AWS + Docker Class Notes

1. What is a Docker file?

A Docker file is a text file that contains a series of instructions used to build a Docker image. Think of it as a recipe for creating a container environment.

To write a Docker file for your application, you should know:

- The technology your app uses (e.g., Node.js, Java, PHP)

- The commands needed to build and run your app (e.g., npm init, npm install, npm start for Node.js)

2. Setting Up in VS Code

- Install the Docker extension in Visual Studio Code.

- Create a file named Dockerfile (no extension).

- Save it in the root directory of your project.

3. Sample Docker file for a Node.js App

```
FROM node:22

WORKDIR /app

COPY package.json ./

RUN npm install

COPY . ./

EXPOSE 5173

CMD ["npm", "start"]
```

Explanation of Each Line

- FROM node:22 – Uses Node.js version 22 as the base image.

- WORKDIR /app – Sets /app as the working directory.

- COPY package.json ./ – Copies your package.json.

- RUN npm install – Installs dependencies.

- COPY . ./ – Copies all project files.

- EXPOSE 5173 – Exposes port 5173.

- CMD ["npm", "start"] – Starts the app.

4. Deploying on AWS EC2 (Ubuntu)

1. Launch an Ubuntu EC2 instance from AWS Console.

2. SSH into EC2 and become root: sudo su

3. Clone GitHub repo: git clone https://github.com/sibasish934/testing.git

4. Navigate to the project directory: cd testing

5. (Optional) Rename Dockerfile properly if needed.

6. Update system packages and install Docker:

   apt update

   apt install docker.io -y

7. Start Docker service:

   systemctl start docker

   systemctl enable docker

5. Building and Running Docker Image

- Build the image:

  docker build -t backend:v1 .

- Run the container:

  docker run -it -d -p 3000:3000 backend:v1

- Check containers:

  docker container ls

6. Allow Access to Port 3000 in AWS

- Go to EC2 > Security Groups > Inbound Rules

- Add new rule:

  Type: Custom TCP

  Port Range: 3000

  Source: Anywhere (0.0.0.0/0)

7. Access App in Browser

Open browser and go to:

```
http://<your-ec2-public-ip>:3000/
```

8. Optimizing Docker Image Size

- Initial Size: 1.13 GB

- Optimization 1 (node:22-slim): ~224 MB

- Optimization 2 (node:22-alpine): ~170 MB

9. Multi-Stage Docker Builds

Multi-stage builds define multiple stages in one Dockerfile. Only the final output is kept.

Sample Dockerfile (backend:v4):

```
FROM node:22-slim AS build

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . ./

FROM node:22-alpine

WORKDIR /app

COPY --from=build /app .

EXPOSE 3000

CMD ["node", "index.js"]
```

Resulting image size: ~164 MB

10. Docker Cleanup

```
docker system prune
```

Removes unused containers, networks, images, and cache.

11. Environment Variables in Docker

Example:

ENV NODE_ENV="production"

Access via process.env.NODE_ENV in Node.js

Safer with build arguments:

