

## Lesson 10 Demo 02

### Creating a Container Registry Using AWS ECR

**Objective:** To create a container image repository using AWS ECR and push images into it

**Tools required:** AWS Management Console

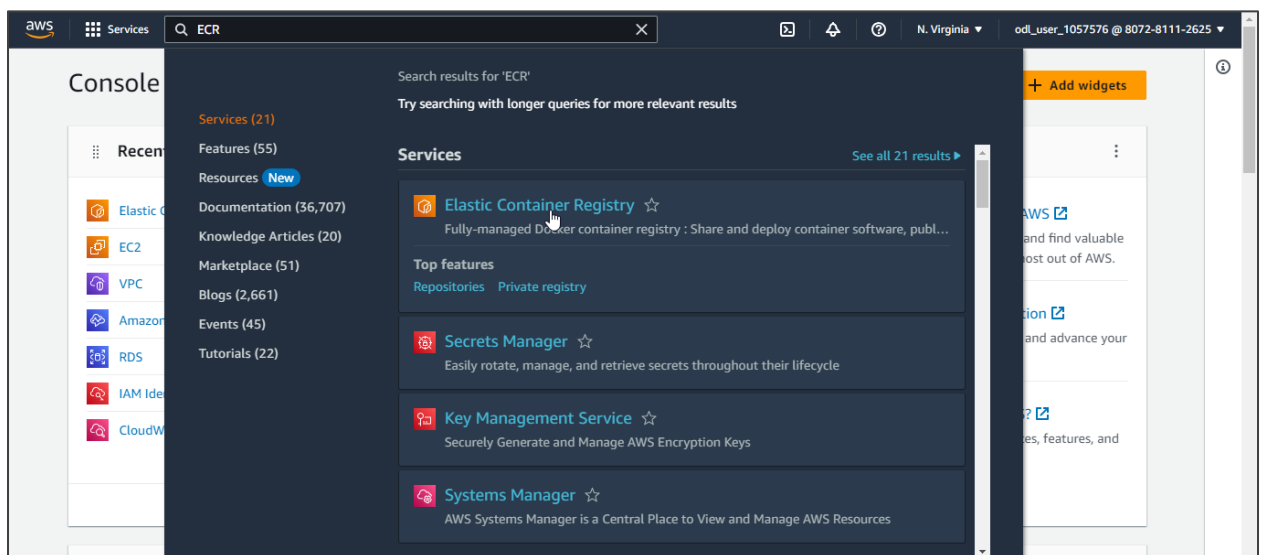
**Prerequisites:** None

Steps to be followed:

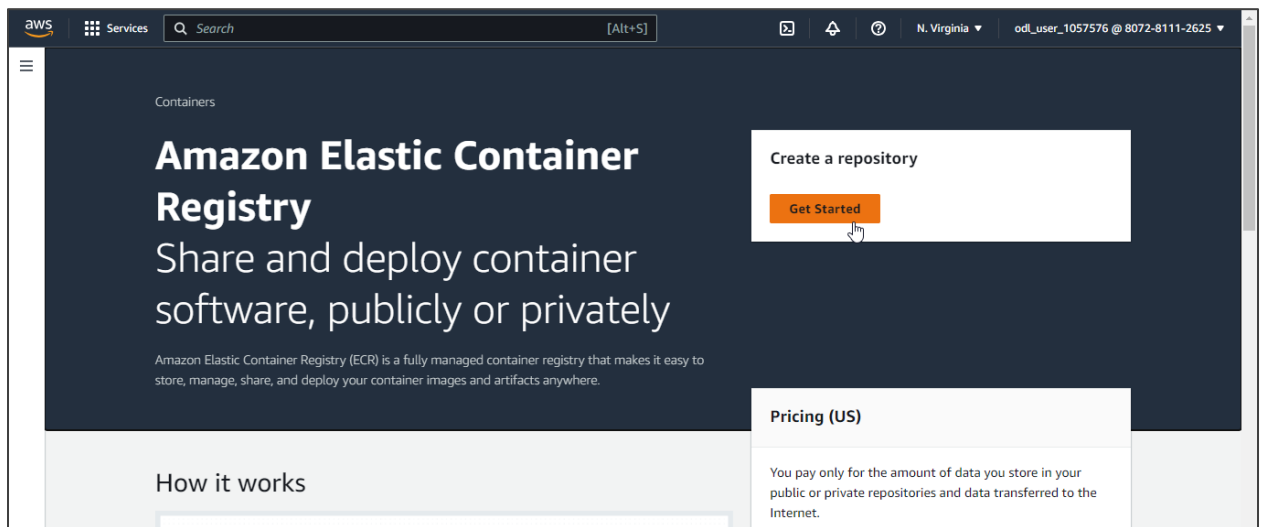
1. Create an ECR repository
2. Launch an EC2 instance
3. Install Docker on the EC2 instance
4. Create and push the Docker image to the repository

