

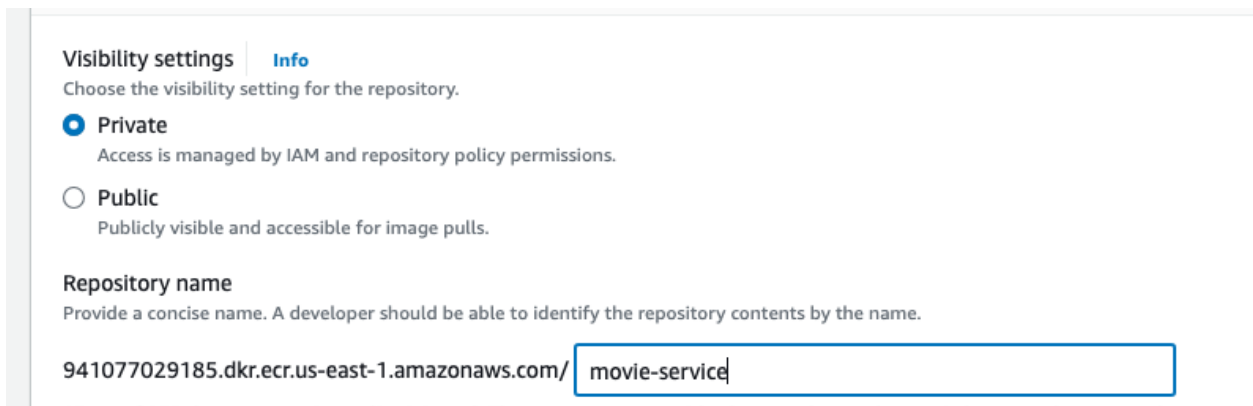
ECR

Creating Repositories	1
Local Access	2
Pushing Docker Images	3
Movie Service	3
Customer Service	4

Creating Repositories

We will create 2 repositories.

- movie-service
- customer-service



The screenshot shows the AWS ECR console interface for creating a new repository. It features two tabs: 'Visibility settings' (active) and 'Info'. Under 'Visibility settings', there are two radio button options: 'Private' (selected) and 'Public'. Below these, the 'Repository name' field is shown with a text input containing 'movie-service'. The field is preceded by the text '941077029185.dkr.ecr.us-east-1.amazonaws.com/'.

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.

941077029185.dkr.ecr.us-east-1.amazonaws.com/

Individual services will provide commands on how to push images.

Push commands for movie-service

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-stdin 941077029185.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 movie-service .
```

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

```
docker tag movie-service:latest 941077029185.dkr.ecr.us-east-1.amazonaws.com/movie-service:latest
```

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