ARG APP_ENV

ENV NODE_ENV=$APP_ENV

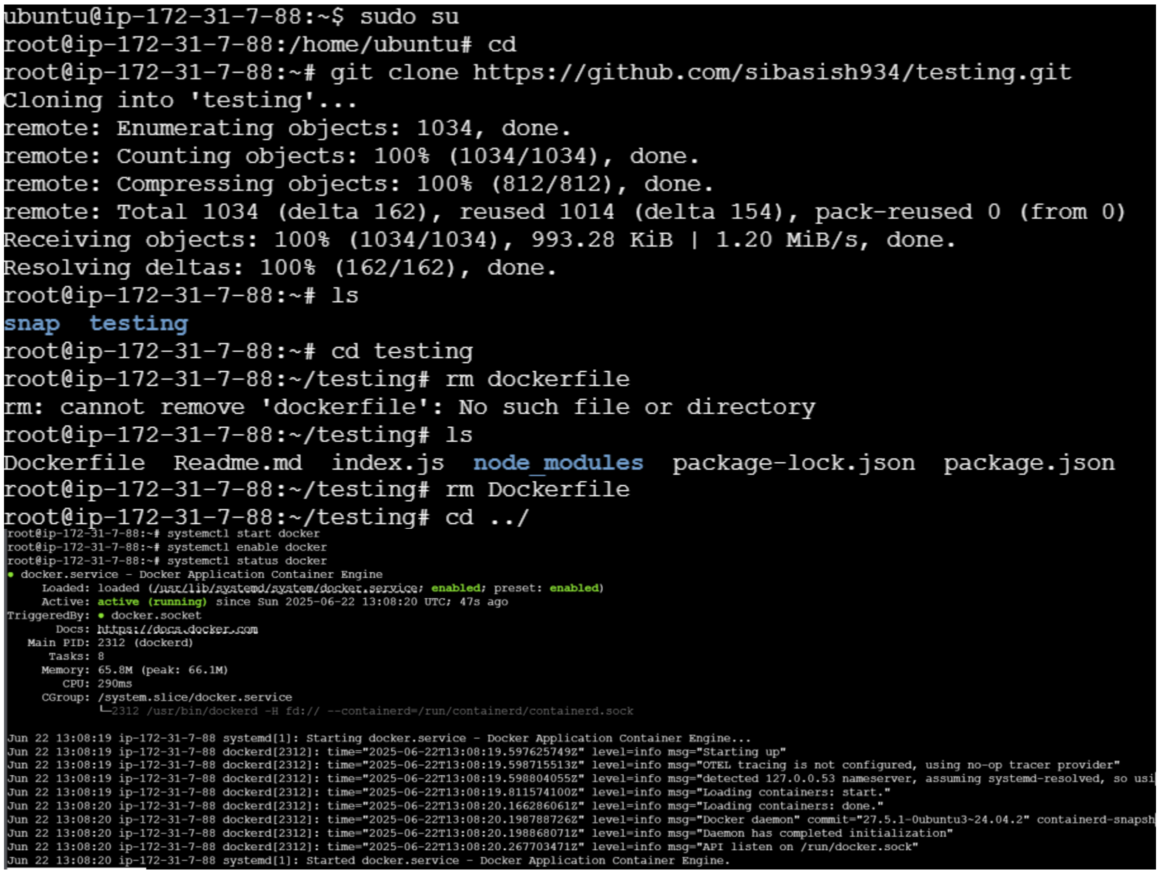
Build with:

docker build --build-arg APP_ENV=production -t backend:v5 .

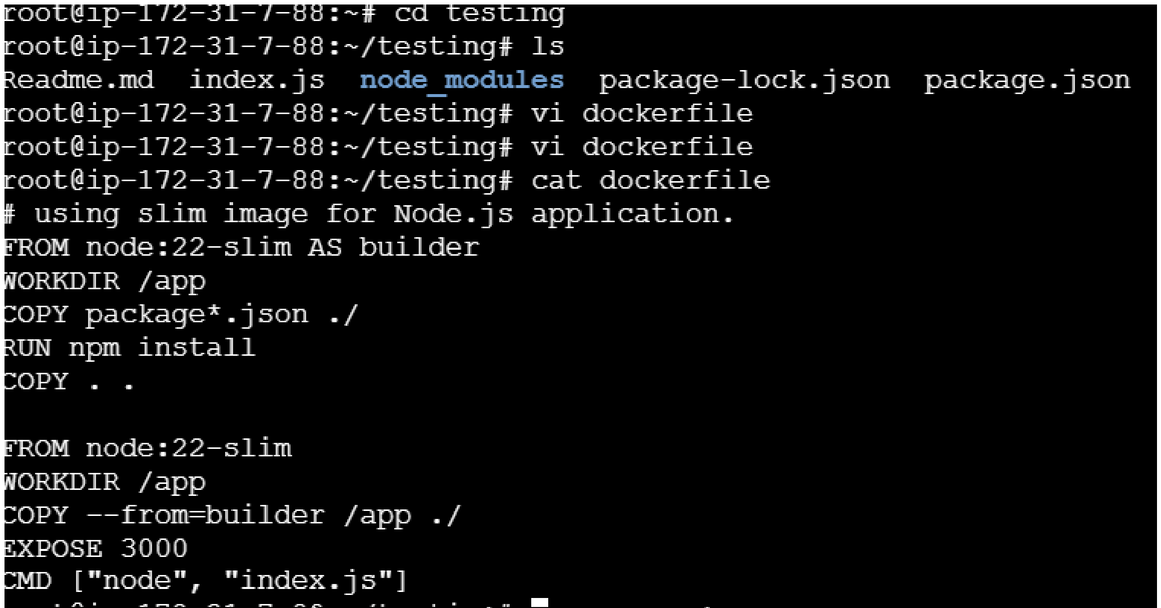
12. Docker Registry & AWS ECR

- AWS provides ECR for storing Docker images.

