

Demo 1: ECR

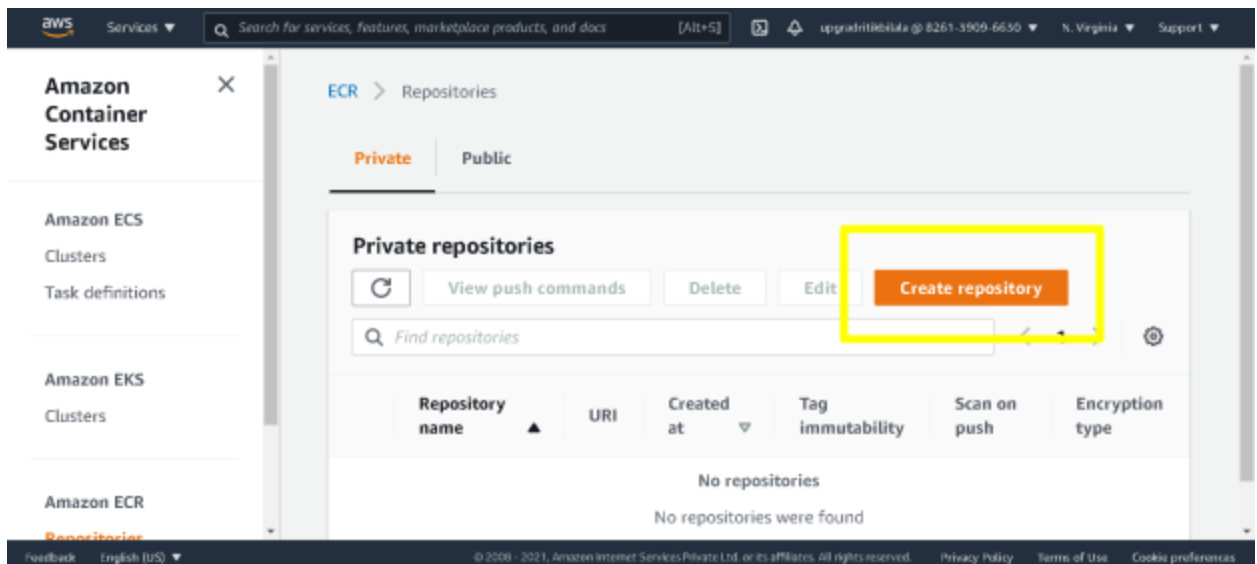
Introduction

Following are the learning objectives of this demonstration:

- Create an ECR repository
- Build an image and push it into the repository from an EC2 instance
- Set up the life cycle policy of ECR

Create an IAM role with permission “[AmazonEC2ContainerRegistryFullAccess](#)” and attach it to the EC2 instance from which you will push images into the repository. If you are using a local machine outside AWS, you will need to configure an AWS account using AWS access keys.

1. Log in to the AWS console and visit the ECR page.



2. Click on create a repository.
3. Provide a repository name and click on the create button.

General settings

Visibility settings [Info](#)
Choose the visibility setting for the repository.

☒ **Private**
Access is managed by IAM and repository policy permissions.

☐ **Public**
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.

826139096630.dkr.ecr.us-east-1.amazonaws.com/

6 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.

Tag immutability [Info](#)
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. [Disable tag immutability to allow image tags to be overwritten.](#)

- Copy your repository URL for the next steps.
(My URL = **826139096630.dkr.ecr.us-east-1.amazonaws.com**)

Repository name	URL	Created at	Tag immutability	Scan on push	Encryption type
upgrad	826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad	19 Mar 2021 12:15:04	Disabled	Disabled	AES-256

- Now go to your repository details and click on the “**view push commands**” button.

upgrad

Images (0)

[View push commands](#) [Delete](#) [Scan](#)

Image tag	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
No images						
No images to display						

- After clicking the button, the following dialogue box will pop-up.

Push commands for upgrad

macOS / Linux

Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

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-
```

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 upgrad .
```
3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag upgrad:latest 826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad:latest
```
4. Run the following command to push this image to your newly created AWS repository:

```
docker push 826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad:latest
```

Close

7. Now SSH into the EC2 instance and log in to the ECR repository using the above image's first command. Remember, your EC2 must have **AWScli** and **docker** installed.

**AWS ecr get-login-password --region us-east-1 | docker login --username AWS
--password-stdin 826139096630.dkr.ecr.us-east-1.amazonaws.com**

```

Last login: Fri Mar 19 13:24:26 2021 from 103.157.221.91
ubuntu@ip-172-31-30-37:~$ aws ecr get-login-password --region us-east-1 | docker login --user
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-30-37:~$

```

8. Now either create your dockerfile or pull your git repository, which contains the dockerfile.
9. Now build an image using **docker build -t <imagename:tag> .**

```

ubuntu@ip-172-31-30-37:~/NGINX$ docker build -t upgrad .
Sending build context to Docker daemon 65.54kB
Step 1/7 : FROM ubuntu
latest: Pulling from library/ubuntu
5d3b2c2d21bb: Pull complete
3fc2062ea667: Pull complete
75adf526d75b: Pull complete
Digest: sha256:b4f9e18267eb98998f6130342baacaeb9553f136142d40959a1b46d6401f0f2b
Status: Downloaded newer image for ubuntu:latest
--> 4dd97cefde62
Step 2/7 : RUN apt-get update && apt-get install -y nginx && rm -rf /var/lib/apt/li
--> Running in f337b975f5c1
--> Running in 888e23383388
Removing intermediate container 6bbd235053db
--> 87366dc14cae
Step 7/7 : EXPOSE 443
--> Running in 815cac12cbc8
Removing intermediate container 815cac12cbc8
--> e19967651a0a
Successfully built e19967651a0a
Successfully tagged upgrad:latest
ubuntu@ip-172-31-30-37:~/NGINX$

```

10. After the build completes, tag your image so you can push the image to this repository using the following commands:

