**Containerization and Automation related projects**

We shall plan to create a landing page with 2 links 2 separate lambda functions each for the above links. The lambda functions are to be containerised, the S3 storage and lambda are to be deployed using cloudformation templates.

Folder structure for required files

landing\_page/

│

**├── index.html**

│

**├── lambda\_functions/**

│ ├── lambda1/

│ │ └── lambda1.py

│ │ └── Dockerfile.lambda1

│ │

**│ ├── lambda2/**

│ │ └── lambda2.py

│ │ └── Dockerfile.lambda2

│

**├── cloudformation/**

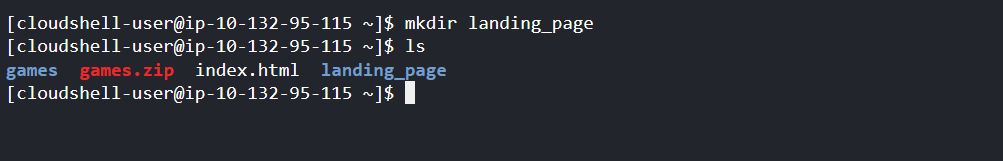
├── lambda1\_stack.yaml

├── lambda2\_stack.yaml

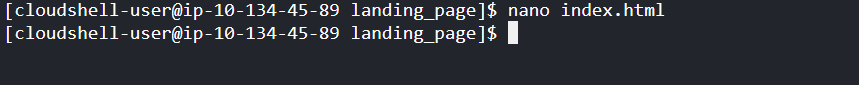
├── api\_gateway\_stack.yaml

└── s3\_bucket\_stack.yaml

1. Create the landing\_page directory

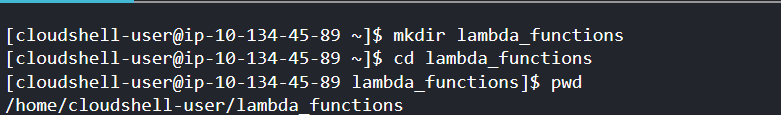


1. Navigate into the landing\_page directory and Create index.html file

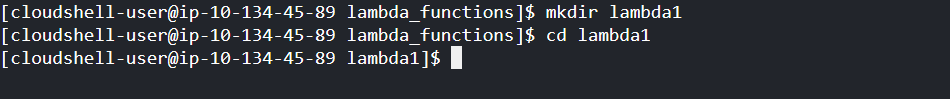




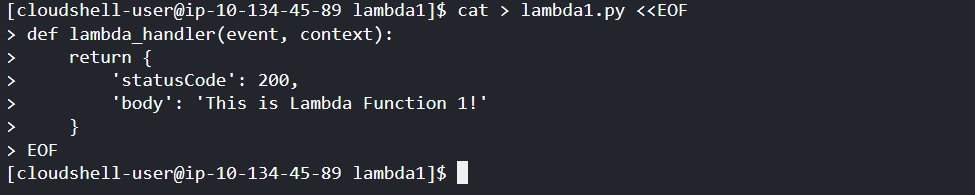
1. Create and navigate into the lambda\_functions directory



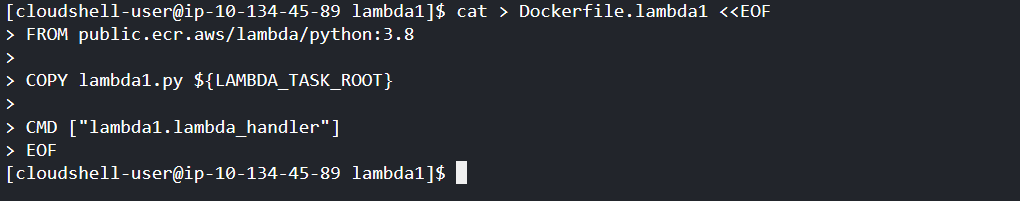
1. Create and navigate into the lambda1 directory



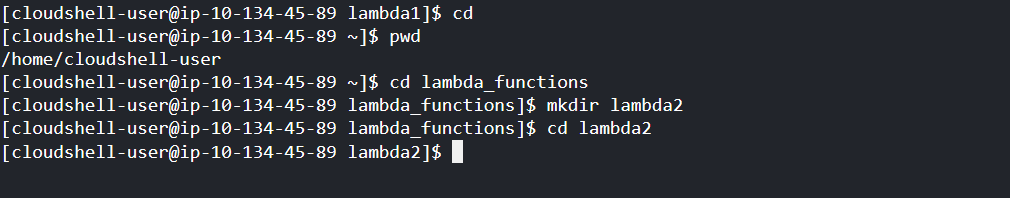
1. Create lambda1.py file



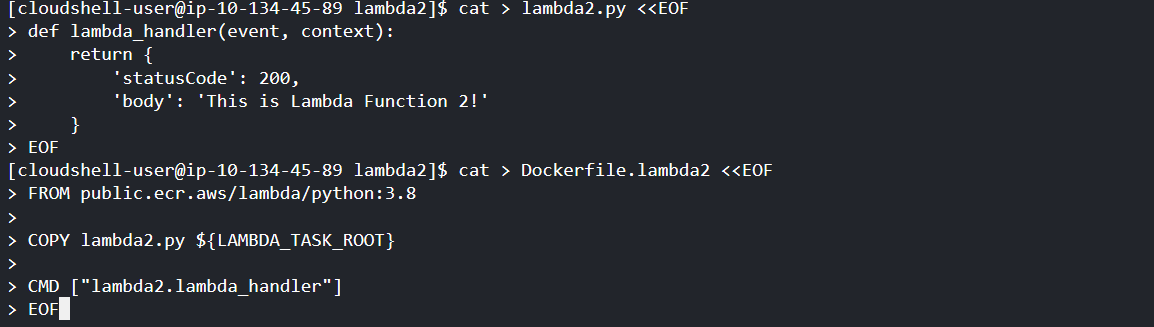
1. Create Dockerfile.lambda1 file



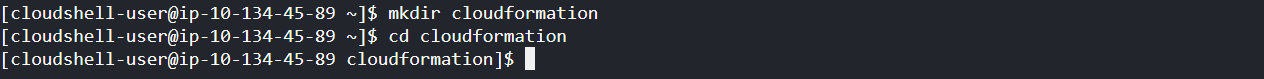
1. Navigate back to the lambda\_functions directory, Create and navigate into the lambda2 directory



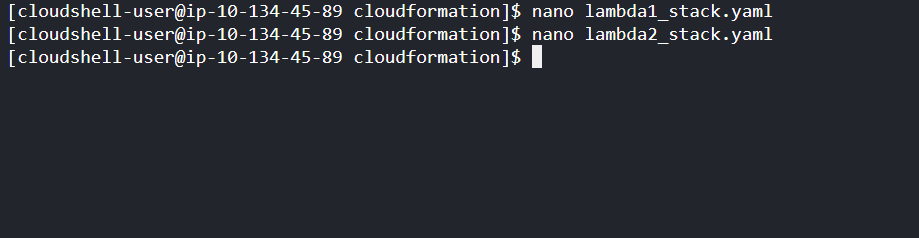
1. Create lambda2.py file and Dockerfile.lambda2 file

