

Basic Descriptive Works

A Summary of why PostgreSQL + PostGIS is an important factor in a GIS stack.

PostgreSQL



PostgreSQL is a general purpose and object-relational database management system, the most advanced open source database system. PostgreSQL (or postgres, for short) is a *database server*, meaning it is a program that accepts connections from one or more *clients*, and allows them to interact with the data the server manages.

PostgreSQL was developed based on POSTGRES 4.2 at Berkeley Computer Science Department, University of California. PostgreSQL was also designed to be portable so that it could run on various platforms such as Mac OS X, Solaris, and Windows.

PostgreSQL is free and open source software. Its source code is available under PostgreSQL license, a liberal open source license. You are free to use, modify and distribute PostgreSQL in any form.

Note: PostgreSQL requires very minimum maintained efforts because of its stability. Therefore, if you develop applications based on PostgreSQL, the total cost of ownership is low in comparison with other database management systems.

PostgreSQL features

- **User-defined types**
- **Table inheritance**
- **Sophisticated locking mechanism**
- **Foreign key referential integrity**
- **Views, rules, subquery**
- **Nested transactions.**
- **Multi-version concurrency control (MVCC)**
- **Asynchronous replication**

Purpose to use PostgreSQL+ PostGIS

PostGIS has become the canonical example of a extremely massive advanced Extension that helps PostgreSQL stay standard and protrusile. Also PostgreSQL does not support Spatial data types. It supports geometric types.

- have no Spatial Reference System ID/SRSID.
- can never be re-projected.
- even on a spheroid, they provide very limited functionality.
- They're not standardized.
- They don't provide GIST indexes.

PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.

PostGIS offers many features rarely found in other competing spatial databases such as Oracle Locator/Spatial and SQL Server.

Note : PostGIS adds extra types (geometry, geography, raster and others) to the PostgreSQL database. It also adds functions, operators, and index enhancements that apply to these spatial types. These additional functions, operators, index bindings and types augment the power of the core PostgreSQL DBMS, making it a fast, feature-plenty, and robust spatial database management system.

POSTGIS



PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.

PostGIS Features

- ☐ Processing and analytic functions for both vector and raster data for splicing, dicing, morphing, reclassifying, and collecting/unioning with the power of SQL .
- ☐ raster map algebra for fine-grained raster processing.
- ☐ Spatial reprojection SQL callable functions for both vector and raster data.
- ☐ 3D object support, spatial index, and functions
- ☐ Network Topology support

Note : In Short PostGIS is an *extension* that is installed on top of PostgreSQL and gives the database server the means to store and manipulate spatial data, as well as perform spatial calculations and analysis. PostGIS can perform spatial operations on both vector and raster data.

``All the above topics clearly shows that PostgreSQL + PostGIS is an important factor in a GIS stack.``

A Summary of what Docker is and why Docker should be used to containerise PostgreSQL (with PostGIS)

DOCKER

Docker: The Modern Platform for High-Velocity Innovation. Docker is a set of platforms as a service product that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

DOCKER'S FEATURES

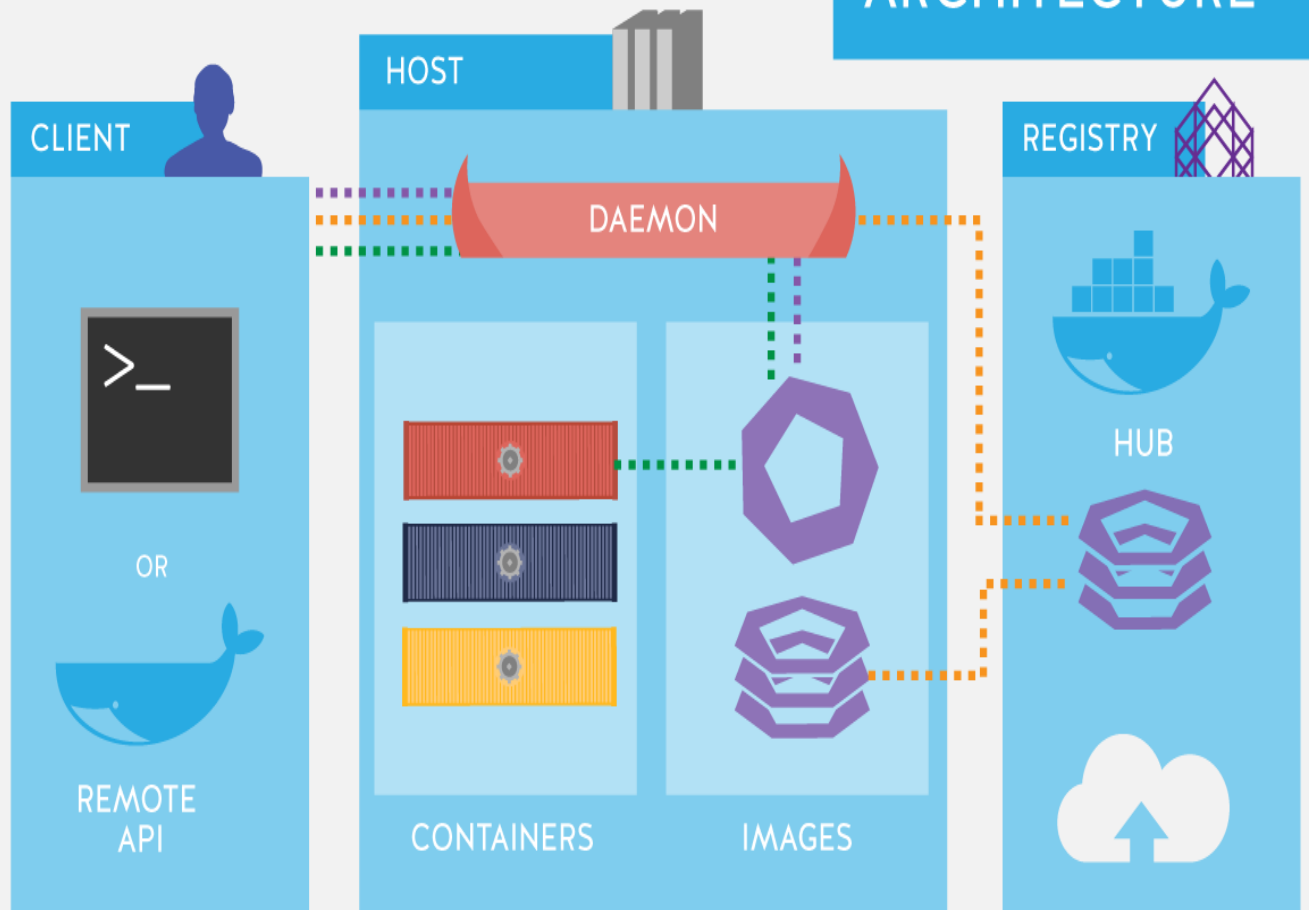
- Easy and Faster Configuration.
- Increase productivity.
- Application Isolation.
- Swarm.
- Routing Mesh.
- Services.
- Security Management.

