GEOSERVER DEPLOYMENT
Best Practice Guide

Radek Furmánek, Lukáš Vojáček, Jakub Konvička, Jan Martinovič
Projects´ affiliation: Floreon+, RODOS, ESA Urban TEP
2022

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

GEOSERVER DEPLOYMENT:
Best Practice Guide

Introduction

This document deals with the deployment of GeoServer in Docker or docker-compose software. For the correct production deployment, it is important to follow a certain procedure. It includes preparing the environment, running docker with GeoServer, functionality testing and other dependencies such as setting up a database or proxy server.
Link to GIT:
https://code.it4i.cz/fur0016/best-practice-guides/-/blob/main/BPG_Geoserver/Best_Practice_Guides_for_GeoServer_EN.rst

1. Docker Solution

Docker is a software platform that allows to quickly build, test, and deploy applications. Docker packages software into standardized units called containers that hold everything the software needs to run, including libraries, system tools, and code. With Docker, you can quickly deploy and scale applications to any environment and be confident that the applications will run.

2. Preparing the server environment

For this case, the CentOS distribution, which is used in our infrastructure, was used. The first step is to update the OS. After that, the necessary software can be installed for the docker to run correctly.

Software installation

- yum-utils - a collection of tools and programs for managing yum repositories
- cifs-utils- file sharing tool
- docker - software see 1.
- unzip - decompression tool
- nano - tool for creating and reading text files
- midnight commander - (not important) tool for viewing directory structure

Enable ports

The default port where GeoServer runs is 8080 for https access is 8443. In our case is port redirect to 8444.

Example for firewalld:

```
sudo firewall-cmd --zone=public --permanent --add-port=8080/tcp

sudo firewall-cmd --reload
```

Set mount

In this case, the disk is mounted on a windows server. The mount is set in the fstab file localized in /etc/fstab. For a correct mount it is important to create a credentials file (geo.storage) with the username and password. On the windows server side, a share is set up for the folder where the geoserver data will be.

```
//geoshare.vsb.cz/docker_geodata /mnt/autofs/geoshare-geodata cifs
uid=1000,gid=1000,rw,nosuid,credentials=/root/geo.storage 0 0
```

geo.storage – sample (reference to this file has to be in attribute "credentials" in fstab):

```
username=usr_name
password=P4$$w0rd
domain=yourserver
```

Set up PostgreSQL

GeoServer communicates with the PostgreSQL database server where the data are stored. Therefore, it is necessary to set up database side access for the server where the GeoServer is located. This is contained in the pg_hba.conf configuration file.

Create keystore

The purpose of keystore is to protect the privacy and integrity of cryptographic keys using password-based algorithms. Privacy means that the keys are kept secret and can only be used by someone who knows the password. This is useful for private keys and secret keys. Integrity means that a change to the keys can be detected by someone who knows the password, this is useful for public keys and secret keys.
For create serverkeyW.pem and server.pem contact administrator of the server. If you are administrator of server, you can use public certificate provider. In our infrastructure we are using CESNET provider (only in CZ language) or SSLMarket or SSLS.

```
openssl pkcs12 -export -in server.pem -inkey serverkeyW.pem -out keystore.p12 -
name tomcat

keytool -importkeystore -deststorepass xxxpass -destkeypass xxxpass -
destkeystore keystore.jks -srckeystore keystore.p12 -srcstoretype PKCS12 -
srcstoretype PKCS12 -srcstorepass xxxpass -alias tomcat

chmod 770 keystore.jks

chown tomcat:tomcat keystore.jks
```

xxxpass – password for keystore.jks

3. Preparing docker-compose

For GeoServer it is necessary to have an image with Java jre8 and above, it is also important to have Ubuntu version 11 and above because of the GDAL library. GeoServer can use the ImageI/O Ext GDAL library to read selected coverage formats, for this it needs a local GDAL library installed on the machine. Geonode's distribution of GeoServer in the docker runs on Ubuntu 9 and because of this, the GDAL library cannot be installed. Therefore, the approach taken was to build our image with Java 8 and Ubuntu 20.04, where the GDAL library can be installed see B) Build image from Dockerfile.

A) Build geoserver.war

GeoServer is packaged as a standalone servlet for use with existing application servers such as Apache Tomcat or Jetty. In a normal GeoServer deployment, all that is required, is copy the geoserver.war file to the webapps directory of the application server and the application will be deployed. The default distribution of geoserver.war includes the gs-wms library, which is responsible for displaying GetCapabilities. However, when time layers are used, it returns all time domains in Capabilities, and disables the application's fast process for loading layers. Therefore, it is necessary to download the source version, and then comment part of the code in dimension and a build of a new GeoServer version.