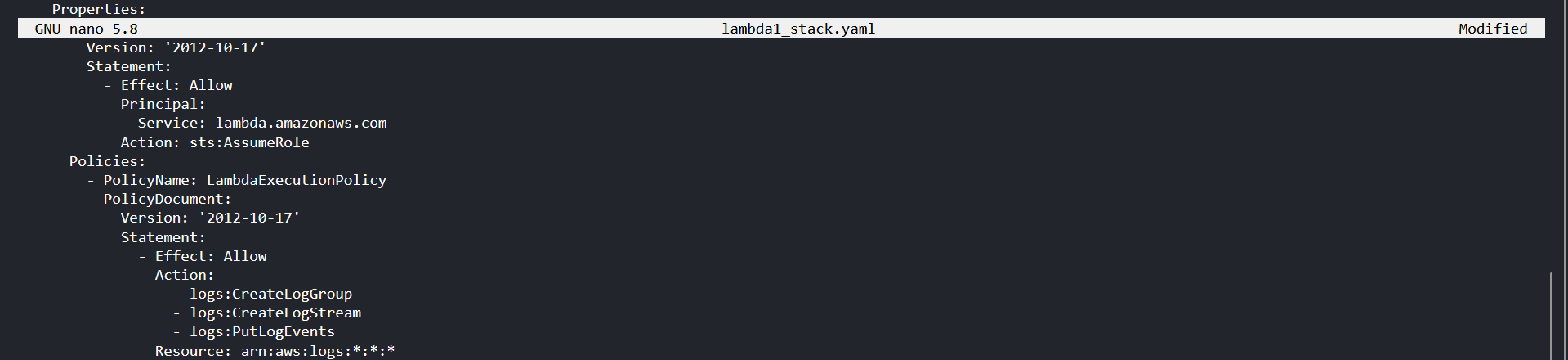


1. Create and navigate into **cloudformation folder**

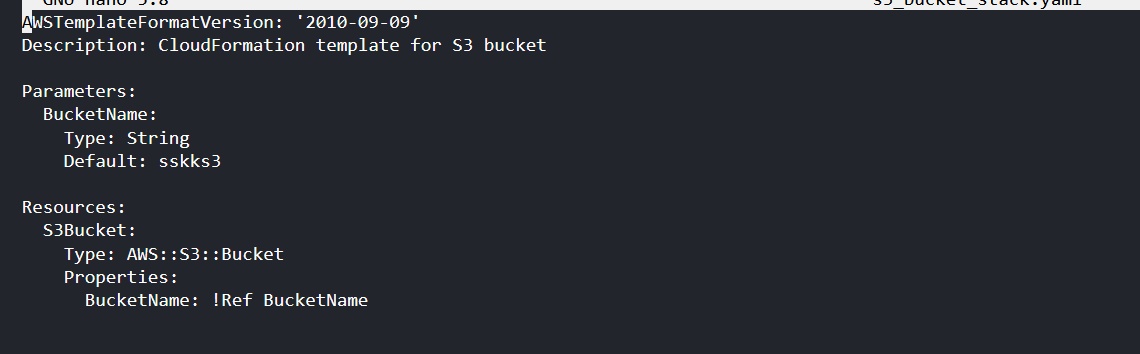


1. Create yaml for **Lambda Function 1 Stack and Lambda Function 2 Stack**

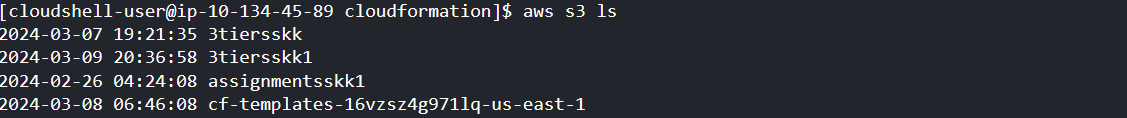
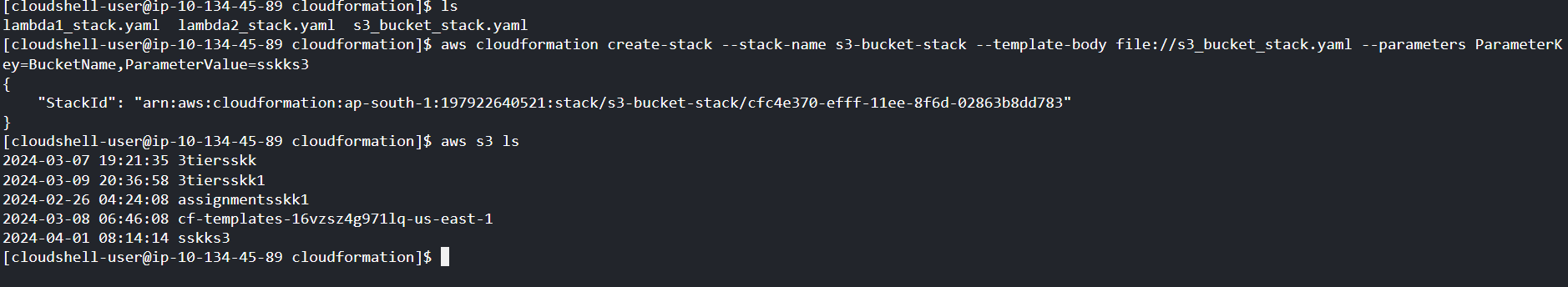
****



1. Create a template for s3 bucket



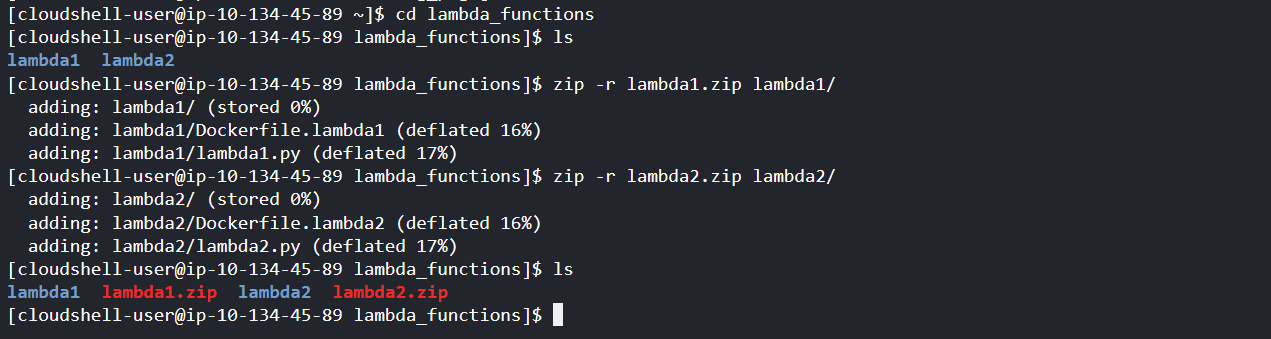
1. Deploy S3 bucket stack

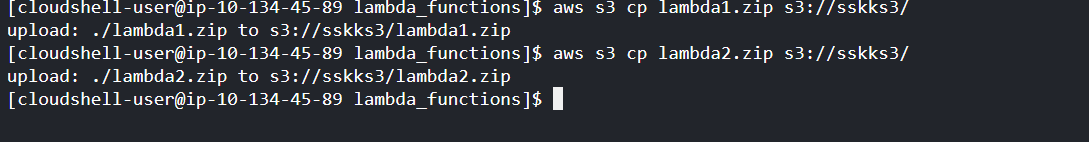
1. **Zip Lambda Function Files and Upload to S3**

