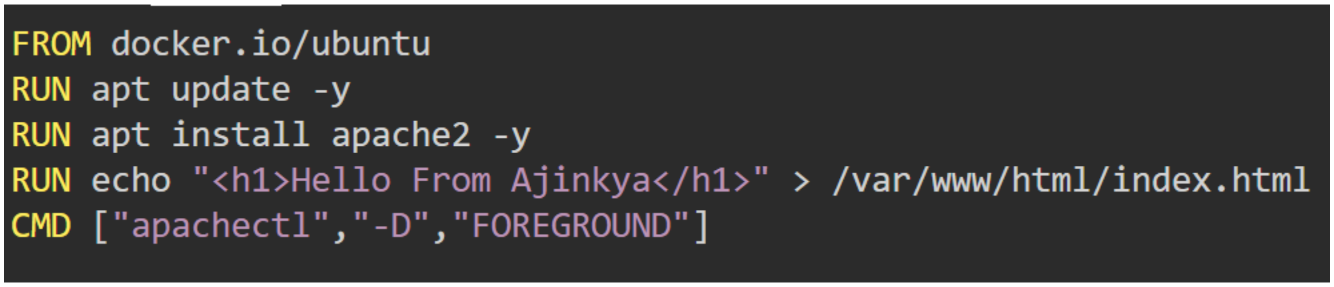
Deploying an Application on AWS ECS with ECR and Docker

Step1: Create a Containerfile

1. Create Dockerfile (Containerfile):
   - Create a **Dockerfile** with the following content.



```
FROM docker.io/ubuntu
RUN apt update -y
RUN apt install apache2 -y
RUN echo "<h1>Hello From Ajinkya</h1>" > /var/www/html/index.html
CMD ["apachectl","-D","FOREGROUND"]
```

Step 2: Configure AWS CLI

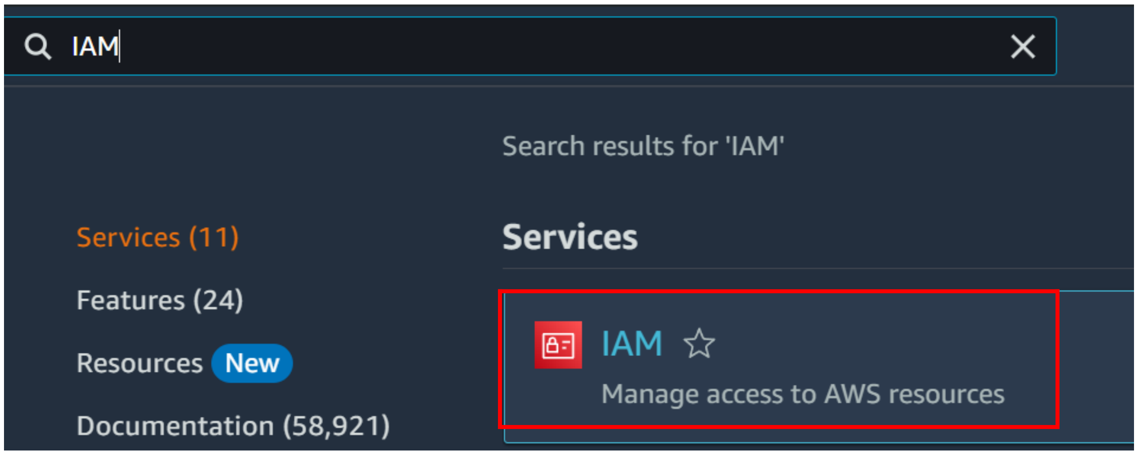
1. Install AWS CLI on KillerCoda (Ubuntu Linux):
   - **To install** the AWS CLI, run the following commands.