WHY DOCKER ?

Docker is a beast, it can be remotely daunting for incipient users who aren't habituated with the command-line, but it provides you with an expedient to build and run software in a very consistent and controlled way by building upon a technology called LXC containers. we can download and run software like Postgres + PostGIS on any machine with minimal configuration very quickly.



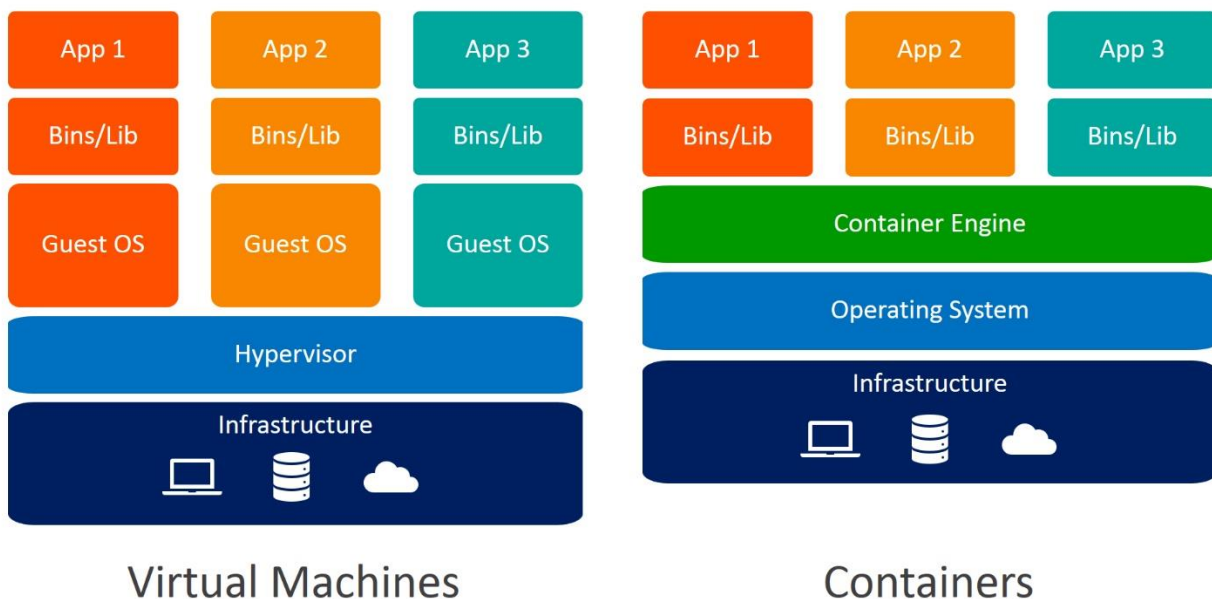
DOCKER ARCHITECTURE



Guys you will be thinking that it is like Virtual Machine. But it's not true. See the difference below.

Virtual Machine	Docker Container
Hardware-level process isolation	OS level process isolation
Each VM has a separate OS	Each container can share OS
Boots in minutes	Boots in seconds
VMs are of few GBs	Containers are lightweight (KBs/MBs)
Ready-made VMs are difficult to find available	Pre-built docker containers are easily
Creating VM takes a relatively longer time	Containers can be created in seconds
More resource usage	Less resource usage

Guys See the below architecture between Virtual Machine and Docker



- With containers, you can approach the database as an on-demand utility, which means that each application can have its own dedicated database that can be spun up as needed. This overcomes the disadvantages of large, monolithic databases with a microservices architecture supported by smaller, containerized databases
- Containerized databases separate storage from compute, denoting storage performance and capacity can be scaled independently of compute resources. This provides more flexibility in upfront database capacity orchestrating and provisioning, since changes are much more facile to make later
- Software-defined containerized databases provide a crucial missing link in high-velocity DevOps cycles, allowing development and operations teams to collaborate seamlessly. At the same time, however, containerized databases have a unique set of challenges in terms of high data availability, backup and recovery, and other critical database performance and compliance requirements.

