Create a Docker container using Dockerfile and store the image in ECR

Lab Steps

Task 1: Sign in to AWS Management Console

1. Click on the **Open Console** button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,

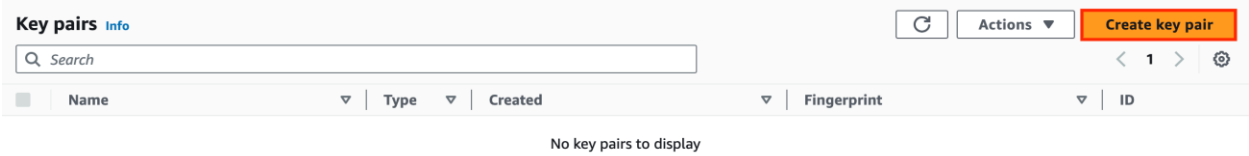
a. Leave the Account ID as default. Never edit/remove the 12-digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.

b. Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign in** button.

3. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1.**

Task 2: Create a Key Pair for the EC2 instances, inside the ECS Cluster

1. In the left navigation pane (scroll down) within **Network & Security**, click on the **Key Pairs**
2. To create a new key pair, click on the **Create Key pair** button.



3. Fill in the details below:

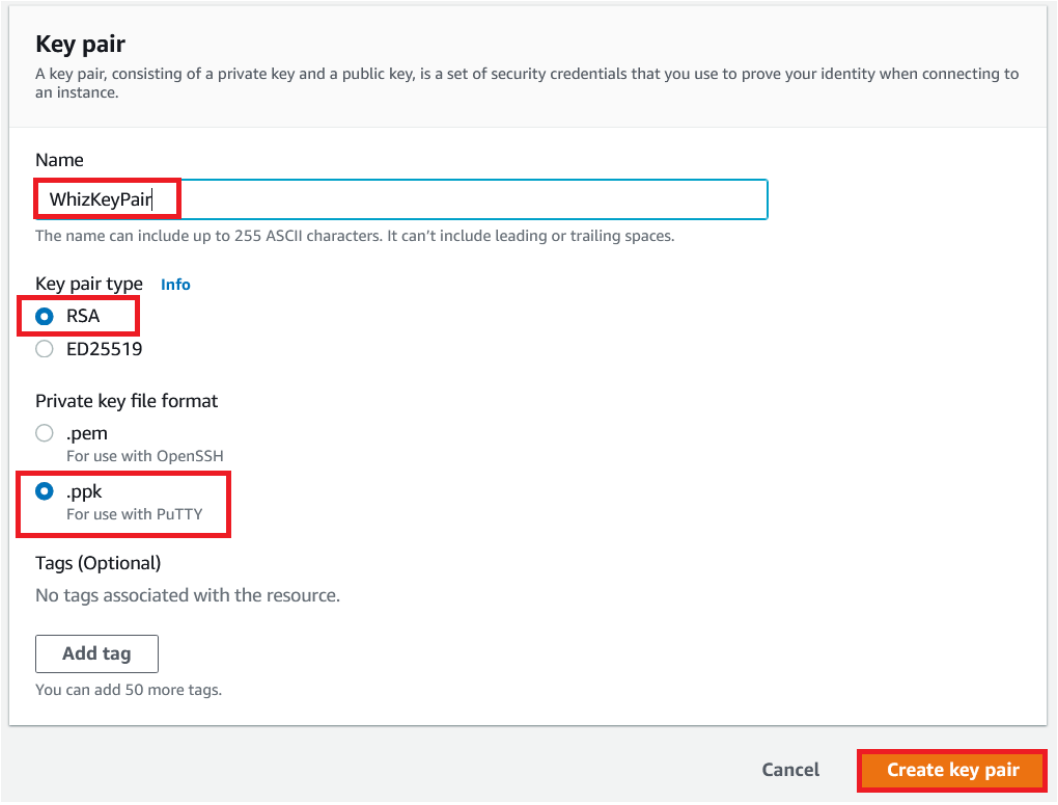
a. Name: Enter ***WhizKeyPair***

b. Key pair type: Select **RSA**

c. Private key file format: Select **pem (Linux & Mac Users)** or **ppk (Windows users)**

d. Leave other options as **default**.

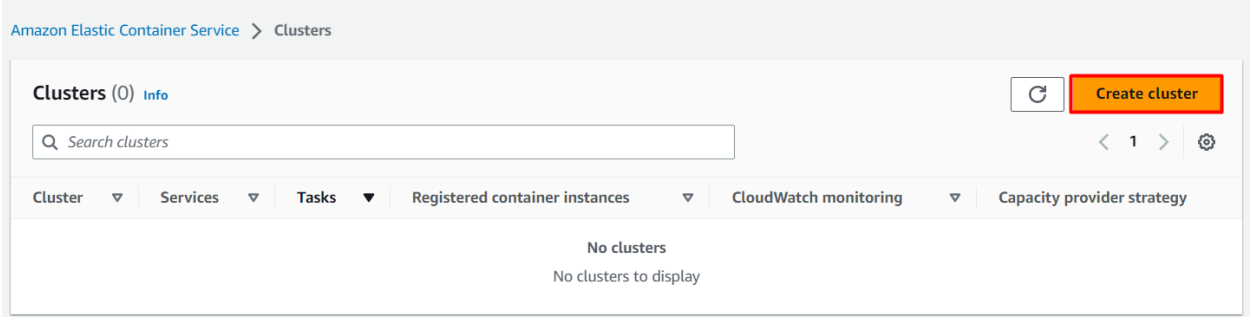
e. Click on the **Create Key pair**.



4. Key pair will be created.

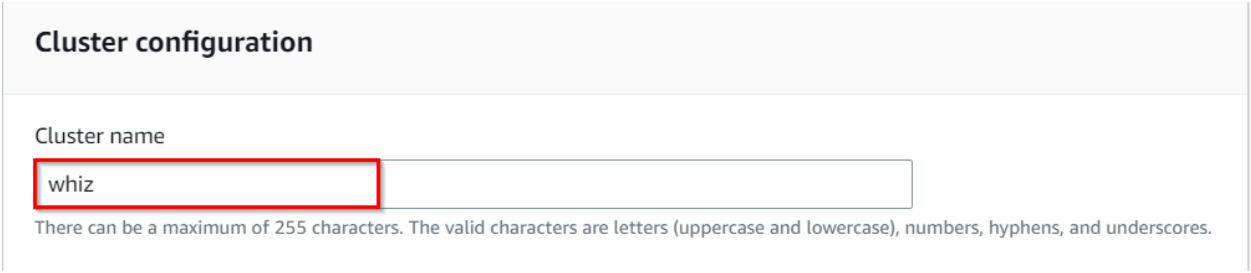
Task 3: Launching an ECS Cluster

1. Make sure you are in the **N.Virginia** Region. Navigate to **Elastic Container Service** by clicking on the **Services** menu in the top, then click on **Elastic Container Service** in the **Containers** section.
2. On the left sidebar, click on the **Clusters** option present under the **Amazon ECS** section, then Click on the **Create cluster** button.



