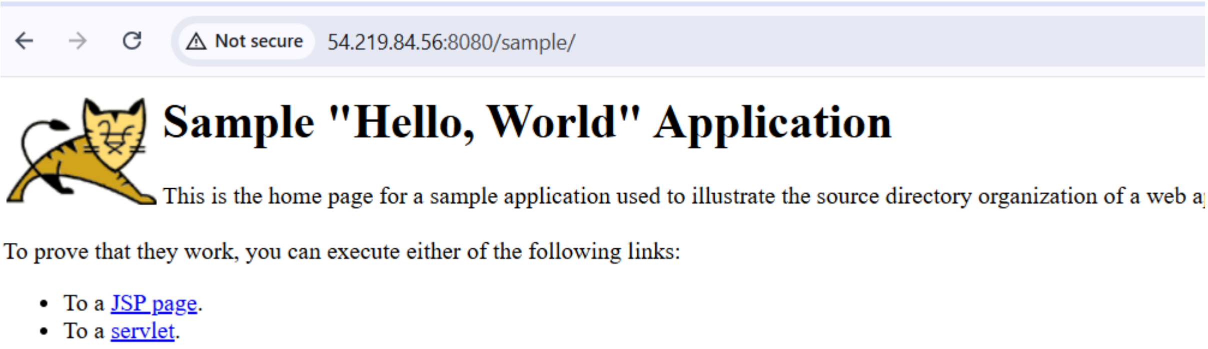
DOCKER 02

1. Create a tomcat container on 8080 and deploy sample application in tomcat.

2.Run the Tomcat container on port 8080

- docker run -d –name mytomcat -p 8080:8080 tomcat:
- port changed to 8080
- Deploy a sample application (a .war file)
- wget https://tomcat.apache.org/tomcat-9.0-doc/appdev/sample/sample.war
- sample war file downloaded
- copy it to webapps location .
- docker cp sample.war mytomcat:/usr/local/tomcat/webapps/
- now,
- copy ip and add 8080 and /sample in browser



Sample "Hello, World" Application

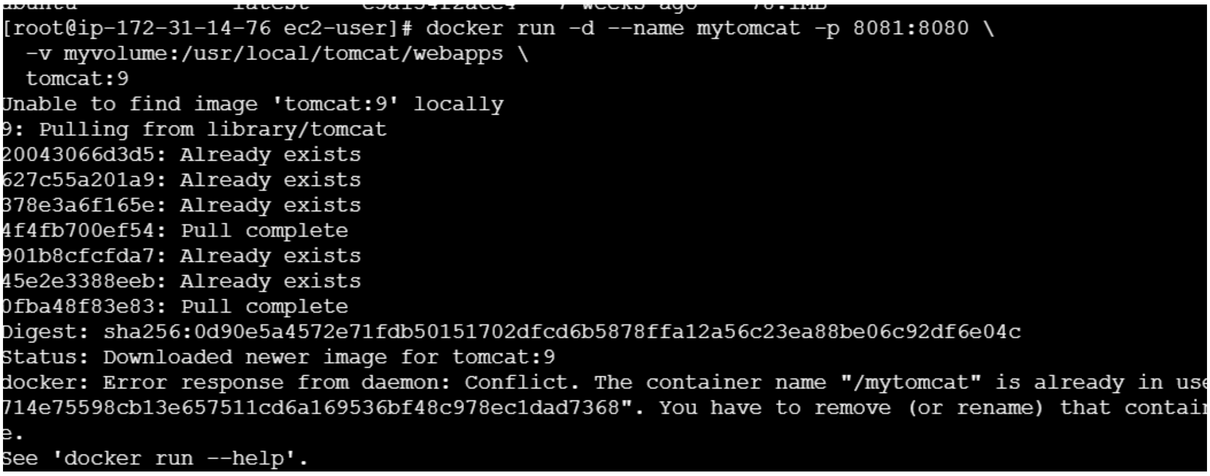
This is the home page for a sample application used to illustrate the source directory organization of a web a

To prove that they work, you can execute either of the following links:

- To a JSP page.
- To a servlet.

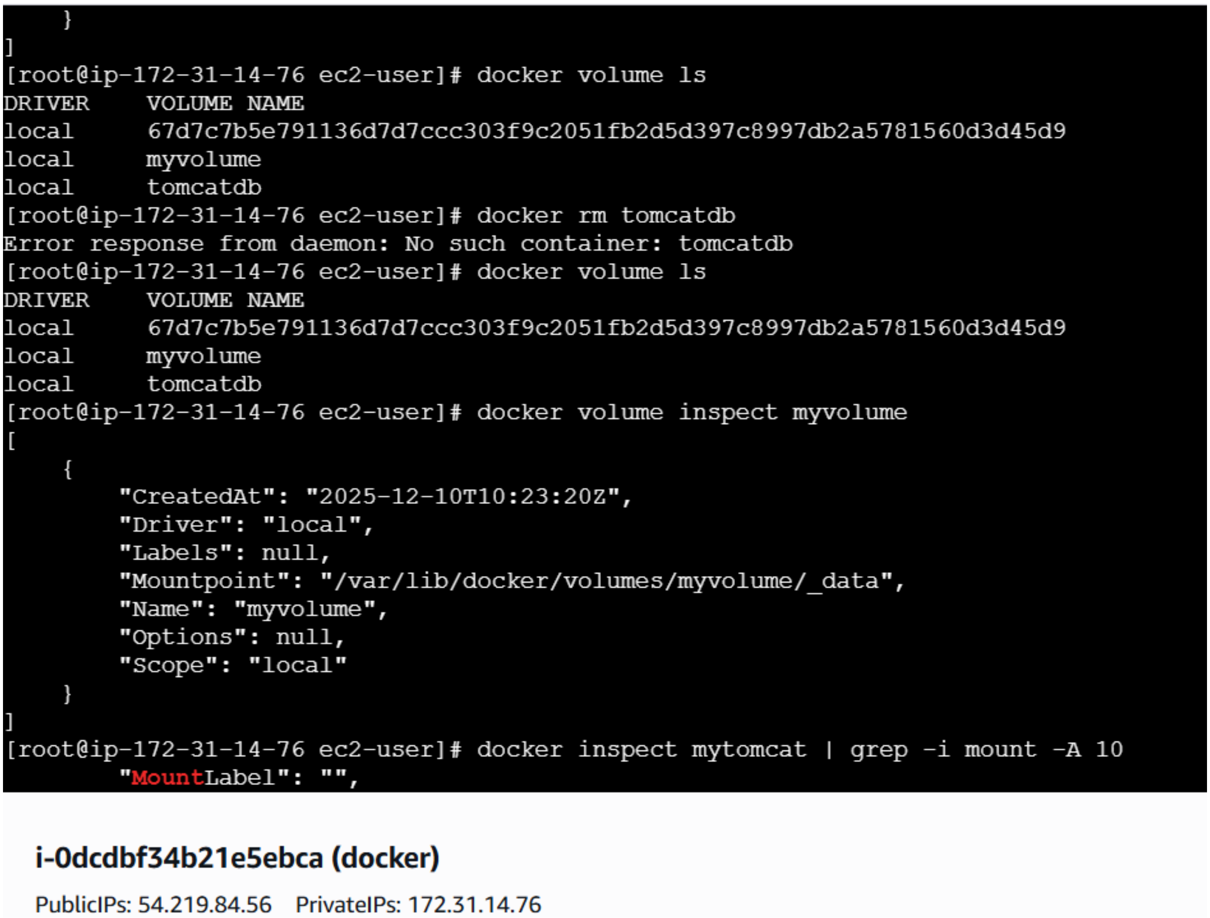
2 . Create volume and deploy tomcat container on port 8080