#### Step 1: Create an ECR repository

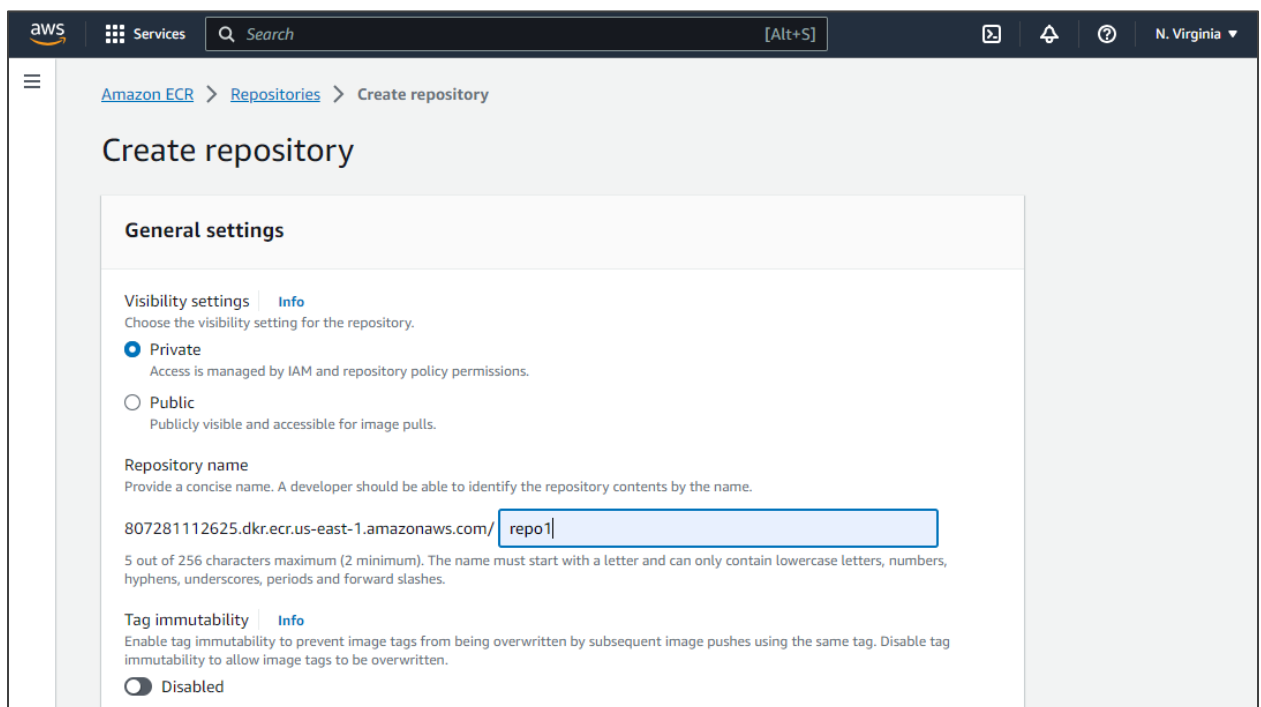
- 1.1 In the AWS Management Console, search for ECR and then click on **Elastic Container Registry**



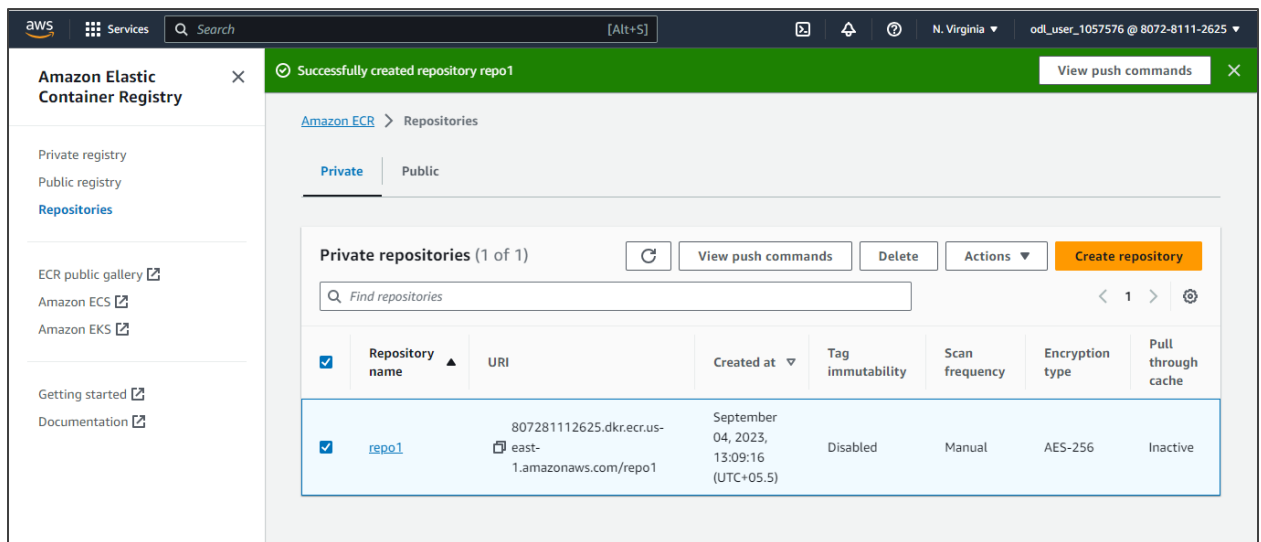
## 1.2 In the ECR console, click on **Get Started**