First, navigate to the directory containing your Lambda function code

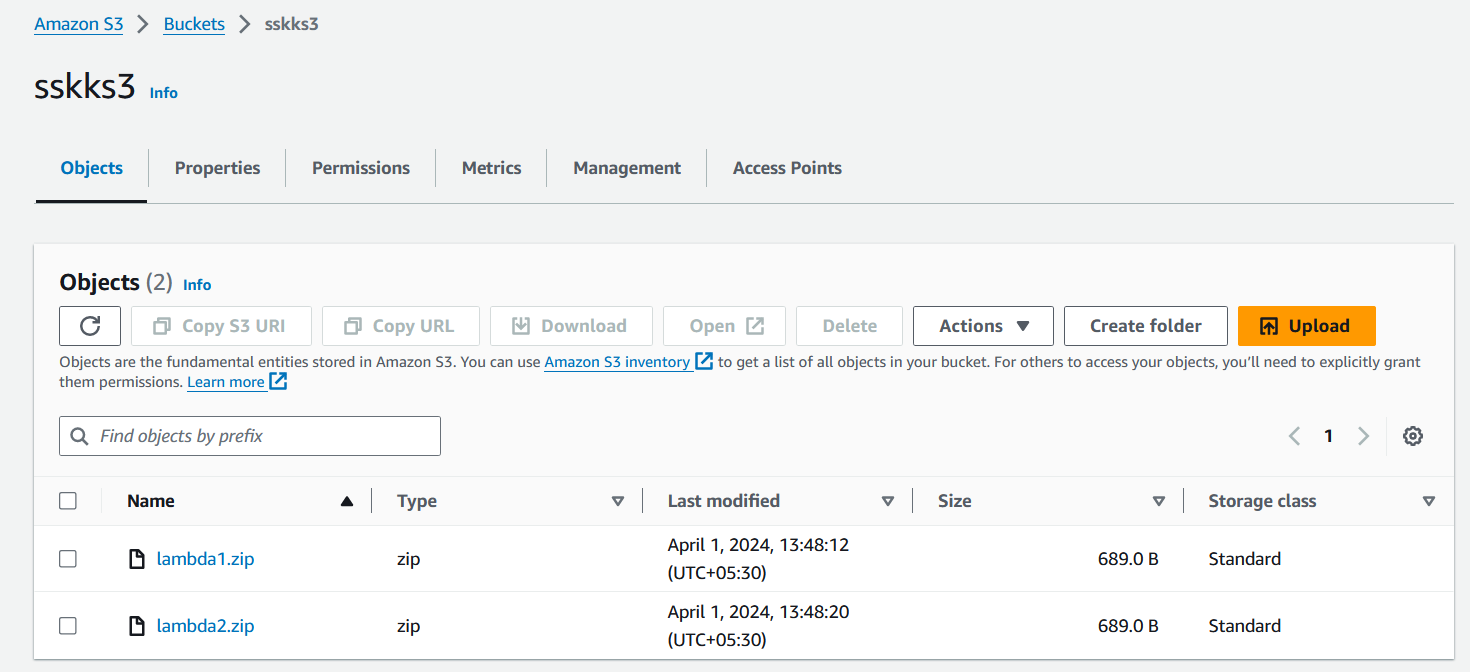
Then, you can use the zip command to create a zip file containing the Lambda function code files



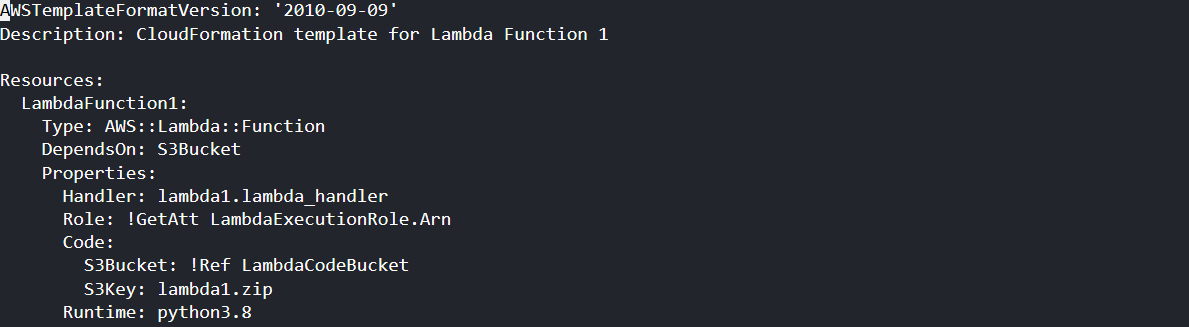
1. **Upload zip files to S3:**

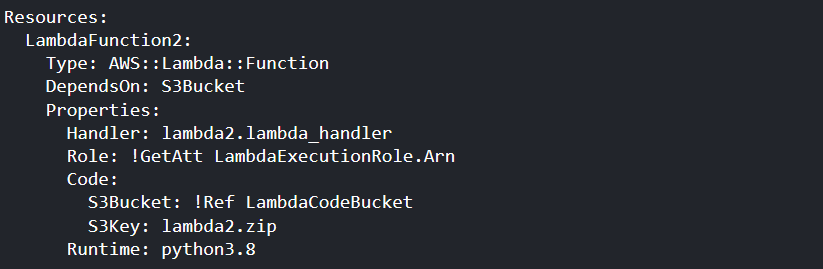


1. **Verify upload:**

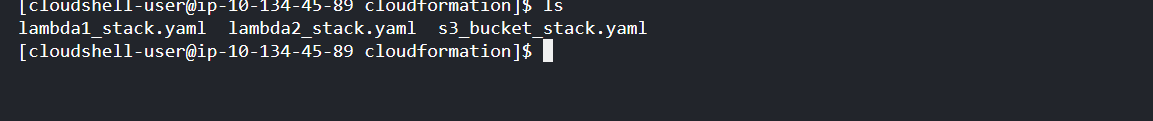
****

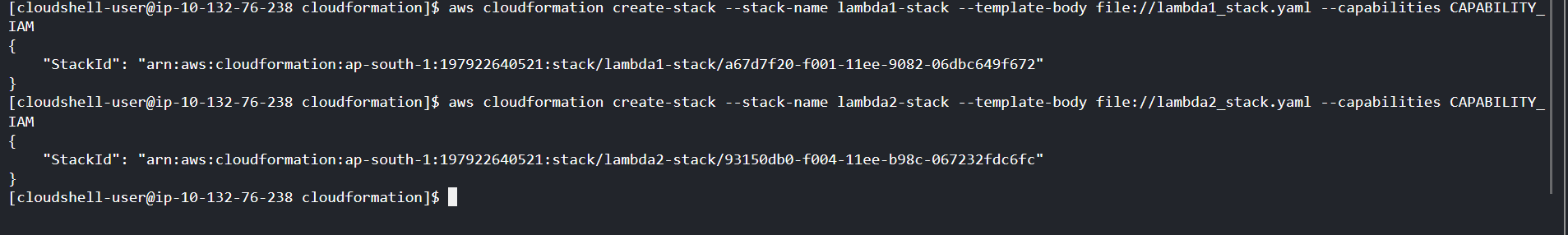
1. Update the CloudFormation templates for the Lambda function stacks to include a dependency on the S3 bucket stack. We can use the DependsOn attribute.

