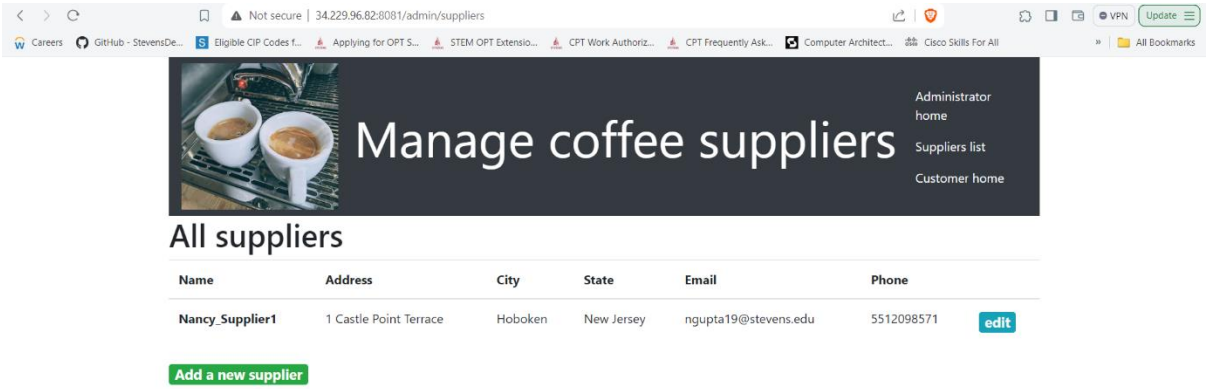
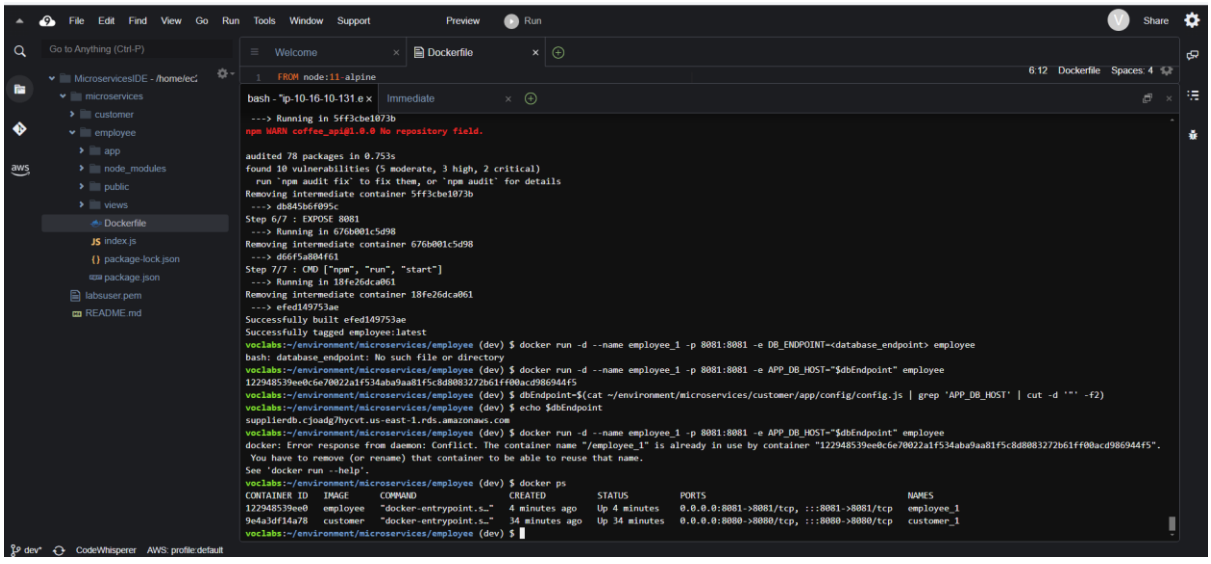
Verify that the supplier no longer appears on the suppliers page.



Troubleshooting tip: If any of the functionality mentioned above isn't working as intended, follow these steps: (1) Stop and delete the running container, (2) modify the microservice source code as appropriate, (3) create an updated Docker image from the source code, (4) launch a new test container, and (5) verify whether the functionality is now available. For a list of commands to run to accomplish these steps, see Updating a test container running on Cloud9 in the appendix of this file.

5. **To observe details about both running test containers, run the following command:**

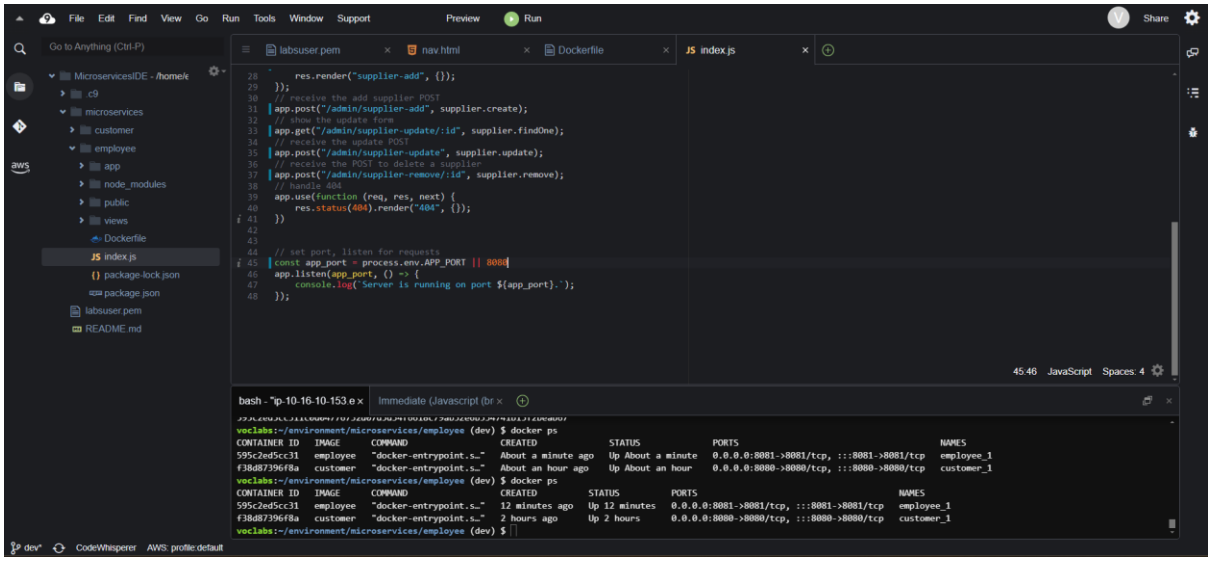
docker ps

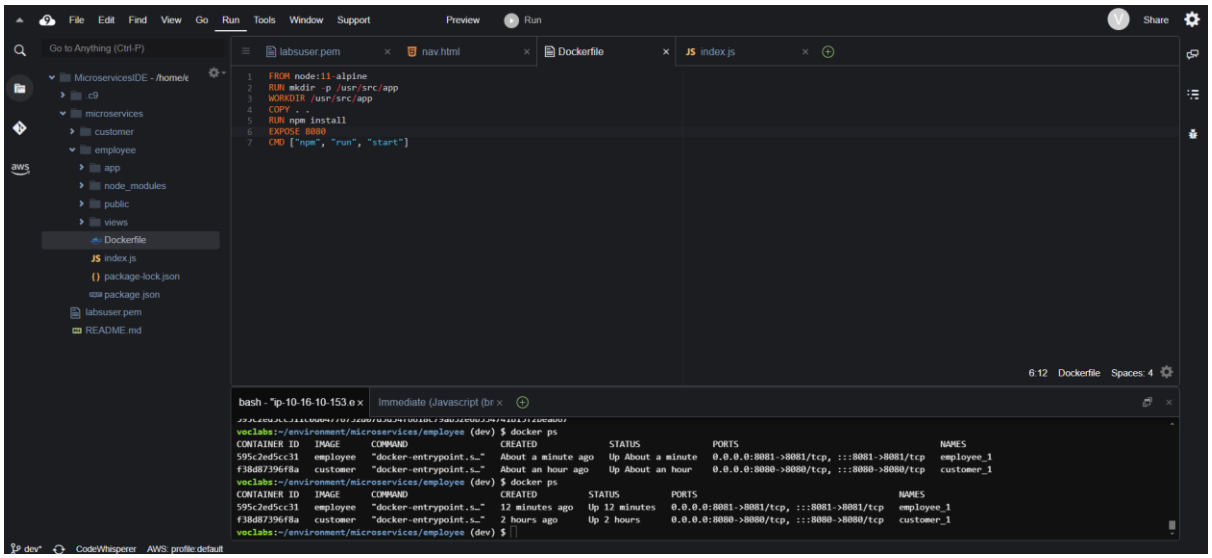


Task 4.6: Adjust the *employee* microservice port and rebuild the image

When you tested the employee microservice on the AWS Cloud9 instance, you ran it on port 8081. However, when you deploy it to Amazon ECS, you will want it to run on port 8080. To adjust this, you need to modify two files.

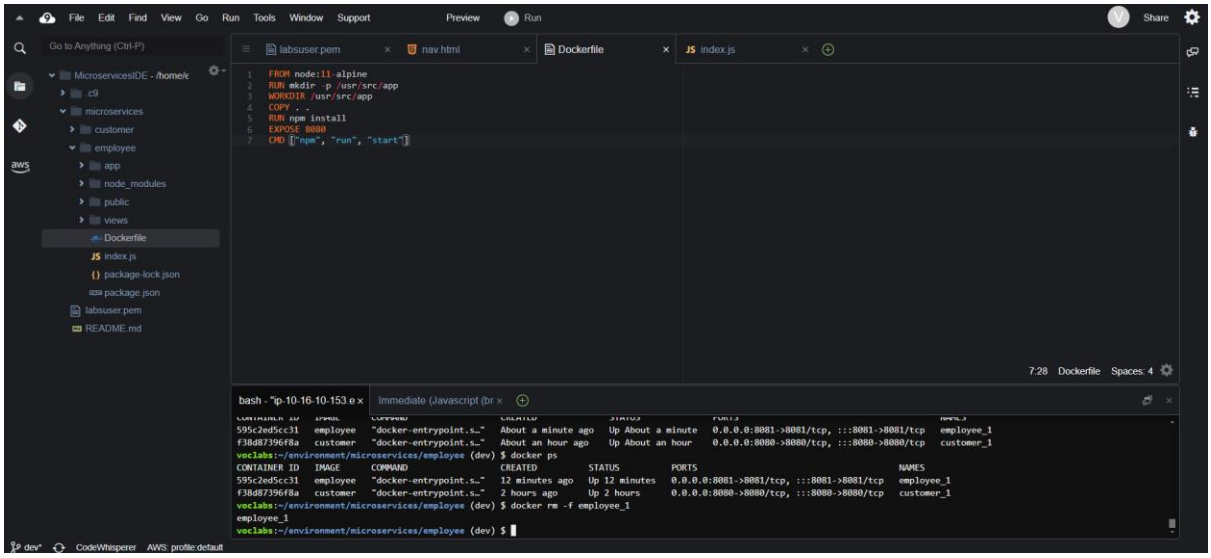
1. **Edit the employee/index.js and employee/Dockerfile files to change the port from 8081 to 8080**





2. **Rebuild the Docker image for the employee microservice.**
    - **To stop and delete the existing container (assumes that the container name is *employee_*), run the following command:**

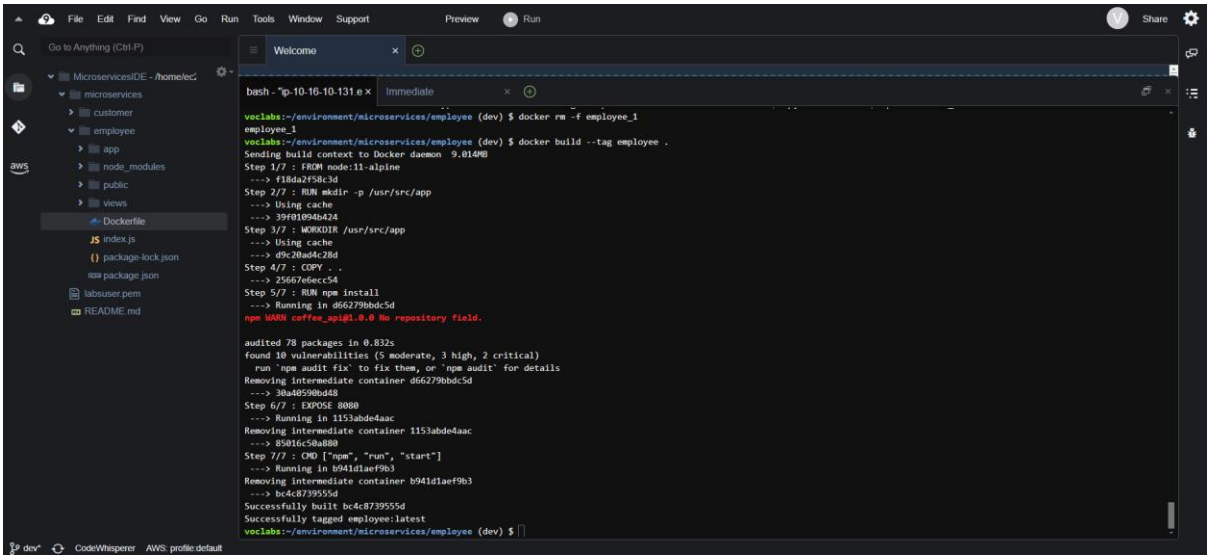
```
docker rm -f employee_1
```



- **Ensure that your terminal is in the employee directory.**
- **Use the docker build command to build a new image from the latest source files that you edited. Use the employee tag.**

Use the following command:

docker build --tag customer .



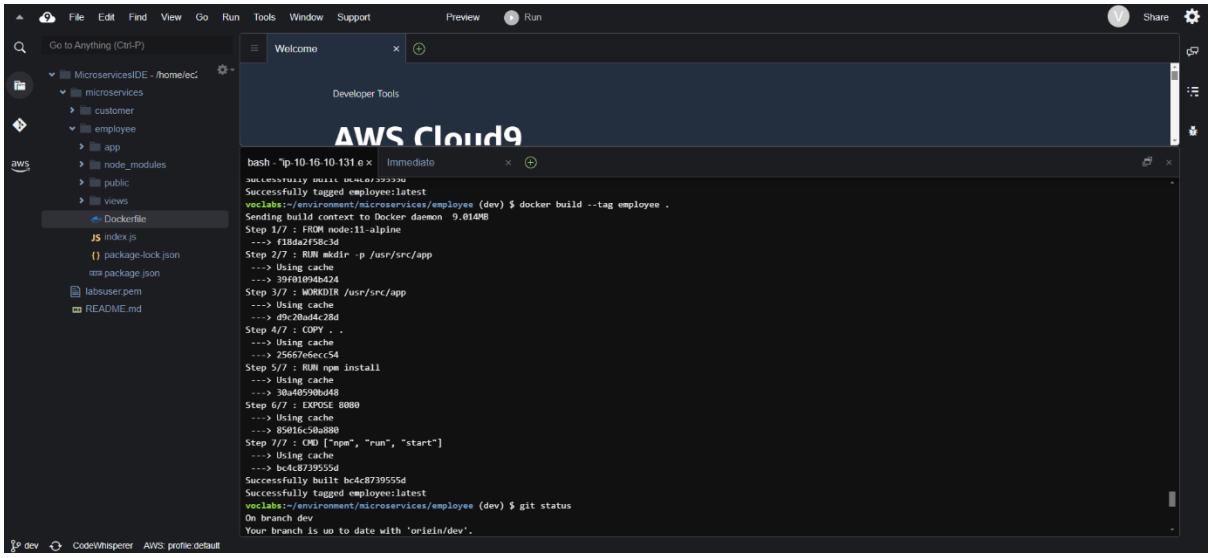
Tip: If you build an image with the name of an existing image, the existing image will be overwritten.

Note: You don't need to run a new test container, so you don't need to run docker run.

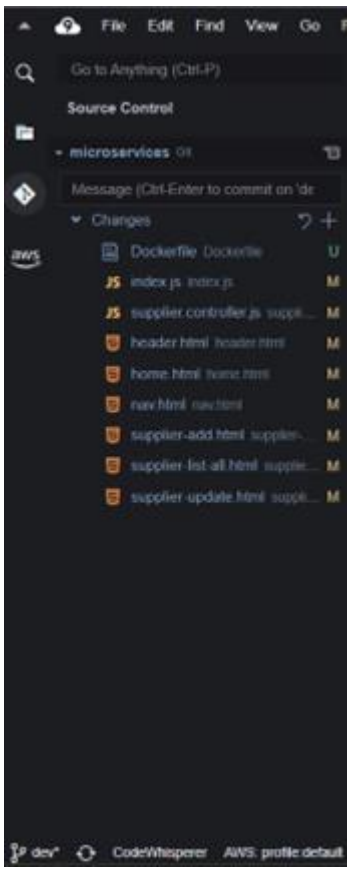
Task 4.7: Check code into CodeCommit

In this task, you will commit and push the changes that you made to the employee microservice to CodeCommit.

1. **Review the updates that you made to the source code. To accomplish this:**
   - **Choose the source control icon in the AWS Cloud9 IDE.**

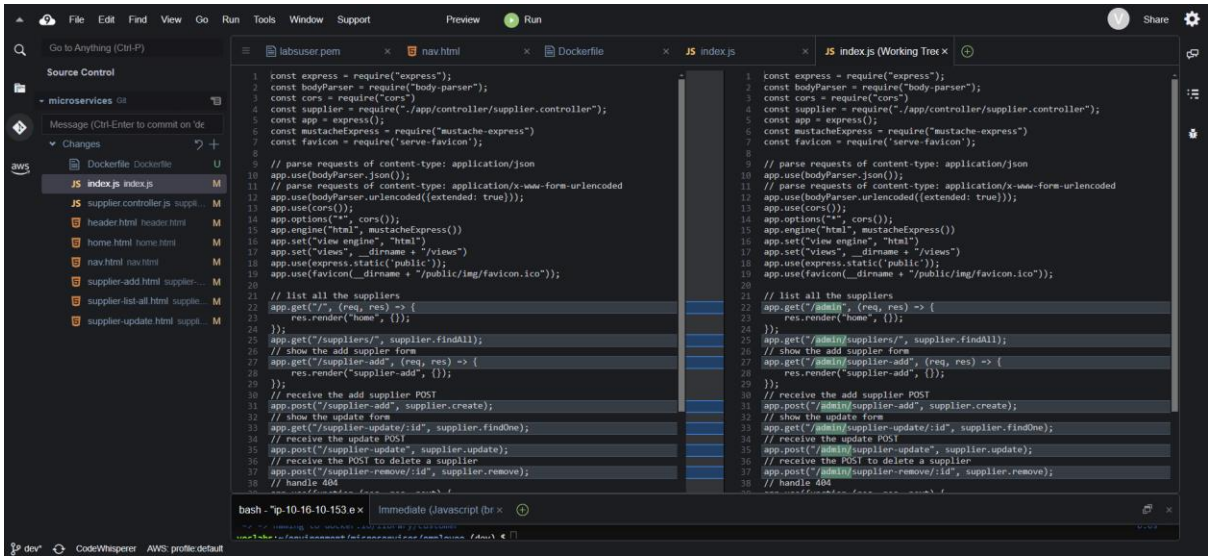


Notice the changes list, which indicates which files were changed since you last checked files in to the remote Git repository (CodeCommit).



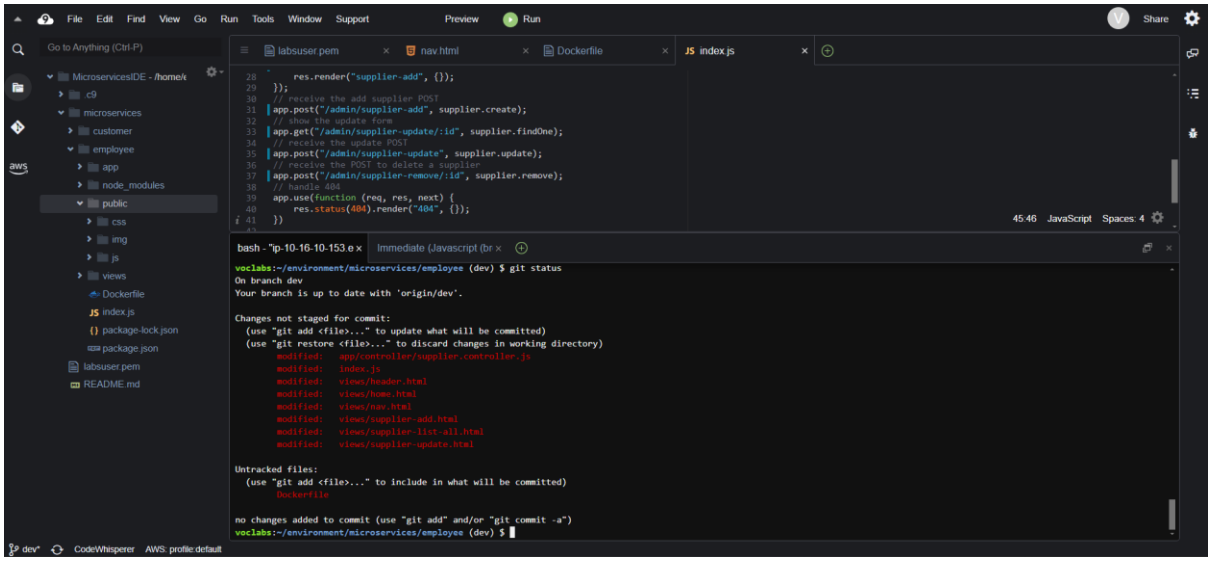
- **Choose one of the files that was modified, such as index.js, to compare the version fro the last Git commit to the latest version. Changes are highlighted.**

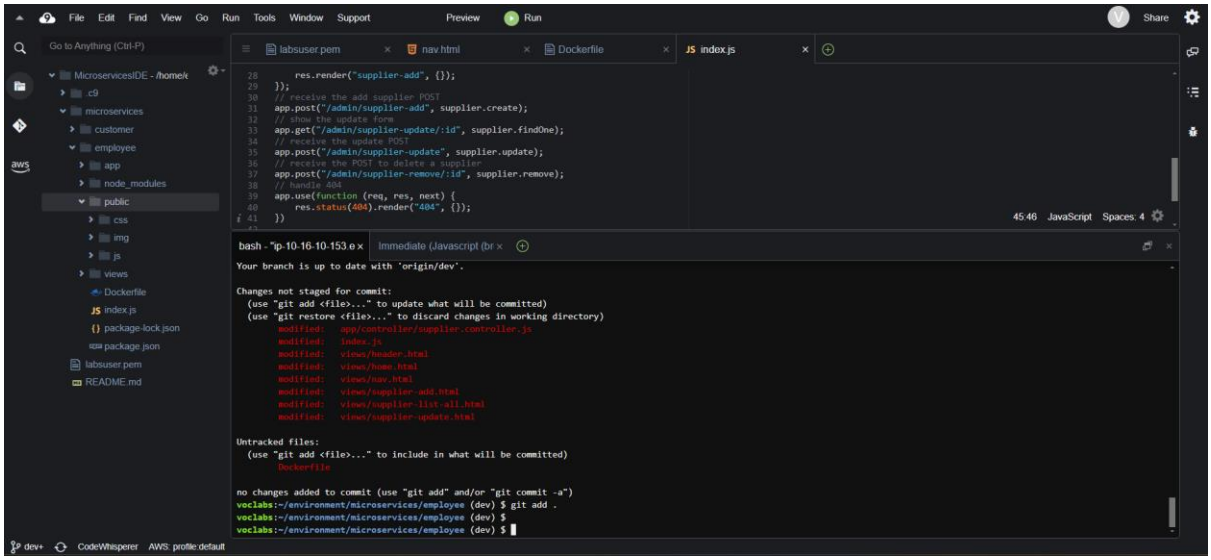
This demonstrates a benefit of using a source control system and a Git-compatible IDE such as AWS Cloud9. You can review your code changes prior to committing.

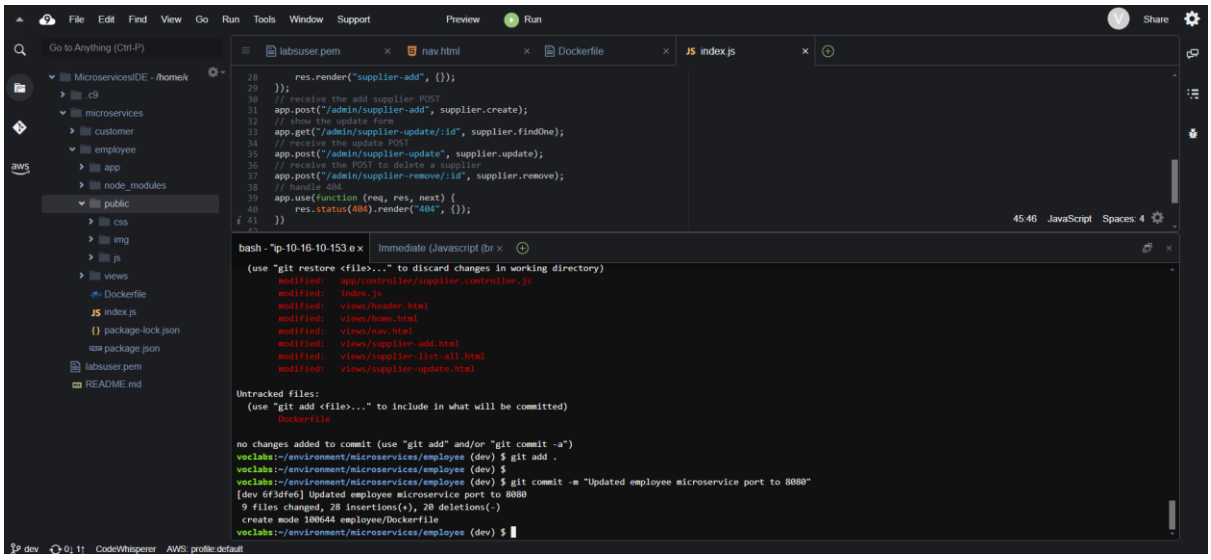


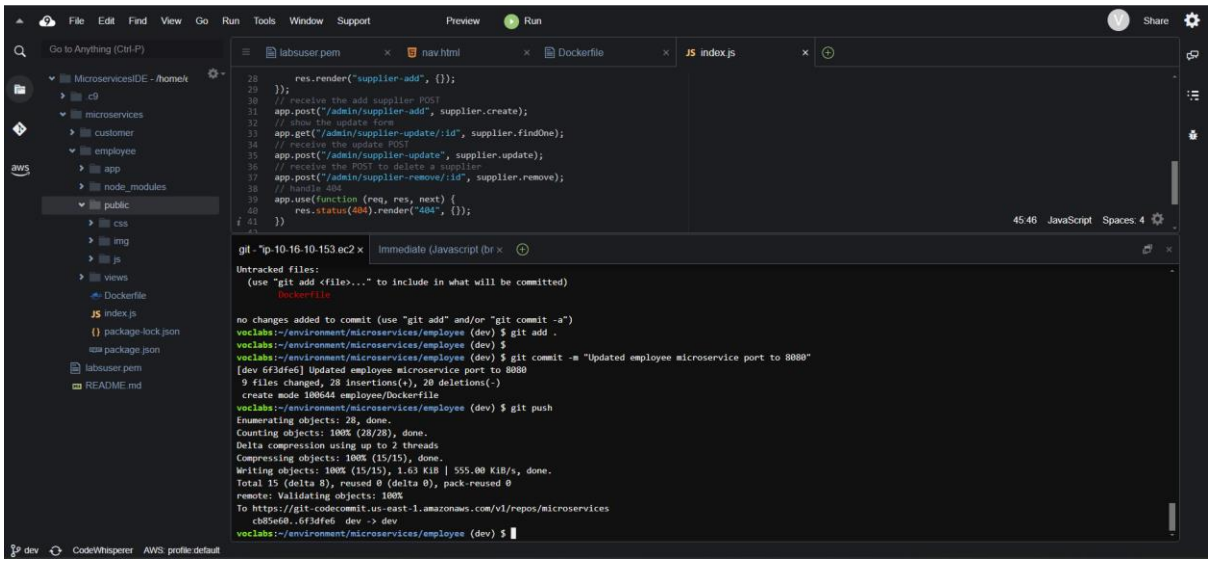
2. **Check your changes into CodeCommit.**

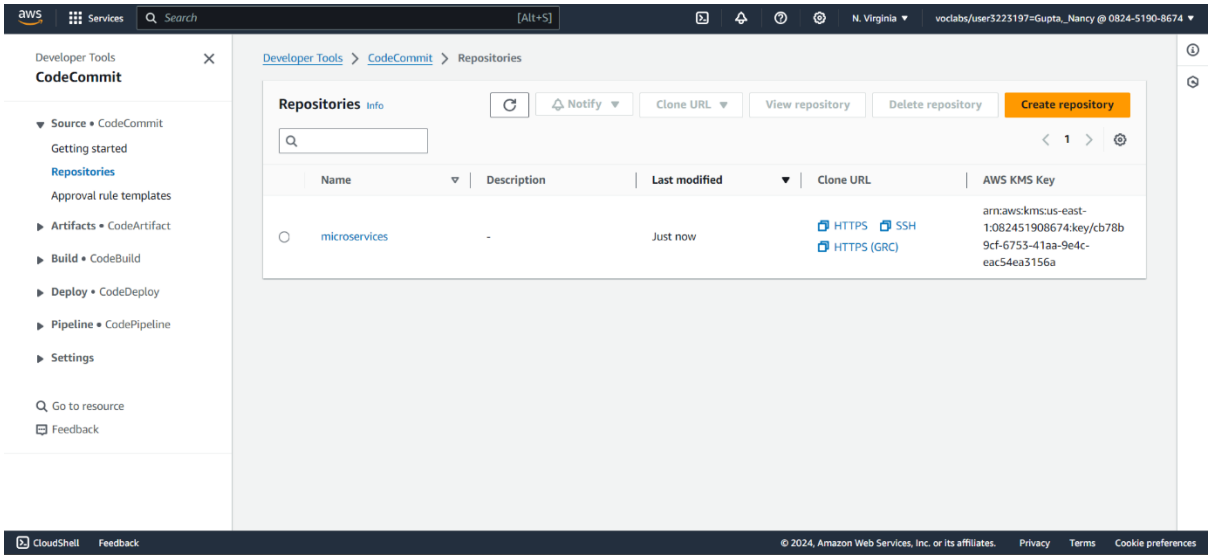
Tip: You performed this same type of action in task 4.3. You can accomplish this step by using the Git source control panel, or you can use the git commit and git push commands in the terminal.