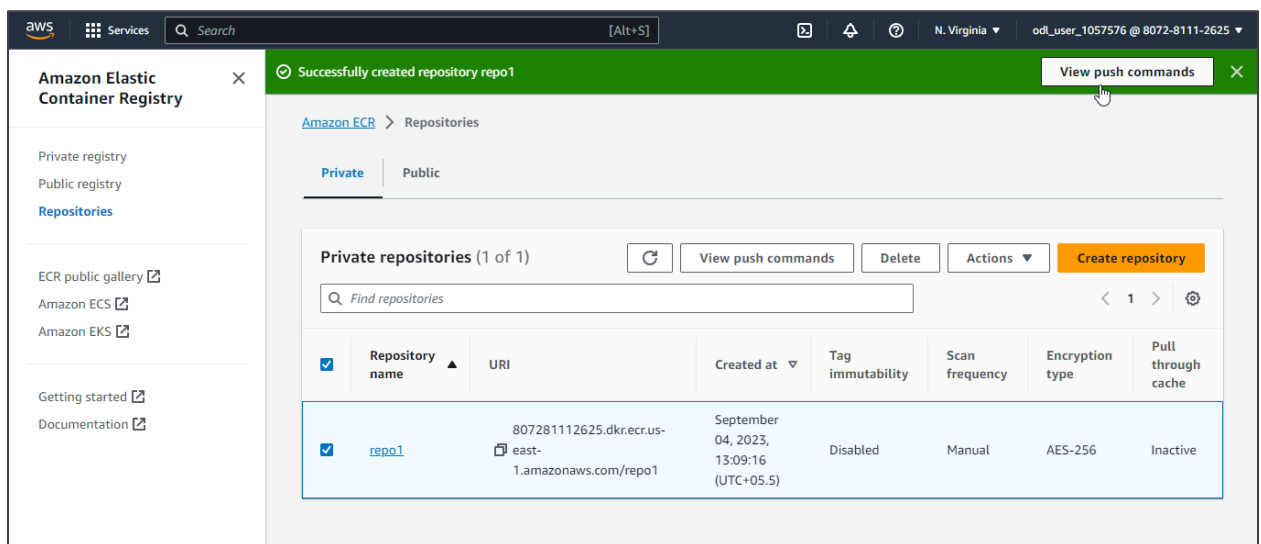
## 1.3 In the **Repository name** section, provide an arbitrary name for your repository, then click on **Create repository**

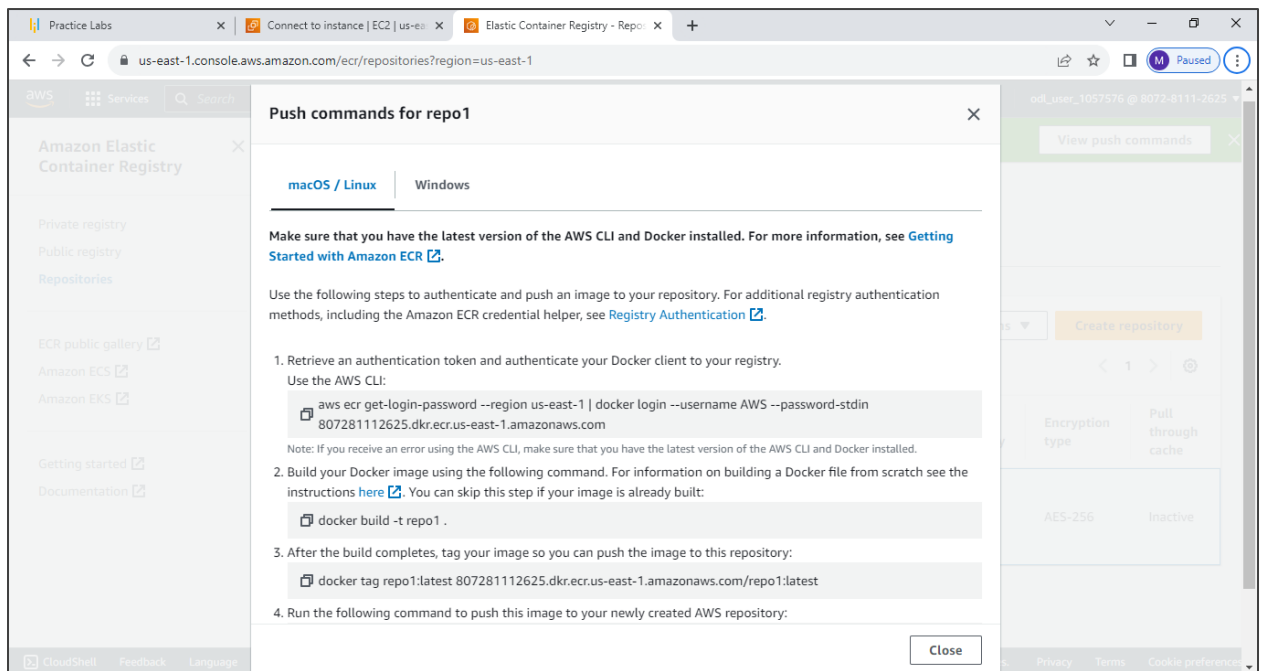


Once the repository is successfully created, it will be visible on the **Repositories** dashboard.



