**1. Setup ‘NGINX’ server using docker.**

- NGINX is a popular lightweight web application that is used for developing server-side applications.

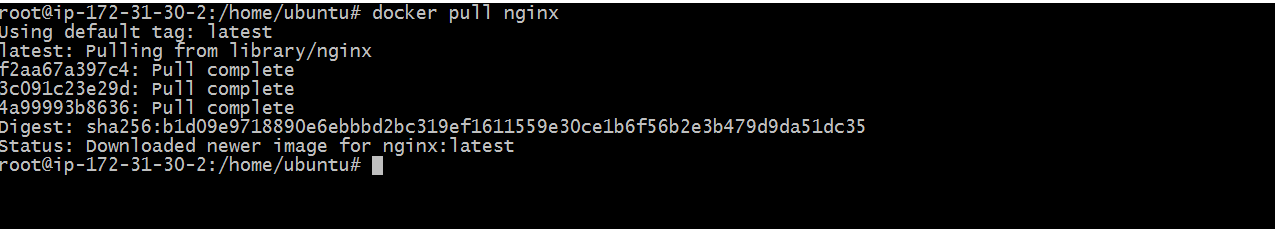
- Use the official nginx image from docker hub registry.

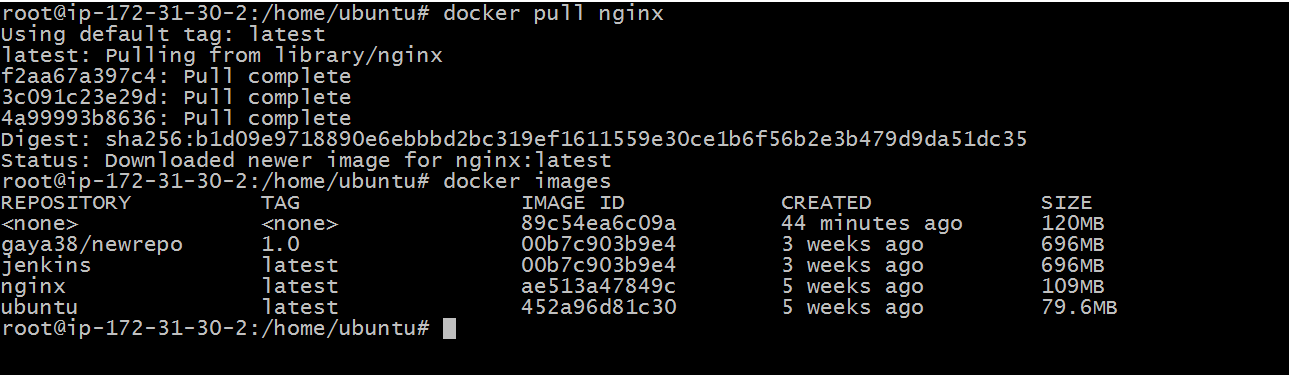
- Pull image and run the docker container on docker host.

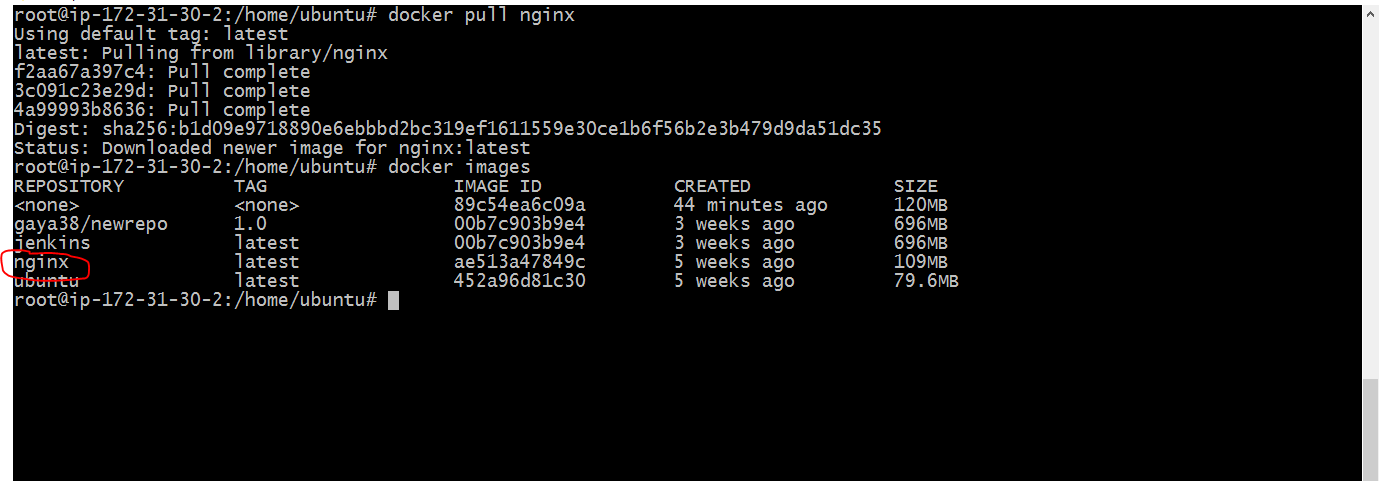
- Open then 8080 port.

- Try accessing http://dockerhost\_ip:8080 and check installation

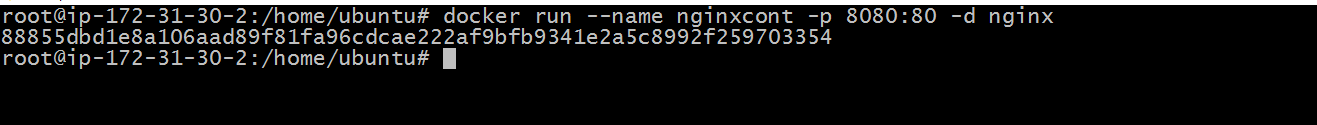
.Pull nginx image from Docker hub registry



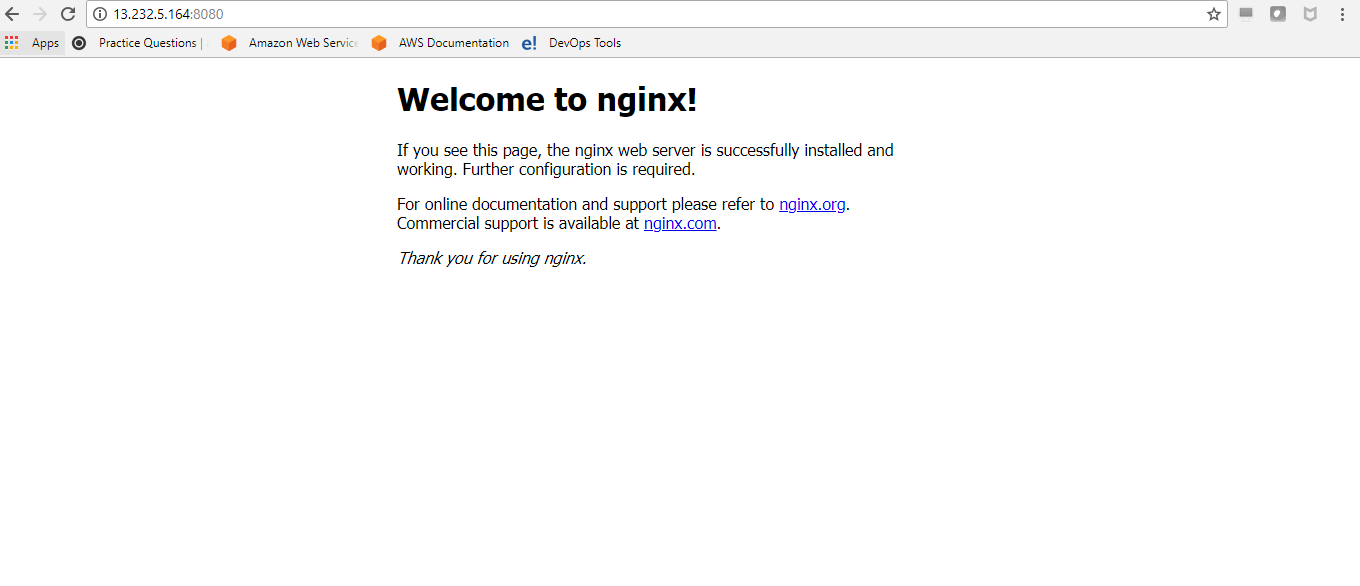




Run the container on 8080 port



Nginx page



**2. Setup simple hello world website using ‘NGINX’ server using docker.**

- NGINX is a popular lightweight web application that is used for developing server-side applications.

