

A summary of why GeoServer is an important factor in a GIS stack.

Why any GIS Application would need a Server

A GIS Application has to deal with huge data in different forms, layer – both raster and vector.

A typical mapping application gives you the tools, maps, and demographic data you need to analyze and understand how geography affects you and your business. Modern day computer systems, with ever increasing compute power and storage capabilities, have extended the use of mapping applications manifold.

When it comes to the deployment of mapping applications, we usually have two broad options.

1. **Desktop GIS:** It allows you to manipulate your map and all the manipulations locally in your desktop. It gives you the possibility to keep your data private. The examples include ArcMap, QGIS, etc.



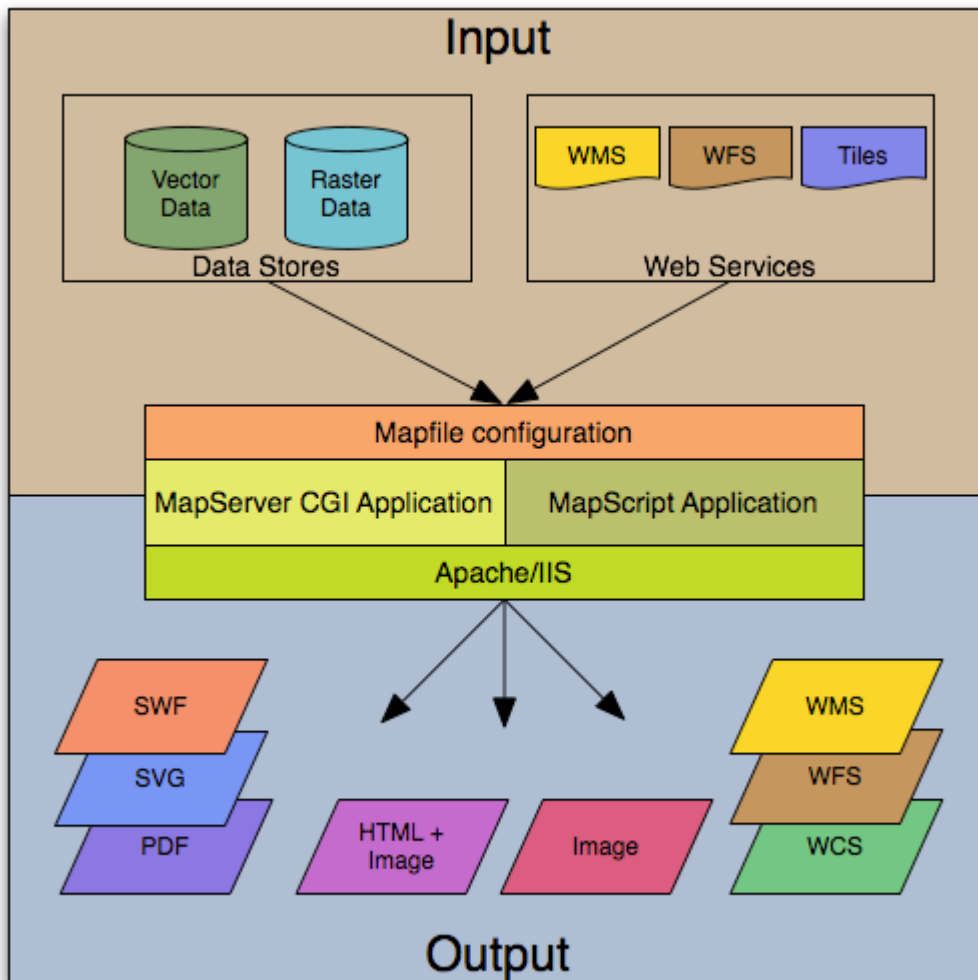
2. **GIS Map Server:** The second option is to use GIS Map server. While desktop application might be suitable for small time works, we need to understand that real world GIS applications require huge resources. They need to deal with hundreds of data formats, multiple rendering technologies, and serve different mapping requirements, while serving multiple teams simultaneously. This is not possible through standalone desktop GIS applications. This need is served by GIS Map Servers or simply Map Servers.



A GIS Map Server Ecosystem

In its most basic form, GIS Map Server is an application that sits inactive on your Web server. It gets active, when a request is sent to it. According to the request, it processes the mapping information stored in its associated databases and serves the mapping image to the client using one of the standard web services protocols like WMS, WFS, Tiles etc.

On the input side, it is closely coupled with the Data Servers and stores, which store the basic mapping data, sourced from various data sources. They may be online data sources or offline as well. And also, they may be vectors sources as well as raster also.



WMS (Web Map Service), WFS (Web Feature Service), and WCS (Web Coverage Service) are the web service standards from the Open Geospatial Consortium (OGC). These allow web clients to query and receive geographic information in the form of image, vector, or coverage data.

WMS is probably the best known of these three standards due to its widespread use by map servers to deliver map images.

On the output side, the rendered mapping output can be in any form as per the requirement. It can be a pdf form, an SWF video file, an SVG image file, an HTML page or simply a combination of image and a webpage.

Features of a Typical Mapping Server

Here we describe some of the basic features of a typical Mapping Server.

- A typical mapping server supports display and querying of hundreds of raster, vector, and database formats
- It has the ability to run on various operating systems (Windows, Linux, Mac OS X, etc.)
- It supports popular scripting languages and development environments (PHP, Python, Perl, Ruby, Java, .NET)
- It allows on-the-fly projections of maps in different projection systems
- It ensures high quality rendering of maps in various formats as per user requirements
- The output is usually fully customizable

(Ref.: <https://mapserver.org/introduction.html>)

Advantage of Using a GIS Map Server

With the help of Internet, clients can access the geospatial information over the web regardless of the fact that the server and client might be far from each other. Map Servers introduces distinct advantages over traditional desktop GIS, including the following:

Applications:

Unlike desktop GIS, which is limited to a certain number of GIS professionals, Map Servers can be used by everyone in an organisation as well as the public at large. This broad audience has diverse demands. Applications such as mapping street road, locating places to tag personal photos, tracing friends and displaying WiFi hotspots are a few examples.