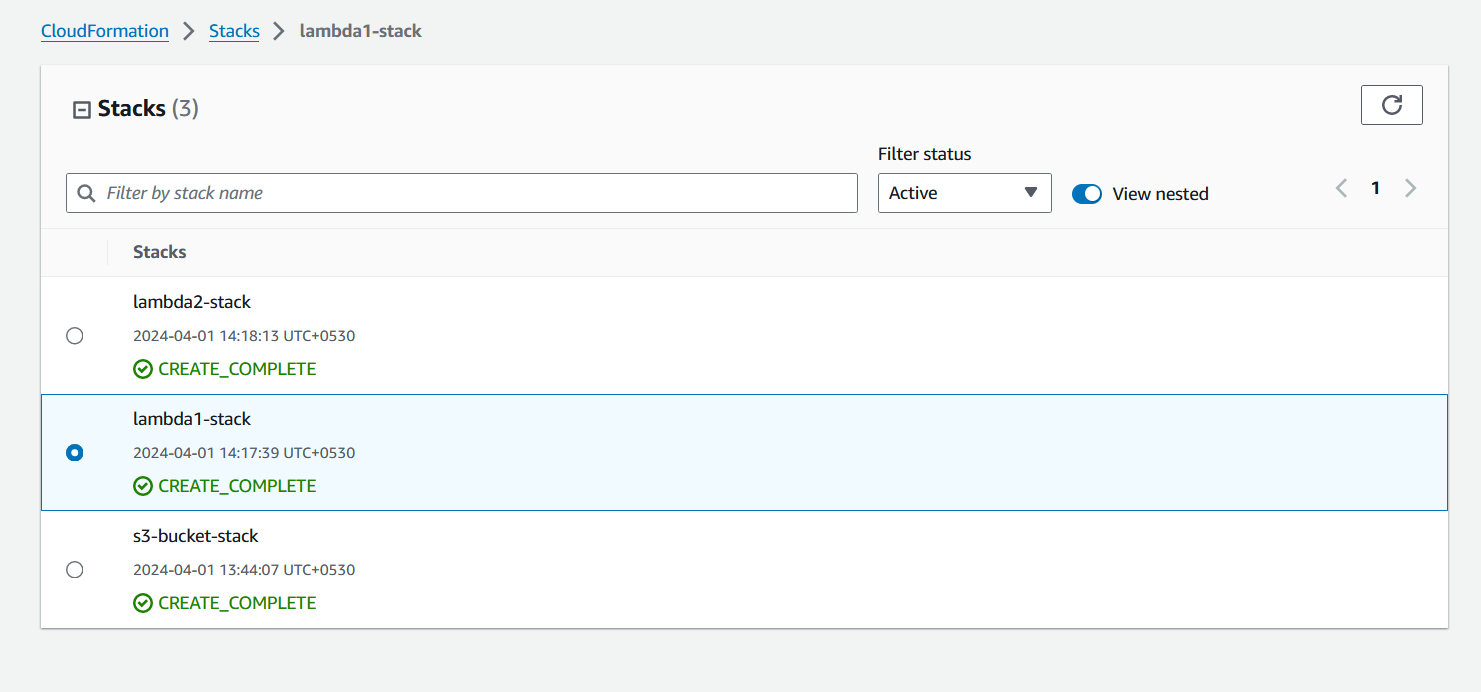




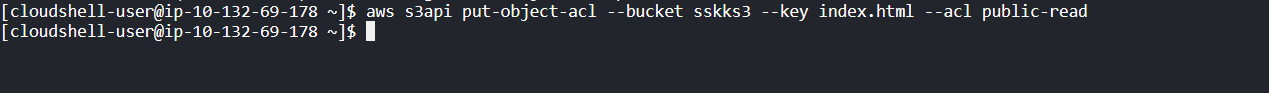
1. Deploy Lambda function stacks with dependency on S3 bucket stack



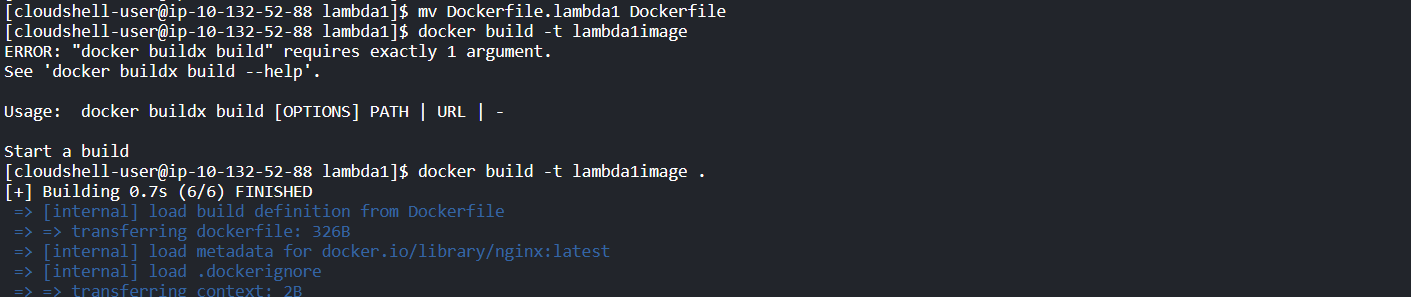
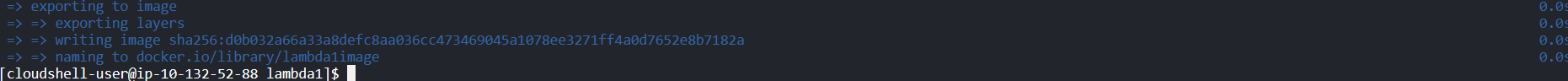
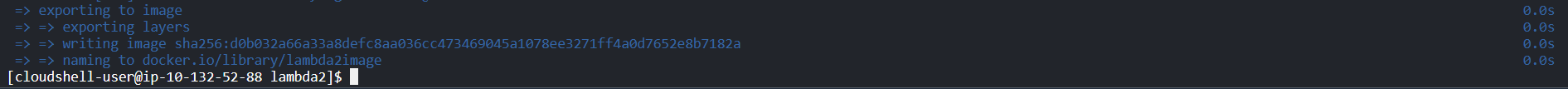


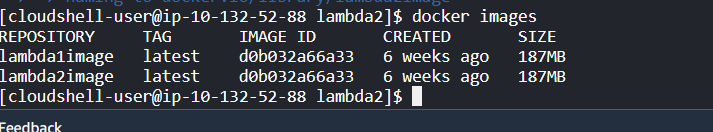


1. **Set** Permissions **for Website Access**

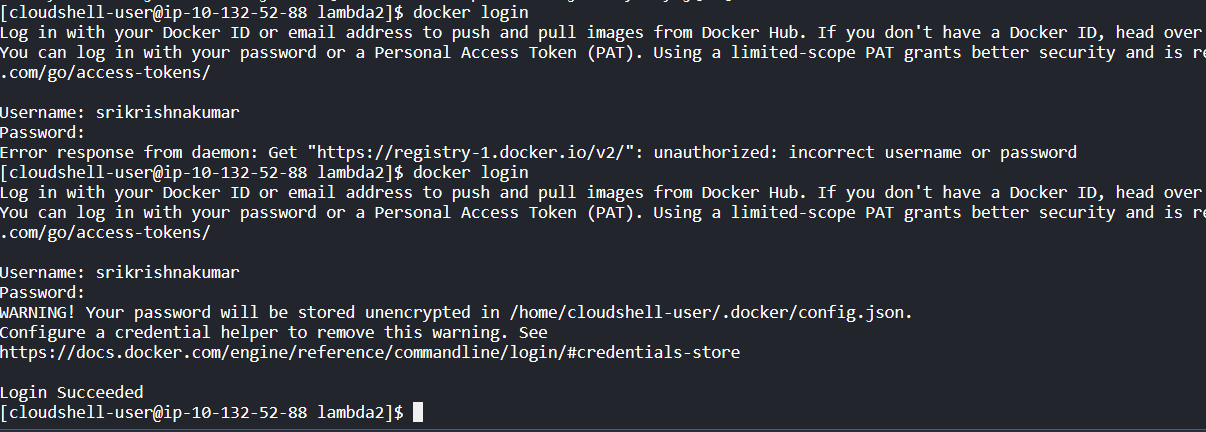


1. **Build Docker Image with already prepared Dockerfiles and verify the creation**