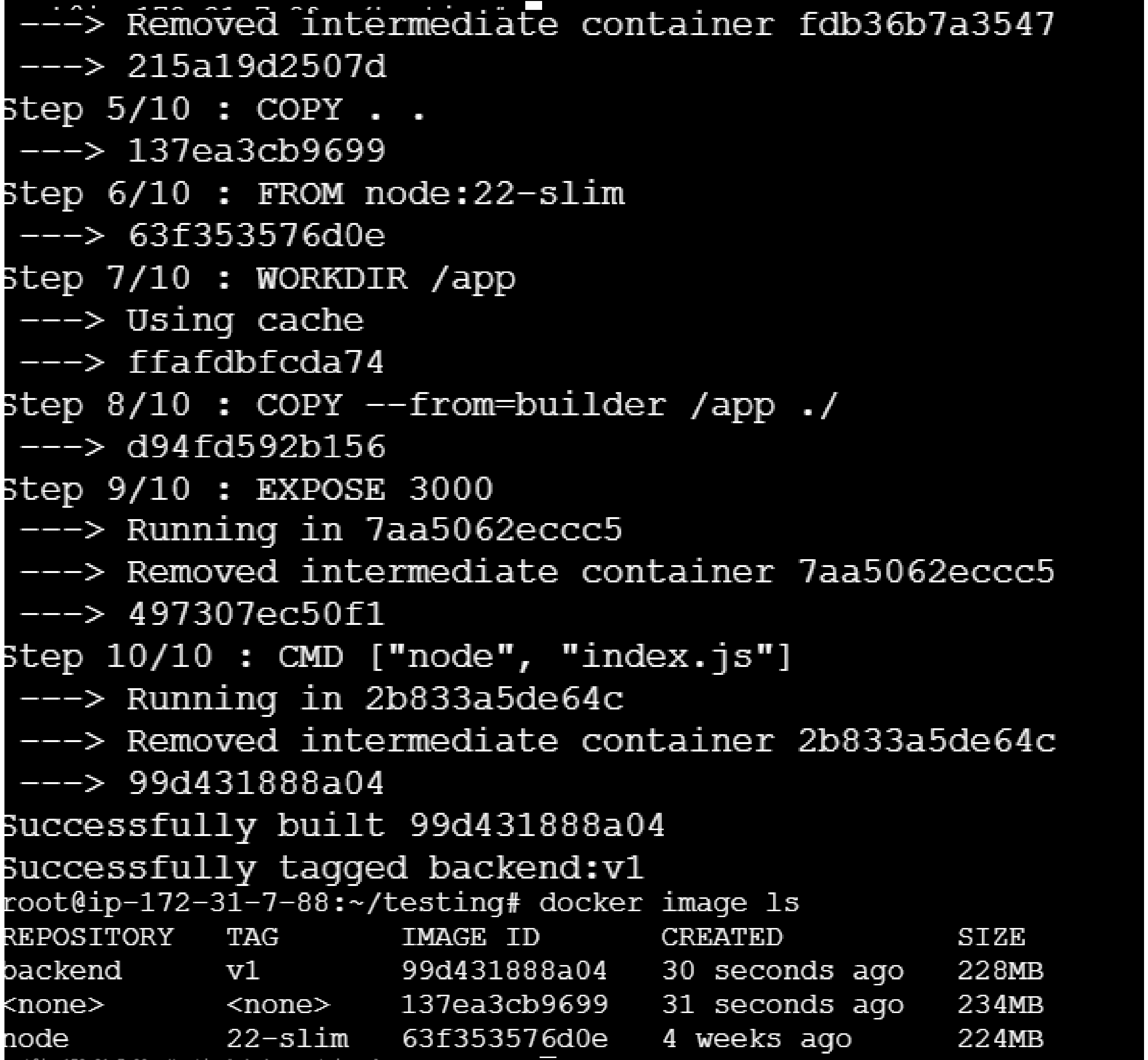
- Steps: Install AWS CLI, create IAM user, configure CLI, create repo, push image.



