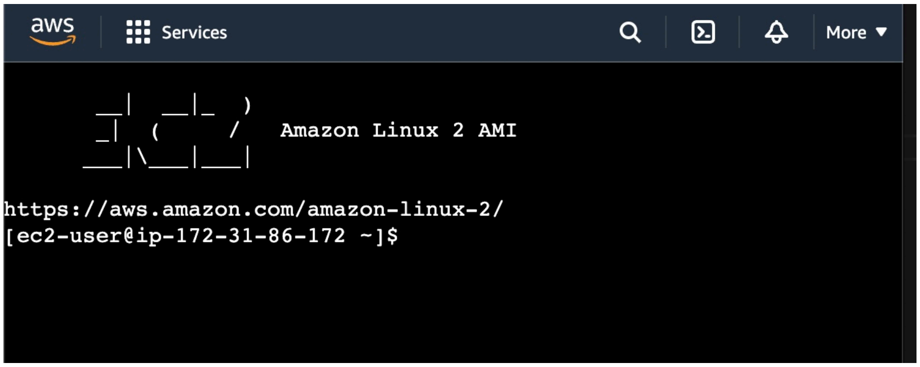
Docker Image Basic Operations

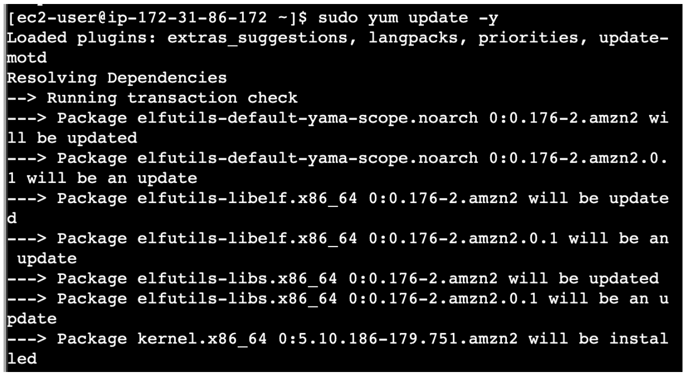
Part 1-Launch a Docker Machine Instance, Connect with SSH and Install Docker on Amazon Linux 2 EC2 Instance.

I launch a Docker machine on Amazon Linux 2 AMI with security group allowing SSH connections.



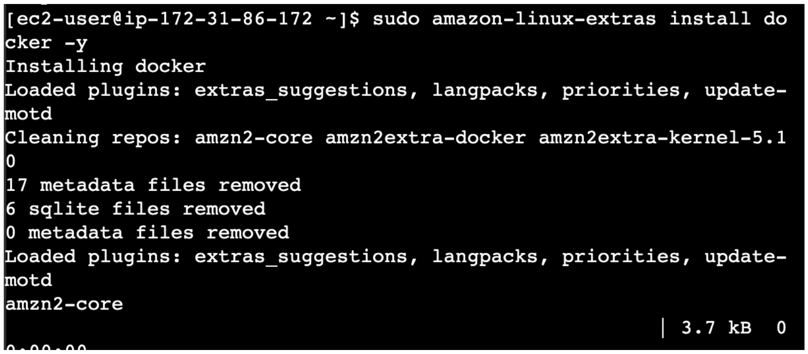
```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-86-172 ~]$
```

I update the installed packages and package cache on my instance by using code ‘sudo yum update -y’