Better cross-platform capability:

Most of the GIS Map Server clients are web browsers like Internet Explorer, Google Chrome, Mozilla Firefox, Apple Safari etc. Because these web browsers are compatible with HTML and JavaScript standards, MapServers that relies on HTML clients will typically support different operating systems such as Microsoft Windows, Linux, and Apple Mac OS.

GIS Map Servers over web have world-wide reach:

The geographic data and maps can be presented to the world through GIS MapServers. Anybody locating anywhere in the world can access the geographical information from their computers, desktop or mobile devices.

Supports multiple users simultaneously:

In general, a traditional desktop supporting GIS application software is used by only one user at a time, while a MapServers can be used by dozens or hundreds of users simultaneously. Thus, GIS MapServers provide much higher performance and scalability than single desktop GIS.

Low cost as averaged by the number of users:

The vast majority of content on the internet is free of charge to end users, and this also applies on GIS MapServers. Generally, users do not need to buy software or pay to use GIS MapServers service. Organizations that need to provide GIS service to individual users can also minimize their costs through GIS MapServers. Instead of buying and setting up desktop GIS for every user, an organization can set up just one MapServers, and this single system can be shared by many users: from home, at work, or in the field.

User-friendly:

Desktop GIS is intended for professional users with months of training and experience in GIS. MapServers is intended for a broad audience, including public users who may know nothing about GIS.

Automatic updates:

For desktops, GIS to be updated to a new version, the update needs to be installed on every computer. For MapServers, one update works for all clients.

More about GeoServer

- GeoServer is an open-source GIS Map Server written in Java that allows users to share, process and edit geospatial data.
- GeoServer conforms to the Web Feature Service (WFS) standard, and Web Coverage Service (WCS) standard which permits the sharing and editing of the data that is used to generate the maps.

- It also uses the Web Map Tile Service standard to split your published maps into tiles for ease of use by web mapping and mobile applications.
- It is a modular application with additional functionality added via extensions.

Essentially, the above write-up summarizes, why GeoServer is an important factor of a GIS stack.

A summary of what Docker is and why Docker should be used to containerize GeoServer

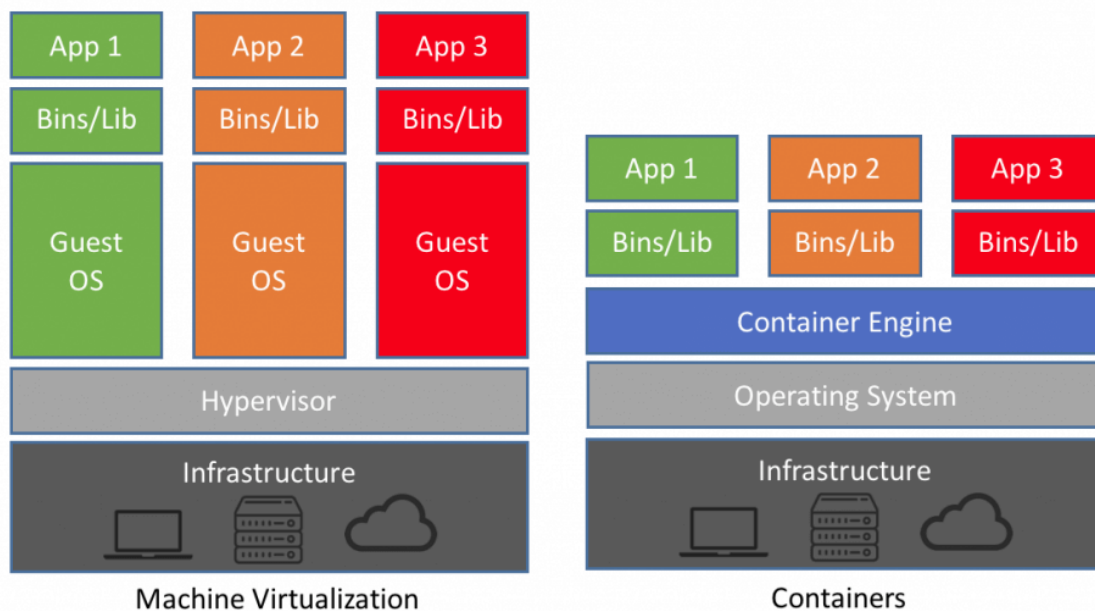
I visualize Docker as an extension of the concept of Virtual Machine in the sense that both provides a user the ability to use the host environment to different other use cases. For example you can run a Linux machine within a Windows environment.

However, there are a lot of difference between them.

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code. [Source: <https://opensource.com/resources/what-docker>]

In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

The difference between Docker and a Virtual Machine can be easily understood from the following pictorial diagram.



Source: <https://www.guru99.com/docker-tutorial.html>

What is Docker Container

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

(Source: <https://www.docker.com/resources/what-container>)

So essentially, to run GeoServer on a Windows machine, you just have to load Docker on the machine, pull a suitable GeoServer image hub from Docker hub and run it within a container. To manage the data, you might use a spatial database like PostGIS. PostgreSQL already come bundled with PostGIS. It hardly takes a set of few commands to establish the setup.

Why to run containerize GeoServer

There are many advantages of running a specialized Web Server Application like GeoServer in a containerized environment like Docker.

Mapping applications usually require a lot of data churning and are quite resource consuming as compared to simple Desktop applications. Often they operate in multi-user environment and

a single standalone desktop application might not suffice the requirements. Hence applications GeoServer, fill that requirement.