- Use the nginx server to host your simple ‘hello world‘ website.

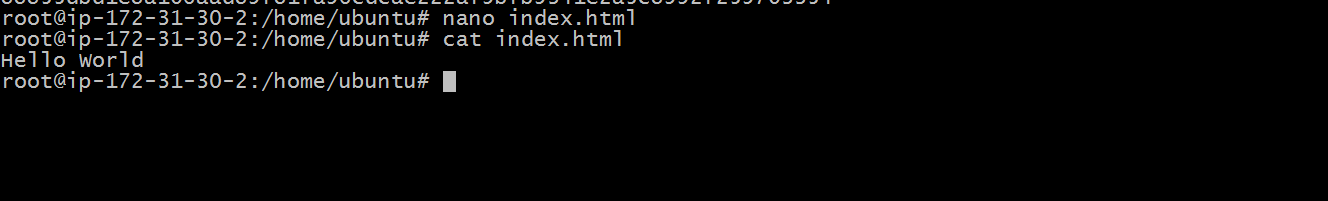
- Create simple html page. Say an ‘Index.html’ page.

- Pull image and run the docker container on docker host with correct commands..

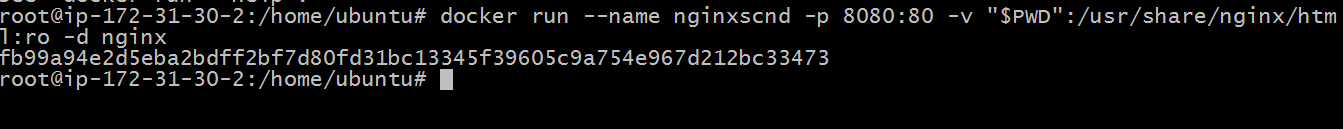
- Open then 8080 port.

- Try accessing http://dockerhost\_ip:8080/index.html and check the welcome message.

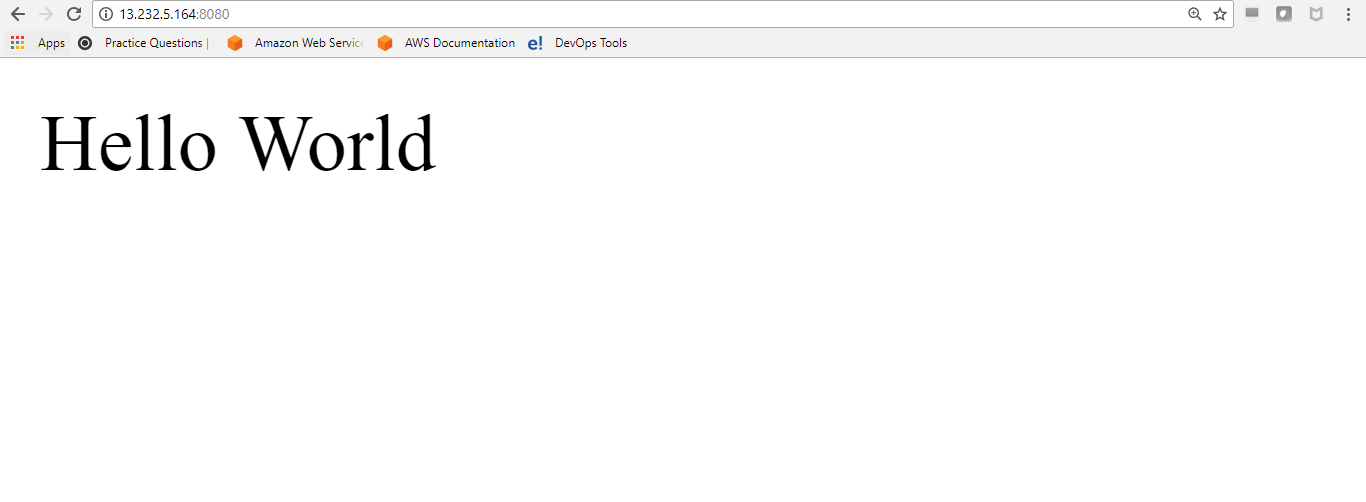
Creating “Hello World” in index.html



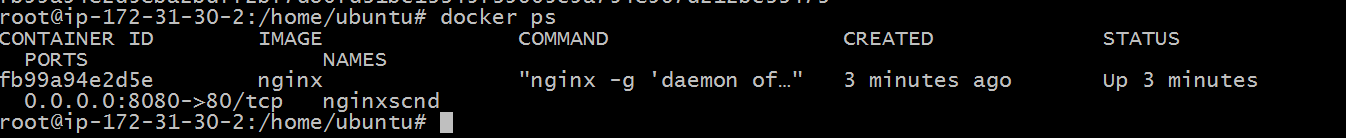
Run the container



Web Page



Docker container



**3. Setup ‘MongoDB’ using docker.**

- MongoDB is a famous document-oriented database that is used by many modern-day web applications.

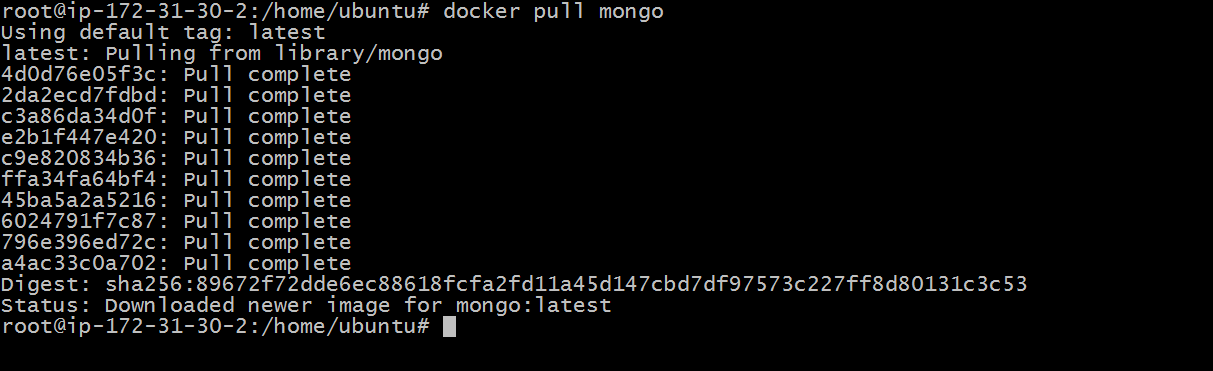
- Use the official image from docker hub registry.