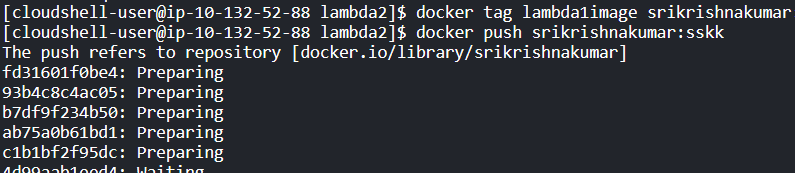
  



1. Login to docker using the credentials

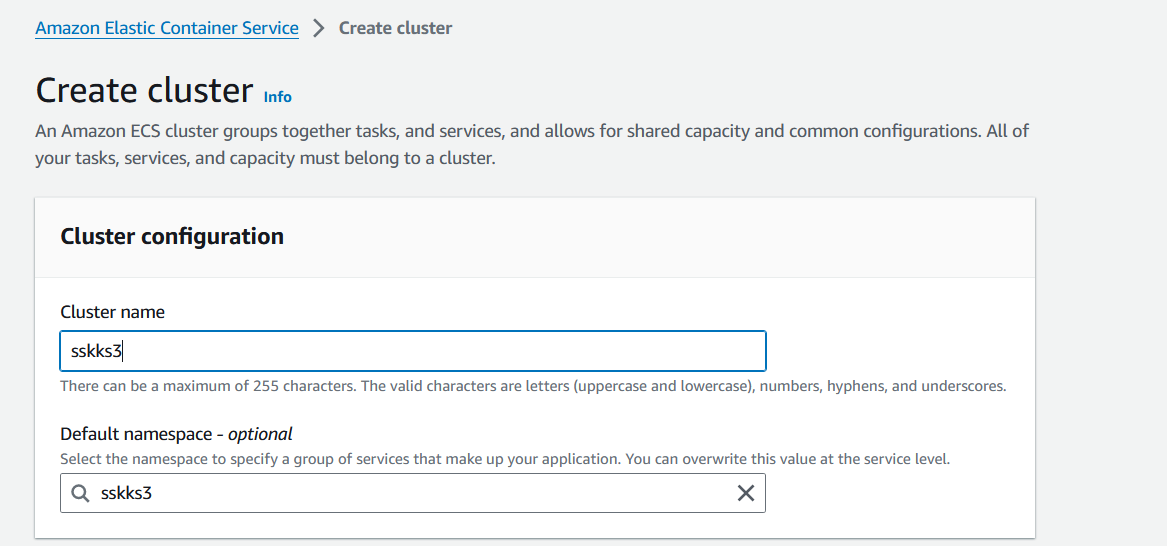


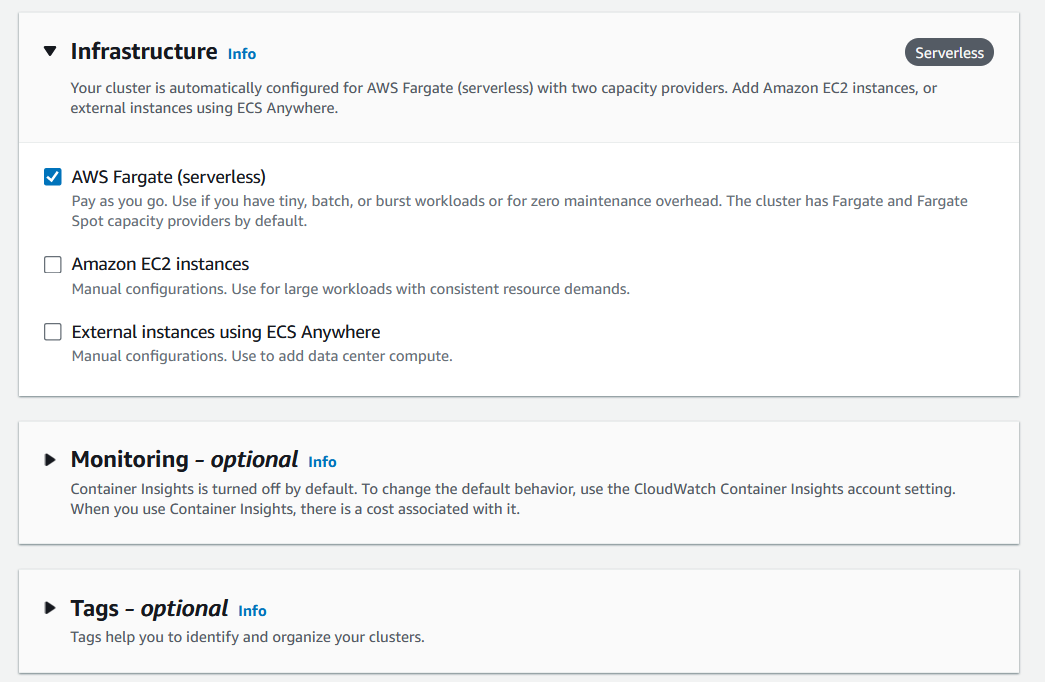
1. Tag the images using ‘docker tag’ command and Push the images using ‘docker push’ cpmmand