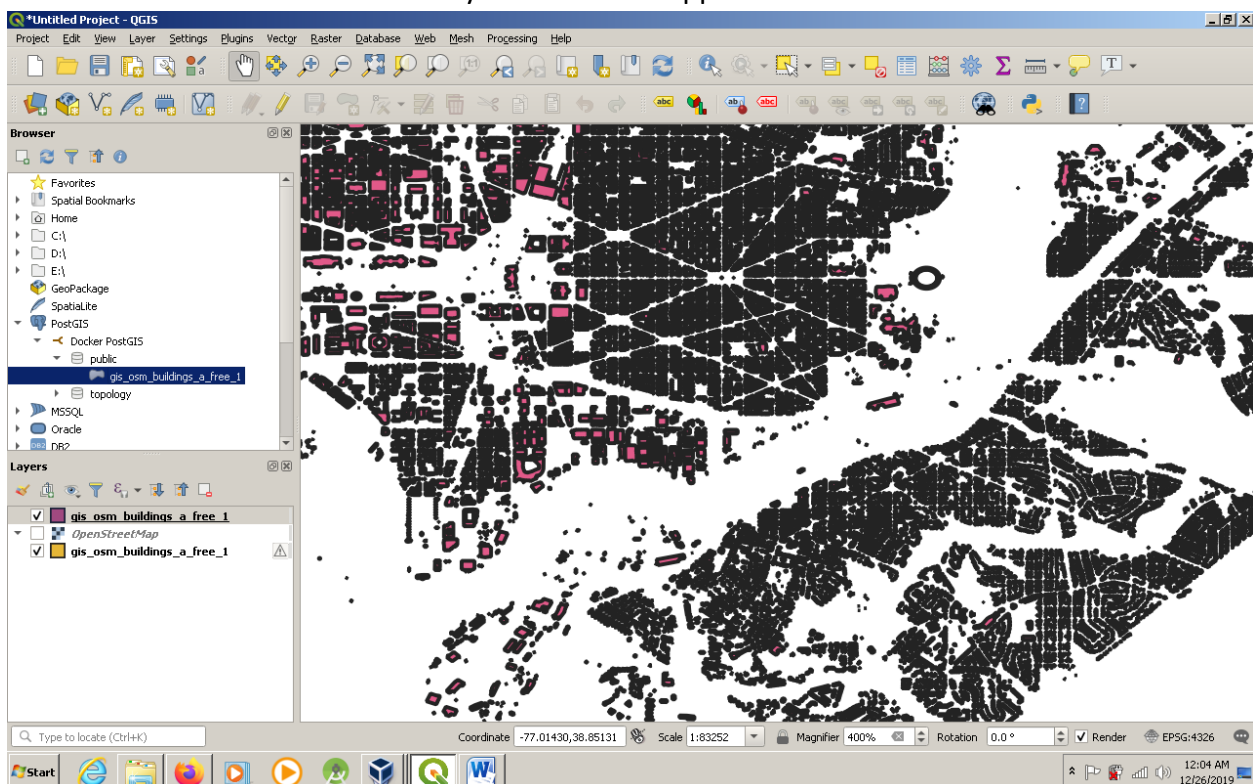
But, in case the requirement is to share the hardware resources with other applications also, then you may not like to invest in a dedicated hardware and so the techniques of virtualization have to come in. Docker provides a lot of advantages as compared to typical Virtual Machines, which have been discussed in the previous sections.

Hence, running a containerized GeoServer in a Docker is the best possible combination to minimize the investment and also optimally utilize the available resources.

My Experience with Docker

My newer machine running Windows 10, had recently got crashed and is not yet up and running. For the past few days, I have been working on my old machine, which is a Windows 7 machine, with a meagre 4 GB of RAM. After a lot of hesitation, I have loaded Docker, PostGIS and also GeoServer on this Windows 7 machine and have been able upload a new layer successfully.

It gave me huge surprise that the Docker was able to run the containers having both PostGIS as well as GeoServer while the Windows 7 machine was also running other applications – it talks a lot about how efficiently these applications utilize the resources.



Installation of Docker on local machine

I am working on an older machine, which does not support the installation of Docker for Windows. So, I had to install Docker Toolbox. I downloaded Docker Toolbox in D:// drive.

Videos	android-studio-ide-191.6010548-windows32	12/13/2019 3:12 PM	Compressed (zippe...	738,359 KB
	district-of-columbia-latest-free.shp	12/19/2019 11:41 PM	Compressed (zippe...	25,085 KB
	DockerToolbox-19.03.1	12/15/2019 9:42 AM	Application	236,837 KB
	Firefox Setup 60.0.1	5/22/2018 11:54 AM	Application	38,063 KB
Computer	Local Disk (C:)			
	Local Disk (D:)			
Network	inkscape-0.92.4-x64	11/14/2019 8:50 PM	Windows Installer P...	83,255 KB
	npp.7.8.1.Installer.x64	11/14/2019 8:16 PM	Application	3,890 KB
	osgeolive-13.0-amd64	12/25/2019 10:28 PM	Virtual Machine Disk...	12,375,744...
	osnervive-13.0-amd64.vmdk.7z	11/14/2019 9:45 PM	77 File	3,335,460 KB

Then I ran the installer to install the Docker in my already existing Oracle VM VirtualBox.

[illegible]

1. **Docker --version:** This command is used to get the currently installed version of Docker.

2. Docker ps: This command is used to list the running containers.

```
mingw64:/c:/Program Files/Docker Toolbox
mkgarg@DLBRNG010211-Home MINGW64 /c:/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        STATUS               IMAGE                  PORTS                COMMAND              NAMES              CREATED
9b28914f5ef6        Up 2 hours           kartoza/geoserver      0.0.0.0:8080->8080/tcp  "/scripts/entrypoint" geoserver          8 days ago
97d5109b5ab9        Up 2 hours           kartoza/postgis:9.4-2.1 5432/tcp             "/bin/sh -c /start-p" postgis            8 days ago
mkgarg@DLBRNG010211-Home MINGW64 /c:/Program Files/Docker Toolbox
$
```

As can be seen, right now my Docker instance is running kartoza/postgis:9.4-2.1 container and also kartoza/geoserver, whose name I have given as “PostGIS” and “geoserver” respectively. The Docker gives every container a unique Container Id. My Docker has given them the container id as 97d5109b5ab9 and 9b28914f5ef6 respectively.