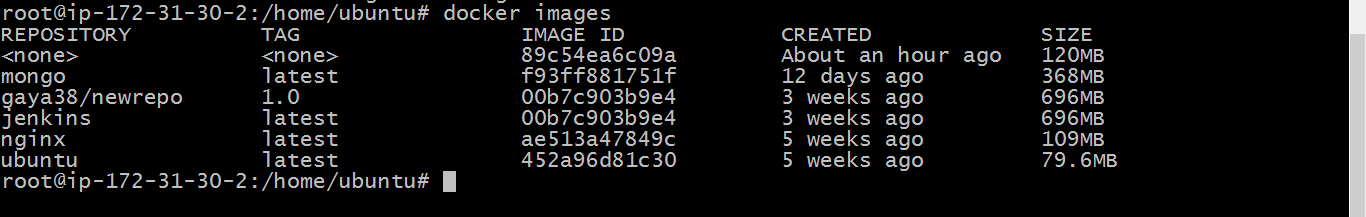
- Pull image and run the docker container on docker host.

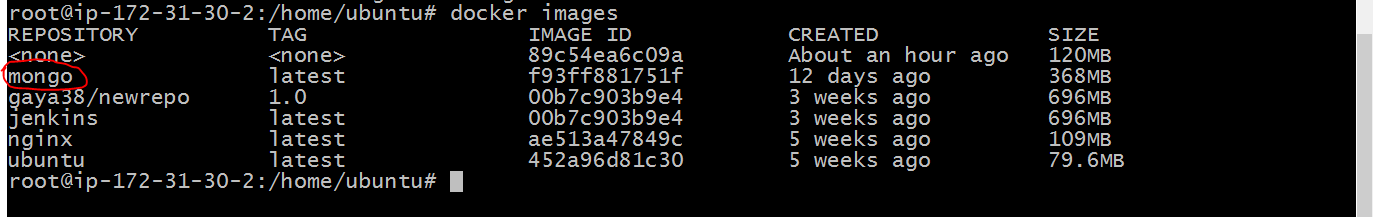
- Open then required port while running container.

- Try connecting to mongodb using desktop clients to verify.

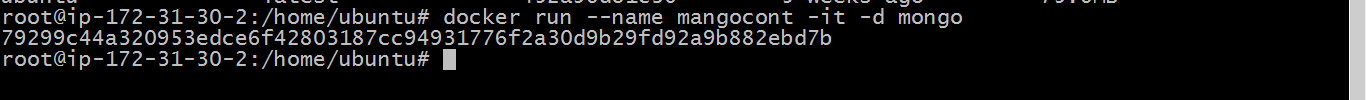
Pull Mongo image from Docker hub registry