➢
```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

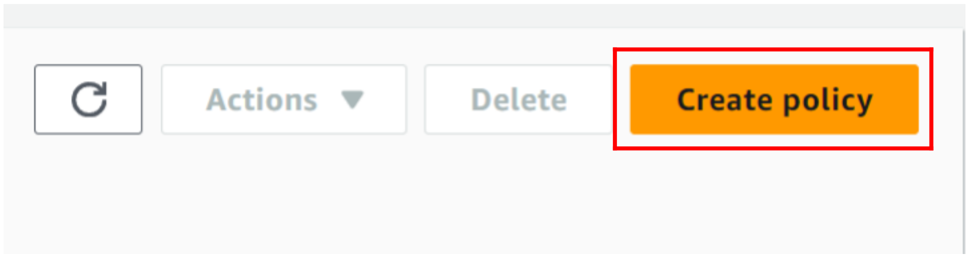
Ajinkya Kale

2. Create IAM Policy for ECR Access:

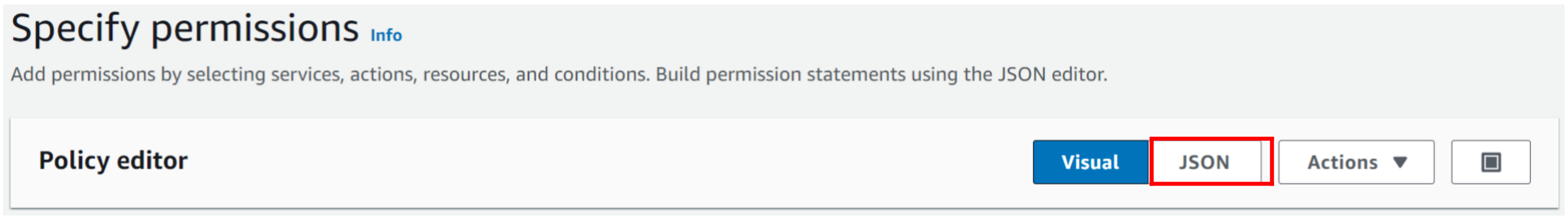
- First, create an IAM policy that allows necessary permissions for Amazon ECR.
- Go to AWS console, search for **IAM**.



- In IAM **Dashboard**, click on **Policies**.
- Click on **Create policy.**

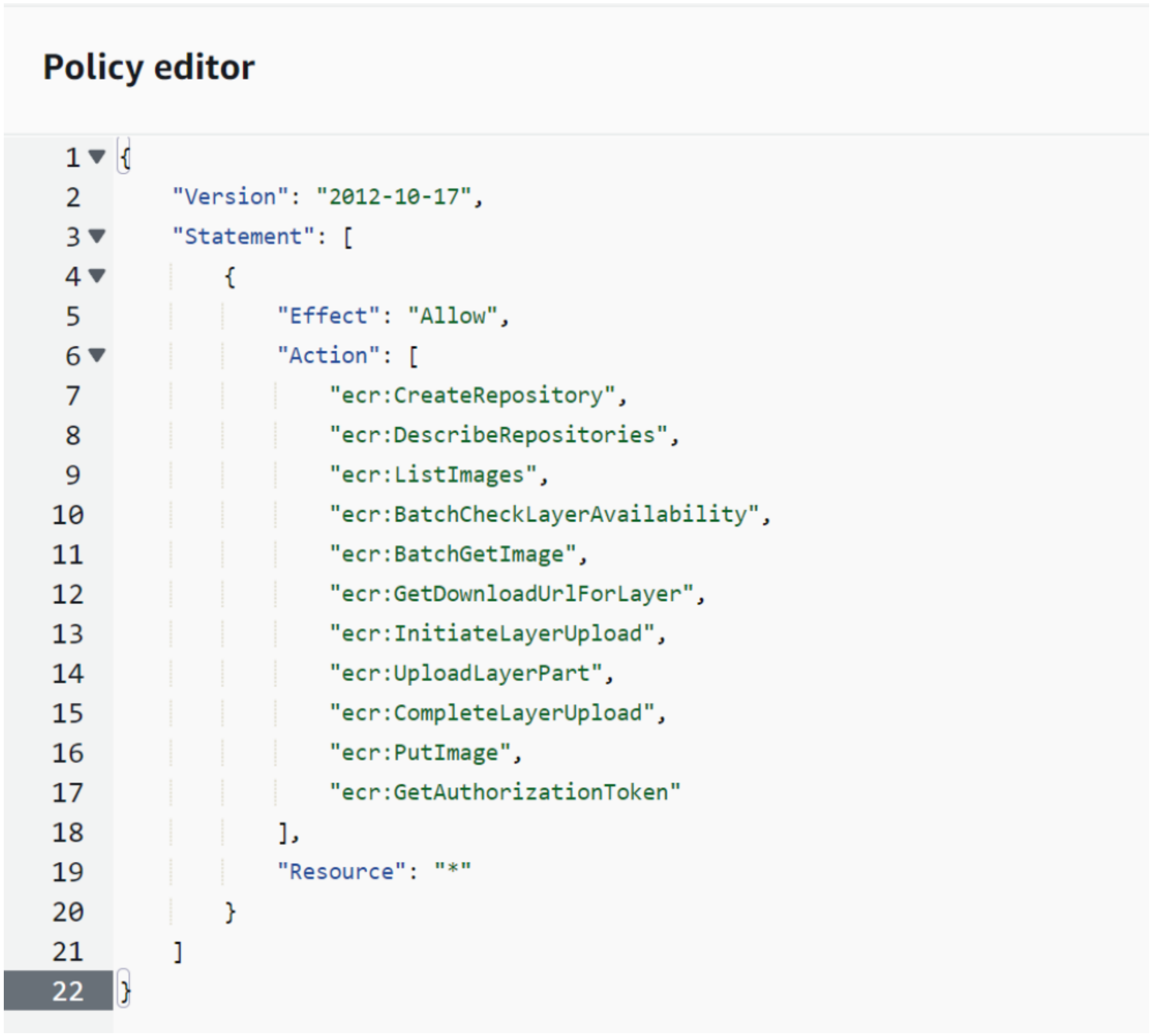


- Click on **JSON.**


Specify permissions Info

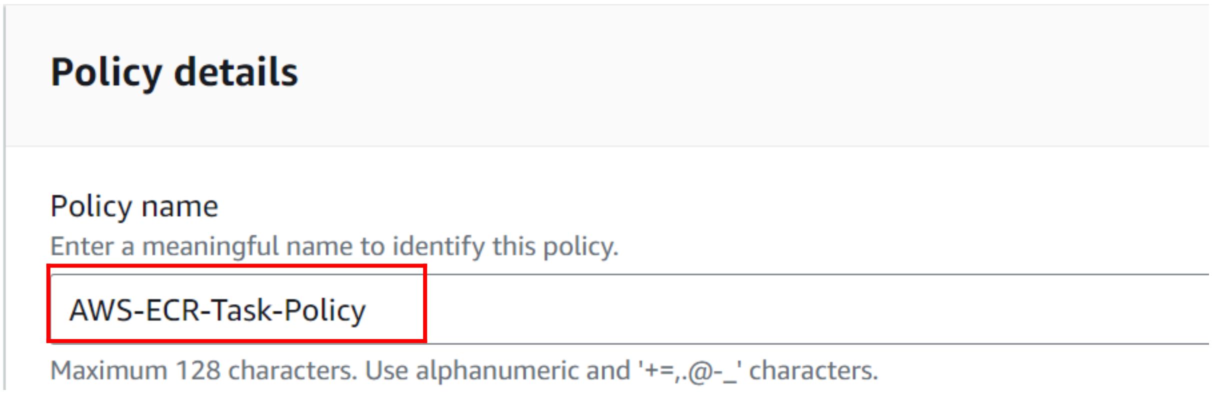
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

- Then use the following JSON code for the IAM user policy to provide Amazon ECR permissions for creating epositories and pushing images.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

- Then click on **Next**.
- Enter **name** for your policy.

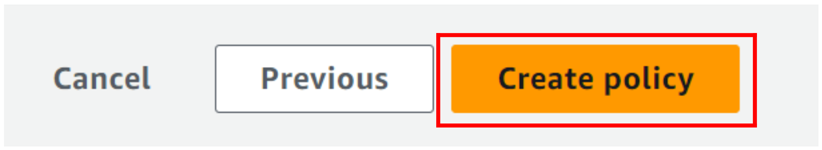


Policy name
Enter a meaningful name to identify this policy.

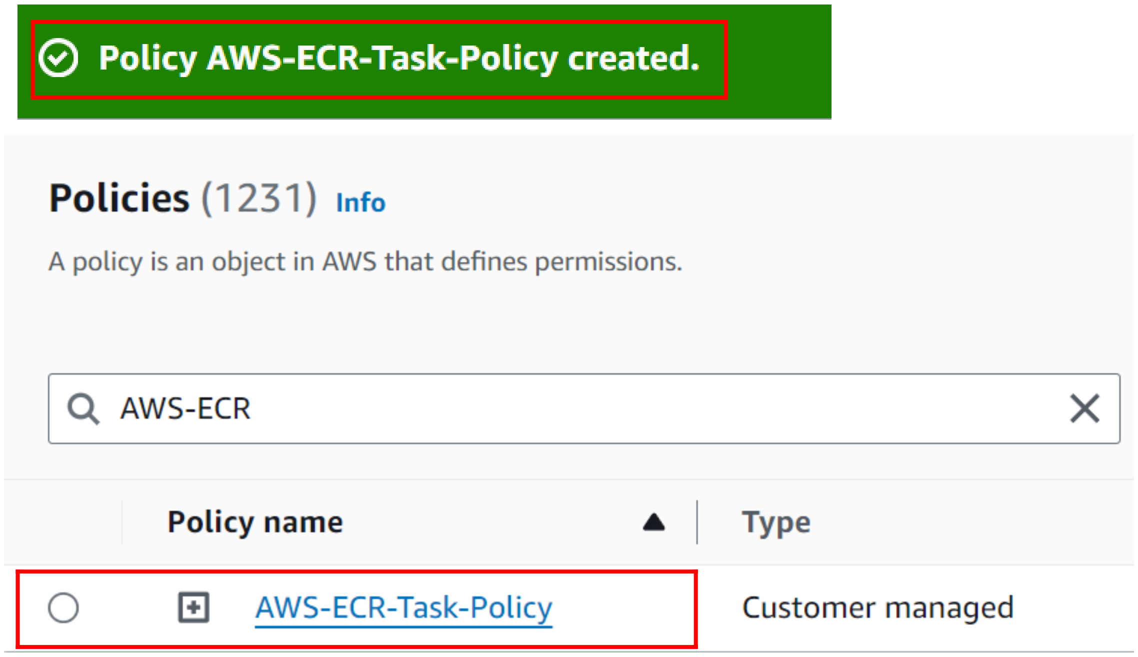
AWS-ECR-Task-Policy

Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

- Click on **Create policy**.



Policy created successfully!!



Policy AWS-ECR-Task-Policy created.

Policies (1231) Info

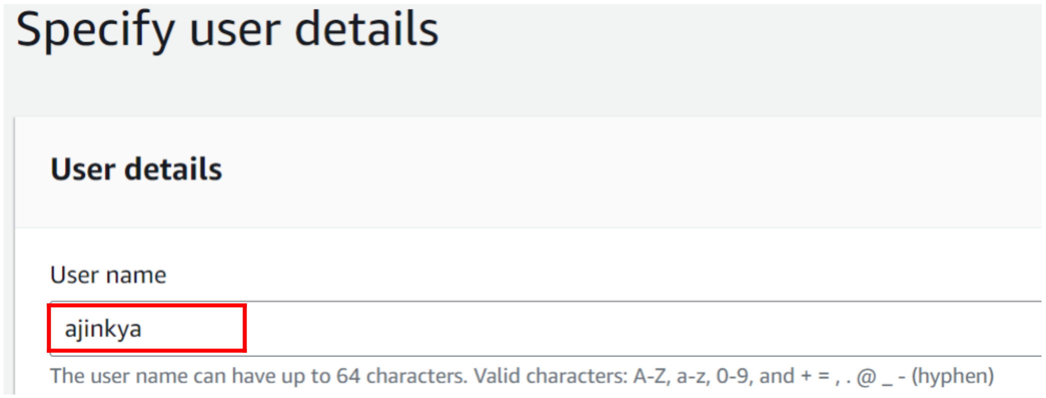
A policy is an object in AWS that defines permissions.

AWS-ECR

Policy name	Type
AWS-ECR-Task-Policy	Customer managed

3. Attach Policy to IAM User and create IAM user:

- Go to the IAM Management Console.
- Navigate to **Users** in the left-hand side.
- Click on **Create user**.
- Specify your user's name.



Specify user details

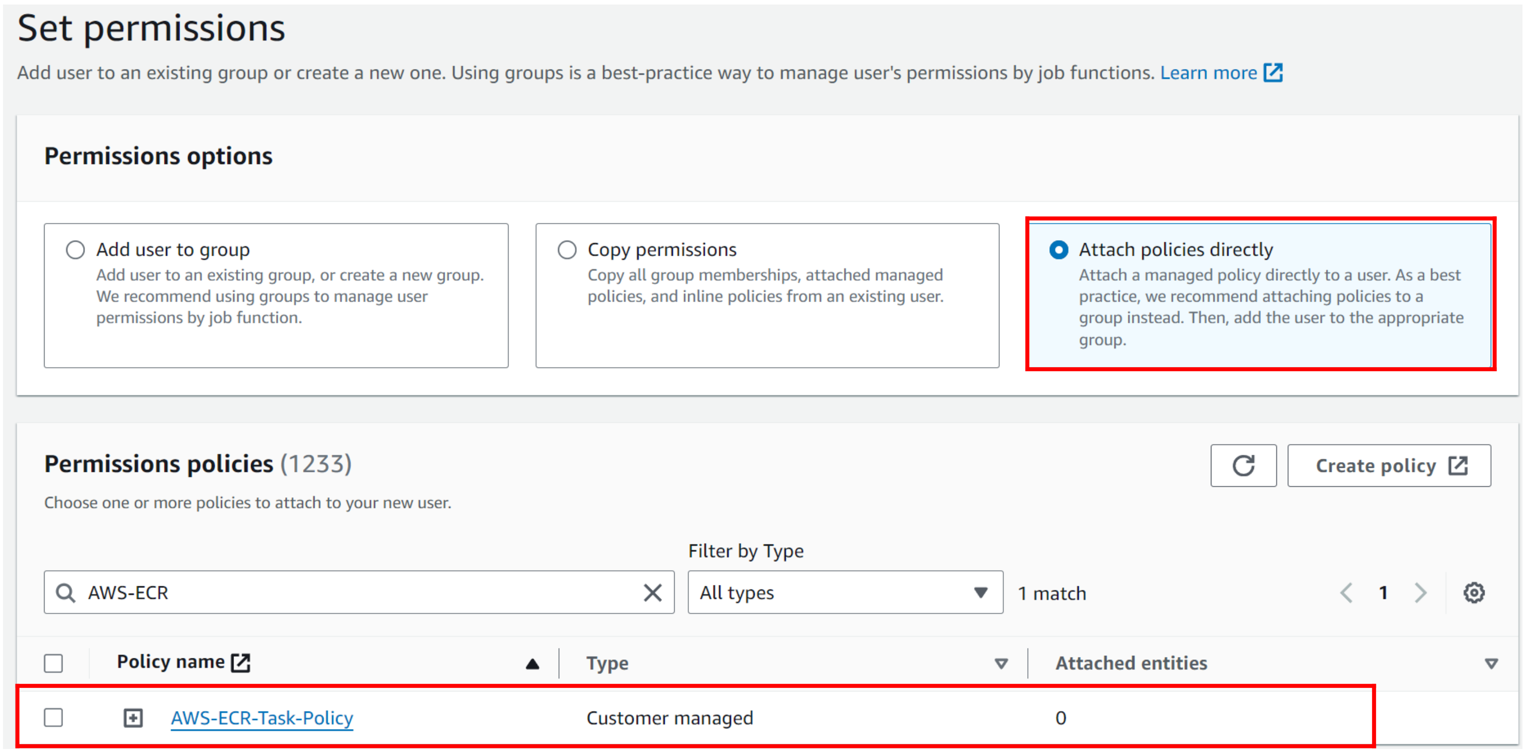
User details

User name

ajinkya

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

- Under **Set permissions**, select **Attach policies directly** and select the policy created (i.e AWS-ECR-Task-Policy).



Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Permissions options

( ) Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

( ) Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

[✓ Attach policies directly]
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1233)

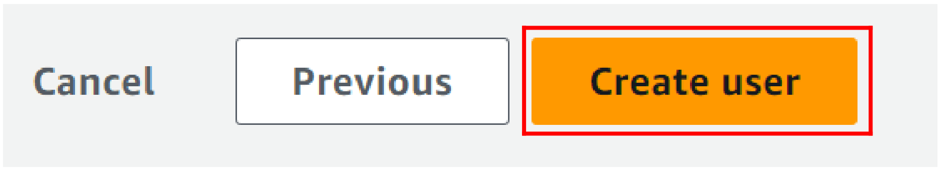
Choose one or more policies to attach to your new user.

Filter by Type

AWS-ECR | All types | 1 match

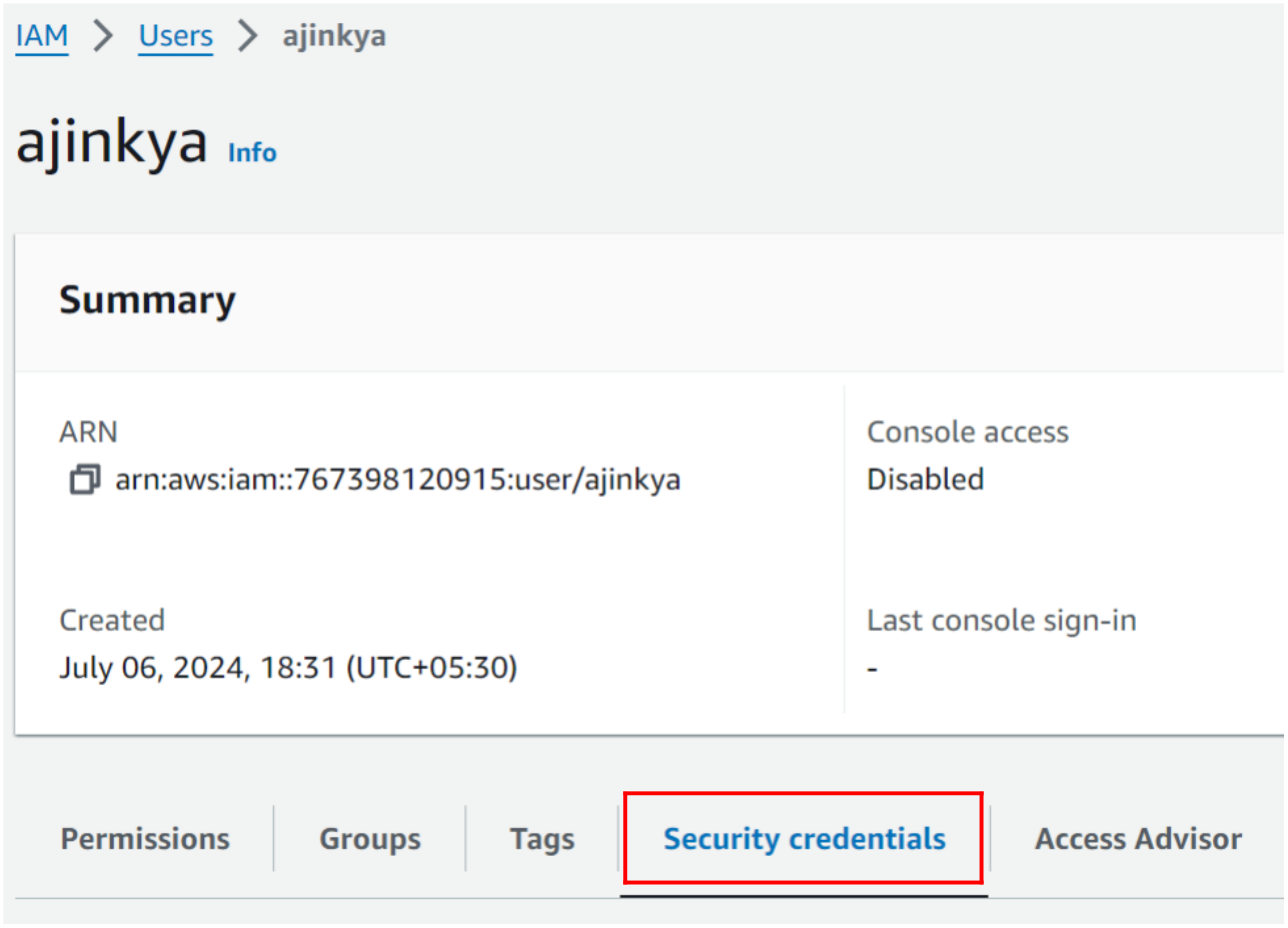
	Policy name	Type	Attached entities
☐	AWS-ECR-Task-Policy	Customer managed	0

- Then click **Next**.
- Review and create, click on **Create user**



4. Create Access Key for IAM User:

- Still on the IAM user detail page:
- Under the **"Security credentials"** tab, click **"Create access key"**.



IAM > Users > ajinkya

ajinkya Info

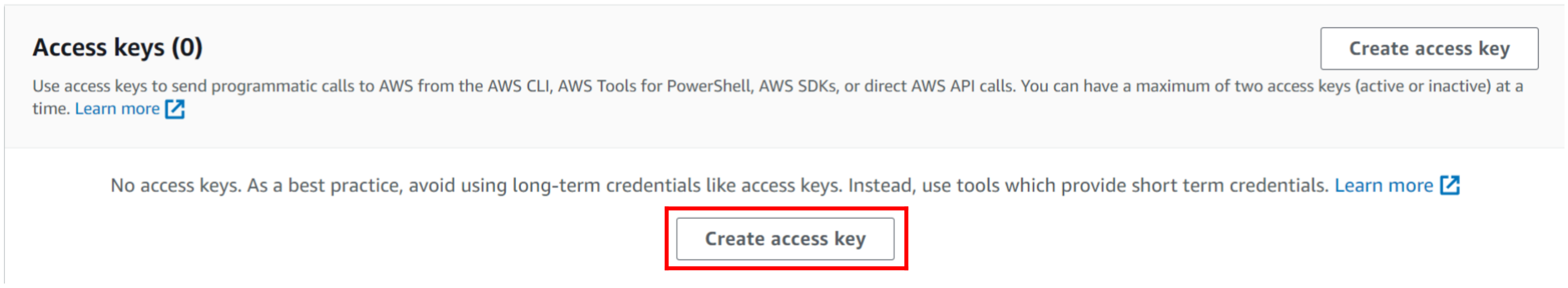
Summary

ARN
arn:aws:iam::767398120915:user/ajinkya

Console access
Disabled

Created
July 06, 2024, 18:31 (UTC+05:30)

Last console sign-in
-



Access keys (0)

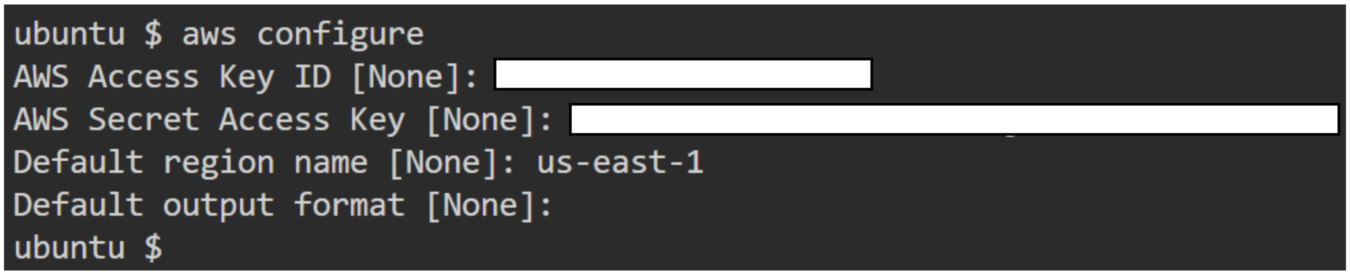
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

- Then you will see **Access Key ID and Secret Access Key**.
- Keep the Access ID and key safe.

5. Configure AWS Credentials:

- Configure AWS credentials using the **aws configure** command.
- Provide your **AWS Access Key ID**, **Secret Access Key**, **AWS Region**, and output format as **JSON**

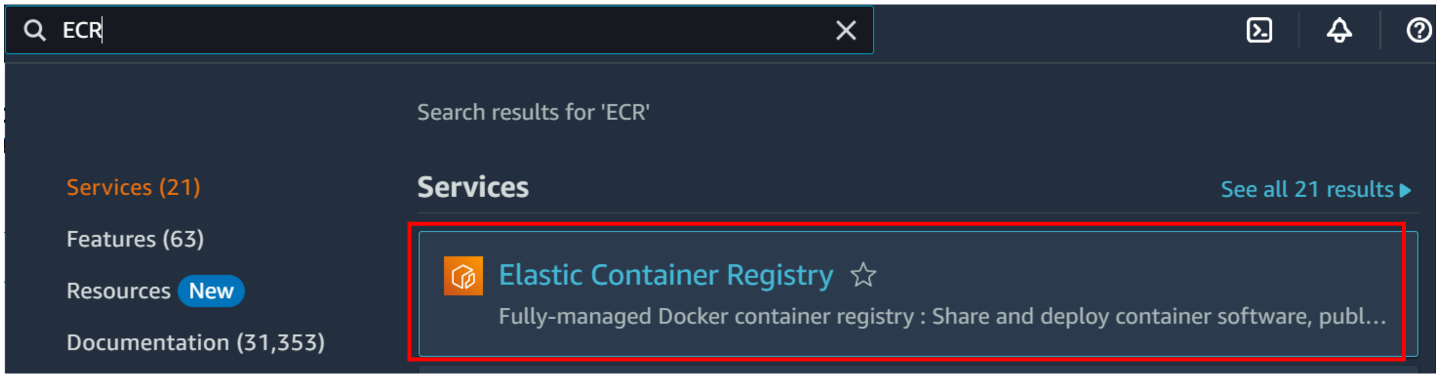