- volume can be added to contiane r onlyb while creation
- use one line command
- docker run -d --name mytomcat -p 8081:8080 \
- -v myvolume:/usr/local/tomcat/webapps \
- tomcat:9



```
[root@ip-172-31-14-76 ec2-user]# docker run -d --name mytomcat -p 8081:8080 \
  -v myvolume:/usr/local/tomcat/webapps \
  tomcat:9
Unable to find image 'tomcat:9' locally
9: Pulling from library/tomcat
20043066d3d5: Already exists
627c55a201a9: Already exists
378e3a6f165e: Already exists
4f4fb700ef54: Pull complete
901b8cfcfda7: Already exists
45e2e3388eeb: Already exists
0fba48f83e83: Pull complete
Digest: sha256:0d90e5a4572e71fdb50151702dfcd6b5878ffa12a56c23ea88be06c92df6e04c
Status: Downloaded newer image for tomcat:9
docker: Error response from daemon: Conflict. The container name "/mytomcat" is already in use
714e75598cb13e657511cd6a169536bf48c978ec1dad7368". You have to remove (or rename) that contai
e.
See 'docker run --help'.
```

- Docker inspect myvolume



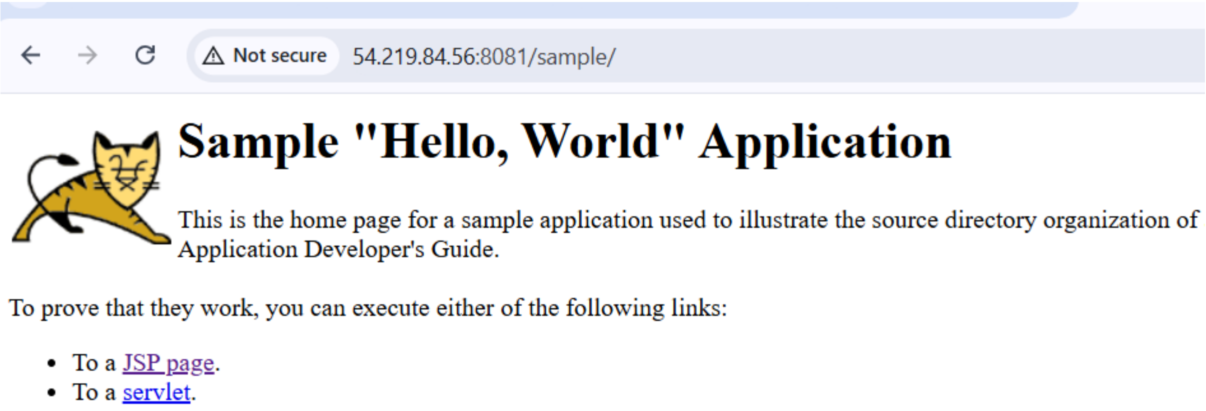
```
    }
]
[root@ip-172-31-14-76 ec2-user]# docker volume ls
DRIVER    VOLUME NAME
local     67d7c7b5e791136d7d7ccc303f9c2051fb2d5d397c8997db2a5781560d3d45d9
local     myvolume
local     tomcatdb
[root@ip-172-31-14-76 ec2-user]# docker rm tomcatdb
Error response from daemon: No such container: tomcatdb
[root@ip-172-31-14-76 ec2-user]# docker volume ls
DRIVER    VOLUME NAME
local     67d7c7b5e791136d7d7ccc303f9c2051fb2d5d397c8997db2a5781560d3d45d9
local     myvolume
local     tomcatdb
[root@ip-172-31-14-76 ec2-user]# docker volume inspect myvolume
[
    {
        "CreatedAt": "2025-12-10T10:23:20Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/myvolume/_data",
        "Name": "myvolume",
        "Options": null,
        "Scope": "local"
    }
]
[root@ip-172-31-14-76 ec2-user]# docker inspect mytomcat | grep -i mount -A 10
        "MountLabel": "",
```

i-0dcdbf34b21e5ebca (docker)

PublicIPs: 54.219.84.56   PrivateIPs: 172.31.14.76

- **Check if your container is using the volume**
- docker inspect mytomcat | grep -i mount -A 10
- Inside the output, look for:
- "Mounts": [

- {
- "Type": "volume",
- "Name": "myvolume",
- "Destination": "/usr/local/tomcat/webapps"
- }
-]
- Now , change ip
- Docker run -d –name mytomcat -p 8081:8080 tomcat:9
- Wget sample war file link
- Now copy sample.warv   fild to
- Mytomcat:/usr/local/tomcat/webapps
- And now ip abnd open browser
- And add ipadn with port number and /sample
- Pag ewill be deployed.



Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory organization of Application Developer's Guide.

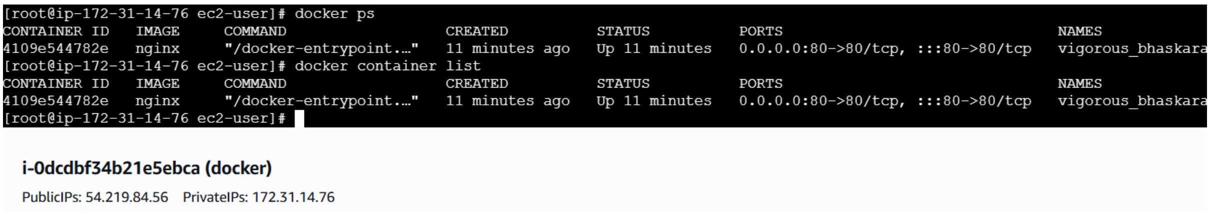
To prove that they work, you can execute either of the following links:

- To a JSP page.
- To a servlet.