#### 1.4 Click on **View push commands** on the Repositories dashboard

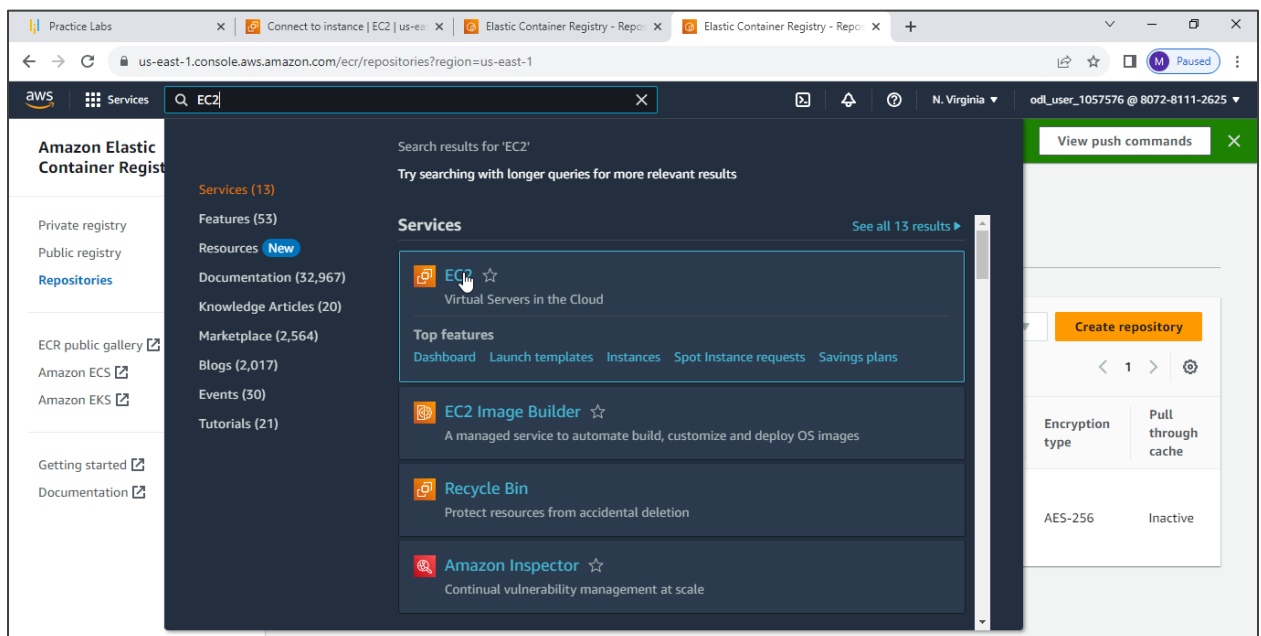




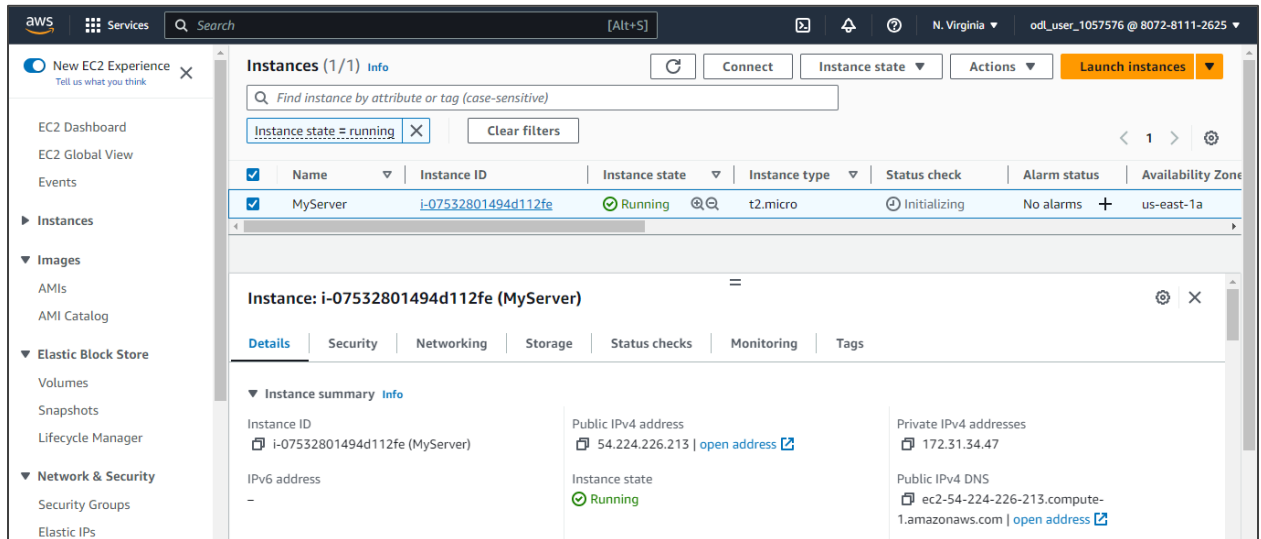
These commands will be used in later steps, so keep this page open and duplicate the new tab for the next steps.