The above three points tell us why Docker should be used to containerise PostgreSQL (with PostGIS).

Environment Setup

How to Install Docker on your local machine ?

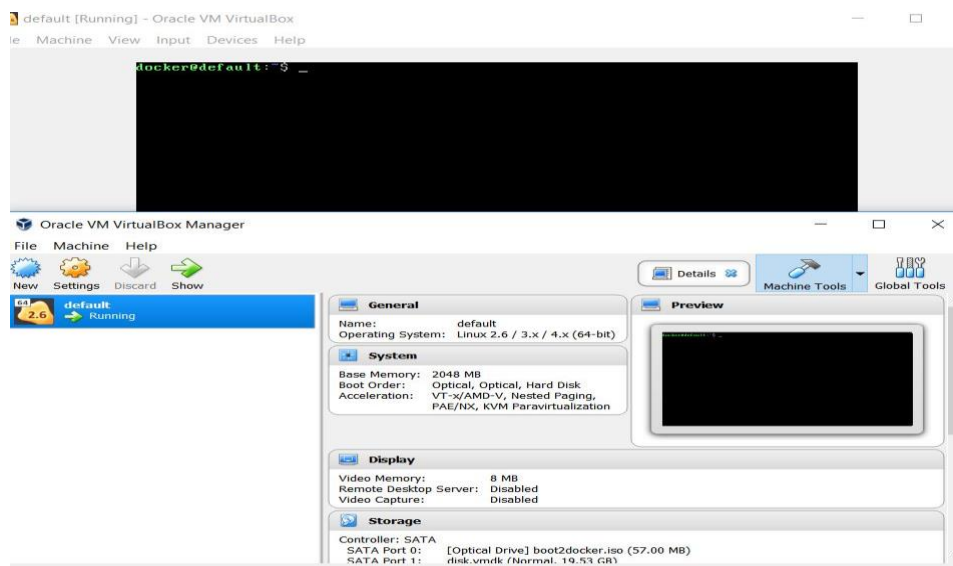
Guys, my experience with docker was amazing I faced a lot of difficulties. But finally, I got to know it's easy to install. So I will tell you how to install docker on your desktop safely and easily.

1. Docker for windows required **windows 10 with Hyper-V virtualization enabled.** which is only available on Windows 10 Professional or Enterprise 64 bit versions. But I will be not using Docker For Desktop as I told you the reason.

2. I will be Installing **Docker Toolbox** which is available for all OS(Operating System). And It uses VirtualBox to run your containers in conjunction with Docker.
3. Download the Docker Toolbox installer from here <https://github.com/docker/toolbox/releases>
4. See the Image below I have Downloaded the Installer.



5. Then I installed Docker Toolbox and it created the image in virtual box and it was running successfully.



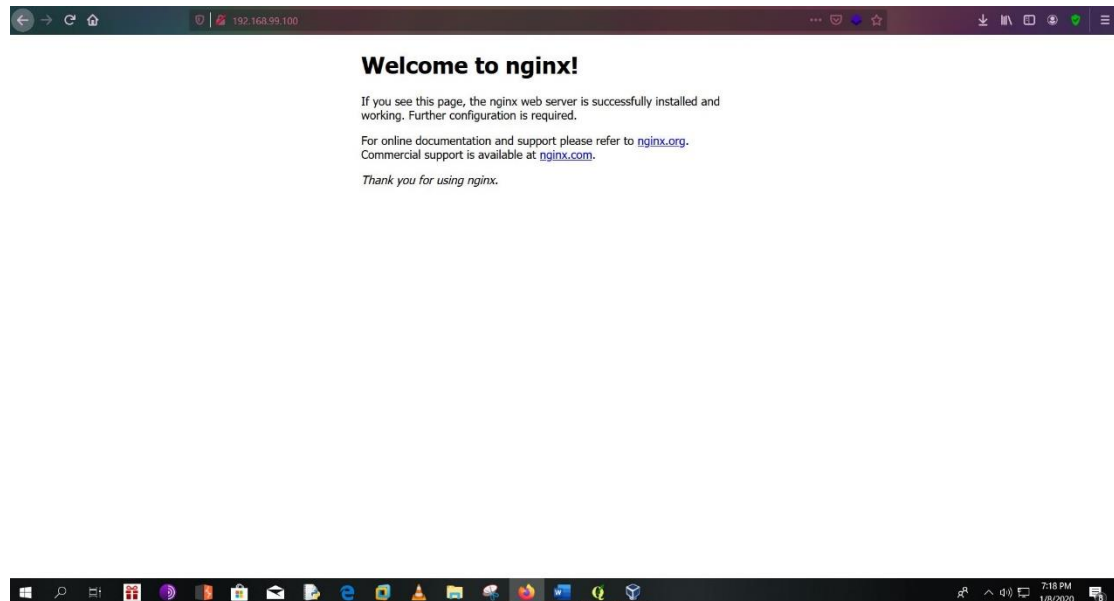
6. When you installed Docker Toolbox, it will installed and configured a Linux virtual machine on VirtualBox, where your containers will actually run. **Lets test out our**

installation by running a container running the NGINX web server. This will not only prove that our container is running, but that we can access it through the network abstractions created by VirtualBox and Docker.