```

docker tag upgrad:latest \
826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad:latest

```

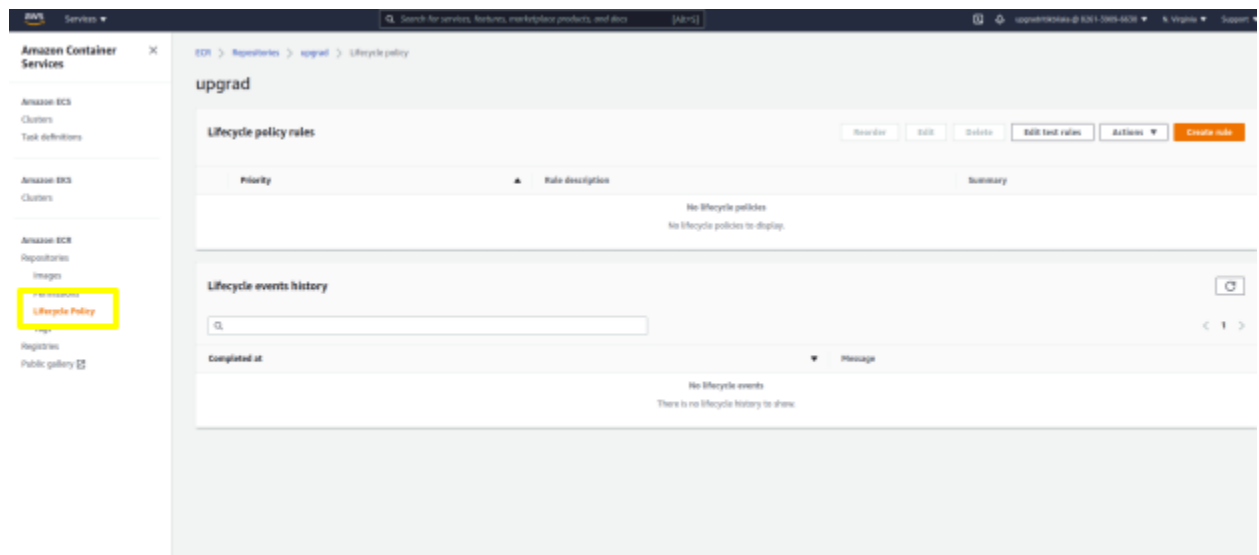
Command : *(docker tag <repo name>:latest <repo url> :latest)*

11. Run the following command to push this image to your newly created AWS repository:
docker push 826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad:latest

```
ubuntu@ip-172-31-30-37:~/NGINX$ docker tag upgrad:latest 826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad:latest
ubuntu@ip-172-31-30-37:~/NGINX$ docker push 826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad:latest
The push refers to repository [826139096630.dkr.ecr.us-east-1.amazonaws.com/upgrad]
03c18ce5e58c: Pushed
c20d459170d8: Pushed
db978cae6a05: Pushed
aeb3f02e9374: Pushed
latest: digest: sha256:17f2d26af3c0d50b16ae6ee1e59f6b81e15d358723f37518f957f71f1feeca71 size: 1155
ubuntu@ip-172-31-30-37:~/NGINX$
```

12. Go to the ECR console and check whether the image is pushed or not.

13. Now from the left panel, select “life cycle policy.”



14. Put an ECR lifecycle policy as given below so that images older than one year get deleted automatically.

Lifecycle rule configuration

If you apply this rule as your lifecycle policy, it may be evaluated immediately. It is recommended that you dry run your rules first.

Rule priority
Specify a rule priority, which must be unique. Values do not need to be sequential across rules in a policy.

1

Rule description
Specify a description for your lifecycle policy.

Remove 1 year old images

Image status
Indicates whether the image is tagged or not.

☐ Tagged
☐ Untagged
☒ Any

Match criteria
Specify the count type to apply to the images. If "countType" is set to "imageCountMoreThan", you also specify "countNumber" to create a rule that sets a limit on the number of images that exist in your repository. If "countType" is set to "sinceImagePushed", you also specify "countUnit" and "countNumber" to specify a time limit on the images that exist in your repository.

Since image pushed 365 Days

Amazon Container Services

- Amazon ECS
 - Clusters
 - Task definitions
- Amazon EKS
 - Clusters
- Amazon ECR
 - Repositories
 - Images
 - Permissions
 - Lifecycle Policy**
 - Tags
 - Registries
 - Public gallery

Lifecycle rule successfully added to policy

ECR > Repositories > upgrad > Lifecycle policy

upgrad

Lifecycle policy rules

Priority	Rule description	Summary
1	1 year old images	expire sinceImagePushed (365 days) any

Lifecycle events history

Completed at

No lifecycle events
There is no lifecycle history to show.

Now images older than 365 days will get deleted from the repository.