3 . Limit the nginx container to 500 MB.

- Create container
- Or pull a image from docker
- If we pull image we need create a container separately , it is a step process
- Docker pull (service name)
- Docker run -d -p 80:80 (service name)

- **If you want *only* auto-generated name**
- (No memory limit)
- `docker run -d -p 80:80 nginx`
- Docker will create and run nginx, giving it a random name like:
- `happy_raven`
- `elegant_tiger`
-
- Or
- We can directly
- Create a container in single step bycommand
- Docker ru -d –name (any name) -p 80:80 (service name)



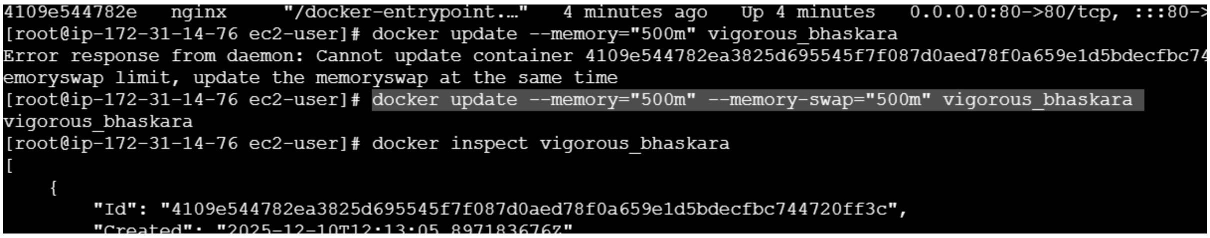
```
[root@ip-172-31-14-76 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                 NAMES
4109e544782e   nginx     "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp     vigorous_bhaskara
[root@ip-172-31-14-76 ec2-user]# docker container list
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                 NAMES
4109e544782e   nginx     "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp     vigorous_bhaskara
[root@ip-172-31-14-76 ec2-user]#
```

i-0dcdbf34b21e5ebca (docker)

PublicIPs: 54.219.84.56   PrivateIPs: 172.31.14.76

To limit the container to 500mb

- We need add memory and swap memory
- docker update --memory="500m" --memory-swap="500m" vigorous_bhaskara



```
4109e544782e   nginx     "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp, :::80->
[root@ip-172-31-14-76 ec2-user]# docker update --memory="500m" vigorous_bhaskara
Error response from daemon: Cannot update container 4109e544782ea3825d695545f7f087d0aed78f0a659e1d5bdecfbc74
emoryswap limit, update the memoryswap at the same time
[root@ip-172-31-14-76 ec2-user]# docker update --memory="500m" --memory-swap="500m" vigorous_bhaskara
vigorous_bhaskara
[root@ip-172-31-14-76 ec2-user]# docker inspect vigorous_bhaskara
[
    {
        "Id": "4109e544782ea3825d695545f7f087d0aed78f0a659e1d5bdecfbc744720ff3c",
        "Created": "2025-12-10T12:13:05.897183676Z"
```

4 . Create a sample docker file using below instructions.

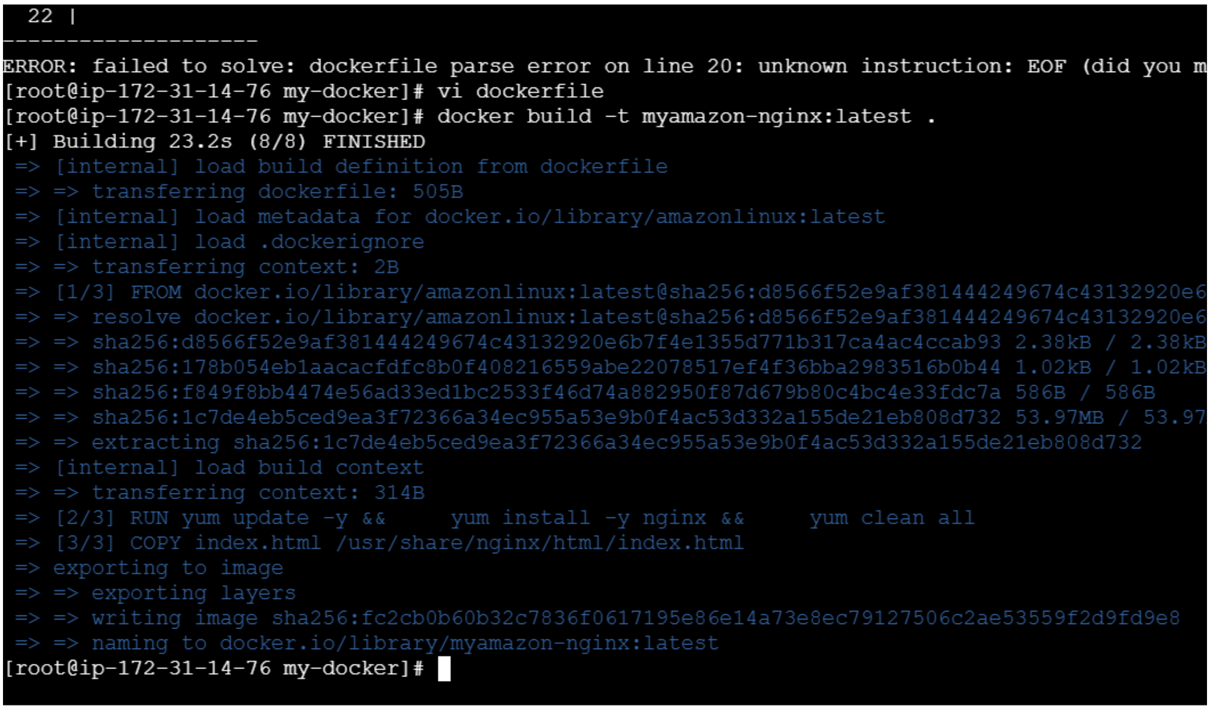
a. Base module as amazonlinux:latest
b. Maintainer you name
c. Install nginx
d. COPY one index.html file to image
e. Expose on port 80
f. Command to start the nginx container