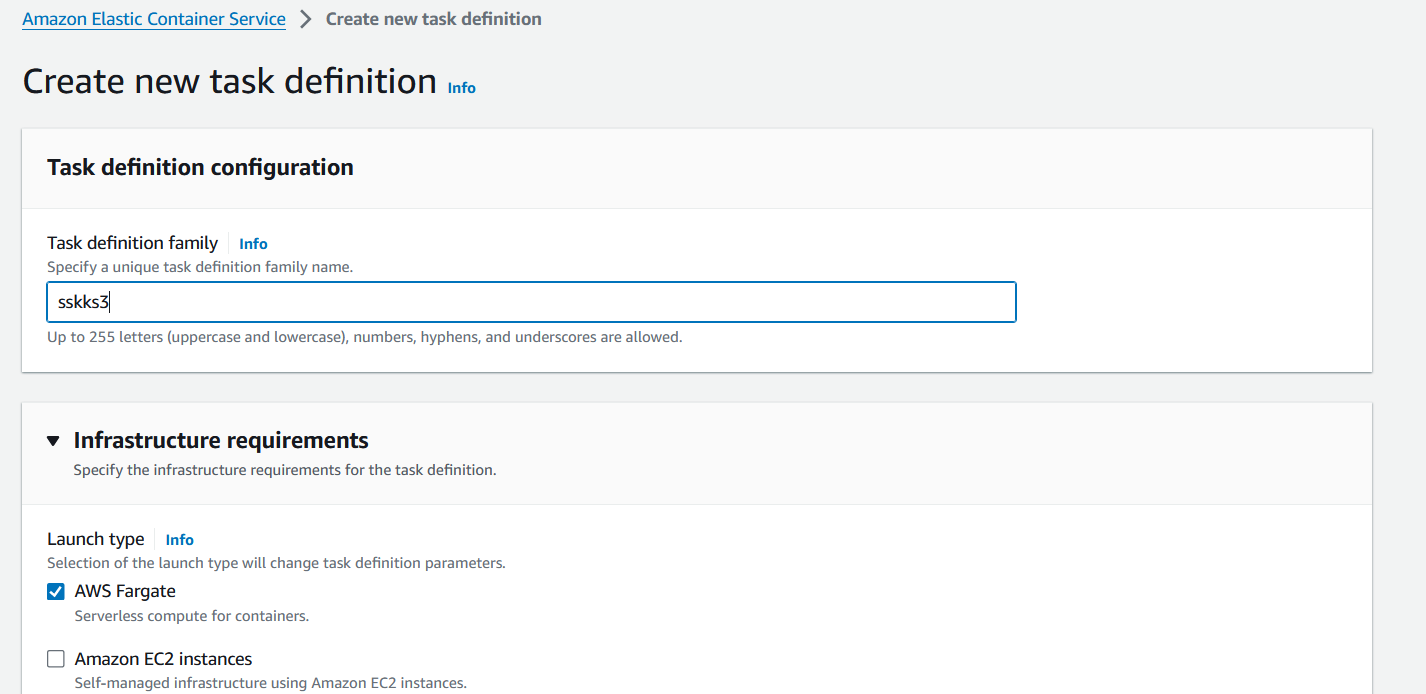


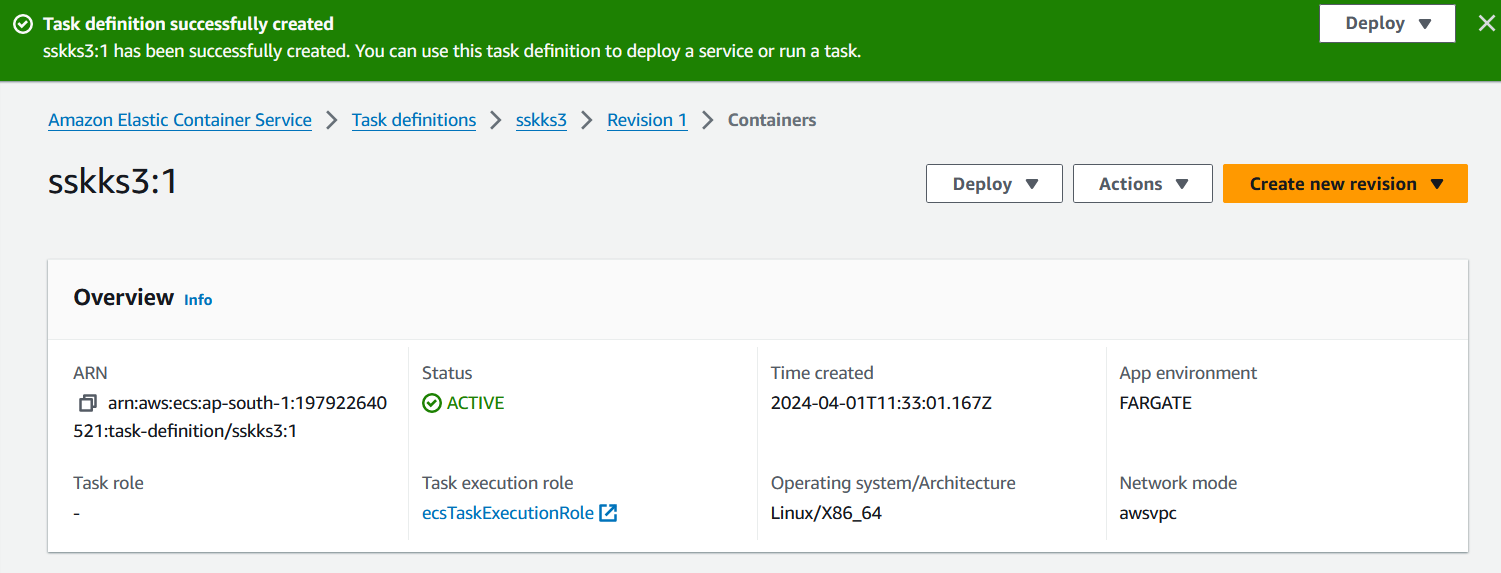
**Check IAM Permissions**: Ensure that the IAM user or role you're using to push the Docker images has the necessary permissions to perform the ecr:PutImage action on the target ECR repository.