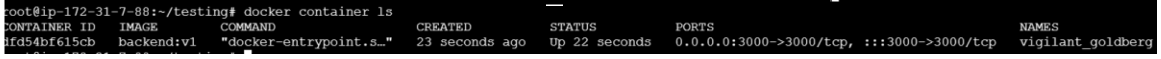
```
ubuntu@ip-172-31-7-88:~$ sudo su
root@ip-172-31-7-88:/home/ubuntu# cd
root@ip-172-31-7-88:~# git clone https://github.com/sibasish934/testing.git
Cloning into 'testing'...
remote: Enumerating objects: 1034, done.
remote: Counting objects: 100% (1034/1034), done.
remote: Compressing objects: 100% (812/812), done.
remote: Total 1034 (delta 162), reused 1014 (delta 154), pack-reused 0 (from 0)
Receiving objects: 100% (1034/1034), 993.28 KiB | 1.20 MiB/s, done.
Resolving deltas: 100% (162/162), done.
root@ip-172-31-7-88:~# ls
snap  testing
root@ip-172-31-7-88:~# cd testing
root@ip-172-31-7-88:~/testing# rm dockerfile
rm: cannot remove 'dockerfile': No such file or directory
root@ip-172-31-7-88:~/testing# ls
Dockerfile  Readme.md  index.js  node_modules  package-lock.json  package.json
root@ip-172-31-7-88:~/testing# rm Dockerfile
root@ip-172-31-7-88:~/testing# cd ../
root@ip-172-31-7-88:~# systemctl start docker
root@ip-172-31-7-88:~# systemctl enable docker
root@ip-172-31-7-88:~# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-06-22 13:08:20 UTC; 47s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2312 (dockerd)
      Tasks: 8
     Memory: 65.8M (peak: 66.1M)
        CPU: 290ms
     CGroup: /system.slice/docker.service
             └─2312 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jun 22 13:08:19 ip-172-31-7-88 systemd[1]: Starting docker.service - Docker Application Container Engine...
Jun 22 13:08:19 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:19.597625749Z" level=info msg="Starting up"
Jun 22 13:08:19 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:19.598715513Z" level=info msg="OTEL tracing is not configured, using no-op tracer provider"
Jun 22 13:08:19 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:19.598804055Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolved, so usi
Jun 22 13:08:19 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:19.811574100Z" level=info msg="Loading containers: start."
Jun 22 13:08:20 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:20.166286061Z" level=info msg="Loading containers: done."
Jun 22 13:08:20 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:20.198788726Z" level=info msg="Docker daemon" commit="27.5.1-0ubuntu3~24.04.2" containerd-snapsh
Jun 22 13:08:20 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:20.198868071Z" level=info msg="Daemon has completed initialization"
Jun 22 13:08:20 ip-172-31-7-88 dockerd[2312]: time="2025-06-22T13:08:20.267703471Z" level=info msg="API listen on /run/docker.sock"
Jun 22 13:08:20 ip-172-31-7-88 systemd[1]: Started docker.service - Docker Application Container Engine.
```



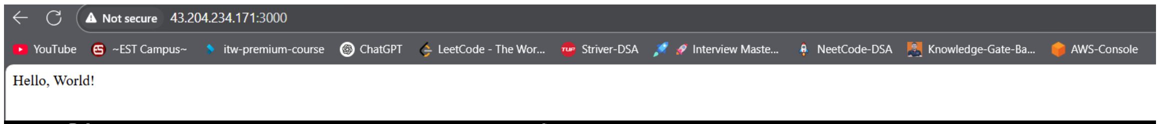
```
root@ip-172-31-7-88:~# cd testing
root@ip-172-31-7-88:~/testing# ls
Readme.md  index.js  node_modules  package-lock.json  package.json
root@ip-172-31-7-88:~/testing# vi dockerfile
root@ip-172-31-7-88:~/testing# vi dockerfile
root@ip-172-31-7-88:~/testing# cat dockerfile
# using slim image for Node.js application.
FROM node:22-slim AS builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .

FROM node:22-slim
WORKDIR /app
COPY --from=builder /app ./
EXPOSE 3000
CMD ["node", "index.js"]
```