Base module as amazonlinux:latest

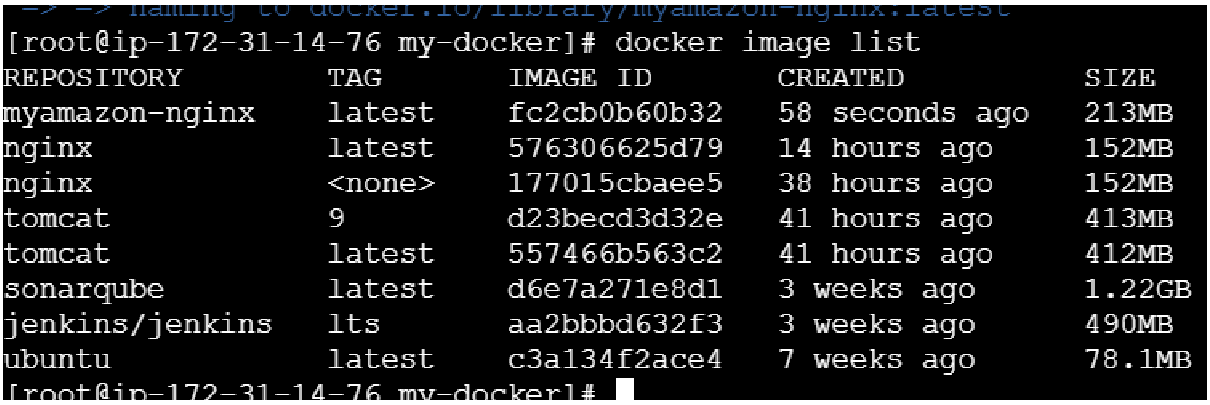
- Launch an instance
- Create a folder my-docker
- Add 2 files
- Index.html and dockerfile
- Index.html >>>> add script
-
- <!doctype html>
- <html>
- <head><meta charset="utf-8"><title>My Amazon Linux NGINX</title></head>
- <body>
- <h1>Hello from my custom Amazon Linux + NGINX image!</h1>
- </body>
- </html>

- Now, docker file
-
- # Base image
- FROM amazonlinux:latest
-
- # Maintainer
- MAINTAINER kamal
-
- # Install nginx
- RUN yum update -y && \
- yum install -y nginx && \

- yum clean all
-
- # Copy custom index.html into nginx's default web directory
- COPY index.html /usr/share/nginx/html/index.html
-
- # Expose port 80 (where nginx listens)
- EXPOSE 80
-
- # Start nginx in foreground so container stays alive
- CMD ["nginx", "-g", "daemon off;"]
-
- Now ---
- Run
- Build docker image
- Command - docker build -t myamazon-nginx:latest .
- Image pulled

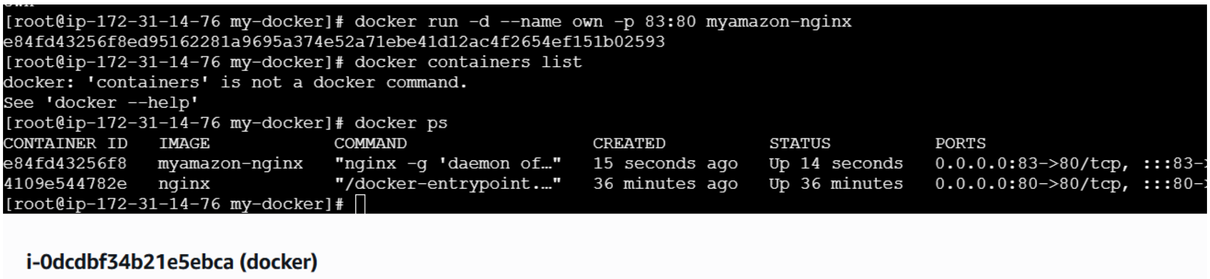


```
  22 |
--------------------
ERROR: failed to solve: dockerfile parse error on line 20: unknown instruction: EOF (did you m
[root@ip-172-31-14-76 my-docker]# vi dockerfile
[root@ip-172-31-14-76 my-docker]# docker build -t myamazon-nginx:latest .
[+] Building 23.2s (8/8) FINISHED
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 505B
 => [internal] load metadata for docker.io/library/amazonlinux:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/3] FROM docker.io/library/amazonlinux:latest@sha256:d8566f52e9af381444249674c43132920e6
 => => resolve docker.io/library/amazonlinux:latest@sha256:d8566f52e9af381444249674c43132920e6
 => => sha256:d8566f52e9af381444249674c43132920e6b7f4e1355d771b317ca4ac4ccab93 2.38kB / 2.38kB
 => => sha256:178b054eb1aacacfdfc8b0f408216559abe22078517ef4f36bba2983516b0b44 1.02kB / 1.02kB
 => => sha256:f849f8bb4474e56ad33ed1bc2533f46d74a882950f87d679b80c4bc4e33fdc7a 586B / 586B
 => => sha256:1c7de4eb5ced9ea3f72366a34ec955a53e9b0f4ac53d332a155de21eb808d732 53.97MB / 53.97
 => => extracting sha256:1c7de4eb5ced9ea3f72366a34ec955a53e9b0f4ac53d332a155de21eb808d732
 => [internal] load build context
 => => transferring context: 314B
 => [2/3] RUN yum update -y &&     yum install -y nginx &&     yum clean all
 => [3/3] COPY index.html /usr/share/nginx/html/index.html
 => exporting to image
 => => exporting layers
 => => writing image sha256:fc2cb0b60b32c7836f0617195e86e14a73e8ec79127506c2ae53559f2d9fd9e8
 => => naming to docker.io/library/myamazon-nginx:latest
[root@ip-172-31-14-76 my-docker]# 
```



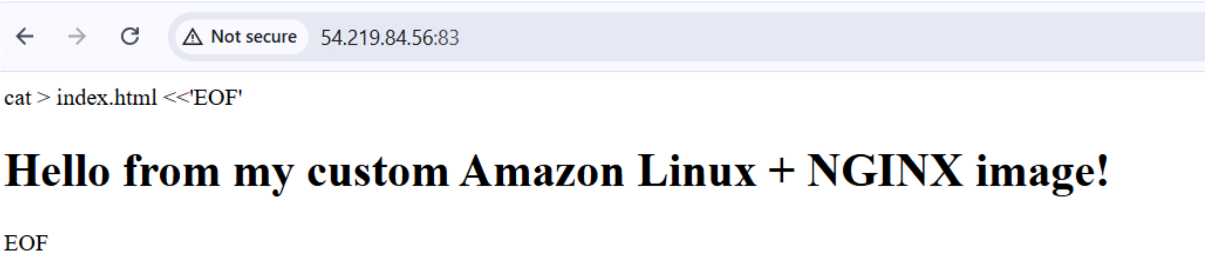
```
=> => naming to docker.io/library/myamazon-nginx:latest
[root@ip-172-31-14-76 my-docker]# docker image list
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
myamazon-nginx      latest    fc2cb0b60b32   58 seconds ago   213MB
nginx               latest    576306625d79   14 hours ago     152MB
nginx               <none>    177015cbaee5   38 hours ago     152MB
tomcat              9         d23becd3d32e   41 hours ago     413MB
tomcat              latest    557466b563c2   41 hours ago     412MB
sonarqube           latest    d6e7a271e8d1   3 weeks ago      1.22GB
jenkins/jenkins     lts       aa2bbbd632f3   3 weeks ago      490MB
ubuntu              latest    c3a134f2ace4   7 weeks ago      78.1MB
[root@ip-172-31-14-76 my-docker]# 
```

