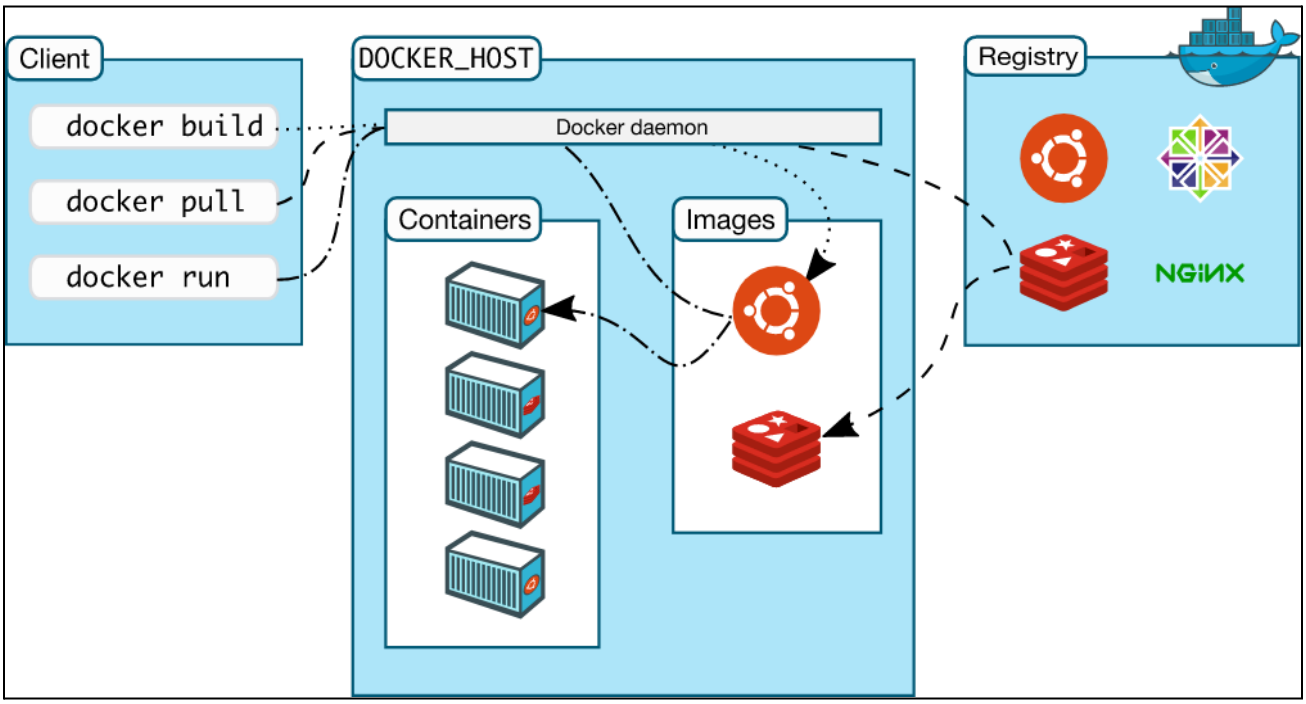
Create Docker Image in a container and expose to custom port - 500

- **What is Docker ?**

    - Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
    - Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows you to run many containers simultaneously on a given host.

- **Docker Architecture**



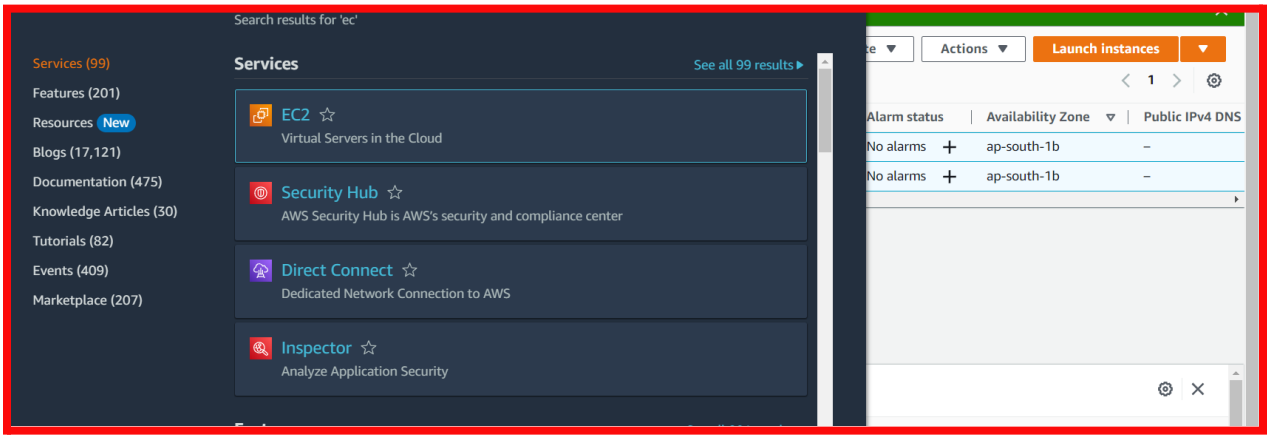
Docker Architecture

- The Docker architecture uses a client-server model and comprises of the Docker Client, Docker Host, Network and Storage components, and the Docker Registry / Hub.
- Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.
- The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.
- The Docker client and daemon communicate using a REST API, over UNIX sockets, or a network interface. Another Docker client is Docker Compose, which lets you work with applications consisting of a set of containers.

Note:For more information reference link - https://docs.docker.com/get-started/overview/

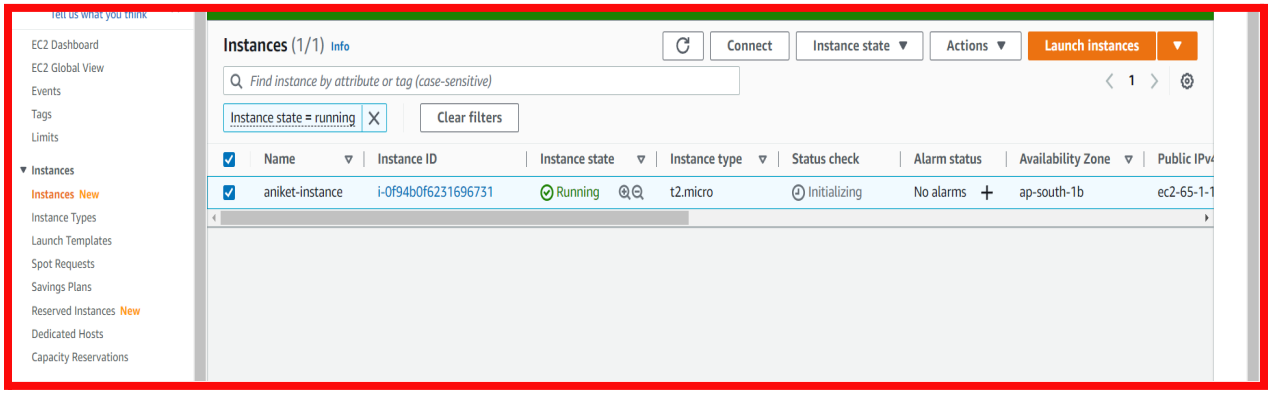
➢ **TASK :-** Create a docker image and run in a container then expose to custom port no. **5000**