3. In Cluster configuration :

   Cluster name : Enter **whiz**



4. In Infrastructure Section :

UnCheck AWS Fargate and **Check** Amazon EC2 Instances checkbox:

Auto Scaling group (ASG) : Select **Create new ASG**

Operating System/Architecture : Select **Amazon Linux 2**

EC2 instance type : Select **t2.micro**

Desired capacity :

Minimum : Enter **1**

Maximum : Enter **2**

SSH Key pair: Select **WhizKeyPair**

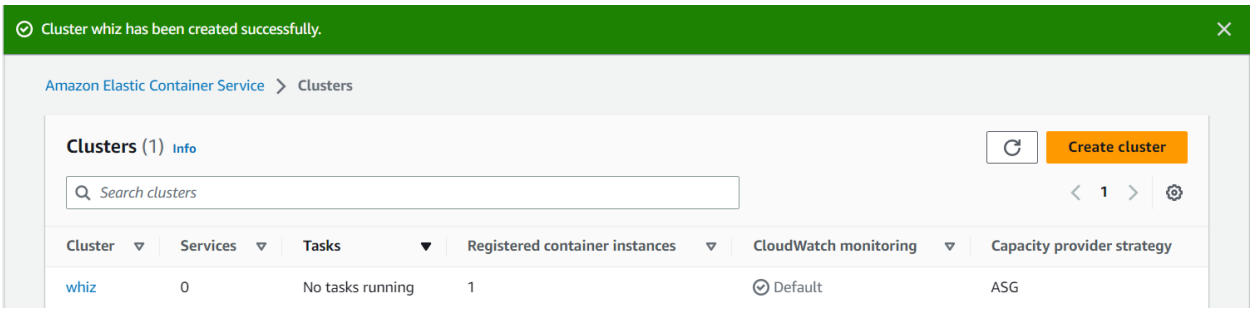
5. In Network settings:

Security group: Select **Create new security group**.
Security name : Enter **mysg**
Security group description : Enter **Security group for instance creation**
Inbound rules for security groups:
Type : Select **SSH**
Source : Select **Anywhere**

6. Keep rest things as default and click on **Create** button

7. **whiz** ECS Cluster will be created

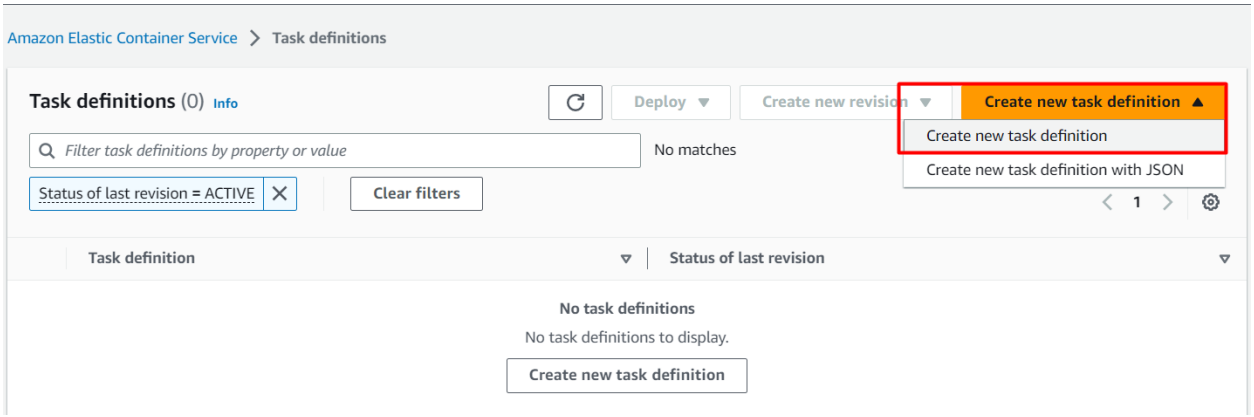
8. It will take few minutes to provision the ECS Instance.



Task 4: Create Task Definitions

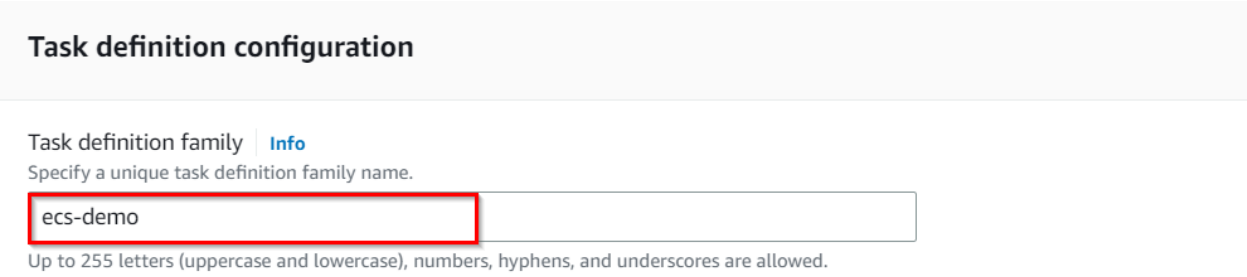
In this task, we are going to create a task definition for the cluster. A task definition provides a blueprint for creating tasks, which are the basic unit of work in ECS. Each task definition can define one or more containers that are run together on the same underlying EC2 instances or Fargate tasks.

1. On the left sidebar, click on the **Task Definitions** option present under the **Amazon ECS** section.
2. Click on the **Create new task definition** button.



3. In Task definition configuration:

Task definition family: Enter ***ecs-demo***



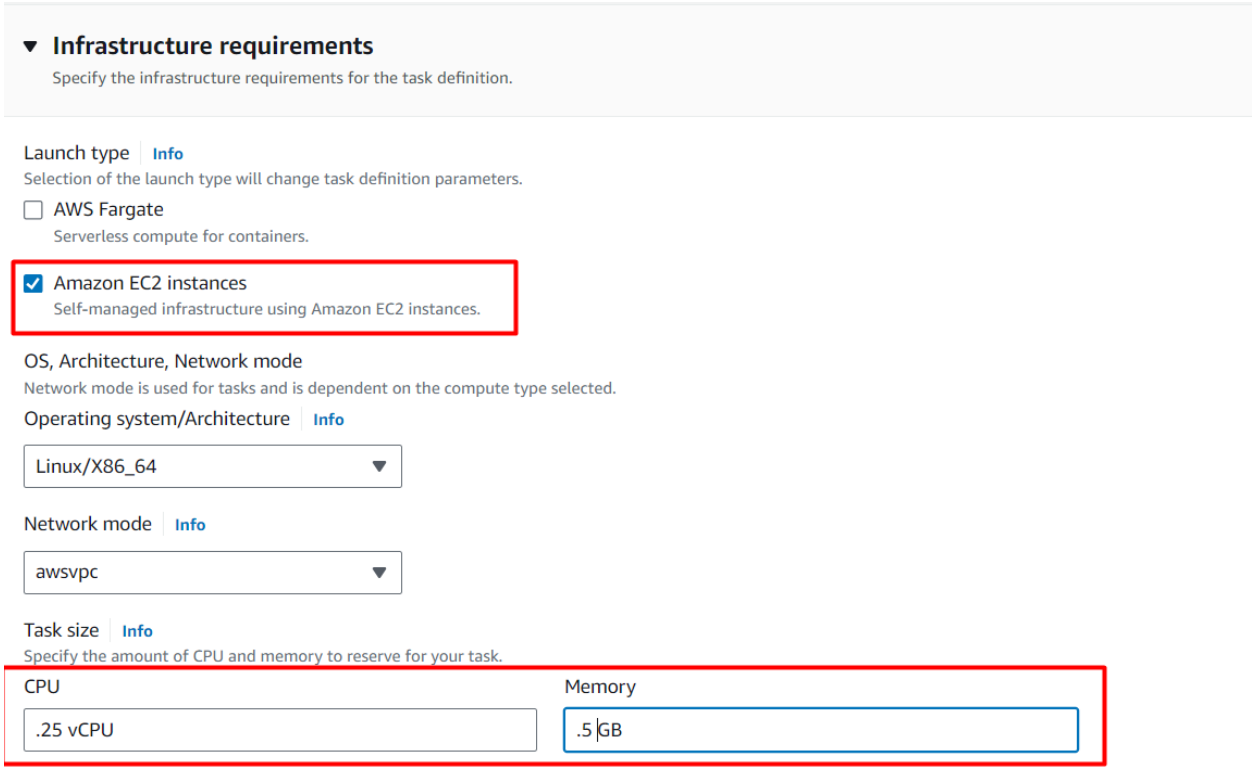
4. For Environment:

App Environment : **Remove AWS Fargate** And Select **Amazon EC2 Instances**

Task Size :

CPU : Enter **.25 VCPU**

Memory : Enter **.5 GB**



For Container Details :

Enter **httpd**

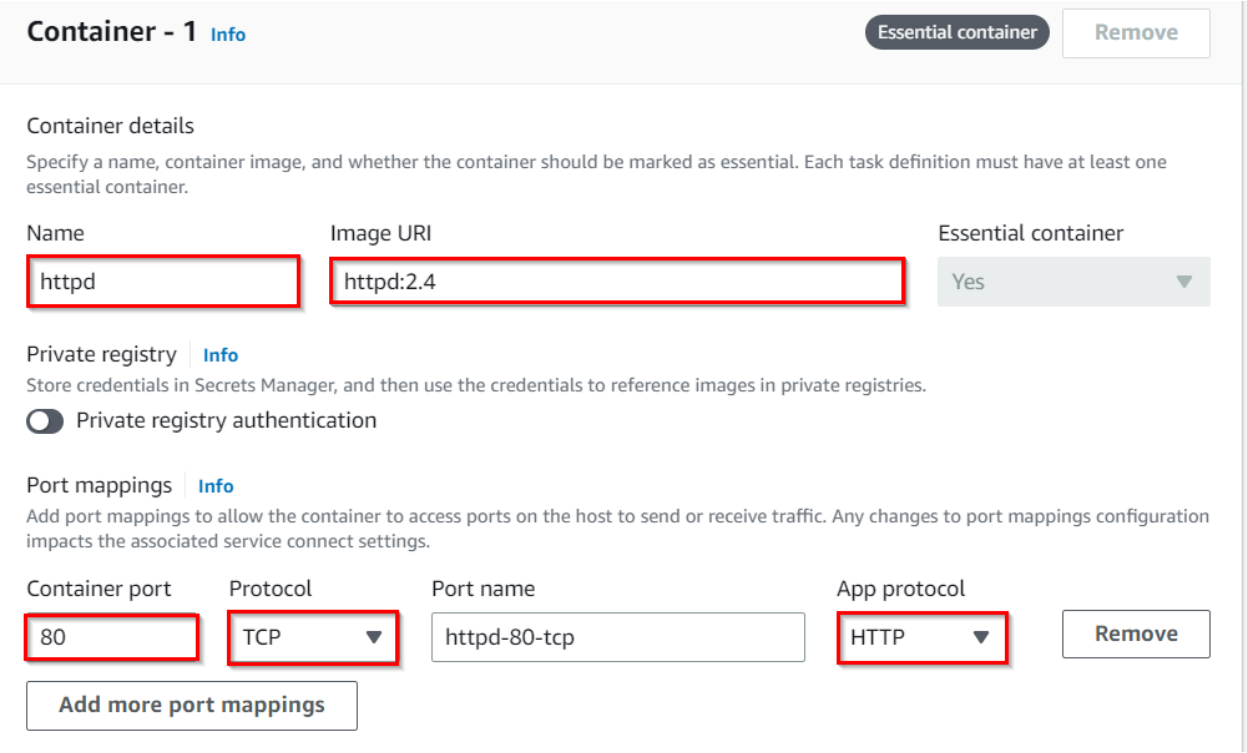
Image URL: Enter **httpd:2.4**

For Port mappings:

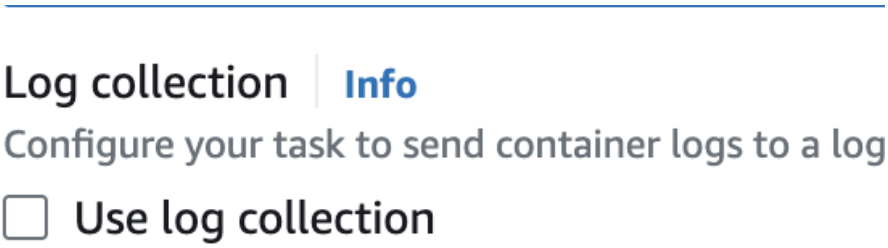
Container Port: Enter **80**

Protocol: Select **TCP**

App protocol: Select **HTTP**



5. **Uncheck** use log collection box.