3. Docker ps -a: If we give Docker ps command with flag -a, then it gives us a list of all the containers, whether they are in running state or not.

```
mingw64:/c:/Program Files/Docker Toolbox
mkgarg@DLBRNG010211-Home MINGW64 /c:/Program Files/Docker Toolbox
$ docker ps -a
CONTAINER ID        STATUS               IMAGE                  PORTS                COMMAND              NAMES              CREATED
9b28914f5ef6        Up 2 hours           kartoza/geoserver      0.0.0.0:8080->8080/tcp  "/scripts/entrypoint" geoserver          8 days ago
97d5109b5ab9        Up 2 hours           kartoza/postgis:9.4-2.1 5432/tcp             "/bin/sh -c /start-p" postgis            8 days ago
mkgarg@DLBRNG010211-Home MINGW64 /c:/Program Files/Docker Toolbox
$
```

As can be seen from the above output, my Docker is having two containers – one is PostGIS and the other one is geoserver. The status of both the containers is “Up for 2 hours”.

4. Docker images: if you want to know what images are already present on your Docker setup locally, you can give this command.

```
MINGW64:/c/Program Files/Docker Toolbox
mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
kartoza/geoserver    latest             b56fd9201710       7 weeks ago
1.36GB
kartoza/postgis      9.4-2.1            24a1421822f9       2 years ago
495MB
mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$
```

This tells us the images, which we have downloaded in the Docker.

5. Docker stop: This command is used to stop the container from running.

```
MINGW64:/c/Program Files/Docker Toolbox
mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
kartoza/geoserver    latest             b56fd9201710       7 weeks ago
1.36GB
kartoza/postgis      9.4-2.1            24a1421822f9       2 years ago
495MB
mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$ docker stop geoserver
geoserver
mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$
```

Here I have stopped the container geoserver.

6. Docker pull: This command is used to download image file from the Docker hub repository. For example, to pull PostGIS image from Docker hub repository, the following commands can be used.

```
docker pull kartoza/geoserver
```

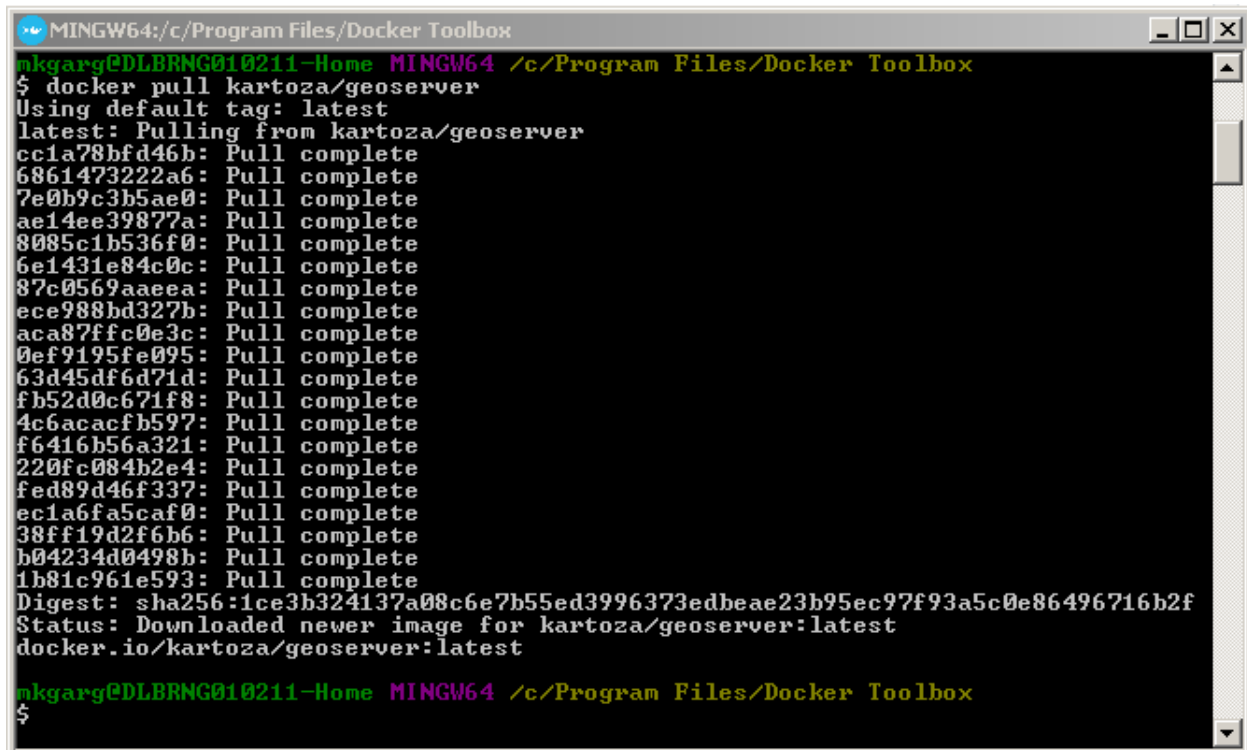
Creating the container

Now, for the installation of the geoserver, following commands were used:

Pull the image of GeoServer from Docker hub

Following command was used for pulling the geoserver image:

docker pull kartoza/geoserver



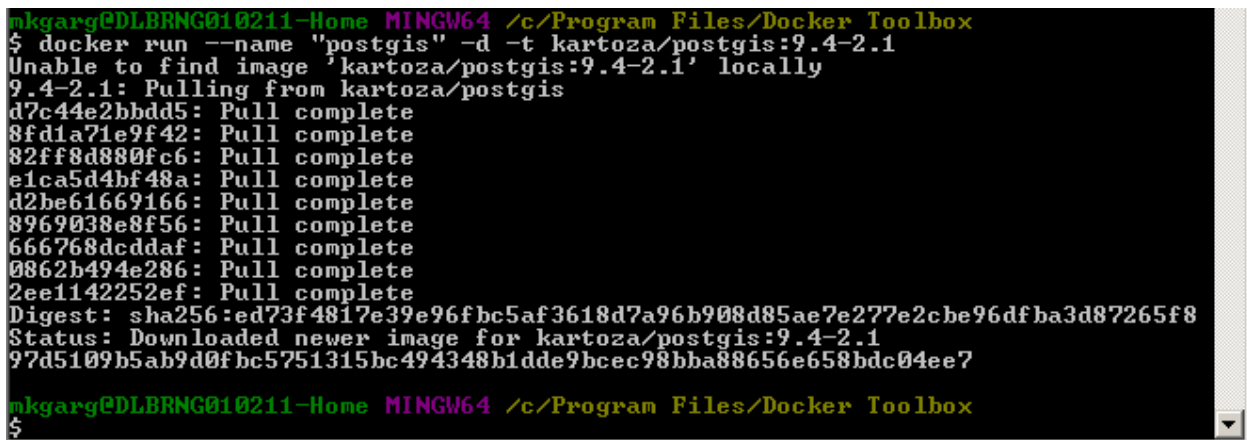
```
mingw64:/c:/Program Files/Docker Toolbox
$ docker pull kartoza/geoserver
Using default tag: latest
latest: Pulling from kartoza/geoserver
cc1a78bfd46b: Pull complete
6861473222a6: Pull complete
7e0b9c3b5ae0: Pull complete
ae14ee39877a: Pull complete
8085c1b536f0: Pull complete
6e1431e84c0c: Pull complete
87c0569aaeea: Pull complete
ece988bd327b: Pull complete
aca87ffc0e3c: Pull complete
0ef9195fe095: Pull complete
63d45df6d71d: Pull complete
fb52d0c671f8: Pull complete
4c6acacfb597: Pull complete
f6416b56a321: Pull complete
220fc084b2e4: Pull complete
fed89d46f337: Pull complete
ec1a6fa5caf0: Pull complete
38ff19d2f6b6: Pull complete
b04234d0498b: Pull complete
1b81c961e593: Pull complete
Digest: sha256:1ce3b324137a08c6e7b55ed3996373edbeae23b95ec97f93a5c0e86496716b2f
Status: Downloaded newer image for kartoza/geoserver:latest
docker.io/kartoza/geoserver:latest

mingw64:/c:/Program Files/Docker Toolbox
$
```

Create and start the container containing PostGIS

To run GeoServer, you need to first create a PostGIS database and start the container containing the PostGIS. Following command was used for this purpose:

```
docker run --name "PostGIS" -d -t kartoza/postgis:9.4-2.1
```



```
mingw64:/c:/Program Files/Docker Toolbox
$ docker run --name "postgis" -d -t kartoza/postgis:9.4-2.1
Unable to find image 'kartoza/postgis:9.4-2.1' locally
9.4-2.1: Pulling from kartoza/postgis
d7c44e2bbdd5: Pull complete
8fd1a71e9f42: Pull complete
82ff8d880fc6: Pull complete
e1ca5d4bf48a: Pull complete
d2be61669166: Pull complete
8969038e8f56: Pull complete
666768dcddaf: Pull complete
0862b494e286: Pull complete
2ee1142252ef: Pull complete
Digest: sha256:ed73f4817e39e96fbc5af3618d7a96b908d85ae7e277e2cbe96dfba3d87265f8
Status: Downloaded newer image for kartoza/postgis:9.4-2.1
97d5109b5ab9d0fbc5751315bc494348b1dde9bcec98bba88656e658bdc04ee7

mingw64:/c:/Program Files/Docker Toolbox
$
```

After the container was created and then started, I used few commands like `docker ps` to display that the container is properly running.

Create and start the container containing geoserver

Following command was used for this purpose:

```
docker run --name "geoserver" --link postgis:postgis -p 8080:8080 -d -t kartoza/geoserver
```