```
ubuntu $ aws configure
AWS Access Key ID [None]: [redacted]
AWS Secret Access Key [None]: [redacted]
Default region name [None]: us-east-1
Default output format [None]:
ubuntu $
```

Step 3: Create an ECR Repository

1. Navigate to **Amazon ECR**:

- Use the AWS services search bar and search for **ECR**



ECR

Search results for 'ECR'

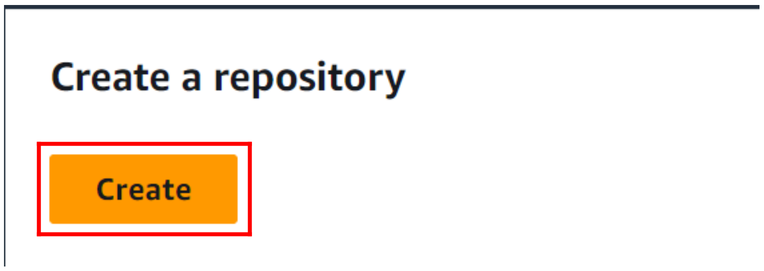
Services (21)
Features (63)
Resources New
Documentation (31,353)

Services
See all 21 results

Elastic Container Registry
Fully-managed Docker container registry : Share and deploy container software, publ...

2. Create a New Repository:

- In the Amazon ECR console, click on **Create**

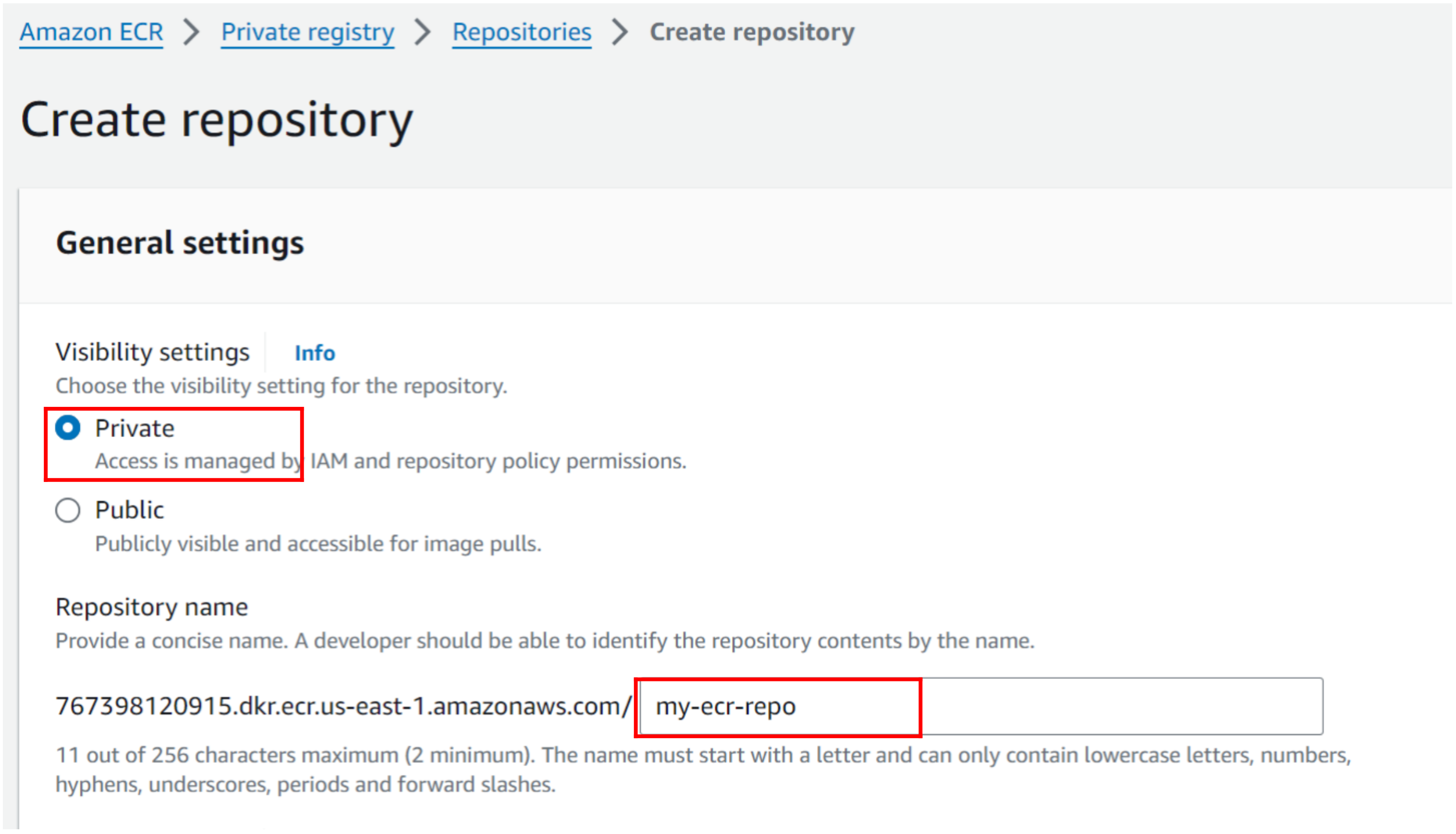


Create a repository

Create

3. Configure Repository Settings:

- Enter a unique name for your repository (e.g., **my-ecr-repo**).
- Choose visibility settings **(Private)**


Amazon ECR > Private registry > Repositories > Create repository

Create repository

General settings

Visibility settings Info
Choose the visibility setting for the repository.

[✓ Private] Access is managed by IAM and repository policy permissions.

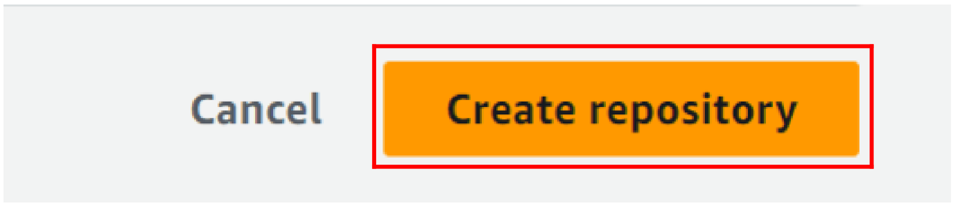
Public Publicly visible and accessible for image pulls.

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

767398120915.dkr.ecr.us-east-1.amazonaws.com/ my-ecr-repo

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

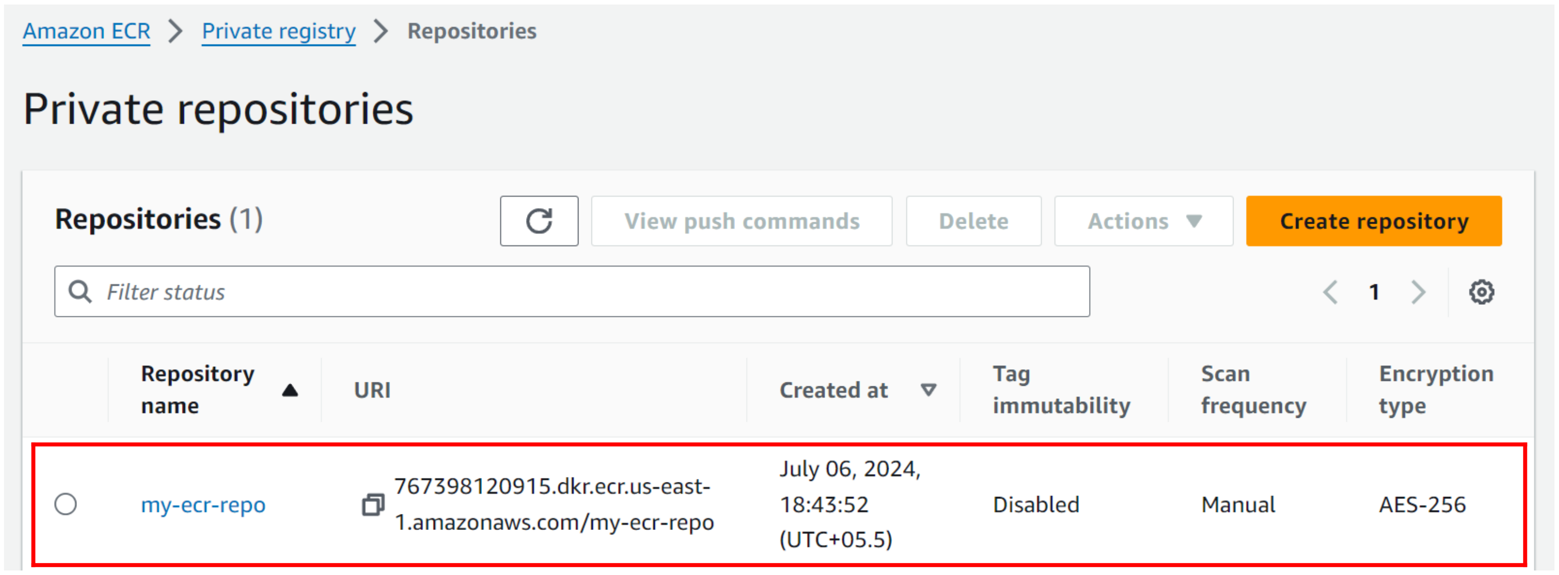
- click the **Create repository** button.



Ajinkya Kale

4. Repository Created:

- **repository has been created successfully!!**



Amazon ECR > Private registry > Repositories

Private repositories

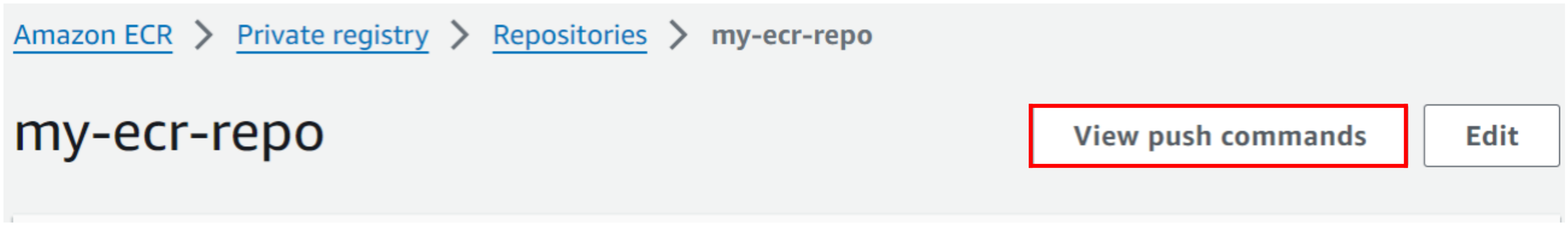
Repositories (1)

	Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type
○	my-ecr-repo	767398120915.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repo	July 06, 2024, 18:43:52 (UTC+05.5)	Disabled	Manual	AES-256

Step5: Push Docker Image to ECR

1. Push commands for **my-ecr-repo**:

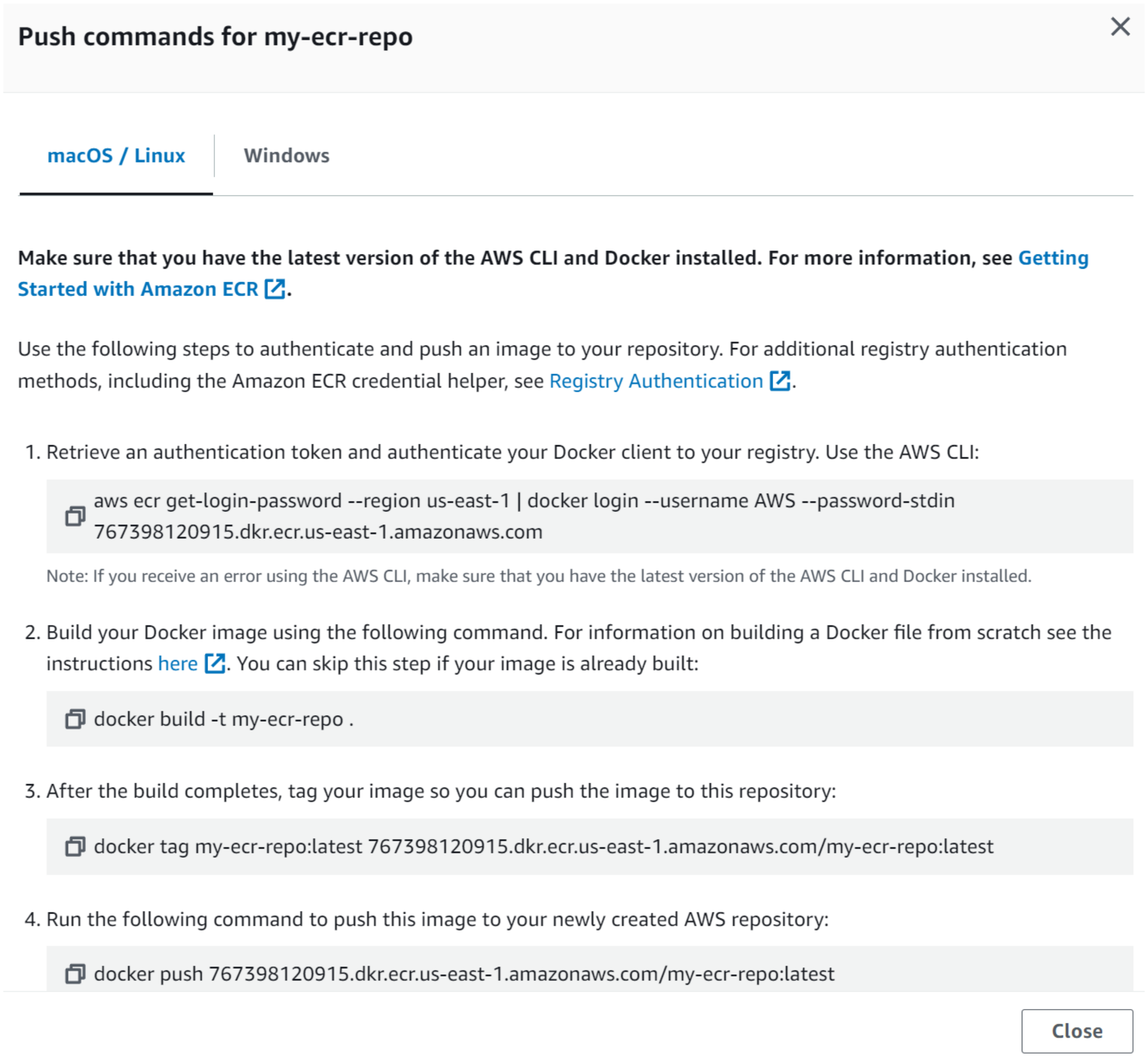
- Click on **Repository name.**
- Then Click on **"View push commands"**.



Amazon ECR > Private registry > Repositories > my-ecr-repo

my-ecr-repo

- **By following below steps, you can successfully push your Docker image to Amazon ECR and make it available for use in ECS**
- Run the following commands one by one.



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-stdin
767398120915.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 my-ecr-repo .
```

