## **Day 17: Docker Project for DevOps Engineers.**

# **Dockerfile**

Docker is a tool that makes it easy to run applications in containers. Containers are like small packages that hold everything an application needs to run. To create these containers, developers use something called a Dockerfile.

A Dockerfile is like a set of instructions for making a container. It tells Docker what base image to use, what commands to run, and what files to include. For example, if you were making a container for a website, the Dockerfile might tell Docker to use an official web server image, copy the files for your website into the container, and start the web server when the container starts.

For more about Dockerfile visit [here](https://rushikesh-mashidkar.hashnode.dev/dockerfile-docker-compose-swarm-and-volumes)

**Task:**

* Create a Dockerfile for a simple web application (e.g. a Node.js or Python app)

Dockerfile is nothig but a set of commands to execute or automate the application deployments

Follow below steps to deploy flask app to DockerHub

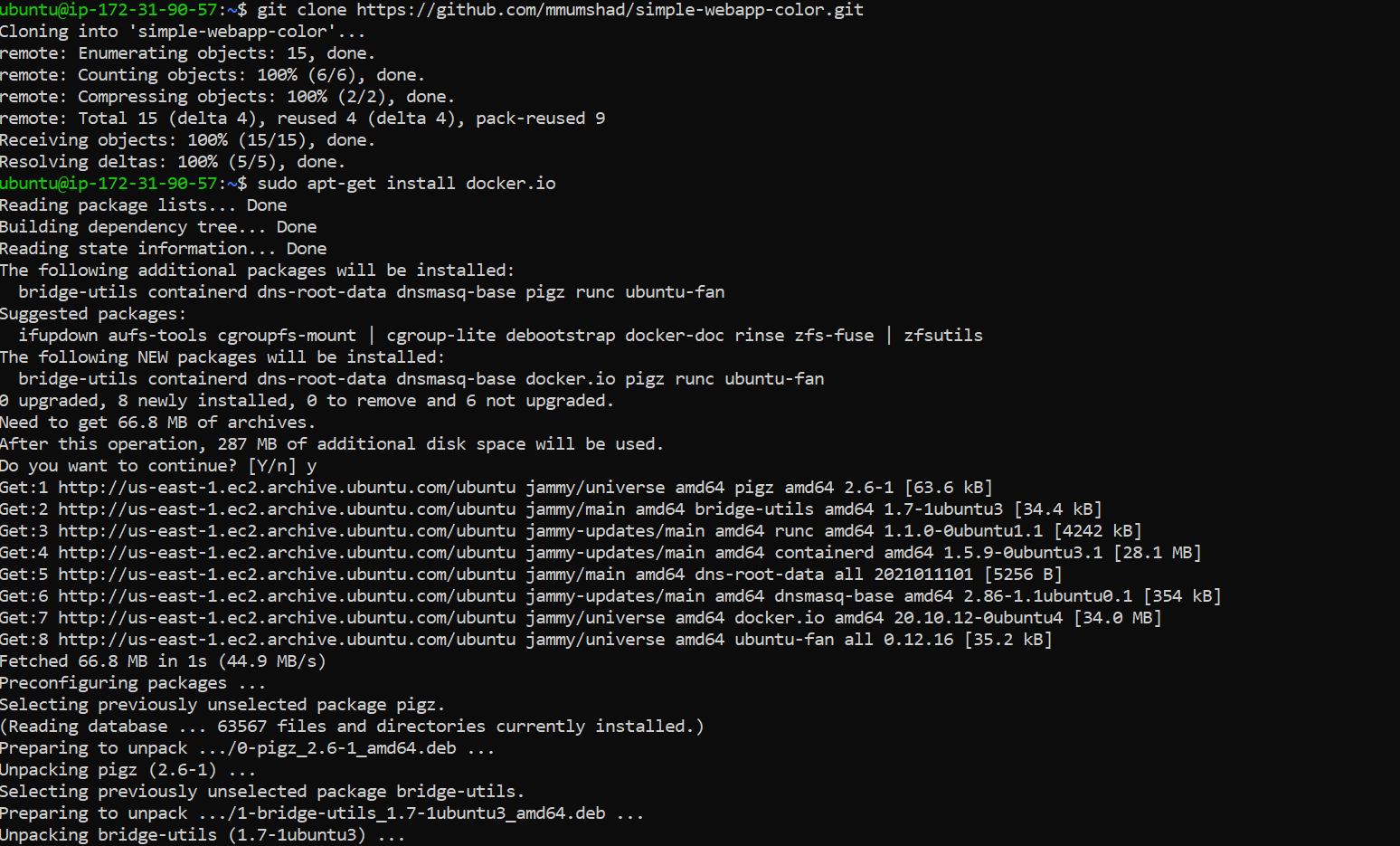
Launch AWS ubuntu ec2 instance

Ref:- https://medium.com/@misalPav103/deploying-nodejs-app-on-aws- ec2-instance-step-by-step-1b00f807cdce