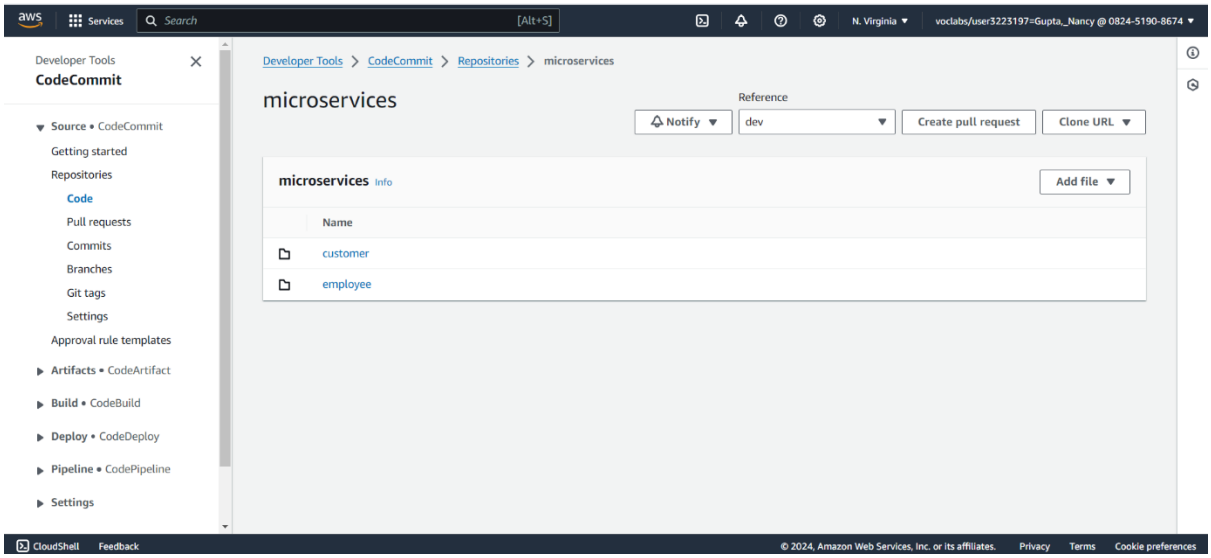


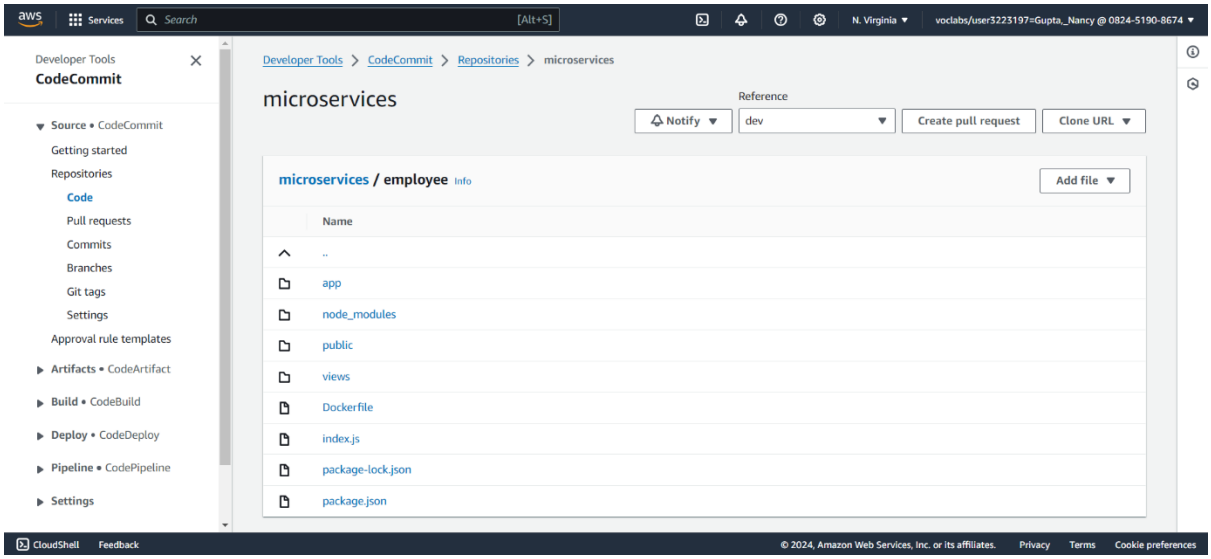

File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
labsuser.pem
nav.html
Dockerfile
index.js
MicroservicesIDE - /home/e
.c9
microservices
customer
employee
app
node_modules
public
css
img
js
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
res.render("supplier-add", {});
});
// receive the add supplier POST
app.post("/admin/supplier-add", supplier.create);
// show the update form
app.get("/admin/supplier-update/:id", supplier.findOne);
// receive the update POST
app.post("/admin/supplier-update", supplier.update);
// receive the POST to delete a supplier
app.post("/admin/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
res.status(404).render("404", {});
})
45:46 JavaScript Spaces: 4
bash - "ip-10-16-10-153.e
Immediate (Javascript (br
Your branch is up to date with 'origin/dev'.
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: app/controller/supplier.controller.js
modified: index.js
modified: views/header.html
modified: views/home.html
modified: views/nav.html
modified: views/supplier-add.html
modified: views/supplier-list-all.html
modified: views/supplier-update.html
Untracked files:
(use "git add <file>..." to include in what will be committed)
Dockerfile
no changes added to commit (use "git add" and/or "git commit -a")
voclabs:~/environment/microservices/employee (dev) $ git add .
voclabs:~/environment/microservices/employee (dev) $
voclabs:~/environment/microservices/employee (dev) $
dev+
CodeWhisperer
AWS: profile:default

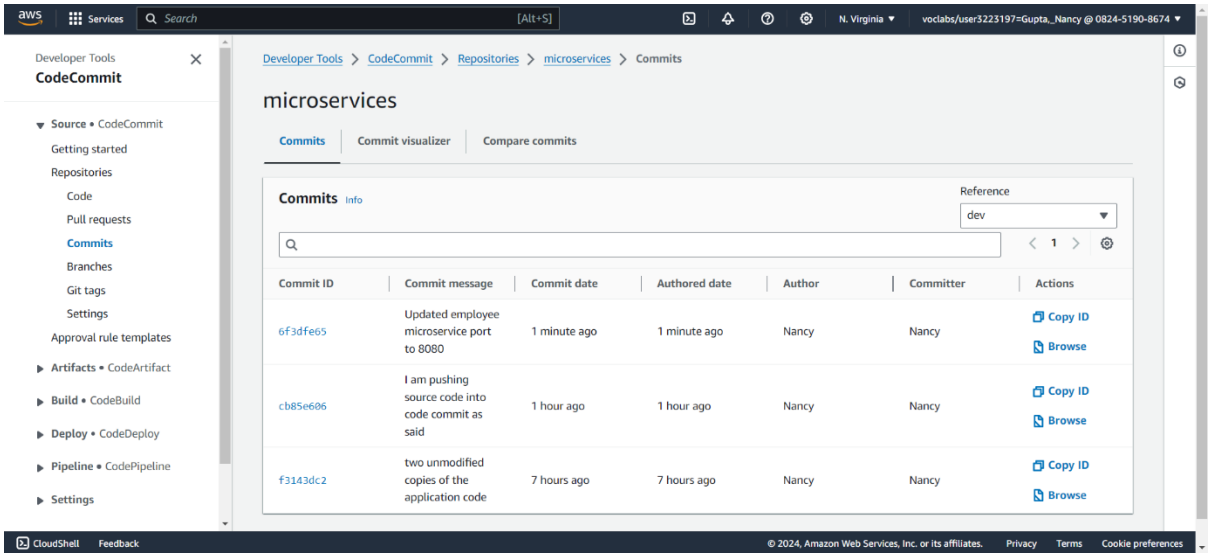

File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
labsuser.pem
nav.html
Dockerfile
index.js
MicroservicesIDE - /home/e
.c9
microservices
customer
employee
app
node_modules
public
css
img
js
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
res.render("supplier-add", {});
});
// receive the add supplier POST
app.post("/admin/supplier-add", supplier.create);
// show the update form
app.get("/admin/supplier-update/:id", supplier.findOne);
// receive the update POST
app.post("/admin/supplier-update", supplier.update);
// receive the POST to delete a supplier
app.post("/admin/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
res.status(404).render("404", {});
})
45:46 JavaScript Spaces: 4
bash - "ip-10-16-10-153.e
Immediate (Javascript (br
(use "git restore <file>..." to discard changes in working directory)
modified: app/controller/supplier.controller.js
modified: index.js
modified: views/header.html
modified: views/home.html
modified: views/nav.html
modified: views/supplier-add.html
modified: views/supplier-list-all.html
modified: views/supplier-update.html
Untracked files:
(use "git add <file>..." to include in what will be committed)
Dockerfile
no changes added to commit (use "git add" and/or "git commit -a")
voclabs:~/environment/microservices/employee (dev) $ git add .
voclabs:~/environment/microservices/employee (dev) $
voclabs:~/environment/microservices/employee (dev) $ git commit -m "Updated employee microservice port to 8080"
[dev 6f3dfe6] Updated employee microservice port to 8080
9 files changed, 28 insertions(+), 20 deletions(-)
create mode 100644 employee/Dockerfile
voclabs:~/environment/microservices/employee (dev) $
dev
0↓ 1↑
CodeWhisperer
AWS: profile:default

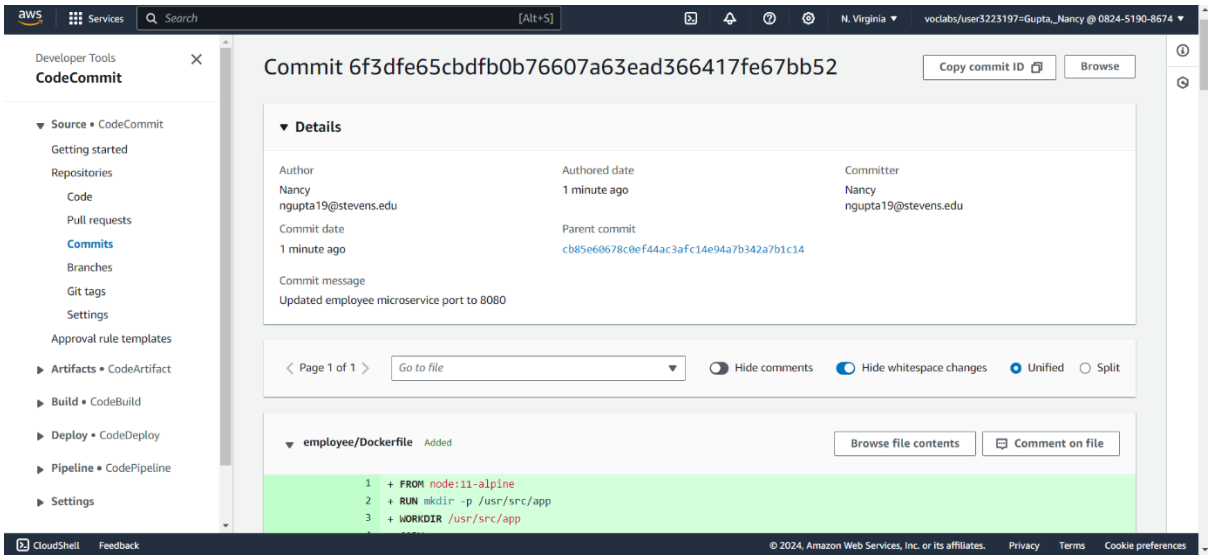

File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
labsuser.pem
nav.html
Dockerfile
index.js
MicroservicesIDE - /home/e
.c9
microservices
customer
employee
app
node_modules
public
css
img
js
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
res.render("supplier-add", {});
});
// receive the add supplier POST
app.post("/admin/supplier-add", supplier.create);
// show the update form
app.get("/admin/supplier-update/:id", supplier.findOne);
// receive the update POST
app.post("/admin/supplier-update", supplier.update);
// receive the POST to delete a supplier
app.post("/admin/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
res.status(404).render("404", {});
})
45:46 JavaScript Spaces: 4
git - "ip-10-16-10-153.ec2
Immediate (Javascript (br
Untracked files:
(use "git add <file>..." to include in what will be committed)
Dockerfile
no changes added to commit (use "git add" and/or "git commit -a")
voclabs:~/environment/microservices/employee (dev) $ git add .
voclabs:~/environment/microservices/employee (dev) $
voclabs:~/environment/microservices/employee (dev) $ git commit -m "Updated employee microservice port to 8080"
[dev 6f3dfe6] Updated employee microservice port to 8080
9 files changed, 28 insertions(+), 20 deletions(-)
create mode 100644 employee/Dockerfile
voclabs:~/environment/microservices/employee (dev) $ git push
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 1.63 KiB | 555.00 KiB/s, done.
Total 15 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
cb85e60..6f3dfe6 dev -> dev
voclabs:~/environment/microservices/employee (dev) $
dev
CodeWhisperer
AWS: profile:default

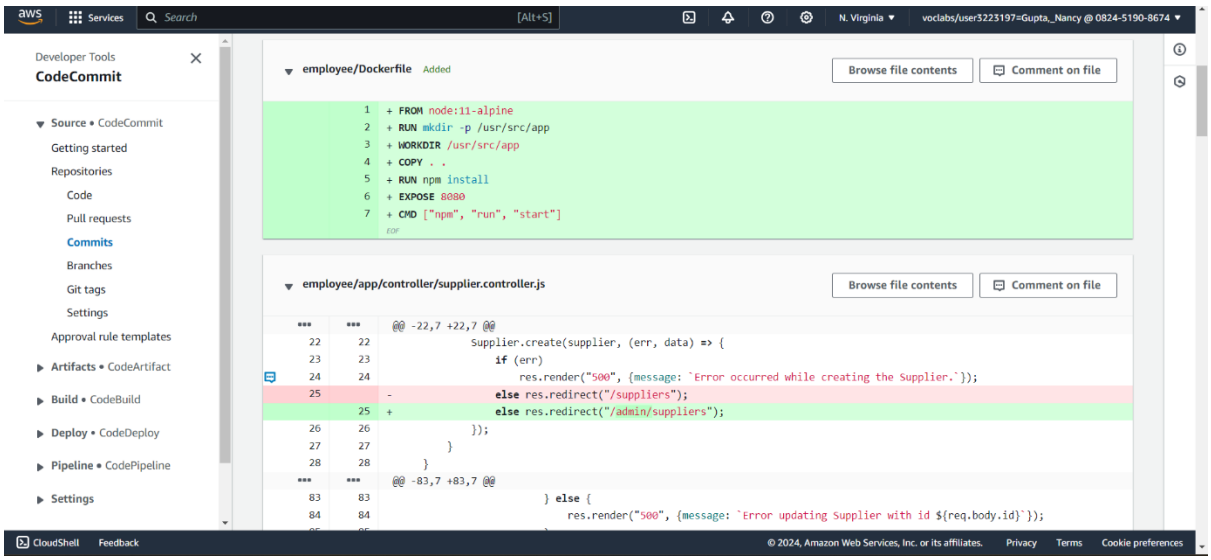

Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Developer Tools > CodeCommit > Repositories
Repositories Info
Notify
Clone URL
View repository
Delete repository
Create repository
Name
Description
Last modified
Clone URL
AWS KMS Key
microservices
-
Just now
HTTPS
SSH
HTTPS (GRC)
arn:aws:kms:us-east-1:082451908674:key/cb78b9cf-6753-41aa-9e4c-eac54ea3156a
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

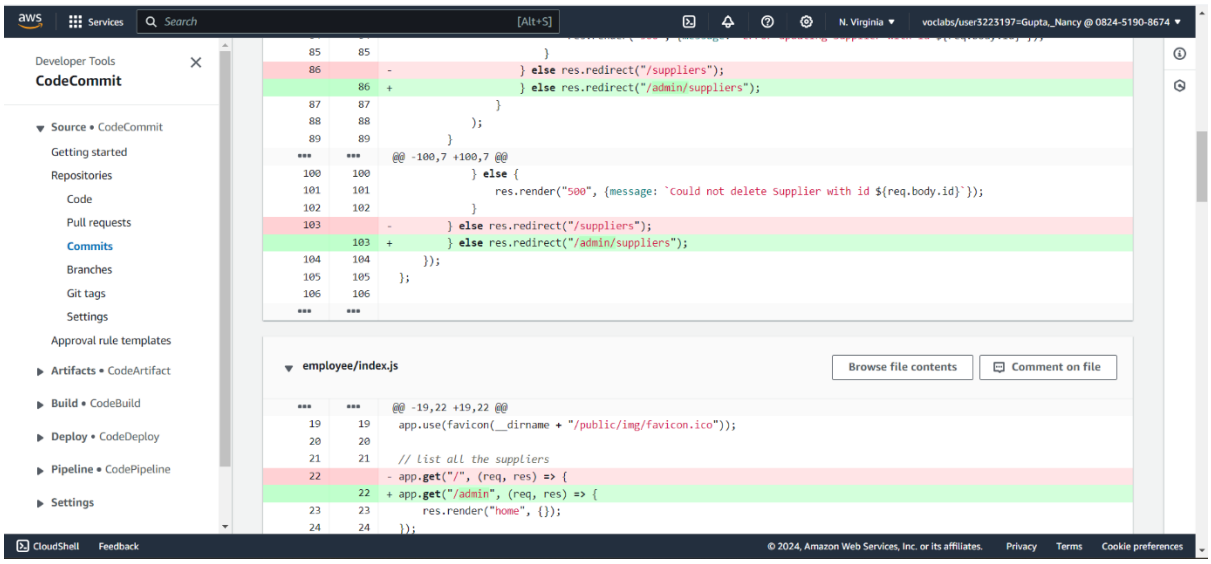

Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Developer Tools > CodeCommit > Repositories > microservices
microservices
Reference
Notify
dev
Create pull request
Clone URL
microservices Info
Add file
Name
customer
employee
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

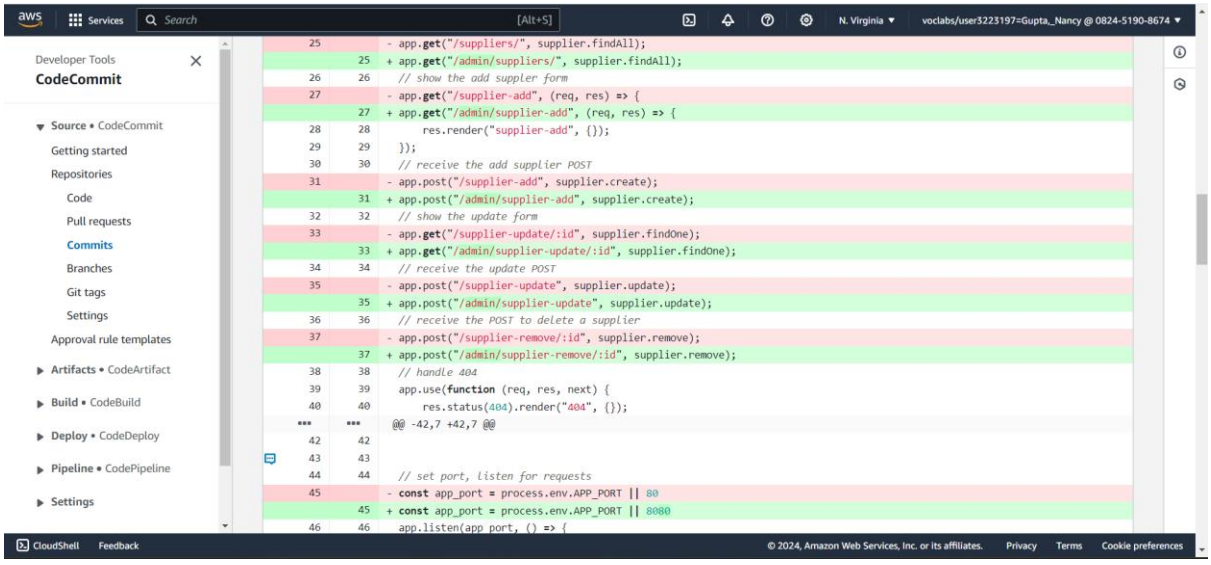

Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Developer Tools > CodeCommit > Repositories > microservices
microservices
Reference
Notify
dev
Create pull request
Clone URL
microservices / employee Info
Add file
Name
..
app
node_modules
public
views
Dockerfile
index.js
package-lock.json
package.json
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

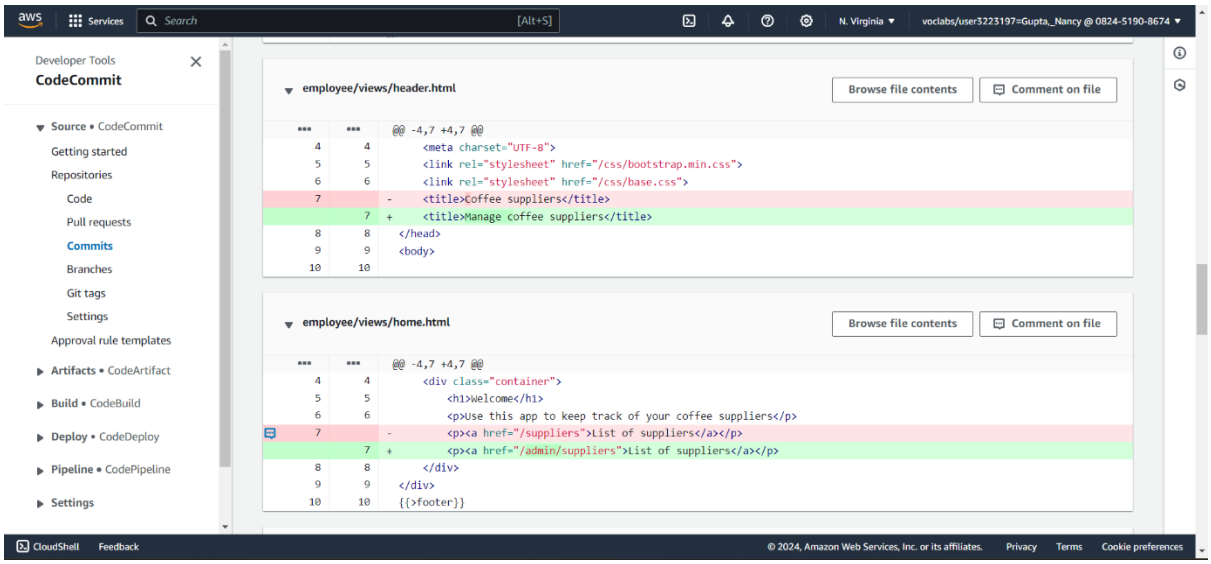

Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Developer Tools > CodeCommit > Repositories > microservices > Commits
microservices
Commits
Commit visualizer
Compare commits
Commits Info
Reference
dev
Commit ID
Commit message
Commit date
Authored date
Author
Committer
Actions
6f3dfe65
Updated employee microservice port to 8080
1 minute ago
1 minute ago
Nancy
Nancy
Copy ID
Browse
cb85e606
I am pushing source code into code commit as said
1 hour ago
1 hour ago
Nancy
Nancy
Copy ID
Browse
f3143dc2
two unmodified copies of the application code
7 hours ago
7 hours ago
Nancy
Nancy
Copy ID
Browse
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Commit 6f3dfe65cbdfb0b76607a63ead366417fe67bb52
Copy commit ID
Browse
Details
Author
Nancy
ngupta19@stevens.edu
Authored date
1 minute ago
Committer
Nancy
ngupta19@stevens.edu
Commit date
1 minute ago
Parent commit
cb85e60678c0ef44ac3afc14e94a7b342a7b1c14
Commit message
Updated employee microservice port to 8080
Page 1 of 1
Go to file
Hide comments
Hide whitespace changes
Unified
Split
employee/Dockerfile
Added
Browse file contents
Comment on file
1 + FROM node:11-alpine
2 + RUN mkdir -p /usr/src/app
3 + WORKDIR /usr/src/app
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
employee/Dockerfile
Added
Browse file contents
Comment on file
1 + FROM node:11-alpine
2 + RUN mkdir -p /usr/src/app
3 + WORKDIR /usr/src/app
4 + COPY . .
5 + RUN npm install
6 + EXPOSE 8080
7 + CMD ["npm", "run", "start"]
EOF
employee/app/controller/supplier.controller.js
Browse file contents
Comment on file
@@ -22,7 +22,7 @@
Supplier.create(supplier, (err, data) => {
if (err)
res.render("500", {message: `Error occurred while creating the Supplier.`});
- else res.redirect("/suppliers");
+ else res.redirect("/admin/suppliers");
});
}
}
@@ -83,7 +83,7 @@
} else {
res.render("500", {message: `Error updating Supplier with id ${req.body.id}`});
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

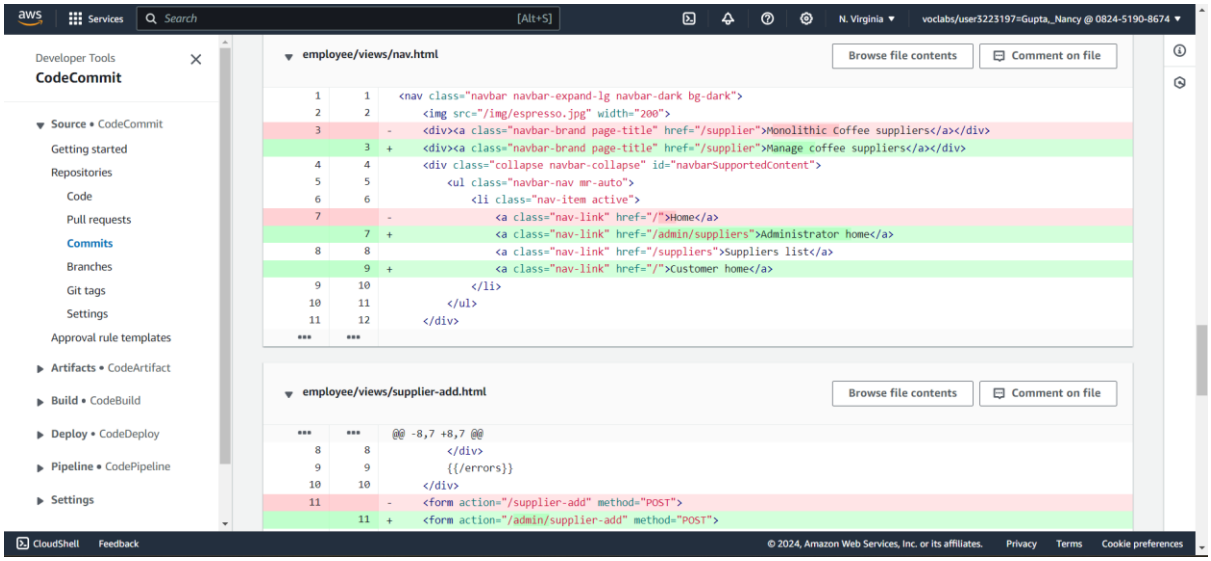

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
} else res.redirect("/suppliers");
} else res.redirect("/admin/suppliers");
@@ -100,7 +100,7 @@
} else {
res.render("500", {message: `Could not delete Supplier with id ${req.body.id}`});
}
} else res.redirect("/suppliers");
} else res.redirect("/admin/suppliers");
});
};
employee/index.js
Browse file contents
Comment on file
@@ -19,22 +19,22 @@
app.use(favicon(__dirname + "/public/img/favicon.ico"));
// list all the suppliers
app.get("/", (req, res) => {
app.get("/admin", (req, res) => {
res.render("home", {});
});
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

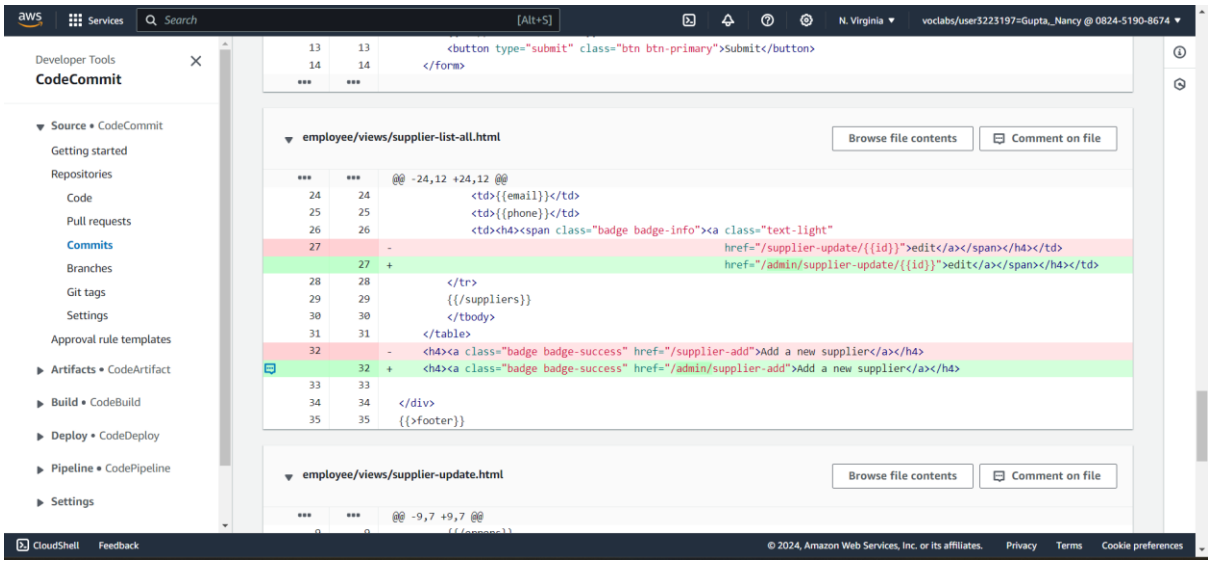

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
app.get("/suppliers/", supplier.findAll);
app.get("/admin/suppliers/", supplier.findAll);
// show the add suppler form
app.get("/supplier-add", (req, res) => {
app.get("/admin/supplier-add", (req, res) => {
res.render("supplier-add", {});
});
// receive the add supplier POST
app.post("/supplier-add", supplier.create);
app.post("/admin/supplier-add", supplier.create);
// show the update form
app.get("/supplier-update/:id", supplier.findOne);
app.get("/admin/supplier-update/:id", supplier.findOne);
// receive the update POST
app.post("/supplier-update", supplier.update);
app.post("/admin/supplier-update", supplier.update);
// receive the POST to delete a supplier
app.post("/supplier-remove/:id", supplier.remove);
app.post("/admin/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
res.status(404).render("404", {});
@@ -42,7 +42,7 @@
// set port, listen for requests
const app_port = process.env.APP_PORT || 80
const app_port = process.env.APP_PORT || 8080
app.listen(app_port, () => {
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

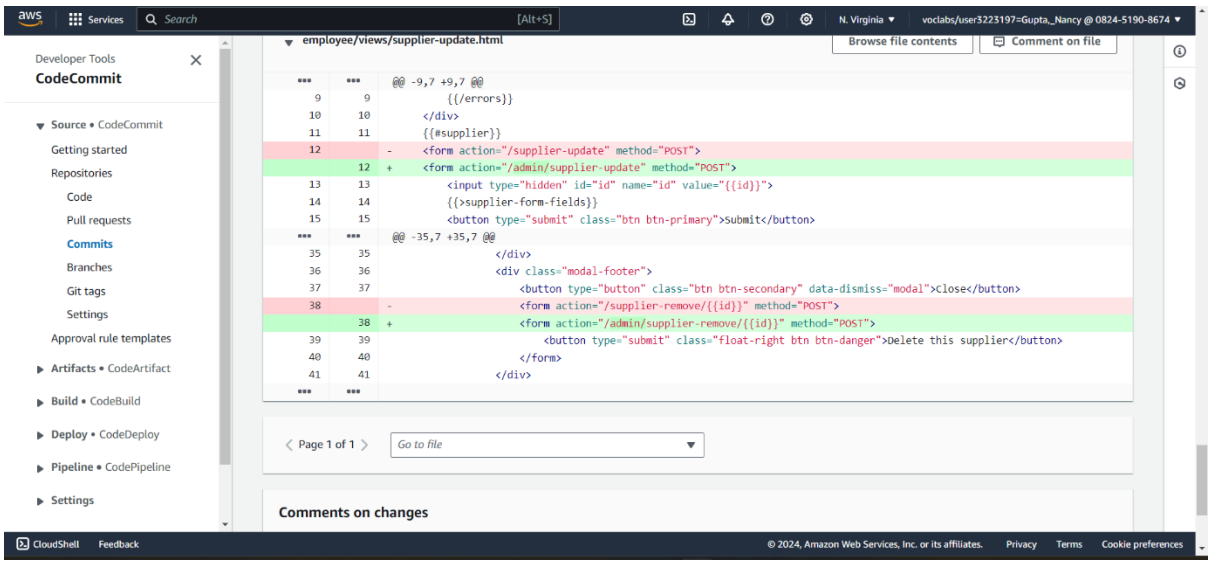

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
employee/views/header.html
Browse file contents
Comment on file
@@ -4,7 +4,7 @@
<meta charset="UTF-8">
<link rel="stylesheet" href="/css/bootstrap.min.css">
<link rel="stylesheet" href="/css/base.css">
<title>Coffee suppliers</title>
<title>Manage coffee suppliers</title>
</head>
<body>
employee/views/home.html
Browse file contents
Comment on file
@@ -4,7 +4,7 @@

<h1>Welcome</h1>
Use this app to keep track of your coffee suppliers
List of suppliers
List of suppliers

{{>footer}}
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences







Phase 5: Creating ECR repositories, an ECS cluster, task definitions, and AppSpec files

At this point, you have successfully implemented numerous solution requirements. You split the monolithic application into two microservices that can run as Docker containers. You have also verified that the containers support the needed application actions, such as adding, editing, and deleting entries from the database. The microservices architecture still uses Amazon RDS to store the coffee supplier entries.

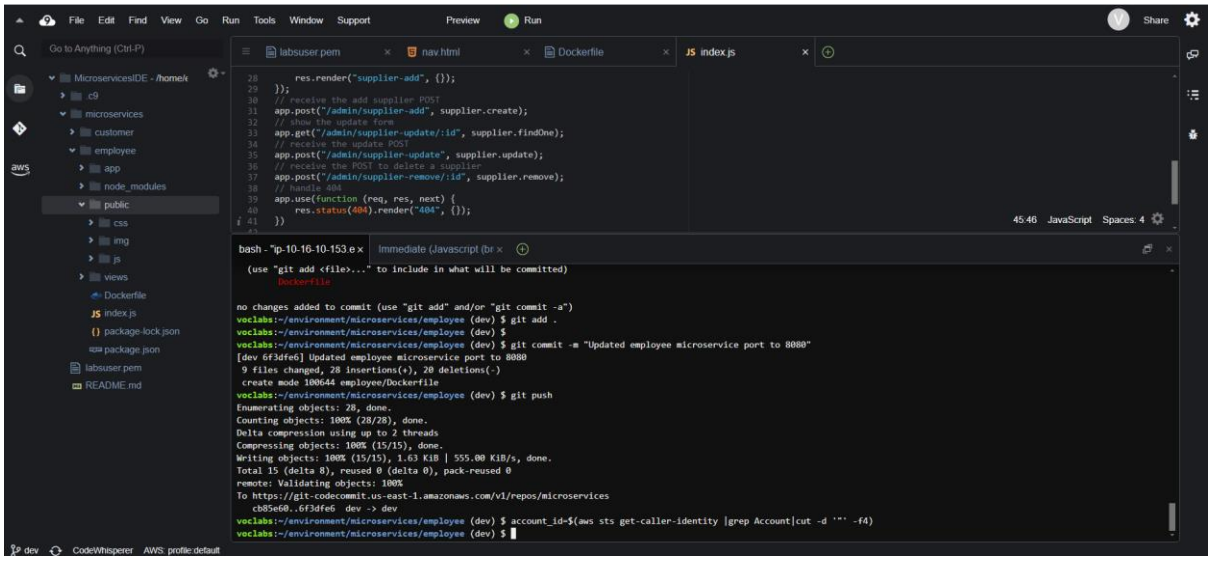
However, your work isn't finished. There are more solution requirements to implement. The containers are able to run on the AWS Cloud9 instance, but that isn't a scalable deployment architecture. You need the ability to scale the number of containers that run on each microservice up and down depending on need. Also, you need to have a load balancer to route traffic to the appropriate microservice. Finally, you need to be able to easily update each application microservice's codebase independently and roll those changes into production. In the remaining phases of the project, you will work to accomplish these solution requirements.

Task 5.1: Create ECR repositories and upload the Docker images

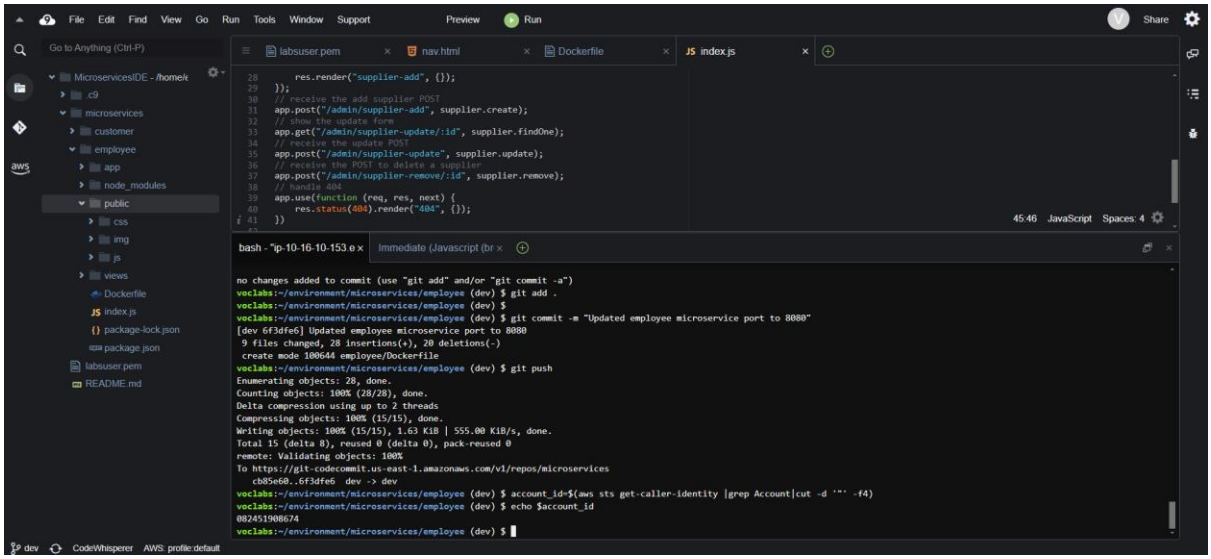
In this phase, you will upload the latest Docker images of the two microservices to separate Amazon ECR repositories.