7. In the terminal type `` docker run --name=nginx -d -p 80:80 nginx`` after running the command you will get the following output.

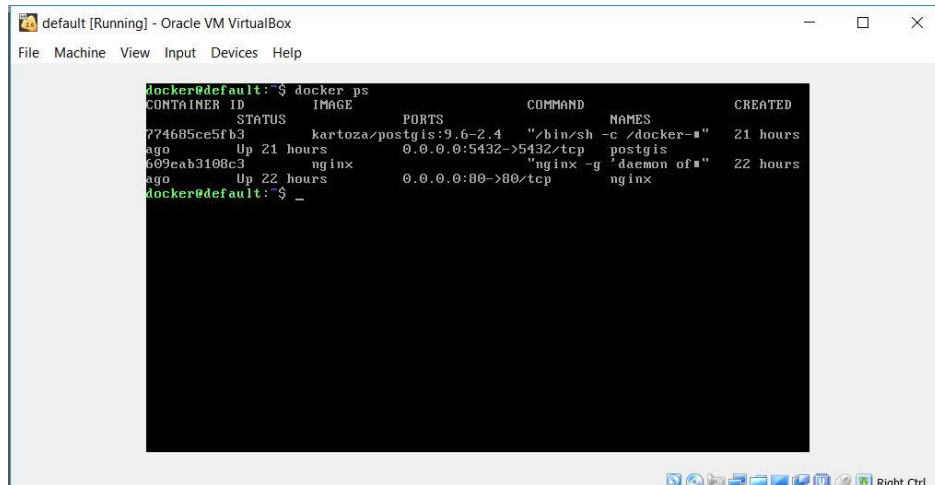
[illegible]

- Now get the docker ip address so we can check in our web browser If docker is actually running. Command : ``**docker-machine ip**`` or type ``**ifconfig**``
- Now in you web browser navigate to this Ip address and you should see the NGINX welcome screen. If yes docker properly installed.



Let's Get familiar with some basic Docker commands

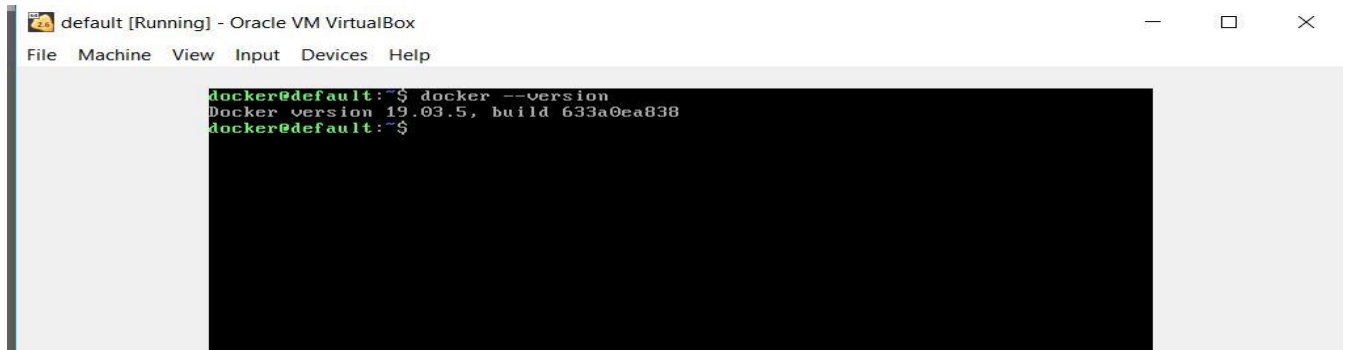
1. ``docker ps`` this command will tell us how many containers are running. At same time. See the output below.



The screenshot shows a terminal window titled "default [Running] - Oracle VM VirtualBox". The terminal output for the command "docker ps" is as follows:

CONTAINER ID	STATUS	IMAGE	PORTS	COMMAND	NAMES	CREATED
774685ce5fb3	Up 21 hours	kartoza/postgis:9.6-2.4	0.0.0.0:5432->5432/tcp	"/bin/sh -c /docker-"	postgis	21 hours
609eab310ec5	Up 22 hours	nginx	0.0.0.0:80->80/tcp	"nginx -g 'daemon of"	nginx	22 hours

2. ``docker --version`` this command will tell you the which version of docker you are running.



The screenshot shows a terminal window titled "default [Running] - Oracle VM VirtualBox". The terminal output for the command "docker --version" is:

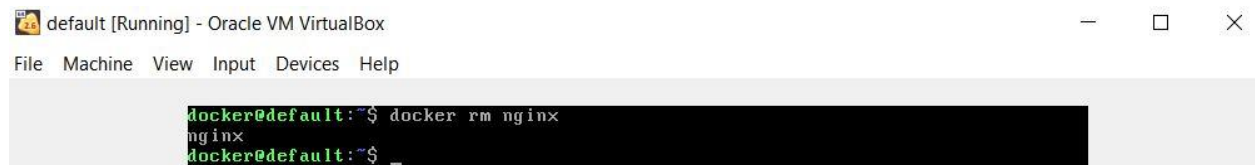
```
Docker version 19.03.5, build 633a0ea838
```

3. ``docker stop`` this command will stop the current container running.