- **Step to create EC2 instance:**

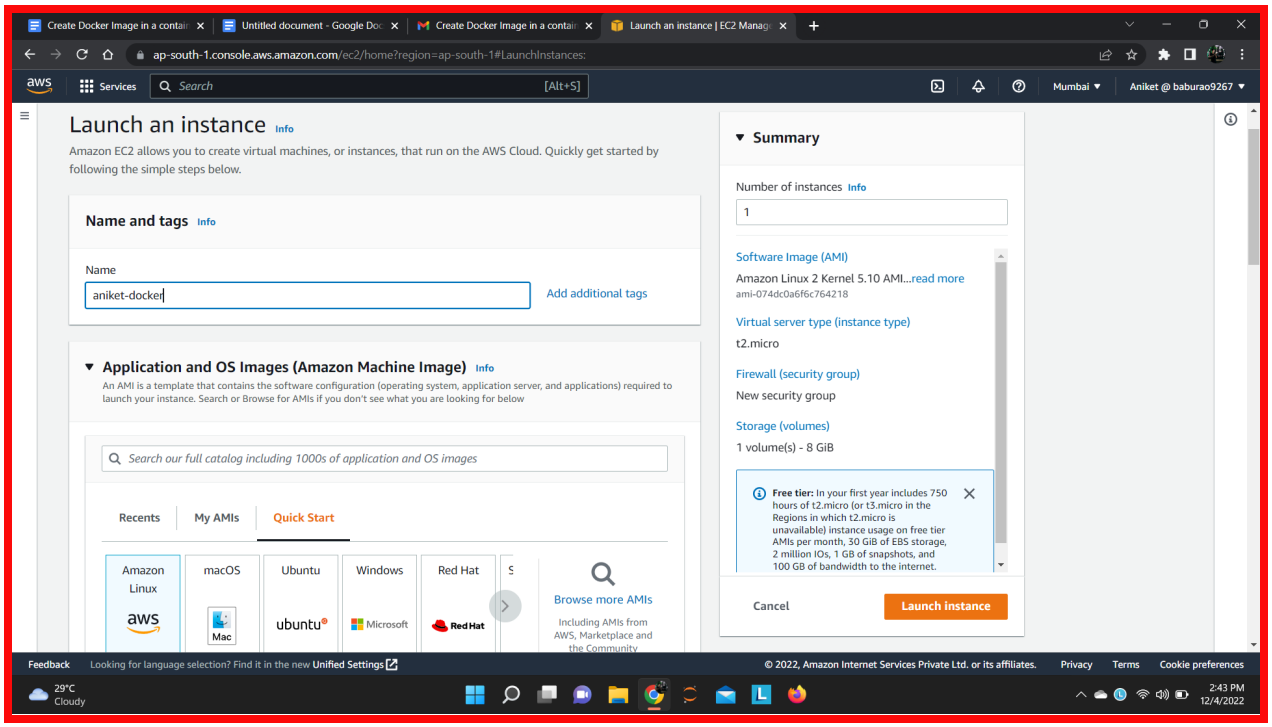


- Go to EC2 dashboard

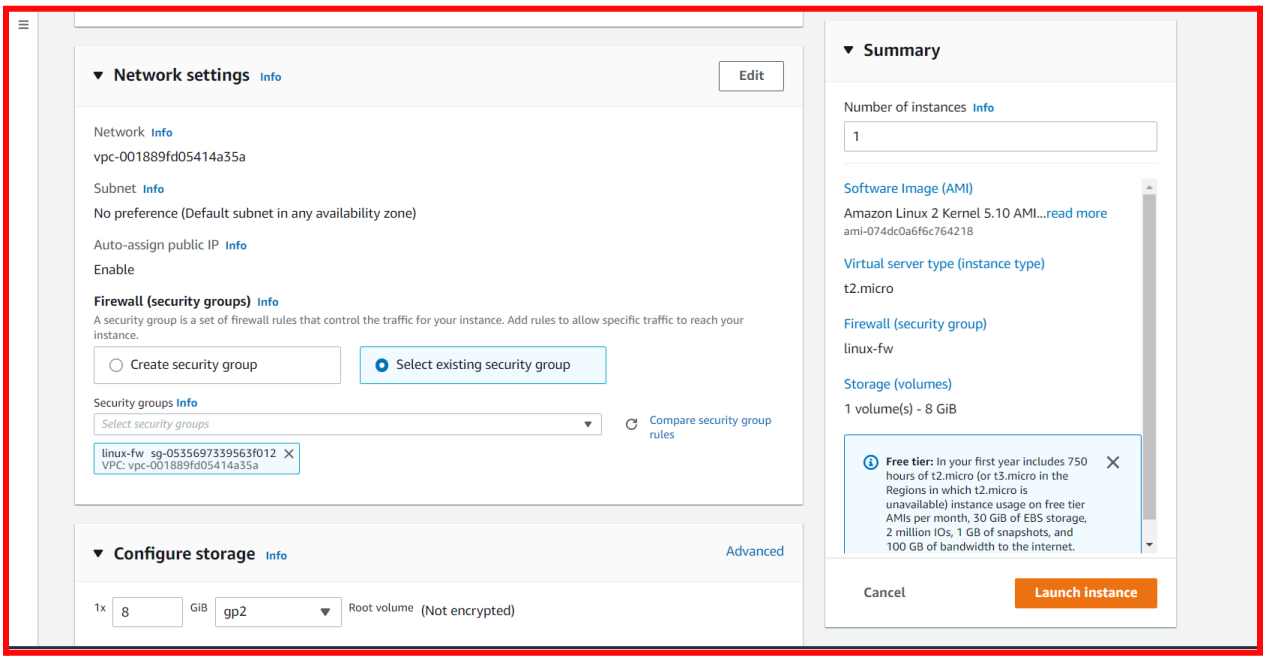
- Launch an EC2 instance



- For launch give proper name for an instance
- Select AMI (**Amazon Machine Image**)is required to launch an instance .
- Select free tier **instance type**(Eg. t2.micro..)
- Select the key pair if not, create it and download the **key** pair file and save in your system.



- In network settings, select existing security or create a new security group and set it as **default**.



- Launch **instance**.

- Connect to an SSH client. In my case I use **Putty** as an ssh terminal.

- Login as **ec2-user.**

```
ec2-user@ip-172-31-12-187:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Fri Nov 25 11:10:21 2022 from 49.15.250.12

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

https://aws.amazon.com/amazon-linux-2/
```

- **The following linux commands are use to create a docker image:**

  - **Sudo yum update -y :** install or update all the packages on the system. **-y** is used for yes to install the packages.
  - **Sudo yum install docker -y :** To install Docker engine on the system.
  - **Sudo docker info** : used for information about the docker.
  - Which docker : about docker version
  - **Sudo usermod -aG docker ec2-user :** Add the ec2-user to the docker group so you can execute Docker commands without using sudo.
  - Log out and log back in again to pick up the new docker group permissions. You can accomplish this by closing your current SSH terminal window and reconnecting to your instance in a new one. Your new SSH session will have the appropriate docker group permissions.
  - **Sudo systemctl start docker** : used to start the docker service
  - **docker info**: Verify that you can run Docker commands without sudo.

