COURSE PROJECT

Building Microservices and a CI/CD Pipeline with AWS

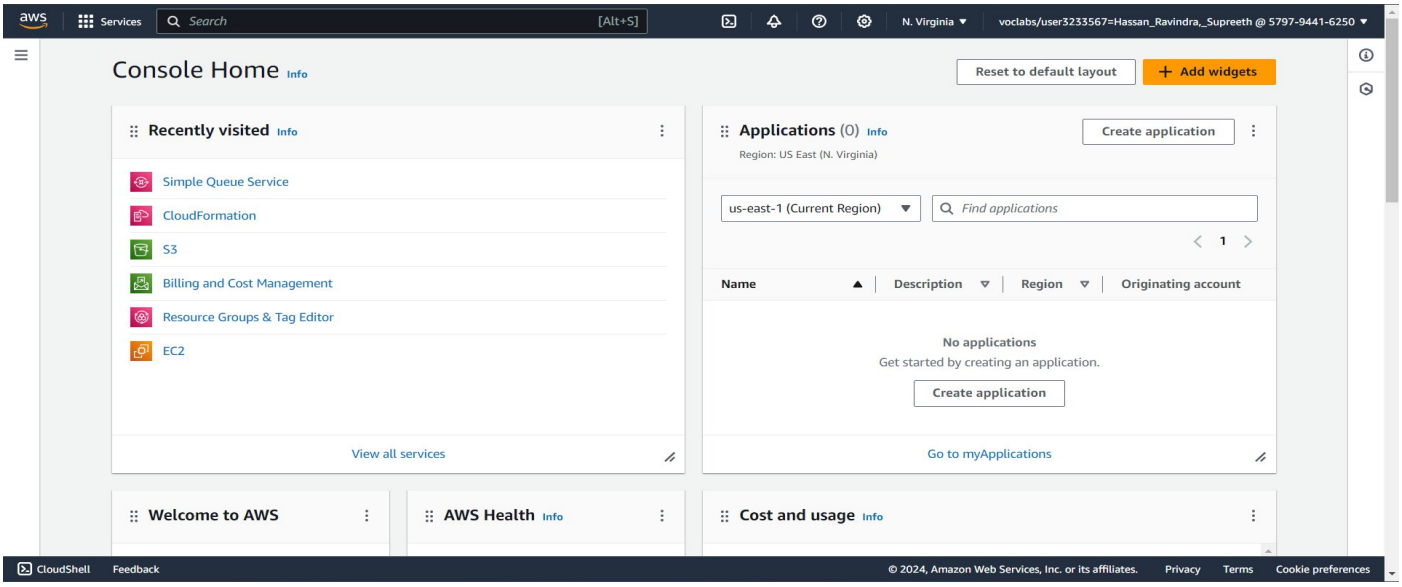
NAME: SUPREETH HASSAN RAVINDRA

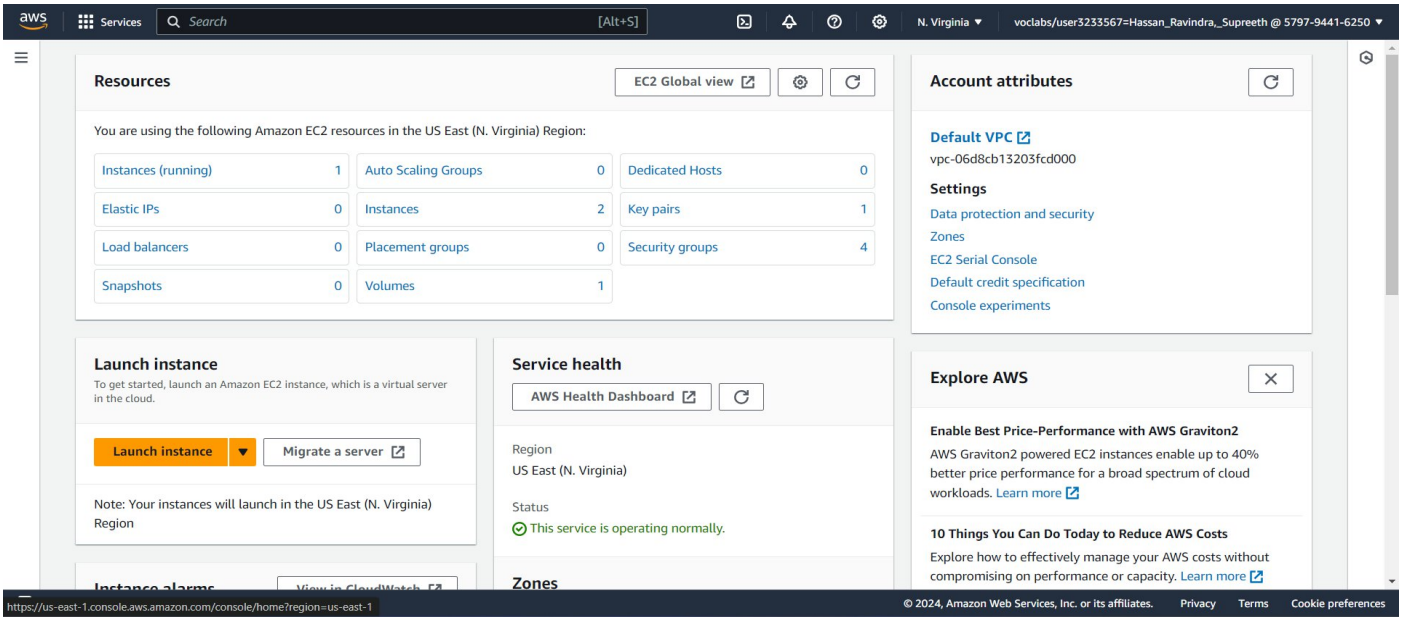
CWID: 20020533

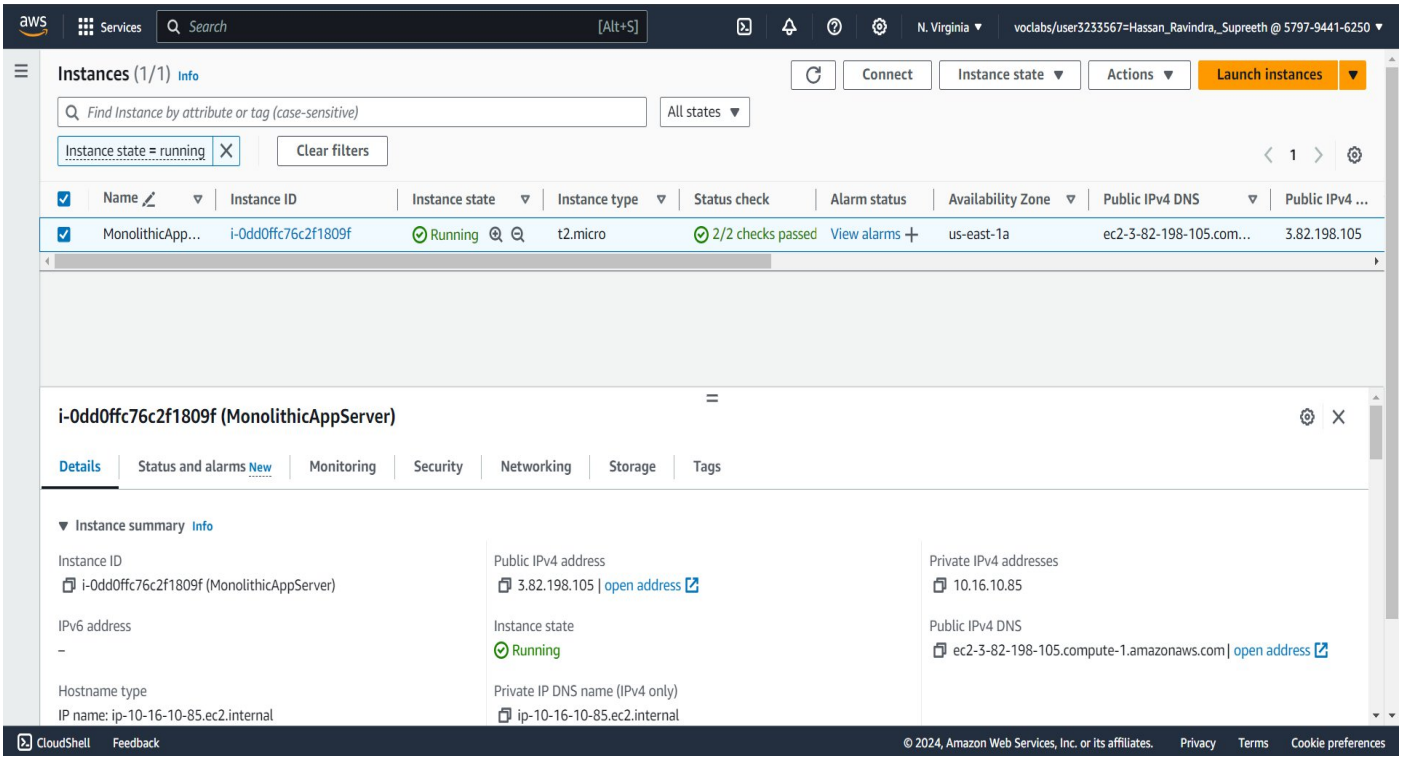
PART-1: PROJECT EXECUTION

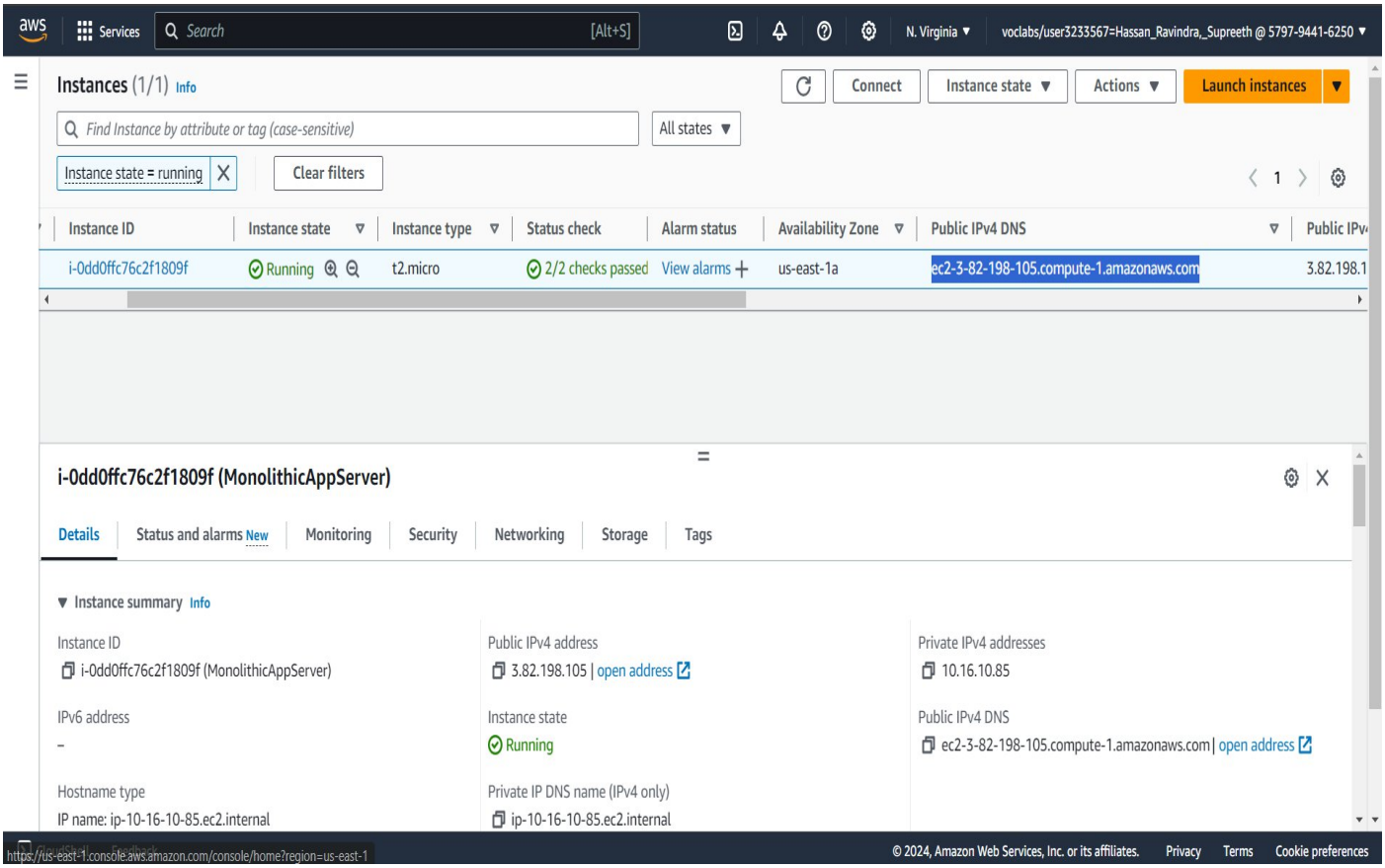
PHASE-2: Analyzing the infrastructure of the monolithic application

- Task 2.1: Loading the coffee suppliers website in the browser to see if it is available from the internet using the Public IPv4 address of the *MonolithicAppServer* instance in the EC2 console.