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

```
docker tag my-ecr-repo:latest 767398120915.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repo:latest
```

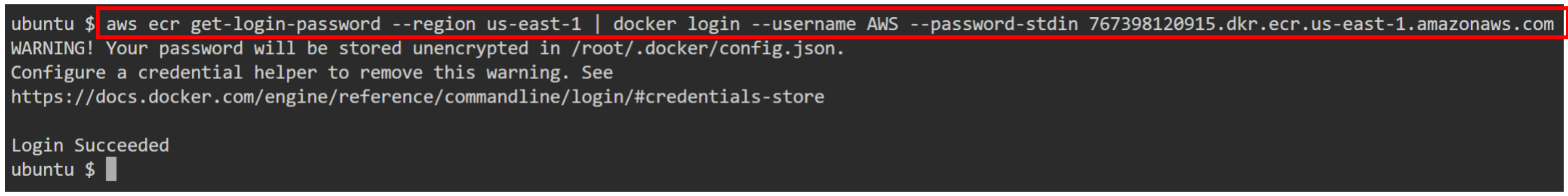
4. Run the following command to push this image to your newly created AWS repository:

```
docker push 767398120915.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repo:latest
```

Ajinkya Kale

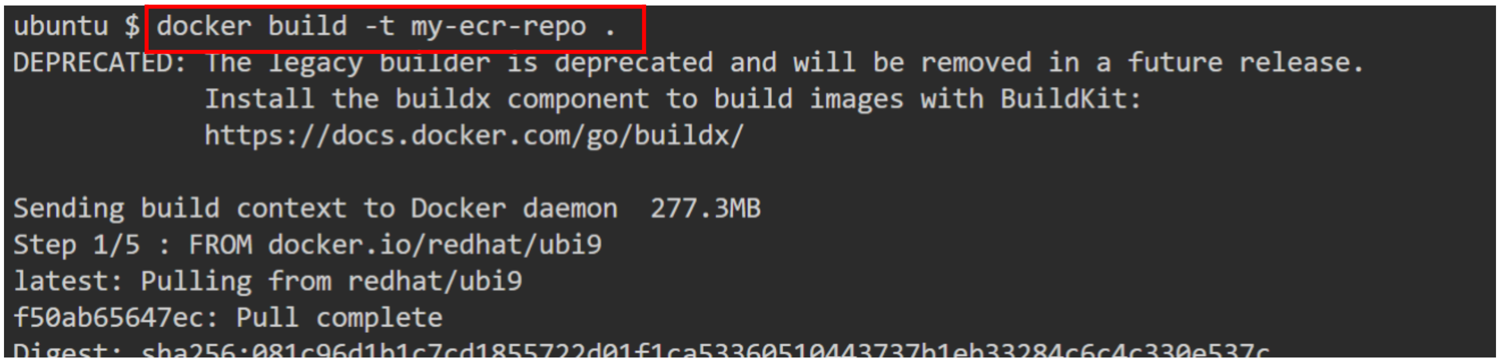
2. Push command for my-ecr-repo:

1. Authenticate Docker to ECR



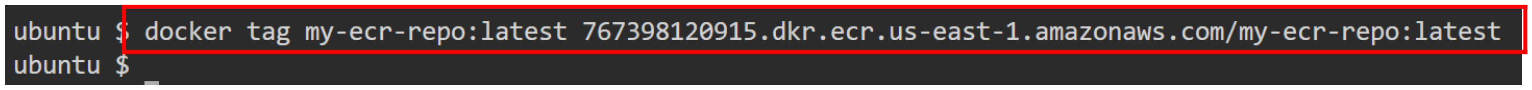
```
ubuntu $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 767398120915.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu $
```

