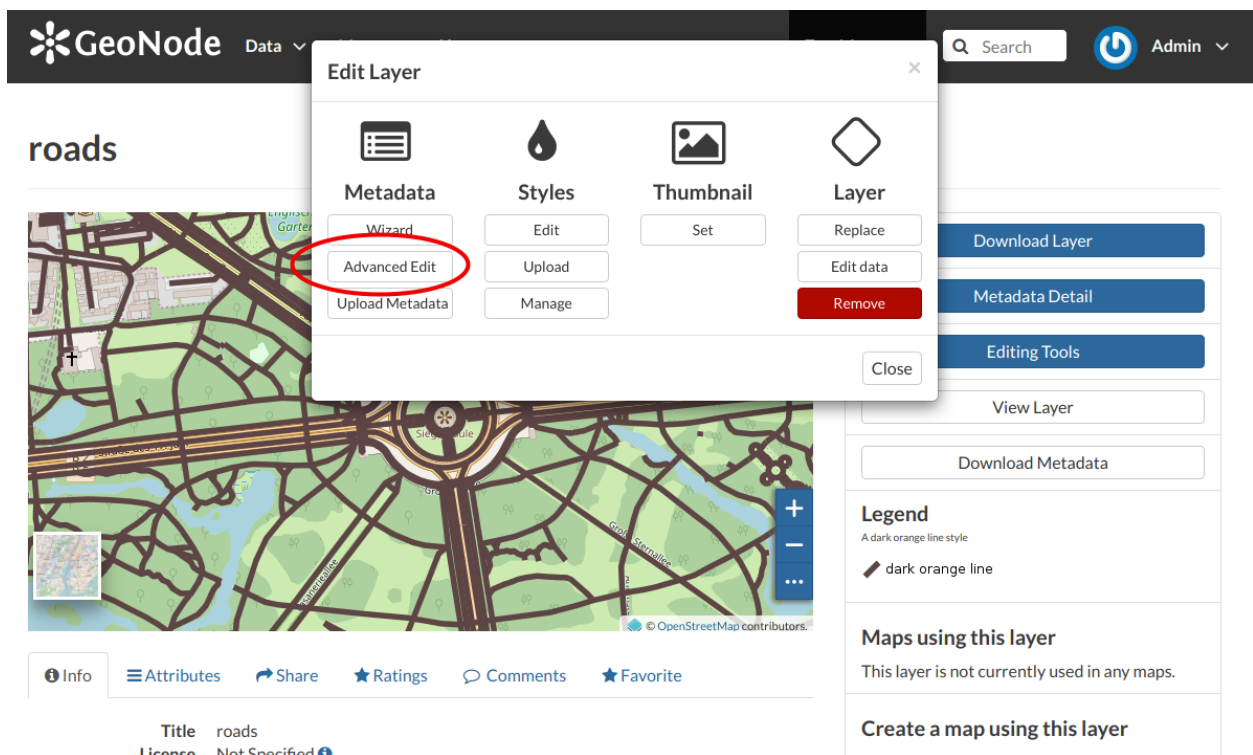


Fig. 127: *The Advanced Edit button*

Click on it to display the *Metadata Advanced Editing Page*. That page allows you to edit all the layer metadata described in the previous paragraph. Once you have finished to edit them click on *Update* to save your changes.

Uploading Metadata

Users may also upload a metadata XML document (in ISO, FGDC, or Dublin Core format) to fill in key GeoNode metadata elements automatically. The picture below shows you how the page looks like.

Fig. 128: *The Metadata Advanced Editing page*

Click on *Choose Files* to select the document from your disk, then click on *Upload files* to trigger the uploading process.

Layer Styling

Maps are helpful because they allow you gain a deeper understanding of your data by allowing you to visualize it in many different ways. So you can tell different stories depending on how the data is presented. For any given data or layer, you should explore different styling options and choose the best style for that.

In GeoNode each layer has a *Default Style* which is determined by the nature of the data you're mapping. When uploading a new layer (see *Layers Uploading*) a new default style will be associated to it.

Referring to the example above, dark orange lines are not very good to represent waterways so we would need to change this style. In the following paragraphs you will learn how to create a new style starting from given templates, how to edit a style, how to upload styles from file and how to manage them.

waterways

Download Layer

Metadata Detail

Editing Tools

View Layer

Download Metadata

Legend
A dark orange line style
dark orange line

Maps using this layer
This layer is not currently used in any maps.

Create a map using this layer
Click the button below to generate a new map based on this layer.
Create a Map

Styles
The following styles are associated with this layer. Choose a style to view it in the preview map.
● (default style) A dark orange line style

Refresh Attributes and Statistics of this layer
Click the button below to allow GeoNode refreshing the list of available layer attributes

Info **Attributes** **Share** **Ratings** **Comments** **Favorite**

Title waterways
License Not Specified ⓘ
Abstract No abstract provided
Publication Date June 11, 2019, 4:21 a.m.
Type Vector Data
Keywords features , waterways
Regions Global
Owner johnsmith
More info -

Layer WMS GetCapabilities document

Fig. 129: Default Style for Layers

Creating new Styles

In order to create a new style, open the *Layer Page* (see [Layer Information](#)) and click on *Editing Tools*. Then click the *Edit* button in the *Styles* section of the *Layer Editing* panel (see the picture below).

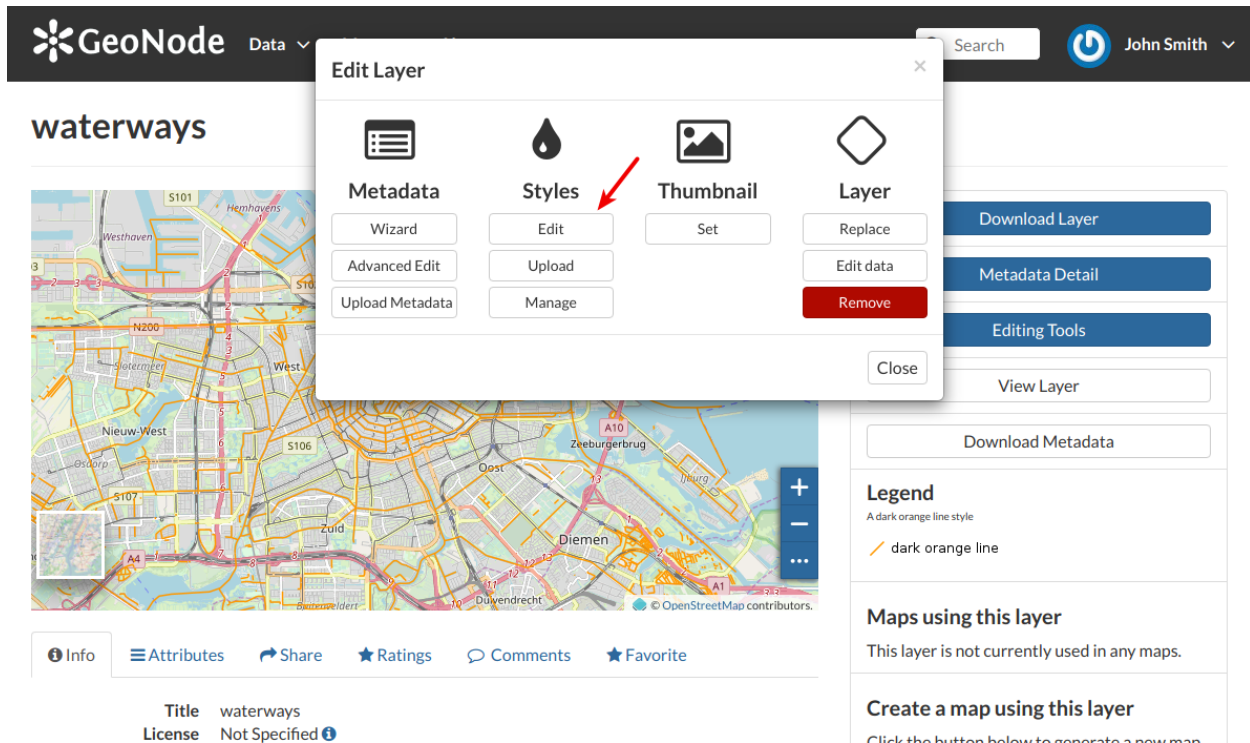




Fig. 130: *Edit Styles button*

The *Layer* will open in a new *Map*. The *Styles Panel* will show you all the available styles for the layer and some useful tools.

Now follow the steps below:

1. Click the  button. The *Style Templates Panel* will open.
2. Choose a *Style Template* from the list (both *CSS* and *SLD* styles are available).
3. Click the  button to add the *Style Template* to the styles list.
4. Insert a *Title* and an *Abstract* (optional), then click on *Save*.

The style you have created is now added to the *Styles List*.

You will also see this new style in the *Layer Page*.

Now you can switch the style by clicking on the corresponding checkbox.

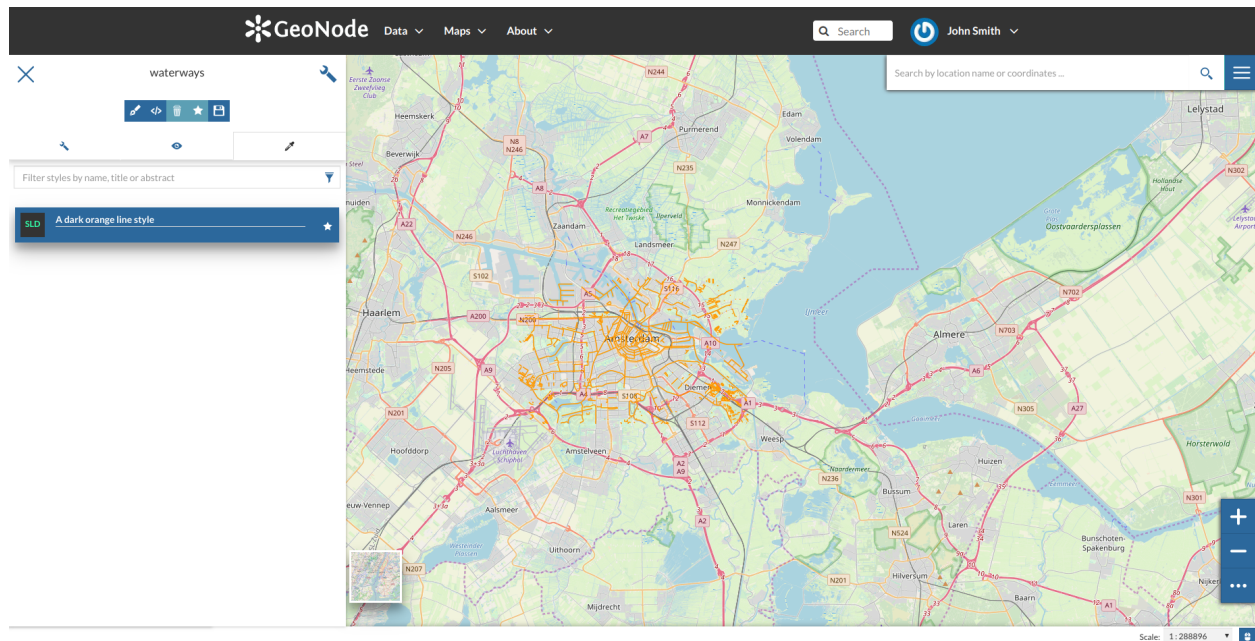





Fig. 131: The Styles Panel in the Map



It would be nice to change the style in order to decrease the opacity of the filling color as well as to reduce the lines width. The embedded [MapStore](#) makes available a powerful *Style Editor* to accomplish that tasks. In the next paragraph we will explain how.

Editing the Layer Style

The following steps show you how to edit styles:

1. From inside the map open the *TOC (Table Of Content)* by clicking the  button
2. Click on 
3. Open the *Style* tab 

Warning: Styles editing is allowed only to those users who have the needed permission. See [Changing the Layer Permissions](#) to read more)

4. Select the *Style* and click on 
5. Edit the style. The *Style Editor* helps you to write valid styles through the *Syntax Validator* which shows you a popup in case of errors (see the picture below).
6. Click on  to save your changes.

See the following gif to recap the whole process.

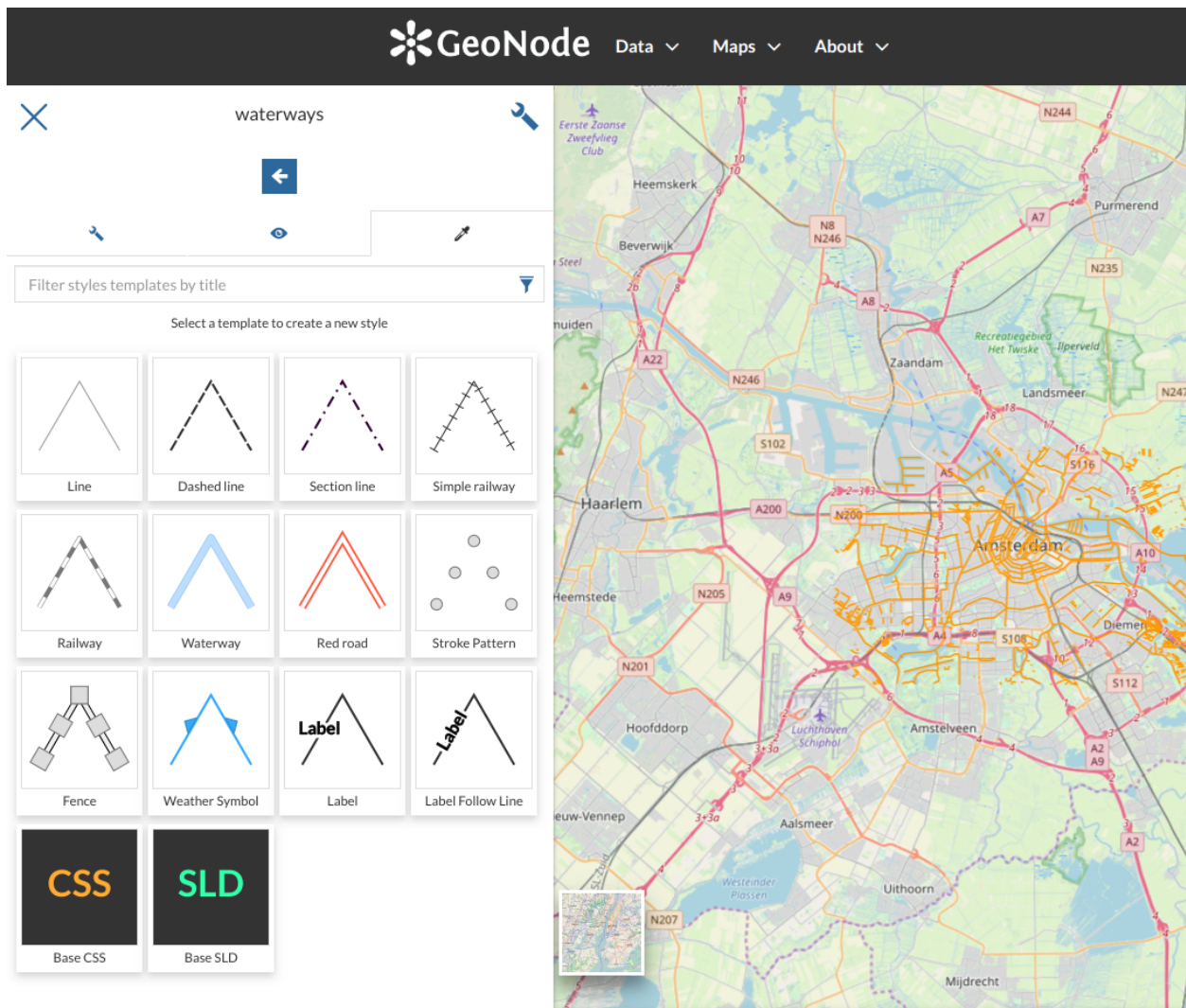


Fig. 132: Create new Styles

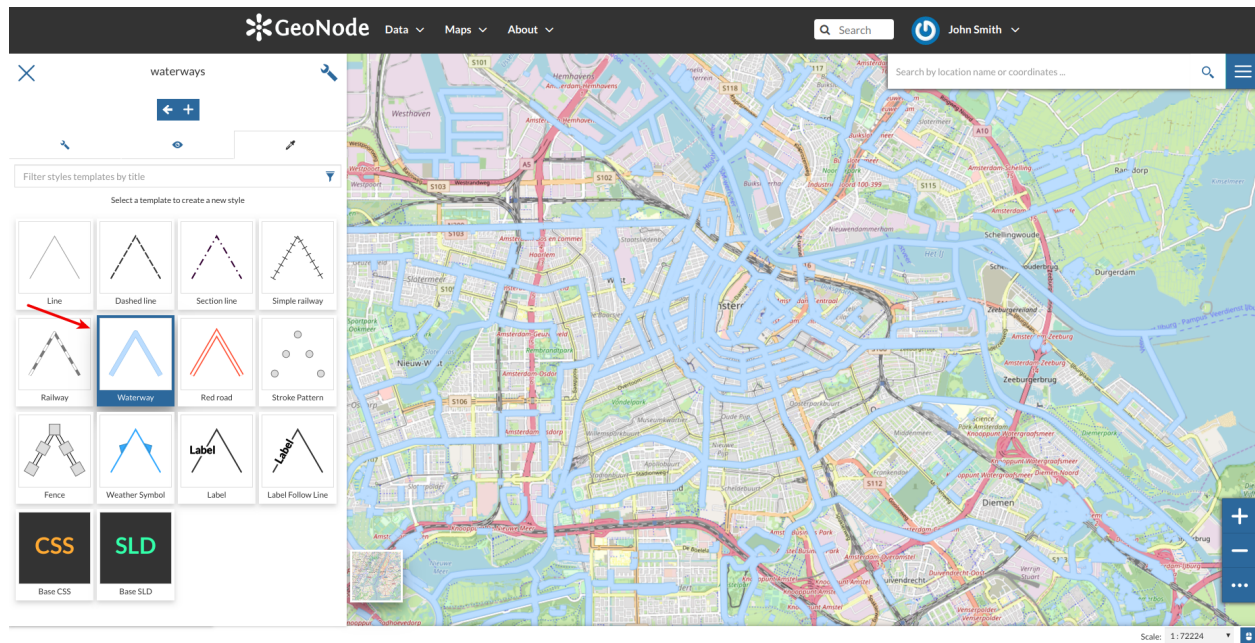


Fig. 133: *Style Templates*

Create new style

Title

Waterway

Abstract

A style for waterways

Save

Fig. 134: *Title and Abstract for new Styles*

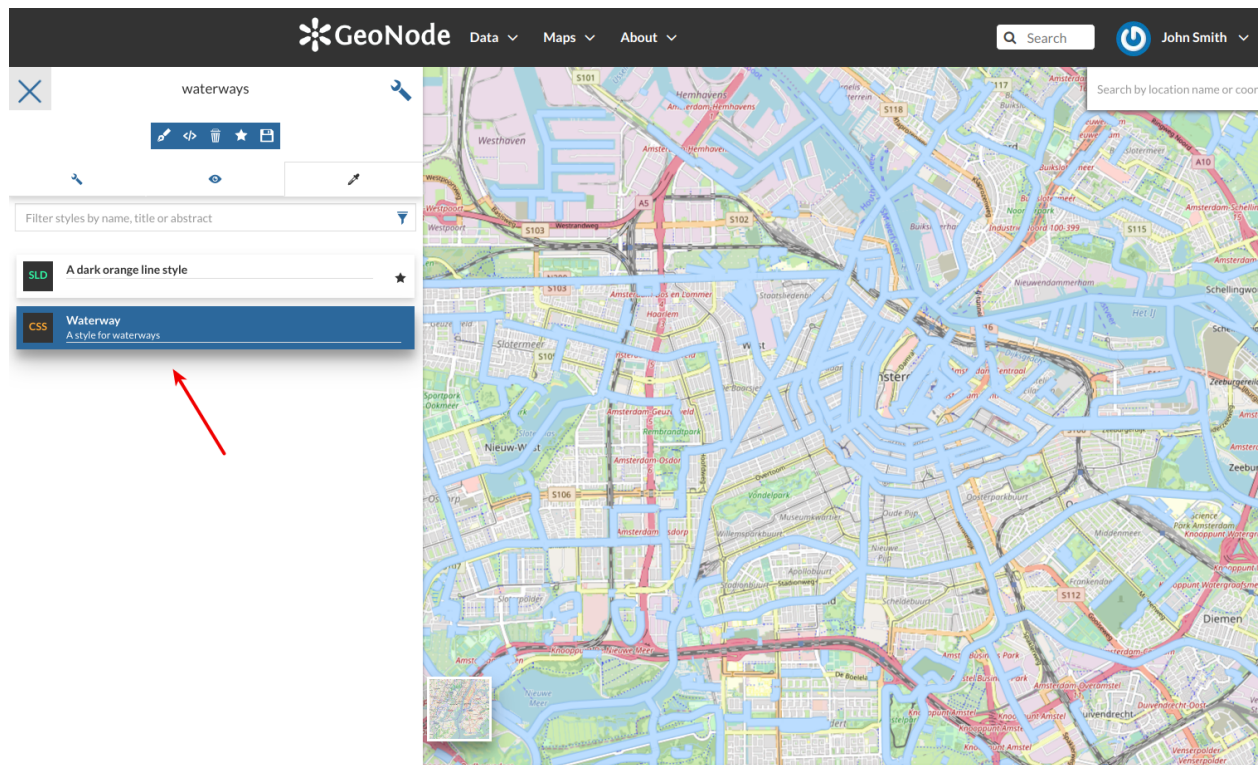



Fig. 135: New Styles into the list

You can also decide to make your new style the *Default Style* of that layer. Click on  to do that.

Click on  to delete the style.

Uploading Styles


In GeoNode it is also possible to upload an existing style from file.


Warning: Currently only styles in **SLD (Style Layer Descriptor 1.0, 1.1)** format can be uploaded in GeoNode.

From the *Layer Page* click on *Editing Tools* to open the *Editing Tools* panel and follow the steps below:

1. Click the *Upload* button of the *Styles* section
2. Click on *Choose Files* and select your style from your disk
3. Click on *Upload files*

Once the process has been finished the new *Style* will be visible in the *Layer Page*.


[Data](#)
[Maps](#)
[About](#)


[John Smith](#)



[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

A dark orange line style



Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

- ☒ (default style) A dark orange line style
- ☐ Waterway

[Refresh Attributes and Statistics of](#)

[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title waterways

License Not Specified

Abstract No abstract provided

Publication Date June 11, 2019, 4:21 a.m.

Type Vector Data

Keywords features , waterways

Regions Global

Owner johnsmith

[More info](#)

[Layer WMS GetCapabilities document](#)

Fig. 136: The Layer Page with the new Style

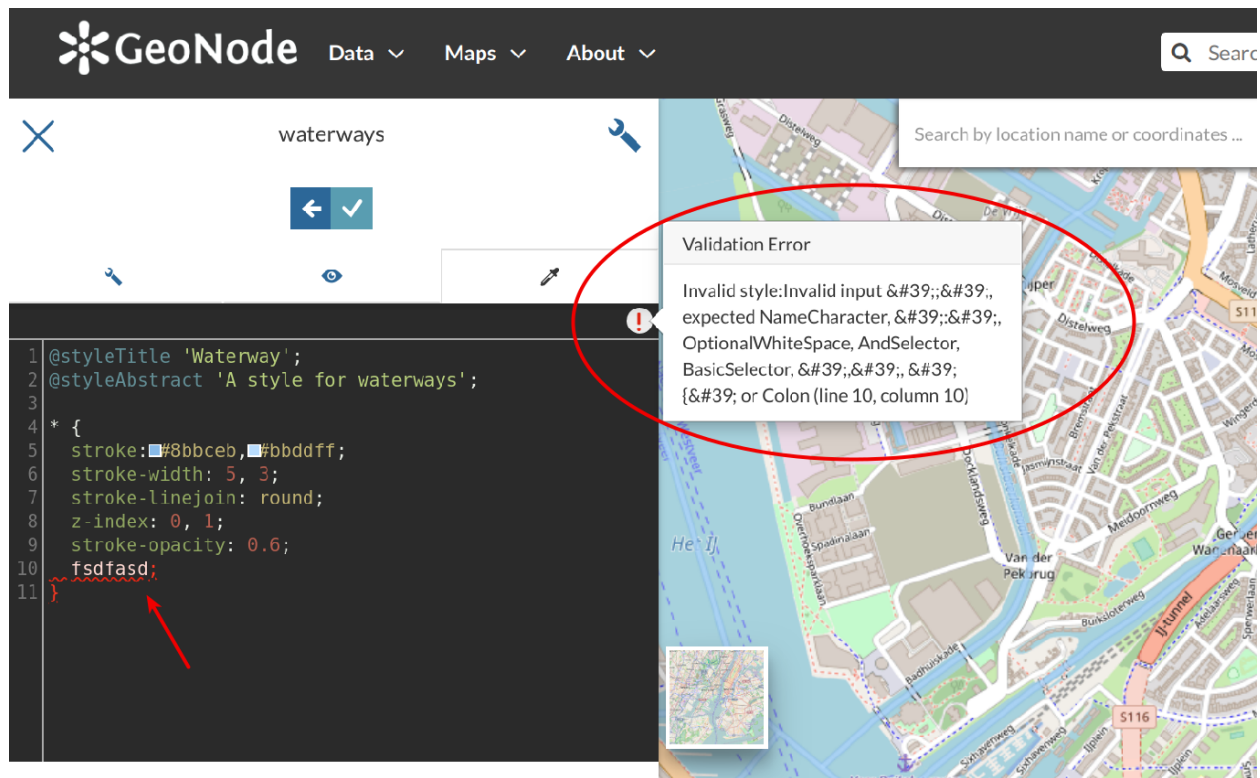


Fig. 137: The Style Editor Syntax Validation

Fig. 138: The Style Editor

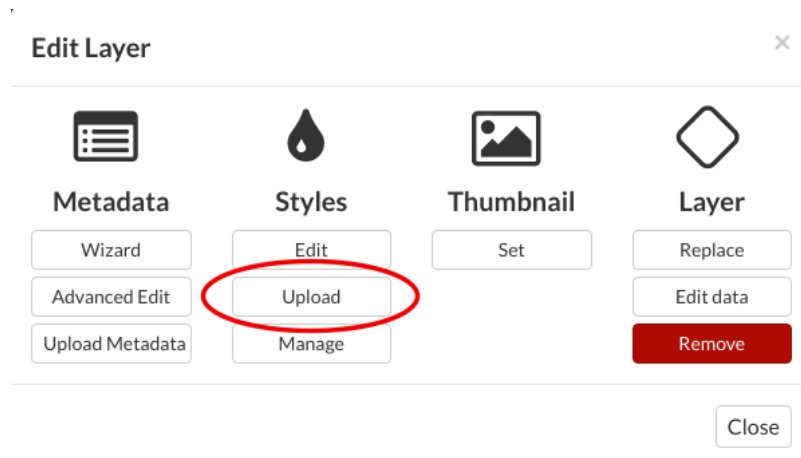


Fig. 139: Upload Styles button

Upload Layer Style (SLD - Style Layer Descriptor 1.0, 1.1) [Return to Layer](#) [Explore Layers](#)

Drop files here

or select them one by one:

[Choose Files](#)

Files to be uploaded

waterways

Style Layer Descriptor

• waterways.sld Remove

WARNING: This will most probably overwrite the current default style!

[Clear](#) [Upload files](#)

Permissions

Who can view it?

☒ Anyone

The following users:

✕ johnsmith

The following groups:

Choose groups...

Who can download it?

Who can change metadata for it?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Fig. 140: Upload Styles

Managing Styles

Given a layer, you can manage all its styles in the *Styles Management Page* accessible from the *Manage* button of the *Layer Editing* panel.

In that page you can:

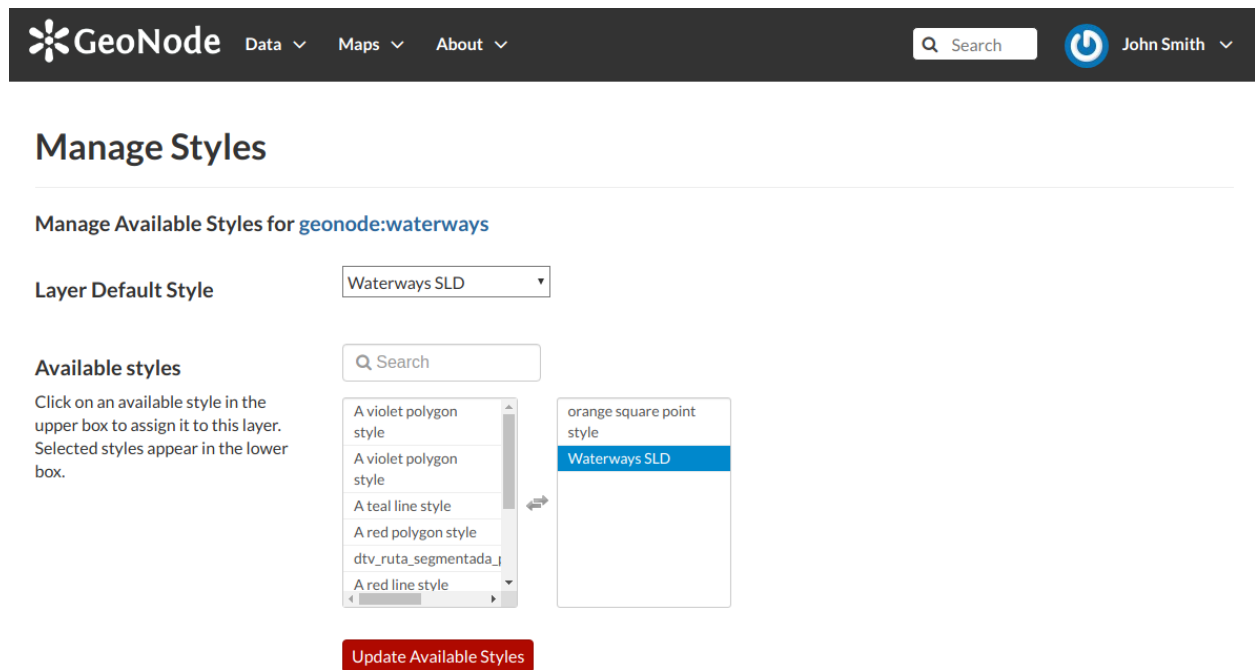
- See the *Layer Name*
- Add/remove styles to/from the *Available styles* list
- Choose the *Layer Default Style* from the *Available styles* list

Click on *Update Available Styles* to save your changes.

1.10.6 Managing Maps

Maps are sets of layers displayed together on an interactive web map. Maps can be composed in the map composer and saved as GeoNode resources. Maps can also be associated with metadata, ratings, and comments.

In this section, you will learn how to create a new map and share it.

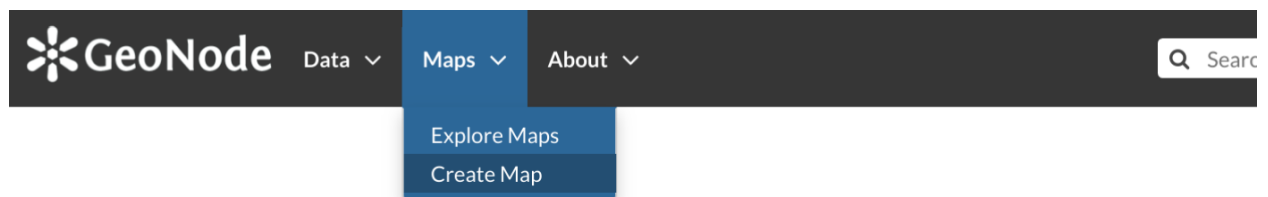
Fig. 141: *Managing Styles*

Creating Maps

In this section, we'll create a *Map* using some uploaded layers, combine them with some other layers from remote web services, and then share the resulting map for public viewing.


In order to create new maps you can use:

- the *Create Map* link of the *Maps* menu in the navigation bar

Fig. 142: *The Create Map link*

- the *Create Map* button in the *Layer Page* (it creates a map using a specific layer)
- the *Create New Map* button in the *Explore Maps* page

The new *Map* will open in a *Map Viewer* like the one in the picture below.

In the upper left corner the  button opens the *Table of Contents (TOC)* of the *Map*. It allows to manage all the layers associated with the map and to add new ones from the *Catalog*.

The *TOC* component makes possible to manage layers overlap on the map by shifting their relative positions in the list (drag and drop them up or down in the list).

The screenshot shows the GeoNode web interface. At the top is a dark navigation bar with the GeoNode logo, menu items (Data, Maps, About), a search bar, and a user profile for John Smith. Below the navigation bar, the page title is 'waterways'. The main content area features a map of Amsterdam with waterways highlighted in blue. To the right of the map is a sidebar with several buttons: 'Download Layer', 'Metadata Detail', 'Editing Tools', 'View Layer', and 'Download Metadata'. Below these buttons is a 'Legend' section, followed by 'Maps using this layer' (which states 'This layer is not currently used in any maps.'), and 'Create a map using this layer' (which says 'Click the button below to generate a new map based on this layer.'). The 'Create a Map' button is circled in red. Below the sidebar, there is a 'Styles' section.

waterways

Download Layer
Metadata Detail
Editing Tools
View Layer
Download Metadata

Legend
Waterways SLD
Rule Title

Maps using this layer
This layer is not currently used in any maps.

Create a map using this layer
Click the button below to generate a new map based on this layer.
Create a Map

Styles
The following styles are associated with this layer. Choose a style to view it in the preview map.

Info **Attributes** **Share** **Ratings** **Comments** **Favorite**

Title waterways
License Not Specified
Abstract No abstract provided
Publication Date June 11, 2019, 4:21 a.m.
Type Vector Data
Keywords features, waterways
Regions Global
Owner johnsmith
More info -

Laver WMS GetCapabilities document

Fig. 143: The Create Map button

The screenshot shows the 'Explore Maps' page on GeoNode. At the top is the same dark navigation bar as in Fig. 143. Below the navigation bar, the page title is 'Explore Maps'. To the right of the title is a button labeled 'Create a New Map', which is circled in red. Below the title, there is a section for 'Selected Maps' with a sub-section 'Add maps through the "checkboxes"'. Below this is a 'Set permissions' button. To the right of the 'Selected Maps' section, it says '0 Maps found' and 'No content created yet.' Below this is a 'Filters' section with a 'Clear' button. At the bottom right, there is a pagination bar showing 'page 1 of 1'.

Explore Maps **Create a New Map**

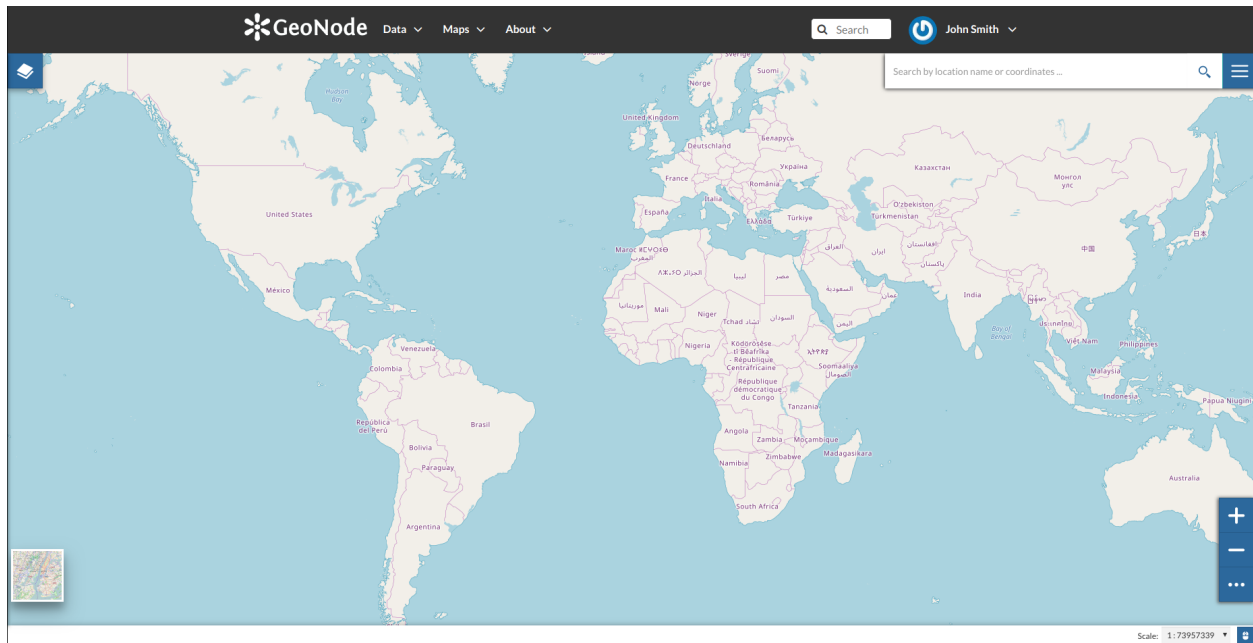
Selected Maps
Add maps through the "checkboxes".
Set permissions





0 Maps found
No content created yet.

Filters **Clear**
TEXT

< page 1 of 1 >

Fig. 144: The Create New Map button

Fig. 145: *The Map Viewer*

It also allows to hide/show layers ( and ), to zoom to layers extents () and to manage their properties ().


Once the map layers have been settled it is possible to save the *Map* by clicking on  and choosing *Save as*.

Fig. 146: *Creating new Maps*

If you followed the steps above, you have just created your first *Map*. Now you should see it in the *Explore Maps* page, see [Map Information](#) for further details.

We will take a closer look at the *Map Viewer* tools in the [Exploring Maps](#) section.


Map Information


As mentioned in the [Maps](#) section, in GeoNode you can see your maps and all the published maps through the *Explore Maps* link of the navigation bar.

Click on the title of the *Map* you are interested in to open its *Information* page, it should look like the following.

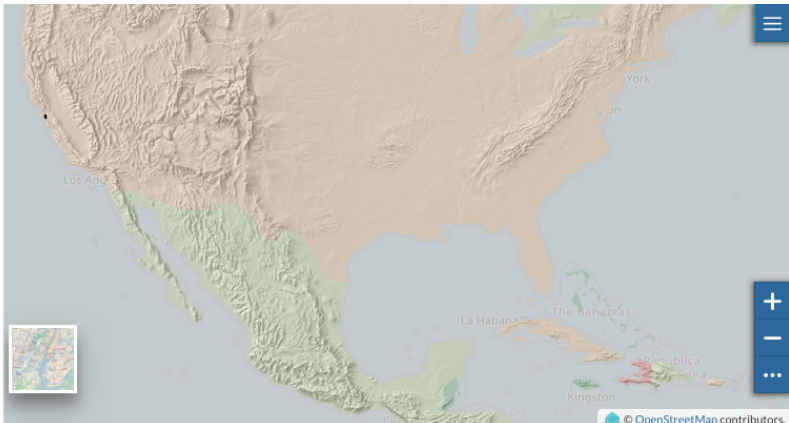
The *Map Page* is divided into three main sections:

1. the *Map Preview* section, under the title
2. the *Tabs* section, under the layer preview
3. the *Tools* section, on the right side of the page


[Data](#)
[Maps](#)
[About](#)

 John Smith

My New Map



[Download Map](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Map](#)

Map Layers

This map uses the following layers:

- epi:epi-environmental-performance-index-2010_biodiversity-and-habitat
- US_MSR
- San Francisco Landuse OSM

Permissions

Specify which users can view or modify this map

[Change Permissions of this Map](#)

Copy this map

Duplicate this map and modify it for your own purposes

[Create a New Map](#)

Map WMS


WMS layer group for local map layers:
(on local OWS)

Publish local map layers as WMS layer group

[Publish Map WMS](#)


About

Owner, Point of Contact, Metadata Author

 johnsmith
John Smith Foundation

[Info](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title My New Map

License Not Specified 

Publication Date June 12, 2019, 8:17 a.m.

Regions Global , Africa , Central Africa , West Africa , Pacific , Kiribati

Owner johnsmith

[More info](#) -


 Map layers WMS GetCapabilities document

Fig. 147: The Map Information page

Map Preview

The *Map Preview* shows the *Map* with very basic functionalities:

- the *Base Map Switcher* that allows you to change the base map;
- the *Zoom in/out* tool to enlarge and decrease the view;
- the *Zoom to max extent* tool for the zoom to fit the layers extents;
- the *Query Objects* tool to retrieve information about the map objects by clicking on the map;
- the *Print* tool to print the preview.

Fig. 148: *Map Preview*

See the [MapStore Documentation](#) to learn more.

Tabs Sections

The *Map Information* page shows you some tabs sections containing different information about the map:

- The tab *Info* is active by default. This tab section shows some metadata such as its Title, the License, the Publication Date etc. The metadata also indicates the map owner and which regions are involved. The Map Layers WMS GetCapabilities document link is also provided.

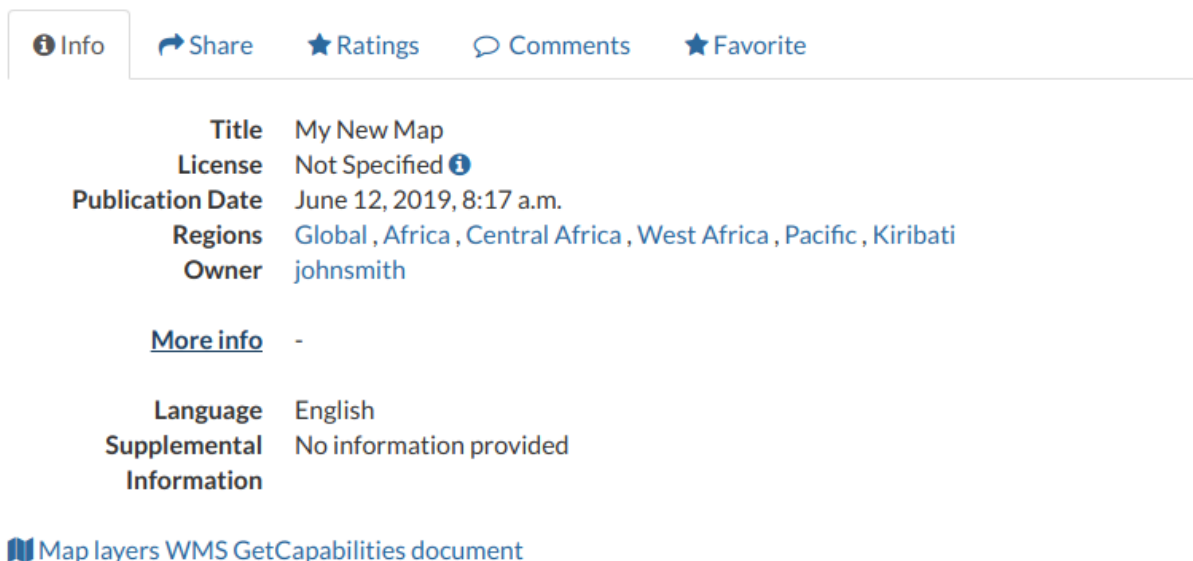


Fig. 149: *Maps Info tab*

- The *Share* tab provides the links for the map to share through social media or email.
- You can *Rate* the map through the *Rating system*.
- In the *Comments* tab section you can post your comment. Click on *Add Comment*, insert your comment and click *Submit Comment* to post it.

Your comment will be added next to the last already existing comment. If you want to remove it click on the red *Delete* button.

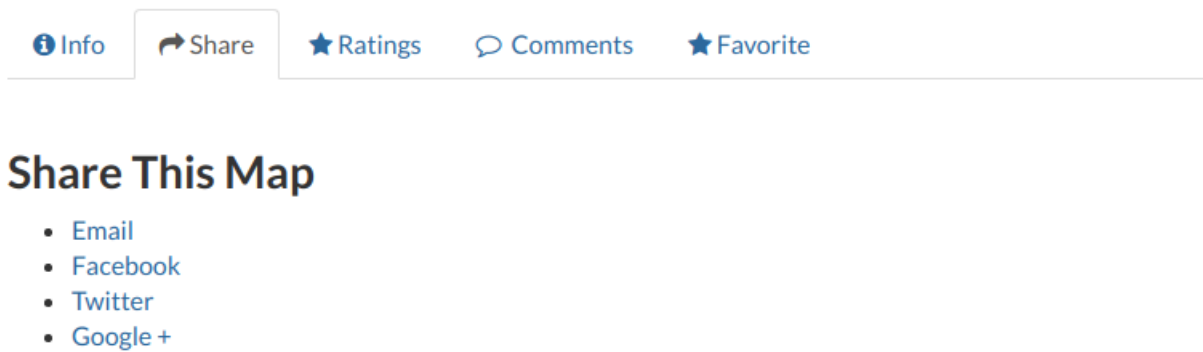


Fig. 150: *Map Sharing*

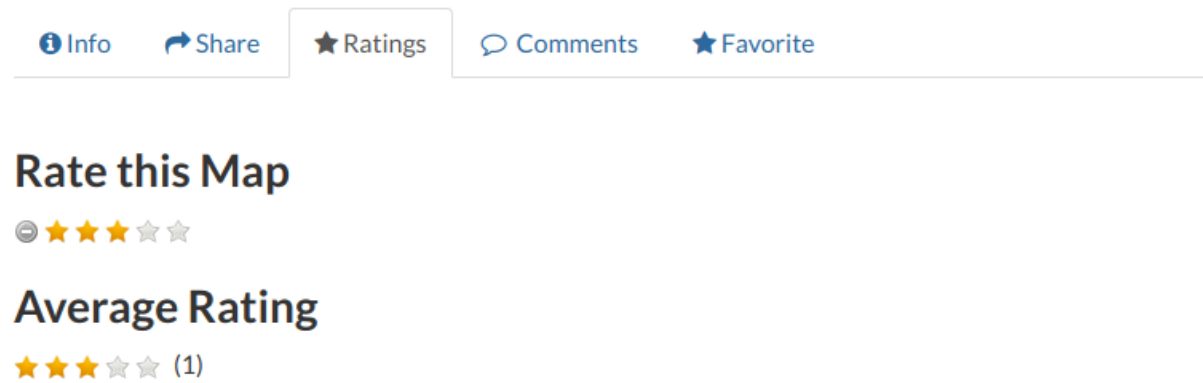


Fig. 151: *Map Rating*

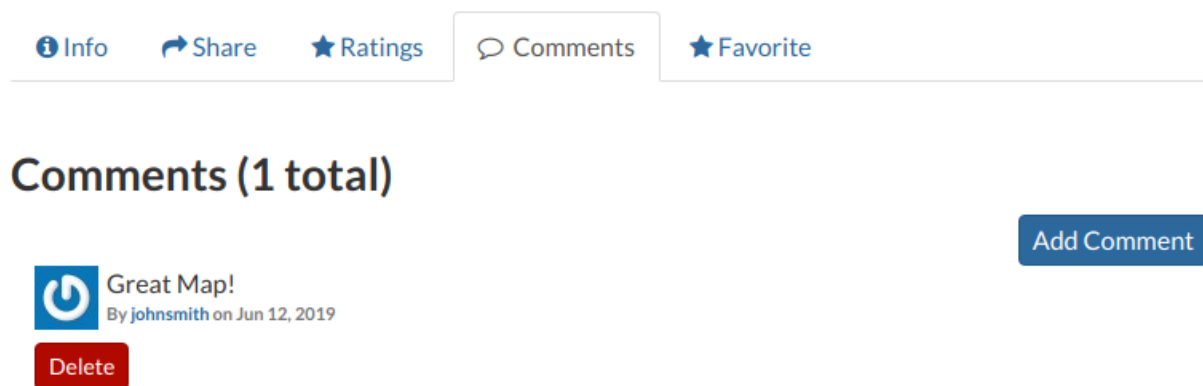


Fig. 152: *Map Comments*

- If you want this map in your *Favorites* (see [Updating the Profile](#)), open the *Favorite* tab and click on *Add to Favorites*.

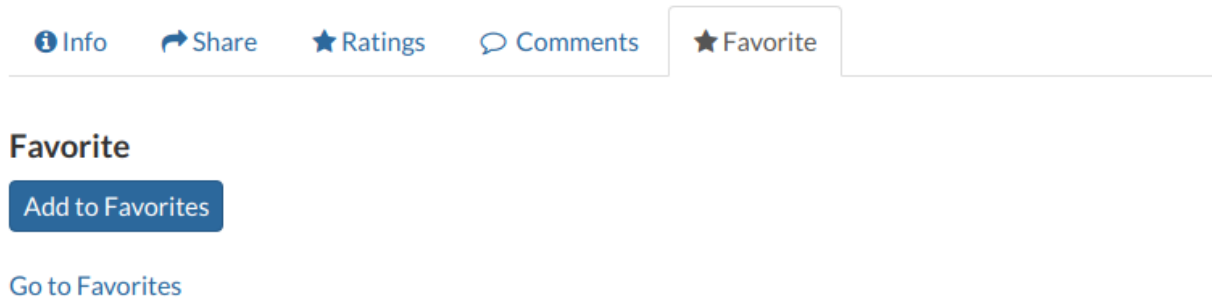


Fig. 153: *Your Favorite Maps*

Map Tools

In the right side of the *Map Information* page there are some tools that can help you to manage your maps. In this paragraph you will learn how to discover and retrieve information about maps.

The following is a list of actions you can take in order to accomplish this task:

- click the *Download Map* button, to download the map as image;
- click the *Metadata Detail* button to see the map metadata, see [Maps Metadata](#);
- click the *Editing Tools* button to access to many editing tools. Those functionalities will be explained in the [Exploring Maps](#) section;
- click the *View Map* button to open the map, see the [Exploring Maps](#) section for more details;
- see the *Map Layers* section to know which layers are used by the map (you can open the *Layer Page* by clicking on its name, available only for local layers);
- click the *Create a Map* button of the *Copy this map* section to duplicate the map;
- click the *Publish Map WMS* of the *Map WMS* section to publish local map layers as WMS layer group;
- see the *About* section to know the map *Owner*, the *Contact* user and the *Metadata Author*.

Maps Metadata

Maps Metadata can be explored by clicking the *Metadata Detail* button from the *Map Information* page.

The *Map Metadata* page will open.

Lots of information are displayed in this page. Those information are grouped as follow:

- *Identification* to uniquely identify the map (Title, License, Publication Date and Regions. There are also some flags which tell you the state of the map, in particular if it is Approved and/or Published);
- the map *Owner*;
- *Information*, the Identification Image, the Spatial Extent, the Projection System and the Extent;
- *Features*, Language, Supplemental and other Information;
- *Contact Points*, the available user to get in contact;

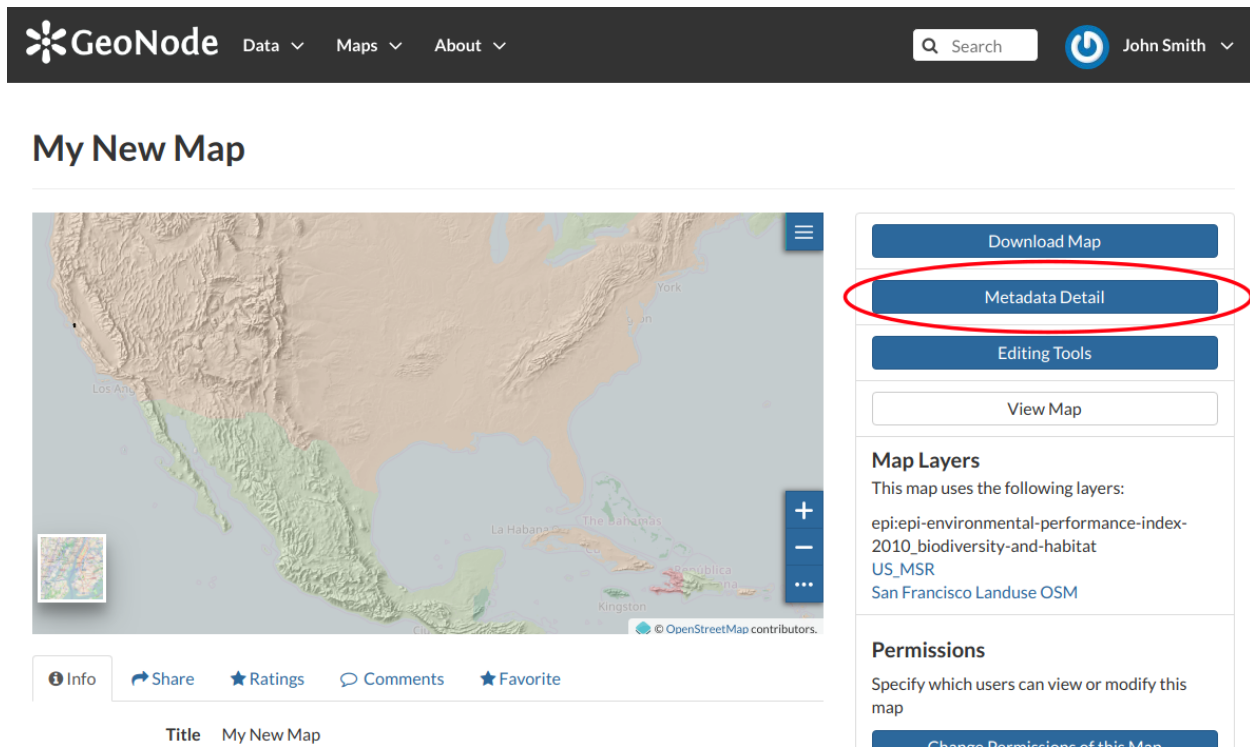


Fig. 154: The Map Metadata Detail button

- *References*, links to the map and its metadata;
- *Metadata Author*, information about the author of the metadata.

Metadata Wizard


Metadata provide essential information for the identification and the comprehension of the map. They also make the map more easily retrievable through the search tools.


Those *Metadata* can be filled out through a three-steps *Wizard* in which you have to provide all mandatory information to complete the process. Those three steps are described below.


- *Basic Metadata*

In the first step the system asks you to insert the following metadata (required fields are highlighted with red outlines):

- The *Thumbnail* of the map (click *Edit* to change it);
- The *Title* of the map, which should be clear and understandable;
- An *Abstract*;
- The *Creation/Publication/Revision Dates* which define the time period that is covered by the map;
- The *Keywords*, which should be chosen within the available list;
- The *Category* which the map belongs to;
- The *Group* which the map is linked to.


Data ▾
Maps ▾
About ▾




John Smith ▾

Metadata : My New Map Return to Map

Identification


Title	My New Map
License	Not Specified ⓘ
Publication Date	June 12, 2019, 8:17 a.m.
Regions	Global, Africa, Central Africa, West Africa, Pacific, Kiribati
Approved	Yes
Published	Yes
Featured	No

Owner

Name	John Smith (johnsmith)
email	john.smith@mail.com
Position	CEO and Founder
Organization	John Smith Foundation
Location	John Smith Avenue 12345 John Smith City John Smith District ZAF
Voice	123456789
Fax	987654321

Information

Identification Image



Spatial Extent	---
Projection System	EPSG:3857
Extension x0	-20037397.023299999535084
Extension x1	666726.142827000003308
Extension y0	-5787726.067250000312924
Extension y1	20037397.023299999535084

Features

Language	English
Supplemental Information	No information provided

Contact Points


Name	John Smith (johnsmith)
email	john.smith@mail.com
Position	CEO and Founder
Organization	John Smith Foundation
Location	John Smith Avenue 12345 John Smith City John Smith District ZAF
Voice	123456789
Fax	987654321


References

Link Online	/maps/47
Metadata Page	/maps/47/metadata_detail


Metadata Author


Name	John Smith (johnsmith)
email	john.smith@mail.com
Position	CEO and Founder
Organization	John Smith Foundation
Location	John Smith Avenue 12345 John Smith City John Smith District ZAF
Voice	123456789
Fax	987654321


Data ▾
Maps ▾
About ▾


John Smith ▾

Metadata for My New Map


Edit


Settings

Mandatory


Mandatory

Optional

1

Basic Metadata

Thumbnail



Edit

2

Location and Licenses

Title

My New Map

Abstract


3

Optional Metadata

Date type

Publication ▾

Date

2019-06-12 08: 

Category

Group

Update

Next >>

Fig. 156: *Basic Map Metadata*

Click *Next >>* to go to the next step.

- *Location and Licenses*

GeoNode Data Maps About Search John Smith

Metadata for My New Map

Completeness: Check Schema mandatory fields 83%

Edit Settings

1 Basic Metadata **2 Location and Licenses** 3 Optional Metadata

Language
English

License
Open Data Commons Open Database License

Regions
Global Pacific Americas
United States of America [North America]

Data quality statement
Field declared Mandatory by the Metadata Schema

Restrictions
Field declared Mandatory by the Metadata Schema

Restrictions other

<< Back Update Next >>

Fig. 157: *Location and Licenses Metadata for Maps*

The following list shows what kinds of metadata you are required to enter (see also the picture below):


- The *Language* of the layer;
- The *License* of the dataset;
- The *Regions* covered by the layers extent. Proposed extents cover the following scales: global, continental, regional, national;
- The *Data Quality statement* (general explanation of the data producer's knowledge about the lineage of a dataset);
- Potential *Restrictions* on layer sharing.

No further mandatory metadata are required in the next step so, once the required fields have been filled out, a green *Done* button will be visible in the screen. Click *Next >>* to go to the next step or << *Back* to go back to the previous step.

- *Optional Metadata*

Complementary information are:

- The *Edition* of the map;

 [Data](#) [Maps](#) [About](#) [John Smith](#)

Metadata for My New Map

Completeness
✔ Metadata Schema mandatory fields completed
100%

[Edit](#) [Settings](#) [Done!](#)

Mandatory

Mandatory

Optional

1

2

3

Basic Metadata

Other, Optional, Metadata

Edition

Purpose

Supplemental information

No information provided

Location and Licenses

temporal extent start

temporal extent end

Maintenance frequency

Spatial representation type

Optional Metadata

Responsible Parties

Point of Contact

✖ johnsmith

Owner and Permissions

Owner

✖ johnsmith

Metadata Author

✖ johnsmith

[<< Back](#) [Update](#)

Fig. 158: *Optional Map Metadata*

- The *Purpose* of the map and its objectives;
- Any *Supplemental information* that can provide a better understanding of the map;
- The *Maintenance frequency* of the map;
- The *Spatial representation type*, the method used to represent geographic information in the dataset;
- The users who are *Responsible* for the layer, its *Owner*, and the *Author* of its metadata;

If you miss some mandatory metadata the *Completeness* bar shows you a red message like the one in the picture below.

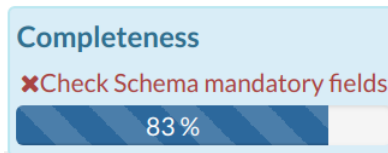


Fig. 159: *Completeness Progress Bar*

Metadata Advanced Editing

The *Advanced Edit* editing tool allows to change the map metadata. You can find this button into the map *Editing Tools*.

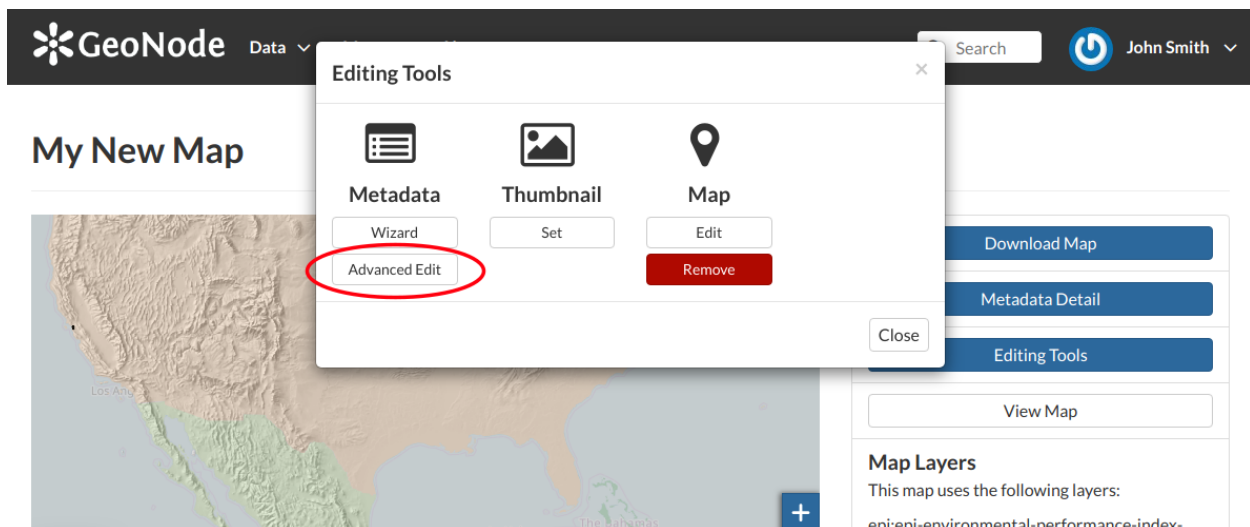


Fig. 160: *The Advanced Edit button*

Click on it to display the *Metadata Advanced Editing Page*. That page allows you to edit all the layer metadata described in the previous paragraph. Once you have finished to edit them click on *Update* to save your changes.

Changing the Map Permissions

In the *Map Information* section of this guide we said that you can see your maps and all the published maps. In GeoNode the permissions management system is indeed more complex. Administrators can choose who can do what for each map. Users can manage only the maps they own or the maps which they are authorize to manage.

By default only owners can edit and manage maps, anyone can view and download them.

In order to modify the *Map Permissions* settings you have to click the *Change the Layer Permissions* button in the *Map Page*.

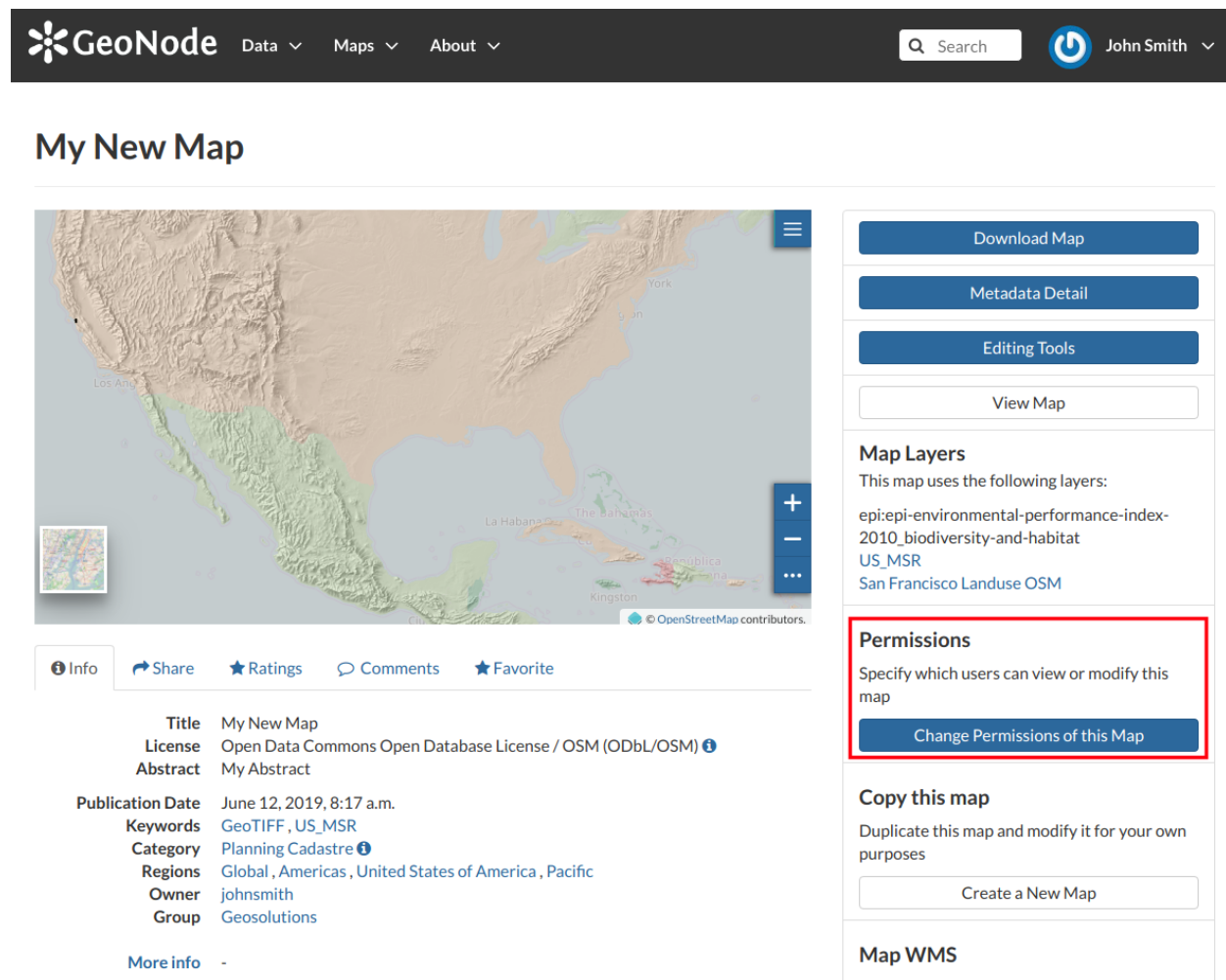


Fig. 161: *Change Map Permissions*

Through the *Permissions Settings Panel* you can add or remove permissions for users and groups. The picture below shows an example.

You can set the following types of permissions:

- *View* allows to view the map;
- *Download* allows to download the map;
- *Change Metadata* allows to change the map metadata;
- *Manage* allows to update, delete, change permissions, publish and unpublish the map.

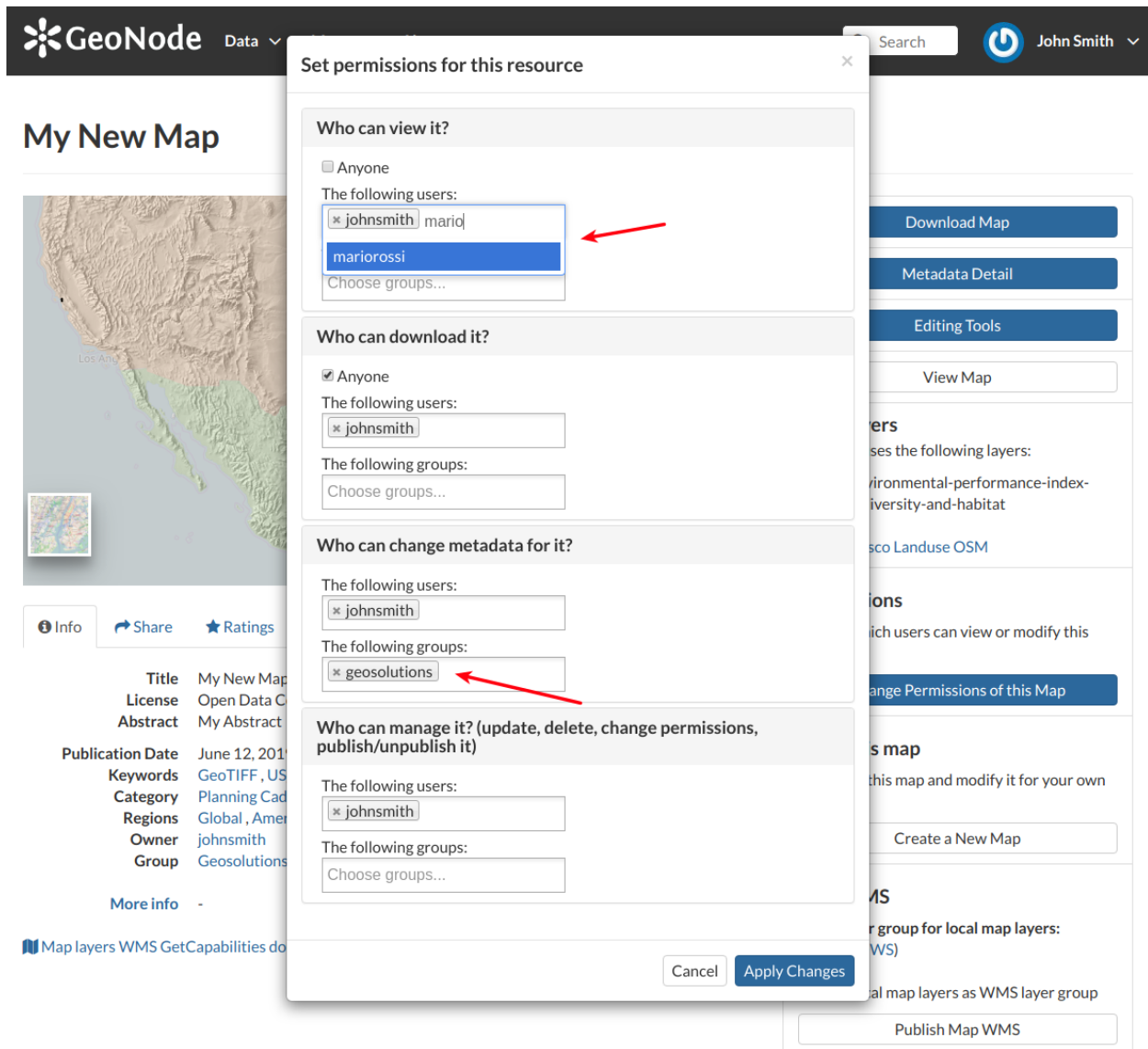


Fig. 162: Map Permissions settings for users and groups

Warning: When assigning permissions to a group, all the group members will have those permissions. Be careful in case of editing permissions.

Click on *Apply Changes* to save these settings.

Exploring Maps

From the *Explore Maps* link of the navigation bar you can reach the *Maps List* page (see [Maps](#)). Select a map you are interested in and click on it, the *Map Page* will open.

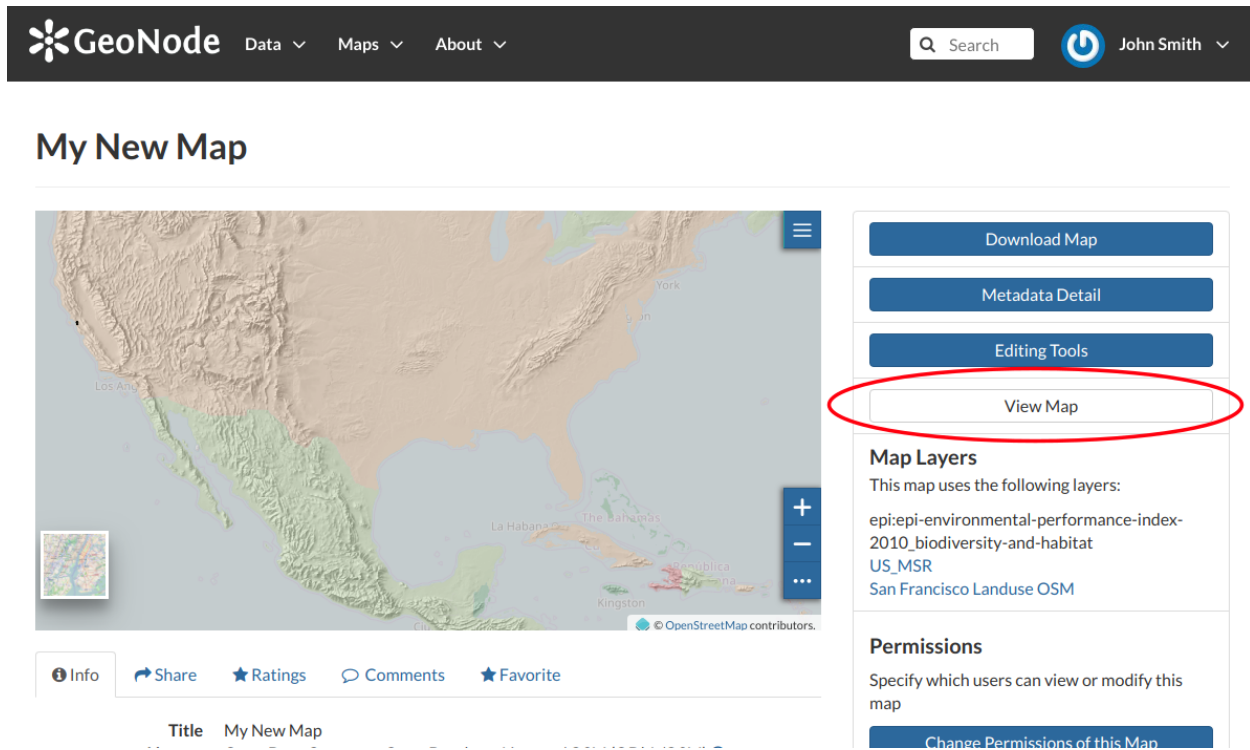


Fig. 163: The *View Map* button

Click on the *View Map* button to open the *Map Viewer*.

The Map Viewer (based on [MapStore](#)) provides the following tools:

- the *Table of Contents (TOC)* to manage the map contents;
- the *Basemap Switcher* to change the basemap (see the next paragraphs);
- the *Search Bar* to search by location, name and coordinates (see the paragraph below);
- the *Options Menu Tools* which contains the link to the *Print* tool, to the layers *Catalog* and to the *Measure* tool;
- the *Sidebar* and its tools such as the *Zoom* tools and the *Get Features Info* tool;
- the *Footer Tools* to manage the scale of the map, to track the mouse coordinates and change the CRS (Coordinates Reference System).

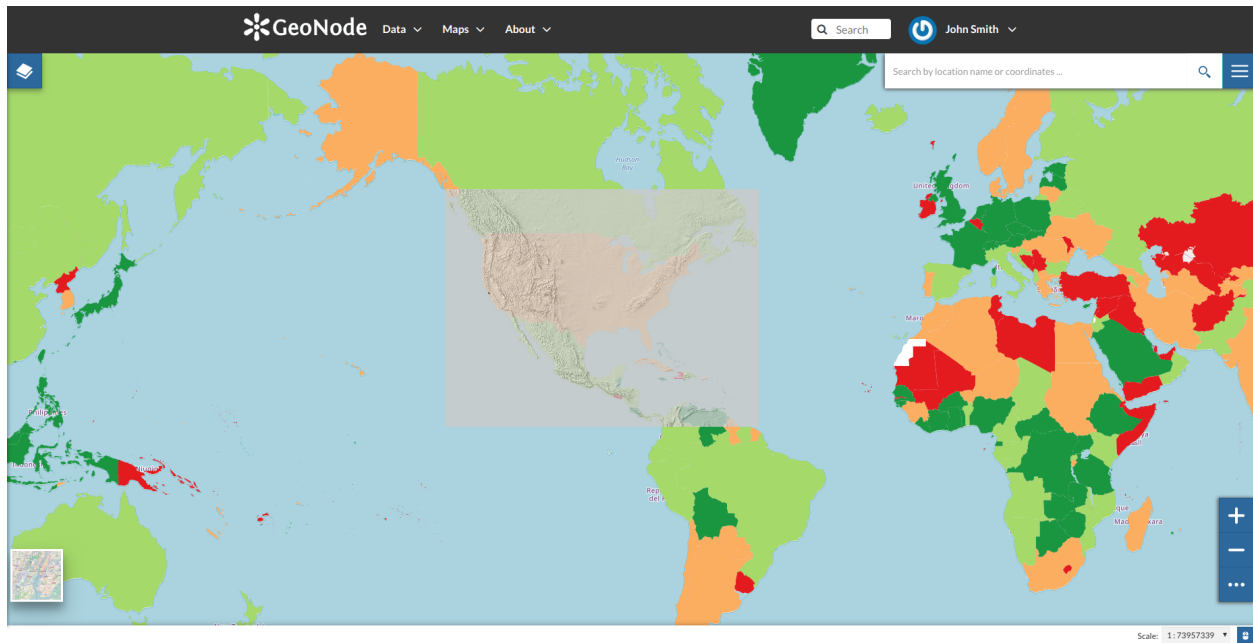





Fig. 164: The Map View

Table of Contents (TOC)





In the upper left corner, click on  to open the *Table Of Contents*, briefly *TOC* from now on, of the map. The *TOC* shows all the layers involved with the *Map* and allows to manage their properties and representations on the map.

From the *TOC* you can:

- manage the layers *Overlap*;
- filter the layers list by typing text in the *Filter Layers* field;
- add new layers from the *Catalog* by clicking the *Add Layer* button;
- manage the layers properties such as *Opacity* (scroll the opacity cursor), *Visibility* (click on  to make the layer not visible, click on  to show it on map);
- manage the *Layer Settings*, see the next paragraph.

Select a *Layer* from the list and click on it, the *Layer Toolbar* should appear in the *TOC*.

The *Toolbar* shows you many buttons:

-  allows you to zoom to the layer extent;
-  drives you through the layer settings customization (see the next paragraph);
-  to explore the features of the layer and their attributes (more information at *Attributes Table*);
-  to delete layers (click on *Delete Layer* to confirm your choice);

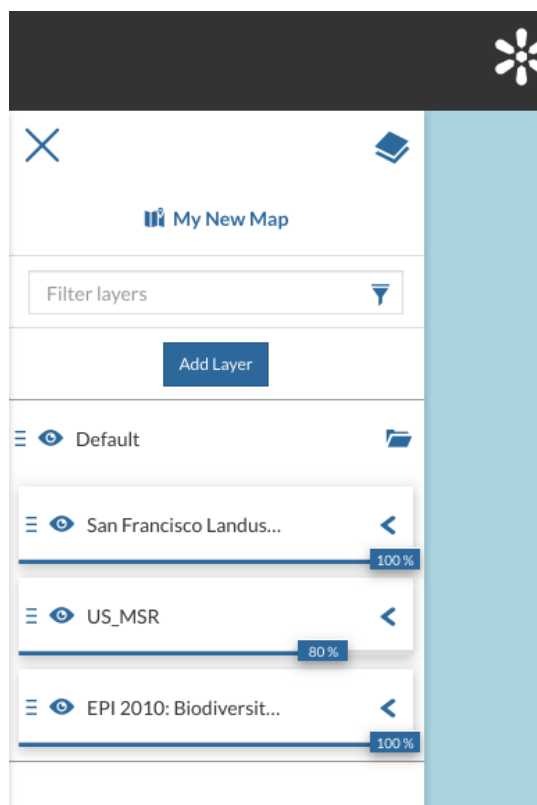
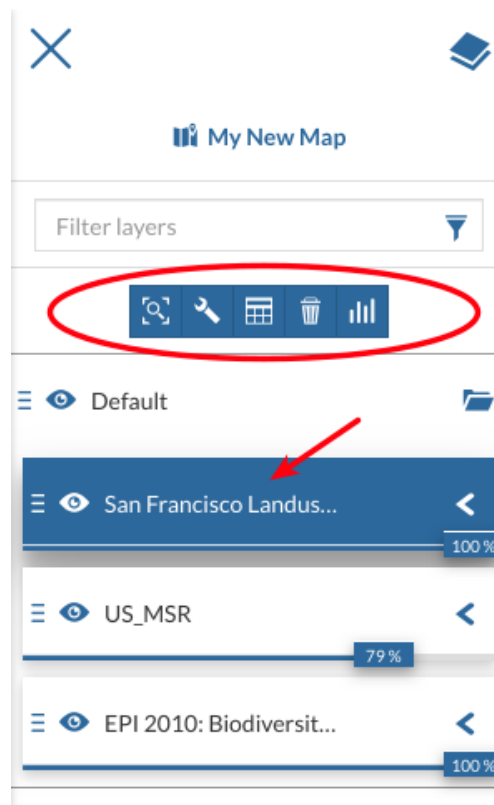
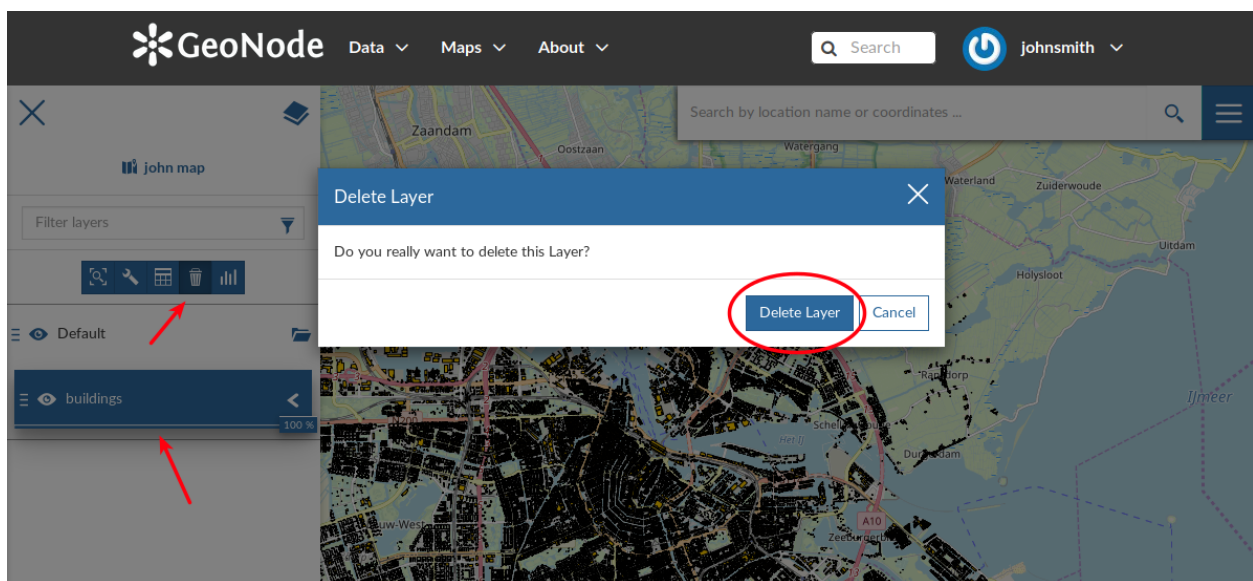



Fig. 165: *The Table Of Contents (TOC)*

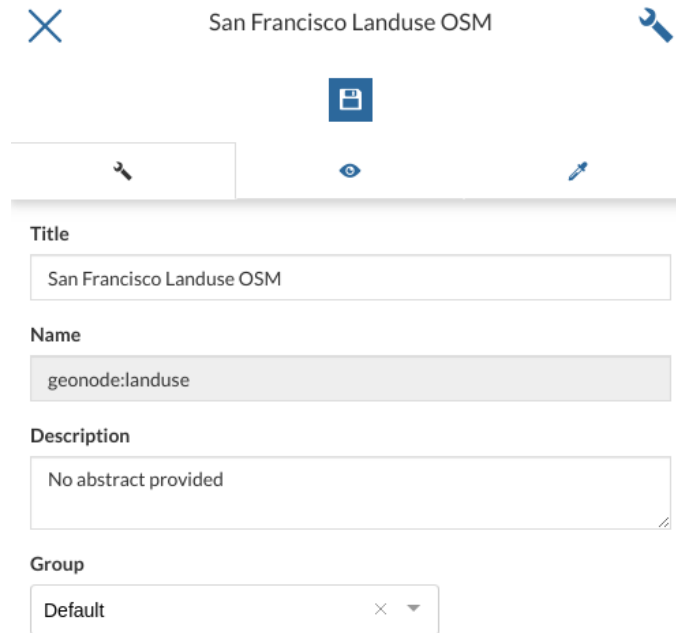
Fig. 166: *Scrolling the Layer Opacity*

Fig. 167: *The Layer Toolbar*Fig. 168: *Deleting Layers*

-  to create *Widgets* (see *Creating Widgets*).

Managing Layer Settings

The *Layer Settings* panel looks like the one below.



The screenshot shows the 'Layer Settings' panel for a layer titled 'San Francisco Landuse OSM'. At the top, there is a close button (X), the layer title, and a settings icon (wrench). Below this is a tab bar with three tabs: 'General' (selected), 'Display', and 'Style'. The 'General' tab contains the following fields:

- Title:** A text input field containing 'San Francisco Landuse OSM'.
- Name:** A text input field containing 'geonode:landuse'.
- Description:** A text area containing 'No abstract provided'.
- Group:** A dropdown menu showing 'Default'.

Fig. 169: *The Layer Settings Panel*

The *Layer Settings* are divided in three groups:

1. *General* settings
2. *Display* settings
3. *Style* settings

In the **General** tab of the *Settings Panel* you can customize the layer *Title*, insert a *Description* and change/create the *Layer Group*.

Click on the **Display** tab to see what are the layer appearance properties you can configure.

The *Format* field allows you to change the output format of the WMS requests.

You can set a numeric value of *Opacity* using the corresponding input field.

You can also set the layer as *Transparent*, decide to *Use cache options* and to use *Single Tile*.

The third tab is the **Style** one. By clicking on it, an advanced *Style Editor* allows you to create new styles and to modify or delete an existing one. See the *Layer Styling* section to read more.

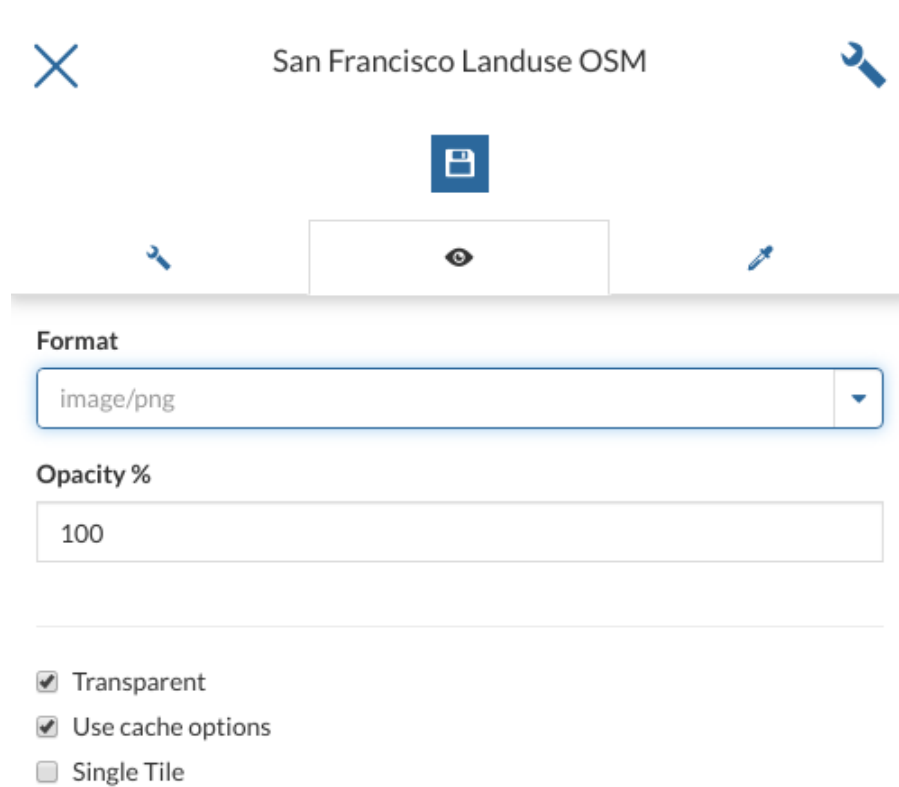



Fig. 170: *The Layer Display Settings Panel*

Attributes Table

When clicking on the  button of the *Table of Contents (TOC)*, the *Attributes Table* panel opens at the bottom of the *Map* page.

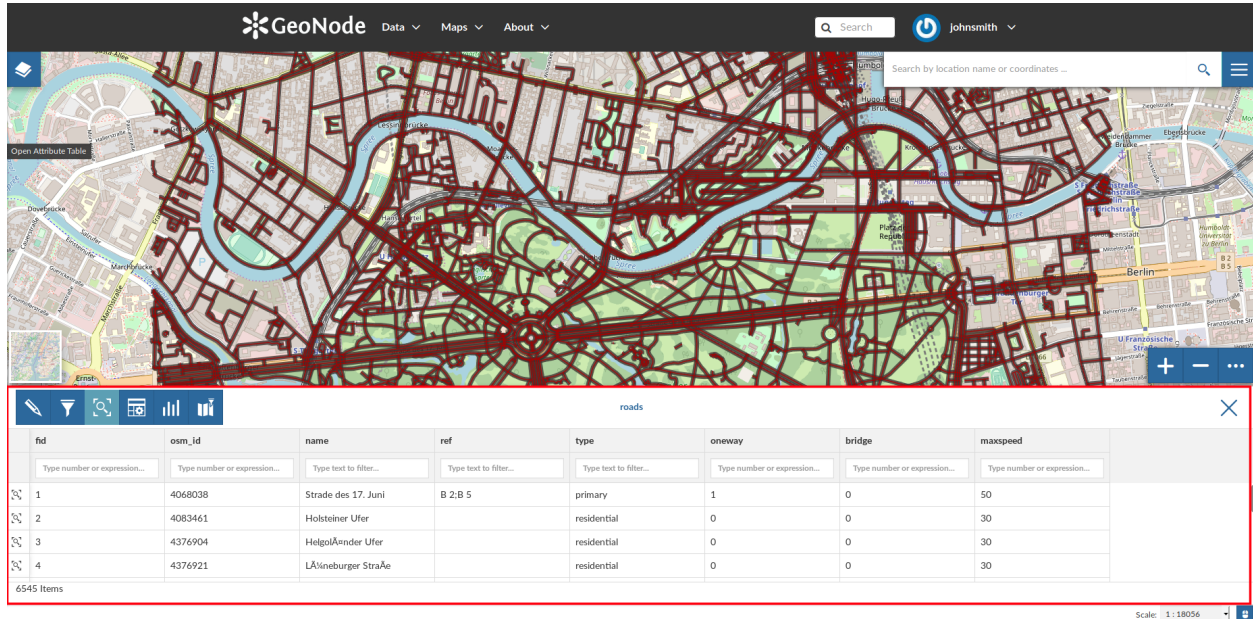



Fig. 171: The Attributes Table Panel

In that panel you can navigate through the features of the layer, zoom to their geometries by clicking on the  icon and explore their attributes.


The *Attribute Tables* has a row for each feature belonging to the layer and a column for each attribute that describes the feature.

Each column has a *Filter* input field through which you can filter the features basing on some value or expression (depending on the data type of the field).


The *Attributes Table* panel contains a *Toolbar* which makes you available some useful functionalities.

Those functionalities are:

- *Edit Mode*

By clicking on  you can start an editing session. It permits you to add new features, to delete or modify the existing ones, to edit geometries. See the *Editing the Layer Data* section for further information.

- *Advanced Search*

Click on , a new panel opens. That panel allows you to filter features in many different ways. This functionality will be explained in depth in the *Advanced Search* section.

- *Zoom to page extent*

Click on  to zoom to the page extent.

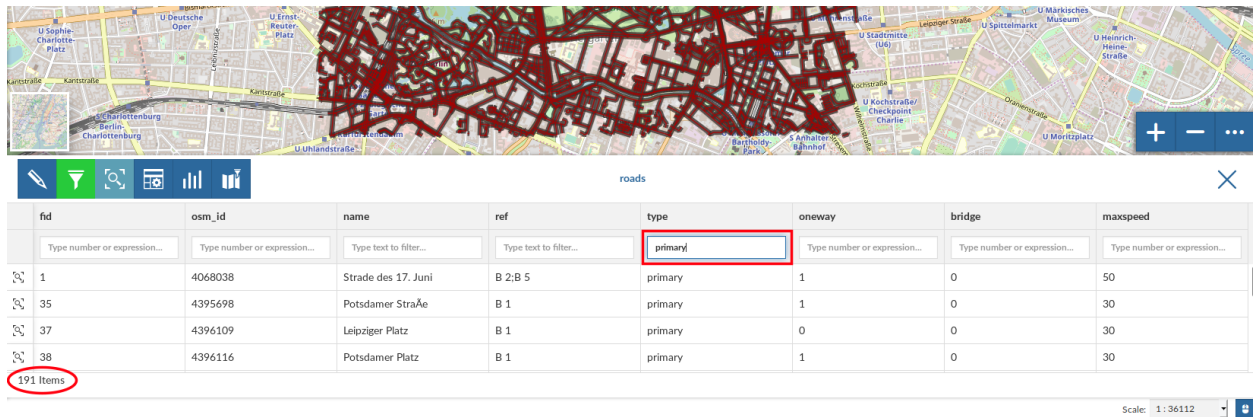


Fig. 172: Filtering Features by Attribute

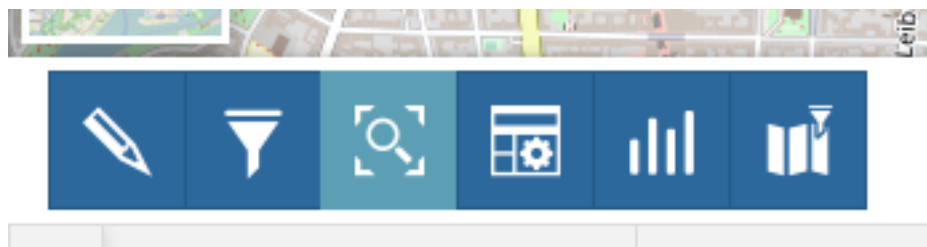


Fig. 173: The Attributes Table Toolbar

- *Hide/show columns*



When clicking on  another panel opens inside the *Attributes Table*. Through that panel you can choose what columns you want to see, see the picture below.

Fig. 174: Hide/Show Columns of the Attributes Table


- *Create a chart*




Through the  button you can open the *Chart Widgets* panel where many functionalities to describe and visualize the layer data are available (see [Creating Widgets](#)).




- *Sync map with filter*


Click on the  icon to synchronize the map with the filter.



Advanced Search


As mentioned before, GeoNode allows both an attribute based and spatial filtering. When clicking on  from the layer *Attributes Table* the *Advanced Search* panel opens and shows you three different filtering functionalities:

- In the **Attribute Filter** section you can compose a series of conditions about the attributes of the layer. Click on  to insert a new empty condition. Select the attribute you are interested in, select an operator and type a comparison value. You can group conditions through the *Add Group*  button. Click on  to perform



Attribute Filter

Match any▼ of the following conditions:


Region of interest

Filter Type

Select...▼

Geometric Operation

Intersects▼

Layer filter

Target layer


Select layer▼

Fig. 175: *Advanced Search*

the search.

Fig. 176: Filtering by Attributes

You can also decide if *All* the conditions have to be met, if only *Any* or *None* of them (see the red arrow in the picture above).

- The **Region of interest** filtering allows you to filter features that have some relationship with a spatial region that you draw on the map. Select the *Filter Type* (Circle, Viewport, Polygon or Rectangle), draw the spatial region of interest on the map, select a *Geometric Operation* (Intersects, Bounding Box, Contains or Is contained) and then click on .
- Through the **Layer Filter** you can select only those features which comply with some conditions on other layers of the map. You can also add conditions on attributes for those layers.

You can read more about the *Attributes Table* and the *Advanced Search* on the [MapStore2 Documentation](#).

Creating Widgets


Widgets are graphical elements that describe the layers data. They can be of different types such as *Charts*, *Texts*, *Tables* and *Counters*. Through the  button of the *Table of Contents (TOC)* you can open the *Widgets* panel.

Chart Widgets

Chart Widgets are graphical representations of the layer data. They can be *Bar Chart*, *Pie Chart* or *Line Chart* as shown in the picture below.

Lets create a new **Bar Chart**.

Click on *Bar Chart* then select the *X Attribute*, the *Y Attribute*, the *Operation* and the *Color* do you prefer. You can also display the *Legend*, *Hide the Y axis*, *Hide the grid* and decide what *Label* display into the legend.

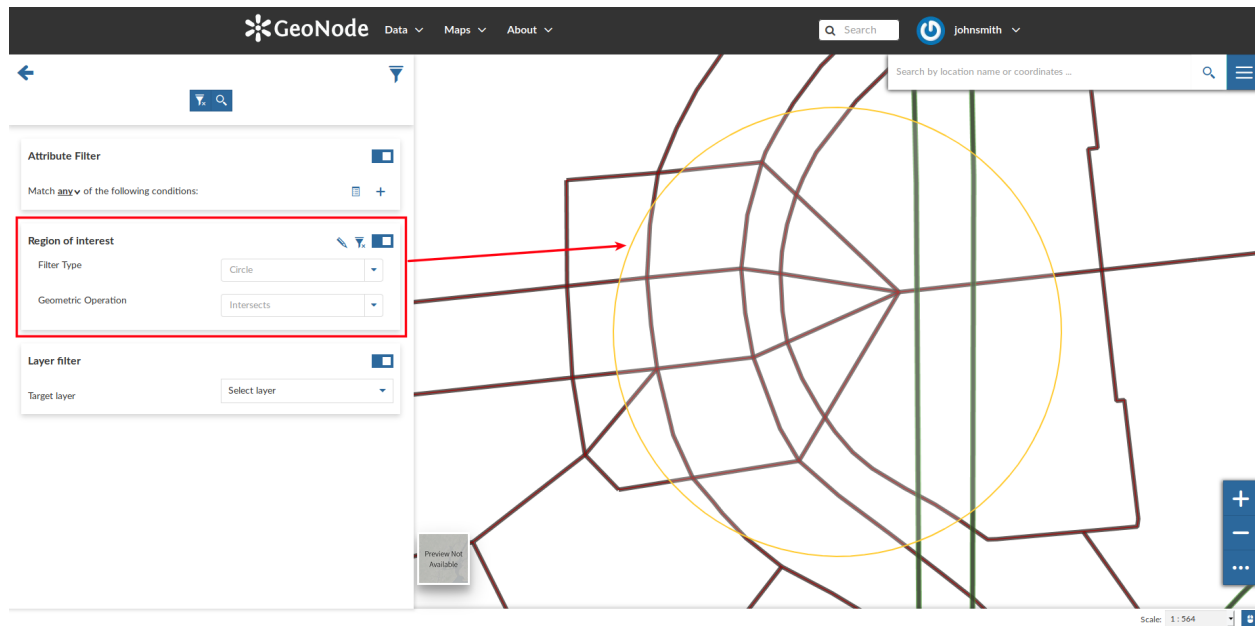


Fig. 177: Filtering by Region Of Interest

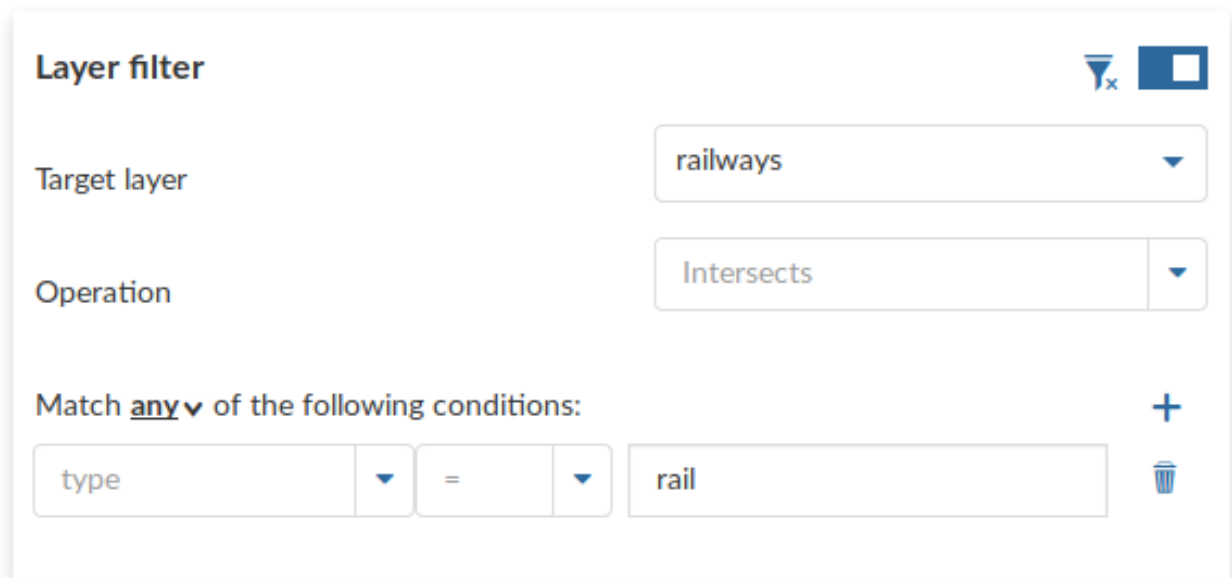


Fig. 178: Layer Filtering

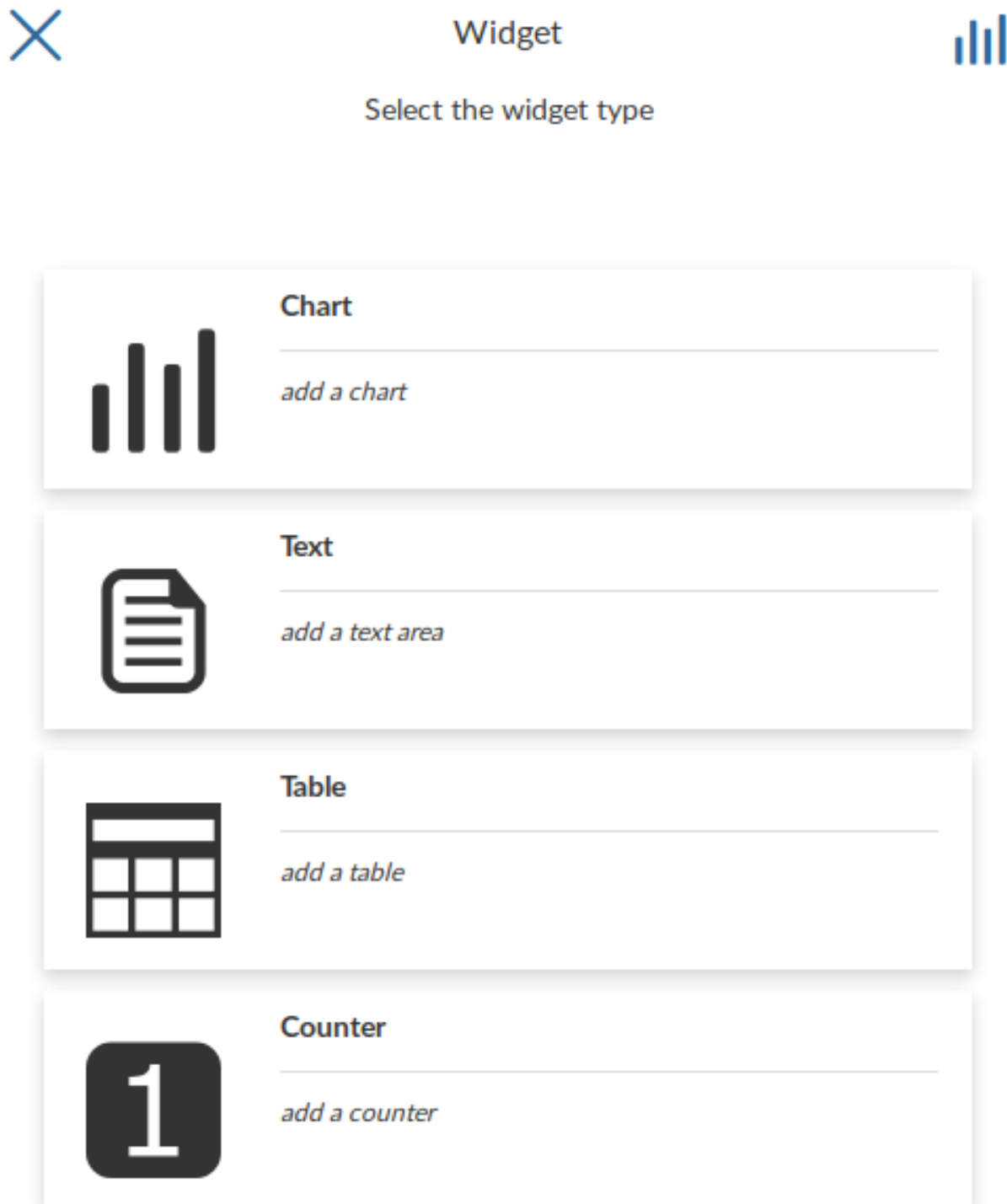
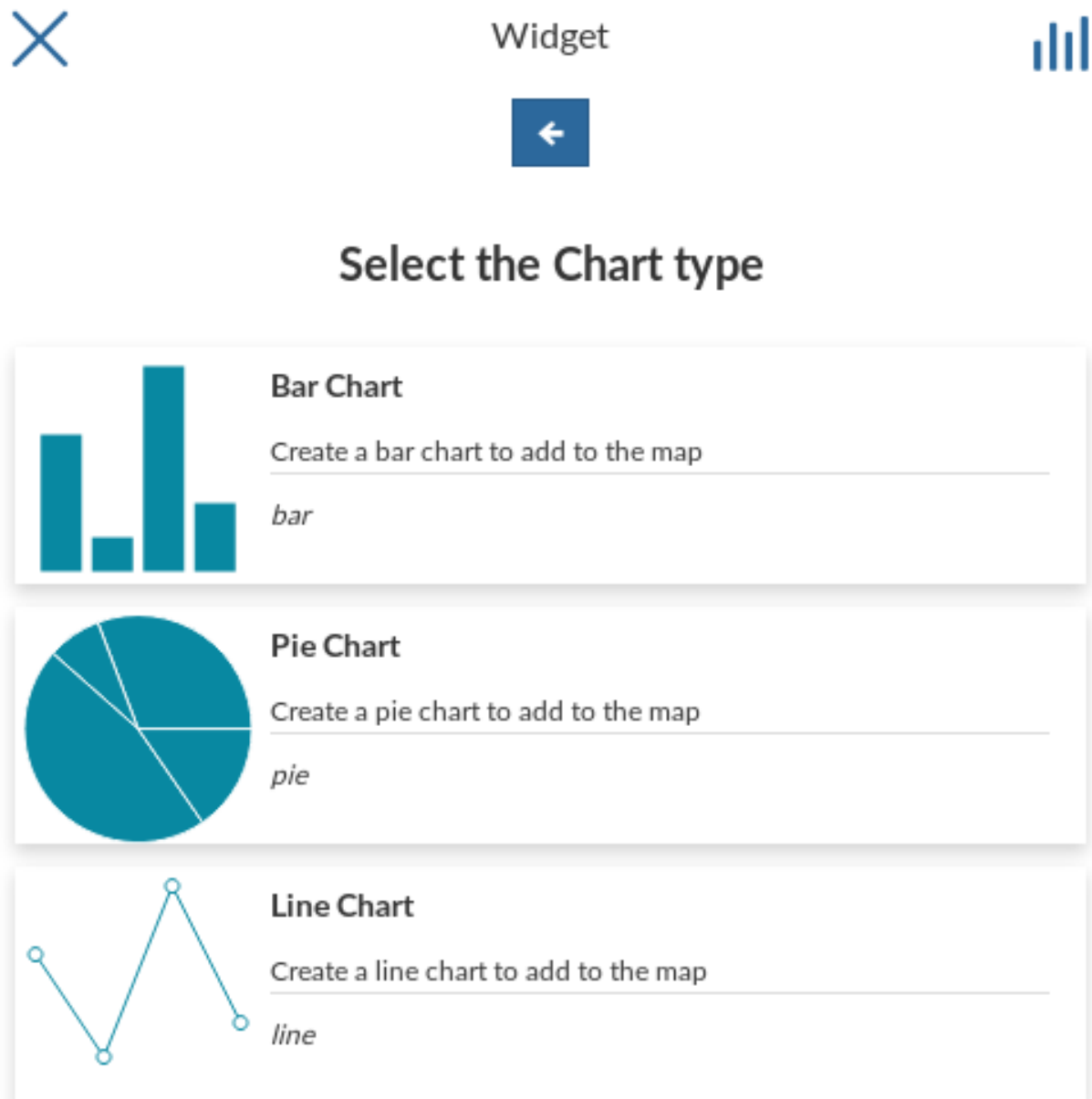


Fig. 179: Creating Widgets

Fig. 180: *Chart Widgets*





Now you can filter the data to be considered for the chart by clicking on . We don't need any filter so click  to configure other widget options. Insert a *Title* and a *Description* and click on *Save* .

Fig. 181: Chart Widgets Creation

The green  icon means that the chart is connected to the viewport.

Expanding the options menu of the widget you can *Show the plotted data*, *Edit* the widget or *Delete* it, *Download* the data as a CSV file or *Export* the image of the graph.

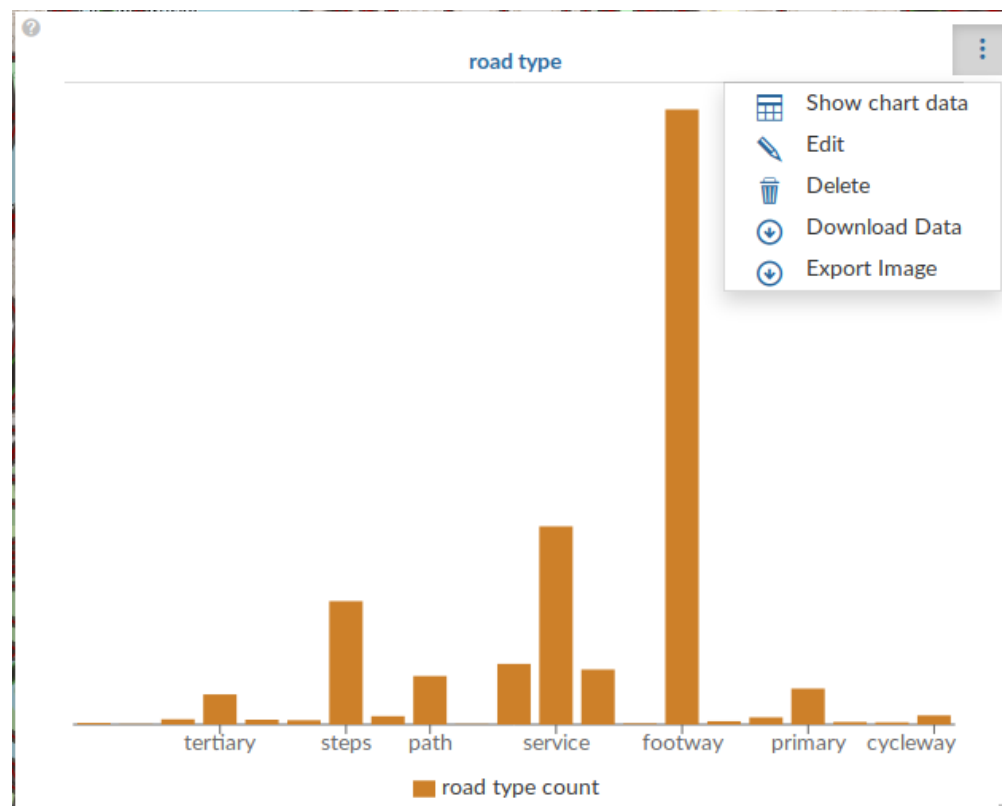



Fig. 182: Chart Widgets Options

Text Widgets

If you select *Text* on the *Widgets* panel you can create *Text Widgets*. Add a *Title* and the desired descriptive text, then click on .

The resulting widget looks like the following.