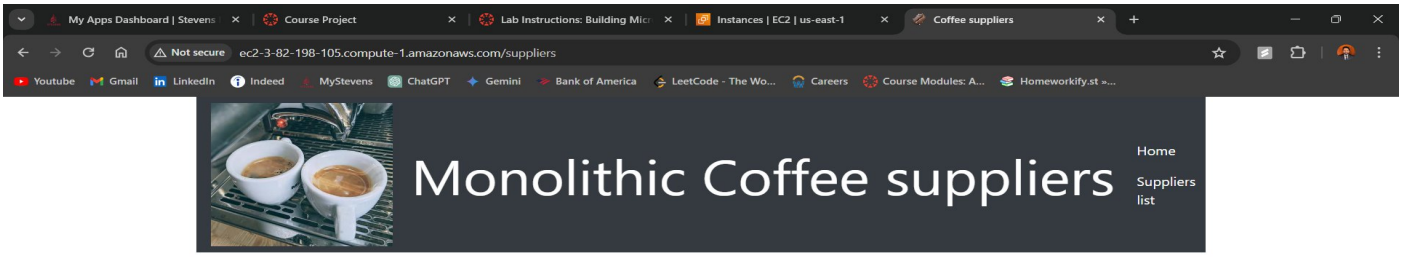


- Task 2.2: Testing the monolithic web application where I accessed the **List of Suppliers** to identify the URL path /suppliers, then added a new supplier via /supplier-add by filling out the required fields and edited an existing supplier with a URL path supplier-update/1 where I modified the record and saved the changes confirming successful updates to the database.

Welcome

Use this app to keep track of your coffee suppliers

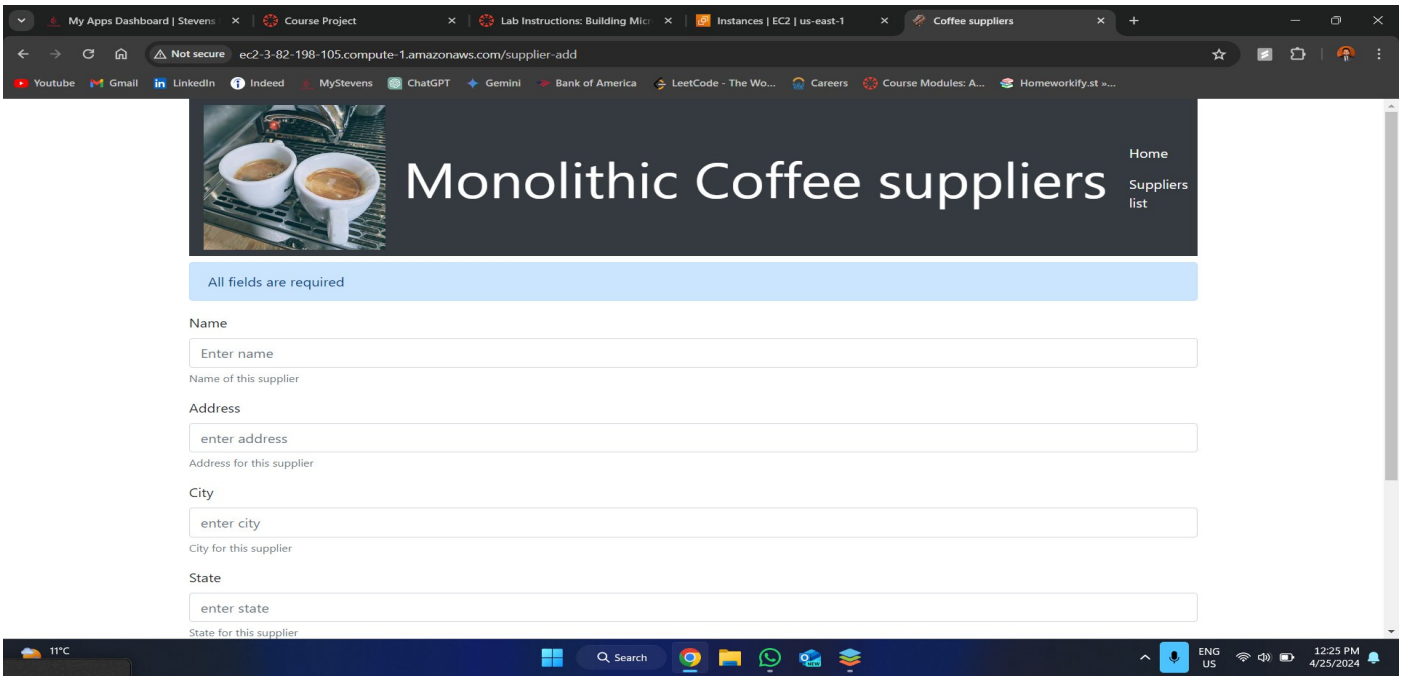
List of suppliers



All suppliers

Name	Address	City	State	Email	Phone

Add a new supplier



All fields are required

Name

Supreeth H R

Name of this supplier

Address

56 Poplar Street, Apt #2

Address for this supplier

City

Jersey City

City for this supplier

State

New Jersey

State for this supplier

All suppliers

Name	Address	City	State	Email	Phone	
Supreeth H R	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263	edit

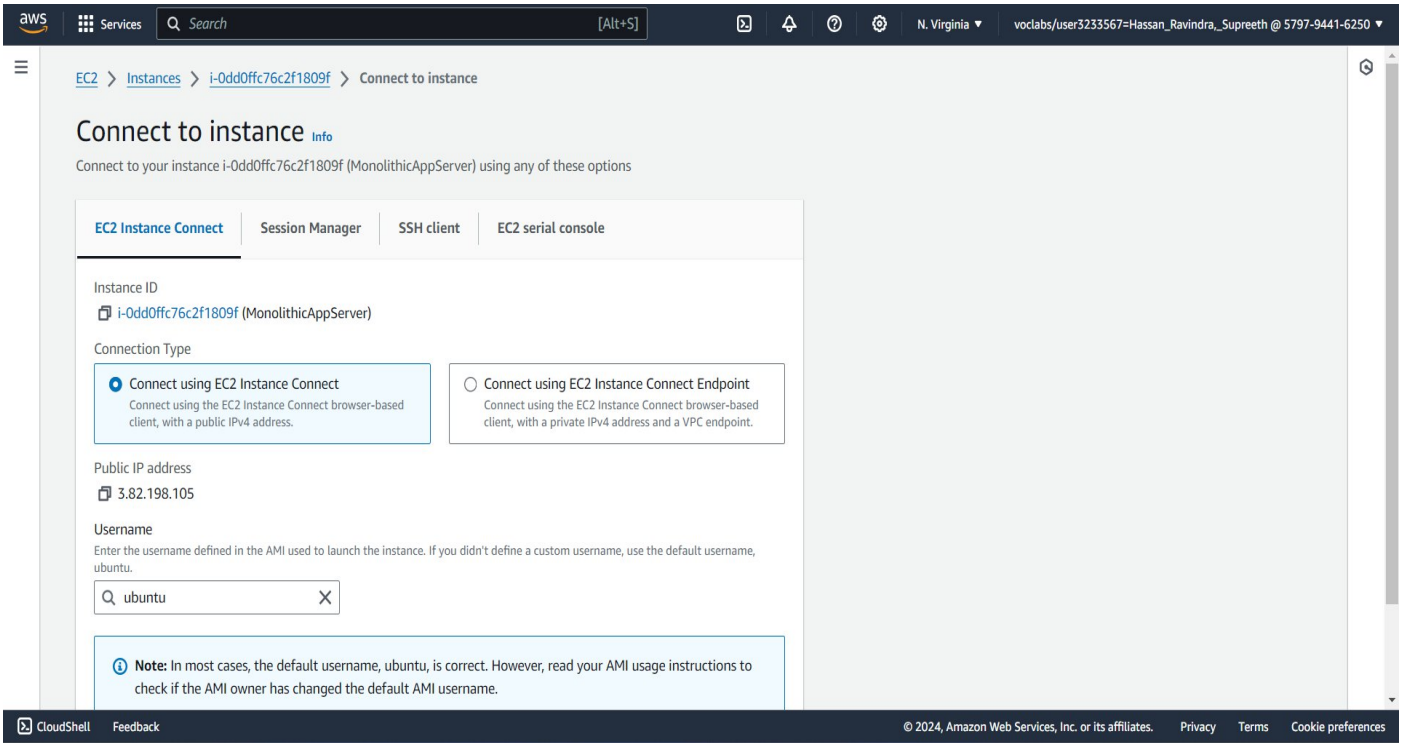
Add a new supplier

All suppliers

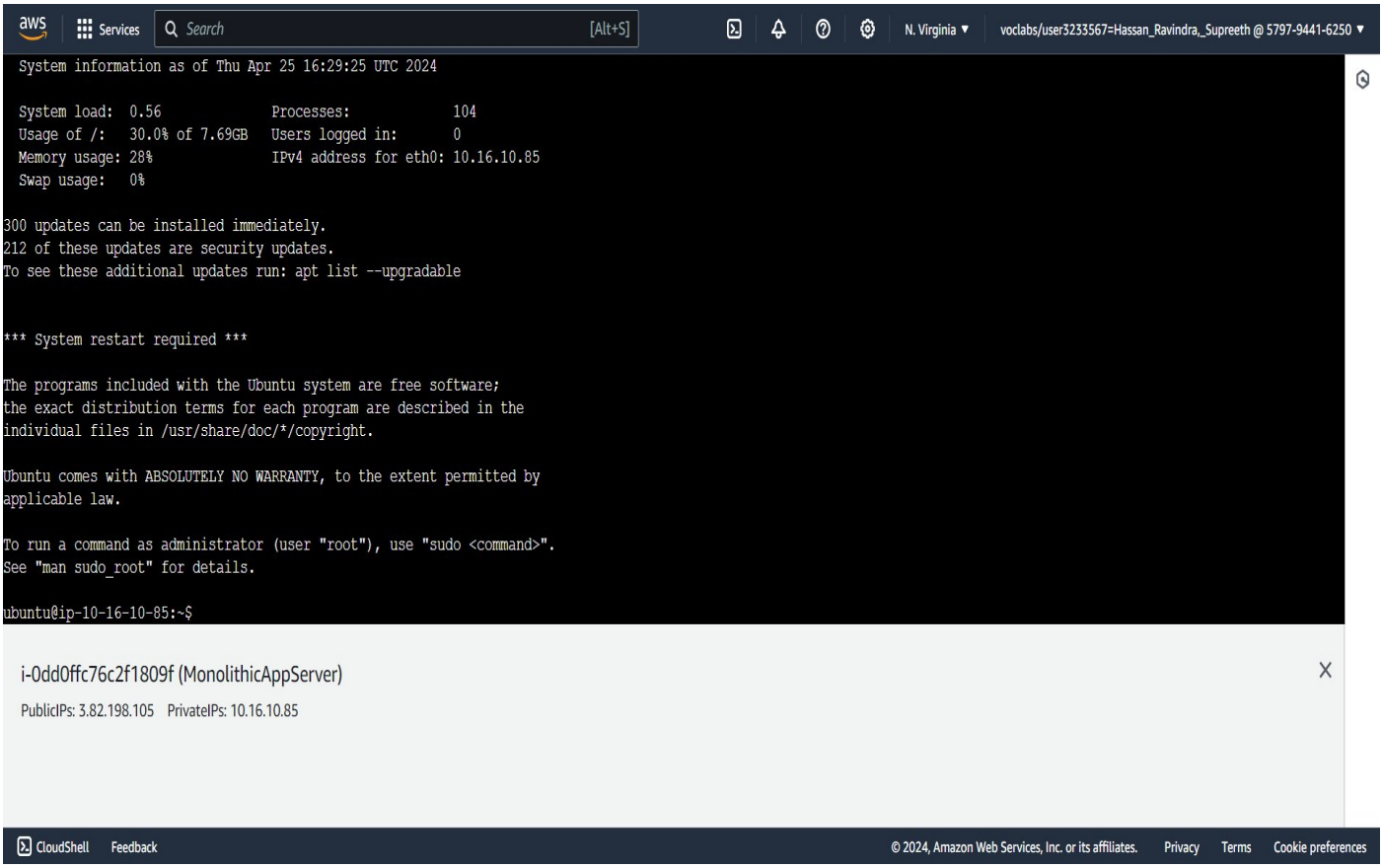
Name	Address	City	State	Email	Phone	
Supreeth Hassan Ravindra	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263	edit