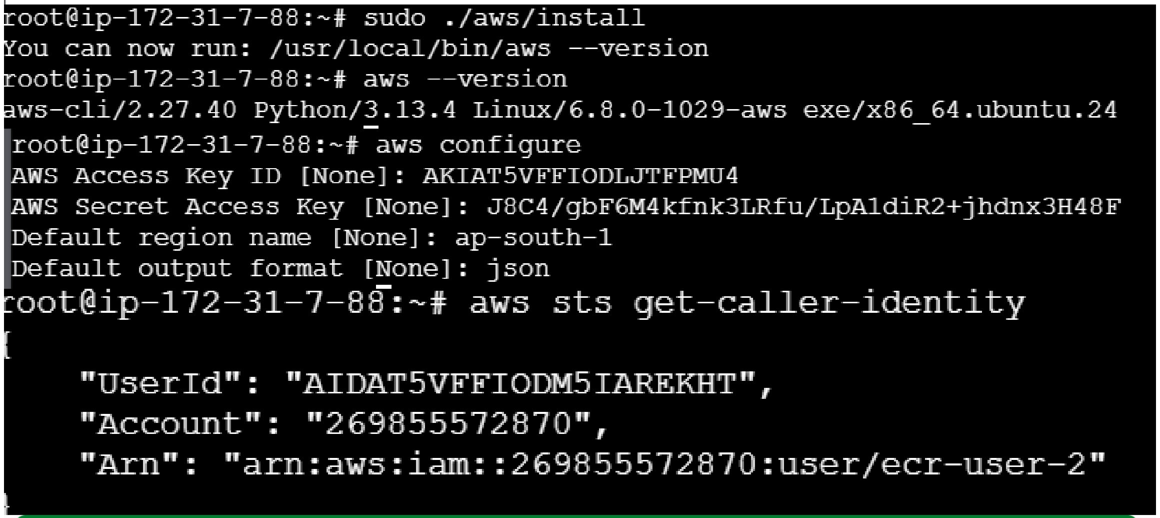
```
 ---> Removed intermediate container fdb36b7a3547
 ---> 215a19d2507d
Step 5/10 : COPY . .
 ---> 137ea3cb9699
Step 6/10 : FROM node:22-slim
 ---> 63f353576d0e
Step 7/10 : WORKDIR /app
 ---> Using cache
 ---> ffafdbfcda74
Step 8/10 : COPY --from=builder /app ./
 ---> d94fd592b156
Step 9/10 : EXPOSE 3000
 ---> Running in 7aa5062eccc5
 ---> Removed intermediate container 7aa5062eccc5
 ---> 497307ec50f1
Step 10/10 : CMD ["node", "index.js"]
 ---> Running in 2b833a5de64c
 ---> Removed intermediate container 2b833a5de64c
 ---> 99d431888a04
Successfully built 99d431888a04
Successfully tagged backend:v1
root@ip-172-31-7-88:~/testing# docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
backend      v1        99d431888a04   30 seconds ago   228MB
<none>       <none>    137ea3cb9699   31 seconds ago   234MB
node         22-slim   63f353576d0e   4 weeks ago      224MB
```



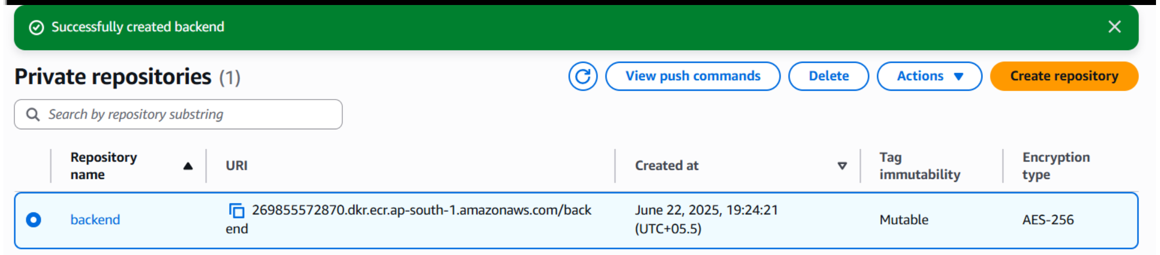
```
root@ip-172-31-7-88:~/testing# docker container ls
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS          PORTS                                       NAMES
dfd54bf615cb   backend:v1   "docker-entrypoint.s…"   23 seconds ago   Up 22 seconds   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   vigilant_goldberg
```



Hello, World!