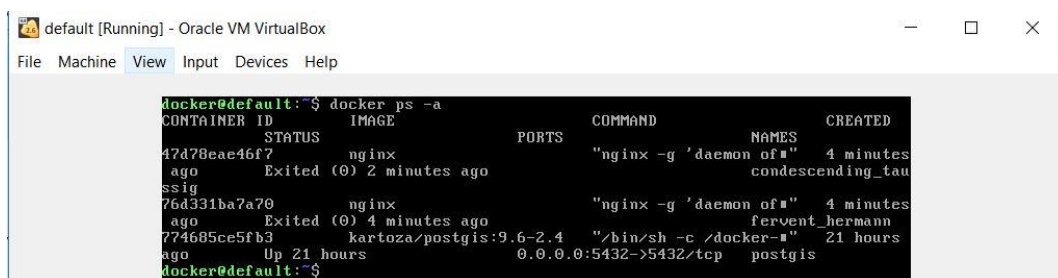
```
default [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
docker@default:~$ docker stop nginx
nginx
docker@default:~$ _
```

4. ``docker rm`` will remove the container from docker.



```
default [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
docker@default:~$ docker rm nginx
nginx
docker@default:~$ _
```

5. ``docker ps -a`` this command will tell us the status of the container whether it's running or not.



```
default [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
docker@default:~$ docker ps -a
```

CONTAINER ID	IMAGE	PORTS	COMMAND	NAMES	CREATED
47d78eae46f7	nginx		"nginx -g 'daemon of#"	condescending_tau	4 minutes ago
76d331ba7a70	nginx		"nginx -g 'daemon of#"	fervent_hermann	4 minutes ago
774685ce5fb3	kartoza/postgis:9.6-2.4	0.0.0.0:5432->5432/tcp	"/bin/sh -c /docker-#"	postgis	21 hours ago

```
Up 21 hours
docker@default:~$
```

6. ``docker pull`` this command is used to download the image file from repository of docker.

Like we will use the image of postgis so the command will be used here is

Command : Docker pullkartoza/postgis:9.6-2.4

Creating the container

Create a volume

It will persist our data it will be used to persist PostgreSQL database files outside of the the container that runs the database process.

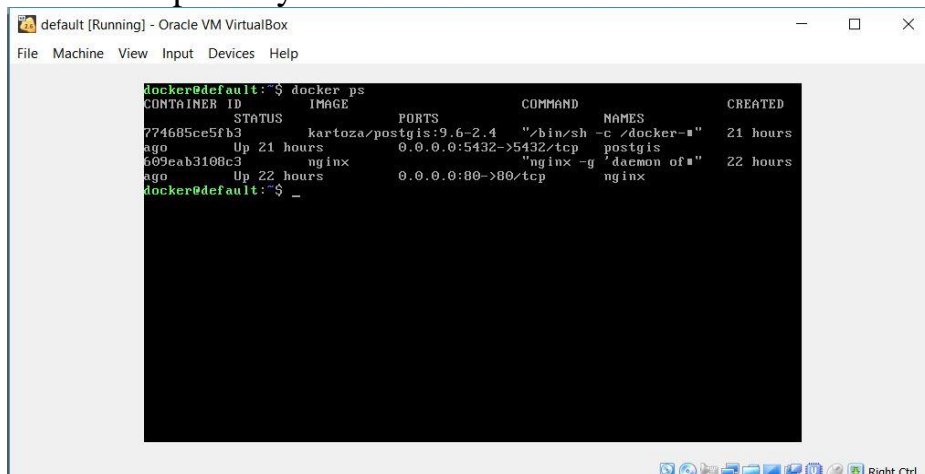
Command : `` docker volume create pg_data ``

Creating the Database Container

Now we'll use docker run to create the PostGIS container.

Command : ``docker run --name=postgis -d -e POSTGRES_USER=shivam -e POSTGRES_PASS=password -e POSTGRES_DBNAME=gis -e ALLOW_IP_RANGE=0.0.0.0/0 -p 5432:5432 -v pg_data:/var/lib/postgresql --restart=always kartoza/postgis:9.6-2.4``

Now use ``docker ps`` to see whether the images has been successfully forked from the repository or not.



```
docker@default:~$ docker ps
CONTAINER ID   STATUS    IMAGE              PORTS              COMMAND              NAMES              CREATED
774685ce5fb3   Up 21 hours kartoza/postgis:9.6-2.4 0.0.0.0:5432->5432/tcp "/bin/sh -c /docker-# postgis              21 hours
609eab3100c3   Up 22 hours nginx              0.0.0.0:80->80/tcp  "nginx -g 'daemon of# nginx              22 hours
docker@default:~$ _
```

Note: If you want to see log output from your container you can do so by using ``docker logs`` as an example just see the below image how the output will be there in your docker.

```
PostgreSQL stand-alone backend 9.6.5
2017-10-29 19:10:41.256 UTC [24] LOG:  could not bind IPv6 socket: Cannot assign requested address
2017-10-29 19:10:41.256 UTC [24] HINT:  Is another postmaster already running on port 5432? If not, wa
2017-10-29 19:10:41.256 UTC [24] WARNING: could not create listen socket for "::1"
backend> backend> postgres ready
2017-10-29 19:10:41.277 UTC [27] LOG:  database system was shut down at 2017-10-29 19:10:41 UTC
2017-10-29 19:10:41.278 UTC [28] [unknown]@[unknown] LOG:  incomplete startup packet
2017-10-29 19:10:41.279 UTC [27] LOG:  MultiXact member wraparound protections are now enabled
2017-10-29 19:10:41.285 UTC [24] LOG:  database system is ready to accept connections
2017-10-29 19:10:41.285 UTC [32] LOG:  autovacuum launcher started
Postgis is missing, installing now
Creating template postgres
Enabling template_postgis as a template
UPDATE 1
Loading postgis extension
CREATE EXTENSION
Enabling hstore in the template
CREATE EXTENSION
Enabling topology in the template
CREATE EXTENSION
```