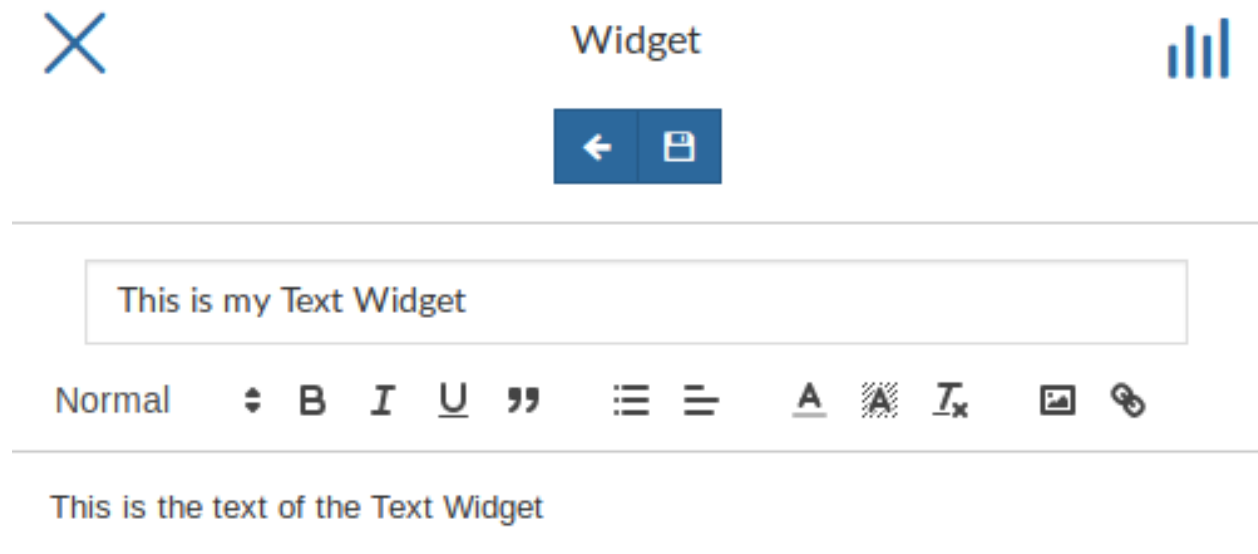


Fig. 183: *Text Widgets Creation*

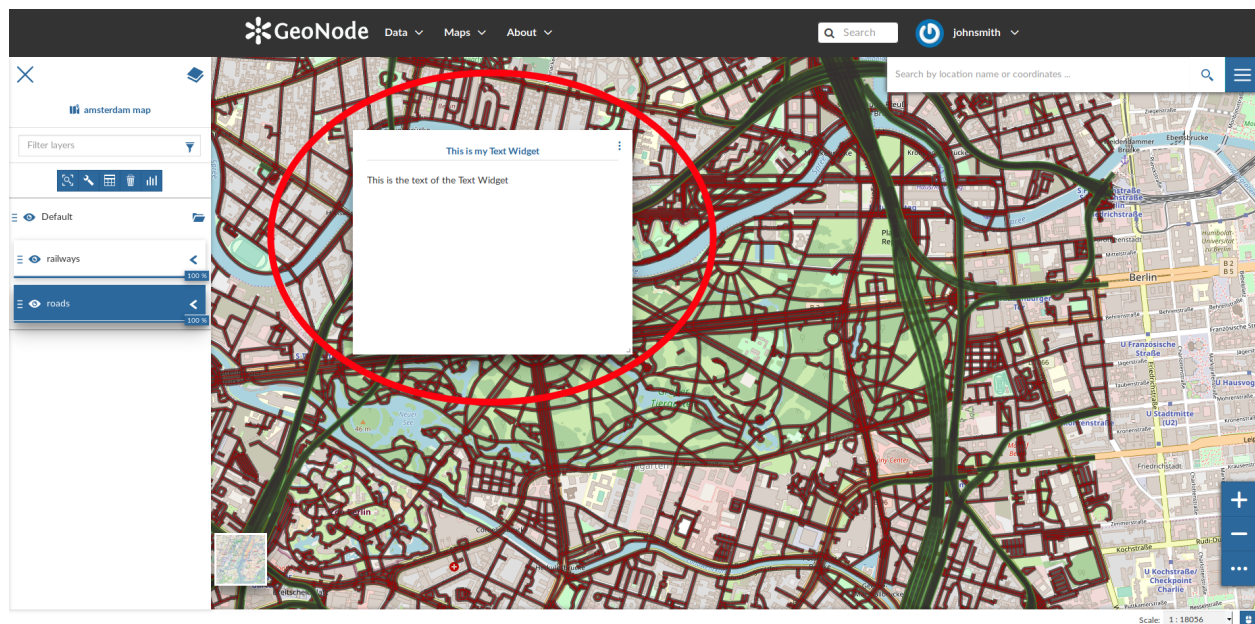



Fig. 184: *My Text Widget*

Table Widgets



Through the *Table Widgets* you can add the *Attributes Table* of the layer to the map. You can decide to show a subset of the features, through filters, and you can select one or more columns/attributes.

So, choose what attributes you are interested in and click on .

Insert *Title* and *Description* (optional) and click on . The example below shows the *Table Widget* on the map.

Counter Widgets

Counter Widgets are numeric representations of some attributes. For example you can represent the average speed limit on a road network.

Click on , insert *Title* and *Description* then click on .

The GeoNode map viewer is [MapStore](#) based, see the [MapStore Documentation](#) for further information.

Timeline

GeoNode can manage layers with a *time dimension*. Those vector layer may vary their data through time so it is useful to represent that variation on the map.

The [MapStore](#) based map viewer used in Geonode makes available the **Timeline** tool which allows you to observe the layers' evolution over time, to inspect the layer configuration at a specific time instant and to view different layer configurations time by time dynamically through animations (see the [MapStore Documentation](#) for further details).


Warning: Timeline actually works only with WMTS-Multidim extension (WMS time in capabilities is not fully supported).

When loading a temporal layer into the map, the *Timeline* opens automatically.

On the left side of the *Timeline* panel you can set the time value in which you want to observe the data. You can type it directly filling out the corresponding input fields or by using the up/down arrows.

On the other side there are the buttons responsible for managing the animations.

In particular you can *Play* the animation by clicking , go back to the previous time instant through , go forward to next time step using  and stop the animation by clicking .

The *Timeline* panel can be expanded through the  button.

The expanded section of the *Timeline* panel contains the *Time Layers List* and an *Histogram* which shows you:

- the distribution of the data over time

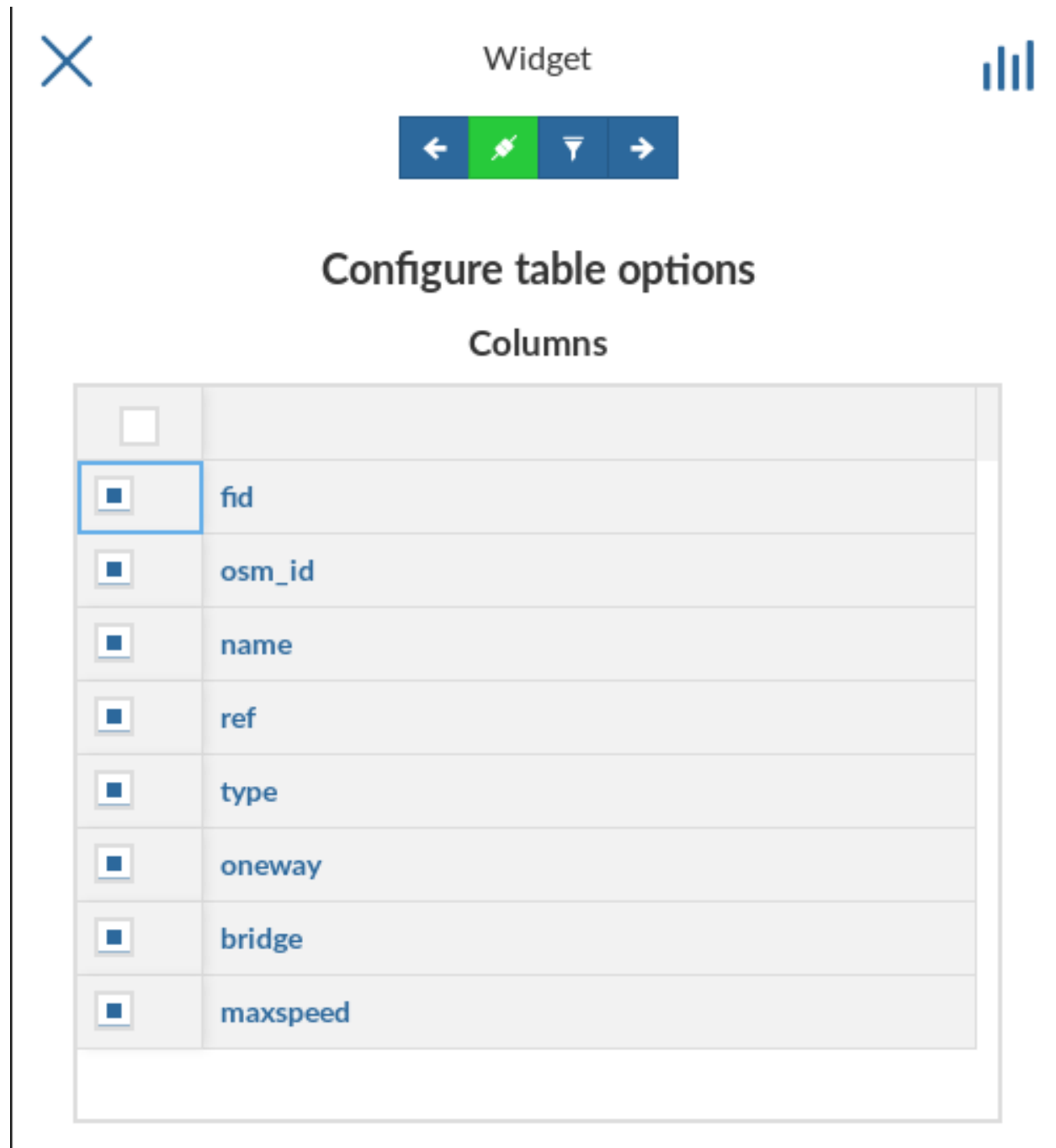


Fig. 185: Table Widgets Columns

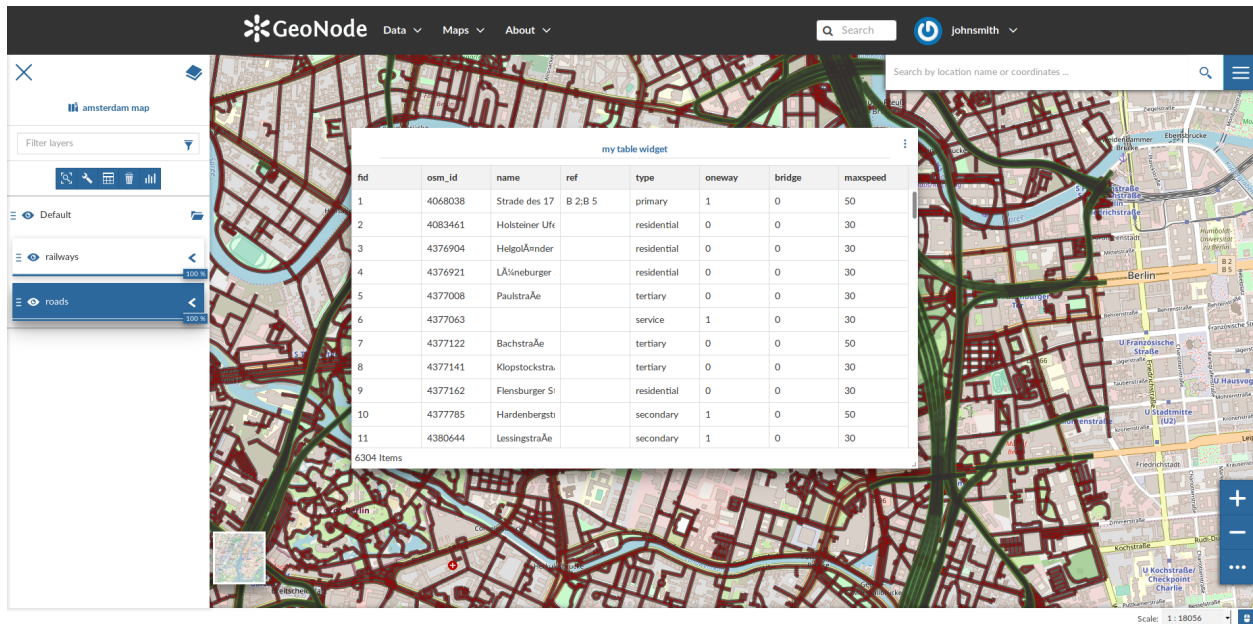


Fig. 186: Table Widget



Fig. 187: Counter Widget Creation

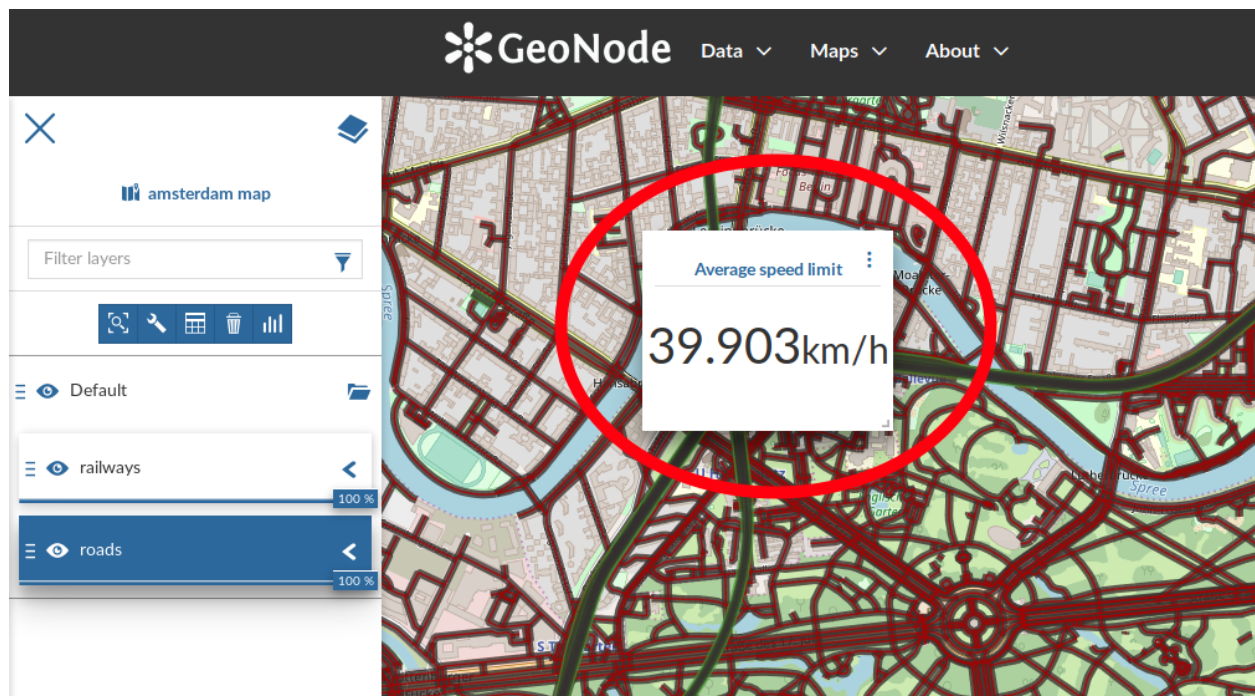


Fig. 188: Counter Widget

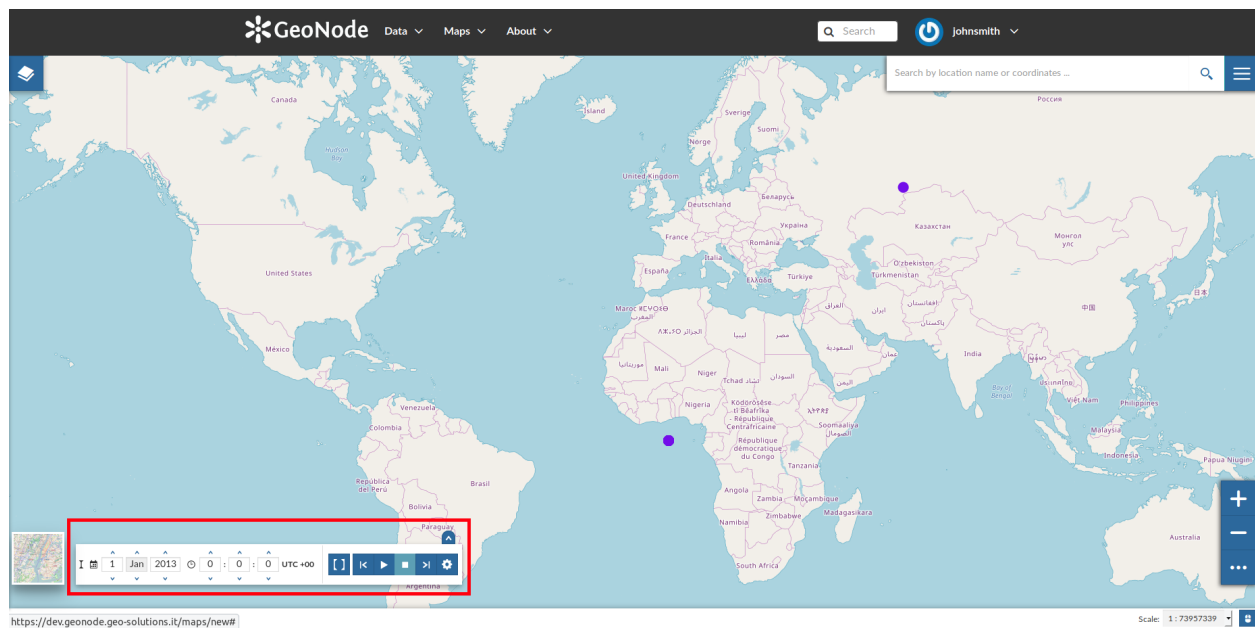


Fig. 189: The Timeline



Fig. 190: *The Time Control Buttons*

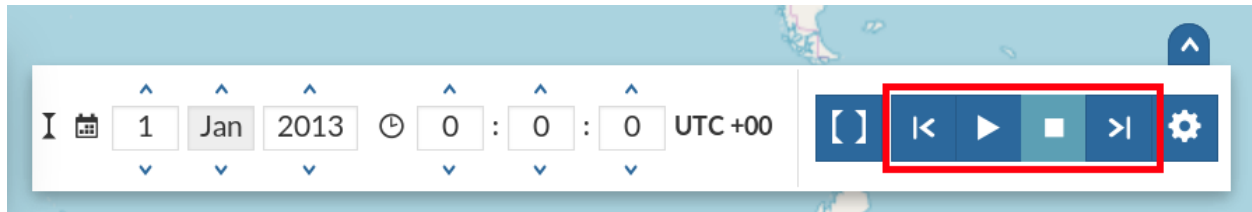


Fig. 191: *The Animation Control Buttons*

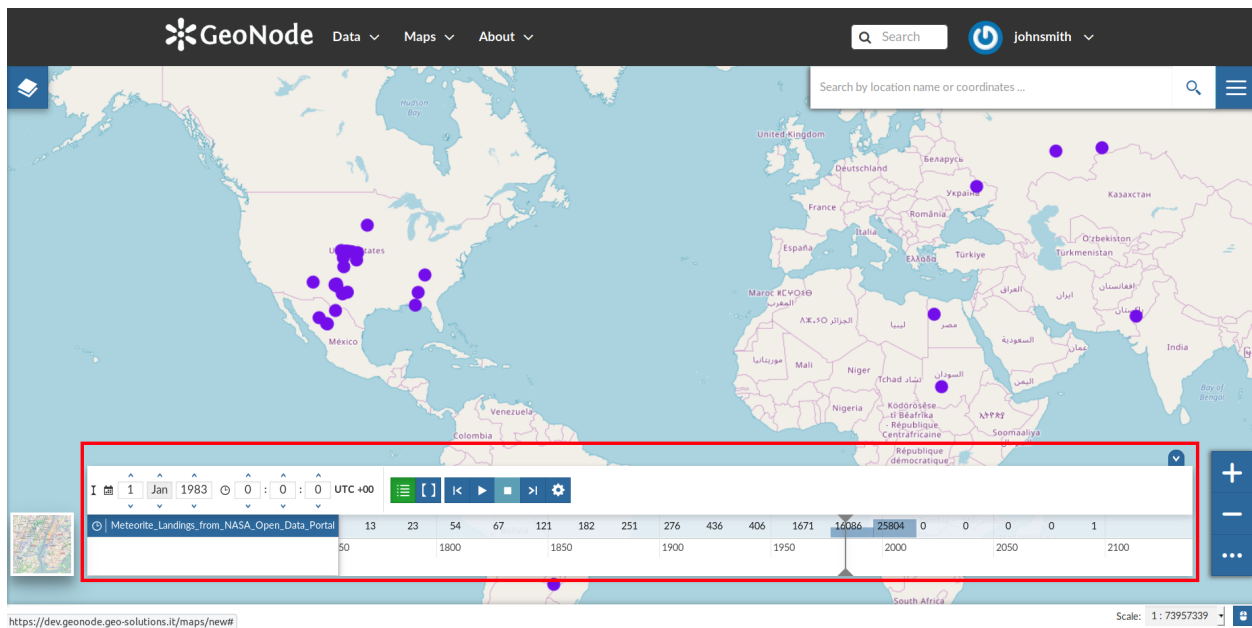


Fig. 192: *The Expanded Timeline*

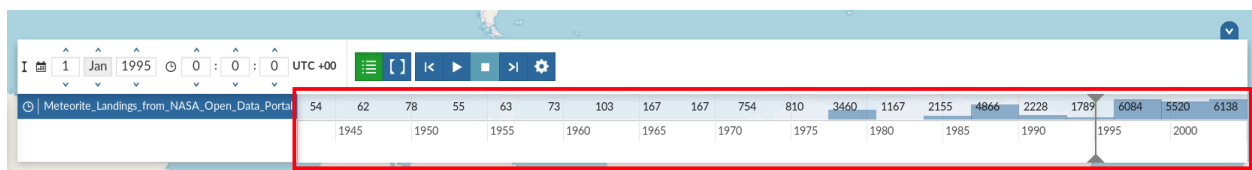
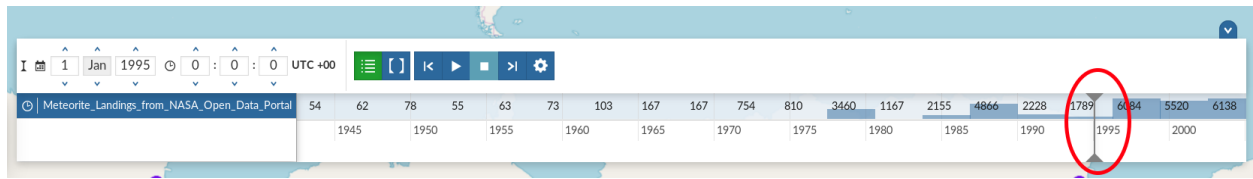




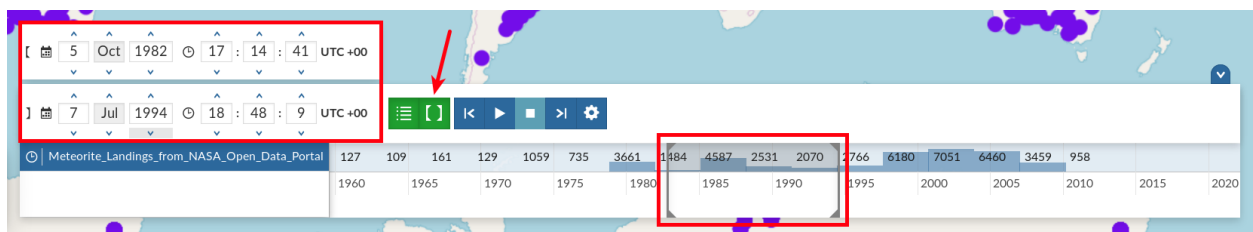
Fig. 193: *The Timeline Histogram*

- the *Time Cursor*

Fig. 194: *The Time Cursor*


You can show/hide the layers list by clicking  (it is active by default).


Through the *Time Range* function you can observe the data in a finite temporal interval. Click on  and set the initial and the final times to use it.

Fig. 195: *The Time Range Settings*

Animations

The *Timeline* allows you to see the data configurations (one for each time in which the data are defined) through ordered sequences of steps.

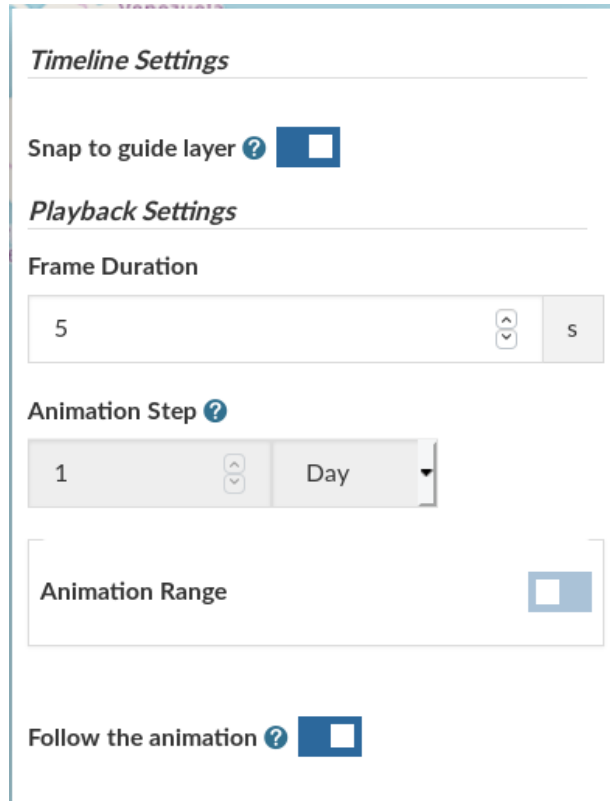
As said before, you can play the resulting *Animation* by clicking the play button . The layer data displayed on map will change accordingly to the time reach by the cursor on the *Histogram*.

By clicking on  you can manage some *Animation Settings*.

You can activate the *Snap to guide layer* so that the time cursor will snap to the selected layer's data. You can also set up the *Frame Duration* (by default 5 seconds).

If the *Snap to guide layer* option is disabled, you can force the animation step to be a fixed value.

The *Animation Range* option lets you to define a temporal range within which the time cursor can move. See the following gif to better understand how the *Animation* works or take a look at the [MapStore Documentation](#).



Timeline Settings

Snap to guide layer ? ☒

Playback Settings

Frame Duration

5 s

Animation Step ?


1 Day

Animation Range ☐

Follow the animation ? ☒

Fig. 196: *The Timeline Settings*Fig. 197: *The Timeline Animation*

Options Menu Tools

At the top-right corner of the *Map* there is a *Burger Menu* button . Click on it to open the *Map Options* panel.

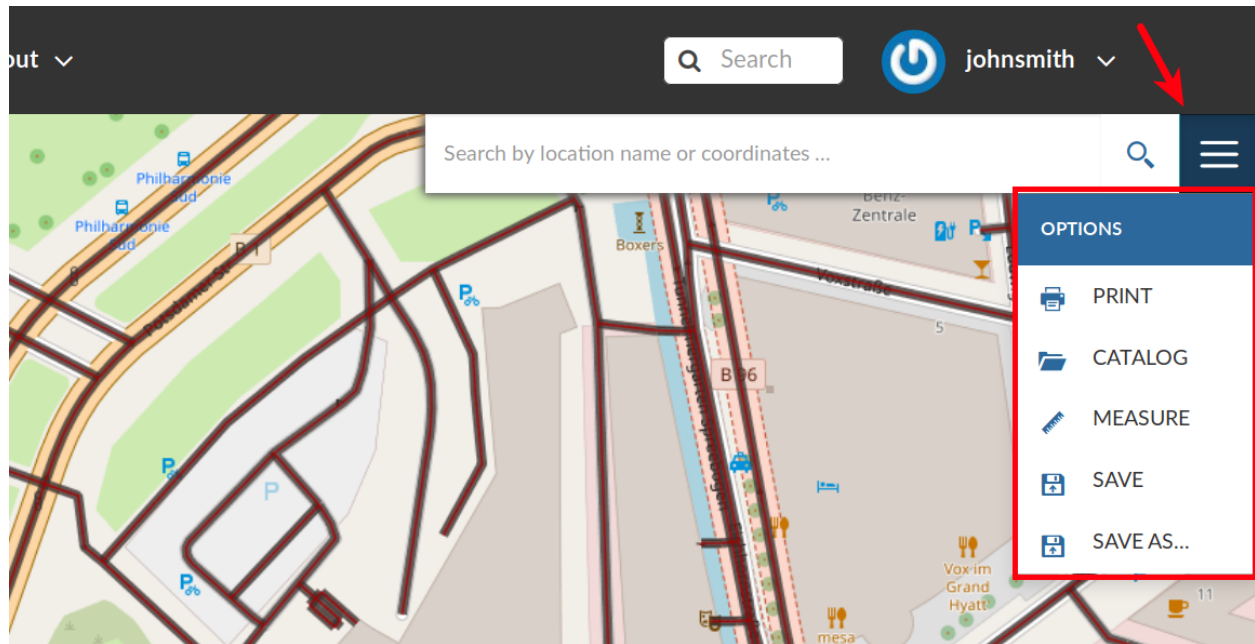


Fig. 198: The Map Options Menu

We will explain those tools more in depth in the next paragraphs.

Printing a Map

The *MapStore* based map viewer of GeoNode allows you to print your map with a customizable layout. Click the *PRINT* option from the *Map Options Menu*, the **Printing Window** will open.

From this window you can:

- enter *Title* and *Description*;
- choose the *Resolution* in dpi;
- customize the *Layout*
 - the *Sheet size* (A3, A4);
 - if include the legend or not;
 - if to put the legend in a separate page;
 - the page *Orientation* (Landscape or Portrait);
- customize the *Legend*
 - the *Label Font*;
 - the *Font Size*;

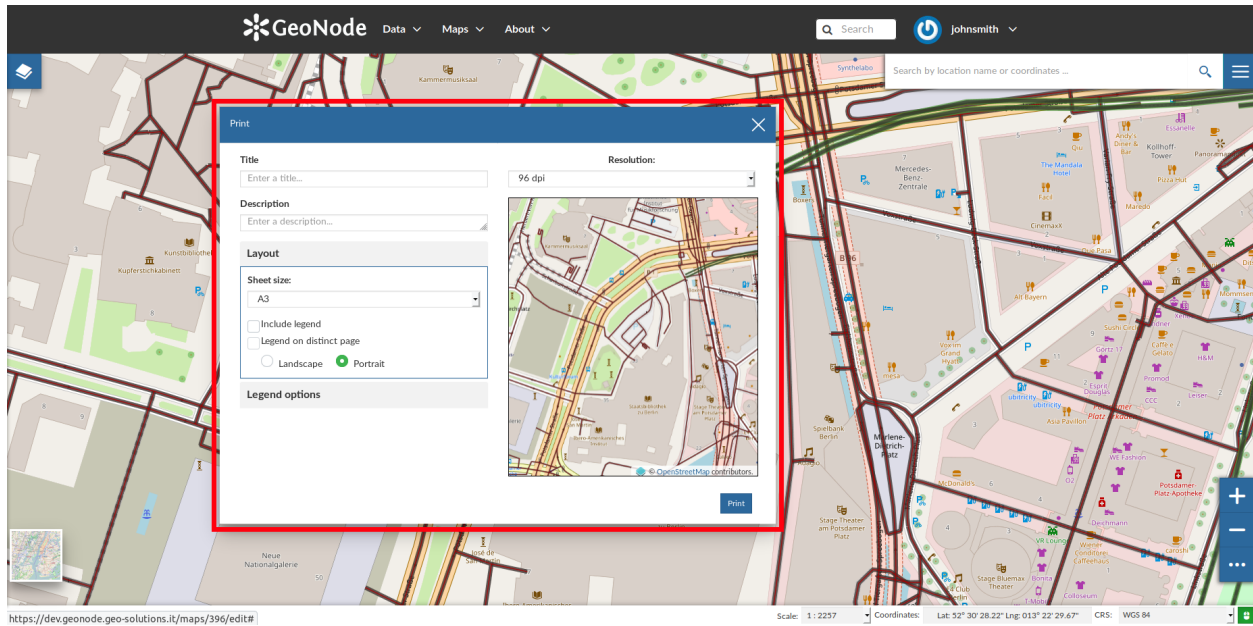


Fig. 199: The Printing Window

- the *Font Emphasis* (bold, italic);
- if *Force Labels*;
- if use *Anti Aliasing Font*;
- the *Icon Size*;
- the *Legend Resolution* in dpi.

To print the map click on *Print*.

The Layers Catalog

All the layers available in GeoNode, both uploaded and remote, can be loaded on the map through the *Catalog*. Click on the *CATALOG* option of the *Map Options Menu* to take a look at the catalog panel.

You can navigate through layers and look at their *Thumbnail* images, *Title*, *Description* and *Abstract*. Click on *Add To Map* to load a layer into the map, it will be also visible in the *Table of Contents (TOC)*.

Performing Measurements

Click on the *MEASURE* option of the *Map Options Menu* to perform a measurement. As you can see in the picture below, this tool allows you to measure *Distances*, *Areas* and the *Bearing* of lines.

To perform a measure draw on the map the geometry you are interested in, the result will be displayed on the left of the unit of measure select menu (this tool allows you to change the unit of measure also).

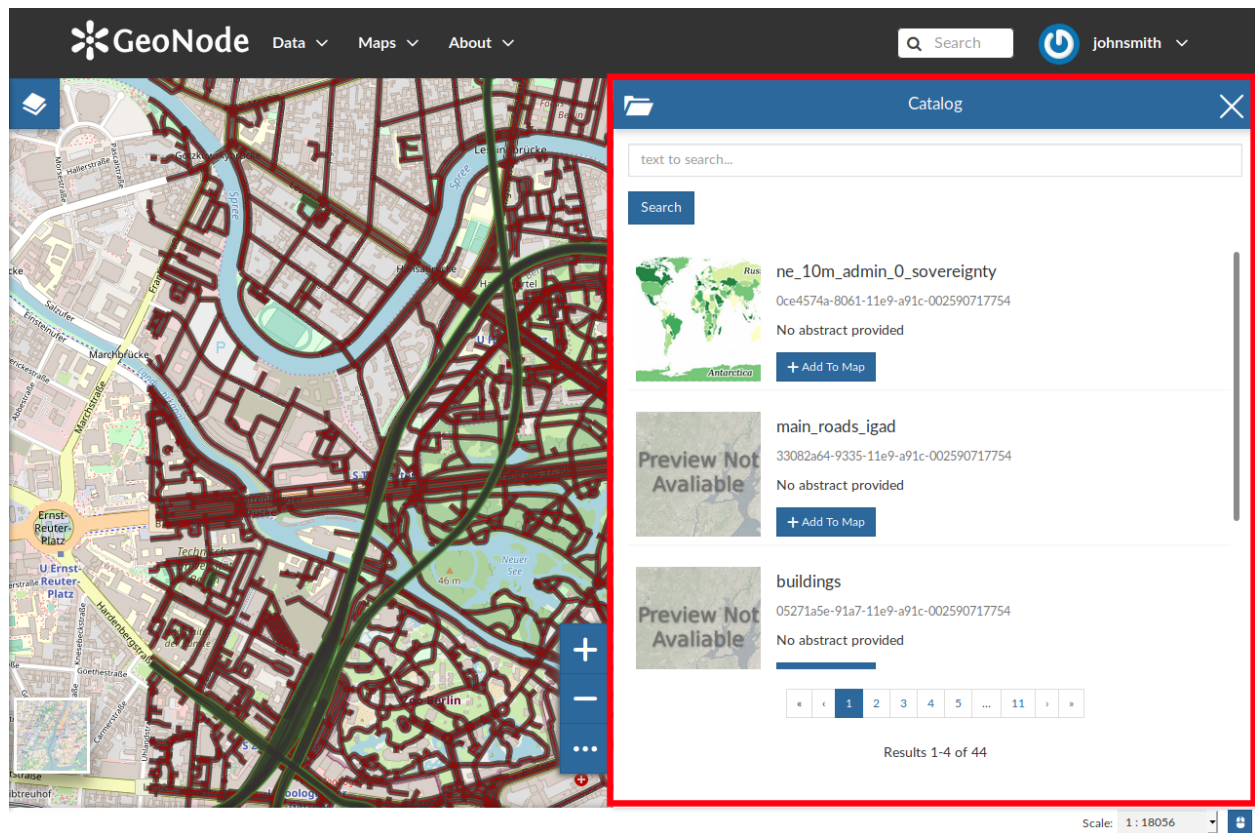
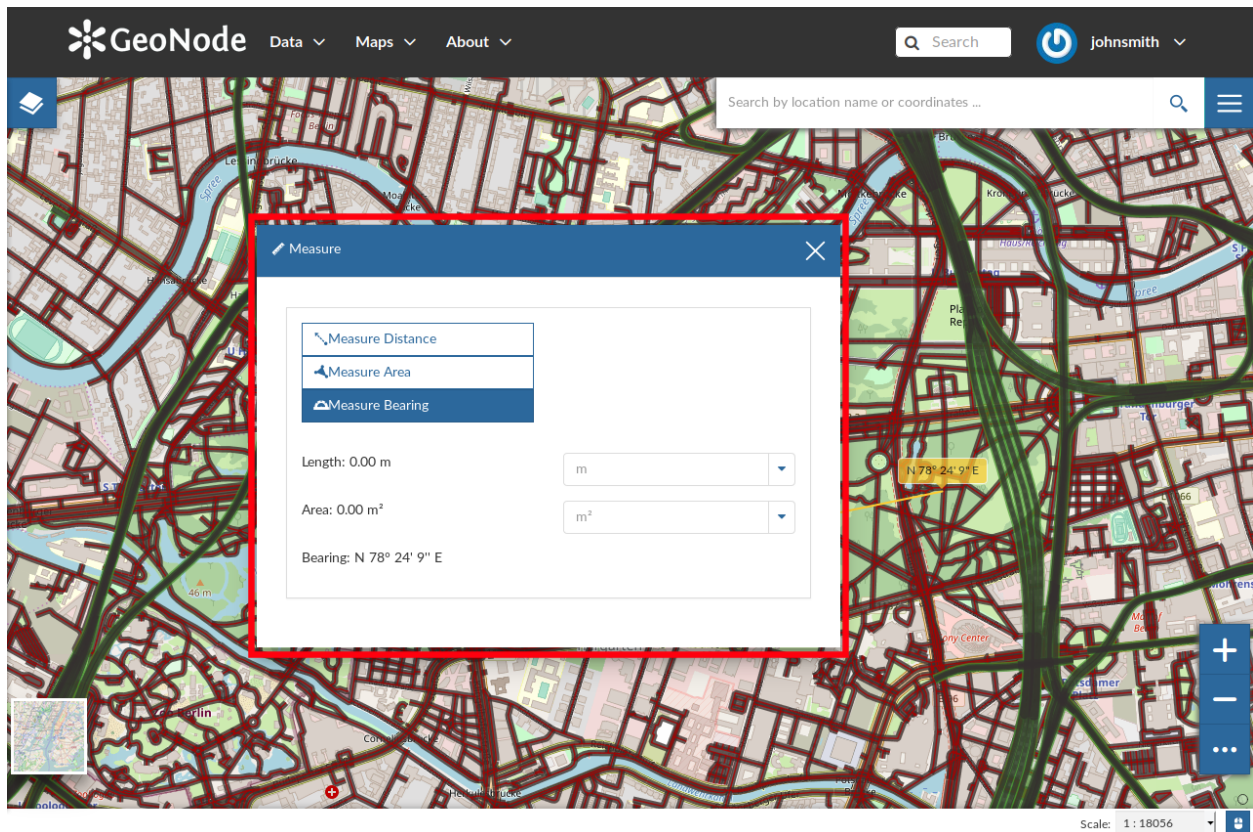
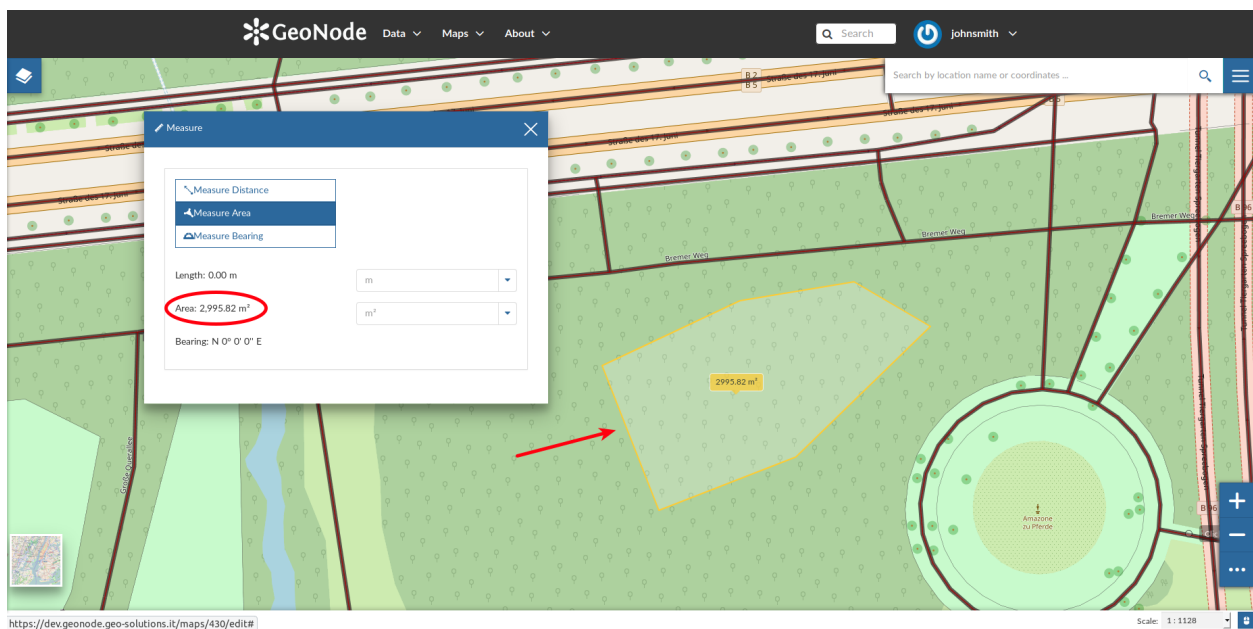


Fig. 200: *The Layers Catalog*

Fig. 201: *The Measure Tool*Fig. 202: *Measuring Areas*

Saving a map

Once all the customizations have been carried out, you can *Save* your map by clicking on the *SAVE AS* option of the *Map Options Menu*.

A new popup window will open.

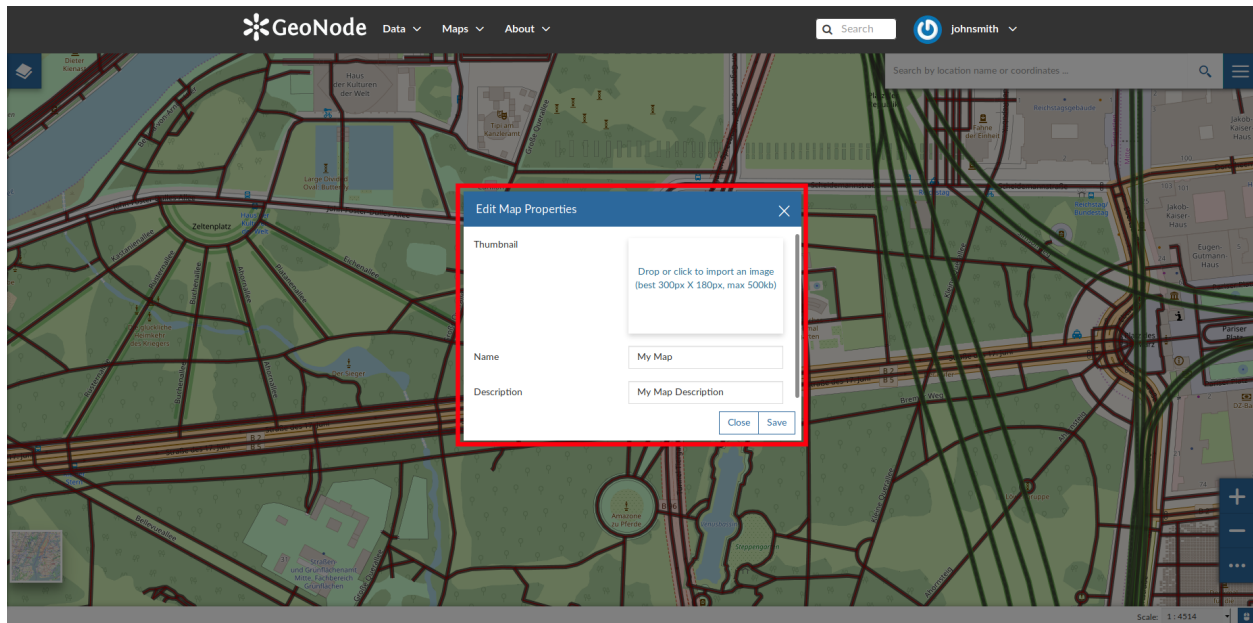


Fig. 203: *Saving Maps*

You have to fill out a *Title* and an optional *Description*, then click on *Save*. The page will reload and your map should be visible in the *Maps* list.

Search Bar


The *Search Bar* of the map viewer allows you to find point of interests (POIs), streets or locations by name. Lets type the name of some place then select the first record.

The map will automatically re-center on that area delimiting it by a polygon in the case of an area, by a line in the case of a linear shape (e.g. streets, streams) and by a marker in the case of a point.

Sidebar Tools

The *Map Viewer* makes also available the *Sidebar*. It is a navigation panel containing various tools that help you to explore the map such as tools for zooming, changing the extent and querying objects on the map.

By default the *Sidebar* shows you the zooming buttons  and , other options can be explored by clicking on

 which expands/collapses the toolbar.

GeoNode Data Maps About Search johnsmith

Explore Maps

Create a New Map

13 Maps found

Selected Maps

Add maps through the "checkboxes".

Set permissions

Filters Clear

TEXT

Search by text

KEYWORDS

CATEGORIES

OWNERS

GROUPS

GROUP CATEGORIES

DATE

REGIONS

EXTENT

My Map

My Map Description

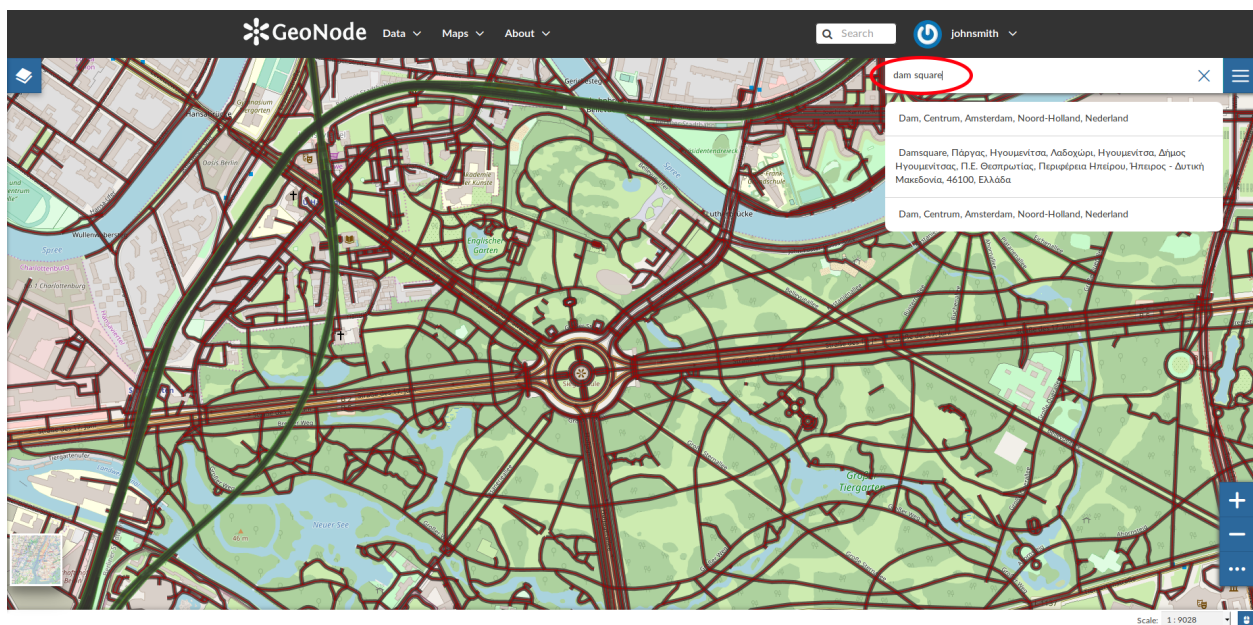
johnsmith 21 Jun 2019 0 0 0 View Map

states

Admin 21 Jun 2019 0 0 0 View Map

taz2

Admin 20 Jun 2019 0 0 0 View Map

Fig. 204: *Your Map into the List*Fig. 205: *The Search Bar*

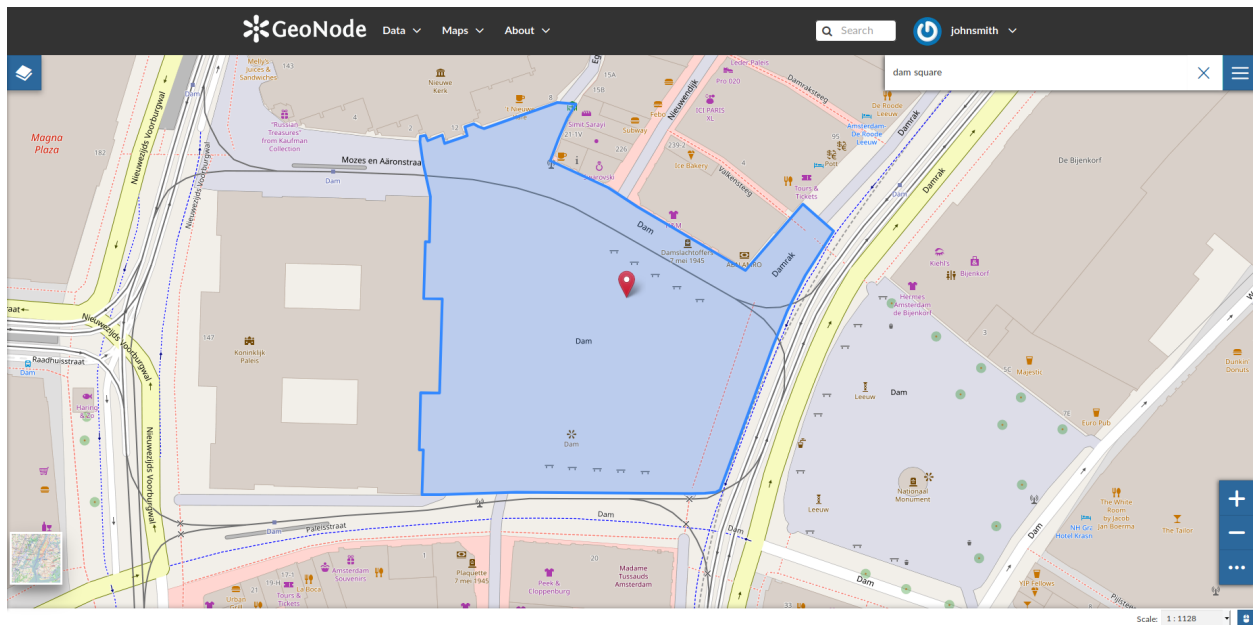


Fig. 206: Result of a Search

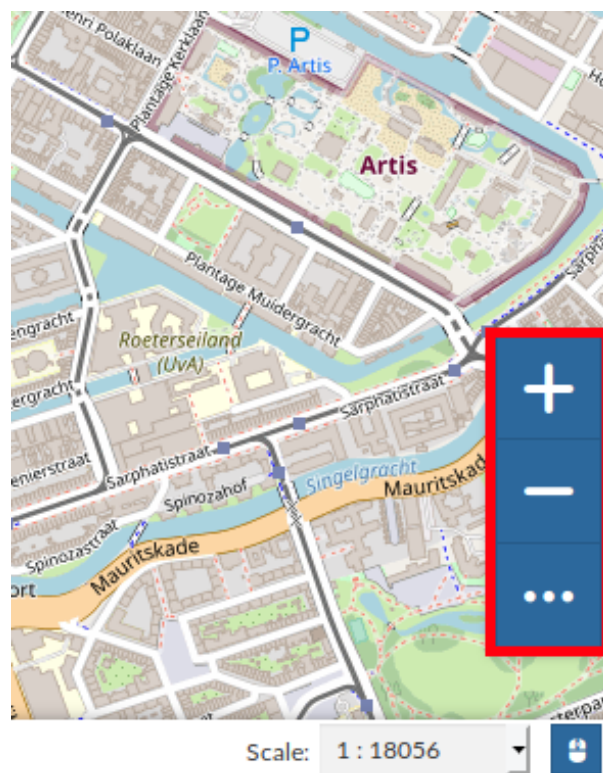


Fig. 207: The Default Sidebar

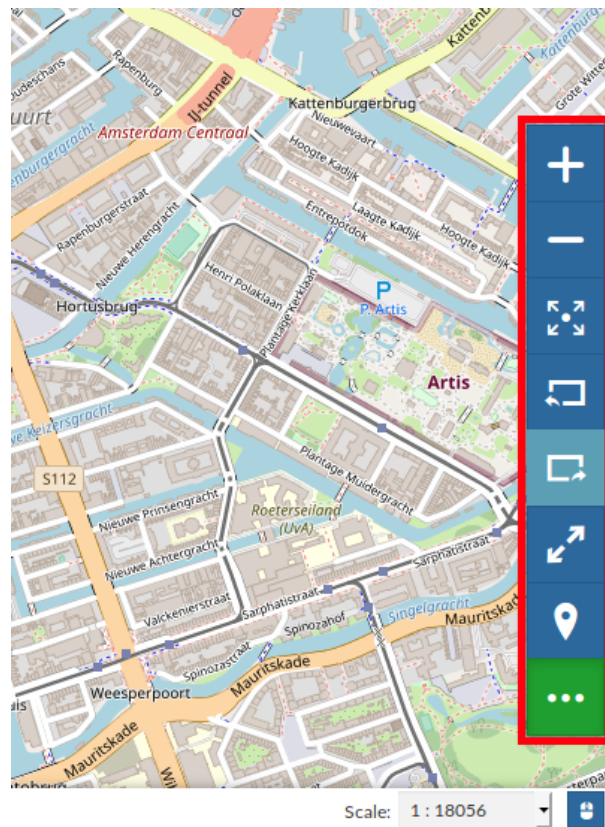


Fig. 208: *The Expanded Sidebar*

The *Sidebar* contains the following tools:






- The *Query Objects on map* allows you to get feature information through the  button. It allows you to retrieve information about the features of some layers by clicking them directly on the map.

Fig. 209: *Querying Objects on map*

When clicking on map a new panel opens. That panel will show you all the information about the clicked features for each active loaded layer.

- You can *Zoom To Max Extent* by clicking .
- You can switch between the previous and the next zoom level through the *Go Back* button  and the *Go Forward* one .
- The *Switch to Full Screen*  button allows to have a full screen map.

Basemap Switcher

By default, GeoNode allows to enrich maps with many world backgrounds:

- *OpenStreetMap*
- *OpenTopoMap*
- *Sentinel-2-cloudless*

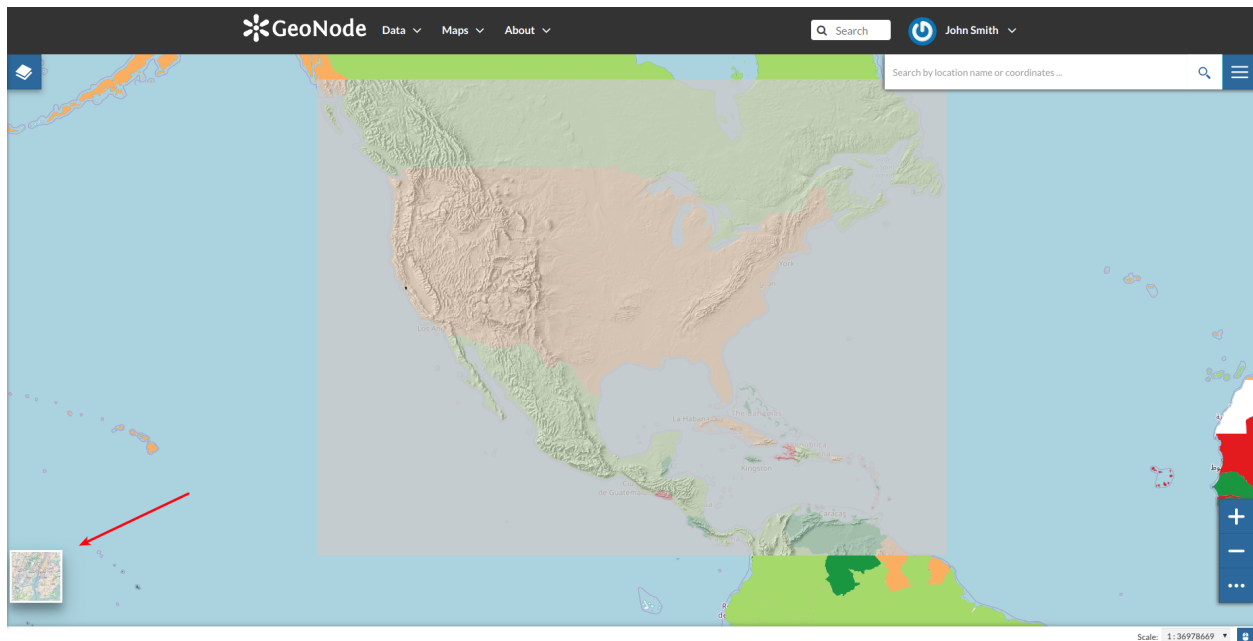



Fig. 210: *The Basemap Switcher Tool*

You can also decide to have an *Empty Background*.

Fig. 211: *Switching the Basemap*

Footer Tools

At the bottom of the map, the *Footer* shows you the *Scale* of the map and allows you to change it.

The  button allows you to see the pointer *Coordinates* and to change the Coordinates Reference System (CRS), WGS 84 by default.

1.10.7 Publishing Data

In GeoNode, each resource can be published in order to share it with other people. Once a *Map* has been published you can embed it in your web pages, your blog or your web site.

An easy way to accomplish that is to use an `iframe`. See the following steps:

- Open the *Map Information* page and copy the *URL*
- Add “/embed” to the URL so that it will be like this “`http://master.demo.geonode.org/maps/11/embed`”
- Use this URL inside an html `iframe` as `src` value

```
<iframe
  style="border: none;" height="400" width="600"
  src="http://master.demo.geonode.org/maps/11/embed"
></iframe>
```

- Put this html block of code inside your web pages to display the map.

Saving an html file with this code you can test your map on your pc, look at the following picture.

As you can see, some basic functionalities will be available to the user: the *Table of Contents (TOC)*, the *Basemap Switcher*, the *Sidebar Tools* and the *Options Menu Tools*.

1.10.8 Using GeoNode with Other Applications

Your GeoNode project is based on core components which are interoperable and as such, it is straightforward for you to integrate with external applications and services. This section will walk you through how to connect to your GeoNode instance from other applications and how to integrate other services into your GeoNode project. When complete, you should have a good idea about the possibilities for integration, and have basic knowledge about how to accomplish it. You may find it necessary to dive deeper into how to do more complex integration in order to accomplish your goals, but you should feel comfortable with the basics, and feel confident reaching out to the wider GeoNode community for help.

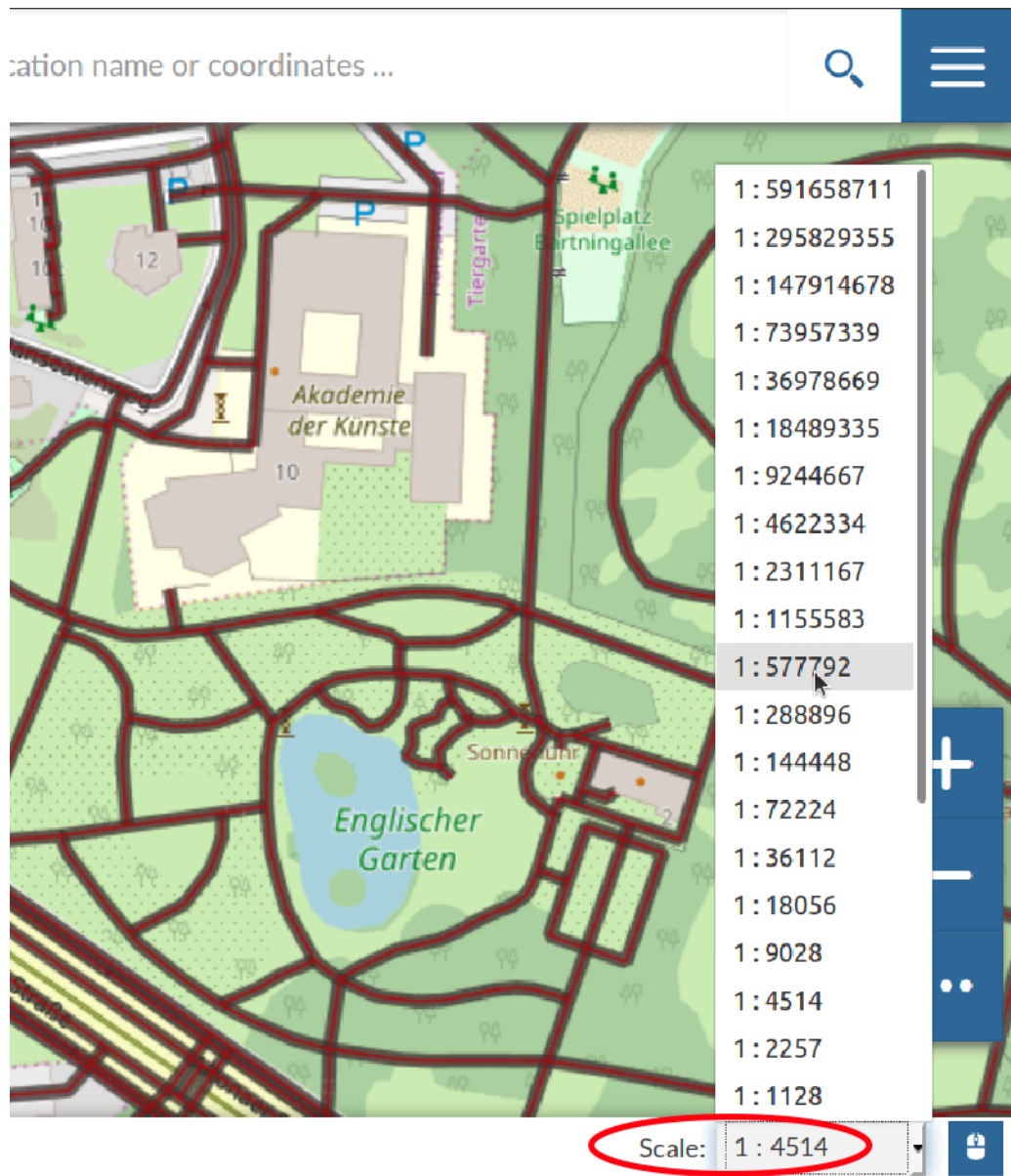


Fig. 212: The Map Scale

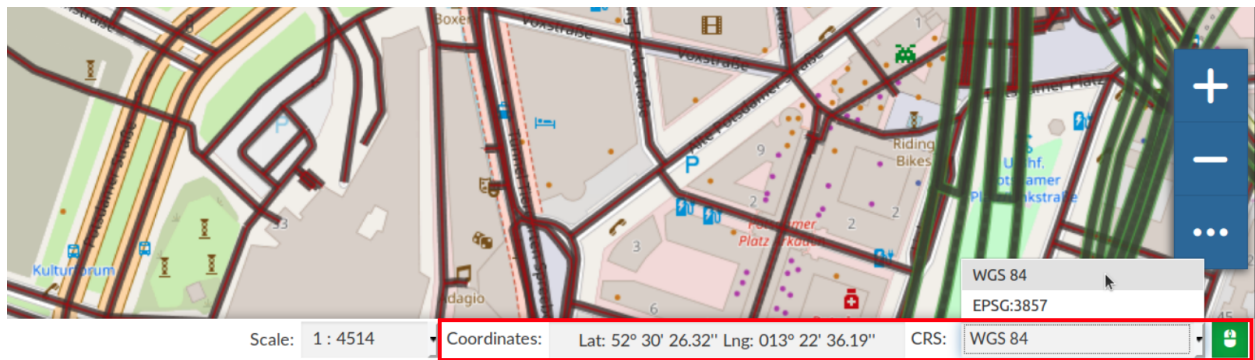


Fig. 213: The Pointer Coordinates and the CRS

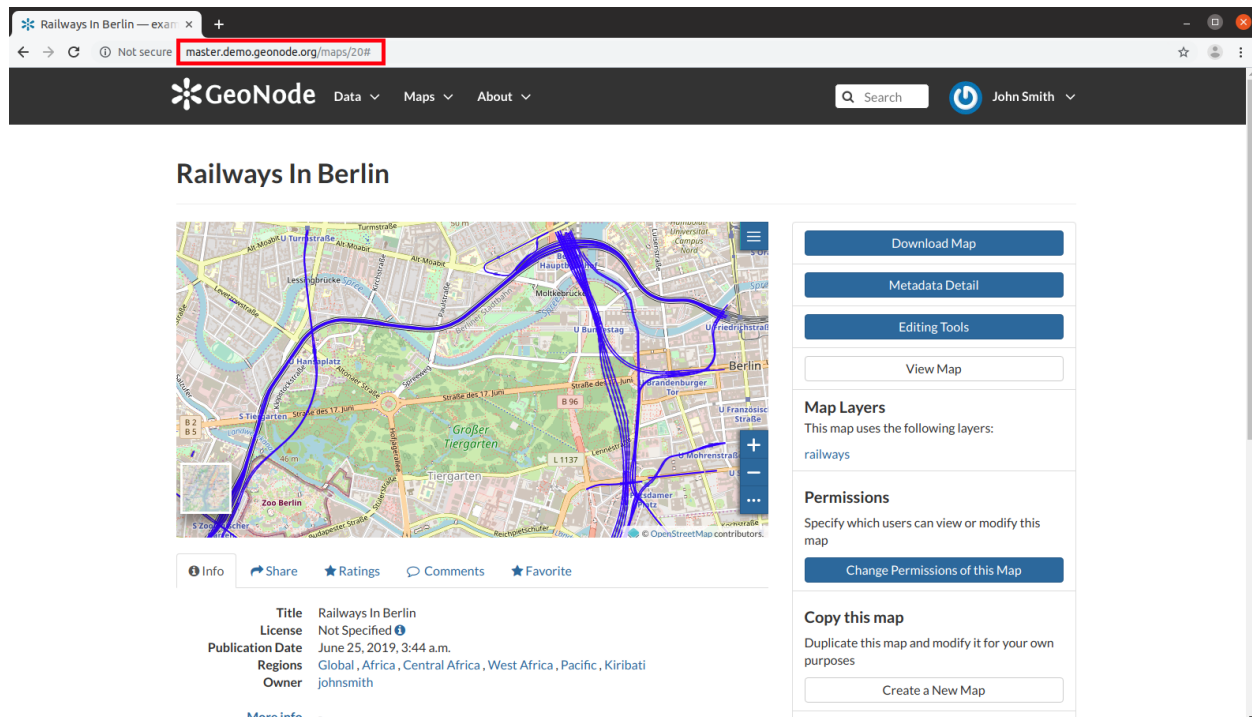


Fig. 214: The Map Information Page URL

1.11 GeoNode Core

1.11.1 Overview

The following steps will guide you to a fresh setup of GeoNode. All guides will first install and configure the system to run it in `DEBUG` mode (also known as `DEVELOPMENT` mode) and then by configuring an `HTTPD` server to serve GeoNode through the standard `HTTP` (80) port.

Those guides **are not** meant to be used on a production system. There will be dedicated chapters that will show you some *hints* to optimize GeoNode for a production-ready machine. In any case, we strongly suggest to task an experienced *DevOp* or *System Administrator* before exposing your server to the `WEB`.

1.11.2 Ubuntu 18.04

This part of the documentation describes the complete setup process for GeoNode on an Ubuntu 18.04 64-bit clean environment (Desktop or Server). All examples use shell commands that you must enter on a local terminal or a remote shell. - If you have a graphical desktop environment you can open the terminal application after login; - if you are working on a remote server the provider or sysadmin should have given you access through an `ssh` client.

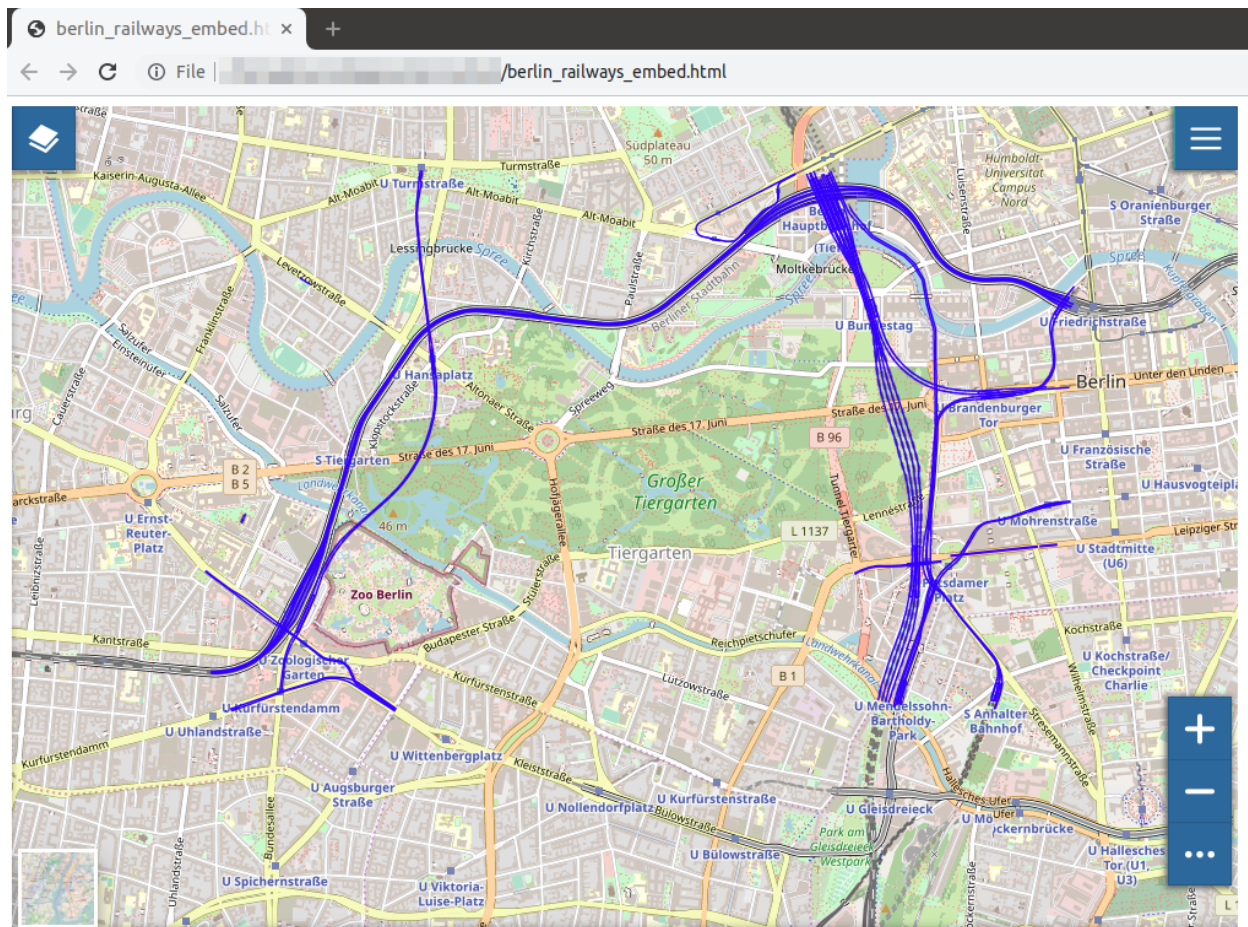


Fig. 215: *The Embedded Map*

Install the dependencies

In this section, we are going to install all the basic packages and tools needed for a complete GeoNode installation. To follow this guide, a basic knowledge about Ubuntu Server configuration and working with a shell is required. This guide uses `vim` as the editor; feel free to use `nano`, `gedit` or others.

Upgrade system packages

Check that your system is already up-to-date with the repository running the following commands:

```
sudo apt update
sudo apt upgrade
```

Packages Installation

Note: You don't need to install the **system packages** if you want to run the project using Docker

We will use **example.org** as fictitious Domain Name.

First, we are going to install all the **system packages** needed for the GeoNode setup. Login to the target machine and execute the following commands:

```
# Install packages from GeoNode core
sudo apt install -y gdal-bin
sudo apt install -y python3-pip python3-dev python3-virtualenv python3-venv
↳virtualenvwrapper
sudo apt install -y libxml2 libxml2-dev gettext
sudo apt install -y libxslt1-dev libjpeg-dev libpng-dev libpq-dev libgdal-dev
↳libgdal20
sudo apt install -y software-properties-common build-essential
sudo apt install -y git unzip gcc zlib1g-dev libgeos-dev libproj-dev
sudo apt install -y sqlite3 spatialite-bin libsqlite3-mod-spatialite

# Install Openjdk
sudo -i apt update
sudo apt install openjdk-8-jdk-headless default-jdk-headless -y
sudo update-java-alternatives --jre-headless --jre --set java-1.8.0-openjdk-amd64

# Install VIM
sudo apt install -y vim

sudo apt update -y
sudo apt upgrade -y
sudo apt autoremove -y
sudo apt autoclean -y
sudo apt purge -y
sudo apt clean -y
```

Create a Dedicated User

In the following steps a User named geonode is created (if needed) and used: to run installation commands the user must be in the sudo group.

Create User geonode **if not present**:

```
# Follow the prompts to set the new user's information.
# It is fine to accept the defaults to leave all of this information blank.
sudo adduser geonode

# The following command adds the user geonode to group sudo
sudo usermod -aG sudo geonode

# make sure the newly created user is allowed to login by ssh
# (out of the scope of this documentation) and switch to User geonode
su geonode
```

GeoNode Installation

This is the most basic installation of GeoNode. It won't use any external server like Apache Tomcat, PostgreSQL or HTTPD.

It will run locally against a file-system based SQLite database.

First of all we need to prepare a new Python Virtual Environment

Since geonode needs a large number of different python libraries and packages, its recommended to use a python virtual environment to avoid conflicts on dependencies with system wide python packages and other installed software. See also documentation of [Virtualenvwrapper](#) package for more information

```
# Create the GeoNode Virtual Environment (first time only)
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 geonode
```

At this point your command prompt shows a (geonode) prefix, this indicates that your virtualenv is active.

Note: The next time you need to access the Virtual Environment just run

```
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
workon geonode
```

Note: In order to save permanently the virtualenvwrapper environment

```
nano ~/.bashrc

# Write to the bottom of the file the following lines
export WORKON_HOME=/home/geonode/.virtualenvs
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

```
# Let's create the GeoNode core base folder and clone it
sudo mkdir -p /opt/geonode/
```

(continues on next page)

(continued from previous page)

```

sudo usermod -a -G www-data geonode
sudo chown -Rf geonode:www-data /opt/geonode/
sudo chmod -Rf 775 /opt/geonode/

# Clone the GeoNode source code on /opt/geonode
cd /opt
git clone https://github.com/GeoNode/geonode.git geonode

# Install the Python packages
cd /opt/geonode
pip install -r requirements.txt --upgrade --no-cache --no-cache-dir
pip install -e . --upgrade

# Install GDAL Utilities for Python
pip install pygdal=="gdal-config --version`.*"

```

Run GeoNode for the first time in DEBUG Mode

Warning: Be sure you have successfully completed all the steps of the section *Install the dependencies*.

This command will run both GeoNode and GeoServer locally after having prepared the SQLite database. The server will start in DEBUG (or DEVELOPMENT) mode, and it will start the following services:

1. GeoNode on `http://localhost:8000/`
2. GeoServer on `http://localhost:8080/geoserver/`

This modality is beneficial to debug issues and/or develop new features, but it cannot be used on a production system.

```

# Prepare the GeoNode SQLite database (the first time only)
paver setup
paver sync

```

Note: In case you want to start again from a clean situation, just run

```
paver reset_hard
```

Warning: This will blow up completely your `local_settings`, delete the SQLite database and remove the GeoServer data dir.

```

# Run the server in DEBUG mode
paver start

```

Once the server has finished the initialization and prints on the console the sentence `GeoNode is now available.`, you can open a browser and go to:

```
http://localhost:8000/
```

Sign-in with:

```
user: admin
password: admin
```

Postgis database Setup

Warning: Be sure you have successfully completed all the steps of the section *Install the dependencies*.

In this section, we are going to setup users and databases for GeoNode in PostgreSQL.

Install and Configure the PostgreSQL Database System

In this section we are going to install the PostgreSQL packages along with the PostGIS extension. Those steps must be done **only** if you don't have the DB already installed on your system.

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg_
↳main" >> /etc/apt/sources.list.d/pgdg.list'
sudo wget --no-check-certificate --quiet -O - https://www.postgresql.org/media/keys/
↳ACCC4CF8.asc | sudo apt-key add -

sudo apt update
sudo apt install -y postgresql-11 postgresql-11-postgis-2.5 postgresql-11-postgis-2.5-
↳scripts postgresql-contrib-11 postgresql-client-11
```

We now must create two databases, geonode and geonode_data, belonging to the role geonode.

Note: This is our default configuration. You can use any database or role you need. The connection parameters must be correctly configured on settings, as we will see later in this section.

Databases and Permissions

First, create the geonode user. GeoNode is going to use this user to access the database

```
sudo -u postgres createuser -P geonode
```

You will be prompted asked to set a password for the user. Enter geonode as password.

Warning: This is a sample password used for the sake of simplicity. This password is very **weak** and should be changed in a production environment.

Create database geonode and geonode_data with owner geonode

```
sudo -u postgres createdb -O geonode geonode
sudo -u postgres createdb -O geonode geonode_data
```

Next let's create PostGIS extensions


```
sudo -u postgres psql -d geonode_data -c 'CREATE EXTENSION postgis;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN_
↳SCHEMA public TO geonode;'
```

Final step is to change user access policies for local connections in the file `pg_hba.conf`

```
sudo vim /etc/postgresql/11/main/pg_hba.conf
```

Scroll down to the bottom of the document. We only need to edit one line.

```
# "local" is for Unix domain socket connections only
# local  all  all  peer
local  all  all  trust
```

Warning: If your PostgreSQL database resides on a **separate/remote machine**, you'll have to **allow** remote access to the databases in the `/etc/postgresql/11/main/pg_hba.conf` to the `geonode` user and tell PostgreSQL to **accept** non-local connections in your `/etc/postgresql/11/main/postgresql.conf` file

Restart PostgreSQL to make the change effective.

```
sudo service postgresql restart
```

PostgreSQL is now ready. To test the configuration, try to connect to the `geonode` database as `geonode` role.

```
psql -U geonode geonode
\q
```

Install GeoServer

When running the command `paver start`, as we have seen before, the script runs automatically a Jetty Servlet Java container running GeoServer with the default settings.

Warning: Before executing the next steps, be sure GeoNode and GeoServer `paver` services have been stopped. In order to do that

```
workon geonode
cd /opt/geonode/
paver stop
```

This is not the optimal way to run GeoServer. This is a fundamental component of GeoNode and we must be sure it is running on a stable and reliable manner.

In this section, we are going to install the Apache Tomcat 8 Servlet Java container, which will be started by default on the internal port 8080.

We will also perform several optimizations to:

1. Correctly setup the Java VM Options, like the available heap memory and the garbage collector options.
2. Externalize the GeoServer and GeoWebcache catalogs in order to allow further updates without the risk of deleting our datasets.

Note: This is still a basic setup of those components. More details will be provided on sections of the documentation concerning the hardening of the system in a production environment. Nevertheless, you will need to tweak a bit those settings accordingly with your current system. As an instance, if your machine does not have enough memory, you will need to lower down the initial amount of available heap memory. **Warnings** and **notes** will be placed below the statements that will require your attention.

```
# Install Openjdk
sudo -i apt update
sudo apt install openjdk-8-jdk-headless default-jdk-headless -y
sudo update-java-alternatives --jre-headless --jre --set java-1.8.0-openjdk-amd64

# Check Java version
java -version
openjdk version "1.8.0_212"
OpenJDK Runtime Environment (build 1.8.0_212-8u212-b03-0ubuntu1.18.04.1-b03)
OpenJDK 64-Bit Server VM (build 25.212-b03, mixed mode)

# Install Apache Tomcat 8
sudo wget http://www-us.apache.org/dist/tomcat/tomcat-8/v8.5.53/bin/apache-tomcat-8.5.
↳53.tar.gz
sudo tar xzf apache-tomcat-8.5.53.tar.gz
sudo mv apache-tomcat-8.5.53 /usr/local/apache-tomcat8
sudo useradd -m -U -s /bin/false tomcat
sudo usermod -a -G www-data tomcat
sudo sed -i -e 's/xom-\\*\\.jar/xom-\\*\\.jar,bcprov\\*\\.jar/g' /usr/local/apache-tomcat8/
↳conf/catalina.properties

export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
echo 'JAVA_HOME=$JAVA_HOME' | sudo tee --append /usr/local/apache-tomcat8/bin/setenv.
↳sh

# Add Tomcat user to www-data group !important!
sudo usermod -a -G www-data tomcat

sudo sh -c 'chmod +x /usr/local/apache-tomcat8/bin/*.sh'
sudo chown -Rf tomcat:www-data /usr/local/apache-tomcat8
```

Let's create a system service to manage tomcat startup

```
sudo vim /etc/systemd/system/tomcat.service
```

```
[Unit]
Description=Tomcat 8.5 servlet container
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME=/usr/lib/jvm/default-java"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"

Environment="CATALINA_BASE=/usr/local/apache-tomcat8"
```

(continues on next page)

(continued from previous page)

```
Environment="CATALINA_HOME=/usr/local/apache-tomcat8"
Environment="CATALINA_PID=/usr/local/apache-tomcat8/temp/tomcat.pid"

ExecStart=/usr/local/apache-tomcat8/bin/startup.sh
ExecStop=/usr/local/apache-tomcat8/bin/shutdown.sh

[Install]
WantedBy=multi-user.target
```

To test the service:

```
sudo systemctl daemon-reload
sudo systemctl restart tomcat
sudo systemctl status tomcat.service
```

To make it enabled by default

```
sudo systemctl enable tomcat
```

GeoServer Optimizations

Let's externalize the GEOSERVER_DATA_DIR and logs

```
# Create the target folders
sudo mkdir -p /opt/data
sudo chown -Rf geonode:www-data /opt/data
sudo chmod -Rf 775 /opt/data
sudo mkdir -p /opt/data/logs
sudo chown -Rf geonode:www-data /opt/data/logs
sudo chmod -Rf 775 /opt/data/logs

# Download and extract the default GEOSERVER_DATA_DIR
sudo wget --no-check-certificate https://build.geo-solutions.it/geonode/geoserver/
↳latest/data-2.16.2.zip
sudo unzip data-2.16.2.zip -d /opt/data/

sudo mv /opt/data/data/ /opt/data/geoserver_data
sudo chown -Rf tomcat:www-data /opt/data/geoserver_data
sudo chmod -Rf 775 /opt/data/geoserver_data

sudo mkdir -p /opt/data/geoserver_logs
sudo chown -Rf tomcat:www-data /opt/data/geoserver_logs
sudo chmod -Rf 775 /opt/data/geoserver_logs

sudo mkdir -p /opt/data/gwc_cache_dir
sudo chown -Rf tomcat:www-data /opt/data/gwc_cache_dir
sudo chmod -Rf 775 /opt/data/gwc_cache_dir

# Download and install GeoServer
sudo wget --no-check-certificate https://build.geo-solutions.it/geonode/geoserver/
↳latest/geoserver-2.16.2.war
sudo mv geoserver-2.16.2.war /usr/local/apache-tomcat8/webapps/geoserver.war
```

Let's now configure the JAVA_OPTS, i.e. the parameters to run the Servlet Container, like heap memory, garbage collector and so on.

```
sudo sed -i -e "s/JAVA_OPTS=/#JAVA_OPTS=/g" /usr/local/apache-tomcat8/bin/setenv.sh

echo 'GEOSERVER_DATA_DIR="/opt/data/geoserver_data"' | sudo tee --append /usr/local/
↪apache-tomcat8/bin/setenv.sh
echo 'GEOSERVER_LOG_LOCATION="/opt/data/geoserver_logs/geoserver.log"' | sudo tee --
↪append /usr/local/apache-tomcat8/bin/setenv.sh
echo 'GEOWEBCACHE_CACHE_DIR="/opt/data/gwc_cache_dir"' | sudo tee --append /usr/local/
↪apache-tomcat8/bin/setenv.sh
echo 'GEOFENCE_DIR="$GEOSERVER_DATA_DIR/geofence"' | sudo tee --append /usr/local/
↪apache-tomcat8/bin/setenv.sh
echo 'TIMEZONE="UTC"' | sudo tee --append /usr/local/apache-tomcat8/bin/setenv.sh

echo 'JAVA_OPTS="-server -Djava.awt.headless=true -Dorg.geotools.shapefile.
↪datetime=true -XX:+UseParallelGC -XX:ParallelGCThreads=4 -Dfile.encoding=UTF8 -
↪Duser.timezone=$TIMEZONE -Xms512m -Xmx4096m -Djavax.servlet.request.encoding=UTF-8 -
↪Djavax.servlet.response.encoding=UTF-8 -DGEOSERVER_CSRF_DISABLED=true -DGEOSERVER_
↪DATA_DIR=$GEOSERVER_DATA_DIR -Dgeofence.dir=$GEOFENCE_DIR -DGEOSERVER_LOG_LOCATION=
↪$GEOSERVER_LOG_LOCATION -DGEOWEBCACHE_CACHE_DIR=$GEOWEBCACHE_CACHE_DIR"' | sudo tee
↪--append /usr/local/apache-tomcat8/bin/setenv.sh
```

Note: After the execution of the above statements, you should be able to see the new options written at the bottom of the file `/usr/local/apache-tomcat8/bin/setenv.sh`.

```
...
# If you run Tomcat on port numbers that are all higher than 1023, then you
# do not need authbind. It is used for binding Tomcat to lower port numbers.
# (yes/no, default: no)
#AUTHBIND=no
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
GEOSERVER_DATA_DIR="/opt/data/geoserver_data"
GEOSERVER_LOG_LOCATION="/opt/data/geoserver_logs/geoserver.log"
GEOWEBCACHE_CACHE_DIR="/opt/data/gwc_cache_dir"
GEOFENCE_DIR="$GEOSERVER_DATA_DIR/geofence"
TIMEZONE="UTC"
JAVA_OPTS="-server -Djava.awt.headless=true -Dorg.geotools.shapefile.datetime=true -
↪XX:+UseParallelGC -XX:ParallelGCThreads=4 -Dfile.encoding=UTF8 -Duser.timezone=
↪$TIMEZONE -Xms512m -Xmx4096m -Djavax.servlet.request.encoding=UTF-8 -Djavax.servlet.
↪response.encoding=UTF-8 -DGEOSERVER_CSRF_DISABLED=true -DGEOSERVER_DATA_DIR=
↪$GEOSERVER_DATA_DIR -Dgeofence.dir=$GEOFENCE_DIR -DGEOSERVER_LOG_LOCATION=
↪$GEOSERVER_LOG_LOCATION -DGEOWEBCACHE_CACHE_DIR=$GEOWEBCACHE_CACHE_DIR"
```

Those options could be updated or changed manually at any time, accordingly to your needs.

Warning: The default options we are going to add to the Servlet Container, assume you can reserve at least 4GB of RAM to GeoServer (see the option `-Xmx4096m`). You must be sure your machine has enough memory to run both GeoServer and GeoNode, which in this case means at least 4GB for GeoServer plus at least 2GB for GeoNode. A total of at least 6GB of RAM available on your machine. If you don't have enough RAM available, you can lower down the values `-Xms512m` `-Xmx4096m`. Consider that with less RAM available, the performances of your services will be highly impacted.

In order to make the changes effective, you'll need to restart the Servlet Container.

```
# Restart the server
sudo systemctl daemon-reload
sudo systemctl restart tomcat
sudo systemctl status tomcat.service

# Follow the startup logs
sudo tail -F -n 300 /opt/data/geoserver_logs/geoserver.log
```

If you can see on the logs something similar to this, without errors

```
...
2019-05-31 10:06:34,190 INFO [geoserver.wps] - Found 5 bindable processes in_
↳GeoServer specific processes
2019-05-31 10:06:34,281 INFO [geoserver.wps] - Found 89 bindable processes in_
↳Deprecated processes
2019-05-31 10:06:34,298 INFO [geoserver.wps] - Found 31 bindable processes in Vector_
↳processes
2019-05-31 10:06:34,307 INFO [geoserver.wps] - Found 48 bindable processes in_
↳Geometry processes
2019-05-31 10:06:34,307 INFO [geoserver.wps] - Found 1 bindable processes in_
↳PolygonLabelProcess
2019-05-31 10:06:34,311 INFO [geoserver.wps] - Blacklisting process_
↳ras:ConvolveCoverage as the input kernel of type class javax.media.jai.KernelJAI_
↳cannot be handled
2019-05-31 10:06:34,319 INFO [geoserver.wps] - Blacklisting process_
↳ras:RasterZonalStatistics2 as the input zones of type class java.lang.Object cannot_
↳be handled
2019-05-31 10:06:34,320 INFO [geoserver.wps] - Blacklisting process_
↳ras:RasterZonalStatistics2 as the input nodata of type class it.geosolutions.jaiext.
↳range.Range cannot be handled
2019-05-31 10:06:34,320 INFO [geoserver.wps] - Blacklisting process_
↳ras:RasterZonalStatistics2 as the input rangeData of type class java.lang.Object_
↳cannot be handled
2019-05-31 10:06:34,320 INFO [geoserver.wps] - Blacklisting process_
↳ras:RasterZonalStatistics2 as the output zonal statistics of type interface java.
↳util.List cannot be handled
2019-05-31 10:06:34,321 INFO [geoserver.wps] - Found 18 bindable processes in Raster_
↳processes
2019-05-31 10:06:34,917 INFO [ows.OWSHandlerMapping] - Mapped URL path [/TestWfsPost]_
↳onto handler 'wfsTestServlet'
2019-05-31 10:06:34,918 INFO [ows.OWSHandlerMapping] - Mapped URL path [/wfs/*] onto_
↳handler 'dispatcher'
2019-05-31 10:06:34,918 INFO [ows.OWSHandlerMapping] - Mapped URL path [/wfs] onto_
↳handler 'dispatcher'
2019-05-31 10:06:42,237 INFO [geoserver.security] - Start reloading user/groups for_
↳service named default
2019-05-31 10:06:42,241 INFO [geoserver.security] - Reloading user/groups successful_
↳for service named default
2019-05-31 10:06:42,357 WARN [auth.GeoFenceAuthenticationProvider] - INIT FROM CONFIG
2019-05-31 10:06:42,494 INFO [geoserver.security] - AuthenticationCache Initialized_
↳with 1000 Max Entries, 300 seconds idle time, 600 seconds time to live and 3_
↳concurrency level
2019-05-31 10:06:42,495 INFO [geoserver.security] - AuthenticationCache Eviction Task_
↳created to run every 600 seconds
2019-05-31 10:06:42,506 INFO [config.GeoserverXMLResourceProvider] - Found_
↳configuration file in /opt/data/gwc_cache_dir
2019-05-31 10:06:42,516 INFO [config.GeoserverXMLResourceProvider] - Found_
↳configuration file in /opt/data/gwc_cache_dir
```

(continues on next page)

(continued from previous page)

```
2019-05-31 10:06:42,542 INFO [config.XMLConfiguration] - Wrote configuration to /opt/
↳data/gwc_cache_dir
2019-05-31 10:06:42,547 INFO [geoserver.importer] - Enabling import store: memory
```

Your GeoServer should be up and running at

```
http://localhost:8080/geoserver/
```

Warning: In case of errors or the file `geoserver.log` is not created, check the Catalina logs in order to try to understand what's happened.

```
sudo less /usr/local/apache-tomcat8/logs/catalina.out
```

It is possible to test the new running GeoServer with the GeoNode paver service (DEBUG mode). To do that

```
workon geonode
cd /opt/geonode/
paver start_django
```

Warning: The `paver reset` command from now on **won't** clean up GeoServer and its catalog anymore. Therefore, every data uploaded during those tests will remain on GeoServer even if GeoNode will be reset.

Web Server

Until now we have seen how to start GeoNode in DEBUG mode from the command line, through the `paver` utilities. This is of course not the best way to start it. Moreover you will need a dedicated HTTPD server running on port 80 if you would like to expose your server to the world.

In this section we will see:

1. How to configure NGINX HTTPD Server to host GeoNode and GeoServer. In the initial setup we will still run the services on `http://localhost`
2. Update the settings in order to link GeoNode and GeoServer to the PostgreSQL Database.
3. Update the settings in order to update GeoNode and GeoServer services running on a **public IP** or **hostname**.
4. Install and enable HTTPS secured connection through the Let's Encrypt provider.

Install and configure NGINX

Warning: Before executing the next steps, be sure GeoNode paver services have been stopped. To do that

```
workon geonode
cd /opt/geonode/
paver stop_django
```

```
# Install the services
sudo apt install -y nginx uwsgi uwsgi-plugin-python3
```

Serving {"geonode", "geoserver"} via NGINX

```
# Create the GeoNode UWSGI config
sudo vim /etc/uwsgi/apps-available/geonode.ini
```

```
[uwsgi]
socket = 0.0.0.0:8000
uid = geonode
gid = www-data

plugins = python3
virtualenv = /home/geonode/.virtualenvs/geonode
env = DEBUG=False
env = DJANGO_SETTINGS_MODULE=geonode.settings
env = SECRET_KEY='Rand0m%3cr3tK3y'
env = SITE_HOST_NAME=localhost
env = SITEURL=http://localhost/
env = LOCKDOWN_GEOCODE=False
env = SESSION_EXPIRED_CONTROL_ENABLED=True
env = FORCE_SCRIPT_NAME=
env = EMAIL_ENABLE=False
env = DJANGO_EMAIL_HOST_USER=
env = DJANGO_EMAIL_HOST_PASSWORD=
env = DJANGO_EMAIL_HOST=localhost
env = DJANGO_EMAIL_PORT=25
env = DJANGO_EMAIL_USE_TLS=False
env = DEFAULT_FROM_EMAIL=GeoNode <no-reply@localhost>
env = MONITORING_ENABLED=True
env = GEOSERVER_PUBLIC_HOST=localhost
env = GEOSERVER_PUBLIC_PORT=
env = GEOSERVER_ADMIN_PASSWORD=geoserver
env = GEOSERVER_LOCATION=http://localhost/geoserver/
env = GEOSERVER_PUBLIC_LOCATION=http://localhost/geoserver/
env = GEOSERVER_WEB_UI_LOCATION=http://localhost/geoserver/
env = RESOURCE_PUBLISHING=False
env = ADMIN_MODERATE_UPLOADS=False
env = GROUP_PRIVATE_RESOURCES=False
env = GROUP_MANDATORY_RESOURCES=False
env = OGC_REQUEST_TIMEOUT=60
env = OGC_REQUEST_MAX_RETRIES=3
env = OGC_REQUEST_POOL_MAXSIZE=100
env = OGC_REQUEST_POOL_CONNECTIONS=100
env = EXIF_ENABLED=True
env = CREATE_LAYER=False
env = FAVORITE_ENABLED=True

chdir = /opt/geonode
module = geonode.wsgi:application

processes = 4
threads = 2
enable-threads = true
```

(continues on next page)

(continued from previous page)

```

master = true

# logging
# path to where uwsgi logs will be saved
logto = /opt/data/logs/geonode.log
daemonize = /opt/data/logs/geonode.log
touch-reload = /opt/geonode/geonode/wsgi.py
buffer-size = 32768
max-requests = 500
harakiri = 300 # respawn processes taking more than 5 minutes (300 seconds)
max-requests = 500 # respawn processes after serving 5000 requests
# limit-as = 1024 # avoid Errno 12 cannot allocate memory
harakiri-verbose = true
vacuum = true
thunder-lock = true

```

```

# Enable the GeoNode UWSGI config
sudo ln -s /etc/uwsgi/apps-available/geonode.ini /etc/uwsgi/apps-enabled/geonode.ini

# Restart UWSGI Service
sudo service uwsgi restart

```

```

# Backup the original NGINX config
sudo mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.orig

# Create the GeoNode Default NGINX config
sudo vim /etc/nginx/nginx.conf

```

```

# Make sure your nginx.config matches the following one
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {
    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    # server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

    include /etc/nginx/mime.types;

```

(continues on next page)

(continued from previous page)

```

default_type application/octet-stream;

##
# SSL Settings
##

ssl_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE
ssl_prefer_server_ciphers on;

##
# Logging Settings
##

access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log;

##
# Gzip Settings
##

gzip on;
gzip_vary on;
gzip_proxied any;
gzip_http_version 1.1;
gzip_disable "MSIE [1-6]\.";
gzip_buffers 16 8k;
gzip_min_length 1100;
gzip_comp_level 6;
gzip_types video/mp4 text/plain application/javascript application/x-javascript_
↪text/javascript text/xml text/css image/jpeg;

##
# Virtual Host Configs
##

include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
}

```

```

# Remove the Default NGINX config
sudo rm /etc/nginx/sites-enabled/default

# Create the GeoNode App NGINX config
sudo vim /etc/nginx/sites-available/geonode

```

```

uwsgi_intercept_errors on;

upstream geoserver_proxy {
    server localhost:8080;
}

# Expires map
map $sent_http_content_type $expires {
    default                off;
    text/html               epoch;
    text/css                max;

```

(continues on next page)

(continued from previous page)

```
application/javascript      max;
~image/                     max;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    charset utf-8;

    etag on;
    expires $expires;
    proxy_read_timeout 600s;
    # set client body size to 2M #
    client_max_body_size 50000M;

    location / {
        etag off;
        uwsgi_pass 127.0.0.1:8000;
        uwsgi_read_timeout 600s;
        include uwsgi_params;
    }

    location /static/ {
        alias /opt/geonode/geonode/static_root/;
    }

    location /uploaded/ {
        alias /opt/geonode/geonode/uploaded/;
    }

    location /geoserver {
        proxy_pass http://geoserver_proxy;
        include proxy_params;
    }
}
```

```
# Enable GeoNode NGINX config
sudo ln -s /etc/nginx/sites-available/geonode /etc/nginx/sites-enabled/geonode

# Restart the services
sudo systemctl restart tomcat
sudo service nginx restart
```

Refresh GeoNode static data

```
workon geonode
cd /opt/geonode
python manage.py collectstatic --no-input
```

Refresh GeoNode and GeoServer OAuth2 settings


```

workon geonode
cd /opt/geonode

# This must be done the first time only
sudo cp package/support/geonode.binary /usr/bin/geonode
sudo cp package/support/geonode.updateip /usr/bin/geonode_updateip
sudo chmod +x /usr/bin/geonode
sudo chmod +x /usr/bin/geonode_updateip
pip install -e git+https://github.com/justquick/django-activity-stream.git#egg=django-
↳activity-stream

# Update the GeoNode ip or hostname
sudo PYTHONWARNINGS=ignore VIRTUAL_ENV=$VIRTUAL_ENV DJANGO_SETTINGS_MODULE=geonode.
↳settings GEONODE_ETC=/opt/geonode GEOSERVER_DATA_DIR=/opt/data/geoserver_data_
↳TOMCAT_SERVICE="service tomcat" APACHE_SERVICE="service nginx" geonode_updateip -p_
↳localhost

```

The GeoNode service should now run on `http://localhost/`

The GeoServer service should now run on `http://localhost/geoserver/`

You should be able to login with the default user admin (pwd admin) and upload your layers.

Update the settings in order to use the PostgreSQL Database

Warning: Make sure you already installed and configured the Database as explained in the previous sections.

```

workon geonode
cd /opt/geonode

cp geonode/local_settings.py.geoserver.sample geonode/local_settings.py

# In case you want to change the DB password, run the following
# sudo sed -i -e "s/'PASSWORD': 'geonode','/PASSWORD': '<your_db_role_password>','/g"_
↳geonode/local_settings.py

# Stop Tomcat
sudo systemctl stop tomcat

# Initialize GeoNode
DJANGO_SETTINGS_MODULE=geonode.local_settings paver reset
DJANGO_SETTINGS_MODULE=geonode.local_settings paver setup
DJANGO_SETTINGS_MODULE=geonode.local_settings paver sync
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py collectstatic --noinput

```

Before finalizing the configuration we will need to update the UWSGI settings

```
sudo vim /etc/uwsgi/apps-enabled/geonode.ini
```

Change `geonode.settings` to `geonode.local_settings`

```

:%s/geonode.settings/geonode.local_settings/g
:wq

```

Restart UWSGI and update OAuth2 by using the new `geonode.local_settings`

Warning: **!IMPORTANT!** In the statement below make sure to use `DJANGO_SETTINGS_MODULE=geonode.local_settings`

```
# Restart UWSGI
sudo service uwsgi restart

# Update the GeoNode ip or hostname
sudo PYTHONWARNINGS=ignore VIRTUAL_ENV=$VIRTUAL_ENV DJANGO_SETTINGS_MODULE=geonode.
↪local_settings GEONODE_ETC=/opt/geonode GEOSERVER_DATA_DIR=/opt/data/geoserver_data_
↪TOMCAT_SERVICE="service tomcat" APACHE_SERVICE="service nginx" geonode_updateip -p_
↪localhost
```

Update the settings in order to update GeoNode and GeoServer services running on a public IP or hostname

Warning: Before exposing your services to the Internet, **make sure** your system is **hardened** and **secure enough**. See the specific documentation section for more details.

Let's say you want to run your services on a public IP or domain, e.g. `www.example.org`. You will need to slightly update your services in order to reflect the new server name.

In particular the steps to do are:

1. Update NGINX configuration in order to serve the new domain name.

```
sudo vim /etc/nginx/sites-enabled/geonode

# Update the 'server_name' directive
server_name example.org www.example.org;

# Restart the service
sudo service nginx restart
```

2. Update UWSGI configuration in order to serve the new domain name.

```
sudo vim /etc/uwsgi/apps-enabled/geonode.ini

# Change everywhere 'localhost' to the new hostname
%s/localhost/www.example.org/g

# Restart the service
sudo service uwsgi restart
```

3. Update OAuth2 configuration in order to hit the new hostname.

```
workon geonode
cd /opt/geonode

# Update the GeoNode ip or hostname
sudo PYTHONWARNINGS=ignore VIRTUAL_ENV=$VIRTUAL_ENV DJANGO_SETTINGS_
↪MODULE=geonode.local_settings GEONODE_ETC=/opt/geonode GEOSERVER_DATA_DIR=/
↪opt/data/geoserver_data TOMCAT_SERVICE="service tomcat" APACHE_SERVICE=
↪"service nginx" geonode_updateip -l localhost -p www.example.org
```

4. Update the existing GeoNode links in order to hit the new hostname.

```
workon geonode
cd /opt/geonode

# Update the GeoNode ip or hostname
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py migrate_
↪baseurl --source-address=http://localhost --target-address=http://www.
↪example.org
```

Install and enable HTTPS secured connection through the Let's Encrypt provider

```
# Install Let's Encrypt Certbot
sudo add-apt-repository ppa:certbot/certbot
sudo apt update -y; sudo apt install python-certbot-nginx -y

# Reload NGINX config and make sure the firewall denies access to HTTP
sudo systemctl reload nginx
sudo ufw allow 'Nginx Full'
sudo ufw delete allow 'Nginx HTTP'

# Create and dump the Let's Encrypt Certificates
sudo certbot --nginx -d example.org -d www.example.org
# ...choose the redirect option when asked for
```

1. Update the GeoNode **OAuth2** Redirect URIs accordingly.

From the GeoNode Admin Dashboard go to Home > Django/GeoNode OAuth Toolkit > Applications > GeoServer

2. Update the GeoServer Proxy Base URL accordingly.

From the GeoServer Admin GUI go to About & Status > Global

3. Update the GeoServer Role Base URL accordingly.

From the GeoServer Admin GUI go to Security > Users, Groups, Roles > geonode REST role service

4. Update the GeoServer OAuth2 Service Parameters accordingly.

From the GeoServer Admin GUI go to Security > Authentication > Authentication Filters > geonode-oauth2

5. Update the UWSGI configuration

```
sudo vim /etc/uwsgi/apps-enabled/geonode.ini

# Change everywhere 'http' to 'https'
%s/http/https/g

# Add three more 'env' variables to the configuration
env = SECURE_SSL_REDIRECT=True
env = SECURE_HSTS_INCLUDE_SUBDOMAINS=True
env = AVATAR_GRAVATAR_SSL=True

# Restart the service
sudo service uwsgi restart
```

Django administration

Home › Django/GeoNode OAuth Toolkit › Applications › GeoServer

Change application

Client id: 6Aa43yllpxN0RcxwXXM7XDPArEtFSSPjv3Y2t

User: 1000 Q admin

Redirect uris:


https://example.org/geoserver/
https://www.example.org/geoserver/

 ←

Allowed URIs list, space separated

Client type: ☒ Confidential ☐ Public

Fig. 216: Redirect URIs

 **GeoServer**

Global Settings

Settings that apply to all OGC services and control the internal behavior of GeoServer.

OGC Services

Service Settings

Proxy Base URL

https://www.example.org/geoserver

 ←

☐ Use headers for Proxy URL

☒ Enable global services

Service Request Settings

☐ Evaluate XML entities from remote servers (security risk)

- About & Status
 - Server Status
 - GeoServer Logs
 - Contact Information
 - About GeoServer
 - Process status
- Data
 - Layer Preview
 - Import Data
 - Workspaces
 - Storage

Fig. 217: Proxy Base URL

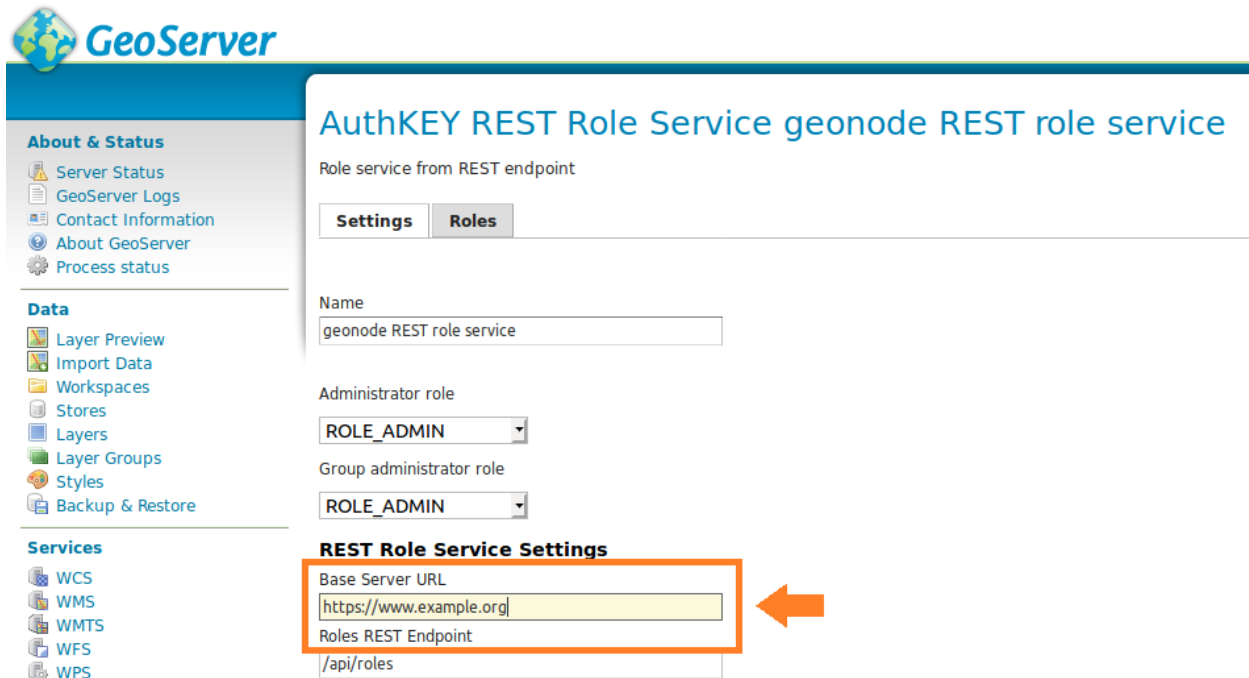


Fig. 218: Role Base URL

1.11.3 CentOS 7.0

- TODO

1.11.4 Docker

In this section we are going to list the passages needed to:

1. Install Docker and docker-compose packages on a Ubuntu host
2. Deploy a vanilla GeoNode 2.10 with Docker
 - a. Override the ENV variables to deploy on a public IP or domain
 - b. Access the django4geonode Docker image to update the code-base and/or change internal settings
 - c. Access the geoserver4geonode Docker image to update the GeoServer version
3. Passages to completely get rid of old Docker images and volumes (prune the environment completely)

Install the Docker and docker-compose packages on a Ubuntu host

Docker Setup (First time only)

```
sudo add-apt-repository universe
sudo apt-get update -y
sudo apt-get install -y git-core git-buildpackage debhelper devscripts
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-
↳ properties-common
```

(continues on next page)

Authentication using a GeoNode OAuth2 geonode-oauth2

Authenticates by looking up for a valid GeoNode OAuth2 access_token key sent as URL parameter

Name
geonode-oauth2

OAuth2 provider connection

☐ Enable Redirect Authentication EntryPoint ⓘ

Login Authentication EndPoint
/j_spring_oauth2_geonode_login

Logout Authentication EndPoint
/j_spring_oauth2_geonode_logout

☒ Force Access Token URI HTTPS Secured Protocol ⓘ

Access Token URI
https://www.example.org/o/token/ ⓘ

☒ Force User Authorization URI HTTPS Secured Protocol ⓘ

User Authorization URI
https://www.example.org/o/authorize/ ⓘ

Redirect URI
https://www.example.org/geoserver/ ⓘ

Check Token Endpoint URL
https://www.example.org/api/o/v4/tokeninfo/ ⓘ

Logout URI
https://www.example.org/account/logout/ ⓘ

Scopes
write ⓘ

Client ID
6Aa43yllpxN0RcxwXXM7XDPArEtFSSPjv3Y2mcDd ⓘ

Client Secret
RHlyGtz1cDO597MKLQmxKvxTogubAaL0Q7kstFb9UeJi ⓘ

Role source

Role service

Fig. 219: OAuth2 Service Parameters

```

[uwsgi]
socket = 0.0.0.0:8000
uid = geonode
gid = www-data

plugins = python
virtualenv = /home/geonode/Envs/geonode
env = DEBUG=False
env = DJANGO_SETTINGS_MODULE=geonode.local_settings
env = SECRET_KEY='Rand0m%3cr3tK3y'
env = SITE_HOST_NAME=www.example.org
env = SITEURL=https://www.example.org/
env = LOCKDOWN_GEOCODE=False
env = SESSION_EXPIRED_CONTROL_ENABLED=True
env = FORCE_SCRIPT_NAME=
env = EMAIL_ENABLE=False
env = DJANGO_EMAIL_HOST_USER=
env = DJANGO_EMAIL_HOST_PASSWORD=
env = DJANGO_EMAIL_HOST=www.example.org
env = DJANGO_EMAIL_PORT=25
env = DJANGO_EMAIL_USE_TLS=False
env = DEFAULT_FROM_EMAIL=GeoNode <no-reply@www.example.org>
env = MONITORING_ENABLED=True
env = GEOSERVER_PUBLIC_HOST=www.example.org
env = GEOSERVER_PUBLIC_PORT=
env = GEOSERVER_ADMIN_PASSWORD=geoserver
env = GEOSERVER_LOCATION=https://www.example.org/geoserver/
env = GEOSERVER_PUBLIC_LOCATION=https://www.example.org/geoserver/
env = GEOSERVER_WEB_UI_LOCATION=https://www.example.org/geoserver/
env = RESOURCE_PUBLISHING=False
env = ADMIN_MODERATE_UPLOADS=False
env = GROUP_PRIVATE_RESOURCES=False
env = GROUP_MANDATORY_RESOURCES=False
env = OGC_REQUEST_TIMEOUT=60
env = OGC_REQUEST_MAX_RETRIES=3
env = OGC_REQUEST_POOL_MAXSIZE=100
env = OGC_REQUEST_POOL_CONNECTIONS=100
env = EXIF_ENABLED=True
env = CREATE_LAYER=False
env = FAVORITE_ENABLED=True
env = SECURE_SSL_REDIRECT=True
env = SECURE_HSTS_INCLUDE_SUBDOMAINS=True

```

Fig. 220: UWSGI Configuration

(continued from previous page)

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
  
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  
→$(lsb_release -cs) stable"  
  
sudo apt-get update -y  
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose  
sudo apt autoremove --purge  
  
sudo usermod -aG docker geonode  
su geonode
```

Install the Docker and docker-compose packages on a CentOS host

Docker Setup (First time only)

Warning: The *centos-extras* repository must be enabled

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2  
  
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.  
→repo  
  
sudo yum install docker-ce docker-ce-cli containerd.io  
  
sudo systemctl start docker  
  
sudo usermod -aG docker geonode  
su geonode
```

Test Docker Compose Instance

Logout and login again on shell and then execute:

```
docker run -it hello-world
```

Deploy a vanilla GeoNode 2.10 with Docker

Clone the Project

```
# Let's create the GeoNode core base folder and clone it  
sudo mkdir -p /opt/geonode/  
sudo usermod -a -G www-data geonode  
sudo chown -Rf geonode:www-data /opt/geonode/  
sudo chmod -Rf 775 /opt/geonode/  
  
# Clone the GeoNode source code on /opt/geonode
```

(continues on next page)

(continued from previous page)

```
cd /opt
git clone https://github.com/GeoNode/geonode.git geonode
```

Start the Docker instances on localhost

Warning: The first time pulling the images will take some time. You will need a good internet connection.

```
cd /opt/geonode
docker-compose -f docker-compose.yml -f docker-compose.override.localhost.yml pull
docker-compose -f docker-compose.yml -f docker-compose.override.localhost.yml up -d
```

Note: If you want to re-build the docker images from scratch, instead of pulling them from the Docker Hub add the `--build` parameter to the up command, for instance:

```
docker-compose -f docker-compose.yml -f docker-compose.override.localhost.yml up --
↳build
```

In this case you can of course skip the pull step to download the pre-built images.