## Step 2: Launch an EC2 instance

2.1 In the AWS Management Console, search for EC2, and then click on EC2

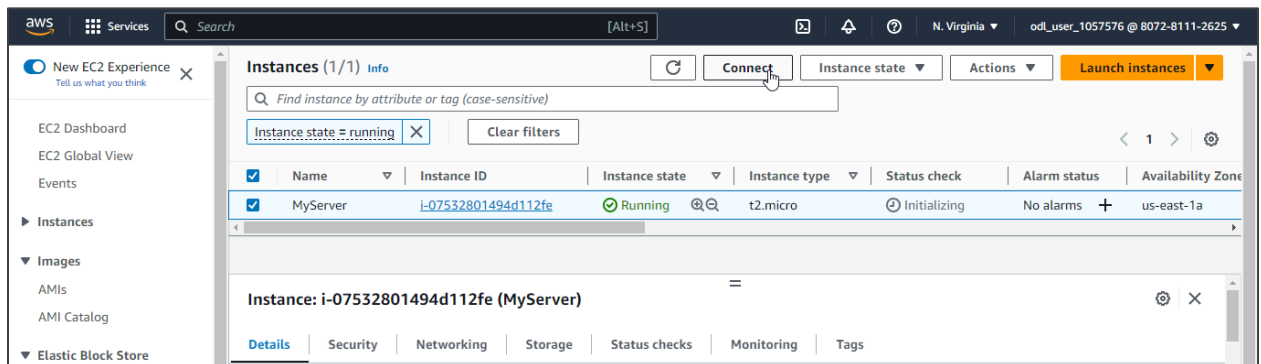


2.2 Launch a new EC2 instance with **Amazon AMI 2** as the operating system. Ensure that you have the necessary security group rules to allow SSH access.



**Note:** Please refer to previous lesson demos on how to launch an EC2 instance.

2.3 Select the instance and click **Connect**



## 2.4 Click Connect

aws Services Search [Alt+S] N. Virginia

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID  
i-07532801494d112fe (MyServer)

Connection Type

☒ Connect using EC2 Instance Connect  
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

☐ Connect using EC2 Instance Connect Endpoint  
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address  
54.224.226.213

User name  
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ec2-user.  
ec2-user

**Note:** In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel Connect

aws Services Search [Alt+S] N. Virginia odl\_user\_1057576 @ 8072-8111-2625

```

  _ _ _ _ _
 _ _ _ _ _ /   Amazon Linux 2 AMI
 _ _ _ _ _

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-34-47 ~]$
  
```

i-07532801494d112fe (MyServer)  
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47

## Step 3: Install Docker on the EC2 instance

3.1 To install Docker, run these commands on your EC2 instance:

```
sudo yum update -y
sudo amazon-linux-extras install docker
sudo systemctl start docker
sudo systemctl enable docker
```

```

aws Services Search [Alt+S] N. Virginia odl_user_1057576 @ 8072-8111-2625
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-34-47 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.7 kB 00:00:00
No packages marked for update
[ec2-user@ip-172-31-34-47 ~]$ sudo amazon-linux-extras install docker
Installing docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
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 00:00:00
amzn2extra-docker | 3.0 kB 00:00:00
amzn2extra-kernel-5.10 | 3.0 kB 00:00:00
(1/7): amzn2-core/2/x86_64/group_gz | 2.5 kB 00:00:00
(2/7): amzn2-core/2/x86_64/updateinfo | 677 kB 00:00:00
(3/7): amzn2extra-docker/2/x86_64/updateinfo | 12 kB 00:00:00
(4/7): amzn2extra-kernel-5.10/2/x86_64/updateinfo | 35 kB 00:00:00

i-07532801494d112fe (MyServer)
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47

```

```

aws Services Search [Alt+S] N. Virginia
51 php8.0 available [ =stable ]
52 tomcat9 available [ =stable ]
53 unbound1.13 available [ =stable ]
54 mariadb10.5 available [ =stable ]
55 kernel-5.10=latest enabled [ =stable ]
56 redis6 available [ =stable ]
57 ruby3.0 available [ =stable ]
58 postgresql12 available [ =stable ]
59 postgresql13 available [ =stable ]
60 mock2 available [ =stable ]
61 dnsmasq2.85 available [ =stable ]
62 kernel-5.15 available [ =stable ]
63 postgresql14 available [ =stable ]
64 firefox available [ =stable ]
65 lustre available [ =stable ]
66 php8.1 available [ =stable ]
67 awscli1 available [ =stable ]
68 php8.2 available [ =stable ]
69 dnsmasq available [ =stable ]
70 unbound1.17 available [ =stable ]
71 golang1.19 available [ =stable ]
72 collectd-python3 available [ =stable ]
[ec2-user@ip-172-31-34-47 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-34-47 ~]$ sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

i-07532801494d112fe (MyServer)
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47