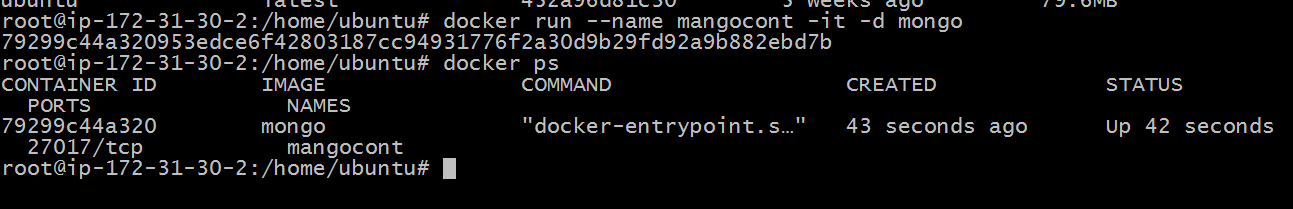


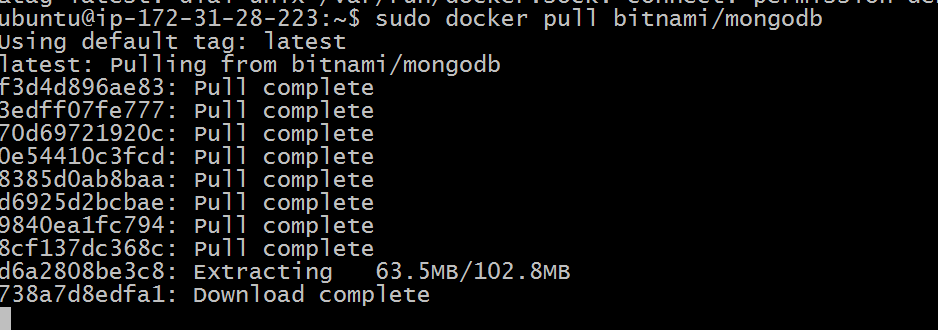


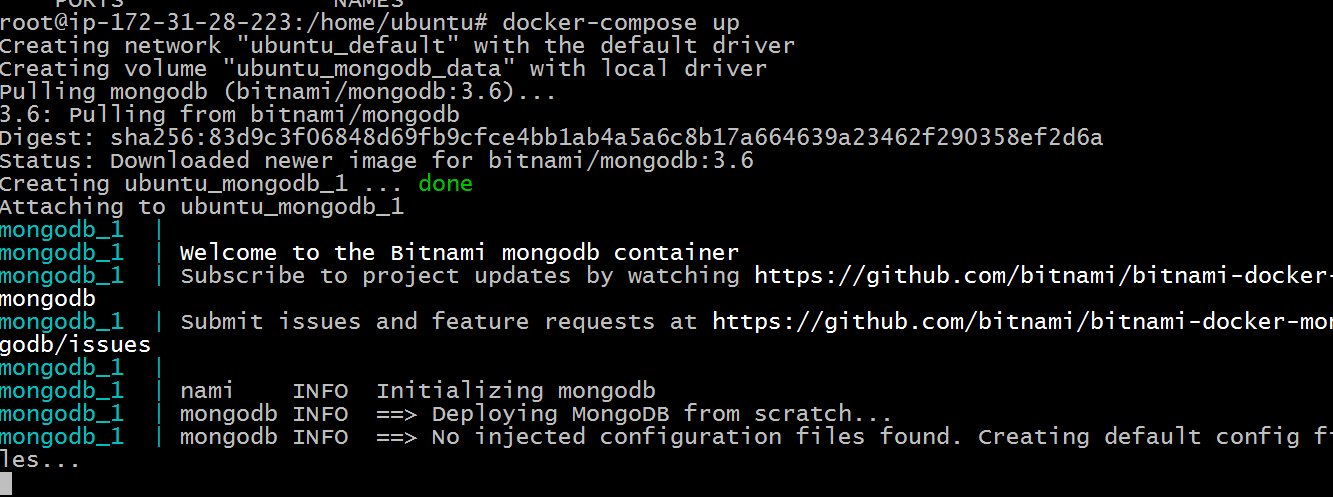


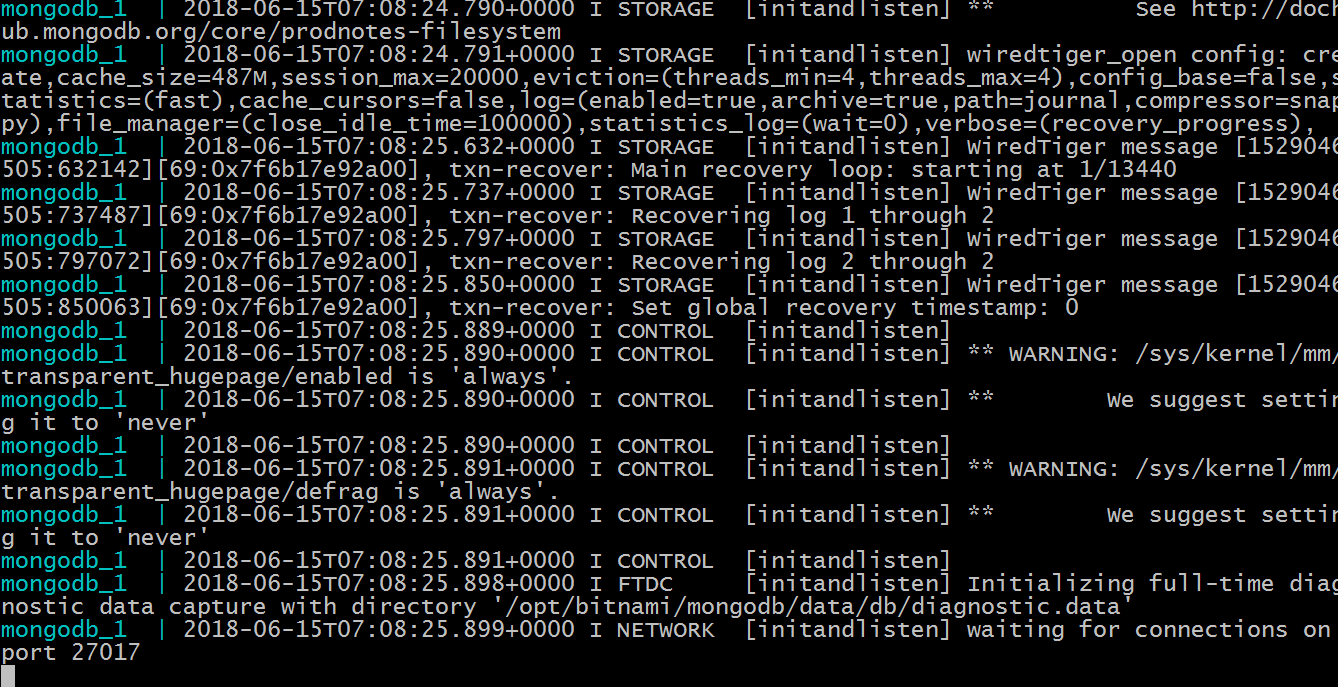
Run the container

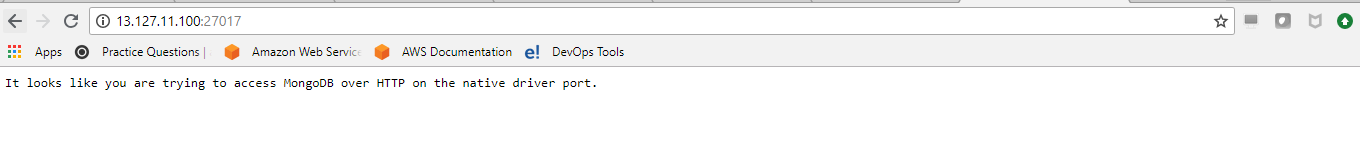












**Level 1**

**1. Create your own dockerfile to install Python.**

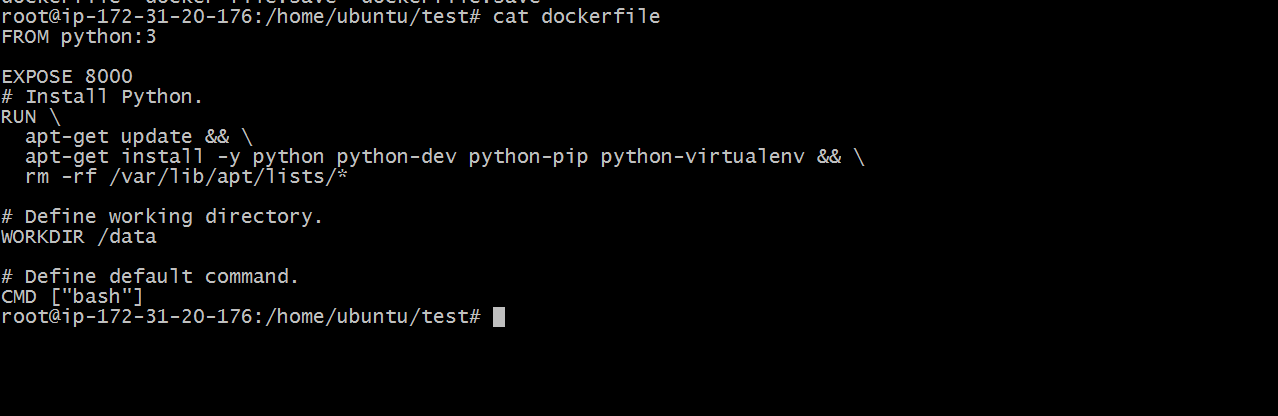
- Create your own dockerfile to install python

- Expose required ports.

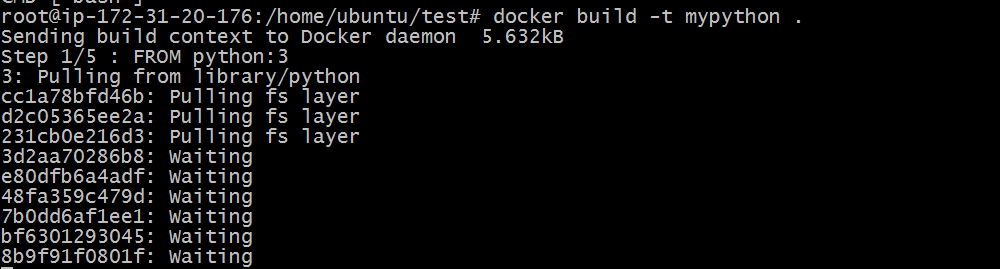
- Verify installation.

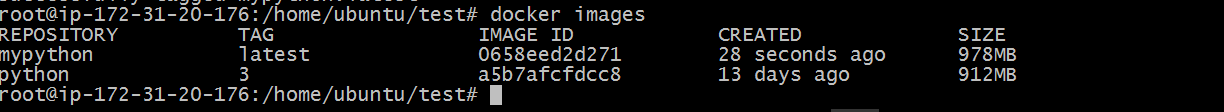
- Publish your image to docker hub registry.

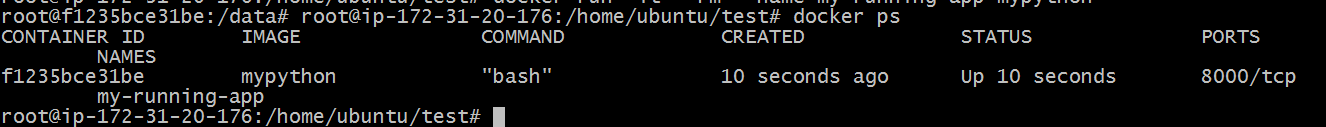
Creating dockerfile to install python and expose required ports



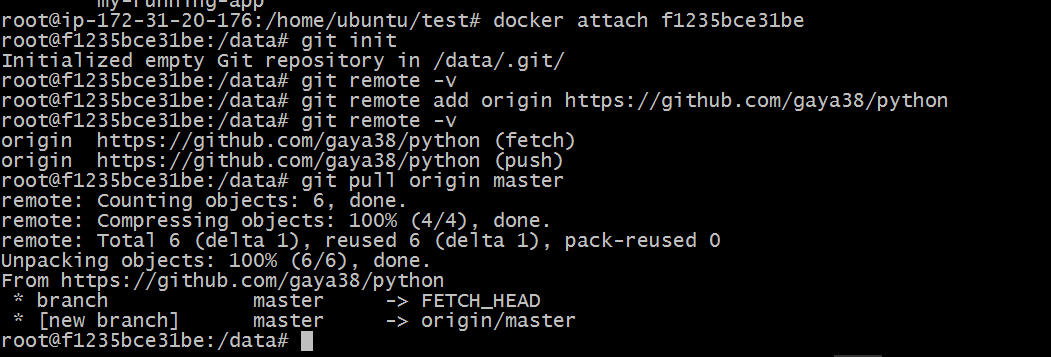
Build an image and running the container by using this file