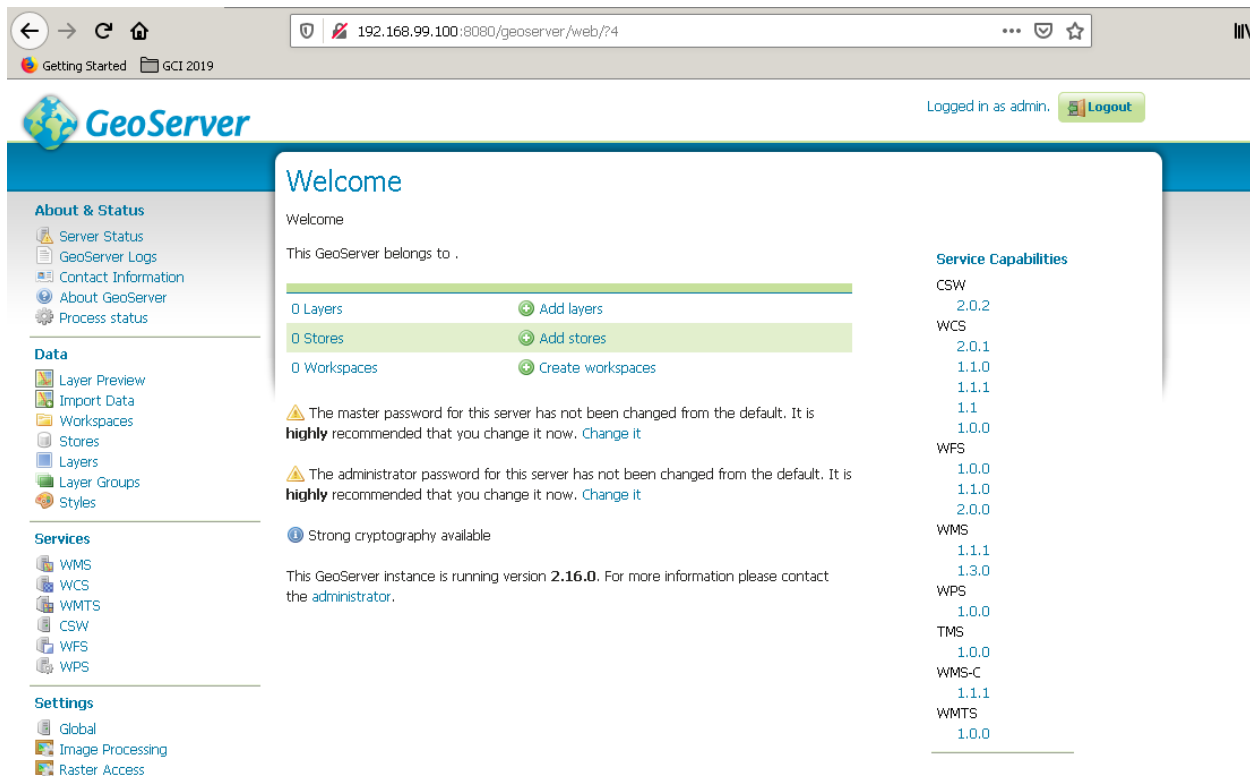
```
mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name "geoserver" --link postgis:postgis -p 8080:8080 -d -t kartoza/geoserver
9b28914f5ef695e6f92c0c88c3bea98440a53353062b8e0b34096c20e766a8ea
mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$
```

Open GeoServer Web Interface

Once both the containers are up and running, we have to start the web interface of Geoserver. Open your browser and go to the following URL

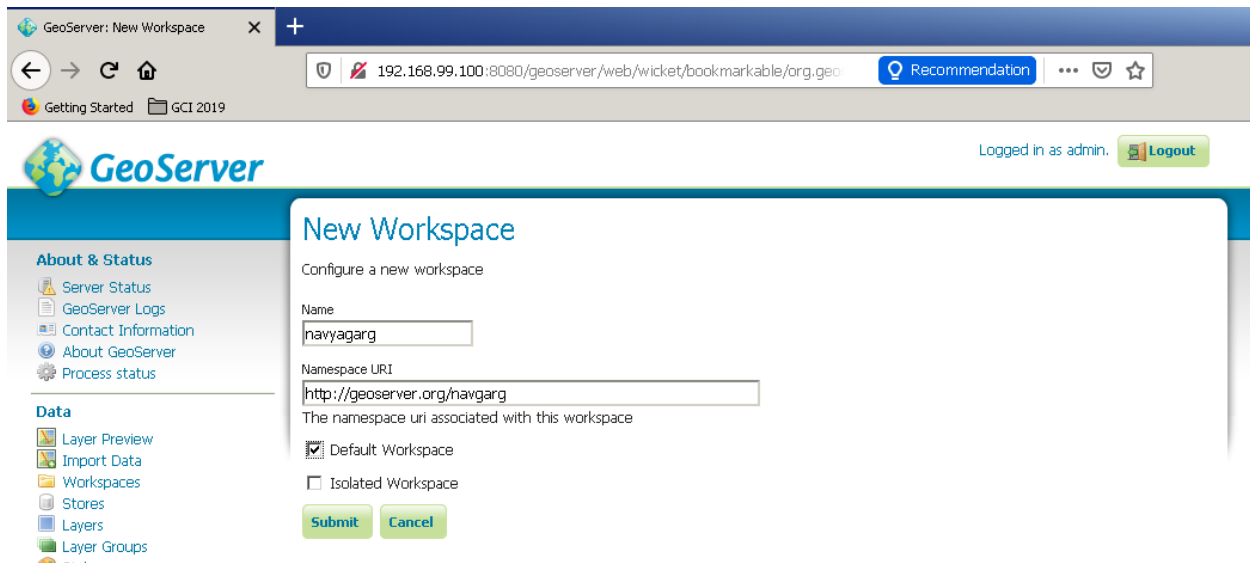
<http://192.168.99.100:8080/geoserver>

192.168.99.100 is the IP assigned to the Docker and 8080 is the port on which GeoServer functions.