```

### 3.2 Add the **ec2-user** to the **docker** group to allow running Docker commands without **sudo**: **sudo usermod -a -G docker ec2-user**

```
aws Services Search [Alt+S]
55 kernel-5.10=latest enabled [ =stable ]
56 redis6 available [ =stable ]
57 ruby3.0 available [ =stable ]
58 postgresql12 available [ =stable ]
59 postgresql13 available [ =stable ]
60 mock2 available [ =stable ]
61 dnsmasq2.85 available [ =stable ]
62 kernel-5.15 available [ =stable ]
63 postgresql14 available [ =stable ]
64 firefox available [ =stable ]
65 lustre available [ =stable ]
66 php8.1 available [ =stable ]
67 awscli1 available [ =stable ]
68 php8.2 available [ =stable ]
69 dnsmasq available [ =stable ]
70 unbound1.17 available [ =stable ]
71 golang1.19 available [ =stable ]
72 collectd-python3 available [ =stable ]
[ec2-user@ip-172-31-34-47 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-34-47 ~]$ sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-34-47 ~]$ sudo usermod -a -G docker ec2-user
```

After running this command, you should log out and log back in to the EC2 instance to apply the group changes.

### 3.3 To log out, simply type: **exit**

```
aws Services Search [Alt+S]
55 kernel-5.10=latest enabled [ =stable ]
56 redis6 available [ =stable ]
57 ruby3.0 available [ =stable ]
58 postgresql12 available [ =stable ]
59 postgresql13 available [ =stable ]
60 mock2 available [ =stable ]
61 dnsmasq2.85 available [ =stable ]
62 kernel-5.15 available [ =stable ]
63 postgresql14 available [ =stable ]
64 firefox available [ =stable ]
65 lustre available [ =stable ]
66 php8.1 available [ =stable ]
67 awscli1 available [ =stable ]
68 php8.2 available [ =stable ]
69 dnsmasq available [ =stable ]
70 unbound1.17 available [ =stable ]
71 golang1.19 available [ =stable ]
72 collectd-python3 available [ =stable ]
[ec2-user@ip-172-31-34-47 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-34-47 ~]$ sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-34-47 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-34-47 ~]$ exit
logout

i-07532801494d112fe (MyServer)
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47
```



```
aws Services [Alt+S] N. Virginia
Last login: Mon Sep 4 09:46:41 2023 from ec2-18-206-107-28.compute-1.amazonaws.com

 _ | _ | _ |
 _ | ( _ | /
 _ | \ _ | _ |

Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-34-47 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[ec2-user@ip-172-31-34-47 ~]$ vi Dockerfile
```

4.2 Paste the following code:

**FROM** ubuntu:18.04

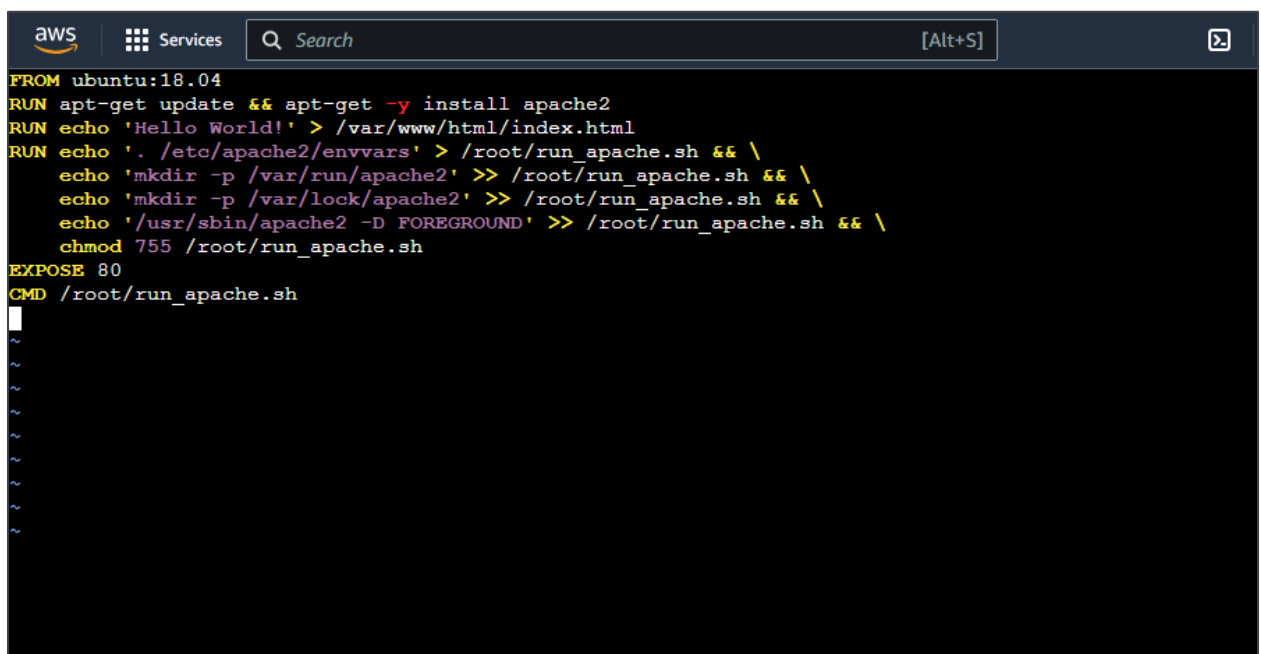
**RUN** apt-get update && apt-get -y install apache2