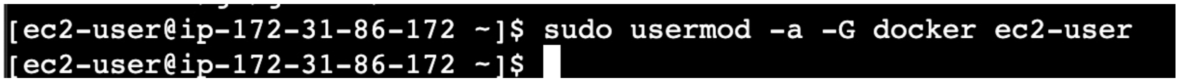
```
[ec2-user@ip-172-31-86-172 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-
motd
Resolving Dependencies
--> Running transaction check
---> Package elfutils-default-yama-scope.noarch 0:0.176-2.amzn2 wi
ll be updated
---> Package elfutils-default-yama-scope.noarch 0:0.176-2.amzn2.0.
1 will be an update
---> Package elfutils-libelf.x86_64 0:0.176-2.amzn2 will be update
d
---> Package elfutils-libelf.x86_64 0:0.176-2.amzn2.0.1 will be an
 update
---> Package elfutils-libs.x86_64 0:0.176-2.amzn2 will be updated
---> Package elfutils-libs.x86_64 0:0.176-2.amzn2.0.1 will be an u
pdate
---> Package kernel.x86_64 0:5.10.186-179.751.amzn2 will be instal
led
```

I install the most recent Docker Community Edition package by using code ‘sudo amazon-linux-extras install docker -y’



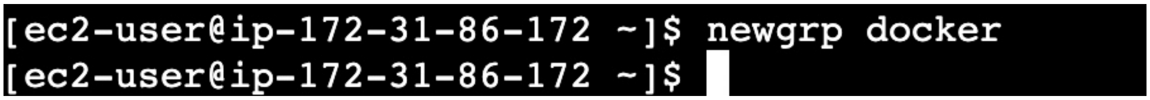
```
[ec2-user@ip-172-31-86-172 ~]$ sudo amazon-linux-extras install do
cker -y
Installing docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-
motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.1
0
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-
motd
amzn2-core
                                                       | 3.7 kB  0
0:00:00
```

I add the ec2-user to the docker group to run docker commands without using sudo. The code ‘sudo usermod -a -G docker ec2-user’



```
[ec2-user@ip-172-31-86-172 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-86-172 ~]$ 
```

Normally, the user needs to re-login into bash shell for the group docker to be effective, but newgrp command can be used activate docker group for ec2-user, not to re-login into bash shell.

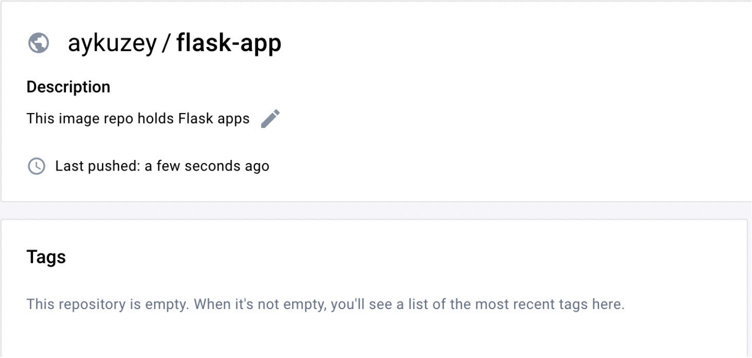


```
[ec2-user@ip-172-31-86-172 ~]$ newgrp docker
[ec2-user@ip-172-31-86-172 ~]$ 
```

Part 2 -Using Docker Image Commands and Docker Hub

I signed up to Docker Hub.

I create a repository with the name of flask-app and description of ‘This image repo holds Flask apps.’



aykuzey / **flask-app**

Description

This image repo holds Flask apps

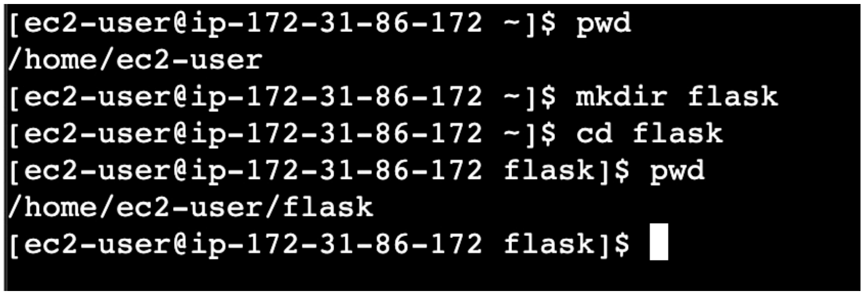
Last pushed: a few seconds ago

Tags

This repository is empty. When it's not empty, you'll see a list of the most recent tags here.