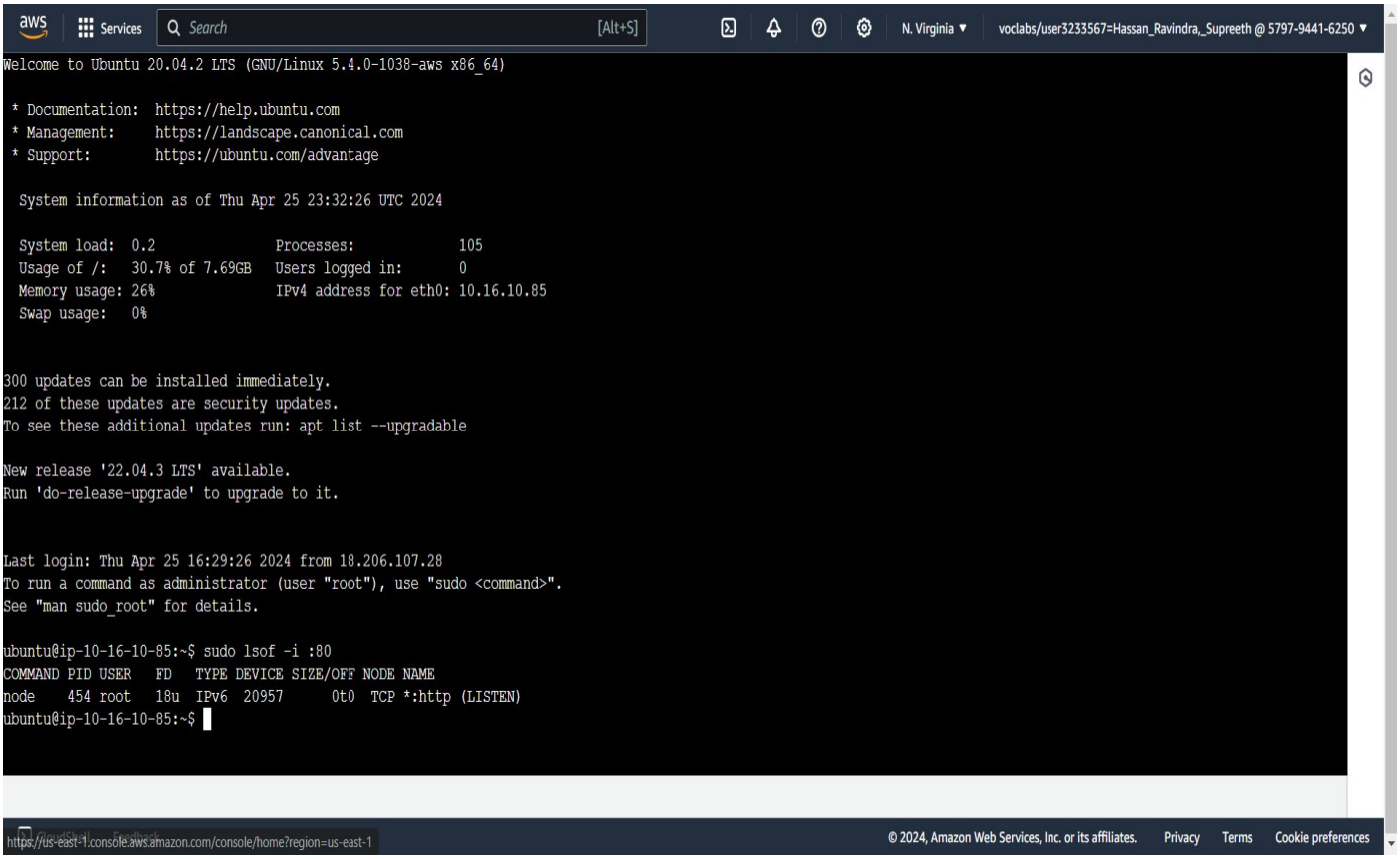
Add a new supplier

- Task 2.3: I used EC2 Instance Connect to connect to the MonolithicAppServer instance.

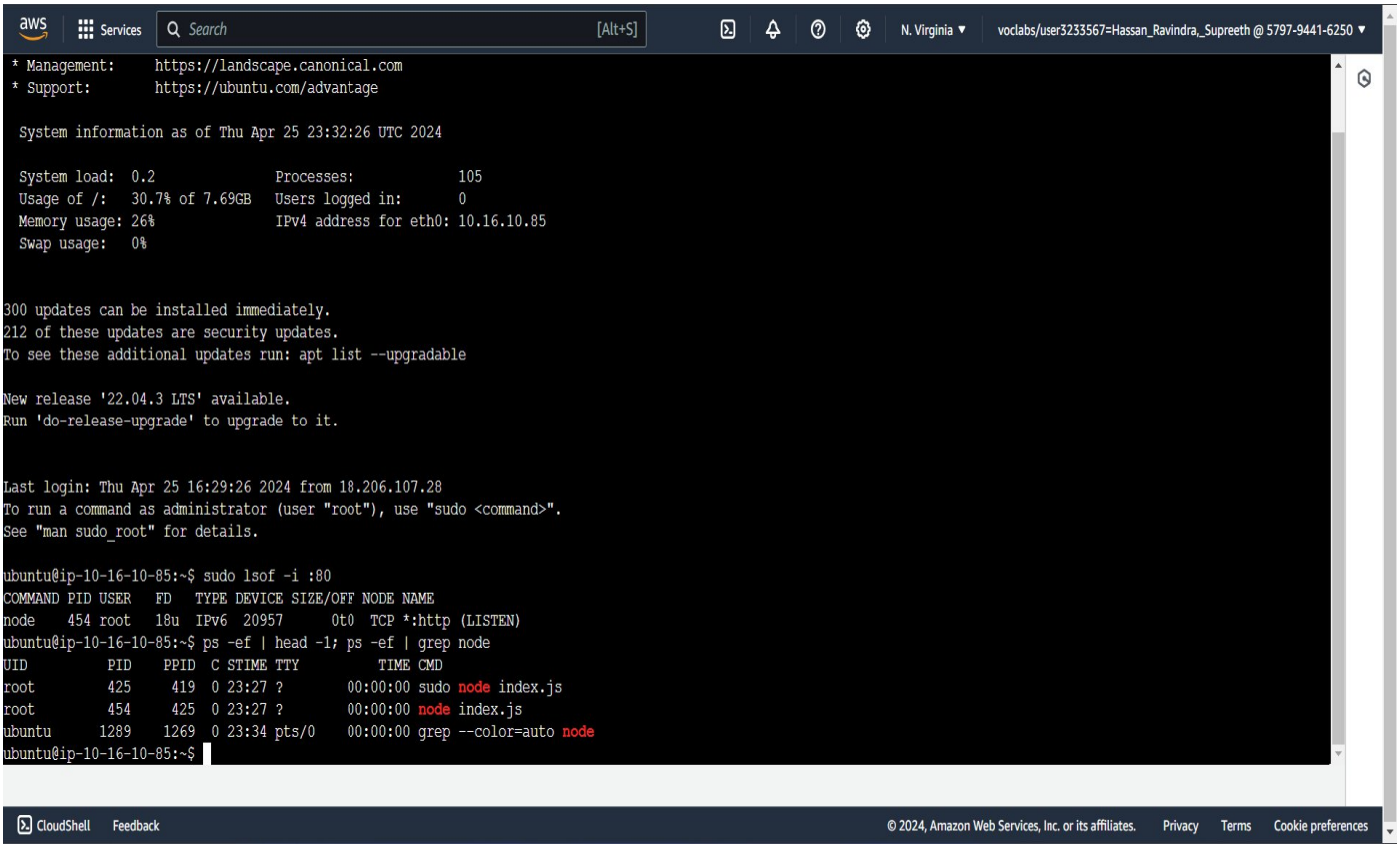


- Running **sudo lsof -i :80**. I observed that the node daemon used port 80 with the HTTP protocol.



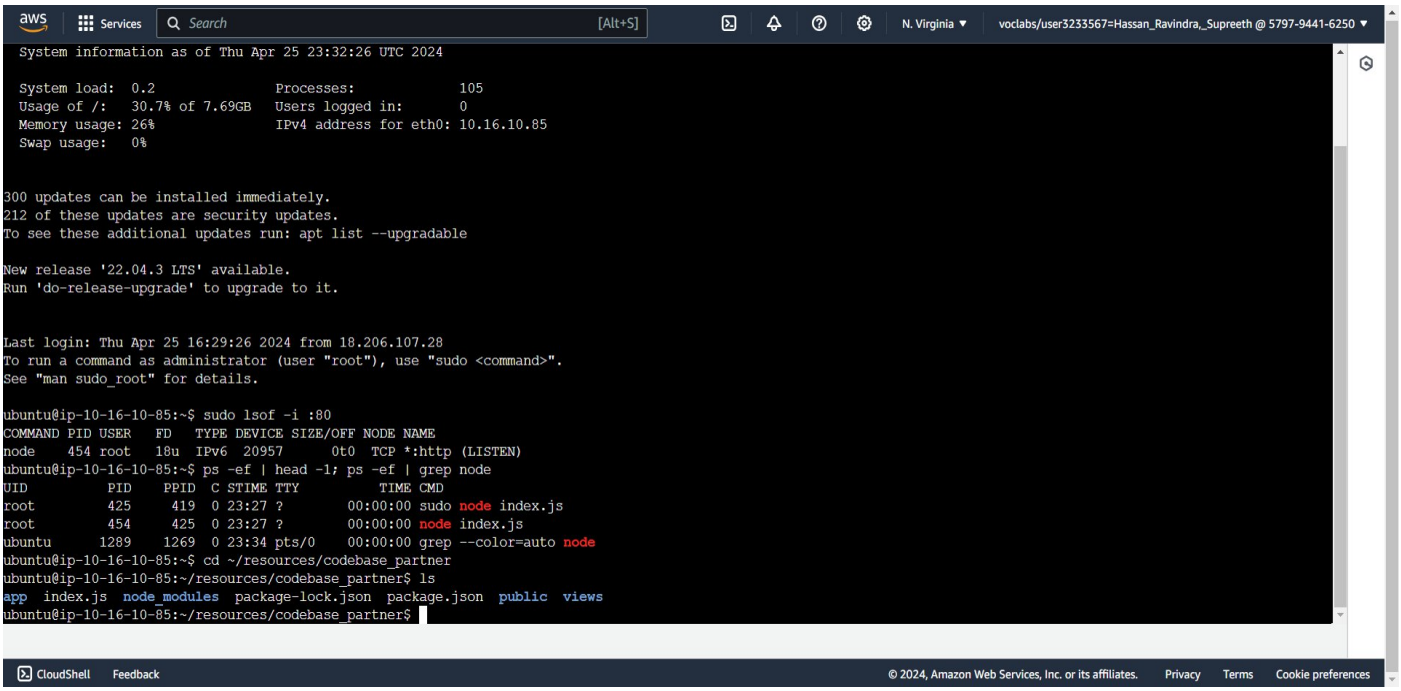


- Using **ps -ef | head -1; ps -ef | grep node**, I identified the user running the node process and checked whether the node Process ID (PID) matched any from the previous command output.