2. Build Docker Image



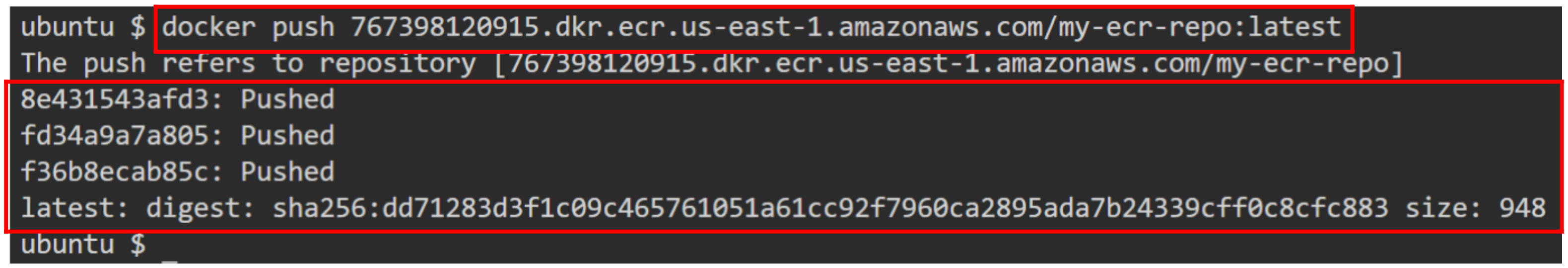
```
ubuntu $ docker build -t my-ecr-repo .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  277.3MB
Step 1/5 : FROM docker.io/redhat/ubi9
latest: Pulling from redhat/ubi9
f50ab65647ec: Pull complete
Digest: sha256:081c96d1b1c7cd1855722d01f1ca53360510443737b1eb33284c6c4c330e537c
```

3. Tag Docker Image



```
ubuntu $ docker tag my-ecr-repo:latest 767398120915.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repo:latest
ubuntu $
```

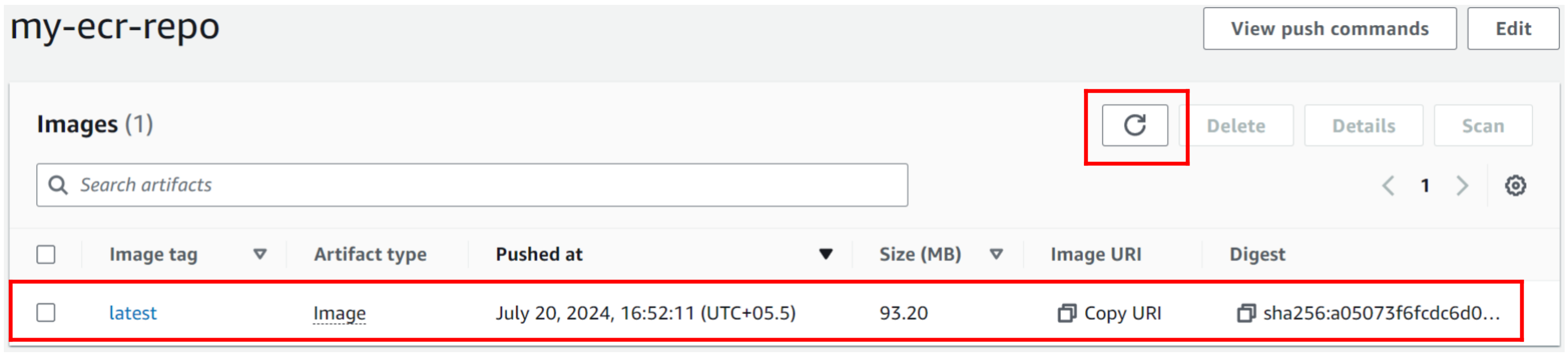
4. Push Docker Image to ECR



```
ubuntu $ docker push 767398120915.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repo:latest
The push refers to repository [767398120915.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repo]
8e431543afd3: Pushed
fd34a9a7a805: Pushed
f36b8ecab85c: Pushed
latest: digest: sha256:dd71283d3f1c09c465761051a61cc92f7960ca2895ada7b24339cff0c8cfc883 size: 948
ubuntu $
```

3. List Images in ECR Repository:

- Click on the refresh button to verify that the Docker image has been uploaded to the ECR repository .



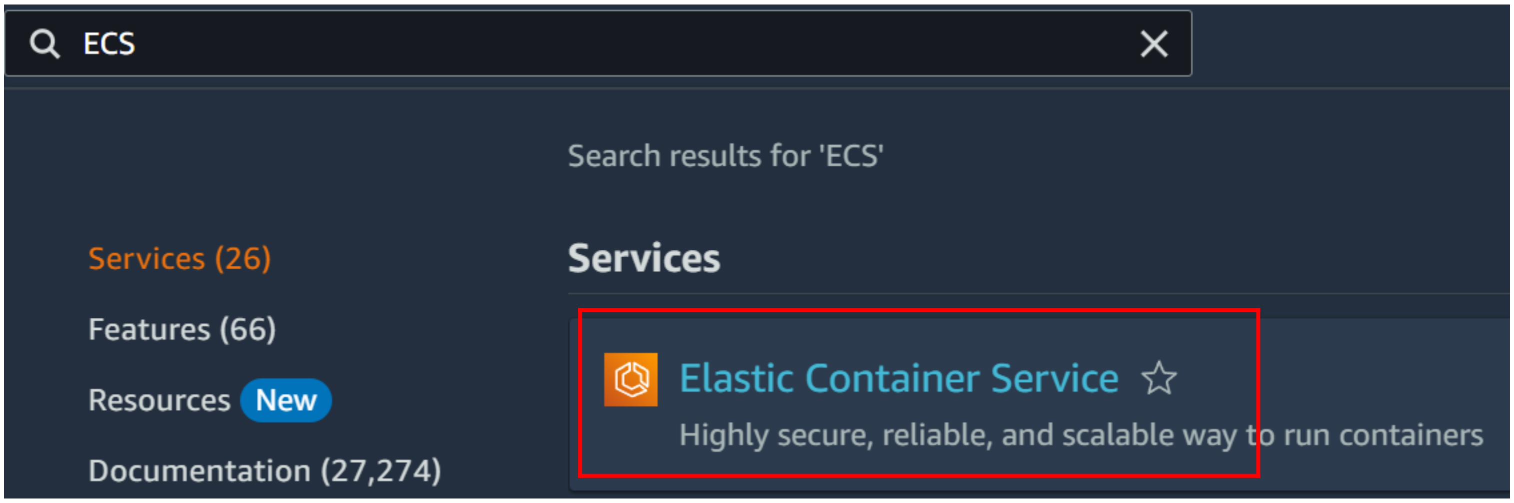
my-ecr-repo

Images (1)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	July 20, 2024, 16:52:11 (UTC+05.5)	93.20	Copy URI	sha256:a05073f6fcdc6d0...

Step 4: Create ECS

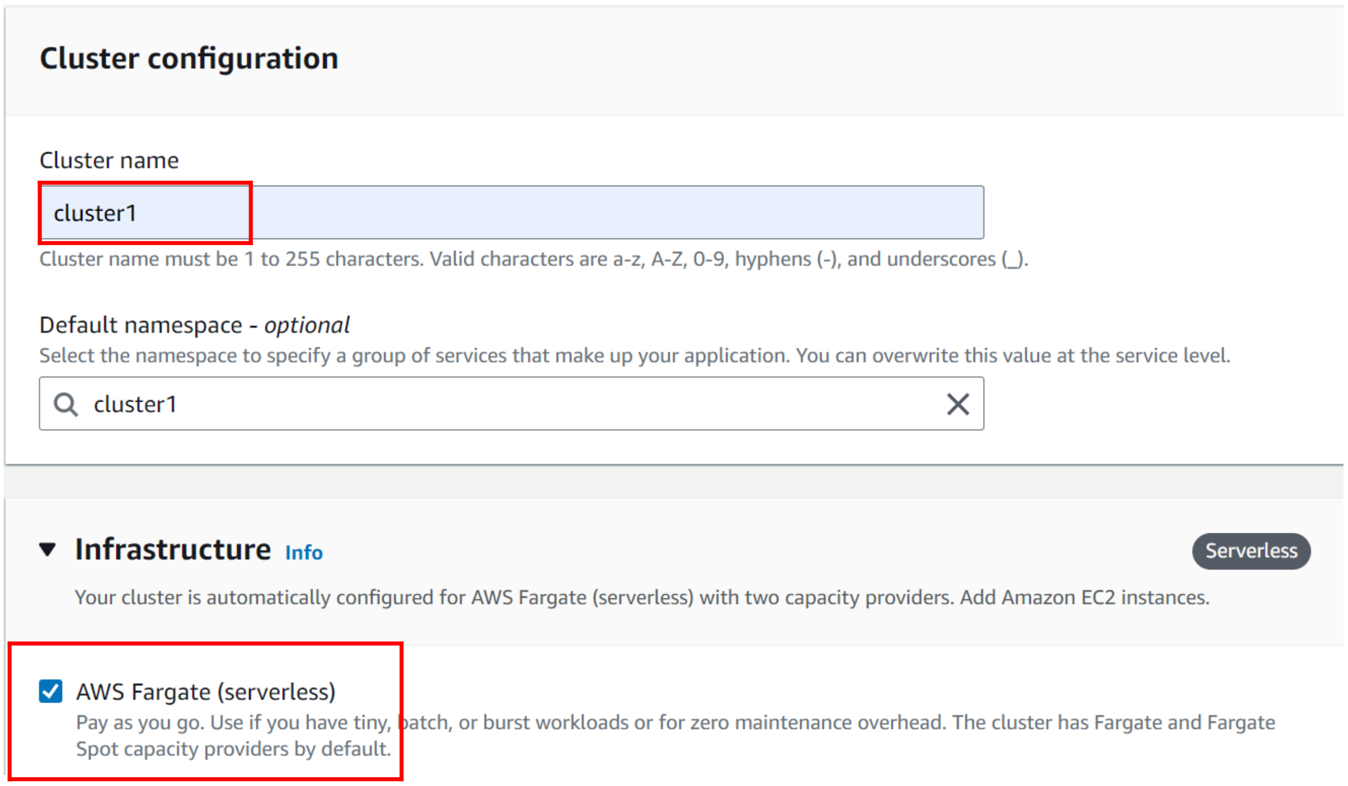
- Go to the AWS Management Console and search for ECS.



Ajinkya Kale

1. Create **ECS Cluster**:

- Enter name for your cluster
- Under the **Infrastructure**, choose **"AWS Fargate"**.
- Click on **Create**.zz



Cluster configuration

Cluster name

cluster1

Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Default namespace - *optional*

Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.

cluster1

Infrastructure Info

Serverless

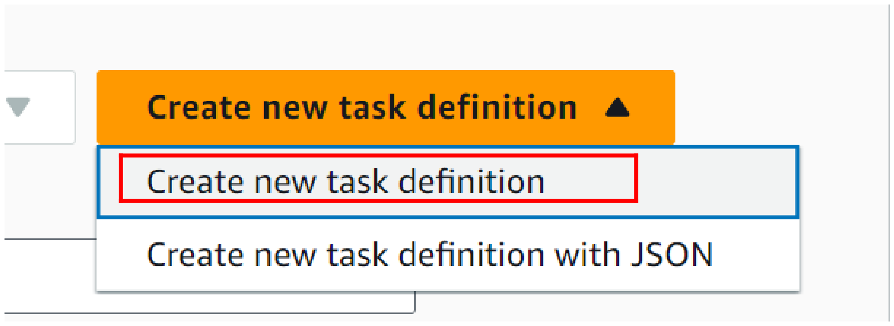
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances.

☑ AWS Fargate (serverless)

Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.