- Create a container
- Docker run -d –name own -p 83:80 myamazon-nginx



```
[root@ip-172-31-14-76 my-docker]# docker run -d --name own -p 83:80 myamazon-nginx
e84fd43256f8ed95162281a9695a374e52a71ebe41d12ac4f2654ef151b02593
[root@ip-172-31-14-76 my-docker]# docker containers list
docker: 'containers' is not a docker command.
See 'docker --help'
[root@ip-172-31-14-76 my-docker]# docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS          PORTS
e84fd43256f8   myamazon-nginx   "nginx -g 'daemon of…"   15 seconds ago   Up 14 seconds   0.0.0.0:83->80/tcp, :::83-
4109e544782e   nginx            "/docker-entrypoint.…"   36 minutes ago   Up 36 minutes   0.0.0.0:80->80/tcp, :::80-
[root@ip-172-31-14-76 my-docker]# 
```

i-0dcdbf34b21e5ebca (docker)



cat > index.html <<'EOF'

Hello from my custom Amazon Linux + NGINX image!

EOF

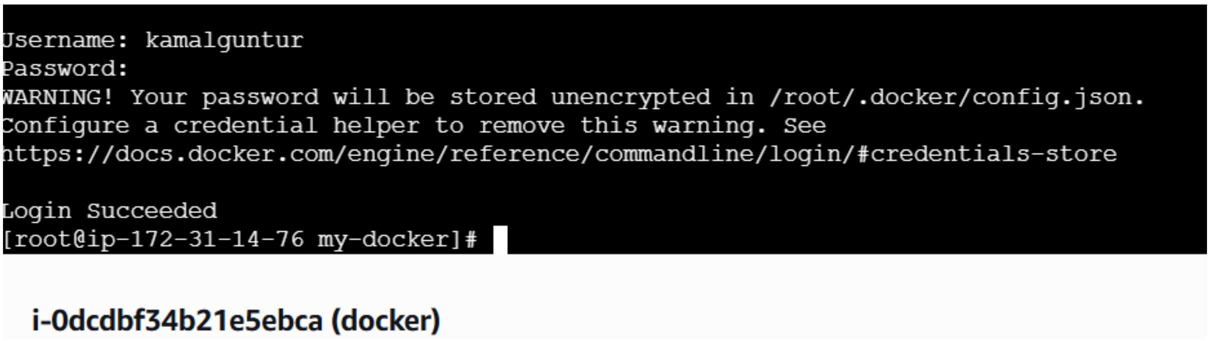
-

5 . Push image to Docker hub

Login to docker

Create a accoubnt in dockerhub gui

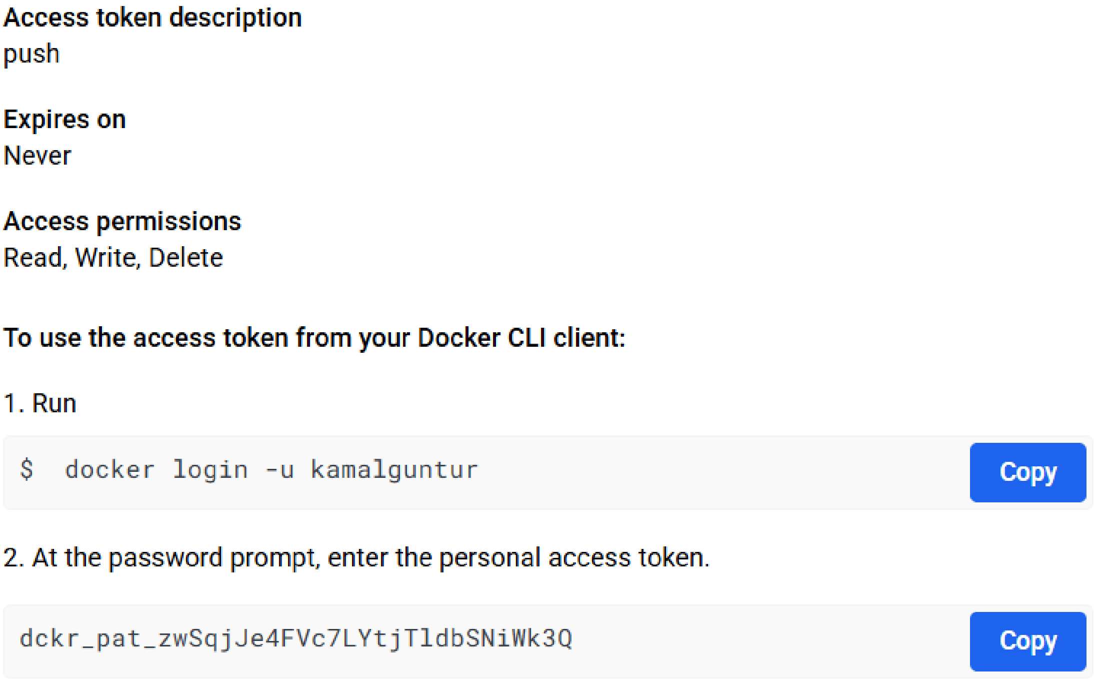
- In instance
- Give command
- Docker login
- Give username andn password



```
Username: kamalguntur
Password:
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-172-31-14-76 my-docker]# 
```

i-0dcdbf34b21e5ebca (docker)

- Now create a repo in gui
- Generate a token
- Add token in instance



Access token description
push

Expires on
Never

Access permissions
Read, Write, Delete

To use the access token from your Docker CLI client:

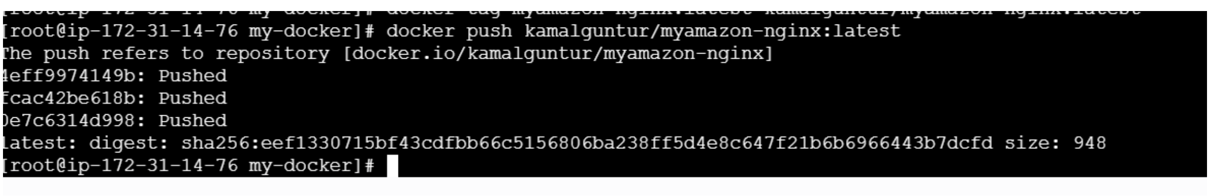
1. Run

```
$  docker login -u kamalguntur
```

2. At the password prompt, enter the personal access token.

```
dckr_pat_zwSqjJe4FVc7LYtjTldbSNiWk3Q
```

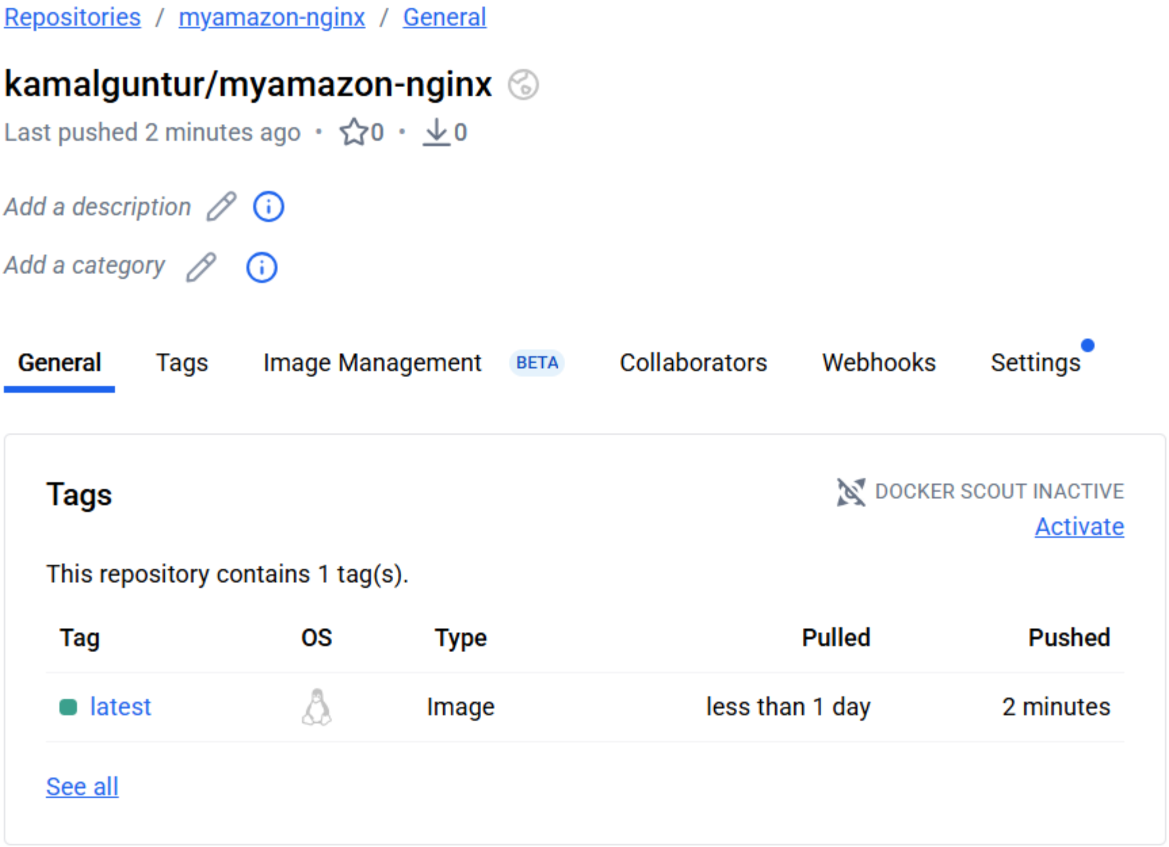
-
-



```
[root@ip-172-31-14-76 my-docker]# docker tag myamazon-nginx:latest kamalguntur/myamazon-nginx:latest
[root@ip-172-31-14-76 my-docker]# docker push kamalguntur/myamazon-nginx:latest
The push refers to repository [docker.io/kamalguntur/myamazon-nginx]
4eff9974149b: Pushed
fcac42be618b: Pushed
0e7c6314d998: Pushed
latest: digest: sha256:eef1330715bf43cdfbb66c5156806ba238ff5d4e8c647f21b6b6966443b7dcfd size: 948
[root@ip-172-31-14-76 my-docker]# 
```

-
- Now,
- Tag the image
- docker tag myamazon-nginx:latest kamalguntur/myamazon-nginx:latest
- and push

- docker push kamalguntur/myamazon-nginx:latest



kamalguntur/myamazon-nginx

Last pushed 2 minutes ago • ☆0 • ↓0

Tags

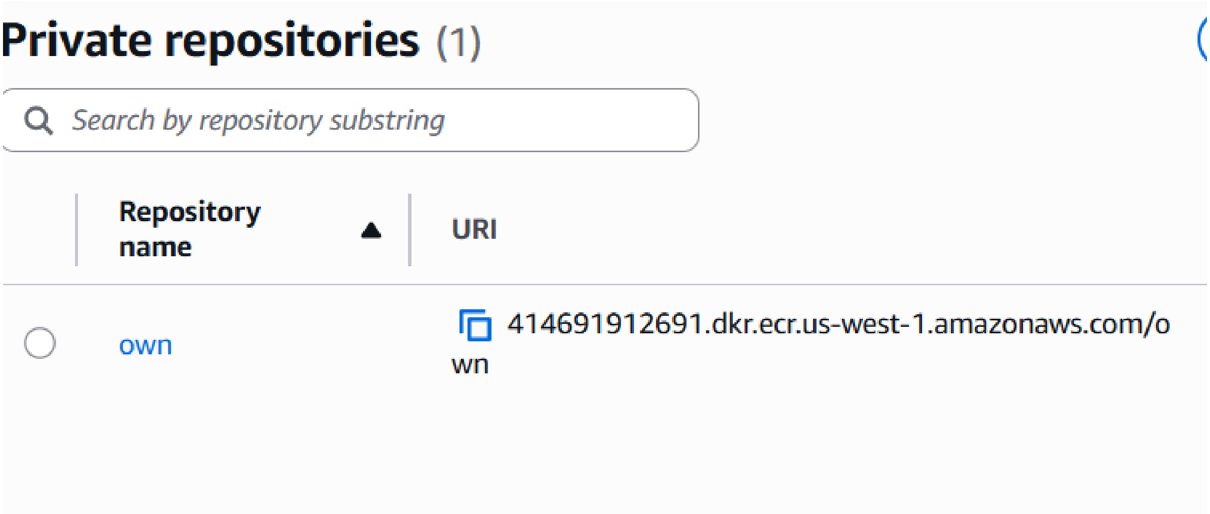
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
🟩 latest	🐧	Image	less than 1 day	2 minutes