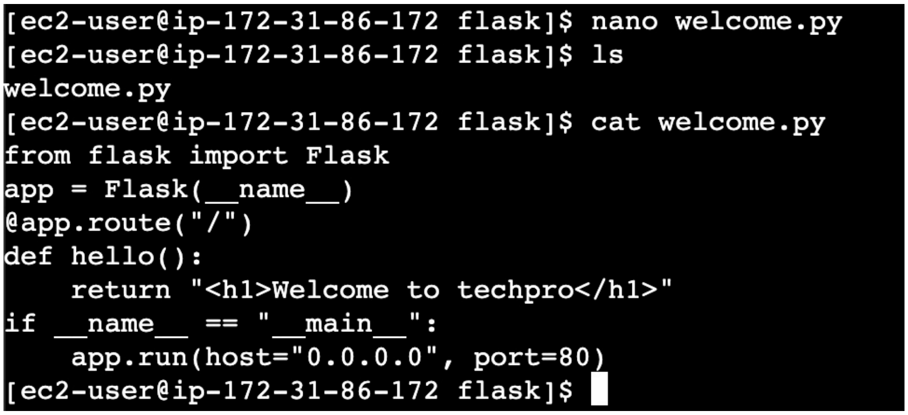
Part 3- Building Docker Images with Dockerfile

I create a folder to hold all files necessary for creating Docker image. Named flask on ec2 and get in to that folder.



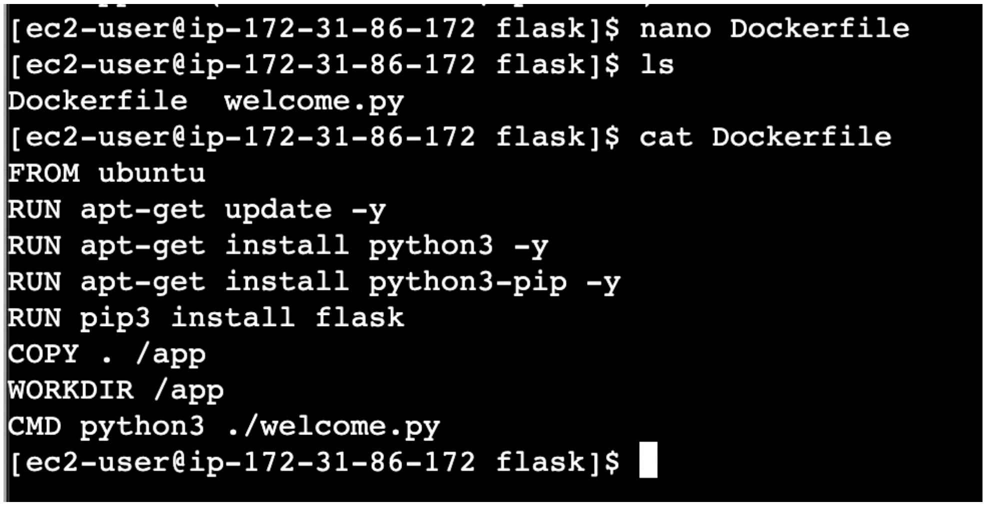
```
[ec2-user@ip-172-31-86-172 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-86-172 ~]$ mkdir flask
[ec2-user@ip-172-31-86-172 ~]$ cd flask
[ec2-user@ip-172-31-86-172 flask]$ pwd
/home/ec2-user/flask
[ec2-user@ip-172-31-86-172 flask]$ 
```

I used nano to create application code and save it to file, and named welcome.py and paste the code given.