1. To authorize your Docker client to connect to the Amazon ECR service, run the following commands:

account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)

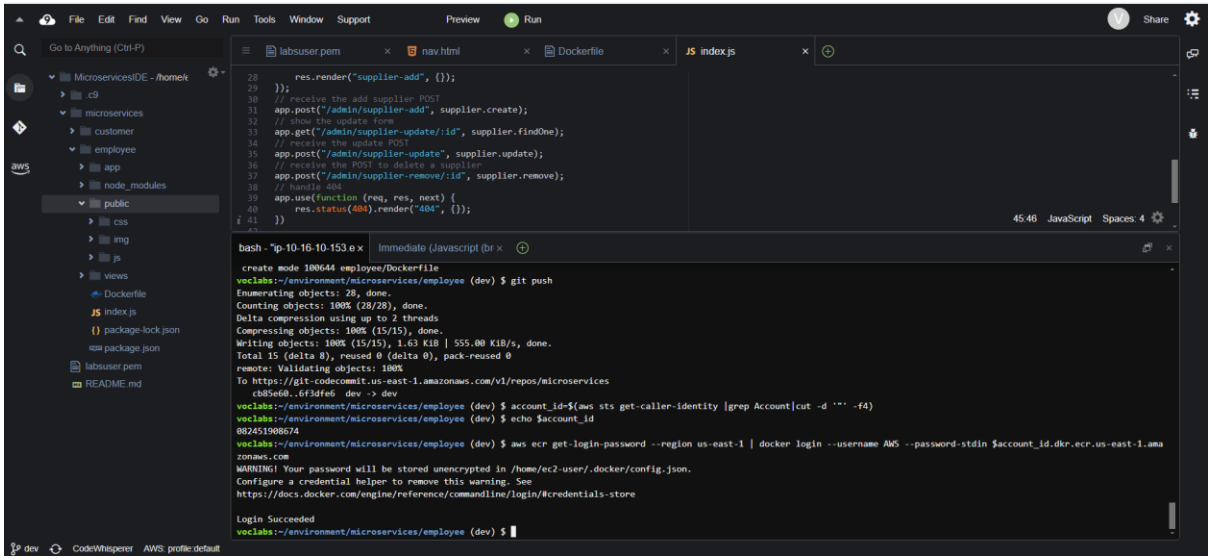


echo $account_id

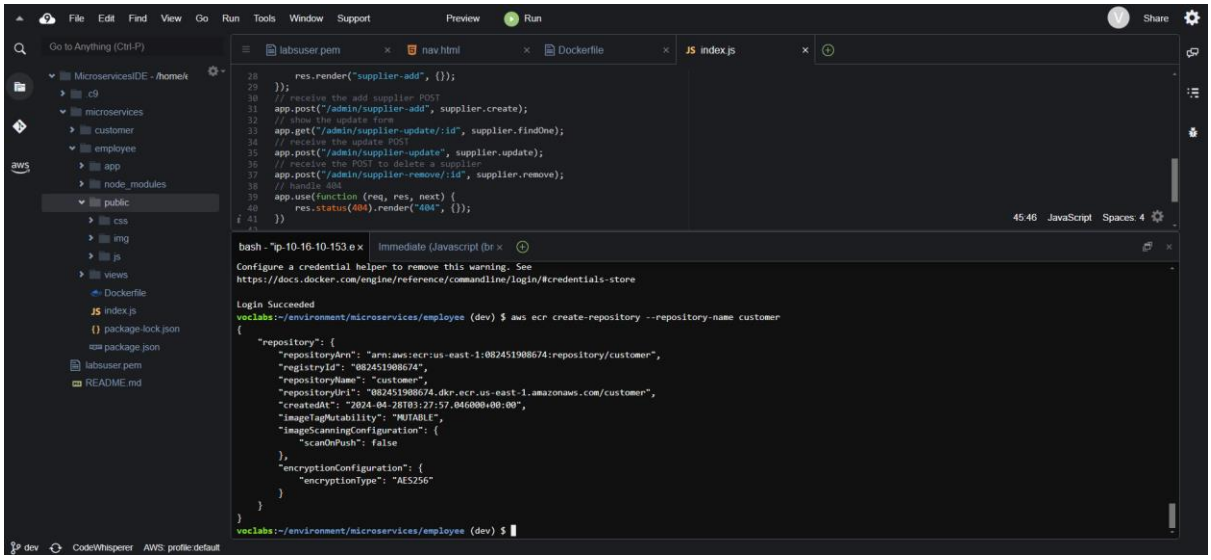


aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com

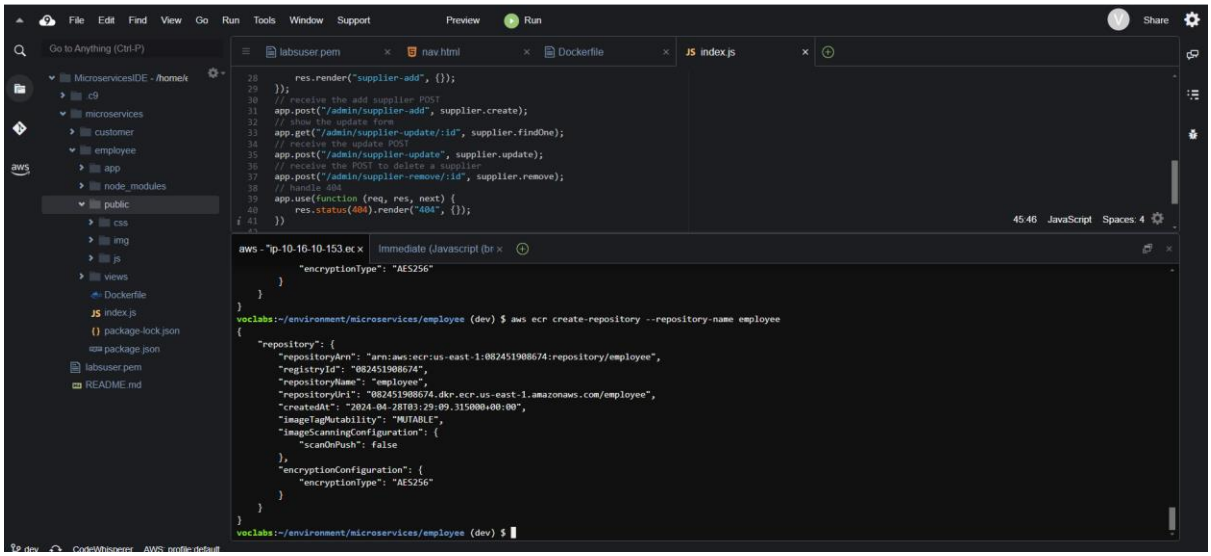
A message in the command output indicates that the login succeeded.



2. Create a separate private ECR repository for each microservice.
    - Name the first repository customer



- Name the second repository employee



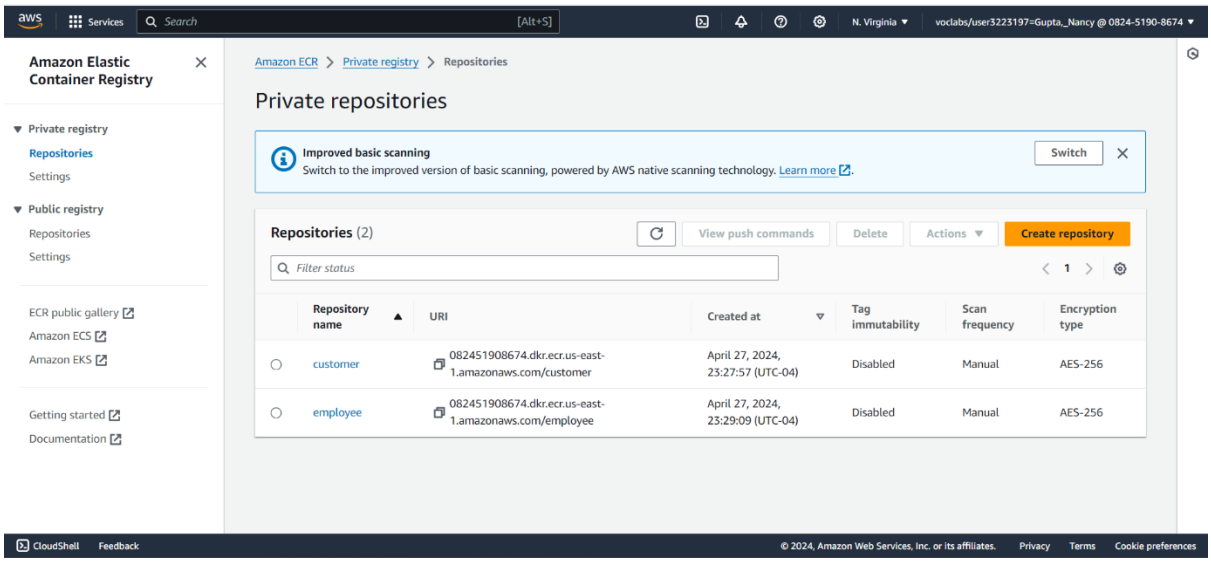
3. Set permissions on the *customer* ECR repository.

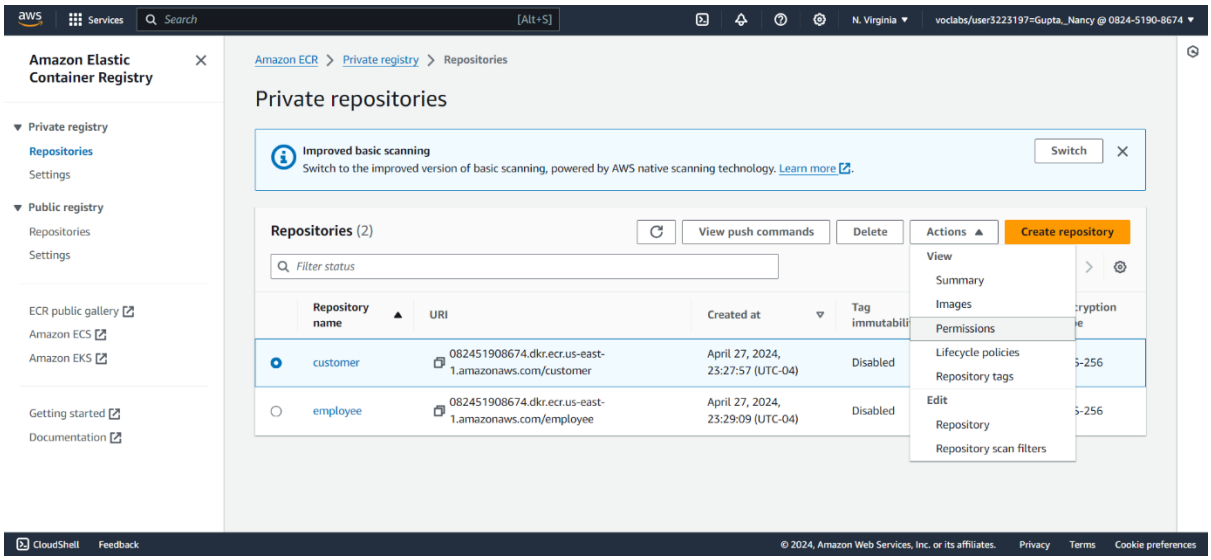
   - For information about editing the existing JSON policy, see Setting a Private Repository Statement in the Amazon ECR User Guide.

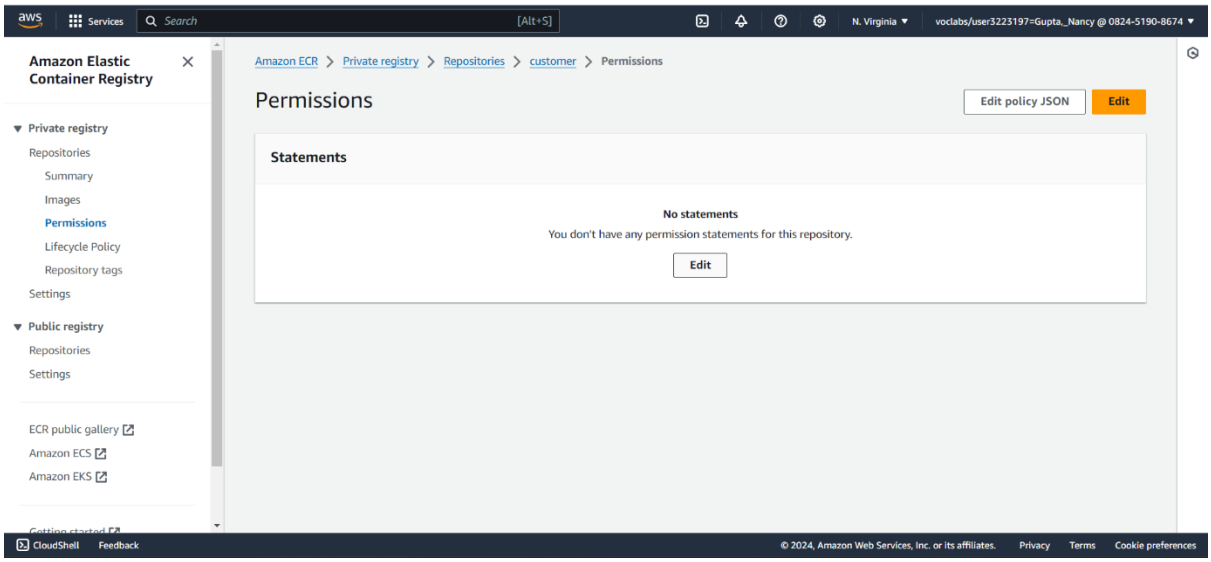
   - Replace the existing lines in the policy with the following:

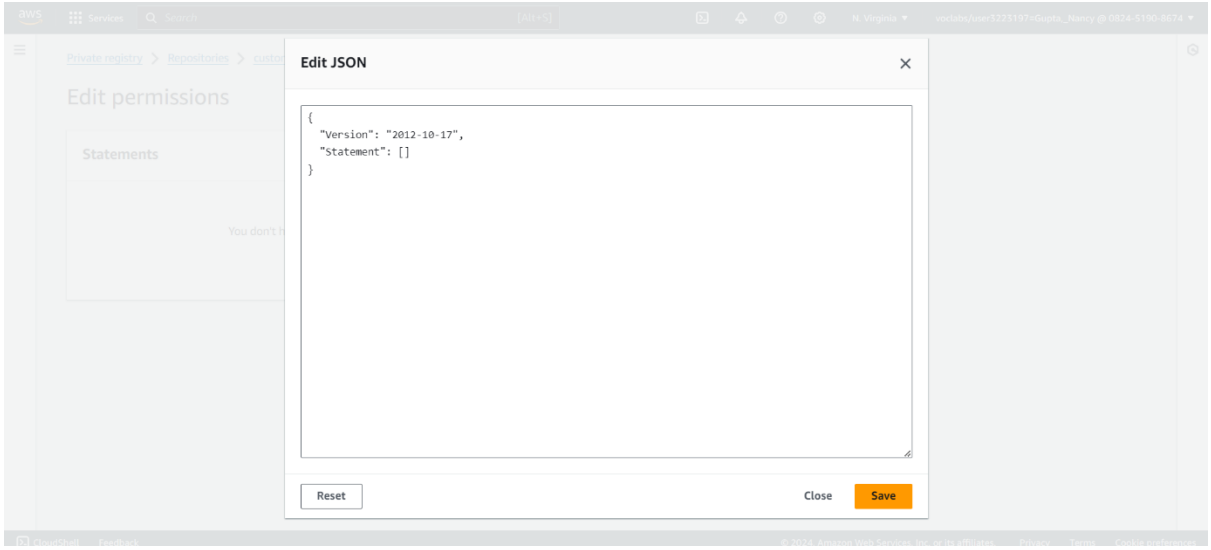
```
{
 "Version": "2008-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": "*",
   "Action": "ecr:*"
```

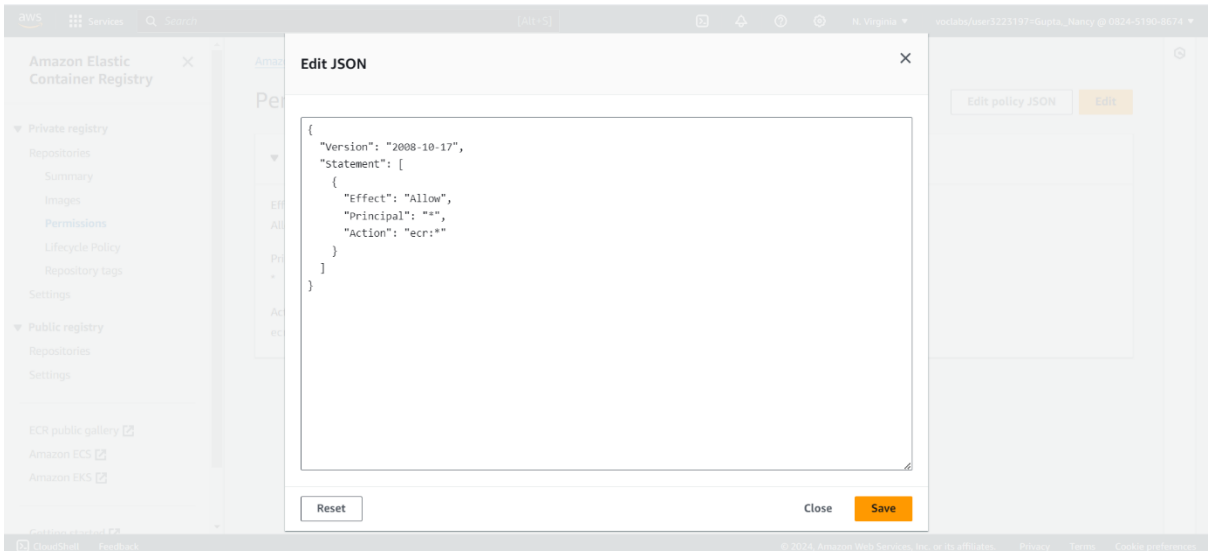
```
    }
  ]
}
```

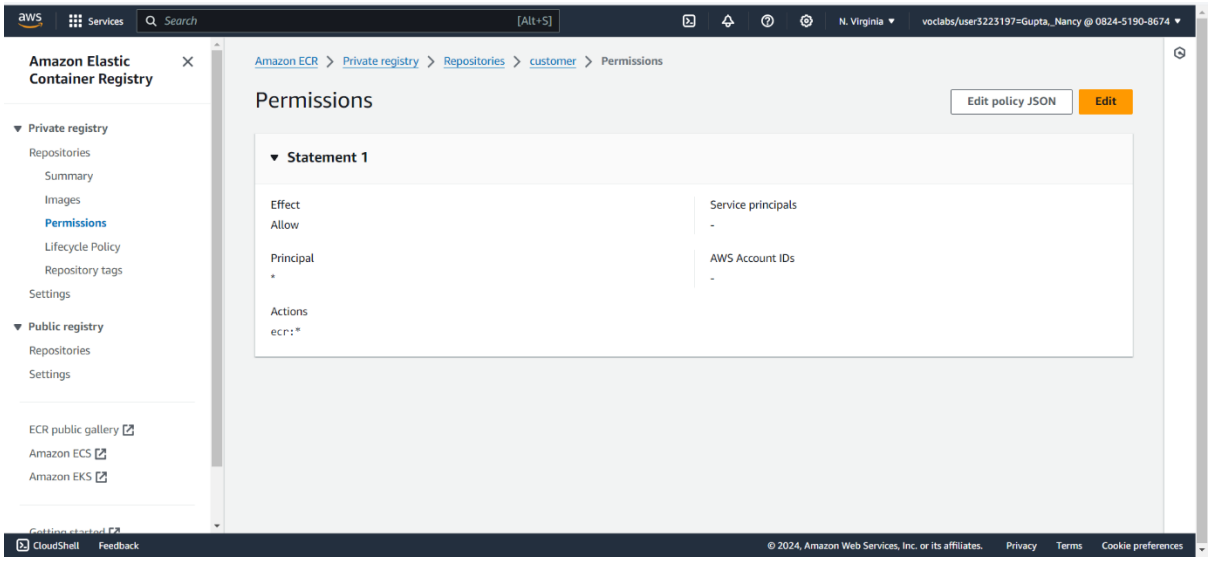

Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Amazon Elastic Container Registry
Amazon ECR > Private registry > Repositories
Private repositories
Private registry
Repositories
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
Improved basic scanning
Switch to the improved version of basic scanning, powered by AWS native scanning technology. Learn more.
Switch
Repositories (2)
View push commands
Delete
Actions
Create repository
Filter status
Repository name
URI
Created at
Tag immutability
Scan frequency
Encryption type
customer
082451908674.dkr.ecr.us-east-1.amazonaws.com/customer
April 27, 2024, 23:27:57 (UTC-04)
Disabled
Manual
AES-256
employee
082451908674.dkr.ecr.us-east-1.amazonaws.com/employee
April 27, 2024, 23:29:09 (UTC-04)
Disabled
Manual
AES-256
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Amazon Elastic Container Registry
Amazon ECR > Private registry > Repositories
Private repositories
Private registry
Repositories
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
Improved basic scanning
Switch to the improved version of basic scanning, powered by AWS native scanning technology. Learn more.
Switch
Repositories (2)
View push commands
Delete
Actions
Create repository
Filter status
View
Summary
Images
Permissions
Lifecycle policies
Repository tags
Edit
Repository
Repository scan filters
Repository name
URI
Created at
Tag immutabili
ryption
customer
082451908674.dkr.ecr.us-east-1.amazonaws.com/customer
April 27, 2024, 23:27:57 (UTC-04)
Disabled
employee
082451908674.dkr.ecr.us-east-1.amazonaws.com/employee
April 27, 2024, 23:29:09 (UTC-04)
Disabled
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

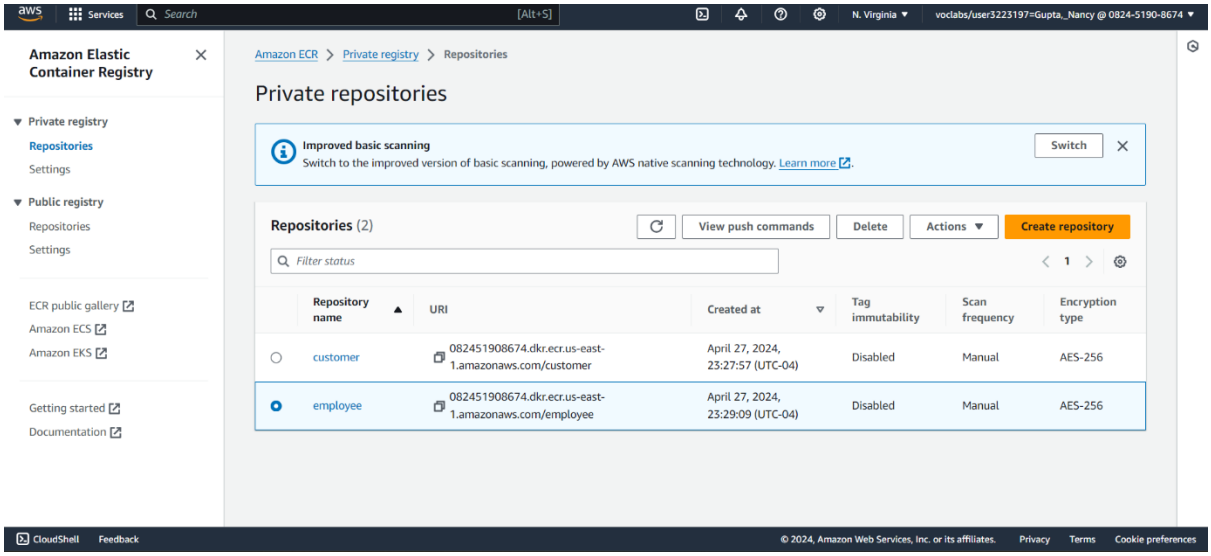

Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Amazon Elastic Container Registry
Private registry
Repositories
Summary
Images
Permissions
Lifecycle Policy
Repository tags
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Amazon ECR > Private registry > Repositories > customer > Permissions
Permissions
Edit policy JSON
Edit
Statements
No statements
You don't have any permission statements for this repository.
Edit
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

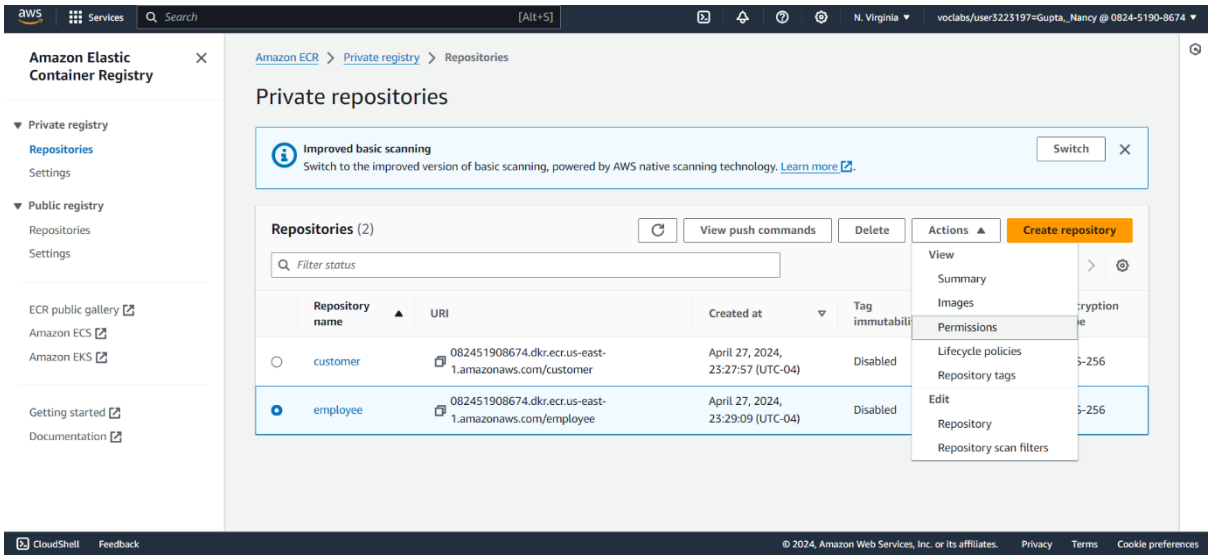

Edit JSON
{
"Version": "2012-10-17",
"Statement": []
}
Reset
Close
Save

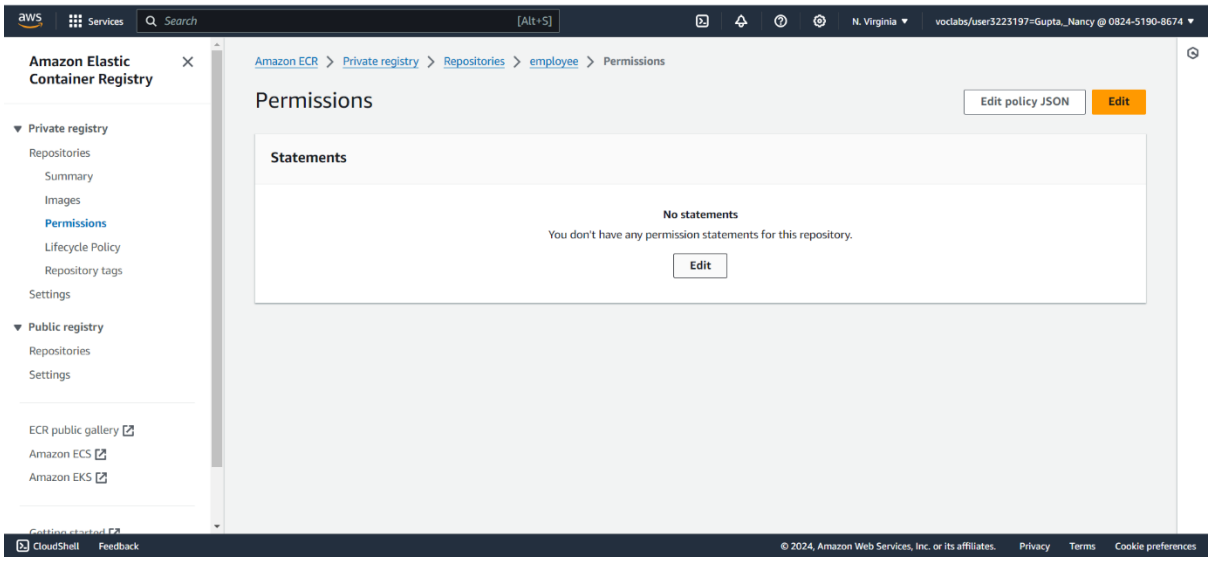

Edit JSON
{
"Version": "2008-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": "*",
"Action": "ecr:*"
}
]
}
Reset
Close
Save

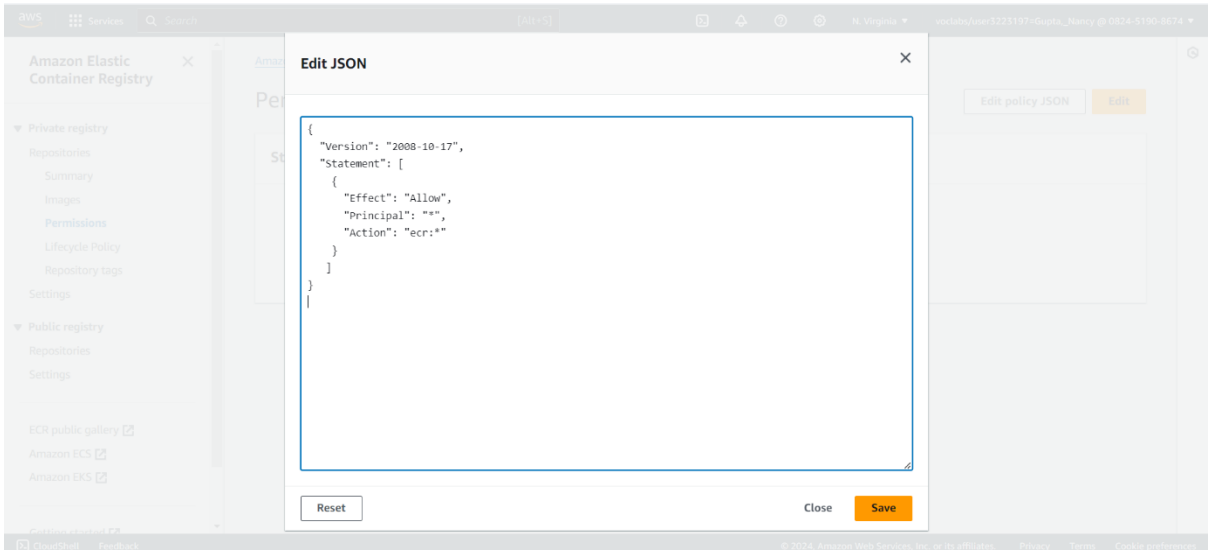


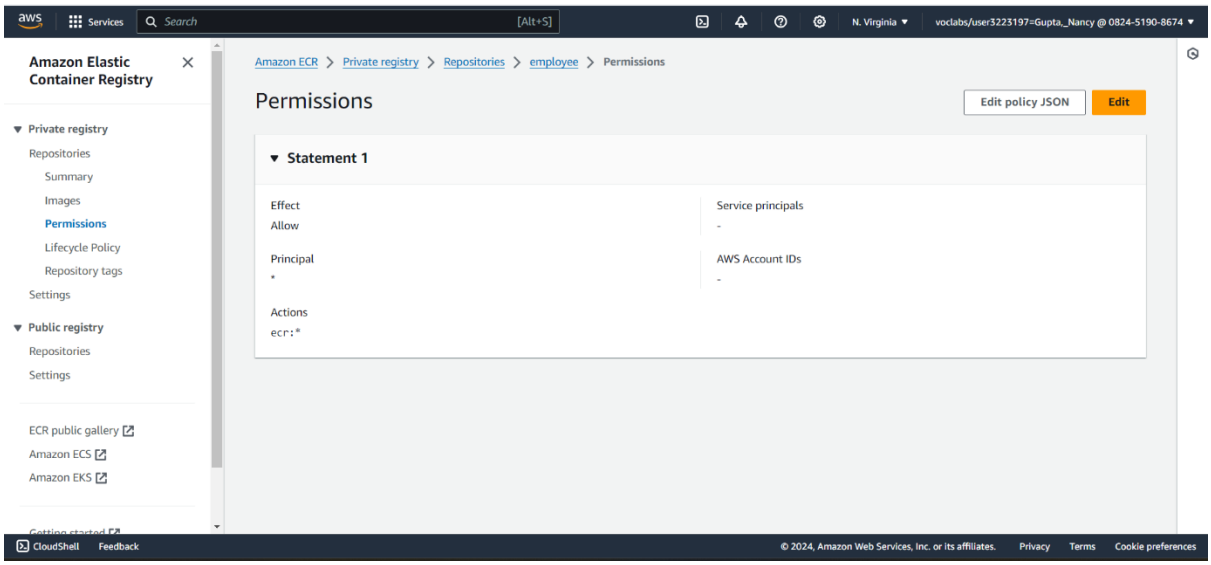
4. Use the same approach to set the same permissions on the *employee* ECR repository.