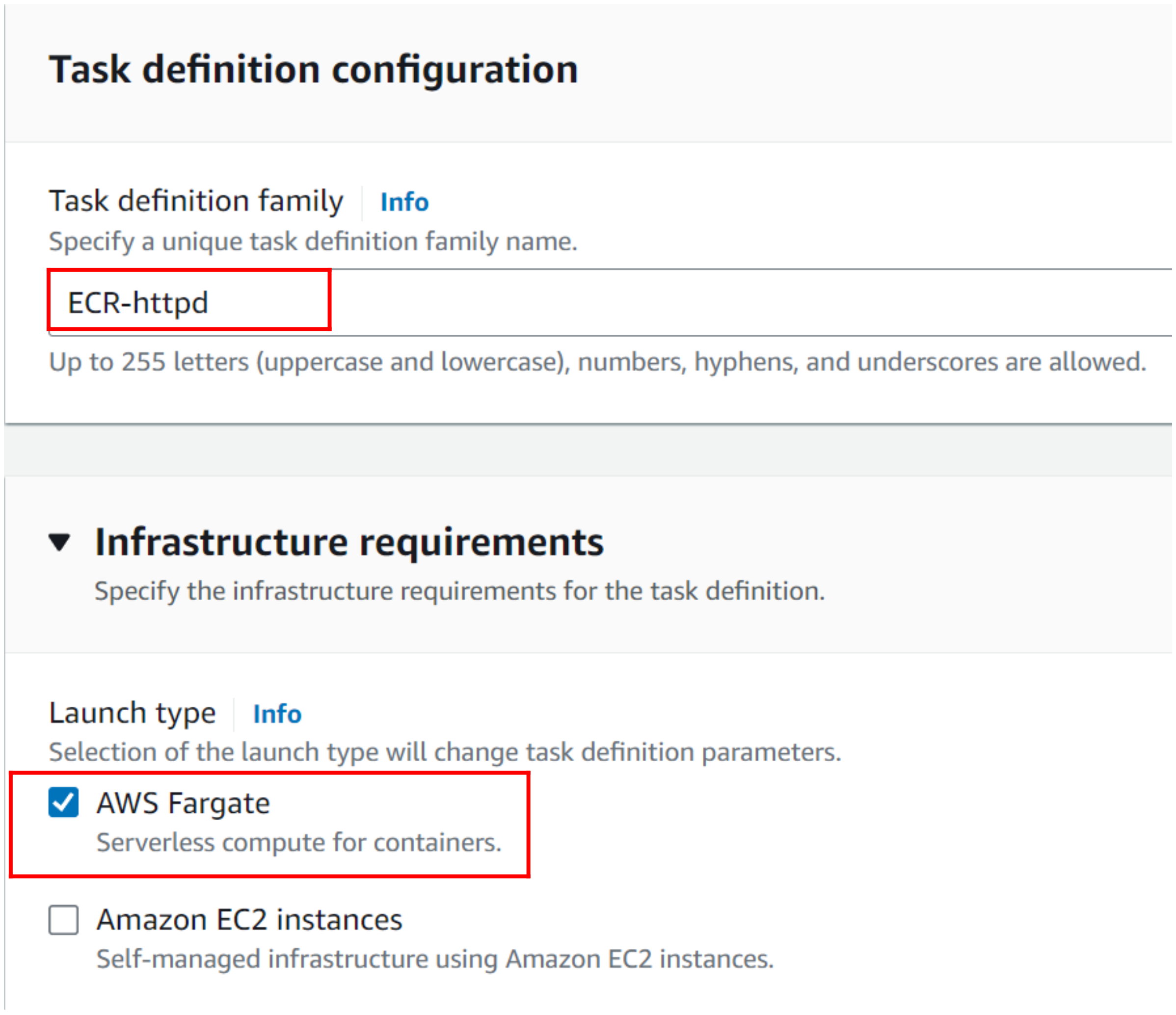
2. Create **Task Definition**:

- Click on **Create new task definition**.


Create new task definition
Create new task definition
Create new task definition with JSON

Ajinkya Kale

- Under **task definition family** enter name for your task.
- Choose **FARGATE** launch type.



Task definition configuration

Task definition family | Info

Specify a unique task definition family name.

ECR-httpd

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Infrastructure requirements

Specify the infrastructure requirements for the task definition.

Launch type | Info

Selection of the launch type will change task definition parameters.

☑ AWS Fargate

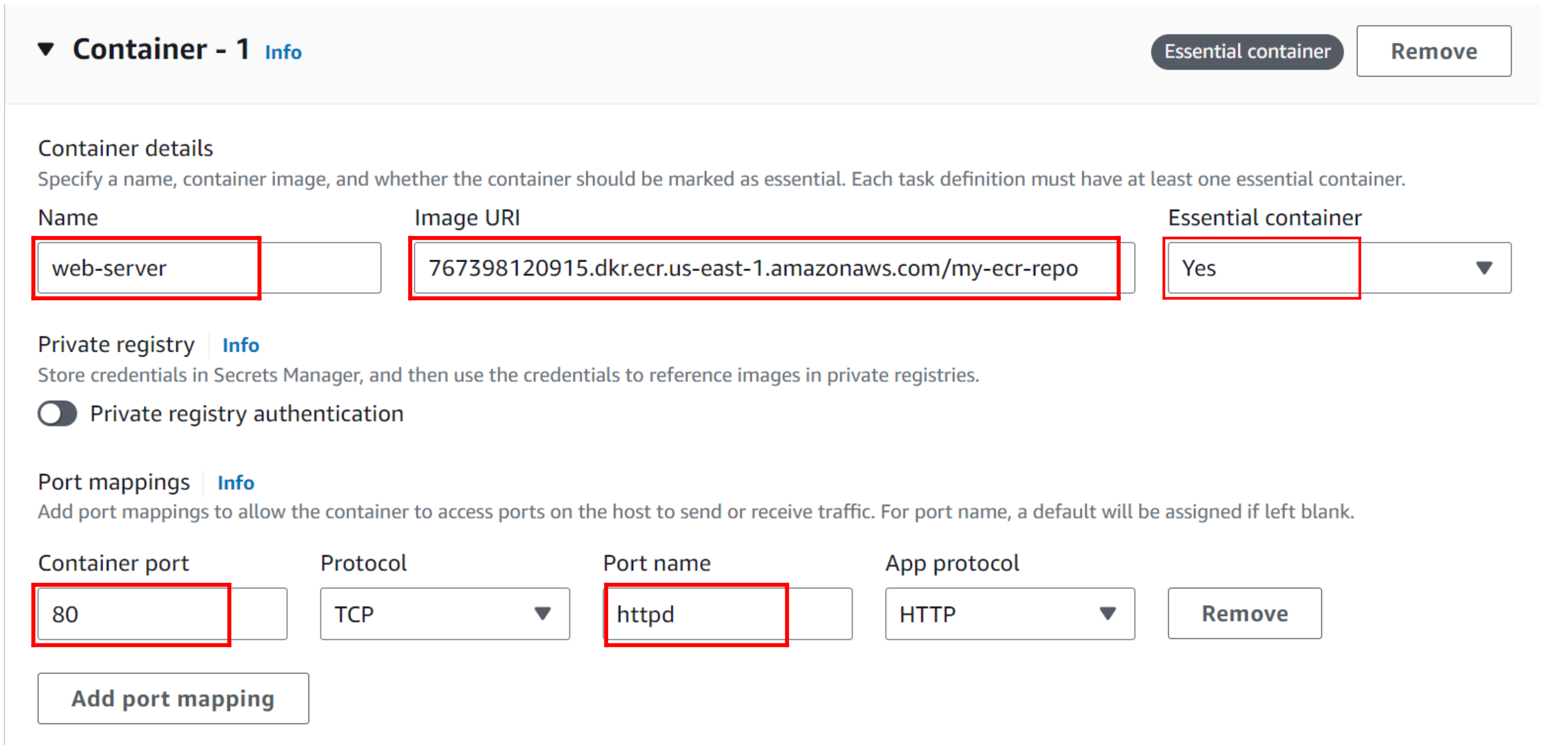
Serverless compute for containers.

☐ Amazon EC2 instances

Self-managed infrastructure using Amazon EC2 instances.

3. Container:

- **Name of container** (web-server)
- **Image URL**: Copy the URI from the Repository that we created earlier
- **Essential Container** (Yes)
- **Port Mapping** Container (Port 80),
- **Port Name** (httpd)


Container - 1 Info

Essential container

Remove

Container details

Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name	Image URI	Essential container
web-server	767398120915.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repo	Yes

Private registry Info

Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

Private registry authentication

Port mappings Info

Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
80	TCP	httpd	HTTP

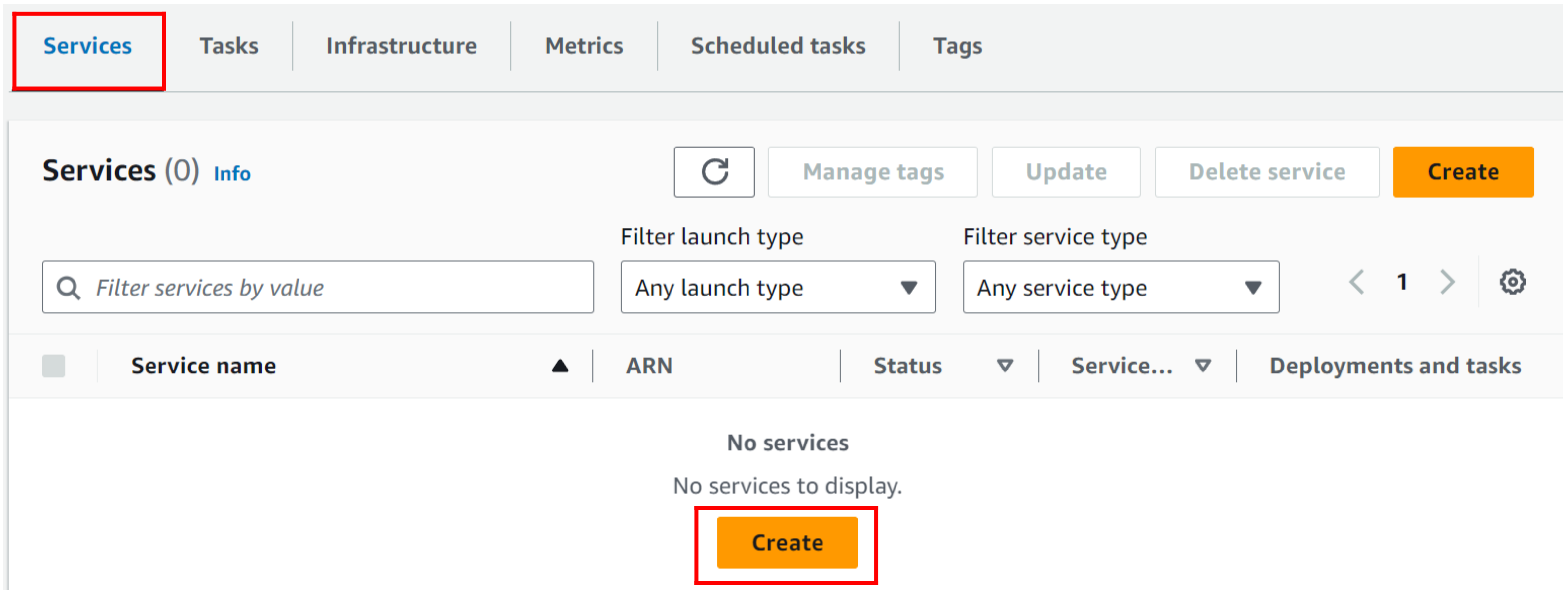
Remove

Add port mapping

- Then click on **Create**

4. Creating ECS Service:

- Go back to the cluster we created.
- Scroll down and click **Create** under **Services**.