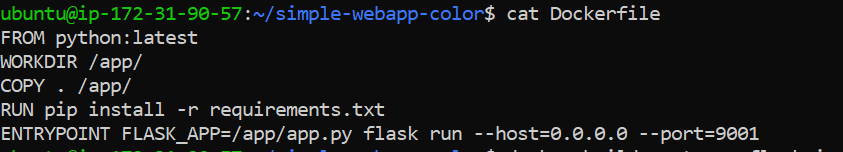
After That Update the system and install docker, using below command

sudo apt-get update && sudo apt-get install docker.io

After successfully installed docker, clone the git repository



After that you create a Dockerfile



FROM python: latest

fetching the latest python image from docker hub

WORKDIR /app/

now we create a directory to hold application code inside the image. This is current working directory of image

COPY . /app/

Copy all local application data to image, inside /app/ folder

RUN pip install -r requirements.txt

Using Run Command, we have to install some dependencies. All dependencies are stored inside requirements.txt.

Using Run Command, we install all the dependencies recursively.

ENTRYPOINT FLASK\_APP=/app/app.py flask run --host=0.0.0.0 --port=9001

ENTRYPOINT is a command is responsible for execute a command, allocate port number to application, allocate host ip, etc...

* Build the image using the Dockerfile and run the container

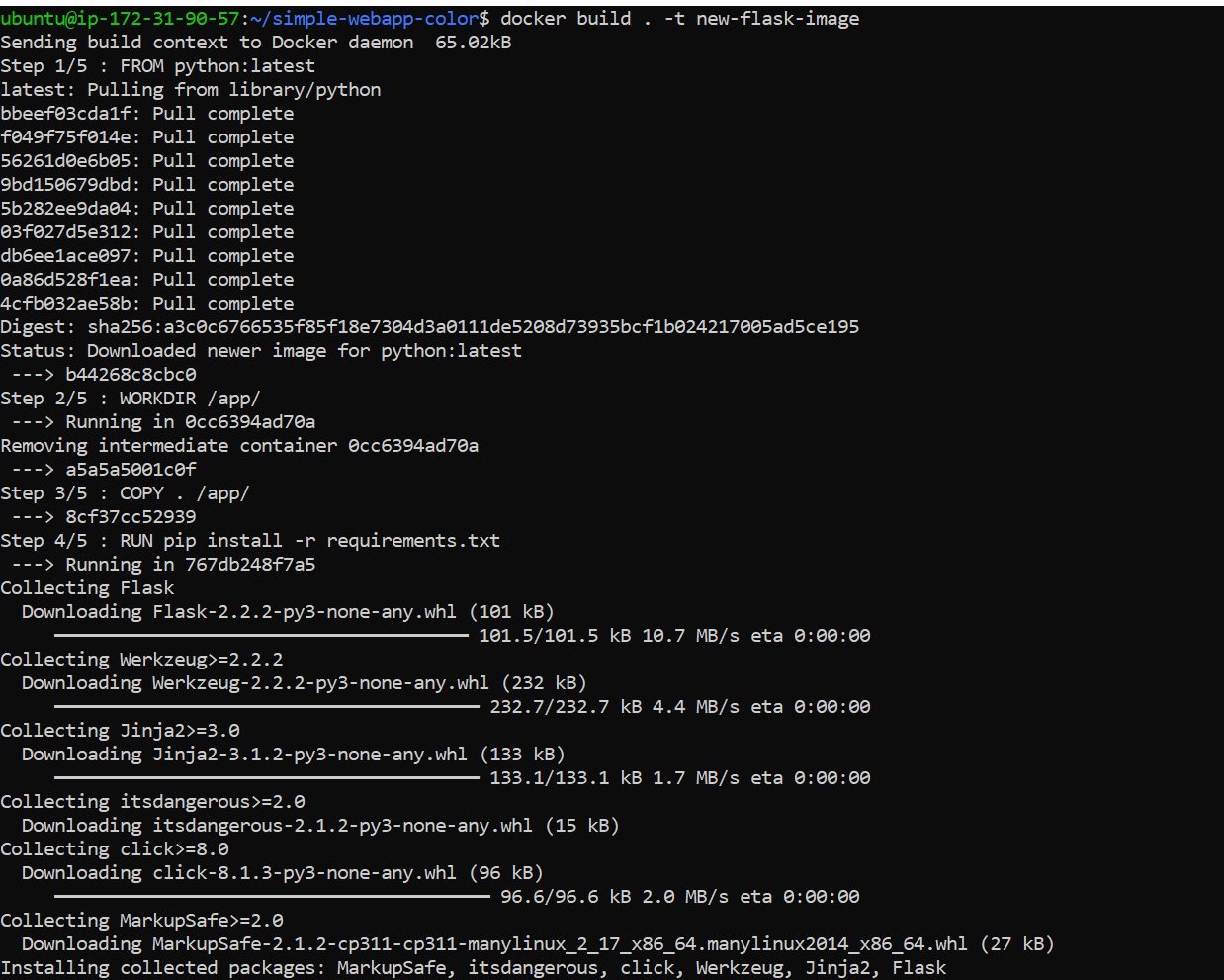
Before building dockerfile,Run the below command to set current user All permissions of executing the Dockerfile.