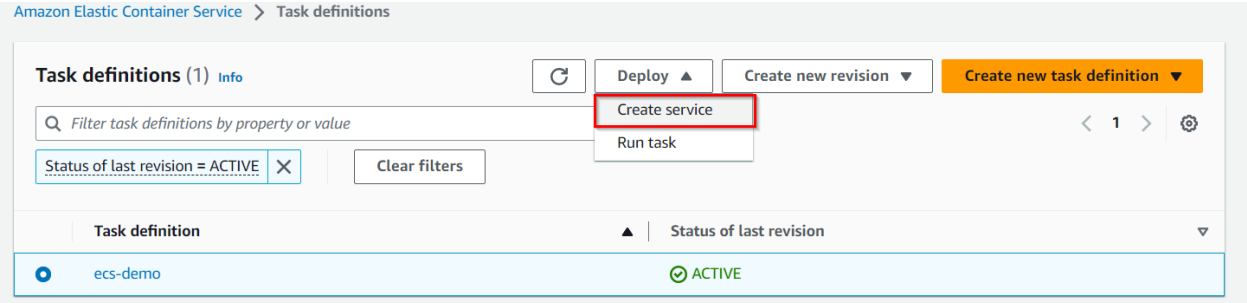


6. Keep rest things as default and click on the **Create** button.
7. Task Definition **ecs-demo** is now created.

Task 5: Create a service and start HTTPD container in ECS

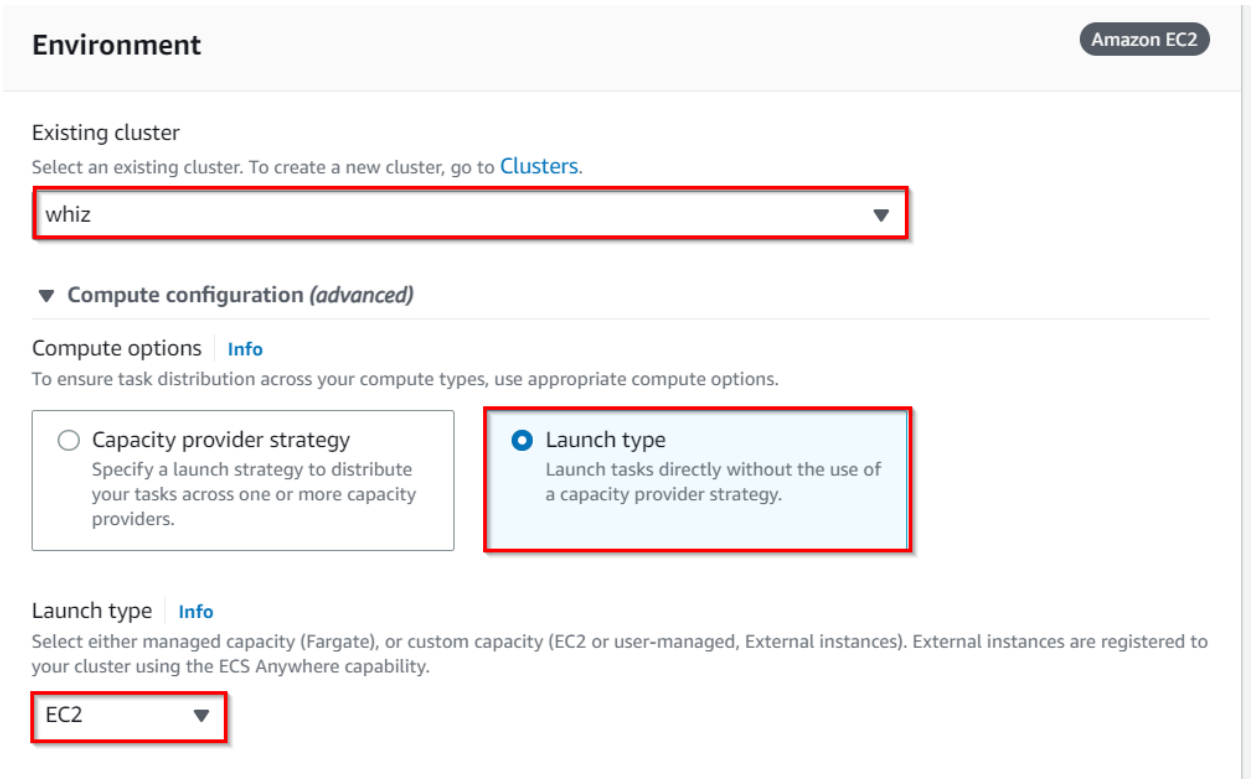
In this task, we are going to create a service and start the HTTPD container. In AWS ECS (Elastic Container Service), a service is a long-running task that ensures that a specified number of instances of a task definition are running and maintained in an ECS cluster. Services allow you to define the desired state of your tasks and automatically handle task placement, scaling, and recovery.

1. On the left sidebar, click on the **Task definitions** option present under the **Amazon ECS** section.
2. Select **ecs-demo** and click on **Deploy** and click **Create service** button.



3. Existing cluster : Select **whiz** cluster
4. Compute options : Select **Launch Type**

   Launch type: Select **EC2**



5. In Deployment configurations :

   Service name: Enter **httpd**

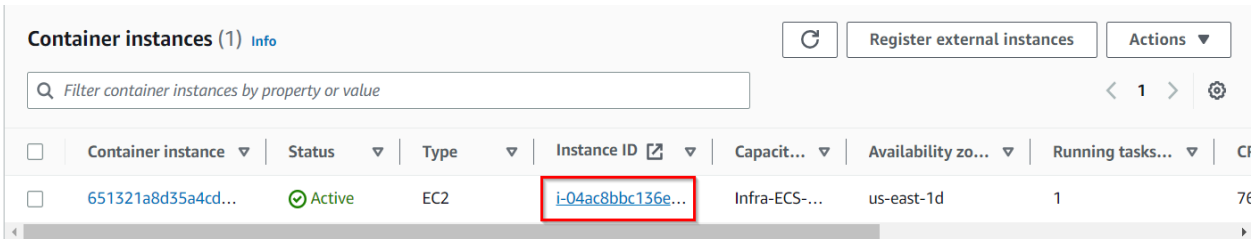
   Service type: Select **REPLICA**

   Desired tasks : Enter **1**

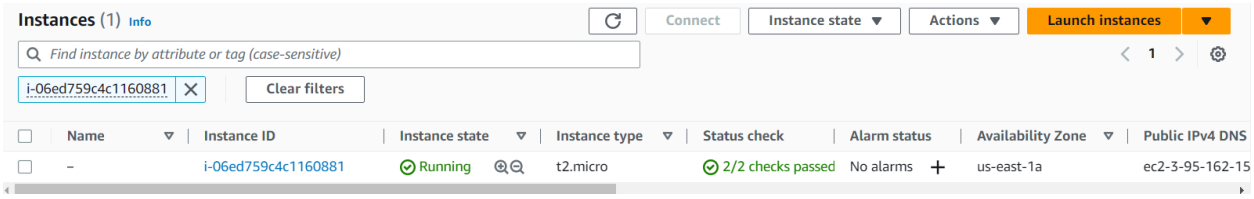
6. Keep other options as default, and click on the **Create** button.

Task 6: Test the HTTPD container in ECS Cluster

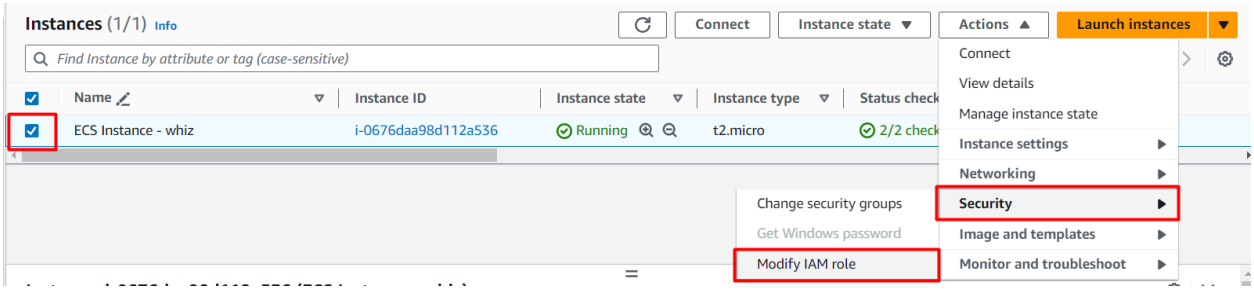
1. On the left sidebar, click on the **Clusters** option present under the **Amazon ECS** section.
2. whiz ECS Cluster will be listed here, click on the **whiz.**
3. To view the ECS Instance, switch to the **Infrastructure** tab.
4. Scroll down to the Container Instances