- **Create a Docker Image:**

Amazon ECS task definitions use **Docker** images to launch containers on the container instances in your clusters. In this section, you create a Docker image of a simple web application, test it on your local system or Amazon EC2 instance, and then push the image to the **Amazon ECR** container registry so you can use it in an Amazon ECS task definition.

```
ec2-user@ip-172-31-12-187:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Fri Nov 25 11:10:21 2022 from 49.15.250.12

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-12-187 ~]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
[ec2-user@ip-172-31-12-187 ~]$ sudo yum install docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package docker-20.10.17-1.amzn2.0.1.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-12-187 ~]$ sudo docker info
Client:
 Context:    default
 Debug Mode: false

Server:
ERROR: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daem
on running?
errors pretty printing info
[ec2-user@ip-172-31-12-187 ~]$ sudo usermod -aG docker ec2-user
[ec2-user@ip-172-31-12-187 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-12-187 ~]$ sudo docker info
Client:
 Context:    default
 Debug Mode: false

Server:
 Containers: 8
  Running: 0
  Paused: 0
  Stopped: 8
 Images: 2
 Server Version: 20.10.17
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
20°C
Haze
```

- Here we create a docker image for a **public repository** and run in a container and exposed to custom port 500.

ec2-user@ip-172-31-12-187:~

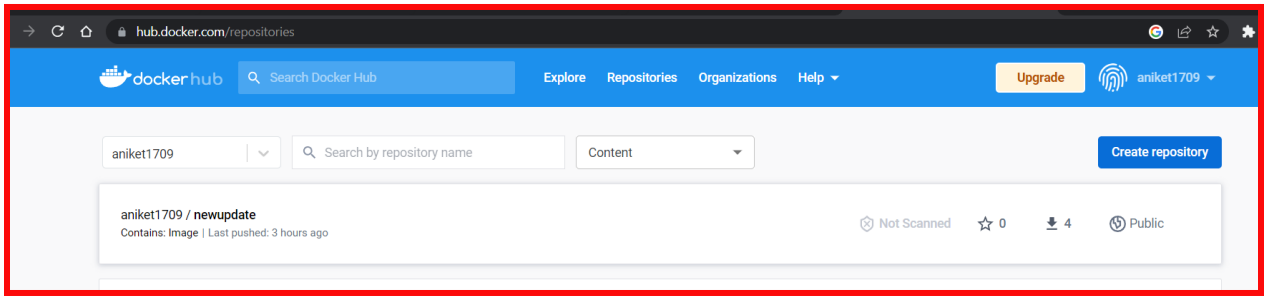
```
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

[ec2-user@ip-172-31-12-187 ~]$
[ec2-user@ip-172-31-12-187 ~]$ vi index.html
[ec2-user@ip-172-31-12-187 ~]$ ls
Dockerfile  index.html
[ec2-user@ip-172-31-12-187 ~]$ cd Dockerfile
-bash: cd: Dockerfile: Not a directory
[ec2-user@ip-172-31-12-187 ~]$ cd index.html
-bash: cd: index.html: Not a directory
[ec2-user@ip-172-31-12-187 ~]$ mkdir sheet
[ec2-user@ip-172-31-12-187 ~]$ ls
Dockerfile  index.html  sheet
[ec2-user@ip-172-31-12-187 ~]$ vi Dockerfile
[ec2-user@ip-172-31-12-187 ~]$ docker images
REPOSITORY             TAG       IMAGE ID       CREATED       SIZE
aniket1709/my_empire   1         c477f4e0cada   6 hours ago   145MB
httpd                  latest    8653efc8c72d   10 days ago   145MB
[ec2-user@ip-172-31-12-187 ~]$ docker rm aniket1709/my_empire:1
Error: No such container: aniket1709/my_empire:1
[ec2-user@ip-172-31-12-187 ~]$ docker images
REPOSITORY             TAG       IMAGE ID       CREATED       SIZE
aniket1709/my_empire   1         c477f4e0cada   6 hours ago   145MB
httpd                  latest    8653efc8c72d   10 days ago   145MB
[ec2-user@ip-172-31-12-187 ~]$ docker build -t aniket1709/newupdate:1
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
[ec2-user@ip-172-31-12-187 ~]$ docker build -t aniket1709/newupdate:1 .
Sending build context to Docker daemon  16.38kB
Step 1/2 : FROM nginx
latest: Pulling from library/nginx
a603fa5e3b41: Already exists
c39e1cda007e: Pull complete
90cfefba34d7: Pull complete
a38226fb7aba: Pull complete
62583498bae6: Pull complete
9802a2cfdb8d: Pull complete
Digest: sha256:e209ac2f37c70c1e0e9873a5f7231e91dcd83fdf1178d8ed36c2ec09974210ba
Status: Downloaded newer image for nginx:latest
 ---> 88736fe82739
Step 2/2 : COPY index.html /usr/share/nginx/html
```

20°C
Haze

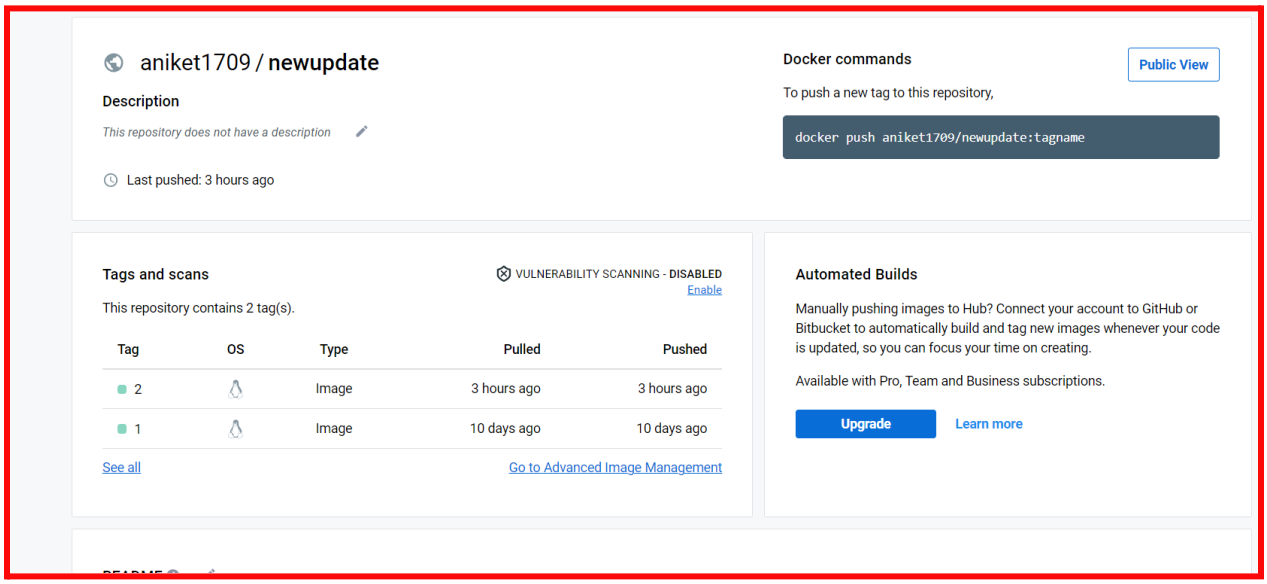
- Now ,edit a new **index.html** file then add content for the web page image.