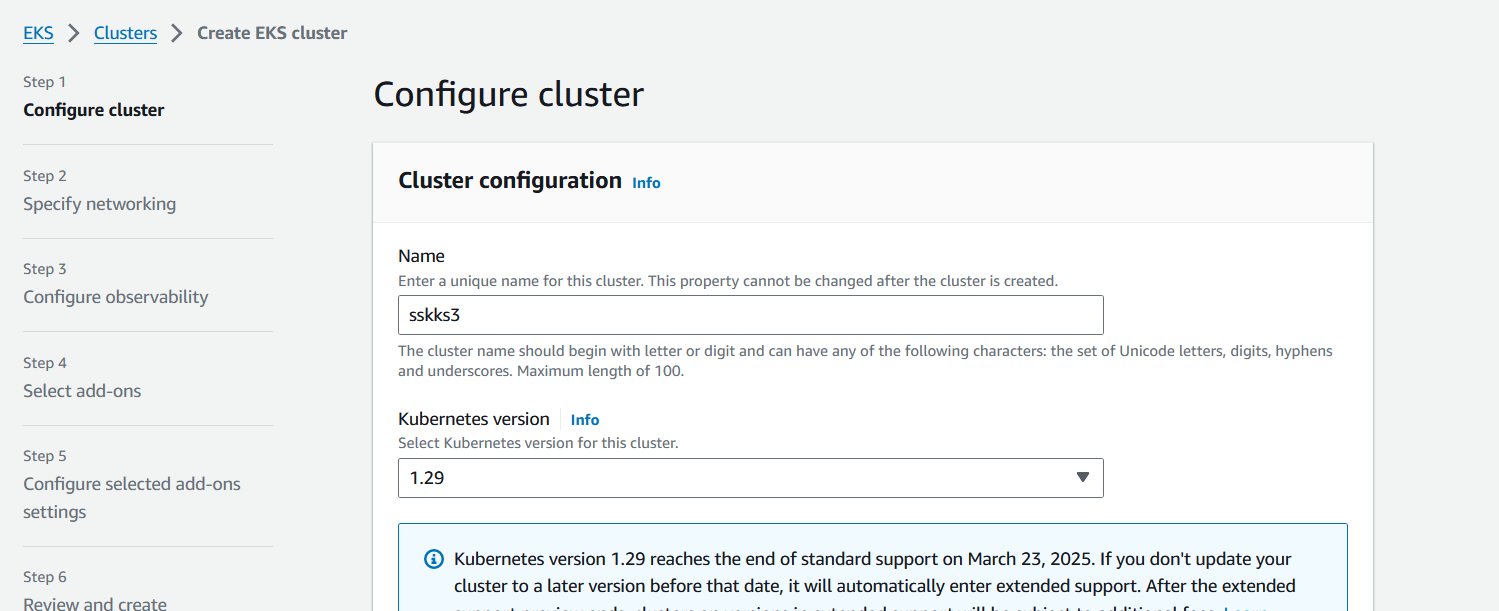
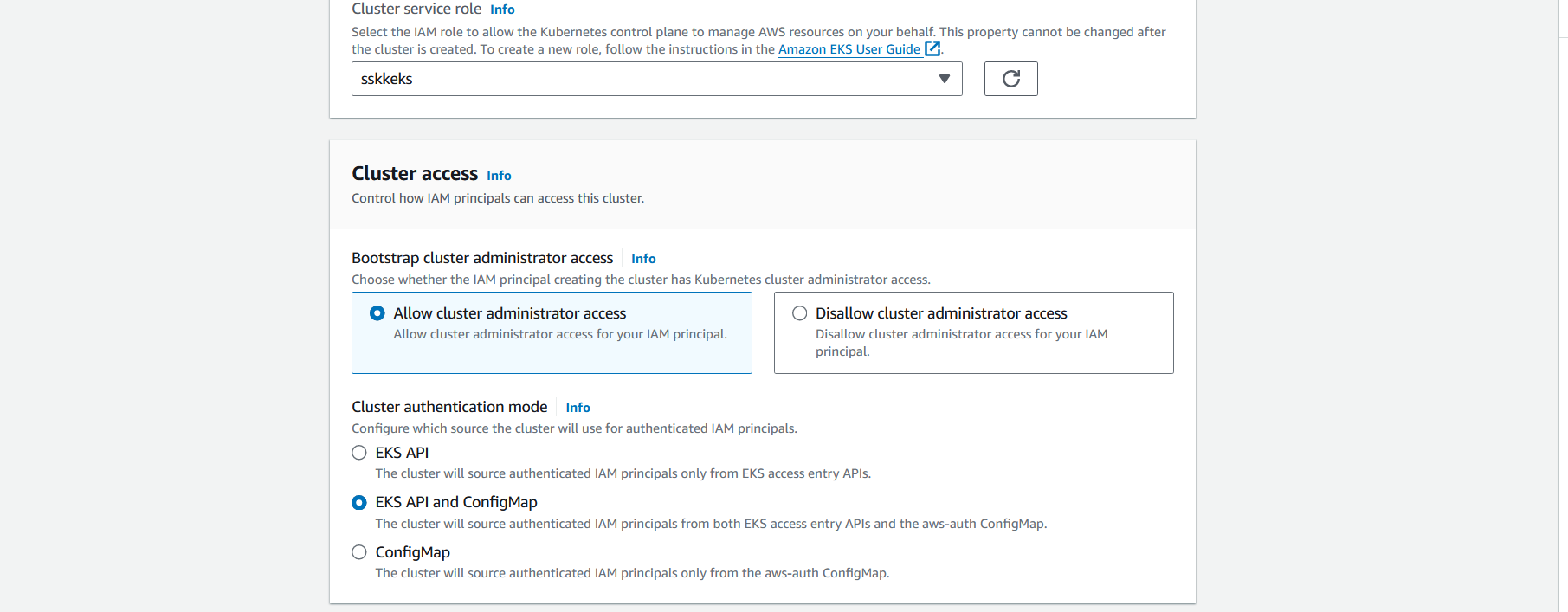
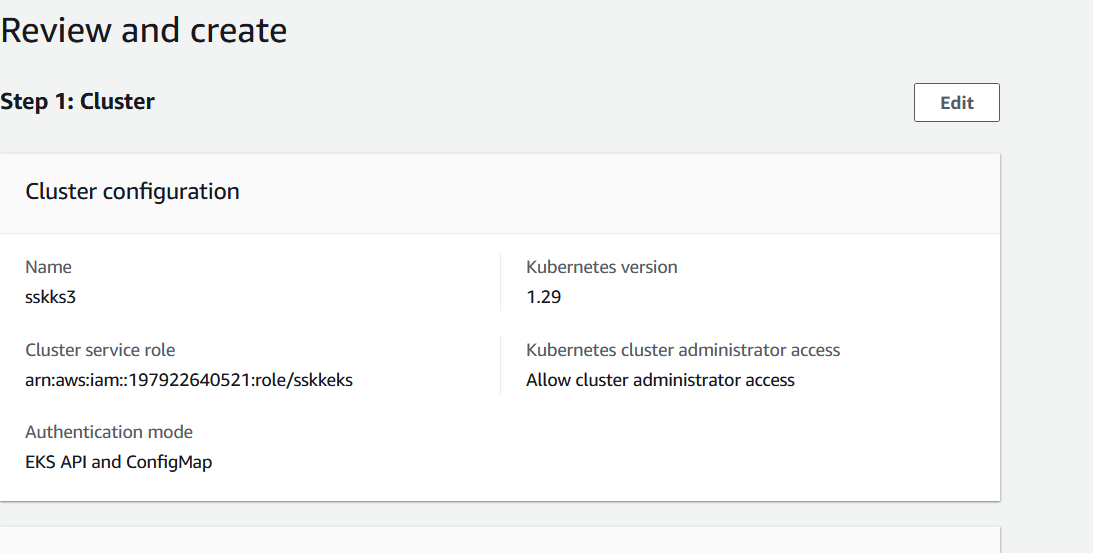
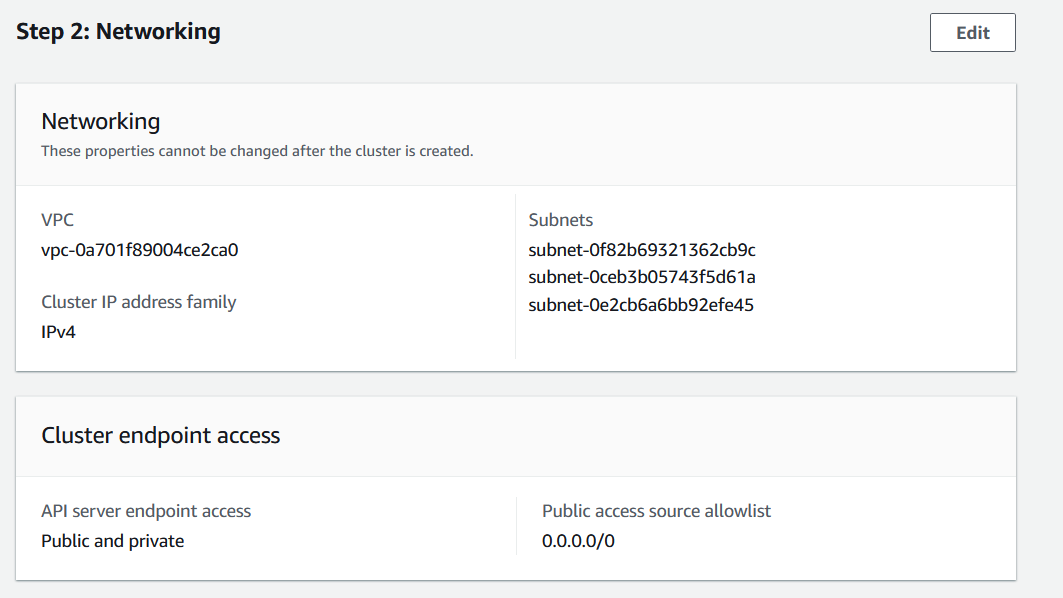
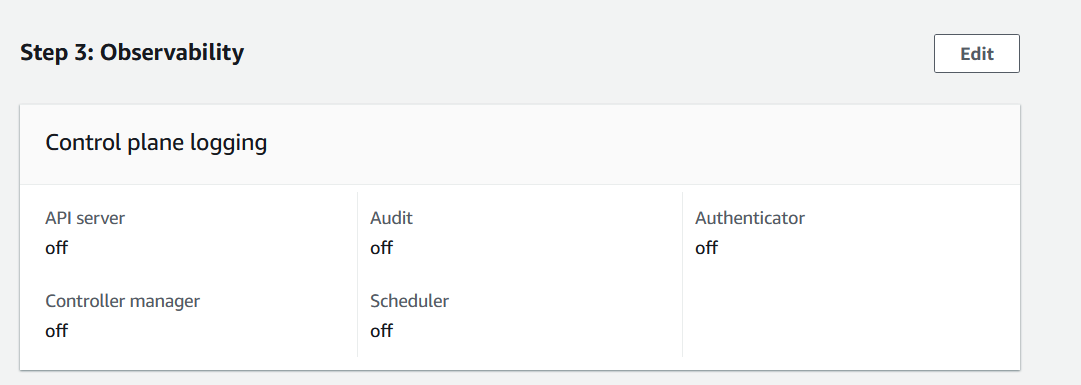
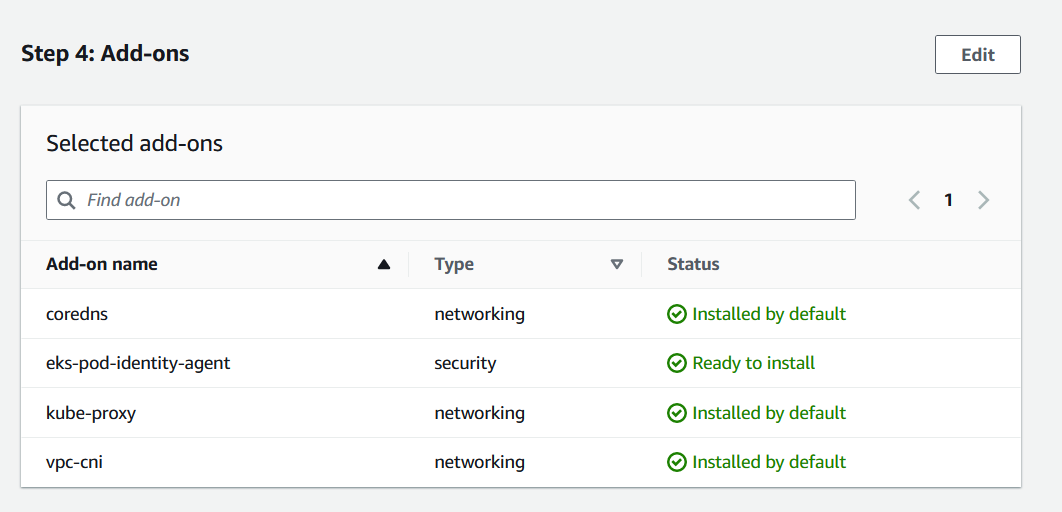
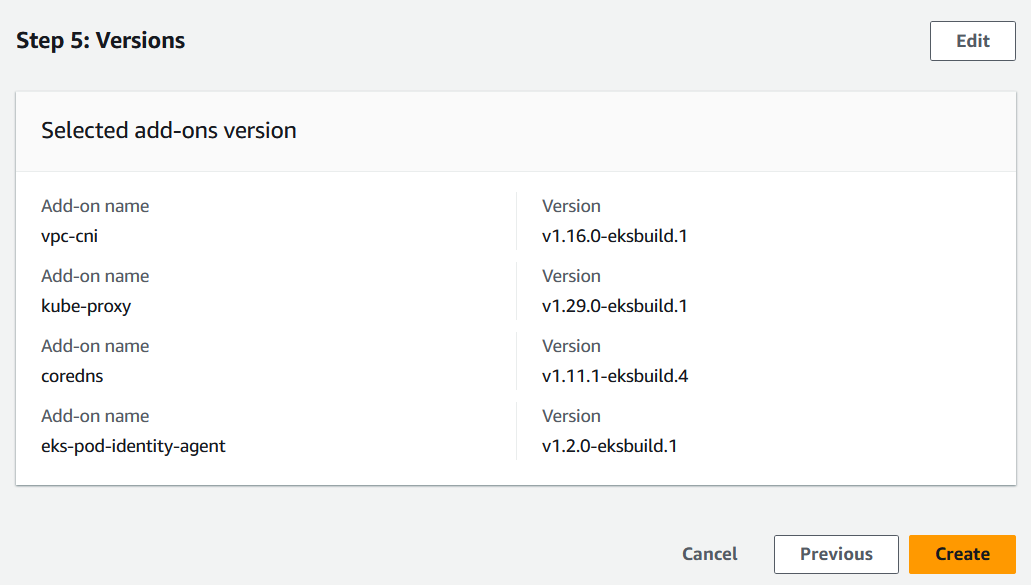
1. Create ECS cluster to deploy the images