Services (0) Info

Filter launch type: Any launch type

Filter service type: Any service type

Filter services by value

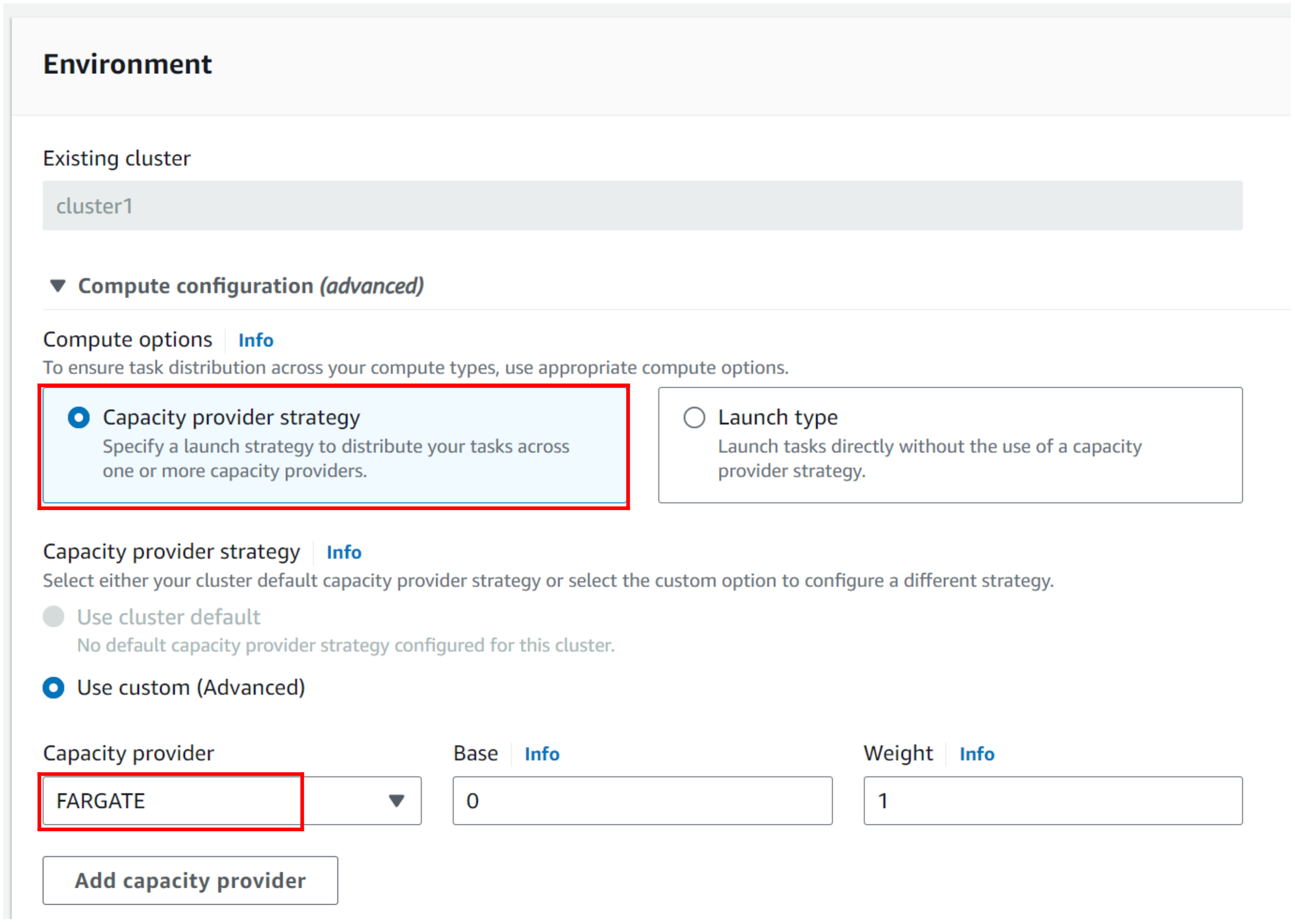
Service name	ARN	Status	Service...	Deployments and tasks

No services

No services to display.

Create

- Under the Compute options menu. Select **Capacity provider strategy.**
- Select **FARGATE** as the capacity provider.



Environment

Existing cluster

cluster1

▼ Compute configuration *(advanced)*

Compute options Info

To ensure task distribution across your compute types, use appropriate compute options.

[✓ Capacity provider strategy] Specify a launch strategy to distribute your tasks across one or more capacity providers.

Launch type — Launch tasks directly without the use of a capacity provider strategy.

Capacity provider strategy Info

Select either your cluster default capacity provider strategy or select the custom option to configure a different strategy.

Use cluster default — No default capacity provider strategy configured for this cluster.

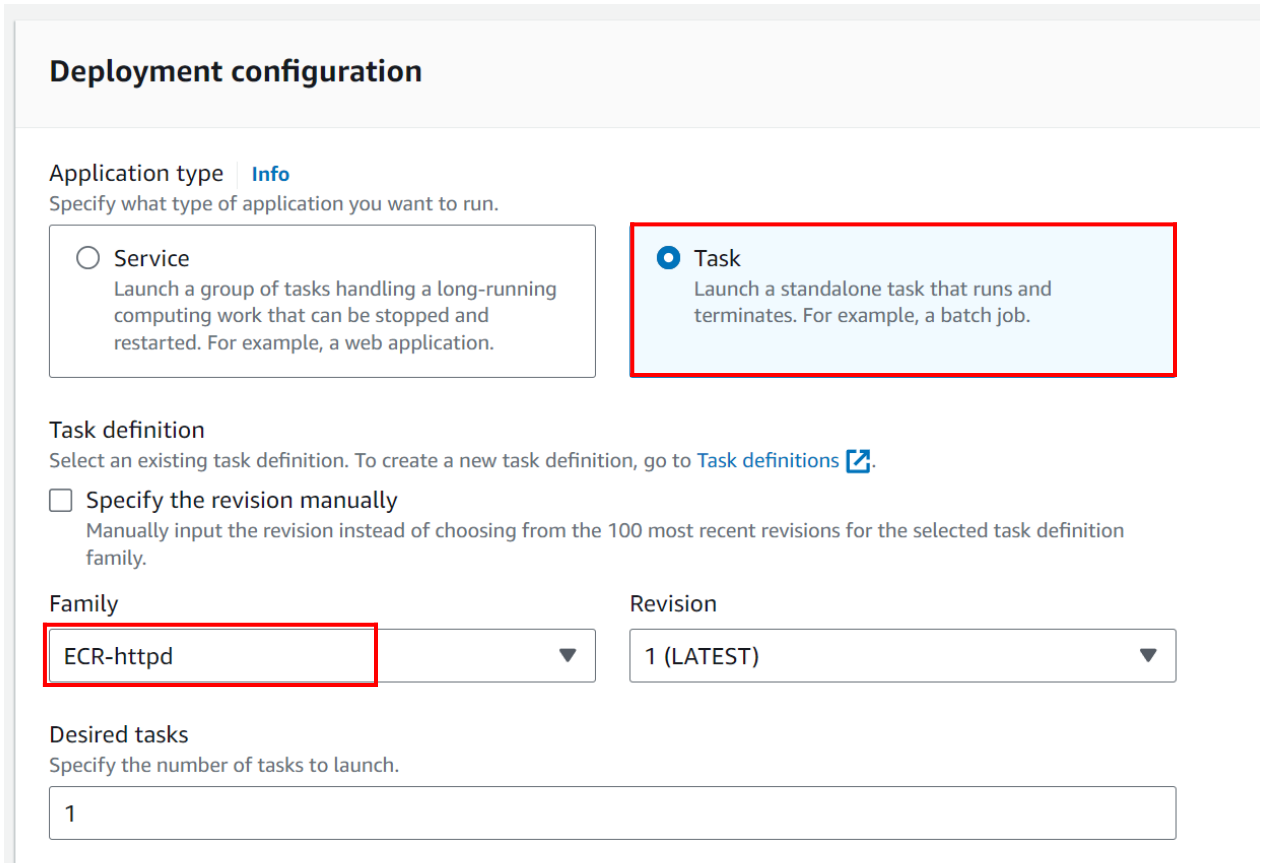
[✓ Use custom (Advanced)]

Capacity provider	Base	Weight
FARGATE	0	1

Add capacity provider

- Under Deployment configuration, choose **Task**.
- In **Task definition** Select the created task definition, **(i.e., ECR-httpd)**

Ajinkya Kale



Deployment configuration

Application type | Info

Specify what type of application you want to run.

( ) Service
Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

(●) Task
Launch a standalone task that runs and terminates. For example, a batch job.

Task definition

Select an existing task definition. To create a new task definition, go to Task definitions.

[ ] Specify the revision manually
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

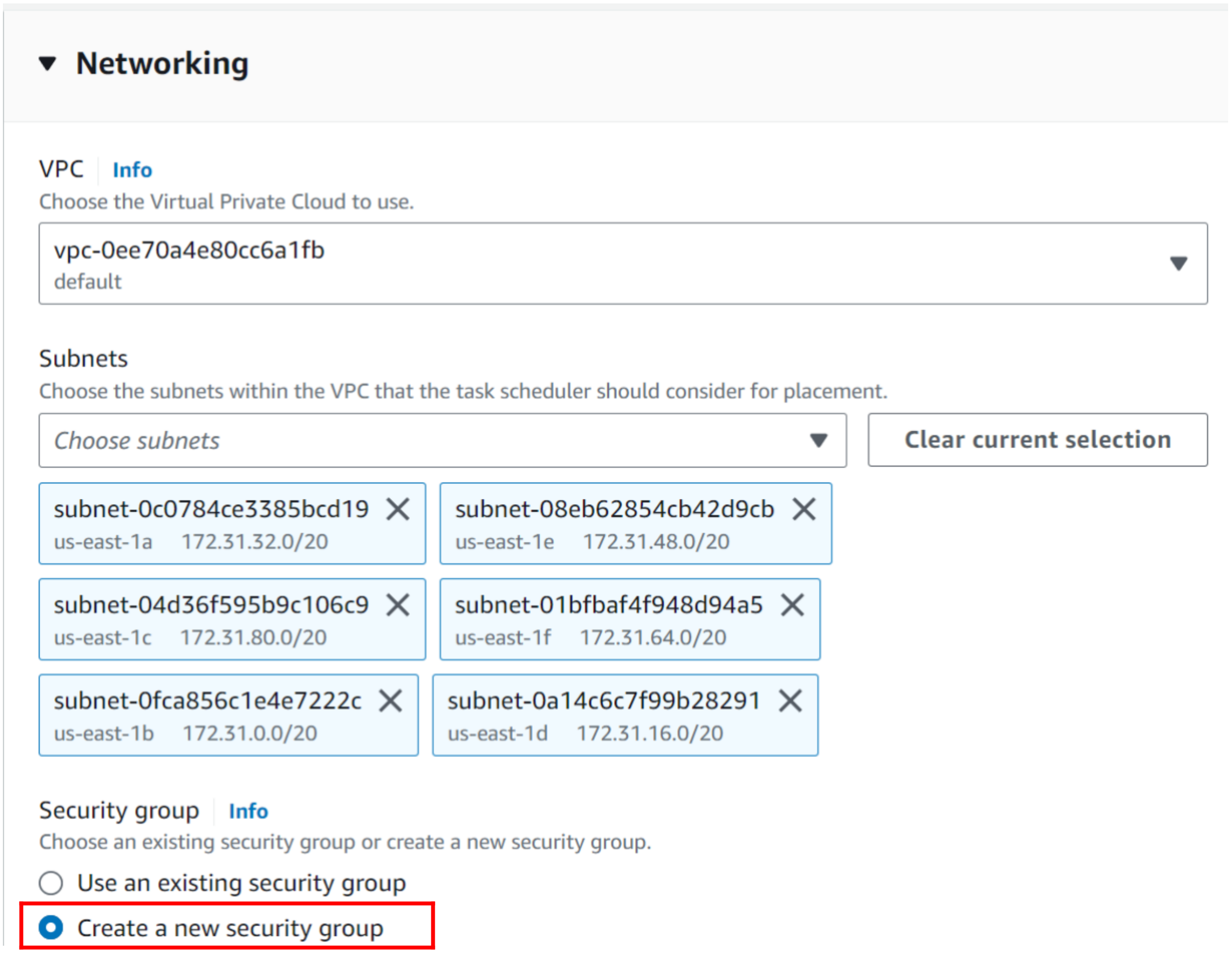
Family	Revision
ECR-httpd	1 (LATEST)

Desired tasks

Specify the number of tasks to launch.

1

- Under **Networking**, click on **Create new security group**



Networking

VPC | Info

Choose the Virtual Private Cloud to use.

vpc-0ee70a4e80cc6a1fb
default

Subnets

Choose the subnets within the VPC that the task scheduler should consider for placement.