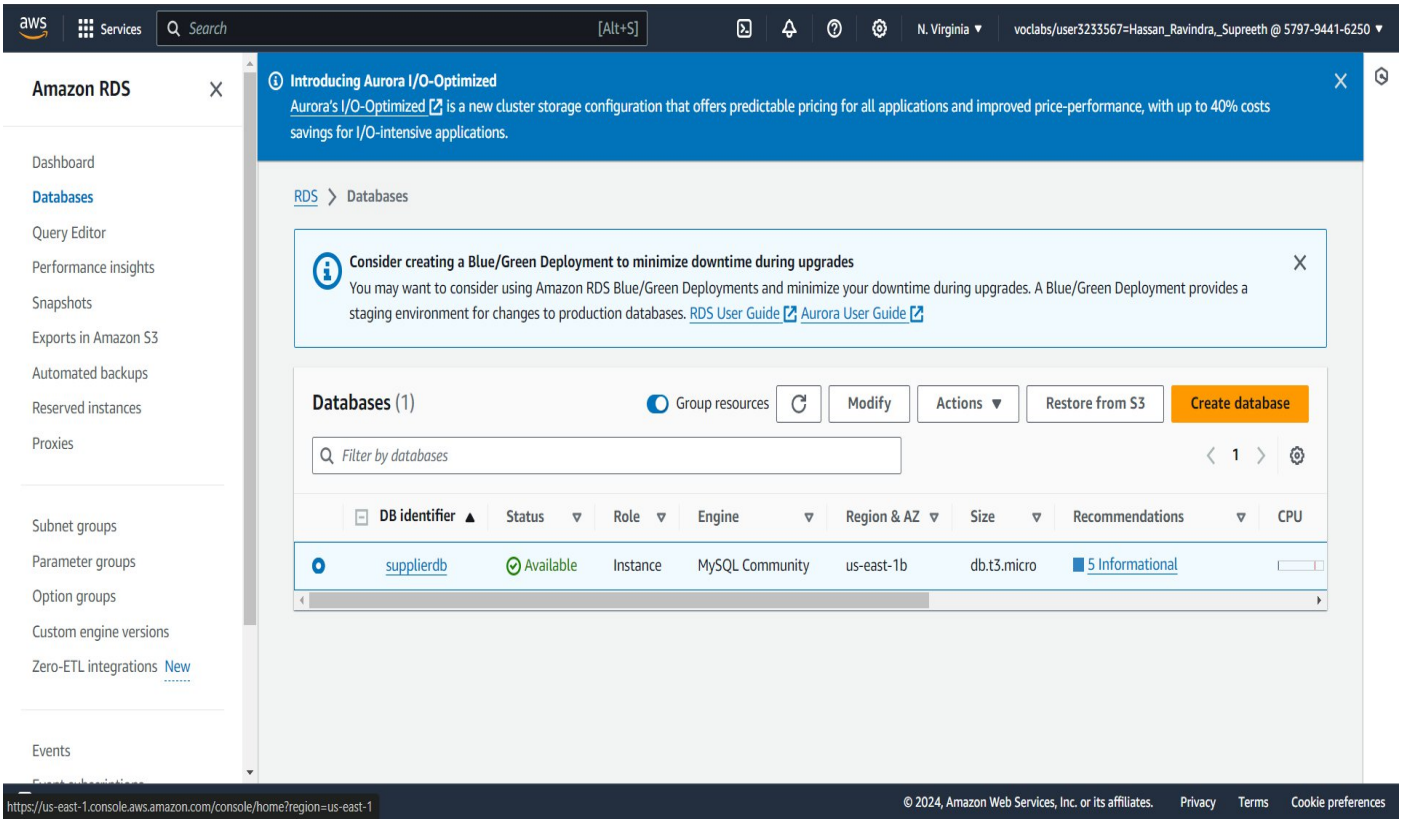


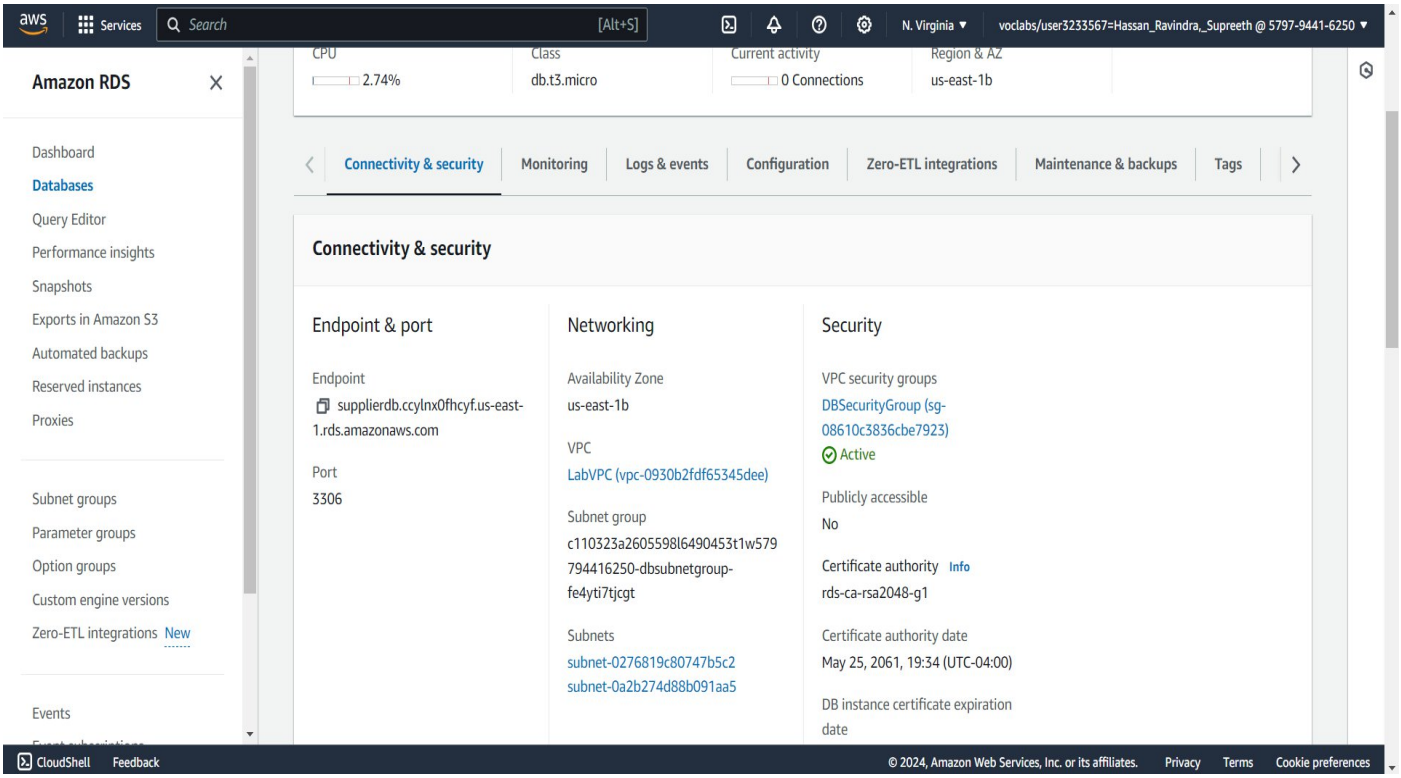
- To understand the structure, I navigated to the codebase directory with

cd ~/resources/codebase_partner and checked the contents with **ls** confirming the presence of index.js which contains the base application logic.