-

1. 6 . push image to AWS ECR

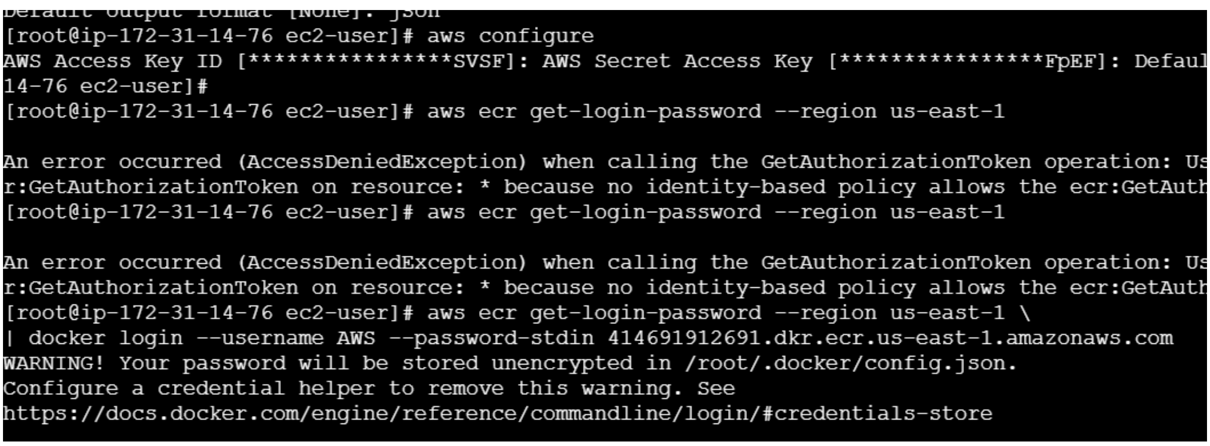
   - **Login to AWS ECR**
   - To push an image, Docker must authenticate with ECR. Replace <REGION> and <ACCOUNT_ID> accordingly.
   - **Command**
   - aws ecr get-login-password --region <REGION> \
   - | docker login --username AWS --password-stdin <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com
   - aws ecr get-login-password --region us-west-1 \
   - | docker login --username AWS --password-stdin 414691912691.dkr.ecr.us-west-1.amazonaws.com
   - Successful login displays:
   - Login Succeeded
   - Create a repo inaws ecr



Private repositories (1)

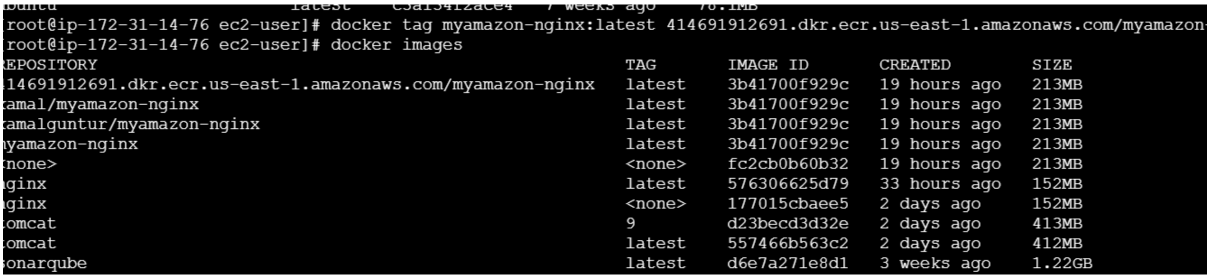
Search by repository substring

	Repository name ▲	URI
○	own	414691912691.dkr.ecr.us-west-1.amazonaws.com/own

- 

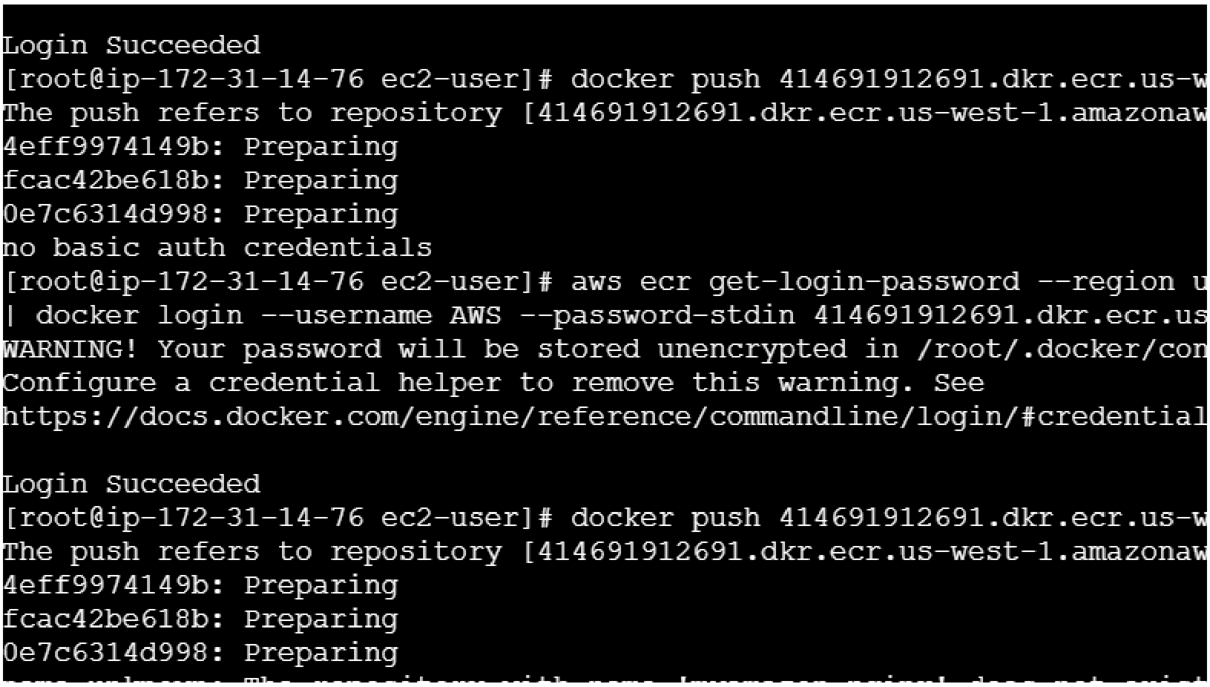
```
Default output format [None]: json
[root@ip-172-31-14-76 ec2-user]# aws configure
AWS Access Key ID [****************SVSF]: AWS Secret Access Key [****************FpEF]: Defaul
14-76 ec2-user]#
[root@ip-172-31-14-76 ec2-user]# aws ecr get-login-password --region us-east-1

An error occurred (AccessDeniedException) when calling the GetAuthorizationToken operation: Us
r:GetAuthorizationToken on resource: * because no identity-based policy allows the ecr:GetAuth
[root@ip-172-31-14-76 ec2-user]# aws ecr get-login-password --region us-east-1

An error occurred (AccessDeniedException) when calling the GetAuthorizationToken operation: Us
r:GetAuthorizationToken on resource: * because no identity-based policy allows the ecr:GetAuth
[root@ip-172-31-14-76 ec2-user]# aws ecr get-login-password --region us-east-1 \
| docker login --username AWS --password-stdin 414691912691.dkr.ecr.us-east-1.amazonaws.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
```