sudo –a –G docker $USER && sudo reboot

After rebooting, again create a connection

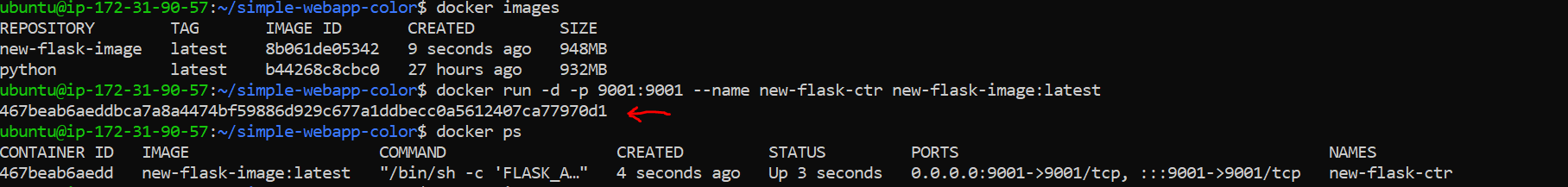
To Build a image using Dockerfile.

docker build . -t new-flask-image



After Building a image Run this image to create a container.

docker run -d -p 9001:9001 --name new-flask-ctr new-flask-image: latest



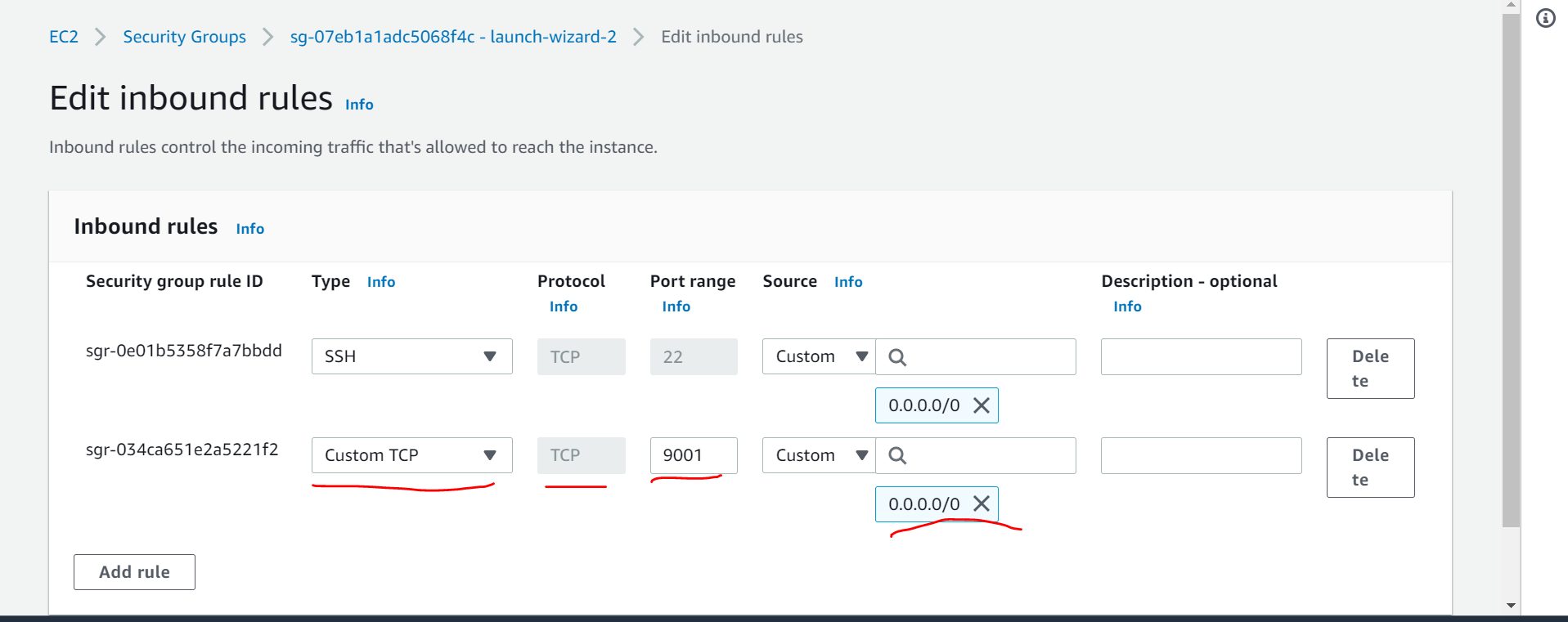
* Verify that the application is working as expected by accessing it in a web browser

Now, container is running, now you go to ec2 instance page, and add a 9001 port in Security Group.

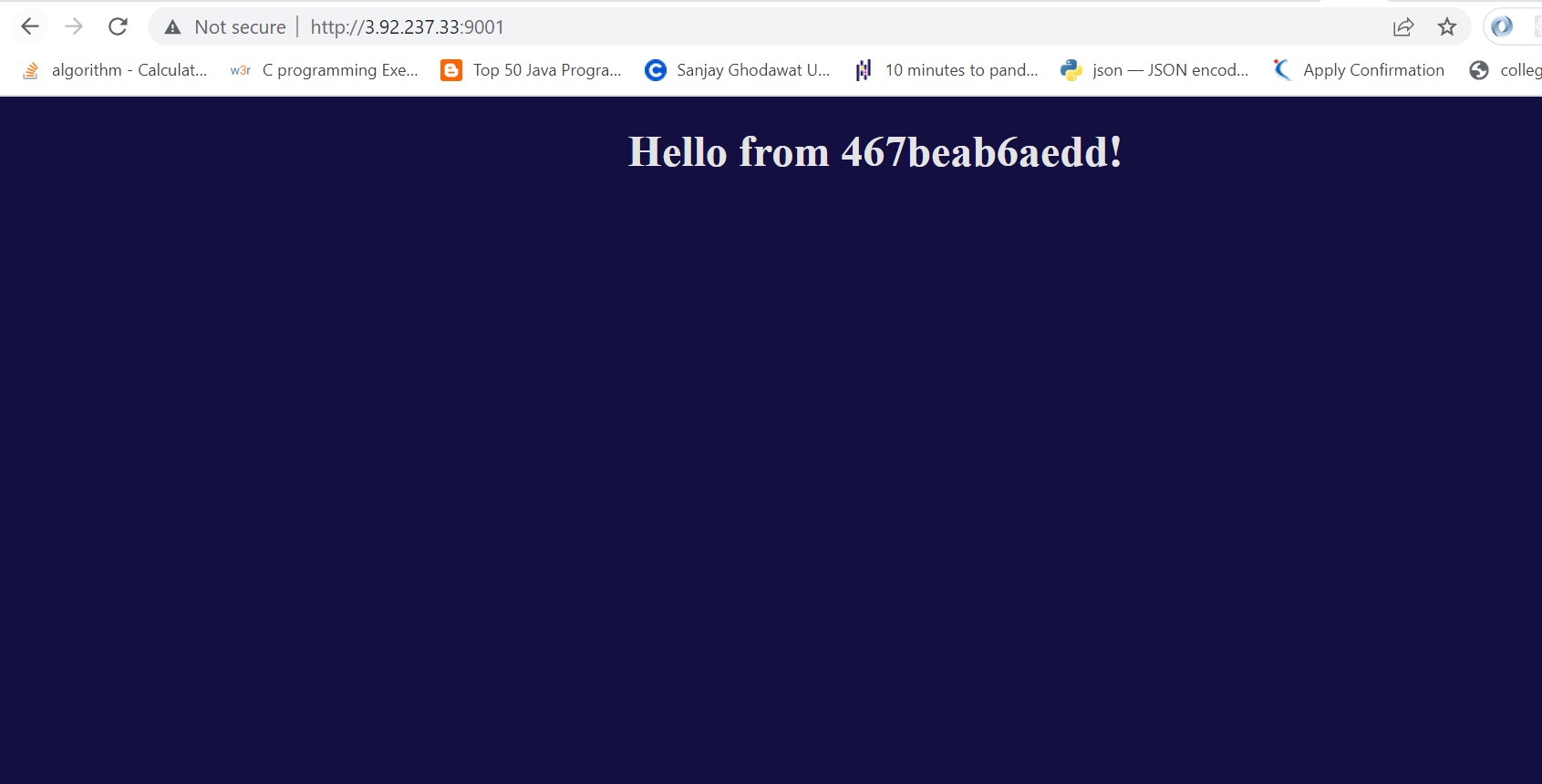
1] select ec2 instance checkbox and click security tab