Create Workspace

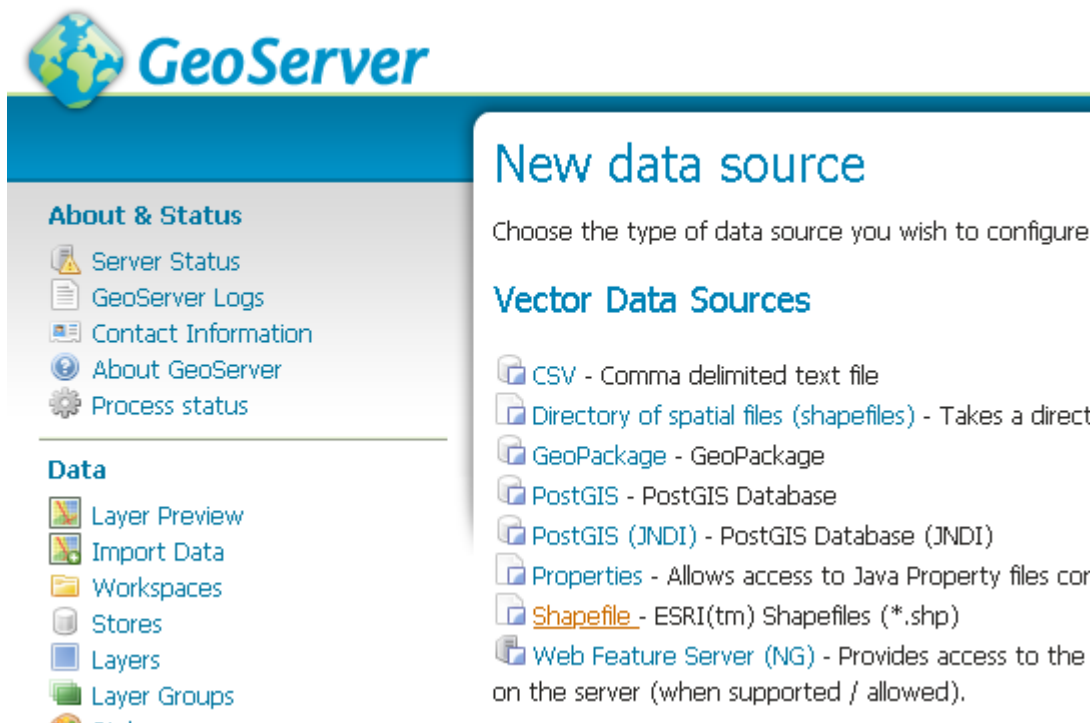
Click on Workspace link in the left sidebar and then click on Add New Workspace.

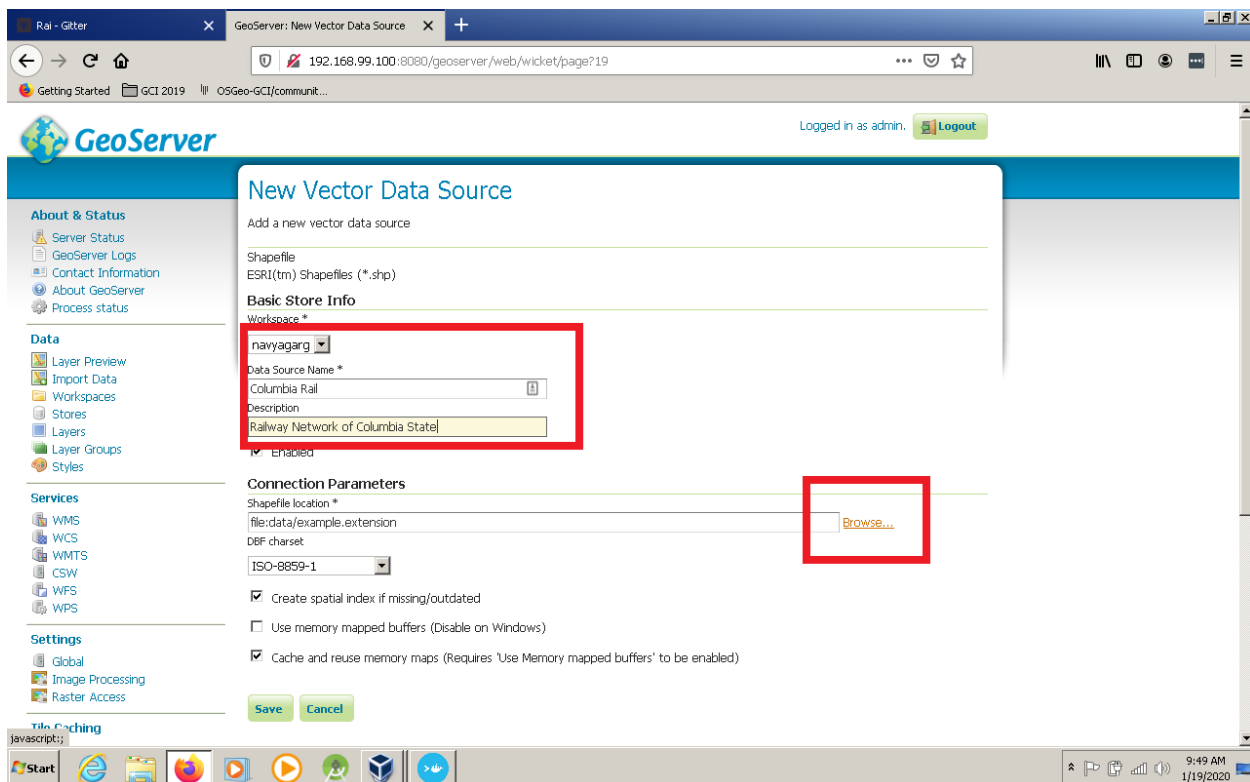


Here I have created a new workspace with my name navyagarg.

Create Store

Click on Store link in the left sidebar and then click on Shapefile link (as the data source, which I am going to add contains vector data.)





Click on the Browse link and browse to the shapefile, in which your data is stored.

Shapefile location		
lib64/	Nov 27, 2019 4:05 PM	
media/	Apr 26, 2018 12:00 AM	
mnt/	Apr 26, 2018 12:00 AM	
opt/	Nov 27, 2019 4:09 PM	
proc/	Jan 19, 2020 3:53 AM	
root/	Jan 11, 2020 3:14 AM	
run/	Apr 26, 2018 12:00 AM	
sbin/	Apr 26, 2018 12:00 AM	
scripts/	Nov 27, 2019 4:08 PM	
srv/	Apr 26, 2018 12:00 AM	
sys/	Jan 19, 2020 3:53 AM	
tmp/	Jan 19, 2020 3:53 AM	
usr/	Apr 26, 2018 12:00 AM	
var/	Apr 26, 2018 12:00 AM	
railways.shp	Dec 18, 2019 11:25 PM	154.1K

For demonstration purpose, I have downloaded the Railway Map of Columbia State from OpenStreetData and saved the shapefile in the home directory of Docker.

```

mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$ ls
boot2docker.iso          docker-start.cmd          start.sh*
docker.exe*              gis_osm_railways_free_1.shp  unins000.dat
docker-compose.exe*      gis_osm_waterways_free_1.shp  unins000.exe*
docker-machine.exe*      installers/
docker-quickstart-terminal.ico kitematic/

mkgarg@DLBRNG010211-Home MINGW64 /c/Program Files/Docker Toolbox
$ docker cp gis_osm_railways_free_1.shp 9b28914f5ef6:/railways.shp

```

As you can see in the above screenshot, I have downloaded the file in Windows and it was stored in the path c://Program Files/Docker Toolbox.