- **vi / vim / nano :** this are text editor to edit content inside in the file
- **vi index.html :** open file with vi editor and add the content to displayed on webpage
- **ls :**list the directory and files in the system
- **vi Dockerfie:** create a directory and in editor copy path from **index.html** file.
- **docker images :** show all top level images, their repository and tags, and their size.
- **docker build -t aniket1709/newupdate:1 . :** build the docker image in the current repository in docker hub.

★ **Note**: Before run above commands you have to create dockerhub account.



dockerhub repository

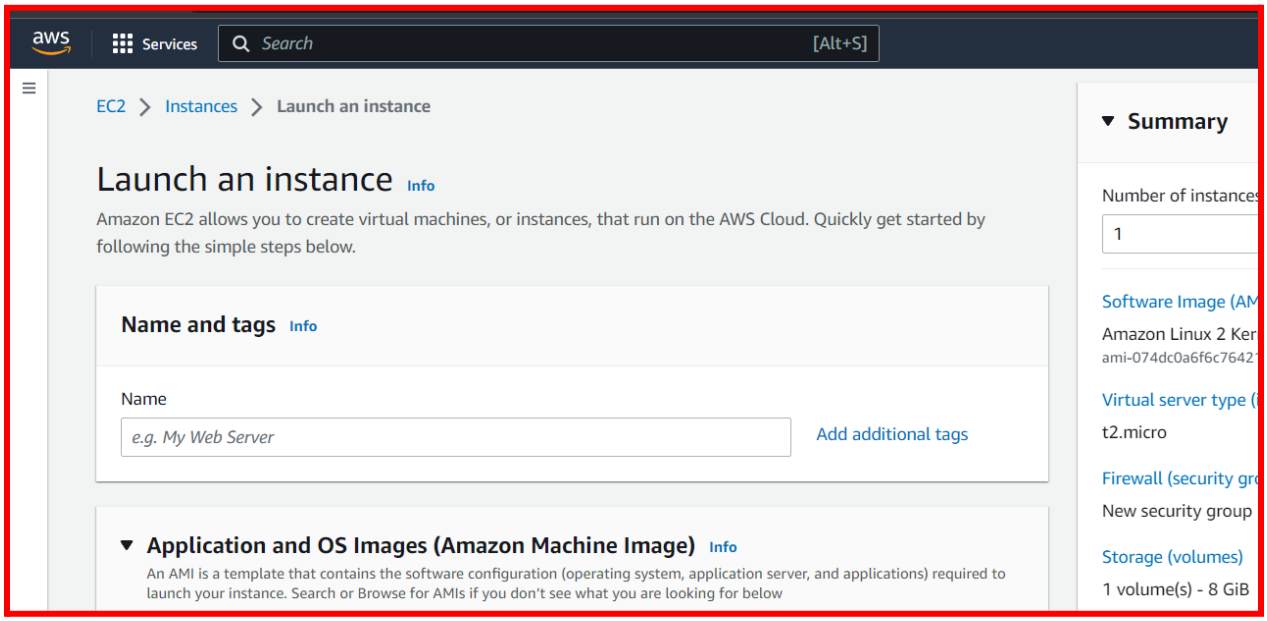


dockerhub repository tags or versions

```
ec2-user@ip-172-31-12-187:~
Step 2/2 : COPY index.html /usr/share/nginx/html
 ---> 5a67b92a34af
Successfully built 5a67b92a34af
Successfully tagged aniket1709/newupdate:1
[ec2-user@ip-172-31-12-187 ~]$ docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
aniket1709/newupdate    1         5a67b92a34af   13 seconds ago   142MB
aniket1709/my_empire    1         c477f4e0cada   6 hours ago      145MB
httpd                   latest    8653efc8c72d   10 days ago      145MB
nginx                   latest    88736fe82739   10 days ago      142MB
[ec2-user@ip-172-31-12-187 ~]$ docker push aniket1709/newupdate:1
The push refers to repository [docker.io/aniket1709/newupdate]
fbcc591834ce: Pushed
6cffb086835a: Mounted from library/nginx
e2d75d87993c: Mounted from library/nginx
5a5bafd53f76: Mounted from library/nginx
f86e88a471f4: Mounted from library/nginx
f7ed3797e296: Mounted from library/nginx
ec4a38999118: Mounted from aniket1709/my_empire
1: digest: sha256:a11405de6a446a7ee5e4bafe0c45fbdd49705d580609bf0c65a860518a0e1ef4 size: 1777
[ec2-user@ip-172-31-12-187 ~]$
[ec2-user@ip-172-31-12-187 ~]$
[ec2-user@ip-172-31-12-187 ~]$
[ec2-user@ip-172-31-12-187 ~]$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-12-187 ~]$
[ec2-user@ip-172-31-12-187 ~]$
[ec2-user@ip-172-31-12-187 ~]$
[ec2-user@ip-172-31-12-187 ~]$ docker build -t aniket1709/newupdate:1
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
[ec2-user@ip-172-31-12-187 ~]$ docker build -t aniket1709/newupdate:1 .
Sending build context to Docker daemon  16.38kB
Step 1/2 : FROM nginx
 ---> 88736fe82739
Step 2/2 : COPY index.html /usr/share/nginx/html
 ---> Using cache
 ---> 5a67b92a34af
Successfully built 5a67b92a34af
Successfully tagged aniket1709/newupdate:1
20°C
Haze
```

- **docker push aniket1709/newupdate:1 :** push an image or a repository to a registry
- **Authentication :** to push an image to the registry we required authentication. After login we can push the docker image.
- **docker login :** Log in with dockerhub account login details.

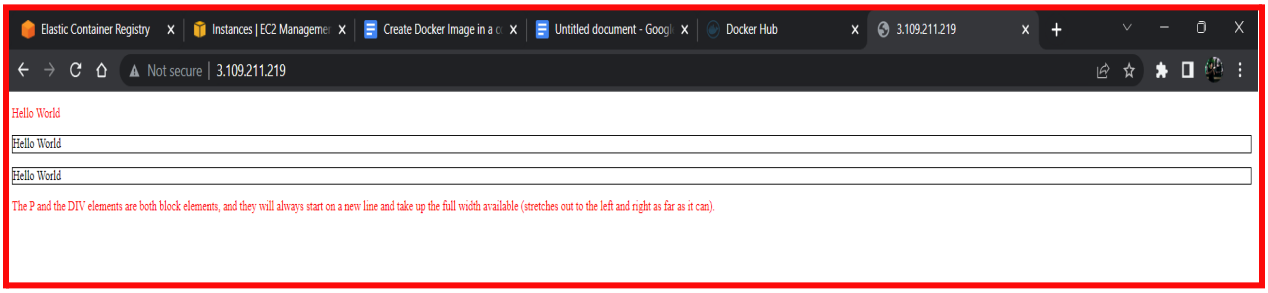
- **docker push aniket1709/newupdate:1 :** After authentication succeeded we can push docker image.