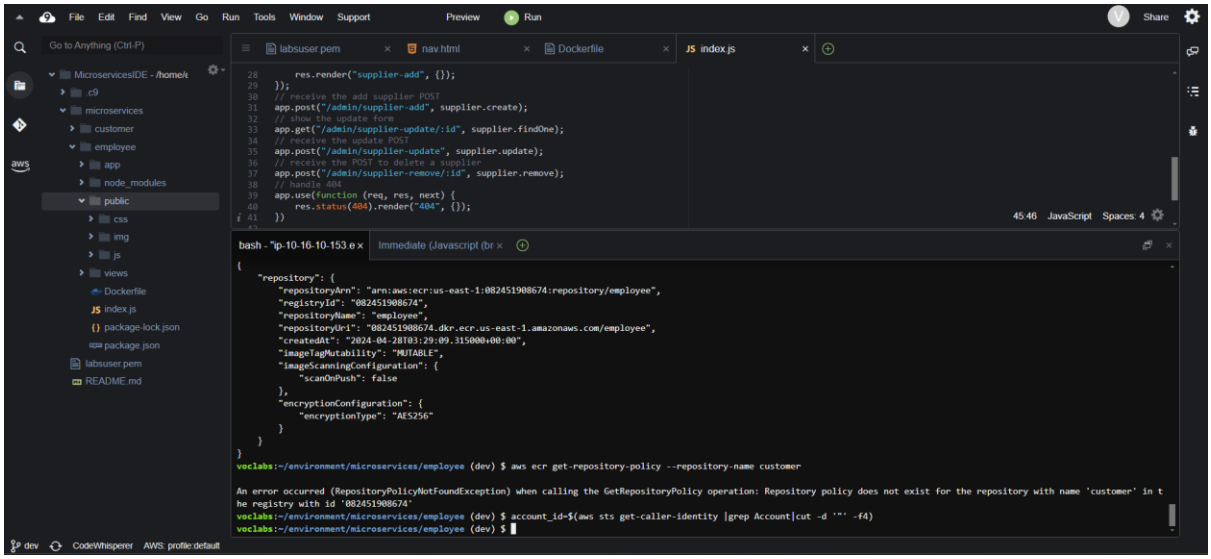






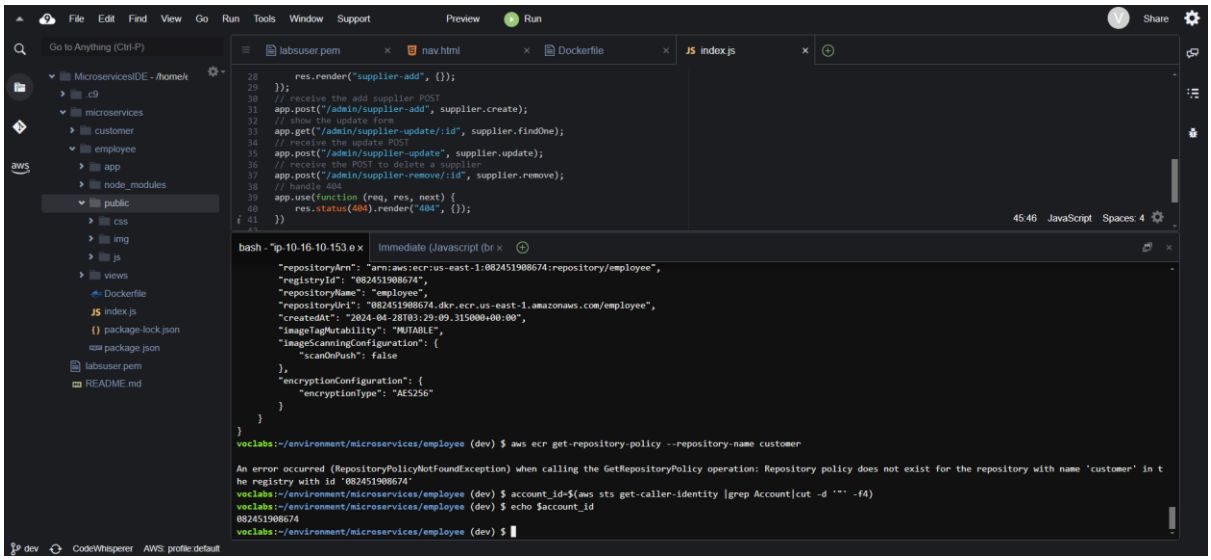
5. Tag the Docker images with your unique *registryId* (account ID) value to make it easier to manage and keep track of these images.
    - In the AWS Cloud9 IDE, run the following commands:

account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)



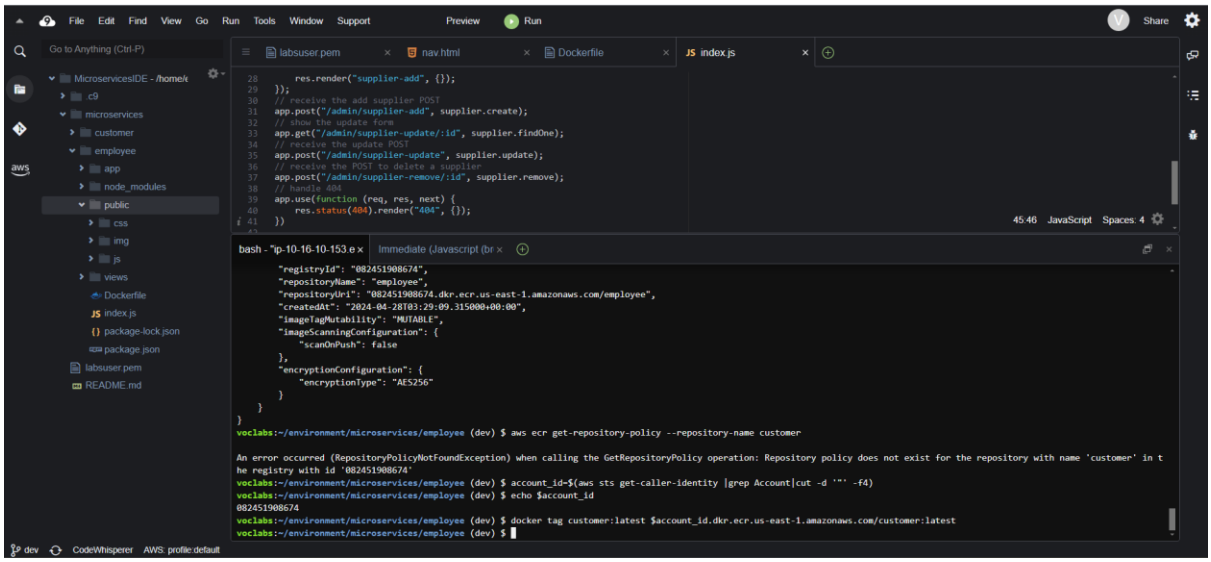
Verify that the account_id value is assigned to the $account_id variable

echo $account_id



Tag the customer image

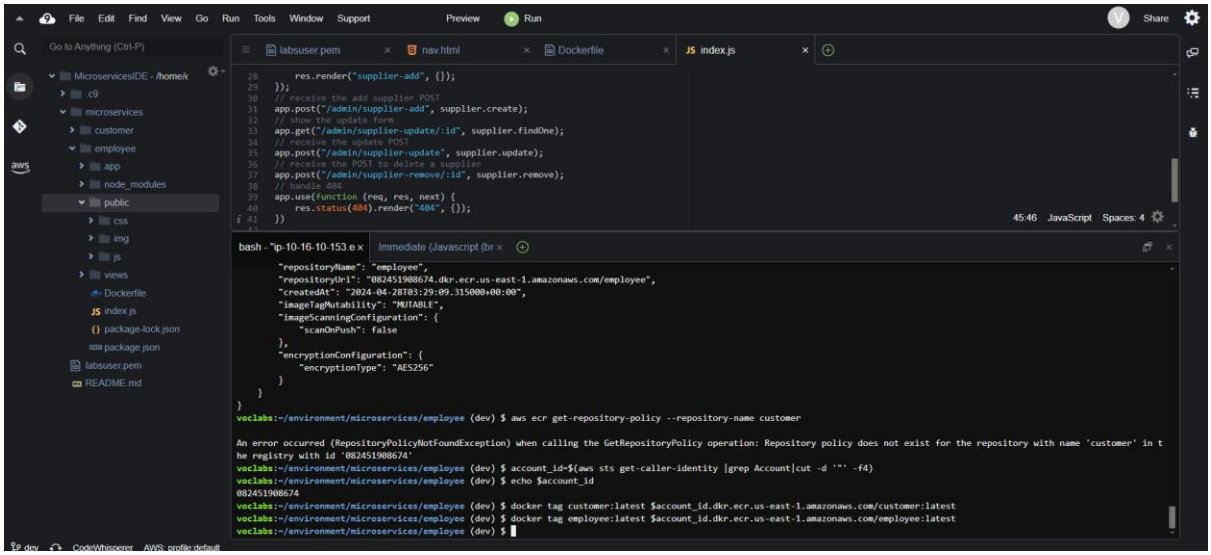
docker tag customer:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest



Tag the employee image

docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest

Note: The commands don't return output.



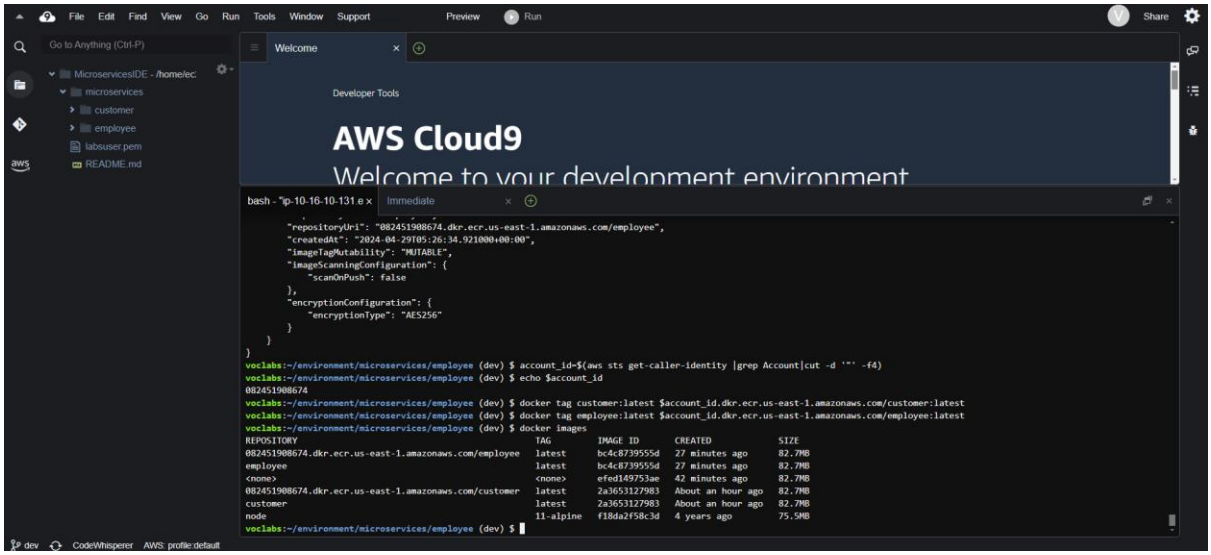
- Run the appropriate docker command to verify that the images exist and the tags were applied.

Tip: To find the command that you need to run, see Use the Docker Command Line in the Docker documentation.

Tip: The output of the command should be similar to the following image. Notice that the *latest* tag was applied and that the image names now include the remote repository name where you intend to store it:

Use the following command:

docker images

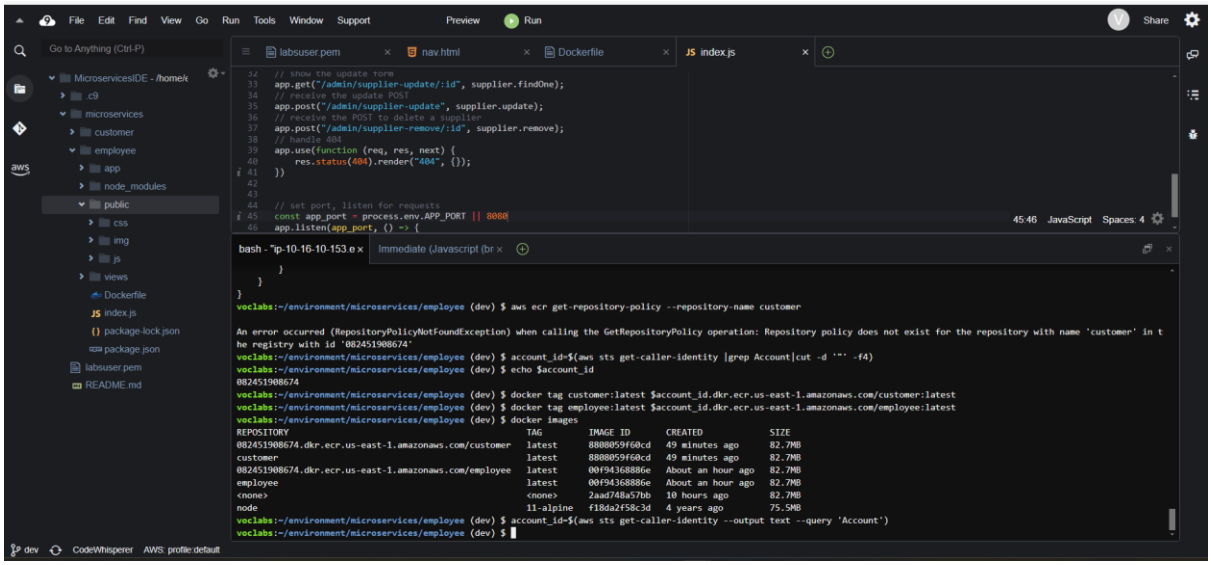


6. Run the appropriate docker command to push each of the Docker images to Amazon ECR.

Tip: To find the command that you need to run, see Use the Docker Command Line in the Docker documentation.

Tip: Before running the Docker commands, run the following command to set account_id as a variable in the terminal. Then, when you run the Docker commands, you can reference the account ID as $account_id.

account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)



Additional tip: The commands that you run should look like the following commands but with **REPLACE_ME** replaced with the correct command:

docker REPLACE_ME $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest

docker REPLACE_ME $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest

The output for *each* Docker command that you run to push each image to Amazon ECR should look similar to the following:

The push refers to repository [642015801240.dkr.ecr.us-east-1.amazonaws.com/node-app]

006e0ec54dba: Pushed

59762f95cb06: Pushed

22736f780b31: Pushed

d81d715330b7: Pushed

1dc7f3bb09a4: Pushed

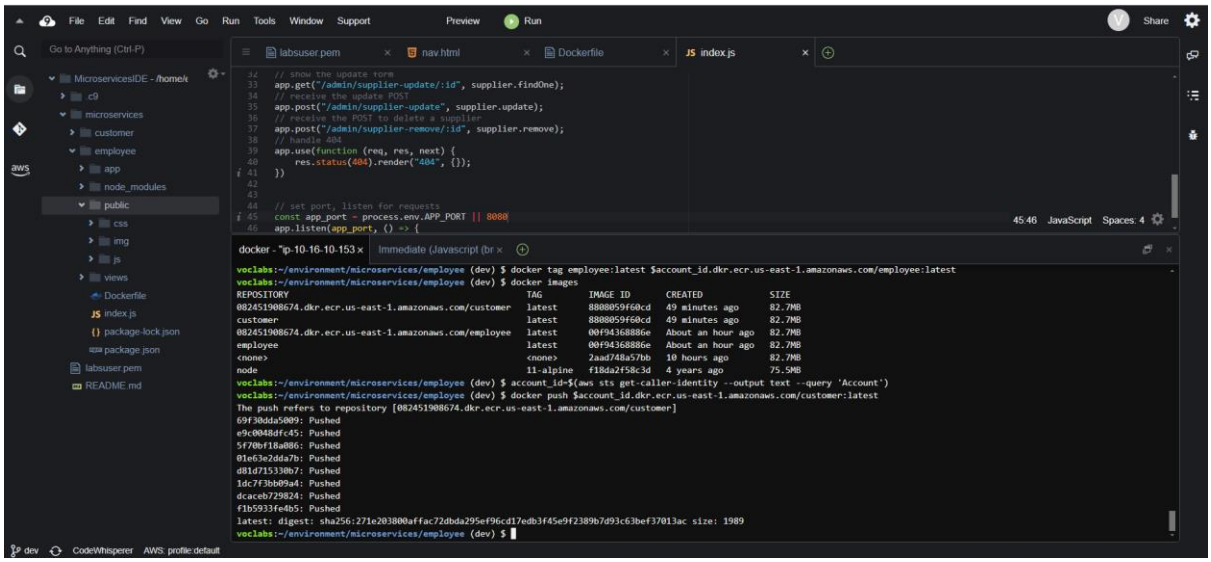
dcaceb729824: Pushed

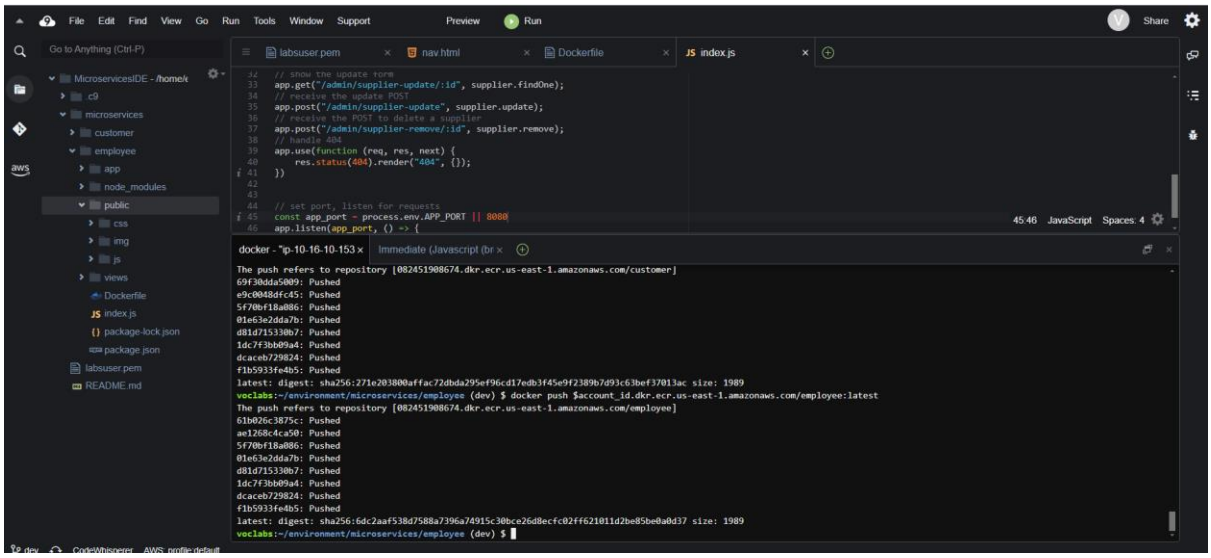
f1b5933fe4b5: Pushed

latest: digest: sha256:f75b60adddb8d6343b9dff690533a1cd1fbb34ccce6f861e84c857ba7a27b77d size: 1783

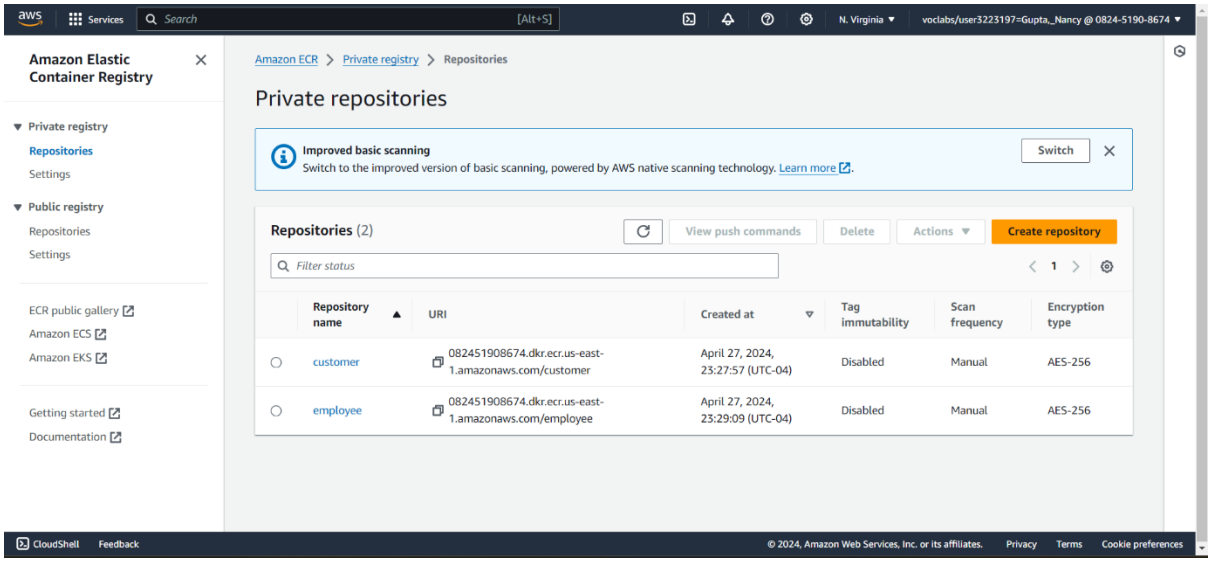
Use the following commands:
docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest

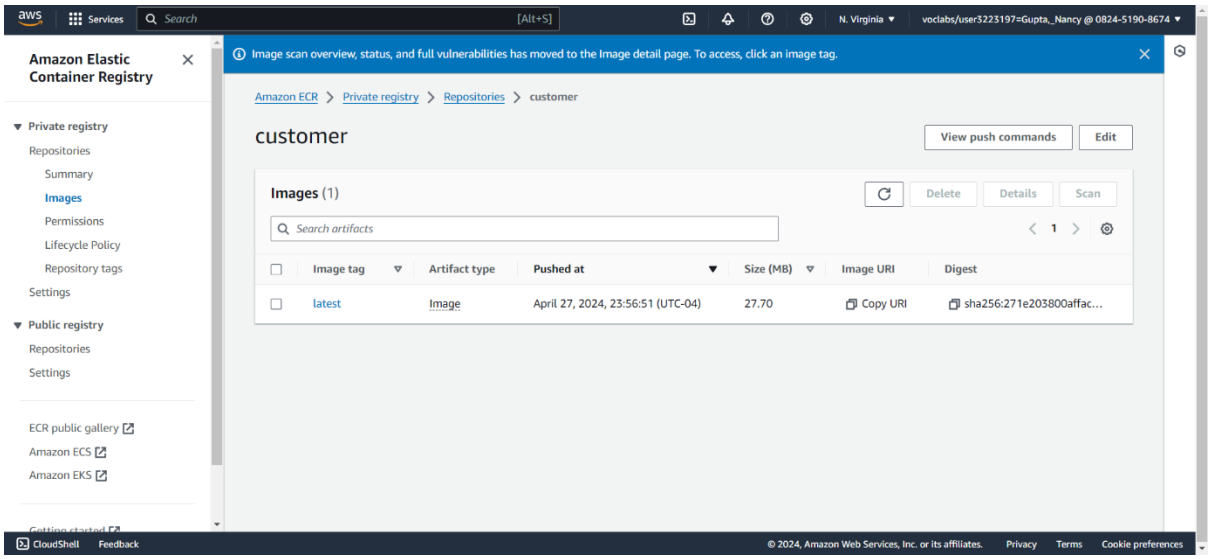
docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest

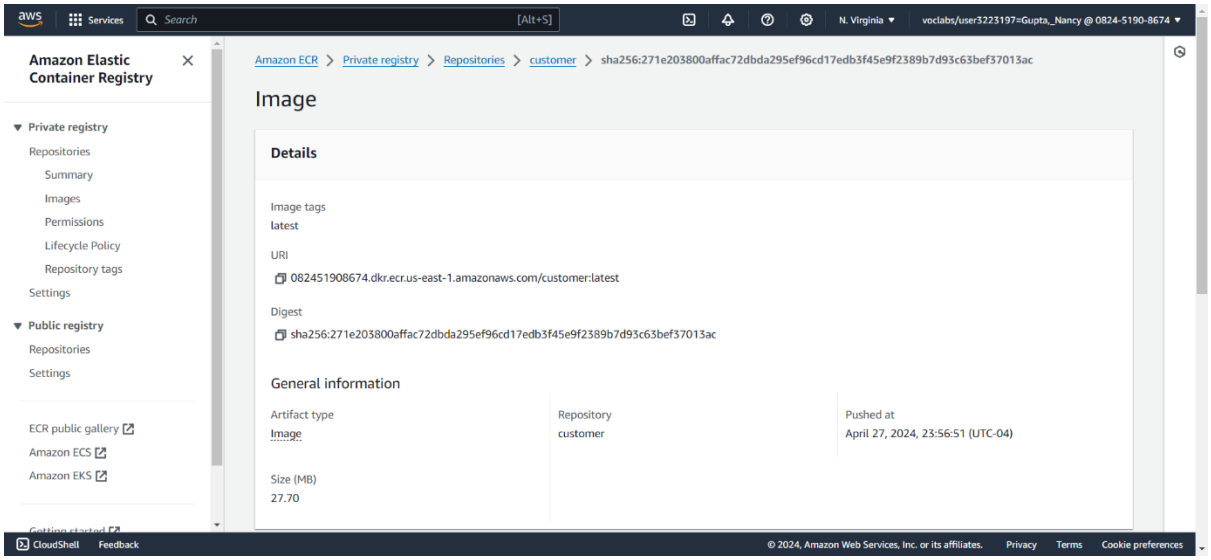


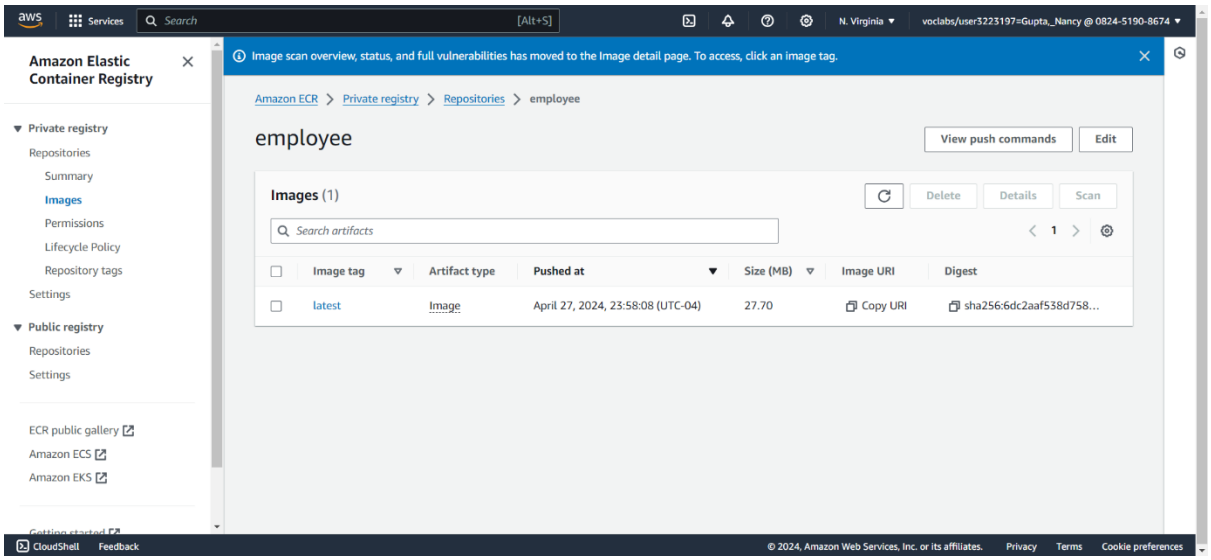


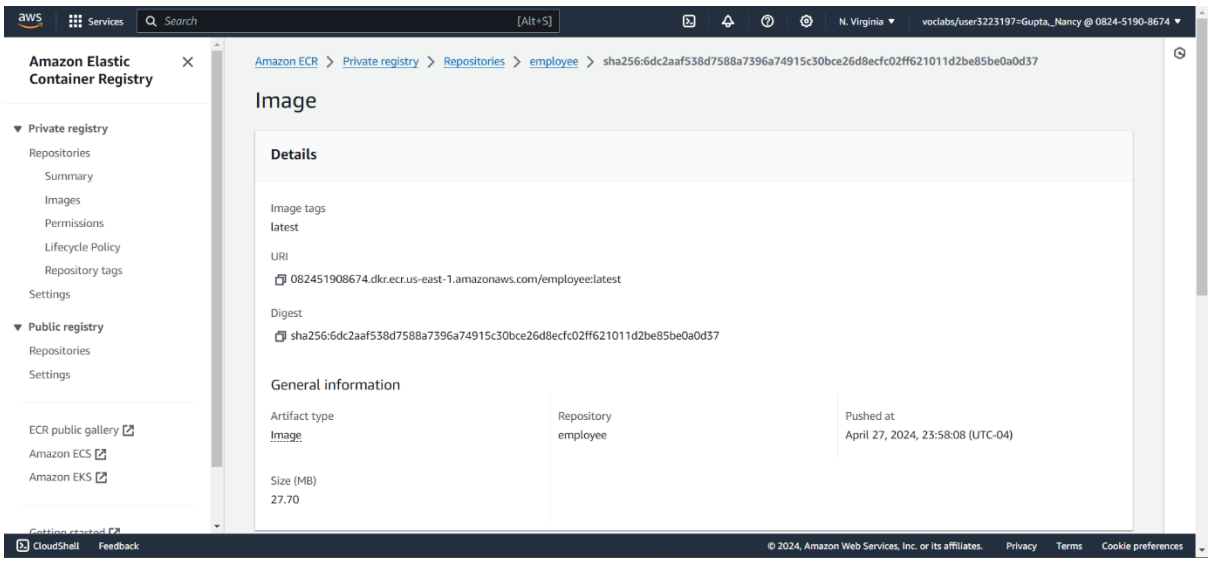
7. Confirm that the two images are now stored in Amazon ECR and that each has the *latest* label applied.











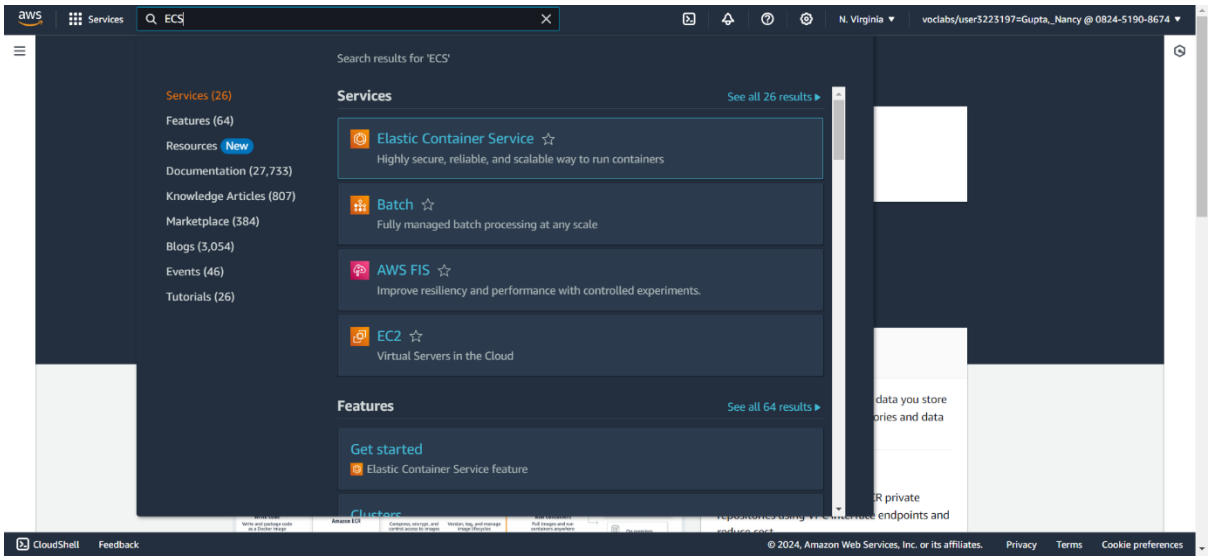
Task 5.2: Create an ECS cluster

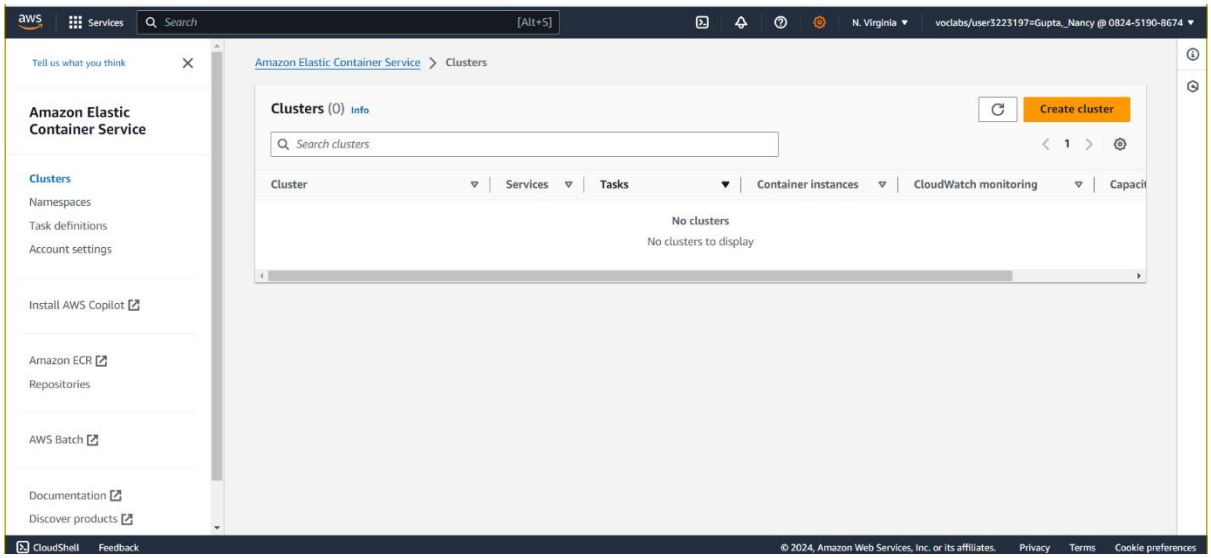
In this task, you will create an Amazon ECS cluster.