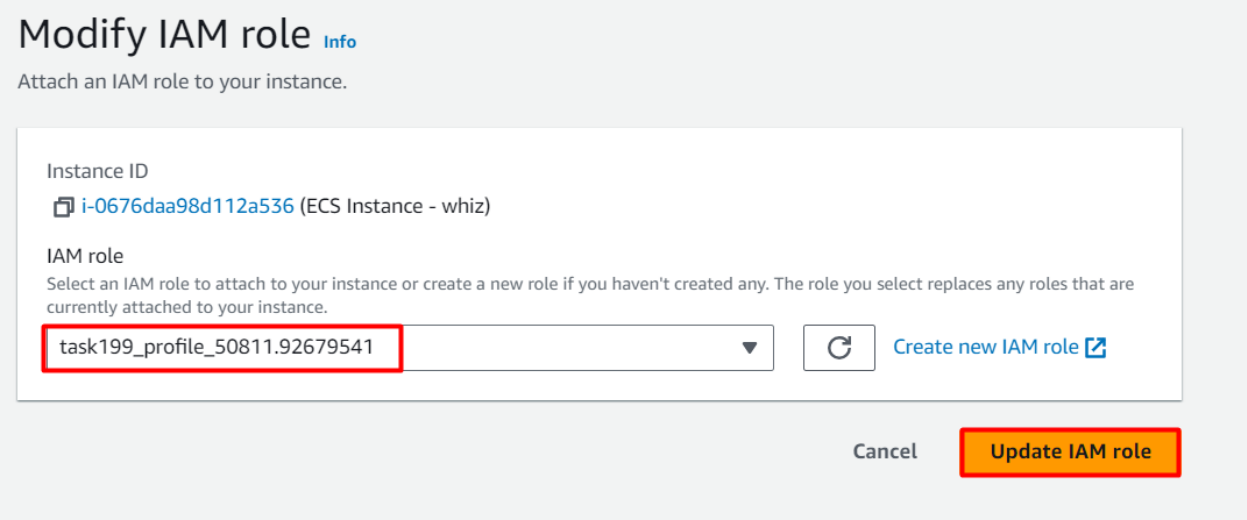
5. **Click** on the EC2 Instance ID, and you will be redirected to see the running EC2 instance.



6. Select the EC2 Instance and Go to **Actions**. Select **Modify IAM role** under **Security.**



7. Select **task_199_<RANDOM_NUMBER>** from the drop-down and click on **Update IAM role** button.



Task 6: SSH into the underlying EC2 instance

1. Please follow the steps in SSH into EC2 Instance.

Task 7: Create a Dockerfile and build a docker image

1. Get the root access using the following command:

```
sudo su
```

2. Now run the updates using the following command:

```
yum -y update
```

3. Check the Docker version by running the following command:

```
docker version
```

4. Check all the docker processes running in the ECS Cluster by running the following command:

```
docker ps
```

5. Install AWS CLI by running the following command:

```
yum install awscli -y
```

6. Create a Dockerfile by running the following command:

```
touch Dockerfile
```

7. Put the instructions in the Dockerfile by running the following command:

```
vi Dockerfile
```

8. Paste the below code in Dockerfile after pressing **i** or insert mode

```
#This is a sample Image
FROM ubuntu
MAINTAINER someone@example.com
RUN apt-get update
RUN apt-get install nginx -y
CMD ["echo","Image created"]
```

9. While pasting the above script in Dockerfile's vi editor, you may get issue with formatting. Please follow the below instructions:

10. Save the file by pressing **esc** and enter: **wq** to close the vi editor.
11. Run the Build command:

```
docker build -t whiz .
```

Note: demo will be the name of our newly created docker image. Don't forget to add a **(.) dot** after the demo (image name).

12. Check the availability of a new image, by running the following command:

```
docker images
```

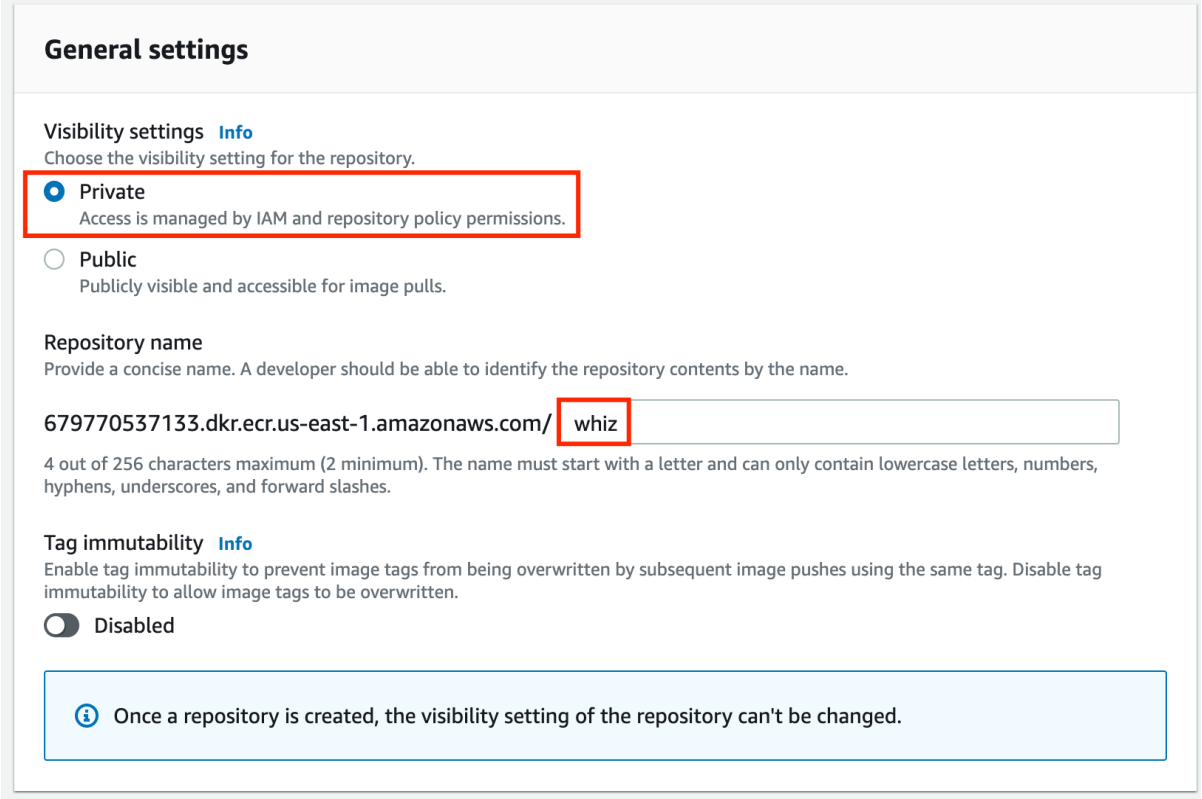
Task 8: Create a repository in ECR and run the push commands

1. Navigate to **Elastic Container Registry** by clicking on the **Services** menu at the top, then click on **Elastic Container Registry** in the **Containers** section.
2. Click on **Get Started** under Create a Repository.
3. Fill in the below details to create an ECR repository.