Note: To startup the containers daemonized, which means they will be started in the background (and keep running if you log out from the server or close the shell) add the `-d` option to the up command as in the following. `docker-compose` will take care to restart the containers if necessary (e.g. after boot).

```
docker-compose -f docker-compose.yml -f docker-compose.override.localhost.yml up -d

# If you want to rebuild the images also
docker-compose -f docker-compose.yml -f docker-compose.override.localhost.yml up --
↳build -d
```

Test the instance and follow the logs

If you run the containers daemonized (with the `-d` option), you can either run specific Docker commands to follow the startup and initialization logs or entering the image shell and check for the GeoNode logs.

In order to follow the startup and initialization logs, you will need to run the following command from the repository folder

```
cd /opt/geonode
docker logs -f django4geonode
```

Alternatively:

```
cd /opt/geonode
docker-compose logs -f django
```

You should be able to see several initialization messages. Once the container is up and running, you will see the following statements

```
...
789 static files copied to '/mnt/volumes/statics/static'.
static data refreshed
Executing UWSGI server uwsgi --ini /usr/src/app/uwsgi.ini for Production
[uWSGI] getting INI configuration from /usr/src/app/uwsgi.ini
```

To exit just hit CTRL+C.

This message means that the GeoNode containers have been started. Browsing to `http://localhost/` will show the GeoNode home page. You should be able to successfully log with the default admin user (admin / admin) and start using it right away.

With Docker it is also possible to run a shell in the container and follow the logs exactly the same as you deployed it on a physical host. To achieve this run

```
docker exec -it django4geonode /bin/bash

# Once logged in the GeoNode image, follow the logs by executing
tail -F -n 300 /var/log/geonode.log
```

Alternatively:

```
docker-compose exec django /bin/bash
```

To exit just hit CTRL+C and `exit` to return to the host.

Override the ENV variables to deploy on a public IP or domain

If you would like to start the containers on a public IP or domain, let's say `www.example.org`, you can

```
cd /opt/geonode

# Stop the Containers (if running)
docker-compose stop
```

Edit the ENV override file in order to deploy on `www.example.org`

```
# Make a copy of docker-compose.override.localhost.yml
cp docker-compose.override.localhost.yml docker-compose.override.example-org.yml
```

Replace everywhere `localhost` with `www.example.org`

```
vim docker-compose.override.example-org.yml
```

```
# e.g.: :%s/localhost/www.example.org/g

version: '2.2'
services:

  django:
    build: .
    # Loading the app is defined here to allow for
    # autoreload on changes it is mounted on top of the
    # old copy that docker added when creating the image
    volumes:
      - './usr/src/app'
```

(continues on next page)

(continued from previous page)

```

environment:
  - DEBUG=False
  - GEONODE_LB_HOST_IP=www.example.org
  - GEONODE_LB_PORT=80
  - SITEURL=http://www.example.org/
  - ALLOWED_HOSTS=['www.example.org', ]
  - GEOSERVER_PUBLIC_LOCATION=http://www.example.org/geoserver/
  - GEOSERVER_WEB_UI_LOCATION=http://www.example.org/geoserver/

celery:
  build: .
  volumes:
    - './usr/src/app'
  environment:
    - DEBUG=False
    - GEONODE_LB_HOST_IP=www.example.org
    - GEONODE_LB_PORT=80
    - SITEURL=http://www.example.org/
    - ALLOWED_HOSTS=['www.example.org', ]
    - GEOSERVER_PUBLIC_LOCATION=http://www.example.org/geoserver/
    - GEOSERVER_WEB_UI_LOCATION=http://www.example.org/geoserver/

geoserver:
  environment:
    - GEONODE_LB_HOST_IP=www.example.org
    - GEONODE_LB_PORT=80
#   - NGINX_BASE_URL=

```

Note: It is possible to override here even more variables to customize the GeoNode instance. See the GeoNode Settings section in order to get a list of the available options.

Run the containers in daemon mode

```
docker-compose -f docker-compose.yml -f docker-compose.override.example-org.yml up --
↳ build -d
```

Access the django4geonode Docker container to update the code-base and/or change internal settings

Access the container bash

```
docker exec -i -t django4geonode /bin/bash
```

You will be logged into the GeoNode instance as `root`. The folder is `/usr/src/app/` where the GeoNode project is cloned. Here you will find the GeoNode source code as in the GitHub repository.

Note: The machine is empty by default, no Ubuntu packages installed. If you need to install text editors or something you have to run the following commands:

```
apt update
apt install <package name>
```

(continues on next page)

(continued from previous page)

```
e.g.:  
apt install vim
```

Update the templates or the Django models. Once in the bash you can edit the templates or the Django models/classes. From here you can run any standard Django management command.

Whenever you change a template/CSS/Javascript remember to run later:

```
python manage.py collectstatic
```

in order to update the files into the `statics` Docker volume.

Warning: This is an external volume, and a simple restart won't update it. You have to be careful and keep it aligned with your changes.

Whenever you need to change some settings or environment variable, the easiest thing to do is to:

```
# Stop the container  
docker-compose stop  
  
# Restart the container in Daemon mode  
docker-compose -f docker-compose.yml -f docker-compose.override.<whatever>.yaml up -d
```

Whenever you change the model, remember to run later in the container via bash:

```
python manage.py makemigrations  
python manage.py migrate
```

Access the geoserver4geonode Docker container to update the GeoServer version

This procedure allows you to access the GeoServer container.

The concept is exactly the same as above, log into the container with bash.

```
# Access the container bash  
docker exec -it geoserver4geonode /bin/bash
```

You will be logged into the GeoServer instance as `root`.

GeoServer is deployed on an Apache Tomcat instance which can be found here

```
cd /usr/local/tomcat/webapps/geoserver
```

Warning: The GeoServer `DATA_DIR` is deployed on an external Docker Volume `geonode_gsdatadir`. This data dir won't be affected by changes to the GeoServer application since it is `external`.

Update the GeoServer instance inside the GeoServer Container

Warning: The old configuration will be kept since it is `external`

```
docker exec -it geoserver4geonode bash
```

```
cd /usr/local/tomcat/
wget --no-check-certificate https://build.geo-solutions.it/geonode/geoserver/latest/
↳ geoserver-2.16.2.war
mkdir tmp/geoserver
cd tmp/geoserver/
unzip /usr/local/tomcat/geoserver-2.16.2.war
rm -Rf data
cp -Rf /usr/local/tomcat/webapps/geoserver/data/ .
cd /usr/local/tomcat/
mv webapps/geoserver/ .
mv tmp/geoserver/ webapps/
exit
```

```
docker restart geoserver4geonode
```

Warning: GeoNode 2.8.1 is **NOT** compatible with GeoServer > 2.13.x

GeoNode 2.8.2 / 2.10.x are **NOT** compatible with GeoServer < 2.14.x

Remove all data and bring your running GeoNode deployment to the initial stage

This procedure allows you to stop all the containers and reset all the data with the deletion of all the volumes.

```
cd /opt/geonode
# stop containers and remove volumes
docker-compose down -v
```

Passages to completely get rid of old Docker images and volumes (reset the environment completely)

Note: For more details on Docker commands, please refer to the official Docker documentation.

It is possible to let docker show which containers are currently running (add `-a` for all containers, also stopped ones)

```
# Show the currently running containers
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
↳ 3b232931f820	geonode/nginx:production	"nginx -g 'daemon of...'"	26 minutes
↳ ago Up 26 minutes	0.0.0.0:80->80/tcp	nginx4geonode	
↳ ff7002ae6e91	geonode/geonode:latest	"/usr/src/app/entryp..."	26 minutes
↳ ago Up 26 minutes	8000/tcp	django4geonode	
↳ 2f155e5043be	geonode/geoserver:2.16.2	"/usr/local/tomcat/t..."	26 minutes
↳ ago Up 26 minutes	8080/tcp	geoserver4geonode	
↳ 97f1668a01b1	geonode_celery	"/usr/src/app/entryp..."	26 minutes
↳ ago Up 26 minutes	8000/tcp	geonode_celery_1	

(continues on next page)

(continued from previous page)

```
1b623598b1bd      geonode/postgis:10      "docker-entrypoint.s..."  About an_
↪hour ago      Up 26 minutes      5432/tcp      db4geonode
```

Stop all the containers by running

```
docker-compose stop
```

Force kill all containers by running

```
docker kill $(docker ps -q)
```

If you want to clean up all containers and images, without deleting the static volumes (i.e. the DB and the GeoServer catalog), issue the following commands

```
# Remove all containers
docker rm $(docker ps -a -q)

# Remove all docker images
docker rmi $(docker images -q)

# Prune the old images
docker system prune -a
```

If you want to remove a volume also

```
# List of the running volumes
docker volume ls

# Remove the GeoServer catalog by its name
docker volume rm -f geonode-gsdatadir

# Remove all dangling docker volumes
docker volume rm $(docker volume ls -qf dangling=true)

# update all images, should be run regularly to fetch published updates
for i in $(docker images| awk 'NR>1{print $1":"$2}'| grep -v '<none>'); do docker_
↪pull "$i" ;done
```

1.12 GeoNode Project

1.12.1 Overview

The following steps will guide you to a new setup of GeoNode Project. All guides will first install and configure the system to run it in `DEBUG` mode (also known as `DEVELOPMENT` mode) and then by configuring an `HTTPD` server to serve GeoNode through the standard `HTTP` (80) port.

Those guides **are not** meant to be used on a production system. There will be dedicated chapters that will show you some *hints* to optimize GeoNode for a production-ready machine. In any case, we strongly suggest to task an experienced *DevOp* or *System Administrator* before exposing your server to the `WEB`.

1.12.2 Ubuntu 18.04

This part of the documentation describes the complete setup process for GeoNode on an Ubuntu 18.04 64-bit clean environment (Desktop or Server). All examples use shell commands that you must enter on a local terminal or a remote shell. - If you have a graphical desktop environment you can open the terminal application after login; - if you are working on a remote server the provider or sysadmin should have given you access through an ssh client.

Install the dependencies

In this section, we are going to install all the basic packages and tools needed for a complete GeoNode installation. To follow this guide, a piece of basic knowledge about Ubuntu Server configuration and working with a shell is required. This guide uses `vim` as the editor; feel free to use `nano`, `gedit` or others.

Upgrade system packages

Check that your system is already up-to-date with the repository running the following commands:

```
sudo apt update
sudo apt upgrade
```

Create a Dedicated User

In the following steps a User named `geonode` is used: to run installation commands the user must be in the `sudo` group.

Create User `geonode` **if not present**:

```
# Follow the prompts to set the new user's information.
# It is fine to accept the defaults to leave all of this information blank.
sudo adduser geonode

# The following command adds the user geonode to group sudo
sudo usermod -aG sudo geonode

# make sure the newly created user is allowed to login by ssh
# (out of the scope of this documentation) and switch to User geonode
su geonode
```

Packages Installation

Note: You don't need to install the **system packages** if you want to run the project using Docker

First, we are going to install all the **system packages** needed for the GeoNode setup.

```
# Install packages from GeoNode core
sudo apt install -y gdal-bin
sudo apt install -y python3-pip python3-dev python3-virtualenv python3-venv
↳ virtualenvwrapper
sudo apt install -y libxml2 libxml2-dev gettext
```

(continues on next page)

(continued from previous page)

```
sudo apt install -y libxslt1-dev libjpeg-dev libpng-dev libpq-dev libgdal-dev
↪libgdal20
sudo apt install -y software-properties-common build-essential
sudo apt install -y git unzip gcc zlib1g-dev libgeos-dev libproj-dev
sudo apt install -y sqlite3 spatialite-bin libsqlite3-mod-spatialite

# Install Openjdk
sudo -i apt update
sudo apt install openjdk-8-jdk-headless default-jdk-headless -y
sudo update-java-alternatives --jre-headless --jre --set java-1.8.0-openjdk-amd64

sudo apt update -y
sudo apt upgrade -y
sudo apt autoremove -y
sudo apt autoclean -y
sudo apt purge -y
sudo apt clean -y

# Install Packages for Virtual environment management
sudo apt install -y virtualenv virtualenvwrapper

# Install text editor
sudo apt install -y vim
```

Geonode Project Installation

Geonode project is the proper way to run a customized installation of Geonode. The repository of geonode-project contains a minimal set of files following the structure of a django-project. Geonode itself will be installed as a requirement of your project. Inside the project structure is possible to extend, replace or modify all geonode components (e.g. css and other static files, templates, models..) and even register new django apps **without touching the original Geonode code**.

Note: You can call your geonode project whatever you like following the naming conventions for python packages (generally lower case with underscores (_)). In the examples below, replace `my_geonode` with whatever you would like to name your project.

See also the [README](#) file on geonode-project repository

First of all we need to prepare a new Python Virtual Environment

Prepare the environment

```
sudo mkdir -p /opt/geonode_custom/
sudo usermod -a -G www-data geonode
sudo chown -Rf geonode:www-data /opt/geonode_custom/
sudo chmod -Rf 775 /opt/geonode_custom/
```

Clone the source code

```
cd /opt/geonode_custom/
git clone https://github.com/GeoNode/geonode-project.git
```

Make an instance out of the Django Template

Note: We will call our instance `my_geonode`. You can change the name at your convenience.

```
vim ~/.bashrc
# add the following line to the bottom
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

```
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 my_geonode
pip install Django==2.2.9

django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,
→env,sample -n monitoring-cron -n Dockerfile my_geonode

# Install the Python packages
cd /opt/geonode_custom/my_geonode
pip install -r requirements.txt --upgrade --no-cache --no-cache-dir
pip install -e . --upgrade

# Install GDAL Utilities for Python
pip install pygdal=="`gdal-config --version`.*"
```

Run GeoNode Project for the first time in DEBUG Mode

Warning: Be sure you have successfully completed all the steps of the section *Install the dependencies*.

This command will run both GeoNode and GeoServer locally after having prepared the SQLite database. The server will start in DEBUG (or DEVELOPMENT) mode, and it will start the following services:

1. GeoNode on `http://localhost:8000/`
2. GeoServer on `http://localhost:8080/geoserver/`

This modality is beneficial to debug issues and/or develop new features, but it cannot be used on a production system.

```
# Prepare the GeoNode SQLite database (the first time only)
paver setup
paver sync
```

Note: In case you want to start again from a clean situation, just run

```
paver reset_hard
```

Warning: This will blow up completely your `local_settings`, delete the SQLite database and remove the GeoServer data dir.

```
# Run the server in DEBUG mode
paver start
```

Once the server has finished the initialization and prints on the console the sentence `GeoNode is now available.`, you can open a browser and go to:

```
http://localhost:8000/
```

Sign-in with:

```
user: admin
password: admin
```

From now on, everything already said for GeoNode Core (please refer to the section *Postgis database Setup* and following), applies to a GeoNode Project.

Be careful to use the **new** paths and names everywhere:

- Everytime you'll find the keyword `geonode`, you'll need to use your geonode custom name instead (in this example `my_geonode`).
- Everytime you'll find paths pointing to `/opt/geonode/`, you'll need to update them to point to your custom project instead (in this example `/opt/geonode_custom/my_geonode`).

1.12.3 Docker

Warning: Before moving with this section, you should have read and clearly understood the `INSTALLATION > GeoNode Core` sections, and in particular the `Docker` one. Everything said for the `GeoNode Core Vanilla` applies here too, except that the `Docker` container names will be slightly different. As an instance if you named your project `my_geonode`, your containers will be called:

```
'django4my_geonode' instead of 'django4geonode' and so on...
```

Deploy an instance of a geonode-project Django template 2.10.x with Docker on localhost

Prepare the environment

```
sudo mkdir -p /opt/geonode_custom/
sudo usermod -a -G www-data geonode
sudo chown -Rf geonode:www-data /opt/geonode_custom/
sudo chmod -Rf 775 /opt/geonode_custom/
```

Clone the source code

```
cd /opt/geonode_custom/
git clone https://github.com/GeoNode/geonode-project.git
```

Make an instance out of the Django Template

Note: We will call our instance `my_geonode`. You can change the name at your convenience.

```
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 my_geonode
pip install Django==2.2.9
```

(continues on next page)

(continued from previous page)

```
django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,
↪env,sample -n monitoring-cron -n Dockerfile my_geonode
cd /opt/geonode_custom/my_geonode
```

Modify the code and the templates and rebuild the Docker Containers

```
docker-compose -f docker-compose.yml build --no-cache
```

Finally, run the containers

```
docker-compose -f docker-compose.yml up -d
```

Deploy an instance of a geonode-project Django template 2.10.x with Docker on a domain

Note: We will use `www.example.org` as an example. You can change the name at your convenience.

Stop the containers

```
cd /opt/geonode_custom/my_geonode

docker-compose -f docker-compose.yml stop
```

Edit the ENV override file in order to deploy on `www.example.org`

Replace everywhere `localhost` with `www.example.org`

```
vim .env
```

```
# e.g.: :%s/localhost/www.example.org/g
```

Note: It is possible to override here even more variables to customize the GeoNode instance. See the GeoNode Settings section in order to get a list of the available options.

Run the containers in daemon mode

```
docker-compose -f docker-compose.yml -f docker-compose.override.example-org.yml up --
↪build -d
```

1.13 SPCGeoNode

1.13.1 Overview

SPCgeonode is a setup for Geonode deployment at SPC. It makes it easy to deploy a production ready Geonode. The setup aims for simplicity over flexibility, so that it will only apply for typical small scale Geonode installations.

The setup is also usable for Geonode development or customization.

1.13.2 Prerequisites

Make sure you have a version of Docker (tested with 17.12) and docker-compose.

- Linux : <https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-from-a-package> and <https://docs.docker.com/compose/install/#install-compose>
- Windows : <https://store.docker.com/editions/community/docker-ce-desktop-windows>
- Mac : <https://store.docker.com/editions/community/docker-ce-desktop-mac>

1.13.3 Usage

All the following commands happen from this folder:

```
cd /path/to/geonode/scripts/spcgeonode
```

Development

To start only the main services (should be enough for development) :

```
docker-compose up --build -d django geoserver postgres nginx
```

To start the whole stack :

```
docker-compose up --build -d
```

If not familiar with Docker, read below to know how to see what's happening. On first start, the containers will restart several times. Once everything started, you should be able to open <http://127.0.0.1> in your browser. See how to edit the configuration below if you install on another computer.

Production (using composer)

Using a text editor, edit the `.env` file (you can also override those with environment variables) :

```
# General configuration
nano .env
```

When ready, start the stack using this commands:

```
# Run the stack
docker-compose -f docker-compose.yml up -d --build
```

Alternatively, you can pull the images from dockerhub instead of rebuilding (only applies if you haven't changed the docker setup):

```
# Pull the images and run the stack
docker-compose -f docker-compose.yml pull
docker-compose -f docker-compose.yml up -d
```

If not familiar with Docker, read below to know how to see what's happening. On first start, the containers will restart several times. Once everything started, you should be able to open http://your_http_host or https://your_https_host in your browser.

1.13.4 Upgrade

If at some point you want to update the SPCgeonode setup (this will work only if you didn't do modifications, if you did, you need to merge them):

```
# Get the update setup
git pull
```

Upgrade the stack

```
docker-compose -f docker-compose.yml up -d --build
```

1.13.5 Development vs Production

Difference of dev setup vs prod setup:

- Django source is mounted on the host and uwsgi does live reload (so that edits to the python code is reloaded live)
- Django static and media folder, Geoserver's data folder and Certificates folder are mounted on the host (just to easily see what's happening)
- Django debug is set to True
- Postgres's port 5432 is exposed (to allow debugging using pgadmin)
- Nginx debug mode is activated (not really sure what this changes)
- Docker tags are set to dev instead of latest

1.13.6 Releases

To make a release:

- checkout spcgeonode-release
- merge spcgeonode
- replace the version tag in docker-compose.yml with the version (format *x.x.x*)
- commit
- create a git tag (format *spc/x.x.x*)
- push spcgeonode-release with tags

This will trigger an automatic build on docker hub.

If you need to manually publish the image (e.g. dockerhub build fail) :

```
docker login
docker-compose -f docker-compose.yml build
docker-compose -f docker-compose.yml push
```

1.13.7 FAQ

Docker-primer - How to see what's happening?

If not familiar with Docker, here are some useful commands:

- `docker ps`: list all containers and their status
- `docker-compose logs -f`: show live stdout from all containers
- `docker-compose logs -f django`: show live stdout from a specific container (replace *django* by *geoserver*, *postgres*, etc.)
- `docker-compose down -v`: brings the stack down including volumes, allowing you to restart from scratch **THIS WILL ERASE ALL DATA !!**

During startup, a lot of container crash and restart, is it normal?

This is the normal startup process. Due to the nature of the setup, the containers are very interdependent. Startup from scratch can take approx. 5-10 minutes, during which all containers may restart a lot of times.

In short, Django will restart until Postgres is up so it can migrate the database. Geoserver will restart until Django has configured OAuth so it can get OAuth2 configuration. Django will restart until Geoserver is running so it can reinitialize the master password.

Backups

Backups are made using [RClone](<https://rclone.org/docs/>). RClone is a flexible file syncing tool that supports all commons cloud provider, regular file transfer protocols as well as local filesystem. It should be able to accommodate almost any setup.

The only available configuration provided with the setup assumes Amazon S3 is being used, in which case you need to replace the following parts of the `rclone.backup.config` file : `YOUR_S3_ACCESS_KEY_HERE`, `YOUR_S3_SECRET_KEY_HERE`, `YOUR_S3_REGION_HERE` and `THE_NAME_OF_YOUR_BUCKET_HERE` (watch [this](<https://www.youtube.com/watch?v=BLTy2tQXQLY>) to learn how to get these keys).

Also consider enabling *versioning* on the Bucket, so that if data won't get lost if deleted accidentally in GeoNode.

If you want to setup backups using another provider, check the [RClone documentation](<https://rclone.org/docs/>). It should be easy to add any RClone supported provider to SPCgeonode.

How to migrate from an existing standard Geonode install

This section lists the steps done to migrate from an apt-get install of Geonode 2.4.1 (with Geoserver 2.7.4) to a fresh SPCGeonode 0.1 install. It is meant as a guide only as some steps may need some tweaking depending on your installation. Do not follow these steps if you don't understand what you're doing.

Prerequisites

- access to the original server
- a new server for the install (can be the same than the first one if you don't fear losing all data) - ideally linux but should be OK as long as it runs docker (64bits)
- an external hard-drive to copy data over

On the old server

```
# Move to the external hard drive
cd /path/to/your/external/drive
```

1. Find the current database password (look for DATABASE_PASSWORD, in my case it was XbFAyE4w)

```
more /etc/geonode/local_settings.py
```

2. Dump the database content (you will be prompted several time for the password above)

```
pg_dumpall --host=127.0.0.1 --username=geonode --file=pg_dumpall.custom
```

3. Copy all uploaded files

```
cp -r /var/www/geonode/uploaded uploaded
```

4. Copy geoserver data directory

```
cp -r /usr/share/geoserver/data geodatadir
```

On the new server

Setup SPCGeonode by following the prerequisite and production steps on <https://github.com/GeoNode/geonode/tree/master/scripts/spcgeonode> up to (but not including) run the stack.

Then run these commands:

```
# Prepare the stack (without running)
docker-compose -f docker-compose.yml pull --no-parallel
docker-compose -f docker-compose.yml up --no-start

# Start the database
docker-compose -f docker-compose.yml up -d postgres

# Initialize geoserver (to create the geodatadir)
docker-compose -f docker-compose.yml run --rm geoserver true

# Go to the external drive
cd /path/to/drive/

# Restore the dump (this can take a while if you have data in postgres)
cat pg_dumpall.custom | docker exec -i spcgeonode_postgres_1 psql -U postgres
# Rename the database to postgres
docker exec -i spcgeonode_postgres_1 dropdb -U postgres postgres
```

(continues on next page)

(continued from previous page)

```

docker exec -i spcgeonode_postgres_1 psql -d template1 -U postgres -c "ALTER DATABASE_
↳geonode RENAME TO postgres;"

# Restore the django uploaded files
docker cp uploaded/. spcgeonode_django_1:/spcgeonode-media/

# Restore the workspaces and styles of the geoserver data directory
docker cp geodatadir/styles/. spcgeonode_geoserver_1:/spcgeonode-geodatadir/styles
docker cp geodatadir/workspaces/. spcgeonode_geoserver_1:/spcgeonode-geodatadir/
↳workspaces
docker cp geodatadir/data/. spcgeonode_geoserver_1:/spcgeonode-geodatadir/data

# Back to SPCgeonode
cd /path/to/SPCgeonode

# Fix some inconsistency that prevents migrations (public.layers_layer shouldn't have_
↳service_id column)
docker exec -i spcgeonode_postgres_1 psql -U postgres -c "ALTER TABLE public.layers_
↳layer DROP COLUMN service_id;"

# Migrate with fake initial
docker-compose -f docker-compose.yml run --rm --entrypoint "python manage.py migrate -
↳-fake-initial" django

# Create the SQL diff to fix the schema # TODO : upstream some changes to django-
↳extensions for this to work directly
docker-compose -f docker-compose.yml run --rm --entrypoint '/bin/sh -c "DJANGO_
↳COLORS=nocolor python manage.py sqldiff -ae"' django >> fix.sql

# Manually fix the SQL command until it runs (you can also drop the tables that have_
↳no model)
nano fix.sql

# Apply the SQL diff (review the sql file first as this may delete some important_
↳tables)
cat fix.sql | docker exec -i spcgeonode_postgres_1 psql -U postgres

# Set all layers as approved
docker exec -i spcgeonode_postgres_1 psql -U postgres -c 'UPDATE base_resourcebase_
↳SET is_approved = TRUE;'

# This time start the stack
docker-compose -f docker-compose.yml up -d

```

One last step was to connect to the GeoServer administration and change the PostGIS store host, user and password to 'postgres'.

On windows, I have error like `standard_init_linux.go:190: exec user process caused "no such file or directory"`

This may be due to line endings. When checking out files, git optionally converts line endings to match the platform, which doesn't work well it `.sh` files.

To fix, use `git config --global core.autocrlf false` and checkout again.

1.14 GeoNode Settings

Settings

1.14.1 Settings

Here's a list of settings available in GeoNode and their default values. This includes settings for some external applications that GeoNode depends on.

For most of them, default values are good. Those should be changed only for advanced configurations in production or heavily hardened systems.

The most common ones can be set through environment variables to avoid touching the `settings.py` file at all. This is a good practice and also the preferred one to configure GeoNode (and Django apps in general). Whenever you need to change them, set the environment variable accordingly (where it is available) instead of overriding it through the `local_settings`.

A

ACCESS_TOKEN_EXPIRE_SECONDS

Default: 86400

Env: ACCESS_TOKEN_EXPIRE_SECONDS

When a user logs into GeoNode, if no ACCESS_TOKEN exists, a new one will be created with a default expiration time of ACCESS_TOKEN_EXPIRE_SECONDS seconds (1 day by default).

ACCOUNT_ADAPTER

Default: `geonode.people.adapters.LocalAccountAdapter`

Custom GeoNode People (Users) Account Adapter.

ACCOUNT_APPROVAL_REQUIRED

Default: False

Env: ACCOUNT_APPROVAL_REQUIRED

If ACCOUNT_APPROVAL_REQUIRED equals True, newly registered users must be activated by a superuser through the Admin gui, before they can access GeoNode.

ACCOUNT_CONFIRM_EMAIL_ON_GET

Default: `True`

This is a [django-allauth setting](#) It allows specifying the HTTP method used when confirming e-mail addresses.

ACCOUNT_EMAIL_REQUIRED

Default: `True`

This is a [django-allauth setting](#) which controls whether the user is required to provide an e-mail address upon registration.

ACCOUNT_EMAIL_VERIFICATION

Default: `optional`

This is a [django-allauth setting](#)

ACCOUNT_LOGIN_REDIRECT_URL

Default: `SITEURL`

Env: `LOGIN_REDIRECT_URL`

This is a [django-user-accounts setting](#) It allows specifying the default redirect URL after a successful login.

ACCOUNT_LOGOUT_REDIRECT_URL

Default: `SITEURL`

Env: `LOGOUT_REDIRECT_URL`

This is a [django-user-accounts setting](#) It allows specifying the default redirect URL after a successful logout.

ACCOUNT_NOTIFY_ON_PASSWORD_CHANGE

Default: `True`

Env: `ACCOUNT_NOTIFY_ON_PASSWORD_CHANGE`

This is a [django-user-accounts setting](#)

ACCOUNT_OPEN_SIGNUP

Default: True

Env: ACCOUNT_OPEN_SIGNUP

This is a [django-user-accounts](#) setting Whether or not people are allowed to self-register to GeoNode or not.

ACCOUNT_SIGNUP_FORM_CLASS

Default: `geonode.people.forms.AllauthReCaptchaSignupForm`

Env: ACCOUNT_SIGNUP_FORM_CLASS

Enabled only when the [RECAPTCHA_ENABLED](#) option is True.

Ref. to [RECAPTCHA_ENABLED](#)

ACTSTREAM_SETTINGS

Default:

```
{
  'FETCH_RELATIONS': True,
  'USE_PREFETCH': False,
  'USE_JSONFIELD': True,
  'GFK_FETCH_DEPTH': 1,
}
```

Actstream Settings.

ADMIN_MODERATE_UPLOADS

Default: False

When this variable is set to True, every uploaded resource must be approved before becoming visible to the public users.

Until a resource is in `PENDING APPROVAL` state, only the superusers, owner and group members can access it, unless specific edit permissions have been set for other users or groups.

A Group Manager *can* approve the resource, but he cannot publish it whenever the setting `RESOURCE_PUBLISHING` is set to True. Otherwise, if `RESOURCE_PUBLISHING` is set to False, the resource becomes accessible as soon as it is approved.

AGON_RATINGS_CATEGORY_CHOICES

Default:

```
{
  "maps.Map": {
    "map": "How good is this map?"
  },
  "layers.Layer": {
    "layer": "How good is this layer?"
  },
  "documents.Document": {
    "document": "How good is this document?"
  }
}
```

ALLOWED_DOCUMENT_TYPES

Default:

```
['doc', 'docx', 'gif', 'jpg', 'jpeg', 'ods', 'odt', 'odp', 'pdf', 'png',
'ppt', 'pptx', 'rar', 'sld', 'tif', 'tiff', 'txt', 'xls', 'xlsx', 'xml',
'zip', 'gz', 'qml']
```

A list of acceptable file extensions that can be uploaded to the Documents app.

ANONYMOUS_USER_ID

Default: -1

Env: ANONYMOUS_USER_ID

The id of an anonymous user. This is an django-guardian setting.

API_INCLUDE_REGIONS_COUNT

Default: False

Env: API_INCLUDE_REGIONS_COUNT

If set to True, a counter with the total number of available regions will be added to the API JSON Serializer.

API_LIMIT_PER_PAGE

Default: 200

Env: API_LIMIT_PER_PAGE

The Number of items returned by the APIs 0 equals no limit. Different from CLIENT_RESULTS_LIMIT, affecting the number of items per page in the resource list.

API_LOCKDOWN

Default: True

Env: API_LOCKDOWN

If this is set to `True` users must be authenticated to get search results when search for for users, groups, categories, regions, tags etc. Filtering search results of Resourcebase-objects like Layers, Maps or Documents by one of the above types does not work. Attention: If `API_LOCKDOWN` is set to `False` all details can be accessed by anonymous users.

ASYNC_SIGNALS

Default: False

Env: ACCOUNT_NOTIFY_ON_PASSWORD_CHANGE

AUTH_EXEMPT_URLS

Default:

```
(r'^/?$',
'/gs/*',
'/static/*',
'/o/*',
'/api/o/*',
'/api/roles',
'/api/adminRole',
'/api/users',
'/api/layers',)
```

A tuple of URL patterns that the user can visit without being authenticated. This setting has no effect if `LOCKDOWN_GEONODE` is not `True`. For example, `AUTH_EXEMPT_URLS = ('/maps',)` will allow unauthenticated users to browse maps.

AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME

Default: True

Env: AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME

Auto assign users to a default `REGISTERED_MEMBERS_GROUP_NAME` private group after `AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_AT`.

AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_AT

Default: activation

Env: AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_AT

Options: "registration" | "activation" | "login"

Auto assign users to a default `REGISTERED_MEMBERS_GROUP_NAME` private group after {"registration" | "activation" | "login"}.

Notice that whenever `ACCOUNT_EMAIL_VERIFICATION == True` and `ACCOUNT_APPROVAL_REQUIRED == False`, users will be able to register and they became active already, even if they won't be able to login until the email has been verified.

AUTO_GENERATE_AVATAR_SIZES

Default: 20, 30, 32, 40, 50, 65, 70, 80, 100, 140, 200, 240

An iterable of integers representing the sizes of avatars to generate on upload. This can save rendering time later on if you pre-generate the resized versions.

AWS_ACCESS_KEY_ID

Default: ''

Env: `AWS_ACCESS_KEY_ID`

This is a [Django storage setting](#) Your Amazon Web Services access key, as a string.

Warning: This works only if `DEBUG = False`

AWS_BUCKET_NAME

Default: ''

Env: `S3_BUCKET_NAME`

The name of the S3 bucket GeoNode will pull static and/or media files from. Set through the environment variable `S3_BUCKET_NAME`. This is a [Django storage setting](#)

Warning: This works only if `DEBUG = False`

AWS_QUERYSTRING_AUTH

Default: `False`

This is a [Django storage setting](#) Setting `AWS_QUERYSTRING_AUTH` to `False` to remove query parameter authentication from generated URLs. This can be useful if your S3 buckets are public.

Warning: This works only if `DEBUG = False`

AWS_S3_BUCKET_DOMAIN

<https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L1661>

```
AWS_S3_BUCKET_DOMAIN = '%s.s3.amazonaws.com' % AWS_STORAGE_BUCKET_NAME
```

Warning: This works only if `DEBUG = False`

AWS_SECRET_ACCESS_KEY

Default: ''

Env: `AWS_SECRET_ACCESS_KEY`

This is a [Django storage setting](#) Your Amazon Web Services secret access key, as a string.

Warning: This works only if `DEBUG = False`

AWS_STORAGE_BUCKET_NAME

Default: ''

Env: `S3_BUCKET_NAME`

This is a [Django storage setting](#) Your Amazon Web Services storage bucket name, as a string.

Warning: This works only if `DEBUG = False`

B

BING_API_KEY

Default: None

Env: `BING_API_KEY`

This property allows to enable a Bing Aerial background.

If using `mapstore` client library, make sure the `MAPSTORE_BASELAYERS` include the following:

```
if BING_API_KEY:
    BASEMAP = {
        "type": "bing",
        "title": "Bing Aerial",
        "name": "AerialWithLabels",
        "source": "bing",
        "group": "background",
        "apiKey": "{{apiKey}}",
        "visibility": False
    }
    DEFAULT_MS2_BACKGROUNDS = [BASEMAP,] + DEFAULT_MS2_BACKGROUNDS
```

BROKER_HEARTBEAT

Default: 0

Heartbeats are used both by the client and the broker to detect if a connection was closed. This is a [Celery setting](#).

BROKER_TRANSPORT_OPTIONS

Default:

```
{
    'fanout_prefix': True,
    'fanout_patterns': True,
    'socket_timeout': 60,
    'visibility_timeout': 86400
}
```

This is a [Celery setting](#).

C

CACHES

Default:

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.dummy.DummyCache',
    },
    'resources': {
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
        'TIMEOUT': 600,
        'OPTIONS': {
            'MAX_ENTRIES': 10000
        }
    }
}
```

A dictionary containing the settings for all caches to be used with Django. This is a [Django setting](#)

The 'default' cache is disabled because we don't have a mechanism to discriminate between client sessions right now, and we don't want all users fetch the same api results.

The 'resources' is not currently used. It might be helpful for [caching Django template fragments](#) and/or [Tastypie API Caching](#).

CACHE_BUSTING_MEDIA_ENABLED

Default: False

Env: CACHE_BUSTING_MEDIA_ENABLED

This is a [Django ManifestStaticFilesStorage](#) storage setting A boolean allowing you to enable the ManifestStaticFilesStorage storage. This works only on a production system.

Warning: This works only if `DEBUG = False`

CACHE_BUSTING_STATIC_ENABLED

Default: False

Env: CACHE_BUSTING_STATIC_ENABLED

This is a [Django ManifestStaticFilesStorage](#) storage setting A boolean allowing you to enable the ManifestStaticFilesStorage storage. This works only on a production system.

Warning: This works only if `DEBUG = False`

CASCADE_WORKSPACE

Default: geonode

Env: CASCADE_WORKSPACE

CATALOGUE

A dict with the following keys:

ENGINE: The CSW backend (default is `geonode.catalogue.backends.pycsw_local`) URL: The FULLY QUALIFIED base URL to the CSW instance for this GeoNode USERNAME: login credentials (if required) PASSWORD: login credentials (if required)

pycsw is the default CSW enabled in GeoNode. pycsw configuration directives are managed in the PYCSW entry.

CELERYD_POOL_RESTARTS

Default: True

This is a [Celery](#) setting.

CELERY_ACCEPT_CONTENT

Default: ['json']

This is a [Celery setting](#).

CELERY_ACKS_LATE

Default: True

This is a [Celery setting](#)

CELERY_BEAT_SCHEDULE

Here you can define your scheduled task.

CELERY_DISABLE_RATE_LIMITS

Default: False

This is a [Celery setting](#).

CELERY_ENABLE_UTC

Default: True

This is a [Celery setting](#).

CELERY_MAX_CACHED_RESULTS

Default: 32768

This is a [Celery setting](#).

CELERY_MESSAGE_COMPRESSION

Default: gzip

This is a [Celery setting](#).

CELERY_RESULT_PERSISTENT

Default: False

This is a [Celery setting](#).

CELERY_RESULT_SERIALIZER

Default: json

This is a [Celery setting](#).

CELERY_SEND_TASK_SENT_EVENT

Default: True

If enabled, a task-sent event will be sent for every task so tasks can be tracked before they are consumed by a worker. This is a [Celery setting](#).

CELERY_TASK_ALWAYS_EAGER

Default: False if ASYNC_SIGNALS else True

This is a [Celery setting](#).

CELERY_TASK_CREATE_MISSING_QUEUES

Default: True

This is a [Celery setting](#).

CELERY_TASK_IGNORE_RESULT

Default: True

This is a [Celery setting](#).

CELERY_TASK_QUEUES

Default:

```
Queue('default', GEONODE_EXCHANGE, routing_key='default'),
Queue('geonode', GEONODE_EXCHANGE, routing_key='geonode'),
Queue('update', GEONODE_EXCHANGE, routing_key='update'),
Queue('cleanup', GEONODE_EXCHANGE, routing_key='cleanup'),
Queue('email', GEONODE_EXCHANGE, routing_key='email'),
```

A tuple with registered Queues.

CELERY_TASK_RESULT_EXPIRES

Default: 43200

Env: CELERY_TASK_RESULT_EXPIRES

This is a [Celery setting](#).

CELERY_TASK_SERIALIZER

Default: json Env: CELERY_TASK_SERIALIZER

This is a [Celery setting](#).

CELERY_TIMEZONE

Default: UTC

Env: TIME_ZONE

This is a [Celery setting](#).

CELERY_TRACK_STARTED

Default: True

This is a [Celery setting](#).

CELERY_WORKER_DISABLE_RATE_LIMITS

Default: False

Disable the worker rate limits (number of tasks that can be run in a given time frame).

CELERY_WORKER_SEND_TASK_EVENTS

Default: False

Send events so the worker can be monitored by other tools.

CLIENT_RESULTS_LIMIT

Default: 5

Env: CLIENT_RESULTS_LIMIT

The Number of results per page listed in the GeoNode search pages. Different from
API_LIMIT_PER_PAGE, affecting the number of items returned by the APIs.

CREATE_LAYER

Default: False

Env: CREATE_LAYER

Enable the create layer plugin.

CKAN_ORIGINS

Default:

```

CKAN_ORIGINS = [{
    "label": "Humanitarian Data Exchange (HDX)",
    "url": "https://data.hdx.rwlab.org/dataset/new?title={name}&notes=
↪ {abstract}",
    "css_class": "hdx"
}]

```

A list of dictionaries that are used to generate the links to CKAN instances displayed in the Share tab. For each origin, the name and abstract format parameters are replaced by the actual values of the Resource-Base object (layer, map, document). This is not enabled by default. To enable, uncomment the following line: `SOCIAL_ORIGINS.extend(CKAN_ORIGINS)`.

CSRF_COOKIE_HTTPONLY

Default: False

Env: CSRF_COOKIE_HTTPONLY

Whether to use HttpOnly flag on the CSRF cookie. If this is set to True, client-side JavaScript will not be able to access the CSRF cookie. This is a [Django Setting](#)

CSRF_COOKIE_SECURE

Default: False

Env: CSRF_COOKIE_SECURE

Whether to use a secure cookie for the CSRF cookie. If this is set to True, the cookie will be marked as “secure,” which means browsers may ensure that the cookie is only sent with an HTTPS connection. This is a [Django Setting](#)

D

DATA_UPLOAD_MAX_NUMBER_FIELDS

Default: 100000

Maximum value of parsed attributes.

DEBUG

Default: `False`

Env: `DEBUG`

One of the main features of debug mode is the display of detailed error pages. If your app raises an exception when `DEBUG` is `True`, Django will display a detailed traceback, including a lot of metadata about your environment, such as all the currently defined Django settings (from `settings.py`). This is a [Django Setting](#)

DEBUG_STATIC

Default: `False`

Env: `DEBUG_STATIC`

Load non minified version of static files.

DEFAULT_ANONYMOUS_DOWNLOAD_PERMISSION

Default: `True`

Whether the uploaded resources should be downloadable by default.

DEFAULT_ANONYMOUS_VIEW_PERMISSION

Default: `True`

Whether the uploaded resources should be public by default.

DEFAULT_LAYER_FORMAT

Default: `image/png`

Env: `DEFAULT_LAYER_FORMAT`

The default format for requested tile images.

DEFAULT_MAP_CENTER

Default: `(0, 0)`

Env: `DEFAULT_MAP_CENTER_X` `DEFAULT_MAP_CENTER_Y`

A 2-tuple with the latitude/longitude coordinates of the center-point to use in newly created maps.

DEFAULT_MAP_CRS

Default: EPSG:3857

Env: DEFAULT_MAP_CRS

The default map projection. Default: EPSG:3857

DEFAULT_MAP_ZOOM

Default: 0

Env: DEFAULT_MAP_ZOOM

The zoom-level to use in newly created maps. This works like the OpenLayers zoom level setting; 0 is at the world extent and each additional level cuts the viewport in half in each direction.

DEFAULT_SEARCH_SIZE

Default: 10

Env: DEFAULT_SEARCH_SIZE

An integer that specifies the default search size when using `geonode.search` for querying data.

DEFAULT_WORKSPACE

Default: geonode

Env: DEFAULT_WORKSPACE

The standard GeoServer workspace.

DELAYED_SECURITY_SIGNALS

Default: False

Env: DELAYED_SECURITY_SIGNALS

This setting only works when `GEOFENCE_SECURITY_ENABLED` has been set to `True` and GeoNode is making use of the GeoServer `BACKEND`.

By setting this to `True`, every time the permissions will be updated/changed for a Layer, they won't be applied immediately but only and only if either:

- A Celery Worker is running and it is able to execute the `geonode.security.tasks.sync_guardian` periodic task; notice that the task will be executed at regular intervals, based on the interval value defined in the corresponding `PeriodicTask` model.
- A periodic `cron` job runs the `sync_security_rules` management command, or either it is manually executed from the Django shell.
- The user, owner of the Layer or with rights to change its permissions, clicks on the GeoNode UI button `Sync permissions` immediately

Warning: Layers won't be accessible to public users anymore until the Security Rules are not synchronized!

DISPLAY_COMMENTS

Default: True

Env: DISPLAY_COMMENTS

If set to False comments are hidden.

DISPLAY_RATINGS

Default: True

Env: DISPLAY_RATINGS

If set to False ratings are hidden.

DISPLAY_SOCIAL

Default: True

Env: DISPLAY_SOCIAL

If set to False social sharing is hidden.

DISPLAY_WMS_LINKS

Default: True

Env: DISPLAY_WMS_LINKS

If set to False direct WMS link to GeoServer is hidden.

DISPLAY_ORIGINAL_DATASET_LINK

Default: True

Env: DISPLAY_ORIGINAL_DATASET_LINK

If set to False original dataset download is hidden.

DOWNLOAD_FORMATS_METADATA

Specifies which metadata formats are available for users to download.

Default:

```
DOWNLOAD_FORMATS_METADATA = [  
    'Atom', 'DIF', 'Dublin Core', 'ebRIM', 'FGDC', 'ISO',  
]
```


DOWNLOAD_FORMATS_VECTOR

Specifies which formats for vector data are available for users to download.

Default:

```

DOWNLOAD_FORMATS_VECTOR = [
    'JPEG', 'PDF', 'PNG', 'Zipped Shapefile', 'GML 2.0', 'GML 3.1.1', 'CSV',
    'Excel', 'GeoJSON', 'KML', 'View in Google Earth', 'Tiles',
]

```

DOWNLOAD_FORMATS_RASTER

Specifies which formats for raster data are available for users to download.

Default:

```

DOWNLOAD_FORMATS_RASTER = [
    'JPEG', 'PDF', 'PNG', 'Tiles',
]

```

E

EMAIL_ENABLE

Default: False

Options:

- EMAIL_BACKEND
 - Default: `django.core.mail.backends.smtp.EmailBackend`
 - Env: `DJANGO_EMAIL_BACKEND`
- EMAIL_HOST
 - Default: `localhost`
- EMAIL_PORT
 - Default: `25`
- EMAIL_HOST_USER
 - Default: `' '`
- EMAIL_HOST_PASSWORD
 - Default: `' '`
- EMAIL_USE_TLS
 - Default: `False`
- DEFAULT_FROM_EMAIL
 - Default: `GeoNode <no-reply@geonode.org>`

EPSG_CODE_MATCHES

Default:

```
{
  'EPSG:4326': '(4326) WGS 84',
  'EPSG:900913': '(900913) Google Maps Global Mercator',
  'EPSG:3857': '(3857) WGS 84 / Pseudo-Mercator',
  'EPSG:3785': '(3785 DEPRECATED) Popular Visualization CRS / Mercator',
  'EPSG:32647': '(32647) WGS 84 / UTM zone 47N',
  'EPSG:32736': '(32736) WGS 84 / UTM zone 36S'
}
```

Supported projections human readable descriptions associated to their EPSG Codes. This list will be presented to the user during the upload process whenever GeoNode won't be able to recognize a suitable projection. Those codes should be aligned to the *UPLOADER* ones and available in GeoServer also.

F

FREETEXT_KEYWORDS_READONLY

Default: `False`

Env: `FREETEXT_KEYWORDS_READONLY`

Make Free-Text Keywords writable from users. Or read-only when set to `False`.

G

GEOFENCE_SECURITY_ENABLED

Default: `True` (False is Test is true)

Env: `GEOFENCE_SECURITY_ENABLED`

Whether the geofence security system is used.

GEOIP_PATH

Default: Path to project

Env: `PROJECT_ROOT`

The local path where GeoIPCities.dat is written to. Make sure your user has to have write permissions.

GEONODE_APPS

If enabled contrib apps are used.

GEONODE_CLIENT_LAYER_PREVIEW_LIBRARY

Default: "mapstore"

The library to use for display preview images of layers. The library choices are:

"mapstore" "leaflet" "react"

GEONODE_EXCHANGE

Default:: Exchange("default", type="direct", durable=True)

The definition of Exchanges published by geonode. Find more about Exchanges at [celery docs](#).

GEOSERVER_EXCHANGE

Default:: Exchange("geonode", type="topic", durable=False)

The definition of Exchanges published by GeoServer. Find more about Exchanges at [celery docs](#).

GEOSERVER_LOCATION

Default: http://localhost:8080/geoserver/

Env: GEOSERVER_LOCATION

Url under which GeoServer is available.

GEOSERVER_PUBLIC_HOST

Default: SITE_HOST_NAME (Variable)

Env: GEOSERVER_PUBLIC_HOST

Public hostname under which GeoServer is available.

GEOSERVER_PUBLIC_LOCATION

Default: SITE_HOST_NAME (Variable)

Env: GEOSERVER_PUBLIC_LOCATION

Public location under which GeoServer is available.

GEOSERVER_PUBLIC_PORT

Default: 8080 (Variable)

Env: GEOSERVER_PUBLIC_PORT

Public Port under which GeoServer is available.

GEOSERVER_WEB_UI_LOCATION

Default: GEOSERVER_PUBLIC_LOCATION (Variable)

Env: GEOSERVER_WEB_UI_LOCATION

Public location under which GeoServer is available.

GROUP_PRIVATE_RESOURCES

Default: False

Env: GROUP_PRIVATE_RESOURCES

If this option is enabled, Resources belonging to a Group won't be visible by others

H

HAYSTACK_FACET_COUNTS

Default: True

Env: HAYSTACK_FACET_COUNTS

If set to True users will be presented with feedback about the number of resources which matches terms they may be interested in.

HAYSTACK_SEARCH

Default: False

Env: HAYSTACK_SEARCH

Enable/disable haystack Search Backend Configuration.

L

LEAFLET_CONFIG

A dictionary used for Leaflet configuration.

LICENSES

Default::

```
{ 'ENABLED': True, 'DETAIL': 'above', 'METADATA': 'verbose',  
  }
```

Enable Licenses User Interface

LOCAL_SIGNALS_BROKER_URL

Default: `memory://`

LOCKDOWN_GEONODE

Default: `False`

Env: `LOCKDOWN_GEONODE`

By default, the GeoNode application allows visitors to view most pages without being authenticated. If this is set to `True` users must be authenticated before accessing URL routes not included in `AUTH_EXEMPT_URLS`.

LOGIN_URL

Default: `{%account/login/'}.format(SITEURL)`

Env: `LOGIN_URL`

The URL where requests are redirected for login.

LOGOUT_URL

Default: `{%account/login/'}.format(SITEURL)`

Env: `LOGOUT_URL`

The URL where requests are redirected for logout.

M

MAP_CLIENT_USE_CROSS_ORIGIN_CREDENTIALS

Default: `False`

Env: `MAP_CLIENT_USE_CROSS_ORIGIN_CREDENTIALS`

Enables cross origin requests for geonode-client.

MAPSTORE_BASELAYERS

Default:

```
[
  {
    "type": "osm",
    "title": "Open Street Map",
    "name": "mapnik",
    "source": "osm",
    "group": "background",
    "visibility": True
  }, {
    "type": "tileprovider",
```

(continues on next page)

(continued from previous page)

```

        "title": "OpenTopoMap",
        "provider": "OpenTopoMap",
        "name": "OpenTopoMap",
        "source": "OpenTopoMap",
        "group": "background",
        "visibility": False
    }, {
        "type": "wms",
        "title": "Sentinel-2 cloudless - https://s2maps.eu",
        "format": "image/jpeg",
        "id": "s2cloudless",
        "name": "s2cloudless:s2cloudless",
        "url": "https://maps.geo-solutions.it/geoserver/wms",
        "group": "background",
        "thumbURL": "%static/mapstorestyle/img/s2cloudless-s2cloudless.png"
↪ % SITEURL,
        "visibility": False
    }, {
        "source": "ol",
        "group": "background",
        "id": "none",
        "name": "empty",
        "title": "Empty Background",
        "type": "empty",
        "visibility": False,
        "args": ["Empty Background", {"visibility": False}]
    }
]

```

Env: MAPSTORE_BASELAYERS

Allows to specify which backgrounds MapStore should use. The parameter `visibility` for a layer, specifies which one is the default one.

A sample configuration using the Bing background without OpenStreetMap, could be the following one:

```

[
  {
    "type": "bing",
    "title": "Bing Aerial",
    "name": "AerialWithLabels",
    "source": "bing",
    "group": "background",
    "apiKey": "{{apiKey}}",
    "visibility": True
  }, {
    "type": "tileprovider",
    "title": "OpenTopoMap",
    "provider": "OpenTopoMap",
    "name": "OpenTopoMap",
    "source": "OpenTopoMap",
    "group": "background",
    "visibility": False
  }, {

```

(continues on next page)

(continued from previous page)

```

        "type": "wms",
        "title": "Sentinel-2 cloudless - https://s2maps.eu",
        "format": "image/jpeg",
        "id": "s2cloudless",
        "name": "s2cloudless:s2cloudless",
        "url": "https://maps.geo-solutions.it/geoserver/wms",
        "group": "background",
        "thumbURL": "%sstatic/mapstorestyle/img/s2cloudless-s2cloudless.png"
    ↪% SITEURL,
        "visibility": False
    }, {
        "source": "ol",
        "group": "background",
        "id": "none",
        "name": "empty",
        "title": "Empty Background",
        "type": "empty",
        "visibility": False,
        "args": ["Empty Background", {"visibility": False}]
    }
]

```

Warning: To use a Bing background, you need to correctly set and provide a valid BING_API_KEY

MAX_DOCUMENT_SIZE

Default: 2

Env: MAX_DOCUMENT_SIZE

Allowed size for documents in MB.

MISSING_THUMBNAIL

Default: geonode/img/missing_thumb.png

The path to an image used as thumbnail placeholder.

MODIFY_TOPICCATEGORY

Default: False

Metadata Topic Categories list should not be modified, as it is strictly defined by ISO (See: <http://www.isotc211.org/2005/resources/Codelist/gmxCodellists.xml> and check the <CodeListDictionary gml:id="MD_MD_TopicCategoryCode"> element).

Some customization is still possible changing the is_choice and the GeoNode description fields.

In case it is necessary to add/delete/update categories, it is possible to set the MODIFY_TOPICCATEGORY setting to True.

MONITORING_ENABLED

Default: `False`

Enable internal monitoring application (*geonode.monitoring*). If set to *True*, add following code to your local settings:

```
MONITORING_ENABLED = True
# add following lines to your local settings to enable monitoring
if MONITORING_ENABLED:
    INSTALLED_APPS + ('geonode.monitoring',)
    MIDDLEWARE_CLASSES + ('geonode.monitoring.middleware.MonitoringMiddleware
    ↪',)
```

See *Read-Only and Maintenance Mode* for details.

MONITORING_DATA_AGGREGATION

Default:

```
(
    (timedelta(seconds=0), timedelta(minutes=1)),
    (timedelta(days=1), timedelta(minutes=60)),
    (timedelta(days=14), timedelta(days=1)),
)
```

Configure aggregation of past data to control data resolution. It lists data age and aggregation in reverse order, by default:

- for current data, 1 minute resolution
- for data older than 1 day, 1-hour resolution
- for data older than 2 weeks, 1 day resolution

See *Read-Only and Maintenance Mode* for further details.

This setting takes effects only if *USER_ANALYTICS_ENABLED* is true.

MONITORING_DATA_TTL

Default: 365

Env: `MONITORING_DATA_TTL`

How long monitoring data should be stored in days.

MONITORING_DISABLE_CSRF

Default: False

Env: MONITORING_DISABLE_CSRF

Set this to true to disable csrf check for notification config views, use with caution - for dev purpose only.

MONITORING_SKIP_PATHS

Default:

```
(
    '/api/o/',
    '/monitoring/',
    '/admin',
    '/jsi18n',
    STATIC_URL,
    MEDIA_URL,
    re.compile('^/[a-z]{2}/admin/'),
)
```

Skip certain useless paths to not to mud analytics stats too much. See *Read-Only and Maintenance Mode* to learn more about it.

This setting takes effects only if *USER_ANALYTICS_ENABLED* is true.

N

NOTIFICATIONS_MODULE

Default: `pinax.notifications`

App used for notifications. (pinax.notifications or notification)

NOTIFICATION_ENABLED

Default: True

Env: NOTIFICATION_ENABLED

Enable or disable the notification system.

O

OAuth2_API_KEY

Default: None

Env: OAUTH2_API_KEY

In order to protect oauth2 REST endpoints, used by GeoServer to fetch user roles and infos, you should set this key and configure the geonode REST role service accordingly. Keep it secret!

Warning: If not set, the endpoint can be accessed by users without authorization.

OAuth2_PROVIDER

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_APPLICATION_MODEL

Default: `oauth2_provider.Application`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_ACCESS_TOKEN_MODEL

Default: `oauth2_provider.AccessToken`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_ID_TOKEN_MODEL

Default: `oauth2_provider.IDToken`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_GRANT_MODEL

Default: `oauth2_provider.Grant`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_REFRESH_TOKEN_MODEL

Default: `oauth2_provider.RefreshToken`

Ref.: [OAuth Toolkit settings](#)

OGC_SERVER_DEFAULT_PASSWORD

Default: `geoserver`

Env: `GEOSERVER_ADMIN_PASSWORD`

The geoserver password.

OGC_SERVER_DEFAULT_USER

Default: admin

Env: GEOSERVER_ADMIN_USER

The GeoServer user.

OGC_SERVER

Default: {} (Empty dictionary)

A dictionary of OGC servers and their options. The main server should be listed in the 'default' key. If there is no 'default' key or if the OGC_SERVER setting does not exist, Geonode will raise an Improperly Configured exception. Below is an example of the OGC_SERVER setting:

```
OGC_SERVER = {
    'default' : {
        'LOCATION' : 'http://localhost:8080/geoserver/',
        'USER' : 'admin',
        'PASSWORD' : 'geoserver',
    }
}
```

- BACKEND

Default: "geonode.geoserver"

The OGC server backend to use. The backend choices are:

'geonode.geoserver'

- BACKEND_WRITE_ENABLED

Default: True

Specifies whether the OGC server can be written to. If False, actions that modify data on the OGC server will not execute.

- DATASTORE

Default: '' (Empty string)

An optional string that represents the name of a vector datastore, where Geonode uploads are imported into. To support vector datastore imports there also needs to be an entry for the datastore in the DATABASES dictionary with the same name. Example:

```
OGC_SERVER = {
    'default' : {
        'LOCATION' : 'http://localhost:8080/geoserver/',
        'USER' : 'admin',
        'PASSWORD' : 'geoserver',
        'DATASTORE': 'geonode_imports'
    }
}

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'development.db',
```

(continues on next page)

(continued from previous page)

```
},
'geonode_imports' : {
    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    'NAME': 'geonode_imports',
    'USER' : 'geonode_user',
    'PASSWORD' : 'a_password',
    'HOST' : 'localhost',
    'PORT' : '5432',
}
}
```

- **GEONODE_SECURITY_ENABLED**

Default: True

A boolean that represents whether GeoNode's security application is enabled.

- **LOCATION**

Default: "http://localhost:8080/geoserver/"

A base URL from which GeoNode can construct OGC service URLs. If using GeoServer you can determine this by visiting the GeoServer administration home page without the /web/ at the end. For example, if your GeoServer administration app is at <http://example.com/geoserver/web/>, your server's location is <http://example.com/geoserver>.

- **MAPFISH_PRINT_ENABLED**

Default: True

A boolean that represents whether the MapFish printing extension is enabled on the server.

- **PASSWORD**

Default: 'geoserver'

The administrative password for the OGC server as a string.

- **PRINT_NG_ENABLED**

Default: True

A boolean that represents whether printing of maps and layers is enabled.

- **PUBLIC_LOCATION**

Default: "http://localhost:8080/geoserver/"

The URL used to in most public requests from Geonode. This setting allows a user to write to one OGC server (the LOCATION setting) and read from a separate server or the PUBLIC_LOCATION.

- **USER**

Default: 'admin'

The administrative username for the OGC server as a string.

- **WMST_ENABLED**

Default: False

Not implemented.

- **WPS_ENABLED**

Default: False

Not implemented.

- **TIMEOUT**

Default: 10

The maximum time, in seconds, to wait for the server to respond.

OGP_URL

Default: `http://geodata.tufts.edu/solr/select`

Env: `OGP_URL`

Endpoint of `geodata.tufts.edu` `getCapabilities`.

OPENGRAPH_ENABLED

Default:: `True`

A boolean that specifies whether Open Graph is enabled. Open Graph is used by Facebook and Slack.

P

PINAX_NOTIFICATIONS_BACKENDS

Default: `("email", _EMAIL_BACKEND, 0),`

Used notification backend. This is a [pinax notification setting](#):

PINAX_NOTIFICATIONS_LOCK_WAIT_TIMEOUT

Default: `-1`

Env: `NOTIFICATIONS_LOCK_WAIT_TIMEOUT`

It defines how long to wait for the lock to become available. Default of `-1` means to never wait for the lock to become available. This is a [pinax notification setting](#):

PINAX_NOTIFICATIONS_QUEUE_ALL

Default: `-1`

Env: `NOTIFICATIONS_LOCK_WAIT_TIMEOUT`

By default, calling `notification.send` will send the notification immediately, however, if you set this setting to `True`, then the default behavior of the `send` method will be to queue messages in the database for sending via the `emit_notices` command. This is a [pinax notification setting](#):

PINAX_RATINGS_CATEGORY_CHOICES

Default:

```
{
    "maps.Map": {
        "map": "How good is this map?"
    },
    "layers.Layer": {
        "layer": "How good is this layer?"
    },
    "documents.Document": {
        "document": "How good is this document?"
    }
}
```

PROXY_ALLOWED_HOSTS

Default: () (Empty tuple)

A tuple of strings representing the host/domain names that GeoNode can proxy requests to. This is a security measure to prevent an attacker from using the GeoNode proxy to render malicious code or access internal sites.

Values in this tuple can be fully qualified names (e.g. 'www.geonode.org'), in which case they will be matched against the request's Host header exactly (case-insensitive, not including port). A value beginning with a period can be used as a subdomain wildcard: .geonode.org will match geonode.org, www.geonode.org, and any other subdomain of geonode.org. A value of '*' will match anything and is not recommended for production deployments.

PROXY_URL

Default /proxy/?url=

The URL to a proxy that will be used when making client-side requests in GeoNode. By default, the internal GeoNode proxy is used but administrators may favor using their own, less restrictive proxies.

PYCSW

A dict with pycsw's configuration. Of note are the sections `metadata:main` to set CSW server metadata and `metadata:inspire` to set INSPIRE options. Setting `metadata:inspire['enabled']` to `true` will enable INSPIRE support. Server level configurations can be overridden in the `server` section. See <http://docs.pycsw.org/en/latest/configuration.html> for full pycsw configuration details.

R

RABBITMQ_SIGNALS_BROKER_URL

Default: `amqp://localhost:5672`

The Rabbitmq endpoint

RECAPTCHA_ENABLED

Default: `False`

Env: `RECAPTCHA_ENABLED`

Allows enabling reCaptcha field on signup form. Valid Captcha Public and Private keys will be needed as specified here <https://pypi.org/project/django-recaptcha/#installation>

More options will be available by enabling this setting:

- **ACCOUNT_SIGNUP_FORM_CLASS**

Default: `geonode.people.forms.AllauthReCaptchaSignupForm`

Env: `ACCOUNT_SIGNUP_FORM_CLASS`

Enabled only when the `RECAPTCHA_ENABLED` option is `True`.

- **INSTALLED_APPS**

The captcha must be present on `INSTALLED_APPS`, otherwise you'll get an error.

When enabling the `RECAPTCHA_ENABLED` option through the environment, this setting will be automatically added by GeoNode as follows:

```
if 'captcha' not in INSTALLED_APPS:
    INSTALLED_APPS += ('captcha',)
```

- **RECAPTCHA_PUBLIC_KEY**

Default: `geonode_RECAPTCHA_PUBLIC_KEY`

Env: `RECAPTCHA_PUBLIC_KEY`

In order to generate reCaptcha keys, please see:

1. <https://pypi.org/project/django-recaptcha/#installation>
2. <https://pypi.org/project/django-recaptcha/#local-development-and-functional-testing>

- **RECAPTCHA_PRIVATE_KEY**

Default: `geonode_RECAPTCHA_PRIVATE_KEY`

Env: `RECAPTCHA_PRIVATE_KEY`

In order to generate reCaptcha keys, please see:

1. <https://pypi.org/project/django-recaptcha/#installation>
2. <https://pypi.org/project/django-recaptcha/#local-development-and-functional-testing>

RECAPTCHA_PUBLIC_KEY

Default: `geonode_RECAPTCHA_PUBLIC_KEY`

Env: `RECAPTCHA_PUBLIC_KEY`

Ref. to *RECAPTCHA_ENABLED*

RECAPTCHA_PRIVATE_KEY

Default: `geonode_RECAPTCHA_PRIVATE_KEY`

Env: `RECAPTCHA_PRIVATE_KEY`

Ref. to *RECAPTCHA_ENABLED*

REDIS_SIGNALS_BROKER_URL

Default: `redis://localhost:6379/0`

The Redis endpoint.

REGISTERED_MEMBERS_GROUP_NAME

Default: `registered-members`

Env: `REGISTERED_MEMBERS_GROUP_NAME`

Used by `AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME` settings.

REGISTERED_MEMBERS_GROUP_TITLE

Default: `Registered Members`

Env: `REGISTERED_MEMBERS_GROUP_TITLE`

Used by `AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME` settings.

REGISTRATION_OPEN

Default: `False`

A boolean that specifies whether users can self-register for an account on your site.

RESOURCE_PUBLISHING

Default: `False`

By default, the GeoNode application allows GeoNode staff members to publish/unpublish resources. By default, resources are published when created. When this setting is set to `True` the staff members will be able to unpublish a resource (and eventually publish it back).

S

S3_MEDIA_ENABLED

Default: `False`

Env: `S3_MEDIA_ENABLED`

Enable/disable Amazon S3 media storage.

S3_STATIC_ENABLED

Default: `False`

Env: `S3_STATIC_ENABLED`

Enable/disable Amazon S3 static storage.

SEARCH_FILTERS

Default:

```
'TEXT_ENABLED': True,
'TYPE_ENABLED': True,
'CATEGORIES_ENABLED': True,
'OWNERS_ENABLED': True,
'KEYWORDS_ENABLED': True,
'H_KEYWORDS_ENABLED': True,
'T_KEYWORDS_ENABLED': True,
'DATE_ENABLED': True,
'REGION_ENABLED': True,
'EXTENT_ENABLED': True,
```

Enabled Search Filters for filtering resources.

SECURE_BROWSER_XSS_FILTER

Default: `True`

Env: `SECURE_BROWSER_XSS_FILTER`

If `True`, the `SecurityMiddleware` sets the `X-XSS-Protection: 1; mode=block` header on all responses that do not already have it. This is [Django settings](https://docs.djangoproject.com/en/2.1/ref/settings/#secure-browser-xss-filter). <https://docs.djangoproject.com/en/2.1/ref/settings/#secure-browser-xss-filter>

SECURE_CONTENT_TYPE_NOSNIFF

Default: `True`

Env: `SECURE_CONTENT_TYPE_NOSNIFF`

If `True`, the `SecurityMiddleware` sets the `X-Content-Type-Options: nosniff` header on all responses that do not already have it. This is [Django settings](#):

SECURE_HSTS_INCLUDE_SUBDOMAINS

Default: `True`

Env: `SECURE_HSTS_INCLUDE_SUBDOMAINS`

This is `Django settings`: <https://docs.djangoproject.com/en/2.1/ref/settings/#secure-hsts-include-subdomains>

SECURE_HSTS_SECONDS

Default: `3600`

Env: `SECURE_HSTS_SECONDS`

This is [Django settings](#): If set to a non-zero integer value, the `SecurityMiddleware` sets the HTTP Strict Transport Security header on all responses that do not already have it.

SECURE_SSL_REDIRECT

If `True`, the `SecurityMiddleware` redirects all non-HTTPS requests to HTTPS (except for those URLs matching a regular expression listed in `SECURE_REDIRECT_EXEMPT`). This is [Django settings](#):

SERVICE_UPDATE_INTERVAL

Default: `0`

The Interval services are updated.

SESSION_COOKIE_SECURE

Default: `False`

Env: `SESSION_COOKIE_SECURE`

This is a [Django setting](#):

SESSION_EXPIRED_CONTROL_ENABLED

Default: True

Env: SESSION_EXPIRED_CONTROL_ENABLED

By enabling this variable, a new middleware `geonode.security.middleware.SessionControlMiddleware` will be added to the `MIDDLEWARE_CLASSES`. The class will check every request to GeoNode and it will force a log out whenever one of the following conditions occurs:

1. The OAuth2 Access Token is not valid anymore or it is expired.

Warning: The Access Token might be invalid for various reasons. Usually a misconfiguration of the OAuth2 GeoServer application. The latter is typically installed and configured automatically at GeoNode bootstrap through the default fixtures.

2. The user has been deactivated for some reason; an Admin has disabled it or its password has expired.

Whenever the middleware terminates the session and the user forced to log out, a message will appear to the GeoNode interface.

SHOW_PROFILE_EMAIL

Default: False

A boolean which specifies whether to display the email in the user's profile.

SITE_HOST_NAME

Default: localhost

Env: SITE_HOST_NAME

The hostname used for GeoNode.

SITE_HOST_PORT

Default: 8000

Env: SITE_HOST_PORT

The Site hostport.

SITEURL

Default: 'http://localhost:8000/'

A base URL for use in creating absolute links to Django views and generating links in metadata.

SKIP_PERMS_FILTER

Default: False

Env: SKIP_PERMS_FILTER

If set to true permissions prefiltering is avoided.

SOCIALACCOUNT_ADAPTER

Default: `geonode.people.adapters.SocialAccountAdapter`

This is a [django-allauth setting](#) It allows specifying a custom class to handle authentication for social accounts.

SOCIALACCOUNT_AUTO_SIGNUP

Default: True

Attempt to bypass the signup form by using fields (e.g. username, email) retrieved from the social account provider. This is a [Django-allauth setting](#):

SOCIALACCOUNT_PROVIDERS

Default:

```
{
  'linkedin_oauth2': {
    'SCOPE': [
      'r_emailaddress',
      'r_basicprofile',
    ],
    'PROFILE_FIELDS': [
      'emailAddress',
      'firstName',
      'headline',
      'id',
      'industry',
      'lastName',
      'pictureUrl',
      'positions',
      'publicProfileUrl',
      'location',
      'specialties',
      'summary',
    ]
  },
  'facebook': {
    'METHOD': 'oauth2',
    'SCOPE': [
      'email',
      'public_profile',
    ],
    'FIELDS': [
      'id',
```

(continues on next page)

(continued from previous page)

```

        'email',
        'name',
        'first_name',
        'last_name',
        'verified',
        'locale',
        'timezone',
        'link',
        'gender',
    ]
},
}

```

This is a `django-allauth` setting. It should be a dictionary with provider specific settings

SOCIALACCOUNT_PROFILE_EXTRACTORS

Default:

```

{
    "facebook": "geonode.people.profileextractors.FacebookExtractor",
    "linkedin_oauth2": "geonode.people.profileextractors.LinkedInExtractor",
}

```

A dictionary with provider ids as keys and path to custom profile extractor classes as values.

SOCIAL_BUTTONS

Default: True

A boolean which specifies whether the social media icons and JavaScript should be rendered in GeoNode.

SOCIAL_ORIGINS

Default:

```

SOCIAL_ORIGINS = [{
    "label": "Email",
    "url": "mailto:?subject={name}&body={url}",
    "css_class": "email"
}, {
    "label": "Facebook",
    "url": "http://www.facebook.com/sharer.php?u={url}",
    "css_class": "fb"
}, {
    "label": "Twitter",
    "url": "https://twitter.com/share?url={url}",
    "css_class": "tw"
}, {
    "label": "Google +",
    "url": "https://plus.google.com/share?url={url}",
    "css_class": "gp"
}]

```

A list of dictionaries that are used to generate the social links displayed in the Share tab. For each origin, the name and URL format parameters are replaced by the actual values of the ResourceBase object (layer, map, document).

SOCIALACCOUNT_WITH_GEONODE_LOCAL_SINGUP

Default: `True`

Variable which controls displaying local account registration form. By default form is visible

SRID

Default:

```
{
  'DETAIL': 'never',
}
```

SEARCH_RESOURCES_EXTENDED

Default: `True`

This will extend search with additional properties. By default its on and search engine will check resource title or purpose or abstract. When set to False just title lookup is performed.

T

TASTYPIE_DEFAULT_FORMATS

Default: `json`

This setting allows you to globally configure the list of allowed serialization formats for your entire site. This is a [tastypie setting](#):

THEME_ACCOUNT_CONTACT_EMAIL

Default: `'admin@example.com'`

This email address is added to the bottom of the password reset page in case users have trouble unlocking their account.

THESAURI

Default = []

A list of Keywords thesauri settings: For example *THESAURI* = [{*'name': 'inspire_themes', 'required': True, 'filter': True*}, {*'name': 'inspire_concepts', 'filter': True*},]

TOPICCATEGORY_MANDATORY

Default: False

Env: TOPICCATEGORY_MANDATORY

If this option is enabled, Topic Categories will become strictly Mandatory on Metadata Wizard

TWITTER_CARD

Default:: True

A boolean that specifies whether Twitter cards are enabled.

TWITTER_SITE

Default:: '@GeoNode '

A string that specifies the site to for the twitter:site meta tag for Twitter Cards.

TWITTER_HASHTAGS

Default:: ['geonode']

A list that specifies the hashtags to use when sharing a resource when clicking on a social link.

U

UNOCONV_ENABLE

Default: False

Env: UNOCONV_ENABLE

UPLOADER

Default:

```
{
  'BACKEND' : 'geonode.rest',
  'OPTIONS' : {
    'TIME_ENABLED': False,
  }
}
```

A dictionary of Uploader settings and their values.

- BACKEND

Default: `'geonode.rest'`

The uploader backend to use. The backend choices are:

`'geonode.importer'` `'geonode.rest'`

The importer backend requires the GeoServer importer extension to be enabled.

- OPTIONS

Default:

```
'OPTIONS' : {  
    'TIME_ENABLED': False,  
}
```

- TIME_ENABLED

Default: `False`

A boolean that specifies whether the upload should allow the user to enable time support when uploading data.

USER_MESSAGES_ALLOW_MULTIPLE_RECIPIENTS

Default: `True`

Env: `USER_MESSAGES_ALLOW_MULTIPLE_RECIPIENTS`

Set to true to have multiple recipients in `/message/create/`

USER_ANALYTICS_ENABLED

Default: `False`

Env: `USER_ANALYTICS_ENABLED`

Set to true to anonymously collect user data for analytics. If true you have to set *MONITORING_DATA_AGGREGATION* and *MONITORING_SKIP_PATHS*.

See *Read-Only and Maintenance Mode* to learn more about it.

USER_ANALYTICS_GZIP

Default: `False`

Env: `USER_ANALYTICS_GZIP`

To be used with `USER_ANALYTICS_ENABLED`. Compress `gzip` json messages before sending to `logstash`.

X

X_FRAME_OPTIONS

Default: 'ALLOW-FROM %s' % SITEURL

This is a [Django setting](#)

1.15 Customize the Look and Feel

1.15.1 GeoNode Themes

We have already explained in [Simple Theming](#) how to change the GeoNode theme directly from the *Admin Interface*. This is an easy way for customizing GeoNode appearance but, in some cases, you might want to have more control on it.

In those cases, you have to venture into the code and it is highly recommended to use a GeoNode Project and customize it instead of the GeoNode default HTML/CSS code. See the following sections to learn more about that.

1.15.2 Theming your GeoNode Project

There are a range of options available to you if you want to change the default look and feel of your *GeoNode Project*. Since GeoNode's style is based on [Bootstrap](#) you will be able to make use of all that Bootstrap has to offer in terms of theme customization. You should consult Bootstrap's documentation as your primary guide once you are familiar with how GeoNode implements Bootstrap and how you can override GeoNode's theme and templates in your own project.

Logos and graphics

GeoNode intentionally does not include a large number of graphics files in its interface. This keeps page loading time to a minimum and makes for a more responsive interface. That said, you are free to customize your GeoNode's interface by simply changing the default logo, or by adding your own images and graphics to deliver a GeoNode experience the way you envision it.

Your GeoNode project has a directory already set up for storing your own images at `<my_geonode>/static/img`. You should place any image files that you intend to use for your project in this directory.

Let's walk through an example of the steps necessary to change the default logo.

1. Change to the `img` directory:

```
$ cd <my_geonode>/static/img
```

2. If you haven't already, obtain your logo image. The URL below is just an example, so you will need to change this URL to match the location of your file or copy it to this location:

```
$ sudo wget https://upload.wikimedia.org/wikipedia/commons/thumb/a/ac/Service_
↪mark.svg/500px-Service_mark.svg.png
$ sudo chown -Rf geonode: .
```

3. Change to the `css` directory:

```
$ cd ../../..
```

4. Override the CSS that displays the logo by editing `<my_geonode>/static/css/site_base.css` with your favorite editor and adding the following lines, making sure to update the width, height, and URL to match the specifications of your image.

```
$ sudo vi site_base.css
```

```
.navbar-brand {  
  width: 350px;  
  height: 80px;  
  background: transparent url("../img/500px-Service_mark.svg.png") no-repeat;  
  background-size: 300px 70px;  
  background-position-y: center;  
}
```

5. Restart your GeoNode project and look at the page in your browser:

```
$ cd /home/geonode  
$ sudo rm -Rf geonode/geonode/static_root/*  
$ cd my_geonode  
$ python manage.py collectstatic  
$ sudo service apache2 restart
```

Note: It is a good practice to cleanup the **static_folder** and the Browser Cache before reloading in order to be sure that the changes have been correctly taken and displayed on the screen.

Visit your site at <http://localhost/> or the remote URL for your site.

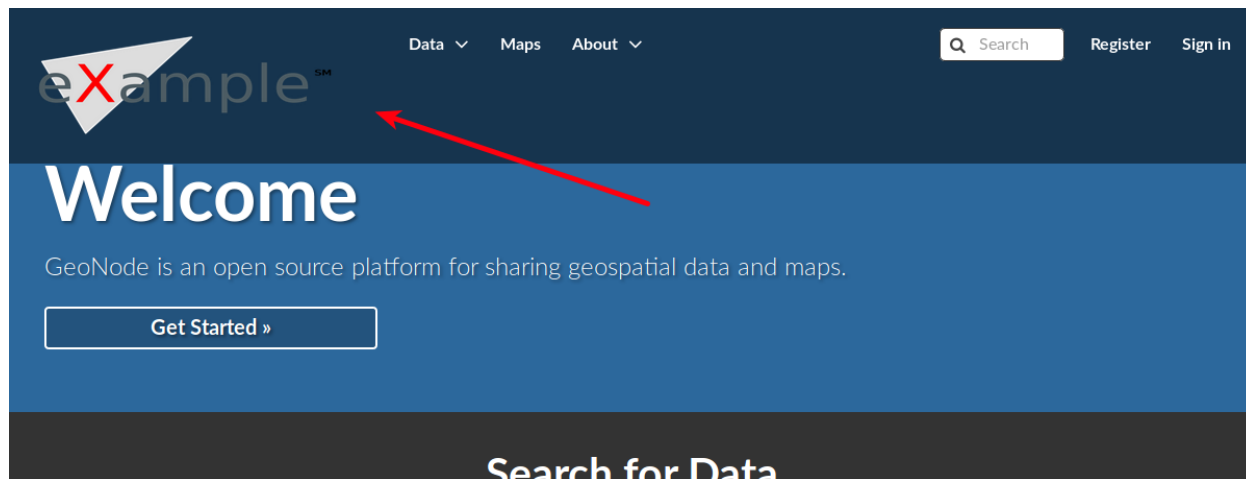


Fig. 221: Custom logo

You can see that the header has been expanded to fit your graphic. In the following sections you will learn how to customize this header to make it as you want.

Note: You should commit these changes to your repository as you progress through this section, and get in the habit of committing early and often so that you and others can track your project on GitHub. Making many atomic commits and staying in sync with a remote repository makes it easier to collaborate with others on your project.

Cascading Style Sheets

In the last section you already learned how to override GeoNode’s default CSS rules to include your own logo. You are able to customize any aspect of GeoNode’s appearance this way. In the last screenshot, you saw that the main area in the homepage is covered up by the expanded header.

First, we’ll walk through the steps necessary to displace it downward so it is no longer hidden, then change the background color of the header to match the color in our logo graphic.

1. Reopen `<my_geonode>/static/css/site_base.css` in your editor:

```
$ cd /home/geonode/my_geonode/my_geonode/static/css
$ sudo vi site_base.css
```

1. Add the following CSS rules to consider the expanded header height:

```
#wrap {
    margin-top: 100px !important;
    padding-top: 0px;
}
```

1. Add a rule to change the background color of the header to match the logo graphic:

```
.navbar-inverse {
    background-color: #ff0000 !important;
}
```

1. Add a background image for the *hero* section:

```
.jumbotron {
    background: url("https://cdn.pixabay.com/photo/2017/09/16/16/09/sea-
↪2755908_960_720.jpg") no-repeat !important;
    background-size: cover !important;
}
```

1. Your project CSS file should now look like this:

```
.navbar-brand {
    width: 350px;
    height: 150px;
    background: transparent url("../img/500px-Service_mark.svg.png") no-
↪repeat;
    background-size: 300px 100px;
    background-position-y: center;
}

#wrap {
    margin-top: 100px !important;
    padding-top: 0px;
}

.navbar-inverse {
    background-color: #ff0000 !important;
}

.jumbotron {
    background: url("https://cdn.pixabay.com/photo/2017/09/16/16/09/sea-
↪2755908_960_720.jpg") no-repeat !important;
```

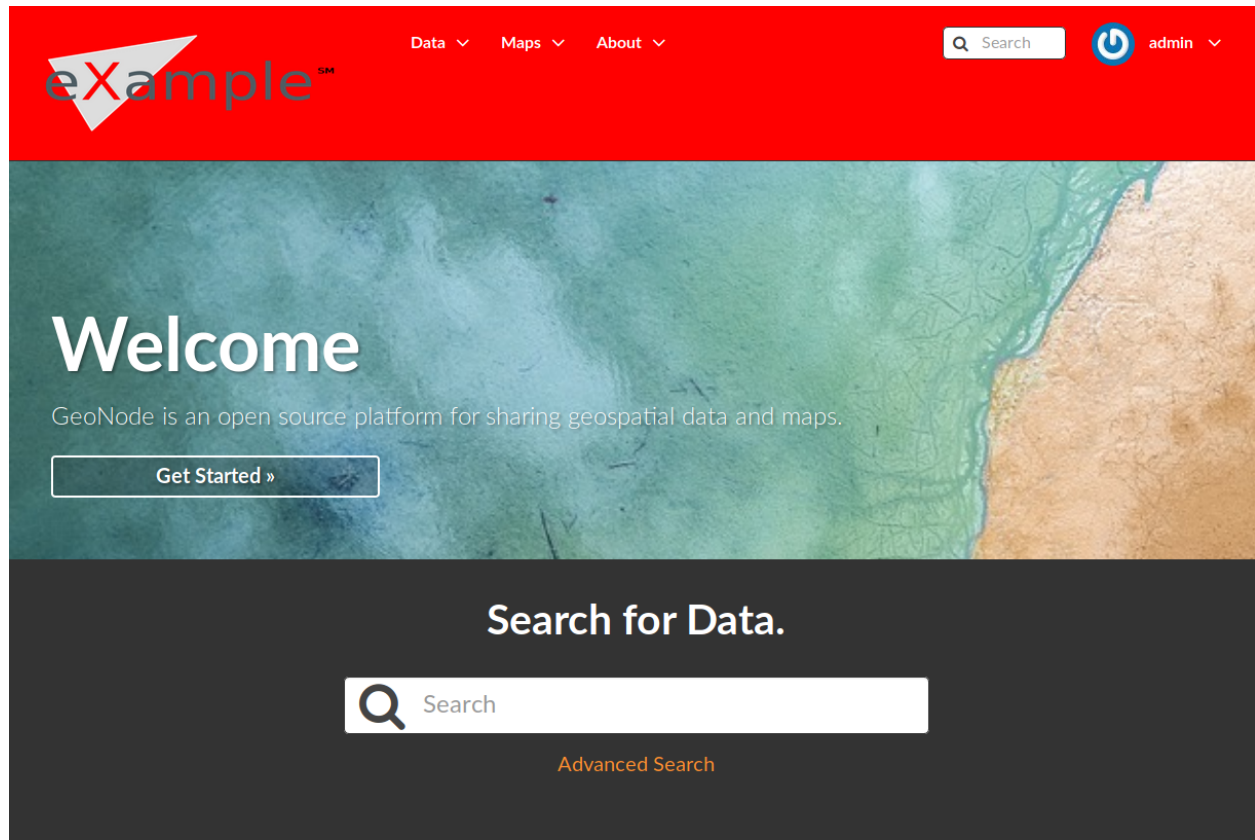
(continues on next page)

(continued from previous page)

```
background-size: cover !important;
}
```

1. Collect the static files into STATIC_ROOT, restart the development server and reload the page:

```
$ python manage.py collectstatic
$ sudo service apache2 restart
```



Discover the available datasets

Fig. 222: CSS override

You can continue adding rules to this file to override the styles that are in the GeoNode base CSS file which is built from [base.less](#).

Note: You may find it helpful to use your browser's development tools to inspect elements of your site that you want to override to determine which rules are already applied. See the screenshot below.

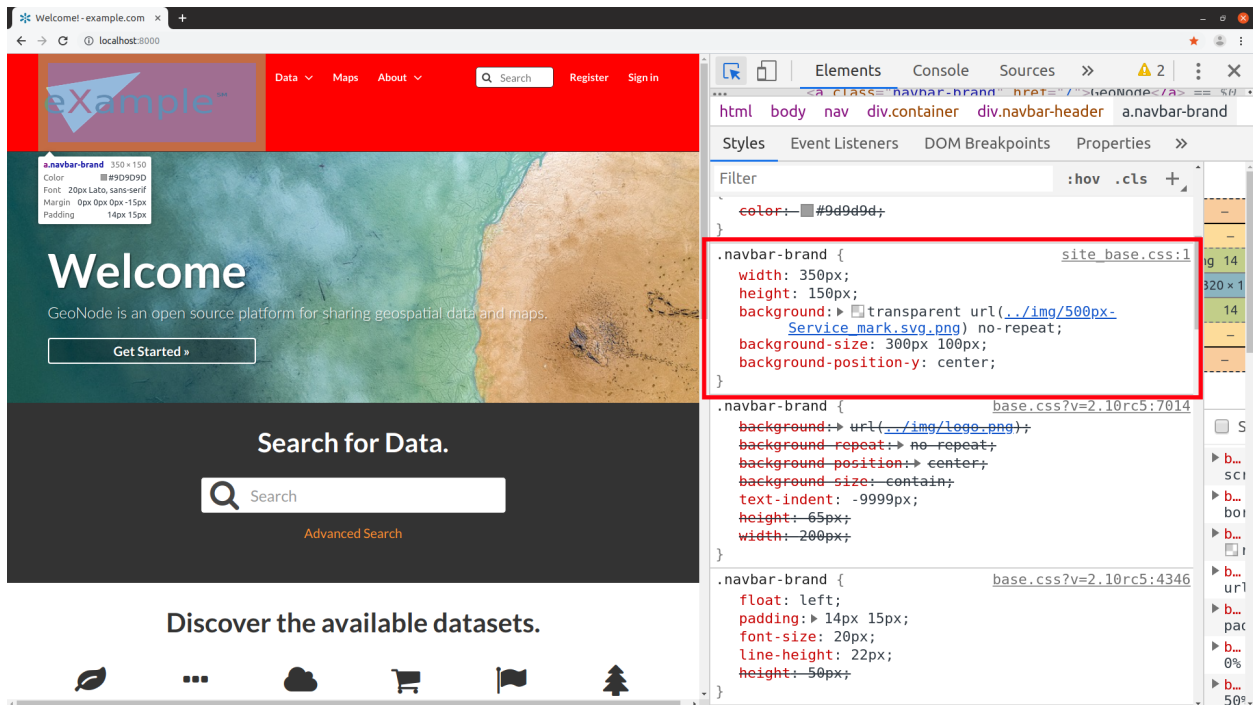


Fig. 223: Screenshot of using browser debugger to inspect the CSS overrides

Templates and static pages

Now that we have changed the default logo and adjusted our main content area to fit the expanded header, the next step is to update the content of the homepage itself. Your GeoNode project includes two basic templates that you will use to change the content of your pages.

The file `site_base.html` (in `<my_geonode>/templates/`) is the basic template that all other templates inherit from and you will use it to update things like the header, navbar, site-wide announcement, footer, and also to include your own JavaScript or other static content included in every page in your site. It's worth taking a look at [GeoNode's base file on GitHub](#). You have several blocks available to you for overriding, but since we will be revisiting this file in future sections of this guide, let's just look at it for now and leave it unmodified.

Open `<my_geonode>/templates/site_base.html` in your editor:

```
$ cd /home/geonode/my_geonode/my_geonode/templates
$ sudo vi site_base.html
```

You will see that it extends from `base.html`, which is the GeoNode template referenced above and it currently only overrides the `extra_head` block to include our project's `site_base.css` which we have modified in the previous section.

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
{% endblock %}
```

You can see on [line 189 of the GeoNode base.html template](#) that this block is included in an empty state and is set up specifically for you to include extra CSS files as your project is already set up to do.

The file `site_index.html` is the template used to define your GeoNode project's homepage. Let's actually override this template.

It extends GeoNode's default `index.html` template and gives you the option to override specific areas of the homepage like the *hero area*, but it also allows you leave other sections as they are. You are of course free to override the sections which you prefer, the following steps give you an example.

1. Open `<my_geonode>/templates/site_index.html` in your editor.
2. Edit the first `<h1>` element inside the `<div class="container">` to say something other than "Welcome":

```
<h1>{{custom_theme.jumbotron_welcome_title|default:_("GeoNode Project Example
↪") }}</h1>
```

Warning: Pay attention to the `custom_theme.jumbotron_welcome_title` part, if you delete it you will cannot use the "admin-based" theme customization option (see [Simple Theming](#))

3. Edit the introductory paragraph to say something about your GeoNode project:

```
<p>
↪ <p>{{custom_theme.jumbotron_welcome_content|default:_("This GeoNode has
↪been customized through my GeoNode Project.")}}</p>
</p>
```

Warning: Take care of the `custom_theme.jumbotron_welcome_content` if you are using the "admin-based" theme customization option (see [Simple Theming](#))

4. Your edited `site_index.html` file should now look like this:

```
{% extends 'index.html' %}
{% load i18n %}

{% comment %}
    This is where you can override the hero area block. You can simply
    ↪modify the content below or replace it wholesale to meet your own needs.
{% endcomment %}

{% block hero %}
    <div class="jumbotron">
        <div class="container">
            <h1>{{custom_theme.jumbotron_welcome_title|default:_("GeoNode
            ↪Project Example") }}</h1>
            <p></p>
            <p>{{custom_theme.jumbotron_welcome_content|default:_("This
            ↪GeoNode has been customized through my GeoNode Project.")}}</p>
            {% if not custom_theme.jumbotron_cta_hide %}
                <p>
                    <a class="btn btn-default btn-lg" target="_blank" role=
                    ↪"button"
                        href="{{custom_theme.jumbotron_cta_link|default:_(
                    ↪'http://docs.geonode.org/en/master/usage/index.html') }}">
                        {{custom_theme.jumbotron_cta_text|default:_("Get
                    ↪Started &raquo;") }}
                </a>
            </p>
        </div>
    </div>
{% endblock %}
```

(continues on next page)

(continued from previous page)

```
        </a>
      </p>
    {% endif %}
  </div>
</div>
{% endblock %}
```

5. Collect the static files into `STATIC_ROOT`, restart the development server and reload the page to see the changes:

```
$ python manage.py collectstatic
$ sudo service apache2 restart
```

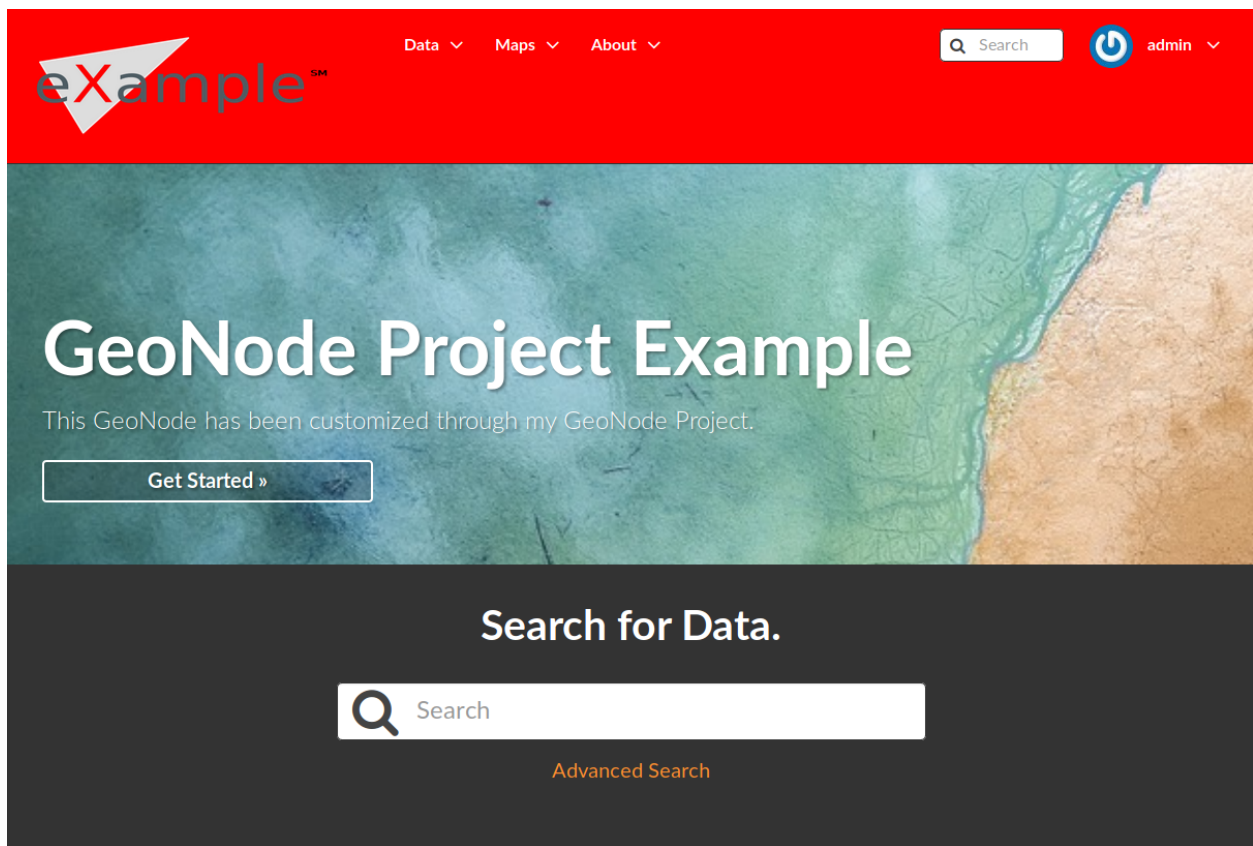


Fig. 224: Customized Geonode Project Home Page

Other theming options

You are able to change any specific piece of your GeoNode project's style by adding CSS rules to `site_base.css`, but since GeoNode is based on Bootstrap, there are many pre-defined themes that you can simply drop into your project to get a whole new look. This is very similar to [WordPress](#) themes and it is a powerful and easy way to change the look of your site without much effort.

Bootswatch

From [Bootswatch](#) you can download ready-to-use themes for Bootstrap-based website.

Warning: Currently GeoNode uses the 3.3.7 version of Bootstrap, so [suitable Bootswatch themes](#) should have been built for the same version.

The following steps will show you how to use a theme from Bootswatch in your own GeoNode Project.