```
docker push 941077029185.dkr.ecr.us-east-1.amazonaws.com/movie-service:latest
```

Create these 2 repositories

Repositories (2)

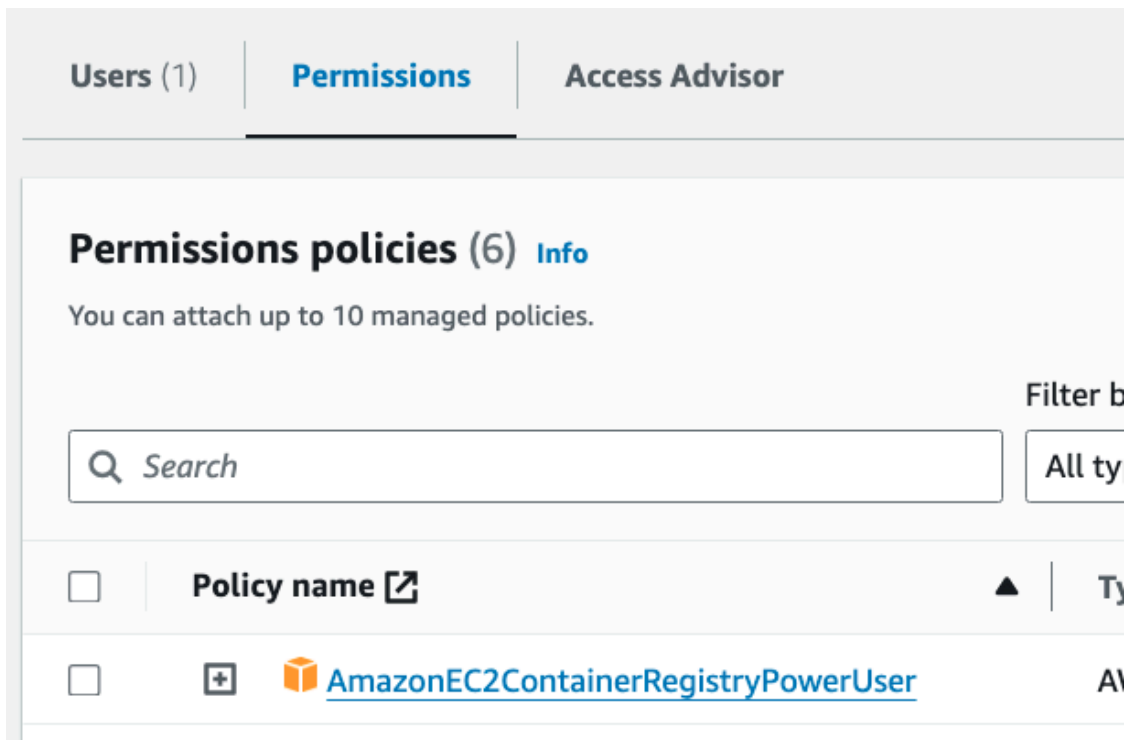
Filter status

	Repository name ▲	URI
<input type="radio"/>	customer-service	941077029185.dkr.ecr.us-east-1.amazonaws.com/customer-service
<input type="radio"/>	movie-service	941077029185.dkr.ecr.us-east-1.amazonaws.com/movie-service

Local Access

- Ensure that your dev user has access to push docker images into the ECR.

- I added the below permission to the “vins-dev” user.



Pushing Docker Images

At this point, we can push docker images. ***Ensure that you have configured your aws credentials*** in your local.

You should have credentials under

```
~/.aws/credentials
```

You should have config directory

```
~/.aws/config
```

Movie Service

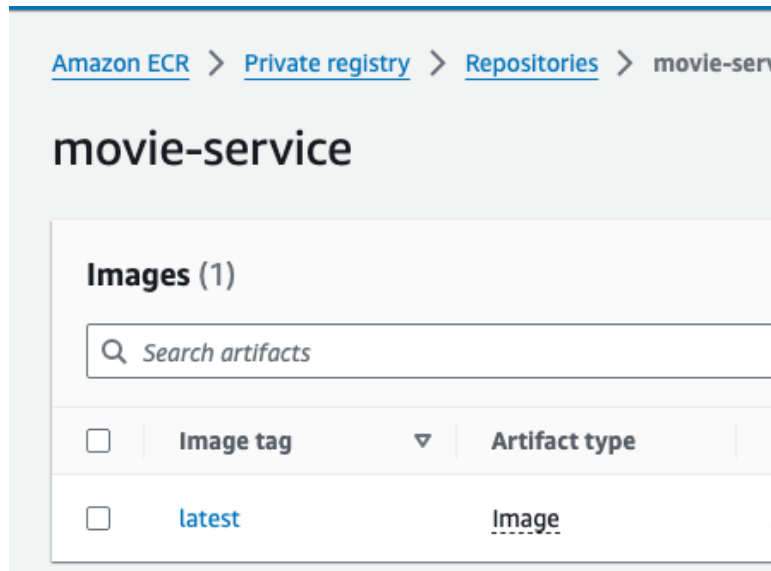
- Create the app jar

```
mvn clean package
```

- Build the docker image - **DO NOT FORGET the --platform**

```
docker build -t movie-service . --platform=linux/amd64
```

- We can follow the rest of the instructions from the ECR push-commands for movie-service



Customer Service

Repeat the steps for customer-service as well.