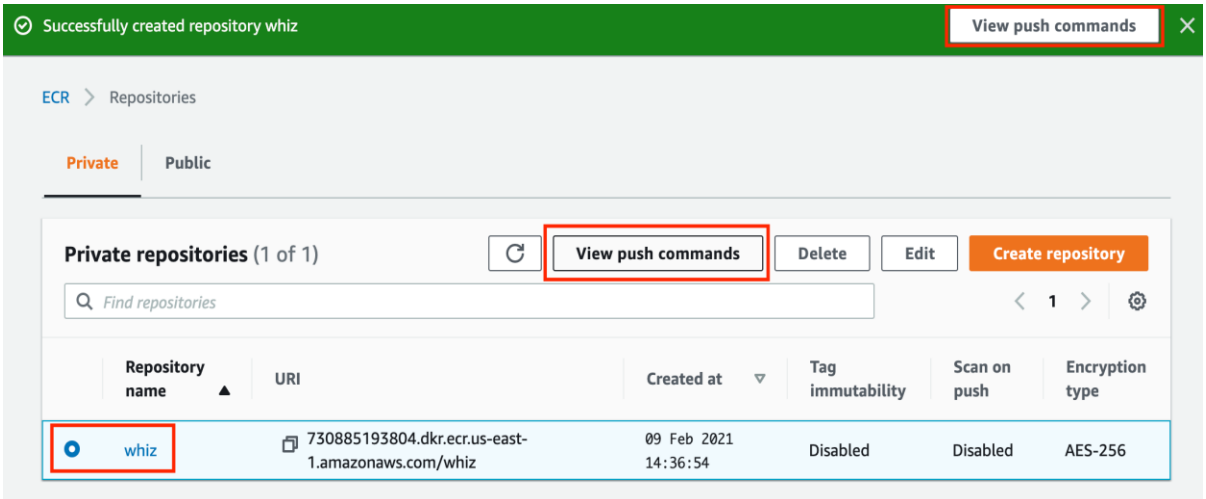
a. Visibility settings: Select **Private**

b. Repository name: Enter ***whiz***

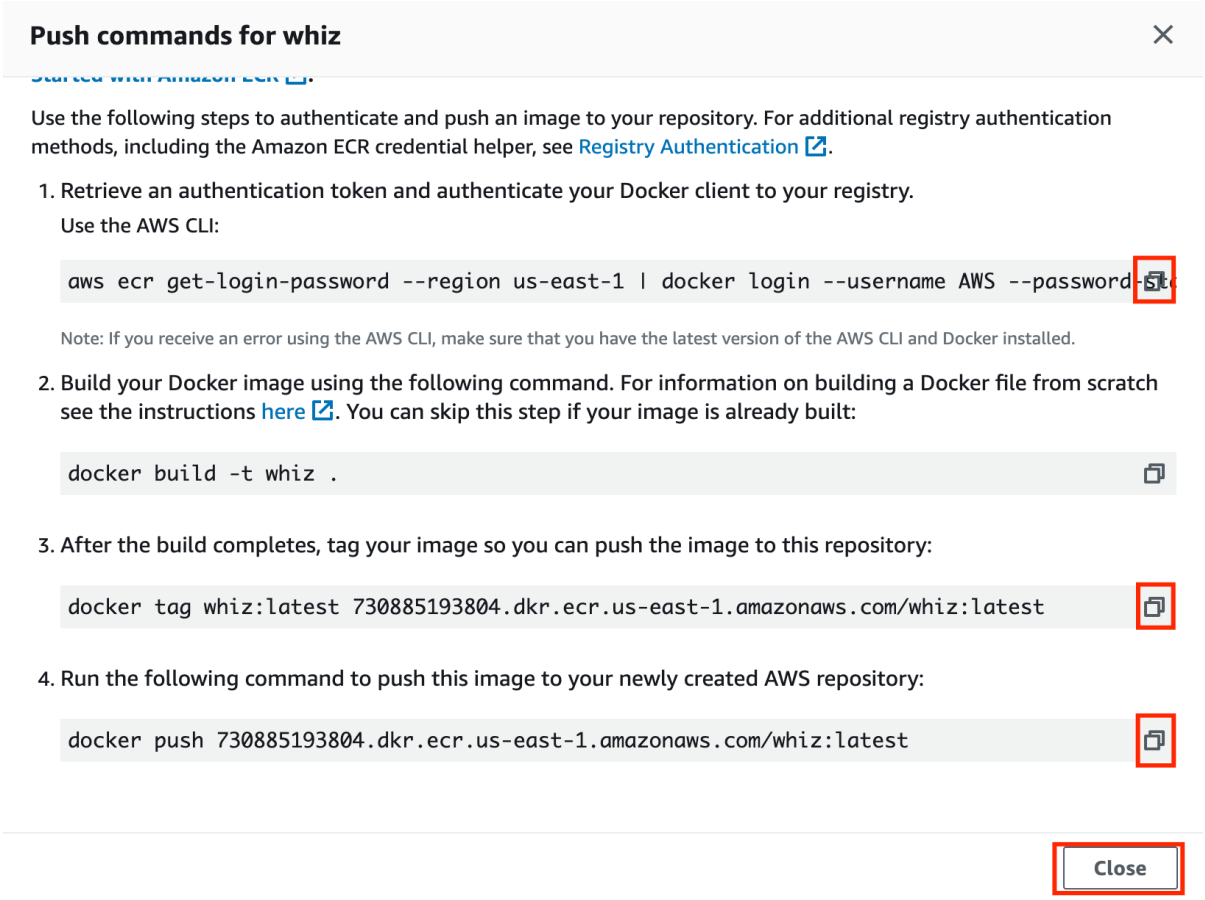
c. Keep other options as default.



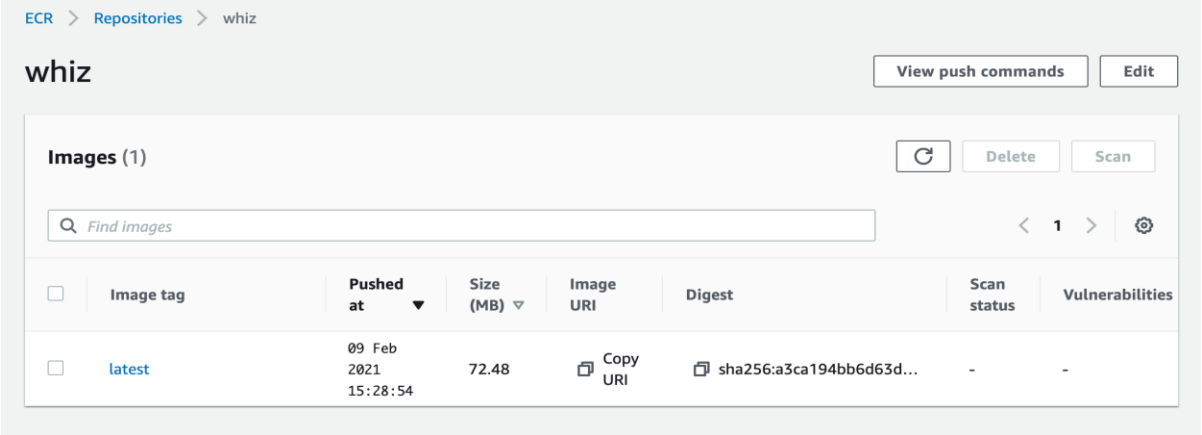
4. Finally, click on the **Create Repository** to create an ECR Repository.
5. The repository is now created, select the repository and click on the **View push commands** button to upload the created custom image in the ECR Repository.



6. **Copy the commands one by one** and **paste the same on the terminal** to execute.
7. Since we have already built the Dockerfile to the docker image. We can **skip copying and pasting the 2nd command**.