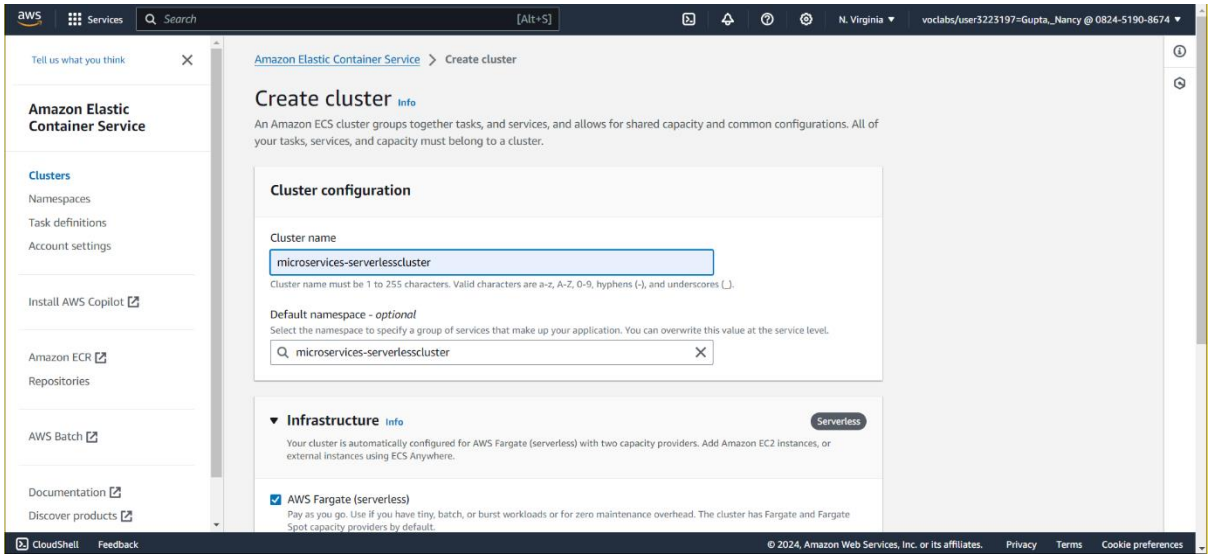
1. **Create a serverless AWS Fargate cluster that is named microservices-serverlesscluster**

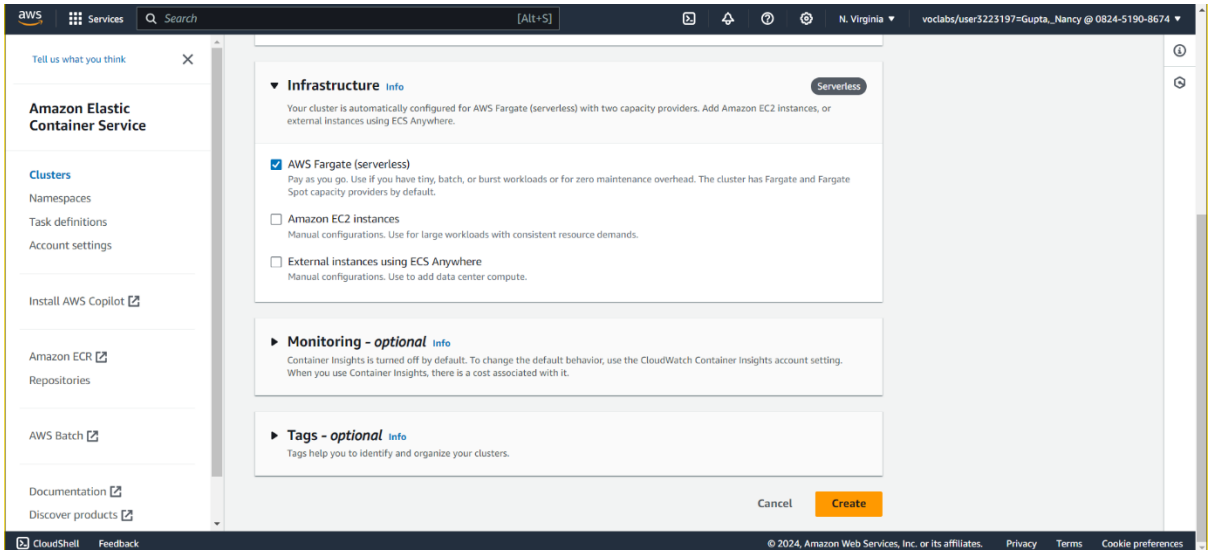
Ensure that it's configured to use *LabVPC*, *PublicSubnet1*, and *PublicSubnet2* (remove any other subnets). *DON'T* select Amazon EC2 instances or ECS Anywhere.

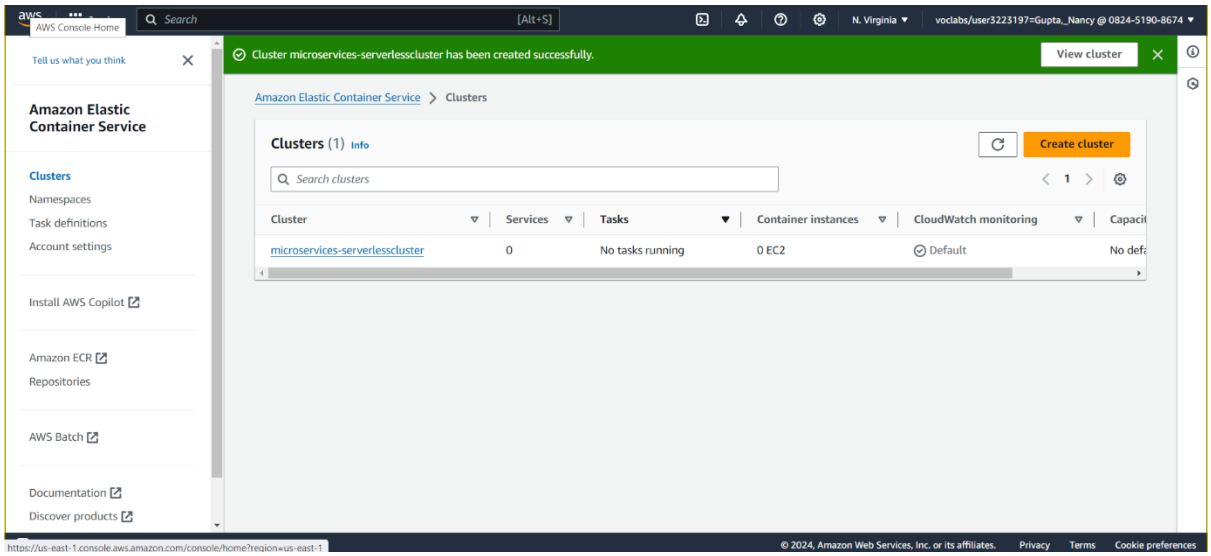
Important: After choosing the button to create the cluster, in the banner that appears across the top of the page, choose **View in CloudFormation**. Wait until the stack that creates the cluster attains the status *CREATE_COMPLETE* before you proceed to the next task. *If the stack fails to create for any reason and therefore rolls back*, repeat these steps to try again. It should succeed the second time.

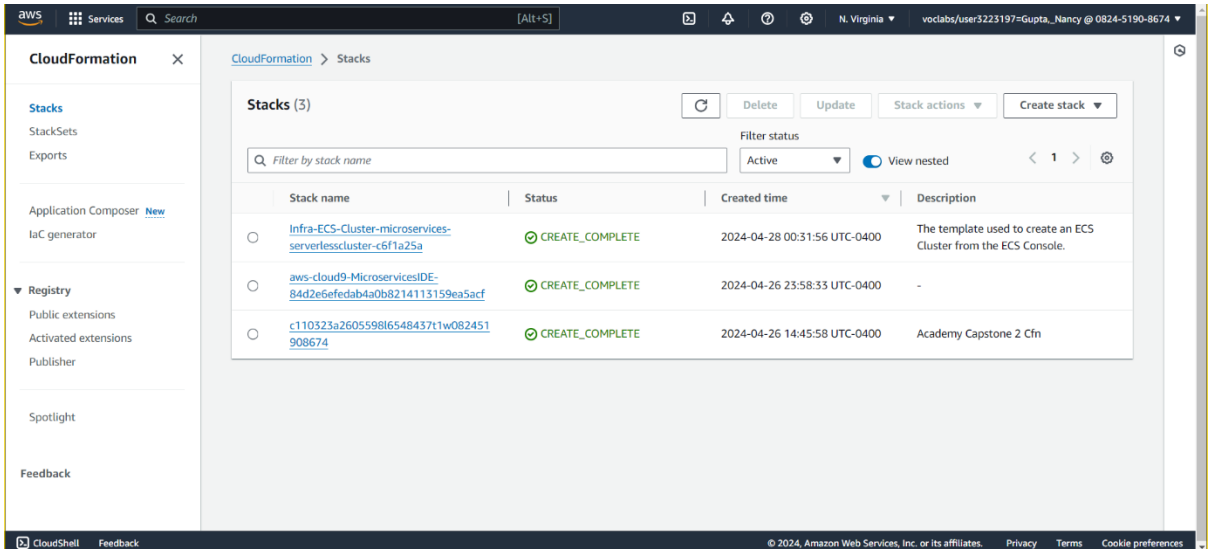


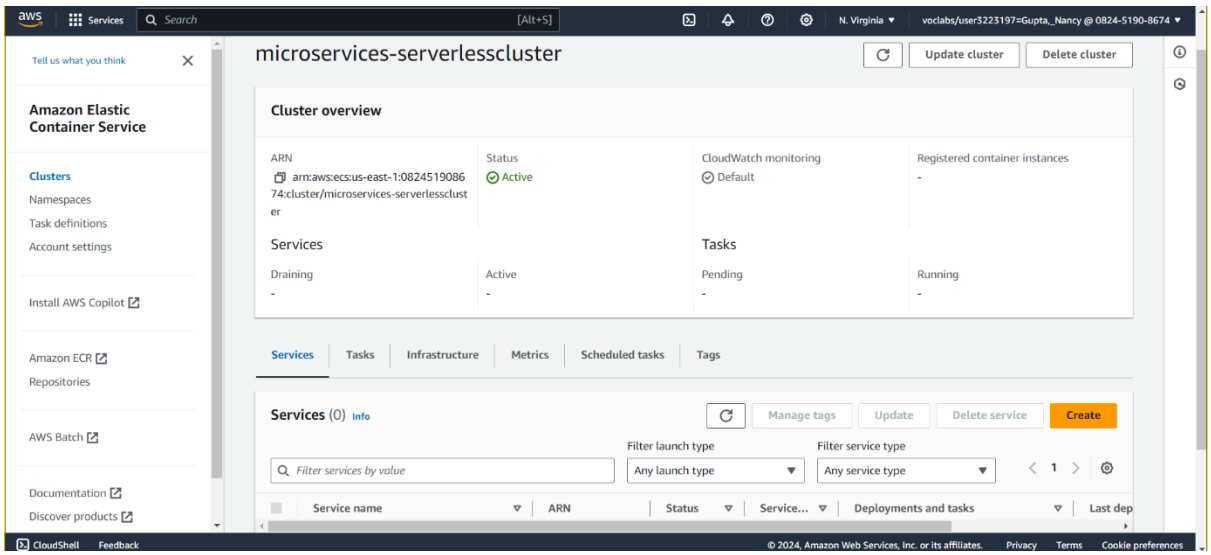



Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Tell us what you think
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Install AWS Copilot
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Amazon Elastic Container Service > Create cluster
Create cluster Info
An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.
Cluster configuration
Cluster name
microservices-serverlesscluster
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.
microservices-serverlesscluster
Infrastructure Info
Serverless
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.
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.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Tell us what you think
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Install AWS Copilot
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Infrastructure Info
Serverless
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.
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.
Amazon EC2 instances
Manual configurations. Use for large workloads with consistent resource demands.
External instances using ECS Anywhere
Manual configurations. Use to add data center compute.
Monitoring - optional Info
Container Insights is turned off by default. To change the default behavior, use the CloudWatch Container Insights account setting. When you use Container Insights, there is a cost associated with it.
Tags - optional Info
Tags help you to identify and organize your clusters.
Cancel
Create
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


AWS Console Home
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Tell us what you think
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Install AWS Copilot
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Cluster microservices-serverlesscluster has been created successfully.
View cluster
Amazon Elastic Container Service > Clusters
Clusters (1) Info
Create cluster
Search clusters
Cluster
Services
Tasks
Container instances
CloudWatch monitoring
microservices-serverlesscluster
0
No tasks running
0 EC2
Default
https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

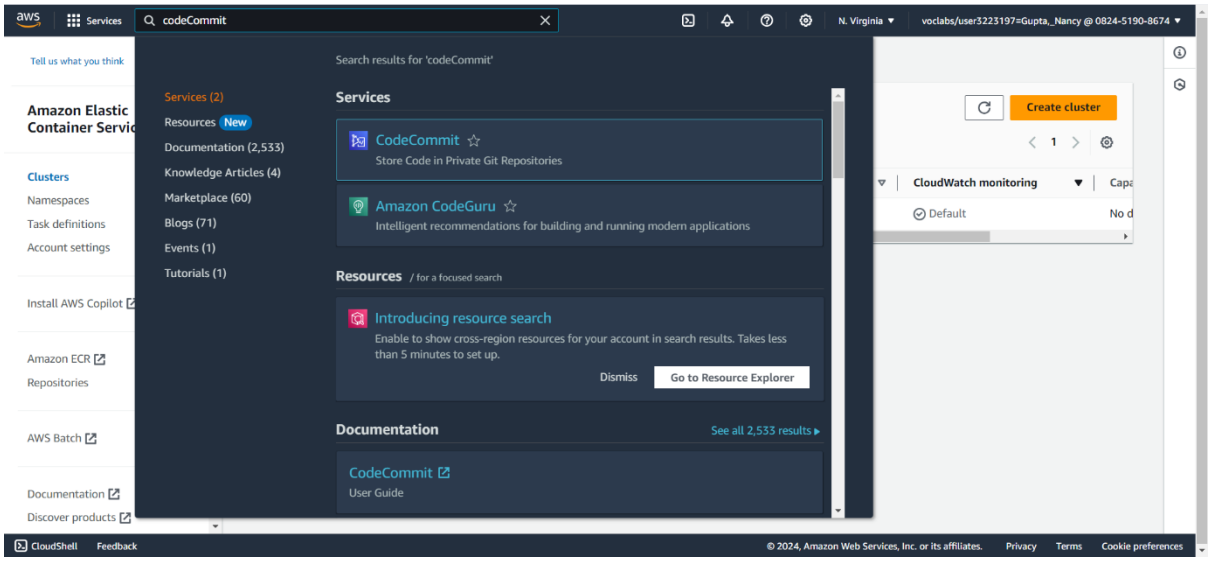


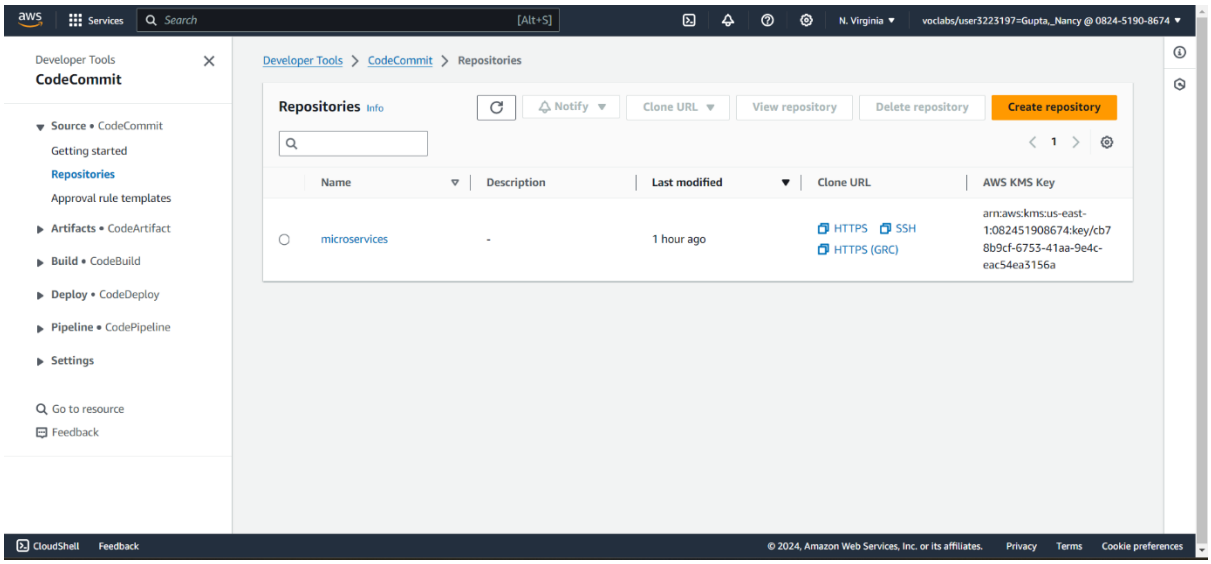


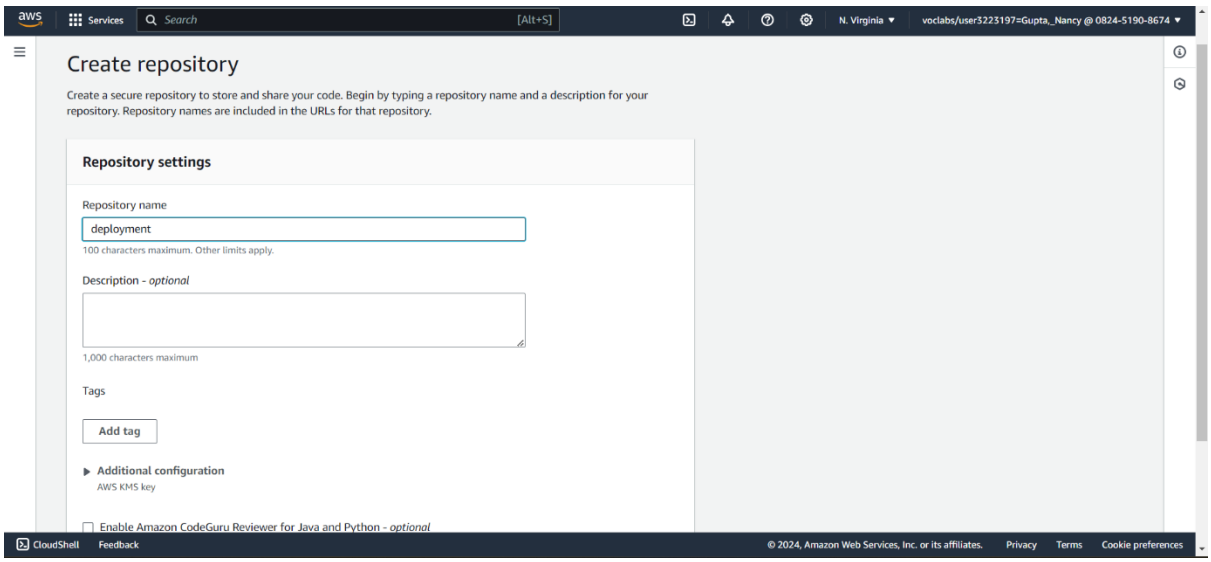
Task 5.3: Create a CodeCommit repository to store deployment files

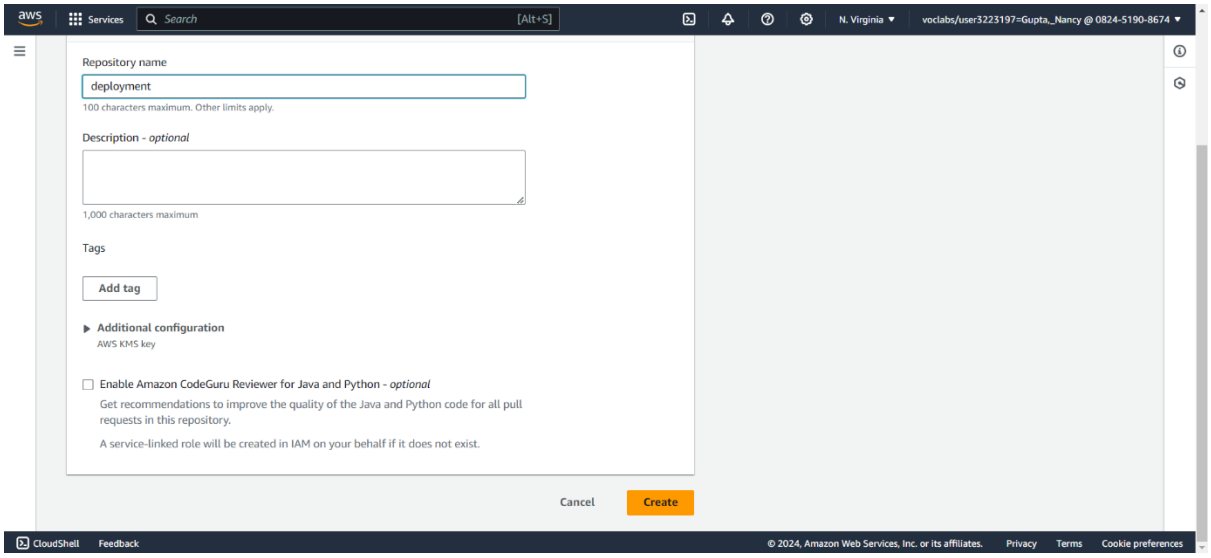
In this task, you will create another CodeCommit repository. This repository will store the task configuration specification files that Amazon ECS will use for each microservice. The repository will also store AppSpec specification files that CodeDeploy will use for each microservice.

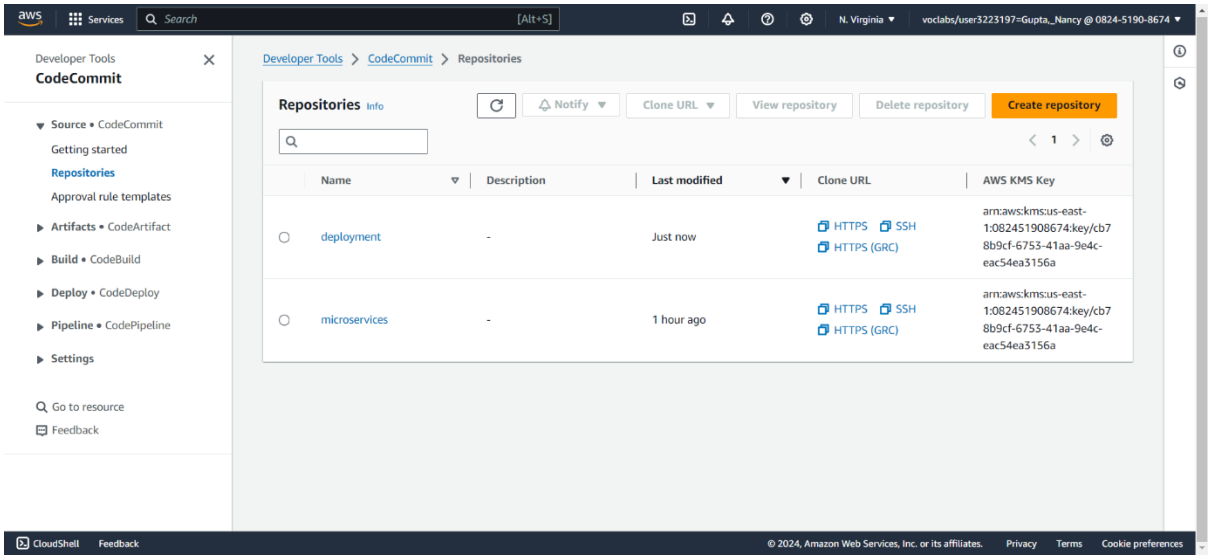
1. **Create a new CodeCommit repository that is named deployment to store deployment configuration files.**


aws
Services
codeCommit
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Tell us what you think
Amazon Elastic Container Servic
Clusters
Namespaces
Task definitions
Account settings
Install AWS Copilot
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Search results for 'codeCommit'
Services (2)
Resources New
Documentation (2,533)
Knowledge Articles (4)
Marketplace (60)
Blogs (71)
Events (1)
Tutorials (1)
Services
CodeCommit
Store Code in Private Git Repositories
Amazon CodeGuru
Intelligent recommendations for building and running modern applications
Resources / for a focused search
Introducing resource search
Enable to show cross-region resources for your account in search results. Takes less than 5 minutes to set up.
Dismiss
Go to Resource Explorer
Documentation
See all 2,533 results
CodeCommit
User Guide
Create cluster
CloudWatch monitoring
Default
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Developer Tools > CodeCommit > Repositories
Repositories Info
Notify
Clone URL
View repository
Delete repository
Create repository
Name
Description
Last modified
Clone URL
AWS KMS Key
microservices
-
1 hour ago
HTTPS
SSH
HTTPS (GRC)
arn:aws:kms:us-east-1:082451908674:key/cb78b9cf-6753-41aa-9e4c-eac54ea3156a
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Create repository
Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.
Repository settings
Repository name
deployment
100 characters maximum. Other limits apply.
Description - optional
1,000 characters maximum
Tags
Add tag
Additional configuration
AWS KMS key
Enable Amazon CodeGuru Reviewer for Java and Python - optional
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



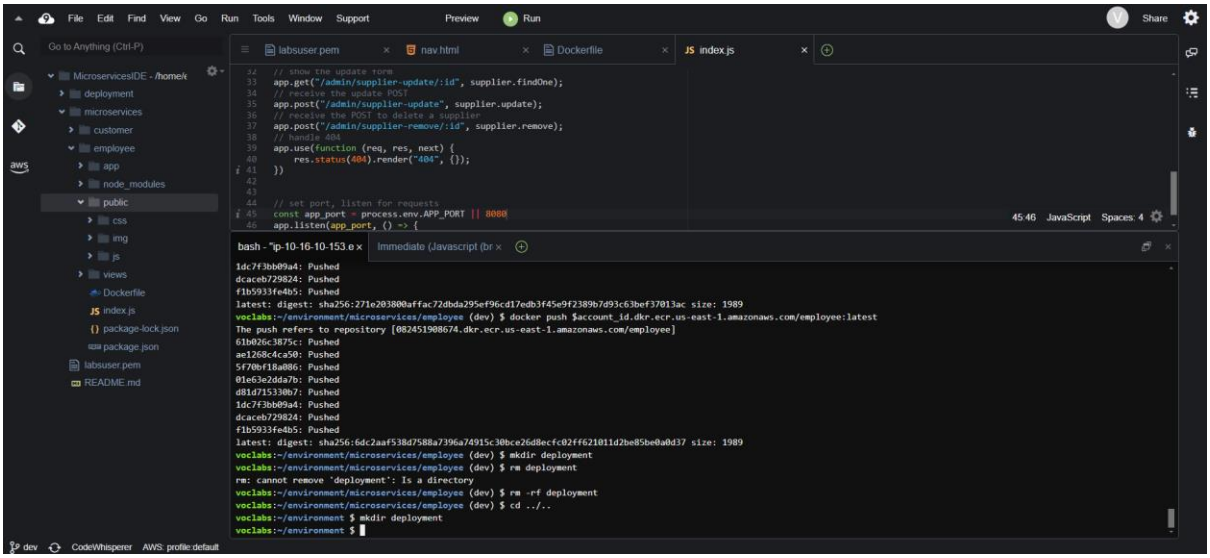


2. **In AWS Cloud9, in the environment directory, create a new directory that is named deployment. Initialize the directory as a Git repository with a branch named dev.**

   Use the following commands:

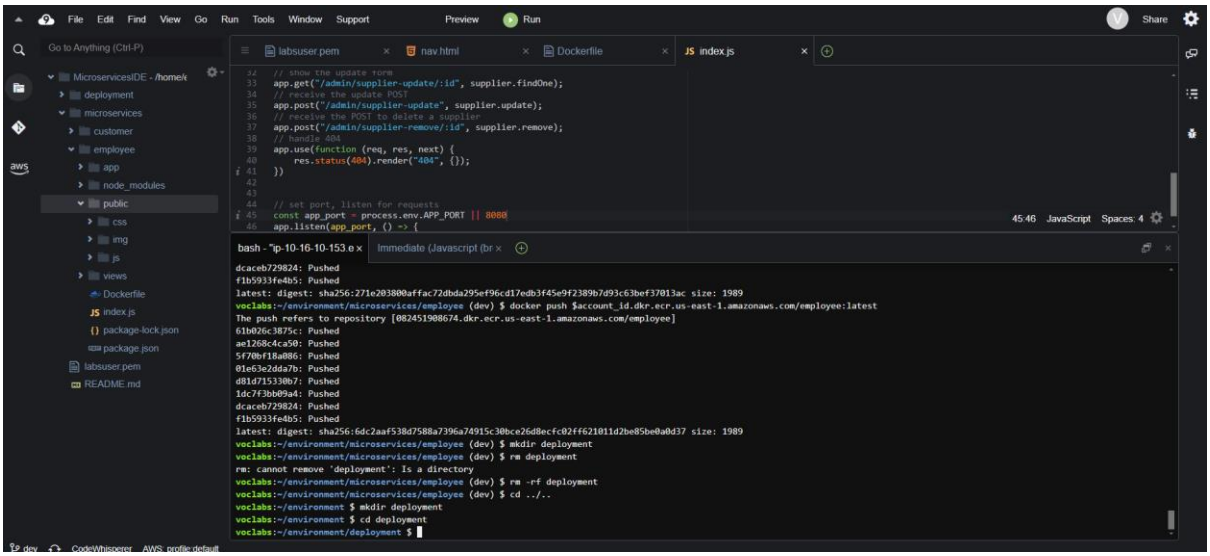
   # Create the deployment directory

   mkdir deployment



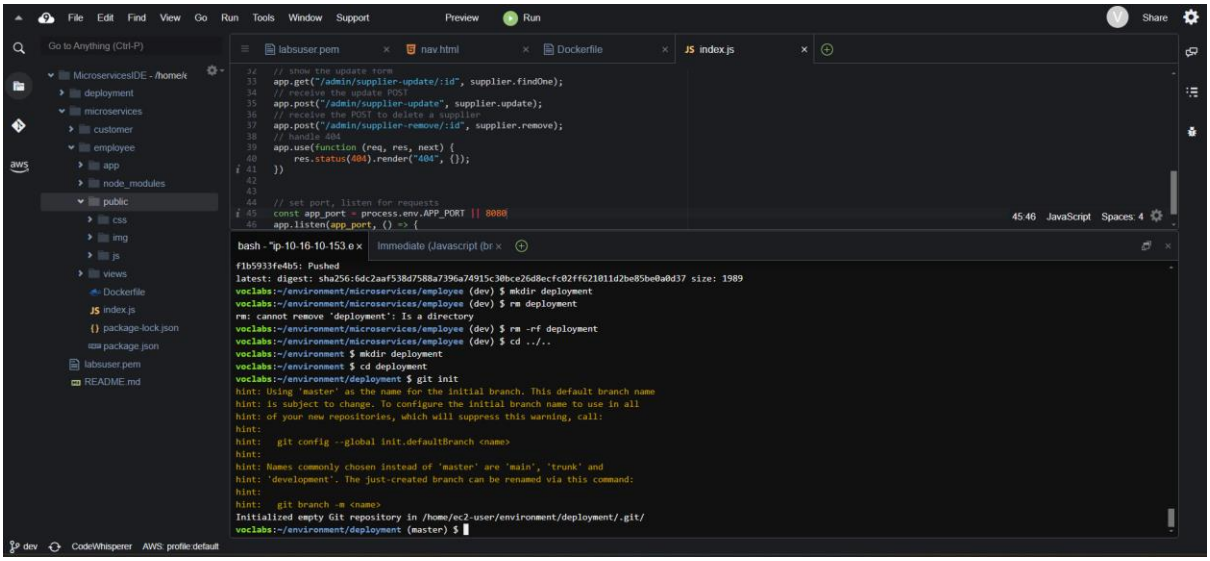
Navigate into the deployment directory

cd deployment



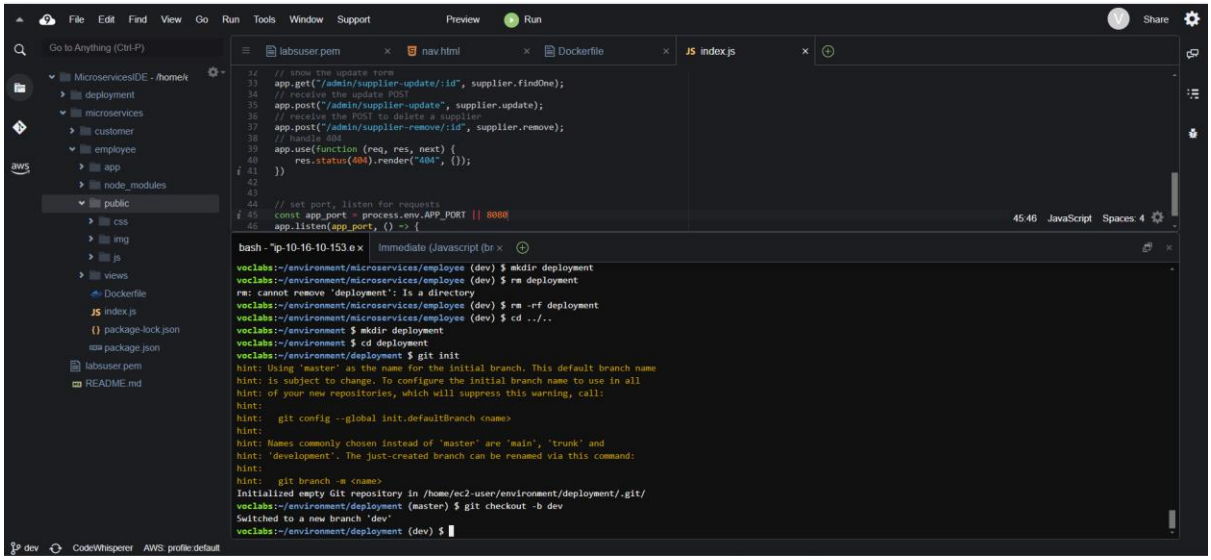
Initialize the directory as a Git repository

git init



Create and switch to a new branch named "dev"

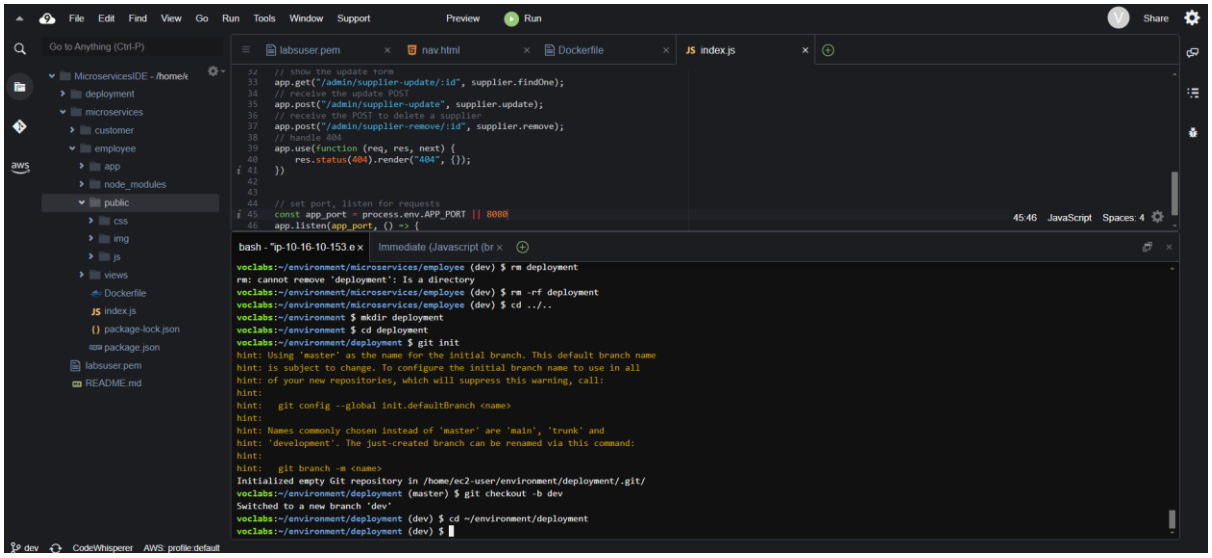
git checkout -b dev

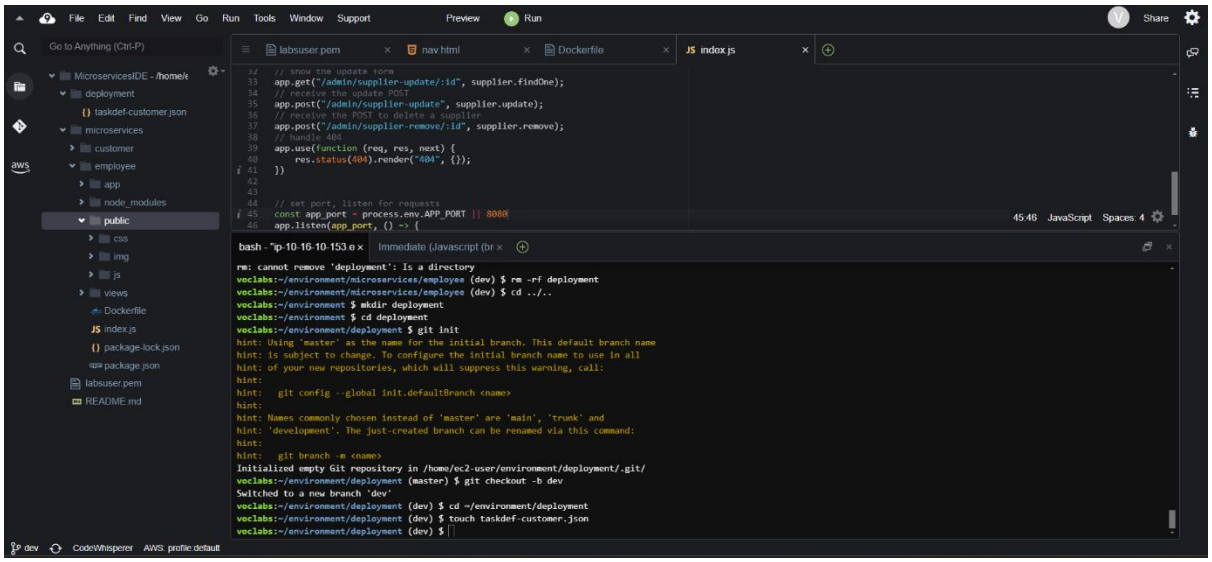


Task 5.4: Create task definition files for each microservice and register them with Amazon ECS

In this task, you will create a task definition file for each microservice and then register the task definitions with Amazon ECS.