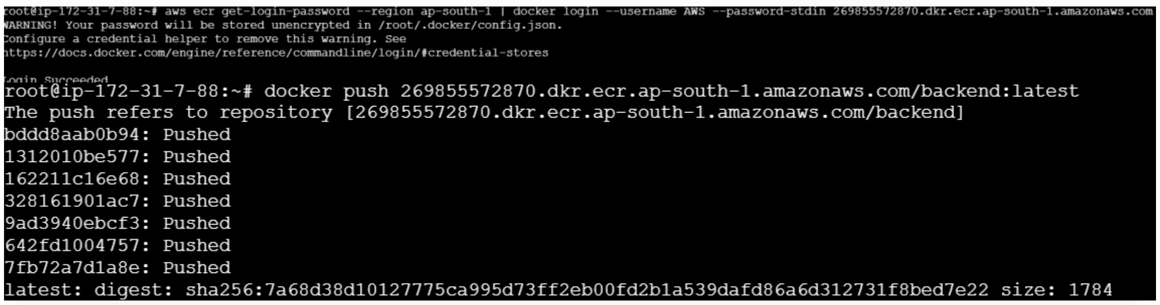
```
root@ip-172-31-7-88:~# sudo ./aws/install
You can now run: /usr/local/bin/aws --version
root@ip-172-31-7-88:~# aws --version
aws-cli/2.27.40 Python/3.13.4 Linux/6.8.0-1029-aws exe/x86_64.ubuntu.24
root@ip-172-31-7-88:~# aws configure
AWS Access Key ID [None]: AKIAT5VFFIODLJTFPMU4
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]: json
root@ip-172-31-7-88:~# aws sts get-caller-identity
{
    "UserId": "AIDAT5VFFIODM5IAREKHT",
    "Account": "269855572870",
    "Arn": "arn:aws:iam::269855572870:user/ecr-user-2"
```



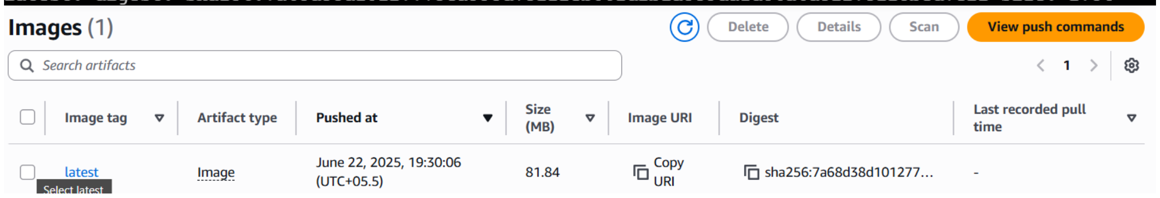
Successfully created backend

Private repositories (1)

Repository name	URI	Created at	Tag immutability	Encryption type
backend	269855572870.dkr.ecr.ap-south-1.amazonaws.com/backend	June 22, 2025, 19:24:21 (UTC+05.5)	Mutable	AES-256



```
root@ip-172-31-7-88:~# aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 269855572870.dkr.ecr.ap-south-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/#credential-stores

Login Succeeded
root@ip-172-31-7-88:~# docker push 269855572870.dkr.ecr.ap-south-1.amazonaws.com/backend:latest
The push refers to repository [269855572870.dkr.ecr.ap-south-1.amazonaws.com/backend]
bddd8aab0b94: Pushed
1312010be577: Pushed
162211c16e68: Pushed
328161901ac7: Pushed
9ad3940ebcf3: Pushed
642fd1004757: Pushed
7fb72a7d1a8e: Pushed
latest: digest: sha256:7a68d38d10127775ca995d73ff2eb00fd2b1a539dafd86a6d312731f8bed7e22 size: 1784
```



Images (1)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
latest	Image	June 22, 2025, 19:30:06 (UTC+05.5)	81.84	Copy URI	sha256:7a68d38d101277...	-