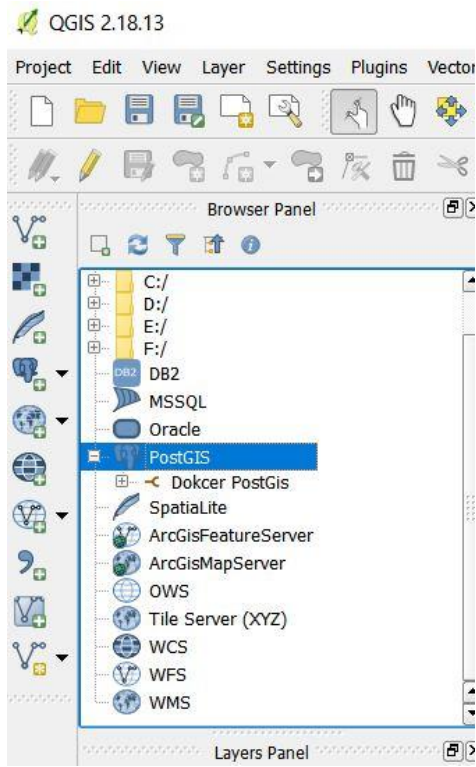
List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
gis	alex	UTF8	C.UTF-8	C.UTF-8	
postgres	postgres	UTF8	C.UTF-8	C.UTF-8	
template0	postgres	UTF8	C.UTF-8	C.UTF-8	=c/postgres +
template1	postgres	UTF8	C.UTF-8	C.UTF-8	postgres=CTc/postgres +
template_postgis	postgres	UTF8	C.UTF-8	C.UTF-8	=c/postgres +
					postgres=CTc/postgres

```
(5 rows)

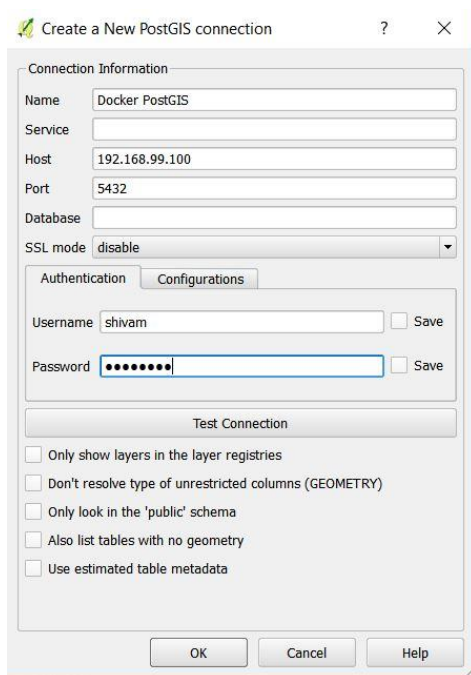
2017-10-29 19:10:43.514 UTC [24] LOG:  received smart shutdown request
2017-10-29 19:10:43.514 UTC [32] LOG:  autovacuum launcher shutting down
2017-10-29 19:10:43.517 UTC [29] LOG:  shutting down
2017-10-29 19:10:43.524 UTC [24] LOG:  database system is shut down
Postgres initialisation process completed .... restarting in foreground
2017-10-29 19:10:43.564 UTC [126] LOG:  database system was shut down at 2017-10-29 19:10:43 UTC
2017-10-29 19:10:43.566 UTC [126] LOG:  MultiXact member wraparound protections are now enabled
2017-10-29 19:10:43.568 UTC [123] LOG:  database system is ready to accept connections
2017-10-29 19:10:43.568 UTC [130] LOG:  autovacuum launcher started
```

Connect to the containerised database using QGIS

You should now be able to add the connection to PostGIS in the browser panel in QGIS. Open QGIS from the start menu. **Under Browser Panel** just check **postgis** is there. And then right click on it and create a new connection.