1. **In the new deployment directory, create an empty file named taskdef-customer.json**

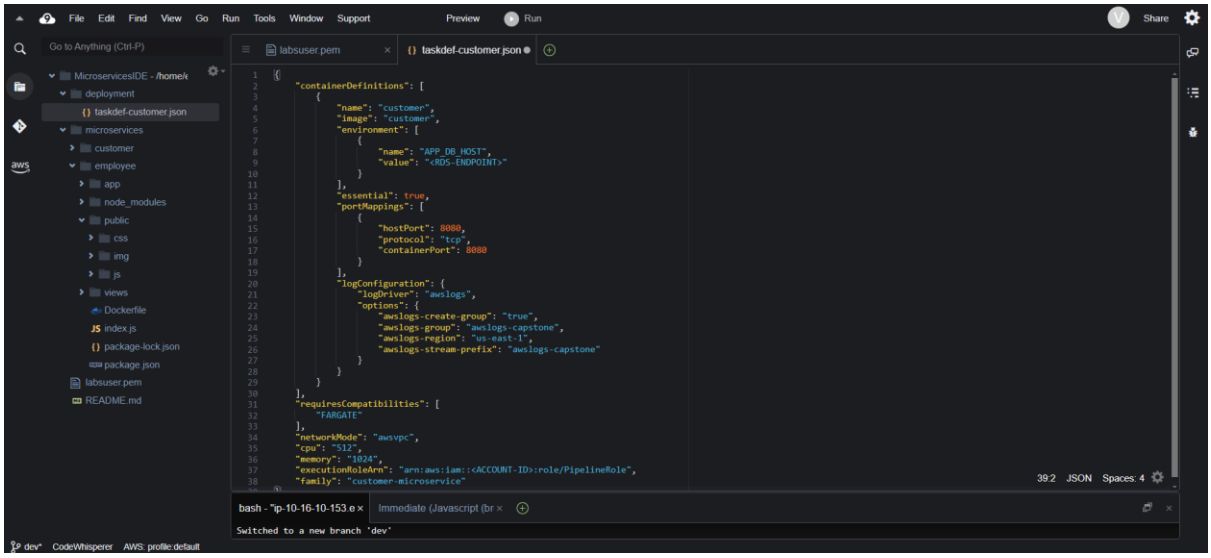




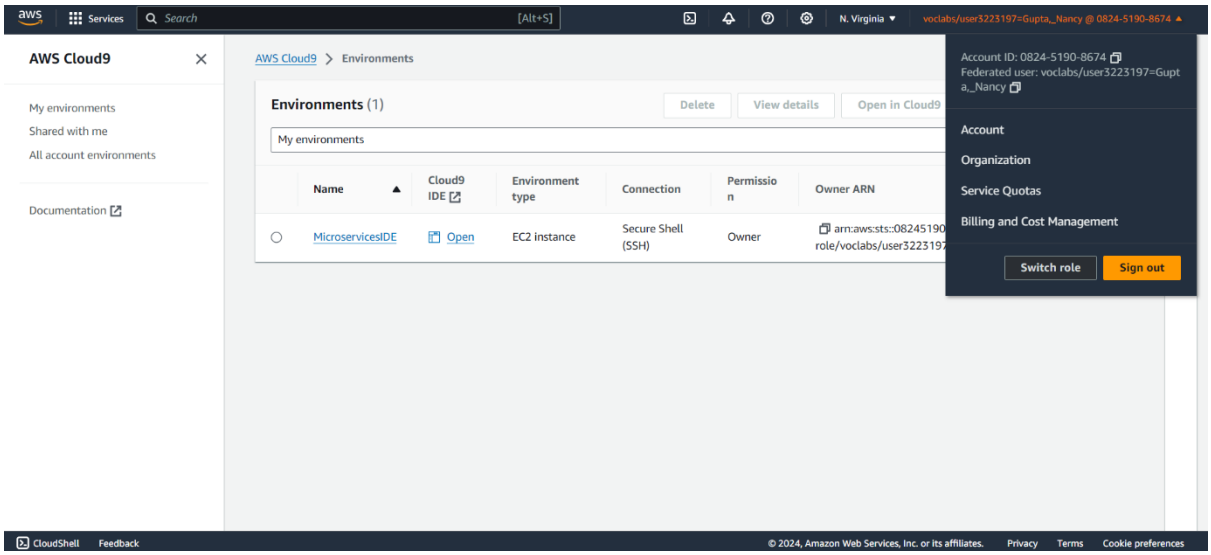
2. **Edit the taskdef-customer.json file.**
    - **Paste the following JSON code into the file:**

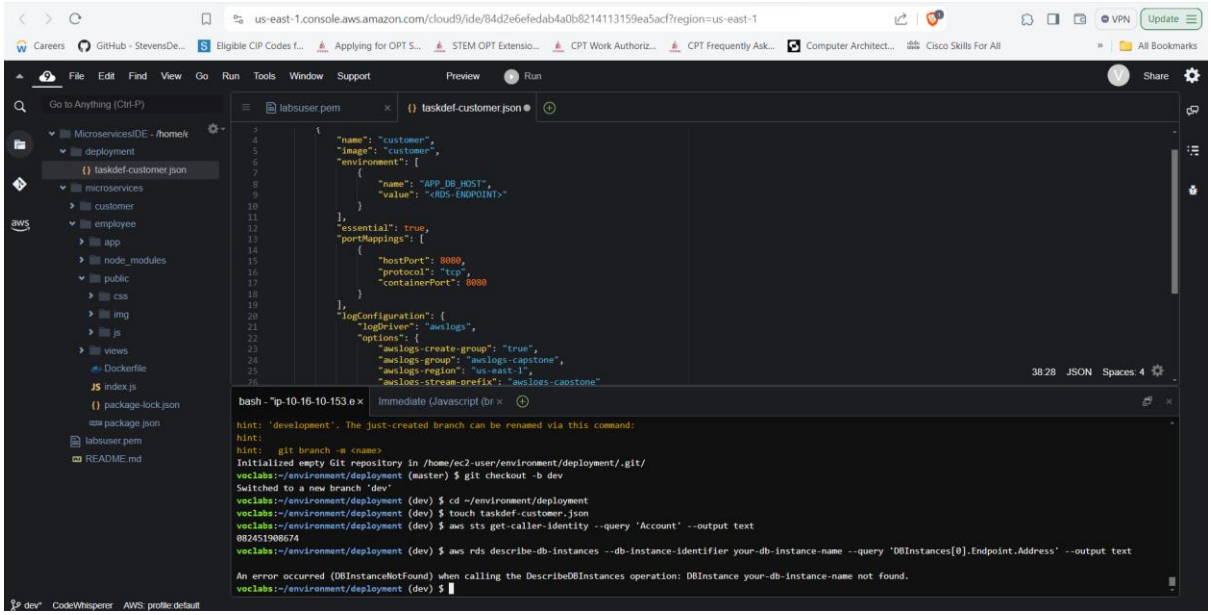
```
{
  "containerDefinitions": [
    {
      "name": "customer",
      "image": "customer",
      "environment": [
        {
          "name": "APP_DB_HOST",
          "value": "<RDS-ENDPOINT>"
```

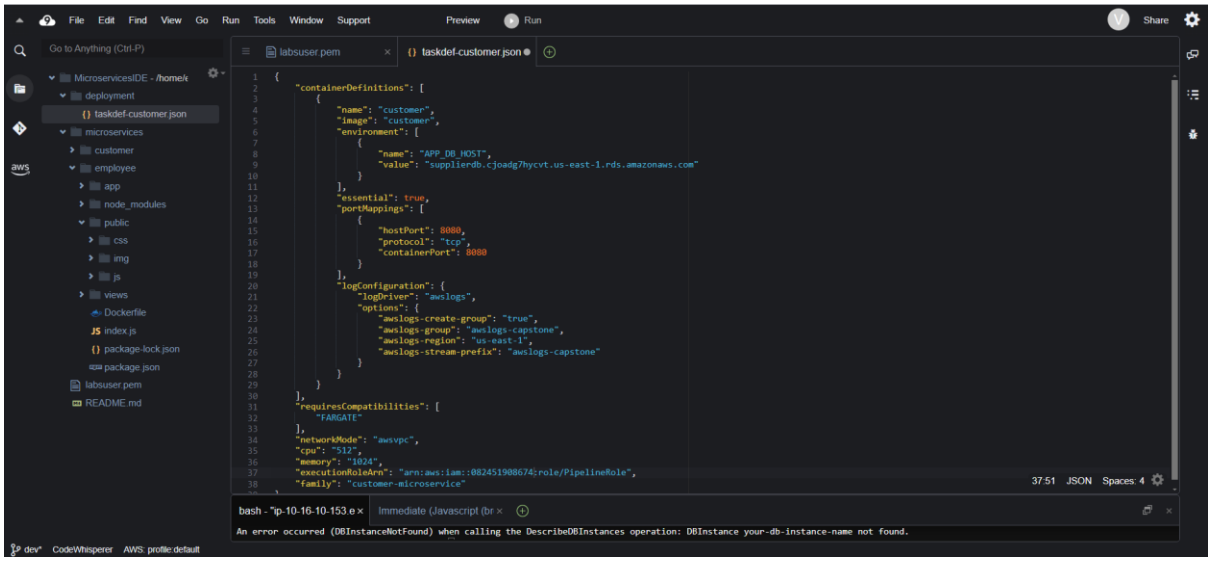
```
            }
         ],
         "essential": true,
         "portMappings": [
            {
               "hostPort": 8080,
               "protocol": "tcp",
               "containerPort": 8080
            }
         ],
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-create-group": "true",
               "awslogs-group": "awslogs-capstone",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "awslogs-capstone"
            }
         }
      }
   ],
   "requiresCompatibilities": [
      "FARGATE"
   ],
   "networkMode": "awsvpc",
   "cpu": "512",
   "memory": "1024",
   "executionRoleArn": "arn:aws:iam::<ACCOUNT-ID>:role/PipelineRole",
   "family": "customer-microservice"
}
```



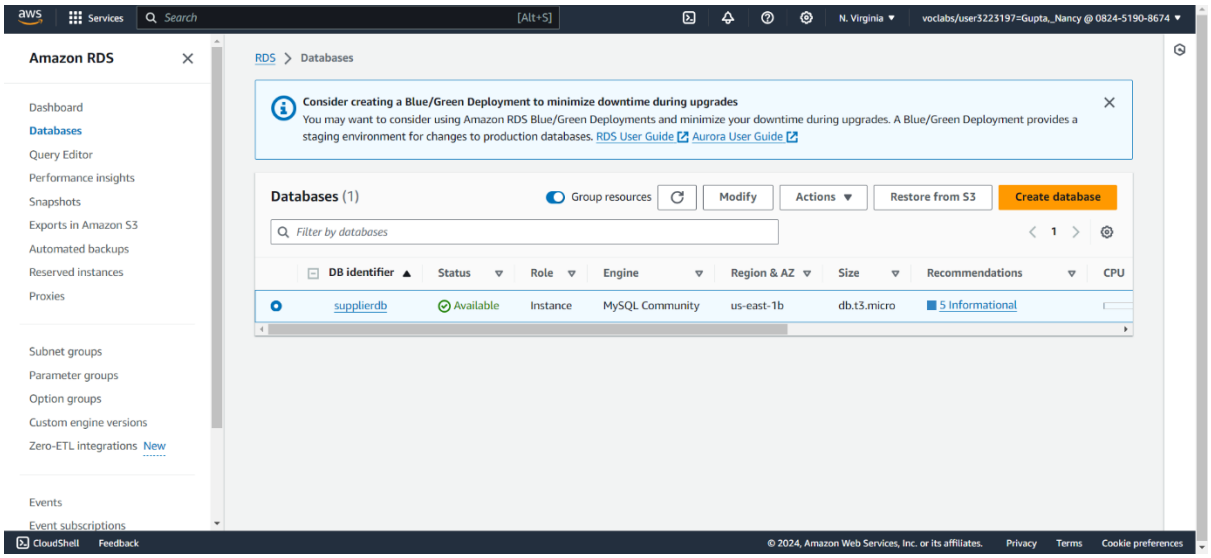
- **Replace a couple values in the file:**
  - **On line 37, replace <ACCOUNT-ID> with the actual account ID.**

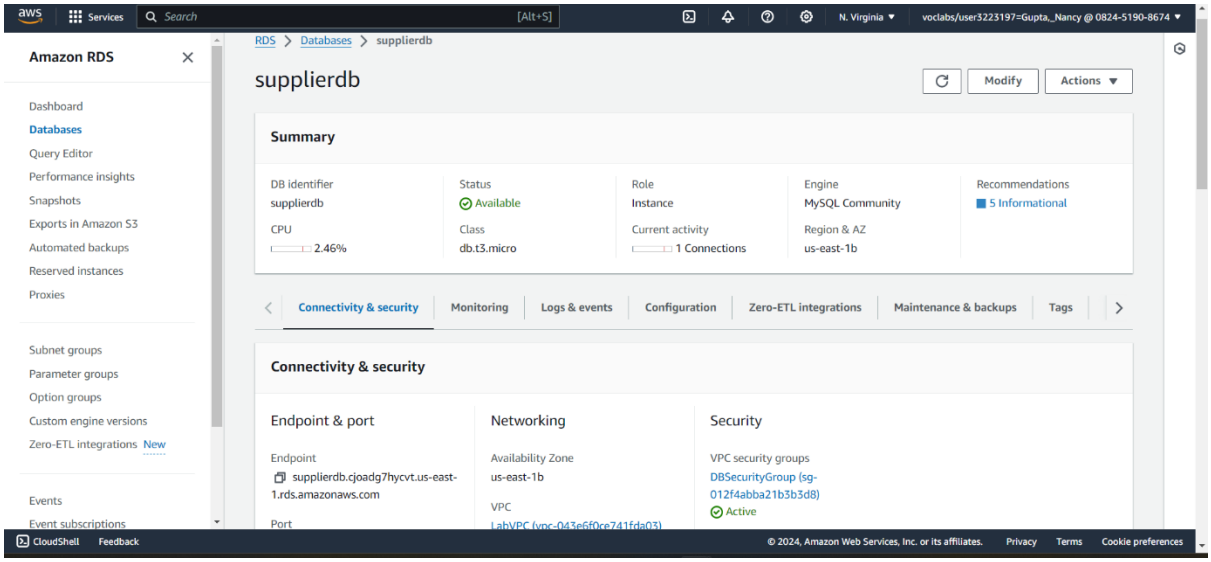


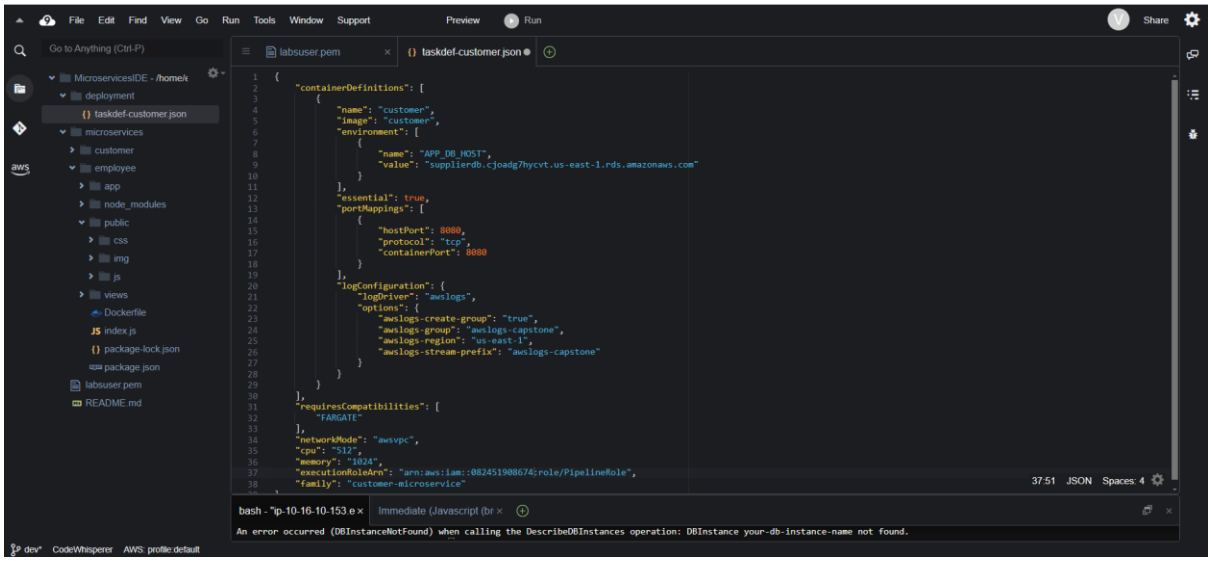




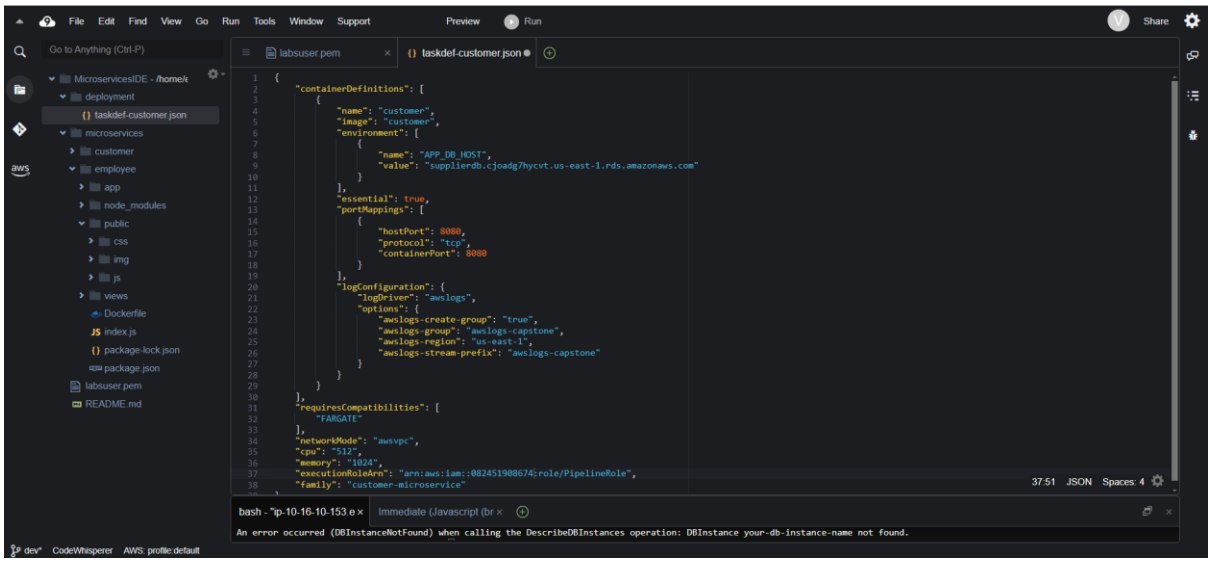
- **On line 9, replace <RDS-ENDPOINT> with the actual RDS endpoint.**





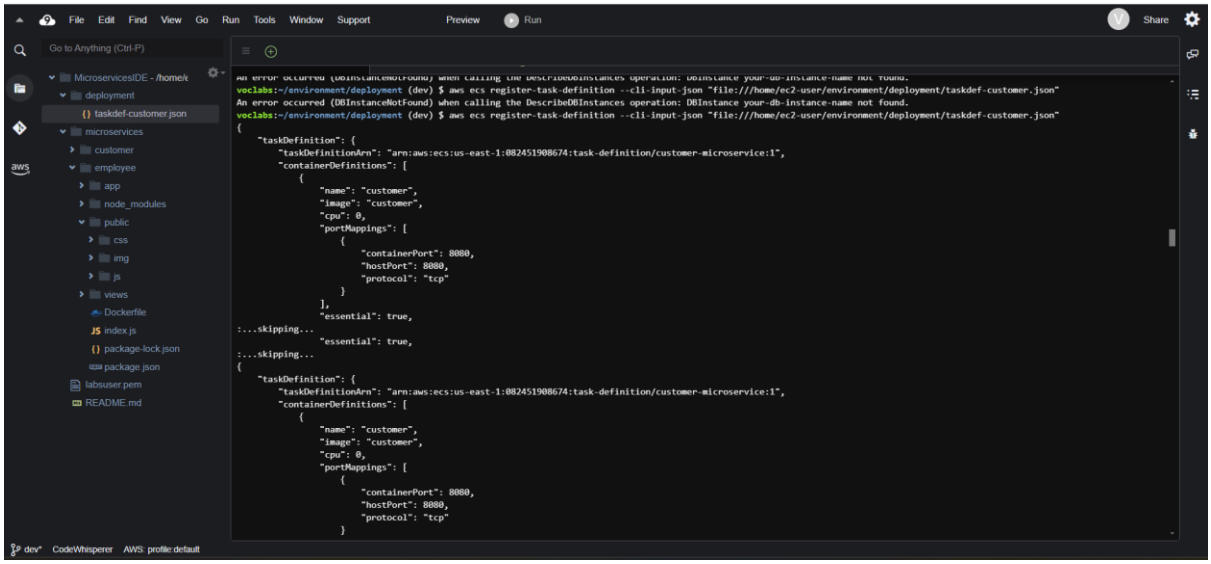


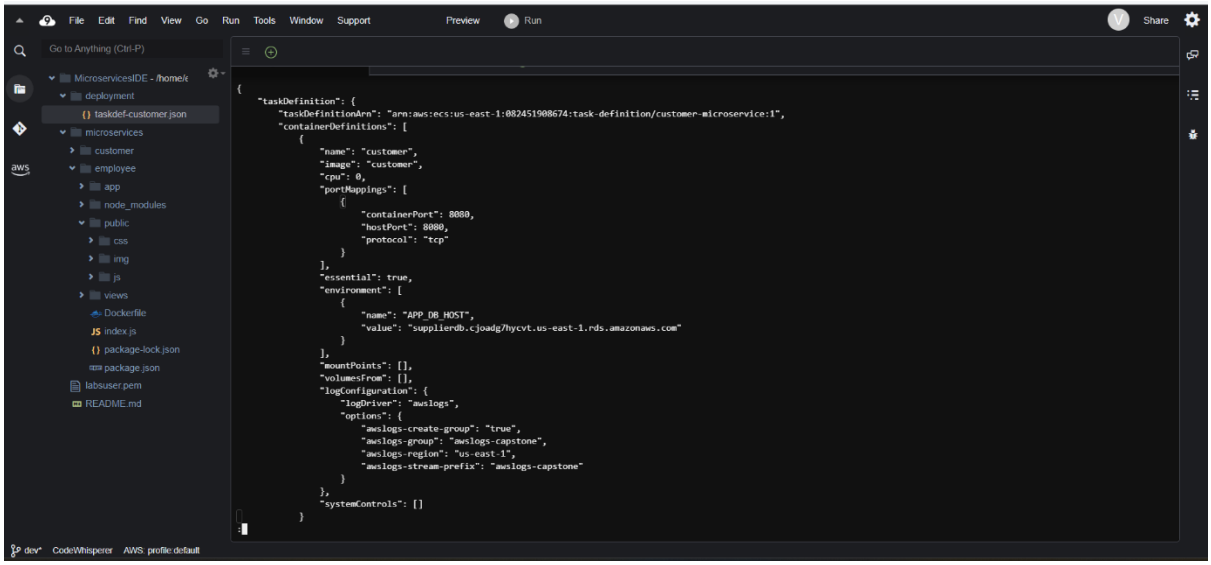
- **Save the changes.**



3. **To register the *customer* microservice task definition in Amazon ECS, run the following command:**

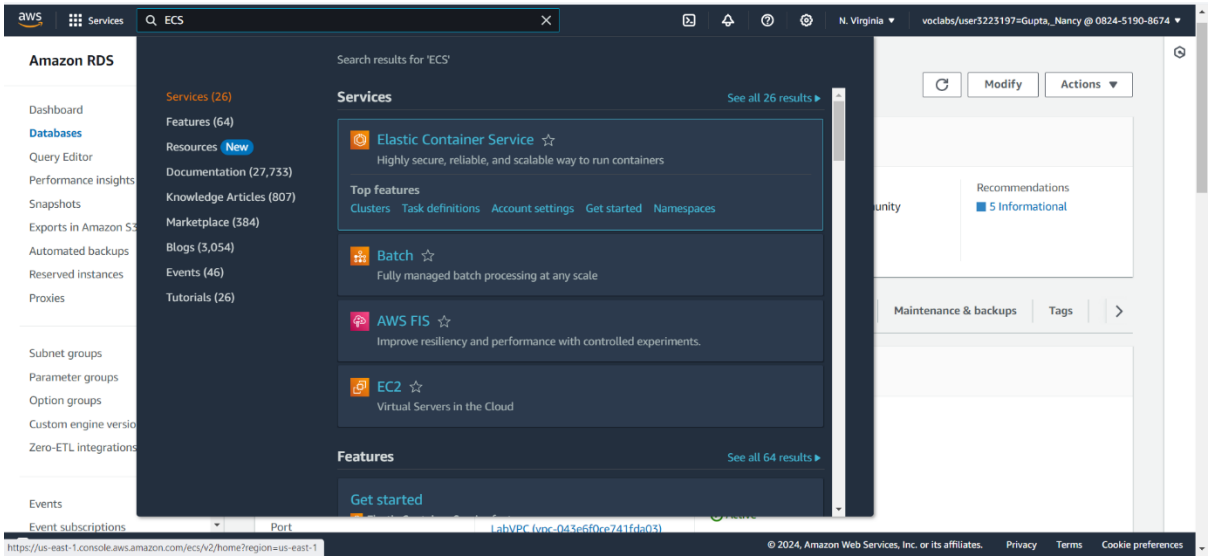
aws ecs register-task-definition --cli-input-json file:///home/ec2-user/environment/deployment/taskdef-customer.json

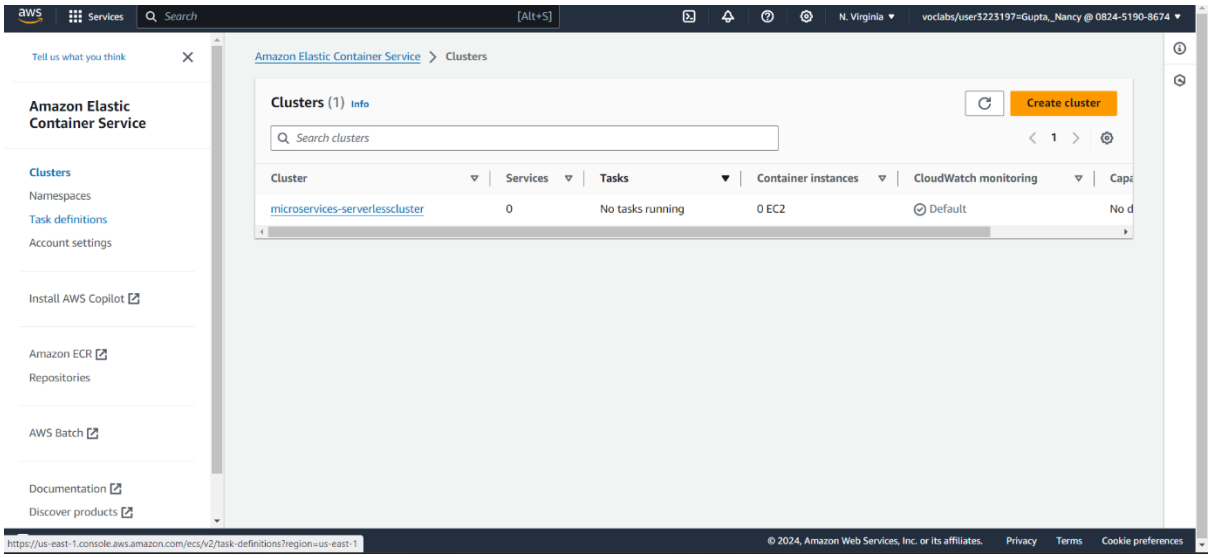


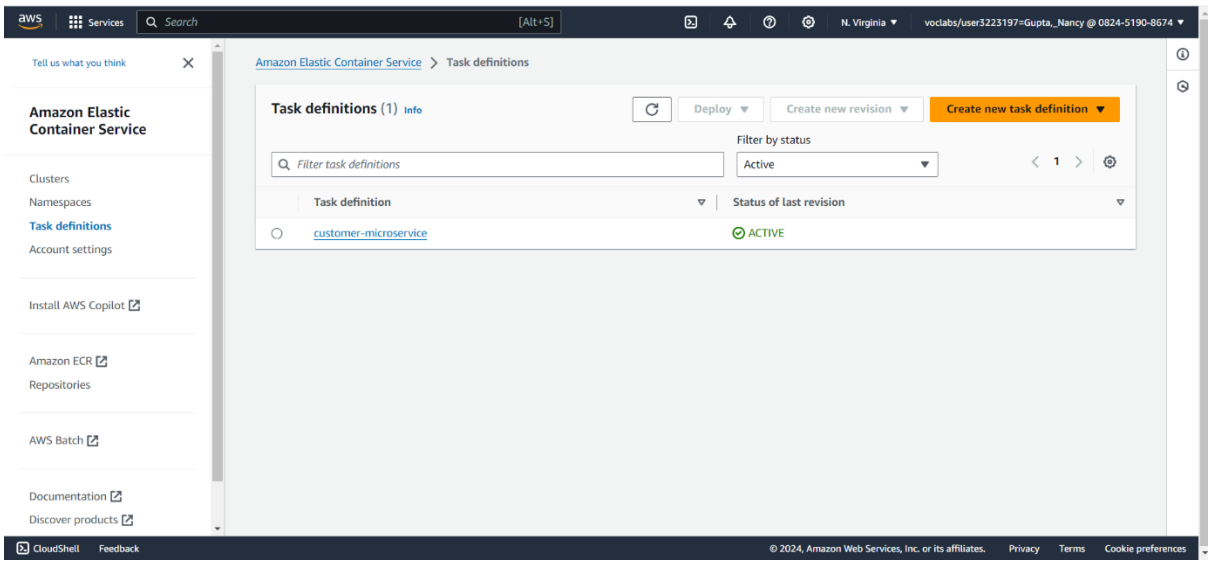


4. **In the Amazon ECS console, verify that the *customer-microservice* task definition now appears in the Task definitions pane. Also, notice that the revision number displays after the task definition name.**

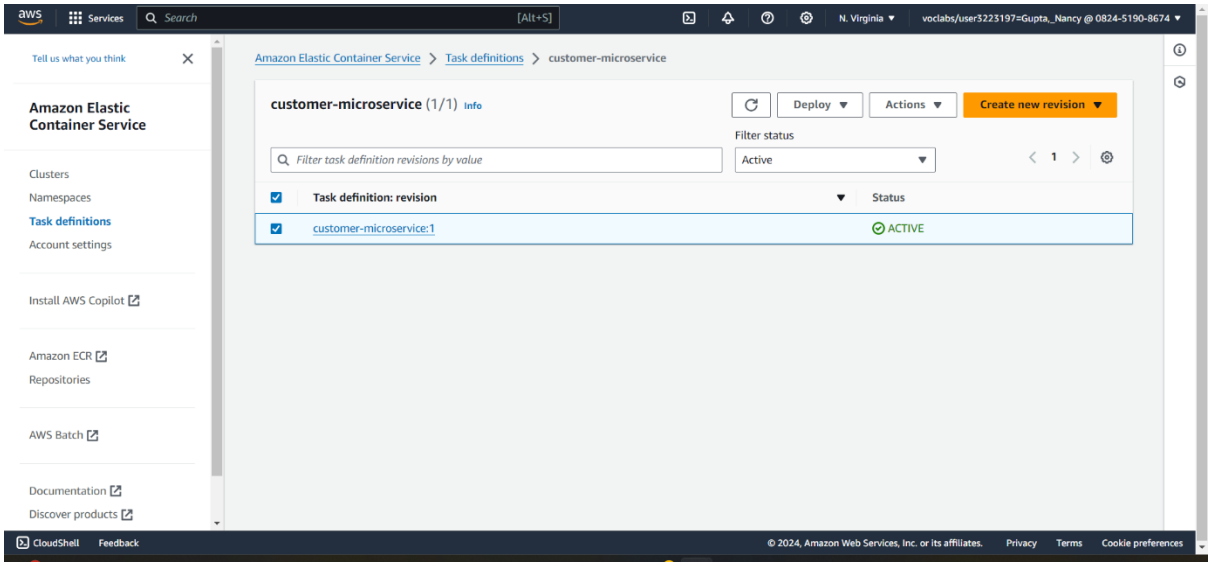
Tip: Consult the ECS documentation if it is helpful.