After downloading the GeoServer source code, the Dimension Helper library is available at the address below, it has to be rebuild.

```
geoserver-
2.18.2\src\wms\src\main\java\org\geoserver\wms\capabilities\DimensionHelper
```

Comment on the lines

```
// Time dimension

//if (hasTime) {

// try {

// handleTimeDimensionVector(typeInfo);

// } catch (IOException e) {

// throw new RuntimeException("Failed to handle time attribute for layer", e);

// }

//}
```

Build

In folder src will run the commands:

```
sudo apt-get update

sudo apt-get install maven
```

```
mvn -DskipTests clean install

cd web/app/

mvn jetty:run

mvn install #(run this in the folder web/app/ and the folder geoserver.war will
be created in the target folder)
```

Copy geoserver.war to the prerequisites folder. A command has to be created for this path in the Dockerfile (B)Build image from Dockerfile, section in Dockerfile – Download and install GeoServer) to copy it to the tomcat folder, where the war file will unpack itself when the tomcat server starts.

B) Build image from Dockerfile

Below is an example of how to create a docker image to run GeoServer. It is possible to use a pre made docker image from GeoNode (https://github.com/GeoNode/geoserver-docker) and build from there. In the attached Dockerfile, the default image from Ubuntu:20.04 is used to create the new image, as well as a modified image from GeoNode.

- The Dockerfile must have install python3, python3-pip, python3-dev, for pip to work correctly
- Download and install GeoServer has to be rewritten to `COPY <PATH-PREREQUISITIES>/geoserver-${GEOSERVER_VERSION}.war /usr/local/tomcat`
- The geoserver.war file must be located where the Dockerfile is, or in another folder at the Dockerfile level
- Change paths to GeoNode prerequisites files, these scripts set up GeoNode authentication and run `entrypoint.sh`

```
COPY <PATH-PREREQUISITIES>/set_geoserver_auth.sh /usr/local/tomcat/tmp

COPY <PATH-PREREQUISITIES>/setup_auth.sh /usr/local/tomcat/tmp

COPY <PATH-PREREQUISITIES>/requirements.txt /usr/local/tomcat/tmp

COPY <PATH-PREREQUISITIES>/get_dockerhost_ip.py /usr/local/tomcat/tmp

COPY <PATH-PREREQUISITIES>/get_nginxhost_ip.py /usr/local/tomcat/tmp

COPY <PATH-PREREQUISITIES>/entrypoint.sh /usr/local/tomcat/tmp
```

It is also possible to add plugins (extensions) for the GeoServer. The version of the extension must match the GeoServer version.

- List of extensions from the documentation: https://geoserver.org/release/stable/

To create an image, run the `docker-compose build` command where the *docker-compose.yml*. file is stored.

The full Dockerfile is in the Docker file.

C) Create the docker-compose.yml file

A Compose file is a YAML file that defines services, networks, and volumes for a Docker application. Each GeoServer is run the same way, it just has different ports set than the others or a different data folder. All prerequisites and data needed for the GeoServer are mapped in the volumes.

Mapping prerequisities

```
<PATH-PREREQUISITIES>/marlin-0.9.3-Unsafe.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/marlin-0.9.3-Unsafe.jar

<PATH-PREREQUISITIES>/marlin-0.9.3-Unsafe-sun-java2d.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/marlin-0.9.3-Unsafe-sun-java2d.jar

<PATH-PREREQUISITIES>/floreonCors.jar:/usr/local/tomcat/lib/floreonCors.jar

<PATH-PREREQUISITIES>/server.xml:/usr/local/tomcat/conf/server.xml

<PATH-PREREQUISITIES>/web.xml:/usr/local/tomcat/webapps/geoserver/WEB-INF/web.xml

<PATH-PREREQUISITIES>/keystore.jks:/usr/local/tomcat/conf/keystore.jks

<PATH-PREREQUISITIES>/tomcat-context/tomcat1/context.xml:/usr/local/tomcat/conf/context.xml
```