**RUN** echo 'Hello World!' > /var/www/html/index.html

**RUN** echo './etc/apache2/envvars' > /root/run\_apache.sh && \  
echo 'mkdir -p /var/run/apache2' >> /root/run\_apache.sh && \  
echo 'mkdir -p /var/lock/apache2' >> /root/run\_apache.sh && \  
echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run\_apache.sh && \  
chmod 755 /root/run\_apache.sh

**EXPOSE** 80

**CMD** /root/run\_apache.sh



The screenshot shows an AWS Cloud9 IDE interface. At the top, there is a header bar with the AWS logo, a 'Services' menu, a search bar, and a keyboard shortcut '[Alt+S]'. Below the header is a terminal window with a dark background and light-colored text. The terminal displays the following Dockerfile code:

```
FROM ubuntu:18.04
RUN apt-get update && apt-get -y install apache2
RUN echo 'Hello World!' > /var/www/html/index.html
RUN echo './etc/apache2/envvars' > /root/run_apache.sh && \
    echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \
    echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \
    chmod 755 /root/run_apache.sh
EXPOSE 80
CMD /root/run_apache.sh
```

Below the code, there are several tilde (~) characters, indicating that the terminal is in a shell state and ready for input.

4.3 To save and quit from the vi editor, press the escape key and enter `:wq`

```

aws Services Search [Alt+S]
FROM ubuntu:18.04
RUN apt-get update && apt-get -y install apache2
RUN echo 'Hello World!' > /var/www/html/index.html
RUN echo './etc/apache2/envvars' > /root/run_apache.sh && \
  echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \
  echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \
  echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \
  chmod 755 /root/run_apache.sh
EXPOSE 80
CMD /root/run_apache.sh

~
~
~
~
~
~
~
:wq

```

**Note:** Before pushing the Docker image into your ECR repository, make sure that AWS CLI on your EC2 instance is configured with the necessary credentials.

IAM > Users > odl user\_1057576 > Create access key

Step 1  
[Access key best practices & alternatives](#)

Step 2 - optional  
[Set description tag](#)

Step 3  
**Retrieve access keys**

### Retrieve access keys [Info](#)

**Access key**  
 If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key Copied

Access key	Secret access key
AKIA3X5NUGYYW4QDAEPE	***** <a href="#">Show</a>

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

```
[ec2-user@ip-172-31-34-47 ~]$ aws configure
AWS Access Key ID [None]: AKIA3X5NUGYY747RS34V
AWS Secret Access Key [None]: EsrB8LHYBkcr4Mw/E2TQBXdfcSBtTvhpeswQ4se9
Default region name [None]: us-east-1
Default output format [None]:
```

#### 4.4 Build and push the Docker image into your ECR repository by following the push commands from the ECR page

The screenshot shows the AWS Elastic Container Registry console. The 'macOS / Linux' tab is selected, displaying instructions for pushing a Docker image. The instructions are as follows:

1. Retrieve an authentication token and authenticate your Docker client to your registry.  
Use the AWS CLI:  

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 807281112625.dkr.ecr.us-east-1.amazonaws.com
```

  
Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:  

```
docker build -t repo1 .
```
3. After the build completes, tag your image so you can push the image to this repository:  

```
docker tag repo1:latest 807281112625.dkr.ecr.us-east-1.amazonaws.com/repo1:latest
```
4. Run the following command to push this image to your newly created AWS repository:  

```
docker push 807281112625.dkr.ecr.us-east-1.amazonaws.com/repo1:latest
```

- Copy and run the first command

The screenshot shows a terminal window with the following output:

```
[ec2-user@ip-172-31-34-47 ~]$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 807281112625.dkr.ecr.us-east-1.amazonaws.com
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-172-31-34-47 ~]$
```

Below the terminal window, a system message box displays the instance ID and IP addresses:

```
i-07532801494d112fe (MyServer)
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47
```

- Run the second command to build the Docker image

```
aws Services Search [Alt+S] N. Virginia odl_user_1057576 @ 8072-8111-2625
[ec2-user@ip-172-31-34-47 ~]$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 807281112625.dkr.ecr.us-east-1.amazonaws.com
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-172-31-34-47 ~]$ docker build -t repo1 .
Sending build context to Docker daemon 10.38MB
Step 1/6 : FROM ubuntu:18.04
18.04: Pulling from library/ubuntu
7c457f213c76: Pull complete
Digest: sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
Status: Downloaded newer image for ubuntu:18.04
--> f9a80a55f492
Step 2/6 : RUN apt-get update && apt-get -y install apache2
--> Running in 13631d543e11
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [83.3 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
i-07532801494d112fe (MyServer)
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47
```

- Use the third command to tag your Docker image