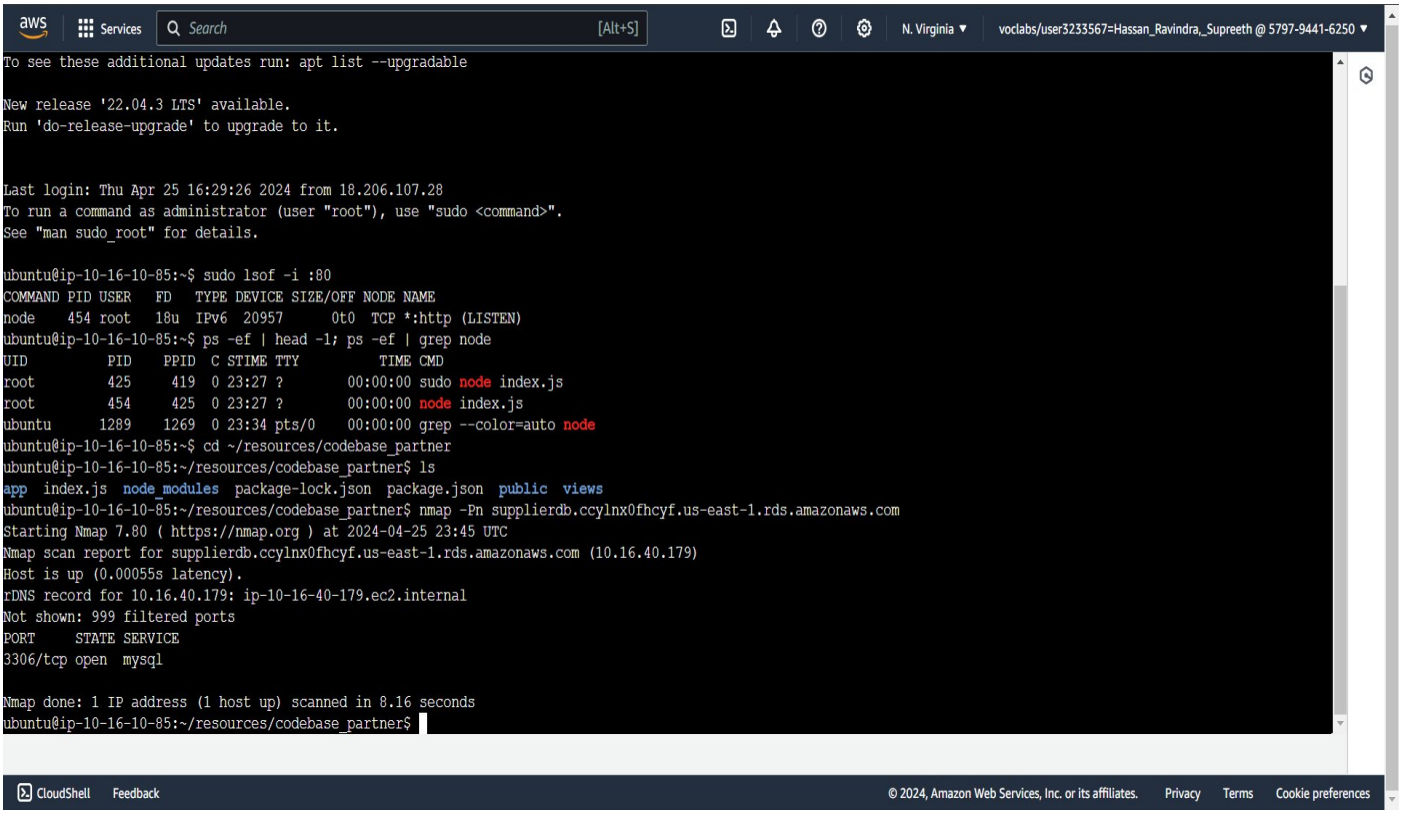


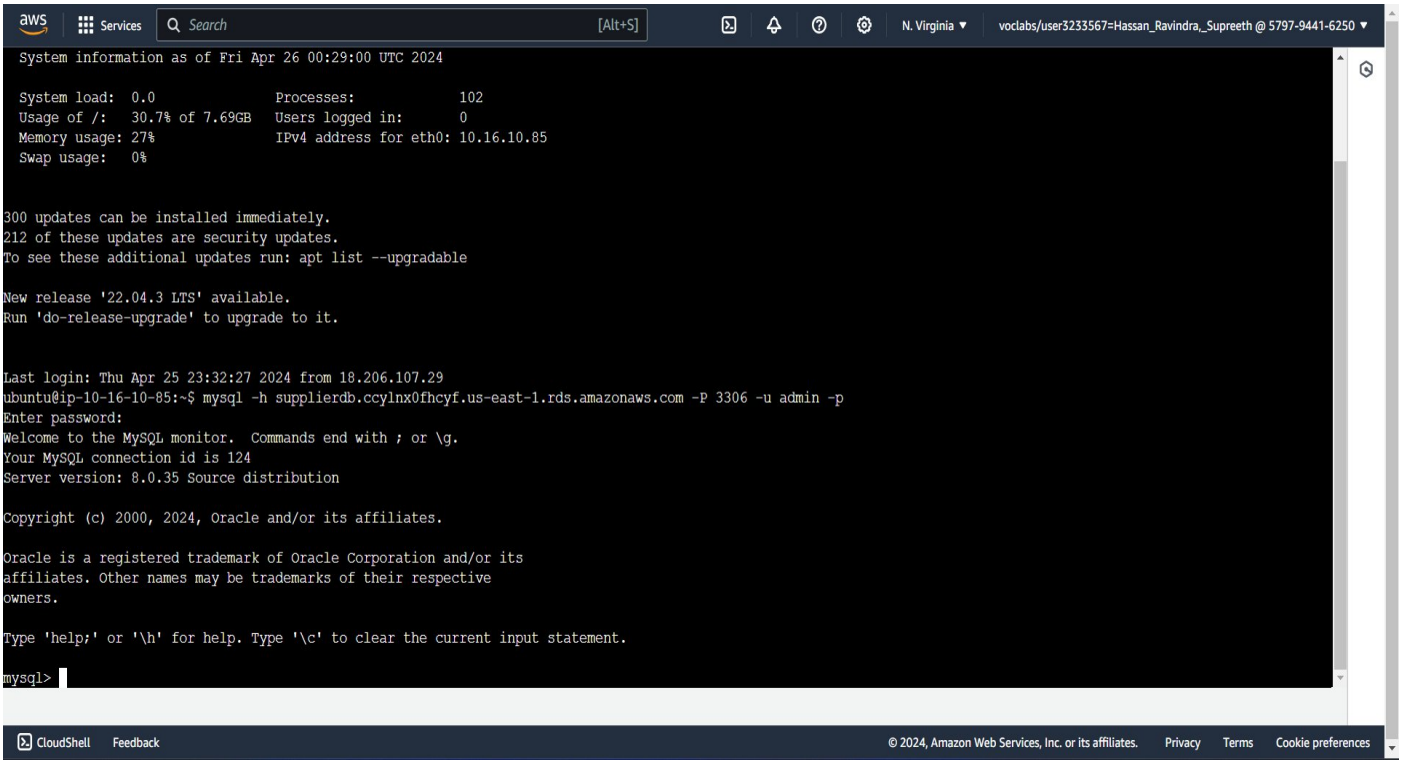
- I connected to the RDS database where the node application stores its data. I found and copied the RDS endpoint from the lab environment and then used **nmap -Pn** to ensure the database was accessible from the MonolithicAppServer instance on the standard MySQL port.

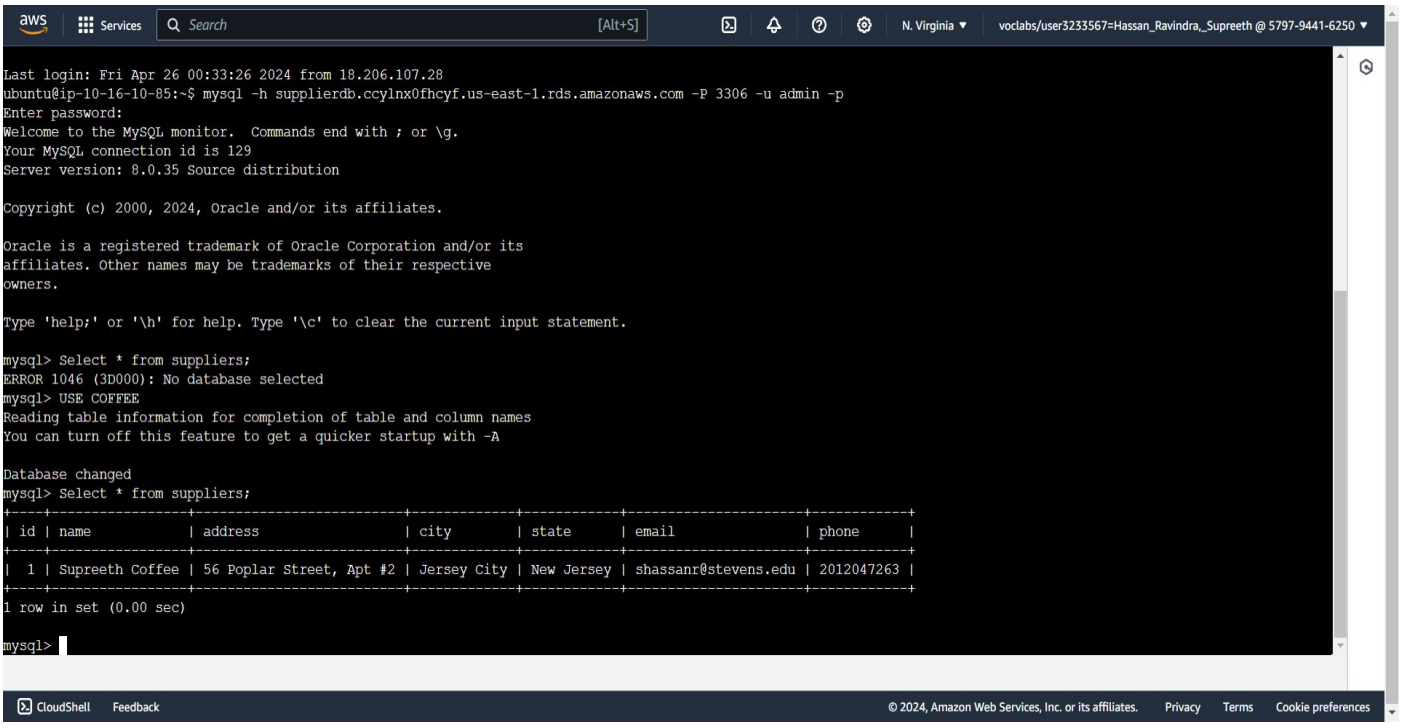




- Using the pre-installed MySQL client on the MonolithicAppServer instance, I connected to the RDS database with the following values:
  Username: admin
  Password: lab-password
- I ran SQL commands to check the data in the COFFEE database specifically the suppliers table where I found the supplier entries added earlier.