Choose subnets

Clear current selection

subnet-0c0784ce3385bcd19	us-east-1a	172.31.32.0/20
subnet-08eb62854cb42d9cb	us-east-1e	172.31.48.0/20
subnet-04d36f595b9c106c9	us-east-1c	172.31.80.0/20
subnet-01bfbaf4f948d94a5	us-east-1f	172.31.64.0/20
subnet-0fca856c1e4e7222c	us-east-1b	172.31.0.0/20
subnet-0a14c6c7f99b28291	us-east-1d	172.31.16.0/20

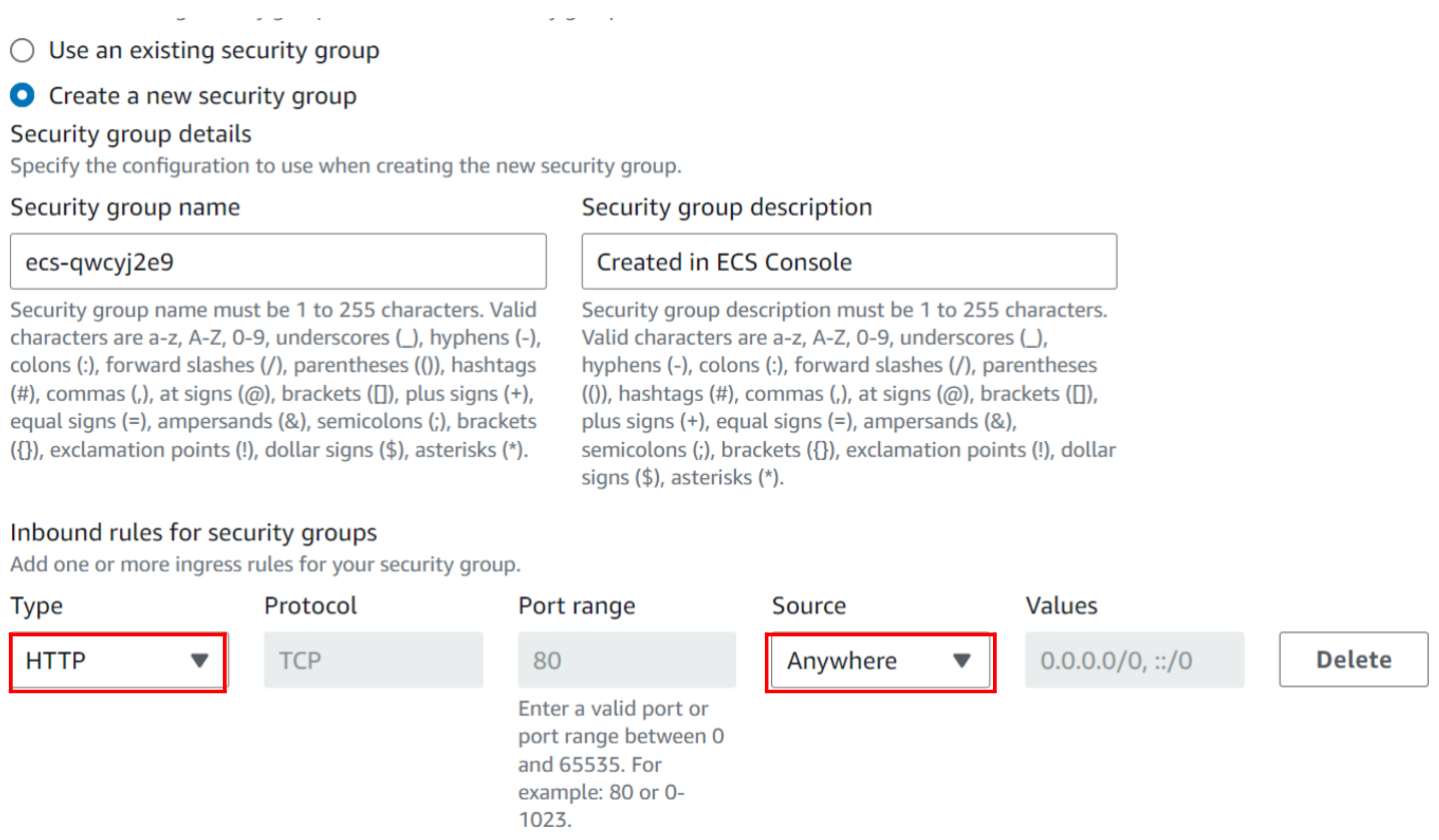
Security group | Info

Choose an existing security group or create a new security group.

( ) Use an existing security group

(●) Create a new security group

- Create a new security group with inbound rule for **HTTP (80)**



Use an existing security group

[✓ Create a new security group]

Security group details

Specify the configuration to use when creating the new security group.

Security group name: ecs-qwcyj2e9

Security group name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, underscores (_), hyphens (-), colons (:), forward slashes (/), parentheses (()), hashtags (#), commas (,), at signs (@), brackets ([]), plus signs (+), equal signs (=), ampersands (&), semicolons (;), brackets ({}), exclamation points (!), dollar signs ($), asterisks (*).

Security group description: Created in ECS Console

Security group description must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, underscores (_), hyphens (-), colons (:), forward slashes (/), parentheses (()), hashtags (#), commas (,), at signs (@), brackets ([]), plus signs (+), equal signs (=), ampersands (&), semicolons (;), brackets ({}), exclamation points (!), dollar signs ($), asterisks (*).

Inbound rules for security groups

Add one or more ingress rules for your security group.

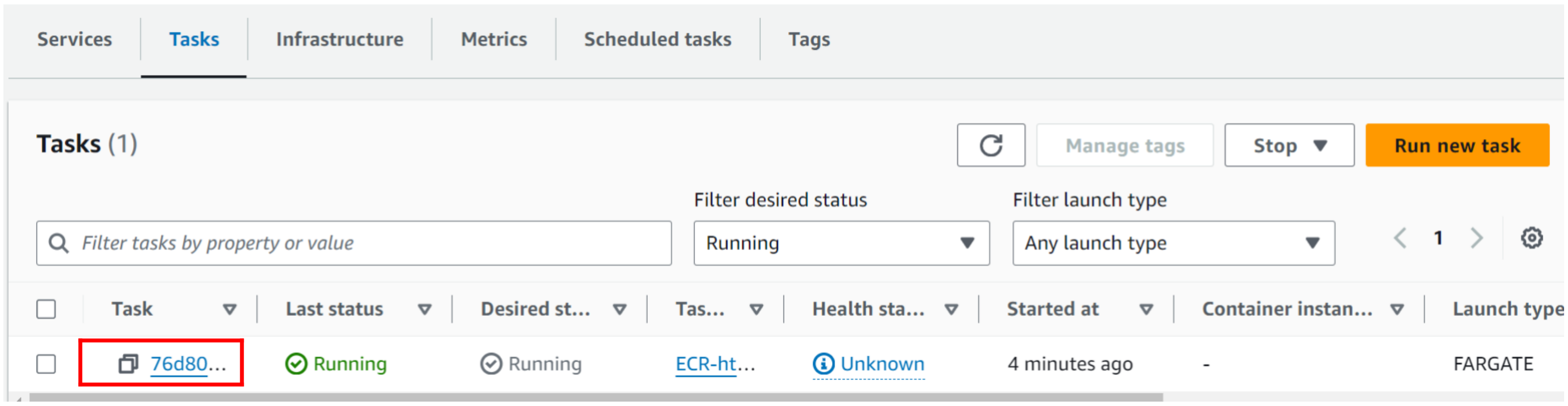
Type	Protocol	Port range	Source	Values	
HTTP	TCP	80	Anywhere	0.0.0.0/0, ::/0	Delete

Enter a valid port or port range between 0 and 65535. For example: 80 or 0-1023.

- Then click on **Create.**

5. Access **HTTPD Page**:

- Click on Task, that we created.



Services | Tasks | Infrastructure | Metrics | Scheduled tasks | Tags

Tasks (1)

Manage tags | Stop | Run new task

Filter tasks by property or value

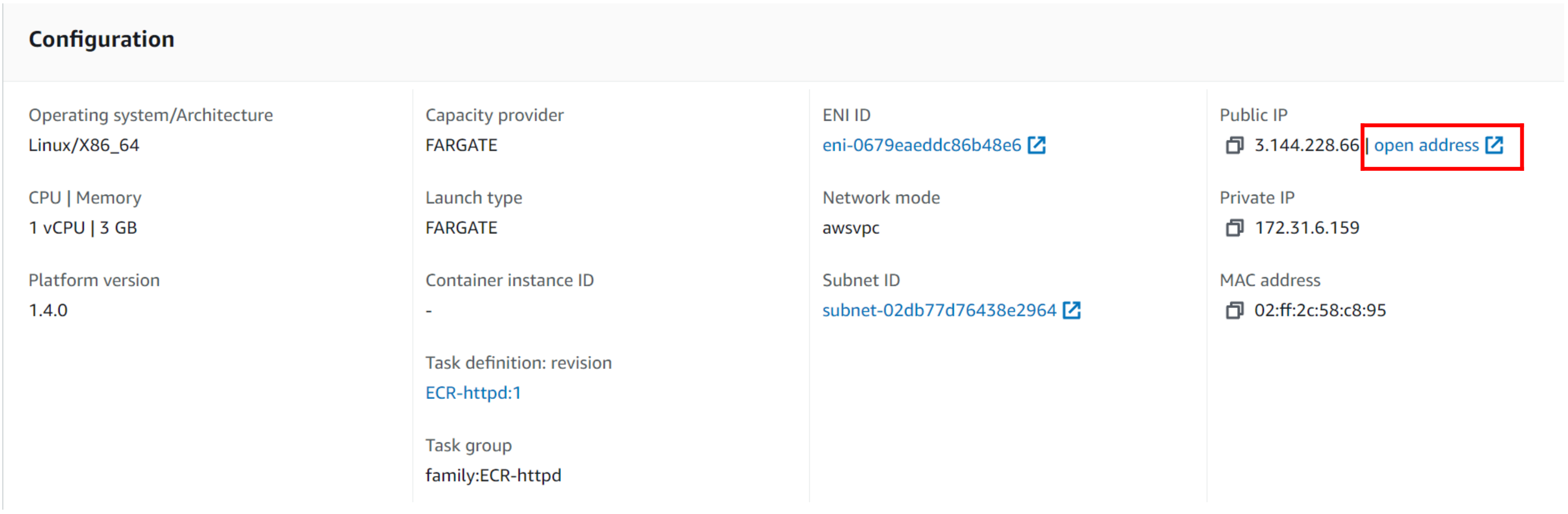
Filter desired status: Running

Filter launch type: Any launch type

Task	Last status	Desired st...	Tas...	Health sta...	Started at	Container instan...	Launch type
76d80...	Running	Running	ECR-ht...	Unknown	4 minutes ago	-	FARGATE

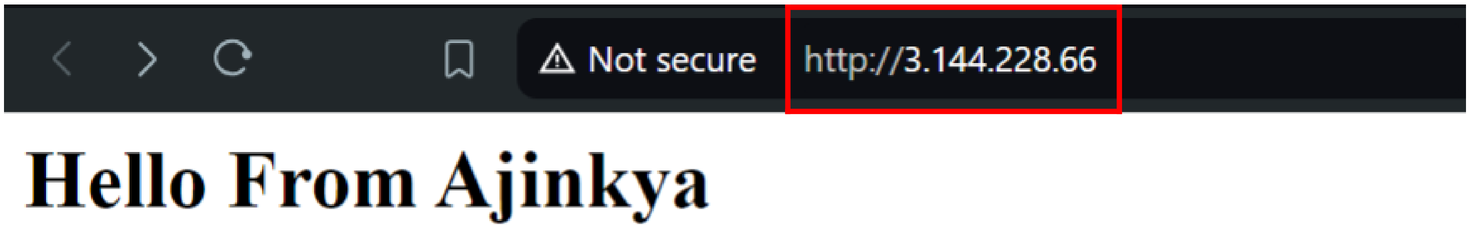
Ajinkya Kale

- Under **Configuration**, click on **open address.**
- Open the address in a web browser to access the **HTTPD** page.



Configuration

Operating system/Architecture Linux/X86_64	Capacity provider FARGATE	ENI ID eni-0679eaeddc86b48e6	Public IP 3.144.228.66 | open address
CPU | Memory 1 vCPU | 3 GB	Launch type FARGATE	Network mode awsvpc	Private IP 172.31.6.159
Platform version 1.4.0	Container instance ID -	Subnet ID subnet-02db77d76438e2964	MAC address 02:ff:2c:58:c8:95
	Task definition: revision ECR-httpd:1		
	Task group family:ECR-httpd		



Not secure http://3.144.228.66

Hello From Ajinkya

By following these steps, you will have built a Docker image with Apache httpd and a custom index page, pushed it to Amazon ECR, and deployed it as a containerized service on Amazon ECS using a task definition.

Ajinkya Kale