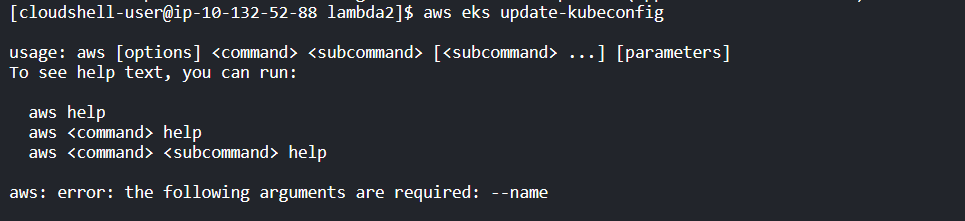


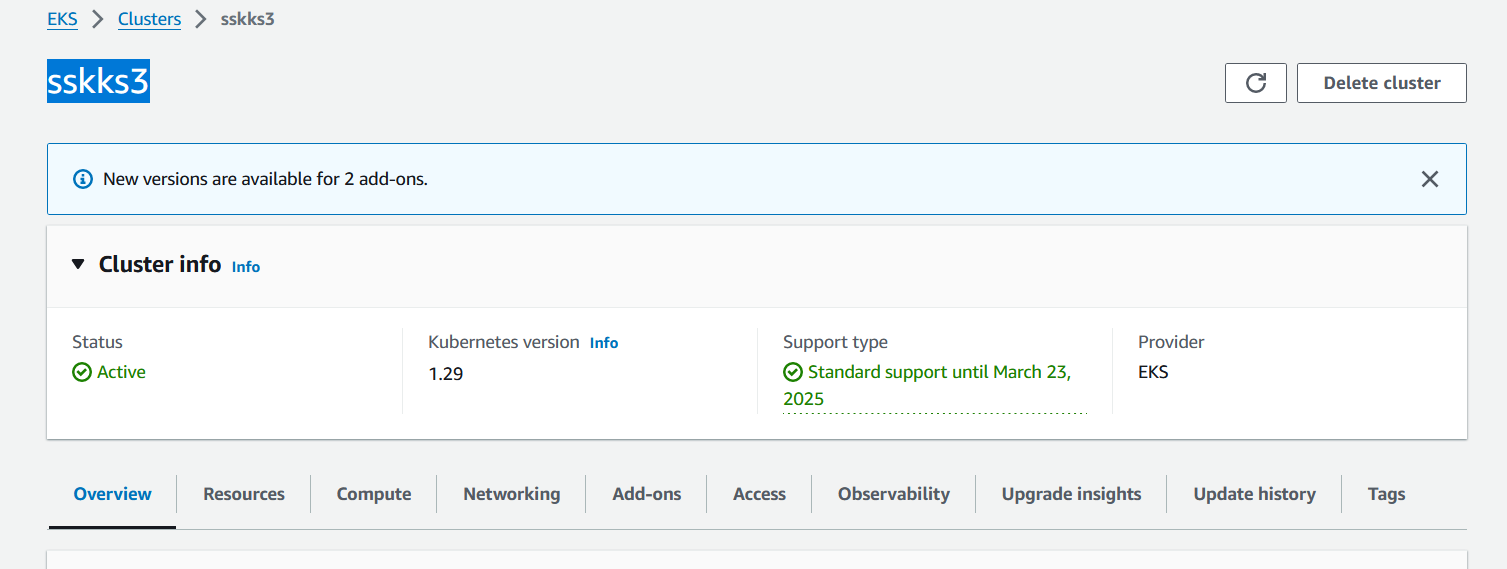




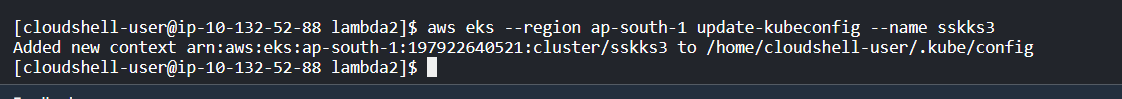
      

1. Configure EKS to deploy the images

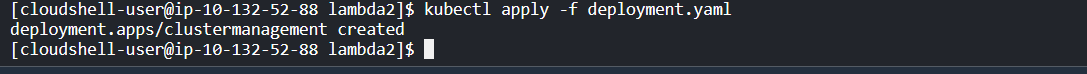
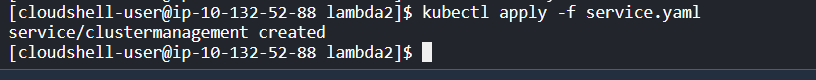




1. Integrate the Lambda functions into index.html, you can use JavaScript to make AJAX requests to invoke the Lambda functions asynchronously



1. Deploy the cloudformation templates

1. Verify the website and check for both lambda functions to be invoked properly