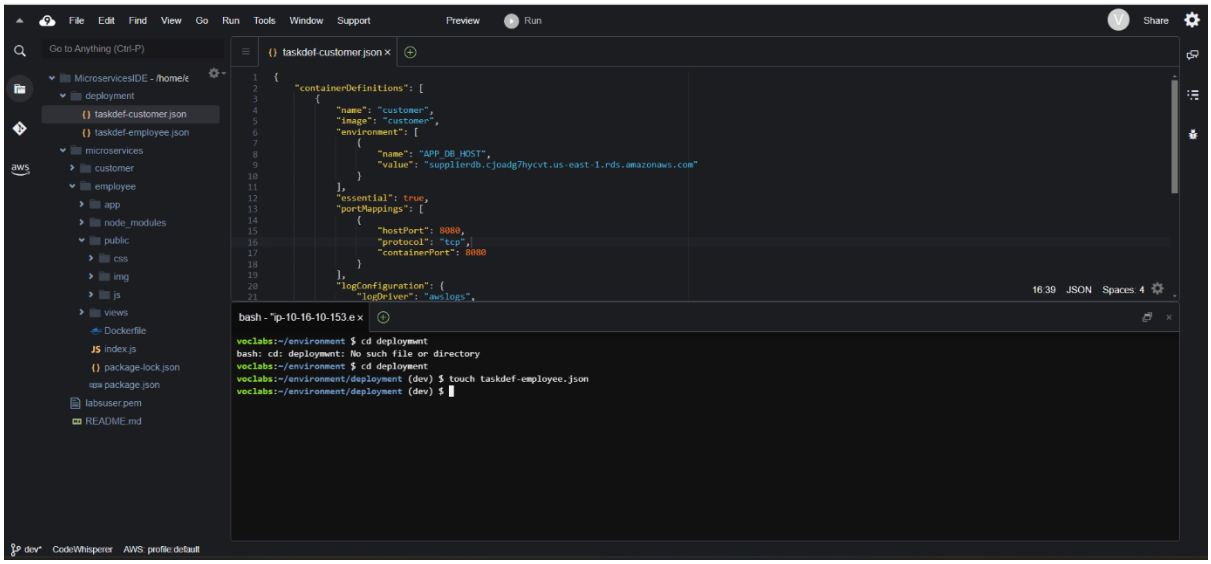




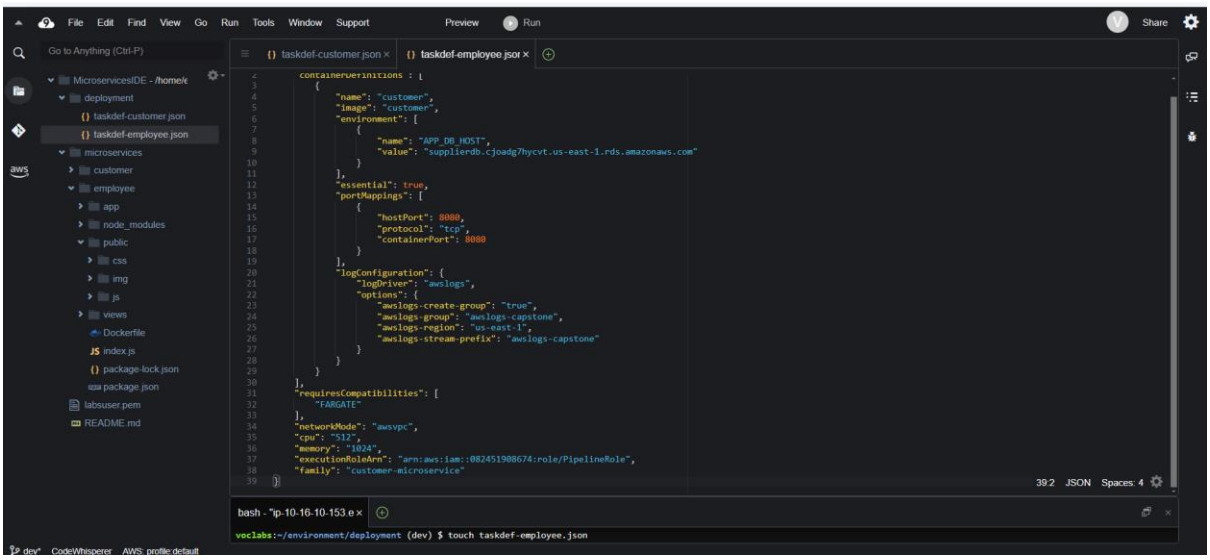
Notice that the revision number displays after the task definition name



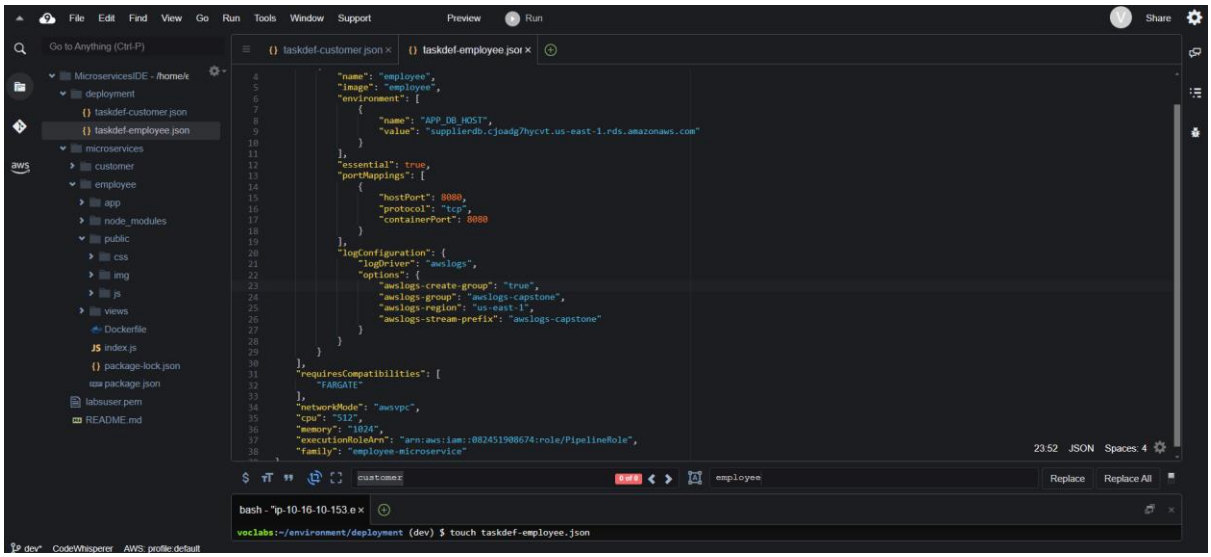
5. **In the *deployment* directory, create a taskdef-employee.json specification file.**



- **Add the same JSON code to it that currently exists in the *taskdef-customer.json* file (where you have already set the account ID and RDS endpoints).**



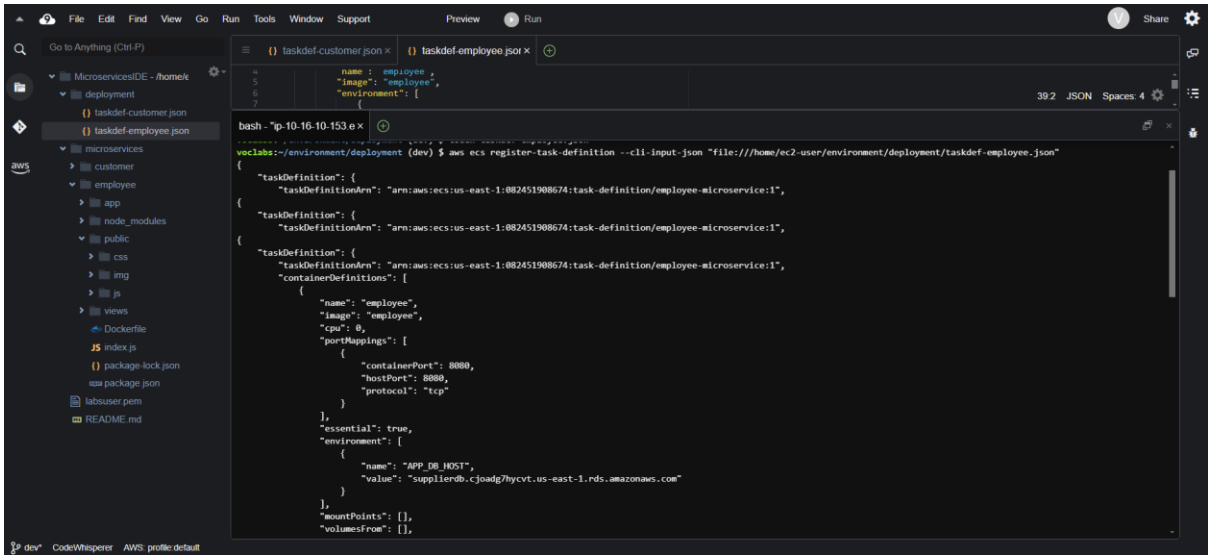
- **After you paste in the code, change the three occurrences of customer to employee**



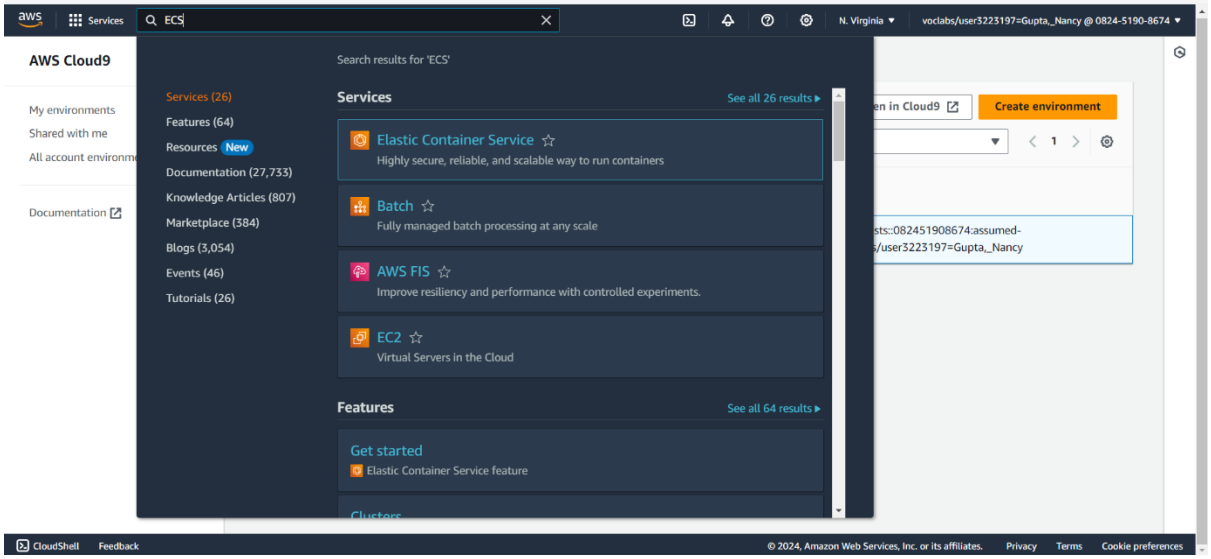
6. **To register the *employee* task definition with Amazon ECS, run an AWS CLI command.**

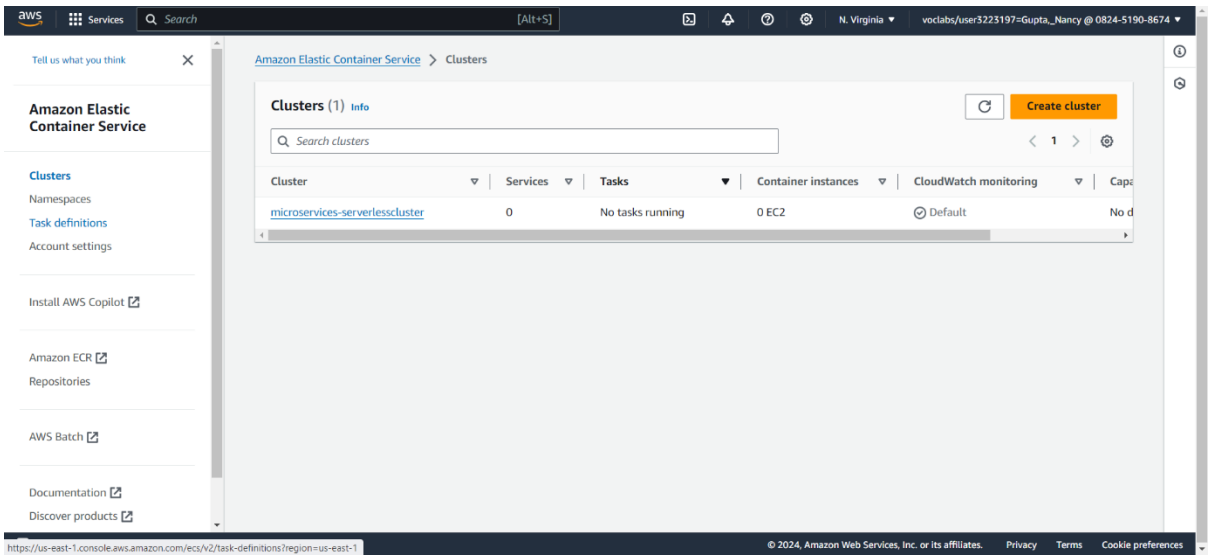
Use the following command:

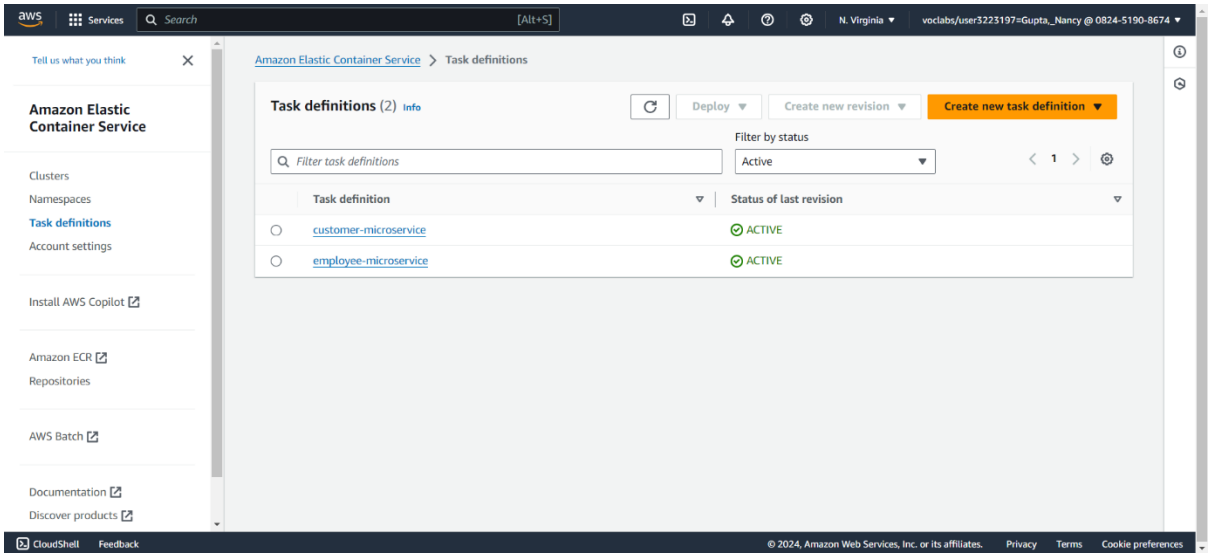
aws ecs register-task-definition --cli-input-json file:///home/ec2-user/environment/deployment/taskdef-employee.json

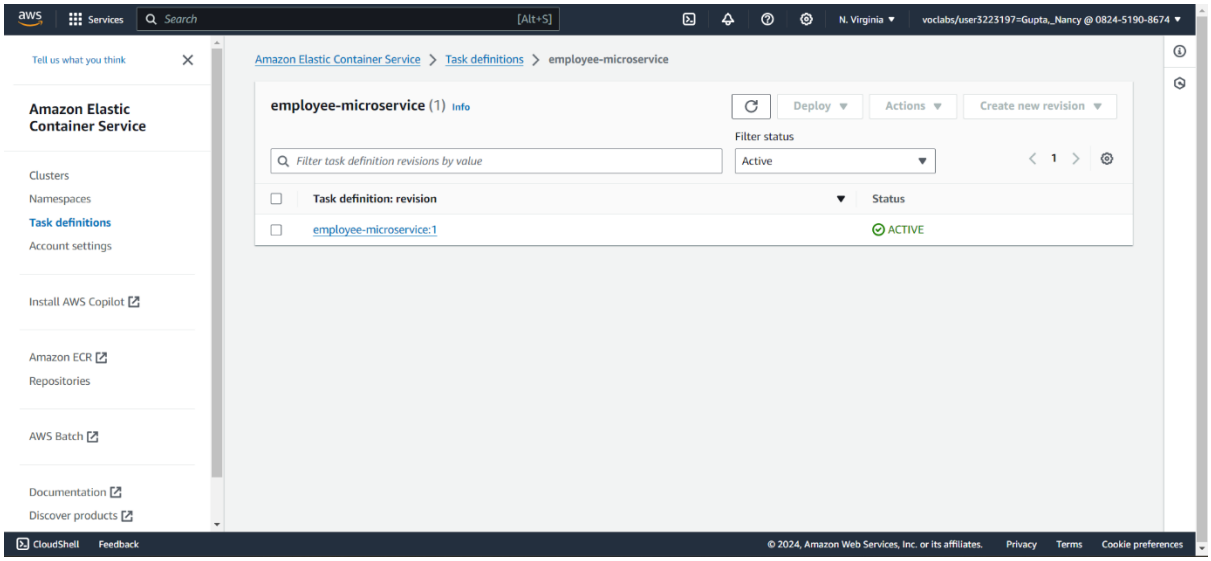


7. **In the Amazon ECS console, verify that the *employee-microservice* task definition now appears in the Task definitions pane. Also, notice that the revision number displays after the task definition name.**





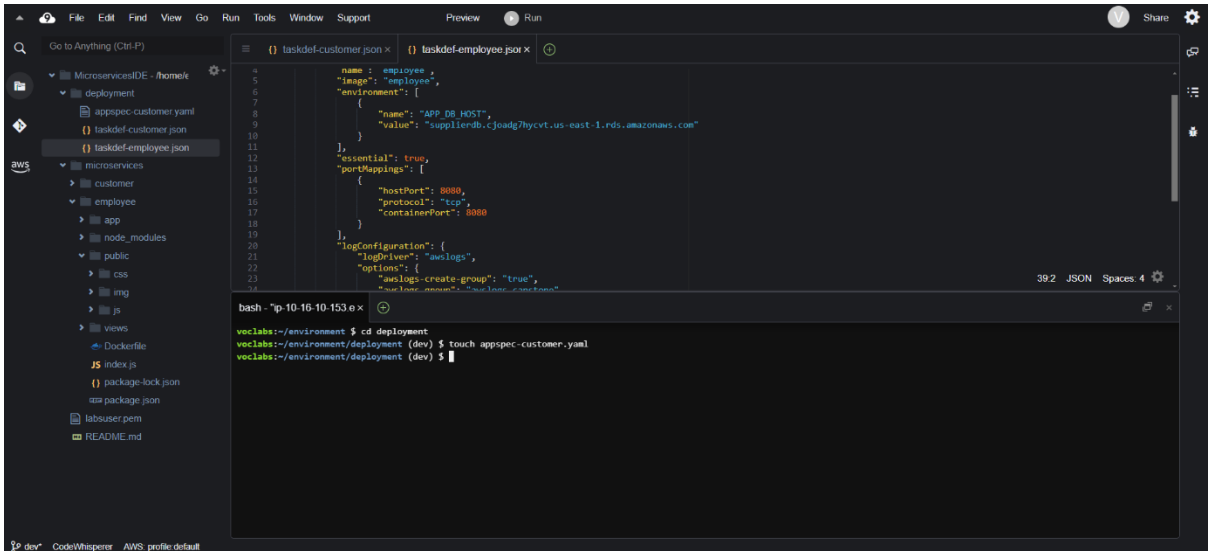




Task 5.5: Create AppSpec files for CodeDeploy for each microservice

In this task, you will continue to complete tasks to support deploying the microservices-based web application to run on an ECS cluster where the deployment is supported by a CI/CD pipeline. In this specific task, you will create two application specification (AppSpec) files, one for each microservice. These files will provide instructions for CodeDeploy to deploy the microservices to the Amazon ECS on Fargate infrastructure.

1. **Create an AppSpec file for the *customer* microservice.**
    - **In the *deployment* directory, create a new file named appspec-customer.yaml**



- **Paste the following YAML code into the file:**

Important: *DON'T* modify <TASK_DEFINITION>. This setting will be updated automatically when the pipeline runs.

version: 0.0

Resources:

- TargetService:

  Type: AWS::ECS::Service

  Properties:

    TaskDefinition: <TASK_DEFINITION>

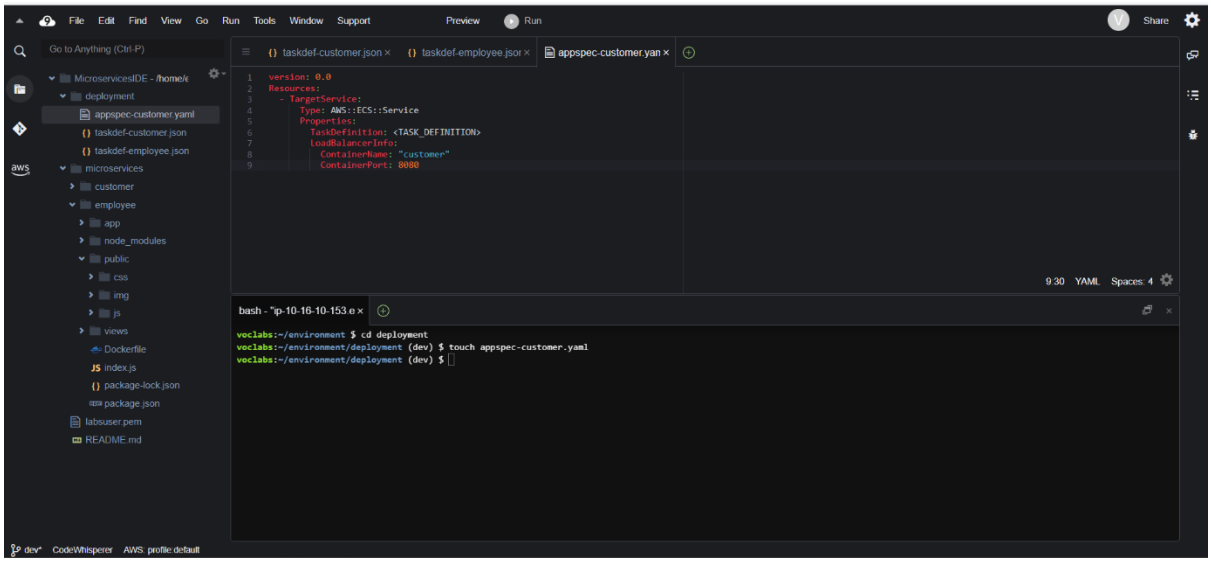
    LoadBalancerInfo:

      ContainerName: "customer"

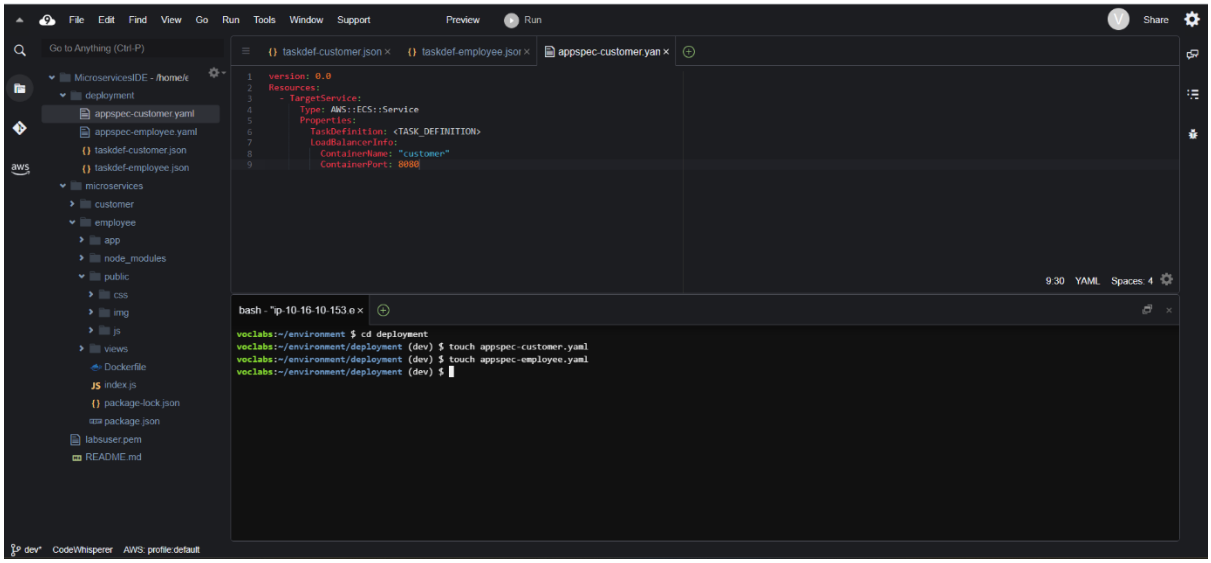
      ContainerPort: 8080

Note: This file is in YAML format. In YAML, indentation is important. Verify that the code in your file maintains the indentation levels as shown in the previous code block.

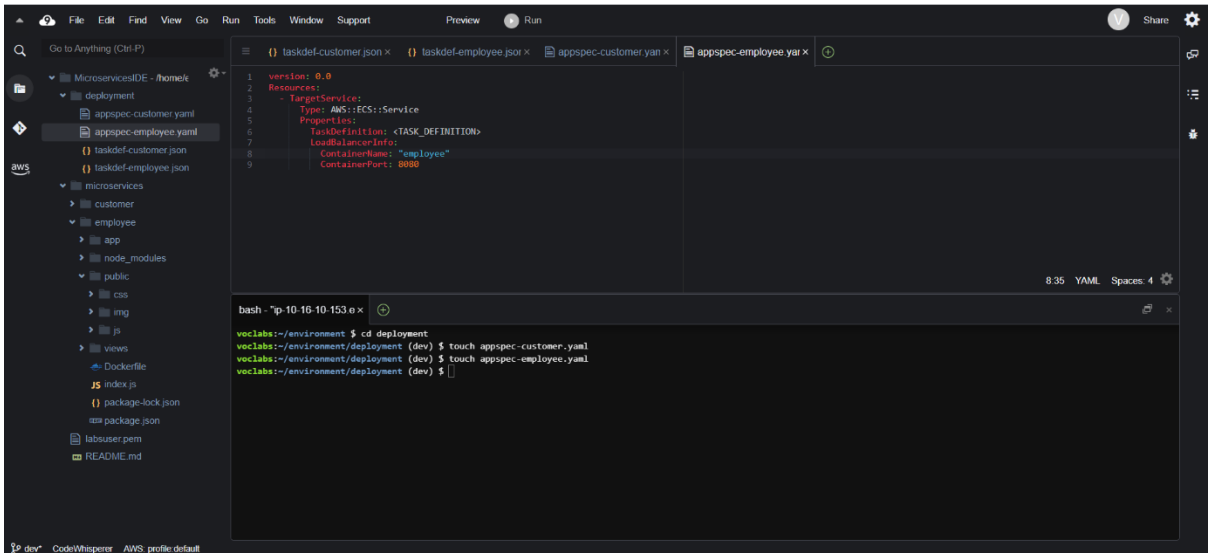
- **Save the changes.**



2. **In the same directory, create an AppSpec file for the *employee* microservice.**
    - **Name the file appspec-employee.yaml**



    - **The contents of the file should be the same as the *appspec-customer.yaml* file. However, change customer`` on the containerName line to be employee`**



Task 5.6: Update files and check them into CodeCommit

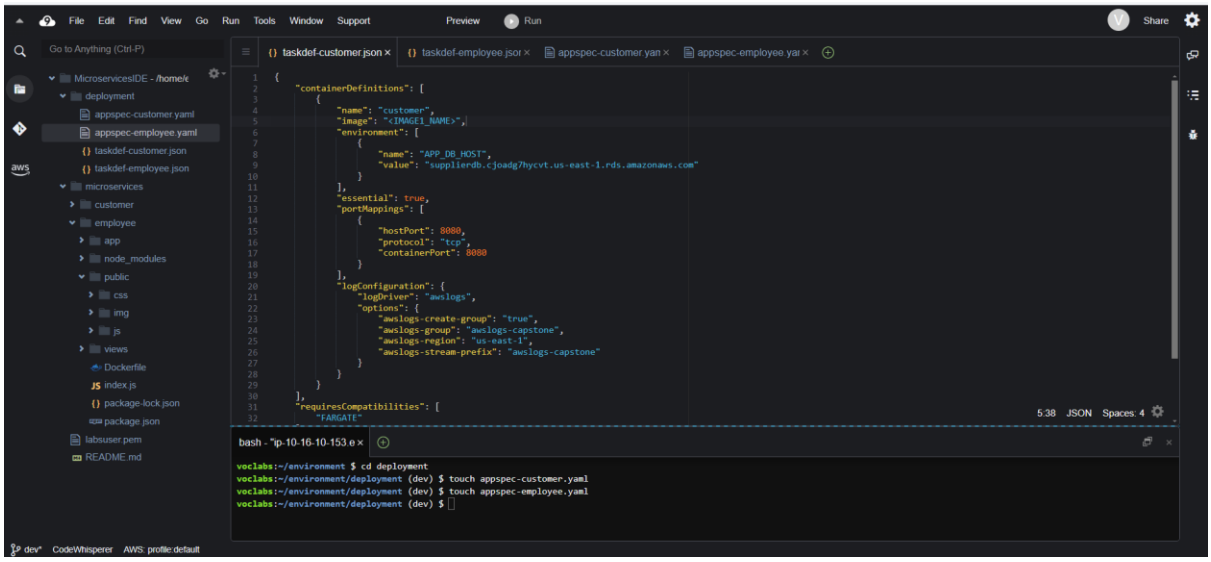
In this task, you will update the two task definition files. Then, you will push the four files that you created in the last two tasks into the *deployment* repository.

1. **Edit the taskdef-customer.json file.**
    - **Modify line 5 to match the following line:**

"image": "<IMAGE1_NAME>",

- **Save the change.**

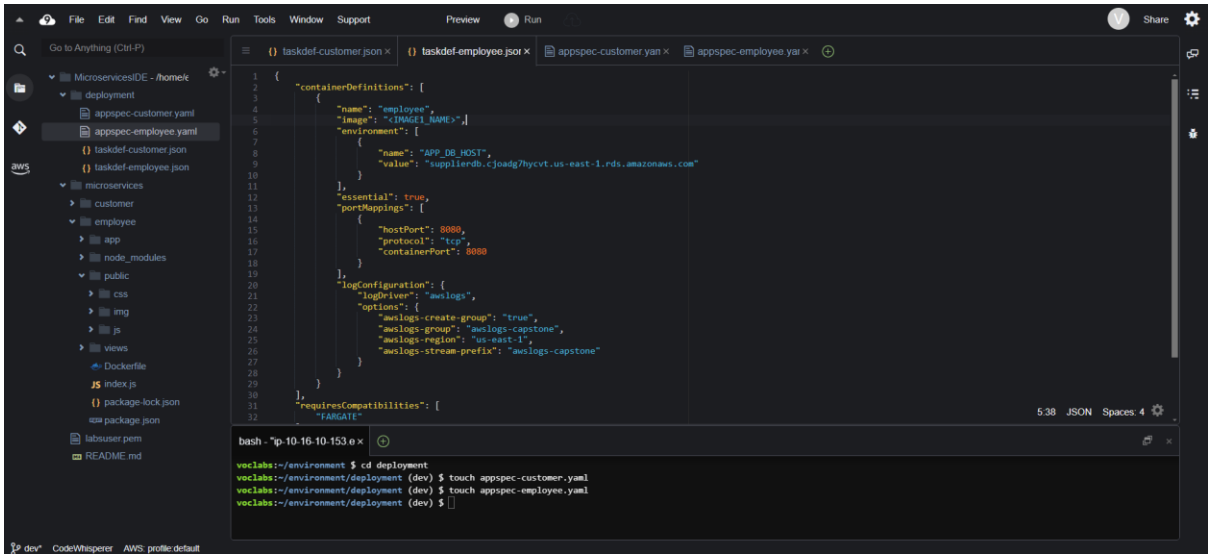
Analysis: <IMAGE1_NAME> is not a valid image name, which is why you originally set the image name to *customer* before running the AWS CLI command to register the first revision of the file with Amazon ECS. However, at this point in the project, it's important to set the image value to a placeholder text value. Later in this project, when you configure a pipeline, you will identify IMAGE1_NAME as placeholder text that can be dynamically updated. In summary, CodePipeline will set the correct image name dynamically at runtime.