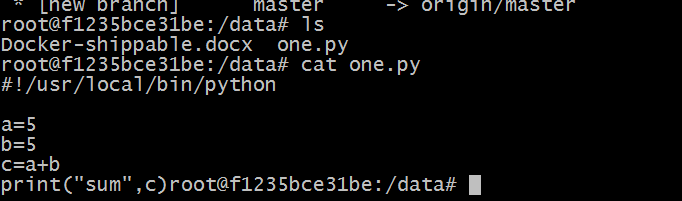




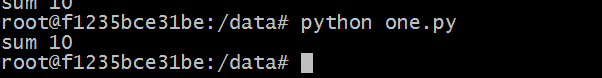
Verifying installation by running the python file



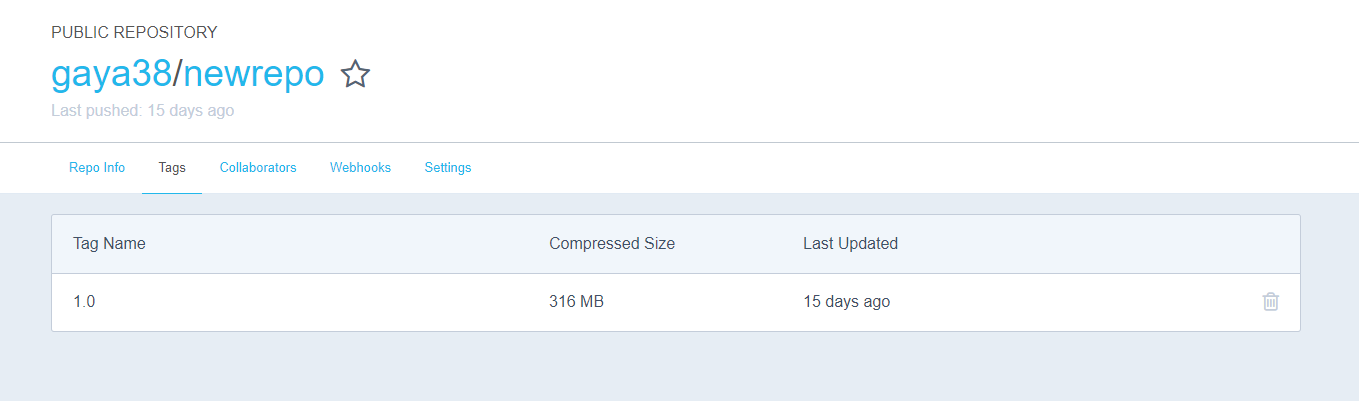
Python program



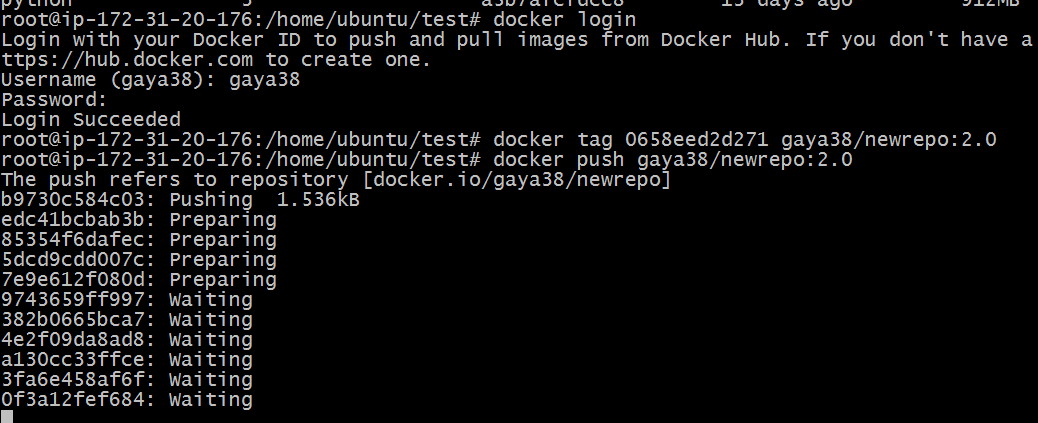
Running the python file



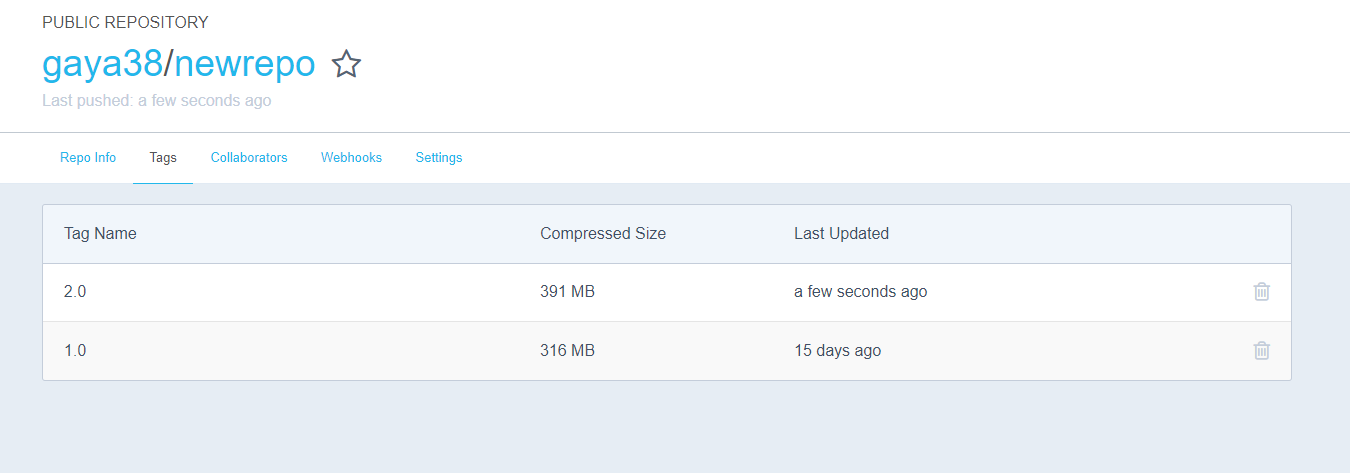
Before publishing the image to Docker hub registry



Publishing the image to Docker hub registry



After publishing the image to Docker hub registry



**2. Create your own dockerfile to setup Elasticsearch.**

- Create your own dockerfile to setup single node ES cluster.

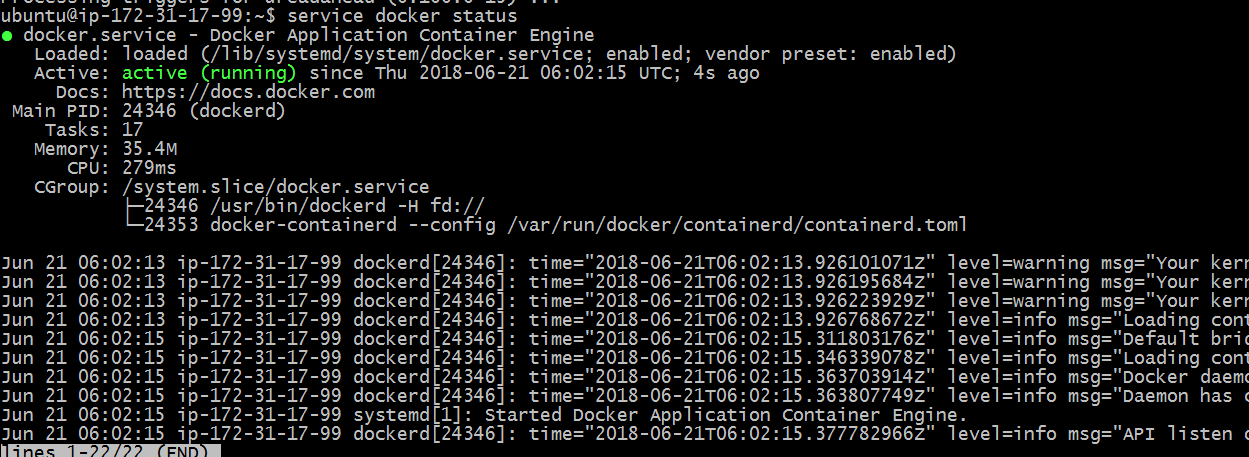
- Expose required ports.