1. Download the [Bootswatch themes for Bootstrap v3.3.7 archive](#) and extract it on some folder in your disk.
2. Select a theme (in this example we will use *Sandstone*) and copy the `bootstrap.css` file inside the theme folder to the `<my_geonode>/static/css` (the static folder of your GeoNode Project).
3. Update the `site_base.html` template to include this file. It should now look like this:

```
$ cd <my_geonode>/templates
$ sudo vi site_base.html
```

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
    <link href="{{ STATIC_URL }}css/bootstrap.css" rel="stylesheet"/>
{% endblock %}
```

5. Collect the static files into `STATIC_ROOT`, restart the development server and reload the page:

```
$ python manage.py collectstatic
$ sudo service apache2 restart
```

1.16 GeoNode permissions

1.16.1 Permissions

Permissions in GeoNode are set per resource, where a resource can be a layer, a map, a document or a service. The way the permissions are set is the same for all of them.

Warning: GeoNode has a set of default permissions that are applied on resource creation **when** you don't explicitly declare them. This is particularly relevant when creating and saving a map, where you won't have the possibility to set its permissions during the creation phase. GeoNode can be tuned to make sure that by default the new created resource are not public, this can be done by changing two settings, see [Default view permissions](#) and [Default download permissions](#)

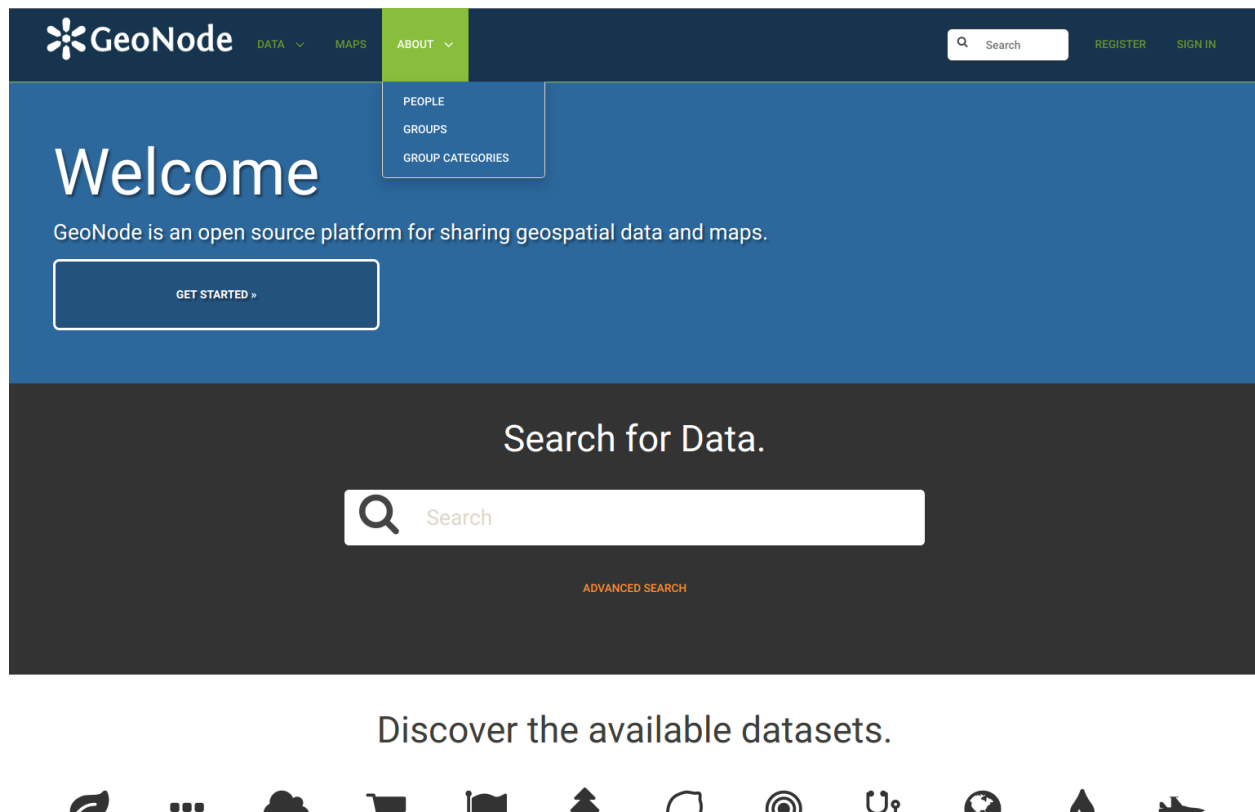


Fig. 225: *Bootstrap Theme for the Geonode Project*

Single Resource permissions

Resource permissions can be generally set from the *resource detail* page. The following figure shows how to open the dialog to set permissions on a layer, the same concept applies to documents and maps.

The dialog for setting the permission allow a granular selection of each permission type to be applied for users and/or groups, each permission type is grouped in tabs that are expanded on click.

The text boxes have an autosuggest feature to help the compilation of user names and groups, it starts upon typing.

You can set the following types of permissions:

- *View* allows to view the layer;
- *Download* allows to download the layer;
- *Change Metadata* allows to change the layer metadata;
- *Edit Data* allows to change attributes and properties of the layers features;
- *Edit Style* allows to change the layer style;
- *Manage* allows to update, delete, change permissions, publish and unpublish the layer.

Warning: When assigning permissions to a group, all the group members will have those permissions. Be careful in case of editing permissions.

Geo Limits permissions

Note: This feature is available **only** when enabling ``GeoServer`` as geospatial backend. Also make sure that the properties ``GEONODE_SECURITY_ENABLED``, ``GEOFENCE_SECURITY_ENABLED`` and ``GEOFENCE_URL`` are correctly set for the ``OGC_SERVER``.

Geo Limits are an extension of the GeoNode standard permissions. *Geo Limits* allows the owner of the resource, or the administrator, to restrict users or groups to a specific geographical area, in order to limit the access to the layer to only the portions contained within that geographic restriction, excluding data outside of it.


In order to be able to set *Geo Limits* you must be an ``administrator`` of the system or the ``owner`` of the resource or you must have ``Manage Permissions`` rights to the resource.

Go to the *Layer Details* page and scroll down to the *Change Layer Permissions* button, as we have seen on the previous section.

If you have the permissions to set the *Geo Limits*, you should be able to see the limits tab beside the permissions one.

You should be able to see an interactive preview of the layers along with few small drawing tools, that allow you to start creating limits on the map manually if you want.

Moreover at the bottom of the panel, there are two other tabs, one listing the available *Users* and another one listing the available *Groups*.


[Data](#)
[Maps](#)
[About](#)

John Smith


[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title Italian Towers

License Not Specified

Abstract No abstract provided

Publication Date June 7, 2019, 4:03 a.m.

Type Vector Data

Keywords features, italian_towers

Category Environment

Regions Global

Owner johnsmith

[More info](#)

[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Default Point

- Red Square

Fig. 226: *Change Layer Permissions*

Set permissions for this resource



Who can view it?
☐ Anyone
The following users:

✕ admin

The following groups:

Choose groups...

Who can download it?
☐ Anyone
The following users:

✕ admin

The following groups:

Choose groups...

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel

Apply Changes

Fig. 227: Resource Permission Dialogue

Set permissions for this resource

People and Groups

Geo Limits

Who can view it?

☒ Anyone

The following users:

× admin

The following groups:

Who can download it?

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel

Apply Changes

Fig. 228: *Geo Limits Tab*

Warning: You will be able to access only *Public* groups and the *Private* ones you belong to.

For each entry of the *Users* and *Groups* tabs, you will have 3 options:

1. Load *Geo Limits*

This button allows you to load the *Geo Limits* already stored on the DB.

Warning: By clicking this button, the geometries present into the map will be cleared. You can add/remove more geometries later on if needed.

2. Upload *Geo Limits*

This button allows you to upload the *Geo Limits* from a ``SHAPEFILE`` on your hard disk. This button **won't** save anything yet. It will **only** load the geometries into the map.

Warning: Be careful using big ``SHAPEFILES``. The geometries will be loaded in memory, and your browser might slow down a lot if you load huge / complex geometries.

Warning: By clicking this button, the geometries present into the map will be cleared. You can add/remove more geometries later on if needed.

3. Save *Geo Limits*

This button allows you to store the *Geo Limits* into the DB. The geometries will be associated to the current ``resource`` and selected ``user`` or ``group``.

Note: By saving the geometries into the DB, the geospatial restrictions won't be applied yet. In order to apply the restrictions you need to:

- a) Set the general permissions to the user / group on the general *Permissions* dialog.
- b) Click on *Apply Changes* button

See the next paragraph for more details.

Once you finished editing your geometries, save them into the DB.

What you have to do now, in order to apply the *Geo Limits* correctly, is to go back to the *Permissions* tab and select *View* and / or *Download* permissions for the users / groups you want to apply the restrictions.

When you are happy with your changes, click on *Apply Changes* button.

The user ``afabiani`` won't be able from now on to access the whole layer data.

Warning: The *Geo Limits* will be persisted on GeoNode DB for that resource. That means that everytime you will update the general permissions, also the geospatial restrictions will be applied.

In order to remove the *Geo Limits* for a certain user or group, you can just *Save* an **empty geometry**. This will **delete** the entry from the DB also.



Fig. 229: *Geo Limits: Preview Window with Drawing Tools*

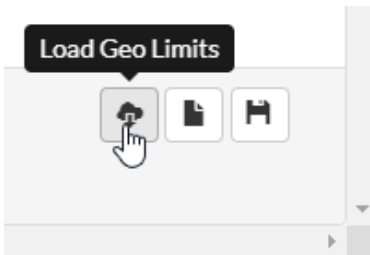


Fig. 230: *Geo Limits: Load from DB*



Fig. 231: *Geo Limits: Upload from a SHAPEFILE*

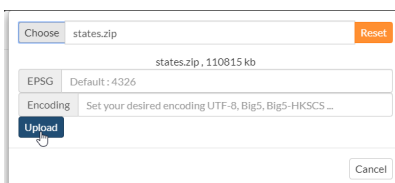


Fig. 232: *Geo Limits: Upload from a SHAPEFILE*

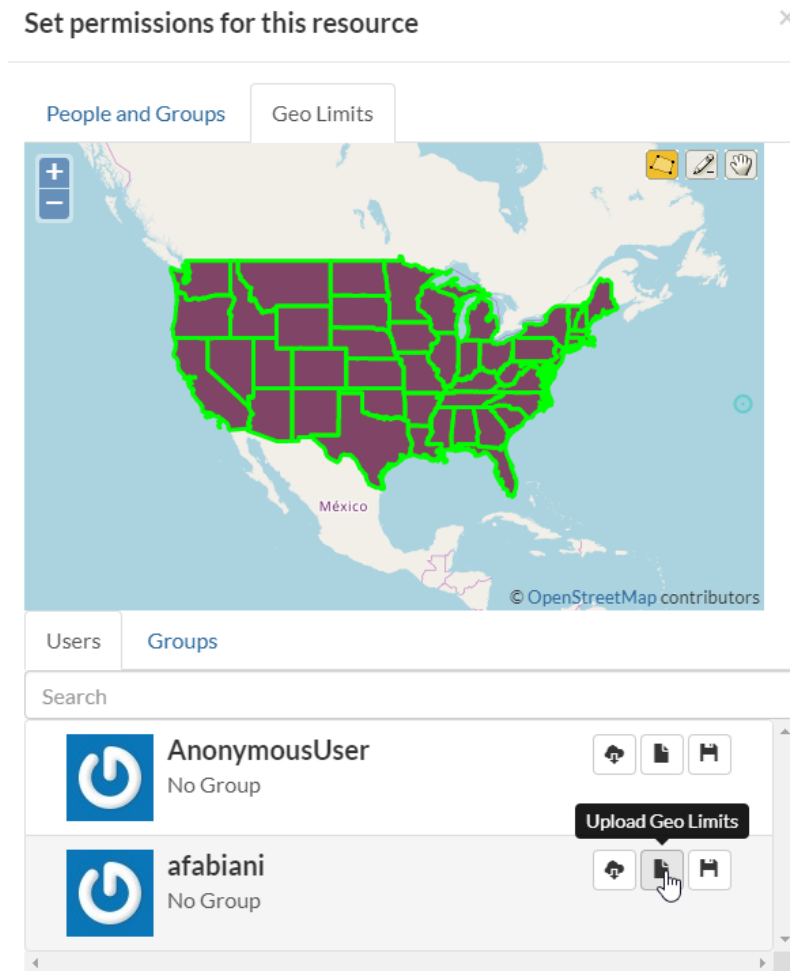


Fig. 233: *Geo Limits: Upload from a SHAPEFILE*

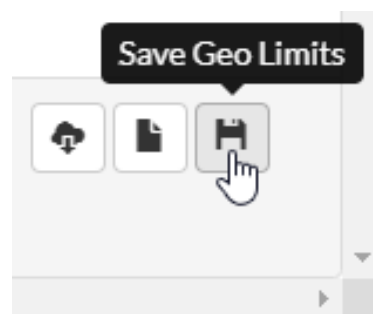
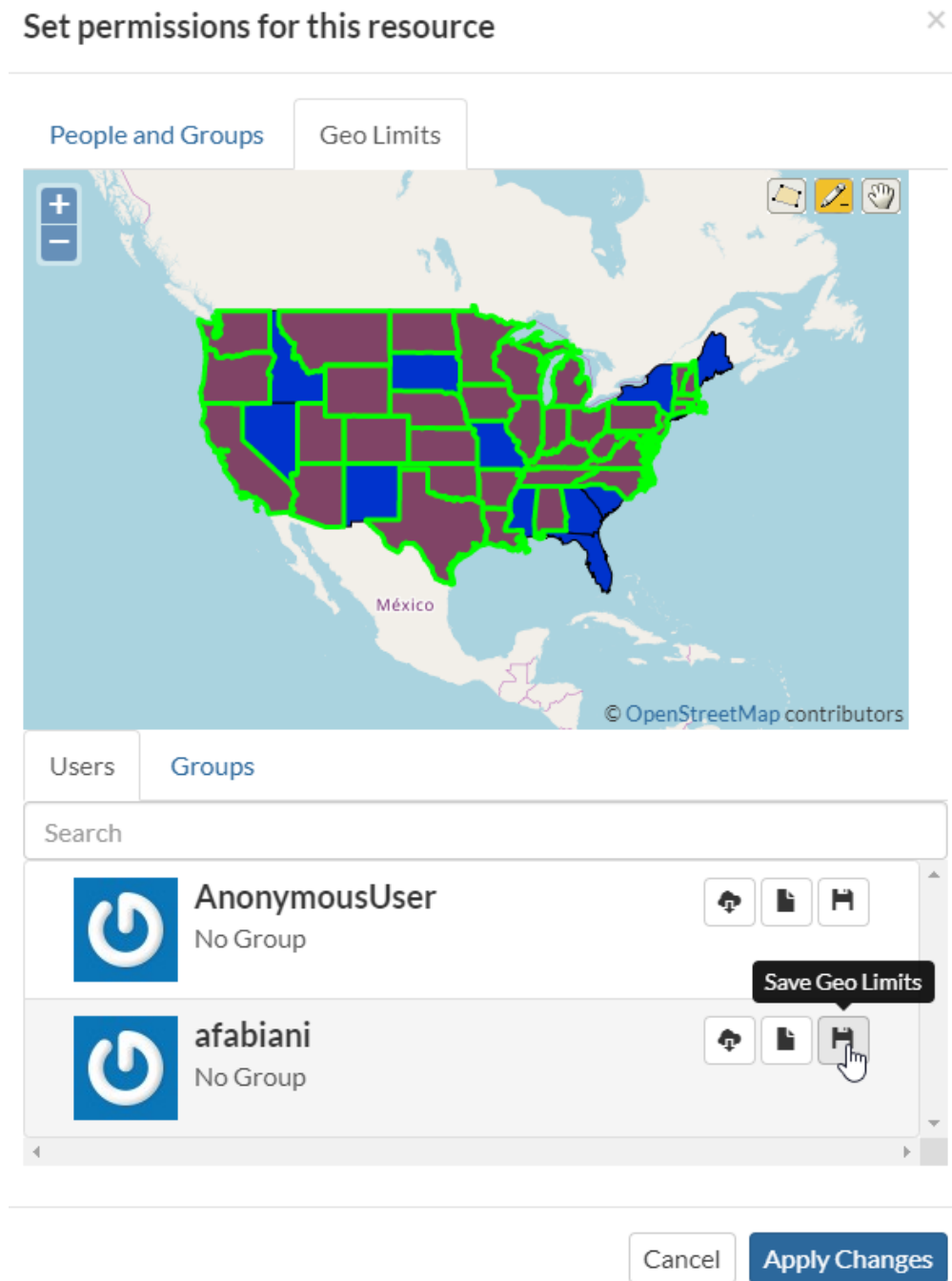


Fig. 234: *Geo Limits: Store the Geo Limits into the DB*

Fig. 235: *Geo Limits: Editing the Geometries*

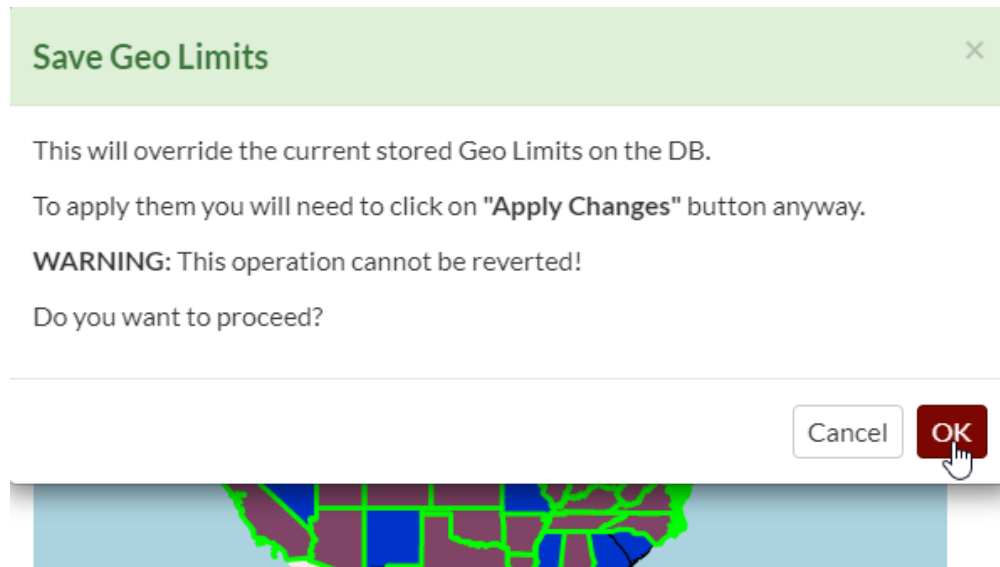


Fig. 236: *Geo Limits: Saving the Geometries for the user afabiani*

Bulk permissions

GeoNode offers the possibility to set permissions in bulk, this can be done in any *list* page.

In order to set bulk permissions you have first to fill the *shopping cart* with the resources you are interested with by clicking the + button on the resource snippet.

Once happy with the selection you can click the *Set Permissions* button under the shopping cart to open the permissions dialogue that will apply the chosen permission to all selected resources.

1.17 Read-Only and Maintenance Mode

1.17.1 Read-Only and Maintenance Modes

Overview

GeoNode gives an option to operate in different modes, according to the needs and demands of the certain application system.

Changing the currently used mode can be done in the admin panel by the user with super-user privileges, by modifying `Configuration` singleton model in the `BASE` application:

Set permissions for this resource ×

People and Groups

Geo Limits

Who can view it?

☒ Anyone

The following users:

× admin

afa

afabiani

Who can download it?

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel

Apply Changes

Fig. 237: *Geo Limits: Set View/Download Permissions for the user afabiani*

Who can download it?

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

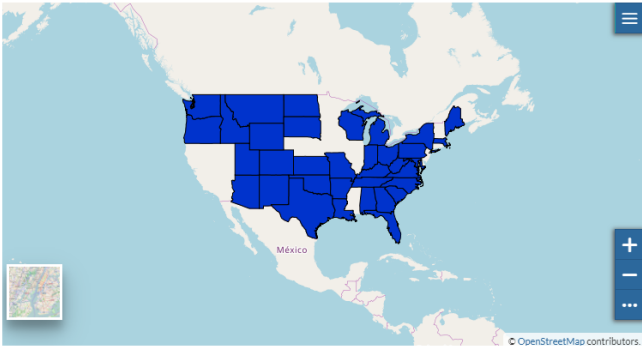
Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel Apply Changes

Fig. 238: *Geo Limits: Apply Permissions and Restrictions to the users and groups*

GeoNode Data Maps About Search afabiani

states



Download Layer

Metadata Detail

View Layer

Download Metadata

Legend

A azure polygon style

■ azure polygon

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

Info Attributes Share Ratings Comments Favorite

Title states

License Not Specified

Fig. 239: *Geo Limits: Geospatial restrictions applies for the user afabiani*



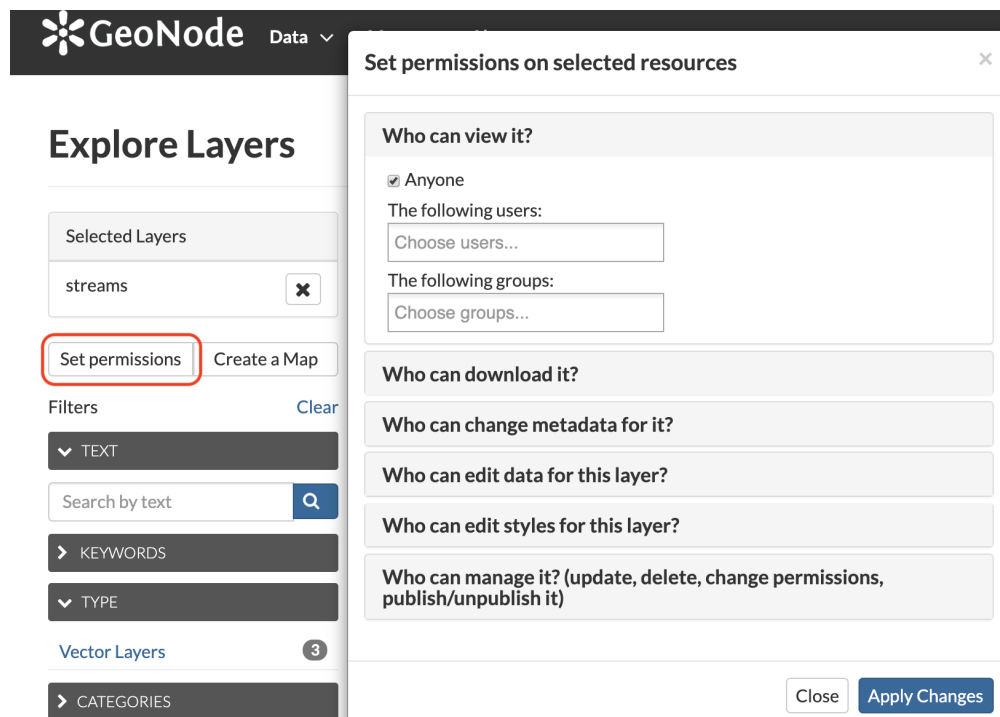
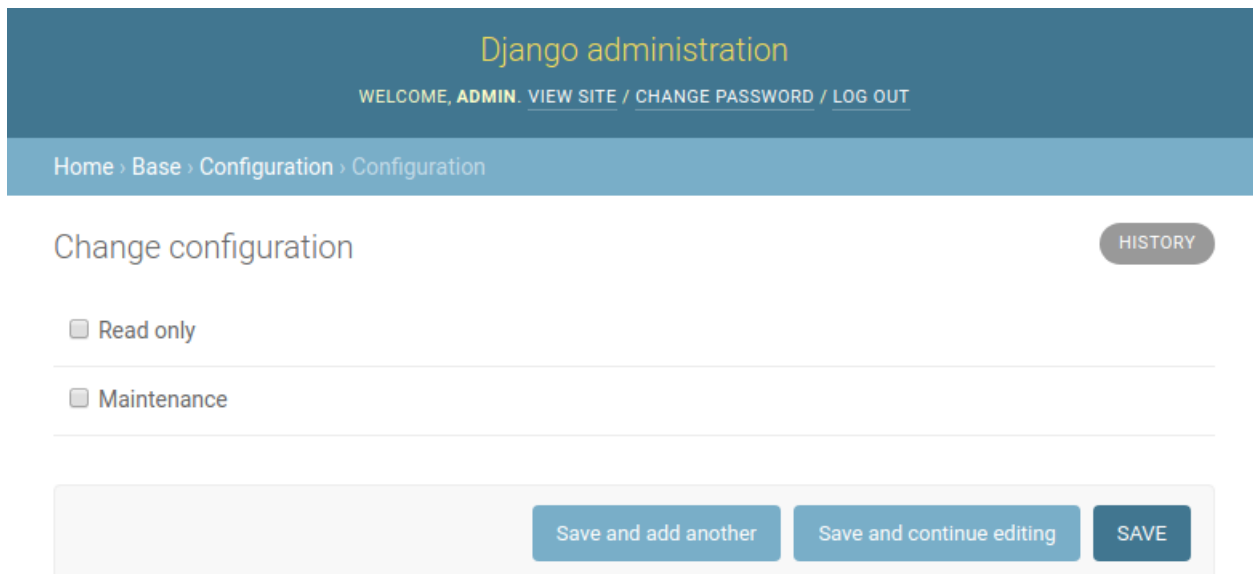
streams

No abstract provided.

simone 12 Jun 2019 0 0 0 Create a Map

+

Fig. 240: *Add Resource To Shopping Cart*

Fig. 241: *Sopping Cart Permissions*Fig. 242: *Configuration change in the admin panel*

Read-Only Mode

Activating the Read-Only Mode (by setting `Read only True` in the `Configuration`) activates a middleware rejecting all modifying requests (POST/PUT/DELETE), with an exception for:

- POST to login view
- POST to logout view
- POST to admin login view
- POST to admin logout view
- all requests to OWS endpoint
- all requests ordered by a super-user

Additionally, all UI elements allowing modifying GeoNode's content are hidden, so e.g. the button "Upload Layer" is not rendered in the templates.

In case a user tries to perform a forbidden request, they will be presented with a static HTML page informing them, the GeoNode is in the Read-Only mode and this action is currently forbidden.

Maintenance Mode

Activating the Maintenance Mode (by setting `Maintenance True` in the `Configuration`) activates the highest level middleware (the one executed as the first) rejecting all requests to the GeoNode instance, with an exception for:

- POST to admin login view
- POST to admin logout view
- all requests ordered by a super-user

In case a user tries to perform any request against the GeoNode (including GET requests), they will be presented with a static HTML page informing them, the maintenance actions are taken on the GeoNode instance, and asking them to try again soon.

The maintenance mode was implemented with a thought of the backup and restore procedures without a necessity to put down the instance, but at the same time with a restriction of any outer interference.

1.18 Monitoring

1.18.1 Monitoring

Internal Monitoring Application (`geonode.monitoring`)

Note: This application requires MaxMind's GeoIP database file.

Base concepts and objects

GeoNode monitoring is a configurable monitoring application, that allows internal resources and hardware resources monitoring for GeoNode installations, including GeoServer deployments.

Monitoring application is configurable, so different deployment scenarios could be handled - from GeoNode and GeoServer running on single host, through distributed installations, where GeoServer is deployed to several hosts.

Monitoring application uses three base entity classes to describe elements of reality: `Host`, `Service Type` and `Service`.

- `Host` is an object describing physical (or virtual) instance of operating system on which GN or GS is running. This object exists only for grouping and is not used directly by monitoring.
- `Service Type` is a description of kind of `Service`. Depending on service type, different metrics are stored, and different data collection mechanisms are used. Additionally, for system monitoring, it's not conducted directly, but with GeoNode or GeoServer as monitoring agent. That means, no additional software installation is needed to monitor system, but also, hosts that don't have GeoNode or GeoServer installed, won't be monitored. There are four service types:
 - `hostgeonode`, `hostgeoserver` - those types describe system monitoring probes that are running with GeoNode or GeoServer respectively,
 - `geonode`, `geoserver` - application-level probes that monitor one specific GeoNode or GeoServer instance.
- `Service` describes one specific instance of probe, either host-level or application-level. Service references `Host` and `Service Type`. Each service must be named, and name should be system-wide unique.

As mentioned above, each `Service Type` keeps a set of `metrics`, specific for that type. A `metric` is a description of measured value, for example: number of requests, response size or time, cpu usage, free memory etc. Each `Service Type` has it's own metrics set. Metric value may be either value counter (like country of user), numeric counter (like number of requests) or rate (like bytes in/out on network interface).

Besides metric data, monitoring will also store exception information for exceptions that were captured during request handling.

Data are collected periodically (at most every 1 minute), aggregated and stored in aggregated form. User can see data from predefined relative periods (last minute, last 10 minutes, last hour, last day, last week).

User can enable and configure automated checks, which will be run after each collection/aggregation cycle, and will emit notifications if metric values in that run exceed configured thresholds.

Analytics

GeoNode monitoring application makes also available information about resources usage at user level.

Those information are collected whenever an event occurs about some resource. Events can be of different types (`EventType`) which refer to common user activities on resources (upload, view, download, etc.). Those data are stored using a dedicated `metric` and aggregated based on a configurable granularity, depending on the time interval considered and the wanted resolution.

So the analytics client, once defined a time interval and a time frame, can retrieve stats such as:

- total number of unique visitors;
- number of unique visitors who trigger a specific type of event;
- number of unique visitors who trigger events on some resource type;
- number of unique visitors in a given country;

- number of unique visitors who trigger events on some specific resource;
- number of unique visitors considering a combination of multiple conditions (for example an event type on some resource type).

Installation

Warning: This plugin requires a Potgresql DB backend enabled

- ensure UTC Timezone to your DB

```
psql -c 'set timezone=UTC;'
```

- enable `MONITORING_ENABLED` flag and ensure that following code is in your settings:

```
# Settings for MONITORING plugin
CORS_ORIGIN_ALLOW_ALL = ast.literal_eval(os.environ.get('CORS_ORIGIN_ALLOW_ALL',
↳ 'False'))
GEOIP_PATH = os.getenv('GEOIP_PATH', os.path.join(PROJECT_ROOT, 'GeoIPCities.dat'))
MONITORING_ENABLED = ast.literal_eval(os.environ.get('MONITORING_ENABLED', 'True'))

MONITORING_CONFIG = os.getenv("MONITORING_CONFIG", None)
MONITORING_HOST_NAME = os.getenv("MONITORING_HOST_NAME", HOSTNAME)
MONITORING_SERVICE_NAME = os.getenv("MONITORING_SERVICE_NAME", 'local-geonode')

# how long monitoring data should be stored
MONITORING_DATA_TTL = timedelta(days=int(os.getenv("MONITORING_DATA_TTL", 7)))

# this will disable csrf check for notification config views,
# use with caution - for dev purpose only
MONITORING_DISABLE_CSRF = ast.literal_eval(os.environ.get('MONITORING_DISABLE_CSRF',
↳ 'False'))

if MONITORING_ENABLED:
    if 'geonode.monitoring' not in INSTALLED_APPS:
        INSTALLED_APPS += ('geonode.monitoring',)
    if 'geonode.monitoring.middleware.MonitoringMiddleware' not in MIDDLEWARE_CLASSES:
        MIDDLEWARE_CLASSES += \
            ('geonode.monitoring.middleware.MonitoringMiddleware',)

# skip certain paths to not to mud stats too much
MONITORING_SKIP_PATHS = ('/api/o/',
                        '/monitoring/',
                        '/admin',
                        '/jsi18n',
                        STATIC_URL,
                        MEDIA_URL,
                        re.compile('^/[a-z]{2}/admin/'),
                        )

# configure aggregation of past data to control data resolution
# list of data age, aggregation, in reverse order
# for current data, 1 minute resolution
# for data older than 1 day, 1-hour resolution
# for data older than 2 weeks, 1 day resolution
```

(continues on next page)

(continued from previous page)

```

MONITORING_DATA_AGGREGATION = (
    (timedelta(seconds=0), timedelta(minutes=1)),
    (timedelta(days=1), timedelta(minutes=60)),
    (timedelta(days=14), timedelta(days=1)),
)

# privacy settings
USER_ANALYTICS_ENABLED = ast.literal_eval(os.getenv('USER_ANALYTICS_ENABLED', 'False
↳'))

```

- run

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py migrate_
↳monitoring

```

to apply db schema changes and insert initial data

- run

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py _
↳updategeoip

```

to fetch MaxMind's GeoIP database file. It will be written to path specified by *GEOIP_PATH* setting.

- run

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py collect_
↳metrics -n -t xml -f --since='<yyyy-mm-dd HH:mm:ss>'

```

to create first metrics.

Warning: Replace `<yyyy-mm-dd HH:mm:ss>` with a real date time to start with.

- update Sites from admin; make sure it contains a correct host name
- do not forget to enable notifications and configure them from user profile

Enable the collect_metrics cron

Warning: Here below you will find instructions for a Ubuntu 16.04/18.04 based machine, but the procedure is similar for other OSs. The basic concept is that you must allow the system to run the command every minute (**without -f and since**):

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py collect_metrics -n -
↳t xml

```

crontab job

```
sudo crontab -e
```

```
# Add the following line at the bottom; this will run the supervisor command every_
↪minute
* * * * * supervisorctl start geonode-monitoring
```

supervisor

```
sudo apt install supervisor
sudo service supervisor restart
sudo update-rc.d supervisor enable
```

```
sudo vim /etc/supervisor/conf.d/geonode-monitoring.conf
```

```
[program:geonode-monitoring]
command=<path_to_virtualenv>/geonode/bin/python -W ignore <path_to_your_project>/
↪geonode/manage.py collect_metrics -n -t xml
directory = <path_to_your_project>
environment=DJANGO_SETTINGS_MODULE="<your_project>.settings"
user=<your_user>
numproc=1
stdout_logfile=/var/log/geonode-celery.log
stderr_logfile=/var/log/geonode-celery.log
autostart = true
autorestart = true
startsecs = 10
stopwaitsecs = 600
priority = 998
```

```
sudo service supervisor restart
sudo supervisorctl start geonode-monitoring
sudo supervisorctl status geonode-monitoring
```

```
sudo vim /etc/hosts
```

```
127.0.0.1      localhost
<public_ip>    <your_host.your_domain> <your_host>

# The following lines are desirable for IPv6 capable hosts
```

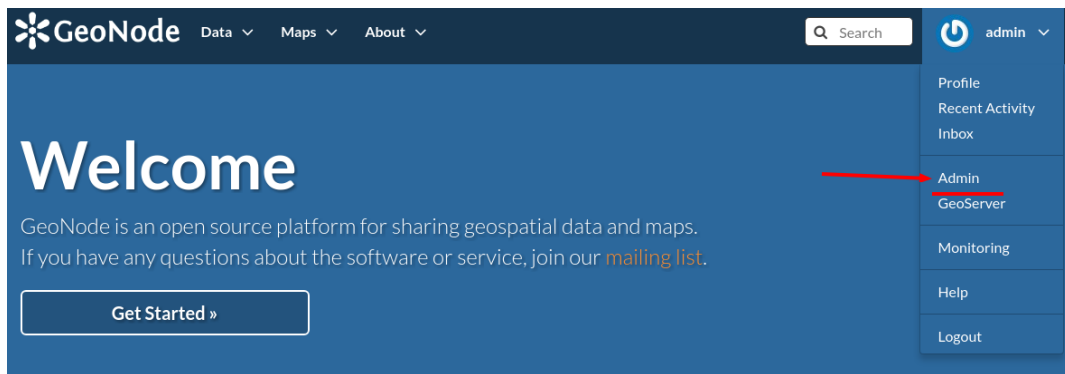
Configuration

In order to have working monitoring, at least `Service` should be configured. Let's assume following deployment scenario:

- there's one machine, `geo01`
- `geo01` hosts both GeoNode and GeoServer (including PostgreSQL).
- applications are served with `nginx+uwsgi`, on port 80, but they are reachable on `localhost` address.
- GeoServer is served from `/geoserver/` path
- GeoNode is served from `/` path

Here's step-by-step instruction how to create monitoring setup for deployment scenario:

1. Log in as admin, and go to admin section:



2. Go to **monitoring** section (or type `/admin/monitoring/` as a path in URL):

Layers	
Attributes	+ Add ✎ Change
Layers	+ Add ✎ Change
Styles	+ Add ✎ Change
Upload sessions	+ Add ✎ Change
Maps	
Map layers	+ Add ✎ Change
Map snapshots	+ Add ✎ Change
Maps	+ Add ✎ Change
Monitoring	
Exception events	+ Add ✎ Change
Hosts	+ Add ✎ Change
Metric labels	+ Add ✎ Change
Metric notification checks	+ Add ✎ Change
Metrics	+ Add ✎ Change
Monitored resources	+ Add ✎ Change

3. Go to **Hosts**:



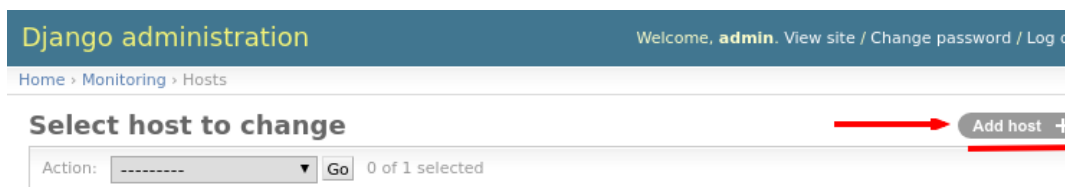
Django administration Welcome, admin.

Home > Monitoring

Monitoring administration

Monitoring	
Exception events	+ Add Change
<u>Hosts</u>	+ Add Change
Metric labels	+ Add Change
Metric notification checks	+ Add Change
Metrics	+ Add Change
Monitored resources	+ Add Change
Notification checks	+ Add Change
Notification metric definitions	+ Add Change
Notification receivers	+ Add Change
Ows services	+ Add Change
Request events	+ Add Change
Service type metrics	+ Add Change
Service types	+ Add Change
<u>Services</u>	+ Add Change

4. Click on **Add host +**:



Django administration Welcome, admin. View site / Change password / Log out

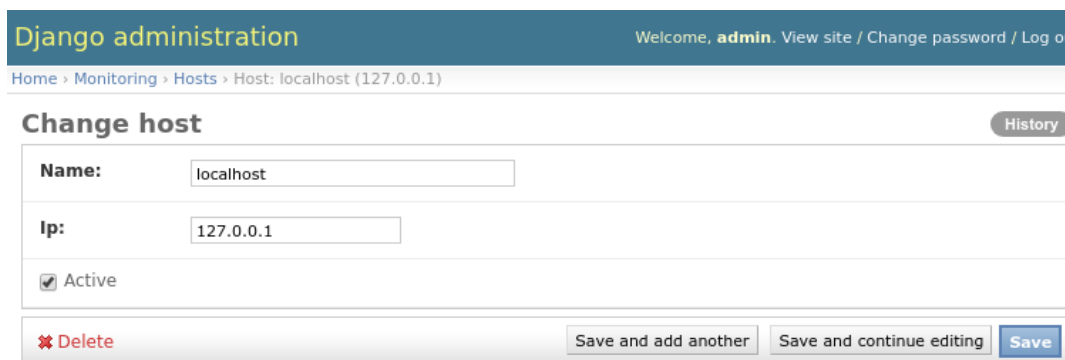
Home > Monitoring > Hosts

Select host to change

[Add host +](#)

Action: 0 of 1 selected

5. Enter following information: * **host**: *localhost* * **ip**: *127.0.0.1* Note, that **host** value is arbitrary. You can enter other name if you like. Don't forget to save.



Django administration Welcome, admin. View site / Change password / Log out

Home > Monitoring > Hosts > Host: localhost (127.0.0.1)

Change host History

Name:

Ip:

☒ Active

[Delete](#) [Save and add another](#) [Save and continue editing](#) [Save](#)

6. Go to **Services**:



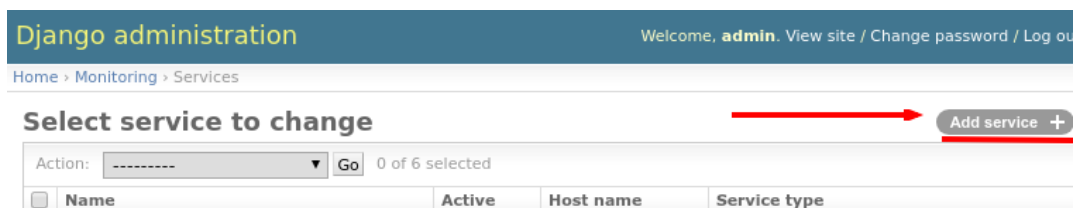
Django administration Welcome, **admin**.

[Home](#) > [Monitoring](#)

Monitoring administration

Monitoring	
Exception events	+ Add ✎ Change
Hosts	+ Add ✎ Change
Metric labels	+ Add ✎ Change
Metric notification checks	+ Add ✎ Change
Metrics	+ Add ✎ Change
Monitored resources	+ Add ✎ Change
Notification checks	+ Add ✎ Change
Notification metric definitions	+ Add ✎ Change
Notification receivers	+ Add ✎ Change
Ows services	+ Add ✎ Change
Request events	+ Add ✎ Change
Service type metrics	+ Add ✎ Change
Service types	+ Add ✎ Change
Services	+ Add ✎ Change

7. Click on **Add service +**:



Django administration Welcome, **admin**. [View site](#) / [Change password](#) / [Log out](#)

[Home](#) > [Monitoring](#) > [Services](#)

Select service to change

[Add service +](#)

Action: 0 of 6 selected

<input type="checkbox"/>	Name	Active	Host name	Service type
--------------------------	------	--------	-----------	--------------

8. Enter following information:



- **name:** *local-geonode*
- **host:** *localhost*
- **service type:** *geonode*

Django administration Welcome, **admin**. [View site](#) / [Change password](#) / [Log out](#)



[Home](#) > [Monitoring](#) > [Services](#) > Service: local-geonode@localhost



Change service History

Name:

Host:  

Check interval:

Last check: **Date:** [Today](#) | 
Time: [Now](#) | 

Service type:  

☒ Active

Notes:

Url:

[✖ Delete](#) [Save and add another](#) [Save and continue editing](#) [Save](#)

9. Add another **Service** Enter following information:



- **name:** *local-system-geonode*
- **host:** *localhost*
- **service type:** *hostgeonode*
- **url:** *http://localhost/* (should point to GeoNode home page)

Django administration Welcome, **admin**. [View site](#) / [Change password](#)



[Home](#) > [Monitoring](#) > [Services](#) > Service: local-system@localhost



Change service

Name:

Host: Host: localhost (127.0.0.1) ▼  

Check interval:

Last check: **Date:** [Today](#)  **Time:** [Now](#) 

Service type: Service Type: hostgeonode ▼  

☒ Active

Notes:

Url: **Currently:** <http://localhost/>
Change:

[✖ Delete](#) [Save and add another](#) [Save and continue editing](#)

10. Add another **Service** and enter following information:



- **name:** *local-geoserver*
- **host:** *localhost*
- **service type:** *geoserver*
- **url:** *http://localhost/geoserver/* (should point to GeoServer home page)

Django administration Welcome, **admin**. [View site](#) / [Change password](#)



[Home](#) > [Monitoring](#) > [Services](#) > Service: local-geoserver@localhost



Change service

Name:

Host:  

Check interval:

Last check: **Date:** [Today](#) | 
Time: [Now](#) | 

Service type:  

☒ Active

Notes:

Url: **Currently:** <http://localhost/geoserver/>
Change:

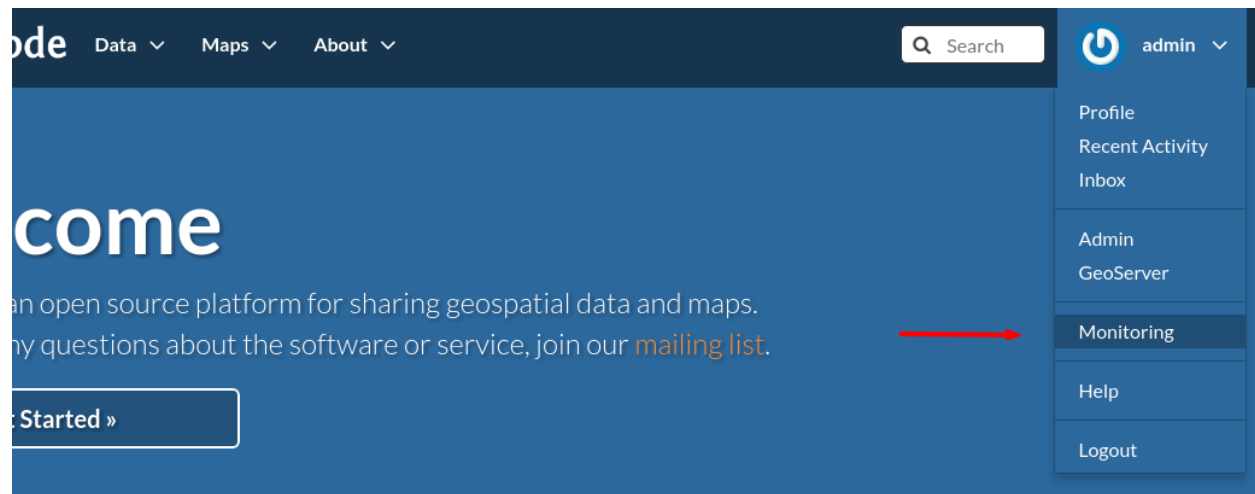
[✖ Delete](#) [Save and add another](#) [Save and continue editing](#)

To summarize, following entries should be created in admin/monitoring:

- Host: localhost, with ip: 127.0.0.1
- **Service: local-geonode:**
 - host localhost
 - type geonode
- **Service: local-geoserver:**
 - url <http://localhost/geoserver/>
 - host localhost
 - type geoserver
- **Service: local-system-geonode**
 - url <http://localhost/>
 - host localhost
 - type hostgeonode

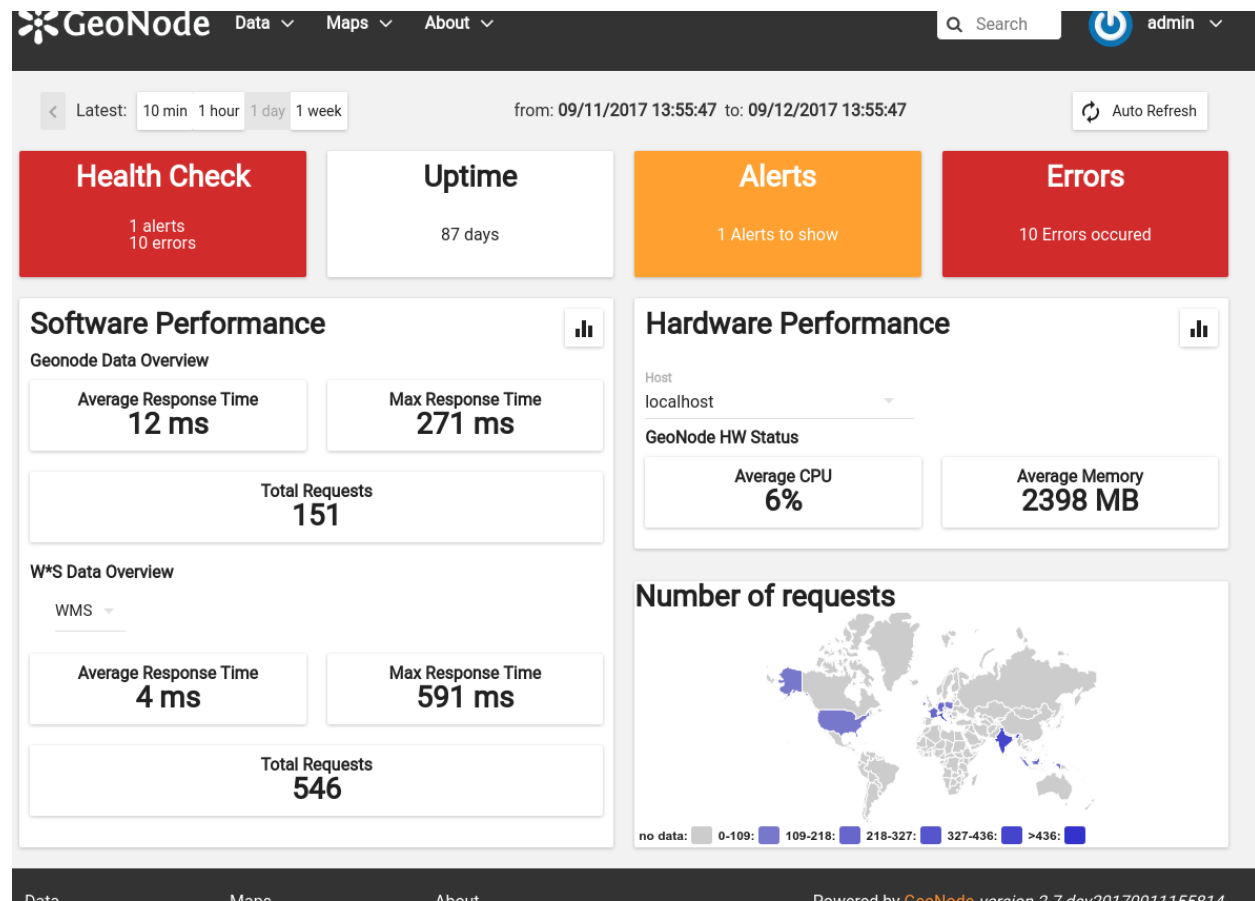
Usage

Monitoring interface is available for superusers only. It's available in profile menu:

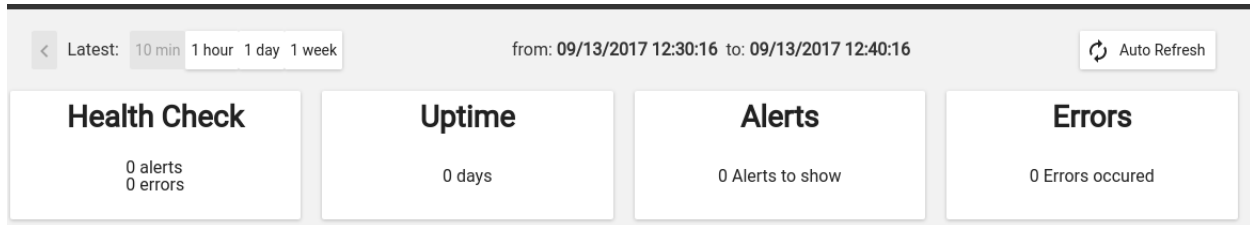


Dashboard

Main view offers overview of recent situation in GeoNode deployment.



Top bar and indicators

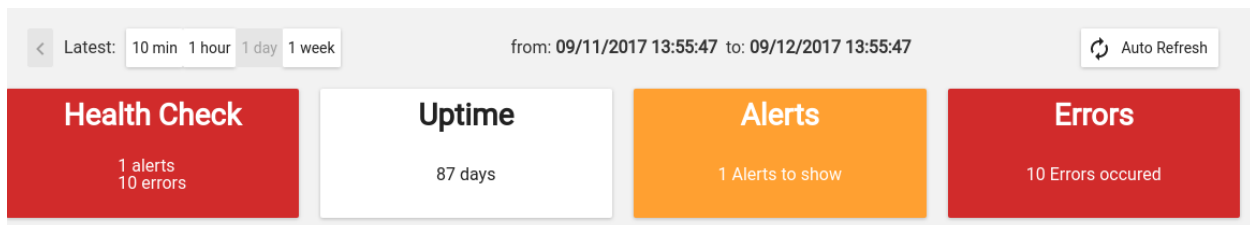


With top bar buttons User can:

- go back from nested interface elements (charts, alerts, errors)
- select time window from which data will be aggregated and shown (last 10 minutes, last 1 hour, last day or last week from now)
- see what's currently used time window
- enable/disable autorefresh

Below there are four main health indicators:

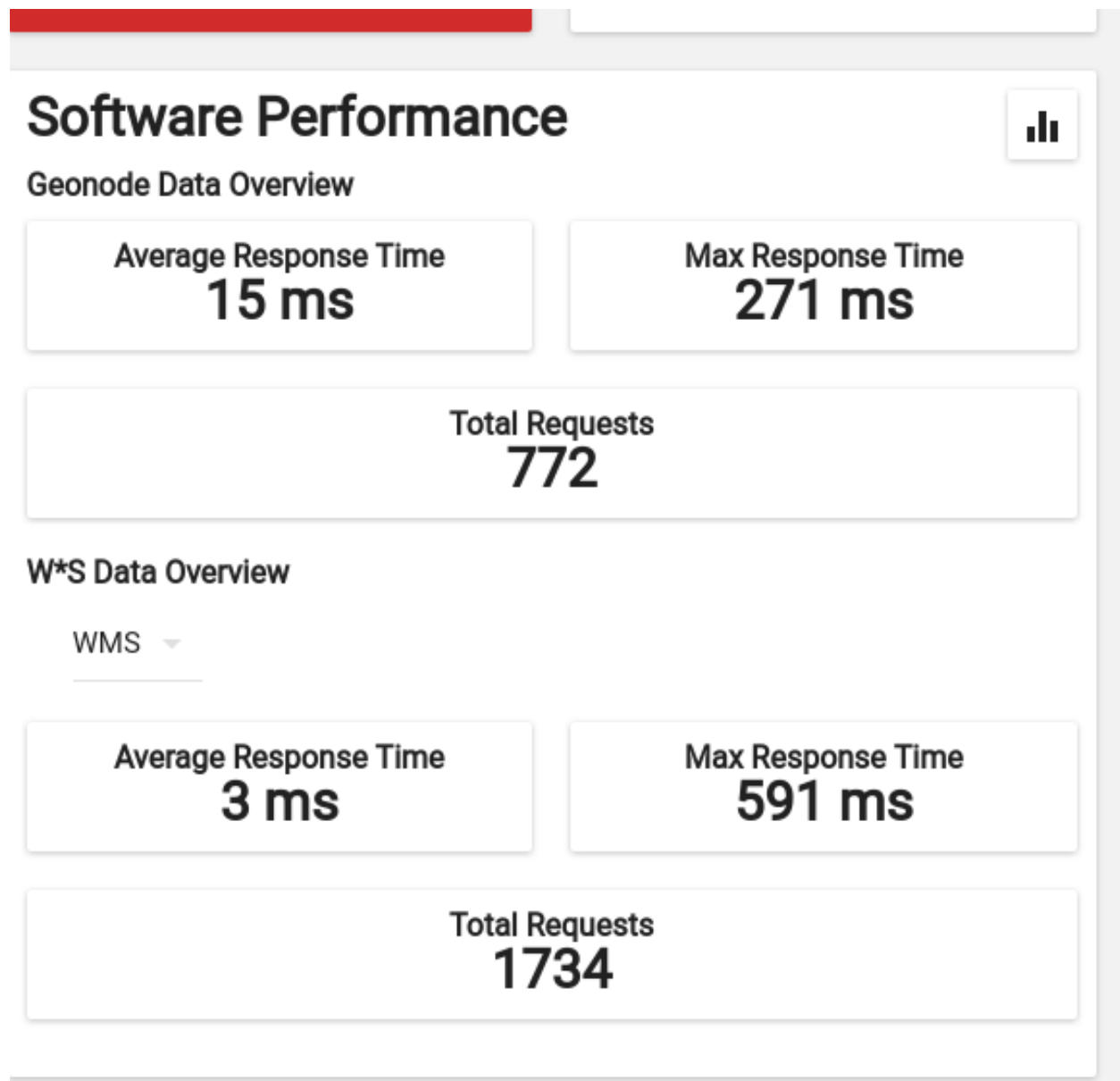
- **aggregated Health Check information.** This element will be:
 - *green* if there is no alerts nor errors
 - *yellow* if there are alerts
 - *red* if there are errors
- **Uptime** that shows GeoNode's system uptime.
- **Alerts** shows number of notifications from defined checks. When clicked, Alerts box will show detailed information. See Notifications description for details.
- **Errors** - shows how many errors were captured during request processing. When clicked, Errors box will show detailed list of captured errors. See Errors description for details.



Indicators in error state

Software Performance

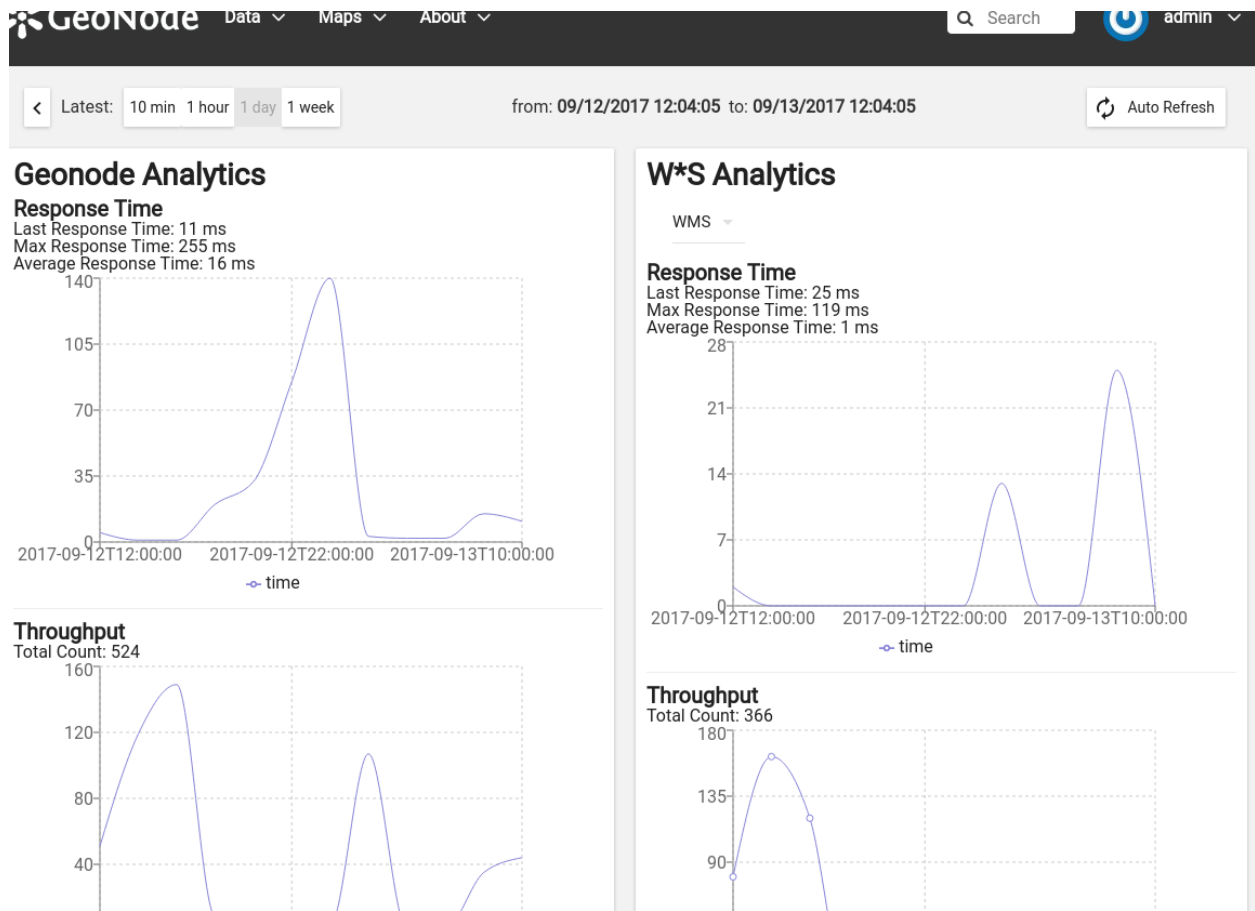
Software Performance view shows GeoServer web service statistics, for all requests monitored and detailed, OWS-specific, per service type (WMS, WFS, OCS etc).



Clicking on




will show charts with data history for overall performance and per-OWS performance:



Hardware Performance

Hardware performance box shows hardware usage statistics for selected host (monitored with any of hostgeonode or hostgeoserver type Service): % of CPU usage and average memory consumption. User can select from which host data will be presented.

017 12:30:16 to: 09/13/2017 12:40:16

 Auto Refresh

Alerts

0 Alerts to show

Errors

0 Errors occurred

Hardware Performance



Host

localhost

GeoNode HW Status

Average CPU
5%

Average Memory
1937 MB

Clicking on

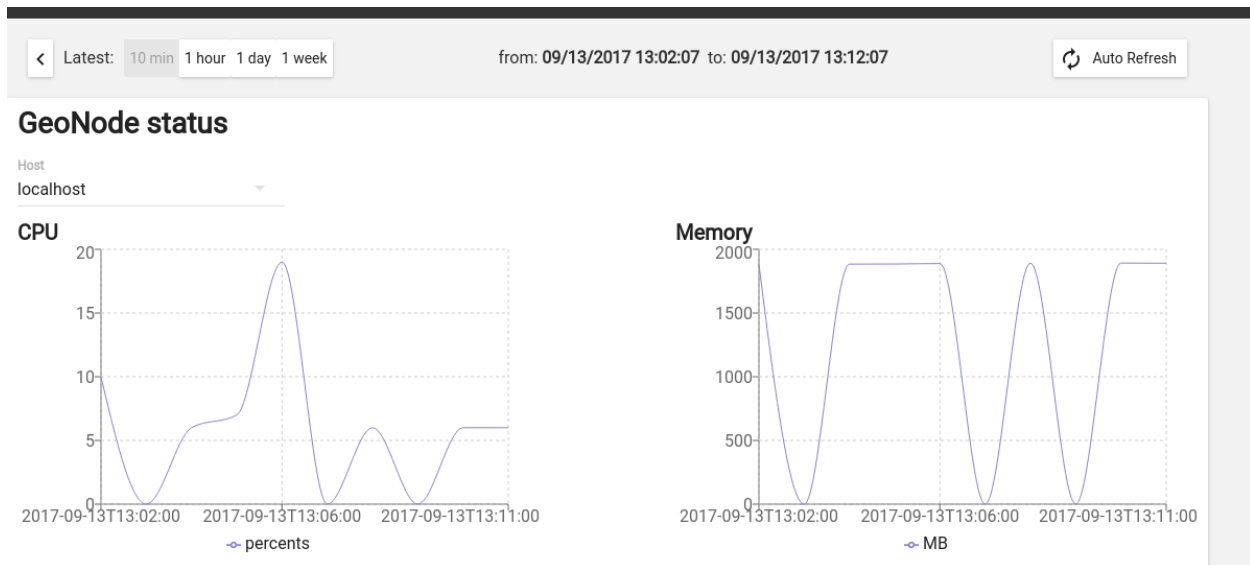
Hardware Performance



Host

localhost

will show charts with data history for selected host and time period



Errors

Errors view will show list of captured errors in GeoNode and GeoServer. List contents is displayed for selected time window.

GeoNode

Latest: 10 min 1 hour 1 day 1 week from: 09/05/2017 16:26:18 to: 09/12/2017 16:26:18 Auto Refresh

Errors

ID	Type	Service	Date
205	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-08T23:58:03.823
206	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-09T06:10:03.556
207	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-10T04:10:03.870
208	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-10T16:20:03.930
209	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T08:16:08.702
210	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T08:16:08.711
211	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T08:16:08.774
212	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T10:14:06.617
213	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T10:14:06.729
214	django.template.base.TemplateSyntax...	local-geonode	2017-09-11T11:10:56.110
215	django.template.base.TemplateSyntax...	local-geonode	2017-09-11T11:11:06.116

For each error, details are available:

- error class, message and stack trace
- basic request context (IP, path, user agent)

<

Latest:

10 min

1 hour

1 day

1 week

from: 09/05/2017 16:26:50 to: 09/12/2017 16:26:50

↺

Auto Refresh

Error id: 206

METADATA	CLIENT
Date: 2017-09-09 06:10:03 Service: demo-geoserver Type: org.geoserver.platform.ServiceException Status code: 200 URL: /geosolutions/wms/reflect?format=application/atom+xml&layers=geosolutions:cleaned&featureid=cleaned.187	

Log

```

org.geoserver.wms.map.GetMapKvpRequestReader.parseLayers(GetMapKvpRequestReader.java:1357)
org.geoserver.wms.map.GetMapKvpRequestReader.read(GetMapKvpRequestReader.java:235)
org.geoserver.wms.map.GetMapKvpRequestReader.read(GetMapKvpRequestReader.java:85)
org.geoserver.ows.Dispatcher.parseRequestKVP(Dispatcher.java:1514)
org.geoserver.ows.Dispatcher.dispatch(Dispatcher.java:680)
org.geoserver.ows.Dispatcher.handleRequestInternal(Dispatcher.java:258)
org.springframework.web.servlet.mvc.AbstractController.handleRequest(AbstractController.java:147)
org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter.handle(SimpleControllerHandlerAdapter.java:50)
org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:959)
org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:893)
org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:968)
org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:859)
javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:844)
javax.servlet.http.HttpServlet.service(HttpServlet.java:722)
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:305)
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
org.geoserver.filters.ThreadLocalsCleanupFilter.doFilter(ThreadLocalsCleanupFilter.java:28)
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:243)
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
org.geoserver.filters.SpringDelegatingFilter$Chain.doFilter(SpringDelegatingFilter.java:75)
org.geoserver.wms.animate.AnimatorFilter.doFilter(AnimatorFilter.java:71)

```

Alerts

An alert is a descriptive information on situation when observed metric contains values outside allowed range (for example, response time is above 30 seconds, or no requests were served within last 30 minutes). Alerts are generated by notifications mechanism described below.

Alerts view will show list of alerts for current moment (alerts that were generated in past are not displayed here):

GeoNode

Data

Maps

About

Search

admin

<

Latest:

10 min

1 hour

1 day

1 week

from: 09/11/2017 13:56:39 to: 09/12/2017 13:56:39

↺

Auto Refresh

Alerts

⚙

Number of handled requests is lower than 800

←

Each alert contains more descriptive information what is wrong:

< Latest: 10 min 1 hour 1 day 1 week from: C

Alerts

Number of handled requests is lower than 800
Number of requests should be at least 800, got 6 instead

Notifications

Notification mechanism (not to be confused with notifications application in GeoNode) is a way to inform selected users about situations, where collected metric data would indicate a problem with deployment. Notifications are accessible from Alerts view:

Alerts

Number of handled requests is lower than 100
2017-09-13 15:31:41.142

There can be several notification configurations available.

Alerts Settings

GeoServer is not working
detects when requests are not handled

Each notification configuration contains two main elements:

- list of email addressess which should be notified when alert is generated
- list of checks (at least one check must be in invalid state to generate alert)

GeoServer is not working

detects when requests are not handled

Who to alert:

When to alert

☒ Number of handled requests is lower than

100 ▾

☒ No response for at least

86400

☒ Response time is higher than

500

s

User can add arbitrary number of emails. Email address doesn't need to point to user registered in GeoNode instance. If email provided doesn't belong to any of users, alert will be send as a regular email. If email provided can be associated with specific user, notifications application (and thus, notification settings for that user) will be used to send alert.

Integration with GeoHealthCheck

GeoNode can also be easily monitored with external tools, like [GeoHealthCheck](#). See [Documentation on adding resources](#) for details.

1.18.2 Monitoring: API

Overview

Geonode monitoring is an optional infrastructure for monitoring resource usage in GeoNode, accompanying GeoServer(s) and hosts on which each service is running. This is not full-fledge monitoring, like zabbix or nagios, rather a moderate size tool to diagnose deployment health. It will be used by users that mostly are not full-time sysops, so usage is simplified.

API

Monitoring API exposes various data to monitoring client.

API root URL is `/monitoring/`, each path in this documentation is relative to that root.

Valid from/valid to

Monitoring collects data periodically, in fixed periods (usually 1 minute). Each metric data is a value (or values if they are split by additional indicators, like resource, label etc) accumulated within that period.

Host

Host is a physical or virtualized instance, on which specific service (GeoNode or GeoServer) is running. This entity is not monitored, but it's used to group services by their deployment location. Hosts list is available in API in `/api/hosts/` endpoint:

```
GET /monitoring/api/hosts/
```

```
{
  "hosts": [
    {
      "ip": "127.0.0.1",
      "name": "localhost"
    }
  ]
}
```

While host is not monitored directly, some service types (and services of those types) are responsible for monitoring underlying host, hardware resources are monitored indirectly (no dedicated system-level agent is needed).

Service

Service is a name of monitored service. Services are configurable from admin interface, and exposed in API in `/api/services/`:

```
GET /monitoring/api/services/
```

```
{
  "services": [
    {
      "name": "local-system",
      "last_check": "2017-08-03T13:33:26.674",
      "host": "localhost",
      "check_interval": 60,
      "type": "hostgeonode",
      "id": 3
    },
    {
      "name": "local-geoserver",
      "last_check": "2017-08-03T13:33:26.455",
      "host": "localhost",
      "check_interval": 60,
      "type": "geoserver",
      "id": 2
    },
    {
      "name": "local-geonode",
      "last_check": "2017-08-03T13:33:27.741",
      "host": "localhost",
      "check_interval": 60,
      "type": "geonode",
      "id": 1
    }
  ]
}
```

Each service is described by properties:

- *name* - unique name of service
- *type* - service type name
- *host* - host on which service is running
- *id* - object id
- *last_check* - timestamp with last check (data collection) on that service
- *check_interval* - interval in seconds, how often data should be collected from this service.

Service type

Service type describes kind of services to which it's assigned. There are several service types available:

- *geonode* - service is a GeoNode instance
- *geoserver* - service is a GeoServer instance
- *hostgeonode* - service is not an application, service is underlying host measured with GeoNode (see [Host](#))
- *hostserver* - service is not an application, service is underlying host measured with GeoServer (see [Host](#))

Resource

Resource is an object that can be served by GeoNode or GeoServer. There are several resource types monitored:

- layer
- document
- map
- url

Resource can be served from either GeoNode or GeoServer. We don't check if specific resource actually exists, just keep list of items used and recorded for monitoring. Also, it won't show renames/copies/moves of the same resource.

Resources list is available in `/api/resources/` endpoint:

GET `/monitoring/api/resources/`

```
{
  "resources": [
    {
      "type": "layer",
      "id": 13,
      "name": "unesco:Unesco_point"
    },
    {
      "type": "layer",
      "id": 7,
      "name": "geonode:test"
    },
    {
      "type": "layer",
      "id": 14,
      "name": "http://www.opengis.net/gml:GridCoverage"
    },
    {

```

(continues on next page)

(continued from previous page)

```

    "type": "map",
    "id": 17,
    "name": "some map"
  },
]
}

```

Resource is described with following attributes:

- *id* - numeric id of resource record in monitoring
- *type* - type of resource
- *name* - name of resource.

Resources list can be filtered with following query sting arguments:

- *metric_name* - name of metric for which resources should be returned
- *resource_type* - name of type of resource (*layer*, *map*, *document*, *style*, *url*)
- *valid_from* - list resources that are available since that timestamp
- *valid_to* - list resources that are available until that timestamp

Example:

GET /monitoring/api/resources/?resource_type=layer&metric_name=request.
count&valid_from=2017-08-01

```

{
  "resources": [
    {
      "type": "layer",
      "id": 24,
      "name": "atlantis:landmarks"
    },
    {
      "type": "layer",
      "id": 2,
      "name": "topp:states"
    },
    {
      "type": "layer",
      "id": 22,
      "name": "atlantis:island"
    },
    {
      "type": "layer",
      "id": 23,
      "name": "atlantis:poi"
    },
    {
      "type": "layer",
      "id": 16,
      "name": "dissolveroad2"
    },
    {
      "type": "layer",
      "id": 21,

```

(continues on next page)

(continued from previous page)

```
    "name": "atlantis:roads"
  }
]
}
```

Resource type

Resource Types describe which types of resource the GeoNode monitoring consider. To retrieve the full list of Resource Types the `/api/resource_types/` is available:

GET `/monitoring/api/resource_types/`

```
{
  "status": "ok",
  "data": {
    "key": "resource_types"
  },
  "errors": {},
  "resource_types": [
    {
      "type": "No resource",
      "name": ""
    },
    {
      "type": "Layer",
      "name": "layer"
    },
    {
      "type": "Map",
      "name": "map"
    },
    {
      "type": "Resource base",
      "name": "resource_base"
    },
    {
      "type": "Document",
      "name": "document"
    },
    {
      "type": "Style",
      "name": "style"
    },
    {
      "type": "Admin",
      "name": "admin"
    },
    {
      "type": "URL",
      "name": "url"
    },
    {
      "type": "Other",
      "name": "other"
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

},
"success": true
}

```

Event Types

Event Types describe the way resources were used in GeoNode. Resource can be accessed as a regular view (through GeoNode, like `/layers/X` url), or through OWS request. Full list of Event Types handled is available in `/api/event_types/` endpoint:

GET `/monitoring/api/event_types/`

```

{
  "status": "ok",
  "errors": { },
  "data": {
    "key": "event_types"
  },
  "event_types": [
    {
      "name": "all"
    },
    {
      "name": "other"
    },
    {
      "name": "download"
    },
    {
      "name": "view"
    },
    {
      "name": "OWS:TMS"
    },
    {
      "name": "OWS:WMS-C"
    },
    {
      "name": "OWS:WMTS"
    },
    {
      "name": "OWS:WCS"
    },
    {
      "name": "OWS:WFS"
    },
    {
      "name": "OWS:WMS"
    },
    {
      "name": "OWS:WPS"
    },
    {
      "name": "OWS:ALL"
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    {
      "name": "create"
    },
    {
      "name": "upload"
    },
    {
      "name": "change"
    },
    {
      "name": "change_metadata"
    },
    {
      "name": "view_metadata"
    },
    {
      "name": "publish"
    },
    {
      "name": "remove"
    }
  ],
  "success": true
}

```

Event types starting with *OWS:* prefix mean they're related to OWS service. *OWS:ALL* is a cumulative event type, which keeps requests for any OWS.

Event type *other* means request not related to OWS. This is also cumulative event type, and should be used as a baseline of all non-ows requests.

In order to retrieve *OWS* only requests the *ows-service* flag (possible values are *True*, *true*, *False*, *false*, *0*, *1*) can be used:

- *OWS* event types

GET /monitoring/api/event_types/?ows_service=true

```

{
  "status": "ok",
  "errors": {},
  "data": {
    "key": "event_types"
  },
  "event_types": [
    {
      "name": "OWS:TMS",
      "type_label": "TMS"
    },
    {
      "name": "OWS:WMS-C",
      "type_label": "WMS-C"
    },
    {
      "name": "OWS:WMTS",
      "type_label": "WMTS"
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    "name": "OWS:WCS",
    "type_label": "WCS"
  },
  {
    "name": "OWS:WFS",
    "type_label": "WFS"
  },
  {
    "name": "OWS:WMS",
    "type_label": "WMS"
  },
  {
    "name": "OWS:WPS",
    "type_label": "WPS"
  },
  {
    "name": "OWS:ALL",
    "type_label": "Any OWS"
  }
],
"success": true
}

```

- *non-OWS* event types

GET /monitoring/api/event_types/?ows_service=false

```

{
  "status": "ok",
  "errors": {},
  "data": {
    "key": "event_types"
  },
  "event_types": [
    {
      "name": "other",
      "type_label": "Not OWS"
    },
    {
      "name": "all",
      "type_label": "All"
    },
    {
      "name": "create",
      "type_label": "Create"
    },
    {
      "name": "upload",
      "type_label": "Upload"
    },
    {
      "name": "change",
      "type_label": "Change"
    },
    {
      "name": "change_metadata",
      "type_label": "Change Metadata"
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    },
    {
      "name": "view_metadata",
      "type_label": "View Metadata"
    },
    {
      "name": "view",
      "type_label": "View"
    },
    {
      "name": "download",
      "type_label": "Download"
    },
    {
      "name": "publish",
      "type_label": "Publish"
    },
    {
      "name": "remove",
      "type_label": "Remove"
    },
    {
      "name": "geoserver",
      "type_label": "Geoserver event"
    }
  ],
  "success": true
}

```

Event type *all* means any request.

Label

Label is a description of subset of metric data that is not described by resources (it's not served as logical data set). Things that can be described with label:

- user tracking id
- volume mount point
- network interface name
- request path
- request method
- response status code
- etc ...

List of all labels recorded is available in `/api/labels/` endpoint:

GET `/monitoring/api/labels/`

```

{
  "labels": [
    {
      "id": 306,

```

(continues on next page)

(continued from previous page)

```

    "name": "Other / Other / Python Requests 2.13"
  },
  {
    "id": 315,
    "name": "Kent"
  },
  {
    "id": 298,
    "name": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/537.36_
↪(KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"
  },
  {
    "id": 261,
    "name": "lo"
  },
  {
    "id": 331,
    "name": "PUT"
  },
  {
    "id": 334,
    "name": "Other / Other / Python Requests 2.18"
  }
]
}

```

Each metric data set will have at least one label attached. List of labels can be filtered with following query sting arguments:

- *metric_name* - name of metric for which labels should be returned
- *valid_from* - list labels that are available since that timestamp
- *valid_to* - list labels that are available until that timestamp

Example:

GET /monitoring/api/labels/?metric_name=request.ua&valid_from=2017-08-05

```

{
  "labels": [
    {
      "id": 298,
      "name": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/537.36_
↪(KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"
    },
    {
      "id": 312,
      "name": "Java/1.8.0_131"
    },
    {
      "id": 293,
      "name": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)_
↪Chrome/36.0.1985.67 Safari/537.36"
    },
    {
      "id": 345,
      "name": "Mozilla/5.0 (Macintosh; Intel Mac OS X) AppleWebKit/538.1 (KHTML, like_
↪Gecko) PhantomJS/2.1.1 Safari/538.1"
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```
    },  
    ...  
  ]  
}
```

Metric name

Metric name is a semi-namespace description of what kind of data metric stores. Typical metric names: - *request.count* - *request.ip* - *response.size* - *response.status*

Each service type has a set of metrics available. Application-level services will have different metric set than host-level services.

Full list of metrics is available in `/api/metrics/` endpoint. Returned list is not filterable. Sample response:

GET `/monitoring/api/metrics/`

```
{  
  "metrics": [  
    {  
      "metrics": [  
        {  
          "type": "count",  
          "name": "request.count",  
          "unit": "Count"  
        },  
        {  
          "type": "count",  
          "name": "request.ip",  
          "unit": "Count"  
        },  
        ...  
      ],  
      "service": "geonode"  
    },  
    { "service": "geoserver",  
      "metrics": [..]  
    }  
  ]  
}
```

Metrics are grouped by service. Each metric has following structure:

```
{  
  "type": "count",  
  "name": "request.ip",  
  "unit": "Count"  
}
```

where:

- *type* is a metric data type (it can be count, value or rate). This is internal description of how to deal with aggregation of data for metric.
- *name* name of metric
- *unit* suggested Y-axis label, describing data units

Metric Data

Core feature of monitoring API is ability to get data for given metric for specified period. Metric value is a data set for fixed period of time, from which data were collected and processed for one specific metric name. Additionally, each metric can have data calculated for specific services, resources, labels and event_types. Metric data API has several features:

- it can show metric data within specific time frame, down to 1 minute granularity (may be less if collection intervals are lower).
- it can show metric data aggregated with custom granularity (for example from last 48 hours with 15 minutes granularity).
- it can show metric data for whole monitored setup or for specific resource, label (like user agent type), monitored service (just for geonode or just for geoserver), Event type. Params can be joined in one query.

API endpoint is: `/api/metric_data/METRIC_NAME/`:

Sample request for *request.ua* metric in specific time window (between 10am and 2pm of 2017-08-03) and data granularity (1h)

```
GET /monitoring/api/metric_data/request.ua/?
valid_from=2017-08-03%2010:00:00&valid_to=2017-08-03%2014:00:00&interval=3600
```

```
{
  "data": {
    "input_valid_from": "2017-08-03T10:00:00",
    "input_valid_to": "2017-08-03T14:00:00",
    "data": [
      {
        "valid_from": "2017-08-03T10:00:00",
        "data": [],
        "valid_to": "2017-08-03T11:00:00"
      },
      {
        "valid_from": "2017-08-03T11:00:00",
        "data": [
          {
            "samples_count": 10,
            "val": "10.0000",
            "min": "1.0000",
            "max": "1.0000",
            "sum": "10.0000",
            "label": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:53.0) Gecko/20100101_
↵Firefox/53.0",
            "metric_count": 10
          },
          {
            "samples_count": 790,
            "val": "790.0000",
            "min": "19.0000",
            "max": "79.0000",
            "sum": "790.0000",
            "label": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like_
↵Gecko) Chrome/60.0.3112.78 Safari/537.36",

```

(continues on next page)

(continued from previous page)

```

        "metric_count": 22
    },
    {
        "samples_count": 150,
        "val": "150.0000",
        "min": "15.0000",
        "max": "15.0000",
        "sum": "150.0000",
        "label": "Mozilla/5.0 (Macintosh; Intel Mac OS X) AppleWebKit/538.1
↪(KHTML, like Gecko) PhantomJS/2.1.1 Safari/538.1",
        "metric_count": 10
    }
],
"valid_to": "2017-08-03T12:00:00"
},
{
    "valid_from": "2017-08-03T12:00:00",
    "data": [],
    "valid_to": "2017-08-03T13:00:00"
},
{
    "valid_from": "2017-08-03T13:00:00",
    "data": [
        {
            "samples_count": 37,
            "val": "37.0000",
            "min": "4.0000",
            "max": "12.0000",
            "sum": "37.0000",
            "label": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
↪Gecko) Chrome/60.0.3112.40 Safari/537.36",
            "metric_count": 4
        }
    ],
    "valid_to": "2017-08-03T14:00:00"
}
],
"metric": "request.ua",
"interval": 3600,
"type": "count",
"axis_label": "Count",
"label": null
}
}

```

Metric data response is wrapped with following envelope:

```

"data": {
    "input_valid_from": "2017-08-03T10:00:00",
    "input_valid_to": "2017-08-03T14:00:00",
    "metric": "request.ua",
    "interval": 3600,
    "type": "count",
    "axis_label": "Count",
    "label": null,
    "data": [
        ... # actual data
    ]
}

```

(continues on next page)

(continued from previous page)

```

    ],
  }
}

```

where:

- *input_valid_from* and *input_valid_to* are parsed and aligned timestamps for which data are returned,
- *metric* is metric name for which response is returned,
- *interval* data aggregation interval used, in seconds (if none is provided, 60 seconds are used, unless time window is larger than 24 hours),
- *type* is metric data type, which describes internally how data are aggregated (sum, average or min/max function).
- *axis_label* is suggested value-axis label to be used in chart
- *label* is metric data label used (no label by default).

Metric data item is build as following structure:

```

{
  "valid_from": "2017-08-03T13:00:00",
  "valid_to": "2017-08-03T14:00:00",
  "data": [
    {
      "samples_count": 37,
      "val": "37.0000",
      "min": "4.0000",
      "max": "12.0000",
      "sum": "37.0000",
      "label": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like_
↪Gecko) Chrome/60.0.3112.40 Safari/537.36",
      "metric_count": 4
    },
    {
      "samples_count": 20,
      "val": "20.0000",
      "min": "4.0000",
      "max": "10.0000",
      "sum": "20.0000",
      "label": "Internet Explorer 4.0",
      "metric_count": 3
    },
  ],
}

```

where:

- *valid_from* and *valid_to* are timestamps of data aggregation period
- *data* is a list of value rows. When *data* is empty, that means no data were collected for input params.
- each *data* element contains:
 - *label* label value associated with metric data value. This can describe user-provided differentiation value (user agent string, request method etc), or, if such value is not in use, default, “count” or “value” label.
 - *val* is metric data aggregated value, which should be used by frontend application. For *request.ua* this means count of requests for given user agent string, for *response.time* that will return average response

time.

- *min*, *max*, *sum* are helper statistical values to give insight on data used,
- *samples_count* is a sum of all samples counts (actual requests) used for this calculation
- *metric_count* is a number of metric data used to calculate the value.
- *resource* (optional) key with resource structure (*id*, *name*, *type*). This element will be visible when grouping by resource is used.
- *event_type* (optional) key with name of event type related to rest of row. This element will be visible when grouping by event type is used

Metric data can be filtered with following params:

- *valid_from* timestamp (date or date + time) meaning that data should be newer than this timestamp
- *valid_to* timestamp (date or date + time) meaning that data should be older than this timestamp
- *interval* data aggregation interval, in seconds. See below notes about intervals and timestamps alignment
- *label* label value only for which data should be returned (see [Labels](#labels))
- *resource* id of resource (see [Resources](#resources)) for which data should be returned
- *service* name of service (see [Services](#services)) for which data should be returned
- *event_type* name of service (see [Event Types](#ows_service)) for which data should be returned
- *resource_type* name of resource type to filter by, for example *layer* to show only data for layer objects (exclude urls, documents, maps).

grouping metric data

Additionally, in some cases client application may want to receive list of data points in one period for several resources (typical usage scenario: list top-most requested layers). In such case, metric data should be queried also with following params:

- *group_by* - name of object which should be used for grouping. At the moment two grouping modes are available:
 - *resource* - group by resource affected. This will produce metrics for the same label but each resource affected will be listed separately. Returned metric data items will have additional *resource* key, which will hold dictionary with keys *name* and *type*. Sample response:

```
GET /monitoring/api/metric_data/request.count/?last=86400&interval=86400&group_by=resource
```

```
{
  "data": {
    "input_valid_from": "2017-09-01T00:00:00",
    "input_valid_to": "2017-09-08T13:50:34.024",
    "data": [
      ..
      {
        "valid_from": "2017-09-04T00:00:00",
        "data": [
          {
            "resource": {
              "type": "layer",
              "name": "nurc:Arc_Sample"
            },
            "samples_count": 300,

```

(continues on next page)

(continued from previous page)

```

        "val": "300.0000",
        "min": "100.0000",
        "max": "100.0000",
        "sum": "300.0000",
        "label": "count",
        "metric_count": 3,
        "id": 10
    },
    {
        "resource": {
            "type": "layer",
            "name": "sde:HYP_HR_SR_OB_DR"
        },
        "samples_count": 72,
        "val": "72.0000",
        "min": "24.0000",
        "max": "24.0000",
        "sum": "72.0000",
        "label": "count",
        "metric_count": 3,
        "id": 25
    }
],
"valid_to": "2017-09-05T00:00:00"
}
],
"valid_to": "2017-09-09T00:00:00"
}
],
"metric": "request.count",
"interval": 86400,
"type": "count",
"axis_label": "Count",
"label": null
}
}

```

- *resource_no_labels* - group by resource affected, but do not distinct by label. This will produce similar result as the other grouping, but it will not contain ‘label’ key.

GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=

```

{
  "data": {
    "input_valid_from": "2018-07-10T15:13:50.784Z",
    "input_valid_to": "2018-07-11T15:13:50.784Z",
    "data": [
      {
        "val" id_from: "2018-07-10T15:13:50.784Z",
        "data": [
          {
            "resource": {
              "type": "url",
              "name": "/layers/",
              "id": 15
            },
            "metric_count": 4,

```

(continues on next page)

(continued from previous page)

```

        "val": 2,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "4.0000",
        "samples_count": 4
    },
    {
        "resource": {
            "type": "url",
            "name": "/",
            "id": 16
        },
        "metric_count": 4,
        "val": 2,
        "min": "1.0000",
        "max": "4.0000",
        "sum": "7.0000",
        "samples_count": 7
    },
    {
        "resource": {
            "type": "url",
            "name": "/maps/",
            "id": 17
        },
        "metric_count": 4,
        "val": 2,
        "min": "1.0000",
        "max": "2.0000",
        "sum": "5.0000",
        "samples_count": 5
    },
    {
        "resource": {
            "type": "url",
            "name": "/maps/3",
            "id": 18
        },
        "metric_count": 1,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "1.0000",
        "samples_count": 1
    },
    {
        "resource": {
            "type": "url",
            "name": "/maps/7",
            "id": 20
        },
        "metric_count": 1,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "1.0000",
        "samples_count": 1
    }

```

(continues on next page)

(continued from previous page)

```

        }
      ],
      "valid_to": "2018-07-11T15:13:50.784Z"
    }
  ],
  "metric": "request.users",
  "interval": 86400,
  "type": "value",
  "axis_label": "Count",
  "label": null
}

```

- *label* - group by label. This will return number of unique label occurrences within selected period.

GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=label

```

{
  "data": {
    "input_valid_from": "2018-07-10T16:29:08.982Z",
    "input_valid_to": "2018-07-11T16:29:08.982Z",
    "data": [
      {
        "valid_from": "2018-07-10T16:29:08.982Z",
        "data": [
          {
            "samples_count": 243,
            "val": 13,
            "min": "0.0000",
            "max": "25.0000",
            "sum": "243.0000",
            "metric_count": 124
          }
        ],
        "valid_to": "2018-07-11T16:29:08.982Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

- *event_type* - group by event type. This will expose *event_type* field in data items. Grouping will return number of requests per each event type.
- *event_type_on_label* - group by event type but use label to do grouping (instead of metric data value). This will expose *event_type* field in data items. Grouping will return number of requests per label (especially for *request.users*, which uses label field as tracking id value, see [User Analytics](<https://github.com/geosolutions-it/geonode/wiki/Monitoring:-User-Analytics>)).

Timestamps alignment

Data collected by monitoring are aggregated into fixed period values. This have several consequences:

- you cannot query for time window smaller than aggregation period
- when querying for time window, input `valid_from` and `valid_to` will be aligned to possible actual `valid_from` and `valid_to` values. Alignment is calculated from 0:00h each day. For best results, you should use intervals that can be aligned without reminders.
- timestamps alignment may produce more rows than you expect in some cases. For example, let's say client application want to have data aggregated with 5 minutes interval. Search for data between 12:04 and 12:06, even if interval between those two (2 minutes) is smaller than data interval (5 minutes), this will be aligned to data intervals, which will be:
 - from 12:00 to 12:05
 - from 12:05 to 12:10

If data aggregation period ends in the future, there's good chance it will not contain any data.

Exceptions

Exceptions are served with separate API endpoints. Those endpoints will return:

- list of exceptions captured
- exception details

List of exceptions is available in `/api/exceptions/` endpoint:

GET `/monitoring/api/exceptions/`

```
{
  "exceptions": [
    {
      "url": "/monitoring/api/exceptions/8/",
      "error_type": "exceptions.ValueError",
      "id": 8,
      "service": {
        "type": "geonode",
        "name": "local-geonode"
      },
      "created": "2017-06-20T17:50:24.922"
    },
    {
      "url": "/monitoring/api/exceptions/9/",
      "error_type": "org.geoserver.platform.ServiceException",
      "id": 9,
      "service": {
        "type": "geoserver",
        "name": "local-geoserver"
      },
      "created": "2017-06-26T15:33:20.152"
    },
    {
      "url": "/monitoring/api/exceptions/10/",
      "error_type": "django.db.utils.ProgrammingError",
      "id": 10,
```

(continues on next page)

(continued from previous page)

```

    "service": {
        "type": "geonode",
        "name": "local-geonode"
    },
    "created": "2017-06-27T12:32:37.032"
},
]
}

```

Each exception in list contains:

- *error_type* which is a class of exception
- *id* object id for given exception recorded
- *service* service object, on which exception was recorded
- *created* exception recorded timestamp
- *url* url with exception details

Exception details:

GET /monitoring/api/exceptions/30/

```

{
  "error_data": "Traceback (most recent call last):\n File \"/home/cezio/.\n
  ↳virtualenvs/geonode/lib/python2.7/site-packages/django/core/handlers/base.py",\n
  ↳line 132, in get_response\n   response = wrapped_callback(request, *callback_args,\n
  ↳**callback_kwargs)\n File \"/home/cezio/.virtualenvs/geonode/lib/python2.7/site-\n
  ↳packages/django/views/generic/base.py", line 71, in view\n   return self.\n
  ↳dispatch(request, *args, **kwargs)\n File \"/home/cezio/.virtualenvs/geonode/lib/\n
  ↳python2.7/site-packages/django/views/generic/base.py", line 89, in dispatch\n
  ↳return handler(request, *args, **kwargs)\n File \"/mnt/work/cezio/geosolutions/\n
  ↳repos/geonode/geonode/contrib/monitoring/views.py", line 176, in get\n   return\n
  ↳json_response({self.output_name: out})\n File \"/mnt/work/cezio/geosolutions/repos/\n
  ↳geonode/geonode/utils.py", line 619, in json_response\n   body = json.dumps(body,\n
  ↳cls=DjangoJSONEncoder)\n File \"/usr/lib64/python2.7/json/__init__.py", line 251,\n
  ↳in dumps\n   sort_keys=sort_keys, **kw).encode(obj)\n File \"/usr/lib64/python2.7/\n
  ↳json/encoder.py", line 207, in encode\n   chunks = self.iterencode(o, _one_\n
  ↳shot=True)\n File \"/usr/lib64/python2.7/json/encoder.py", line 270, in\n
  ↳iterencode\n   return _iterencode(o, 0)\n File \"/home/cezio/.virtualenvs/geonode/\n
  ↳lib/python2.7/site-packages/django/core/serializers/json.py", line 115, in default\n
  ↳return super(DjangoJSONEncoder, self).default(o)\n File \"/usr/lib64/python2.\n
  ↳7/json/encoder.py", line 184, in default\n   raise TypeError(repr(o) + "\n
  ↳is not\n
  ↳JSON serializable")\nTypeError: <Service: Service: local-geoserver@localhost> is\n
  ↳not JSON serializable\n",
  "service": {
    "type": "geonode",
    "name": "local-geonode"
  },
  "created": "2017-07-24T13:29:28.321",
  "error_type": "exceptions.TypeError",
  "request": {
    "event_type": null,
    "client": {
      "ip": "127.0.0.1",
      "position": {
        "lat": null,

```

(continues on next page)

(continued from previous page)

```

        "country": null,
        "lon": null,
        "city": null
    },
    "user_agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like_
↪Gecko) Chrome/60.0.3112.40 Safari/537.36",
    "user_agent_family": "PC / Linux / Chrome 60.0.3112"
},
"request": {
    "path": "/monitoring/api/exceptions/",
    "host": "localhost:8000",
    "method": "GET",
    "created": "2017-07-24T13:29:28.280"
},
"response": {
    "status": 200,
    "time": 30,
    "type": "text/html; charset=utf-8",
    "size": 0
},
"resources": []
},
"error_message": "exceptions.TypeError"
}

```

Details contain:

- *error_type* which is a class of exception
- *error_message* message provided with error
- *error_data* is a plain text with stack trace
- *service* service object, on which exception was recorded
- *created* exception recorded timestamp
- *request* information on request associated with this error:
 - *event_type* name of Event Type associated with request
 - *client* requesting client information
 - *request* details on request received
 - *response* details on response send back
 - *resources* list of resources affected

Autoconfiguration

Autoconfiguration endpoint allows to perform monitoring configuration based on *settings* values. This API endpoint is available to superusers/staff only. Response is wrapped with standard envelope.

POST /monitoring/api/autoconfigure/

```

{
    "status": "ok",
    "success": true,

```

(continues on next page)

(continued from previous page)

```
"errors": {}
}
```

1.18.3 Monitoring: User Analytics

Purpose

UA should provide information about GeoNode resources usage at user level (not request level, like plain monitoring).

Requests

1. total number of unique sessions on GeoNode (excluding ows requests) per day. This gives a base view of the reach.

- requests from all sessions of all types, ows and non-ows

```
GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group=
```

- non-ows related

```
GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group=
```

- only ows related

```
GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group=
```

```
{
  "data": {
    "input_valid_from": "2018-07-11T15:41:06.419Z",
    "input_valid_to": "2018-07-12T15:41:06.419Z",
    "data": [
      {
        "valid_from": "2018-07-11T15:41:06.419Z",
        "data": [
          {
            "samples_count": 82,
            "val": 9,
            "min": "0.0000",
            "max": "24.0000",
            "sum": "82.0000",
            "metric_count": 16
          }
        ],
        "valid_to": "2018-07-12T15:41:06.419Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}
```

2. total number of unique sessions per URL (excluding ows requests). Let me see how many users visits the layers page or the maps page

- get number of unique tracking ids for urls

GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group

```
{
  "data": {
    "input_valid_from": "2018-07-11T15:39:25.126Z",
    "input_valid_to": "2018-07-12T15:39:25.126Z",
    "data": [
      {
        "valid_from": "2018-07-11T15:39:25.126Z",
        "data": [
          {
            "resource": {
              "type": "url",
              "name": "/layers/",
              "id": 15
            },
            "metric_count": 2,
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "2.0000",
            "samples_count": 2
          },
          {
            "resource": {
              "type": "url",
              "name": "/",
              "id": 16
            },
            "metric_count": 2,
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "2.0000",
            "samples_count": 2
          },
          {
            "resource": {
              "type": "url",
              "name": "/documents/",
              "id": 21
            },
            "metric_count": 1,
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "1.0000",
            "samples_count": 1
          }
        ],
        "valid_to": "2018-07-12T15:39:25.126Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
```

(continues on next page)

(continued from previous page)

```

    "axis_label": "Count",
    "label": null
  }
}

```

3. total number of unique sessions per event_type: for example total number of unique visits of resource pages (independently by resource type and id)

- to get number of requests

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

- to get number of unique tracking ids

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

- to get number of unique tracking ids for each event_type on a given resource type

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

```

{
  "data": {
    "input_valid_from": "2018-07-11T17:54:41.467Z",
    "input_valid_to": "2018-07-12T17:54:41.467Z",
    "data": [
      {
        "valid_from": "2018-07-11T17:54:41.467Z",
        "data": [
          {
            "samples_count": 5,
            "event_type": "all",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "other",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "view",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          }
        ],
        "valid_to": "2018-07-12T17:54:41.467Z"
      }
    ]
  }
}

```

(continues on next page)

(continued from previous page)

```

    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

- to get number of unique users for each event type on specific resource type

GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=

```

{
  "data": {
    "input_valid_from": "2018-07-11T17:54:41.467Z",
    "input_valid_to": "2018-07-12T17:54:41.467Z",
    "data": [
      {
        "valid_from": "2018-07-11T17:54:41.467Z",
        "data": [
          {
            "samples_count": 5,
            "event_type": "all",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "other",
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "view",
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          }
        ],
        "valid_to": "2018-07-12T17:54:41.467Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

(continues on next page)

(continued from previous page)

}

4. total number of unique sessions per event_type and single resource: let me see what was the most visited map page in this day, or what was the most downloaded document, what was the most requested ows layer, etc.

- list of most visited resources of *url* type

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

- list of unique tracking ids for each resource (can be narrowed down to specific resource type with *resource_type* values).

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

```
{
  "data": {
    "input_valid_from": "2018-07-11T17:56:49.381Z",
    "input_valid_to": "2018-07-12T17:56:49.381Z",
    "data": [
      {
        "valid_from": "2018-07-11T17:56:49.381Z",
        "data": [
          {
            "resource": {
              "type": "",
              "name": "",
              "id": 1
            },
            "metric_count": 16,
            "val": 9,
            "min": "0.0000",
            "max": "24.0000",
            "sum": "82.0000",
            "samples_count": 82
          },
          {
            "resource": {
              "type": "layer",
              "name": "geonode:ne_50m_admin_0_countries_lakes",
              "id": 2
            },
            "metric_count": 4,
            "val": 3,
            "min": "0.0000",
            "max": "2.0000",
            "sum": "3.0000",
            "samples_count": 3
          },
          {
            "resource": {
              "type": "layer",
              "name": "geonode:world_iso2",
              "id": 12
            },
            "metric_count": 4,
            "val": 2,
            "min": "0.0000",
            "max": "5.0000",
```

(continues on next page)

(continued from previous page)

```

        "sum": "8.0000",
        "samples_count": 8
    },
    {
        "resource": {
            "type": "url",
            "name": "/layers/",
            "id": 15
        },
        "metric_count": 2,
        "val": 2,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "2.0000",
        "samples_count": 2
    },
    {
        "resource": {
            "type": "url",
            "name": "/",
            "id": 16
        },
        "metric_count": 2,
        "val": 2,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "2.0000",
        "samples_count": 2
    },
    {
        "resource": {
            "type": "url",
            "name": "/documents/",
            "id": 21
        },
        "metric_count": 1,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "1.0000",
        "samples_count": 1
    },
    {
        "resource": {
            "type": "document",
            "name": "GeoServer Configuration.pdf",
            "id": 22
        },
        "metric_count": 1,
        "val": 1,
        "min": "5.0000",
        "max": "5.0000",
        "sum": "5.0000",
        "samples_count": 5
    }
],
"valid_to": "2018-07-12T17:56:49.381Z"

```

(continues on next page)

(continued from previous page)

```

    }
  ],
  "metric": "request.users",
  "interval": 86400.0,
  "type": "value",
  "axis_label": "Count",
  "label": null
}
}

```

5. total number of unique visitor (user) per event_type and single resource: let me see how many users visited the map page in this day, or how many users download some resource, etc.

- number of unique visitors (users) in a year for a given event_type:

```
GET /monitoring/api/metric_data/request.users/ ?
valid_from=2019-01-01+00:00:00&valid_to=2019-12-31+23:59:59
&interval=31536000&event_type=upload&group_by=user
```

- number of unique visitors (users) in a given time interval and for a given resource_type.

```
GET /monitoring/api/metric_data/request.users/ ?
valid_from=2019-01-01+00:00:00&valid_to=2019-12-31+23:59:59
&interval=31536000&resource_type=layer&group_by=user
```

the responses should look like this:

```

{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 3,
            "val": 2,
            "min": "1.0000",
            "max": "2.0000",
            "sum": "3.0000",
            "metric_count": 2
          }
        ],
        "valid_to": "2020-01-01T00:00:00Z"
      }
    ],
    "metric": "request.users",
    "interval": 31536000,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

6. total number of unique tracking ids/sessions for a given user.

- sessions count for anonymous users:

```
GET /monitoring/api/metric_data/request.users/?valid_from=2019-01-01+00:00:00&valid_to=2019-12-31T23:59:59Z&interval=31536000&group_by=label&user=AnonymousUser
```

```
{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 5,
            "val": 5,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          }
        ],
        "valid_to": "2020-01-01T00:00:00Z"
      }
    ],
    "metric": "request.users",
    "interval": 31536000,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}
```

7. total number of unique tracking ids/sessions for each user.

- sessions count for each users:

```
GET /monitoring/api/metric_data/request.users/?valid_from=2019-01-01+00:00:00&valid_to=2019-12-31T23:59:59Z&interval=31536000&group_by=user_on_label
```

```
{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 5,
            "val": 5,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "user": "AnonymousUser",
            "metric_count": 5
          }
        ],
        {
          "samples_count": 16,
          "val": 3,
          "min": "1.0000",

```

(continues on next page)

(continued from previous page)

```

        "max": "2.0000",
        "sum": "16.0000",
        "user": "admin",
        "metric_count": 14
    },
    {
        "samples_count": 4,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "4.0000",
        "user": "user1_username",
        "metric_count": 4
    }
],
    "valid_to": "2020-01-01T00:00:00Z"
}
],
"metric": "request.users",
"interval": 31536000,
"type": "value",
"axis_label": "Count",
"label": null
}
}

```

- sessions count for each users which do something with a layer:

GET /monitoring/api/metric_data/request.users/?valid_from=2019-01-01+00:00:00&valid_to=2020-01-01+00:00:00&interval=31536000&resource_type=layer&group_by=user_on_label

```

{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 2,
            "val": 1,
            "min": "2.0000",
            "max": "2.0000",
            "sum": "2.0000",
            "user": "admin",
            "metric_count": 1
          },
          {
            "samples_count": 1,
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "1.0000",
            "user": "user1_username",
            "metric_count": 1
          }
        ]
      }
    ]
  }
}

```

(continues on next page)

(continued from previous page)

```

    ],
    "valid_to": "2020-01-01T00:00:00Z"
  }
],
"metric": "request.users",
"interval": 31536000,
"type": "value",
"axis_label": "Count",
"label": null
}
}

```

8. total number of unique users for each monitored resource.

GET /monitoring/api/metric_data/request.users/ ?
 last=31536000&interval=31536000&group_by=resource_on_user

```

{
  "data": {
    "input_valid_from": "2018-09-10T14:15:39.454Z",
    "input_valid_to": "2019-09-10T14:15:39.454Z",
    "data": [
      {
        "valid_from": "2018-09-10T14:15:39.454Z",
        "data": [
          {
            "resource": {
              "href": "/",
              "type": "url",
              "name": "/",
              "id": 1
            },
            "metric_count": 36,
            "val": 4,
            "max": "2.0000",
            "sum": "35.0000",
            "min": "0.0000",
            "samples_count": 35
          },
          {
            "resource": {
              "href": "/maps/",
              "type": "url",
              "name": "/maps/",
              "id": 3
            },
            "metric_count": 3,
            "val": 2,
            "max": "1.0000",
            "sum": "3.0000",
            "min": "1.0000",
            "samples_count": 3
          },
          {
            "resource": {
              "href": "",
              "type": "layer",

```

(continues on next page)

(continued from previous page)

```

        "name": "geonode:railways",
        "id": 4
    },
    "metric_count": 5,
    "val": 2,
    "max": "2.0000",
    "sum": "3.0000",
    "min": "0.0000",
    "samples_count": 3
},
{
    "resource": {
        "href": "/layers/",
        "type": "url",
        "name": "/layers/",
        "id": 2
    },
    "metric_count": 4,
    "val": 1,
    "max": "1.0000",
    "sum": "4.0000",
    "min": "1.0000",
    "samples_count": 4
},
{
    "resource": {
        "href": "/documents/2",
        "type": "document",
        "name": "test_doc_1.txt",
        "id": 5
    },
    "metric_count": 2,
    "val": 1,
    "max": "2.0000",
    "sum": "4.0000",
    "min": "2.0000",
    "samples_count": 4
},
{
    "resource": {
        "href": "/maps/3",
        "type": "map",
        "name": "test_map",
        "id": 6
    },
    "metric_count": 1,
    "val": 1,
    "max": "1.0000",
    "sum": "1.0000",
    "min": "1.0000",
    "samples_count": 1
},
{
    "resource": {
        "href": "",
        "type": "layer",
        "name": "geonode:waterways",

```

(continues on next page)

(continued from previous page)

```

        "id": 7
      },
      "metric_count": 2,
      "val": 1,
      "max": "2.0000",
      "sum": "2.0000",
      "min": "0.0000",
      "samples_count": 2
    }
  ],
  "valid_to": "2019-09-10T14:15:39.454Z"
}
],
"metric": "request.users",
"interval": 31536000,
"type": "value",
"axis_label": "Count",
"label": null
}
}

```

9. total number of resource monitored in a given time range.

GET /monitoring/api/metric_data/request.users/
last=31536000&interval=31536000&group_by=count_on_resource

?

```

{
  "data": {
    "input_valid_from": "2018-09-10T14:20:27.335Z",
    "input_valid_to": "2019-09-10T14:20:27.335Z",
    "data": [
      {
        "valid_from": "2018-09-10T14:20:27.335Z",
        "data": [
          {
            "samples_count": 52,
            "val": 7,
            "min": "0.0000",
            "max": "2.0000",
            "sum": "52.0000",
            "metric_count": 53
          }
        ],
        "valid_to": "2019-09-10T14:20:27.335Z"
      }
    ],
    "metric": "request.users",
    "interval": 31536000,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

1.18.4 Monitoring: Notifications

Notifications are part of monitoring that is run after each data collection cycle. Its configurable mechanism to check if metrics values are within allowed value range, and if not, send notification to designated receivers (registered users or external emails).

Data model

Notification mechanism is composed of several classes, responsible for different aspects:

- High-level configuration: *NotificationCheck*:
 - keeps general description, list of metric check definition, send grace period configuration and last send marker, list of users to which notification should be delivered (in helper table, *NotificationReceiver* class).
- Per-metric definition: *MetricNotificationDefinition*:
 - keeps per-metric-per-check configuration: name of metric, min, max values allowed for user, check type (if value should be below or above given threshold, or should last read be not older than specific period from metric check), additional scope for check (resource, label, ows service - this part is partially implemented). Definition object is created from *NotificationCheck.user_thresholds* data, and is used to generate validation form. Note, that one *NotificationCheck* can have several definition items, for set of different metrics. Definition rows are created when *NotificationCheck* is created, or updated.
- Per-metric check configuration: *MetricNotificationCheck*
 - Keeps per-metric-per-check configuration: metric and threshold values. It is created after user submits configuration form for specific notification.

Workflow

Notifications are checked after each collection/processing period in collection script, by calling *CollectorAPI.emit_notifications(for_timestamp)*. This will do following:

- get all notifications,
- for each notification, will get all notification checks
- for each notification check, it will get metric valid for given timestamp and check if value matches given criteria
- each check can raise exception, which will be captured in caller, and for each notification, list of errors will be returned
- based on list of notifications and errors, alerts will be generated and send to users, unless last delivery was before grace period is finished.

Additionally, notifications expose `/monitoring/api/status/` *Status API*, which will show errors detected at the moment of request.

1.18.5 Web API

Status API

Status endpoint presents current state of error checking performed by notifications. Frontend can make requests periodically to this endpoint. There is no history view for status at the moment. Status response is wrapped with standard response envelope. Non-error response will have *status* key set to *ok* and *success* to *true*, otherwise *errors* will be not empty.

No errors response:

GET /monitoring/api/status/

```
{
  "status": "ok",
  "data": [],
  "success": true
}
```

Response with errors reported:

```
{
  "status": "ok",
  "data": [
    {
      "problems": [
        {
          "threshold_value": "2017-08-29T10:45:26.142",
          "message": "Value collected too far in the past",
          "name": "request.count",
          "severity": "warning",
          "offending_value": "2017-08-25T16:41:00"
        }
      ],
      "check": {
        "grace_period": {
          "seconds": 600,
          "class": "datetime.timedelta"
        },
        "last_send": null,
        "description": "detects when requests are not handled",
        "severity": "warning",
        "user_threshold": {
          "3": {
            "max": 10,
            "metric": "request.count",
            "steps": null,
            "description": "Number of handled requests is lower than",
            "min": 0
          },
          "4": {
            "max": null,
            "metric": "request.count",
            "steps": null,
            "description": "No response for at least",
            "min": 60
          },
          "5": {
```

(continues on next page)

(continued from previous page)

```

        "max": null,
        "metric": "response.time",
        "steps": null,
        "description": "Response time is higher than",
        "min": 500
    }
},
    "id": 2,
    "name": "geonode is not working"
}
],
"success": true
}

```

Response with reported errors contains list of check elements in *data* element. Each check element contains:

- *check* - serialized *NotificationCheck* object, which was used
- *problems* - list of metric checks that failed. Each element contains name of metric, severity, error message, measured and threshold value.

Severity

Severity is a textual description of potential impact of error. There are three values: *warning*, *error* and *fatal*.

Notification list

This call will return list of available notifications:

GET /monitoring/api/notifications/

```

{
  "status": "ok",
  "data": {
    "problems": [
      {
        "threshold_value": "10.0000",
        "check_url": "/monitoring/api/notifications/config/2/",
        "name": "request.count",
        "check_id": 2,
        "description": "Metric value for request.count should be at least 10, got 4_
↪instead",
        "offending_value": "4.0000",
        "message": "Number of handled requests is lower than 4",
        "severity": "error"
      }
    ],
    "health_level": "error"
  },
  "success": true
}

```

Response will contain list of notifications summary in *data* key. Each element will have:

- *name* of metric checked

- *message* is error message generated by notification. This describes what the problem is.
- *description* more detailed information what which check failed.
- *offending_value* and *threshold_value* are values that were compared (*offending_value* is actual value from metric data)
- *check_url* to notification details
- *severity* of error

Also, *data* will have highest *severity* value available in *health_level*.

Notification details

This will return details for notification, including form and list of allowed fields:

GET /monitoring/api/notifications/config/{{notification_id}}/

```
{
  "status": "ok",
  "errors": {},
  "data": {
    "fields": [
      {
        "is_enabled": true,
        "use_resource": false,
        "description": "Number of handled requests is lower than",
        "max_value": "10.0000",
        "metric": {
          "class": "geonode.contrib.monitoring.models.Metric",
          "name": "request.count",
          "id": 2
        },
        "min_value": "0.0000",
        "use_label": false,
        "use_ows_service": false,
        "field_option": "min_value",
        "use_service": false,
        "steps_calculated": [
          "0.0000",
          "3.33",
          "6.67",
          "10.0"
        ],
        "current_value": "30.0000",
        "steps": 3,
        "notification_check": {
          "class": "geonode.contrib.monitoring.models.NotificationCheck",
          "name": "geonode is not working",
          "id": 2
        },
        "field_name": "request.count.min_value",
        "id": 3,
        "unit": ""
      },
      {
        "is_enabled": true,
        "use_resource": false,
```

(continues on next page)

(continued from previous page)

```

    "description": "No response for at least",
    "max_value": null,
    "metric": {
        "class": "geonode.contrib.monitoring.models.Metric",
        "name": "request.count",
        "id": 2
    },
    "min_value": "60.0000",
    "use_label": false,
    "use_ows_service": false,
    "field_option": "max_timeout",
    "use_service": false,
    "steps_calculated": null,
    "current_value": {
        "seconds": 120,
        "class": "datetime.timedelta"
    },
    "steps": null,
    "notification_check": {
        "class": "geonode.contrib.monitoring.models.NotificationCheck",
        "name": "geonode is not working",
        "id": 2
    },
    "field_name": "request.count.max_timeout",
    "id": 4,
    "unit": ""
},
{
    "is_enabled": false,
    "use_resource": false,
    "description": "Response time is higher than",
    "max_value": null,
    "metric": {
        "class": "geonode.contrib.monitoring.models.Metric",
        "name": "response.time",
        "id": 11
    },
    "min_value": "500.0000",
    "use_label": false,
    "use_ows_service": false,
    "field_option": "max_value",
    "use_service": false,
    "steps_calculated": null,
    "current_value": null,
    "steps": null,
    "notification_check": {
        "class": "geonode.contrib.monitoring.models.NotificationCheck",
        "name": "geonode is not working",
        "id": 2
    },
    "field_name": "response.time.max_value",
    "id": 5,
    "unit": "s"
},
{
    "is_enabled": false,
    "use_resource": false,

```

(continues on next page)

(continued from previous page)

[illegible]

(continued from previous page)

```

"notification": {
  "grace_period": {
    "seconds": 60,
    "class": "datetime.timedelta"
  },
  "last_send": "2017-09-04T13:13:15.203",
  "description": "detects when requests are not handled",
  "severity": "error",
  "user_threshold": {
    "request.count.max_timeout": {
      "max": null,
      "metric": "request.count",
      "steps": null,
      "description": "No response for at least",
      "min": 60
    },
    "response.time.max_value": {
      "max": null,
      "metric": "response.time",
      "steps": null,
      "description": "Response time is higher than",
      "min": 500
    },
    "request.count.min_value": {
      "max": 10,
      "metric": "request.count",
      "steps": 3,
      "description": "Number of handled requests is lower than",
      "min": 0
    }
  },
  "active": true,
  "id": 2,
  "name": "geonode is not working"
},
"success": true
}

```

Returned keys in *data* element:

- *fields* - list of form fields, including detailed per-resource configuration flags
- *form* - rendered user form, which can be displayed
- *notification* - serialized notification object with *user_thresholds* list (this is a base to create *fields* objects)

Frontend should use *fields* list to create whole form in client-side:

- field name is stored in *field_name*.
- field label can be constructed from *description*
- unit can be extracted from *unit* field
- if field definition provides list in *steps_calculated*, this should be used to construct selection/dropdown, otherwise text input should be displayed. If possible, validation should take into account *min_value* and *max_value*.
- currently set value is available in *current_value* field.

- each field has *is_enabled* property, which tells if field is enabled. Currently this value is calculated in following way: field is enabled if *current_value* is not *None*. This may change in the future.

Additionally, each notification configuration accepts list of emails in *emails* field. This field should be send as a list of emails joined with new line char (*n*).

Form should be submitted to the same url as configuration source (`/monitoring/api/notifications/config/{id}/`), see below.

Notification edition (by user)

Following API call allows user to configure notification by setting receivers and adjust threshold values for checks:

POST `/monitoring/api/notifications/config/{notification_check_id}/`

```
request.count.max_value=val
request.count.min_value=1
emails=list of emails
```

Response contains serialized *NotificationCheck* in *data* element, if no errors were captured during form processing:

```
{
  "status": "ok",
  "errors": {},
  "data": {
    "grace_period": {
      "seconds": 600,
      "class": "datetime.timedelta"
    },
    "last_send": null,
    "description": "more test",
    "severity": "error",
    "user_threshold": {
      "request.count.max_value": {
        "max": null,
        "metric": "request.count",
        "steps": null,
        "description": "Max number of request",
        "min": 1000
      },
      "request.count.min_value": {
        "max": 100,
        "metric": "request.count",
        "steps": null,
        "description": "Min number of request",
        "min": 0
      }
    },
    "id": 293,
    "name": "test"
  },
  "success": true
}
```

Error (non-200) response will have *errors* key populated:

```
{
  "status": "error",
```

(continues on next page)

(continued from previous page)

```

"errors": {
  "user_threshold": [
    "This field is required."
  ],
  "name": [
    "This field is required."
  ],
  "description": [
    "This field is required."
  ]
},
"data": [],
"success": false
}

```

Notification creation

This API call allows to create new notification, it's different in form layout from edition:

POST /monitoring/api/notifications/

```

name=Name of notification (geonode doesn't work)
description=This will check if geonode is serving any data
emails=
user_thresholds=
severity=

```

Payload elements:

- *name*, *description* are values visible for user
- *severity* severity value
- *emails* is a list of emails, however, it is encoded to a string, where each email is in new line:

```

email1@test.com
email2@test.com

```

- *user_thresholds* is a json encoded list of per-metric-per-check configurations. Each element of list should be a 10-element list, containing:
 - name of metric
 - field check option (one of three values: *min_value*, *max_value* or *max_timeout*)
 - flag, if metric check can use service
 - flag, if metric check can use resource
 - flag, if metric check can use label
 - flag, if metric check can use ows service
 - minimum value for user input (no minimum check if None)
 - maximum value for user input (no maximum check if None)
 - steps count is a number of steps to generate for user input, so user can select value from select list instead of typing. This will have effect only if both min and max values are also provided Sample payload for *user_thresholds*:

```
[
    ('request.count', 'min_value', False, False, False, False, 0, 100, None,
↪ "Min number of request"),
    ('request.count', 'max_value', False, False, False, False, 1000, None,
↪ None, "Max number of request"),
]
```

Response is a serialized *NotificationCheck* wrapped with standard response envelope (status, errors etc). Actual data is in *data* key. If processing failed, for example because of form validation errors, response will be non-200 OK, and *errors* key will be populated.

```
{
  "status": "ok",
  "errors": {},
  "data": {
    "grace_period": {
      "seconds": 600,
      "class": "datetime.timedelta"
    },
    "last_send": null,
    "description": "more test",
    "user_threshold": {
      "request.count.max_value": {
        "max": 100,
        "metric": "request.count",
        "steps": null,
        "description": "Min number of request",
        "min": 0
      },
      "request.count.min_value": {
        "max": null,
        "metric": "request.count",
        "steps": null,
        "description": "Max number of request",
        "min": 1000
      }
    },
    "id": 257,
    "name": "test"
  },
  "success": true
}
```

1.19 GeoNode Backup and Restore

1.19.1 Full GeoNode Backup & Restore

The admin command to backup and restore GeoNode, allows to extract consistently the GeoNode and GeoServer data models in a serializable meta-format which is being interpreted later by the restore procedure in order to exactly rebuild the whole structure.

In particular the tool helps developers and administrators to correctly extract and serialize the following resources:

- **GeoNode** (Resource Base Model):
 1. Layers (both raster and vectors)

2. Maps
 3. Documents
 4. People with Credentials
 5. Permissions
 6. Associated Styles
 7. Static data and templates
- **GeoServer** (Catalog):
 1. OWS Services configuration and limits
 2. Security model along with auth filters configuration, users and credentials
 3. Workspaces
 4. Stores (both DataStores and CoverageStores)
 5. Layers
 6. Styles

The tool exposes two GeoNode Management Commands, 'backup' and 'restore'.

The commands allow to:

1. Fully backup GeoNode data and fixtures on a zip archive
2. Fully backup GeoServer configuration (physical datasets - tables, shapefiles, geotiffs)
3. Fully restore GeoNode and GeoServer fixtures and catalog from the zip archive

The usage of those commands is quite easy and straightforward.

The first step is to ensure that everything is correctly configured and the requisites respected in order to successfully perform a backup and restore of GeoNode.

Warning: It is worth to notice that this functionality requires the latest [GeoServer Extension](#) (2.9.x or greater) for GeoNode in order to correctly work.

Note: GeoServer full documentation is also available here [GeoServer Docs](#)

Requisites and Setup

Before running a GeoNode backup / restore, it is necessary to ensure everything is correctly configured and setup.

Settings

Accordingly to the admin needs, the file `settings.ini` must be created before running a backup or restore.

The default files can be found at `geonode/br/management/commands/settings_sample.ini` and `geonode/br/management/commands/settings_docker_sample.ini` for the classic and Docker environments accordingly. The content is similar in both of them (an example from `settings_sample.ini`):

```
[database]
pgdump = pg_dump
pgrestore = pg_restore

[geoserver]
datadir = geoserver/data
dumpvectordata = yes
dumprasterdata = yes

[fixtures]
# NOTE: Order is important
apps = contenttypes,auth,people,groups,account,guardian,admin,actstream,
↪announcements,avatar,base,dialogs,documents,geoserver,invitations,pinax_
↪notifications,layers,maps,oauth2_provider,services,sites,socialaccount,taggit,
↪tastypie,upload,user_messages
dumps = contenttypes,auth,people,groups,account,guardian,admin,actstream,
↪announcements,avatar,base,dialogs,documents,geoserver,invitations,pinax_
↪notifications,layers,maps,oauth2_provider,services,sites,socialaccount,taggit,
↪tastypie,upload,user_messages
```

The `settings.ini` file can be created in any directory accessible by GeoNode, and its path can be passed to the backup / restore procedures using `-c` (`-config`) argument.

There are few different sections of the configuration file, that must be carefully checked before running a backup / restore command.

Settings: [database] Section

```
[database]
pgdump = pg_dump
pgrestore = pg_restore
```

This section is quite simple. It contains only two properties:

- `pgdump`; the path of the `pg_dump` local command.
- `pgrestore`; the path of the `pg_restore` local command.

Warning: Those properties are ignored in case GeoNode is not configured to use a DataBase as backend (see `settings.py` and `local_settings.py` sections)

Note: Database connection settings (both for GeoNode and GeoServer) will be taken from `settings.py` and `local_settings.py` configuration files. Make sure they are correctly configured (on the target GeoNode instance, too) and the DataBase server is accessible while executing a backup / restore command.