8. After running all the 3 commands, can click on the repository name to check the created image.



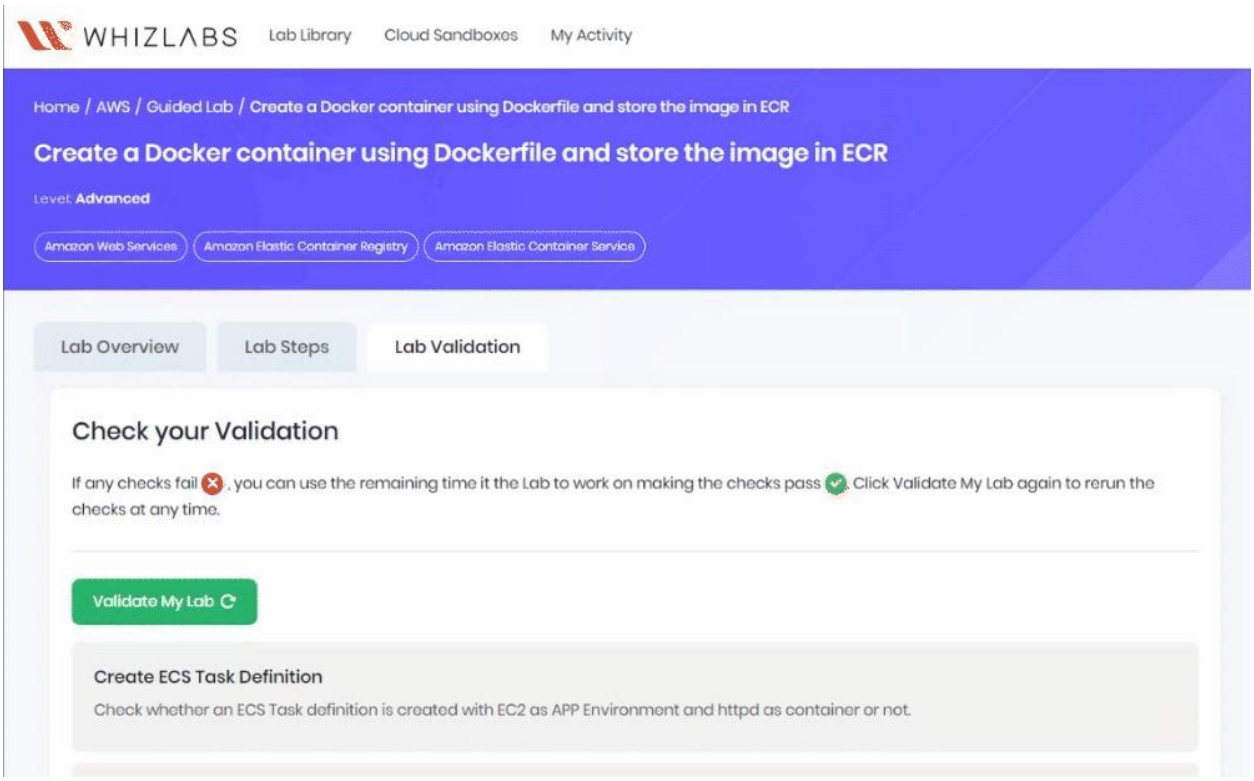
9. Image is now pushed to ECR repository, it can now be used in task definition to create services and tasks.

Do you know ?

AWS ECS is often used in conjunction with other AWS services, such as Amazon ECR (Elastic Container Registry) to store and manage container images, AWS CloudFormation for infrastructure as code, AWS IAM for access management, and AWS CloudWatch for monitoring and logging.

Task 9: Validation Test

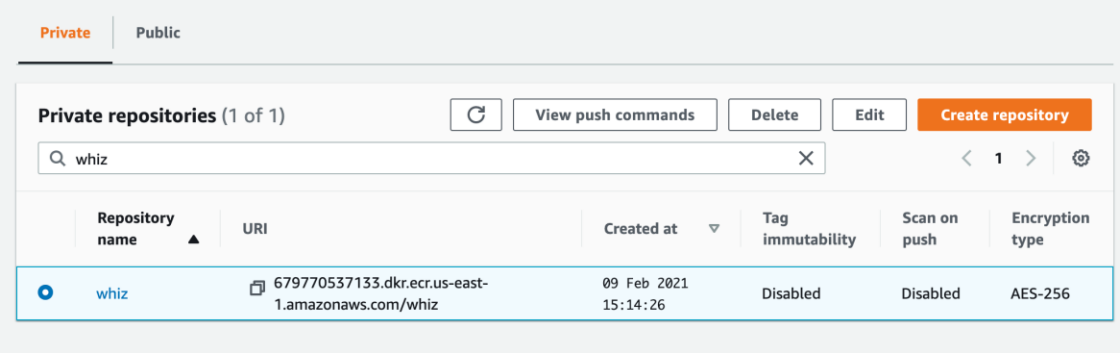
1. Once the lab steps are completed, please click on the **Validation** button on the Right side panel.
2. This will validate the resources in the AWS account and displays whether you have completed this lab successfully or not.
3. Sample output :



Task 10: Delete AWS Resources

Deleting ECR repository

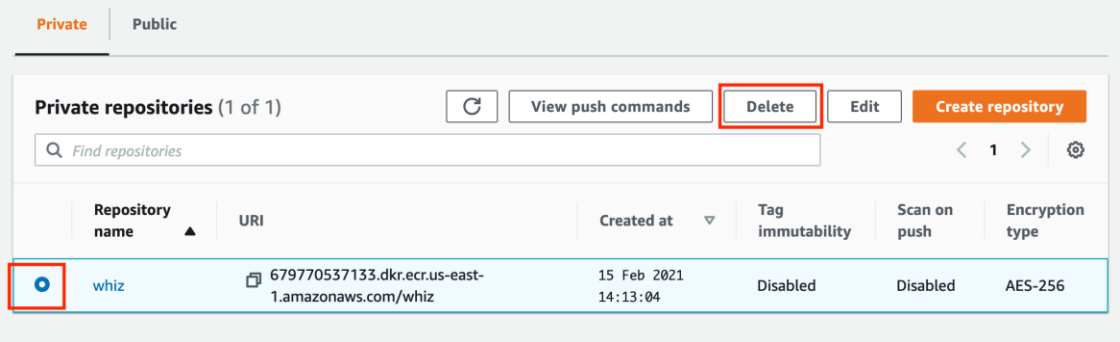
1. Navigate to **Elastic Container Registry** by clicking on the **Services** menu in the top, then click on **Elastic Container Registry** in the **Containers** section.
2. On the left sidebar, click on the **Repositories** option present under the **Amazon ECR** section.
3. whiz ECS repository will be listed here.