- marlin.jar - Java2D RenderingEngine optimized for performance and better visual quality based on openjdk's pisces implementation. (Must be mapped to Java library folder)
- floreonCors.jar - cross-origin resource sharing, for development (path to Tomcat libraries)
- server.xml – http and port settings (path to Tomcat configuration directory)
- web.xml - settings CORS (path inside GeoServer)
- keystore.jks - for https, must be mapped to server.xml file (path to the Tomcat configuration directory)
- context.xml - JNDI connectors settings on DBs (path to Tomcat configuration directory, each GeoServer has its own according to JNDI connectors)

Environment settings

- *JAVA_HOME* – folder with JAVA
- *GEOSERVER_DATA_DIR* – folder where the data for GeoServer
- *GEOSERVER_REQUIRE_FILE* – path to global.xml
- *GEOWEBCACHE_CACHE_DIR* – path to GeoWebCache
- *GEOSERVER_LOG_LOCATION* – path to logging information from GeoServer
- *CATALINA_OPTS*

There are other parameters in *CATALINA_OPTS* that must also be set. These are in particular the paths for the Marlin libraries (-Xbootclasspath/a, -Xbootclasspath/p), maximum resources for the tomcat container (-Xms6G -Xmx6G).

The single-instance part of docker-composer is in the [docker-compose.yml]() folder.

4. Functionality testing

Various situations can occur when GeoServer is started, and a bad setting causes a malfunction. Testing and repair options are described below.
For run docker-compose, run `docker-compose up -d` command in the folder where docker-compose.yml file is stored.

1. Verify Apache Tomcat – The first step is to view the Apache Tomcat home page. It is usually displayed after typing the URL into the browser. e.g.: https://yourserver.cz:8444

- If it works and the Apache Tomcat home page is displayed with a picture of a cat, we can proceed to step 2.
- If it doesn't work, you need to try an address without security – just HTTP and port 8080 See: http://yourserver.cz:8080.
- If the address is working without security, the *keystore.jks* is probably wrong – it is necessary to check the validity of the certificate, the correct user, the read permissions, the path to the file, and the connector settings in server.xml – correct key=value values for https
- If http://yourserver.cz:8080 doesn't work either – you can try localhost, if that doesn't work either, Apache Tomcat is installed wrong – probably a mismatch between Apache Tomcat and JAVA version

2. Verify GeoServer – The first step is to view the GeoServer home page. It is usually displayed after entering the URL address into the browser. e.g.: https://yourserver.cz:8444/geoserver/web

- If it doesn't work, you need to look in the log by command `docker-compose logs`. There it usually says why GeoServer cannot be started. Focus on the wrong paths. In an emergency, you can find the problem by commenting out all paths and unpacking a clean GeoServer that is functional. By uncommenting the paths one by one, you can figure out the specific problem.
- If it works, the GeoServer environment is displayed. For the first login, the default user: *admin* and password: *geoserver* are set. After the first login, the site responds to the default login and asks for a change.

3. Verify GeoServer functionality – Loading GeoServer does not mean full functionality.

- Need to check the functionality of extensions: in the Server Status -> Modules sidebar – in the Table, find the Available? Enabled?, there must be a green check mark (check symbol)
- Set Disk Quota – JNDI connector definition on PostgreSQL database – used for GeoWebCache(GWC)
- Verify Authentication – sidebar panel Security -> Authentication – if an error pops up, the "Key authentication module" is missing
- Verify Data – correct connection to the database with data – sidebar panel Data -> Stores -> add new store. By adding a new Store, you can ensure proper connection to the database.
- Check available layers in the Data -> Layer Preview panel, open a layer for visualization (usually OpenLayers). Everything is fine in the view.

- Verify Capabilities – When you click on the GeoServer logo, Service Capabilities will appear on the right side. Clicking on WMS 1.1.1 or 1.3.0 will display the Capabilities for WMS. In the given Capabilities you need to find the required layers.

5. Server production deployment

For production deployment, a proxy server is used, which distributes the load to multiple GeoServers with different ports. Set ports to the proxy server for the server where the GeoServers are. The proxies tested are IIS, nginx and HAProxy.
WCF server is also used for building *service*–oriented applications. WCF server is used for authentication and authorization users who are registered in the system. Maintains user environment such as default layers, map viewports, time stamp, etc.

Contact

radek.furmanek@vsb.cz

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Bulgaria, Austria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, the United Kingdom, France, the Netherlands, Belgium, Luxembourg, Slovakia, Norway, Switzerland, Turkey, the Republic of North Macedonia, Iceland, and Montenegro. This project has received funding from the Ministry of Education, Youth and Sports of the Czech Republic (ID:MC2101).