Settings: [geoserver] Section

```
[geoserver]
datadir = /opt/gs_data_dir
datadir_exclude_file_path =
dumpvectordata = yes
dumprasterdata = yes
data_dt_filter =
data_layername_filter =
data_layername_exclude_filter =
```

This section allows to enable / disable a full data backup / restore of GeoServer.

- *datadir*: the full path of GeoServer Data Dir, by default `/opt/gs_data_dir`. The path **must** be accessible and **fully writable** by the geonode and / or httpd server users when executing a backup / restore command.
- *datadir_exclude_file_path*: comma separated list of paths to exclude from `geoserver_catalog.zip`; This list will be sent and managed directly by the GeoServer Backup REST API.
- *dumpvectordata*: a boolean flag enabling or disabling creation of a vector data dump from GeoServer (shapefiles or DB tables). If `false` (or `no`) vector data won't be stored / re-stored.
- *dumprasterdata*: a boolean flag enabling or disabling creation of a raster data dump from GeoServer (geotiffs). If `false` (or `no`) raster data won't be stored / re-stored.
- *data_dt_filter*: {cmp_operator} {ISO8601} e.g. `> 2019-04-05T24:00` which means "include on backup archive only the files that have been modified later than 2019-04-05T24:00"
- *data_layername_filter*: comma separated list of layer names, optionally with glob syntax e.g.: `tuscany_*,italy`; Only RASTER original data and VECTORIAL table dumps matching those filters will be **included** into the backup ZIP archive
- *data_layername_exclude_filter*: comma separated list of layer names, optionally with glob syntax e.g.: `tuscany_*,italy`; The RASTER original data and VECTORIAL table dumps matching those filters will be **excluded** from the backup ZIP archive

Warning: Enabling these options **requires** the GeoServer Data Dir to be accessible and **fully writable** for the geonode and / or httpd server users when executing a backup / restore command.

Settings: [fixtures] Section

```
[fixtures]
#NOTE: Order is important
apps = people,account,avatar.avatar,base.backup,base.license,base.topiccategory,
↳base.region,base.resourcebase,base.contactrole,base.link,base.restrictioncodetype,
↳base.spatialrepresentation,guardian.userobjectpermission,guardian.
↳groupobjectpermission,layers.uploadsession,layers.style,layers.layer,layers.
↳attribute,layers.layerfile,maps.map,maps.maplayer,maps.mapsnapshot,documents.
↳document,taggit

dumps = people,accounts,avatars,backups,licenses,topiccategories,regions,
↳resourcebases,contactroles,links,restrictioncodetypes,spatialrepresentation,
↳userpermissions,grouppermissions,uploadsessions,styles,layers,attributes,
↳layerfiles,maps,maplayers,mapsnapshots,documents,tags
```

This section is the most complex one. Usually you don't need to modify it. Only an expert user who knows Python and GeoNode model structure should modify this section.

What its properties mean:

- *apps*; an ordered list of GeoNode Django applications. The backup / restore procedure will dump / restore the fixtures in a portable format.
- *dumps*; this is the list of `files` associated to the Django applications. The order **must** be the same as in the *apps* property above. Each name represents the `file name` where to dump to / read from the single app's fixtures.

Executing from the CLI

The following sections shows instructions on how to perform backup / restore from the command line by using the Django Admin Management Commands.

In order to obtain a basic user guide for the management command from the command line, just run

```
python manage.py backup --help
python manage.py restore --help
```

`--help` will provide the list of available command line options with a brief description.

By default both procedures activate *Read Only* mode, disabling any content modifying requests, which is reverted to the previous state (from before the execution) after finish, regardless of the command's result (success or failure). To disable activation of this mode, `--skip-read-only` argument can be passed to the command.

It is worth notice that both commands allows the following option

```
python manage.py backup --force / -f
python manage.py restore --force / -f
```

Which enables a non-interactive mode, meaning the user will not be asked for an explicit confirmation.

Backup

In order to perform a backup just run the command:

```
python manage.py backup --backup-dir=<target_bk_folder_path> --config=</path/
↪to/settings.ini>
```

The management command will automatically generate a `.zip` archive file on the target folder in case of success. In the target directory `.md5` file with the same name as backup will be created. It contains the MD5 hash of the backup file, which can be used to check archive's integrity before restoration.

It is worth to mention that `br` (Backup & Restore GeoNode application) will not be dumped, even if specified in the `settings.ini` as its content is strictly related to the certain GeoNode instance.

Currently, GeoNode does not support any automatic extraction of the backup file. It should be manually transferred, if needed to the target instance environment.

Restore

The `restore` command has a number of arguments, modifying its execution:

`-c` / `--config`: path to the `settings.ini` configuration file. If the Backup archive is provided with his settings, the latter will be used by the restore command and this option won't be mandatory anymore

1. `--skip-geoserver`: the GeoServer backup restoration won't be performed
2. `--skip-geoserver-info`: {Default: True} Skips GeoServer Global Infos, like the proxy base url and other global GeoServer metadata info
3. `--skip-geoserver-security`: {Default: True} Skips GeoServer all the Security Settings
4. `--backup-file`: (exclusive together with `--backup-files-dir`) path to the backup `.zip` archive
5. `--backup-files-dir`: (exclusive together with `--backup-file`) directory containing backup archives. The directory may contain a number of files, but **only** backup archives are allowed with a `.zip` extension. In case multiple archives are present in the directory, the newest one, created after the last already restored backup creation time, will be restored. This option was implemented with a thought of automated restores.
6. `--recovery-file`: Backup archive containing GeoNode data to restore in case of failure.
7. `-l` / `--with-logs`: the backup file will be checked against the restoration logs (history). In case this backup has already been restored (MD5 based comparison), `RuntimeError` is raised, preventing restore execution.
8. `-n` / `--notify`: the restore procedure outcome will be send by an e-mail notification to the superusers of the instance (note: notification will be sent to the superusers of the instance before restoration).
9. `--skip-read-only`: the restore procedure will be conducted without setting *Read Only* mode during execution.

In order to perform a default backup restoration just run the command:

```
python manage.py restore --backup-file=<target_restore_file_path> --config=</
↳path/to/settings.ini>
```

For restore to run it requires either `--backup-file` or `--backup-files-dir` argument defined.

Warning: The Restore will **overwrite** the whole target instances of GeoNode (and by default GeoServer) including users, catalog and database, so be very careful.

GeoNode Admin GUI Inspection

The history of restored backups can be verified in the admin panel.

Login to the admin panel and select `Restored backups` table from `BACKUP/RESTORE` application.

BACKUP/RESTORE

Restored backups

 View

A list will be displayed with a history of all restored backups. You can select a certain backup to view its data.

The detailed view of the restored backup shows backup archive's name, its MD5 hash, its creation/modification date (in the target folder), and the date of the restoration. Please note Restored Backup history cannot be modified.

Select restored backup to view

NAME	RESTORATION DATE	ARCHIVE MD5	CREATION DATE
2020-03-19_151016.zip	March 24, 2020, 1:33 p.m.	3dfdec2ac8df0ce78e2ef046bed2ef81	March 19, 2020, 3:11 p.m.
2020-03-19_151016.zip	March 24, 2020, 12:11 p.m.	3dfdec2ac8df0ce78e2ef046bed2ef81	March 19, 2020, 3:11 p.m.

2 restored backups

View restored backup

HISTORY

Name: 2020-03-19_151016.zip

Restoration date: March 24, 2020, 1:33 p.m.

Archive md5: 3dfdec2ac8df0ce78e2ef046bed2ef81

Creation date: March 19, 2020, 3:11 p.m.

Close

B/R in Docker environment

When executing B/R in the Docker environment, creation backup to / restoration from should be executed in / backup_restore directory. It is a shared volume between Geoserver and Geonode images, created for this purpose only. Pointing at another location will fail, as one of the images won't have an access to the files.

Warning: When executing B/R in Docker environment **remember** to create `settings.ini` file basing on `settings_docker_sample.ini` to point at a proper Geoserver data directory! In other case configuration mismatch may cause unexpected errors.

Warning: The only other volume shared between images is `/geoserver_data/data`, but backup creation **should not** be performed there, as the recursive Geoserver backups may be created in such case.

1.20 Viewer and Hooksets

1.20.1 GXP

1.20.2 MapStore 2

1.20.3 Leaflet

1.21 GeoNode Components and Architecture

1.21.1 Overview

TODO*

1.21.2 Django

TODO

1.21.3 WebServers

TODO

Apache

TODO

NGINX

TODO

1.21.4 GeoServer

TODO

1.21.5 Databases

TODO

1.21.6 OAuth2 Security: Authentication and Authorization

GeoNode interacts with GeoServer through an advanced security mechanism based on OAuth2 Protocol and GeoFence. This section is a walk through of the configuration and setup of GeoNode and GeoServer Advanced Security.

What we will see in this section is:

- **Introduction**
- **GeoNode (Security Backend):**
 1. Django Authentication
 2. Django OAuth Toolkit Setup and Configuration
 3. Details on `settings.py` Security Settings
- **GeoServer (Security Backend):**
 1. GeoServer Security Subsystem
 2. Introduction to the GeoServer OAuth2 Security Plugin
 3. Configuration of the GeoNode `REST Role Service`
 4. Configuration of the GeoNode `OAuth2 Authentication Filter`
 5. The GeoServer Authentication Filter Chains
 6. Introduction to GeoFence Plugin, the Advanced Security Framework for GeoServer
- **Troubleshooting and Advanced Features:**
 1. Common Issues and Fixes
 2. How to setup `HTTPS` secured endpoints
 3. GeoFence Advanced Features

Introduction

GeoServer, i.e. the geospatial backend server of GeoNode, is a spatial server which needs authenticated users in order to access protected resources or administration functions.

GeoServer supports several kind of Authentication and Authorization mechanisms. Those systems are pluggable and GeoServer can use them at the same time by the use of a `Filter Chain`. Briefly this mechanism allows GeoServer to check for different A&A protocols one by one. The first one matching is used by GeoServer to authorize the users.

GeoNode Authentication is based by default on Django Security Subsystem. Django authentication allows GeoNode to manage its internal users, groups, roles and sessions.

GeoNode has some external components, like GeoServer or QGIS Server, which are pluggable and stand-alone services, devoted to the management of geospatial data. Those external services have their own authentication and authorization mechanisms which must be synchronized somehow with the GeoNode one. Also, those external services maintain, in most of the cases and unless specific configuration does not disable this, alternative security access which for instance allow GeoNode to modify the geospatial catalog under the hood, or a system administrator to have independent and privileged access to the servers.

Before going deeply on how GeoServer/GeoNode A&A works and how it can be configured in order to work correctly with GeoNode, let's quickly clarify the difference between the `Authentication` and `Authorization` concepts.

Authentication

Authentication is the process of verifying the identity of someone through the use of some sort of credentials and a handshake protocol. If the credentials are valid, the authorization process starts. Authentication process always proceeds to Authorization process (although they may often seem to be combined). The two terms are often used synonymously but they are two different processes.

For more details and explanation about the authentication concepts, take a look [here](#).

Authorization

Authorization is the process of allowing authenticated users to access protected resources by checking its roles and rights against some sort of security rules mechanism or protocol. In other words it allows to control access rights by granting or denying specific permissions to specific authorized users.

GeoNode Security Backend

Django Authentication

The Django authentication system handles both authentication and authorization.

The auth system consists of:

1. Users
2. Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
3. Groups: A generic way of applying labels and permissions to more than one user.
4. A configurable password hashing system
5. Forms and view tools for logging in users, or restricting content
6. A pluggable backend system

The authentication system in Django aims to be very generic and doesn't provide some features commonly found in web authentication systems. Solutions for some of these common problems have been implemented in third-party packages:

1. Password strength checking
2. Throttling of login attempts
3. Authentication against third-parties (OAuth, for example)

Note: For more details on installation and configuration of Django authentication system, please refer to the official guide <https://docs.djangoproject.com/en/1.10/topics/auth/>.

GeoNode communicates with GeoServer through Basic Authentication under the hood, in order to configure the data and the GeoServer catalog.

In order to do this, you must be sure that GeoNode knows the **internal** admin user and password of GeoServer.

Warning: This must be an internal GeoServer user with admin rights, not a GeoNode one.

Make sure the credentials are correctly configured into the file `settings.py`

OGC_SERVER

Ensure that the OGC_SERVER settings are correctly configured.

Notice that the two properties `LOGIN_ENDPOINT` and `LOGOUT_ENDPOINT` must specify the GeoServer OAuth2 Endpoints (see details below). The default values `'j_spring_oauth2_geonode_login'` and `'j_spring_oauth2_geonode_logout'` work in most of the cases, unless you need some specific endpoints different from the later. In any case those values **must** be coherent with the GeoServer OAuth2 Plugin configuration.

If in doubt, please use the default values here below.

Default values are:

```
...
# OGC (WMS/WFS/WCS) Server Settings
# OGC (WMS/WFS/WCS) Server Settings
OGC_SERVER = {
    'default': {
        'BACKEND': 'geonode.geoserver',
        'LOCATION': GEOSERVER_LOCATION,
        'LOGIN_ENDPOINT': 'j_spring_oauth2_geonode_login',
        'LOGOUT_ENDPOINT': 'j_spring_oauth2_geonode_logout',
        # PUBLIC_LOCATION needs to be kept like this because in dev mode
        # the proxy won't work and the integration tests will fail
        # the entire block has to be overridden in the local_settings
        'PUBLIC_LOCATION': GEOSERVER_PUBLIC_LOCATION,
        'USER': 'admin',
        'PASSWORD': 'geoserver',
        'MAPFISH_PRINT_ENABLED': True,
        'PRINT_NG_ENABLED': True,
        'GEONODE_SECURITY_ENABLED': True,
        'WMST_ENABLED': False,
        'BACKEND_WRITE_ENABLED': True,
```

(continues on next page)

(continued from previous page)

```

        'WPS_ENABLED': False,
        'LOG_FILE': '%s/geoserver/data/logs/geoserver.log' % os.path.abspath(os.path.
→join(PROJECT_ROOT, os.pardir)),
        # Set to name of database in DATABASES dictionary to enable
        'DATASTORE': '', # 'datastore',
        'TIMEOUT': 10 # number of seconds to allow for HTTP requests
    }
}
...

```

GeoNode and GeoServer A&A Interaction

The GeoServer instance used by GeoNode, has a particular setup that allows the two frameworks to correctly interact and exchange informations on users credentials and permissions.

In particular GeoServer is configured with a `Filter Chain` for Authorization that makes use of the two following protocols:

1. **Basic Authentication; this is the default GeoServer Authentication mechanism. This makes use of [rfc2617 - Basic and Digest Authentication](#)**

In other words, GeoServer takes a username and a password encoded [Base64](#) on the HTTP Request Headers and compare them against its internal database (which by default is an encrypted XML file on the GeoServer Data Dir). If the user's credentials match, then GeoServer checks for Authorization through its `Role Services` (we will see those services in details on the *GeoServer (Security Backend)* section below).

Note: GeoServer ships by default with `admin` and `geoserver` as the default administrator user name and password. Before putting the GeoServer on-line it is imperative to change at least the administrator password.

2. **OAuth2 Authentication;** this module allows GeoServer to authenticate against the [OAuth2 Protocol](#). If the Basic Authentication fails, GeoServer falls back to this by using GeoNode as OAuth2 Provider by default.

Note: Further details can be found directly on the official GeoServer documentation at section “[Authentication Chain](#)”

From the **GeoNode backend (server) side**, the server will make use of **Basic Authentication** with administrator credentials to configure the GeoServer catalog. GeoServer must be reachable by GeoNode of course, and GeoNode must know the internal GeoServer admin credentials.

From the **GeoNode frontend (browser and GUI) side**, the *Authentication* goal is to allow GeoServer to recognize as valid a user which has been already logged into GeoNode, providing kind of an [SSO](#) mechanism between the two applications.

GeoServer must know and must be able to access GeoNode via HTTP/HTTPS. In other words, an external user connected to GeoNode must be authenticated to GeoServer with same permissions. This is possible through the **OAuth2 Authentication** Protocol.

GeoNode / GeoServer Authentication Mechanism

GeoNode as OAuth2 Provider (OP)

OpenID Connect is an identity framework built on OAuth 2.0 protocol which extends the authorization of OAuth 2.0 processes to implement its authentication mechanism. OpenID Connect adds a discovery

mechanism allowing users to use an external trusted authority as an identity provider. From another point of view, this can be seen as a single sign on (SSO) system.

OAuth 2.0 is an authorization framework which is capable of providing a way for clients to access a resource with restricted access on behalf of the resource owner. OpenID Connect allows clients to verify the users with an authorization server based authentication.

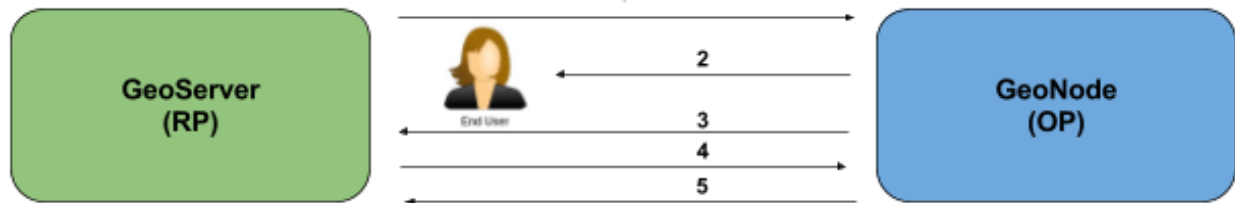
As an OP, GeoNode will be able to act as trusted identity provider, thus allowing the system working on an isolated environment and/or allow GeoNode to authenticate private users managed by the local Django auth subsystem.

GeoServer as OAuth2 Relying Party (RP)

Thanks to the **OAuth2 Authentication** GeoServer is able to retrieve an end user's identity directly from the OAuth2 Provider (OP).

With GeoNode acting as an OP, the mechanism will avoid the use of cookies relying, instead, on the OAuth2 secure protocol.

How the OAuth2 Protocol works:



1. The relying party sends the request to the OAuth2 provider to authenticate the end user
2. The OAuth2 provider authenticates the user
3. The OAuth2 provider sends the ID token and access token to the relying party
4. The relying party sends a request to the user info endpoint with the access token received from OAuth2 provider
5. The user info endpoint returns the claims.

GeoNode / GeoServer Authorization Mechanism

Allowing GeoServer to make use of a OAuth2 in order to act as an OAuth2 RP, is not sufficient to map a user identity to its roles though.

On GeoServer side we will still need to a `RoleService` which would be able to talk to GeoNode and transform the tokens into a User Principal to be used within the GeoServer Security subsystem itself.

In other words after a successful Authentication, GeoServer needs to Authorize the user in order to understand which resources he is enable to access or not. A `REST` based `RoleService` on GeoNode side, allows GeoServer to talk to GeoNode via `REST` to get the current user along with the list of its Roles.

Nevertheless knowing the Roles associated to a user is not sufficient. The complete GeoServer Authorization needs to catch a set of `Access Rules`, associated to the Roles, in order to establish which resources and data are accessible by the user.

The GeoServer Authorization is based on Roles only, therefore for each authenticated user we need also to know:

1. The Roles associated to a valid user session
2. The access permissions associated to a GeoServer Resource

The Authentication mechanism above allows GeoServer to get information about the user and his Roles, which addresses point 1.

About point 2, GeoServer makes use of the [GeoFence Embedded Server](#) plugin. GeoFence is a java web application that provides an advanced authentication / authorization engine for GeoServer using the interface described in [here](#). GeoFence has its own rules database for the management of Authorization rules, and overrides the standard GeoServer security management system by implementing a sophisticated Resource Access Manager. Least but not last, GeoFence implements and exposes a [REST API](#) allowing remote authorized clients to read / write / modify security rules.

The advantages using such plugin are multiple:

1. The Authorizations rules have a fine granularity. The security rules are handled by GeoFence in a way similar to the iptables ones, and allow to define security constraints even on sub-regions and attributes of layers.
2. GeoFence exposes a REST interface to its internal rule database, allowing external managers to update the security constraints programmatically
3. GeoFence implements an internal caching mechanism which improves considerably the performances under load.

GeoNode interaction with GeoFence

GeoNode itself is able to push/manage Authorization rules to GeoServer through the GeoFence [REST API](#), acting as an administrator for GeoServer. GeoNode properly configures the GeoFence rules anytime it is needed, i.e. the permissions of a Resource / Layer are updated.

GeoServer must know and must be able to access GeoNode via HTTP/HTTPS. In other words, an external user connected to GeoNode must be authenticated to GeoServer with same permissions. This is possible through the **GeoNodeCookieProcessingFilter**.

Summarizing we will have different ways to access GeoNode Layers:

1. Through GeoNode via Django Authentication and **GeoNodeCookieProcessingFilter**; basically the users available in GeoNode are also valid for GeoServer or any other backend.

Warning: If a GeoNode user has “administrator” rights, he will be able to administer GeoServer too.

2. Through GeoServer Security Subsystem; it will be always possible to access to GeoServer using its internal security system and users, unless explicitly disabled (**warning** this is dangerous, you must know what you are doing).

Let’s now see in details how the single pieces are configured and how they can be configured.

Django OAuth Toolkit Setup and Configuration

As stated above, GeoNode makes use of the OAuth2 protocol for all the frontend interactions with GeoServer. GeoNode must be configured as an OAuth2 Provider and provide a Client ID and a Client Secret key to GeoServer. This is possible by enabling and configuring the [Django OAuth Toolkit Plugin](#).

Warning: GeoNode and GeoServer won’t work at all if the following steps are not executed at the first installation.

Default settings.py Security Settings for OAuth2

Double check that the OAuth2 Provider and Security Plugin is enabled and that the settings below are correctly configured.

AUTH_IP_WHITELIST

AUTH_IP_WHITELIST property limits access to users/groups REST Role Service endpoints to the only whitelisted IP addresses. Empty list means ‘allow all’. If you need to limit ‘api’ REST calls to only some specific IPs fill the list like this: AUTH_IP_WHITELIST = ['192.168.1.158', '192.168.1.159']

Default values are:

```
...
AUTH_IP_WHITELIST = []
...
```

INSTALLED_APPS

In order to allow GeoNode to act as an OAuth2 Provider, we need to enable the `oauth2_provider` Django application provided by the “Django OAuth Toolkit”.

Default values are:

```
...
INSTALLED_APPS = (

    'modeltranslation',

    ...
    'guardian',
    'oauth2_provider',
    ...

) + GEONODE_APPS
...
```

MIDDLEWARE_CLASSES

Installing the `oauth2_provider` Django application is not sufficient to enable the full functionality. We need also GeoNode to include additional entities to its internal model.

Default values are:

```
...
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',

    # The setting below makes it possible to serve different languages per
    # user depending on things like headers in HTTP requests.
    'django.middleware.locale.LocaleMiddleware',
```

(continues on next page)

(continued from previous page)

```

'pagination.middleware.PaginationMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',

# If you use SessionAuthenticationMiddleware, be sure it appears before
→OAuth2TokenMiddleware.
# SessionAuthenticationMiddleware is NOT required for using django-oauth-toolkit.
'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
'oauth2_provider.middleware.OAuth2TokenMiddleware',
)
...

```

AUTHENTICATION_BACKENDS

In order to allow GeoNode to act as an OAuth2 Provider, we need to enable the `oauth2_provider.backends.OAuth2Backend` Django backend provided by the “Django OAuth Toolkit”. Also notice that we need to specify the OAuth2 Provider scopes and declare which generator to use in order to create OAuth2 Client IDs.

Default values are:

```

...
# Replacement of default authentication backend in order to support
# permissions per object.
AUTHENTICATION_BACKENDS = (
    'oauth2_provider.backends.OAuth2Backend',
    'django.contrib.auth.backends.ModelBackend',
    'guardian.backends.ObjectPermissionBackend',
)

OAUTH2_PROVIDER = {
    'SCOPES': {
        'read': 'Read scope',
        'write': 'Write scope',
        'groups': 'Access to your groups'
    },

    'CLIENT_ID_GENERATOR_CLASS': 'oauth2_provider.generators.ClientIdGenerator',
}
...

```

Django OAuth Toolkit Admin Setup

Once the `settings.py` and `local_settings.py` have been correctly configured for your system:

1. Complete the GeoNode setup steps

- Prepare the model

```

python manage.py makemigrations
python manage.py migrate
python manage.py syncdb

```

- Prepare the static data

```
python manage.py collectstatic
```

- Make sure the database has been populated with initial default data

Warning: *Deprecated* this command will be replaced by migrations in the future, so be careful.

```
python manage.py loaddata initial_data.json
```

- Make sure there exists a superuser for your environment

Warning: *Deprecated* this command will be replaced by migrations in the future, so be careful.

```
python manage.py createsuperuser
```

Note: Read the base tutorials on GeoNode Developer documentation for details on the specific commands and how to use them.

2. Start the application

Start GeoNode accordingly on how the setup has been done; run debug mode through `paver`, or proxied by an HTTP Server like Apache2 HTTPD, Nginx or others.

3. Finalize the setup of the OAuth2 Provider

First of all you need to configure and create a new OAuth2 Application called `GeoServer` through the GeoNode Admin Dashboard

- Access the GeoNode Admin Dashboard
- Go to Django OAuth Toolkit > Applications
- Update or create the Application named `GeoServer`

Warning: The Application name **must** be `GeoServer`











- `Client id`; An alphanumeric code representing the OAuth2 Client Id. `GeoServer` OAuth2 Plugin **will** use **this** value.

Warning: In a production environment it is **highly** recommended to modify the default value provided with GeoNode installation.

- `User`; Search for the admin user. Its ID will be automatically updated into the form.

Menu




-  Upload Layers
-  Profile
-  Recent Activity
-  Inbox
-  Announcements
-  Remote Services
-  Invite User
-  GeoServer
-  **Admin**
-  Help

Log out

Django OAuth Toolkit	
Access tokens	+ Add ✎ Change
Applications	+ Add ✎ Change
Grants	+ Add ✎ Change
Refresh tokens	+ Add ✎ Change

Change application

Client id:	<input type="text" value="Jrchz2oPY3akmzndmgUTYrs9gcZlgoV2I"/>
User:	<input type="text" value="2"/>  admin
Redirect uris:	<div><div>http://localhost:8080/geoserver http://localhost:8080/geoserver/ http://<host_name_or_ip>/geoserver http://<host_name_or_ip>/geoserver/</div><div>Allowed URIs list, space separated</div></div>
Client type:	<input type="text" value="Confidential"/>
Authorization grant type:	<input type="text" value="Authorization code"/>
Client secret:	<input type="text" value="rCnp5txobUo83EpQEblM8fVj3QT5zb5ql"/>
Name:	<input type="text" value="GeoServer"/>
<input type="checkbox"/> Skip authorization	

- Redirect uris; It is possible to specify many URIs here. Those must coincide with the GeoServer instances URIs.
- Client type; Choose Confidential
- Authorization grant type; Choose Authorization code
- Client secret; An alphanumeric code representing the OAuth2 Client Secret. GeoServer OAuth2 Plugin **will** use **this** value.

Warning: In a production environment it is **highly** recommended to modify the default value provided with GeoNode installation.

- Name; **Must** be GeoServer

GeoServer Security Backend

GeoServer Security Subsystem

GeoServer has a robust security subsystem, modeled on Spring Security. Most of the security features are available through the Web administration interface.

For more details on how this works and how to configure and modify it, please refer to the official GeoServer guide <http://docs.geoserver.org/stable/en/user/security/webadmin/index.html>

By using the GeoServer Data Dir provided with GeoNode build, the following configuration are already available. You will need just to update them accordingly to your environment (like IP addresses and Host names, OAuth2 Keys, and similar things). However it is recommended to read carefully all the following passages in order to understand exactly how the different component are configured and easily identify any possible issue during the deployment.

The main topics of this section are:

1. Connection to the GeoNode REST Role Service
2. Setup of the GeoServer OAuth2 Authentication Filter
3. Configuration of the GeoServer Filter Chains
4. Setup and test of the GeoFence Server and Default Rules

Connection to the GeoNode REST Role Service

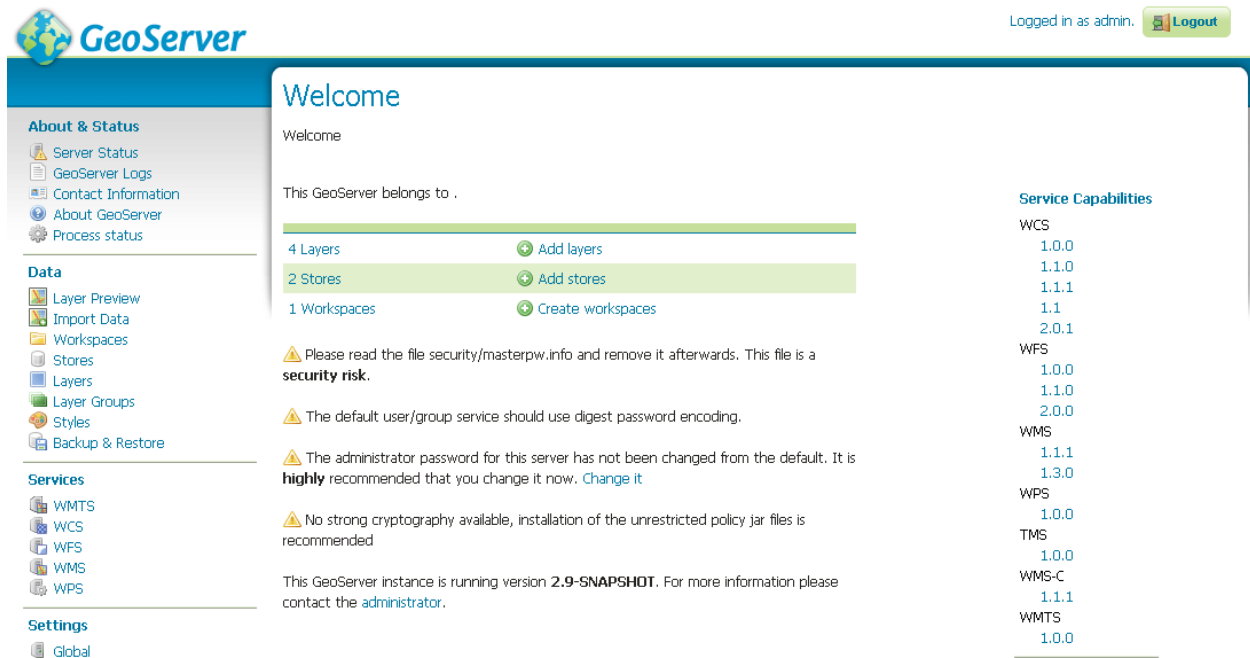
Preliminary checks

- GeoServer is up and running and you have admin rights
- GeoServer must reach the GeoNode instance via HTTP
- The GeoServer Host IP Address must be allowed to access the GeoNode Role Service APIs (see the section AUTH_IP_WHITELIST above)

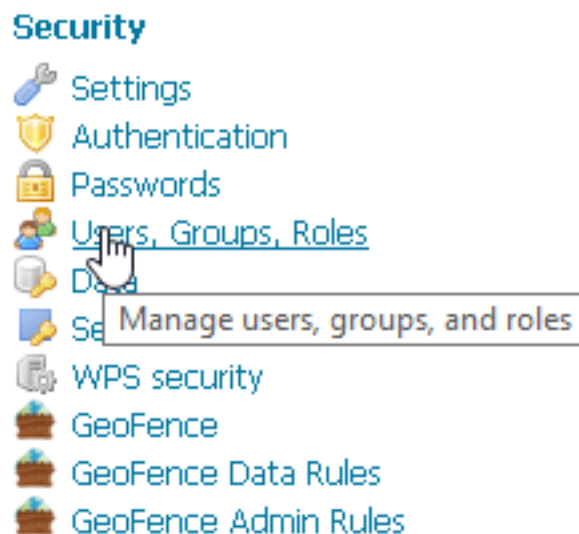
Setup of the GeoNode REST Role Service

1. Login as admin to the GeoServer GUI

Warning: In a production system remember to change the default admin credentials `admin` `geoserver`



2. Access the Security > Users, Groups, Roles section



3. If not yet configured the service geonode REST role service, click on Role Services > Add

new

Note: This passage is **not** needed if the `geonode REST role service` has been already created. If so it will be displayed among the Role Services list

Role Services

[+ Add new](#)
[- Remove selected](#)

Search

<input type="checkbox"/>	Name	Type	Administrator Role
<input type="checkbox"/>	default	Default XML role service	ADMIN
<input type="checkbox"/>	geonode REST role service	AuthKEY REST Role Service	ROLE_ADMIN

<< < 1 > >> Results 1 to 2 (out of 2 items)

Users, Groups, and Roles

Manage user group and role services

Services Users/Groups Roles

User Group Services

[+ Add new](#)
[- Remove selected](#)

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	default	Default XML user/group service

<< < 1 > >> Results 1 to 1 (out of 1 ite

Role Services

[+ Add new](#)
[- Remove selected](#)

- If not yet configured the service `geonode REST role service`, choose `AuthKEY REST - Role`

service from REST endpoint

New Role Service

Create and configure a new Role Service

XML - Default role service stored as XML

J2EE - Role service extracting roles from web.xml

AuthKEY REST - Role service from REST endpoint

JDBC - Role service stored in database

LDAP - Role service stored in LDAP repository

5. Create / update the geonode REST role service accordingly

- Name; **Must be** geonode REST role service
- Base Server URL; **Must point to the GeoNode instance base URL** (e.g. `http://<geonode_host_url>`)
- Roles REST Endpoint; **Enter** `/api/roles`
- Admin Role REST Endpoint; **Enter** `/api/adminRole`
- Users REST Endpoint; **Enter** `/api/users`
- Roles JSON Path; **Enter** `$.groups`
- Admin Role JSON Path; **Enter** `$.adminRole`
- Users JSON Path; **Enter** `$.users[0].groups`

Once everything has been setup and it is working, choose the Administrator role and Group administrator role as `ROLE_ADMIN`

Allow GeoFence to validate rules with ROLES

Warning: The following instruction are different accordingly to the GeoServer version you are currently using.

GeoServer 2.9.x and 2.10.x

1. Access the Security > Settings section
2. Choose the geonode REST role service as Active role service

AuthKEY REST Role Service

Role service from REST endpoint

Settings

Roles

Name

geonode REST role service

Administrator role

ROLE_ADMIN ▼

Group administrator role

ROLE_ADMIN ▼

REST Role Service Settings

Base Server URL

http://<geonode_host_url>

Roles REST Endpoint

/api/roles

Admin Role REST Endpoint

/api/adminRole

Users REST Endpoint

/api/users

Roles JSON Path

\$.groups

Admin Role JSON Path

\$.adminRole

Users JSON Path

\$.users

Security

-  [Settings](#)
-  [Authentication](#)
-  [Passwords](#) Configure global security settings
-  [Users, Groups, Roles](#)
-  [Data](#)

Security Settings

Configure security settings

Active role service

geonode REST role service ▼

default

geonode REST role service

☐ Encrypt web admin URL parameters

Password encryption

Weak PBE ▼

⚠ No strong cryptography available

Save

Cancel

GeoServer 2.12.x and above

With the latest updates to GeoFence Plugin, the latter no more recognizes the Role Service from the default settings but from the `geofence-server.properties` file.

That said, it is important that the `Security > Settings` role service will be set to **default**, in order to allow GeoServer following the standard authorization chain.

On the other side, you will need to be sure that the `geofence-server.properties` file under the `$GEOSERVER_DATA_DIR/geofence` folder, contains the two following additional properties:

```
gwc.context.suffix=gwc
org.geoserver.rest.DefaultUserGroupName=geonode REST role service
```

Setup of the GeoServer OAuth2 Authentication Filter

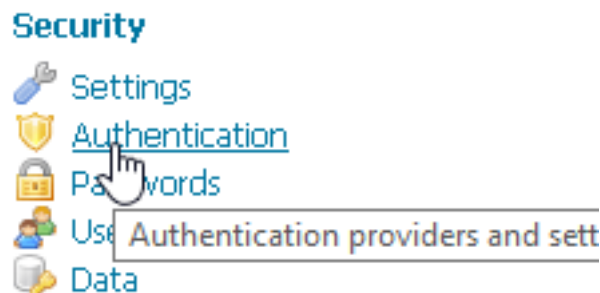
It is necessary now check that GeoServer can connect to OAuth2 Providers (specifically to GeoNode OP), and being able to Authenticate users through it.

Preliminary checks

- GeoServer is up and running and you have admin rights
- GeoServer must reach the GeoNode instance via HTTP
- OAuth2 Client ID and Client Secret have been generated on GeoNode and known

Setup of the GeoNode OAuth2 Security Filter

1. Access the `Security > Authentication` section



2. If **not yet configured** the Authentication Filter `geonode-oauth2 - Authentication` using a GeoNode OAuth2, click on `Authentication Filters > Add new`

Note: This passage is **not** needed if the `geonode-oauth2 - Authentication` using a GeoNode OAuth2 has been already created. If so it will be displayed among the Authentication Filters list

Authentication Filters

[+ Add new](#)[- Remove selected](#)

Search	
<input type="checkbox"/> Name	Type
<input type="checkbox"/> anonymous	Anonymous authentication
<input type="checkbox"/> basic	Basic HTTP authentication
<input type="checkbox"/> form	Form authentication
<input type="checkbox"/> geonode-oauth2	Authentication using a GeoNode OAuth2
<input type="checkbox"/> geonodeAnonymousFilter	org.geonode.security.GeoNodeAnonymousProcessingFilter
<input type="checkbox"/> geonodeCookieFilter	org.geonode.security.GeoNodeCookieProcessingFilter
<input type="checkbox"/> rememberme	Remember me authentication

<< < 1 > >> Results 1 to 7 (out of 7 items)

Authentication Filters



3. If **not yet configured** the Authentication Filter `geonode-oauth2` - Authentication using a GeoNode OAuth2, choose `GeoNode OAuth2` - Authenticates by looking up for a valid GeoNode OAuth2 `access_token` key sent as URL parameter

New Authentication Filter

Create and configure a new Authentication Filter




`J2EE` - Delegates to servlet container for authentication`GeoNode OAuth2` - Authenticates by looking up for a valid GeoNode OAuth2

4. Create / update the `geonode-oauth2` - Authentication using a GeoNode OAuth2 accordingly

- Name; **Must be** `geonode-oauth2`
- Enable Redirect Authentication EntryPoint; It is recommended to put this to `False`, otherwise GeoServer won't allow you to connect to its Admin GUI through the `Form` but only through GeoNode
- Login Authentication EndPoint; Unless you have specific needs, keep the default value `/j_spring_oauth2_geonode_login`

Authentication using a GeoNode OAuth2 geonode-oauth2

Authenticates by looking up for a valid GeoNode OAuth2 access_token key sent as URL parameter

Name	<input type="text" value="geonode-oauth2"/>	
OAuth2 provider connection	<input type="checkbox"/>	
Enable Redirect Authentication EntryPoint	<input type="checkbox"/>	
Login Authentication EndPoint	<input type="text" value="/j_spring_oauth2_geonode_login"/>	
Logout Authentication EndPoint	<input type="text" value="/j_spring_oauth2_geonode_logout"/>	
Force Access Token URI HTTPS Secured Protocol	<input type="checkbox"/>	
Access Token URI	<input type="text"/>	

- Logout Authentication EndPoint; Unless you have specific needs, keep the default value `/j_spring_oauth2_geonode_logout`
- Force Access Token URI HTTPS Secured Protocol; This must be False unless you enabled a Secured Connection on GeoNode. In that case you will need to trust the GeoNode Certificate on the GeoServer JVM Keystore. Please see details below
- Access Token URI; Set this to `http://<geonode_host_base_url>/o/token/`
- Force User Authorization URI HTTPS Secured Protocol; This must be False unless you enabled a Secured Connection on GeoNode. In that case you will need to trust the GeoNode Certificate on the GeoServer JVM Keystore. Please see details below
- User Authorization URI; Set this to `http://<geonode_host_base_url>/o/authorize/`
- Redirect URI; Set this to `http://<geoserver_host>/geoserver`. This address **must** be present on the Redirect uris of GeoNode OAuth2 > Applications > GeoServer (see above)
- Check Token Endpoint URL; Set this to `http://<geonode_host_base_url>/api/o/v4/tokeninfo/`
- Logout URI; Set this to `http://<geonode_host_base_url>/account/logout/`
- Scopes; Unless you have specific needs, keep the default value `read,write,groups`
- Client ID; The Client id alphanumeric key generated by the GeoNode OAuth2 > Applications > GeoServer (see above)
- Client Secret; The Client secret alphanumeric key generated by the GeoNode OAuth2 > Applications > GeoServer (see above)
- Role source; In order to authorize the user against GeoNode, choose Role service > geonode REST role service

Configuration of the GeoServer Filter Chains

The following steps ensure GeoServer can adopt more Authentication methods. As stated above, it is possible to Authenticate to GeoServer using different protocols.

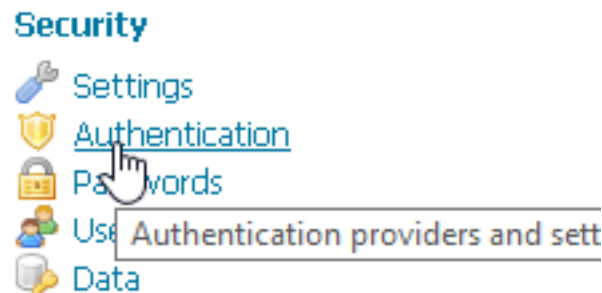
GeoServer scans the authentication filters chain associated to the specified path and tries them one by one sequentially. The first one matching the protocol and able to grant access to the user, breaks the cycle by creating a `User Principal` and injecting it into the `GeoServer SecurityContext`. The Authentication process, then, ends here and the control goes to the Authorization one, which will try to retrieve the authenticated user's Roles through the available GeoServer Role Services associated to the Authentication Filter that granted the access.

Preliminary checks

- GeoServer is up and running and you have admin rights
- GeoServer must reach the GeoNode instance via HTTP
- The `geonode-oauth2` - Authentication using a GeoNode OAuth2 Authentication Filter and the `geonode REST` role service have been correctly configured

Setup of the GeoServer Filter Chains

1. Access the `Security > Authentication` section













2. Identify the section `Filter Chains`
3. Make sure the web `Filter Chain` is configured as shown below





Warning: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.





Filter Chains

 [Add service chain](#)

 [Add HTML chain](#)

Position	Name	Patterns
	web	/web/**,/gwc/rest/web/**,/
 	webLogin	/j_spring_security_check,/j_spring_security_check/,/
 	webLogout	/j_spring_security_logout,/j_spring_security_logout/,/
 	rest	/rest/**
 	gwc	/gwc/rest/**
	default	/**



1


 Results 1 to 6 (out of 6 items)

Available		Selected
basic geonodeAnonymousFilter geonodeCookieFilter	   	geonode-oauth2 rememberme form anonymous

☐ `georence` `org.geoserver.geoserver.auth`
☒ `geonodeAuthProvider` `org.geonode.security.GeoNodeAuthProvider`

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available
geonodeAuthProvider

⇒
 ⇐
 ⇑
 ⇓

Save Cancel

4. Make sure the `rest` Filter Chain is configured as shown below

Available		Selected
geonodeAnonymousFilter	⇒	basic
geonodeCookieFilter	⇐	geonode-oauth2
	⇑	anonymous
	⇓	

Warning: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

☐ georence org.geoserver.geoserver.auth
☒ geonodeAuthProvider org.geonode.security.GeoNodeAuthProvider

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available	
geonodeAuthProvider	⇒ ⇐ ⬆ ⬇

5. Make sure the gwc Filter Chain is configured as shown below

Available		Selected
anonymous geonodeAnonymousFilter geonodeCookieFilter	⇒ ⇐ ⬆ ⬇	basic geonode-oauth2

Warning: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

☐ georence org.geoserver.geoserver.auth
☒ geonodeAuthProvider org.geonode.security.GeoNodeAuthProvider

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available
geonodeAuthProvider

⇒
 ⇐
 ⬆
 ⬇

Save Cancel

6. Make sure the default Filter Chain is configured as shown below

Available		Selected
geonodeAnonymousFilter	⇒	basic
geonodeCookieFilter	⇐	geonode-oauth2
	⬆	anonymous
	⬇	

Warning: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

The screenshot shows a web interface for configuring authentication providers. At the top, there is a list of providers with checkboxes. The first provider is 'georence' with the class 'org.geoserver.geoserver.authr'. The second provider, 'geonodeAuthProvider', is highlighted in green and has the class 'org.geonode.security.GeoNodeAuthProvider'. Below this list are navigation buttons: '<<', '<', '1', '>', and '>>', followed by the text 'Results 1 to 3 (out of 3 items)'. The main section is titled 'Provider Chain'. It contains a table with a header 'Available' and a single row containing 'geonodeAuthProvider'. To the right of the table are four buttons: '⇒', '⇐', '⇕', and '⇓'. At the bottom of the interface are two buttons: 'Save' and 'Cancel'. A mouse cursor is pointing at the 'Save' button.

Available
geonodeAuthProvider

7. Add the GeoNode Login Endpoints to the comma-delimited list of the webLogin Filter Chain

Warning: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

Filter chain

Configure an individual filter chain

Chain settings

Name

Comma delimited list of ANT patterns (with optional query string)

- ☐ Disable security for this chain
- ☐ Allow creation of an HTTP session for storing the authentication token
- ☐ Accept only SSL requests

Role filter

☐ georence org.geoserver.geoserver.auth
☒ geonodeAuthProvider org.geonode.security.GeoNodeAuthProvider

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available
geonodeAuthProvider

Save Cancel

8. Add the GeoNode Logout Endpoints to the comma-delimited list of the webLogout Filter Chain

Warning: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

Filter chain

Configure an individual filter chain

Chain settings

Name

Comma delimited list of ANT patterns (with optional query string)

- ☐ Disable security for this chain
- ☐ Allow creation of an HTTP session for storing the authentication
- ☐ Accept only SSL requests

Role filter

9. Add the GeoNode Logout Endpoints to the comma-delimited list of the `formLogoutChain` XML node in `<GEOSERVER_DATA_DIR>/security/filter/formLogout/config.xml`

You will need a text editor to modify the file.

Note: If the `<formLogoutChain>` XML node does not exist at all, create a **new one** as specified below

```
<logoutFilter>
...
<redirectURL>/web/</redirectURL>
<formLogoutChain>/j_spring_security_logout,/j_spring_security_logout/,/
↪j_spring_oauth2_geonode_logout,/j_spring_oauth2_geonode_logout/</
↪formLogoutChain>
</logoutFilter>
```

Warning: The value `j_spring_oauth2_geonode_logout` **must** be the same specified as Logout Authentication EndPoint in the `geonode-oauth2 - Authentication` using a GeoNode OAuth2 above.

Setup and test of the GeoFence Server and Default Rules

In order to work correctly, GeoServer needs the [GeoFence Embedded Server](#) plugin to be installed and configured on the system.

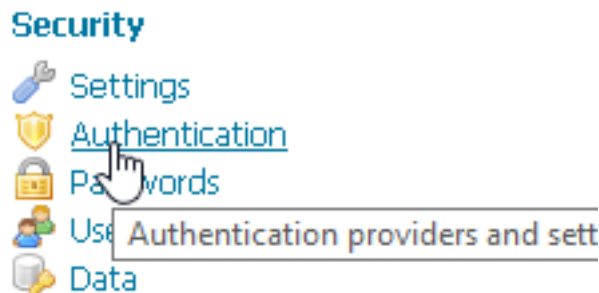
The GeoServer configuration provided for GeoNode, has the plugin already installed with a default configuration. In that case double check that the plugin works correctly and the default rules have been setup by following the next steps.

Preliminary checks

- GeoServer is up and running and you have admin rights
- The [GeoFence Embedded Server](#) plugin has been installed on GeoServer

Setup of the GeoServer Filter Chains

1. Access the `Security > Authentication` section



2. Identify the section `Authentication Providers` and make sure the `geofence Authentication Provider` is present

Authentication Providers ?

+ Add new - Remove selected

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	default	Basic username/password authentication
<input type="checkbox"/>	geofence	org.geoserver.geoserver.authentication.auth.GeoFenceAuthenticationProvider
<input type="checkbox"/>	geonodeAuthProvider	org.geonode.security.GeoNodeAuthenticationProvider

<<
 <

>
 >>
 Results 1 to 3 (out of 3 items)

3. Make sure the Provider Chain is configured as shown below

Provider Chain

Available		Selected
geonodeAuthProvider	⇒	default
	⇐	geofence
	↑↑	
	↓↓	

Warning: Every time you modify an Authentication Providers, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

☐ geofence org.geoserver.geoserver.auth

☒ geonodeAuthProvider org.geonode.security.GeoNode

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available	
geonodeAuthProvider	⇒
	⇐
	↑↑
	↓↓

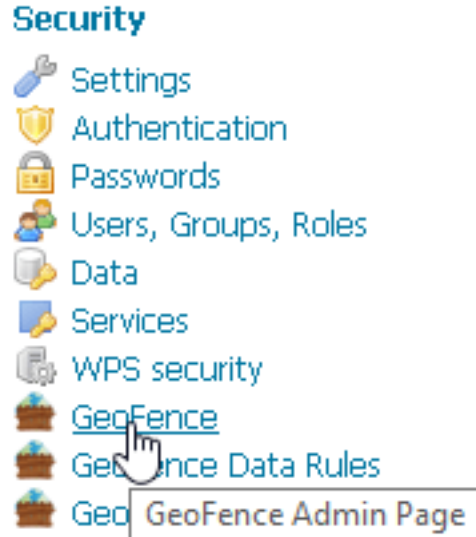
Save

Cancel

Setup of the GeoFence Server and Rules

1. Make sure GeoFence server works and the default settings are correctly configured

- Access the Security > GeoFence section



- Make sure the Options are configured as follows and the server works well when performing a Test Connection
 - Allow remote and inline layers in SLD; Set it to True
 - Allow SLD and SLD_BODY parameters in requests; Set it to True
 - Authenticated users can write; Set it to True
 - Use GeoServer roles to get authorizations; Set it to False

2. Check the GeoFence default Rules

- Access the Security > GeoFence Data Rules section
- Make sure the DENY ALL Rule is present by default, otherwise your data will be accessible to everyone

Note: This rule is **always** the last one

Warning: If that rule does not exists **at the very bottom** (this rule is **always** the last one), add it manually.

- Access the Security > GeoFence Admin Rules section
- No Rules needed here

Connection successful

GeoFence Admin Page

GeoFence options Administration Page

General settings

GeoServer Instance name for GeoFence

default-gs

GeoFence services URL (GeoServer restart is required if changed)

internal:/

Test Connection

Options

- ☒ Allow remote and inline layers in SLD
- ☒ Allow SLD and SLD_BODY parameters in requests
- ☒ Authenticated users can write
- ☐ Use GeoServer roles to get authorizations



GeoFence Data Rules

Configure data rules for the internal GeoFence server.

- Add new rule
- Remove selected rules

<< < 1 > >>

Results 1 to 1 (out of 1 items)

Search

<input type="checkbox"/>	P	Role	User	Service	Request	Workspace	Layer	Access	
<input type="checkbox"/>	0	*	*	*	*	*	*	DENY	

<< < 1 > >>

Results 1 to 1 (out of 1 items)



GeoFence Admin Rules

Configure admin rules for the internal GeoFence server.

-  Add new rule
-  Remove selected rules

<< < > >> Results 0 to 0 (out of 0 items)					<input type="text" value="Search"/>	
	P	Role	User	Workspace	Access	
<< < > >> Results 0 to 0 (out of 0 items)						

Troubleshooting and Advanced Features

Common Issues and Fixes

- GeoServer/GeoNode OAuth2 does not authenticate as Administrator even using GeoNode admin users

Symptoms

When trying to authenticate with an `admin` user using OAuth2, the process correctly redirects to GeoServer page but I'm not a GeoServer Administrator.

Cause

That means that somehow GeoServer could not successfully complete the Authorization and Authentication process.

The possible causes of the problem may be the following ones:

1. The OAuth2 Authentication fails on GeoServer side

This is usually due to an exception while trying to complete the Authentication process.

- A typical cause is that GeoServer tries to use HTTPS connections but the GeoNode certificate is not trusted;

In that case please refer to the section below. Also take a look at the logs (in particular the GeoServer one) as explained in `debug_geonode`. The GeoServer logs should contain a detailed Exception explaining the cause of the problem. If no exception is listed here (even after raised the log level to *DEBUG*), try to check for the GeoNode Role Service as explained below.

- Another possible issue is that somehow the OAuth2 handshake cannot complete successfully;

1. Login into GeoServer as administrator through its WEB login form.

2. Double check that all the `geonode-oauth2 - Authentication` using a GeoNode OAuth2 parameters are correct. If everything is ok, take a look at the logs (in particular the GeoServer one) as explained in `debug_geonode`. The GeoServer logs should contain a detailed Exception explaining the cause of the problem. If no exception is listed here (even after raised the log level to *DEBUG*), try to check for the GeoNode Role Service as explained below.

2. GeoServer is not able to retrieve the user Role from a Role Service

Always double check both HTTP Server and GeoServer log as specified in section `debug_geonode`. This might directly guide you to the cause of the problem.

- Check that the GeoServer host is granted to access GeoNode Role Service REST APIs in the `AUTH_IP_WHITELIST` of the `settings.py`
- Check that the `geonode REST role service` is the default Role service and that the GeoServer OAuth2 Plugin has been configured to use it by default
- Check that the GeoNode REST Role Service APIs are functional and produce correct JSON.

This is possible by using simple `cUrl` GET calls like

```
curl http://localhost/api/adminRole
$> {"adminRole": "admin"}

curl http://localhost/api/users
$> {"users": [{"username": "AnonymousUser", "groups":
↪": ["anonymous"]}, {"username": "afabiani",
↪"groups": ["anonymous", "test"]}, {"username":
↪"admin", "groups": ["anonymous", "test", "admin"]}
↪]}

curl http://localhost/api/roles
$> {"groups": ["anonymous", "test", "admin"]}

curl http://localhost/api/users/admin
$> {"users": [{"username": "admin", "groups": [
↪"anonymous", "test", "admin"]}]}
```

How to setup HTTPS secured endpoints

In a production system it is a good practice to encrypt the connection between GeoServer and GeoNode. That would be possible by enabling HTTPS Protocol on the GeoNode REST Role Service APIs and OAuth2 Endpoints.

Most of the times you will rely on a self-signed HTTPS connection using a generated certificate. That makes the connection *untrusted* and you will need to tell to the GeoServer Java Virtual Machine to trust it.

This can be done by following the steps below.

For any issue take a look at the logs (in particular the GeoServer one) as explained in `debug_geonode`. The GeoServer logs should contain a detailed Exception explaining the cause of the problem.

SSL Trusted Certificates

When using a custom `Keystore` or trying to access a non-trusted or self-signed SSL-protected OAuth2 Provider from a non-SSH connection, you will need to add the certificates to the JVM `Keystore`.

In order to do this you can follow the next steps:

In this example we are going to

1. Retrieve SSL Certificate from GeoNode domain:

“Access Token URI” = https://<geonode_host_base_url>/o/token/ therefore we need to trust https://<geonode_host_base_url> or (<geonode_host_base_url>:443)

Note: You will need to get and trust certificates from every different HTTPS URL used on OAuth2 Endpoints.

2. Store SSL Certificates on local hard-disk
 3. Add SSL Certificates to the Java Keystore
 4. Enable the JVM to check for SSL Certificates from the Keystore
1. Retrieve the SSL Certificate from GeoNode domain

Use the `openssl` command in order to dump the certificate

For https://<geonode_host_base_url>

```
openssl s_client -connect <geonode_host_base_url>:443
```

```
Loading 'screen' into random state - done
CONNECTED(00000188)
depth=3 C = US, O = Equifax, OU = Equifax Secure Certificate Authority
verify return:1
depth=2 C = US, O = GeoTrust Inc., CN = GeoTrust Global CA
verify return:1
depth=1 C = US, O = Google Inc, CN = Google Internet Authority G2
verify return:1
depth=0 C = US, ST = California, L = Mountain View, O = Google Inc, CN = accounts.google.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=accounts.google.com
  i:/C=US/O=Google Inc/CN=Google Internet Authority G2
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority G2
  i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
 2 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
  i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEoTCCA4mgAwIBAgIIeEP870cN1RQwDQYJKoZIhvcNAQELBQAwSTELMAkGA1UE
BhMCVVMxZzARBgNVBAMoTCKdvb2dsZSBjb250bWxJTAjBgNVBAMTHEdwb2dsZSBjb250bWxJ
cm5ldCBDb2R3JpdHkgRzIwHhcNMTYxMDE5MTcxNjU3WbcNMTcxMTEwMTcxMzAw
WjBtMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcn5pYTEwMBQGA1UEBwwN
TW91bnRhaW4gVmlldzETMBEGA1UECgwKR29vZ2x1IEluYzEcmBoGA1UEAwwTYWNj
b3VudHMuz29vZ2x1LmVudTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AK71x4DYCS7NcmQyp8UAq7067Wb7LLhkgE5QJcUVdvFfLrXmG3PAIcflMvdIK+F
RODn5d79I18qgn90XgB/xyFgx381MR/HoglavblCTfgbiotvhaMsxyY/h55efbcn
i2GGOXCVcx119xPOnWfvd7aUaOLSutMeE27T4Hqj3o0WFYVJq+dQNGdkGwqtUap
bosQFuyWg+rKINM6Opvy6ay8fRU/4JeKNVcaZUuIu12bPchBzIXiKtLkMJSIYzRz
ySjJmFLm++/ipMQ549xeyviB4pSz44athBuIgENhgE8Xb7eTS5TPuWFjJ913nhdT
IDsa1PMX2PpsinA1IF8Z+kEaAwEAaOCAWcgggFjMB0GA1UdJQQWMBQGCCsGAQUF
BwMBBggrBgEFBQcDAjA1BgNVHREELjAsghNhY2NvdW50cy5nb29nbGUuY29tghUq
-----END CERTIFICATE-----
```

2. Store SSL Certificate on local hard-disk

Copy-and-paste the section `-BEGIN CERTIFICATE-`, `-END CERTIFICATE-` and save it into a `.cert` file

Note: .cert file are plain text files containing the ASCII characters included on the `-BEGIN CERTIFICATE-`, `-END CERTIFICATE-` sections

geonode.cert (or whatever name you want with .cert extension)

```

1 -----BEGIN CERTIFICATE-----
2 MIIEoTCCA4mgAwIBAgIIeEP870cN1RQwDQYJKoZIhvcNAQELBQAwSTELMAkGA1UE
3 BhMCVVMxMzEzARBgNVBAoTCkdvdj2dsZSBjb250aW50aW50aW50aW50aW50aW50
4 cm5ldCBkdj2dsZSBjb250aW50aW50aW50aW50aW50aW50aW50aW50aW50aW50aW50
5 WjBtMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZlc29vZ2x1IEluYzEcMBoGA1UEAwTYWNj
6 TW91bnRhaW4gVmlldzETMBEGA1UECgwKR29vZ2x1IEluYzEcMBoGA1UEAwTYWNj
7 b3VudHMuz29vZ2x1LmNvbTCCASIwDQYJKoZIhvcNAQEBBQAdggEPADCCAQoCggEB
8 AK71x4DYCS7NcMqyp8Uaq7067Wb7LLhkgE5QJcUVdvfLrXmG3PAIcfiLMvdIK+F
9 RODn5d79I18qgn90XgB/xyFgx381MR/Hoglavb1CTfghiotvhAMsxyY/h55efbcn
10 i2GGOXCWVcx119xPONWfvd7aUa0LSutMeE27T4Hqj3o0WFYVJq+dQNGdkGwqtUap
11 bosQFuyWg+KINM6Opvy6ay8fRU/4JeKNVcaZUuIu12bPchBzIXiKtLkMJSIYzRz
12 ySjJmFlm++/ipMQ549xeyviB4pSz44athBuIgeNhgE8Xb7eTS5TPuWFjJ913nhdT
13 IDsa1PMX2PpsinA1IF8Z+kECaWAAaOCaWcwgGfjMBOGA1UdJQQWMBQGCCsGAQUF
14 BwMBBggrBgEFBQcDAjA1BgNVHREELjAsghNhy2NvdW50cy5nb29nbGUuY29tghUq
15 LnBhcnRuZ2x1Y29vZ2x1LmNvbTCCASIwDQYJKoZIhvcNAQEBBQAdggEPADCCAQoCggEB
16 hh9odHRwOi8vc29vZ2x1LmNvbTCCASIwDQYJKoZIhvcNAQEBBQAdggEPADCCAQoCggEB
17 dHRwOi8vc29vZ2x1LmNvbTCCASIwDQYJKoZIhvcNAQEBBQAdggEPADCCAQoCggEB
18 EZz38qRdr+B1Le1iyTAMBgNVHRMBAf8EAjAAMB8GA1UdIwQYMBaAFerdBhYbvPZo
19 tXb1gba7Yhq6WoEvMCEGA1UdIAQaMBgwDAYKKwYBBAHwEQIDATAIBgZngQwBAGIw
20 MAYDVROfBCkwIzA1OCQIVYfaHR0cDovL3BraS5nb29nbGUuY29tghUqI0JicwIwNv

```

3. Add SSL Certificates to the Java Keystore

You can use the Java command `keytool` like this

geonode.cert (or whatever name you want with .cert extension)

```
keytool -import -noprompt -trustcacerts -alias geonode -
↪file geonode.cert -keystore ${KEYSTOREFILE} -storepass $
↪{KEYSTOREPASS}
```

or, alternatively, you can use some graphic tool which helps you managing the SSL Certificates and Keystores, like [Portecle](#)

```
java -jar c:\apps\portecle-1.9\portecle.jar
```

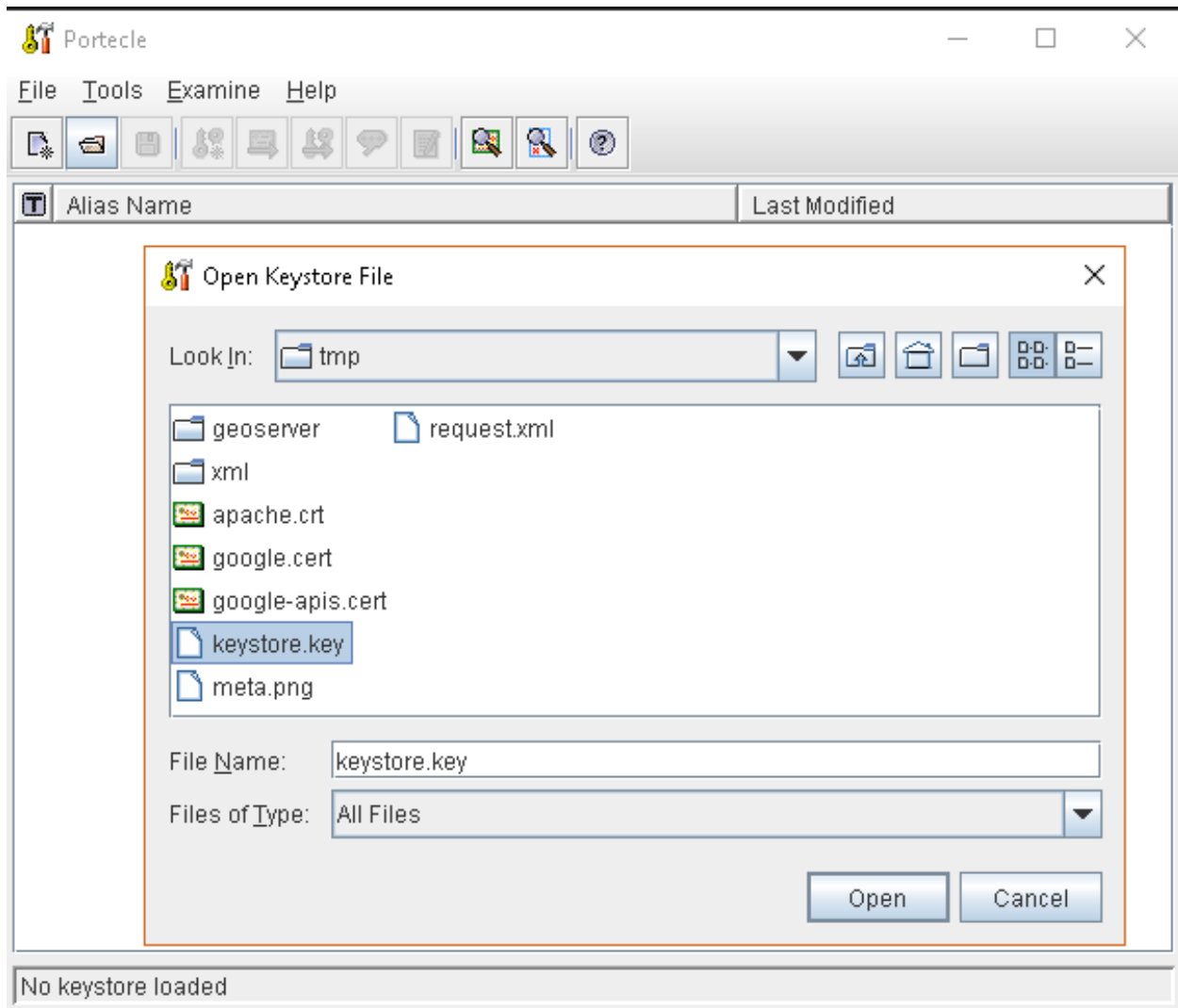
4. Enable the JVM to check for SSL Certificates from the Keystore

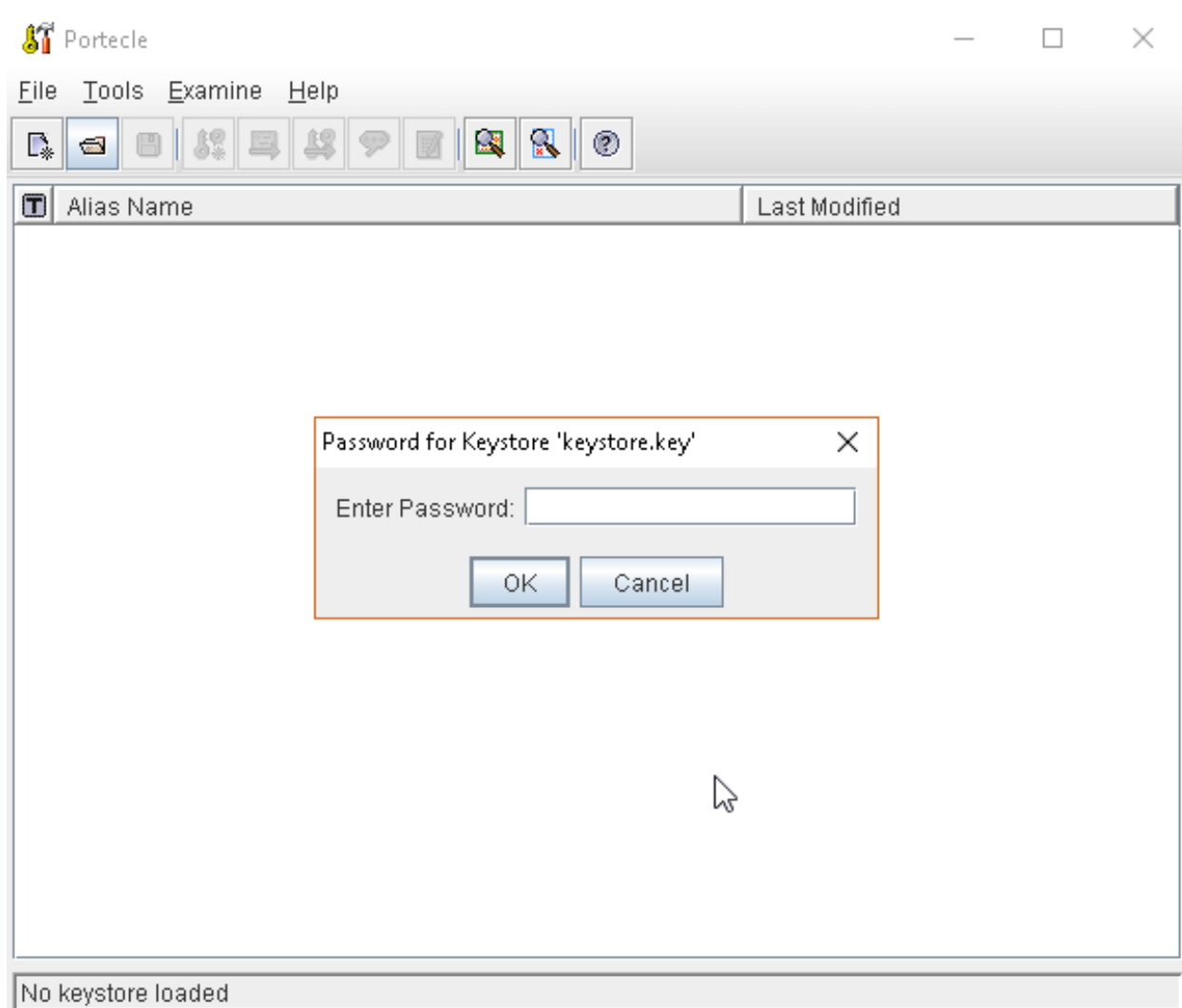
In order to do this, you need to pass a `JAVA_OPTION` to your JVM:

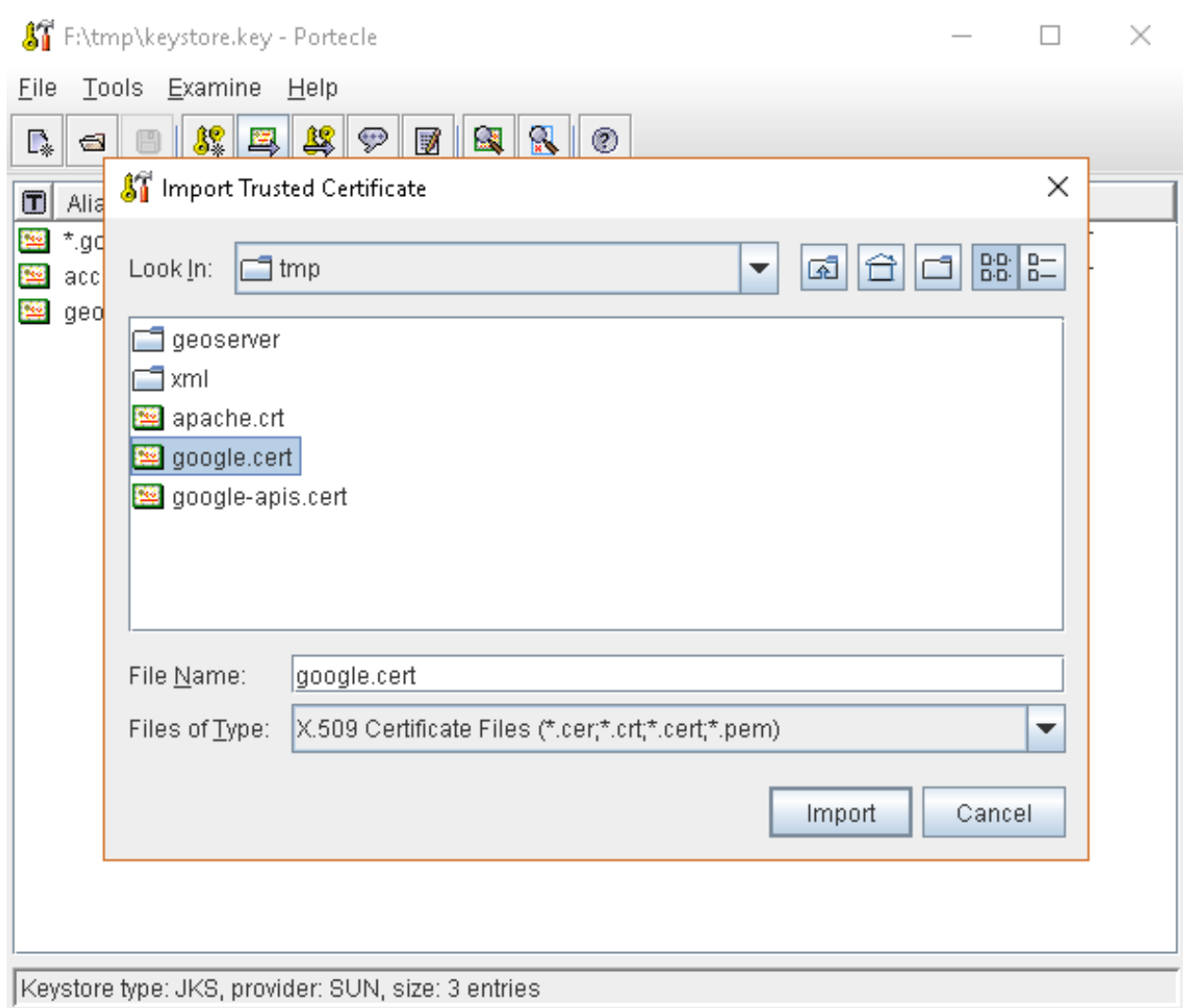
```
-Djavax.net.ssl.trustStore=F:\tmp\keystore.key
```

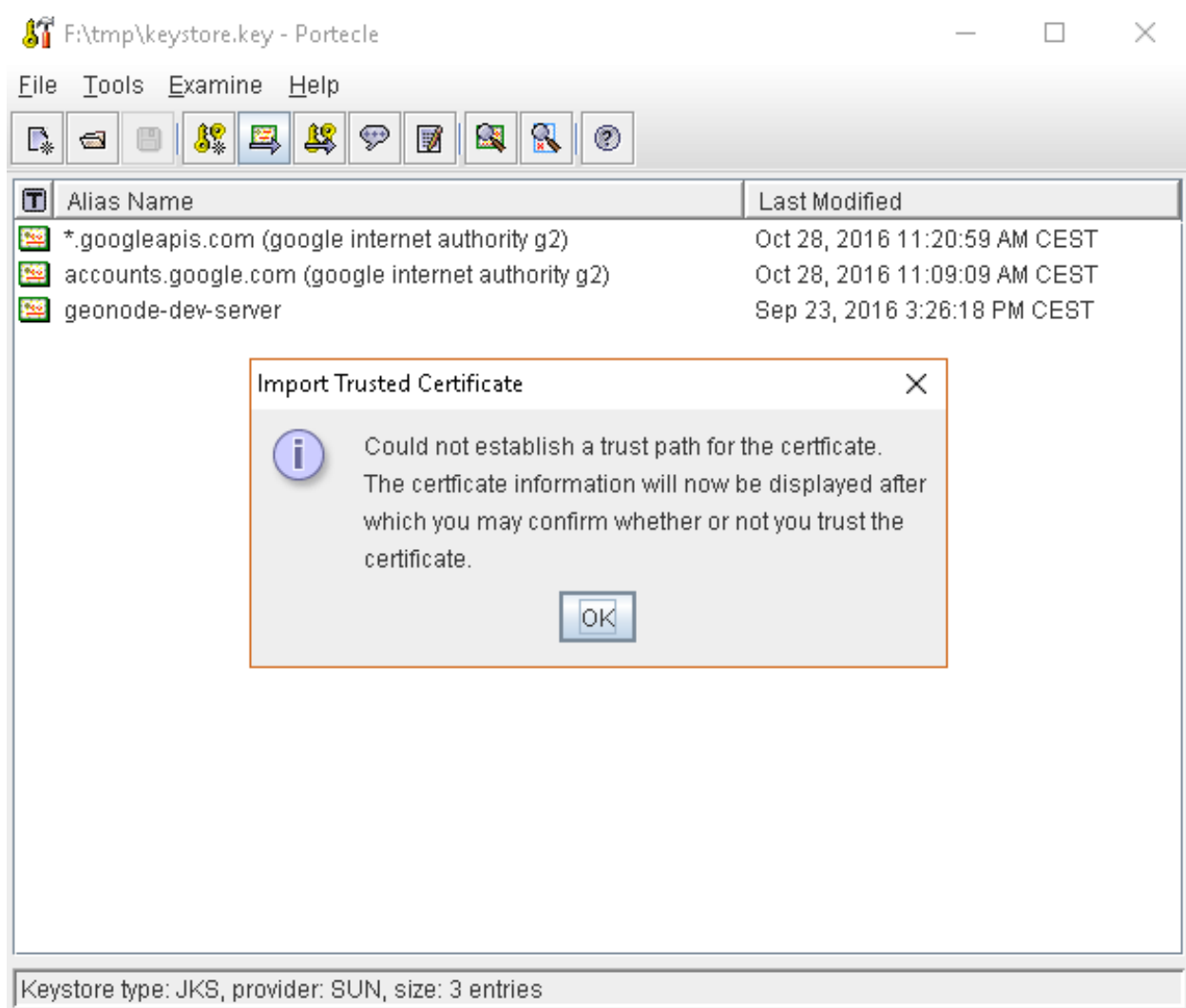
5. Restart your server

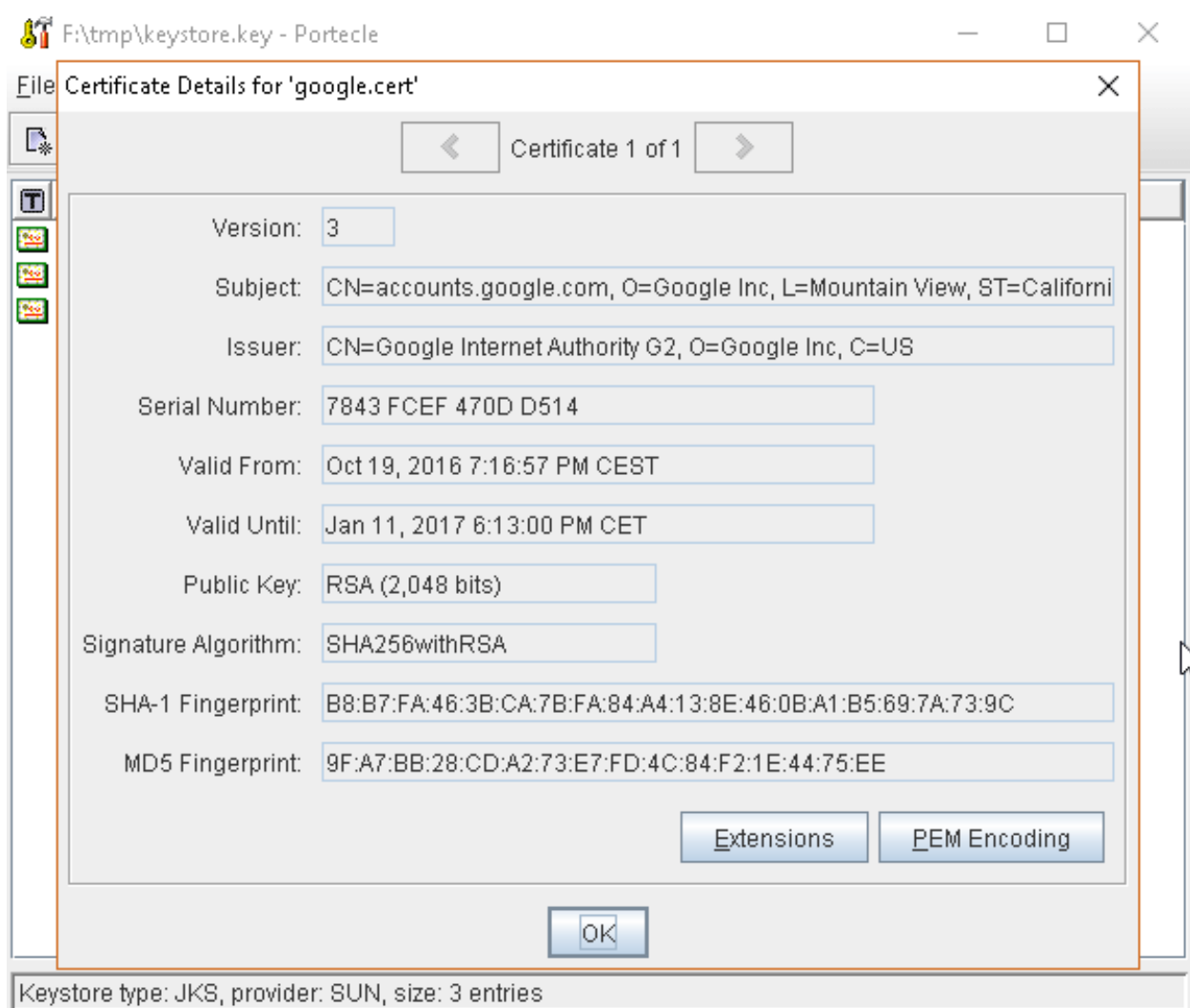
Note: Here below you can find a bash script which simplifies the Keystore SSL Certificates importing. Use it at your convenience.

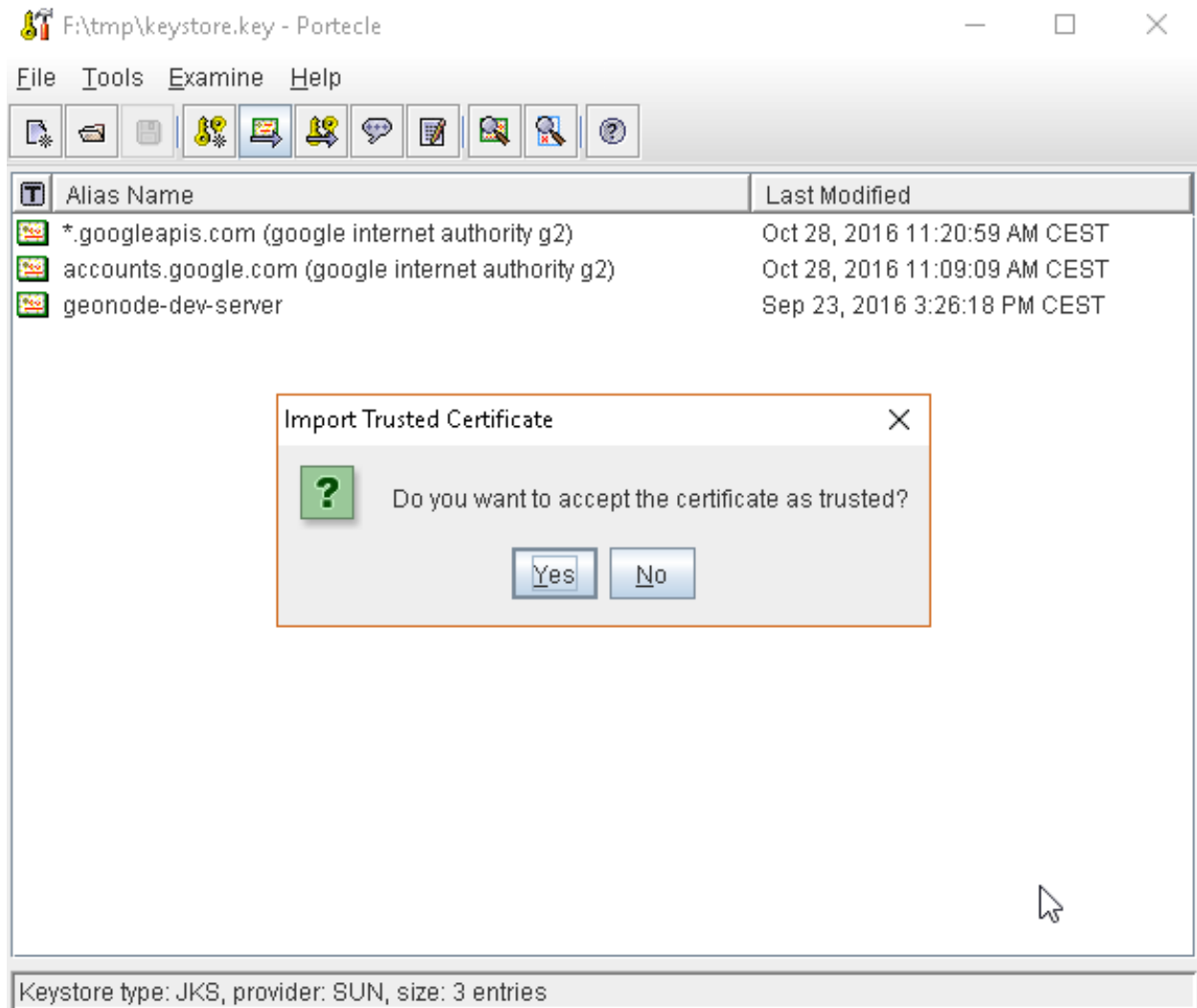


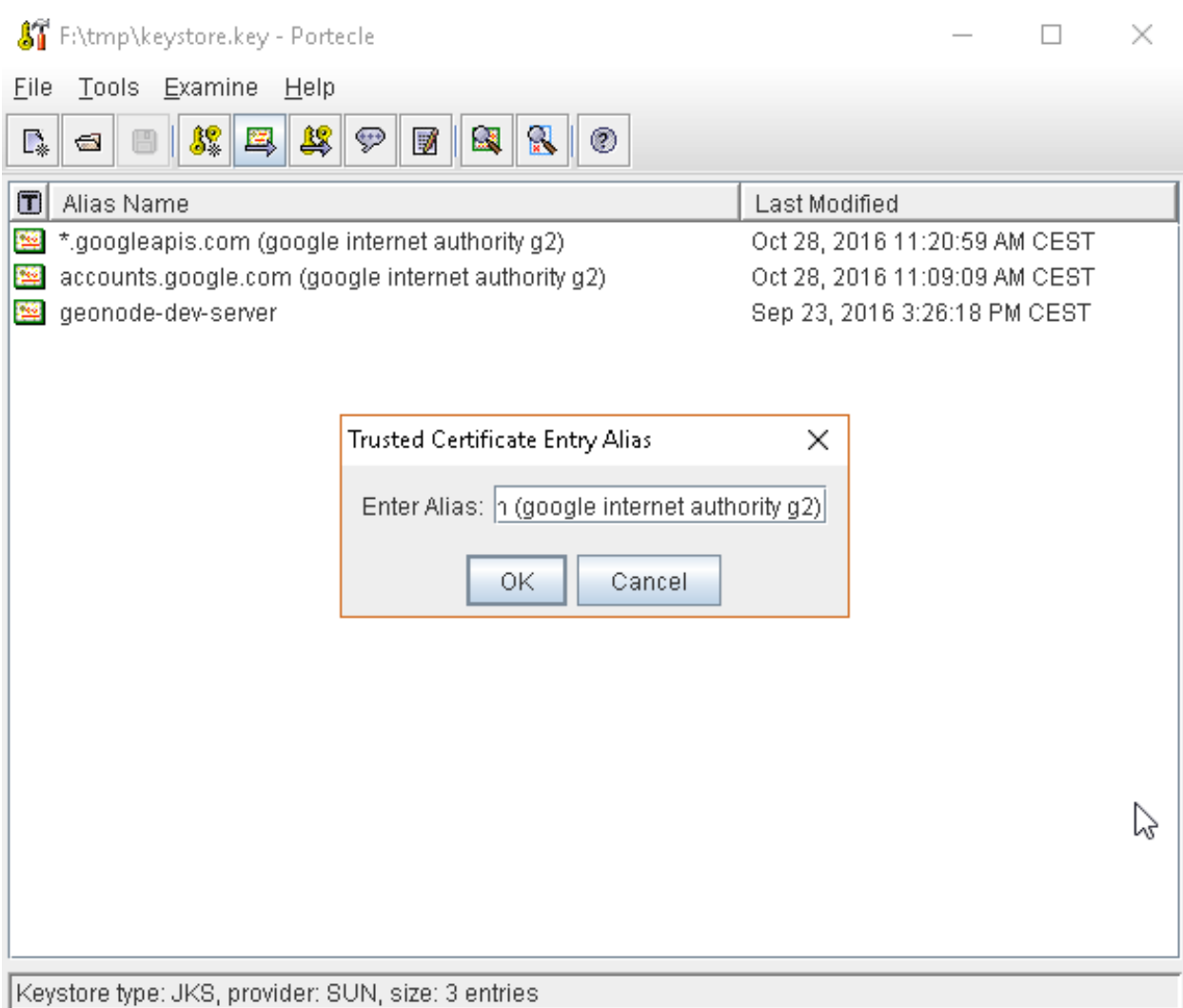


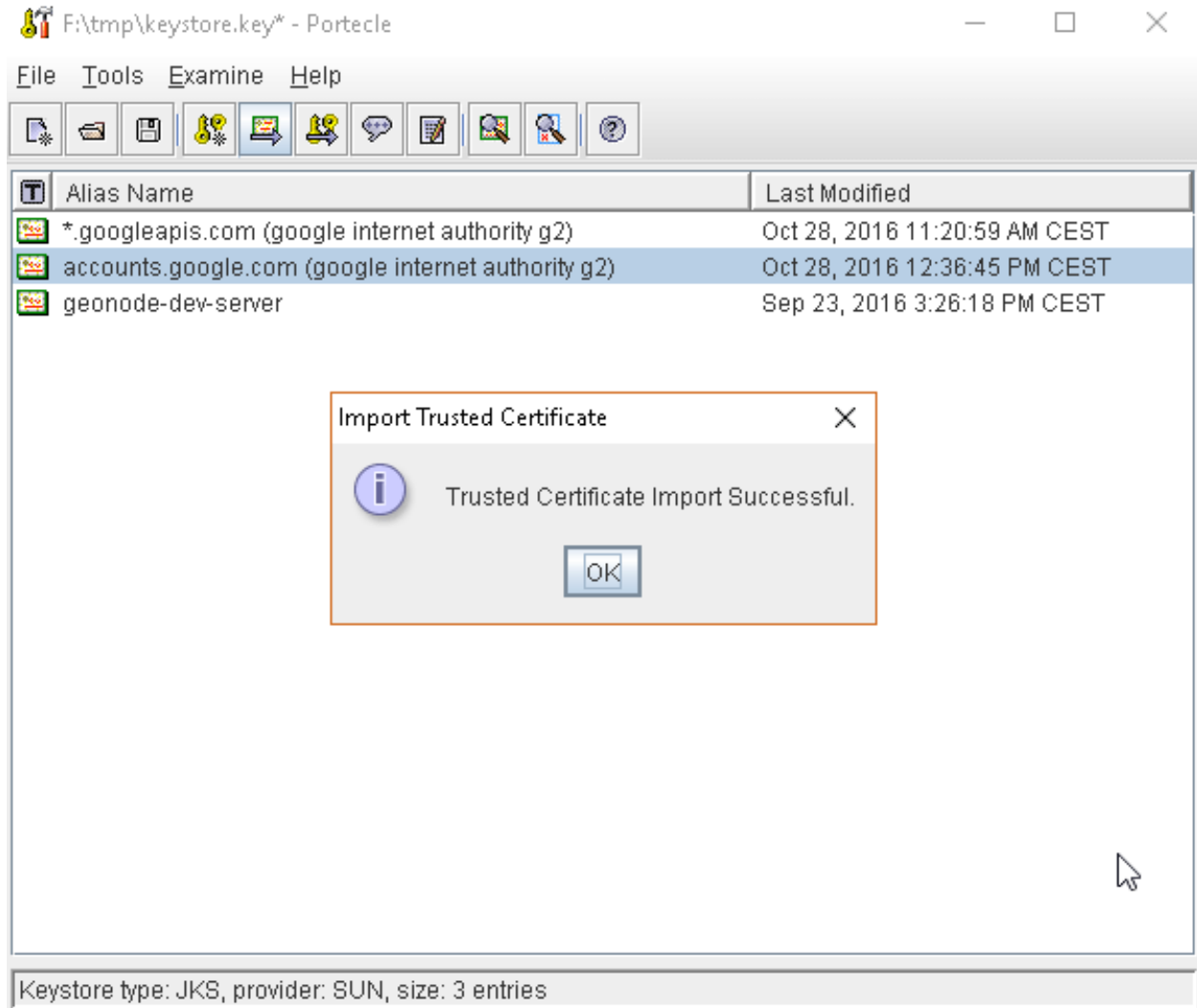


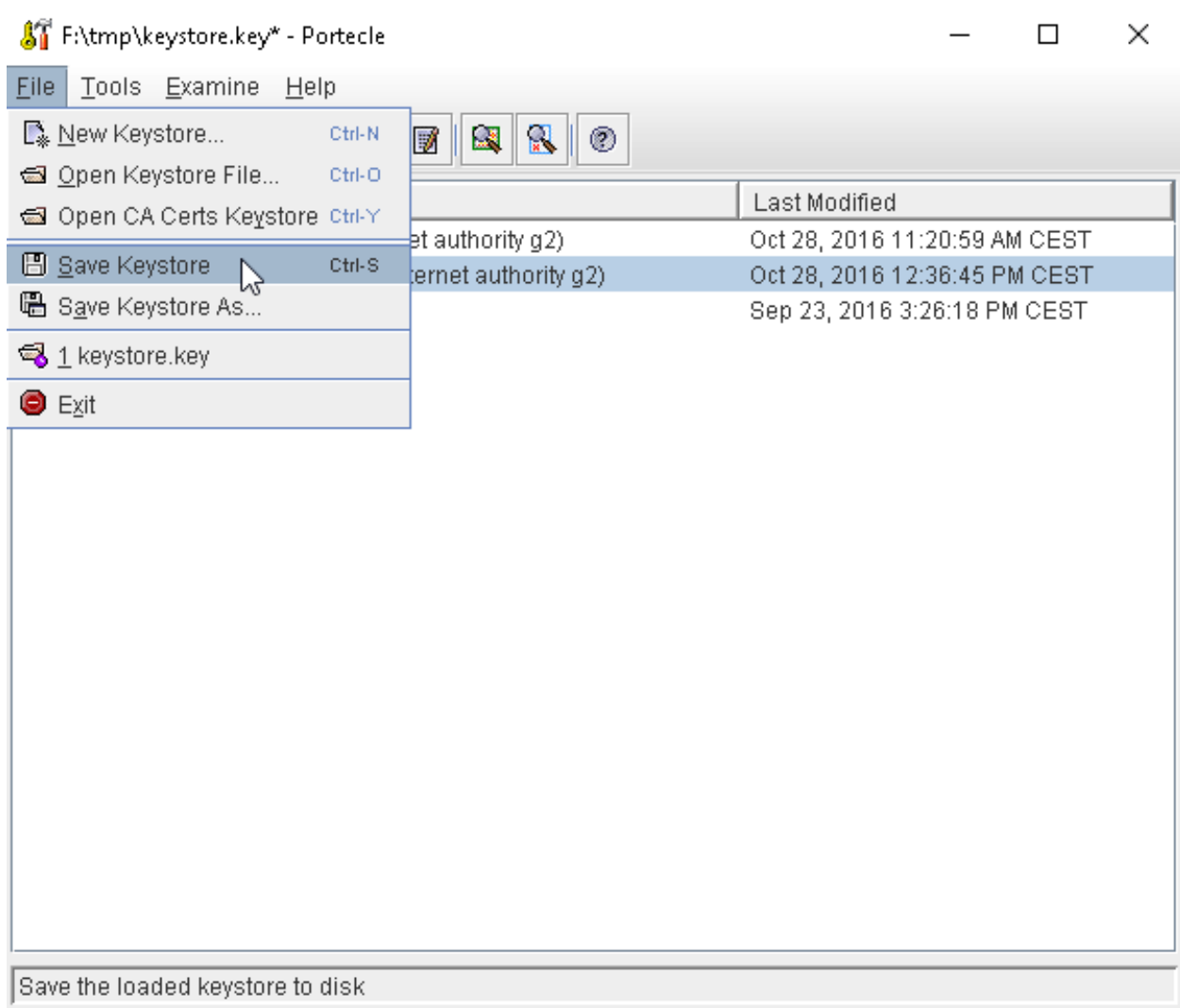












```
HOST=myhost.example.com
PORT=443
KEYSTOREFILE=dest_keystore
KEYSTOREPASS=changeme

# get the SSL certificate
openssl s_client -connect ${HOST}:${PORT} </dev/null \
    | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ${HOST}.cert

# create a keystore and import certificate
keytool -import -noprompt -trustcacerts \
    -alias ${HOST} -file ${HOST}.cert \
    -keystore ${KEYSTOREFILE} -storepass ${KEYSTOREPASS}

# verify we've got it.
keytool -list -v -keystore ${KEYSTOREFILE} -storepass ${KEYSTOREPASS} -alias ${HOST}
```

GeoFence Advanced Features

GeoFence Rules Management and Tutorials

- This [tutorial](#) shows how to install and configure the Geofence Internal Server plug-in. It shows how to create rules in two ways: using the GUI and REST methods.
- GeoFence Rules can be created / updated / deleted through a REST API, accessible only by a GeoServer Admin user. You can find more details on how the GeoFence REST API works [here](#).

GeoFence Rules Storage Configuration

By default GeoFence is configured to use a filesystem based DB stored on the GeoServer Data Dir <GEOSERVER_DATA_DIR/geofence.

- It is possible also to configure GeoFence in order to use an external PostgreSQL / PostGIS Database. For more details please refer to the official GeoFence documentation [here](#).

1. Add Java Libraries to GeoServer

```
wget --no-check-certificate https://build.geo-solutions.it/geonode/
↳ geoserver/latest/hibernate-spatial-postgis-1.1.3.1/hibernate-spatial-
↳ postgis-1.1.3.1.jar
wget --no-check-certificate https://build.geo-solutions.it/geonode/
↳ geoserver/latest/postgis-jdbc-1.3.3/postgis-jdbc-1.3.3.jar

cp hibernate-spatial-postgis-1.1.3.1.jar <GEOSERVER_WEBAPP_DIR>/WEB-INF/
↳ lib
cp postgis-jdbc-1.3.3.jar <GEOSERVER_WEBAPP_DIR>/WEB-INF/lib

restart geoserver
```

2. Either create a DB with the updated schema here https://github.com/geoserver/geofence/blob/master/doc/setup/sql/002_create_schema_postgres.sql or enable the hbm2ddl auto creation through the configuration file (see step 3)

Note: Notice that “update” also creates the tables if they do not exist. In production, however, I would suggest to change it to “validate”

```
# If you want to create a new DB for GeoFence
sudo -u postgres createdb -O geonode geofence; \
sudo -u postgres psql -d geofence -c 'CREATE EXTENSION postgis;'; \
sudo -u postgres psql -d geofence -c 'GRANT ALL ON geometry_columns TO_
↳PUBLIC;'; \
sudo -u postgres psql -d geofence -c 'GRANT ALL ON spatial_ref_sys TO_
↳PUBLIC;'; \
sudo -u postgres psql -d geofence -c 'GRANT ALL PRIVILEGES ON ALL TABLES_
↳IN SCHEMA public TO geonode;'
```

3. Add configuration similar to geofence-datasource-ovr.properties sample below (if loaded as GeoServer extension)

<GEOSERVER_DATA_DIR>/geofence/geofence-datasource-ovr.properties

```
# /* (c) 2019 Open Source Geospatial Foundation - all rights reserved
# * This code is licensed under the GPL 2.0 license, available at the_
↳root
# * application directory.
# */
#
geofenceVendorAdapter.databasePlatform=org.hibernate.spatial.postgis.
↳PostgisDialect
geofenceDataSource.driverClassName=org.postgresql.Driver
geofenceDataSource.url=jdbc:postgresql://localhost:5432/geofence
geofenceDataSource.username=postgres
geofenceDataSource.password=postgres
geofenceEntityManagerFactory.jpaPropertyMap[hibernate.default_
↳schema]=public

#####
↳#####
## Other setup entries
#####
↳#####
## hbm2ddl.auto may assume one of these values:
## - validate: validates the DB schema at startup against the internal_
↳model. May fail on oracle spatial.
## - update: updates the schema, according to the internal model._
↳Updating automatically the production DB is dangerous.
## - create-drop: drop the existing schema and recreates it according to_
↳the internal model. REALLY DANGEROUS, YOU WILL LOSE YOUR DATA.
## You may want not to redefine the property entirely, in order to leave_
↳the default value (no action).

geofenceEntityManagerFactory.jpaPropertyMap[hibernate.hbm2ddl.
↳auto]=update
geofenceEntityManagerFactory.jpaPropertyMap[javax.persistence.validation.
↳mode]=none
geofenceEntityManagerFactory.jpaPropertyMap[hibernate.validator.apply_to_
↳ddl]=false
geofenceEntityManagerFactory.jpaPropertyMap[hibernate.validator.
↳autoregister_listeners]=false
```

(continues on next page)

(continued from previous page)

```

##
## ShowSQL is set to true in the configuration file; putting
↳ showsql=false in
## this file, you can easily check that this override file has been
↳ properly applied.

# geofenceVendorAdapter.generateDdl=false
# geofenceVendorAdapter.showSql=false

## Set to "true" in specific use cases
# workspaceConfigOpts.showDefaultGroups=false

#####
↳ #####
## Disable second level cache.
## This is needed in a geofence-clustered environment.

#geofenceEntityManagerFactory.jpaPropertyMap[hibernate.cache.use_second_
↳ level_cache]=false

#####
↳ #####
## Use external ehcache configuration file.
## Useful to change cache settings, for example diskStore path.
#geofenceEntityManagerFactory.jpaPropertyMap[hibernate.cache.provider_
↳ configuration_file_resource_path]=file:/path/to/geofence-ehcache-
↳ override.xml

```

1.22 Hardening GeoNode

1.22.1 Publish on HTTPS

TBD

1.22.2 OAuth2 Fixtures Update and Base URL Migration

TBD

1.22.3 GeoNode Security Subsystem

TBD

1.22.4 OAuth2 Tokens and Sessions

TBD (ref to *OAuth2 Access Tokens*)

1.23 Social Login

1.23.1 Overview

1.23.2 Configuration

1.23.3 LinkedIn

1.23.4 Facebook

1.24 GeoNode Django Contrib Apps

1.24.1 Geonode auth via LDAP

This package provides utilities for using LDAP as an authentication and authorization backend for geonode.

The `django_auth_ldap` package is a very capable way to add LDAP integration with django projects. It provides a lot of flexibility in mapping LDAP users to geonode users and is able to manage user authentication.

However, in order to provide full support for mapping LDAP groups with geonode's and enforce group permissions on resources, a custom geonode authentication backend is required. This contrib package provides such a backend, based on `django_auth_ldap`.

Installation

Installing this contrib package is a matter of:

1. Installing geonode
2. Installing system LDAP libraries (development packages needed)
3. Cloning this repository locally
4. Change to the `ldap` directory and install this contrib package

```
# 1. install geonode (not shown here for brevity)
# 2. install systemwide LDAP libraries
sudo apt install \
    libldap2-dev \
    libsasl2-dev

# 3. get geonode/contribs code
git clone https://github.com/GeoNode/geonode-contribs.git

# 4. install geonode ldap contrib package
cd geonode-contribs/ldap
pip install .
```

Configuration

1. Add `geonode_ldap.backend.GeonodeLdapBackend` as an additional auth backend.

```
# e.g. by updating your settings.py or local_settings.py
AUTHENTICATION_BACKENDS += (
    "geonode_ldap.backend.GeonodeLdapBackend",
)
```

You may use additional auth backends, the django authentication framework tries them all according to the order listed in the settings. This means that geonode can be setup in such a way as to permit internal organization users to login with their LDAP credentials, while at the same time allowing for casual users to use their facebook login (as long as you enable facebook social auth provider).

Note: The django's `django.contrib.auth.backends.ModelBackend` must also be used in order to provide full geonode integration with LDAP. However this is included by default on GeoNode settings

```
# The GeoNode default settings are the following
AUTHENTICATION_BACKENDS = (
    'oauth2_provider.backends.OAuth2Backend',
    'django.contrib.auth.backends.ModelBackend',
    'guardian.backends.ObjectPermissionBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
)
```

2. Set some additional configuration values. Some of these variables are prefixed with `AUTH_LDAP` (these are used directly by `django_auth_ldap`) while others are prefixed with `GEONODE_LDAP` (these are used by `geonode_ldap`). The geonode custom variables are:
 - `GEONODE_LDAP_GROUP_PROFILE_FILTERSTR` - This is an LDAP search fragment with the filter that allows querying for existing groups. See example below
 - `GEONODE_LDAP_GROUP_NAME_ATTRIBUTE` - This is the name of the LDAP attribute that will be used for deriving the geonode group name. If not specified it will default to `cn`, which means that the LDAP object's *common name* will be used for generating the name of the geonode group
 - `GEONODE_LDAP_GROUP_PROFILE_MEMBER_ATTR` - This is the name of the LDAP attribute that will be used for deriving the geonode membership. If not specified it will default to `member`

Example configuration:

```
# add these import lines to the top of your geonode settings file
from django_auth_ldap import config as ldap_config
from geonode_ldap.config import GeonodeNestedGroupOfNamesType
import ldap

# add both standard ModelBackend auth and geonode.contrib.ldap auth
AUTHENTICATION_BACKENDS += (
    'geonode_ldap.backend.GeonodeLdapBackend',
)

# django_auth_ldap configuration
AUTH_LDAP_SERVER_URI = env("LDAP_SERVER_URI")
AUTH_LDAP_BIND_DN = env("LDAP_BIND_DN")
AUTH_LDAP_BIND_PASSWORD = env("LDAP_BIND_PASSWORD")
AUTH_LDAP_USER_SEARCH = ldap_config.LDAPSearch(
```

(continues on next page)

(continued from previous page)

```

env("LDAP_USER_SEARCH_DN"),
ldap.SCOPE_SUBTREE,
env("LDAP_USER_SEARCH_FILTERSTR")
)
AUTH_LDAP_GROUP_SEARCH = ldap_config.LDAPSearch(
env("LDAP_GROUP_SEARCH_DN"),
ldap.SCOPE_SUBTREE,
env("LDAP_GROUP_SEARCH_FILTERSTR")
)
AUTH_LDAP_GROUP_TYPE = GeonodeNestedGroupOfNamesType()
AUTH_LDAP_USER_ATTR_MAP = {
    "first_name": "givenName",
    "last_name": "sn",
    "email": "mailPrimaryAddress"
}
AUTH_LDAP_FIND_GROUP_PERMS = True
AUTH_LDAP_MIRROR_GROUPS_EXCEPT = [
    "test_group"
]

# these are not needed by django_auth_ldap - we use them to find and match
# GroupProfiles and GroupCategories
GEONODE_LDAP_GROUP_NAME_ATTRIBUTE = env("LDAP_GROUP_NAME_ATTRIBUTE", default="cn")
GEONODE_LDAP_GROUP_PROFILE_FILTERSTR = env("LDAP_GROUP_SEARCH_FILTERSTR", default=
↳ '(ou=research group)')
GEONODE_LDAP_GROUP_PROFILE_MEMBER_ATTR = env("LDAP_GROUP_PROFILE_MEMBER_ATTR",
↳ default='member')

```

Example environment variables:

```

LDAP_SERVER_URL=ldap://<the_ldap_server>
LDAP_BIND_DN=uid=ldapinfo,cn=users,dc=ad,dc=example,dc=org
LDAP_BIND_PASSWORD=<something_secret>
LDAP_USER_SEARCH_DN=dc=ad,dc=example,dc=org
LDAP_USER_SEARCH_FILTERSTR=( &(uid=%(user)s)(objectClass=person))
LDAP_GROUP_SEARCH_DN=cn=groups,dc=ad,dc=example,dc=org
LDAP_GROUP_SEARCH_FILTERSTR=(|(cn=abt1)(cn=abt2)(cn=abt3)(cn=abt4)(cn=abt5)(cn=abt6))
LDAP_GROUP_PROFILE_MEMBER_ATTR=uniqueMember

```

The configuration seen in the example above will allow LDAP users to login to geonode with their LDAP credentials.

On first login, a geonode user is created from the LDAP user and its LDAP attributes `cn` and `sn` are used to populate the geonode user's `first_name` and `last_name` profile fields.

Any groups that the user is a member of in LDAP (under the `cn=groups,dc=ad,dc=example,dc=org` search base and belonging to one of `(|(cn=abt1)(cn=abt2)(cn=abt3)(cn=abt4)(cn=abt5)(cn=abt6))` groups) will be mapped to the corresponding geonode groups, even creating these groups in geonode in case they do not exist yet. The geonode user is also made a member of these geonode groups.

Upon each login, the user's geonode group memberships are re-evaluated according to the information extracted from LDAP. The `AUTH_LDAP_MIRROR_GROUPS_EXCEPT` setting can be used to specify groups whose memberships will not be re-evaluated.

Note: Users mapped from LDAP will be marked with an `ldap` tag. This will be used to keep them in sync.

Warning: If you remove the `ldap` tag, the users will be threaten as pure internal GeoNode ones.

You may also manually generate the geonode groups in advance, before users login. In this case, when a user logs in and the mapped LDAP group already exists, the user is merely added to the geonode group

Be sure to check out `django_auth_ldap` for more information on the various configuration options.