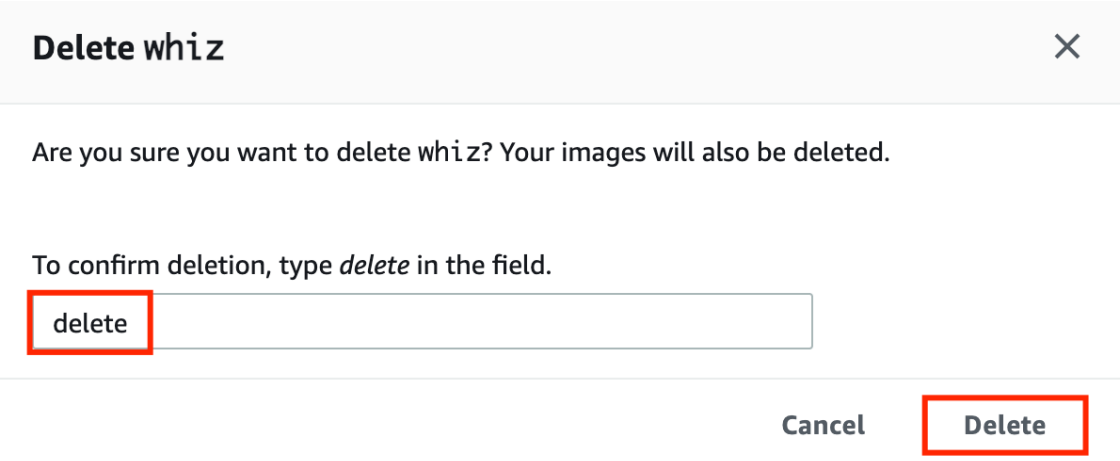
4. To delete the service, do the following task:

a. Select the present repository,

b. Click on the **Delete** button.



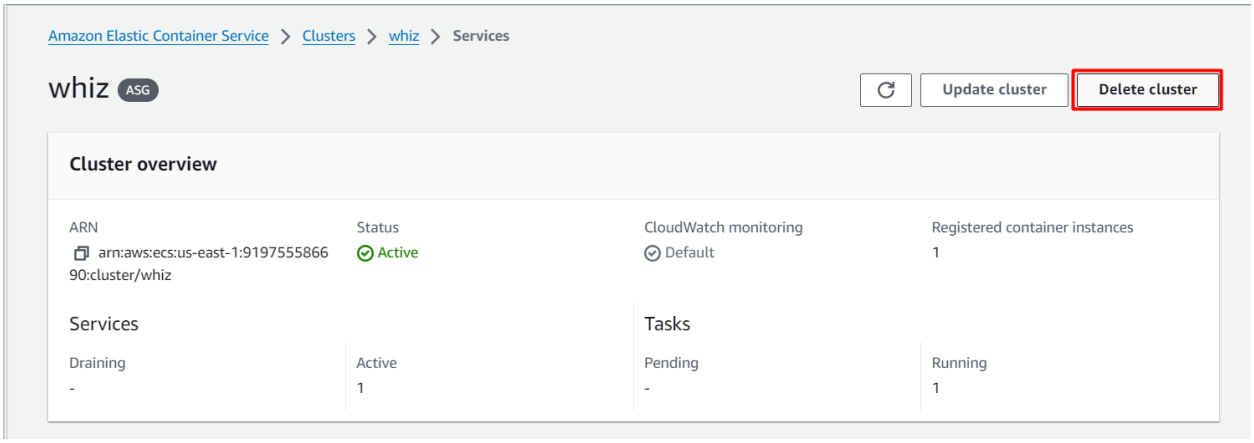
5. Confirm the deletion by typing the ***delete*** phrase in the required field, and click on the **Delete** button.



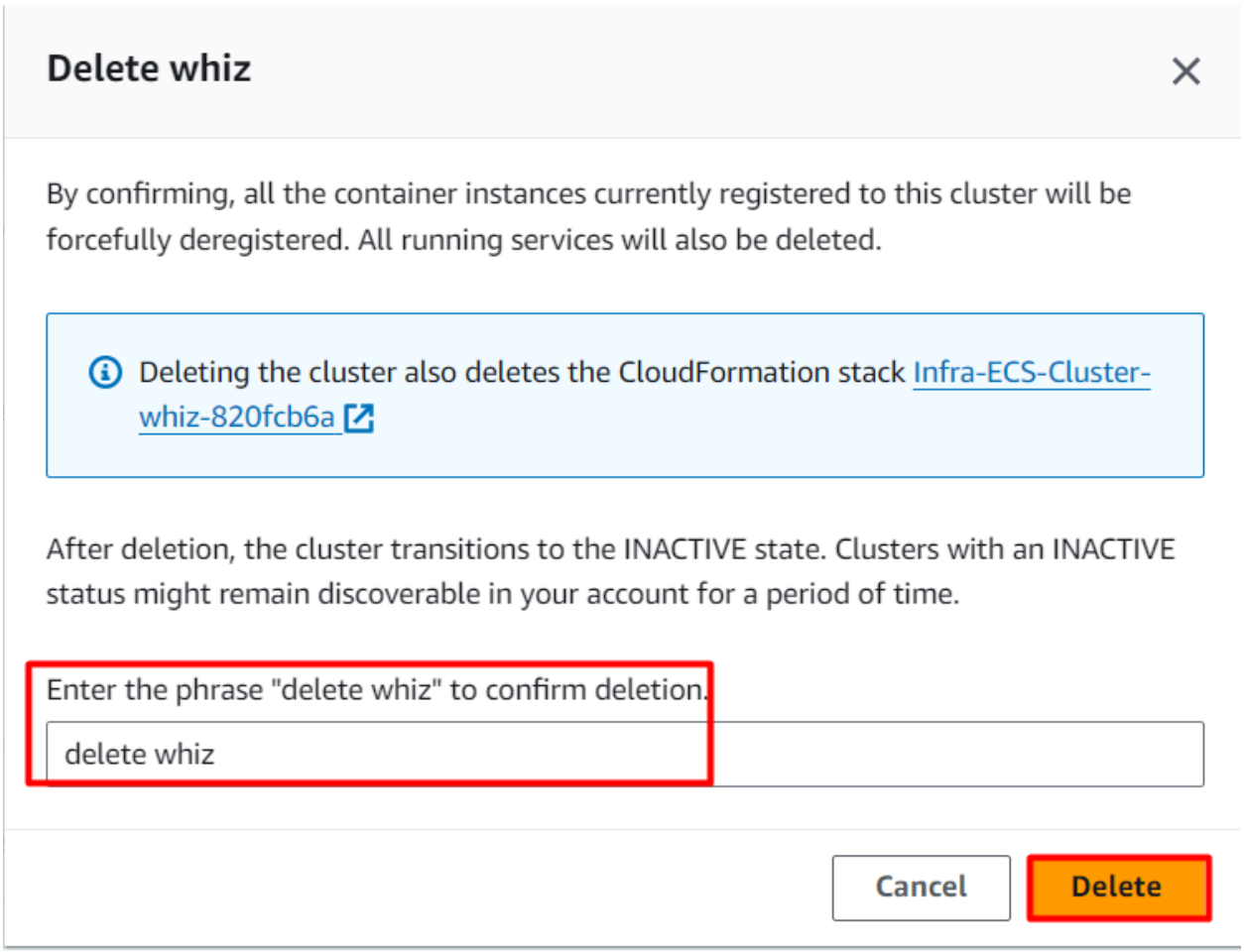
6. It will be deleted automatically.

Deleting ECS Cluster

1. Navigate to **Clusters** from the left navigation panel. Select the cluster and Click on the **Delete Cluster** button.



2. Confirm the deletion by entering the phrase ***delete whiz*** in the pop-up window.



3. Deletion will take up to 3-5 minutes.

Completion and Conclusion

1. You have successfully created and launched Amazon ECS Cluster.
2. You have successfully created an ECR repository.
3. You have successfully pushed the custom image to the ECR repository.

End Lab

1. Sign out of AWS Account.

2. You have successfully completed the lab.
3. Once you have completed the steps, click on **End Lab** from your Whizlabs lab console and wait till the process gets completed.