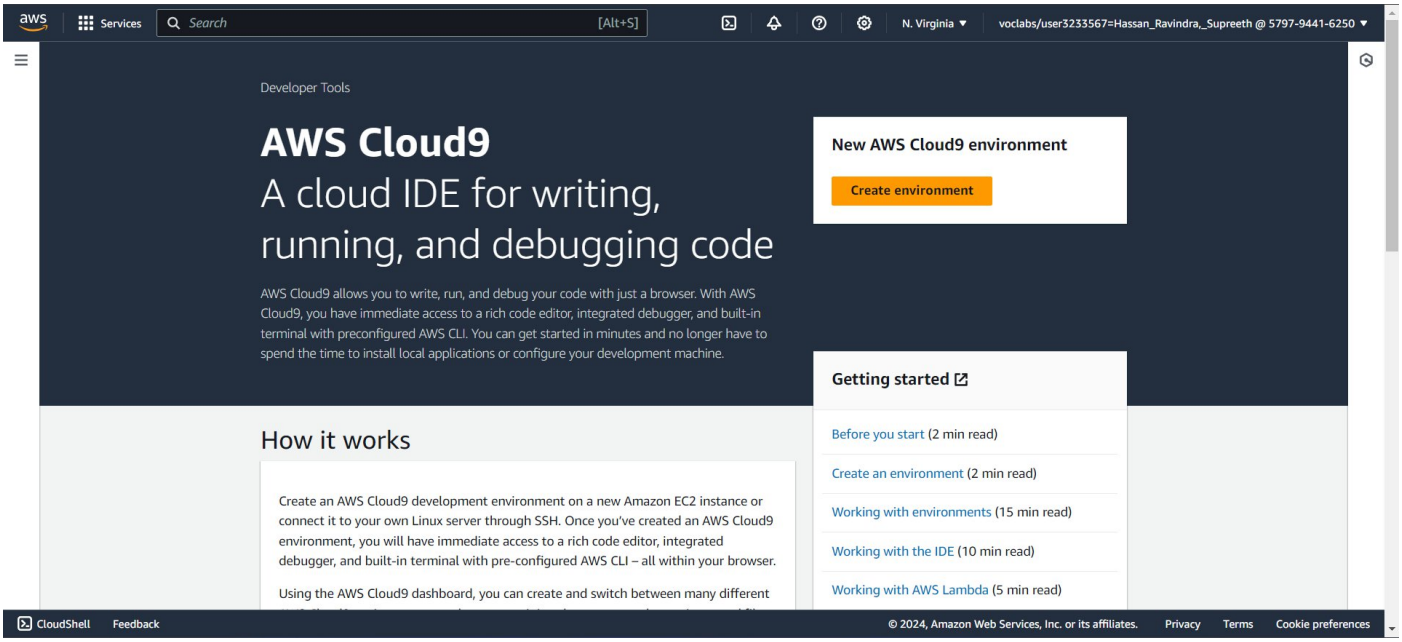


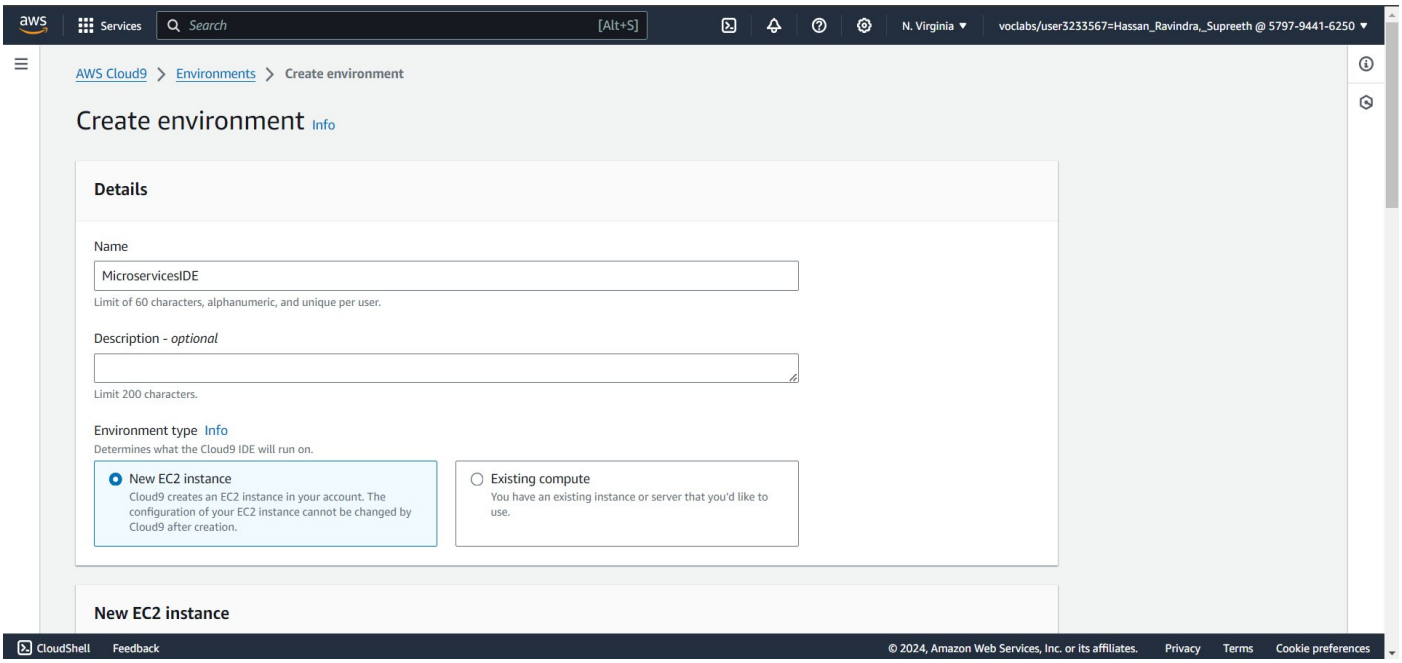


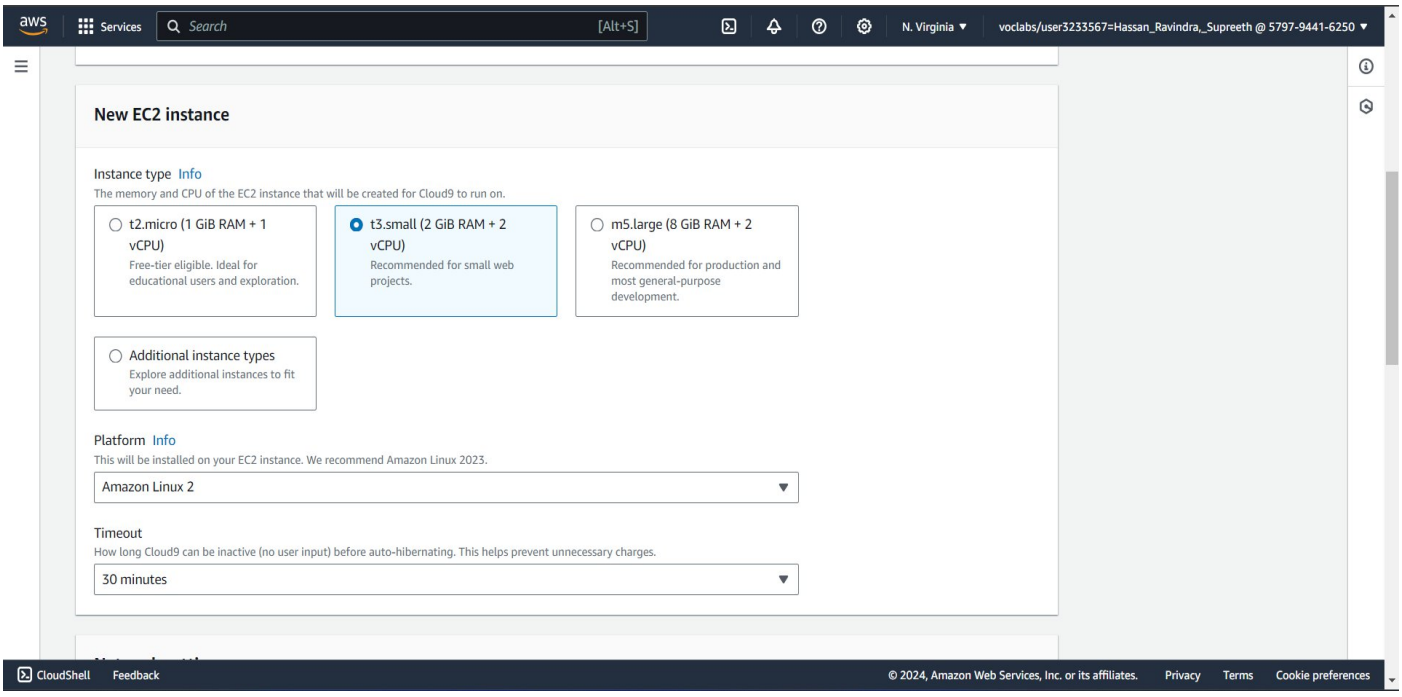


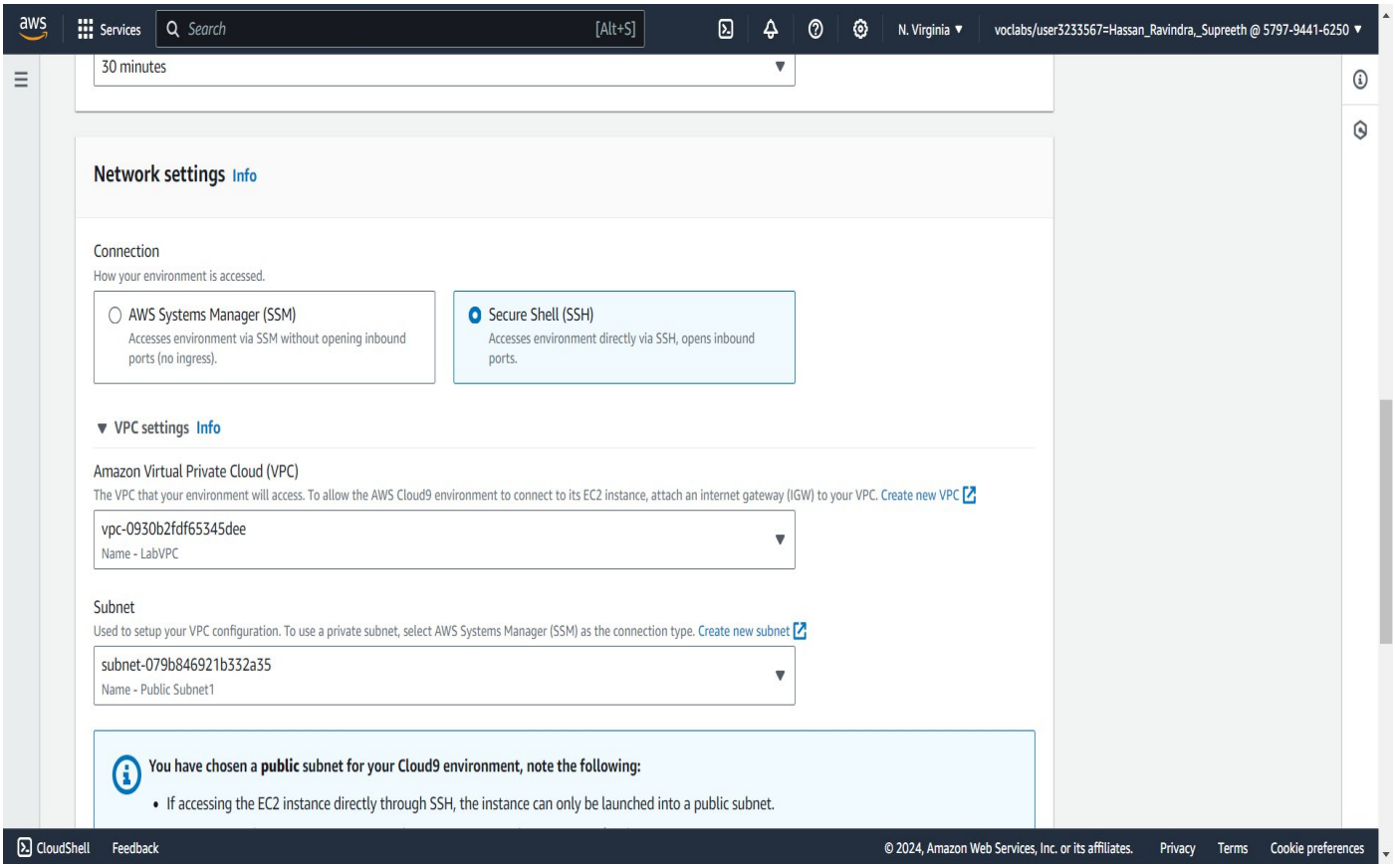
PHASE-3: Creating a development environment and checking code into a Git repository

- Task 3.1: Creating an AWS Cloud9 IDE as my work environment. I created AWS Cloud9 instance named MicroserviceIDE with all the specified specification in the lab instructions.

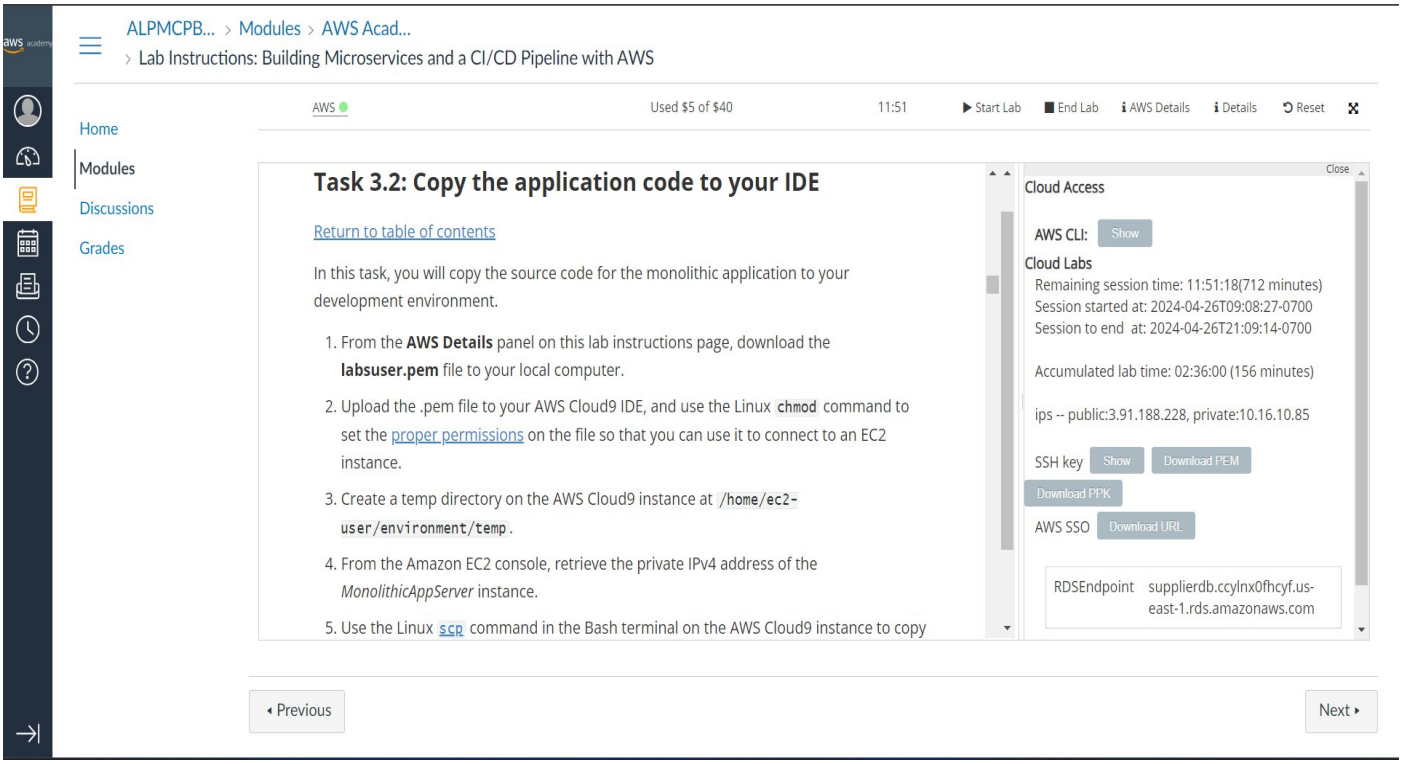

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
AWS Cloud9
A cloud IDE for writing, running, and debugging code
AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.
New AWS Cloud9 environment
Create environment
Getting started
Before you start (2 min read)
Create an environment (2 min read)
Working with environments (15 min read)
Working with the IDE (10 min read)
Working with AWS Lambda (5 min read)
How it works
Create an AWS Cloud9 development environment on a new Amazon EC2 instance or connect it to your own Linux server through SSH. Once you've created an AWS Cloud9 environment, you will have immediate access to a rich code editor, integrated debugger, and built-in terminal with pre-configured AWS CLI – all within your browser.
Using the AWS Cloud9 dashboard, you can create and switch between many different
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