Keep Users and Groups Synchronized

In order to constantly keep the remote LDAP Users and Groups **synchronized** with GeoNode, you will need to run periodically some specific management commands.

```
* /10 * * * * /opt/geonode/my-geonode/manage.sh updateldapgroups >> /var/log/cron.log
↪ 2>&1
* /10 * * * * /opt/geonode/my-geonode/manage.sh updateldapusers >> /var/log/cron.log
↪ 2>&1
```

Where the `manage.sh` is a bash script similar to the following one:

manage.sh

```
export $(grep -v '^#' /opt/geonode/my-geonode/.env | xargs -d '\n'); /home/<my_user>/.
↪ virtualenvs/geonode/bin/python /opt/geonode/my-geonode/manage.py $@
```

and the `/opt/geonode/my-geonode/.env` is something similar to the following one:

/opt/geonode/my-geonode/.env

```
DEBUG=False
DJANGO_ALLOWED_HOSTS=<geonode_public_host>,localhost,127.0.0.1
DJANGO_DATABASE_URL=postgres://my_geonode:*****@localhost:5432/my_geonode_db
DEFAULT_BACKEND_UPLOADER=geonode.importer
DEFAULT_FROM_EMAIL=geonode@example.org
DJANGO_EMAIL_HOST=smtp.example.org
DJANGO_EMAIL_HOST_PASSWORD=*****
DJANGO_EMAIL_HOST_USER=geonode
DJANGO_EMAIL_PORT=465
DJANGO_EMAIL_USE_SSL=True
DJANGO_SETTINGS_MODULE=my_geonode.settings
DJANGO_SECRET_KEY=*****
OAUTH2_API_KEY=*****
PROXY_URL=/proxy/?url=
EXIF_ENABLED=True
EMAIL_ENABLE=True
TIME_ENABLED=True
ACCOUNT_OPEN_SIGNUP=True
ACCOUNT_APPROVAL_REQUIRED=True
ACCOUNT_EMAIL_REQUIRED=True
ACCOUNT_EMAIL_VERIFICATION=optional
AVATAR_GRAVATAR_SSL=True
GEONODE_DB_URL=postgres://my_geonode:*****@localhost:5432/my_geonode_data
GEOSERVER_ADMIN_PASSWORD=*****
GEOSERVER_LOCATION=https://<geonode_public_host>/geoserver/
GEOSERVER_PUBLIC_HOST=<geonode_public_host>
GEOSERVER_PUBLIC_LOCATION=https://<geonode_public_host>/geoserver/
GEOSERVER_WEB_UI_LOCATION=https://<geonode_public_host>/geoserver/
```

(continues on next page)

(continued from previous page)

```

LDAP_SERVER_URL=ldap://<the_ldap_server>
LDAP_BIND_DN=uid=ldapinfo,cn=users,dc=ad,dc=example,dc=org
LDAP_BIND_PASSWORD=<something_secret>
LDAP_USER_SEARCH_DN=dc=ad,dc=example,dc=org
LDAP_USER_SEARCH_FILTERSTR=(&(uid=%(user)s)(objectClass=person))
LDAP_GROUP_SEARCH_DN=cn=groups,dc=ad,dc=example,dc=org
LDAP_GROUP_SEARCH_FILTERSTR=(|(cn=abt1)(cn=abt2)(cn=abt3)(cn=abt4)(cn=abt5)(cn=abt6))
LDAP_GROUP_PROFILE_MEMBER_ATTR=uniqueMember
OGC_REQUEST_MAX_RETRIES=3
OGC_REQUEST_POOL_CONNECTIONS=100
OGC_REQUEST_POOL_MAXSIZE=100
OGC_REQUEST_TIMEOUT=60
SITEURL=https://<geonode_public_host>/
SITE_HOST_NAME=<geonode_public_host>
FREETEXT_KEYWORDS_READONLY=False
# Advanced Workflow Settings
ADMIN_MODERATE_UPLOADS=False
GROUP_MANDATORY_RESOURCES=False
GROUP_PRIVATE_RESOURCES=False
RESOURCE_PUBLISHING=False

```

Note: You might want to use the same `/opt/geonode/my-geonode/.env` for your UWSGI configuration too:

```

[uwsgi]
socket = 0.0.0.0:8000
uid = <my_user>
gid = www-data

plugins = python3
virtualenv = /home/<my_user>/virtualenvs/geonode

# set environment variables from .env file
env LANG=en_US.utf8
env LC_ALL=en_US.UTF-8
env LC_LANG=en_US.UTF-8

for-readline = /opt/geonode/my-geonode/.env
    env = %(_)
endfor =

chdir = /opt/geonode/my-geonode
module = my_geonode.wsgi:application

processes = 12
threads = 2
enable-threads = true
master = true

# logging
# path to where uwsgi logs will be saved
logto = /storage/my_geonode/logs/geonode.log
daemonize = /storage/my_geonode/logs/geonode.log
touch-reload = /opt/geonode/my-geonode/my_geonode/wsgi.py
buffer-size = 32768
max-requests = 500

```

(continues on next page)

(continued from previous page)

```
harakiri = 300 # respawn processes taking more than 5 minutes (300 seconds)
# limit-as = 1024 # avoid Errno 12 cannot allocate memory
harakiri-verbose = true
vacuum = true
thunder-lock = true
```

1.24.2 Geonode Logstash for centralized monitoring/analytics

This contrib app, along with the GeoNode internal monitoring app, lets administrators to configure a service for sending metrics data to a **centralized server** which comes with [Logstash](#).

So it will be possible to visualize stats and charts about one or more GeoNode instances outside the application. Having a server configured with the [ELK stack](#), it is possible to visualize those information on a Kibana dashboard for example.

If you manage more than one GeoNode instances, that server can receive data from many GeoNode(s) so it can make available both *single-instance dashboards* (referred to individual instances) and *global dashboards* (stats calculated on the whole set of instances).

Warning: The centralized monitoring service cannot be active if the settings variables `USER_ANALYTICS_ENABLED` and `monitoring-enabled` are set to `False`.

Overview

By default, GeoNode will send data to the centralized server every **3600 seconds** (1 hour) so, if enabled, the monitoring app will collect 1-hour-aggregated data. This time interval can be configured, see the next paragraphs to know how.

Formatted and compressed data will be sent on a **TCP** connection (on the `443` standard port by default) through a **scheduled celery task** which basically logs information via [python-logstash-async](#).

Warning: This feature requires [python-logstash-async](#).

Data and events formats

Each time the centralized monitoring service is called, 4 types of *JSON* formatted events are sent to the server:

1. Instance overview

```
{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  "time": {
    "startTime": UTC now - 1 hour (default)
    "endTime": UTC now
  },
}
```

(continues on next page)

(continued from previous page)

```

"hits": total number of requests,
"unique_visits": total number of unique sessions,
"unique_visitors": total number of unique users,
"registered_users": total number of registered users at the end time,
"layers": total number of layers at the end time,
"documents": total number of documents at the end time,
"maps": total number of maps at the end time,
"errors": total number of errors
}

```

2. Resources details

```

{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  "time": {
    "startTime": UTC now - 1 hour (default)
    "endTime": UTC now
  },
  "resources": [
    ...
    {
      "type": resource type,
      "name": resource name,
      "url": resource URL,
      "hits": total number of requests about this resource,
      "unique_visits": total number of unique sessions about this resource,
      "unique_visitors": total number of unique users about this resource,
      "downloads": total number of resource downloads,
      "ogcHits": total number of OGC service requests about this resource,
      "publications": total number of publication events
    },
    ...
  ]
}

```

3. Countries details

```

{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  "time": {
    "startTime": UTC now - 1 hour (default)
    "endTime": UTC now
  },
  "countries": [
    ...
    {
      "name": country name,
      "hits": total number of requests about the country
    },
    ...
  ]
}

```

(continues on next page)

(continued from previous page)

```
    ...
  ]
}
```

4. UA (User Agent) Family details

```
{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  "time": {
    "startTime": UTC now - 1 day
    "endTime": UTC now
  },
  "ua_families": [
    ...
    {
      "name": UA family name
      "hits": total number of requests about the UA family
    },
    ...
  ]
}
```

These messages will be [gzip](#) compressed in order to improve transport performances and they should be parsed by a [logstash filter](#) on the server side (see [Logstash configuration](#)).

Configuration


The centralized monitoring service is disabled by default because it needs the internal monitoring to be active and service-specific configurations.

GeoNode configuration

On the GeoNode side, all needed configurations can be set up from the Django admin interface.
If enabled, the **GEONODE LOGSTASH** section will show the **Centralized servers** feature:

GEONODE_LOGSTASH

Centralized servers

 [Change](#)

Let's add one:

Django administration
WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · Geonode_Logstash · Centralized servers · CentralizedServer object

Change centralized server
HISTORY

Host: 192.168.1.91
Centralized Server IP address.

Port: 5000
Centralized Server TCP port number.

Interval: 3600
Data aggregation time interval (in seconds).

Db path: logstash.db
The local SQLite database to cache log events between emitting and transmission to the Logstash server. This way log events are cached even across process restarts (and crashes).

Socket timeout: 5
Timeout in seconds for TCP connections.

Queue check interval: 2
Interval in seconds to check the internal queue for new messages to be cached in the database.

Queue events flush interval: 0.1
Interval in seconds to send cached events from the database to Logstash.

Queue events flush count: 50
Count of cached events to send from the database to Logstash; events are sent to Logstash whenever QUEUED_EVENTS_FLUSH_COUNT or QUEUED_EVENTS_FLUSH_INTERVAL is reached, whatever happens first.

Queue events batch size: 50
Maximum number of events to be sent to Logstash in one batch. Depending on the transport, this usually means a new connection to the Logstash is established for the event batch.

Logstash db timeout: 5
Timeout in seconds to 'connect' the SQLite database.

Last successful deliver: Oct 17, 2019, 2:39 p.m.
Timestamp of the last successful deliver.

Next scheduled deliver: Oct 17, 2019, 3:39 p.m.
Timestamp of the next scheduled deliver.

Last failed deliver: -
Timestamp of the last failed deliver.

Delete Save and continue editing SAVE

Test connection

The **Host** IP address and the **Port** number are mandatory as well as the time **Interval** (3600 seconds by default) which defines the service invocation polling (so the time range on which data should be aggregated).

Note: Once the service configured, the user can test the configuration by clicking on **Test connection**. It will test the connection with the centralized server without saving the configuration.

Other settings come with a default value:

- **Db path** → the local SQLite database to cache events between emitting and transmission to the Logstash server (log events are cached even across process restarts and crashes);
- **Socket timeout** → timeout in seconds for TCP connections;
- **Queue check interval** → interval in seconds to check the internal queue for new messages to be cached in the database;
- **Queue events flush interval** → interval in seconds to send cached events from the database to Logstash;
- **Queue events flush count** → count of cached events to send from the database to Logstash;

- **Queue events batch size** -> maximum number of events to be sent to Logstash in one batch;
- **Logstash db timeout** -> timeout in seconds to 'connect' the SQLite database.

To better understand what these variables mean, it is recommended to read the [python-logstash-async options for the asynchronous processing and formatting](#).

Other three read-only fields will be visible:

- **Last successful deliver** -> timestamp of the last successful deliver (if exists);
- **Next scheduled deliver** -> timestamp of the next scheduled deliver;
- **Last failed deliver** -> timestamp of the last failed deliver (if exists).

Logstash configuration

On the server side, a proper Logstash configuration should be set up.

Some events formats contain arrays (see [Data and events formats](#)) so Logstash should be able to retrieve a single event for each element of the array. The [Split filter plugin](#) helps to correctly parse those messages.

As mentioned above, events messages will be gzip compressed so the [Gzip_lines codec plugin](#) should be installed along with Logstash and the "gzip_lines" codec should be used for the *tcp* input.

An example of the logstash configuration:

```
input {
  tcp {
    port => <logstash_port_number>
    codec => "gzip_lines"
  }
}

filter {
  json {
    source => "message"
  }
  if [format_version] == "1.0" {
    if [countries] {
      split {
        field => "countries"
      }
    }
    if [resources] {
      split {
        field => "resources"
      }
    }
    if [ua_families] {
      split {
        field => "ua_families"
      }
    }
    mutate {
      remove_field => "message"
    }
  }
}

geoip {
```

(continues on next page)

(continued from previous page)

```

    source => "[instance][ip]"
  }
}

output {
  elasticsearch {
    hosts => "elasticsearch:<elastic_port_number>"
    index => "logstash-%{[instance][name]}-%{+YYYY.MM.dd}"
    user => "elastic"
    password => "changeme"
  }
  stdout { codec => rubydebug }
}

```

Usage

When saving the service configuration, if monitoring enabled, GeoNode will create/update a celery [Periodic Task](#) which will be executed at regular intervals based on the *interval* configured.

You can check this behavior on the *Periodic Tasks* section of the admin UI:

PERIODIC TASKS		
Crontabs	+ Add	✎ Change
Intervals	+ Add	✎ Change
Periodic tasks	+ Add	✎ Change
Solar events	+ Add	✎ Change

The *dispatch-metrics-task* task:

Django administration
WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home · Periodic Tasks · Periodic tasks

Select periodic task to change
ADD PERIODIC TASK +

Action: Go 0 of 3 selected

<input type="checkbox"/>	PERIODIC TASK	ENABLED
<input type="checkbox"/>	dispatch-metrics-task: every 3600 seconds	✓
<input type="checkbox"/>	delayed-security-sync-task: every 60 seconds	✓
<input type="checkbox"/>	celery.backend_cleanup: 0 4 * * * (m/h/d/dM/MY)	✓

3 periodic tasks

The task details:

Django administration
WELCOME, ADMIN . [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home › Periodic Tasks › Periodic tasks › dispatch-metrics-task: every 3600 seconds

Change periodic task
HISTORY

Name:



Useful description



Task (registered):

Task (custom):



☒ Enabled

Schedule

Interval:  

Crontab:  

Use one of interval/crontab

Solar:  

Use a solar schedule

Arguments (Show)

Execution Options (Show)

Delete

Save and add another

Save and continue editing

SAVE

Warning: When disabling monitoring is a **good practice** to disable the corresponding Periodic Task too.

Management command

In addition to the scheduled task, this contrib app makes also available the **dispatch_metrics** command to manually send metrics to the server.

Obviously the time interval considered will start at the last successful delivery and will finish at the current time.

When the monitoring plugin is enabled (*USER_ANALYTICS_ENABLED* and *monitoring-enabled* are set to *True*) and a *Geonode Logstash for centralized monitoring/analytics* configured, Geonode sends (hourly by default) metrics data to an external server (which comes with Logstash) for stats visualization and analysis.

The command can be launched using the `manage.py` script. No options are required.

```
$ DJANGO_SETTINGS_MODULE=<your_settings_module> python manage.py dispatch_metrics
```

Possible exceptions raised during the execution will be reported to GeoNode log.

1.25 GeoNode Admins Guide

GeoNode has an administration panel, based on the Django admin, which can be used to do some database operations. Although most of the operations can and should be done through the normal GeoNode interface, the admin panel provides a quick overview and management tool over the database.

The following sections will explain more in depth what functionalities the admin panel makes you available. It should be highlighted that the sections not covered in this guide are meant to be managed through GeoNode UI.

1.25.1 Accessing the panel

The *Admin Panel* is a model-centric interface where trusted users can manage content on GeoNode. Only the staff users can access the admin interface.

Note: The “staff” flag, which controls whether the user is allowed to log in to the admin interface, can be set by the admin panel itself.

The panel can be reached from *Admin* link of the *User Menu* in the navigation bar (see the picture below) or through this URL: `http://<your_geonode_host>/admin`.

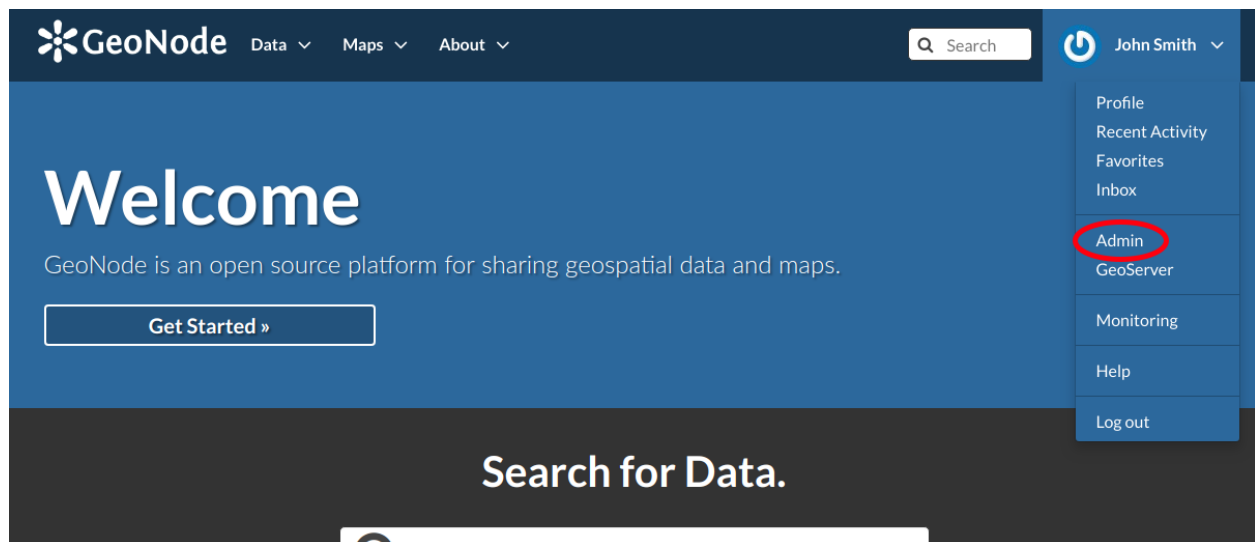


Fig. 243: The Admin Link of the User Menu

When clicking on that link the Django-based *Admin Interface* page opens and shows you all the Django models registered in GeoNode.

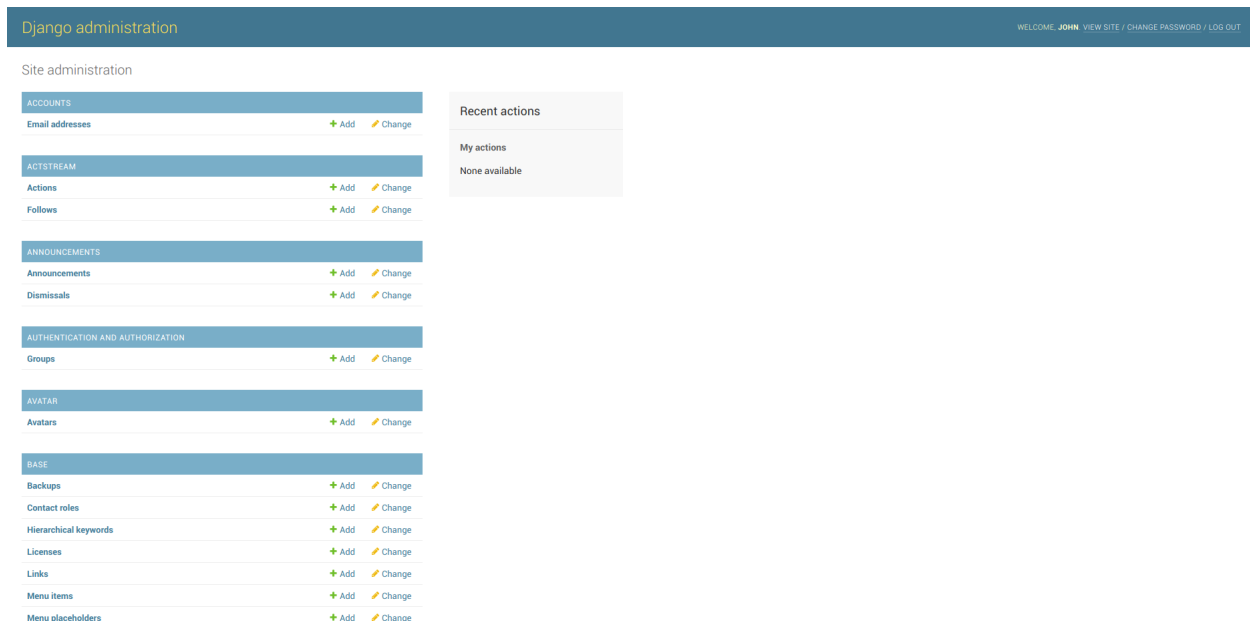


Fig. 244: The GeoNode Admin Interface

1.25.2 Reset or Change the admin password

From the *Admin Interface* you can access the *CHANGE PASSWORD* link on the right side of the navigation bar.

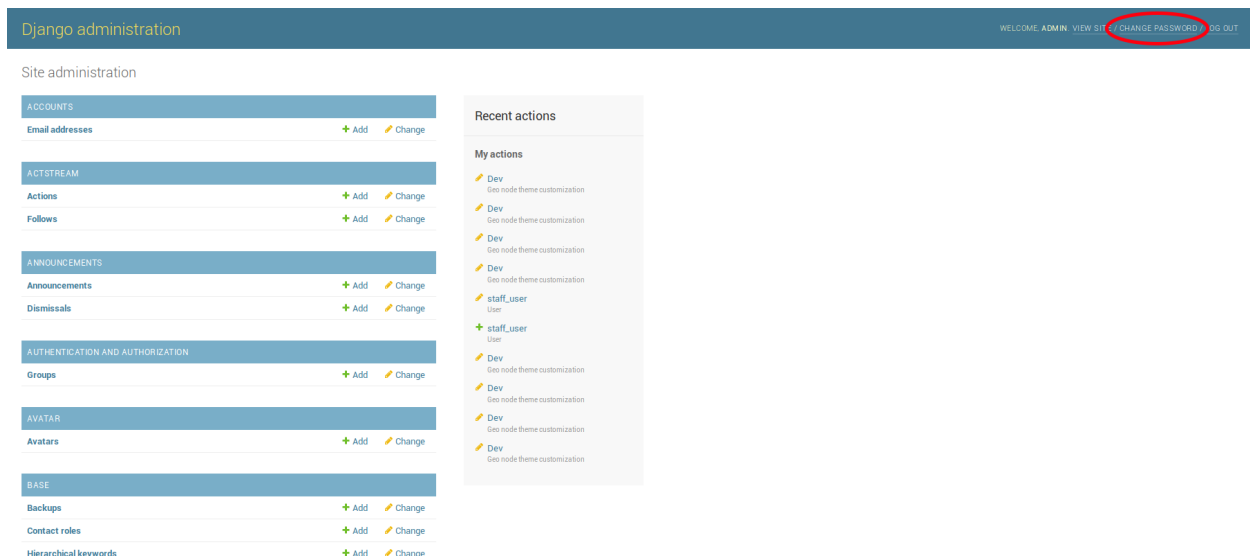


Fig. 245: The Change Password Link

It allows you to access the *Change Password Form* through which you can change your password.

Once the fields have been filled out, click on *CHANGE MY PASSWORD* to perform the change.

Django administration

Home / Password change

WELCOME ADMIN / CHANGE PASSWORD / LOG OUT

Password change

Please enter your old password, for security's sake, and then enter your new password twice so we can verify you typed it in correctly.

Old password:

New password:

New password confirmation:

CHANGE MY PASSWORD

Fig. 246: *The Change Password Form*

1.25.3 Simple Theming

GeoNode provides by default some theming options manageable directly from the Administration panel. Most of the times those options allows you to easily change the GeoNode look and feel without touching a single line of *HTML* or *CSS*.

As an *administrator* go to `http://<your_geonode_host>/admin/geonode_themes/geonodethemecustomization/`.

Django administration

Home / GeoNode Themes Library / Themes

WELCOME ADMIN / VIEW SITE / CHANGE PASSWORD / LOG OUT

Select geo node theme customization to change

ADD GEO NODE THEME CUSTOMIZATION +

Action: 0 of 1 selected

ID	IS ENABLED	NAME	DATE	DESCRIPTION
2	<input checked="" type="checkbox"/>	Dev	May 3, 2019, 6:01 p.m.	

1 geo node theme customization

Fig. 247: *List of available Themes*

The panel shows all the available GeoNode themes, if any, and allows you to create new ones.

Warning: Only one theme at a time can be **activated** (aka *enabled*). By disabling or deleting all the available themes, GeoNode will turn the gui back to the default one.

Editing or creating a new Theme, will actually allow you to customize several properties.

At least you'll need to provide a Name for the Theme. Optionally you can specify also a Description, which will allow you to better identify the type of Theme you created.

Just below the Description field, you will find the Enabled checkbox, allowing you to toggle the Theme.

Change geo node theme customization

Name:
This will not appear anywhere.

Description:

This will not appear anywhere.

Fig. 248: Theme Name and Description

Description:

This will not appear anywhere.

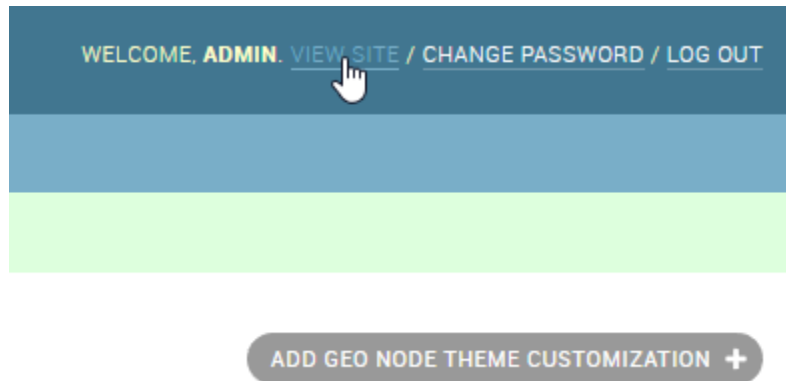
☒ **Is enabled**
Enabling this theme will disable the current enabled theme (if any).



Fig. 249: Theme Name and Description

Jumbotron and Get Started link

Note: Remember, everytime you want to apply some changes to the Theme, you **must** save the Theme and reload the GeoNode browser tab. In order to quickly switch back to the Home page, you can just click the `VIEW SITE` link on the top-right corner of the Admin dashboard.



The next section, allows you to define the first important Theme properties. This part involves the GeoNode main page sections.

By changing those properties as shown above, you will easily change your default home page from this to this

It is possible to optionally **hide** the `Jumbotron` text and/or the `Call to action` button

Copyright and contact info footer

The default GeoNode footer does not present any type of contact info.

By enabling and editing the `contact us` box fields

it will be possible to show a simple *Contact Us* info box on the GeoNode footer section.

Similarly, by editing the `Copyright` text box and/or background color

it will be possible to show the Copyright statement to the bottom of the page

Partners

GeoNode simple theming, allows also a `Partners` section, in order to easily list links to third-party institutions collaborating to the project.

The example below shows the `Partners` section of [WorldBank CHIANG MAI URBAN FLOODING](#) GeoNode instance made through integrating theming options.

The `Partners` items can be managed through the `http://<your_geonode_host>/admin/geonode_themes/partner/` Admin section

From here it is possible to add, modify or delete partners items.

A new partner is defined by few elements, a `Logo`, a `Name`, a `Display Name` and a `Website`


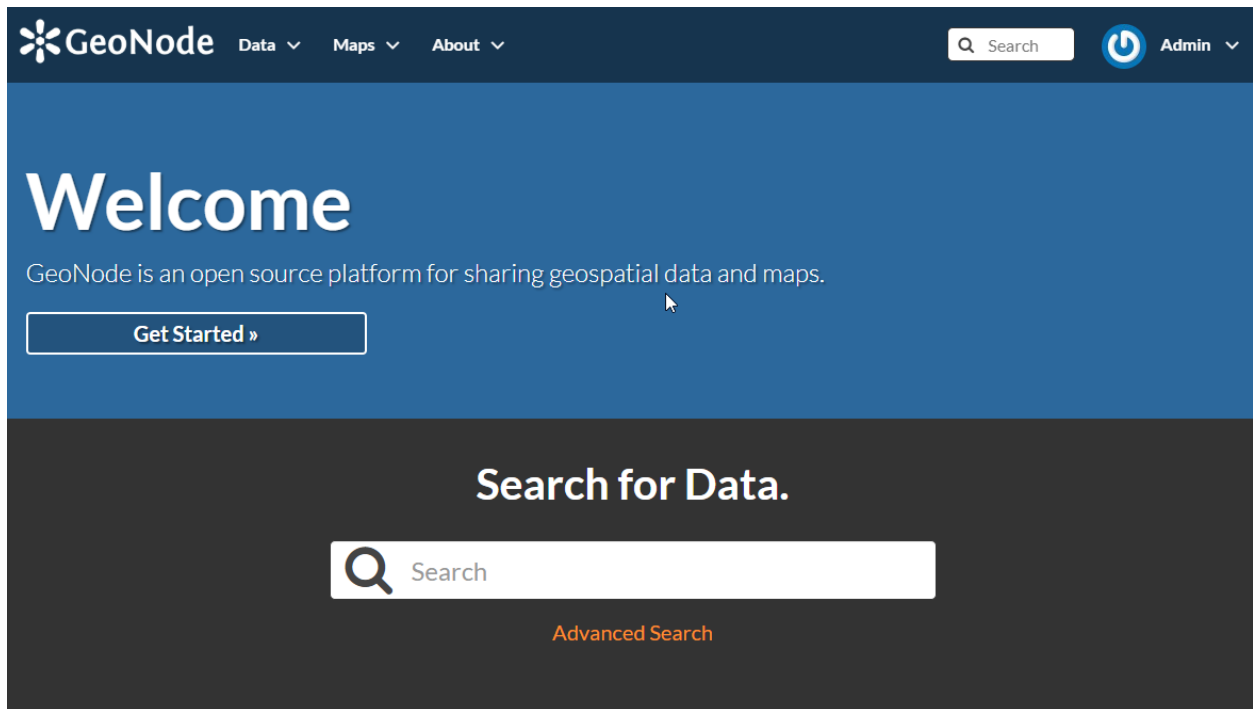
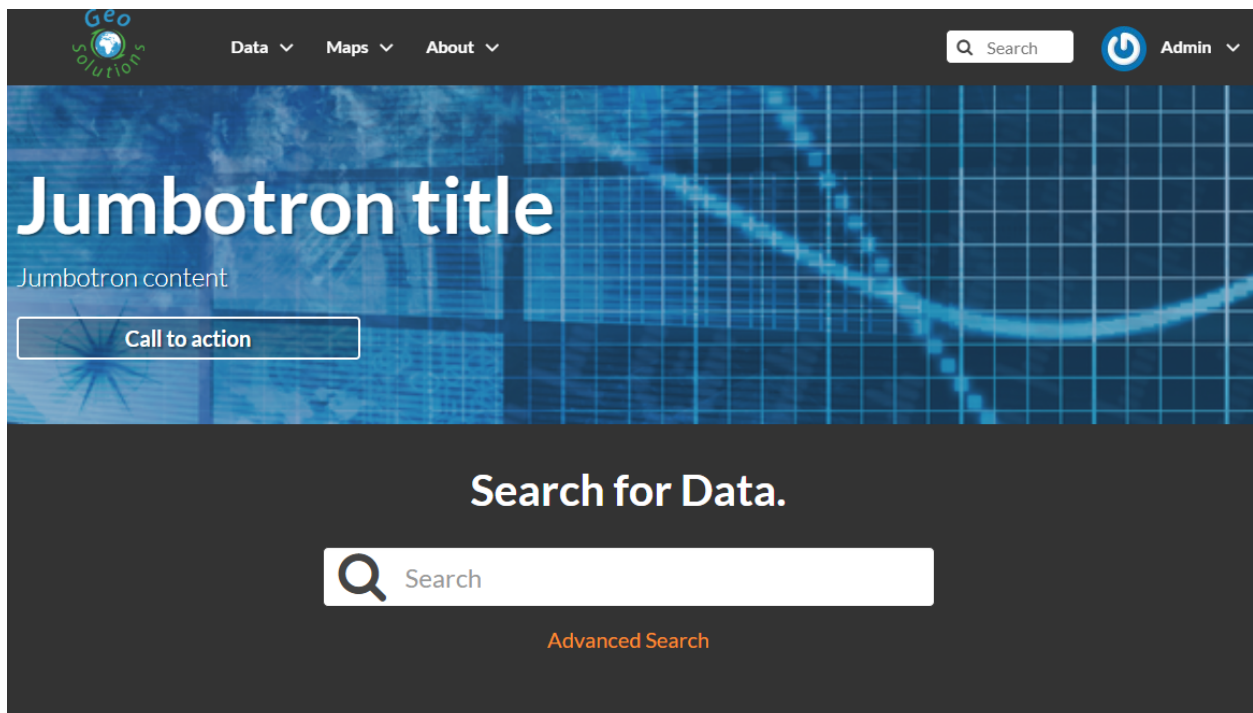
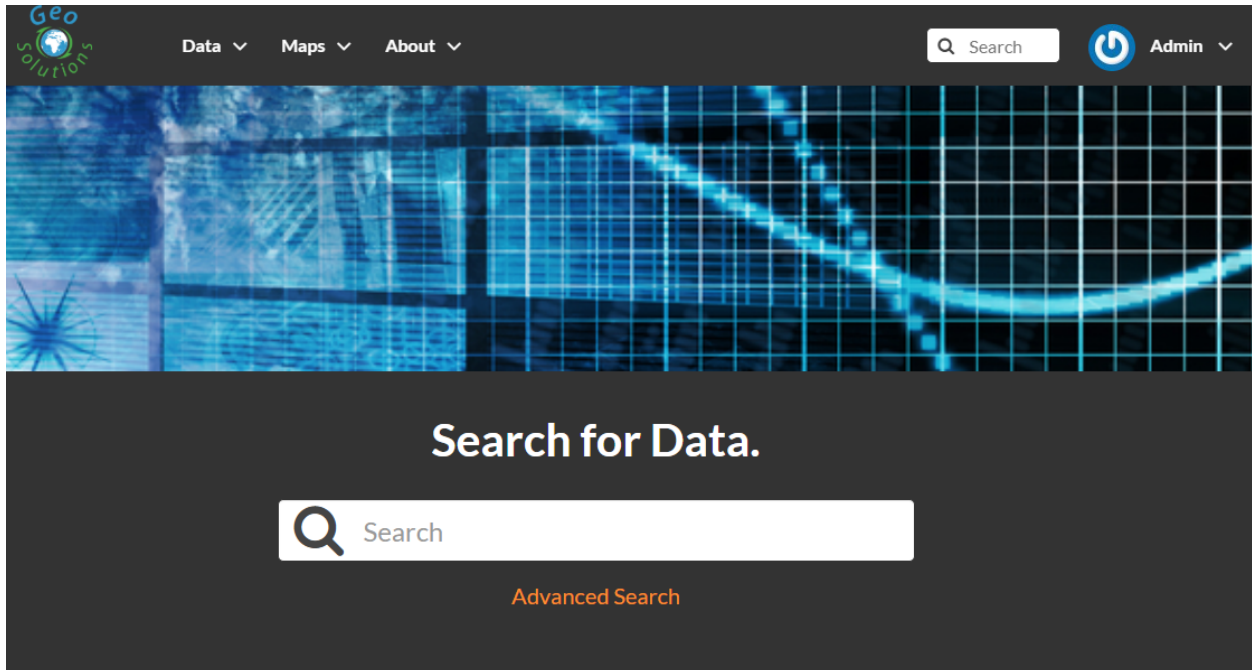
Logo:	<input type="button" value="Choose File"/> logo_GeoSol...uadrato.png
Jumbotron background:	<input type="button" value="Choose File"/> geospatial.png
<input type="checkbox"/> Hide text in the jumbotron Check this if the jumbotron background image already contains text	
Jumbotron title:	<input type="text" value="Jumbotron title"/>
Jumbotron content:	<div><div>Jumbotron content</div><div></div></div>
<input type="checkbox"/> Hide call to action	
Call to action text:	<input type="text" value="Call to action"/>
Call to action link:	<input type="text" value="www.call_to_action_link"/>

Fig. 250: *Jumbotron and Logo options*

Fig. 251: *GeoNode Default Home*Fig. 252: *Updating Jumbotron and Logo*

☒ Hide text in the jumbotron
Check this if the jumbotron background image already contains text

☒ Hide call to action



The screenshot shows the GeoNode homepage. At the top is a dark navigation bar with the GeoNode logo, 'Data', 'Maps', and 'About' menus, a search bar, and an 'Admin' button. Below this is a large jumbotron with a blue grid background. The jumbotron contains the text 'Search for Data.' in white, a large search bar with a magnifying glass icon, and a link for 'Advanced Search' in orange. The jumbotron is currently visible, as indicated by the checked checkbox in the form above.

Fig. 253: Hide Jumbotron text and Call to action button

Data

- Layers
- Documents
- Remote Services
- Upload Layer
- Create Layer
- Upload Document
- Add Remote Service

Maps

- Explore Maps
- Create Map

About

- People
- Groups
- Announcements
- Invite Users
- Add User
- Create Group

Powered by GeoNode version 2.10rc5

[Developers](#) | [About](#)

English

Fig. 254: Default GeoNode Footer


<input checked="" type="checkbox"/> Enable contact us box	
Contact name:	Mr. Pibody
Contact position:	Chief Executive Officer
Contact administrative area:	Paperopoli
Contact street:	113th street
Contact postal code:	113117
Contact city:	Topolinia
Contact country:	Disney
Contact delivery point:	
Contact voice:	+00 000.000.113
Contact facsimile:	+00 000.000.117
Contact email:	pibody@paperopoli.dis

Fig. 255: Enable contact us box

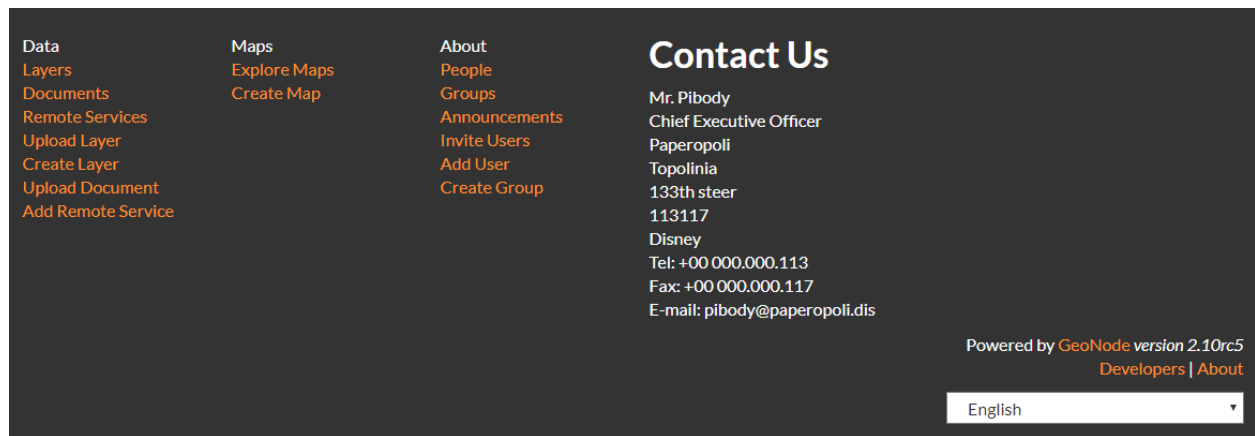


Fig. 256: *Contact Us Footer*

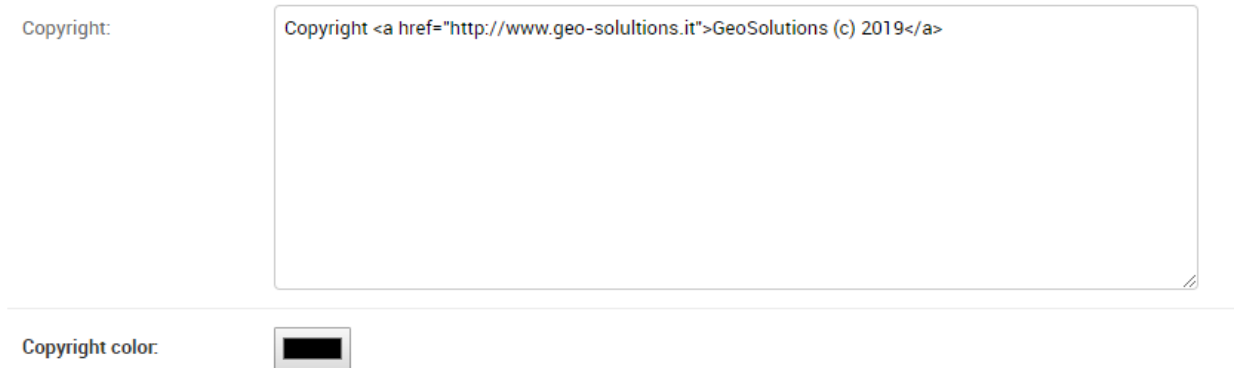


Fig. 257: *Copyright Text and Color*



Fig. 258: *Copyright*

With support from:



[Data](#)
[Layers](#)
[Documents](#)
[Remote Services](#)

[Maps](#)
[Explore Maps](#)

[About](#)
[People](#)
[Groups](#)
[Invite Users](#)

Powered by [GeoNode version 2.10rc5](#)
[Developers](#) | [About](#)
 English

GEOSOLUTIONS S.A.S. 2019 (C)

Fig. 259: Urban-flooding GeoNode Partners Section

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home : [GeoNode Themes Library](#) : [Partners](#)

Select partner to change

Action: Go 0 of 8 selected

ID	NAME	DISPLAY NAME	WEBSITE
8	co-risk	co-risk	http://co-risk.org/
1	Disaster Risk Financing & Insurance Program	Disaster Risk Financing & Insurance Program	http://www.worldbank.org/en/programs/disaster-risk-financing-and-insurance-program
5	EOS	EOS	https://earthobservatory.sg/
3	GFDRR	GFDRR	https://www.gfdr.org/en
2	Humanitarian OpenStreetMap Team	Humanitarian OpenStreetMap Team	https://www.hotosm.org/what-we-do
7	NATCAP	NATCAP	http://www.naturalcapitalproject.org/
6	NTU	NTU	https://ase.ntu.edu.sg/
4	WORLD BANK GROUP	WORLD BANK GROUP	https://www.worldbank.org/

8 Partners

Fig. 260: GeoNode Partners Admin Section

Add partner

Logo:	<input type="button" value="Choose File"/> No file chosen
Name:	<input type="text"/>
	<small>This will not appear anywhere.</small>
Display name:	<input type="text"/>
Website:	<input type="text"/>

Fig. 261: *Add a Partner*

In order to attach or detach a Partner to an existing Theme on GeoNode, you will need to edit the Theme and go to the Partners section

From here you will be able to either to change the Partners title text and/or select/deselect Partners from the multi-select box.

Note: In order to select/deselect elements from the multi-select box, you **must** use the CTRL+CLICK button combination.

Privacy Policies and Cookie settings

By enabling the Cookies Law Info Bar checkbox (True by default)

it will be possible to allow GeoNode presenting the *Privacy Policies and Cookie settings* pop-ups and links at the bottom of the home page

There are plenty of options available, allowing you to customize contact info as long as colors of the bar and page.

One of the most important to consider it is for sure the Cookie law info bar text

The default text contained in this section is the following one

This website uses cookies to improve your experience,
check **this page
→ for details.
We'll assume you're ok with this, but you can opt-out if you wish.**

The text can be changed and customized, of course. Nevertheless it points by default to the following page

/privacy_cookies/

aka `http://<your_geonode_host>/privacy_cookies/`

Contact email:

Partners title: With support from:

Partners:

co-risk
Disaster Risk Financing & Insurance Program
EOS
GFDRR
Humanitarian OpenStreetMap Team
NATCAP
NTU
WORLD BANK GROUP


+

Hold down "Control", or "Command" on a Mac, to select more than one.

Copyright: `GeoSolutions S`

Fig. 262: Theme Partners Section

Copyright color:

☒ Cookies Law Info Bar 

Cookie law info bar head:

This website uses cookies

Cookie law info bar text:

This website uses cookies to improve your experience, check `this page` for details. We'll assume you're ok with this, but you can opt out if you wish.

Fig. 263: Cookies Law Info Bar checkbox

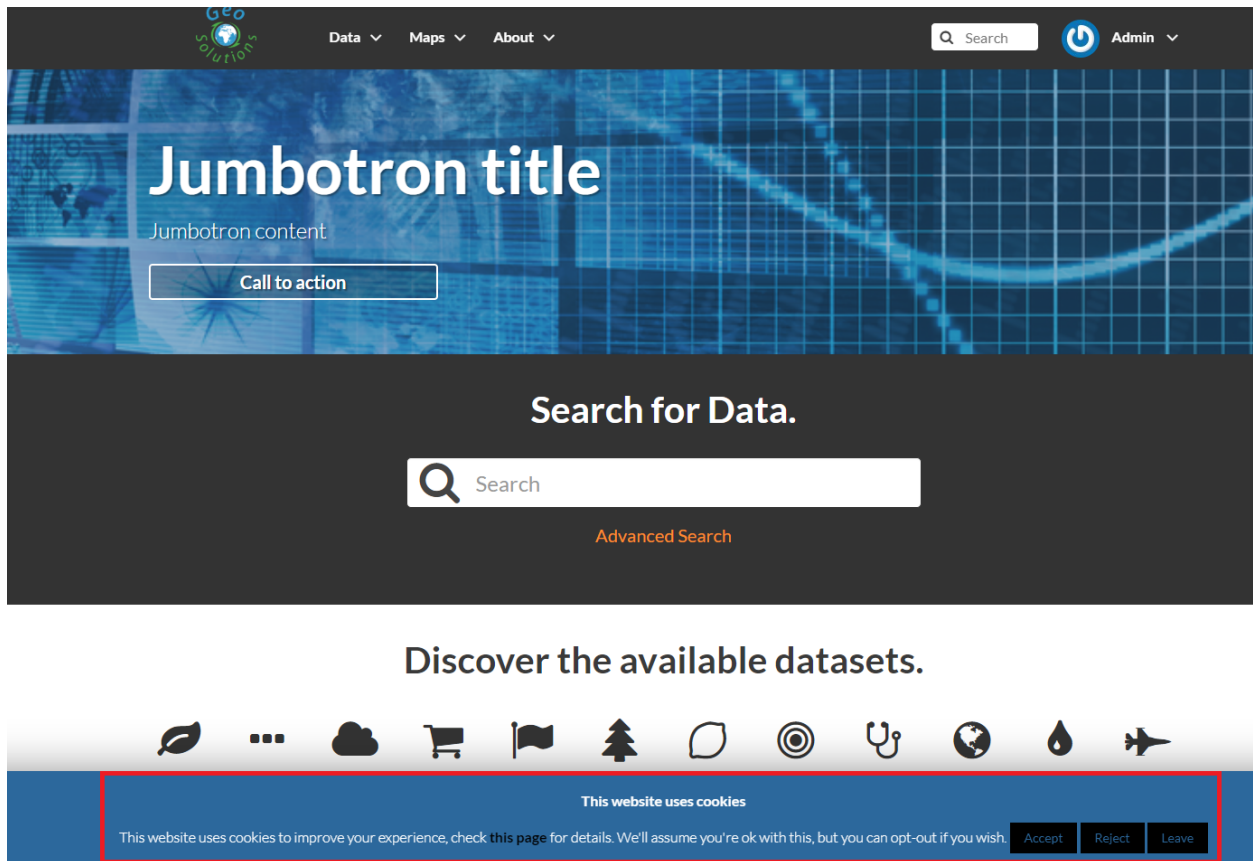


Fig. 264: Cookies Law Info Bar

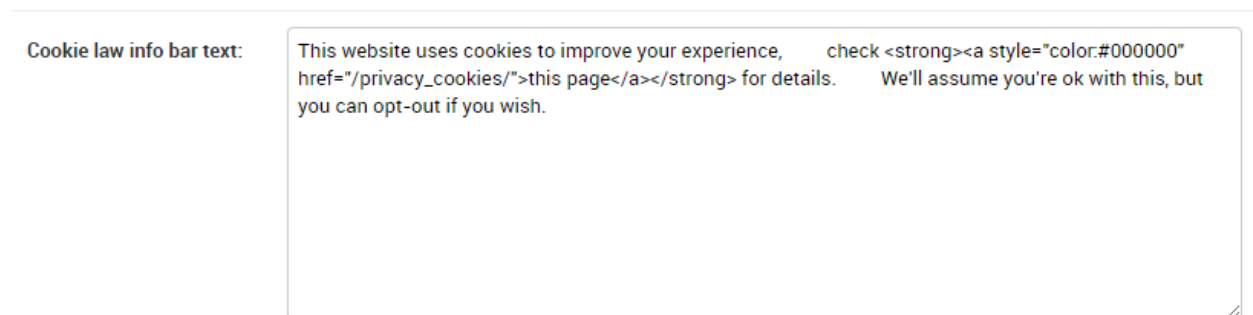


Fig. 265: Cookie law info bar text



Privacy & Cookies Policy

Privacy & Cookies Policy.

This site agrees to respect the privacy of the Website user in accordance with the applicable regulations on the protection of personal data and in particular EU Regulation 2016/679 (hereinafter the "Regulation", "GDPR").

This document ("Privacy & Cookies Policy") provides information on the processing of personal data collected through this Website (hereinafter "Website") and therefore constitutes information to the data subjects in accordance with the aforementioned regulations. Within the specific area of the Website, which collects the personal information of the user, a specific policy is normally published. The following information applies only to this Website and not to other websites accessed via links.

Pursuant to Article 13 of the Regulation, we hereby provide the following information:

DATA CONTROLLER

The Data Controller is **GEOSOLUTIONS DI GIANNECCHINI SIMONE & C.**, #DATA_CONTROLLER_ADDRESS; Tel. ##DATA_CONTROLLER_PHONE e-mail: #DATA_CONTROLLER_EMAIL.

WHAT DATA DO WE PROCESS?

The following data may be subject to processing:

Browsing Data

The processing of personal data of users who visit only the Website (i.e. without sending communications or using reserved areas) is limited to the navigation data, i.e. those for which the transmission to the Website is necessary for the operation of IT systems responsible for the management of the Website and the Internet communication protocols. This category includes the IP addresses or domain of the computer used to visit the Website and other parameters relative to the operating system used by the user to connect to the Website. The Company collects these and other data (such as, for example, the number of visits and the time spent on the Website) only for statistical purposes and in anonymous form in order to control the operation of the Website and improve its functionality. This is information that is not collected for the association with other information about users and to identify the latter; however, by their very nature, these data can allow the identification of users through processing and association with data held by third parties.

The legal basis for this processing is the legitimate interest of the Data Controller in the technical management related to the functionality and safety of the Website as defined by Art. 6.1. (f) of the Regulation

Cookies

Cookies are small text files, which the Web site places on the devices in use, such as computers or mobile devices, stored in directories used by the user's web browser. There are various types of cookies, some make the Website experience more efficient, others to enable certain functions.

The Website uses "technical" cookies, such as navigation or session cookies, or tools to make functional and optimize the navigation and use of the Website.

Fig. 266: */privacy_cookies/ Default Page*

The page contains a default generic text along with some placeholders, which, most probably, won't meet your needs.

In order to change this you have two options:

1. Change the link reported into the `Cookie law info bar` text section, to make it pointing to an external/static page.
2. Change the contents of `/geonode/templates/privacy-cookies.html` Django template accordingly to your needs; this is basically a plain HTML page which can be easily customized by using a standard text editor.

Switching between different themes

In the case you have defined more Themes, switching between them is as easy as enabling one and disabling the others.

Remember to save the Themes everytime and refresh the GeoNode home page on the browser to see the changes.

It is also important that there is **only one** Theme enabled **at a time**.

In order to go back to the standard GeoNode behavior, just disable or delete all the available Themes.

1.25.4 Add a new user

In GeoNode, administrators can manage other users. For example, they can *Add New Users* through the following form.

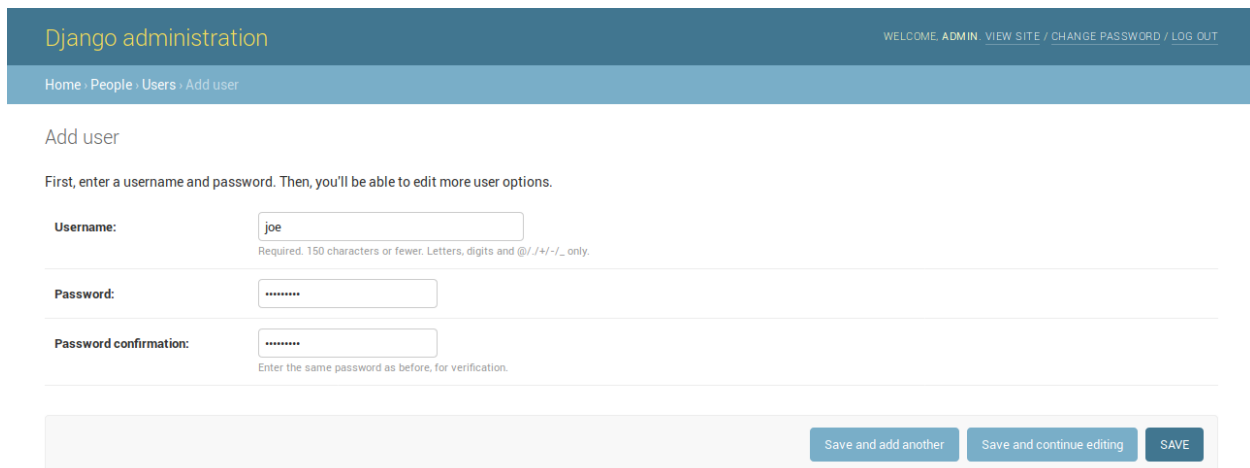
The image shows a screenshot of the Django administration interface for adding a new user. At the top, there's a dark blue header with 'Django administration' on the left and 'WELCOME, ADMIN | VIEW SITE | CHANGE PASSWORD | LOG OUT' on the right. Below the header is a light blue breadcrumb trail: 'Home > People > Users > Add user'. The main content area is white and titled 'Add user'. It contains a sub-header: 'First, enter a username and password. Then, you'll be able to edit more user options.' There are three input fields: 'Username:' with the value 'joe' and a note 'Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.'; 'Password:' with masked characters; and 'Password confirmation:' with a note 'Enter the same password as before, for verification.' At the bottom right, there are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

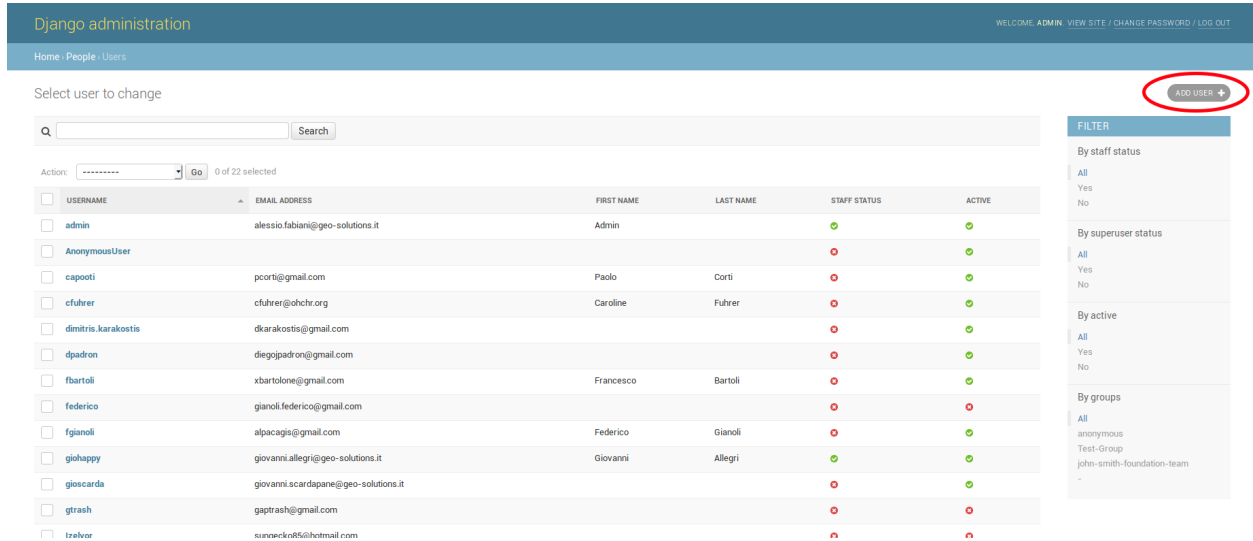
Fig. 267: Adding New Users

The form above can be reached from the *Admin Panel* at the following path: *Home > People > Users*. Click on *ADD USER +* to open the form page.

It is also available, in the GeoNode UI, the *Add User* link of the *About* menu in the navigation bar.

To perform the user creation fill out the required fields (*username* and *password*) and click on *SAVE*. You will be redirected to the *User Details Page* which allows to insert further information about the user.

The user will be visible into the *Users List Page* of the *Admin Panel* and in the *People Page* (see [Viewing other users information](#)).



Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · People · Users

Select user to change

Q Search

Action: Go 0 of 22 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS	ACTIVE
<input type="checkbox"/>	admin	alessio.fabiani@geo-solutions.it	Admin		✓	✓
<input type="checkbox"/>	AnonymousUser				✗	✓
<input type="checkbox"/>	capoti	pcorti@gmail.com	Paolo	Corti	✗	✓
<input type="checkbox"/>	cfuhrer	cfuhrer@ohchr.org	Caroline	Fuhrer	✗	✓
<input type="checkbox"/>	dimitris.karakostis	dkarakostis@gmail.com			✗	✓
<input type="checkbox"/>	dpadron	diegopadron@gmail.com			✗	✓
<input type="checkbox"/>	fbartoli	xbartolone@gmail.com	Francesco	Bartoli	✗	✓
<input type="checkbox"/>	federico	gianoli.federico@gmail.com			✗	✗
<input type="checkbox"/>	fgianoli	alpacagis@gmail.com	Federico	Gianoli	✗	✓
<input type="checkbox"/>	giohappy	giovanni.allegri@geo-solutions.it	Giovanni	Allegri	✓	✓
<input type="checkbox"/>	gioscarda	giovanni.scardapane@geo-solutions.it			✗	✓
<input type="checkbox"/>	gtrash	gaptrash@gmail.com			✗	✗
<input type="checkbox"/>	tzalvor	sinopceknob5@hotmail.com			✗	✗

FILTER

By staff status

All
Yes
No

By superuser status

All
Yes
No

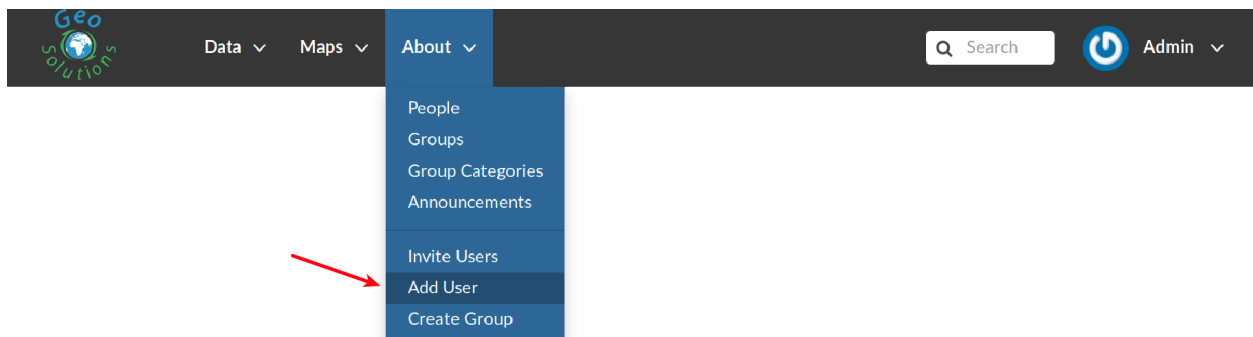
By active

All
Yes
No

By groups

All
anonymous
Test-Group
john-smith-foundation-team
-

Fig. 268: The Add User button in the Users List page



Geo Solutions

Data ▾ Maps ▾ About ▾

Search Admin ▾

- People
- Groups
- Group Categories
- Announcements
- Invite Users
- Add User**
- Create Group

Fig. 269: Add User Link

The screenshot shows the Django administration interface for changing user details. The page title is 'Django administration' with links for 'Home', 'People', 'Users', and 'joe'. The breadcrumb trail is 'Home > People > Users > joe'. The page has two tabs: 'HISTORY' and 'VIEW ON SITE'. The main content area is titled 'Change user' and contains the following sections:

- Username:** A text input field containing 'joe'. Below it, a note says 'Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.'
- Password:** A text input field containing a long alphanumeric string. Below it, a note says 'Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.'
- Personal info:** A section with three text input fields: 'First name' (containing 'Joe'), 'Last name' (containing 'Banana'), and 'Email address' (containing 'banana.joe@mail.com').
- Permissions:** A section with three checkboxes:
 - ☒ **Active**: Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
 - ☐ **Staff status**: Designates whether the user can log into this admin site.
 - ☐ **Superuser status**: Designates that this user has all permissions without explicitly assigning them.
- Groups:** A section with two panels. The left panel, 'Available groups', has a search filter and lists 'joe', 'smith', and 'foundation'. The right panel, 'Chosen groups', lists 'anonymous'. A green plus sign is next to the right panel.

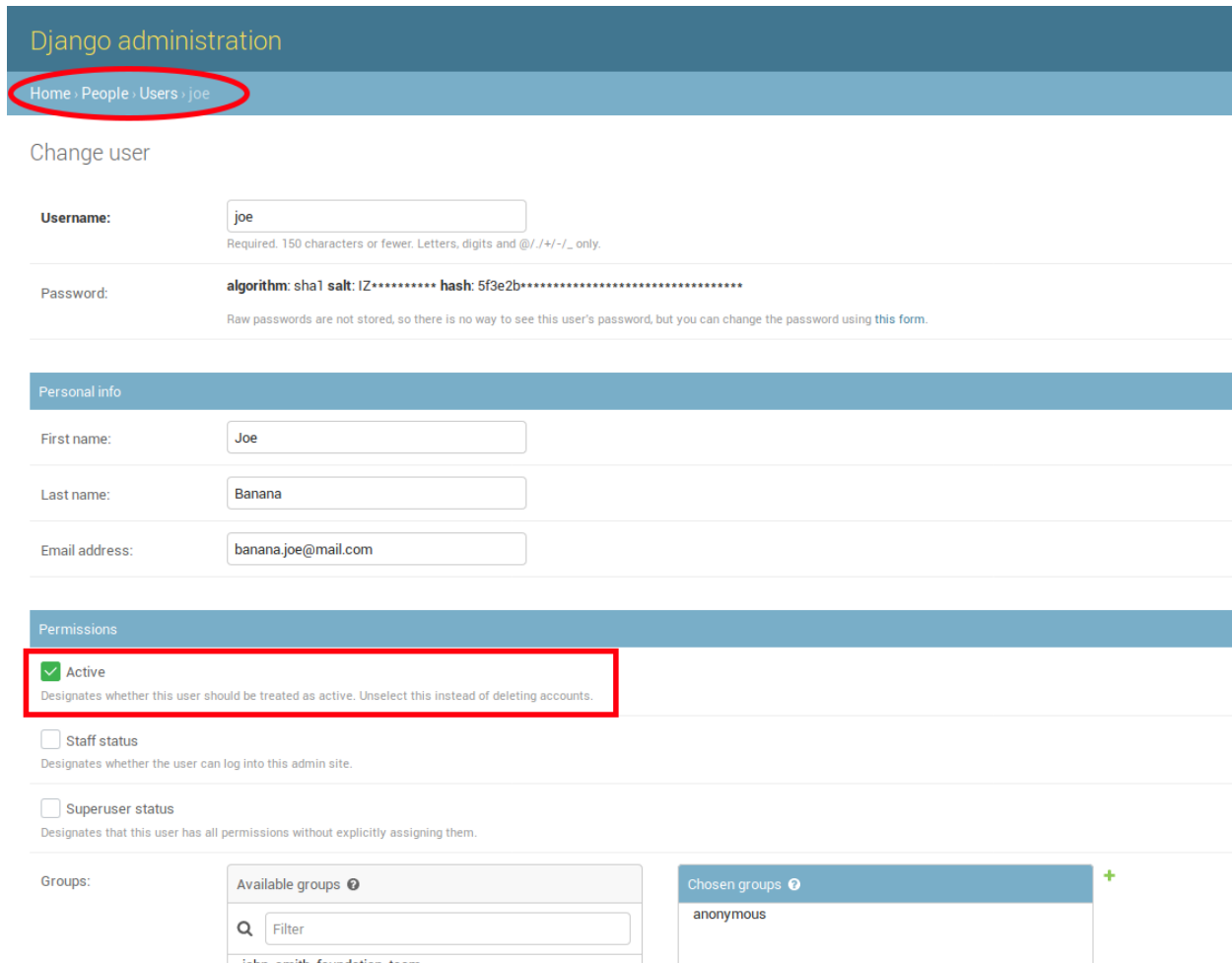
Fig. 270: The User Details Page

The screenshot shows the GeoNode 'Explore People' page. The top navigation bar includes the GeoNode logo, links for 'Data', 'Maps', and 'About', a search bar, and an 'Admin' button. The main heading is 'Explore People'. Below the heading, there is a search bar with a dropdown arrow and the word 'SEARCH'. The search bar contains the text 'joe', which is circled in red. To the right of the search bar, it says 'Total: 1'. Below the search bar, there is a user profile card for 'Joe Banana'. The card has a blue circular icon with a white 'G' and the text 'Joe Banana', 'No Organization Info', and three icons (a diamond, a location pin, and a document) each followed by a '0'.

Fig. 271: The User in the People page

1.25.5 Activate/Disable a User

When created, new users are *active* by default. You can check that in the *User Details Page* from the *Admin Panel* (see the picture below).



Django administration

Home > People > Users > joe

Change user

Username:
Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm: sha1 salt: lZ***** hash: 5f3e2b*******
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).

Personal info

First name:

Last name:

Email address:

Permissions

☒ **Active**
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ Staff status
Designates whether the user can log into this admin site.

☐ Superuser status
Designates that this user has all permissions without explicitly assigning them.

Groups:

Available groups ⓘ

Filter

infn_emith_foundation_team

Chosen groups ⓘ

anonymous

Fig. 272: New Users Active by default

Active users can interact with other users and groups, can manage resources and, more in general, can take actions on the GeoNode platform.

Untick the *Active* checkbox to disable the user. It will be not considered as user by the GeoNode system.

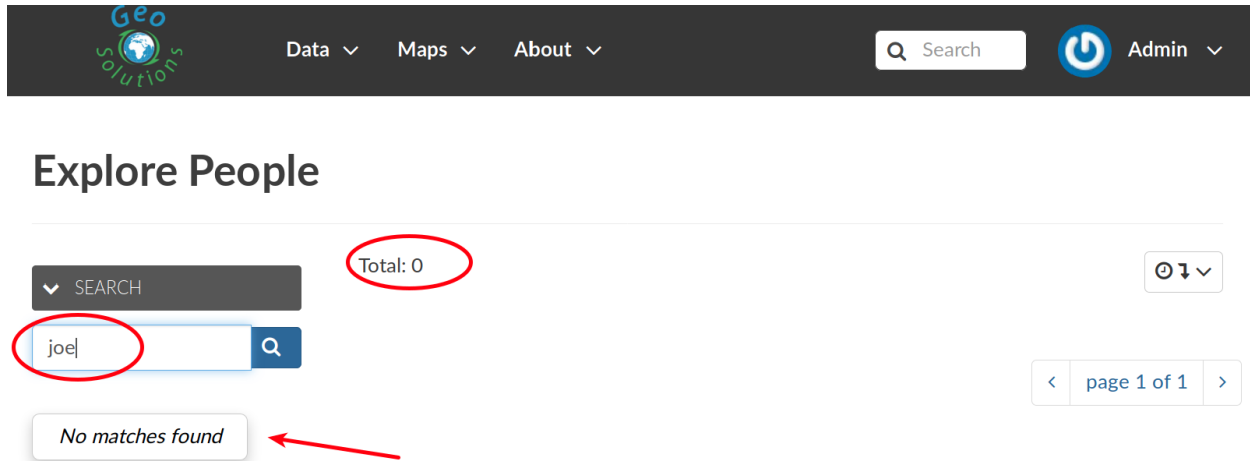


Fig. 273: Disabled Users

1.25.6 Change a User password

GeoNode administrators can also change/reset the password for those users who forget it. As shown in the picture below, click on [this](#) form link from the *User Details Page* to access the *Change Password Form*.

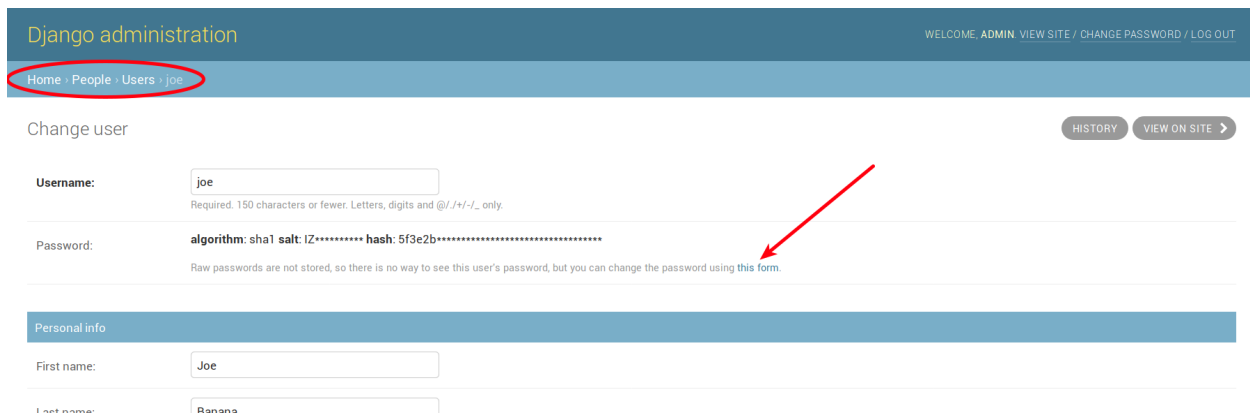


Fig. 274: Changing Users Passwords

The *Change User Password Form* should look like the following one. Insert the new password two times and click on **CHANGE PASSWORD**.

1.25.7 Promoting a User to Staff member or superuser

Active users have not access to admin tools. GeoNode makes available those tools only to *Staff Members* who have the needed permissions. *Superusers* are staff members with full access to admin tools (all permissions are assigned to them).

Administrators can promote a user to *Staff Member* by ticking the **Staff status** checkbox in the *User Details Page*. To make some user a *Superuser*, the **Superuser status** checkbox should be ticked. See the picture below.

GeoNode administration

Home › People › Users › joe › Change password

Change password: joe

Enter a new password for the user **joe**.

Password:

Password (again):
Enter the same password as before, for verification.

CHANGE PASSWORD

Fig. 275: Changing Users Passwords

Django administration WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › People › Users › joe

Change user HISTORY VIEW ON SITE

Username:
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm:** sha1 **salt:** IZ***** **hash:** 5f3e2b*****
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:

Last name:

Email address:

Permissions

☒ **Active**
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☒ **Staff status**
Designates whether the user can log into this admin site.

☒ **Superuser status**
Designates that this user has all permissions without explicitly assigning them.

Groups:

Available groups:

Chosen groups:

Fig. 276: Staff and Superuser permissions

1.25.8 Creating a Group

In GeoNode is possible to create new groups with set of permissions which will be inherited by all the group members.

The creation of a Group can be done both on the GeoNode UI and on the *Admin Panel*, we will explain how in this paragraph.

The *Create Groups* link of *About* menu in the navigation bar allows administrators to reach the *Group Creation Page*.

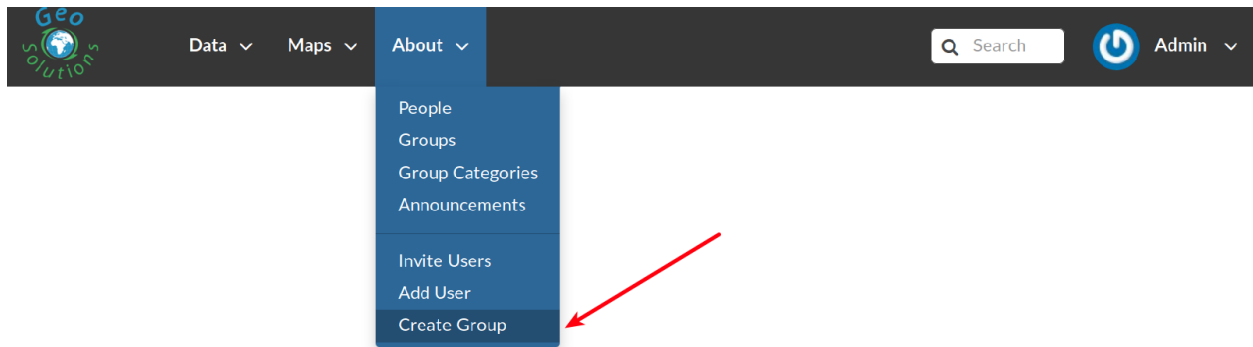


Fig. 277: The Create Group Link

The following form will open.

Fill out all the required fields and click *Create* to create the group. The *Group Details Page* will open.

The new created group will be searchable in the *Groups List Page*.

Note: The *Create a New Group* button on the *Groups List Page* allows to reach the *Group Creation Form*.

As already mentioned above, groups can also be created from the Django-based *Admin Interface* of GeoNode.

The *Groups* link of the *AUTHENTICATION AND AUTHORIZATION* section allows to manage basic Django groups which only care about permissions.

To create a GeoNode group you should take a look at the *GROUPS* section.

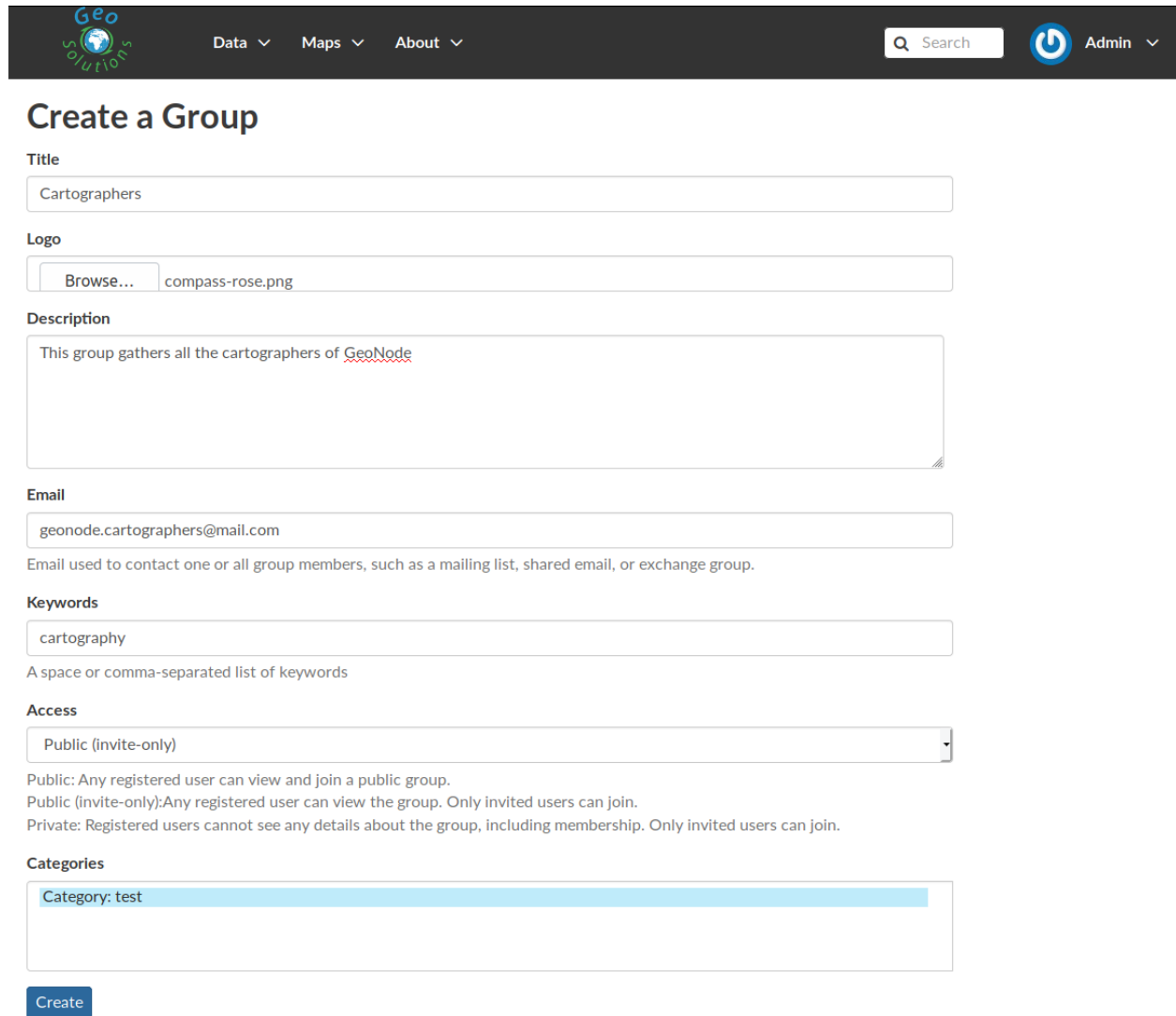
As you can see, GeoNode provides two types of groups. You will learn more about that in the next paragraph.

Types of Groups

In GeoNode users can be grouped through a *Group Profile*, an enhanced Django group which can be enriched with some further information such as a description, a logo, an email address, some keywords, etc. It is also possible to define some *Group Categories* based on which those group profiles can be divided and filtered.

A new **Group Profile** can be created as follows:

- click on the *Group Profile + Add* button
- fill out all the required fields (see the picture below), *Group Profiles* can be explicitly related to group categories
- click on *SAVE* to perform the creation, the new created group profile will be visible in the *Group Profiles List*



Create a Group

Title
Cartographers

Logo
Browse... compass-rose.png

Description
This group gathers all the cartographers of GeoNode

Email
geonode.cartographers@mail.com
Email used to contact one or all group members, such as a mailing list, shared email, or exchange group.


Keywords
cartography
A space or comma-separated list of keywords


Access
Public (invite-only)
Public: Any registered user can view and join a public group.
Public (invite-only): Any registered user can view the group. Only invited users can join.
Private: Registered users cannot see any details about the group, including membership. Only invited users can join.

Categories
Category: test

Create


Fig. 278: *The Group Creation Form*


Data ▾
Maps ▾
About ▾


Admin ▾

Cartographers

Last Modified: June 26, 2019, 12:41 p.m.



This group gathers all the cartographers of GeoNode

cartography

✉ geonode.cartographers@mail.com

test

[Edit Group Details](#)

[Manage Group Members](#)


[Delete this Group](#)

[Group Activities](#)

Permissions

This group is **Public (invite-only)**. Anyone may view this group but membership is by invitation only.


Managers



admin

GeoSolutions

Members

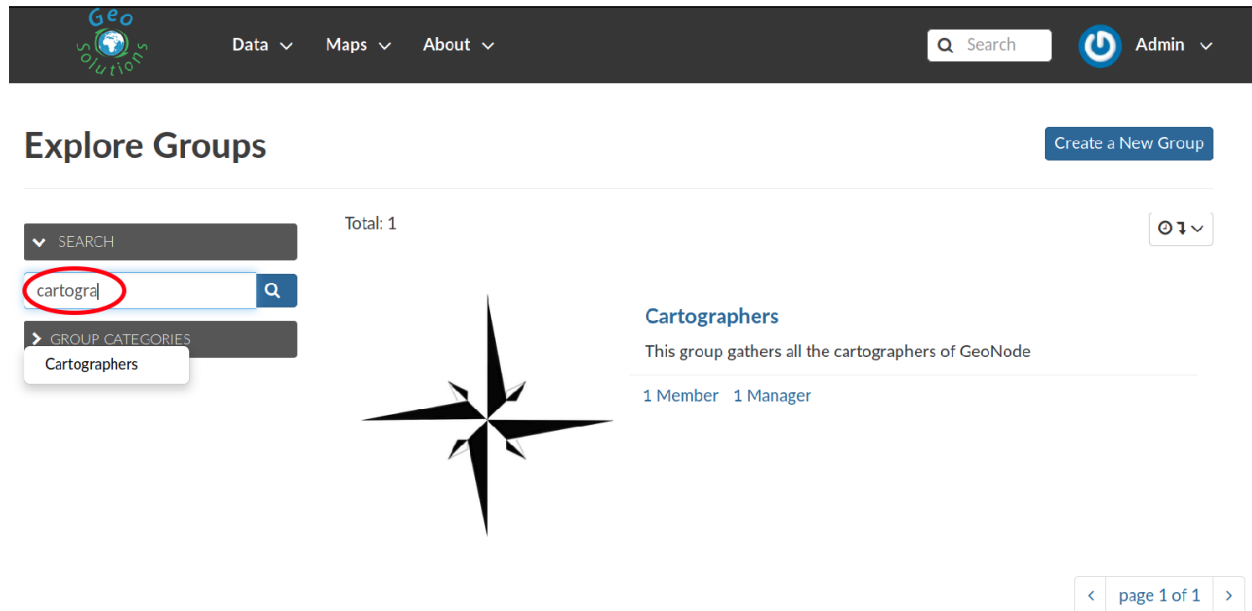


Admin

GeoSolutions

◇ 39
📍 10
📄 1

Fig. 279: *The Group Details Page*

Fig. 280: *The Groups List Page*

GROUPS		
Group Categories	+ Add	✎ Change
Group profiles	+ Add	✎ Change

Fig. 281: *The Groups Section on the Admin Panel*

Django administration
WELCOME, ADMIN / [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home / Groups / Group profiles / Add group profile

Add group profile

Title:

Title [en]:

Slug:

Logo:

Description:

Users interested in transports

Description [en]:

Email:

Email used to contact one or all group members, such as a mailing list, shared email, or exchange group.

Keywords:

A space or comma-separated list of keywords

Access:

Public

Public: Any registered user can view and join a public group.
Public (invite-only): Any registered user can view the group. Only invited users can join.
Private: Registered users cannot see any details about the group, including membership. Only invited users can join.

Categories:

Category: Transport

+

Hold down "Control", or "Command" on a Mac, to select more than one.

GROUP MEMBERS

USER	ROLE	JOINED	DELETE?
<input type="text" value="johnsmith"/> <input type="button" value="edit"/> <input type="button" value="add"/>	<input type="text" value="Manager"/>	Date: <input type="text" value="2019-06-26"/> Today <input type="button" value="calendar"/> Time: <input type="text" value="15:20:58"/> Now <input type="button" value="clock"/> <small>Note: You are 2 hours ahead of server time.</small>	
<input type="text" value="joe"/> <input type="button" value="edit"/> <input type="button" value="add"/>	<input type="text" value="Member"/>	Date: <input type="text" value="2019-06-26"/> Today <input type="button" value="calendar"/> Time: <input type="text" value="15:20:58"/> Now <input type="button" value="clock"/> <small>Note: You are 2 hours ahead of server time.</small>	
<input type="text" value="-----"/> <input type="button" value="edit"/> <input type="button" value="add"/>	<input type="text" value="-----"/>	Date: <input type="text" value="2019-06-26"/> Today <input type="button" value="calendar"/> Time: <input type="text" value="15:20:58"/> Now <input type="button" value="clock"/> <small>Note: You are 2 hours ahead of server time.</small>	

Add another Group member

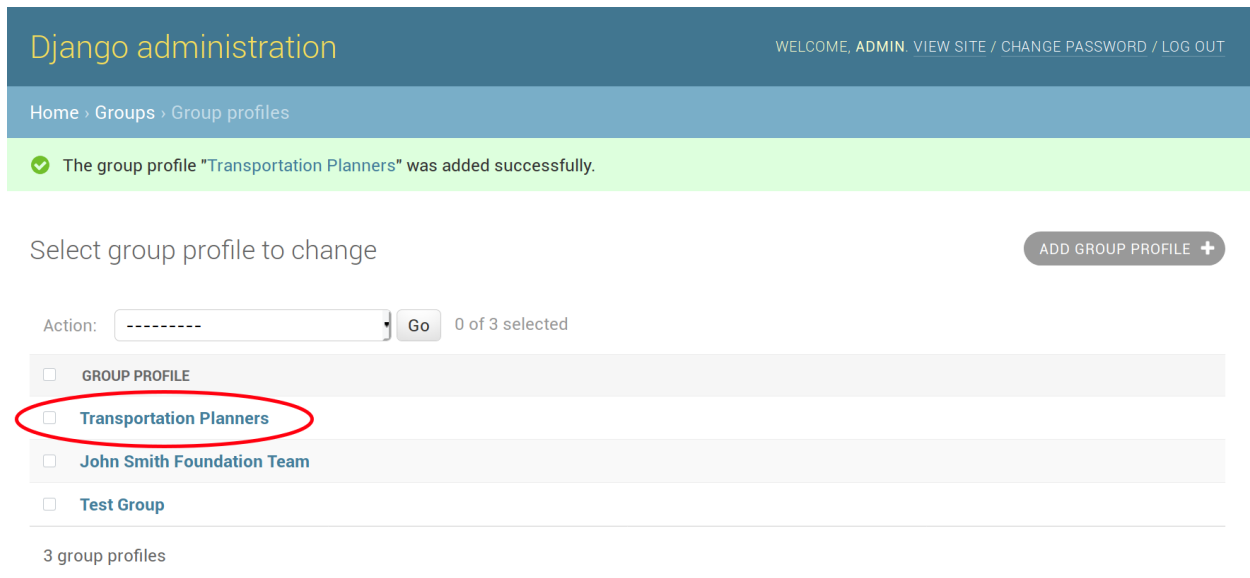


Fig. 283: The Group Profiles List

Group Categories

Group Profiles can also be related to *Group Categories* which represents common topics between groups. In order to add a new **Group Category** follow these steps:

- click on the *Group Categories + Add* button
- fill out the creation form (type *name* and *description*)
- click on *SAVE* to perform the creation, the new created category will be visible in the *Group Categories List*

When a GeoNode resource (layer, document or maps) is associated to some *Group Profile*, it is also possible to retrieve the *Group Category* it belongs to.

So when searching for resources (see [Finding Data](#)) you can also filter the data by group category.

1.25.9 Managing a Group

Through the *Groups* link of *About* menu in the navigation bar, administrators can reach the *Groups List Page*.

In that page all the GeoNode *Group Profiles* are listed.

For each group some summary information (such as the *title*, the *description*, the number of *members* and *managers*) are displayed near the *Group Logo*.

Administrators can manage a group from the *Group Profile Details Page* which is reachable by clicking on the *title* of the group.

As shown in the picture above, all information about the group are available on that page:

- the group *Title*;
- the *Last Editing Date* which shows a timestamp corresponding to the last editing of the group properties;
- the *Keywords* associated with the group;

Django administration WELCOME, ADMIN [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) > [Groups](#) > [Group Categories](#) > Add group category

Add group category

Name [en]:

Description:

Slug:

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

Fig. 284: A new Group Category

Django administration WELCOME, ADMIN [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) > [Groups](#) > [Group Categories](#)

Select group category to change [ADD GROUP CATEGORY +](#)

Action: [Go](#) 0 of 1 selected

<input type="checkbox"/>	NAME	SLUG
<input type="checkbox"/>	Transport	transport

1 group category


Fig. 285: The Group Categories List

The screenshot shows the 'Explore Layers' interface. On the left, a sidebar contains a 'Selected Layers' section and a 'Filters' section. The 'Filters' section has a 'Clear' button and a list of filter categories: TEXT, KEYWORDS, TYPE, CATEGORIES, OWNERS, GROUPS, GROUP CATEGORIES, DATE, REGIONS, and EXTENT. The 'GROUP CATEGORIES' filter is expanded, and 'Transport' is selected and circled in red. A red arrow points to the 'GROUP CATEGORIES' filter. The main content area shows '2 Layers found'. The first layer is 'roads' (Berlin Roads) by johnsmith, dated 10 Jun 2019, with 5 views and 0 shares. The second layer is 'railways' by johnsmith, dated 7 Jun 2019, with 2 views and 0 shares. Both layers are categorized under 'TRANSPORTATION' and 'TRANSPORTATION PLANNERS', which are circled in red. A red arrow points to the 'TRANSPORTATION PLANNERS' category for the 'roads' layer. The bottom right of the main content area shows 'page 1 of 1'.


Fig. 286: Filtering Layers by Group Category

The screenshot shows the top navigation bar of the GeoNode interface. The 'About' menu is open, displaying a list of options: People, Groups, Group Categories, Announcements, Invite Users, Add User, and Create Group. A red arrow points to the 'Groups' link in the dropdown menu.

Fig. 287: The Groups Link in the navigation bar




[Data](#) [Maps](#) [About](#)

 [Admin](#)

Explore Groups


Create a New Group


SEARCH



GROUP CATEGORIES

Total: 4






Test Group

Test


1 Member 1 Manager



John Smith Foundation Team

Members of the Team of John Smith


1 Member 1 Manager



Transportation Planners

Users interested in transports

3 Members 1 Manager




Cartographers

This group gathers all the cartographer of GeoNode


1 Member 1 Manager

< page 1 of 1 >

Fig. 288: Group Profiles List Page



[Data](#) [Maps](#) [About](#)

 [Admin](#)

John Smith Foundation Team

Last Modified: June 10, 2019, 2:51 p.m.

Members of the Team of John Smith

OSM john smith transport

[Edit Group Details](#)

[Manage Group Members](#)


[Delete this Group](#)

[Group Activities](#)


Permissions

This group is **Public**. Anyone may join this group.


Managers


 [admin](#)
GeoSolutions


Members



Admin
GeoSolutions

 39

 10

 1

[<](#) [page 1 of 1](#) [>](#)

Fig. 289: Group Profile Details Page

- *Permissions* on the group (Public, Public(invite-only), Private);
- *Members* who join the group;
- *Managers* who manage the group.

There are also four links:

- The *Edit Group Details* link opens the *Group Profile Form* through which the following properties can be changed:
 - *Title*.
 - *Logo* (see next paragraphs).
 - *Description*.
 - *Email*, to contact one or all group members.
 - *Keywords*, a comma-separated list of keywords.
 - *Access*, which regulates permissions:
 - * *Public*: any registered user can view and join a public group.
 - * *Public (invite-only)*: only invited users can join, any registered user can view the group.
 - * *Private*: only invited users can join the group, registered users cannot see any details about the group, including membership.
 - *Categories*, the group categories the group belongs to.
- *Managing Group Members* (see next paragraphs).
- the *Delete this Group*, click on it to delete the Group Profile. GeoNode requires you to confirm this action.
- the *Group Activities* drives you to the *Group Activities Page* where you can see all layers, maps and documents associated with the group. There is also a *Comments* tab which shows comments on those resources.

Group Logo

Each group represents something in common between its members. So each group should have a *Logo* which graphically represents the idea that identify the group.

On the *Group Profile Form* page you can insert a logo from your disk by click on *Browse...*

Click on *Update* to apply the changes.

Take a look at your group now, you should be able to see that logo.

The screenshot shows the GeoNode interface for the "John Smith Foundation Team". The top navigation bar includes the GeoSolutions logo, links for Data, Maps, and About, a search bar, and an Admin dropdown menu. The group title "John Smith Foundation Team" is prominently displayed, followed by the text "Last Modified: June 10, 2019, 2:51 p.m.". Below the title, a section titled "Members of the Team of John Smith" lists three members: OSM, john smith, and transport. To the right, a sidebar contains several management options: "Edit Group Details", "Manage Group Members", "Delete this Group", and "Group Activities". Below these is a "Permissions" section stating the group is "Public" and anyone can join. The "Managers" section shows the "admin" user from GeoSolutions. On the left, under the "Members" heading, a detailed view of the "Admin" member is shown, including their profile picture, name, organization, and statistics (39 diamonds, 10 location pins, 1 document). At the bottom right, a pagination control shows "page 1 of 1".

Fig. 290: *Group Profile Details Page*

Are you sure you want to remove the group: John Smith Foundation Team?

[Yes, I am sure](#) [No, don't remove it](#)

Fig. 291: *Confirm Group Deletion*

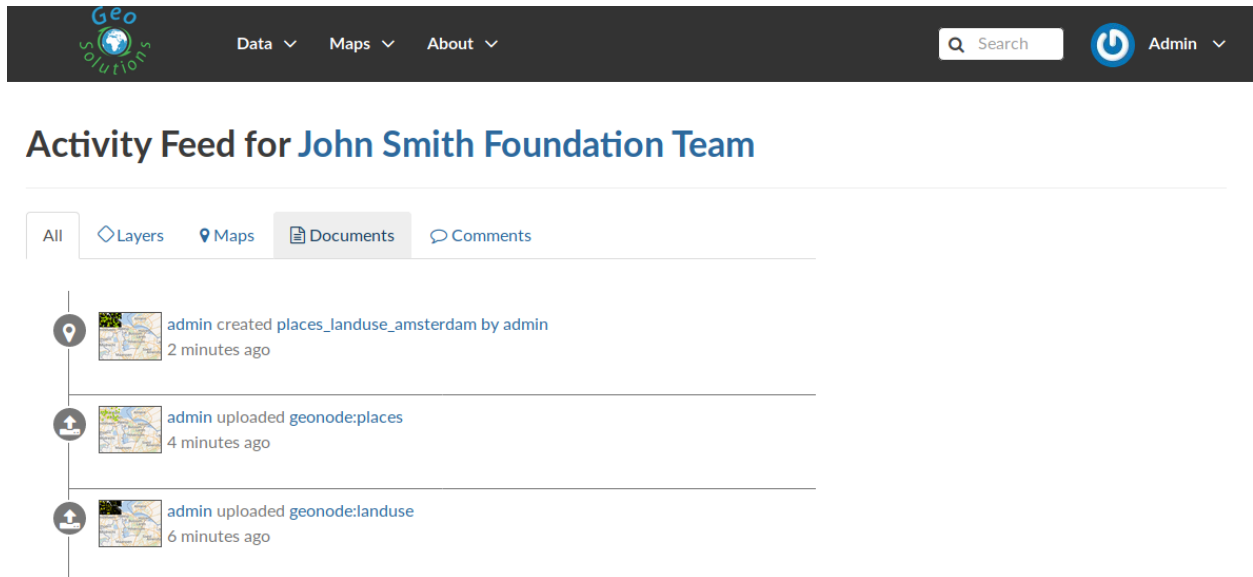


Fig. 292: Group Activities

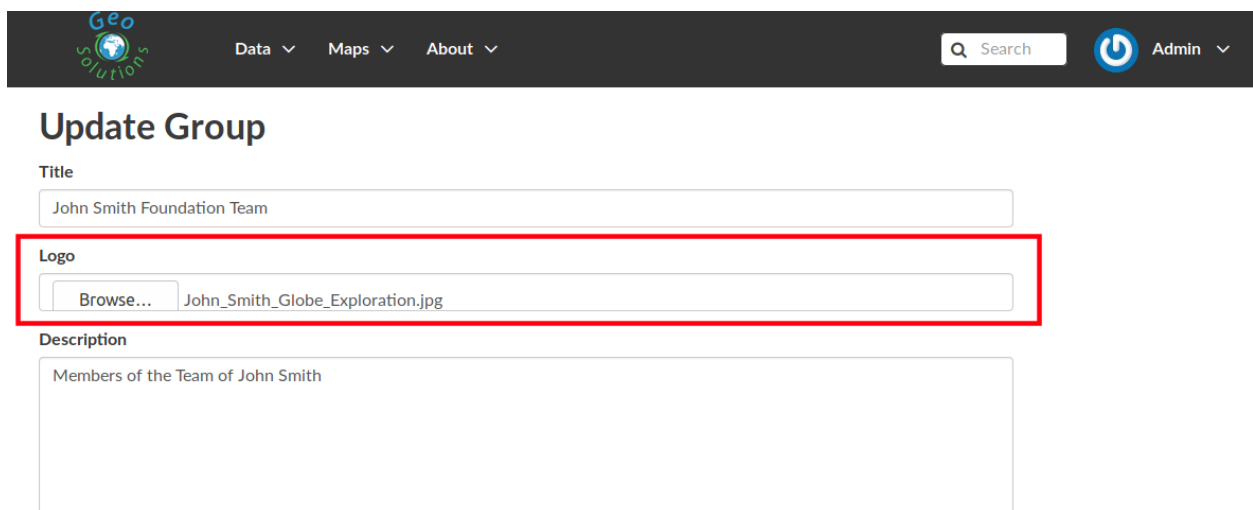




Fig. 293: Editing the Group Logo

[Data](#) [Maps](#) [About](#) [Admin](#)

John Smith Foundation Team

Last Modified: June 27, 2019, 9:14 a.m.



Members of the Team of John Smith


[OSM](#) [john smith](#) [transport](#)

[Edit Group Details](#)[Manage Group Members](#)[Delete this Group](#)[Group Activities](#)


Permissions

This group is **Public**. Anyone may join this group.

Managers

 **admin**
GeoSolutions

Members



Admin
GeoSolutions




 41  11  1

Fig. 294: *The Group Logo*

Managing Group members

The *Manage Group Members* link opens the *Group Members Page* which shows *Group Members* and *Group Managers*. **Managers** can edit group details, can delete the group, can see the group activities and can manage memberships. Other **Members** can only see the group activities.

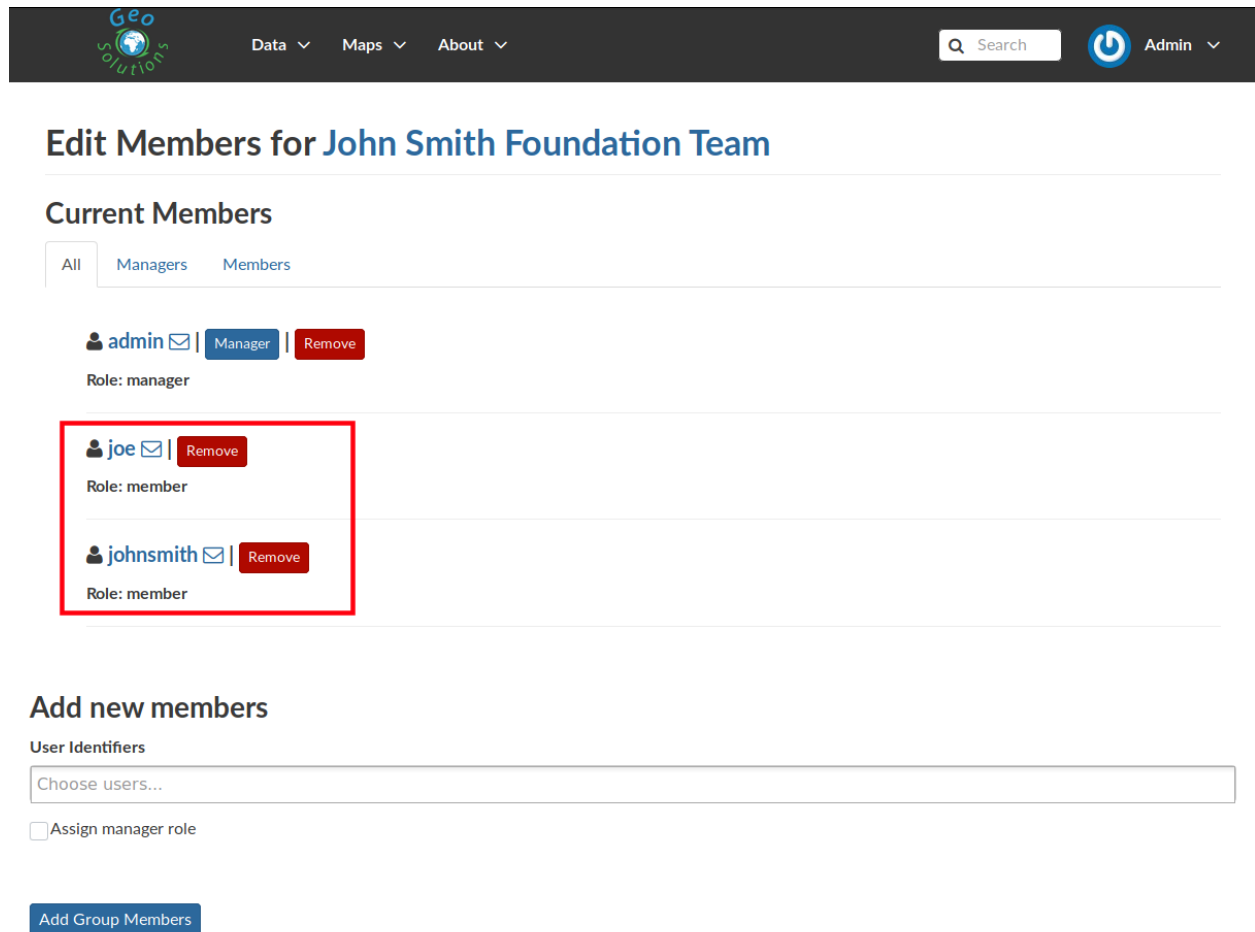
In Public Groups, users can join the group without any approval. Other types of groups require the user to be invited by the group managers.

Only group managers can *Add new members*. In the picture below, you can see the manager can search for users by typing their names into the *User Identifiers* search bar. Once found, he can add them to the group by clicking the *Add Group Members* button. The *Assign manager role* flag implies that all the users found will become managers of the group.

Fig. 295: Adding a new Member to the Group

The following picture shows you the results.

If you want to change the role of group members after adding them, you can use the “promote” button to make a member into a manager, and the “demote” button to make a manager into a regular member.



Edit Members for John Smith Foundation Team

Current Members

All Managers Members

admin | Manager | Remove
Role: manager

joe | Remove
Role: member

johnsmith | Remove
Role: member

Add new members

User Identifiers

Choose users...

☐ Assign manager role

Add Group Members

Fig. 296: *New Members of the Group*

1.25.10 Group based advanced data workflow

By default GeoNode is configured to make every resource (Layer, Document or Map) suddenly available to everyone, i.e. publicly accessible even from anonymous/non-logged in users.

It is actually possible to change few configuration settings in order to allow GeoNode to enable an advanced publication workflow.

With the advanced workflow enabled, your layer, document or map won't be automatically published (i.e. made visible and accessible for all, contributors or simple users).

For now, your item is only visible by yourself, the manager of the group to which the layer, document or map is linked (this information is filled in the metadata), the members of this group, and the GeoNode Administrators.

Before being published, the layer, document or map will follow a two-stage review process, which is described below:

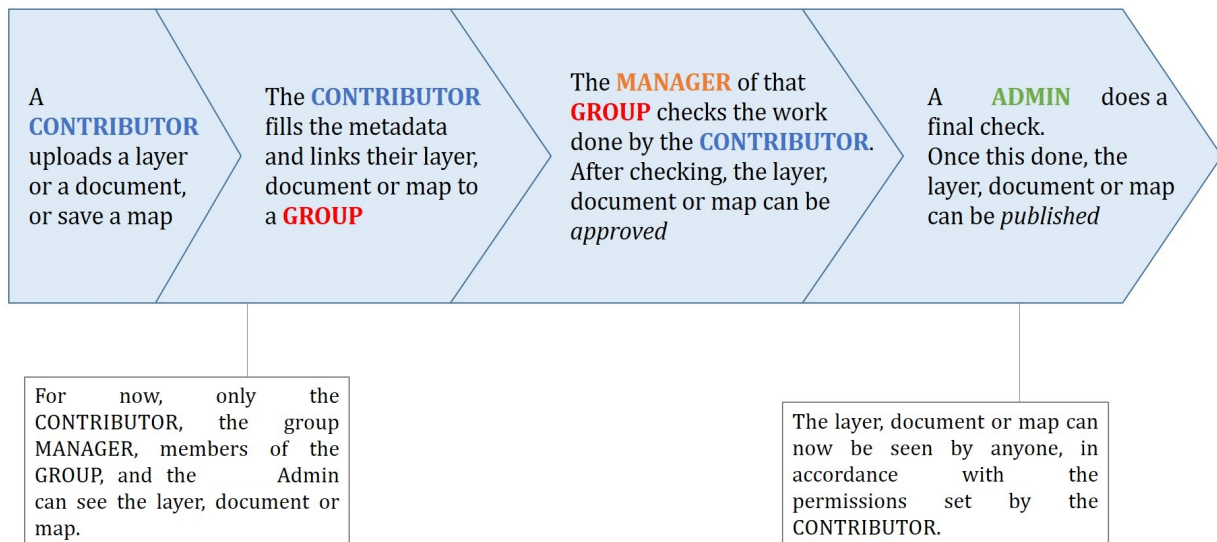


Fig. 297: From upload to publication: the review process on GeoNode

How to enable the advanced workflow

You have to tweak the GeoNode settings accordingly.

Please see the details of the following GeoNode Settings:

- ADMIN_MODERATE_UPLOADS
- GROUP_PRIVATE_RESOURCES
- RESOURCE_PUBLISHING

Change in owner rights in case of advanced workflow is on

After switching `ADMIN_MODERATE_UPLOADS` to True and resource is approved owner is no longer able to modify it. He will see new button on the resource detail page: Request change. After clicking this, view with short form is shown. On this view user can write short message why he want to modify the resource.

This message will be sent through messaging and email system to administrators:

After administrator unapprove the resource owner is again able to modify it.

The group Manager approval

Here, the role of the Manager of the group to which your layer, document or map is linked is to check that the uploaded item is correct. Particularly, in the case of a layer or a map, it consists of checking that the chosen cartographic representation and the style are fitting but also that the discretization is appropriate.

The Manager must also check that the metadata are properly completed and that the mandatory information (Title, Abstract, Edition, Keywords, Category, Group, Region) are filled.

If needed, the Manager can contact the contributor responsible of the layer, document or map in order to report potential comments or request clarifications.

Members of the group can also take part in the reviewing process and give some potential inputs to the responsible of the layer, document or map.

When the Manager considers that the layer, document or map is ready to be published, he should approve it. To do so, the Manager goes to the layer, document or map page, then opens the *Wizard* in order to edit the metadata. In the *Settings* tab, the manager checks the *Approved* box, and then updates the metadata and saves the changes:

Fig. 298: *The approbation process of an item by a Manager*

Following this approval, the GeoNode Administrators receive a notification informing them that an item is now waiting for publication

The publication by the GeoNode Administrator

Prior to the public release of an approved layer, a document or a map, the Administrator of the platform performs a final validation of the item and its metadata, notably to check that it is in line with license policies.

If needed, the GeoNode Administrator can contact the Manager who has approved the layer, document or map, as well as its responsible.

Once the layer, document or map is validated, the item is made public by the Administrator. It can now be viewed, accessed, and downloaded in accordance with the `Permissions` set by the responsible contributor.

1.25.11 Manage profiles using the admin panel

So far GeoNode implements two distinct roles, that can be assigned to resources such as layers, maps or documents:

- party who authored the resource
- party who can be contacted for acquiring knowledge about or acquisition of the resource

These two profiles can be set in the GeoNode interface by accessing the metadata page and setting the `Point of Contact` and `Metadata Author` fields respectively.

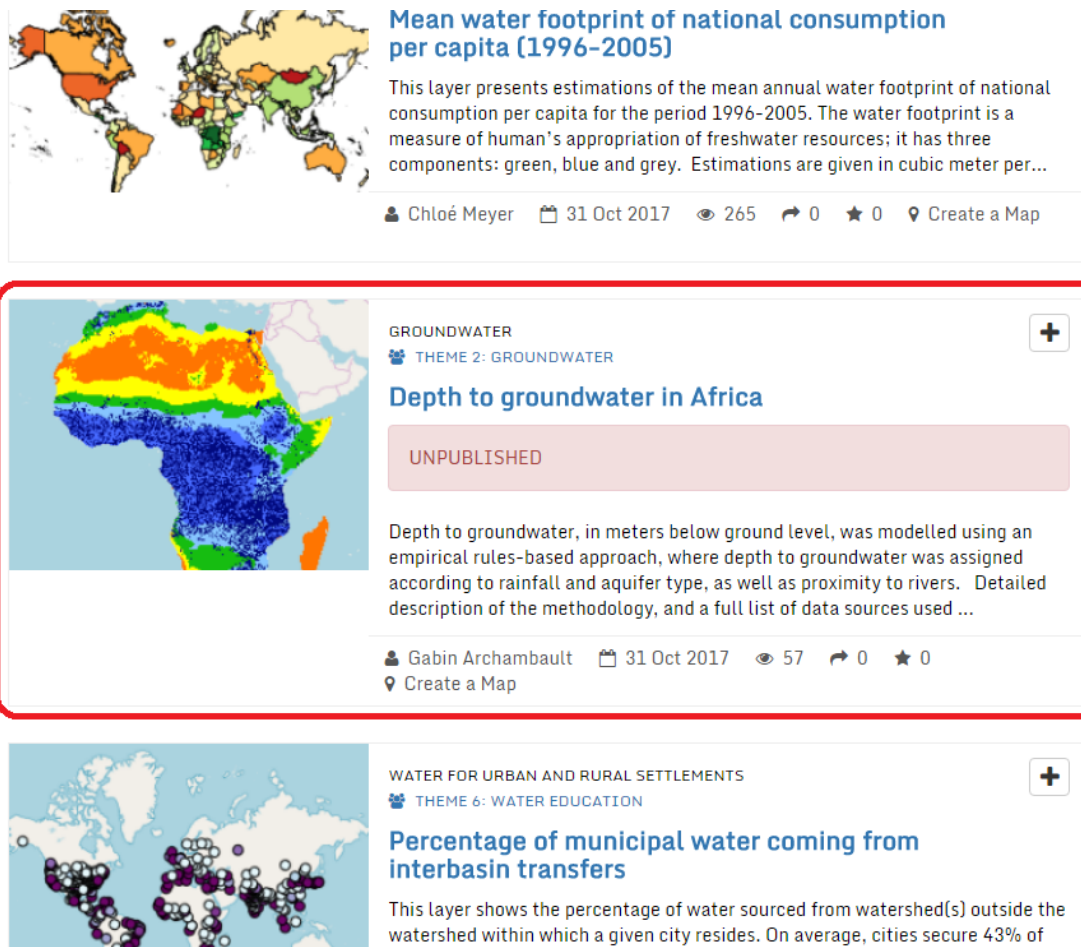
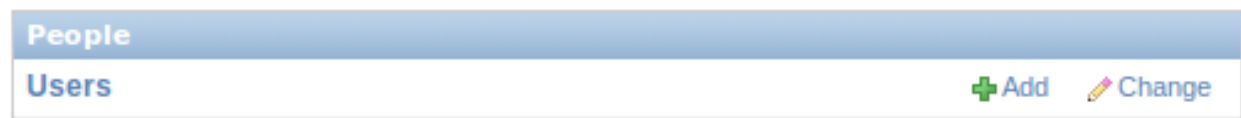


Fig. 299: An approved layer, waiting for publication by the GeoNode administrators

Is possible for an administrator to add new roles if needed, by clicking on the *Add Role* button in the *Base -> Contact Roles* section:

Clicking on the *People* section (see figure) will open a web for with some personal information plus a section called *Users*.



Is important that this last section is not modified here unless the administrator is very confident in that operation.

1.25.12 Manage layers using the admin panel

Some of the Layers information can be edited directly through the admin interface although the best place is in the *Layer -> Metadata Edit* in GeoNode.

Clicking on the *Admin > Layers* link will show the list of available layers.

Warning: It is not recommended to modify the Layers' *Attributes* or *Styles* directly from the Admin dashboard unless you are aware of your actions.

The *Metadata* information can be changed for multiple Layers at once through the *Metadata batch edit* action.

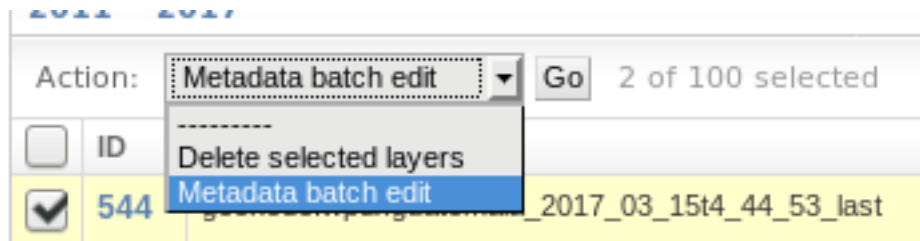
By clicking over one Layer link, it will show a detail page allowing you to modify some of the resource info like the metadata, the keywords, the title, etc.

Note: It is strongly recommended to always use the GeoNode *Metadata Wizard* or *Metadata Advanced* tools in order to edit the metadata info.

The *Permissions* can be changed also for multiple Layers at once through the *Set layers permissions* action.

By clicking over one Layer link, it will show a detail page allowing you to modify the permissions for the selected resources.

Layers	
Attributes	+ Add ✎ Change
Layers	+ Add ✎ Change
Styles	+ Add ✎ Change
Upload sessions	+ Add ✎ Change



Django administration

Home › Layers › Layers

Select layer to change

Q

◀ 2019 November 14

Action: 2 of 2 selected

<input checked="" type="checkbox"/>	ID	ALTERNATE	TITLE [EN]	DATE	CATE
<input checked="" type="checkbox"/>	28	geonode:tasmania_water_bodies	<input type="text" value="tasmania_water_bodies"/>	Nov. 14, 2019, 3:36 p.m.	---
<input checked="" type="checkbox"/>	27	geonode:tasmania_state_boundaries	<input type="text" value="tasmania_state_boundaries"/>	Nov. 14, 2019, 3:36 p.m.	---



Layers Permissions

Group

----- ▼

User

----- ▼

Permission Type

☒ Read

☐ Write

☐ Download

Mode

☒ Set

☐ Unset

Cancel Submit

1.25.13 Manage the maps using the admin panel

Similarly to the Layers, it is possible to manage the available GeoNode Maps through the Admin panel also.