From there, I copied this file (file name is 'gis_osm_railways_free_1.shp') to home folder of geoserver container using the following command:

```
docker cp gis_osm_railways_free_1.shp 9b28914f5ef6:/railways.shp
```

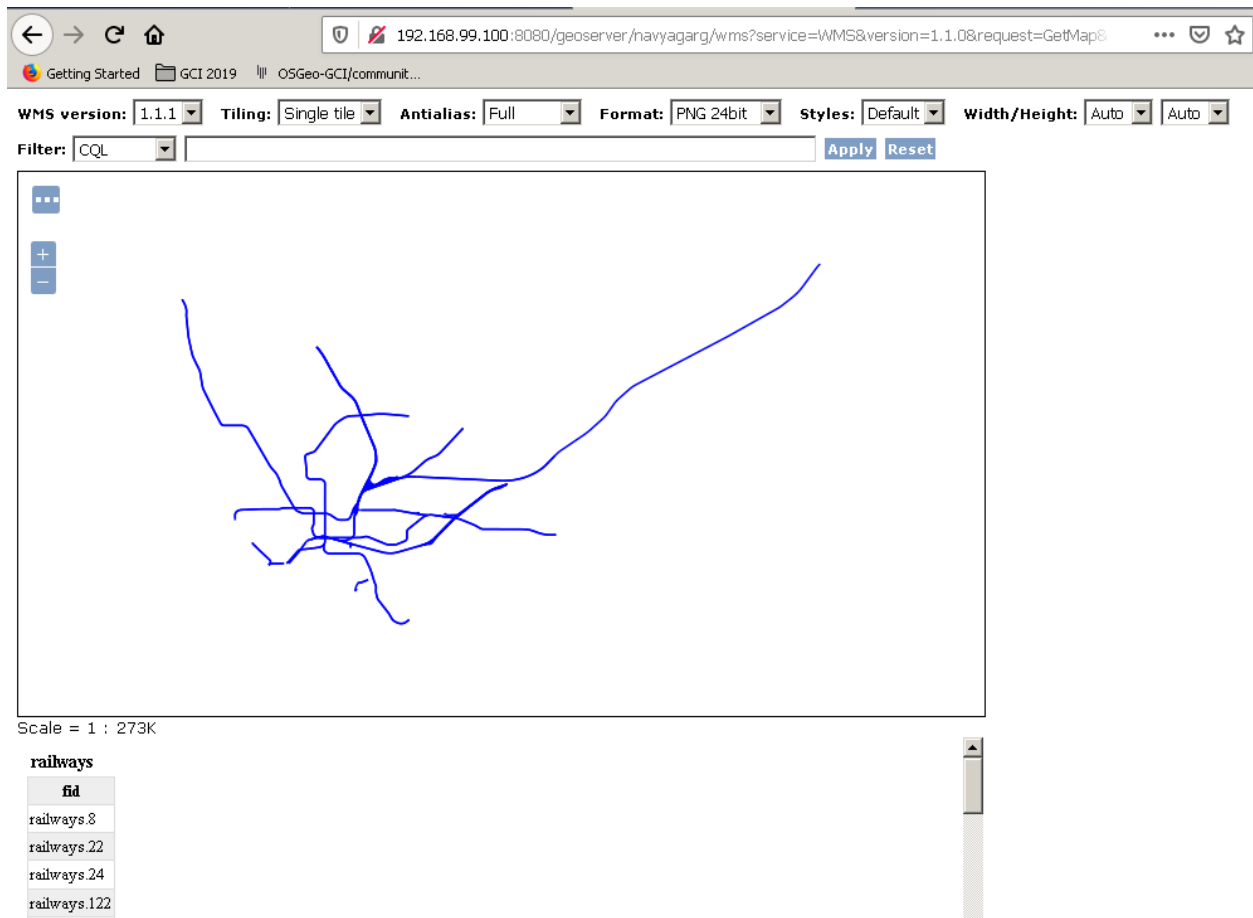
Here, 9b28914f5ef6 is the container ID of the geoserver container. This command also renamed the file to railways.shp

Publish the Layer

The screenshot shows the GeoServer web interface in a browser window. The browser's address bar displays the URL `192.168.99.100:8080/geoserver/web/wicket/page?20`. The page title is "New Layer". On the left sidebar, there are sections for "About & Status", "Data", "Services", "Settings", and "File Caching". The "Data" section is expanded, showing a list of resources in the store "Columbia Rail". The list has a header row with "Published", "Layer name", and "Action". Below the header, there is one row with the value "railways" under "Layer name" and a "Publish" button under "Action". The "Publish" button is highlighted with a red rectangle. The "Action" column header is also highlighted with a red rectangle. The "Published" column header is highlighted with a green background. The "Layer name" column header is highlighted with a light green background. The "railways" row is highlighted with a light green background. The "Publish" button is highlighted with a red rectangle. The "Action" column header is highlighted with a red rectangle. The "Published" column header is highlighted with a green background. The "Layer name" column header is highlighted with a light green background. The "railways" row is highlighted with a light green background. The "Publish" button is highlighted with a red rectangle. The "Action" column header is highlighted with a red rectangle.

Preview Layer

Once the layer is published, you can preview it using the Layer Preview link in the left toolbar.



Now, we have successfully uploaded GeoServer on Docker and added a layer locally!

That's all! Thank You!