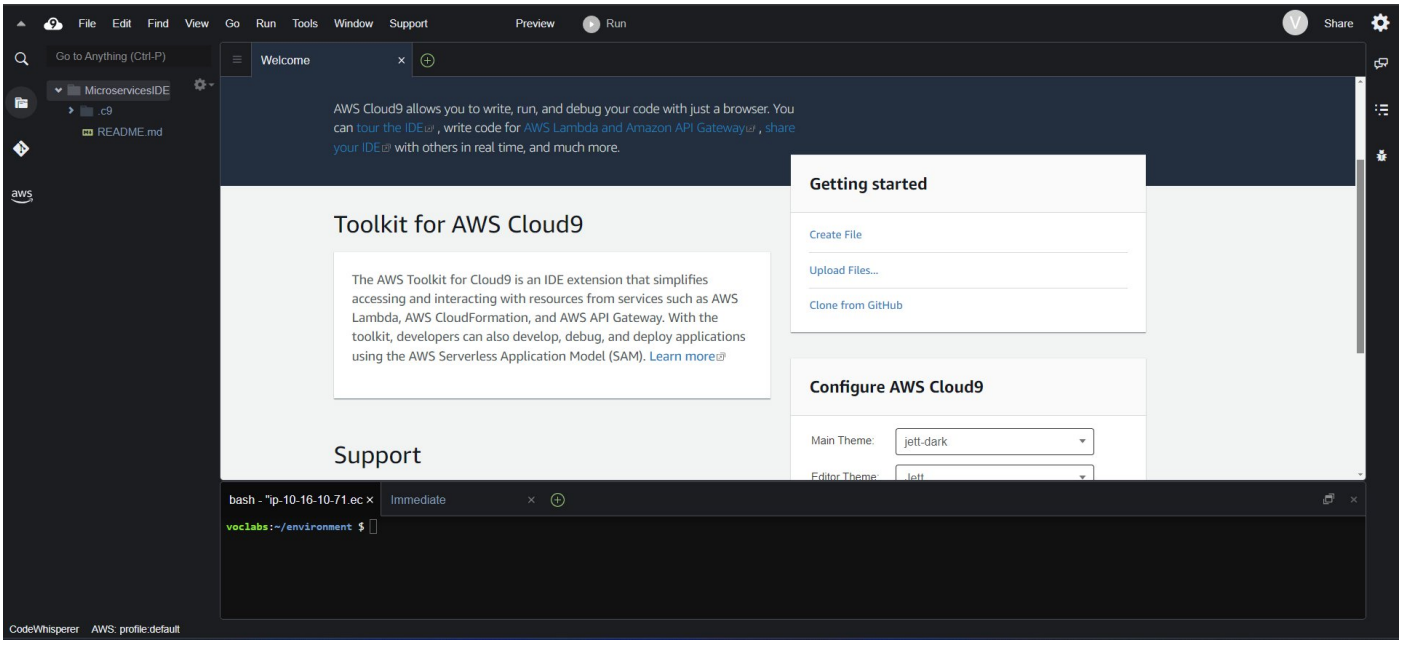

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
AWS Cloud9 > Environments > Create environment
Create environment Info
Details
Name
MicroservicesIDE
Limit of 60 characters, alphanumeric, and unique per user.
Description - optional
Limit 200 characters.
Environment type Info
Determines what the Cloud9 IDE will run on.
New EC2 instance
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.
Existing compute
You have an existing instance or server that you'd like to use.
New EC2 instance
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

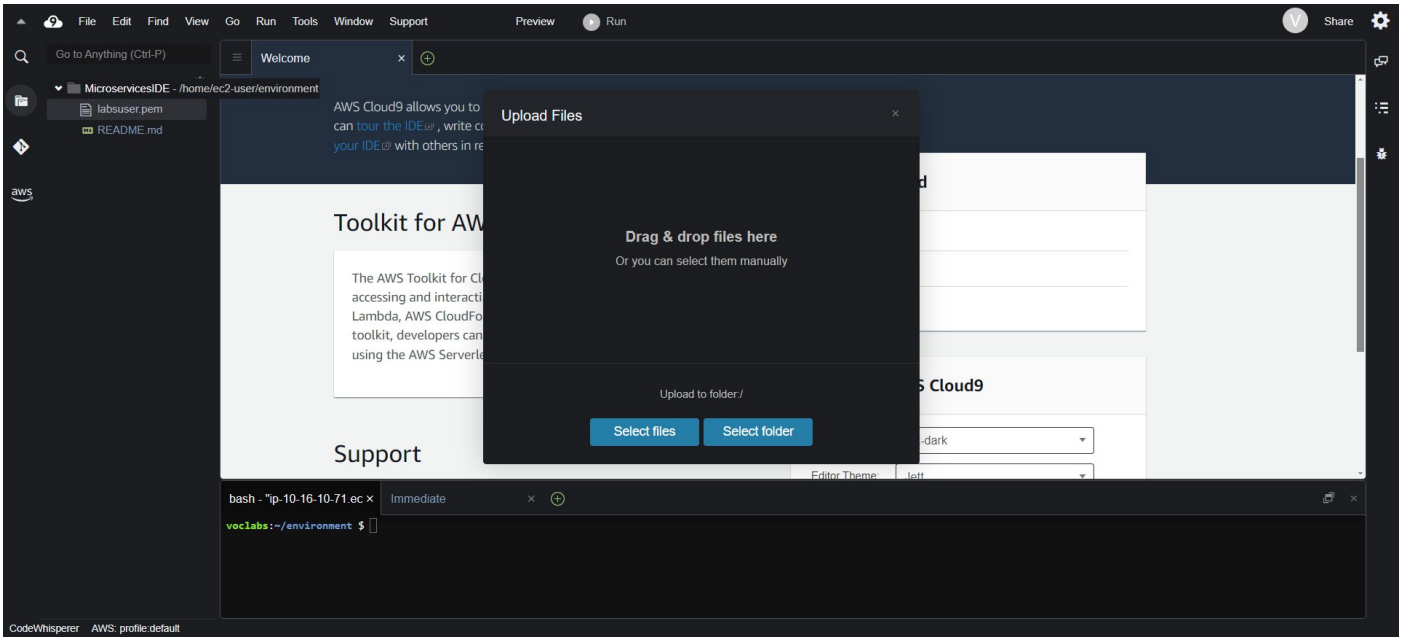

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
New EC2 instance
Instance type Info
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.
t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.
t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.
m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.
Additional instance types
Explore additional instances to fit your need.
Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.
Amazon Linux 2
Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

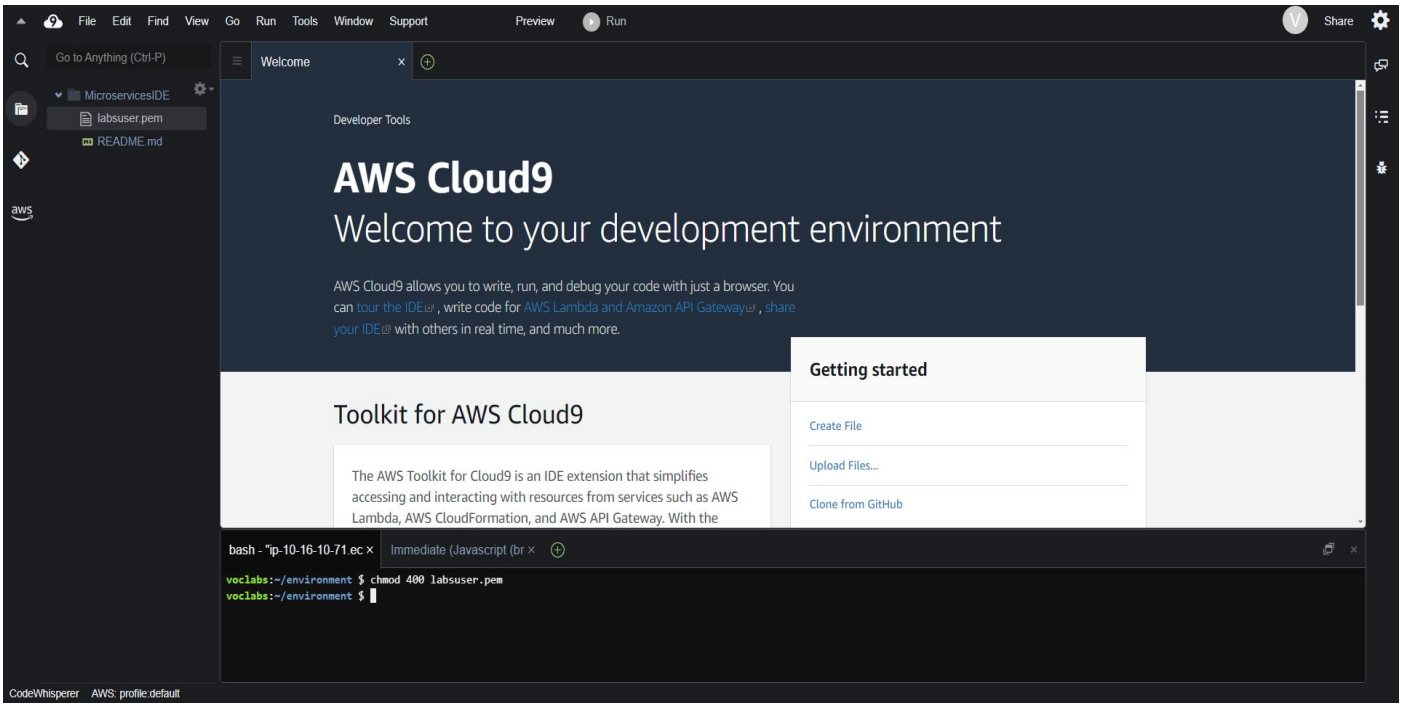


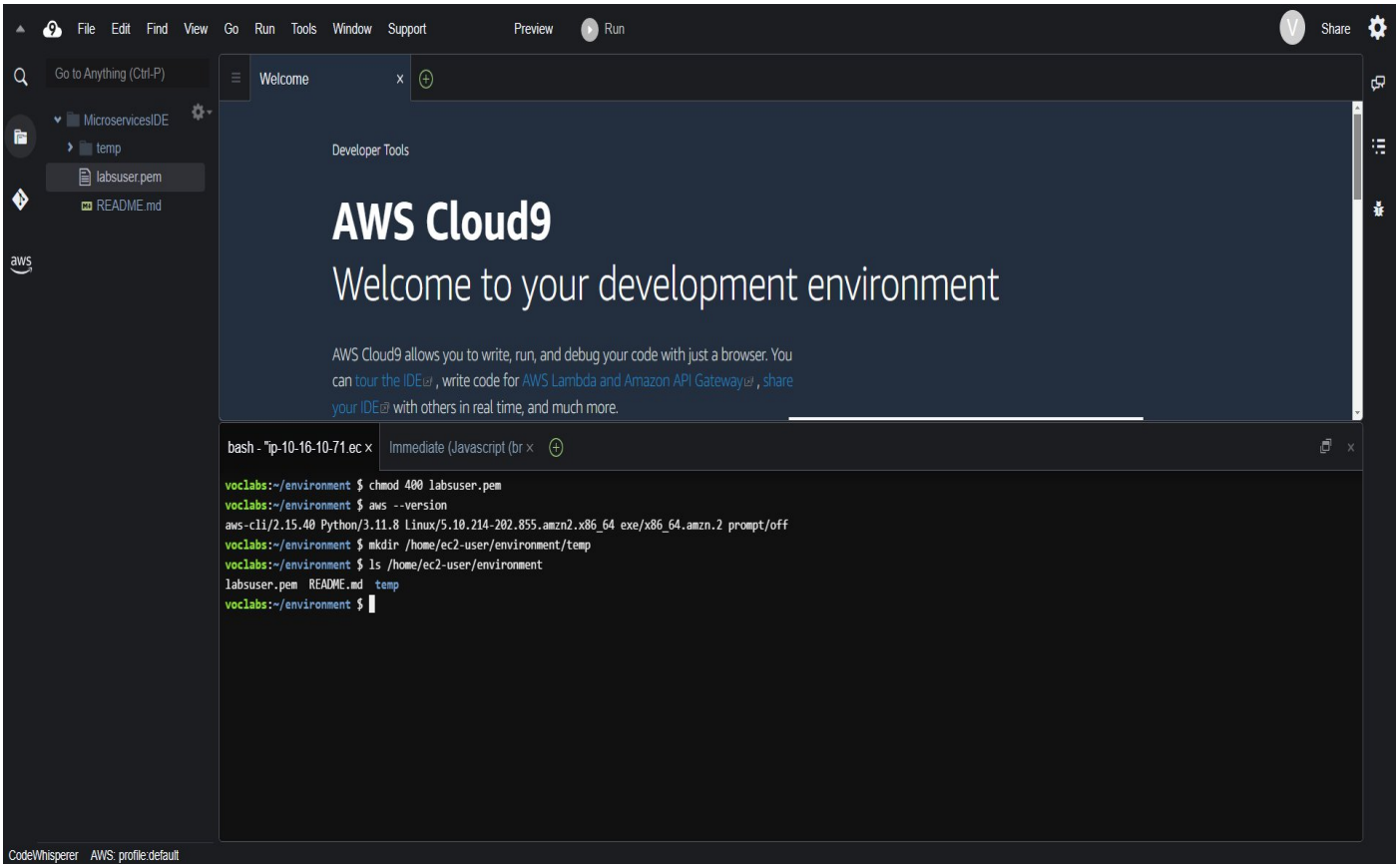
- Task 3.2: Copying the application code to my IDE. I downloaded the labsuser.pem file from the AWS Details panel to my local computer and then uploaded it to the AWS Cloud9 IDE. Using the Linux **chmod** command I have set the correct permissions on the .pem file for secure connections. I created a temporary directory at **/home/ec2-user/environment/temp** on the AWS Cloud9 instance.