- Install ES plugins.

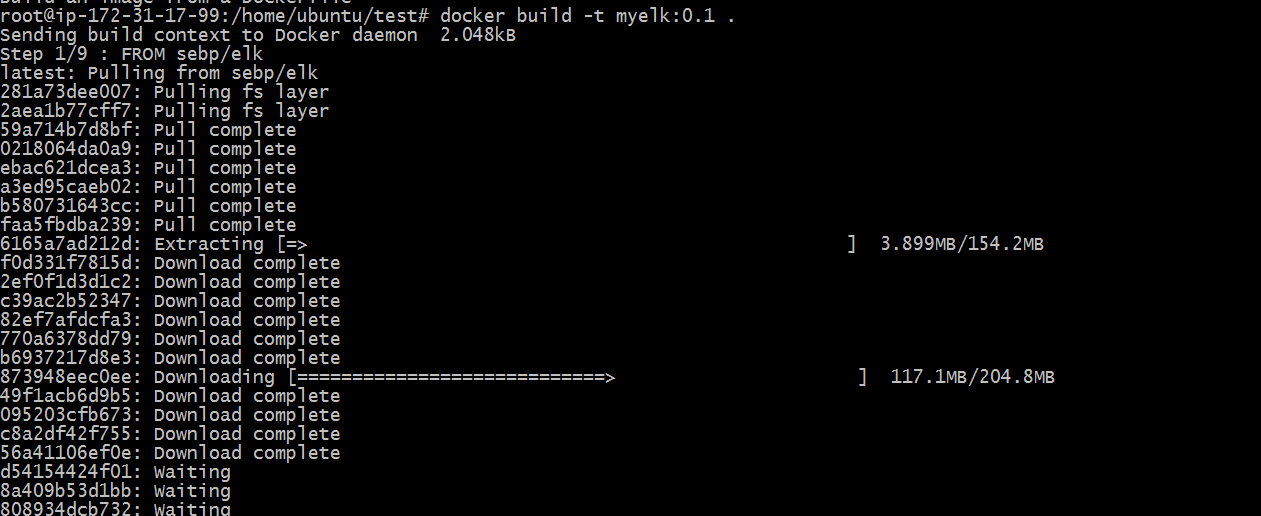
- Verify installation by accessing through web browser.

- Publish your image to docker hub registry.

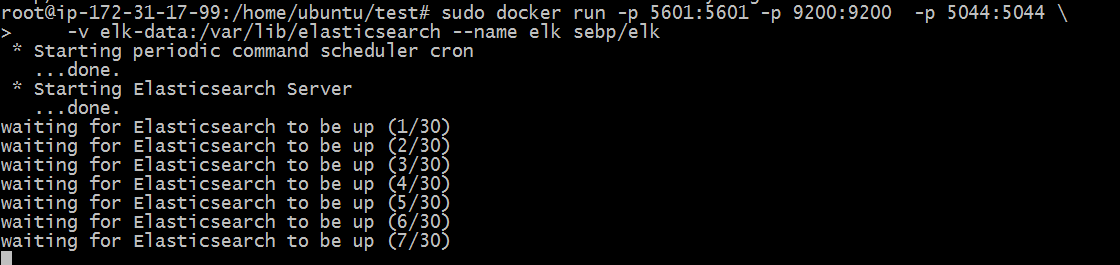
Installed Docker



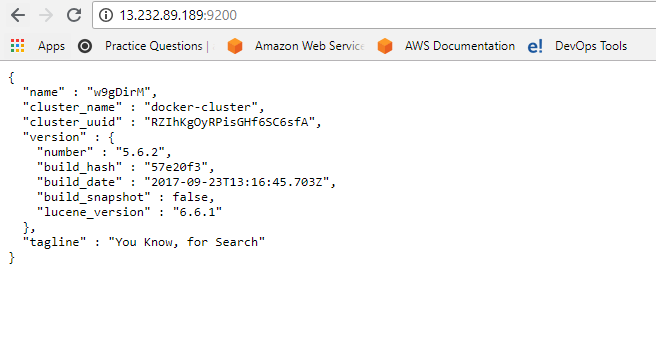
Building an image from Docker file



Running docker container



Elastic search on Port:9200



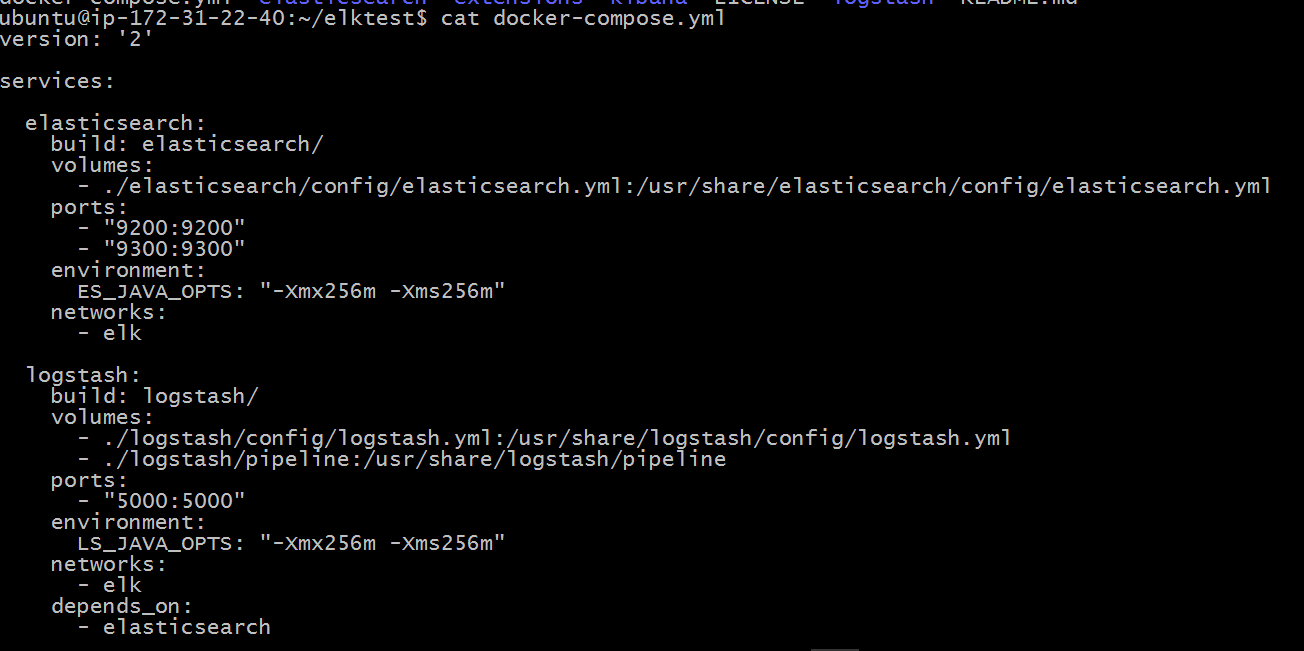
**1. Use Docker-Compose to setup multi container ELK stack.**