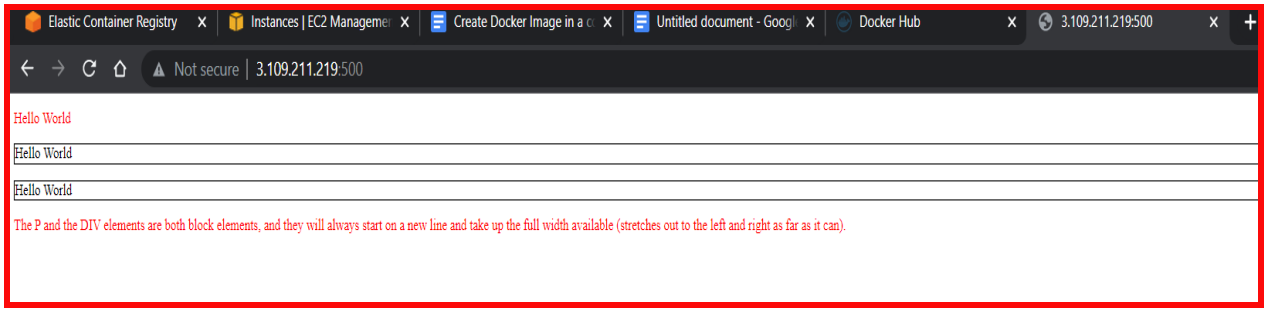
- Launch a new **instance.**
- Connect the **SSH client.**
- Apply same commands as we applied before upto **docker info**
- **docker pull aniket1709/newupdate:1** : Pull an image or a repository from a registry
- **docker images** : This command is used to display all the images currently installed on the system.
- Now run the docker image in a container
- **docker run -d -p 80:80 aniket709/newupdate:1**
- Now check the web server.
- **docker run -d -p 500:80 aniket709/newupdate:1** : run and exposed to custom port no. 500.

```
root@ip-172-31-7-252:/home/ec2-user
[root@ip-172-31-7-252 ec2-user]# docker pull aniket1709/newupdate:2
2: Pulling from aniket1709/newupdate
Digest: sha256:99ba23b510e2294969db75e7e607588b1ca4f621c7eb5889cc39d6fab3077365
Status: Image is up to date for aniket1709/newupdate:2
docker.io/aniket1709/newupdate:2
[root@ip-172-31-7-252 ec2-user]# docker images
REPOSITORY                                                  TAG       IMAGE ID       CREATED
     SIZE
784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo   1         46951748d1a8   3 hours a
go   142MB
aniket1709/newupdate                                        2         46951748d1a8   3 hours a
go   142MB
[root@ip-172-31-7-252 ec2-user]# docker run -d -p 80:80 ^C
[root@ip-172-31-7-252 ec2-user]# docker run  -f -p 80:80 aniket1709/newupdate:2
unknown shorthand flag: 'f' in -f
See 'docker run --help'.
[root@ip-172-31-7-252 ec2-user]#
[root@ip-172-31-7-252 ec2-user]#
[root@ip-172-31-7-252 ec2-user]# docker run  -d -p 80:80 aniket1709/newupdate:2
6825bb35abdfcbb8b6eea06b46dab2f61df463fc0b306b57a328db354213b92d
[root@ip-172-31-7-252 ec2-user]#
```

★ **In the above image, we created a new repository to run docker image otherwise the same process applied.**



Docker image webpage



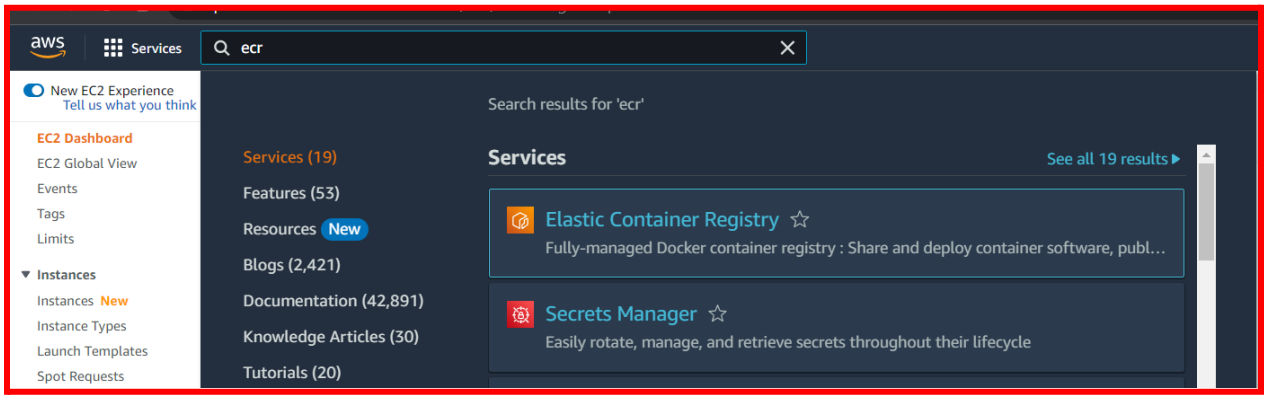
Docker image is exposed on port 500.

★ **Thus we create docker image in a container and then expose to custom port number 500.**

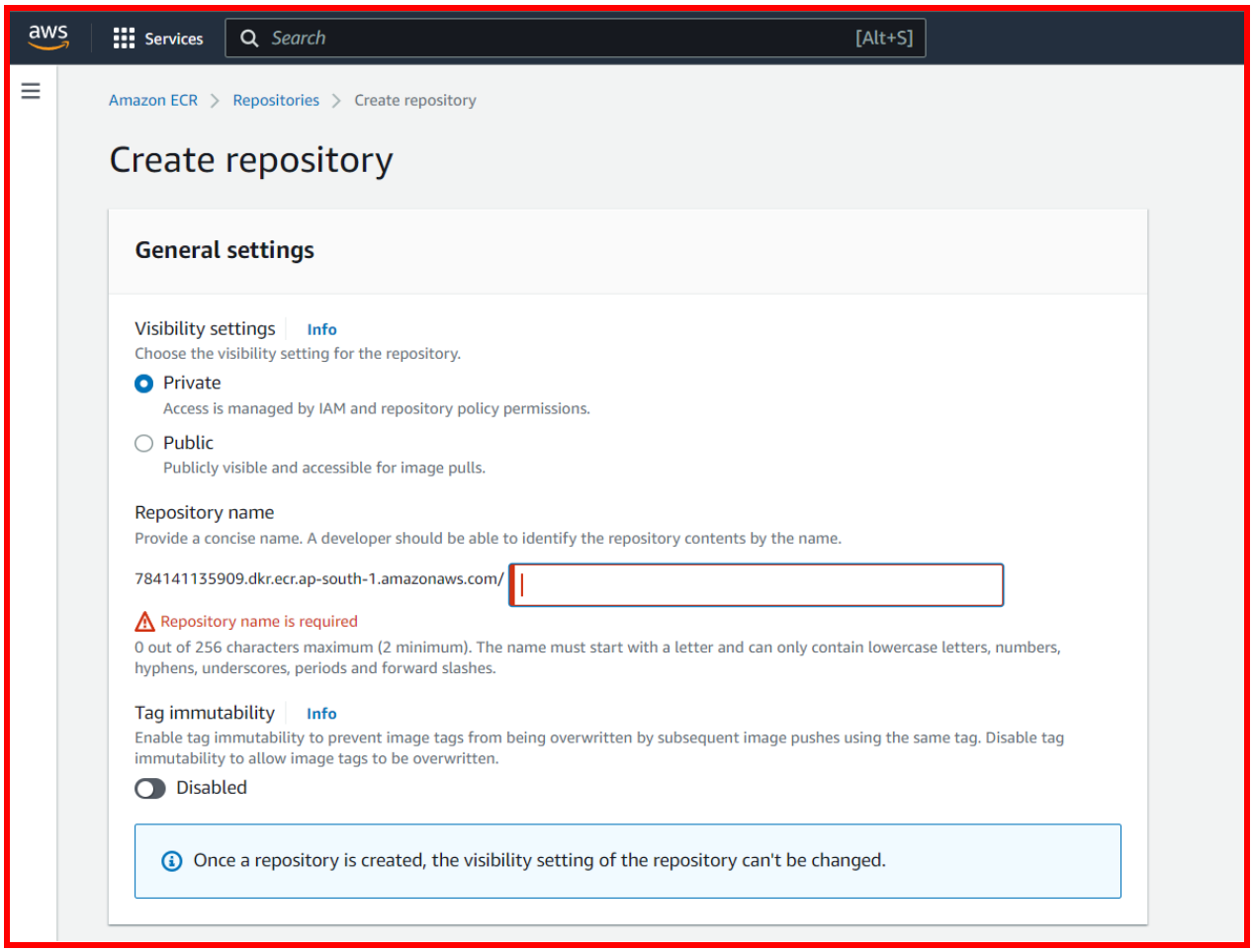
★ Now stop the docker machine.