Move to *Admin > Maps* to access the Maps list.

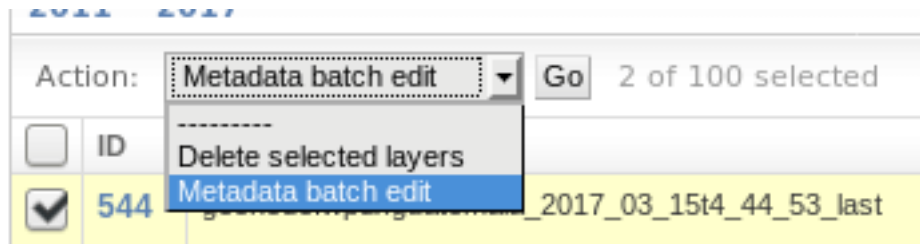
Maps	
Map layers	+ Add ✎ Change
Map snapshots	+ Add ✎ Change
Maps	+ Add ✎ Change

The Metadata information can be changed for multiple Maps at once through the *Metadata batch edit* action.

By clicking over one Map link, it will show a detail page allowing you to modify some of the resource info like the metadata, the keywords, the title, etc.

Note: It is strongly recommended to always use the GeoNode *Metadata Wizard* or *Metadata Advanced* tools in order to edit the metadata info.

Notice that by enabling the `Featured` option here, will allow GeoNode to show the Map thumbnail and the Map detail link on the *Home Page*



Metadata upload

☐ Metadata uploaded preserve

Metadata xml:

```
<gmd:MD_Metadata xmlns:gmd="http://www.isotc211.org/2005/gmd"/>
```

Popular count:

2

Share count:

0

☒ Featured

Should this resource be advertised in home page?



☒ Is Published


Should this resource be published and searchable?

☒ Approved

Is this resource validated from a publisher or editor?

Click to search for geospatial data published by other users, organizations and public sources. Download data in standard formats.

[Add layers »](#)




2 Documents

As for the layers and maps GeoNode allows to publish tabular and text data, manage their metadata and associated documents.

[Add documents »](#)

Data is available for browsing, aggregating and styling to generate maps which can be saved, downloaded, shared publicly or restricted to specify users only.

[Create maps »](#)

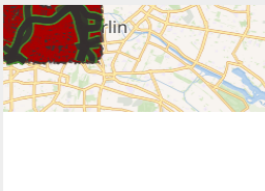


22 Users

Geonode allows registered users to easily upload geospatial data and various documents in several formats.

[See users »](#)

[Explore all datasets](#)



My Map

Data
Layers
Documents
Remote Feeds

Maps
Explore Maps
Create Map

About
People
Groups
Announcements

Contact Us
Mr. Pibody

1.25.14 Manage the documents using the admin panel

Similarly to the Layers and Maps, it is possible to manage the available GeoNode Documents through the Admin panel also.

Move to *Admin > Documents* to access the Documents list.

Documents

Documents [+ Add](#) [Change](#)

The Metadata information can be changed for multiple Documents at once through the *Metadata batch edit* action.

Action: **Metadata batch edit** [Go](#) 2 of 100 selected

<input type="checkbox"/>	ID	
<input checked="" type="checkbox"/>	544	_2017_03_15t4_44_53_last

Metadata batch edit

By clicking over one Document link, it will show a detail page allowing you to modify some of the resource info like the metadata, the keywords, the title, etc.

Note: It is strongly recommended to always use the GeoNode *Metadata Wizard* or *Metadata Advanced* tools in order to edit the metadata info.

1.25.15 Manage the base metadata choices using the admin panel

Admin > Base contains almost all the objects you need to populate the resources metadata choices.

In other words the options available from the *select-boxes* of the *Metadata Wizard* and *Metadata Advanced* panels.

Note: When editing the resource metadata through the *Metadata Wizard*, some fields are marked as `mandatory` and by filling those information the `Completeness` progress will advance accordingly.

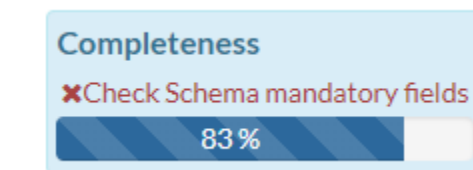


Fig. 303: *Metadata Completeness*

Even if not all the fields have been filled, the system won't prevent you to update the metadata; this is why the `Mandatory` fields are mandatory to be fully compliant with an ISO 19115 metadata schema, but are only recommended to be compliant with GeoNode.

Also the `Completeness` indicates how far the metadata is to be compliant with an ISO 19115 metadata schema.

Of course, it is **highly** recommended to always fill as much as possible at least all the metadata fields marked as `Mandatory`.

This will improve not only the quality of the data stored into the system, but will help the users to easily search for them on GeoNode.

All the `Search & Filter` panels and options of GeoNode are, in fact, based on the resources metadata fields. Too much generic descriptions and too empty metadata fields, will give highly un-precise and very wide search results to the users.

Hierarchical keywords

Through the *Admin > Base > Hierarchical keywords* panel it will be possible to manage all the keywords associated to the resources.

- The *Name* is the human readable text of the keyword, what users will see.
- The *Slug* is a unique label used by the system to identify the keyword; most of the times it is equal to the name.

Notice that through the *Position* and *Relative to* selectors, it is possible to establish a hierarchy between the available keywords. The hierarchy will be reflected in the form of a tree from the metadata panels.

By default each user with editing metadata rights on any resource, will be able to insert new keywords into the system by simply typing a free text on the keywords metadata field.

Django administration

Site administration

ACCOUNTS		
Email addresses	+ Add	Change

ACTSTREAM		
Actions	+ Add	Change
Follows	+ Add	Change

ANNOUNCEMENTS		
Announcements	+ Add	Change
Dismissals	+ Add	Change

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change

AVATAR		
Avatars	+ Add	Change

BASE		
Backups	+ Add	Change
Contact roles	+ Add	Change
Hierarchical keywords	+ Add	Change
Licenses	+ Add	Change
Links	+ Add	Change
Menu items	+ Add	Change
Menu placeholders	+ Add	Change
Menus	+ Add	Change
Metadata Regions	+ Add	Change
Metadata Restriction Code Types		Change
Metadata Spatial Representation Types		Change
Metadata Topic Categories	+ Add	Change

DIALOGOS		
Comments	+ Add	Change

Recent actions

My actions

- [Warning dear users](#)
Announcement
- [afabiani](#)
User
- [afabiani2](#)
User
- [John Smith Foundation Team](#)
Group profile
- [Warning dear users](#)
Announcement
- [Warning dear users](#)
Announcement
- [Warning dear users](#)
Announcement
- [Warning dear users](#)
Announcement
- [Transportation Planners](#)
Group profile
- [Dismissal object](#)
Dismissal

Fig. 300: Admin dashboard Base Panel

Metadata for places

Completeness
 ✖ Check Schema mandatory fields
 83%

Edit
Preview
Settings

Mandatory
Mandatory
Optional

1

Basic Metadata

2

Location and Licenses

3

Optional Metadata

4

Dataset Attributes

Language ?

English ▼

License ?

Not Specified ▼

 NextView
 Not Specified
 Open Data Commons Open Database License / OSM
Public Domain
 Public Domain / USG
 Varied / Derived
 Varied / Original

Regions

× Global

Data quality statement ?

General explanation of the data producer's knowledge about the lineage of a dataset

 Field declared Mandatory by the Metadata Schema

Restrictions ?

 * Field declared Mandatory by the Metadata Schema

Restrictions other ?

other restrictions and legal prerequisites for accessing and using the resource or metadata

Return to Layer
<< Back
Update
Next >>

Fig. 301: Metadata Wizard Panel

It is possible to force the user to select from a fixed list of keywords through the `FREE-TEXT_KEYWORDS_READONLY` setting.

When set to *True* keywords won't be writable from users anymore. Only admins can will be able to manage them through the *Admin > Base > Hierarchical keywords* panel.

Licenses

Through the *Admin > Base > Licenses* panel it will be possible to manage all the licenses associated to the resources. The license description and the info URL will be shown on the resource detail page.

The license text will be shown on the catalogue metadata XML documents.

Warning: It is **strongly** recommended to not publish resources without an appropriate license. Always make sure the data provider specifies the correct license and that all the restrictions have been honored.

Maintenance frequency

----- ▼

Free-text Keywords

features ×

ne_10m_populated_places_simple ×

buildings

builtup_area

catastro

catastro_sc

climate outlook 40 June July and August

Regions

× Global

Restrictions

----- ▼

Restrictions other

other restrictions and legal prerequisites for
accessing and using the resource or metadata

Fig. 302: Metadata Advanced Panel



Fig. 304: Hierarchical keywords list

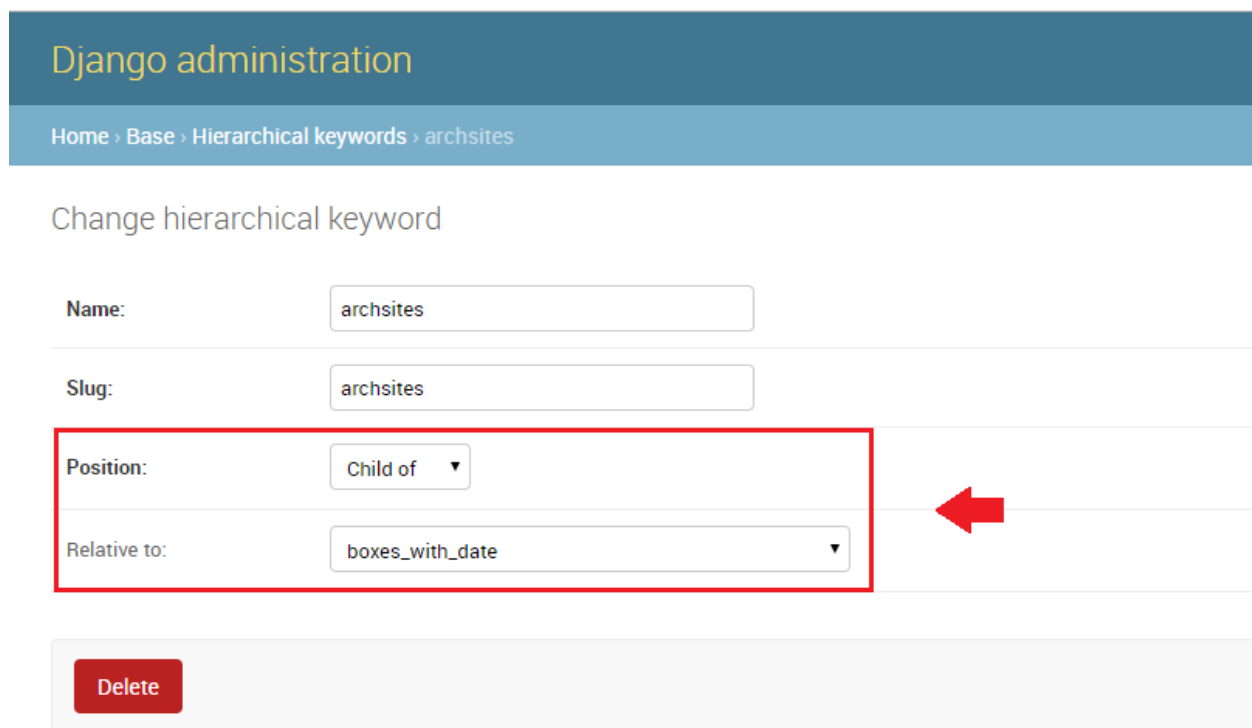


Fig. 305: Hierarchical keywords edit

GeoNode Metadata Editor interface showing the 'Licenses' section.

Navigation tabs: Edit, Preview, Settings.

Progress bar: Mandatory (Green), Mandatory (Red).

Diagram showing two steps:

- 1 Basic Metadata
- 2 Location and Licenses

Form fields and sections:

- Language ?**: English
- License ?**: Open Data Commons Open Database License / OSM (selected from dropdown menu)
- Regions**: × Global
- Data quality statement ?**: General explanation of the data and knowledge about the lineage (text area)

Field declared Mandatory by the Metadata

Fig. 306: Metadata editor Licenses



Fig. 307: *Resource detail License*

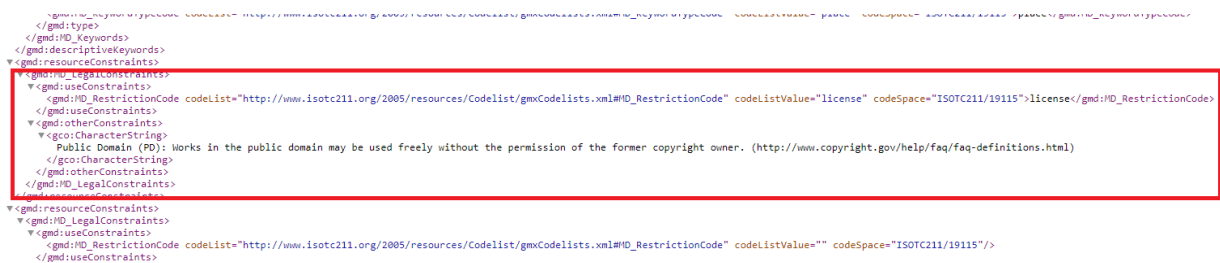


Fig. 308: *Resource Metadata ISO License*

Metadata Regions

Through the *Admin > Base > Metadata Regions* panel it will be possible to manage all the admin areas associated to the resources.

Fig. 309: Resource Metadata Regions

Notice that those regions are used by GeoNode to filter search results also through the resource list view.

Note: GeoNode tries to guess the Regions intersecting the data bounding boxes when uploading a new layer. Those should be refined by the user layer on anyway.

Metadata Restriction Code Types and Spatial Representation Types

Through the *Admin > Base > Metadata Restriction Code Types* and *Admin > Base > Metadata Spatial Representation Types* panels, it will be possible to **update only** the metadata descriptions for restrictions and spatial representation types.

Such lists are *read-only* by default since they have been associated to the specific codes of the ISO 19115 metadata schema. Changing them would require the system to provide a custom dictionary through the metadata catalog too. Such functionality is not supported actually by GeoNode.

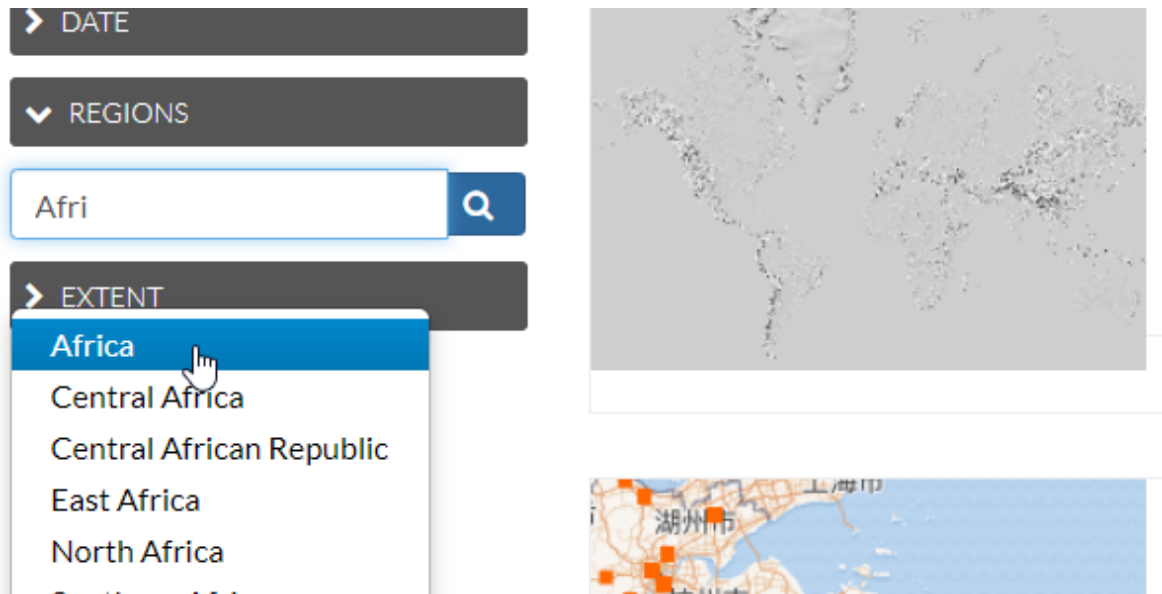


Fig. 310: GeoNode filtering by Metadata Regions

Metadata Topic Categories

Through the *Admin > Base > Metadata Topic Categories* panel it will be possible to manage all the resource metadata categories available into the system.

Notice that by default, GeoNode provides the standard topic categories available with the ISO 19115 metadata schema. Changing them means that the system won't be compliant with the standard ISO 19115 metadata schema anymore. ISO 19115 metadata schema extensions are not currently supported natively by GeoNode.

It is worth notice that GeoNode allows you to associate [Font Awesome Icons](#) to each topic category through their `fa-icon` code. Those icons will be used by GeoNode to represent the topic category on both the *Search & Filter* menus and *Metadata* panels.

Warning: The list of the Metadata Topic Categories on the home page is currently fixed. To change it you will need to update or override the `GeoNode index.html` HTML template.

By default the Metadata Topic Categories are *writable*. Meaning that they can be removed or created by the *Admin* panel.

It is possible to make them fixed (it will be possible to update their descriptions and icons only) through the `MODIFY_TOPICCATEGORY` setting.

1.25.16 Announcements

As an Administrator you might need to broadcast announcements to the world about your portal or simply to the internal contributors.

GeoNode `Announcements` allow actually to do that; an admin has the possibility to create three types of messages, accordingly to their severity, decide their validity in terms of time period (start date and expiring date of the announcement), who can view them or not (everyone or just the registered members) and whenever a user can hide the message or not and how long.

A GeoNode announcement actually looks like this:

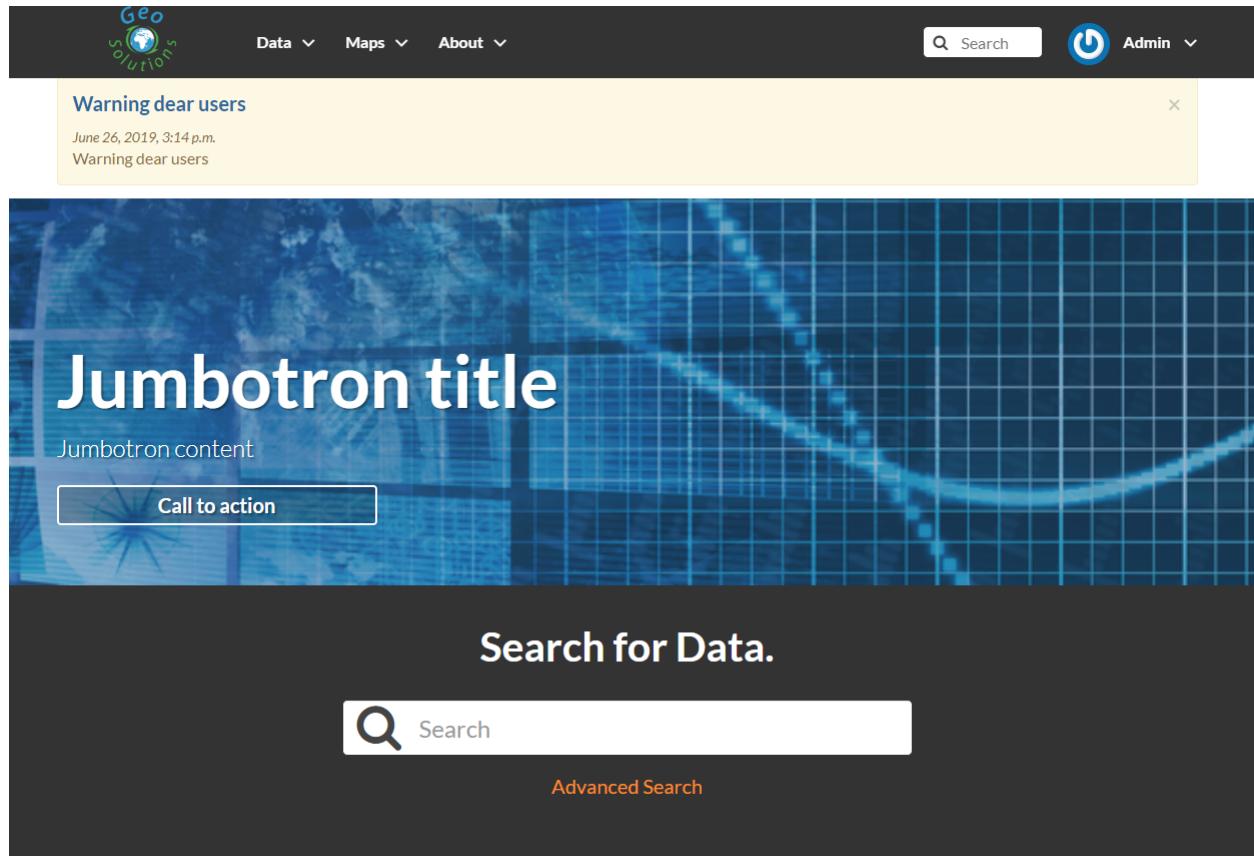


Fig. 311: A sample Warning Announcement

There are three types of announcements accordingly to their severity level: General, Warning and Critical. The difference is mainly the color of the announcement box.

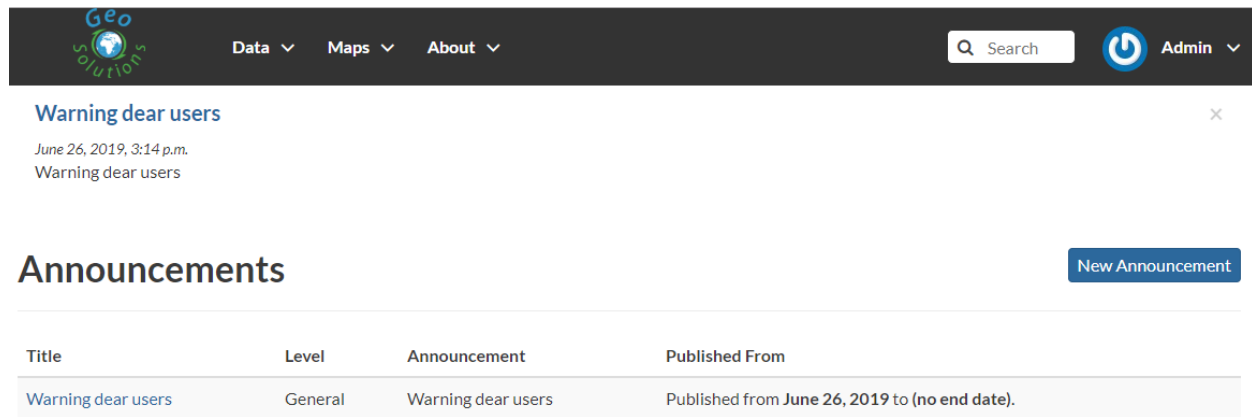
Only administrators and staff members can create and manage announcements.

Currently there are two ways to access and manage the announcements list:

1. Via the GeoNode interface, from the *Profile* panel

Note: Those are accessible by both admins and staff members.

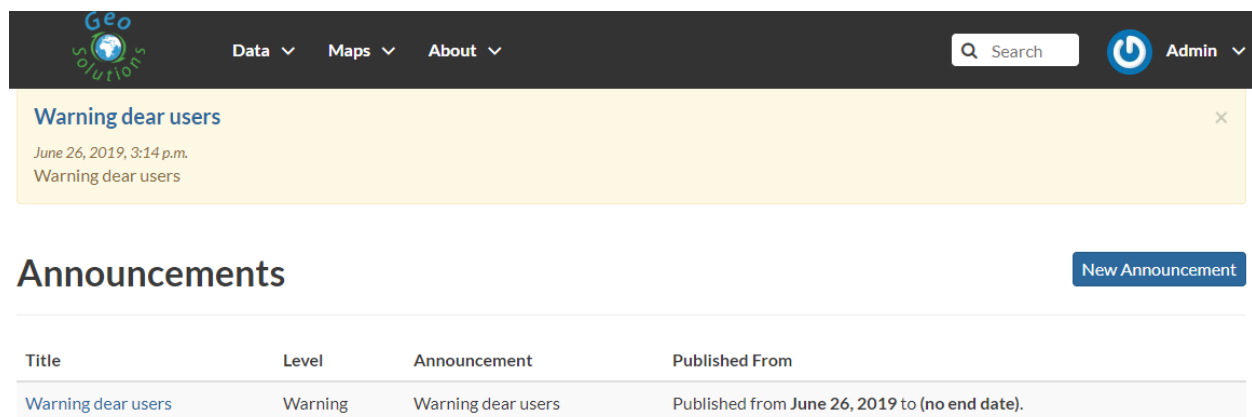
2. Via the GeoNode *Admin* panel



The screenshot shows the GeoNode web interface. At the top is a dark navigation bar with the GeoNode logo, 'Data', 'Maps', and 'About' menus, a search bar, and an 'Admin' button. Below the navigation bar, a yellow banner displays the announcement title 'Warning dear users', the timestamp 'June 26, 2019, 3:14 p.m.', and the message 'Warning dear users'. The main content area is titled 'Announcements' and includes a 'New Announcement' button. A table below lists the announcement details:

Title	Level	Announcement	Published From
Warning dear users	General	Warning dear users	Published from June 26, 2019 to (no end date).

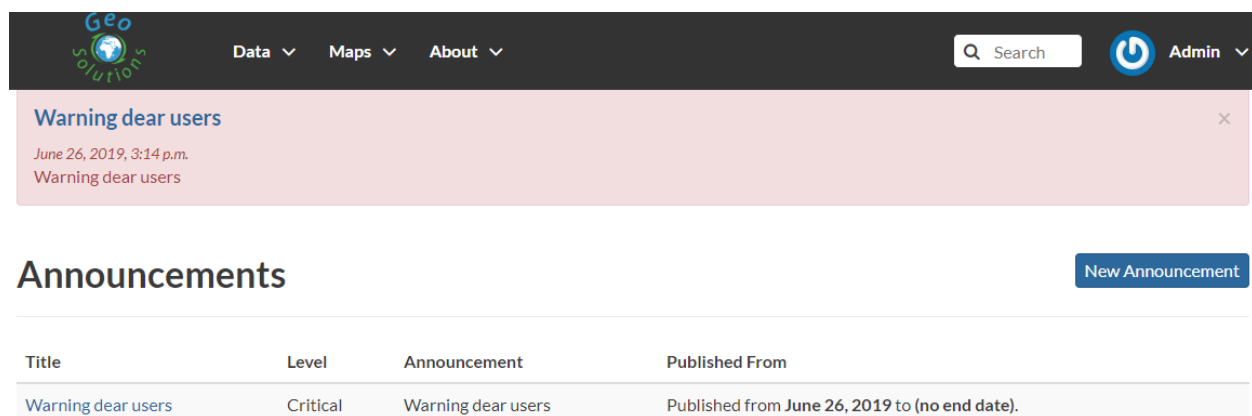
Fig. 312: General Announcement



This screenshot is similar to the previous one but shows the announcement with a 'Warning' level. The yellow banner at the top remains the same. The table below the 'Announcements' section is updated as follows:

Title	Level	Announcement	Published From
Warning dear users	Warning	Warning dear users	Published from June 26, 2019 to (no end date).

Fig. 313: Warning Announcement



This screenshot shows the announcement with a 'Critical' level. The yellow banner at the top remains the same. The table below the 'Announcements' section is updated as follows:

Title	Level	Announcement	Published From
Warning dear users	Critical	Warning dear users	Published from June 26, 2019 to (no end date).

Fig. 314: Critical Announcement

Admin (admin)

Admin

Position	Not provided.
Organization	GeoSolutions
Location	56017 PI ITA
Voice	Not provided.
Fax	Not provided.
Description	Not provided.
Keywords	Not provided

[User layers WMS GetCapabilities document](#)

- Message User
- Edit profile
- Connected social accounts
- Associated e-mails
- Set/Change password
- Upload new layers
- Create a new layer
- Create a new map
- My Activities
- Favorites
- Notifications
- Announcements**
- Invite Users
- GeoServer
- Admin

Fig. 315: *Announcements from the Profile panel*

Note: Those are accessible by admins only.

The functionalities are almost the same for both the interfaces, except that from the *Admin* panel it is possible to manage the dismissals too.

Dismissals are basically records of members that have read the announcement and closed the message box. An announcement can have one `dismissal` type among the three below:

1. *No Dismissal Allowed* it won't be possible to close the announcement's message box at all.
2. *Session Only Dismissal* (*) the default one, it will be possible to close the announcement's message box for the current browser session. It will show up again at next access.
3. *Permanent Dismissal Allowed* once the announcement's message box is closed, it won't appear again for the current member.

How to create and manage Announcements

From the *Profile* panel, click on `Announcements` link

Click either on *New Announcement* to create a new one or over a title of an existing one to manage its contents.

Create a new announcement is quite straight; you have to fill the fields provided by the form.

Warning: In order to be visible, you will need to check the *Site wide* option **in any case**. You might want to hide the message to *anonymous* users by enabling the *Members only* option too.

Managing announcements from the *Admin* panel, is basically the same; the fields for the form will be exactly the same.

Accessing announcements options from the *Admin* panel, allows you to manage dismissals also. Through this interface you will be able to selectively decide members which can or cannot view a specific announcement, or force them to visualize the messages again by deleting the dismissals accordingly.

1.25.17 Menus, Items and Placeholders

GeoNode provides some integrated functionalities allowing you to quickly and easily customize the top-bar menu (see the example below).

With minor changes of the `basic.html` template, potentially, it could be possible to use the same approach for a more complex customization. Let's start with the simple one.

By default GeoNode provides a custom placeholder already defined into the `basic.html` template, called `TOPBAR_MENU`

```
...
<ul class="nav navbar-nav navbar-right">

    {% block my_extra_right_tab %}

        {% render_nav_menu 'TOPBAR_MENU' %}

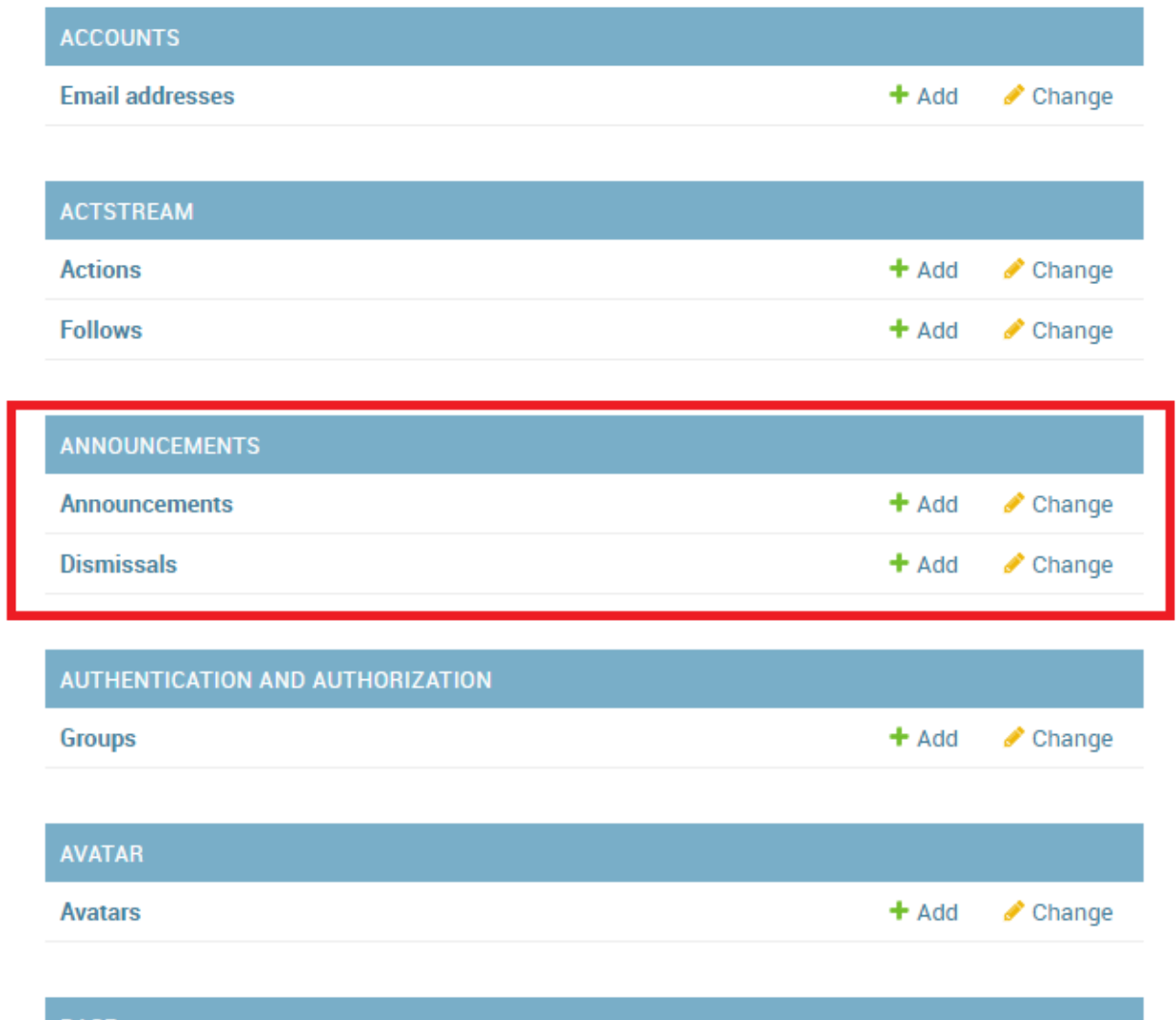
    {% endblock my_extra_right_tab %}

</li>
```

(continues on next page)

Django administration

Site administration



ACCOUNTS

Email addresses [+ Add](#) [Change](#)

ACTSTREAM

Actions [+ Add](#) [Change](#)

Follows [+ Add](#) [Change](#)

ANNOUNCEMENTS

Announcements [+ Add](#) [Change](#)

Dismissals [+ Add](#) [Change](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#) [Change](#)

AVATAR

Avatars [+ Add](#) [Change](#)

Fig. 316: *Announcements from the Admin panel*

Announcements



[New Announcement](#)

Title	Level	Announcement	Published From
Warning dear users 	Critical	Warning dear users	Published from June 26, 2019 to (no end date).

Fig. 317: *Announcements List from the Profile panel*

Create Announcement

Title

Level

General ▼

Content

☐ Site wide
☐ Members only

Dismissal type

Session Only Dismissal ▼

Publish_start

2019-06-27 13:24:04

Publish_end

Fig. 318: Create Announcement from the Profile panel

The screenshot shows the 'Change announcement' form in the Django admin panel. The form includes fields for Title (set to 'Warning dear users'), Level (set to 'Critical'), and Content (a text area with 'Warning dear users'). There are checkboxes for 'Site wide' and 'Members only'. The 'Publish_start' section has date and time pickers (set to 2019-06-26 15:12:57). The 'Publish_end' section also has date and time pickers. The 'Dismissal type' is set to 'Session Only Dismissal'. At the bottom, there are buttons for 'Delete', 'Save and add another', 'Save and continue editing', and 'SAVE'.

Fig. 319: Create Announcement from the Admin panel

The screenshot shows the 'Add dismissal' form in the Django admin panel. It includes a 'User' dropdown (set to 'afabiani'), an 'Announcement' dropdown (set to 'Warning dear users'), and a 'Dismissed at' section with date and time pickers (set to 2019-06-27 13:30:56). At the bottom, there are buttons for 'Save and add another', 'Save and continue editing', and 'SAVE'.

Fig. 320: Create Dismissal from the Admin panel

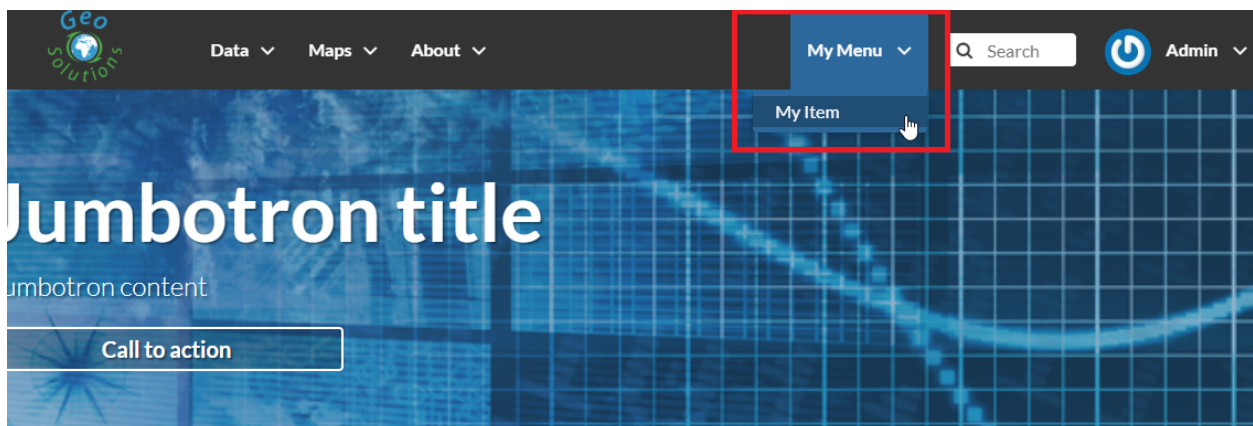


Fig. 321: GeoNode Top-Bar Menu customization

(continued from previous page)

```

<div class="search">
  <form id="search" action="{% url 'search' %}" >
    <span class="fa fa-search"></span>
    {% if HAYSTACK_SEARCH %}
    <input id="search_input" type="text" placeholder="{% trans 'Search' %}"
↪name="q">
    {% else %}
    <input id="search_input" type="text" placeholder="{% trans 'Search' %}"
↪name="title__icontains">
    {% endif %}
  </form>
</div>
</li>
...

```

From the *Admin > Base* panel, it is possible to access to the Menu, Menu Items and Menu Placeholder options.

BASE		
Backups	+ Add	Change
Contact roles	+ Add	Change
Hierarchical keywords	+ Add	Change
Licenses	+ Add	Change
Links	+ Add	Change
Menu items	+ Add	Change
Menu placeholders	+ Add	Change
Menus	+ Add	Change
Metadata Regions	+ Add	Change
Metadata Restriction Code Types		Change
Metadata Spatial Representation Types		Change
Metadata Topic Categories	+ Add	Change
DIALOGOS		

Fig. 322: Menu, Menu Items and Menu Placeholder options on the Admin panel

The hierarchical structure of a custom Menu is the following one:

1. **Menu Placeholder**; first of all you need to define a *placeholder* both into the *Admin > Base* panel and the `basic.html`.

By default GeoNode provides an already defined one called `TOPBAR_MENU`

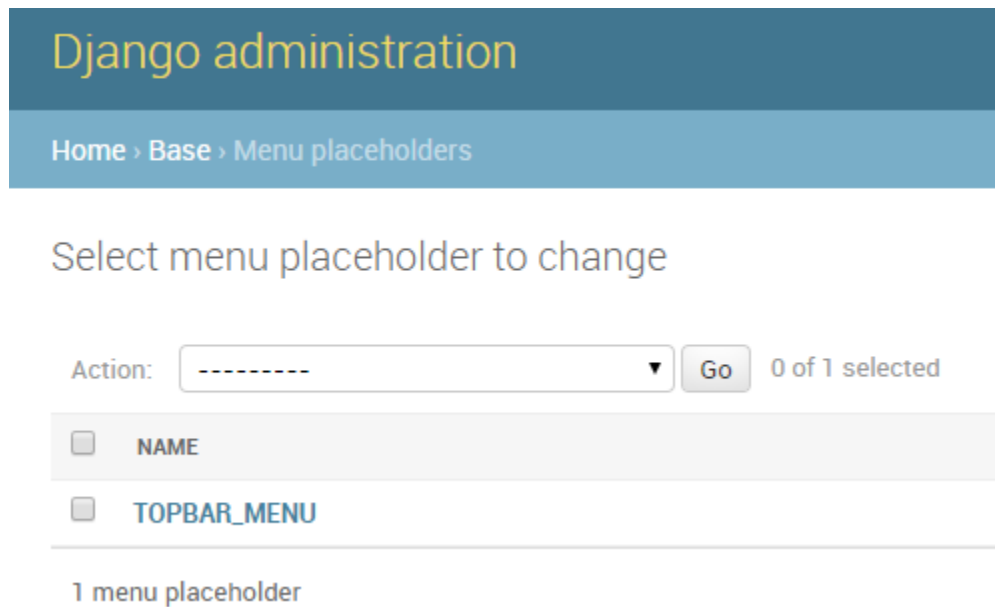


Fig. 323: The default `TOPBAR_MENU` Menu Placeholder on the Admin panel

2. **Menu;** second thing to do is to create a new *menu* associated to the corresponding *placeholder*. This is still possible from the *Admin > Base* panel

You will need to provide:

- A `Title`, representing the name of the `Menu` visible by the users

Warning: By using this approach, internationalization won't be supported. For the time being GeoNode does not support this for menus created from the *Admin > Base* panel.

- A `Menu Placeholder` from the existing ones.
- A `Order` in the case you'll create more menus associated to the same placeholder.

3. `Menu Item`; finally you will need to create voices belonging to the *menu*. For the time being, GeoNode allows you to create only `href` links.



Warning: The `Menu` won't be visible until you add at least one `Menu Item`

Django administration

Home › Base › Menus › My Menu

Change menu

Title:

Placeholder:  

Order:

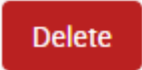


Fig. 324: Create a new Menu from the Admin panel

Django administration

Home › Base › Menu items › My Item

Change menu item



Title:	<input type="text" value="My Item"/>
Menu:	<div>My Menu ▼  </div>
Order:	<input type="text" value="1"/>
<input checked="" type="checkbox"/> Blank target	
Url:	<input type="text" value="https://dev.geonode.geo-solutions.it/"/>

Fig. 325: Create a new Menu Item from the Admin panel

1.25.18 OAuth2 Access Tokens

This small section won't cover entirely the GeoNode OAuth2 security integration, this is explained in detail in other sections of the documentation (refer to *OAuth2 Fixtures Update and Base URL Migration* and *OAuth2 Tokens and Sessions*).

Here we will focus mainly on the *Admin > DJANGO/GEONODE OAUTH TOOLKIT* panel items with a specific attention to the `Access tokens` management.

The *Admin > DJANGO/GEONODE OAUTH TOOLKIT* panel (as shown in the figure below) allows an admin to manage everything related to GeoNode OAuth2 grants and permissions.

As better explained in other sections of the documentation, this is needed to correctly handle the communication between GeoNode and GeoServer.



Fig. 326: *DJANGO/GEONODE OAUTH TOOLKIT Admin panel*

Specifically from this panel an admin can create, delete or extend OAuth2 `Access tokens`.

The section *OAuth2 Tokens and Sessions* better explains the concepts behind OAuth2 sessions; we want just to refresh the mind here about the basic concepts:

- If the `SESSION_EXPIRED_CONTROL_ENABLED` setting is set to *True* (by default it is set to *True*) a registered user cannot login to neither GeoNode nor GeoServer without a valid `Access token`.
- When logging-in into GeoNode through the sign-up form, GeoNode checks if a valid `Access token` exists and it creates a new one if not, or extends the existing one if expired.
- New `Access tokens` expire automatically after `ACCESS_TOKEN_EXPIRE_SECONDS` setting (by default 86400)
- When an `Access token` expires, the user will be kicked out from the session and forced to login again

Create a new token or extend an existing one

It is possible from the *Admin > DJANGO/GEONODE OAUTH TOOLKIT* panel to create a new `Access token` for a user.

In order to do that, just click on the *Add* button beside `Access tokens` topic

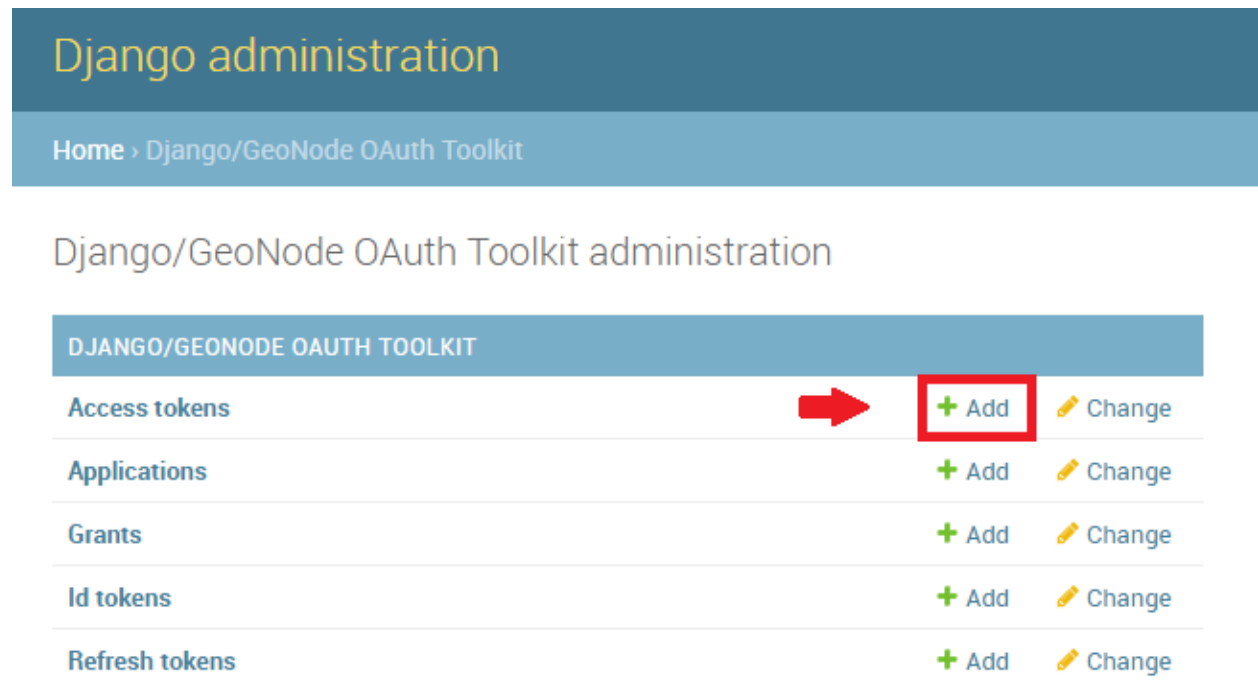


Fig. 327: Add a new `Access token`

On the new form


select the followings:




1. `User`; use the search tool in order to select the correct user. The form want the user PK, which is a number, and **not** the username. The search tool will do everything for you.
2. `Source refresh token`; this is not mandatory, leave it blank.
3. `Token`; write here any alphanumeric string. This will be the `access_token` that the member can use to access the OWS services. We suggest to use a service like <https://passwordsgenerator.net/> in order to generate a strong token string.
4. `Application`; select **GeoServer**, this is mandatory
5. `Expires`; select an expiration date by using the *date-time* widgets.
6. `Scope`; select **write**, this is mandatory.

Django administration



Home › Django/GeoNode OAuth Toolkit › Access tokens › Add access token

Add access token


User: 


Source refresh token:   

Token:

Application:  

Expires:

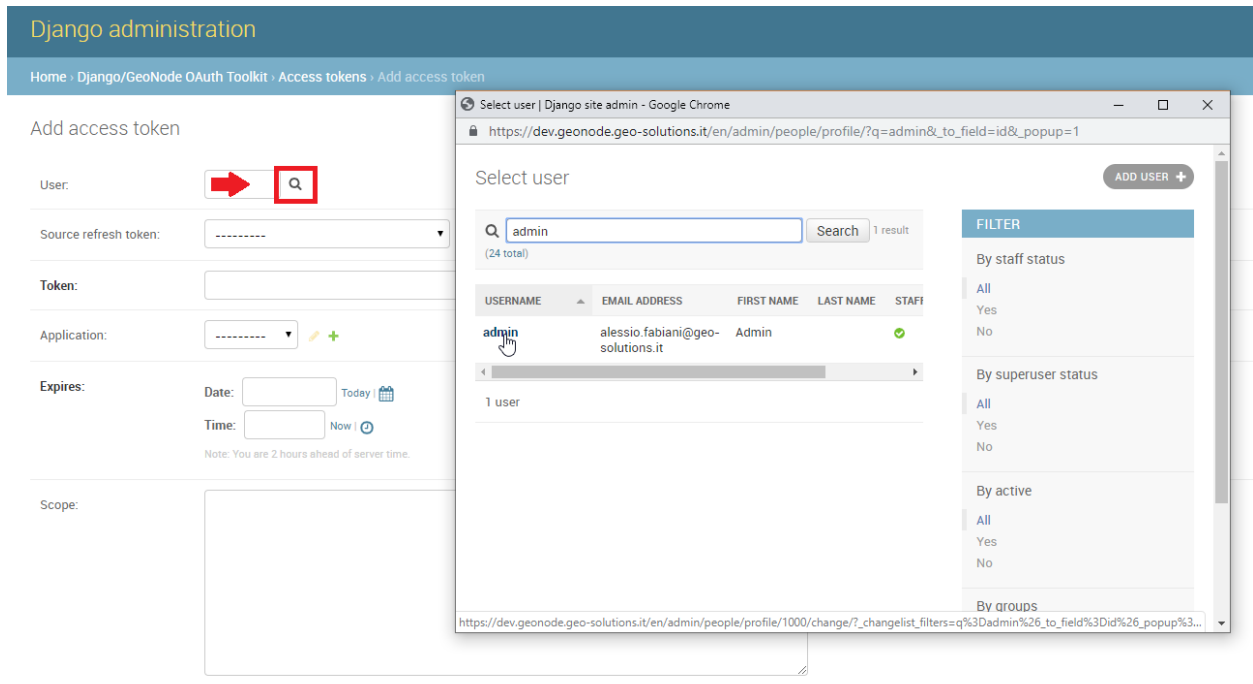
Date: Today 

Time: Now 

Note: You are 2 hours ahead of server time.

Scope:

Fig. 328: Create an ``Access token``

Fig. 329: *Select a User*

User:

Source refresh token:

Token:

Application:

Expires: Date: Today

Time: Now



Fig. 330: *Select a Token*

Add access token

User: 

Source refresh token:   

Token:

Application:   







Expires:   


Fig. 331: Select the GeoServer Application


Source refresh token:   

Token:

Application:  

Expires:

Date:  Today |

Time:  Now |

Note: You are 2 hours ahead of server time.


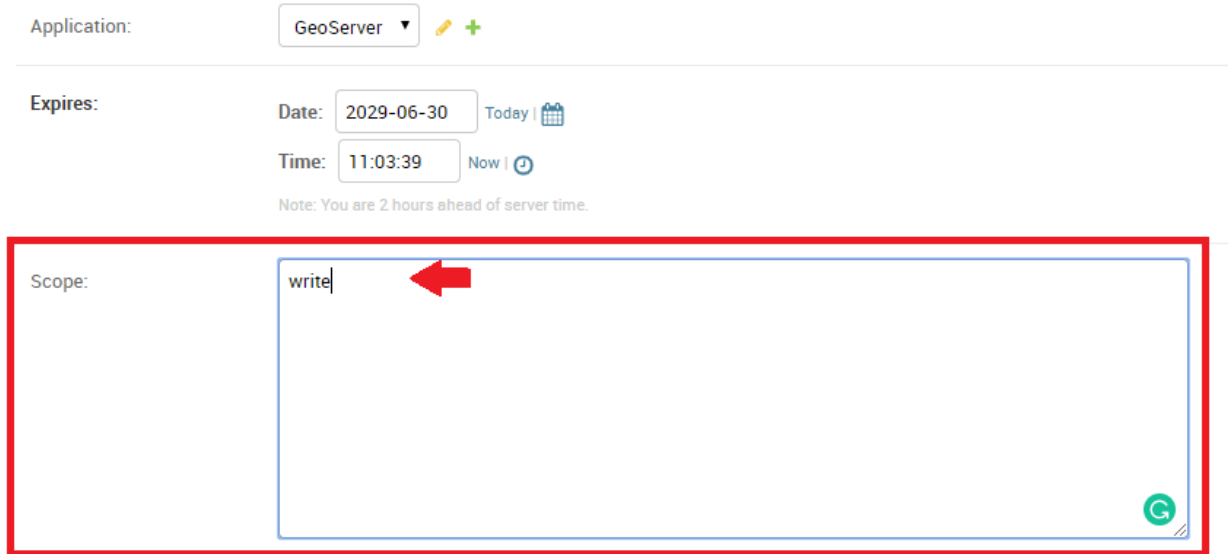






Fig. 332: Select the Token Expiration



Application: GeoServer  

Expires: Date: 2029-06-30 Today 
Time: 11:03:39 Now 
Note: You are 2 hours ahead of server time.


Scope: write 

Fig. 333: Select the Application Scope

Do not forget to *Save*.

From now on, GeoNode will use this `Access Token` to control the user session (notice that the user need to login again if closing the browser session), and the user will be able to access the OWS Services by using the new `Access Token`, e.g.:

```
https://dev.geonode.geo-solutions.it/geoserver/ows?service=wms&version=1.3.0&
↪request=GetCapabilities&access_token=123456
```

Notice the `...quest=GetCapabilities&access_token=123456` (**access_token**) parameter at the end of the URL.

Force a User Session to expire

Everything said about the creation of a new `Access Token`, applies to the deletion of the latter.

From the same interface an admin can either select an expiration date or delete all the `Access Tokens` associated to a user, in order to force its session to expire.

Remember that the user could activate another session by logging-in again on GeoNode with its credentials.

In order to be sure the user won't force GeoNode to refresh the token, reset first its password or de-activate it.

1.26 GeoNode Management Commands

1.26.1 Migrate GeoNode Base URL

The `migrate_baseurl Management Command` allows you to fix all the GeoNode Links whenever, for some reason, you need to change the *Domain Name* of *IP Address* of GeoNode.

This **must** be used also in the cases you'll need to change the network schema from HTTP to HTTPS, as an instance.

First of all let's take a look at the `--help` option of the `migrate_baseurl` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl --help
```

Note: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py migrate_baseurl --help
```

This will produce output that looks like the following

```
usage: manage.py migrate_baseurl [-h] [--version] [-v {0,1,2,3}]
                                [--settings SETTINGS]
                                [--pythonpath PYTHONPATH] [--traceback]
                                [--no-color] [-f]
                                [--source-address SOURCE_ADDRESS]
                                [--target-address TARGET_ADDRESS]

Migrate GeoNode VM Base URL

optional arguments:
-h, --help            show this help message and exit
--version            show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback          Raise on CommandError exceptions
--no-color           Don't colorize the command output.
-f, --force          Forces the execution without asking for confirmation.
--source-address SOURCE_ADDRESS
                        Source Address (the one currently on DB e.g.
                        http://192.168.1.23)
--target-address TARGET_ADDRESS
                        Target Address (the one to be changed e.g. http://my-
                        public.geonode.org)
```

- **Example 1:** I want to move my GeoNode instance from `http://127.0.0.1` to `http://example.org`

Warning: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl_
↪--source-address=127.0.0.1 --target-address=example.org
```

- **Example 2:** I want to move my GeoNode instance from `http://example.org` to `https://example.org`

Warning: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl_
↪--source-address=http:\\example.org --target-address=https:\\example.
↪org
```

- **Example 3:** I want to move my GeoNode instance from `https:\\example.org` to `https:\\geonode.example.org`

Warning: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl_
↪--source-address=example.org --target-address=geonode.example.org
```

Note: After migrating the base URL, make sure to sanitize the links and catalog metadata also (*Update Permissions, Metadata, Legends and Download Links*).

1.26.2 Update Permissions, Metadata, Legends and Download Links

The following three utility *Management Commands*, allow to fixup:

1. *Users/Groups Permissions on Layers*; those will be refreshed and synchronized with the *GIS Server* ones also
2. *Metadata, Legend and Download links on Layers and Maps*
3. Cleanup *Duplicated Links* and *Outdated Thumbnails*

Management Command `sync_geonode_layers`

This command allows to sync already existing permissions on Layers. In order to change/set Layers' permissions refer to the section *Batch Sync Permissions*

The options are:

- **filter**; Only update data the layer names that match the given filter.
- **username**; Only update data owned by the specified username.
- **updatepermissions**; Update the layer permissions; synchronize it back to the GeoSpatial Server. This option is also available from the *Layer Details* page.
- **updateattributes**; Update the layer attributes; synchronize it back to the GeoSpatial Server. This option is also available from the *Layer Details* page.
- **updatethumbnails**; Update the map styles and thumbnails. This option is also available from the *Layer Details* page.
- **remove-duplicates**; Removes duplicated Links.

First of all let's take a look at the `-help` option of the `sync_geonode_layers` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_layers --help
```

Note: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py sync_geonode_layers --
→help
```

This will produce output that looks like the following

```
usage: manage.py sync_geonode_layers [-h] [--version] [-v {0,1,2,3}]
                                     [--settings SETTINGS]
                                     [--pythonpath PYTHONPATH] [--traceback]
                                     [--no-color] [-i] [-d] [-f FILTER]
                                     [-u USERNAME] [--updatepermissions]
                                     [--updatethumbnails] [--updateattributes]

Update the GeoNode layers: permissions (including GeoFence database),
statistics, thumbnails

optional arguments:
-h, --help            show this help message and exit
--version             show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback           Raise on CommandError exceptions
--no-color            Don't colorize the command output.
-i, --ignore-errors  Stop after any errors are encountered.
-d, --remove-duplicates
                        Remove duplicates first.
-f FILTER, --filter FILTER
                        Only update data the layers that match the given
                        filter.
-u USERNAME, --username USERNAME
                        Only update data owned by the specified username.
--updatepermissions  Update the layer permissions.
--updatethumbnails   Update the layer styles and thumbnails.
--updateattributes    Update the layer attributes.
```

- **Example 1:** I want to update/sync all layers permissions and attributes with the GeoSpatial Server

Warning: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_
→layers --updatepermissions --updateattributes
```

- **Example 2:** I want to regenerate the Thumbnails of all the Layers belonging to afabiani

Warning: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_
↳layers -u afabiani --updatethumbnails
```

Management Command `sync_geonode_maps`

This command is basically similar to the previous one, but affects the *Maps*; with some limitations.

The options are:

- **filter**; Only update data the maps titles that match the given filter.
- **username**; Only update data owned by the specified username.
- **updatethumbnails**; Update the map styles and thumbnails. This option is also available from the *Map Details* page.
- **remove-duplicates**; Removes duplicated Links.

First of all let's take a look at the `-help` option of the `sync_geonode_maps` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_maps --help
```

Note: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py sync_geonode_maps --
↳help
```

This will produce output that looks like the following

```
usage: manage.py sync_geonode_maps [-h] [--version] [-v {0,1,2,3}]
                                   [--settings SETTINGS]
                                   [--pythonpath PYTHONPATH] [--traceback]
                                   [--no-color] [-i] [-d] [-f FILTER]
                                   [-u USERNAME] [--updatethumbnails]

Update the GeoNode maps: permissions, thumbnails

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
```

(continues on next page)

(continued from previous page)

```

"/home/djangoprojects/myproject".
--traceback          Raise on CommandError exceptions
--no-color           Don't colorize the command output.
-i, --ignore-errors  Stop after any errors are encountered.
-d, --remove-duplicates
                    Remove duplicates first.
-f FILTER, --filter FILTER
                    Only update data the maps that match the given filter.
-u USERNAME, --username USERNAME
                    Only update data owned by the specified username.
--updatethumbnails  Update the map styles and thumbnails.

```

- **Example 1:** I want to regenerate the Thumbnail of the Map This is a test Map

Warning: Make always sure you are using the **correct** settings

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_
↪maps --updatethumbnails -f 'This is a test Map'

```

Management Command `set_all_layers_metadata`

This command allows to reset **Metadata Attributes** and **Catalogue Schema** on Layers. The command will also update the *CSW Catalogue XML* and Links of GeoNode.

The options are:

- **filter;** Only update data the layers that match the given filter.
- **username;** Only update data owned by the specified username.
- **remove-duplicates;** Update the map styles and thumbnails.
- **delete-orphaned-thumbs;** Removes duplicated Links.

First of all let's take a look at the `-help` option of the `set_all_layers_metadata` management command in order to inspect all the command options and features.

Run

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py set_all_layers_metadata --
↪help

```

Note: If you enabled `local_settings.py` the command will change as following:

```

DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py set_all_layers_
↪metadata --help

```

This will produce output that looks like the following

```

usage: manage.py set_all_layers_metadata [-h] [--version] [-v {0,1,2,3}]
                                         [--settings SETTINGS]
                                         [--pythonpath PYTHONPATH]
                                         [--traceback] [--no-color] [-i] [-d]

```

(continues on next page)

(continued from previous page)

```

[-t] [-f FILTER] [-u USERNAME]

Resets Metadata Attributes and Schema to All Layers

optional arguments:
-h, --help            show this help message and exit
--version            show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback          Raise on CommandError exceptions
--no-color           Don't colorize the command output.
-i, --ignore-errors  Stop after any errors are encountered.
-d, --remove-duplicates
                        Remove duplicates first.
-t, --delete-orphaned-thumbnails
                        Delete Orphaned Thumbnails.
-f FILTER, --filter FILTER
                        Only update data the layers that match the given
                        filter
-u USERNAME, --username USERNAME
                        Only update data owned by the specified username

```

- **Example 1:** After having changed the Base URL, I want to regenerate all the Catalogue Schema and eventually remove all duplicates.

Warning: Make always sure you are using the **correct** settings

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py set_all_layers_
↪ metadata -d

```

1.26.3 Loading Data into GeoNode

There are situations where it is not possible or not convenient to use the *Upload Form* to add new Layers to GeoNode via the web interface. As an instance:

- The dataset is simply too big to be uploaded through a web interface.
- We would like to import some data from the mass storage programmatically.
- We would like to import some tables from a DataBase.
- We need to process the data first and, maybe, transform it to another format.

This section will walk you through the various options available to load data into your GeoNode from GeoServer, from the command-line or programmatically.

Warning: Some parts of this section have been taken from the [GeoServer](#) project and training documentation.

Management Command `importlayers`

The `geonode.layers` Django app includes 2 management commands that you can use to load or configure data in your GeoNode.

Both of them can be invoked by using the `manage.py` script.

First of all let's take a look at the `-help` option of the `importlayers` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py importlayers --help
```

Note: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py importlayers --help
```

This will produce output that looks like the following

```
usage: manage.py importlayers [-h] [--version] [-v {0,1,2,3}]
                             [--settings SETTINGS] [--pythonpath PYTHONPATH]
                             [--traceback] [--no-color] [-u USER] [-i] [-o]
                             [-k KEYWORDS] [-l LICENSE] [-c CATEGORY]
                             [-r REGIONS] [-n LAYERNAME] [-t TITLE]
                             [-a ABSTRACT] [-d DATE] [-p] [-m] [-C CHARSET]
                             [path [path ...]]

Brings a data file or a directory full of data files into a GeoNode site.
Layers are added to the Django database, the GeoServer configuration, and the
pycsw metadata index.

positional arguments:
path                    path [path...]

optional arguments:
-h, --help              show this help message and exit
--version               show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS    The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback             Raise on CommandError exceptions
--no-color              Don't colorize the command output.
-u USER, --user USER  Name of the user account which should own the imported
                        layers
```

(continues on next page)

(continued from previous page)

```

-i, --ignore-errors    Stop after any errors are encountered.
-o, --overwrite        Overwrite existing layers if discovered (defaults
                        False)
-k KEYWORDS, --keywords KEYWORDS
                        The default keywords, separated by comma, for the
                        imported layer(s). Will be the same for all imported
                        layers if multiple imports are done in one command
-l LICENSE, --license LICENSE
                        The license for the imported layer(s). Will be the
                        same for all imported layers if multiple imports are
                        done in one command
-c CATEGORY, --category CATEGORY
                        The category for the imported layer(s). Will be the
                        same for all imported layers if multiple imports are
                        done in one command
-r REGIONS, --regions REGIONS
                        The default regions, separated by comma, for the
                        imported layer(s). Will be the same for all imported
                        layers if multiple imports are done in one command
-n LAYERNAME, --name LAYERNAME
                        The name for the imported layer(s). Can not be used
                        with multiple imports
-t TITLE, --title TITLE
                        The title for the imported layer(s). Will be the same
                        for all imported layers if multiple imports are done
                        in one command
-a ABSTRACT, --abstract ABSTRACT
                        The abstract for the imported layer(s). Will be the
                        same for all imported layers if multiple imports are
                        done in one command
-d DATE, --date DATE   The date and time for the imported layer(s). Will be
                        the same for all imported layers if multiple imports
                        are done in one command. Use quotes to specify both
                        the date and time in the format 'YYYY-MM-DD HH:MM:SS'.
-p, --private          Make layer viewable only to owner
-m, --metadata_uploaded_preserve
                        Force metadata XML to be preserved
-C CHARSET, --charset CHARSET
                        Specify the charset of the data

```

While the description of most of the options should be self explanatory, its worth reviewing some of the key options a bit more in details.

- The *-i* option will force the command to stop when it first encounters an error. Without this option specified, the process will skip over errors that have layers and continue loading the other layers.
- The *-o* option specifies that layers with the same name as the base name will be loaded and overwrite the existing layer.
- The *-u* option specifies which will be the user that owns the imported layers. The same user will be the point of contact and the metadata author as well for that layer
- The *-k* option is used to add keywords for all of the layers imported.
- The *-C* option specifies the character encoding of the data.

The import layers management command is invoked by specifying options as described above and specifying the path to a single layer file or to a directory that contains multiple files. For purposes of this exercise, let's use the default set of testing layers that ship with geonode. You can replace this path with the directory to your own shapefiles.

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py importlayers -v 3 /var/lib/  
↳geonode/lib/python2.7/site-packages/gisdata/data/good/vector/
```

This command will produce the following output to your terminal

```
Verifying that GeoNode is running ...  
Found 8 potential layers.  
No handlers could be found for logger "pycsw"  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_administrative.shp' (1/8)  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_coastline.shp' (2/8)  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_highway.shp' (3/8)  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_location.shp' (4/8)  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_natural.shp' (5/8)  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_poi.shp' (6/8)  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_water.shp' (7/8)  
[created] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/single_point.shp' (8/8)  
  
Detailed report of failures:  
  
Finished processing 8 layers in 30.0 seconds.  
  
8 Created layers  
0 Updated layers  
0 Skipped layers  
0 Failed layers  
3.750000 seconds per layer
```

If you encounter errors while running this command, you can use the `-v` option to increase the verbosity of the output so you can debug the problem.

The verbosity level can be set from 0-3 with 0 being the default.

An example of what the output looks like when an error is encountered and the verbosity is set to 3 is shown below:

```
Verifying that GeoNode is running ...  
Found 8 potential layers.  
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_administrative.shp' (1/8)  
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_coastline.shp' (2/8)  
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_highway.shp' (3/8)  
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_location.shp' (4/8)  
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_natural.shp' (5/8)  
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_poi.shp' (6/8)  
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/  
↳gisdata/data/good/vector/san_andres_y_providencia_water.shp' (7/8)
```

(continues on next page)

(continued from previous page)

```
[failed] Layer for '/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/
↳gisdata/data/good/vector/single_point.shp' (8/8)

Detailed report of failures:

/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/
↳vector/san_andres_y_providencia_administrative.shp
=====
Traceback (most recent call last):
  File "/Users/geosolutions/projects/geonode/geonode/layers/utils.py", line 682, in _
↳upload
    keywords=keywords,
  File "/Users/geosolutions/projects/geonode/geonode/layers/utils.py", line 602, in _
↳file_upload
    keywords=keywords, title=title)
  File "/Users/geosolutions/projects/geonode/geonode/layers/utils.py", line 305, in _
↳save
    store = cat.get_store(name)
  File "/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/geoserver/
↳catalog.py", line 176, in get_store
    for ws in self.get_workspaces():
  File "/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/geoserver/
↳catalog.py", line 489, in get_workspaces
    description = self.get_xml("%s/workspaces.xml" % self.service_url)
  File "/Users/geosolutions/.venvs/geonode/lib/python2.7/site-packages/geoserver/
↳catalog.py", line 136, in get_xml
    response, content = self.http.request(rest_url)
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1445, in request
    (response, content) = self._request(conn, authority, uri, request_uri, method, _
↳body, headers, redirections, cachekey)
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1197, in _
↳request
    (response, content) = self._conn_request(conn, request_uri, method, body, headers)
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1133, in _conn_
↳request
    conn.connect()
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 799, in connect
    raise socket.error, msg
error: [Errno 61] Connection refused
```

Note: This last section of output will be repeated for all layers, and only the first one is show above.

This error indicates that GeoNode was unable to connect to GeoServer to load the layers. To solve this, you should make sure GeoServer is running and re-run the command.

If you encounter errors with this command that you cannot solve, you should bring them up on the geonode users mailing list.

You should now have the knowledge necessary to import layers into your GeoNode project from a directory on the servers filesystem and can use this to load many layers into your GeoNode at once.

Note: If you do not use the `-u` command option, the ownership of the imported layers will be assigned to the primary superuser in your system. You can use GeoNodes Django Admin interface to modify this after the fact if you want them to be owned by another user.

Management Command `updateLayers`

While it is possible to import layers directly from your servers filesystem into your GeoNode, you may have an existing GeoServer that already has data in it, or you may want to configure data from a GeoServer which is not directly supported by uploading data.

GeoServer supports a wide range of data formats and connections to database, and while many of them are not supported as GeoNode upload formats, if they can be configured in GeoServer, you can add them to your GeoNode by following the procedure described below.

GeoServer supports 3 types of data: *Raster*, *Vector*, *Databases* and *Cascaded*.

For a list of the supported formats for each type of data, consult the following pages:

- <https://docs.geoserver.org/latest/en/user/data/vector/index.html>
- <https://docs.geoserver.org/latest/en/user/data/raster/index.html>
- <https://docs.geoserver.org/latest/en/user/data/database/index.html>
- <https://docs.geoserver.org/latest/en/user/data/cascaded/index.html>

Note: Some of these raster or vector formats or database types require that you install specific plugins in your GeoServer in order to use the. Please consult the GeoServer documentation for more information.

Data from a PostGIS database

Lets walk through an example of configuring a new PostGIS database in GeoServer and then configuring those layers in your GeoNode.

First visit the GeoServer administration interface on your server. This is usually on port 8080 and is available at <http://localhost:8080/geoserver/web/>

1. You should login with the superuser credentials you setup when you first configured your GeoNode instance.

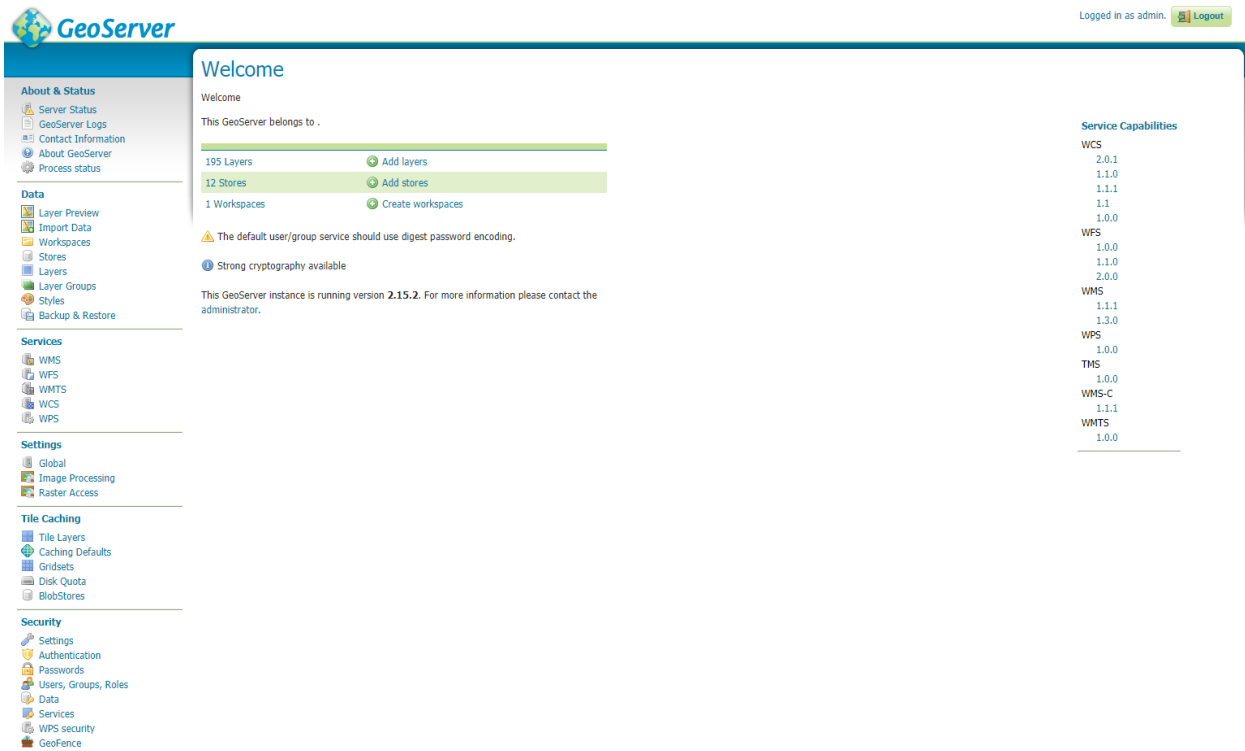
Once you are logged in to the GeoServer Admin interface, you should see the following.

Note: The number of stores, layers and workspaces may be different depending on what you already have configured in your GeoServer.

2. Next you want to select the “Stores” option in the left hand menu, and then the “Add new Store” option. The following screen will be displayed.
3. In this case, we want to select the PostGIS store type to create a connection to our existing database. On the next screen you will need to enter the parameters to connect to your PostGIS database (alter as necessary for your own database).

Note: If you are unsure about any of the settings, leave them as the default.

4. The next screen lets you configure the layers in your database. This will of course be different depending on the layers in your database.



GeoServer

Logged in as admin. [Logout](#)

Welcome

Welcome

This GeoServer belongs to .

195 Layers	Add layers
12 Stores	Add stores
1 Workspaces	Create workspaces

⚠ The default user/group service should use digest password encoding.

🔒 Strong cryptography available

This GeoServer instance is running version **2.15.2**. For more information please contact the administrator.

Service Capabilities

WCS	2.0.1
	1.1.0
	1.1.1
	1.1
	1.0.0
WFS	1.0.0
	1.1.0
	2.0.0
WMS	1.1.1
	1.3.0
WPS	1.0.0
	1.0.0
TMS	1.0.0
WMS-C	1.1.1
WMTS	1.0.0

About & Status

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer
- Process status

Data

- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles
- Backup & Restore

Services

- WMS
- WFS
- WMTS
- WCS
- WPS

Settings

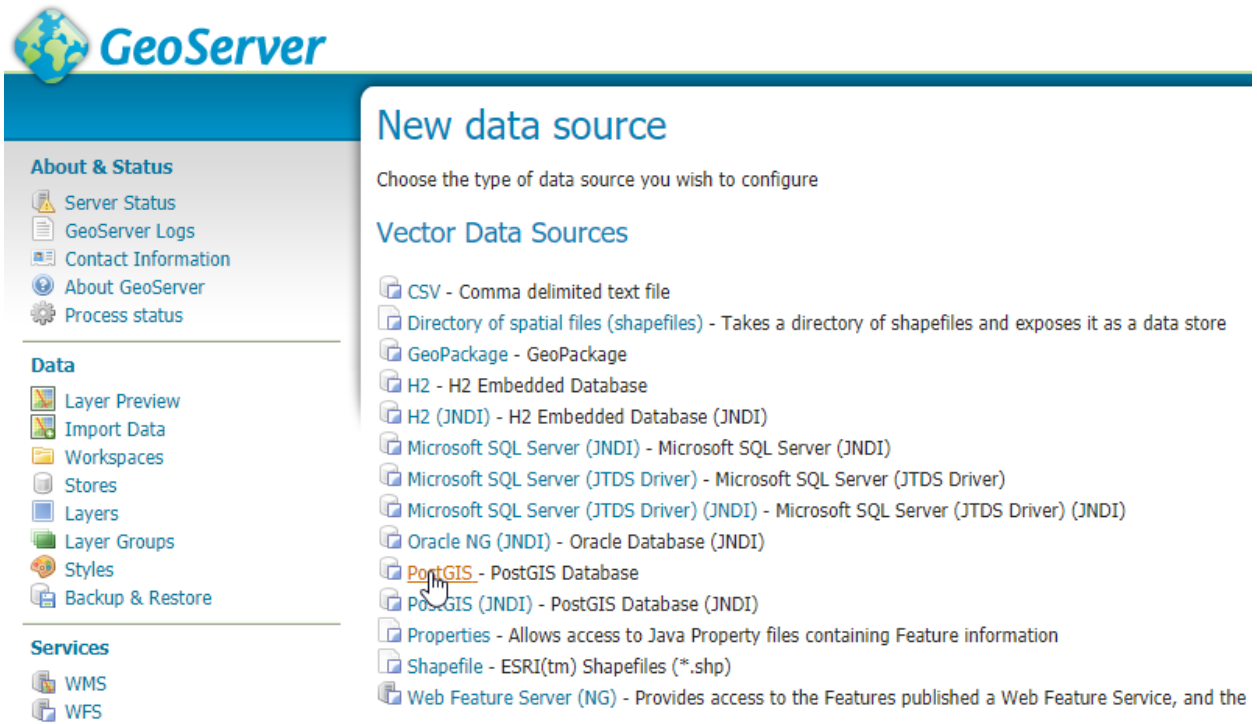
- Global
- Image Processing
- Raster Access

Tile Caching

- Tile Layers
- Caching Defaults
- Gridsets
- Disk Quota
- BlobStores

Security

- Settings
- Authentication
- Passwords
- Users, Groups, Roles
- Data
- Services
- WFS security
- GeoFence




GeoServer

New data source

Choose the type of data source you wish to configure

Vector Data Sources

- CSV - Comma delimited text file
- Directory of spatial files (shapefiles) - Takes a directory of shapefiles and exposes it as a data store
- GeoPackage - GeoPackage
- H2 - H2 Embedded Database
- H2 (JNDI) - H2 Embedded Database (JNDI)
- Microsoft SQL Server (JNDI) - Microsoft SQL Server (JNDI)
- Microsoft SQL Server (JTDS Driver) - Microsoft SQL Server (JTDS Driver)
- Microsoft SQL Server (JTDS Driver) (JNDI) - Microsoft SQL Server (JTDS Driver) (JNDI)
- Oracle NG (JNDI) - Oracle Database (JNDI)
- PostGIS** - PostGIS Database
- PostGIS (JNDI) - PostGIS Database (JNDI)
- Properties - Allows access to Java Property files containing Feature information
- Shapefile - ESRI(tm) Shapefiles (*.shp)
- Web Feature Server (NG) - Provides access to the Features published a Web Feature Service, and the

 **GeoServer**

About & Status

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer
- Process status

Data

- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles
- Backup & Restore

Services

- WMS
- WFS
- WMTS
- WCS
- WPS

Settings

- Global
- Image Processing
- Raster Access

Tile Caching

- Tile Layers
- Caching Defaults

New Vector Data Source

Add a new vector data source

PostGIS
PostGIS Database

Basic Store Info

Workspace *

geonode ▼

Data Source Name *

my_db_data

Description

☐ Enabled

Connection Parameters

host *

localhost

port *

5432

database

geonode_data

schema

public

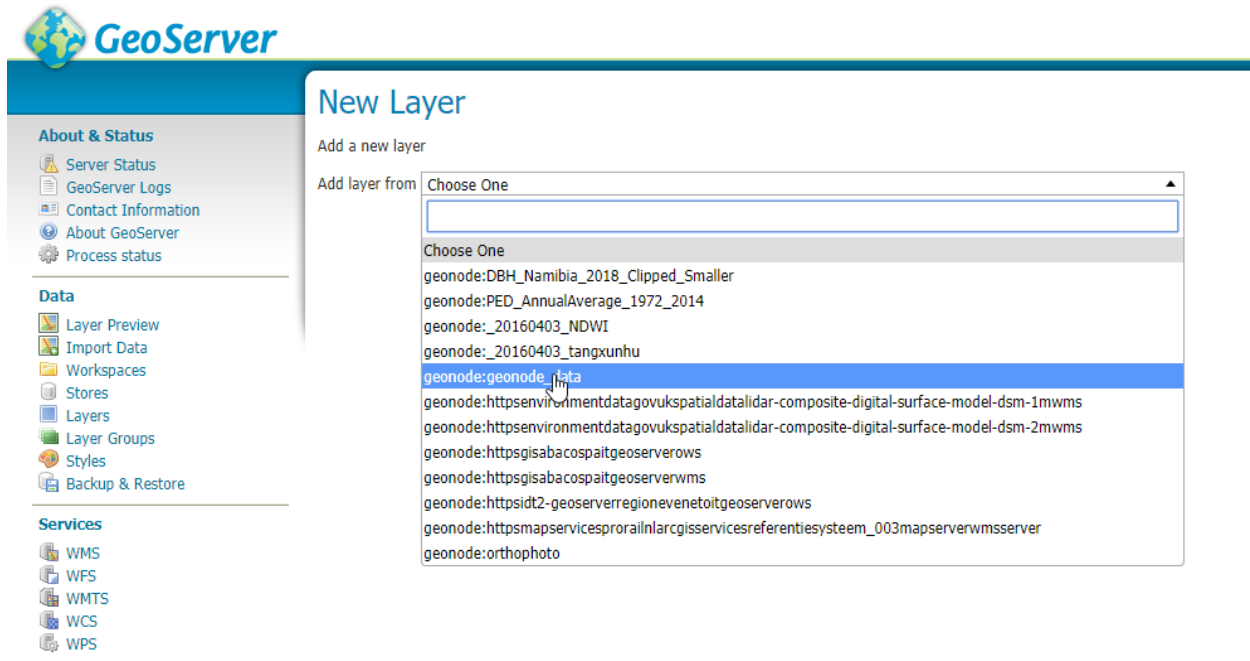
user *

geonode

passwd

Namespace *

http://www.geonode.org/



5. Select the “Publish” button for one of the layers and the next screen will be displayed where you can enter metadata for this layer. Since we will be managing this metadata in GeoNode, we can leave these alone for now.
6. The things that *must* be specified are the Declared SRS and you must select the “Compute from Data” and “Compute from native bounds” links after the SRS is specified.
7. Click save and this layer will now be configured for use in your GeoServer.
8. The next step is to configure these layers in GeoNode. The `updatelayers` management command can be used for this purpose. As with `importlayers`, it’s useful to look at the command line options for this command by passing the `-help` option

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py updatelayers --help
```

Note: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py updatelayers --help
```

This will produce output that looks like the following

```
usage: manage.py updatelayers [-h] [--version] [-v {0,1,2,3}]
                             [--settings SETTINGS] [--pythonpath PYTHONPATH]
                             [--log-level LOG_LEVEL] [--quiet]
                             [--help]
```

(continues on next page)

New Layer

Add a new layer

Add layer from

You can create a new feature type by manually configuring the attribute names and types. [Create new feature type...](#)

On databases you can also create a new feature type by configuring a native SQL statement. [Configure new SQL view...](#)

Here is a list of resources contained in the store 'geonode_data'. Click on the layer you wish to configure

Published	Layer name	Action
<input checked="" type="checkbox"/>	testspnrc	Publish again
<input checked="" type="checkbox"/>	trial_7thsept_29Records	Publish again
<input checked="" type="checkbox"/>	trial_7thsept_30Records	Publish again
<input checked="" type="checkbox"/>	ua_test	Publish again
<input checked="" type="checkbox"/>	verger	Publish again
<input checked="" type="checkbox"/>	verger0	Publish again
<input checked="" type="checkbox"/>	voirie_gen	Publish again
<input checked="" type="checkbox"/>	waterways	Publish again
<input checked="" type="checkbox"/>	waterways0	Publish again
<input type="checkbox"/>	_1_SARMIENTO_ENERO_2018	Publish
<input type="checkbox"/>	cambodia_oilfields	Publish
<input type="checkbox"/>	kagaa	Publish
<input type="checkbox"/>	map4utm2	Publish
<input type="checkbox"/>	study001_ins00_Sept6_Buffer	Publish
<input type="checkbox"/>	study_as_geojson_7thSep	Publish
<input type="checkbox"/>	study_as_logitude_latitude	Publish
<input type="checkbox"/>	study_as_logitude_latitude0	Publish
<input type="checkbox"/>	study_as_logitude_latitude1	Publish
<input type="checkbox"/>	study_as_logitude_latitude2	Publish

[Image Processing](#)
[Raster Access](#)

Tile Caching

[Tile Layers](#)
[Caching Defaults](#)
[Gridsets](#)
[Disk Quota](#)
[BlobStores](#)

Security

[Settings](#)
[Authentication](#)
[Passwords](#)
[Users, Groups, Roles](#)
[Data](#)
[Services](#)
[WPS security](#)
[GeoFence](#)
[GeoFence Data Rules](#)
[GeoFence Admin Rules](#)

Monitor

[Activity](#)
[Reports](#)

Demos

Tools

Keywords

Current Keywords

[Remove selected](#)

New Keyword

Vocabulary

[Add Keyword](#)

Metadata links

No metadata links so far

[Add link](#) *Note only FGDC and TC211 metadata links show up in WMS 1.1.1 capabilities*

Data links

No data links so far

[Add link](#)

Coordinate Reference Systems

Native SRS

[EPSG:WGS 84...](#)

Declared SRS

[Find...](#) [EPSG:WGS 84...](#)

SRS handling

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
-69.078485273	-45.60003889	-69.057928671999	-45.579602872

[Compute from data](#)

[Compute from bounds](#)

Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

[Compute from native bounds](#)

Curved geometries control

(continued from previous page)

```

                                [--traceback] [--no-color] [-i]
                                [--skip-unadvertised]
                                [--skip-geonode-registered] [--remove-
→deleted]
                                [-u USER] [-f FILTER] [-s STORE] [-w
→WORKSPACE]
                                [-p PERMISSIONS]

Update the GeoNode application with data from GeoServer

optional arguments:
-h, --help            show this help message and exit
--version            show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal
→output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't
→provided, the
                        DJANGO_SETTINGS_MODULE environment variable will
→be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback          Raise on CommandError exceptions
--no-color           Don't colorize the command output.
-i, --ignore-errors  Stop after any errors are encountered.
--skip-unadvertised  Skip processing unadvertised layers from GeoSever.
--skip-geonode-registered
                        Just processing GeoServer layers still not
→registered
                        in GeoNode.
--remove-deleted     Remove GeoNode layers that have been deleted from
                        GeoSever.
-u USER, --user USER
→imported            Name of the user account which should own the
                        layers
-f FILTER, --filter FILTER
                        Only update data the layers that match the given
                        filter
-s STORE, --store STORE
→geoserver           Only update data the layers for the given
                        store name
-w WORKSPACE, --workspace WORKSPACE
                        Only update data on specified workspace
-p PERMISSIONS, --permissions PERMISSIONS
                        Permissions to apply to each layer

```

Warning: One of the `--workspace` or `--store` must be always specified if you want to ingest Layers belonging to a specific Workspace. As an instance, in order to ingest the layers present into the `geonode` workspace, you will need to specify the option `-w geonode`.

9. Let's ingest the layer `geonode:_1_SARMIENTO_ENERO_2018` from the geonode workspace.

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py updatelayers -w_
↳geonode -f _1_SARMIENTO_ENERO_2018
```

```
Inspecting the available layers in GeoServer ...
Found 1 layers, starting processing
/usr/local/lib/python2.7/site-packages/owslib/iso.py:117: FutureWarning:
↳the .identification and .serviceidentification properties will merge
↳into .identification being a list of properties. This is currently
↳implemented in .identificationinfo. Please see https://github.com/
↳geopython/OWSLib/issues/38 for more information
FutureWarning)
/usr/local/lib/python2.7/site-packages/owslib/iso.py:495: FutureWarning:
↳The .keywords and .keywords2 properties will merge into the .keywords
↳property in the future, with .keywords becoming a list of MD_Keywords
↳instances. This is currently implemented in .keywords2. Please see
↳https://github.com/geopython/OWSLib/issues/301 for more information
FutureWarning)
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A new layer has been uploaded
From: webmaster@localhost
To: mapadeldelito@chubut.gov.ar
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:17 -0000
Message-ID: <20191008122617.28801.94967@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The user <i><a href="http://master.demo.geonode.org/people/profile/admin
↳">admin</a></i> uploaded the following layer:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↳org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↳geonode.org
</p>
</body>

-----
↳-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A new layer has been uploaded
From: webmaster@localhost
To: giacomovinci@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:17 -0000
Message-ID: <20191008122617.28801.53784@d3cf85425231>
```

(continues on next page)

(continued from previous page)

```

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The user <i><a href="http://master.demo.geonode.org/people/profile/admin
↪">admin</a></i> uploaded the following layer:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↪org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↪geonode.org
</p>
</body>

```

```

-----
↪-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A new layer has been uploaded
From: webmaster@localhost
To: fmgagliano@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:17 -0000
Message-ID: <20191008122617.28801.26265@d3cf85425231>

```

```

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The user <i><a href="http://master.demo.geonode.org/people/profile/admin
↪">admin</a></i> uploaded the following layer:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↪org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↪geonode.org
</p>
</body>

```

```

-----
↪-----
Found geoserver resource for this layer: _1_SARMIENTO_ENERO_2018
... Creating Default Resource Links for Layer [geonode:_1_SARMIENTO_
↪ENERO_2018]
-- Resource Links[Prune old links]...
-- Resource Links[Prune old links]...done!
-- Resource Links[Compute parameters for the new links]...

```

(continues on next page)

(continued from previous page)

```

-- Resource Links[Create Raw Data download link]...
-- Resource Links[Create Raw Data download link]...done!
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...done!
-- Resource Links[Legend link]...
-- Resource Links[Legend link]...done!
-- Resource Links[Thumbnail link]...
-- Resource Links[Thumbnail link]...done!
-- Resource Links[OWS Links]...
-- Resource Links[OWS Links]...done!
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A layer has been updated
From: webmaster@localhost
To: mapadeldelito@chubut.gov.ar
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:20 -0000
Message-ID: <20191008122620.28801.81598@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The following layer was updated:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong>, owned by <i><a href="http://
↪master.demo.geonode.org/people/profile/admin">admin</a></i><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↪org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↪geonode.org
</p>
</body>

-----
↪-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A layer has been updated
From: webmaster@localhost
To: giacomio8vinci@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:20 -0000
Message-ID: <20191008122620.28801.93778@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The following layer was updated:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong>, owned by <i><a href="http://
↪master.demo.geonode.org/people/profile/admin">admin</a></i><br/>

```

(continues on next page)

(continued from previous page)

```

You can visit the layer's detail page here: http://master.demo.geonode.
↳org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↳geonode.org
</p>
</body>

-----
↳-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A layer has been updated
From: webmaster@localhost
To: fmgagliano@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:20 -0000
Message-ID: <20191008122620.28801.58585@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The following layer was updated:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong>, owned by <i><a href="http://
↳master.demo.geonode.org/people/profile/admin">admin</a></i><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↳org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↳geonode.org
</p>
</body>

-----
↳-----
Found geoserver resource for this layer: _1_SARMIENTO_ENERO_2018
/usr/local/lib/python2.7/site-packages/geoserver/style.py:80:
↳FutureWarning: The behavior of this method will change in future_
↳versions. Use specific 'len(elem)' or 'elem is not None' test instead.
if not user_style:
/usr/local/lib/python2.7/site-packages/geoserver/style.py:84:
↳FutureWarning: The behavior of this method will change in future_
↳versions. Use specific 'len(elem)' or 'elem is not None' test instead.
if user_style:
... Creating Default Resource Links for Layer [geonode:_1_SARMIENTO_
↳ENERO_2018]
-- Resource Links[Prune old links]...
-- Resource Links[Prune old links]...done!
-- Resource Links[Compute parameters for the new links]...
-- Resource Links[Create Raw Data download link]...

```

(continues on next page)

(continued from previous page)

```
-- Resource Links[Create Raw Data download link]...done!
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...done!
-- Resource Links[Legend link]...
-- Resource Links[Legend link]...done!
-- Resource Links[Thumbnail link]...
-- Resource Links[Thumbnail link]...done!
-- Resource Links[OWS Links]...
-- Resource Links[OWS Links]...done!
[created] Layer _1_SARMIENTO_ENERO_2018 (1/1)

Finished processing 1 layers in 5.0 seconds.

1 Created layers
0 Updated layers
0 Failed layers
5.000000 seconds per layer
```

Note: In case you don't specify the `-f` option, the layers that already exist in your GeoNode will be just updated and the configuration synchronized between GeoServer and GeoNode.

Warning: When updating from GeoServer, the configuration on GeoNode will be changed!

Using GDAL and OGR to convert your Data for use in GeoNode

GeoNode supports uploading data in *ESRI shapefiles*, *GeoTIFF*, *CSV*, *GeoJSON*, *ASCII-GRID* and *KML / KMZ* formats (for the last three formats only if you are using the `geonode.importer` backend).

- If your data is in other formats, you will need to convert it into one of these formats for use in GeoNode.
- If your *Raster* data is not correctly processed, it might be almost unusable with GeoServer and GeoNode. You will need to process it using *GDAL*.

You need to make sure that you have the GDAL library installed on your system. On Ubuntu you can install this package with the following command:

```
sudo apt-get install gdal-bin
```

OGR (Vector Data)

OGR is used to manipulate vector data. In this example, we will use MapInfo `.tab` files and convert them to shapefiles with the `ogr2ogr` command. We will use sample MapInfo files from the website linked below.

<http://services.land.vic.gov.au/landchannel/content/help?name=sampleddata>

You can download the Admin;(Postcode) layer by issuing the following command:

```
$ wget http://services.land.vic.gov.au/sampleddata/shape/admin_postcode_vm.zip
```

You will need to unzip this dataset by issuing the following command:

```
$ unzip admin_postcode_vm.zip
```

This will leave you with the following files in the directory where you executed the above commands:

```
|-- ANZVI0803003025.htm
|-- DSE_Data_Access_Licence.pdf
|-- VMADMIN.POSTCODE_POLYGON.xml
|-- admin_postcode_vm.zip
--- vicgrid94
    --- mif
        --- lga_polygon
            --- macedon\ ranges
                |-- EXTRACT_POLYGON.mid
                |-- EXTRACT_POLYGON.mif
            --- VMADMIN
                |-- POSTCODE_POLYGON.mid
                --- POSTCODE_POLYGON.mif
```

First, lets inspect this file set using the following command:

```
$ ogrinfo -so vicgrid94/mif/lga_polygon/macedon\ ranges/VMADMIN/POSTCODE_POLYGON.mid_
↳ POSTCODE_POLYGON
```

The output will look like the following:

```
Had to open data source read-only.
INFO: Open of `vicgrid94/mif/lga_polygon/macedon ranges/VMADMIN/POSTCODE_POLYGON.mid'
      using driver `MapInfo File' successful.

Layer name: POSTCODE_POLYGON
Geometry: 3D Unknown (any)
Feature Count: 26
Extent: (2413931.249367, 2400162.366186) - (2508952.174431, 2512183.046927)
Layer SRS WKT:
PROJCS["unnamed",
  GEOGCS["unnamed",
    DATUM["GDA94",
      SPHEROID["GRS 80",6378137,298.257222101],
      TOWGS84[0,0,0,-0,-0,-0,0]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]],
  PROJECTION["Lambert_Conformal_Conic_2SP"],
  PARAMETER["standard_parallel_1",-36],
  PARAMETER["standard_parallel_2",-38],
  PARAMETER["latitude_of_origin",-37],
  PARAMETER["central_meridian",145],
  PARAMETER["false_easting",2500000],
  PARAMETER["false_northing",2500000],
  UNIT["Meter",1]]
PFI: String (10.0)
POSTCODE: String (4.0)
FEATURE_TYPE: String (6.0)
FEATURE_QUALITY_ID: String (20.0)
PFI_CREATED: Date (10.0)
UFI: Real (12.0)
UFI_CREATED: Date (10.0)
UFI_OLD: Real (12.0)
```


This gives you information about the number of features, the extent, the projection and the attributes of this layer.

Next, lets go ahead and convert this layer into a shapefile by issuing the following command:

```
$ ogr2ogr -t_srs EPSG:4326 postcode_polygon.shp vicgrid94/mif/lga_polygon/macedon\_  
→ranges/VMADMIN/POSTCODE_POLYGON.mid POSTCODE_POLYGON
```

Note that we have also reprojected the layer to the WGS84 spatial reference system with the `-t_srs ogr2ogr` option.

The output of this command will look like the following:

```
Warning 6: Normalized/laundered field name: 'FEATURE_TYPE' to 'FEATURE_TY'  
Warning 6: Normalized/laundered field name: 'FEATURE_QUALITY_ID' to 'FEATURE_QU'  
Warning 6: Normalized/laundered field name: 'PFI_CREATED' to 'PFI_CREATE'  
Warning 6: Normalized/laundered field name: 'UFI_CREATED' to 'UFI_CREATE'
```

This output indicates that some of the field names were truncated to fit into the constraint that attributes in shapefiles are only 10 characters long.

You will now have a set of files that make up the `postcode_polygon.shp` shapefile set. We can inspect them by issuing the following command:

```
$ ogrinfo -so postcode_polygon.shp postcode_polygon
```

The output will look similar to the output we saw above when we inspected the MapInfo file we converted from:

```
INFO: Open of `postcode_polygon.shp'  
      using driver `ESRI Shapefile' successful.  
  
Layer name: postcode_polygon  
Geometry: Polygon  
Feature Count: 26  
Extent: (144.030296, -37.898156) - (145.101137, -36.888878)  
Layer SRS WKT:  
GEOGCS["GCS_WGS_1984",  
  DATUM["WGS_1984",  
    SPHEROID["WGS_84",6378137,298.257223563]],  
  PRIMEM["Greenwich",0],  
  UNIT["Degree",0.017453292519943295]]  
PFI: String (10.0)  
POSTCODE: String (4.0)  
FEATURE_TY: String (6.0)  
FEATURE_QU: String (20.0)  
PFI_CREATE: Date (10.0)  
UFI: Real (12.0)  
UFI_CREATE: Date (10.0)  
UFI_OLD: Real (12.0)
```

These files can now be loaded into your GeoNode instance via the normal uploader.

Visit the upload page in your GeoNode, drag and drop the files that composes the shapefile that you have generated using the GDAL `ogr2ogr` command (`postcode_polygon.dbf`, `postcode_polygon.prj`, `postcode_polygon.shp`, `postcode_polygon.shx`). Give the permissions as needed and then click the “Upload files” button.

As soon as the import process completes, you will have the possibility to go straight to the layer info page (“Layer Info” button), or to edit the metadata for that layer (“Edit Metadata” button), or to manage the styles for that layer (“Manage Styles”).

admin

[HOME](#)
[LAYERS](#)
[MAPS](#)
[DOCUMENTS](#)
[PEOPLE](#)
[SEARCH](#)

UPLOAD LAYERS

Drop files here

or select them one by one:

No file chosen

FILES TO BE UPLOADED

POSTCODE_POLYGON

ESRI SHAPEFILE

- [postcode_polygon.dbf](#) Remove
- [postcode_polygon.prj](#) Remove
- [postcode_polygon.shp](#) Remove
- [postcode_polygon.shx](#) Remove

Select the charset or leave default

PERMISSIONS

Who can view and download this data?

☒ Anyone
 ☐ Any registered user
 ☐ Only users who can edit

Who can edit this data?

☒ Any registered user
 ☐ Only the following users or groups:

Who can manage and edit this data?

Powered by GeoNode version 2.0.dev20130911150113 | Developers | About

Language English

admin

[HOME](#)
[LAYERS](#)
[MAPS](#)
[DOCUMENTS](#)
[PEOPLE](#)
[SEARCH](#)

POSTCODE_POLYGON

Title: postcode_polygon

MAPS USING THIS LAYER

This layer is not currently used in any maps.

GDAL (Raster Data)

Let's see several examples on how to either convert raster data into different formats and/or process it to get the best performances.

References:

- a) https://geoserver.geo-solutions.it/edu/en/raster_data/processing.html
- b) https://geoserver.geo-solutions.it/edu/en/raster_data/advanced_gdal/

Raster Data Conversion: Arc/Info Binary and ASCII Grid data into GeoTIFF format.

Let's assume we have a sample ASCII Grid file compressed as an archive.

```
# Un-tar the files
tar -xvf sample_asc.tar
```

You will be left with the following files on your filesystem:

```
|-- batemans_ele
|   |-- dblbnd.adf
|   |-- hdr.adf
|   |-- metadata.xml
|   |-- prj.adf
|   |-- sta.adf
|   |-- w001001.adf
|   |-- w001001x.adf
|-- batemans_elevation.asc
```

The file `batemans_elevation.asc` is an Arc/Info ASCII Grid file and the files in the `batemans_ele` directory are an Arc/Info Binary Grid file.

You can use the `gdalinfo` command to inspect both of these files by executing the following command:

```
gdalinfo batemans_elevation.asc
```

The output should look like the following:

```
Driver: AAIGrid/Arc/Info ASCII Grid
Files: batemans_elevation.asc
Size is 155, 142
Coordinate System is ``
Origin = (239681.0000000000000000,6050551.0000000000000000)
Pixel Size = (100.0000000000000000,-100.0000000000000000)
Corner Coordinates:
Upper Left  ( 239681.000, 6050551.000)
Lower Left  ( 239681.000, 6036351.000)
Upper Right ( 255181.000, 6050551.000)
Lower Right ( 255181.000, 6036351.000)
Center      ( 247431.000, 6043451.000)
Band 1 Block=155x1 Type=Float32, ColorInterp=Undefined
      NoData Value=-9999
```

You can then inspect the `batemans_ele` files by executing the following command:

```
gdalinfo batemans_ele
```

And this should be the corresponding output:

```
Driver: AIG/Arc/Info Binary Grid
Files: batemans_ele
       batemans_ele/dblbnf.adf
       batemans_ele/hdr.adf
       batemans_ele/metadata.xml
       batemans_ele/prj.adf
       batemans_ele/sta.adf
       batemans_ele/w001001.adf
       batemans_ele/w001001x.adf
Size is 155, 142
Coordinate System is:
PROJCS["unnamed",
  GEOGCS["GDA94",
    DATUM["Geocentric_Datum_of_Australia_1994",
      SPHEROID["GRS 1980",6378137,298.257222101,
        AUTHORITY["EPSG","7019"]],
      TOWGS84[0,0,0,0,0,0,0],
      AUTHORITY["EPSG","6283"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4283"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",153],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",1000000],
  UNIT["METERS",1]]
Origin = (239681.0000000000000000,6050551.0000000000000000)
Pixel Size = (100.0000000000000000,-100.0000000000000000)
Corner Coordinates:
Upper Left  ( 239681.000, 6050551.000) (150d 7'28.35"E, 35d39'16.56"S)
Lower Left  ( 239681.000, 6036351.000) (150d 7'11.78"E, 35d46'56.89"S)
Upper Right ( 255181.000, 6050551.000) (150d17'44.07"E, 35d39'30.83"S)
Lower Right ( 255181.000, 6036351.000) (150d17'28.49"E, 35d47'11.23"S)
Center      ( 247431.000, 6043451.000) (150d12'28.17"E, 35d43'13.99"S)
Band 1 Block=256x4 Type=Float32, ColorInterp=Undefined
      Min=-62.102 Max=142.917
NoData Value=-3.4028234663852886e+38
```

You will notice that the `batemans_elevation.asc` file does *not* contain projection information while the `batemans_ele` file does. Because of this, let's use the `batemans_ele` files for this exercise and convert them to a GeoTiff for use in GeoNode. We will also reproject this file into WGS84 in the process. This can be accomplished with the following command.

```
gdalwarp -t_srs EPSG:4326 batemans_ele batemans_ele.tif
```

The output will show you the progress of the conversion and when it is complete, you will be left with a `batemans_ele.tif` file that you can upload to your GeoNode.

You can inspect this file with the `gdalinfo` command:

```
gdalinfo batemans_ele.tif
```

Which will produce the following output:

```

Driver: GTiff/GeoTIFF
Files: batemans_ele.tif
Size is 174, 130
Coordinate System is:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4326"]]
Origin = (150.119938943722502,-35.654598806259330)
Pixel Size = (0.0010111114155919,-0.0010111114155919)
Metadata:
    AREA_OR_POINT=Area
Image Structure Metadata:
    INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 150.1199389, -35.6545988) (150d 7'11.78"E, 35d39'16.56"S)
Lower Left  ( 150.1199389, -35.7860436) (150d 7'11.78"E, 35d47' 9.76"S)
Upper Right ( 150.2958728, -35.6545988) (150d17'45.14"E, 35d39'16.56"S)
Lower Right ( 150.2958728, -35.7860436) (150d17'45.14"E, 35d47' 9.76"S)
Center      ( 150.2079059, -35.7203212) (150d12'28.46"E, 35d43'13.16"S)
Band 1 Block=174x11 Type=Float32, ColorInterp=Gray

```

Raster Data Optimization: Optimizing and serving big raster data

(ref: https://geoserver.geo-solutions.it/edu/en/raster_data/advanced_gdal/example5.html)

When dealing with big raster datasets it could be very useful to use tiles.

Tiling allows large raster datasets to be broken-up into manageable pieces and are fundamental in defining and implementing a higher level raster I/O interface.

In this example we will use the original dataset of the `chiangMai_ortho_optimized` public raster layer which is currently available on the Thai [CHIANG MAI Urban Flooding GeoNode platform](#).

This dataset contains an orthorectified image stored as RGBa GeoTiff with 4 bands, three bands for the RGB and one for transparency (the alpha channel).

Calling the `gdalinfo` command to see detailed information:

```
gdalinfo chiangMai_ortho.tif
```

It will produce the following results:

```

Driver: GTiff/GeoTIFF
Files: chiangMai_ortho.tif
Size is 63203, 66211
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],

```

(continues on next page)

(continued from previous page)

```

    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",99],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32647"]]]
Origin = (487068.7747500000040513,2057413.8898100000080615)
Pixel Size = (0.028850000000000,-0.028850000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=LZW
INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.702) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.181, 2057413.890) ( 98d53'40.94"E, 18d36'27.38"N)
Lower Right ( 488892.181, 2055503.702) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.478, 2056458.796) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=63203x1 Type=Byte, ColorInterp=Red
NoData Value=-10000
Mask Flags: PER_DATASET ALPHA
Band 2 Block=63203x1 Type=Byte, ColorInterp=Green
NoData Value=-10000
Mask Flags: PER_DATASET ALPHA
Band 3 Block=63203x1 Type=Byte, ColorInterp=Blue
NoData Value=-10000
Mask Flags: PER_DATASET ALPHA
Band 4 Block=63203x1 Type=Byte, ColorInterp=Alpha
NoData Value=-10000

```

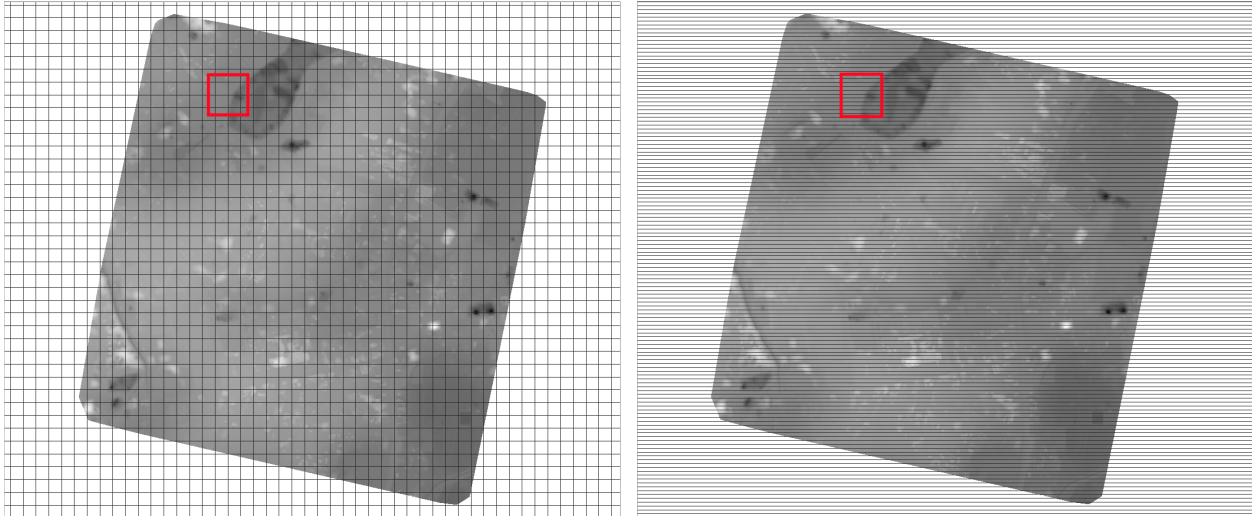
As you can see, this GeoTiff has not been tiled. For accessing subsets though, tiling can make a difference. With tiling, data are stored and compressed in blocks (tiled) rather than line by line (stripped).

In the command output above it is visible that each band has blocks with the same width of the image (63203) and a unit length. The grids in the picture below show an image with equally sized tiles (left) and the same number of strips (right). To read data from the red subset, the intersected area will have to be decompressed.

In the tiled image we will have to decompress only 16 tiles, whereas in the stripped image on the right we'll have to decompress many more strips.

Drone images data usually have a stripped structure so, in most cases, they need to be optimized to increase performances.

Let's take a look at the `gdal_translate` command used to optimize our GeoTiff:



```
gdal_translate -co TILED=YES -co COMPRESS=JPEG -co PHOTOMETRIC=YCBCR
--config GDAL_TIFF_INTERNAL_MASK YES -b 1 -b 2 -b 3 -mask 4
chiangMai_ortho.tif
chiangMai_ortho_optimized.tif
```

Note: For the details about the command parameters see https://geoserver.geo-solutions.it/edu/en/raster_data/advanced_gdal/example5.html

Once the process ended, call the `gdalinfo` command on the resulting tif file:

```
gdalinfo chiangMai_ortho_optimized.tif
```

The following should be the results:

```
Driver: GTiff/GeoTIFF
Files: chiangMai_ortho_optimized.tif
Size is 63203, 66211
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",99],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
```

(continues on next page)

(continued from previous page)

```

    AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32647"]]]
Origin = (487068.7747500000040513,2057413.8898100000080615)
Pixel Size = (0.028850000000000,-0.028850000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=YCbCr JPEG
INTERLEAVE=PIXEL
SOURCE_COLOR_SPACE=YCbCr
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.702) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.181, 2057413.890) ( 98d53'40.94"E, 18d36'27.38"N)
Lower Right ( 488892.181, 2055503.702) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.478, 2056458.796) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
NoData Value=-10000
Mask Flags: PER_DATASET
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
NoData Value=-10000
Mask Flags: PER_DATASET
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
NoData Value=-10000
Mask Flags: PER_DATASET

```

Our GeoTiff is now tiled with 256x256 tiles, has 3 bands and a 1-bit mask for nodata.