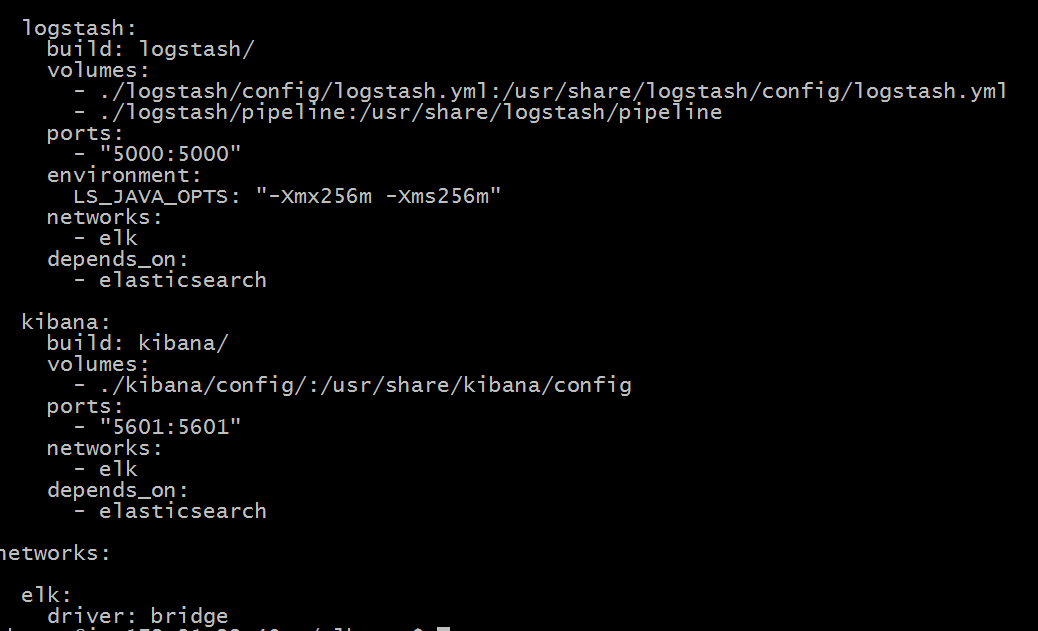
- Create your own docker compose file to setup Elasticsearch, logstash and Kibana (ELK) stack.

- Expose required ports as needed.

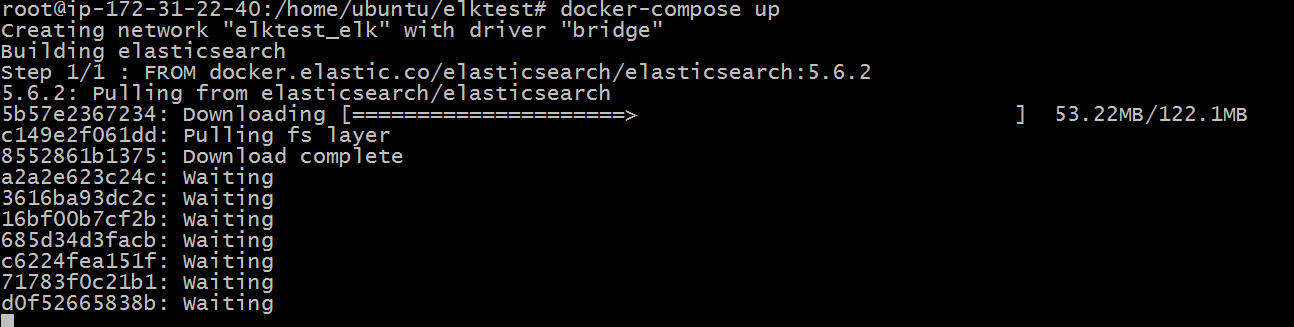
- Create the stack and test it.