★ Now create a docker image in a container with a **private repository.**

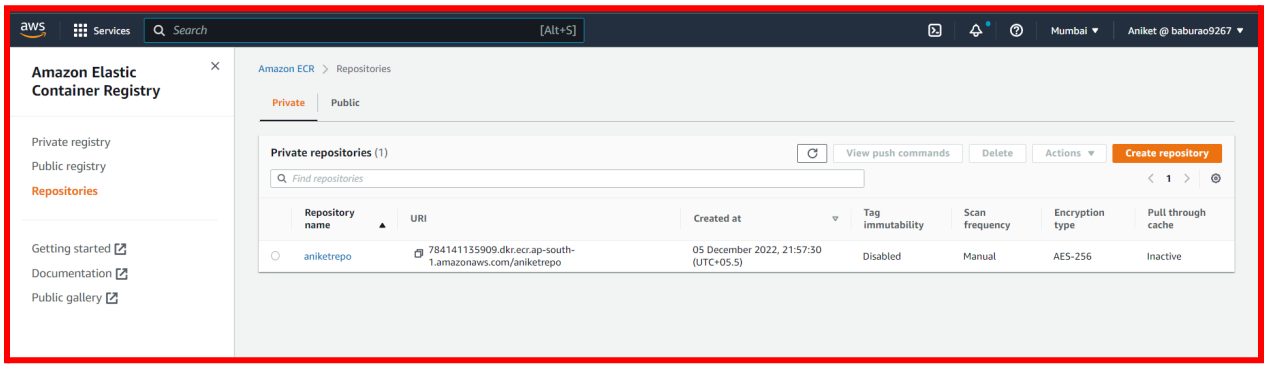
- **For private repository** we required authentication when we push and pull docker image.
- Go to **Elastic Container Registry (ECR)**



- In ECR go to repositories.
- Create a new **repository**.
- Give the proper name and click on create repository.



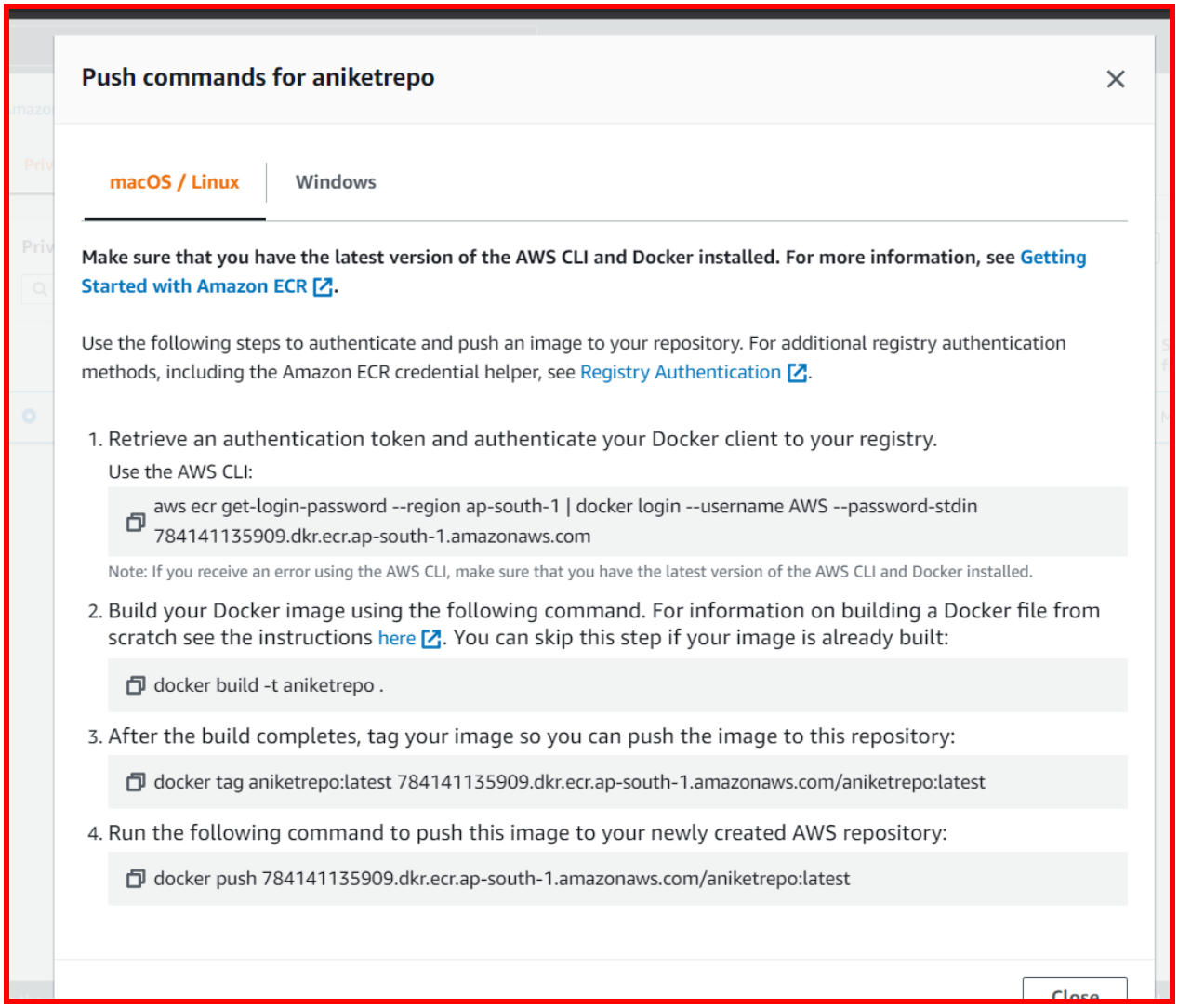
- The repository created.