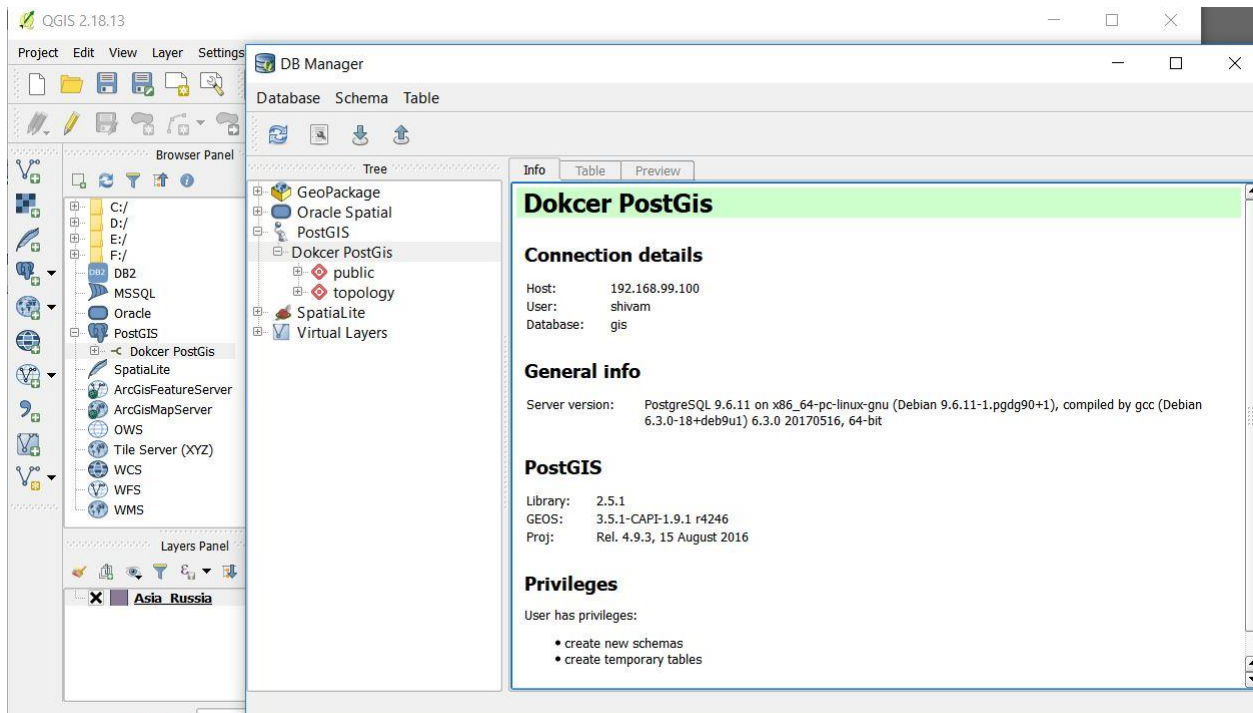
Now Enter the database connection parameters you used in the docker run command. The “Host” parameter will be the IP of your Docker VM (you can get it by running `docker-machine ip` in the terminal window)



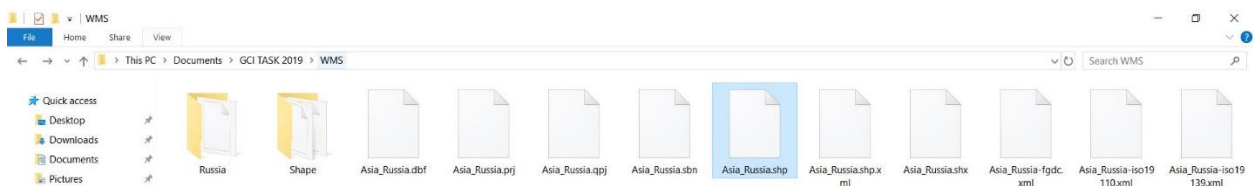
And click ok it will send some messages **Connection Successful**

Importing the Geometry

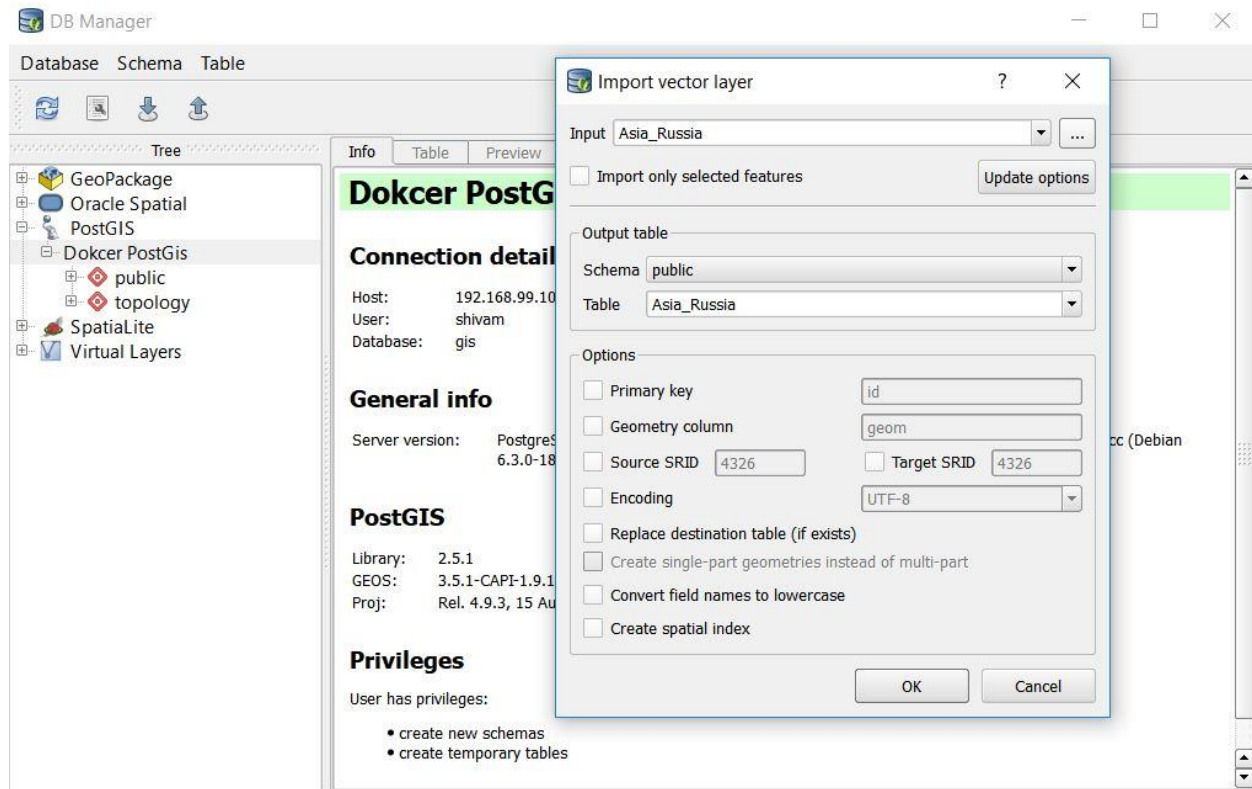
We will use the DB Manager plugin to create new database schemas, as well as import and export files from the database. Open the Db Manager