2] click on security group

3]inside inbound rules add your mapped port number and select custom ip.



Open the public IPv4 address of your EC2 instance with port.no. 9001



* Push the image to a public or private repository (e.g., Docker Hub)

Before directly Pushing the image into DockerHub,first login using DockerHub account credentials.

docker login

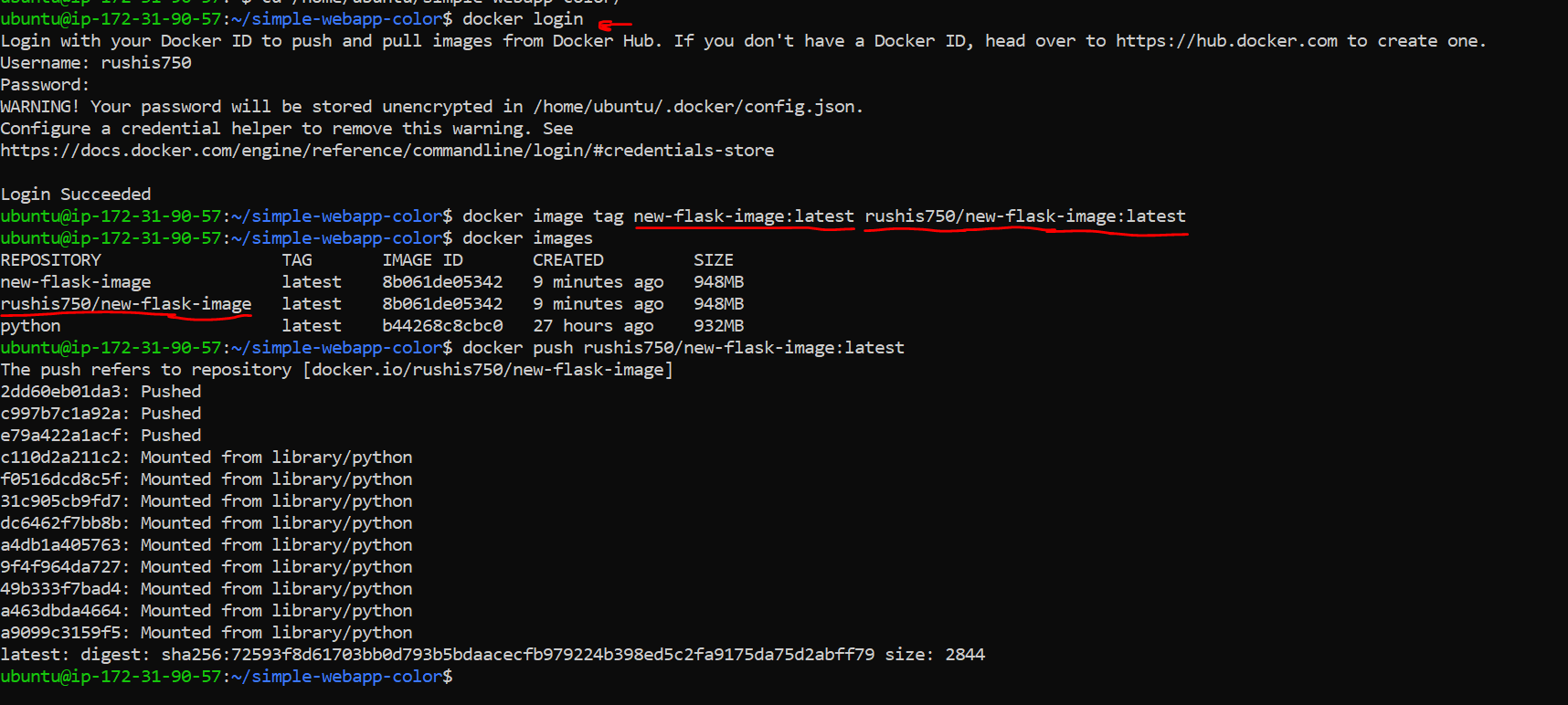
-add your username and password

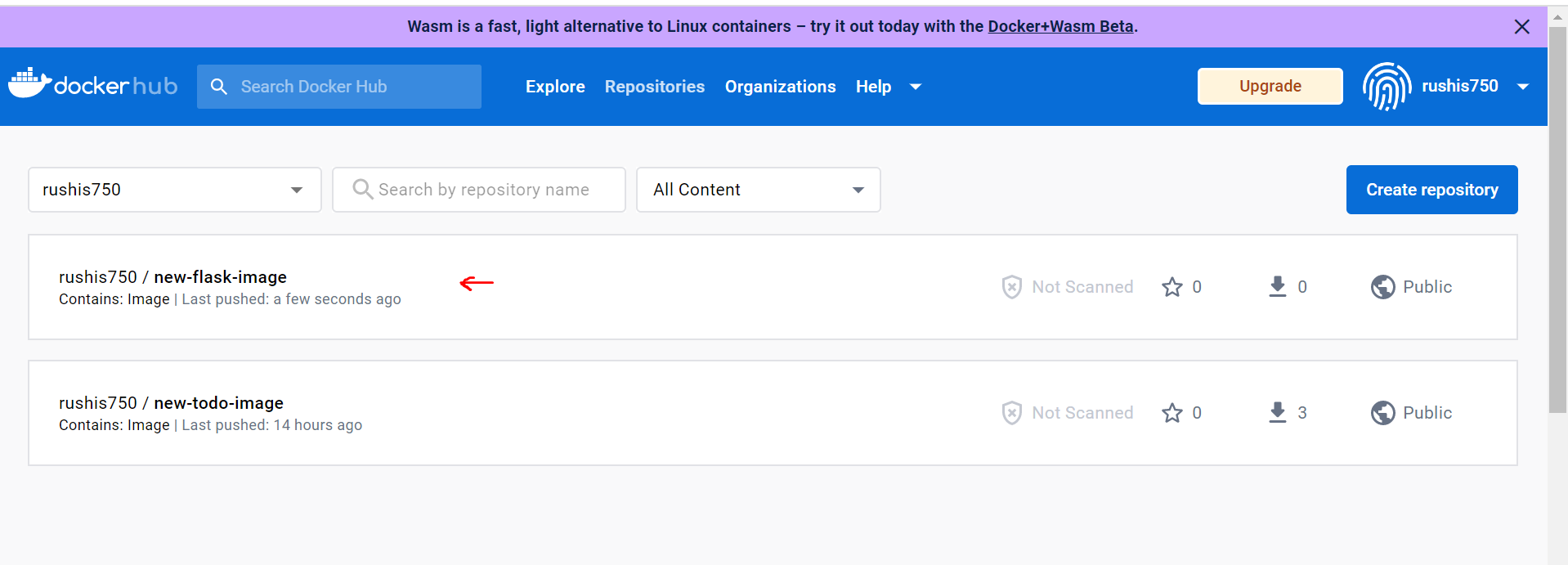
After that rename or tag your image

docker image tag <your image name> <dockerhub username>/<image Name>

Push this latest image to DockerHub

docker push <latest image name>





- ------------------------Happy Learning😊----------------------