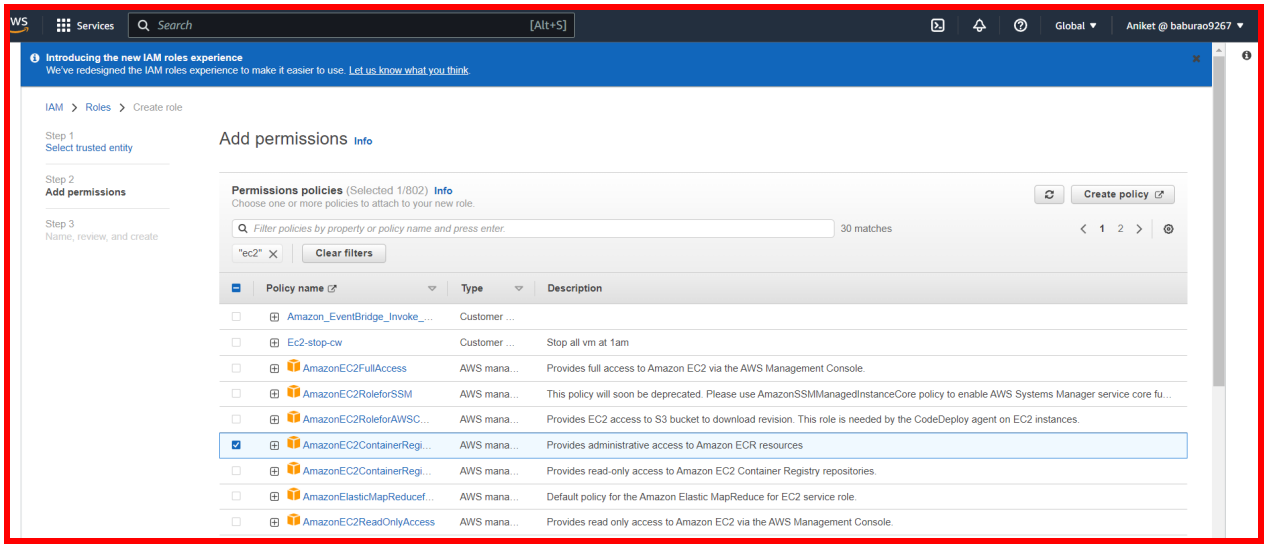
- Reconnect 1st instance SSH client
- Now, build an image of the recently created repository.

```
[root@ip-172-31-5-206 ec2-user]#
[root@ip-172-31-5-206 ec2-user]# ls
Dockerfile  index.html
[root@ip-172-31-5-206 ec2-user]# docker build -t 784141135909.dkr.ecr.ap-south-1
.amazonaws.com/aniketrepo:1 .
Sending build context to Docker daemon  8.704kB
Step 1/2 : FROM nginx
 ---> 88736fe82739
Step 2/2 : COPY index.html /usr/share/nginx/html
 ---> Using cache
 ---> 46951748d1a8
Successfully built 46951748d1a8
Successfully tagged 784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo:1
[root@ip-172-31-5-206 ec2-user]# docker images
REPOSITORY                                                  TAG       IMAGE ID
    CREATED          SIZE
784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo   1         46951748d1a
8   16 minutes ago   142MB
aniket1709/newupdate                                        2         46951748d1a
8   16 minutes ago   142MB
nginx                                                       latest    88736fe8273
9   2 weeks ago      142MB
[root@ip-172-31-5-206 ec2-user]# docker push 784141135909.dkr.ecr.ap-south-1.ama
zonaws.com/aniketrepo:1
The push refers to repository [784141135909.dkr.ecr.ap-south-1.amazonaws.com/ani
ketrepo]
036b32a8f6cb: Preparing
6cffb086835a: Preparing
e2d75d87993c: Preparing
5a5bafd53f76: Preparing
f86e88a471f4: Preparing
f7ed3797e296: Preparing
ec4a38999118: Preparing
no basic auth credentials
[root@ip-172-31-5-206 ec2-user]# aws ecr get-login-password --region ap-south-1
| docker login --username AWS --password-stdin 784141135909.dkr.ecr.ap-south-1.a
mazonaws.com
Unable to locate credentials. You can configure credentials by running "aws conf
igure".
Error: Cannot perform an interactive login from a non TTY device
[root@ip-172-31-5-206 ec2-user]#
```

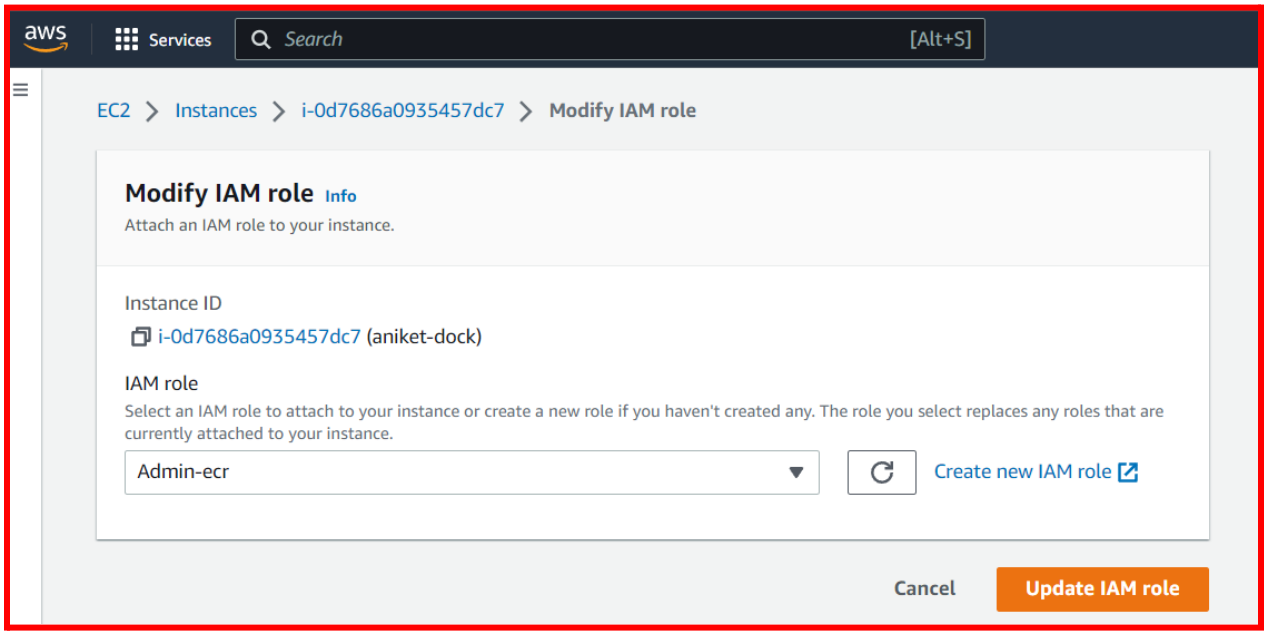
- **docker build -t 784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo:1 .** - build an image of **(aniketrepo:1)** current repository.
- **docker push 784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo:1** - push docker image but we do not have authentication credentials so first authenticate for push docker image.



- In the **ECR** repository click on **push commands** here you will find login commands.
- Copy and paste these commands but login auth failed because the instance does not have an **IAM** role for login so create an IAM role.
- Create IAM role
- Select **AWS service** then in users case select **EC2** then click next
- Add following **marked** permission and then give role name and create role.