```
aws Services Search [Alt+S] N. Virginia odl_user_1057576 @ 8072-8111-2625
invoke-rc.d: policy-rc.d denied execution of start.
Processing triggers for libc-bin (2.27-3ubuntu1.6) ...
Removing intermediate container 13631d543e11
--> b9f45fc50dfe
Step 3/6 : RUN echo 'Hello World!' > /var/www/html/index.html
--> Running in 17680b483b10
Removing intermediate container 17680b483b10
--> a9fdacff0fb
Step 4/6 : RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && echo '/usr/sbin/apachectl -D FOREGROUND' >> /root/run_apache.sh && chmod 755 /root/run_apache.sh
--> Running in fcdc5b904d9b
Removing intermediate container fcdc5b904d9b
--> 1d8e9e1e74ba
Step 5/6 : EXPOSE 80
--> Running in 0a45a8854b51
Removing intermediate container 0a45a8854b51
--> 885c39606e83
Step 6/6 : CMD /root/run_apache.sh
--> Running in bdce326f5f5b
Removing intermediate container bdce326f5f5b
--> 9da3708b837f
Successfully built 9da3708b837f
Successfully tagged repo1:latest
[ec2-user@ip-172-31-34-47 ~]$ docker tag repo1:latest 807281112625.dkr.ecr.us-east-1.amazonaws.com/repo1:latest
[ec2-user@ip-172-31-34-47 ~]$
i-07532801494d112fe (MyServer)
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47
```

- Push the image into the repository using the fourth command

```

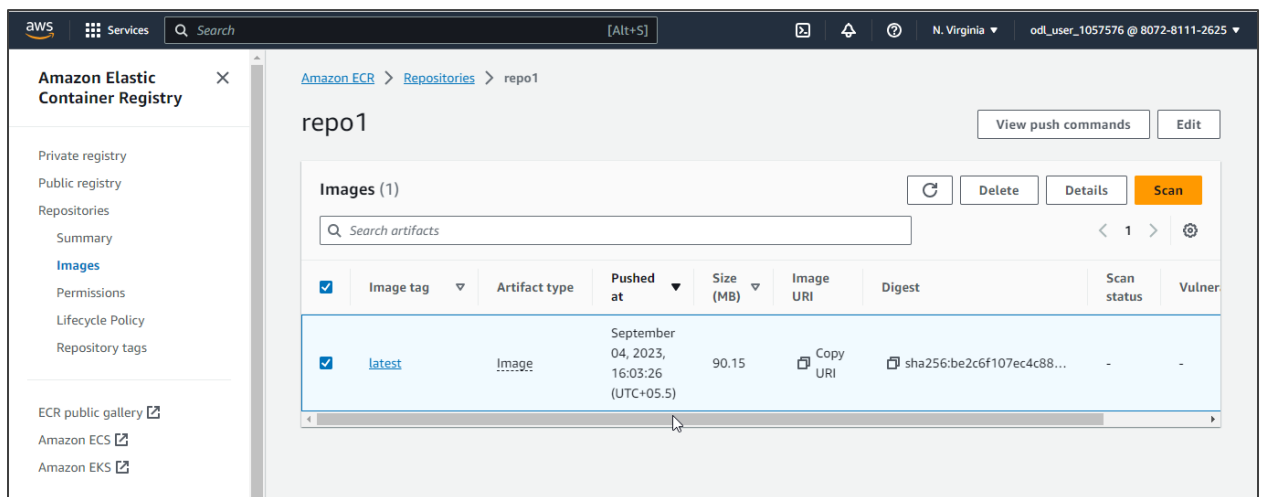
aws
Services
Search
[Alt+S]
N. Virginia
odl_user_1057576 @ 8072-8111-2625

---> a9fdacffd0fb
Step 4/6 : RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && chmod 755 /root/run_apache.sh
---> Running in fcdc5b904d9b
Removing intermediate container fcdc5b904d9b
---> 1dbec9e74ba
Step 5/6 : EXPOSE 80
---> Running in 0a45a8854b51
Removing intermediate container 0a45a8854b51
---> 885c3960e83
Step 6/6 : CMD /root/run_apache.sh
---> Running in bdce326f5f5b
Removing intermediate container bdce326f5f5b
---> 9da3708b837f
Successfully built 9da3708b837f
Successfully tagged repol:latest
[ec2-user@ip-172-31-34-47 ~]$ docker tag repol:latest 807281112625.dkr.ecr.us-east-1.amazonaws.com/repol:latest
[ec2-user@ip-172-31-34-47 ~]$ docker push 807281112625.dkr.ecr.us-east-1.amazonaws.com/repol:latest
The push refers to repository [807281112625.dkr.ecr.us-east-1.amazonaws.com/repol]
a2c7567ceeba: Pushed
87edd34079e1: Pushed
6ba892931abb: Pushed
548a79621a42: Pushed
latest: digest: sha256:be2c6f107ec4c8852c3a7abfbcfc65b543dc77be69c5d586910eb472c1717b61 size: 1155
[ec2-user@ip-172-31-34-47 ~]$

i-07532801494d112fe (MyServer)
PublicIPs: 54.224.226.213 PrivateIPs: 172.31.34.47

```

#### 4.5 In your ECR repository console, find the image you just pushed in step 4.3



You can observe that the Docker image has been successfully pushed into your ECR repository.

By following these steps, you have successfully created an AWS ECR container registry, configured Docker on your EC2 instance, and pushed a Docker image into your ECR repository.