2. **Edit the taskdef-employee.json file.**
    - **Modify line 5 to match the following line:**

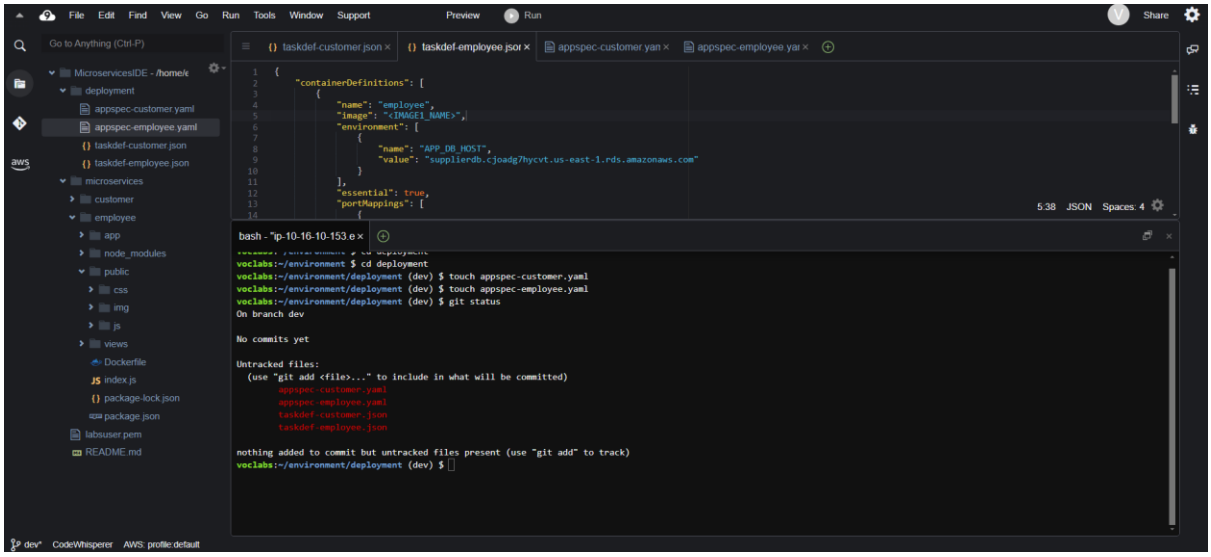
"image": "<IMAGE1_NAME>",

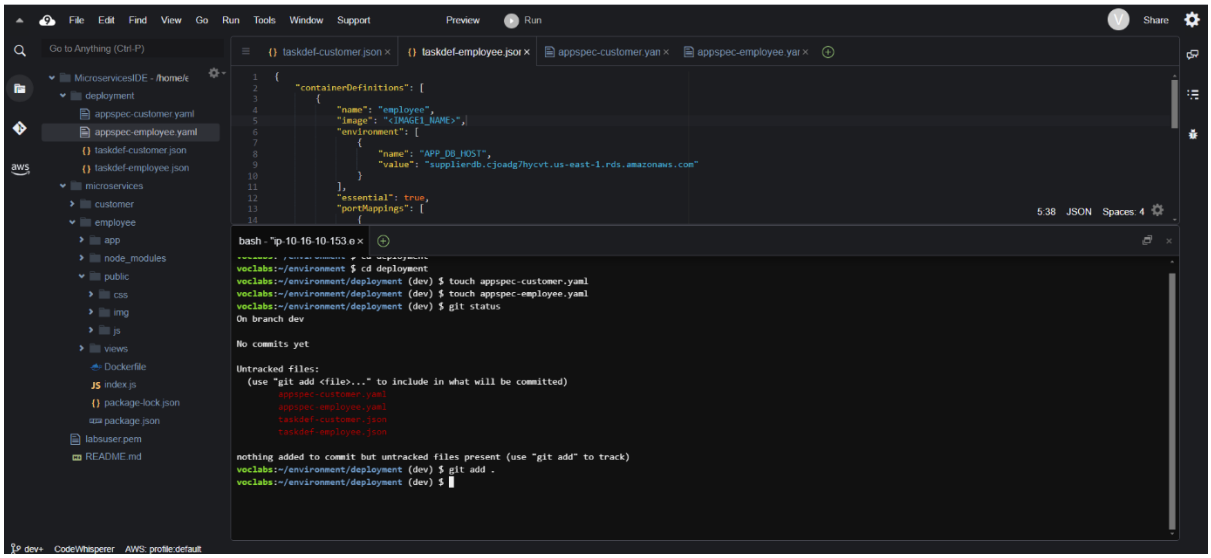
    - **Save the change.**

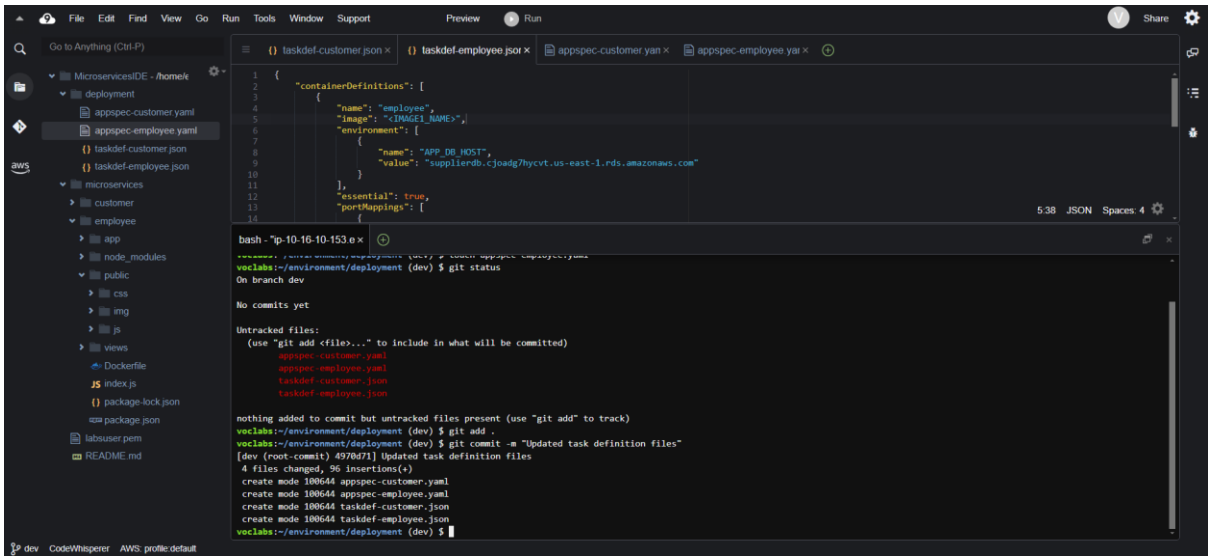


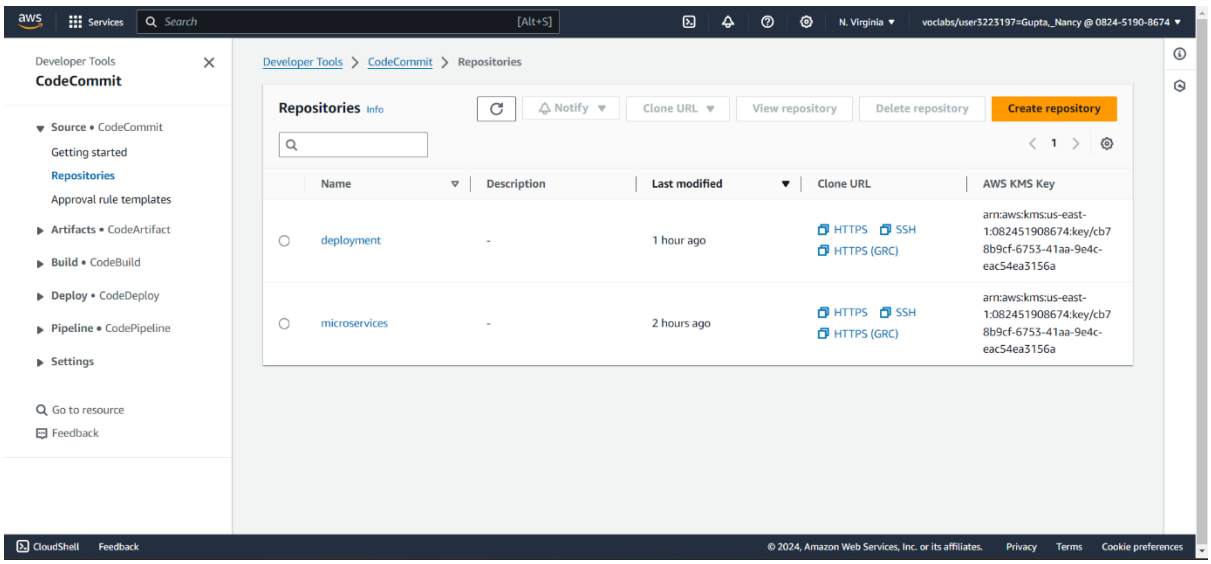
3. **Push all four files to CodeCommit.**

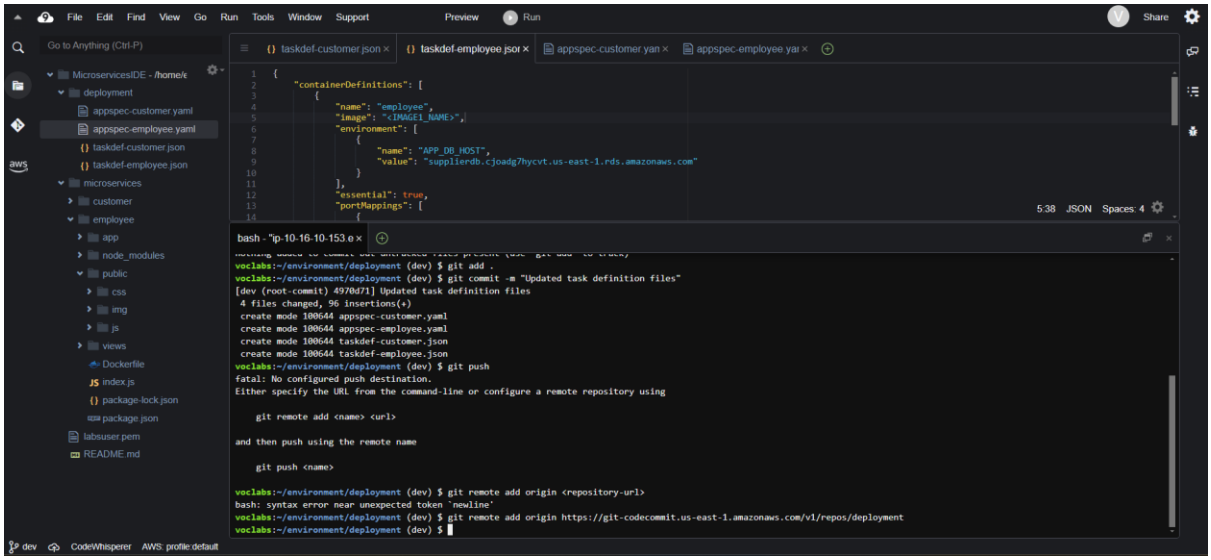
Note: Pushing the latest files to CodeCommit is essential. Later, when you create the CI/CD pipeline, the pipeline will pull these files from CodeCommit and use the details in them as instructions to deploy updates for your microservices to the Amazon ECS cluster.

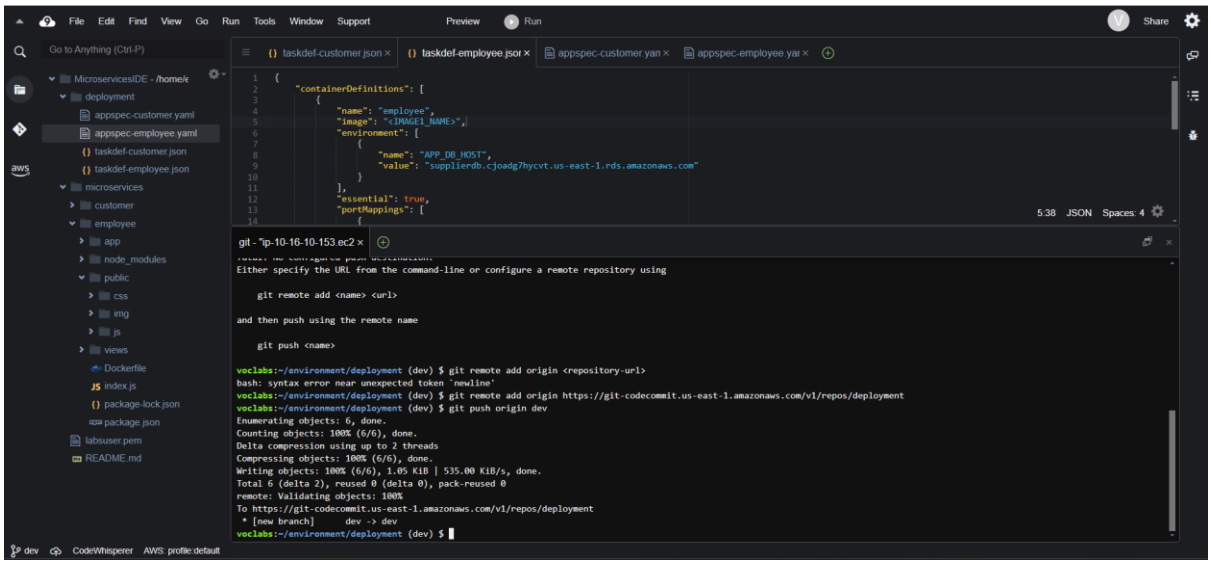


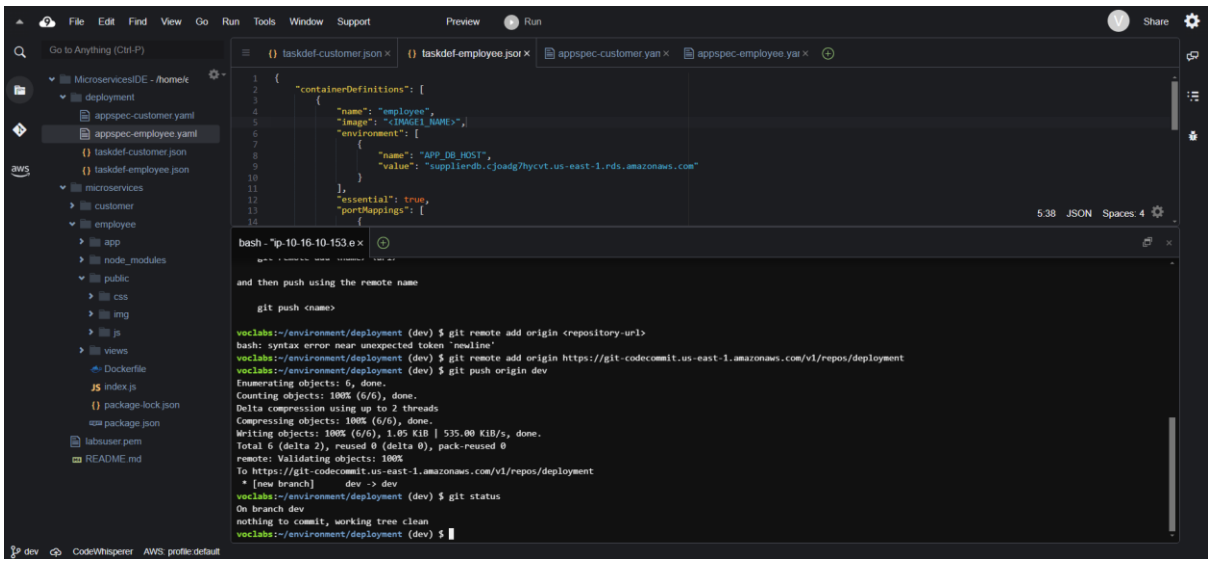

taskdef-customer.json
taskdef-employee.json
appspec-customer.yaml
appspec-employee.yaml
{
"containerDefinitions": [
{
"name": "employee",
"image": "<IMAGE1_NAME>",
"environment": [
{
"name": "APP_DB_HOST",
"value": "supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com"
}
],
"essential": true,
"portMappings": [
voclabs:~/environment $ cd deployment
voclabs:~/environment/deployment (dev) $ touch appspec-customer.yaml
voclabs:~/environment/deployment (dev) $ touch appspec-employee.yaml
voclabs:~/environment/deployment (dev) $ git status
On branch dev
No commits yet
Untracked files:
(use "git add <file>..." to include in what will be committed)
appspec-customer.yaml
appspec-employee.yaml
taskdef-customer.json
taskdef-employee.json
nothing added to commit but untracked files present (use "git add" to track)
voclabs:~/environment/deployment (dev) $ git add .
voclabs:~/environment/deployment (dev) $

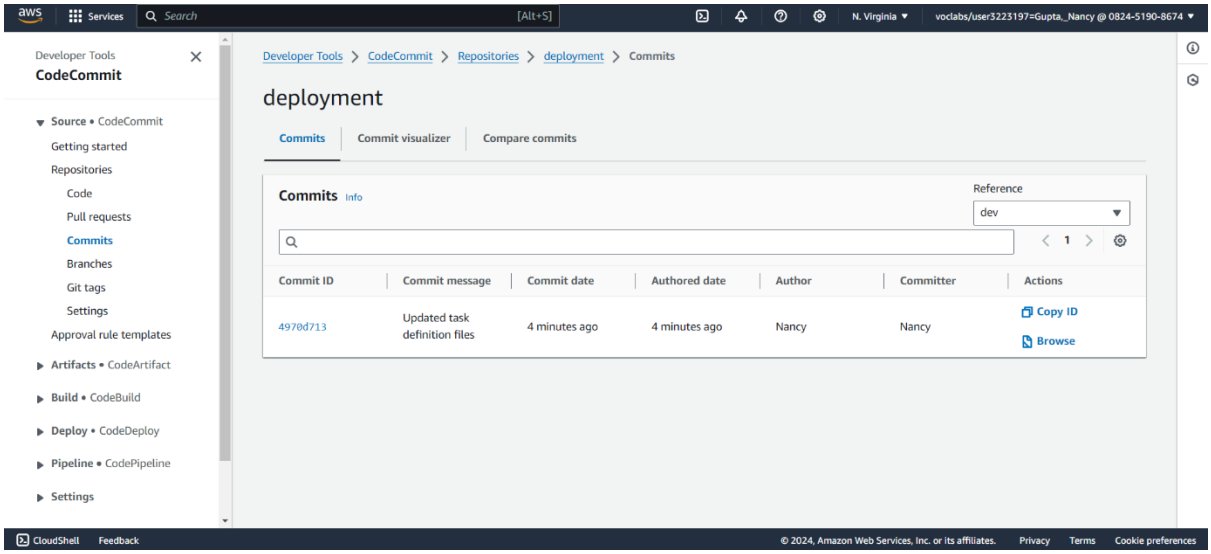

taskdef-customer.json
taskdef-employee.json
appspec-customer.yaml
appspec-employee.yaml
{
"containerDefinitions": [
{
"name": "employee",
"image": "<IMAGE1_NAME>",
"environment": [
{
"name": "APP_DB_HOST",
"value": "supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com"
}
],
"essential": true,
"portMappings": [
voclabs:~/environment/deployment (dev) $ git status
On branch dev
No commits yet
Untracked files:
(use "git add <file>..." to include in what will be committed)
appspec-customer.yaml
appspec-employee.yaml
taskdef-customer.json
taskdef-employee.json
nothing added to commit but untracked files present (use "git add" to track)
voclabs:~/environment/deployment (dev) $ git add .
voclabs:~/environment/deployment (dev) $ git commit -m "Updated task definition files"
[dev (root-commit) 4970d71] Updated task definition files
4 files changed, 96 insertions(+)
create mode 100644 appspec-customer.yaml
create mode 100644 appspec-employee.yaml
create mode 100644 taskdef-customer.json
create mode 100644 taskdef-employee.json
voclabs:~/environment/deployment (dev) $

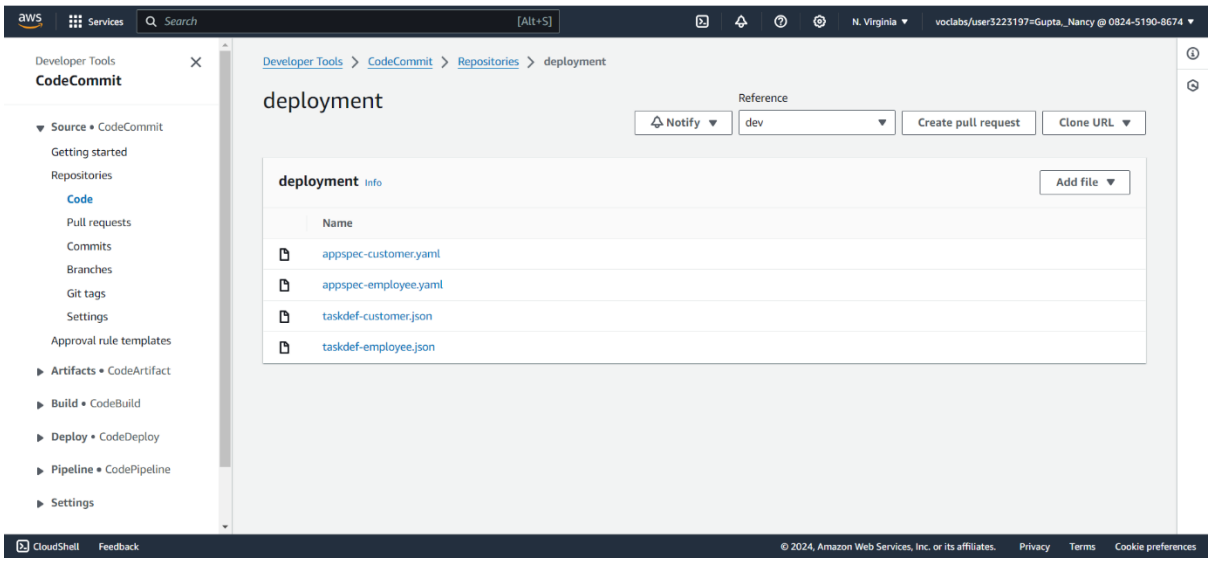

Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Developer Tools > CodeCommit > Repositories
Repositories Info
Notify
Clone URL
View repository
Delete repository
Create repository
Name
Description
Last modified
Clone URL
AWS KMS Key
deployment
1 hour ago
HTTPS
SSH
HTTPS (GRC)
arn:aws:kms:us-east-1:082451908674:key/cb78b9cf-6753-41aa-9e4c-eac54ea3156a
microservices
2 hours ago
HTTPS
SSH
HTTPS (GRC)
arn:aws:kms:us-east-1:082451908674:key/cb78b9cf-6753-41aa-9e4c-eac54ea3156a
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
taskdef-customer.json
taskdef-employee.jsor
appspec-customer.yan
appspec-employee.yar
MicroservicesIDE - /home/e
deployment
appspec-customer.yaml
appspec-employee.yaml
taskdef-customer.json
taskdef-employee.json
microservices
customer
employee
app
node_modules
public
css
img
js
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
{
"containerDefinitions": [
{
"name": "employee",
"image": "<IMAGE1_NAME>",
"environment": [
{
"name": "APP_DB_HOST",
"value": "supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com"
}
],
"essential": true,
"portMappings": [
{
5:38 JSON Spaces: 4
bash - "ip-10-16-10-153.e
voclabs:~/environment/deployment (dev) $ git add .
voclabs:~/environment/deployment (dev) $ git commit -m "Updated task definition files"
[dev (root-commit) 4970d71] Updated task definition files
4 files changed, 96 insertions(+)
create mode 100644 appspec-customer.yaml
create mode 100644 appspec-employee.yaml
create mode 100644 taskdef-customer.json
create mode 100644 taskdef-employee.json
voclabs:~/environment/deployment (dev) $ git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using
git remote add <name> <url>
and then push using the remote name
git push <name>
voclabs:~/environment/deployment (dev) $ git remote add origin <repository-url>
bash: syntax error near unexpected token `newline'
voclabs:~/environment/deployment (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
voclabs:~/environment/deployment (dev) $
dev
CodeWhisperer
AWS: profile:default


File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
taskdef-customer.json
taskdef-employee.jsor
appspec-customer.yan
appspec-employee.yar
MicroservicesIDE - /home/e
deployment
appspec-customer.yaml
appspec-employee.yaml
taskdef-customer.json
taskdef-employee.json
microservices
customer
employee
app
node_modules
public
css
img
js
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
{
"containerDefinitions": [
{
"name": "employee",
"image": "<IMAGE1_NAME>",
"environment": [
{
"name": "APP_DB_HOST",
"value": "supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com"
}
],
"essential": true,
"portMappings": [
{
5:38 JSON Spaces: 4
git - "ip-10-16-10-153.ec2
Either specify the URL from the command-line or configure a remote repository using
git remote add <name> <url>
and then push using the remote name
git push <name>
voclabs:~/environment/deployment (dev) $ git remote add origin <repository-url>
bash: syntax error near unexpected token `newline'
voclabs:~/environment/deployment (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
voclabs:~/environment/deployment (dev) $ git push origin dev
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.05 KiB | 535.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
* [new branch] dev -> dev
voclabs:~/environment/deployment (dev) $
dev
CodeWhisperer
AWS: profile:default


File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
taskdef-customer.json
taskdef-employee.jsor
appspec-customer.yan
appspec-employee.yar
MicroservicesIDE - /home/e
deployment
appspec-customer.yaml
appspec-employee.yaml
taskdef-customer.json
taskdef-employee.json
microservices
customer
employee
app
node_modules
public
css
img
js
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
{
"containerDefinitions": [
{
"name": "employee",
"image": "<IMAGE1_NAME>",
"environment": [
{
"name": "APP_DB_HOST",
"value": "supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com"
}
],
"essential": true,
"portMappings": [
{
5:38 JSON Spaces: 4
bash - "ip-10-16-10-153.e
and then push using the remote name
git push <name>
voclabs:~/environment/deployment (dev) $ git remote add origin <repository-url>
bash: syntax error near unexpected token `newline'
voclabs:~/environment/deployment (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
voclabs:~/environment/deployment (dev) $ git push origin dev
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.05 KiB | 535.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
* [new branch] dev -> dev
voclabs:~/environment/deployment (dev) $ git status
On branch dev
nothing to commit, working tree clean
voclabs:~/environment/deployment (dev) $
dev
CodeWhisperer
AWS: profile:default





Phase 6: Creating target groups and an Application Load Balancer

In this phase, you will create an Application Load Balancer, which provides an endpoint URL. This URL will act as the HTTPS entry point for customers and employees to access your application through a web browser. The load balancer will have listeners, which will have routing and access rules that determine which target group of running containers the user request should be directed to.

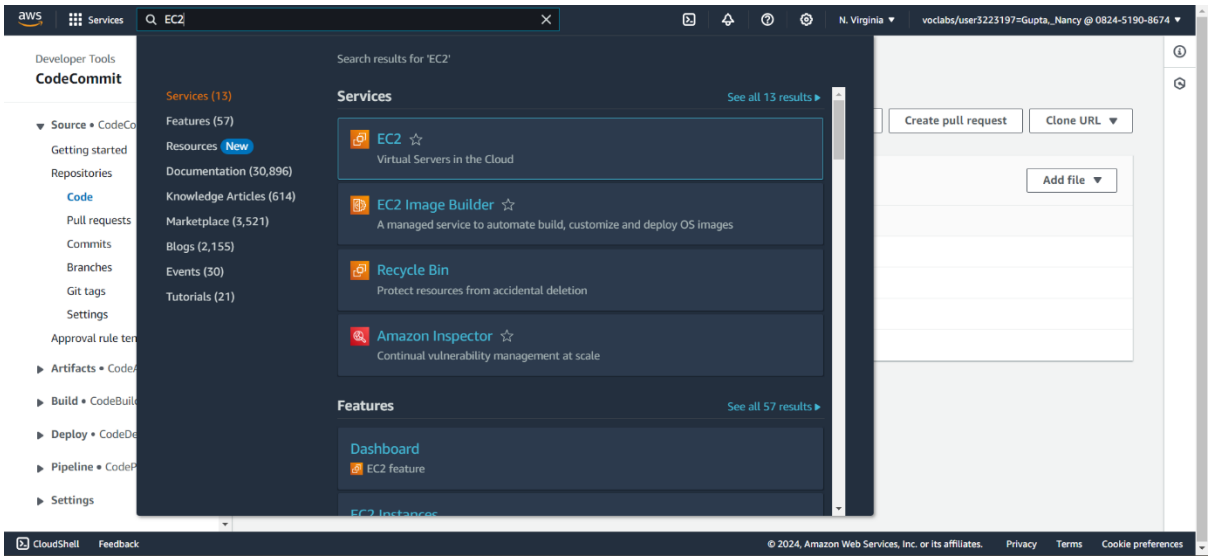
Task 6.1: Create four target groups

In this task, you will create four target groups—two for each microservice. Because you will configure a blue/green deployment, CodeDeploy requires two target groups for each deployment group.

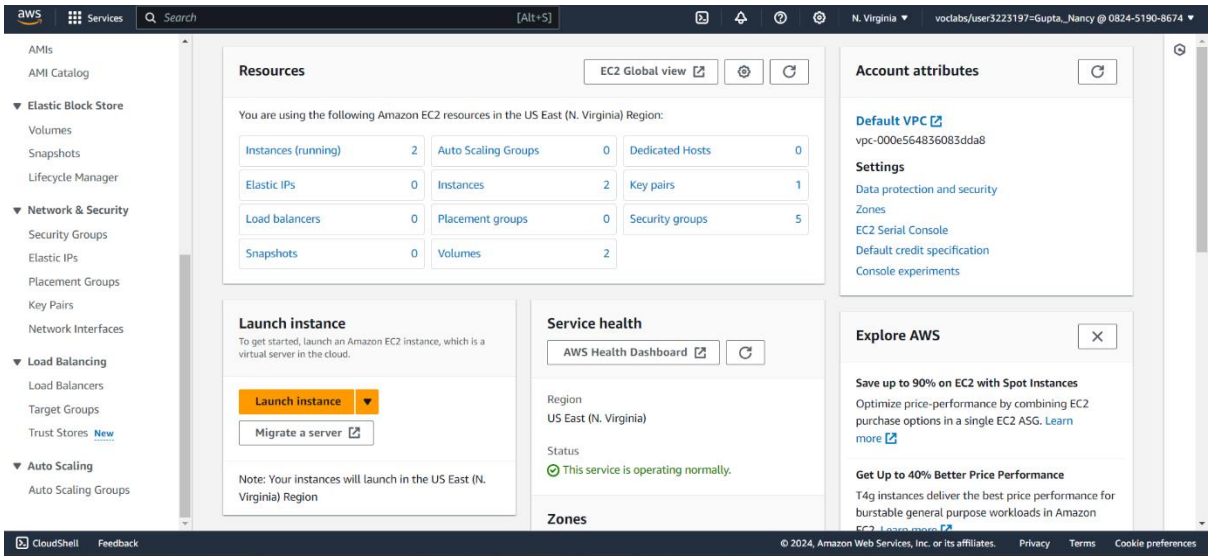
Note: Blue/green is a deployment strategy where you create two separate but identical environments. One environment (blue) runs the current application version, and one environment (green) runs the new application version. For more information, see Blue/Green Deployments in the *Overview of Deployment Options on AWS* whitepaper.

1. **Create the first target group for the *customer* microservice.**

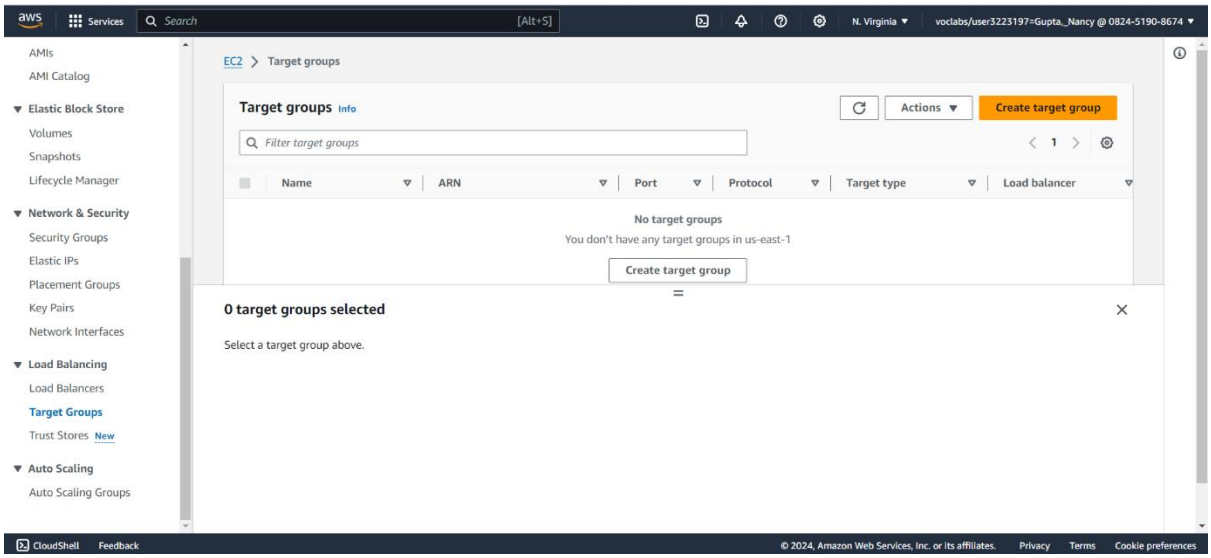
   - **Navigate to the Amazon EC2 console.**



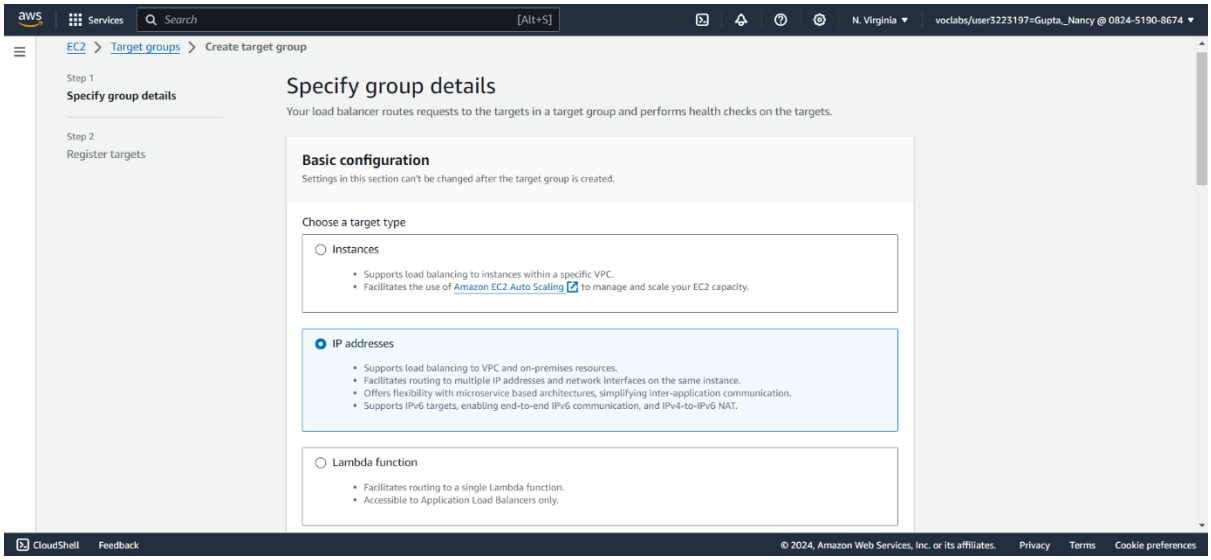
   - **In the navigation pane, choose Target Groups.**



   - **Choose Create target group and configure the following:**



- **Choose a target type: Choose IP addresses.**



- **Target group name: Enter customer-tg-one**