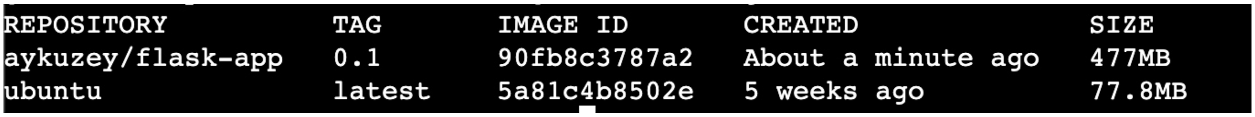
```
[ec2-user@ip-172-31-86-172 flask]$ nano welcome.py
[ec2-user@ip-172-31-86-172 flask]$ ls
welcome.py
[ec2-user@ip-172-31-86-172 flask]$ cat welcome.py
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "<h1>Welcome to techpro</h1>"
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80)
[ec2-user@ip-172-31-86-172 flask]$
```

I create a Dockerfile listing necessary packages and modules, and name it Dockerfile.



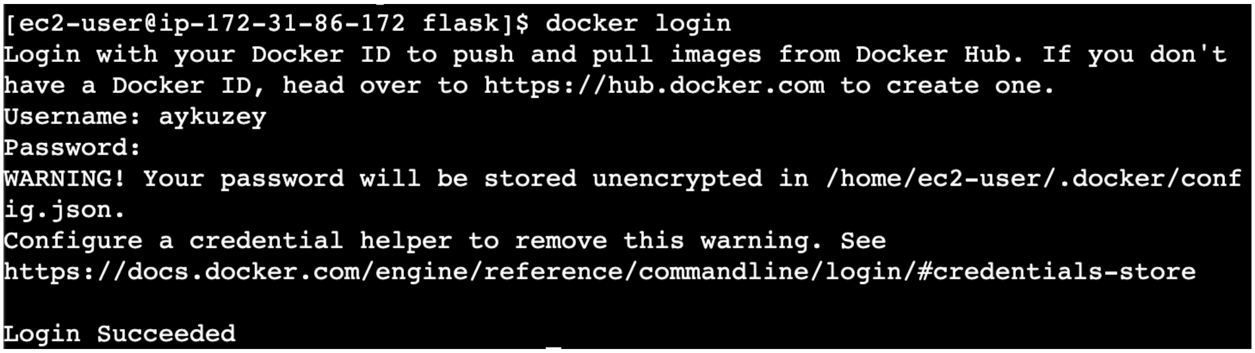
```
[ec2-user@ip-172-31-86-172 flask]$ nano Dockerfile
[ec2-user@ip-172-31-86-172 flask]$ ls
Dockerfile  welcome.py
[ec2-user@ip-172-31-86-172 flask]$ cat Dockerfile
FROM ubuntu
RUN apt-get update -y
RUN apt-get install python3 -y
RUN apt-get install python3-pip -y
RUN pip3 install flask
COPY . /app
WORKDIR /app
CMD python3 ./welcome.py
[ec2-user@ip-172-31-86-172 flask]$
```

Now I can build the flask-app by using command of ‘docker build -t “<dockerhubusername>/flask-app:0.1” .



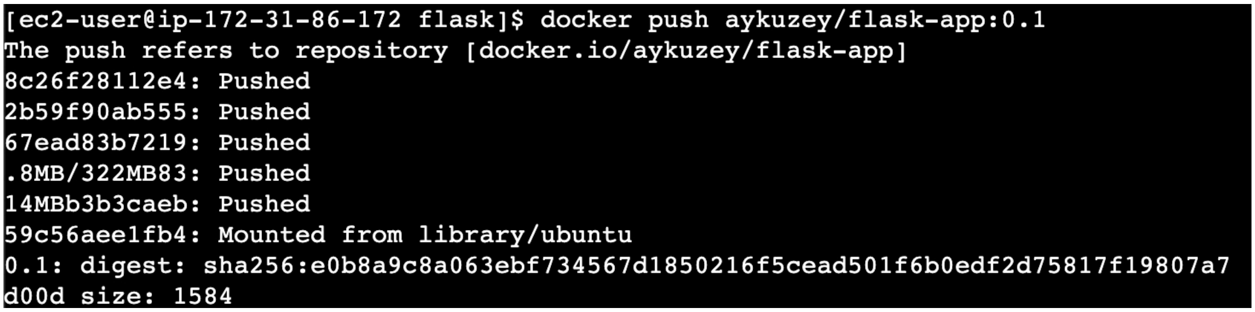
```
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
aykuzey/flask-app   0.1       90fb8c3787a2   About a minute ago   477MB
ubuntu              latest    5a81c4b8502e   5 weeks ago          77.8MB
```