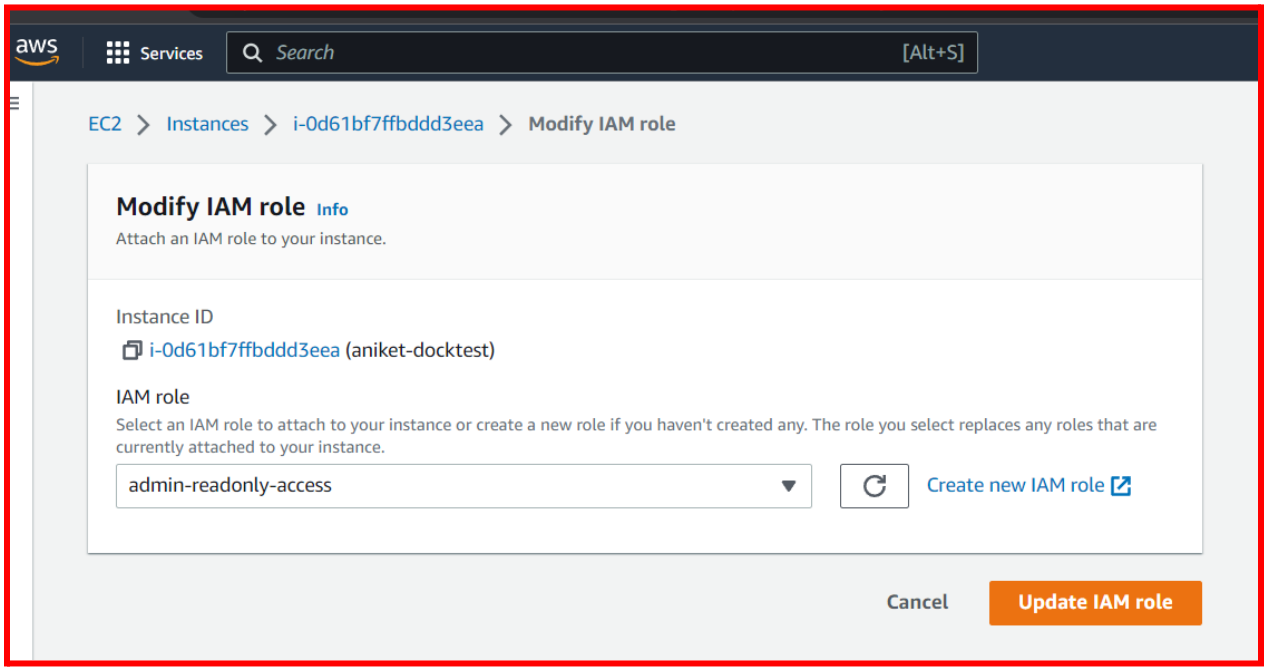


- Modify **IAM** role to server instance.



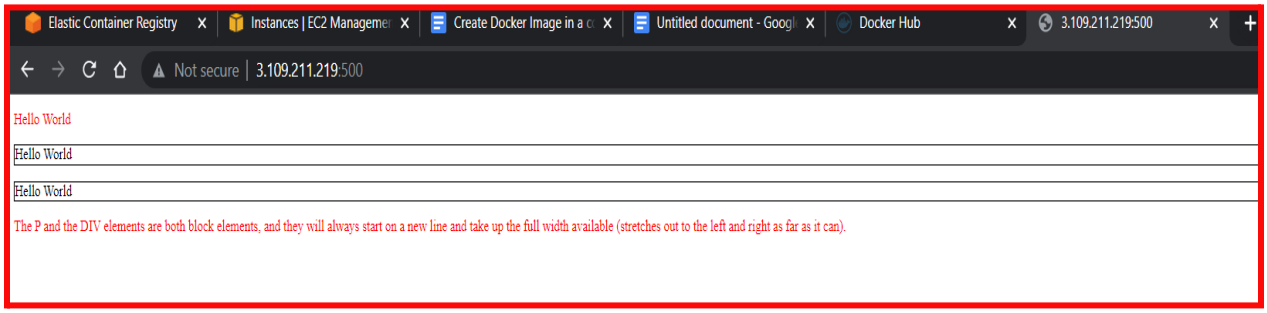
- Now, again login with push command and login succeeded .
- **docker pull 784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo:1 -** Pull an image or a repository from a registry
- The new docker image is pulled successfully.
- For the test server instance we have to modify the IAM role to **ECR read only access to an EC2 instance.**

- Create an **IAM** role then we can run docker image.



- **docker run -d -p 80:80 784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo:1 -** run an docker image in a container and deploy.
- **docker run -d -p 80:80 784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo:1 -**run a docker image in container and deployed at custom port 500**.**

```
root@ip-172-31-7-252:/home/ec2-user
[root@ip-172-31-7-252 ec2-user]# aws ecr get-login-password --region ap-south-1
| docker login --username AWS --password-stdin 784141135909.dkr.ecr.ap-south-1.a
mazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@ip-172-31-7-252 ec2-user]# docker pull 784141135909.dkr.ecr.ap-south-1.ama
zonaws.com/aniketrepo:1
1: Pulling from aniketrepo
Digest: sha256:99ba23b510e2294969db75e7e607588b1ca4f621c7eb5889cc39d6fab3077365
Status: Downloaded newer image for 784141135909.dkr.ecr.ap-south-1.amazonaws.com
/aniketrepo:1
784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo:1
[root@ip-172-31-7-252 ec2-user]# docker images
REPOSITORY                                                  TAG       IMAGE ID
    CREATED          SIZE
784141135909.dkr.ecr.ap-south-1.amazonaws.com/aniketrepo   1         46951748d1a
8   39 minutes ago   142MB
aniket1709/newupdate                                        2         46951748d1a
8   39 minutes ago   142MB
[root@ip-172-31-7-252 ec2-user]# docker run -d -p 80:80 784141135909.dkr.ecr.ap-south-1.amazonaws.c
om/aniketrepo:1
f0e5a86fa508550dbd9835a910b81e679794b5d3f56111e0fee810666fb851bd
docker: Error response from daemon: driver failed programming external connectivity on endpoint gra
cious_shamir (ac41cb781ef0dc0c35df410c1cf6526e9d0b0314ab71516fd04cfe5b4f3a08df): Bind for 0.0.0.0:8
0 failed: port is already allocated.
[root@ip-172-31-7-252 ec2-user]# docker run -d -p 80:80 784141135909.dkr.ecr.ap-south-1.amazonaws.c
om/aniketrepo:1
9a0af2e82c590816e12674bfd2519569d94d423956e077c0b4514d41e4d12fa6
docker: Error response from daemon: driver failed programming external connectivity on endpoint com
petent_liskov (cc89e9f205dacc6316a17dd6316bf403fc2c393c3bf821d02f560eec8307bd2b): Bind for 0.0.0.0:
80 failed: port is already allocated.
[root@ip-172-31-7-252 ec2-user]# docker run -d -p 500:80 784141135909.dkr.ecr.ap-south-1.amazonaws.
com/aniketrepo:1
f4252d3509047b8760251c8555a966d02553d997cbc06c8cdefad1bbad08437c
[root@ip-172-31-7-252 ec2-user]#
```



★ **Thus we create docker image in private repository and run in a container and then expose to custom port number 500.**