We can also add internal overviews to the file using the `gdaladdo` command:

```
gdaladdo -r average chiangMai_ortho_optimized.tif 2 4 8 16 32 64 128 256 512
```

Overviews are duplicate versions of your original data, but resampled to a lower resolution, they can also be compressed with various algorithms, much in the same way as the original dataset.

By default, overviews take the same compression type and transparency masks of the input dataset (applied through the `gdal_translate` command), so the parameters to be specified are:

- `-r average`: computes the average of all non-NODATA contributing pixels
- `2 4 8 16 32 64 128 256 512`: the list of integral overview levels to build (from gdal version 2.3 levels are no longer required to build overviews)

Calling the `gdalinfo` command again:

```
gdalinfo chiangMai_ortho_optimized.tif
```

It results in:

```

Driver: GTiff/GeoTIFF
Files: chiangMai_ortho_optimized.tif
Size is 63203, 66211
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
    GEOGCS["WGS 84",

```

(continues on next page)

(continued from previous page)

```

    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",99],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32647"]]]
Origin = (487068.774750000040513,2057413.889810000080615)
Pixel Size = (0.028850000000000,-0.028850000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=YCbCr JPEG
INTERLEAVE=PIXEL
SOURCE_COLOR_SPACE=YCbCr
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.702) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.181, 2057413.890) ( 98d53'40.94"E, 18d36'27.38"N)
Lower Right ( 488892.181, 2055503.702) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.478, 2056458.796) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
NoData Value=-10000
Overviews: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, 988x1035, ↵
↵494x518, 247x259, 124x130
Mask Flags: PER_DATASET
Overviews of mask band: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, ↵
↵988x1035, 494x518, 247x259, 124x130
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
NoData Value=-10000
Overviews: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, 988x1035, ↵
↵494x518, 247x259, 124x130
Mask Flags: PER_DATASET
Overviews of mask band: 31602x3Results in:3106, 15801x16553, 7901x8277, 3951x4139, ↵
↵1976x2070, 988x1035, 494x518, 247x259, 124x130
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
NoData Value=-10000
Overviews: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, 988x1035, ↵
↵494x518, 247x259, 124x130
Mask Flags: PER_DATASET
Overviews of mask band: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, ↵
↵988x1035, 494x518, 247x259, 124x130

```

Notice that the transparency masks of internal overviews have been applied (their compression does not show up in the file metadata).

UAVs usually provide also two other types of data: DTM (Digital Terrain Model) and DSM (Digital Surface Model).

Those data require different processes to be optimized. Let's look at some examples to better understand how to use `gdal` to accomplish that task.

From the [CHIANG MAI Urban Flooding GeoNode platform](#) platform it is currently available the `chiangMai_dtm_optimized` layer, let's download its original dataset.

This dataset should contain the DTM file `chiangMai_dtm.tif`.

Calling the `gdalinfo` command on it:

```
gdalinfo chiangMai_dtm.tif
```

The following information will be displayed:

```
Driver: GTiff/GeoTIFF
Files: chiangMai_dtm.tif
Size is 12638, 13240
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",99],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AXIS["Easting",EAST],
  AXIS["Northing",NORTH],
  AUTHORITY["EPSG","32647"]]
Origin = (487068.774750000040513,2057413.889810000080615)
Pixel Size = (0.144270000000000,-0.144270000000000)
Metadata:
  AREA_OR_POINT=Area
  TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
  COMPRESSION=LZW
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.755) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.059, 2057413.890) ( 98d53'40.94"E, 18d36'27.37"N)
Lower Right ( 488892.059, 2055503.755) ( 98d53'40.98"E, 18d35'25.22"N)
```

(continues on next page)

(continued from previous page)

```
Center      ( 487980.417, 2056458.822) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=12638x1 Type=Float32, ColorInterp=Gray
NoData Value=-10000
```

Reading this image could be very slow because it has not been tiled yet. So, as discussed above, its data need to be stored and compressed in tiles to increase performances.

The following `gdal_translate` command should be appropriate for that purpose:

```
gdal_translate -co TILED=YES -co COMPRESS=DEFLATE chiangMai_dtm.tif chiangMai_dtm_
↳ optimized.tif
```

When the data to compress consists of imagery (es. aerial photographs, true-color satellite images, or colored maps) you can use lossy algorithms such as JPEG. We are now compressing data where the precision is important, the band data type is Float32 and elevation values should not be altered, so a lossy algorithm such as JPEG is not suitable. JPEG should generally only be used with Byte data (8 bit per channel) so we have chosen the lossless DEFLATE compression through the `COMPRESS=DEFLATE` creation option.

Calling the `gdalinfo` command again:

```
gdalinfo chiangMai_dtm_optimized.tif
```

We can observe the following results:

```
Driver: GTiff/GeoTIFF
Files: chiangMai_dtm_optimized.tif
Size is 12638, 13240
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",99],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AXIS["Easting",EAST],
  AXIS["Northing",NORTH],
  AUTHORITY["EPSG","32647"]]
Origin = (487068.7747500000040513,2057413.8898100000080615)
Pixel Size = (0.144270000000000,-0.144270000000000)
Metadata:
  AREA_OR_POINT=Area
  TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
  COMPRESSION=DEFLATE
```

(continues on next page)

(continued from previous page)

```

INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.755) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.059, 2057413.890) ( 98d53'40.94"E, 18d36'27.37"N)
Lower Right ( 488892.059, 2055503.755) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.417, 2056458.822) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Float32, ColorInterp=Gray
NoData Value=-10000

```

We need also to create overviews through the `gdaladdo` command:

```
gdaladdo -r nearest chiangMai_dtm_optimized.tif 2 4 8 16 32 64
```

Unlike the previous example, overviews will be created with the **nearest resampling algorithm**. That is due to the nature of the data we are representing: we should not consider the average between two elevation values but simply the closer one, it is more reliable regarding the conservation of the original data.

Calling the `gdalinfo` command again:

```
gdalinfo chiangMai_dtm_optimized.tif
```

We can see the following information:

```

Driver: GTiff/GeoTIFF
Files: chiangMai_dtm_optimized.tif
Size is 12638, 13240
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",99],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32647"]]
Origin = (487068.774750000040513,2057413.889810000080615)
Pixel Size = (0.144270000000000,-0.144270000000000)
Metadata:
  AREA_OR_POINT=Area
  TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
  COMPRESSION=DEFLATE

```

(continues on next page)

(continued from previous page)

```

INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.755) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.059, 2057413.890) ( 98d53'40.94"E, 18d36'27.37"N)
Lower Right ( 488892.059, 2055503.755) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.417, 2056458.822) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Float32, ColorInterp=Gray
NoData Value=-10000
Overviews: 6319x6620, 3160x3310, 1580x1655, 790x828, 395x414, 198x207

```

Overviews have been created. By default, they inherit the same compression type of the original dataset (there is no evidence of it in the gdalinfo output).

Other Raster Data Use Cases

- Serving a large number of GrayScale GeoTiff with Palette
- Serving a large number of DTM ASCII Grid Files
- Serving a large number of Cartographic Black/White GeoTiff with Palette
- Serving a large number of satellite/aerial RGB GeoTiff with compression
- Optimizing and serving UAV data
- Optimizing and serving 16-bits satellite/aerial RGB GeoTiff

Process Raster Datasets Programmatically

In this section we will provide a set of *shell* scripts which might be very useful to batch process a lot of raster datasets programmatically.

1. process_gray.sh

```

for filename in *.tif*; do echo gdal_translate -co TILED=YES -co
→COMPRESS=DEFLATE $filename ${filename//.tif/.optimized.tif}; done >
→gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh

```

```

for filename in *.optimized.tif*; do echo gdaladdo -r nearest $filename_
→2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv "${filename}" "${filename//
→.optimized.tif/.tif}\""; done > rename.sh
chmod +x *.sh
./gdaladdo.sh
./rename.sh

```

2. process_rgb.sh

```

for filename in *.tif*; do echo gdal_translate -co TILED=YES -co
→COMPRESS=JPEG -co PHOTOMETRIC=YCBCR -b 1 -b 2 -b 3 $filename $
→{filename//.tif/.optimized.tif}; done > gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh

```

```
for filename in *.optimized.tif*; do echo gdaladdo -r average $filename_
↪2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv \"$filename\" \"${filename//
↪.optimized.tif/.tif}\"; done > rename.sh
chmod +x *.sh
./gdaladdo.sh
./rename.sh
```

3. process_rgb_alpha.sh

```
for filename in *.tif*; do echo gdal_translate -co TILED=YES -co_
↪COMPRESS=JPEG -co PHOTOMETRIC=YCBCR --config GDAL_TIFF_INTERNAL_MASK_
↪YES -b 1 -b 2 -b 3 -mask 4 $filename ${filename//.tif/.optimized.tif};_
↪done > gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh
```

```
for filename in *.optimized.tif*; do echo gdaladdo -r average $filename_
↪2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv \"$filename\" \"${filename//
↪.optimized.tif/.tif}\"; done > rename.sh
chmod +x *.sh
./gdaladdo.sh
./rename.sh
```

4. process_rgb_palette.sh

```
for filename in *.tif*; do echo gdal_translate -co TILED=YES -co_
↪COMPRESS=DEFLATE $filename ${filename//.tif/.optimized.tif}; done >_
↪gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh
```

```
for filename in *.optimized.tif*; do echo gdaladdo -r average $filename_
↪2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv \"$filename\" \"${filename//
↪.optimized.tif/.tif}\"; done > rename.sh
chmod +x *.sh
./gdaladdo.sh
./rename.sh
```

1.26.4 Create Users and Super Users

Your first step will be to create a user. There are three options to do so, depending on which kind of user you want to create you may choose a different option. We will start with creating a *superuser*, because this user is the most important. A superuser has all the permissions without explicitly assigning them.

The easiest way to create a superuser (in linux) is to open your terminal and type:

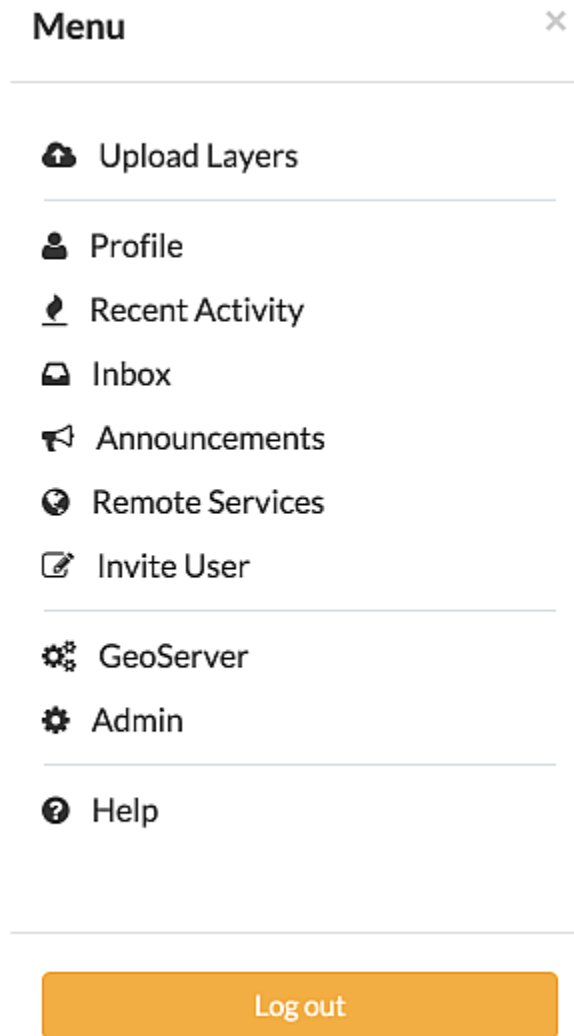
```
$ DJANGO_SETTINGS_MODULE=geonode.settings python manage.py createsuperuser
```

Note: If you enabled `local_settings.py` the command will change as following:

```
$ DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py_
→createsuperuser
```

You will be asked a username (in this tutorial we will call the superuser you now create *your_superuser*), an email address and a password.

Now you've created a superuser you should become familiar with the *Django Admin Interface*. As a superuser you are having access to this interface, where you can manage users, layers, permission and more. To learn more detailed about this interface check this [LINK](#). For now it will be enough to just follow the steps. To attend the *Django Admin Interface*, go to your geonode website and *sign in* with *your_superuser*. Once you've logged in, the name of your user will appear on the top right. Click on it and the following menu will show up:





























Clicking on *Admin* causes the interface to show up.

Go to *Auth -> Users* and you will see all the users that exist at the moment. In your case it will only be *your_superuser*. Click on it, and you will see a section on *Personal Info*, one on *Permissions* and one on *Important dates*. For the

Django administration

Site administration

Account		
Account deletions	 Add	 Change
Accounts	 Add	 Change
Signup codes	 Add	 Change
Actstream		
Actions	 Add	 Change
Follows	 Add	 Change
Announcements		
Announcements	 Add	 Change
Dismissals	 Add	 Change
Auth		
Groups	 Add	 Change
Users	 Add	 Change
Avatar		
Avatars	 Add	 Change
Base		
Contact roles	 Add	 Change
Links	 Add	 Change
Metadata Regions	 Add	 Change

moment, the section on *Permissions* is the most important.

Permissions	
<input checked="" type="checkbox"/> Active	Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
<input checked="" type="checkbox"/> Staff status	Designates whether the user can log into this admin site.
<input checked="" type="checkbox"/> Superuser status	Designates that this user has all permissions without explicitly assigning them.

As you can see, there are three boxes that can be checked and unchecked. Because you've created a superuser, all three boxes are checked as default. If only the box *active* would have been checked, the user would not be a superuser and would not be able to access the *Django Admin Interface* (which is only available for users with the *staff* status). Therefore keep the following two things in mind:

- a superuser is able to access the *Django Admin Interface* and he has all permissions on the data uploaded to GeoNode.
- an ordinary user (created from the GeoNode interface) only has *active* permissions by default. The user will not have the ability to access the *Django Admin Interface* and certain permissions have to be added for him.

Until now we've only created superusers. So how do you create an ordinary user? You have two options:

1. Django Admin Interface

First we will create a user via the *Django Admin Interface* because we've still got it open. Therefore go back to *Auth -> Users* and you should find a button on the right that says *Add user*.

Now you should be directed to the site where you could change the permissions on the user *test_user*. As default only *active* is checked. If you want this user also to be able to attend this admin interface you could also check *staff status*. But for now we leave the settings as they are!

To test whether the new user was successfully created, go back to the GeoNode web page and try to sign in.

2. GeoNode website

To create an ordinary user you could also just use the GeoNode website. If you installed GeoNode using a release, you should see a *Register* button on the top, beside the *Sign in* button (you might have to log out before).



Hit the button and again a form will appear for you to fill out. This user will be named *geonode_user*

SIGN UP

Username

Password

Password (again)

Email

[Sign up](#)

By hitting *Sign up* the user will be signed up, as default only with the status *active*.

1.26.5 Batch Sync Permissions

GeoNode provides a very useful management command `set_layers_permissions` allowing an administrator to easily add / remove permissions to groups and users on one or more layers.

The `set_layers_permissions` command arguments are:

- **permissions** to set/unset → read (r), write (w), download (d), owner (o)

```
READ_PERMISSIONS = [  
    'view_resourcebase'  
]  
WRITE_PERMISSIONS = [  
    'change_layer_data',  
    'change_layer_style',  
    'change_resourcebase_metadata'  
]  
DOWNLOAD_PERMISSIONS = [  
    'download_resourcebase'  
]  
OWNER_PERMISSIONS = [  
    'change_resourcebase',  
    'delete_resourcebase',  
    'change_resourcebase_permissions',
```

(continues on next page)

(continued from previous page)

```
'publish_resourcebase'
]
```

- **resources** (layers) which permissions will be assigned on → type the layer title (use quotation mark for titles with white space), multiple choices can be typed with white space separator, if no titles are provided all the layers will be considered
- **users** who permissions will be assigned to, multiple choices can be typed with a white space separator
- **groups** who permissions will be assigned to, multiple choices can be typed with a white space separator
- **delete** flag (optional) which means the permissions will be unset

Usage examples:

1. Assign **write** permissions on the layers **layer_X** and **layer Y** to the users **user_A** and **user_B** and to the group **group_C**.

```
python manage.py set_layers-permissions -p write -u user_A user_B -g
→group_C -r layer_X 'layer Y'
```

2. Assign **owner** permissions on all the layers to the group **group_C**.

```
python manage.py set_layers-permissions -p owner -g group_C
```

3. Unset **download** permissions on the layer **layer_X** for the user **user_A**.

```
python manage.py set_layers-permissions -p download -u user_A -r layer_X
→-d
```

The same functionalities, with some limitations, are available also from the *Admin Dashboard >> Layers*.

Django administration

Home › Layers › Layers

Select layer to change

< 2019 November 14

Action: Set Layers Permissions ▼ 2 of 2 selected

<input checked="" type="checkbox"/>	ID	ALTERNATE	TITLE [EN]	DATE	CATE
<input checked="" type="checkbox"/>	28	geonode:tasmania_water_bodies	<input type="text" value="tasmania_water_bodies"/>	Nov. 14, 2019, 3:36 p.m.	---
<input checked="" type="checkbox"/>	27	geonode:tasmania_state_boundaries	<input type="text" value="tasmania_state_boundaries"/>	Nov. 14, 2019, 3:36 p.m.	---

An action named *Set layers permissions* is available from the list, redirecting the administrator to a form to set / unset read, write, download and ownership permissions on the selected layers.



1.26.6 Delete Certain GeoNode Resources

The `delete_resources` *Management Command* allows to remove resources meeting a certain condition, specified in a form of a serialized django Q() expression.

First of all let's take a look at the `--help` option of the `delete_resources` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_resources --help
```

Note: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py delete_resources --help
```

This will produce output the following output:

```
usage: manage.py delete_resources [-h] [-c CONFIG_PATH]
                                  [-l LAYER_FILTERS [LAYER_FILTERS ...]]
                                  [-m MAP_FILTERS [MAP_FILTERS ...]]
                                  [-d DOCUMENT_FILTERS [DOCUMENT_FILTERS ...]]
                                  [--version] [-v {0,1,2,3}]
                                  [--settings SETTINGS]
```

(continues on next page)

(continued from previous page)

```

                                [--pythonpath PYTHONPATH] [--traceback]
                                [--no-color] [--force-color]

Delete resources meeting a certain condition

optional arguments:
  -h, --help                show this help message and exit
  -c CONFIG_PATH, --config CONFIG_PATH
                            Configuration file path. Default is:
                            delete_resources.json
  -l LAYER_FILTERS [LAYER_FILTERS ...], --layer_filters LAYER_FILTERS [LAYER_FILTERS .
  ↪ ..]
  -m MAP_FILTERS [MAP_FILTERS ...], --map_filters MAP_FILTERS [MAP_FILTERS ...]
  -d DOCUMENT_FILTERS [DOCUMENT_FILTERS ...], --document_filters DOCUMENT_FILTERS_
  ↪ [DOCUMENT_FILTERS ...]
  --version                show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                            Verbosity level; 0=minimal output, 1=normal output,
                            2=verbose output, 3=very verbose output
  --settings SETTINGS      The Python path to a settings module, e.g.
                            "myproject.settings.main". If this isn't provided, the
                            DJANGO_SETTINGS_MODULE environment variable will be
                            used.
  --pythonpath PYTHONPATH
                            A directory to add to the Python path, e.g.
                            "/home/djangoprojects/myproject".
  --traceback              Raise on CommandError exceptions
  --no-color               Don't colorize the command output.
  --force-color            Force colorization of the command output.

```

There are two ways to declare Q() expressions filtering which resources should be deleted:

1. With a JSON configuration file: passing `-c` argument specifying the path to the JSON configuration file.

- **Example 1:** Relative path to the config file (to `manage.py`)

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
↪resources -c geonode/base/management/commands/delete_resources.json

```

- **Example 2:** Absolute path to the config file

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
↪resources -c /home/User/Geonode/configs/delete_resources.json

```

2. With CLI: passing `-l` `-d` `-m` list arguments for each of resources (layers, documents, maps)

- **Example 3:** Delete resources without configuration file

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
↪resources -l 'Q(pk__in: [1, 2]) | Q(title__icontains:"italy")'
↪'Q(owner__name=admin)' -d '*' -m "Q(pk__in=[1, 2])"

```

Configuration File

The JSON configuration file should contain a single *filters* object, which consists of *layer*, *map* and *document* lists. Each list specifies the filter conditions applied to a corresponding queryset, defining which items will be deleted. The filters are evaluated and directly inserted into Django `.filter()` method, which means the filters occurring as separated list items are treated as AND condition. To create OR query `|` operator should be used. For more info please check Django [documentation](<https://docs.djangoproject.com/en/3.0/topics/db/queries/#complex-lookups-with-q-objects>)). The only exception is passing a list with `'*'` which will cause deleting all the queryset of the resource.

- **Example 4:** Example content of the configuration file, which will delete layers with ID's 1, 2, and 3, those owned by *admin* user, along with all defined maps.

```
{
  "filters": {
    "layer": [
      "Q(pk__in=[1, 2, 3]) | Q(title__icontains='italy')",
      "Q(user__name=admin)"
    ],
    "map": ["*"],
    "document": []
  }
}
```

CLI

The CLI configuration can be specified with `-l -d -m` list arguments, which in fact are a translation of the configuration JSON file. `-l -d -m` arguments are evaluated in the same manner as `filters.layer`, `filters.map` and `filter.document` accordingly from the Example 4. The following example's result will be equivalent to Example 4:

- **Example 5:** Example CLI configuration, which will delete layers with ID's 1, 2, and 3, along with all maps.

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
→resources -l 'Q(pk__in: [1, 2, 3]) | Q(title__icontains:"italy")'
→'Q(owner__name=admin)' -m '*'
```

1.27 Changing the default Languages

1.27.1 Changing the Default Language

GeoNode's default language is English, but GeoNode users can change the interface language with the pulldown menu at the top-right of most GeoNode pages. Once a user selects a language GeoNode remembers that language for subsequent pages.

1.27.2 GeoNode Configuration

As root edit the geonode config file `/home/geonode/geonode/geonode/settings.py` (or `/etc/geonode/settings.py` if GeoNode has been installed using **apt-get**) and change `LANGUAGE_CODE` to the desired default language.

Note: A list of language codes can be found in the global django config file `/usr/local/lib/python2.7/dist-packages/django/conf/global_settings.py` (or `/var/lib/geonode/lib/python2.7/site-packages/django/conf/global_settings.py` if GeoNode has been installed using **apt-get**).

For example, to make French the default language use:

```
LANGUAGE_CODE = 'fr'
```

Unfortunately Django overrides this setting, giving the language setting of a user's browser priority. For example, if `LANGUAGE_CODE` is set to French, but the user has configured their operating system for Spanish they may see the Spanish version when they first visit GeoNode.

1.27.3 Additional Steps

If this is not the desired behaviour, and all users should initially see the default `LANGUAGE_CODE`, regardless of their browser's settings, do the following steps to ensure Django ignores the browser language settings. (Users can always use the pulldown language menu to change the language at any time.)

As **root** create a new directory within GeoNode's site packages

```
mkdir /usr/lib/python2.7/dist-packages/setmydefaultlanguage
```

or

```
mkdir /var/lib/geonode/lib/python2.7/site-packages/setmydefaultlanguage
```

if GeoNode has been installed using **apt-get**.

As root create and edit a new file `/usr/lib/python2.7/dist-packages/setmydefaultlanguage/__init__.py` and add the following lines

```
class ForceDefaultLanguageMiddleware(object):
    """
    Ignore Accept-Language HTTP headers

    This will force the I18N machinery to always choose settings.LANGUAGE_CODE
    as the default initial language, unless another one is set via sessions or cookies

    Should be installed *before* any middleware that checks request.META['HTTP_ACCEPT_
    →LANGUAGE'],
    namely django.middleware.locale.LocaleMiddleware
    """
    def process_request(self, request):
        if request.META.has_key('HTTP_ACCEPT_LANGUAGE'):
            del request.META['HTTP_ACCEPT_LANGUAGE']
```

At the end of the GeoNode configuration file `/home/geonode/geonode/geonode/settings.py` (or `/etc/geonode/settings.py` if GeoNode has been installed using **apt-get**) add the following lines to ensure the above class is executed

```
MIDDLEWARE_CLASSES += (  
    'setmydefaultlanguage.ForceDefaultLanguageMiddleware',  
)
```

1.27.4 Restart

You will need to restart GeoNode accordingly to the installation method you have choosen.

As an instance in case you are using *NGINX* with *UWSGI*, as root you will need to run the following commands

```
service uwsgi restart  
service nginx restart
```

Please refer to Translating GeoNode for information on editing GeoNode pages in different languages and create new GeoNode Translations.

1.28 GeoNode Upgrade from older versions

1.28.1 Upgrade from 2.8.x

1.28.2 Upgrade from 2.7.x

1.28.3 Upgrade from 2.6.x

1.28.4 Upgrade from 2.4.x

These are the notes of a migration from 2.4.x to 2.10.1. These notes could possibly work also when migrating from 2.6.x, 2.7.x, 2.8.x but are not tested in that scenarios. You should run this procedure on your local machine and once you successfully migrated the database move the backup to your GeoNode 2.10.1 production instance.

PostgreSQL

Create a role and a database for Django GeoNode 2.4:

```
create role user with superuser login with password '***';  
create database gn_24 with owner user;  
\c gn_24  
create extension postgis;
```

Restore backup from your production backup:

```
psql gn_24 < gn_24.sql
```


Run GeoNode migrations

Activate your GeoNode virtualenv and set the env vars:

```
. env/bin/Activate
export vars_210
```

Here are the variables to export - update them to your environment settings:

```
export DATABASE_URL=postgis://user:***@localhost:5432/dbname
export DEFAULT_BACKEND_DATASTORE=data
export GEODATABASE_URL=postgis://user:***@localhost:5432/geonode_data
export ALLOWED_HOSTS="['localhost', '192.168.100.10']"
export STATIC_ROOT=~/.www/geonode/static/
export GEOSERVER_LOCATION=http://localhost:8080/geoserver/
export GEOSERVER_PUBLIC_LOCATION=http://localhost:8080/geoserver/
export GEOSERVER_ADMIN_PASSWORD=geoserver
export SESSION_EXPIRED_CONTROL_ENABLED=False
```

Downgrade psycopg2:

```
pip install psycopg2==2.7.7
```

Apply migrations and apply basic fixtures:

```
cd wfp-geonode
./manage.py migrate --fake-initial
paver sync
```

Regenerate from scratch the upload application tables in the database:

```
delete from django_migrations where app = 'upload';
drop table upload_upload cascade;
drop table upload_uploadfile;
```

Regenerate upload tables with migrate:

```
./manage.py migrate upload
```

Upgrade psycopg2:

```
pip install -r geonode/requirements.txt
```

Create superuser

To create a superuser you should drop the following constraints (they can be re-enabled if needed):

```
alter table people_profile alter column last_login drop not null;
```

```
./manage.py createsuperuser
```

Fixes on database

For some reason some resources were unpublished:

```
UPDATE base_resourcebase SET is_published = true;
```

Remove a foreign key from account_account which is not used anymore (GeoNode dev team: maybe even better let's remove all of the account tables, I think they are stale now):

```
ALTER TABLE account_account DROP CONSTRAINT user_id_refs_id_726cb6b4;  
ALTER TABLE account_signupcode DROP CONSTRAINT "inviter_id_refs_id_49a7c0d9";
```

Fix the remote service layers by running this script:

```
python migration/fixes_remote_layers.py
```

1.29 GeoNode Async Signals

1.29.1 Supervisord and Systemd

1.29.2 Celery

1.29.3 Rabbitmq and Redis

1.30 GeoNode add a thesaurus

1.30.1 Loading a thesaurus

You can add a thesaurus into you GeoNode using the `load_thesaurus` command:

```
python manage.py load_thesaurus --help  
  
-d, --dry-run          Only parse and print the thesaurus file, without perform_  
↳insertion in the DB.  
--name=NAME            Identifier name for the thesaurus in this GeoNode instance.  
--file=FILE            Full path to a thesaurus in RDF format.
```

In order add the inspire-themes thesaurus into a geonode instance, download it as file `inspire-theme.rdf` with the command:

```
wget -O inspire-theme.rdf https://raw.githubusercontent.com/geonetwork/core-  
↳geonetwork/master/web/src/test/resources/thesaurus/external/thesauri/theme/  
↳httpinspireeceuropaeutheme-theme.rdf
```

and then issue the command:

```
python manage.py load_thesaurus --file inspire-theme.rdf --name inspire_themes
```

The name is the identifier you'll use to refer to this thesaurus in your GeoNode instance.

If you only want to make sure that a thesaurus file will be properly parsed, give the `--dry-run` parameter, so that nothing will be added to the DB.

Note: if the name starts with the string `fake`, the file will not be accessed at all, and some test keywords will be added to a fake new thesaurus. In this case the `dry-run` param will not be used.

1.30.2 Configure a thesaurus in GeoNode

After you loaded a thesaurus into GeoNode, it should be configured in the `settings.py` file (or in the `local_settings`) in this way:

```
THESAURUS = {'name': 'THESAURUS NAME', 'required': True|False, 'filter': True|False,}
```

- `name`: (mandatory string) the identifier you used in the `load_thesaurus` commands.
- `required`: (optional boolean) if `True`, a keyword of this thesaurus is mandatory to complete the metadata. *Currently not implemented.*
- `filter`: (optional boolean) if `True`, a faceted list of keywords of this thesaurus will be presented on the search page.

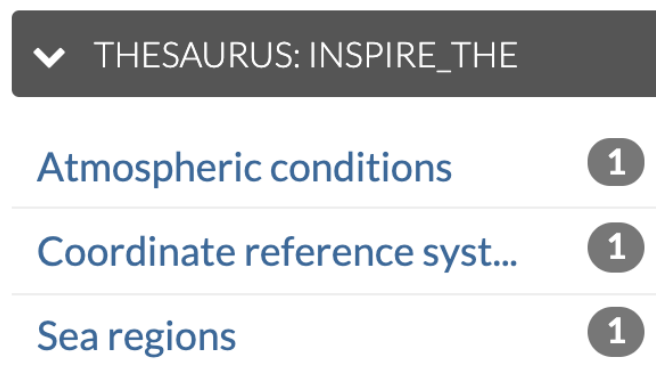
So, in order to set up the INSPIRE themes thesaurus you may set the `THESAURUS` value as:

```
THESAURUS = {'name': 'inspire_themes', 'required': True, 'filter': True}
```

1.30.3 Apply a thesaurus to a resource

After you've finished the setup you should find a new input widget in each resource metadata wizard allowing you to choose a thesaurus for your resource.

After applying a thesaurus to resources those should be listed in the filter section in GeoNodes resource list views.



1.31 Participate in the Discussion

1.31.1 Join the community, ask for help or report bugs

In case of general questions the GeoNode Community is present at following *channels*

- User Mailing List: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-users>
- Developer Mailing List: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-devel>
- Gitter Chat: <https://gitter.im/GeoNode/general>

For reporting bugs please open a ticket at Github issues:

- <https://github.com/GeoNode/geonode/issues>

1.32 Write Documentation

1.32.1 How to contribute to GeoNode's Documentation

If you feel like adding or changing something in the GeoNode documentation you are very welcome to do so. The documentation always needs improvement as the development of the software is going quite fast.

To contribute to the GeoNode documentation you should:

- Read the GeoServer Style Guidelines
- Create an account on GitHub
- Fork the GeoNode repository
- Edit the files
- Submit pull requests

All these things can generally be done within your browser, you won't need to download anything. However, if you need to add images or planning bigger changes working locally is recommended.

Style Guidelines

While we do not have strict rules for writing docs, we encourage you to read GeoServer Style Guidelines before you start writing: <https://docs.geoserver.org/latest/en/docguide/style.html>

Create an account on GitHub

The first step is to create an account on GitHub. Just go to [Github](#), find a username that suits you, enter your email and a password and hit *Sign up for GitHub*. After you've signed in, visit the `geonode_documentation` repository <https://github.com/geonode/documentation>.

Fork the documentation repository

In order to make changes, you first have to fork the repository. On the top right of the website, you will find a button named "fork" to do so.

If you want to read more about forking please visit the official GitHub docs: <https://help.github.com/articles/fork-a-repo>.

Edit files on Github

For smaller changes you can use the GitHub website. Navigate your Browser to your forked repository. To make changes to files, navigate to the file in question and hit the *edit* button on the right top.

Note: The documentation is written in *reStructuredText*, a lightweight markup language. To learn how to use it see: <https://docutils.sourceforge.net/docs/user/rst/quickref.html>.

By hitting the *preview* button you will be able to see how your changes will look like. To save your changes, click on *Commit Changes* at the bottom of the site.

To ask the documentation maintainers to integrate your changes the creation of a *Pull Request* is needed. Therefore use the *new pull request* button to start the process. Find more about Pull requests at the official GitHub documentation: <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests> .

Edit files locally

If you're planning bigger changes on the structure of the documentation, it is advisable to make your changes locally. Further, while you can work on your master branch, it is recommended to create a dedicated branch for your changes.

Start by navigating to a folder where you like to keep your repository locally and install the needed dependencies

```
$ cd /opt
$ git clone https://github.com/your_documentation_repository
$ git remote add upstream https://github.com/geonode/documentation
# add the GeoNode documentation repository as "upstream" source

$ cd your_documentation_repository
$ git fetch upstream;
# get last commits from upstream

$ git merge upstream/master master
# merge the upstream with your fork
# if you like, you can also use 'git pull', which is nothing else than fetching and
↪merging in one step

$ git push
# update your repository at GitHub (origin)
```

Your repository should now be up to date! For more information on those commands go to <https://git-scm.com/docs>. Let's install the dependencies

```
$ pip install virtualenv
$ virtualenv docs_env
$ source docs_env/bin/activate
$ pip install sphinx sphinx_rtd_theme sphinx-autobuild
```

You can now start the sphinx development server which will serve and live-reload your docs at <https://localhost:8000>

```
$ sphinx-autobuild . _build
```

When finished create a build with following command

```
$ make html
# for a last check you can open the index.html in _build subdirectory
```

Create a pull request

As with directly editing files in your browser, you will need to create a Pull request to ask for integrating your changes into the main repository.

```
$ git status
# will list all changed files

$ git add ...
# add the files of interest

$ git commit -m 'Fixes #1234 Updated docs for ...'
# choose a meaningful commit message

$ git push <branch>
```

After running these commands, navigate your browser to your GitHub repository and create a pull request as explained above.

1.33 Provide Translations

1.33.1 Contribute to Translations

Behind the scenes, GeoNode is using a software called GNU gettext further text-based translation files (django.po and djangojs.po) for translating content. If you'd like to know more about how all of this works you'll find a full description at the [Django Docs](#). Following will concentrate on what is needed for edit existing or contribute a new translation.

Download the translation File

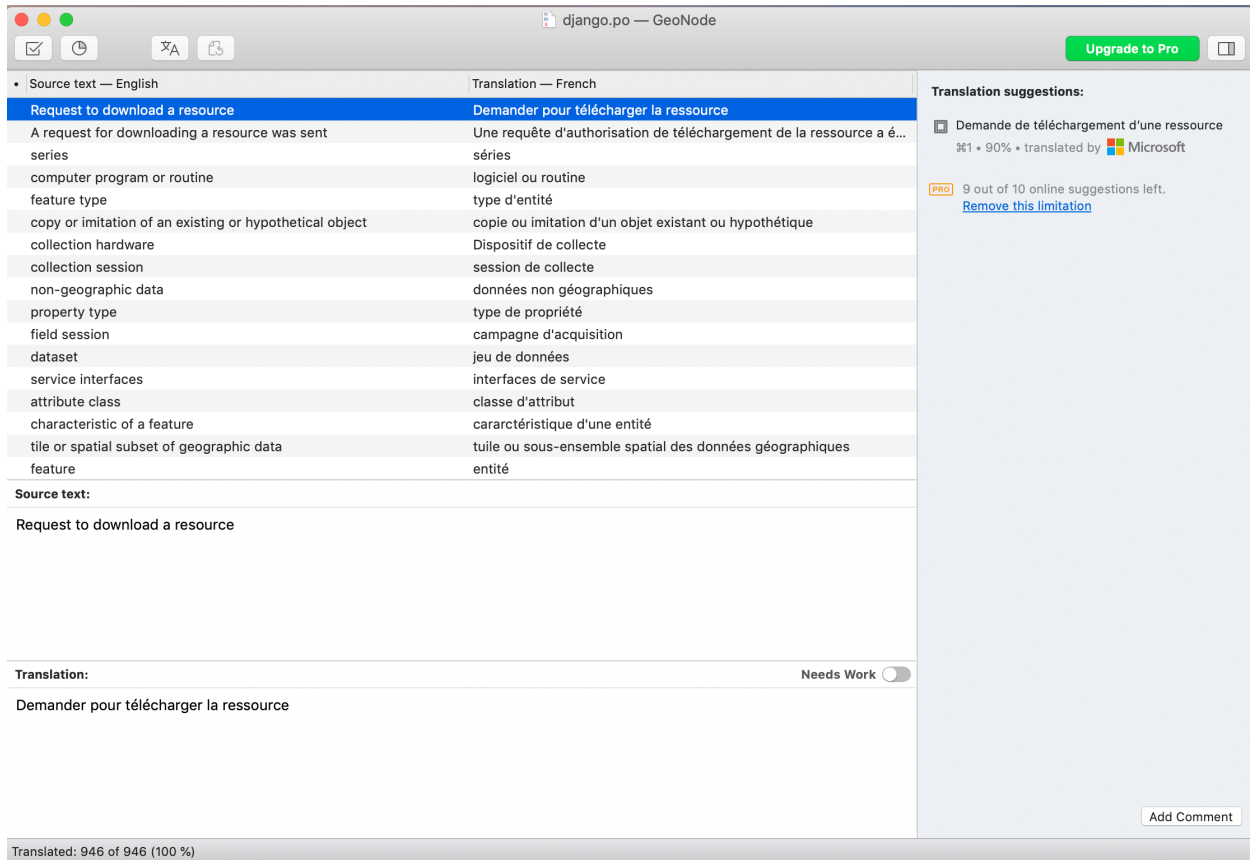
All language files live in a specific subfolder called after their [iso code](#) within the [locale folder](#). For example, for French, the main translation file called django.po can be downloaded from [here](#).

Next, to download the language file, we need to install an OpenSource Editor called “poedit” for editing from: <https://poedit.net/download>

Translation process

Make a copy of the file before starting the translation so that you can revert in case of errors.

After installing ‘poedit’, you should be able to double click on the ‘.po’ file to open it. Poedit’s interface should look similar to the one shown in the picture below:



Identifying translation issues

From the 'poedit' menu 'View', make sure that 'Entries with Errors first' is checked:

Next click on 'Validate Translations' from the 'Catalogue' menu:

'Poedit' will place translations which may require additional consideration on top of the list. A warning mark means that the interpretation might be not entirely consistent with the original phrase. This is not necessarily an error, just a warning asking the user to double check.

Following to marked phrases, 'Poedit' will show untranslated sentences. When clicking on one, it can be translated through the bottom panel.

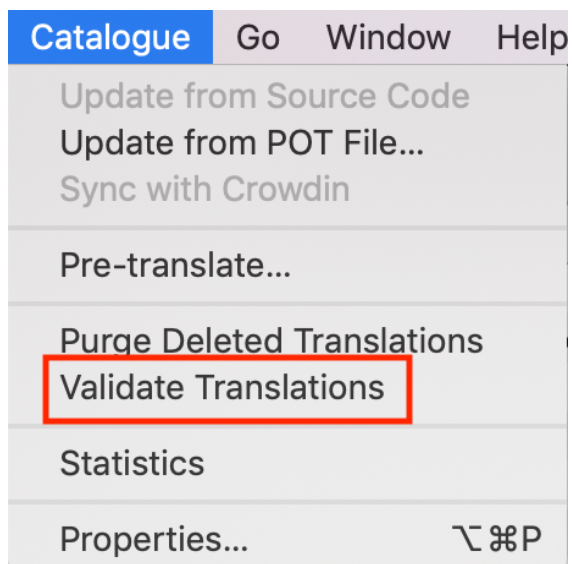
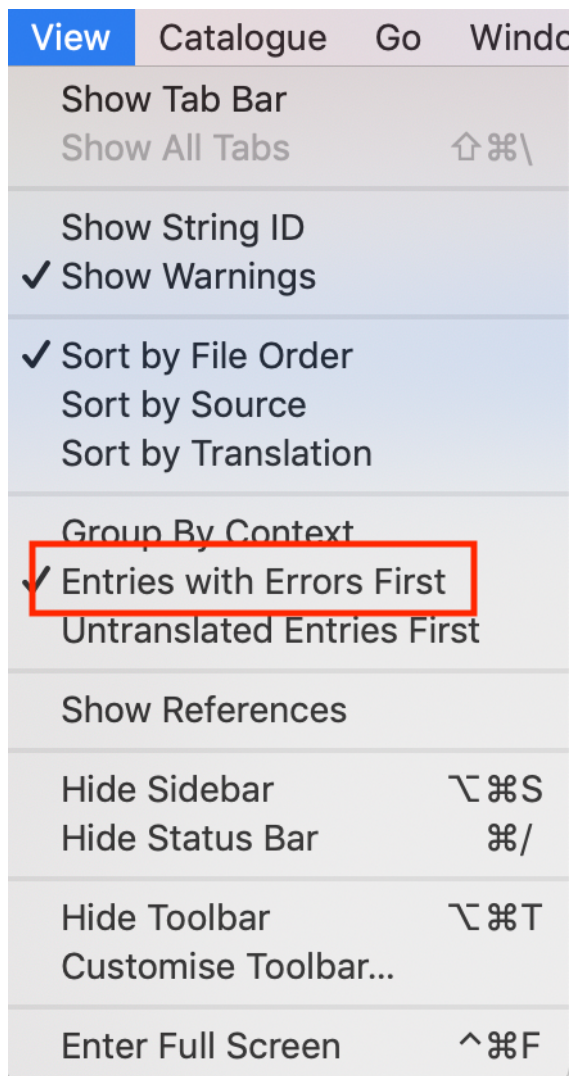
During translation pay special attention to the button saying 'needs work'. In case this button is checked, the phrase will be marked as 'fuzzy' and ignored in GeoNode.

Saving translations

As soon as the translation is complete, it must be saved and compiled. Saving is straightforward. All you have to do is clicking the 'Save' button from the top menu.

As a last step we compile the file. Compiling the translation means to create a binary ".mo" file out of the edited ".po" file. To do so, click on "Compile to MO"

Poedit will ask where to write the ".mo" file to, by default, this is the same folder as the edited '.po' resides in. The '.mo' file can be overwritten if necessary.



django.po — GeoNode

Upgrade to Pro

Source text — English	Translation — French
last modified	Dernière modification
The following styles are associated with this layer. Choose a style t...	Les styles suivants sont associés à cette couche. Choisissez un st...
last updated on	Dernière mise à jour sur
or select them one by one:	ou sélectionnez les un par un:
Replace Layer:	Remplacer la couche :
Provide CRS for	Fournir des CRS pour
Editing details for	Modification des informations relatives
Or just go	Ou tout simplement aller
A rating was given to a map	Une évaluation a été donnée à une carte
ows URL	URL ows
local OWS	OWS local
Note: this map's original metadata was populated by importing a m...	Note: les métadonnées originales de cette carte ont été remplies à...
An unknown error has occurred.	Une erreur inconnue s'est produite
party that accepts accountability and responsibility for the data an...	groupe qui accepte la responsabilité des données et assure la mai...
You can use the login form at	Vous pouvez vous authentifier sur la page d'authentification sur
Please go to resource page and assign the download permissions i...	S'il vous plaît rendez-vous à la page décrivant la ressource et assig...
Request to download a resource	Demander pour télécharger la ressource

Source text:

Request to download a resource

Translation: Needs Work ☐

Demander pour télécharger la ressource

Translation suggestions:

☒ Demande de téléchargement d'une ressource
 100% • 90% • translated by Microsoft

PRO 9 out of 10 online suggestions left.
[Remove this limitation](#)

Add Comment

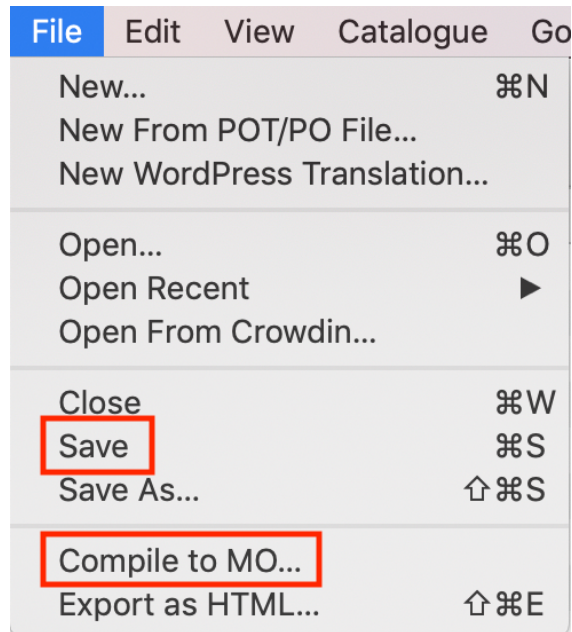
Translated: 946 of 946 (100 %)

Source text:

Request to download a resource

Translation: Needs Work ☐

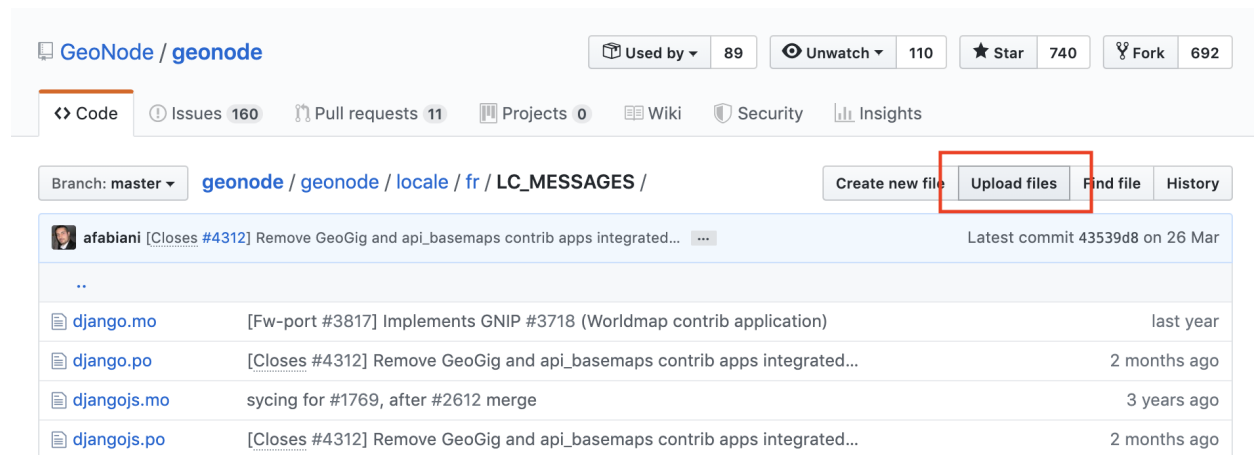
Demander pour télécharger la ressource



Push translations to the repository

For sharing our updates, we must upload the files to GeoNode’s GitHub repository. Go to the correct file position which, in case for French is: https://github.com/GeoNode/geonode/tree/master/geonode/locale/fr/LC_MESSAGES

Click on “Upload Files”



Drag the updated files into the Upload form, and write a title/description of the changes

Click on “Create a new branch for this commit...” and then click on the green button.


The last step will create a *PULL REQUEST* which can be reviewed and then approved by a developer.

GeoNode / geonode

Used by 89Unwatch 110★ Star 740🔗 Fork 692

<> Code🔔 Issues 160🔗 Pull requests 11📁 Projects 0📖 Wiki🛡 Security📊 Insights

geonode / geonode / locale / fr / LC_MESSAGES



Drag files here to add them to your repository

Or [choose your files](#)

Commit changes

Add files via upload

Add an optional extended description...

- ☐ ⓘ You can't commit to `master` because it is a [protected branch](#).
- ☒ ⓘ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests](#).

🔗 New branch name

Commit changes

Cancel

Activate updated translation at your server

Once the files have been pushed to GitHub, it will be necessary to update your server to respect changed files.

At this time, this can be done only by an administrator. From the server 'shell' following commands need to be executed:

```
workon geonode
cd /opt/geonode
DJANGO_SETTINGS_MODULE=geonode.settings python -W ignore manage.py collectstatic --
↳noinput
sudo service uwsgi restart
```

Texts not listed in .po files

In case you find a template output without corresponding translation you can add it as follows:

Identify the corresponding template file which is responsible for outputting the text. Add a `{% trans "TEXT" %}` tag. Save the template file and run the following:

```
django-admin makemessages -l en -d django -e "html,txt,py" -i docs
django-admin makemessages -l en -d djangojs -e "js" -i docs -i node_modules -i lib
```

This will update the english .po file. also to update the language which should be edited by settings the `-l fr` parameter. Continue with updating the .po file as described above.

1.34 Write Code

1.35 Frontend Development

1.35.1 Frontend development

Knowledge of handling node/npm is required.

The GeoNode frontend dependencies can be found in `./geonode/static`. To manage dependencies, we recommend the use of yarn package manager (<https://yarnpkg.com/lang/en>).

First steps:

```
yarn install
```

Installs the required libraries to `./node_modules`

```
yarn install <package>@version [--dev]
```

Installs a package with a defined version. Using `-dev` installs a dependency that is only available for the build process (see: `package.json devDependencies`).

```
yarn remove <package>
```

Removes a package.

```
yarn outdated
```

Shows version information.

```
yarn why <package>
```

Get information on why this package was installed.

For further information on how to install or use please visit the official yarn documentation.

File/Folder overview:

```
./static_dependencies.json
```

includes all dependencies associated with each file. For example all files which should be minified to `assets.min.js` are named as values. All files that should be copied to `lib` folder (for `DEBUG_STATIC`) are values of key `other_dependencies` and so on. Before you can use a dependency it has to be added to `package.json` by use of yarn.

```
./Gruntfile.js
```

reads the dependencies from `static_dependencies.json` and contains all workflows.

geonode/static/geonode

The `./geonode` folder contains GeoNode's stylesheets and javascript files. The CSS files are generated via less. CSS files should therefore never be changed directly but it's corresponding less file. Further this folder should never be deleted!

geonode/static/lib

The `./lib` folder contains all the third-party files. This folder can be deleted as it will be fully generated by use of `grunt development|production`

Example 1 – Change styling:

1. In your settings set `DEBU_STATIC=True`. This will load unminified assets in your template.
2. Start the development server with `paver start`.
3. Use `grunt watch` to watch all less files for change.
4. Change styling in `./geonode/static/geonode/less`
5. If our changes are as expected create a new build with `grunt development` (files are not minimized) or `grunt production` (files are minimized)

Example 2 – add/update a new library:

1. In your settings set `DEBU_STATIC=True`. This will load unminified assets in your template.
2. `yarn add angular@1.7`
3. `vim static_dependencies.json` Edit the file and add your dependency to its fitting destination. For example, `assets.min.js`
4. Check if some Django template (for example, `base.html`) includes the file and add it or adjust the version
5. use `grunt production` to build the package

For further tasks have a look at `gruntfile.js` or ask for help in the development mailing list

Note: Please make maintainers work easier and add a message to your commit why a library has been added. (For example, `commit -m 'select2 added for permissions form on layer detail page'`)

1.36 How to Develop

1.36.1 Start to develop with Docker

How to run the instance for development

Set the variable `SET_DOCKER_ENV` for development

```
vi .env
```

Change to

```
SET_DOCKER_ENV=development
```

Use dedicated docker-compose files while developing

Note: In this example we are going to keep `localhost` as the target IP for GeoNode

```
docker-compose -f docker-compose.async.yml -f docker-compose.development.yml up
```

How to debug

Note: We are supposing to use `ipdb` for debugging which is already available as package from the container

Stop the container for the `django` service:

```
docker-compose stop django
```

Run the container again with the option for *service ports*:

```
docker-compose run \
-e DOCKER_ENV=development \
-e IS_CELERY=False \
-e DEBUG=True \
-e GEONODE_LB_HOST_IP=localhost \
-e GEONODE_LB_PORT=80 \
-e SITEURL=http://localhost/ \
-e ALLOWED_HOSTS="['localhost', ]" \
-e GEOSERVER_PUBLIC_LOCATION=http://localhost/geoserver/ \
```

(continues on next page)

(continued from previous page)

```
-e GEOSERVER_WEB_UI_LOCATION=http://localhost/geoserver/ \
--rm --service-ports django python manage.py runserver --settings=geonode.
↪ settings 0.0.0.0:8000
```

Access the site on <http://localhost/>

Note: If you set an ipdb debug point with `import ipdb ; ipdb.set_trace()` then you should be facing its console and you can see the django server which is restarting at any change of your code from your local machine.

1.36.2 How to Install GeoNode-Core for development

Summary of installation

This section demonstrates a summarization of the steps to be followed in order to install GeoNode-Core for development using Ubuntu 18.04. The following steps will be customized to fit both GeoNode-Project and GeoNode-Core for development purpose.

The steps to be followed are:

- 1- Install build tools and libraries
- 2- Install dependencies and supporting tools
- 3- Setup Python virtual environment
- 4- Clone and install GeoNode from Github
- 5- Install and start Geoserver
- 6- Start GeoNode

Note: The following commands/steps will be executed on your terminal

Warning: If you have a running GeoNode service, you will need to stop it before starting the following steps. To stop GeoNode you will need to run:

```
service apache2 stop    # or your installed server
service tomcat7 stop    # or your version of tomcat
```

Install GeoNode-Core for development

GeoNode-Core installation is considered the most basic form of GeoNode. It doesn't require any external server to be installed and it can run locally against a file-system based SQLite database.

Installation steps

1- Install build tools and libraries

```
$ sudo apt-get install -y build-essential libxml2-dev libxslt1-dev libpq-dev zlib1g-  
↳dev
```

2- Install dependencies and supporting tools

Install python native libraries and tools

```
$ sudo apt-get install -y python3-dev python3-pil python3-lxml python3-pyproj python3-  
↳shapely python3-nose python3-httpplib2 python3-pip software-properties-common
```

Install python virtual environment

```
$ sudo pip install virtualenvwrapper
```

Install postgresql and postgis

```
$ sudo apt-get install postgresql-10 postgresql-10-postgis-2.4
```

Change postgres password expiry and set a password

```
$ sudo passwd -u postgres # change password expiry information  
$ sudo passwd postgres # change unix password for postgres
```

Create geonode role and database

```
$ su postgres  
$ createdb geonode_dev  
$ createdb geonode_dev-imports  
$ psql  
$ postgres=#  
$ postgres=# CREATE USER geonode_dev WITH PASSWORD 'geonode_dev'; # should be same as  
↳password in setting.py  
$ postgres=# GRANT ALL PRIVILEGES ON DATABASE "geonode_dev" to geonode_dev;  
$ postgres=# GRANT ALL PRIVILEGES ON DATABASE "geonode_dev-imports" to geonode_dev;  
$ postgres=# \q  
$ psql -d geonode_dev-imports -c 'CREATE EXTENSION postgis;'  
$ psql -d geonode_dev-imports -c 'GRANT ALL ON geometry_columns TO PUBLIC;'  
$ psql -d geonode_dev-imports -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'  
$ exit
```

Edit PostgreSQL configuration file

```
sudo gedit /etc/postgresql/10/main/pg_hba.conf
```

Scroll to the bottom of the file and edit this line

```
# "local" is for Unix domain socket connections only  
local    all             all                                     peer
```


To be as follows

```
# "local" is for Unix domain socket connections only
local    all                all                                trust
```

Then restart PostgreSQL to make the changes effective

```
sudo service postgresql restart
```

Java dependencies

```
$ sudo apt-get install -y openjdk-11-jdk --no-install-recommends
```

Install supporting tools

```
$ sudo apt-get install -y ant maven git gettext
```

3- Setup Python virtual environment (Here is where Geonode will be running)

Add the virtualenvwrapper to your new environment.

Since we are using Ubuntu, you can add the following settings to your .bashrc file. Please note that the Ubuntu account here is called “geonode”. So you will need to change it according to the name you picked.

```
$ echo export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python >> ~/.bashrc
$ echo export WORKON_HOME=/home/geonode/dev/.venvs >> ~/.bashrc
$ echo source /usr/local/bin/virtualenvwrapper.sh >> ~/.bashrc
$ echo export PIP_DOWNLOAD_CACHE=$HOME/.pip-downloads >> ~/.bashrc
```

And reload the settings by running

```
$ source ~/.bashrc
```

Set up the local virtual environment for Geonode

```
$ vim ~/.bashrc
# add the following line to the bottom
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

```
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
$ mkvirtualenv --python=/usr/bin/python3 geonode
$ workon geonode # or $ source /home/geonode/dev/.venvs/geonode/bin/activate
This creates a new directory where you want your project to be and creates a new
↳ virtualenvironment
```

4- Download/Clone GeoNode from Github

To download the latest geonode version from github, the command “git clone” is used

Note: If you are following the GeoNode training, skip the following command. You can find the cloned repository in /home/geonode/dev

```
$ git clone https://github.com/GeoNode/geonode.git
```

Install Nodejs PPA and other tools required for static development

This is required for static development

Note: If you are following GeoNode’s training, nodejs is already installed in the Virtual Machine skip the first three command and jump to `cd geonode/geonode/static`

```
$ sudo apt-get install nodejs npm
$ cd geonode/geonode/static
$ npm install --save-dev
```

Note: Every time you want to update the static files after making changes to the sources, go to `geonode/static` and run ‘`grunt production`’.

Warning: Starting from the following step, you have to make sure that you installed GDAL correctly according to the documentation page “Install GDAL for Development”

Install GeoNode in the new active local virtualenv

```
$ cd /home/geonode/dev # or to the directory containing your cloned GeoNode
$ pip install -e geonode
$ cd geonode/geonode
```

Create `local_settings.py`

Copy the sample file `/home/geonode/dev/geonode/geonode/local_settings.py.geoserver.sample` and rename it to be `local_settings.py`

```
$ cd /home/geonode/dev/geonode
$ cp geonode/local_settings.py.geoserver.sample geonode/local_settings.py
$ gedit geonode/local_settings.py
```

In the `local_settings.py` file, add the following line after the import statements:

```
SITEURL = "http://localhost:8000/"
```

In the `DATABASES` dictionary under the ‘default’ key, change only the values for the keys `NAME`, `USER` and `PASSWORD` to be as follows:

```
DATABASES = {
'default': {
    'ENGINE': 'django.db.backends.postgresql_psycopg2',
    'NAME': 'geonode_dev',
    'USER': 'geonode_dev',
    'PASSWORD': 'geonode_dev',
    .....
    .....
    ....
    ...
    ...
} ... }
```

In the `DATABASES` dictionary under the ‘datastore’ key, change only the values for the keys `NAME`, `USER` and `PASSWORD` to be as follows:

```
# vector datastore for uploads
'datastore' : {
    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    # 'ENGINE': '', # Empty ENGINE name disables
    'NAME': 'geonode_dev-imports',
    'USER' : 'geonode_dev',
    'PASSWORD' : 'geonode_dev',
    .....
    .....
    .....
    ....
    ...
}
```

In the CATALOGUE dictionary under the 'default' key, uncomment the USER and PASSWORD keys to activate the credentials for GeoNetwork as follows:

```
CATALOGUE = {
'default': {
    # The underlying CSW implementation
    # default is pycsw in local mode (tied directly to GeoNode Django DB)
    'ENGINE': 'geonode.catalogue.backends.pycsw_local',
    # pycsw in non-local mode
    # 'ENGINE': 'geonode.catalogue.backends.pycsw_http',
    # GeoNetwork opensource
    # 'ENGINE': 'geonode.catalogue.backends.geonetwork',
    # deegree and others
    # 'ENGINE': 'geonode.catalogue.backends.generic',
    # The FULLY QUALIFIED base url to the CSW instance for this GeoNode
    'URL': urljoin(SITEURL, '/catalogue/csw'),
    # 'URL': 'http://localhost:8080/geonetwork/srv/en/csw',
    # 'URL': 'http://localhost:8080/deegree-csw-demo-3.0.4/services',
    # login credentials (for GeoNetwork)
    'USER': 'admin',
    'PASSWORD': 'admin',
    # 'ALTERNATES_ONLY': True,
    }}
}
```

5- Install and Start Geoserver

From the virtual environment, first you need to align the database structure using the following command :

```
$ cd /home/geonode/dev/geonode
$ python manage.py migrate
```

Warning: If the start fails because of an import error related to osgeo or libgeos, then please consult the [Install GDAL for Development](#)

then setup GeoServer using the following command:

```
$ paver setup
$ paver sync
```

6- Now we can start our geonode instance

Warning: Don't forget to stop the GeoNode Production services if enabled

```
service apache2 stop
service tomcat7 stop
```

```
$ paver start
```

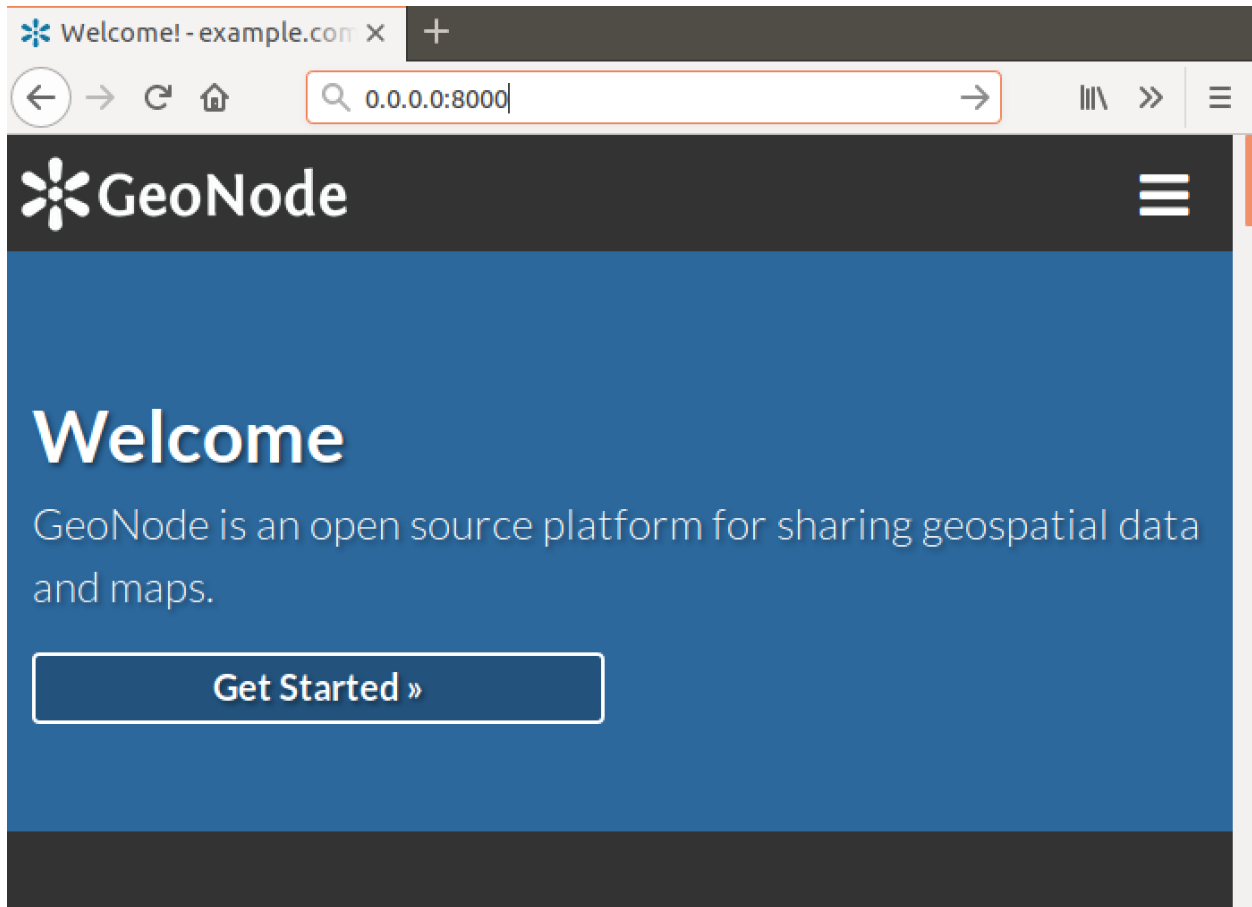
The starting process will take around 20 seconds (depends on your machine) and at the end it shows the following message:

```
[tasks]
. geonode.documents.tasks.create_document_thumbnail
. geonode.documents.tasks.delete_orphaned_document_files
. geonode.documents.tasks.delete_orphaned_thumbnails
. geonode.geoserver.tasks.geoserver_update_layers
. geonode.layers.tasks.delete_layer
. geonode.maps.tasks.delete_map
. geonode.security.tasks.synch_guardian
. geonode.services.tasks.update.harvest_resource
. geonode.tasks.email.send_mail
. geonode.tasks.notifications.send_queued_notifications
. imagekit.cachefiles.backends._generate_file

Performing system checks...

System check identified no issues (1 silenced).
January 21, 2020 - 22:49:50
Django version 1.11.27, using settings 'geonode.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
GeoNode is now available.
```

Now you can visit the geonode site by typing <http://0.0.0.0:8000> into your browser window



Install GeoNode-Project for development after installing GeoNode-Core

Geonode-Project gives the user flexibility to customize the installation of the GeoNode. Geonode itself will be installed as a requirement of your project. Inside the project structure it is possible to extend, replace or modify all geonode components (e.g. css and other static files, templates, models..) and even register new django apps without touching the original Geonode code. In order to install GeoNode-Project, the following steps need to be executed alongside the previous GeoNode-Core installation steps.

1- Use `django-admin.py` to create a project “my_geonode” from a GeoNode-Project template as follows:

Note: Before running the following command, make sure that you are currently working on the virtual environment and just outside geonode directory. The command will create a new project called “my_geonode” which should be located at the level of geonode-core installation directory “inside /home/geonode/dev”

```
$ django-admin.py startproject my_geonode --template=https://github.com/GeoNode/
→geonode-project/archive/master.zip -e py,rst,json,yml,ini,env,sample -n Dockerfile
$ ls /home/geonode/dev # should output: geonode my_geonode
```

Note: Although the following command might show that the majority of requirements are already satisfied “because GeoNode-Core was already installed”, it is recommended to still execute it as it might update or install any missing

package.

2- Install all the required packages/tools for GeoNode-Project as follows:

```
$ pip install -e my_geonode
```

Note: As mentioned earlier, GeoNode will be installed as requirement for the GeoNode-Project in order to be able to extend it

Install GeoNode-Project directly from scratch

If you didn't install GeoNode-Core earlier and you wanted to install GeoNode-Project directly, please follow these steps

1- Create a virtual environment as follows:

```
$ vim ~/.bashrc
# add the following line to the bottom
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

```
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
$ mkvirtualenv --python=/usr/bin/python3 my_geonode
```

2- Clone the geonode-project repo from Github

```
$ git clone https://github.com/GeoNode/geonode-project.git
```

3- Install Django framework as follows

```
$ pip install Django==2.2.9
```

4- Use django-admin.py to create a project "my_geonode" from a GeoNode-Project template as follows:

```
$ django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,
→env,sample -n monitoring-cron -n Dockerfile my_geonode
```

5- Install all the requirements for the GeoNode-Project and install the GeoNode-Project using pip

```
$ cd my_geonode
$ pip install -r requirements.txt --upgrade
$ pip install -e . --upgrade
```

6- Install GDAL Utilities for Python

```
$ pip install pygdal=="gdal-config --version`.*" # or refer to the link <Install_
→GDAL for Development <https://training.geonode.geo-solutions.it/005_dev_workshop/
→004_devel_env/gdal_install.html>
```

7- Install GeoServer and Tomcat using paver

```
$ paver setup
$ paver sync
```

(continues on next page)

(continued from previous page)

```
$ paver start
```

8- Visit <http://localhost:8000/>

1.36.3 How to run GeoNode Core for development

In order to start Geonode Core for development, you need to make sure that no Geonode instance is running first. This can be done by running the following commands:

```
$ cd /home/user/geonode
$ paver stop_geoserver
$ paver stop_django
```

Then you need to start both geoserver and django services as follows:

```
$ paver start_geoserver
$ paver start_django
```

Now you can visit your Geonode GUI by typing <http://localhost:8000> into your browser window

1.36.4 How to run GeoNode Project for development

In order to run a project for development, the following steps have to be followed:

1- Make sure there is no running instance of Geonode first by running the following command:

```
$ cd /home/user/my_geonode
$ paver stop
```

The above command will stop all services related to Geonode if running

2- Start the servers by running paver start as follows:

```
$ paver start
```

Now you can visit your geonode project site by typing <http://localhost:8000> into your browser window

1.36.5 Workshops

The workshops documentation demonstrates few examples on how to utilize GeoNode-Project in order to extend/customize GeoNode's functionalities according to your business. The covered topics include the following:

- 1- Customize your GeoNode with the geonode-project
- 2- Customize the look and feel
- 3- Create your ResourceBase Metadata
- 4- Create your own django app
- 5- Add a custom model

6- Permissions and APIs

7- Deploy your GeoNode

1- Customize your GeoNode with the geonode-project

In this example, GeoNode-Project is cloned to create a template instance in which the rest of the examples will be building on top of it.

1- Assuming you already installed GeoNode-Core, firstly we need to create a GeoNode-Project template and this can be achieved from the following command:

```
$ django-admin.py startproject my_geonode --template=https://github.com/GeoNode/  
↳geonode-project/archive/master.zip -e py,rst,json,yml,ini,env,sample -n Dockerfile
```

Here, django-admin is used with startproject option to create my_geonode project copying the template which is passed as GeoNode-project Github repo. It also includes “py,rst,json,yml,ini,env,sample” extensions

2- Once the cloning finished, the next step is to install the GeoNode-Project we just downloaded as follows:

```
$ pip install -e my_geonode
```

3- Install geoserver using paver as follows

```
$ cd /home/geonode/my_geonode  
$ paver setup
```

4- Note the GeoNode database connection parameters mentioned in the local_settings.py configuration file. If not found, copy local_settings.py.sample and rename it to local_settings.py then use psql to create the required user and grant the required privileges as follows:

```
$ su postgres  
$ createdb geonode  
$ psql  
postgres=# CREATE USER geonode WITH PASSWORD 'geonode';  
CREATE ROLE  
postgres=# GRANT ALL PRIVILEGES ON DATABASE "geonode" to geonode;  
GRANT  
postgres=# \q
```

Warning: Don’t forget to exit from postgres user before executing the following commands

5- Run GeoNode using paver

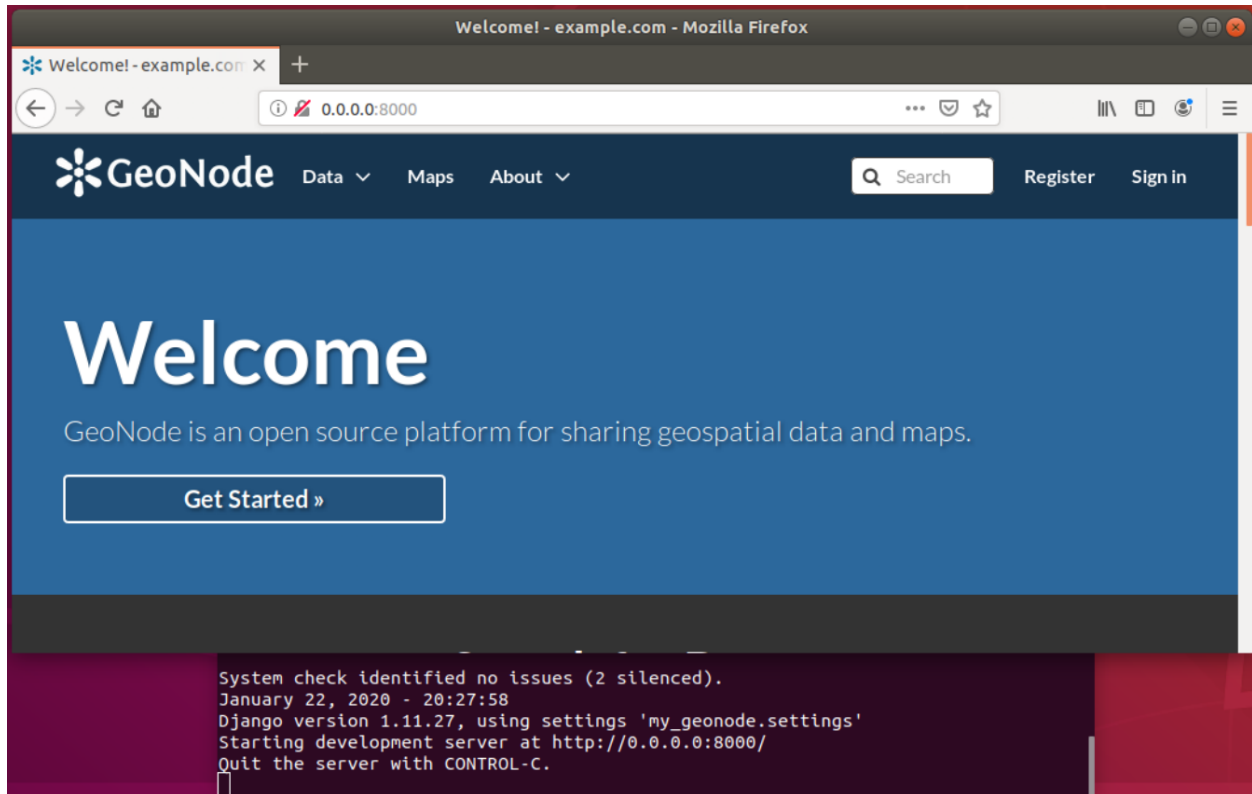
```
$ cd /home/geonode/my_geonode  
$ paver start
```

Note: You may find this warning message: You have 132 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): account, actstream, admin, agon_ratings, announcements, auth, avatar, base, contenttypes, dialogos, documents, favorite, geonode_client, geonode_themes, groups, guardian, invitations, layers, maps, mapstore2_adapter, monitoring, oauth2_provider, people, pinax_notifications, services, sessions, sites, socialaccount, taggit, tastypie, upload, user_messages. Run ‘python manage.py migrate’ to apply them.

Which means you have some sql statements not executed yet and you need to run the “migrate” to sync your database first then “paver start” again as follows:

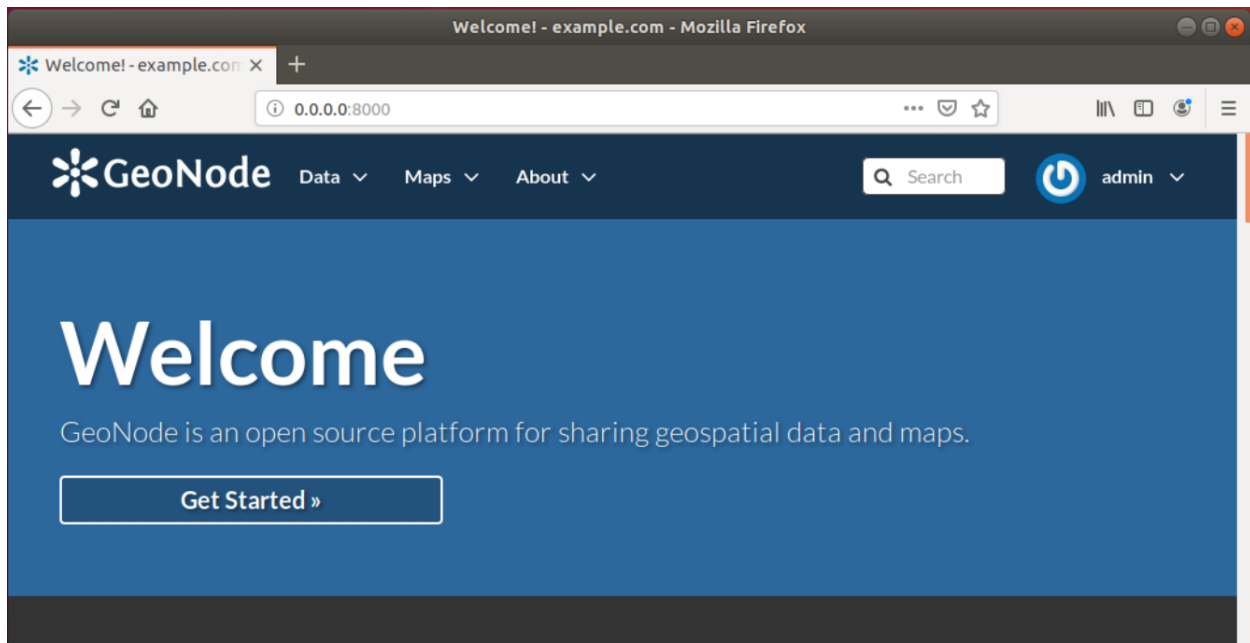
```
$ python manage.py migrate
$ paver start
```

Warning: If encountered this message: (Invalid HTTP_HOST header: '0.0.0.0:8000'. You may need to add u'0.0.0.0' to ALLOWED_HOSTS) It can be fixed in the settings.py file. You will need to add: `ALLOWED_HOSTS = ['0.0.0.0']` in settings.py



6- Once the previous step is done, you can visit 0.0.0.0:8000 to view the GUI of GeoNode. However, we still don't have an account in order to login from the GUI. This can be done using “paver sync”. The command will create sync with latest fixtures and also creates a superuser “admin” with default password “admin”

7- Use the created account to login from the GUI through localhost:8000 or 0.0.0.0:8000



2- Customize the look and feel

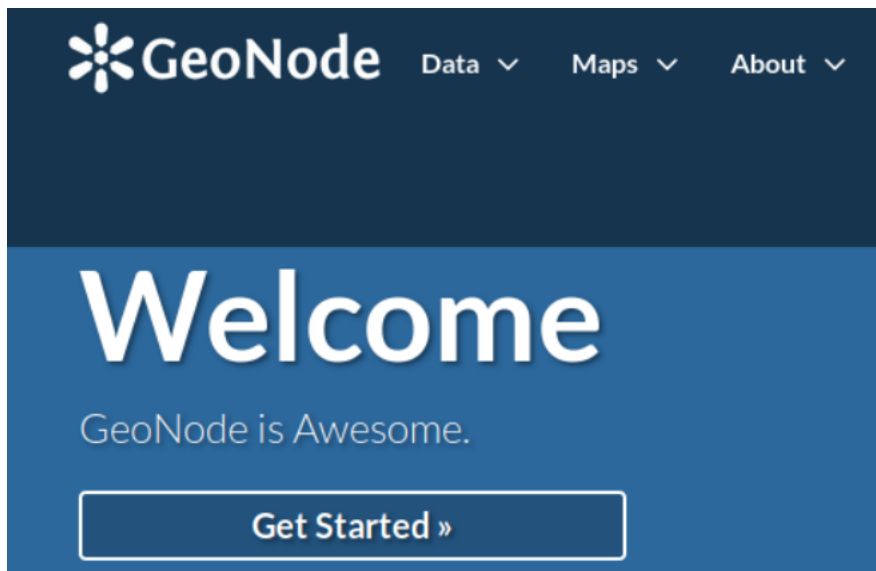
In this section we will change the look and feel of GeoNode, in particular we will do some customization to help understanding how the template inheritance works and how to add new stuff to your GeoNode. The changes will include the home page, the top menu, the footer and a generic GeoNode page.

Homepage:

The geonode-project provides some predefined templates to change the home page and the general site content.

In the “my_geonode/my_geonode/templates” directory we can edit the site_index.html.

Try to edit the content of the “jumbotron” box in the page, save and refresh your browser to see the changes.



The theme:

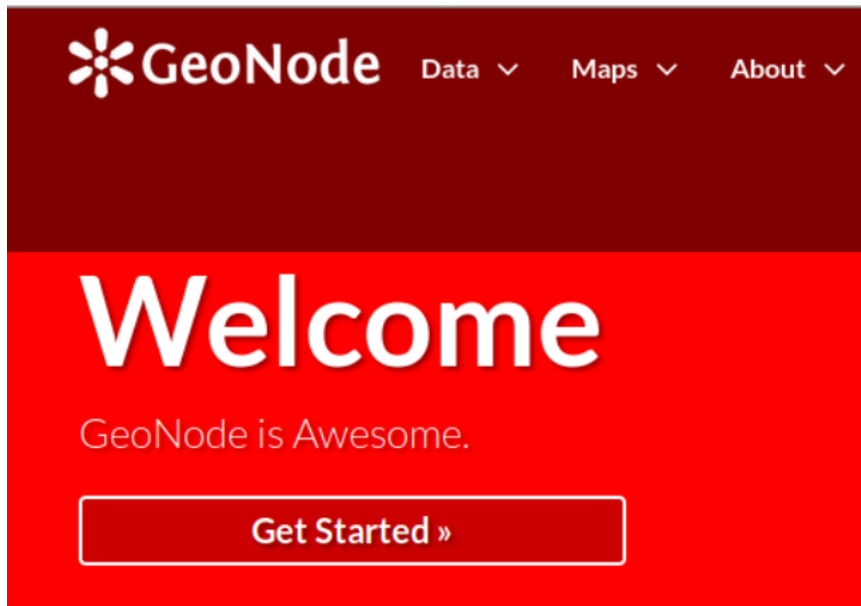
To change the theme of our geonode-project we can act on the `site_base.css` file available in the “`my_geonode/my_geonode/static/css`” folder.

The file is empty so we can inspect elements of the home page with the browser’s developer tools and define css rules in there.

For example, if we want to change the background of the jumbotron, in this file we can add

```
.home .jumbotron { background: red }
```

Then once we refreshed the browser, we should see the change as follows:



Adding the “`.home`” class is necessary in order to let the rule have precedence/priority over the GeoNode’s one. We can see this by inspecting the element in the developer console.

The top menu:

Now we can make some changes that will apply to the whole site. We can add a Geocollections entry in the top menu bar.

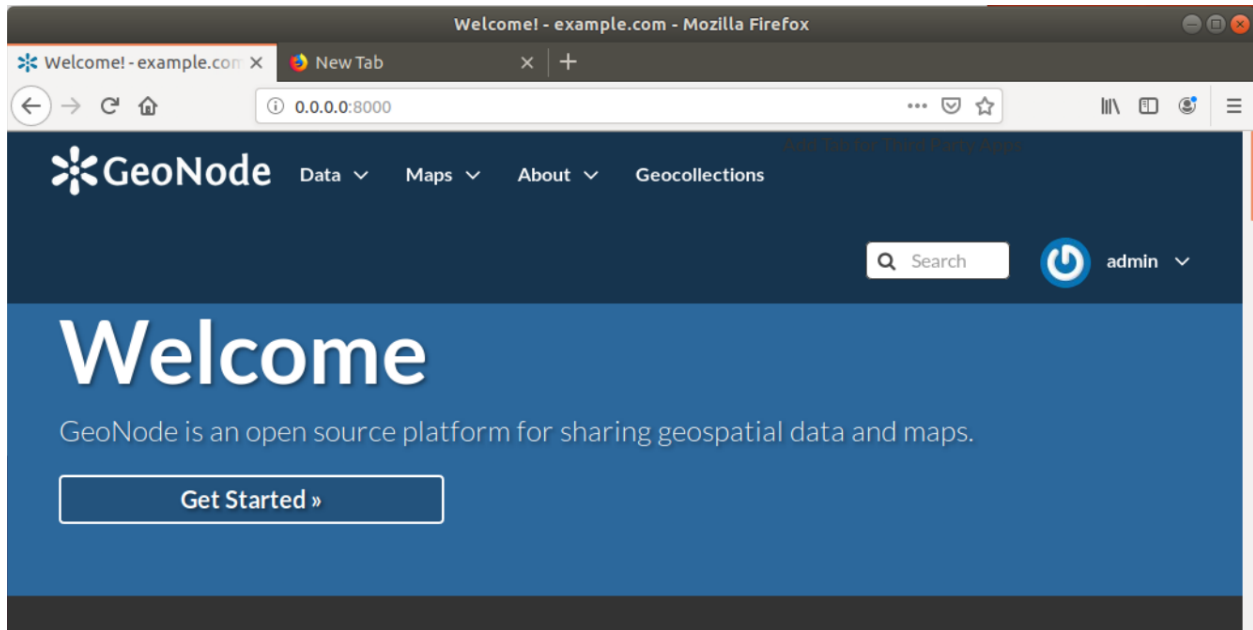
Edit the `site_base.html` file in the templates folder and uncomment the list item adapting the text as well from:

```
{% comment %}
Add Tab for Third Party Apps
<li>
  <a href="{{ PROJECT_ROOT }}app">App</a>
</li>
{% endcomment %}
```

To:

```
<li>
  <a href="{{ PROJECT_ROOT }}/geocollections">Geocollections</a>
</li>
```

On browser refresh you will see a new entry in the nav bar which is persistent to the whole site.



GeoNode generic page

As you can see in the templates folder there are only the `site_index.html` and the `site_base.html` files. In order to customize another GeoNode page, for example the layers list page, you need to recreate the same folder structure of the GeoNode templates folder and add a file with the same name.

For the layers list page we can create a directory named “layers” inside the template directory and a file named “layer_list.html” inside layers. The changes made in this file will only affect the layer list page.

```
mkdir -p my_geonode/templates/layers/

cp geonode/geonode/layers/templates/layers/layer_list.html my_geonode/templates/
↳ layers/layer_list.html

vim my_geonode/templates/layers/layer_list.html
```

For example change in page title to be:

```
<h2 class="page-title">{% trans "Explore My Layers" %}</h2>
```

then refresh the browser to see the update.



Modify functionality

In this section, we will patch the ResourceBase of GeoNode and update the Templates in order to add one more field to the Metadata Schema.

We will add a DOI field to the ResourceBase model and modify the Templates in order to show the new field both into the Metadata Wizard and the Layer Details page.

Note: Make sure to be inside “my_geonode” directory to execute the following commands

Customizing metadata can be achieved from the model which is defined in the core at “geonode/geonode/base/models.py” as follows:

```
# internal fields
uuid = models.CharField(max_length=36)
owner = models.ForeignKey(
    settings.AUTH_USER_MODEL,
    blank=True,
    null=True,
    related_name='owned_resource',
    verbose_name=_("Owner"))
contacts = models.ManyToManyField(
    settings.AUTH_USER_MODEL,
    through='ContactRole')
title = models.CharField(_('title'), max_length=255, help_text=_(
    'name by which the cited resource is known'))
alternate = models.CharField(max_length=128, null=True, blank=True)
date = models.DateTimeField(
    _('date'),
    default=now,
    help_text=date_help_text)
date_type = models.CharField(
    _('date type'),
    max_length=255,
    choices=VALID_DATE_TYPES,
    default='publication',
    help_text=date_type_help_text)
edition = models.CharField(
```

(continues on next page)

(continued from previous page)

```

_('edition'),
    max_length=255,
    blank=True,
    null=True,
    help_text=edition_help_text)
abstract = models.TextField(
    _('abstract'),
    max_length=2000,
    blank=True,
    help_text=abstract_help_text)
purpose = models.TextField(
    _('purpose'),
    max_length=500,
    null=True,
    blank=True,
    help_text=purpose_help_text)
maintenance_frequency = models.CharField(
    _('maintenance frequency'),
    max_length=255,
    choices=UPDATE_FREQUENCIES,
    blank=True,
    null=True,
    help_text=maintenance_frequency_help_text)

```

To add fields directly to the ResourceBase Class without actually modifying it, this can be done from “my_geonode/my_geonode/apps.py” file

The “ready” method is invoked at initialization time and can be currently used to tweak your app in several ways

```

class AppConfig(BaseAppConfig):

    name = "my_geonode"
    label = "my_geonode"

    def ready(self):
        super(AppConfig, self).ready()
        run_setup_hooks()

```

Now we will add the “patch_resource_base” method to the AppConfig and execute it from the ready method as follows:

```

from django.db import models
from django.utils.translation import ugettext_lazy as _

class AppConfig(BaseAppConfig):

    name = "my_geonode"
    label = "my_geonode"

    def _get_logger(self):
        import logging
        return logging.getLogger(self.__class__.__module__)

    def patch_resource_base(self, cls):
        self._get_logger().info("Patching Resource Base")
        doi_help_text = _('a DOI will be added by Admin before publication.')
        doi = models.TextField(

```

(continues on next page)

(continued from previous page)

```

        _('DOI'),
        blank=True,
        null=True,
        help_text=doi_help_text)
    cls.add_to_class('doi', doi)

def ready(self):
    super(AppConfig, self).ready()
    run_setup_hooks()

from geonode.base.models import ResourceBase
self.patch_resource_base(ResourceBase)

```

Note: you will need to perform migrations as follows: - Add field doi to resourcebase

Once you run python manage.py migrate:

```

Running migrations:
Applying announcements.0002_auto_20200119_1257... OK
Applying base.0031_resourcebase_doi... OK
Applying people.0027_auto_20200119_1257... OK

```

Till now, we have patched the DB. however, it is not yet sufficient as we still need to display the added field.

Let's extend the default templates so that we can show the newly added field

Overriding the Metadata Wizard Template Page

Similar to what we have done before in the Templates directory, we will need to create "layouts" directory under "my_geonode/my_geonode/templates". This directory will contain a copy from "geonode/src/geonode/geonode/layers/templates/layouts/panels.html" as follows:

```

$ mkdir -p my_geonode/templates/layouts
$ cp ~/geonode/src/geonode/geonode/layers/templates/layouts/panels.html my_geonode/
→ templates/layouts/panels.html
$ vim my_geonode/templates/layouts/panels.html

```

Inside panels.html, we will add a new div with text input as follows:

```

{{ layer_form.data_quality_statement }}
</div>
<div>
    <span><label for="{{ layer_form.doi.id }}">{{ layer_form.doi.label }}</
→label></span>
    <input id="id_resource-doi" name="resource-doi"
        type="text"
        class="has-external-popover"
        data-container="body"
        data-content="a DOI will be added by Admin before publication." data-
→html="true" data-placement="right"
        placeholder="a DOI will be added by Admin before publication."
        value="{{ layer_form.doi.value }}">
    </div>
</div>

```

In addition, we will override the Layer Detail template page as follows:

```
mkdir -p my_geonode/templates/base

cp /home/geo/Envs/geonode/src/geonode/geonode/base/templates/base/_resourcebase_info_
panel.html my_geonode/templates/base/

vim my_geonode/templates/base/_resourcebase_info_panel.html
```

```
<dd><a href="/groups/group/{{ resource.group.name }}/activity/">{{ group }}</a> </dd>

<dt>DOI</dt>
<dd>{{ resource.doi }}</dd>

</dl>
```

Now from the layer details page, you can see the DOI metadata entry per layer

Title	low_res_test_dataset_areas64_id_91_2_0
License	Not Specified
Abstract	No abstract provided
Publication Date	Jan. 22, 2020, 9:50 p.m.
Type	Raster Data
Keywords	low_res_test_dataset_areas64_id_91_2_0, WCS, WorldImage
Regions	Global
Owner	admin
DOI	None

3- Create your own django app

In this section, we will demonstrate how to create and setup the skeleton of a custom app using the django facilities. The app will add a geocollections functionality to our GeoNode.

The Geocollections app allows to present in a single page, resources and users grouped by a GeoNode Group. We can assign arbitrary resources to a Geocollection, a Group and a name that will be also used to build a dedicated URL.

Note: Make sure to be inside “my_geonode” directory to execute the following commands

Create the django app

Django gives us an handy command to create apps. We already used startproject to create our geonode-project, now we can use startapp to create the app.

```
python manage.py startapp geocollections
```

This will create a folder named geocollections that contains empty models and views.

We need to add the new app to the INSTALLED_APPS of our project. inside “my_geonode/settings.py” line 54 change:

```
INSTALLED_APPS += (PROJECT_NAME,) to be: INSTALLED_APPS += (PROJECT_NAME,
↳ 'geocollections',)
```

Add a custom model

In this section, we will add a custom model and the related logic as follows:

- Add a new model
- Add urls and views
- Add admin panel
- Add the template

```
vim geocollections/models.py
```

```
from django.db import models

from geonode.base.models import ResourceBase
from geonode.groups.models import GroupProfile

class Geocollection(models.Model):
    """
    A collection is a set of resources linked to a GeoNode group
    """
    group = models.ForeignKey(GroupProfile, related_name='group_collections')
    resources = models.ManyToManyField(ResourceBase, related_name='resource_
↳ collections')
    name = models.CharField(max_length=128, unique=True)
    slug = models.SlugField(max_length=128, unique=True)

    def __unicode__(self):
        return self.name
```

At this point we need to ask django to create the database table. Django since version 1.8 has embedded migrations mechanism and we need to use them in order to change the state of the db.

Note: Make sure to be inside “my_geonode” directory to execute the following commands

```
python manage.py makemigrations

# the above command informs you with the migrations to be executed on the database

python manage.py migrate
```

Next we will use django generic view to show the collections detail. Add the following code in the views.py file:

```
vim geocollections/views.py
```

```
from django.views.generic import DetailView

from .models import Geocollection

class GeocollectionDetail(DetailView):
    model = Geocollection
```

Add url configuration

In order to access the created view we also need url mapping. We can create a `urls.py` file containing a url mapping to our generic view:

```
vim geocollections/urls.py
```

```
from django.conf.urls import url

from .views import GeocollectionDetail

urlpatterns = [
    url(r'^(?P<slug>[-\w]+)/$',
        GeocollectionDetail.as_view(),
        name='geocollection-detail'),
]
```

We also need to register the app urls in the project urls. So let's modify the “my_geonode” `urls.py` file adding the following:

```
vim my_geonode/urls.py
```

```
...
urlpatterns += [
    ## include your urls here
    url(r'^geocollections/', include('geocollections.urls')),
]
...
```

Enable the admin panel

We need a user interface where we can create geocollections. Django makes this very easy, we just need the `admin.py` file as follows:

```
vim geocollections/admin.py
```

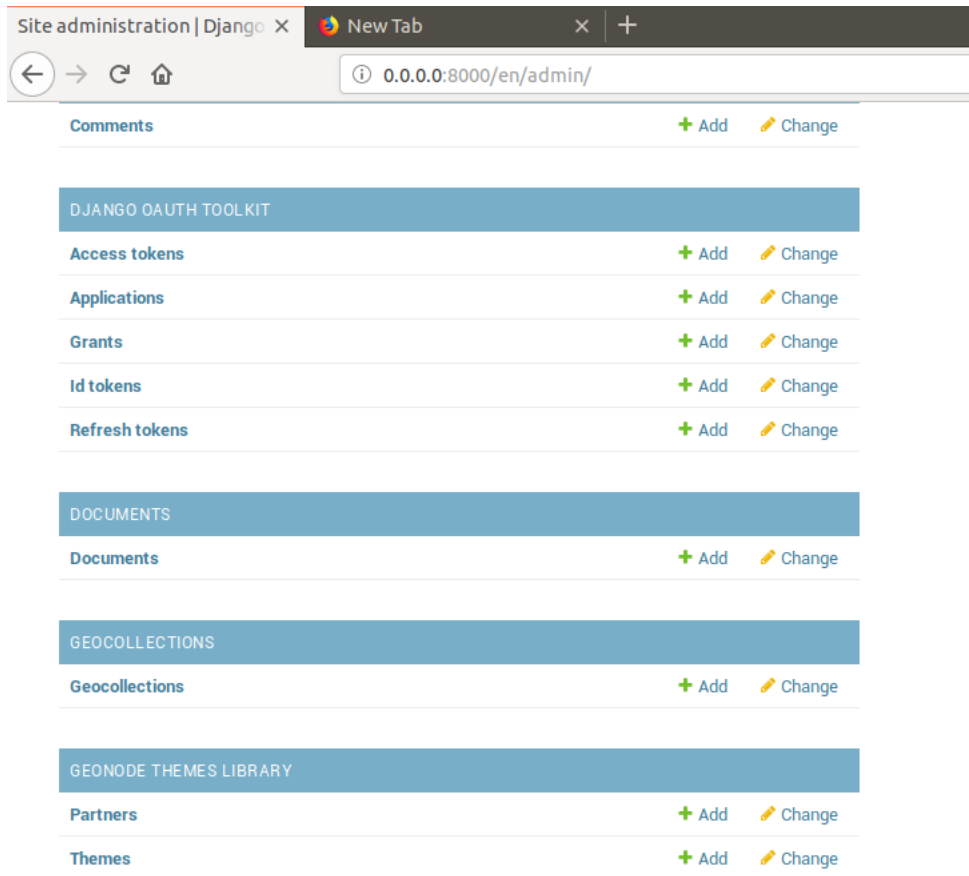
```
from django.contrib import admin

from .models import Geocollection

class GeocollectionAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("name",)}
    filter_horizontal = ('resources',)

admin.site.register(Geocollection, GeocollectionAdmin)
```

Now we can visit the admin page and create a geocollection from there as follows:



Adding the template

Now we need the template where the geocollection detail will be rendered. Let's create a geocollections directory inside the "my_geonode/templates" directory with a file named geocollection_detail.html:

```
mkdir -p my_geonode/templates/geocollections/

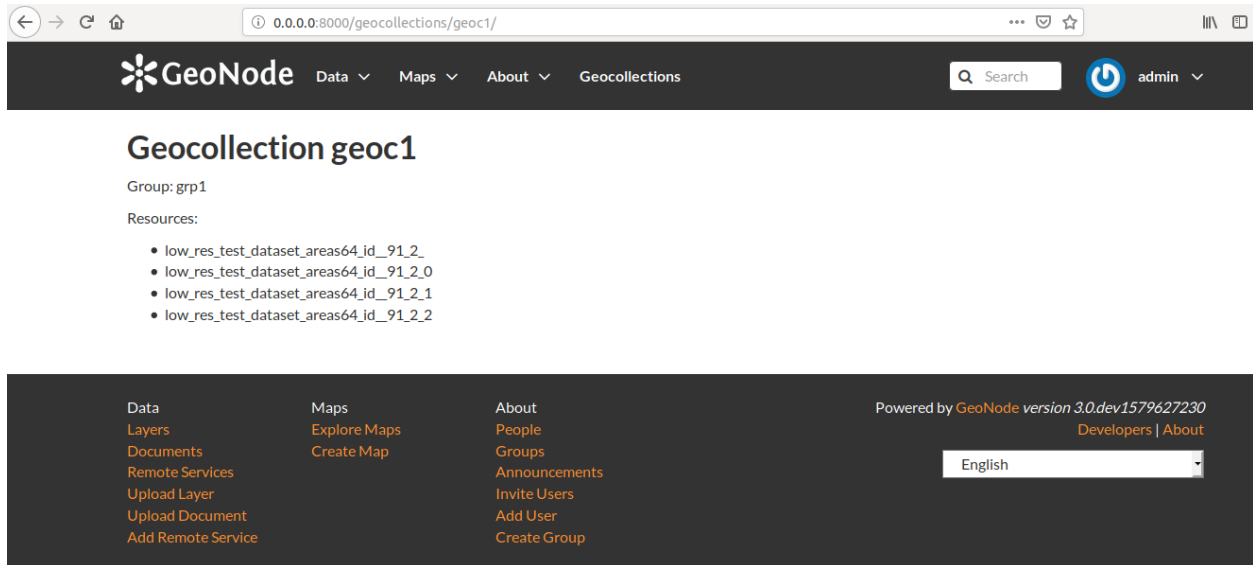
vim my_geonode/templates/geocollections/geocollection_detail.html
```

```
{% extends "geonode_base.html" %}
{% block body %}
    <h2>Geocollection {{ object.name }}</h2>
    <p>Group: {{ object.group.title }}</p>
    <p>Resources:</p>
    <ul>
        {% for resource in object.resources.all %}
            <li>{{ resource.title }}</li>
        {% endfor %}
    </ul>
{% endblock %}
```

To check the results, create a group in the geonode ui interface and load one or more layers/documents

login into the admin panel -> geocollections and create a geocollections

Visit <http://localhost:8000/geocollections/<the-name-of-the-created-geocollection>> and view the results.



Now you know how to customize an html template so you can tune this page as you prefer.

Permissions and APIs

In this section we will add some more advanced logic like permissions and APIs. The permissions in GeoNode are managed with django-guardian, a library which allow to set object level permissions (django has table level authorization).

The APIs are implemented through django-tastypie.

The topics to be covered include:

- Permissions on who can view the geocollection
- How to add templated and js to embed a permission ui in our geocollection detail page
- API to serve json serialized searchable endpoint

Permissions logic (permissions objects)

We need to add the permissions object to the database. We can do this by adding the following meta class to our Geocollection model, guardian will take care of creating the objects for us.

```
vim geocollections/models.py
```

```
class Meta:
    permissions = (
        ('view_geocollection', 'Can view geocollection'),
    )
```

Then run “python manage.py makemigrations” and “python manage.py migrate” to install them

Permissions logic (set_default)

Let’s add a method that will be used to set the default permissions on the Geocollections. We can add this logic to the Geocollection model but could also be a generic Mix-in similar to how it is implemented in GeoNode.

```
vim geocollections/models.py
```

```

from django.contrib.auth.models import Group
from django.contrib.auth import get_user_model
from django.contrib.contenttypes.models import ContentType
from django.conf import settings
from guardian.shortcuts import assign_perm

def set_default_permissions(self):
    """
    Set default permissions.
    """

    self.remove_object_permissions()

    # default permissions for anonymous users
    anonymous_group, created = Group.objects.get_or_create(name='anonymous')

    if settings.DEFAULT_ANONYMOUS_VIEW_PERMISSION:
        assign_perm('view_geocollection', anonymous_group, self)

    # default permissions for group members
    assign_perm('view_geocollection', self.group, self)

```

Permissions logic (methods)

Now we need a method to add generic permissions, we want to be able to assign view permissions to groups and single users. We can add this to our Geocollection model

```
vim geocollections/models.py
```

```

def set_permissions(self, perm_spec):
    anonymous_group = Group.objects.get(name='anonymous')
    self.remove_object_permissions()
    if 'users' in perm_spec and "AnonymousUser" in perm_spec['users']:
        assign_perm('view_geocollection', anonymous_group, self)
    if 'users' in perm_spec:
        for user, perms in perm_spec['users'].items():
            user = get_user_model().objects.get(username=user)
            assign_perm('view_geocollection', user, self)
    if 'groups' in perm_spec:
        for group, perms in perm_spec['groups'].items():
            group = Group.objects.get(name=group)
            assign_perm('view_geocollection', group, self)
def remove_object_permissions(self):
    from guardian.models import UserObjectPermission, GroupObjectPermission
    UserObjectPermission.objects.filter(content_type=ContentType.objects.get_for_
↪model(self),
                                object_pk=self.id).delete()
    GroupObjectPermission.objects.filter(content_type=ContentType.objects.get_for_
↪model(self),
                                object_pk=self.id).delete()

```

Permissions logic (views.py)

We can add now a view to receive and set our permissions, in views.py:

```
vim geocollections/views.py
```

```

import json
from django.core.exceptions import PermissionDenied
from django.http import HttpResponse
from django.contrib.auth import get_user_model

User = get_user_model()

def geocollection_permissions(request, collection_id):

    collection = Geocollection.objects.get(id=collection_id)
    user = User.objects.get(id=request.user.id)

    if user.has_perm('view_geocollection', collection):
        return HttpResponse(
            'You have the permission to view. please customize a template for this view'
            ↪,
            content_type='text/plain')

    if request.method == 'POST':
        success = True
        message = "Permissions successfully updated!"
        try:
            permission_spec = json.loads(request.body)
            collection.set_permissions(permission_spec)

            return HttpResponse(
                json.dumps({'success': success, 'message': message}),
                status=200,
                content_type='text/plain'
            )
        except:
            success = False
            message = "Error updating permissions :("
            return HttpResponse(
                json.dumps({'success': success, 'message': message}),
                status=500,
                content_type='text/plain'
            )

```

Permissions logic (url)

Lastly we need a url to map our client to our view, in urls.py

```
vim geocollections/urls.py
```

```

from django.conf.urls import url

from .views import GeocollectionDetail, geocollection_permissions

urlpatterns = [
    url(r'^(?P<slug>[-\w]+)/$',
        GeocollectionDetail.as_view(),
        name='geocollection-detail'),

    url(r'^permissions/(?P<collection_id>\d+)$',
        geocollection_permissions,

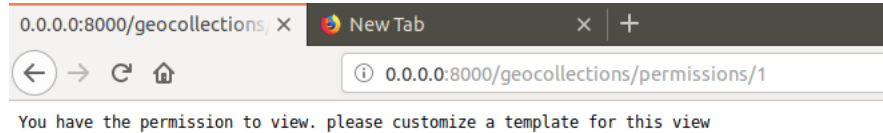
```

(continues on next page)

(continued from previous page)

```
name='geocollection_permissions'),
]
```

This url will be called with the id of the geocollection, the id will be passed to the view in order to get the permissions.



Warning: A note on the client part, the server side logic is just one part necessary to implement permissions.

A checklist of what is necessary:

- A template snippet that can be embedded in the geocollection_detail.html, you can copy and simplify: _permissions_form.html and _permissions.html (in geonode/templates)
- A javascript file that will collect permissions settings and send them to the server, you can copy and simplify: _permissions_form_js.html (in geonode/templates)

API

The GeoNode API system easily allows to plug in new APIs. This section demonstrates the required steps:

We need first to create an api.py file in our geocollection app.

```
vim geocollections/api.py
```

```
import json
from tastypie.resources import ModelResource
from tastypie import fields
from tastypie.constants import ALL_WITH_RELATIONS, ALL

from geonode.api.api import ProfileResource, GroupResource
from geonode.api.resourcebase_api import ResourceBaseResource

from .models import Geocollection
class GeocollectionResource(ModelResource):

    users = fields.ToManyField(ProfileResource, attribute=lambda bundle: bundle.obj.
    group.group.user_set.all(), full=True)
    group = fields.ToOneField(GroupResource, 'group', full=True)
    resources = fields.ToManyField(ResourceBaseResource, 'resources', full=True)

    class Meta:
        queryset = Geocollection.objects.all().order_by('-group')
        ordering = ['group']
        allowed_methods = ['get']
        resource_name = 'geocollections'
        filtering = {
            'group': ALL_WITH_RELATIONS,
            'id': ALL
        }
```

API authorization

We want the API to respect our custom permissions, we can easily achieve this by adding the following to the beginning of `api.py`:

```
vim geocollections/api.py
```

```
from tastypie.authorization import DjangoAuthorization
from guardian.shortcuts import get_objects_for_user

class GeocollectionAuth(DjangoAuthorization):

    def read_list(self, object_list, bundle):
        permitted_ids = get_objects_for_user(
            bundle.request.user,
            'geocollections.view_geocollection').values('id')

        return object_list.filter(id__in=permitted_ids)

    def read_detail(self, object_list, bundle):
        return bundle.request.user.has_perm(
            'view_geocollection',
            bundle.obj)
```

And this to the `GeocollectionResource` Meta class:

```
authorization = GeocollectionAuth()
```

Add a url for our API

In order to publish our API we need a url and we want that url to appear under the GeoNode's `/api` domain.

The final url for our API has to be `/api/geocollections`.

We can inject the url into the GeoNode API by adding the following lines to “`my_geonode/urls.py`” file:

```
vim my_geonode/urls.py
```

```
from geonode.api.urls import api

from geocollections.api import GeocollectionResource

api.register(GeocollectionResource())
```

And add the following in the `urlpatterns`:

```
url(r'', include(api.urls)),
```

The final result will be:

```
from django.conf.urls import url, include
from django.views.generic import TemplateView

from geonode.urls import urlpatterns

from geonode.api.urls import api
from geocollections.api import GeocollectionResource
```

(continues on next page)

(continued from previous page)

```
api.register(GeocollectionResource())

urlpatterns += [
    ## include your urls here
    url(r'', include(api.urls)),
    url(r'^geocollections/', include('geocollections.urls')),
]
```

Let's test permissions on API

We can test the permissions on API by manually set a permission from the command line and check that the API respects it.

With running `python manage.py shell` from inside our “my_geonode” folder, it opens a geonode shell.

A perm spec could look like this:

```
perms = {
    'users': {
        'AnonymousUser': ['view_geocollection'],
        'alessio': ['view_geocollection']}
}
```

and we can assign the permissions with:

```
from geocollections.models import Geocollection

Geocollection.objects.first().set_permissions(perms)
```

our `http://localhost:8000/api/geocollections` should now list the geocollection.

If you remove the ‘AnonymousUser’ line from perms and assign again the permissions it will disappear.

```
perms = {
    'users': {
        'alessio': ['view_geocollection']
    }
}
```

Deploy your GeoNode

So far we demonstrated how to modify, extend and style our GeoNode in dev mode but now it's time to go on production. In this section we will clarify how to:

- commit your work on GitHub
- setup your server
- setup your GeoNode for production

Push to GitHub It is always a good practice to keep your code in a remote repository, GitHub is one of the options and is indeed the most used.

It is assumed that you already have a GitHub account and that you have git installed and configured with your name and email.

We will push only the my_geonode folder to GitHub and as we knew earlier, GeoNode for us is a dependency and we'll just reinstall it as it is on the server.

Steps to push your code to GitHub:

- Create an empty repository in GitHub and copy it's address

- In my_geonode, run `git init` to initialize an empty repository
- Add your remote repository address with `git remote add yourname yourremoteaddress`
- edit `.gitignore` adding all `*.pyc` files
- `git add *` to add all content of my_geonode
- `git commit -m 'initial import'` to make the initial commit
- `git push yourname master` to push the code to the GitHub repository

Setup the server

There are several options for deploying GeoNode projects on servers. In this section, we explain how to deploy it on Ubuntu server 18.04 using system-wide installation

Note: For quick installation, follow the INSTALLING documentation at <http://docs.geonode.org/en/master/install/core/index.html>

Setup our my_geonode

We need now to install the developed “my_geonode” project following these steps:

- `git clone` from your repository (in the folder of your preference)
- `sudo pip install -e my_geonode`
- edit the settings where needed
- edit `/etc/apache2/sites-enabled/geonode.conf` replacing the wsgi path to the `my_geonode/my_geonode/wsgi.py` file
- add the apache rights to the “my_geonode” folder with a directory like

```
<Directory "/path/to/my_geonode/">
    Order allow,deny
    Require all granted
</Directory>
```

- Test your server.

This documentation helps developers to install GeoNode-Core and GeoNode-Project from different scenarios. GeoNode-Project can be installed on top of GeoNode-Core if already installed. Also GeoNode-Project can be installed from scratch as it has GeoNode-Core as a prerequisite.