In order to send something to Ducker, I have to login from ec2 machine by using code ‘docker login’

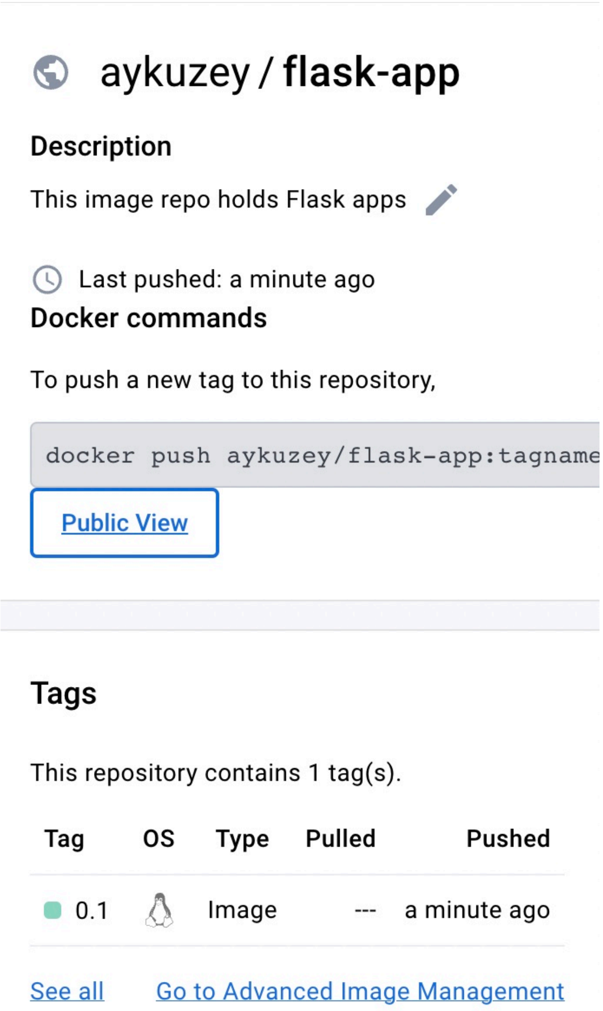


```
[ec2-user@ip-172-31-86-172 flask]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: aykuzey
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/conf
ig.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

I push newly built image to Docker Hub and show the updated repo on Docker Hub by using command ‘ docker push <dockerhubusername>/flask-app:0.1’



```
[ec2-user@ip-172-31-86-172 flask]$ docker push aykuzey/flask-app:0.1
The push refers to repository [docker.io/aykuzey/flask-app]
8c26f28112e4: Pushed
2b59f90ab555: Pushed
67ead83b7219: Pushed
.8MB/322MB83: Pushed
14MBb3b3caeb: Pushed
59c56aee1fb4: Mounted from library/ubuntu
0.1: digest: sha256:e0b8a9c8a063ebf734567d1850216f5cead501f6b0edf2d75817f19807a7
d00d size: 1584
```



aykuzey / **flask-app**

Description

This image repo holds Flask apps

Last pushed: a minute ago

Docker commands

To push a new tag to this repository,

```
docker push aykuzey/flask-app:tagname
```

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
● 0.1	🐧	Image	---	a minute ago