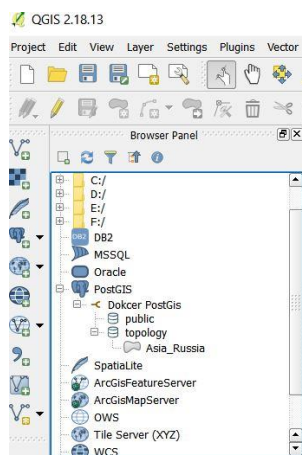
As you can see the connection details also. Now locate in which directory your layer file is saved. I will be using my Asia_Russia ShapeFile.



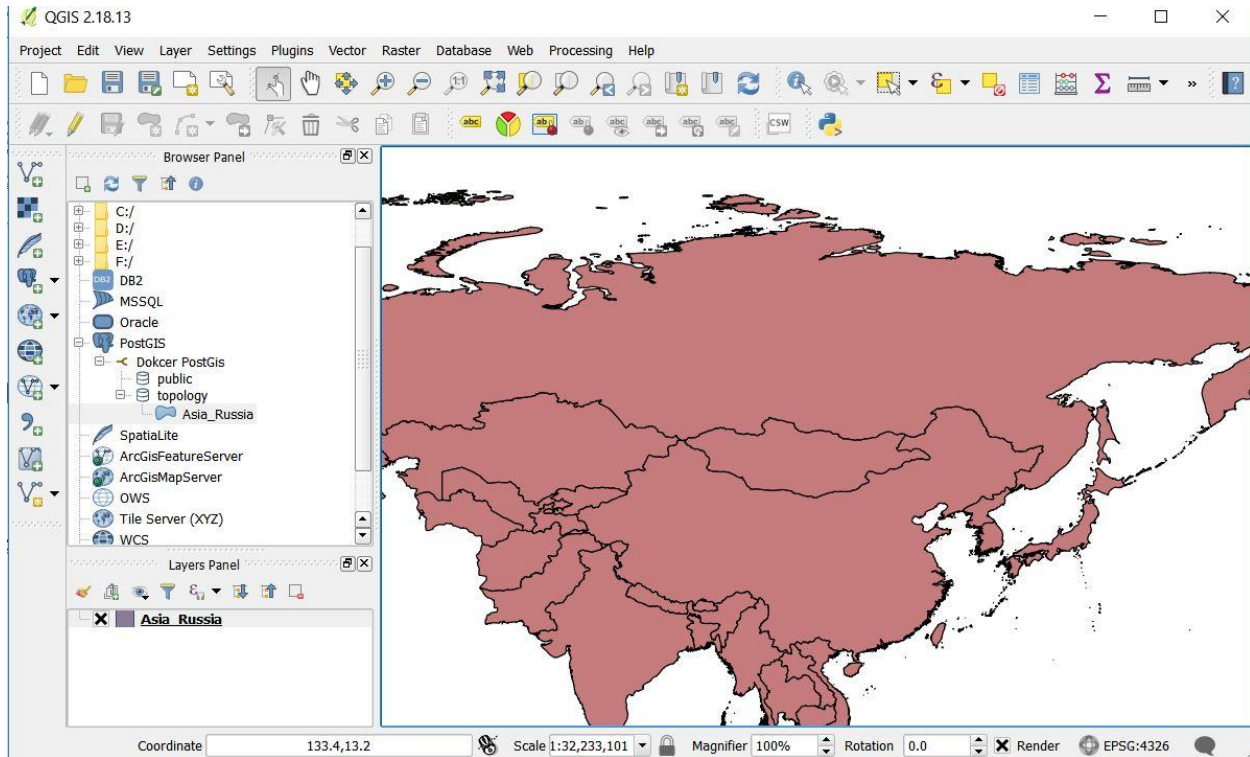
Now **Import the layer/file from DB manager**. And click on **Update Options** it will automatically will the required fields. And don't forgot to choose the Schema under output table.



Now Click On Okay and Under Browser Panel just refresh the Schema (Public, Topology) and you will see that we successfully imported the layer in PostGis.



Last, Click on that layer and drag it to the **Layers Panel**. and the data would be visible to workspace area.



■ Thank You !

#SOURCE USED

<https://alexurquhart.com/post/set-up-postgis-with-docker/>

<https://gis.stackexchange.com/questions/223487/what-is-the-purpose-of-postgis-on-postgresql>

<http://postgis.net/features/>

<https://containerjournal.com/topics/container-management/11-things-to-know-about-databases-and-postgres-in-containers/>