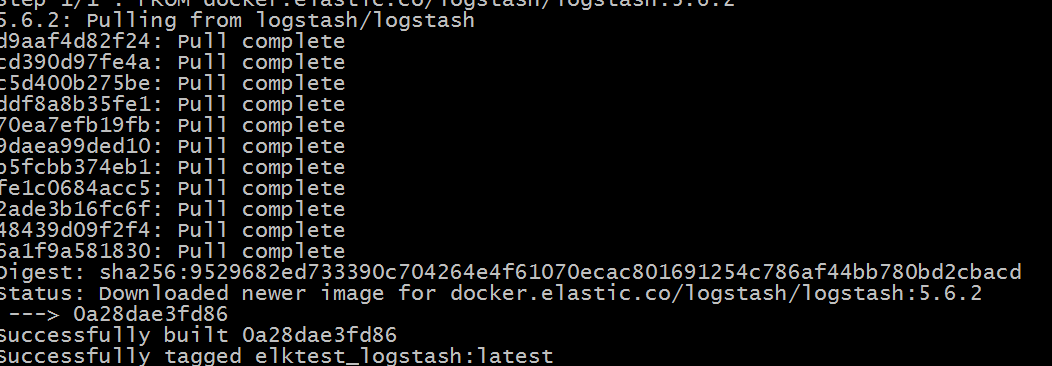
Creating docker compose file

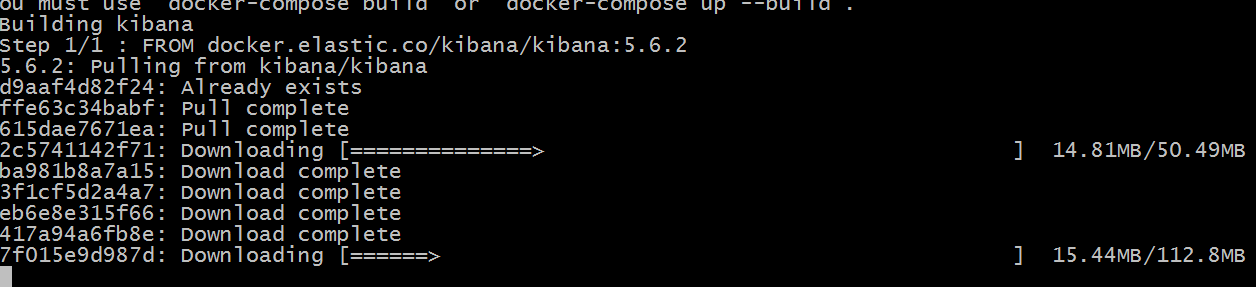


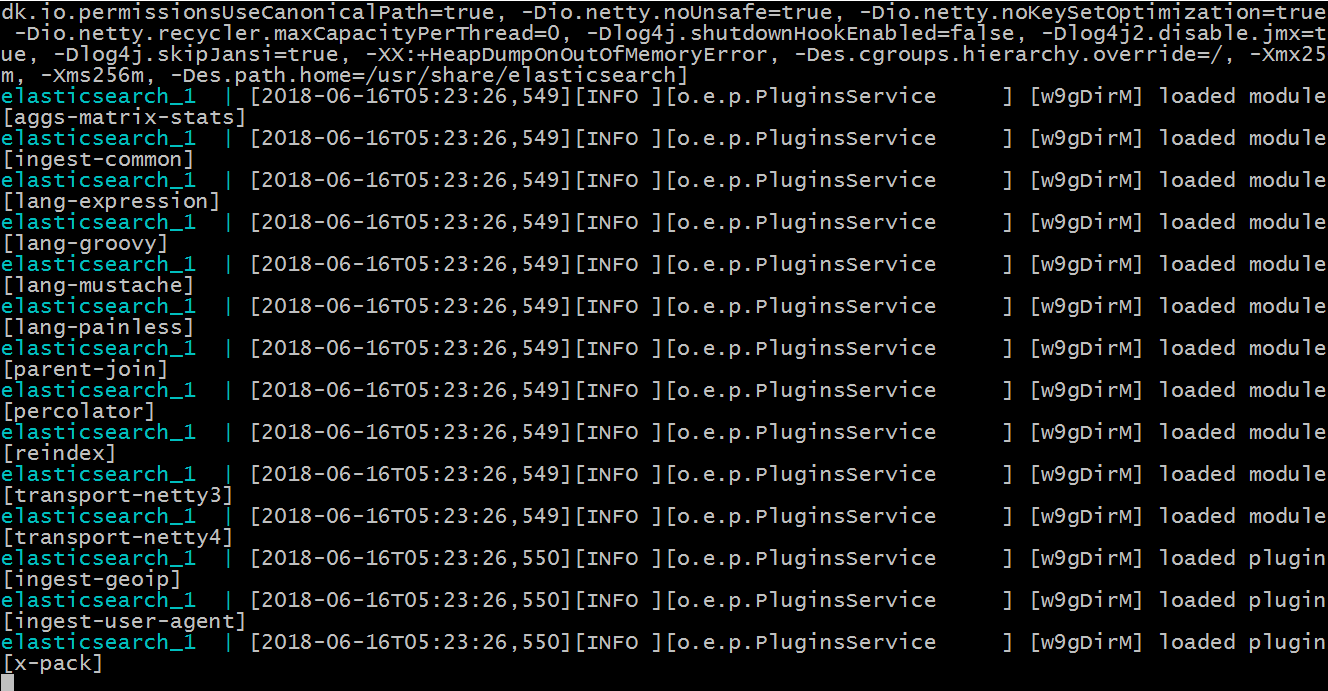


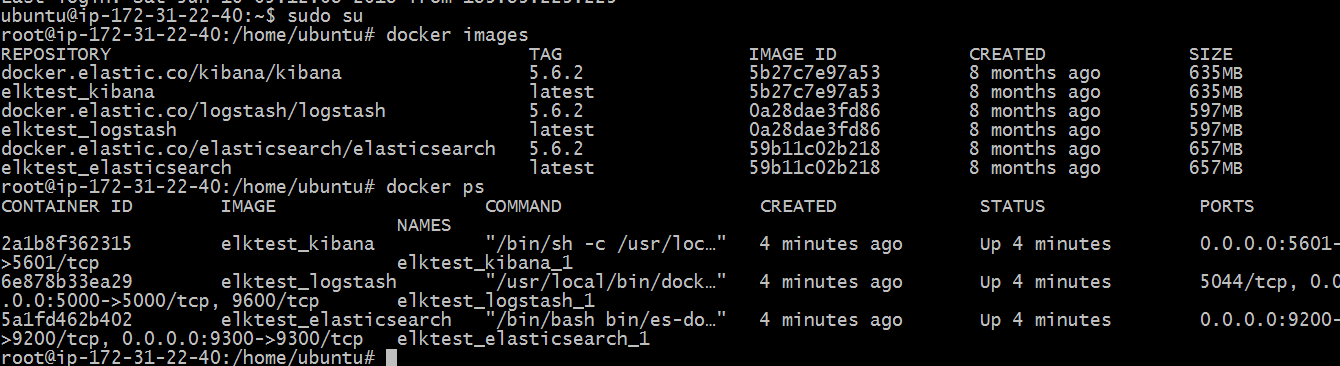
Downloading images



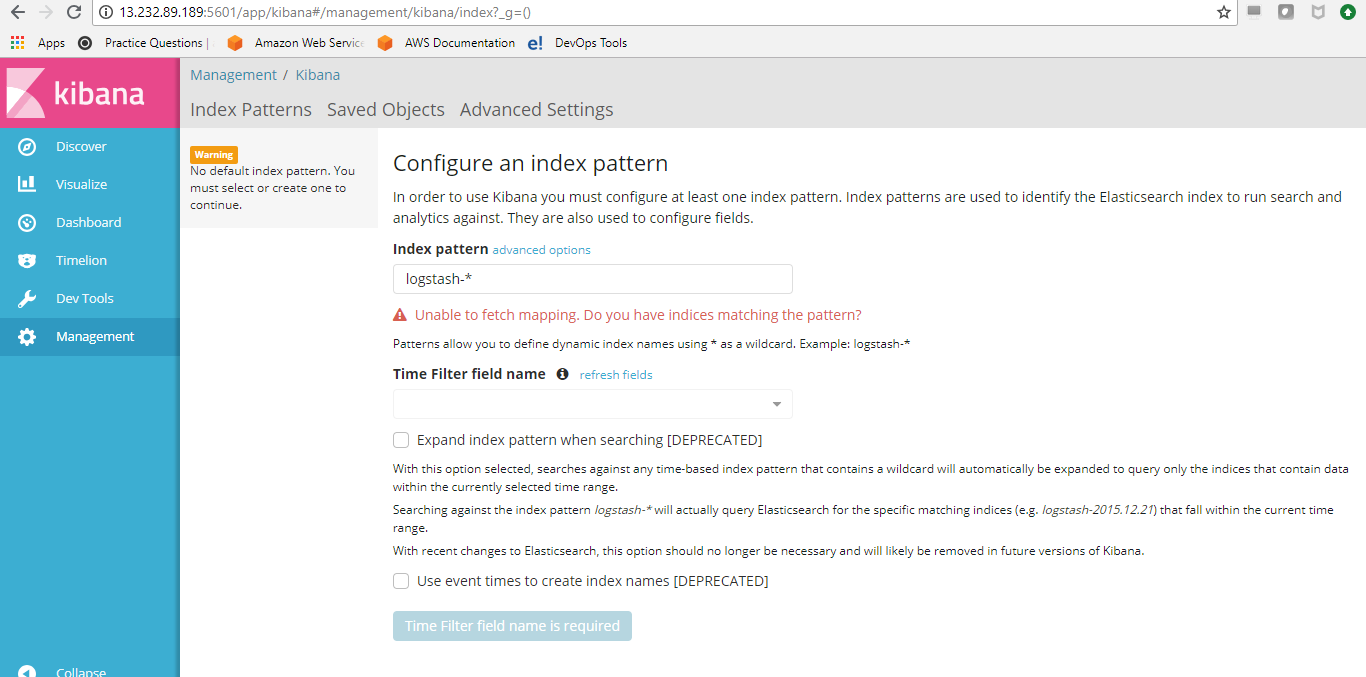




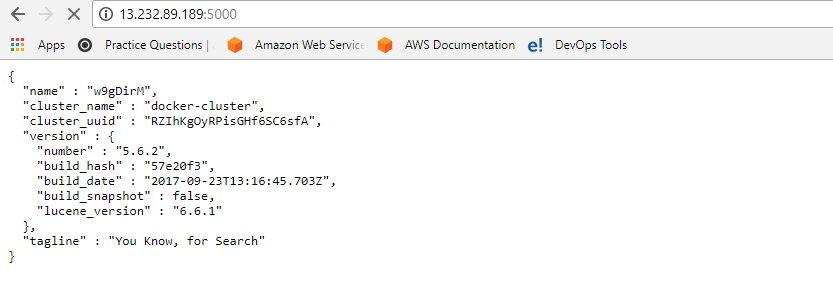




Kibana on Port:5601



Logstash on Port:5000



Elastic search on Port:9200