- Docker images must be tagged with the ECR registry URL before pushing.



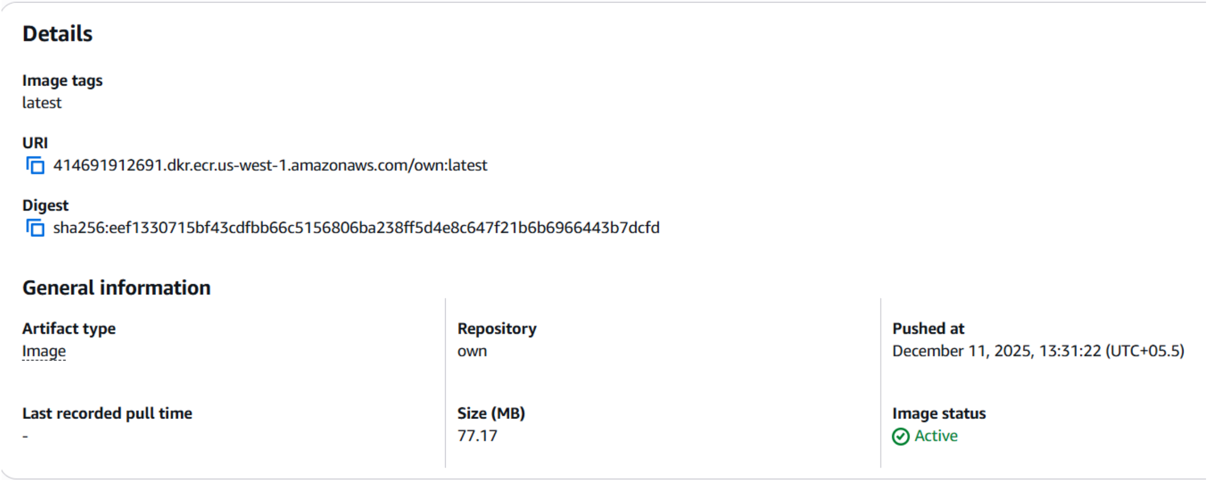
```
ubuntu                   latest   c3a1341zace4   7 weeks ago   78.1MB
root@ip-172-31-14-76 ec2-user]# docker tag myamazon-nginx:latest 414691912691.dkr.ecr.us-east-1.amazonaws.com/myamazon-
root@ip-172-31-14-76 ec2-user]# docker images
EPOSITORY                                                  TAG       IMAGE ID       CREATED         SIZE
14691912691.dkr.ecr.us-east-1.amazonaws.com/myamazon-nginx   latest    3b41700f929c   19 hours ago    213MB
amal/myamazon-nginx                                        latest    3b41700f929c   19 hours ago    213MB
amalguntur/myamazon-nginx                                  latest    3b41700f929c   19 hours ago    213MB
yamazon-nginx                                              latest    3b41700f929c   19 hours ago    213MB
none>                                                      <none>    fc2cb0b60b32   19 hours ago    213MB
ginx                                                       latest    576306625d79   33 hours ago    152MB
ginx                                                       <none>    177015cbaee5   2 days ago      152MB
omcat                                                      9         d23becd3d32e   2 days ago      413MB
omcat                                                      latest    557466b563c2   2 days ago      412MB
onarqube                                                   latest    d6e7a271e8d1   3 weeks ago     1.22GB
```

- docker tag <LOCAL_IMAGE_NAME>:<TAG> <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/<REPO_NAME>:<TAG>



```
Login Succeeded
[root@ip-172-31-14-76 ec2-user]# docker push 414691912691.dkr.ecr.us-w
The push refers to repository [414691912691.dkr.ecr.us-west-1.amazonaw
4eff9974149b: Preparing
fcac42be618b: Preparing
0e7c6314d998: Preparing
no basic auth credentials
[root@ip-172-31-14-76 ec2-user]# aws ecr get-login-password --region u
| docker login --username AWS --password-stdin 414691912691.dkr.ecr.us
WARNING! Your password will be stored unencrypted in /root/.docker/con
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential

Login Succeeded
[root@ip-172-31-14-76 ec2-user]# docker push 414691912691.dkr.ecr.us-w
The push refers to repository [414691912691.dkr.ecr.us-west-1.amazonaw
4eff9974149b: Preparing
fcac42be618b: Preparing
0e7c6314d998: Preparing
```

- docker tag myamazon-nginx:latest 414691912691.dkr.ecr.us-west-1.amazonaws.com/myamazon-nginx:latest
- This command does not copy the image; it only assigns a new tag pointing to ECR.
- ---
- Once the image is tagged, it can be pushed to the repository.
- **Command**
- docker push <ACCOUNT_ID>.dkr.ecr.<REGION>.amazonaws.com/<REPO_NAME>:latest
- **Example**
- docker push 414691912691.dkr.ecr.us-west-1.amazonaws.com/myamazon-nginx:latest



Details

Image tags
latest

URI
414691912691.dkr.ecr.us-west-1.amazonaws.com/own:latest

Digest
sha256:eef1330715bf43cdfbb66c5156806ba238ff5d4e8c647f21b6b6966443b7dcfd

General information

Artifact type
Image

Repository
own

Pushed at
December 11, 2025, 13:31:22 (UTC+05.5)

Last recorded pull time
-

Size (MB)
77.17

Image status
✓ Active