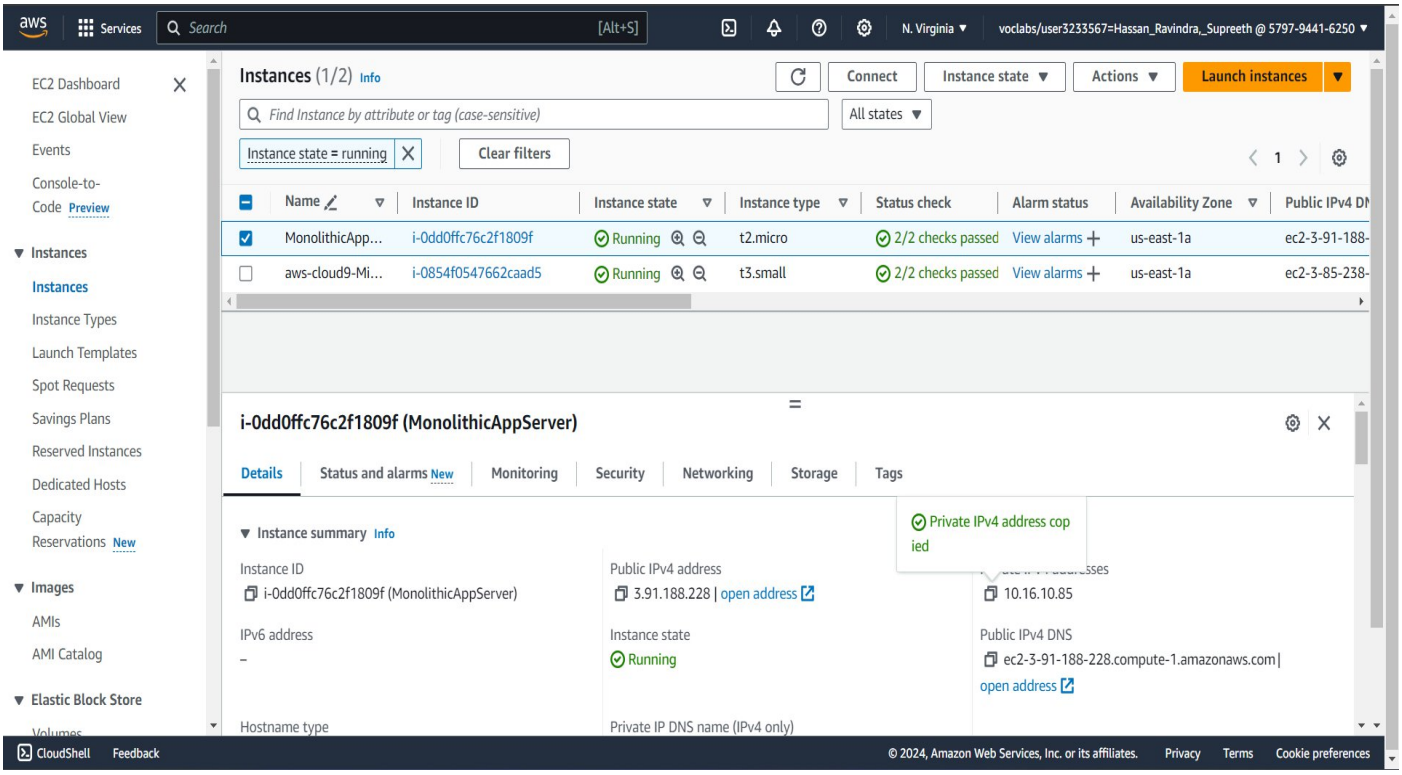

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE
.c9
README.md
Welcome
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.
Toolkit for AWS Cloud9
The AWS Toolkit for Cloud9 is an IDE extension that simplifies accessing and interacting with resources from services such as AWS Lambda, AWS CloudFormation, and AWS API Gateway. With the toolkit, developers can also develop, debug, and deploy applications using the AWS Serverless Application Model (SAM). Learn more
Support
Getting started
Create File
Upload Files...
Clone from GitHub
Configure AWS Cloud9
Main Theme:
jett-dark
bash - "ip-10-16-10-71.ec
Immediate
voclabs:~/environment $
CodeWhisperer AWS: profile:default

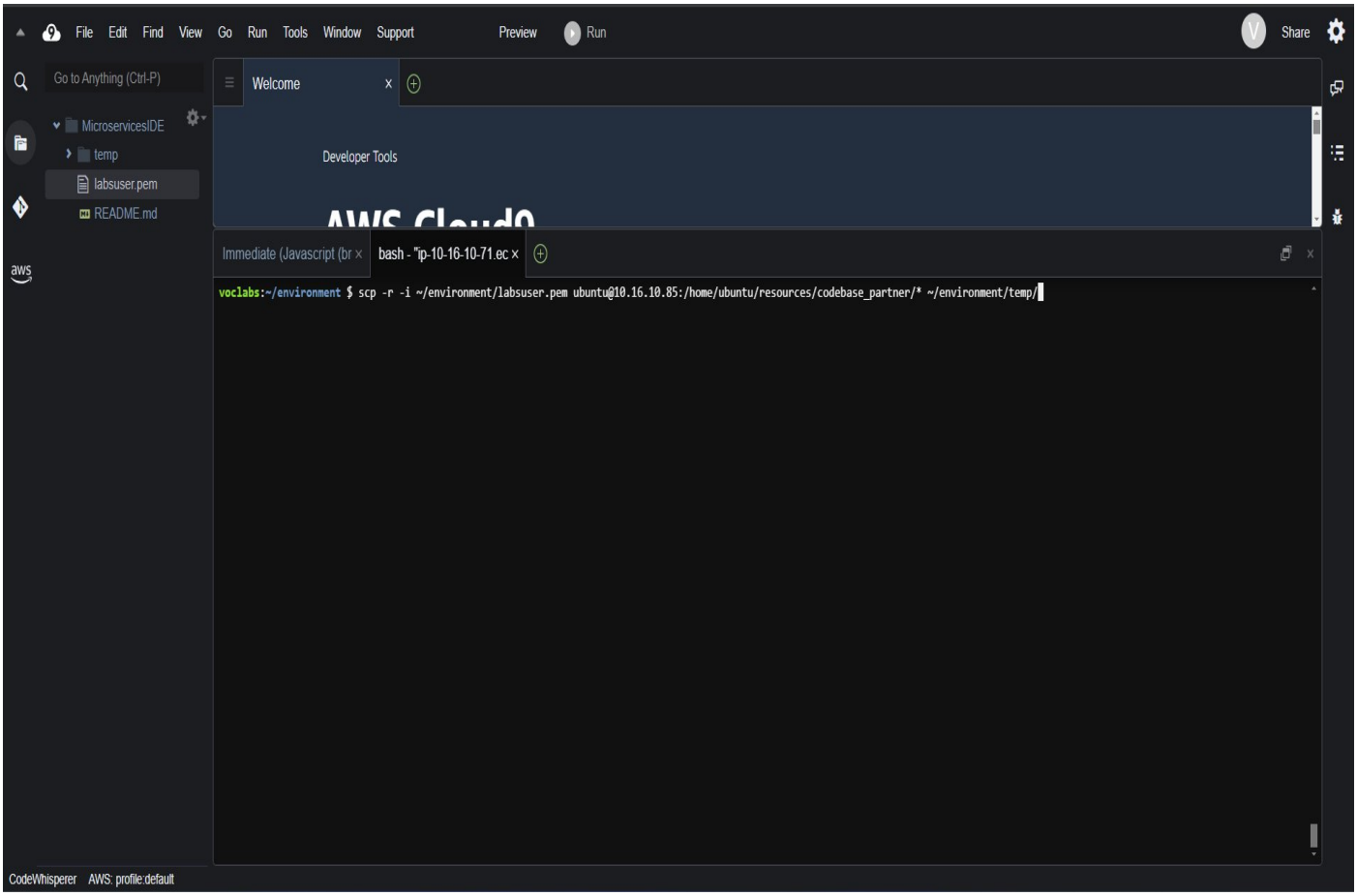

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /home/ec2-user/environment
labsuser.pem
README.md
Welcome
Upload Files
Drag & drop files here
Or you can select them manually
Upload to folder:/
Select files
Select folder
Support
bash - "ip-10-16-10-71.ec
Immediate
voclabs:~/environment $
CodeWhisperer AWS: profile:default

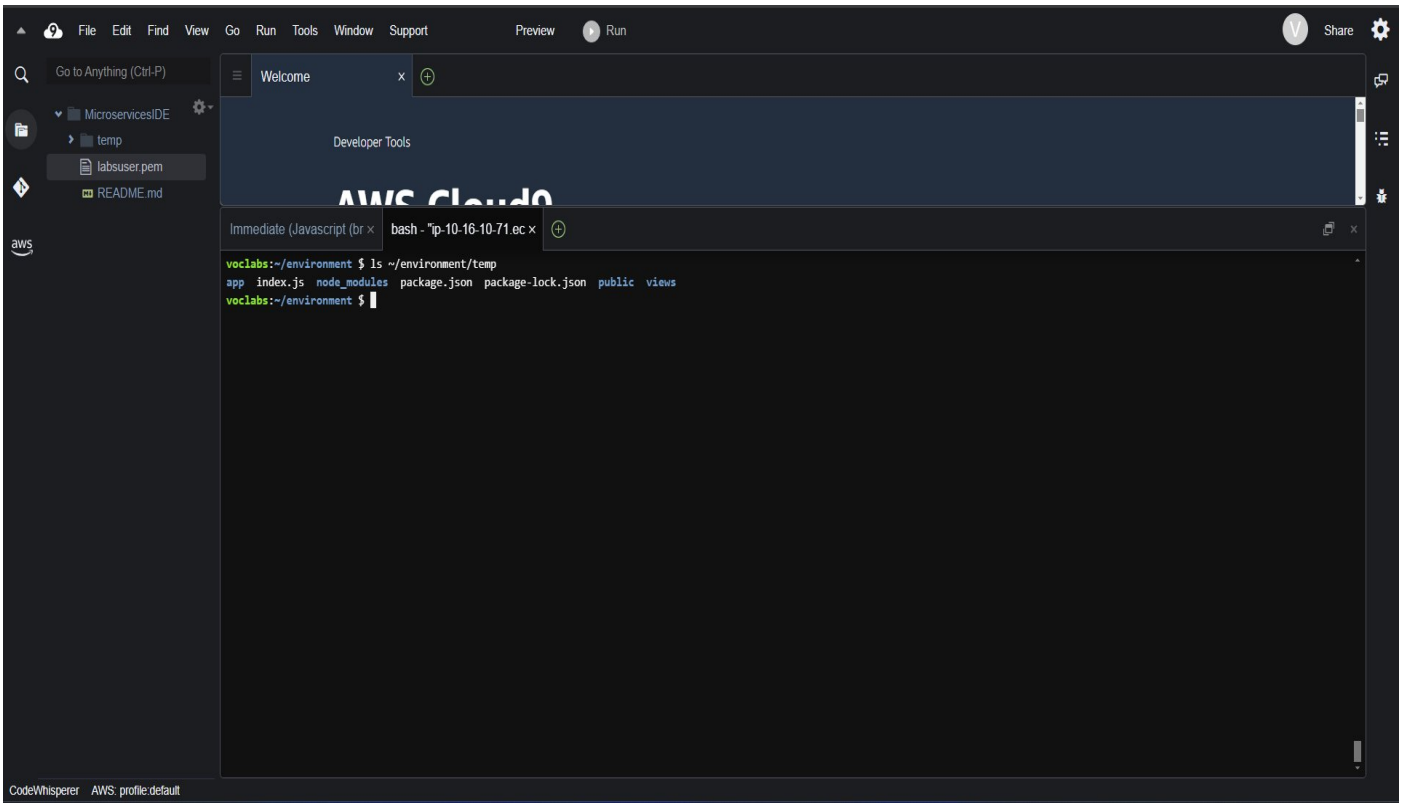

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE
labsuser.pem
README.md
Welcome
Developer Tools
AWS Cloud9
Welcome to your development environment
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.
Toolkit for AWS Cloud9
The AWS Toolkit for Cloud9 is an IDE extension that simplifies accessing and interacting with resources from services such as AWS Lambda, AWS CloudFormation, and AWS API Gateway. With the
Getting started
Create File
Upload Files...
Clone from GitHub
bash - "ip-10-16-10-71.ec
Immediate (Javascript (br
voclabs:~/environment $ chmod 400 labsuser.pem
voclabs:~/environment $
CodeWhisperer AWS: profile:default



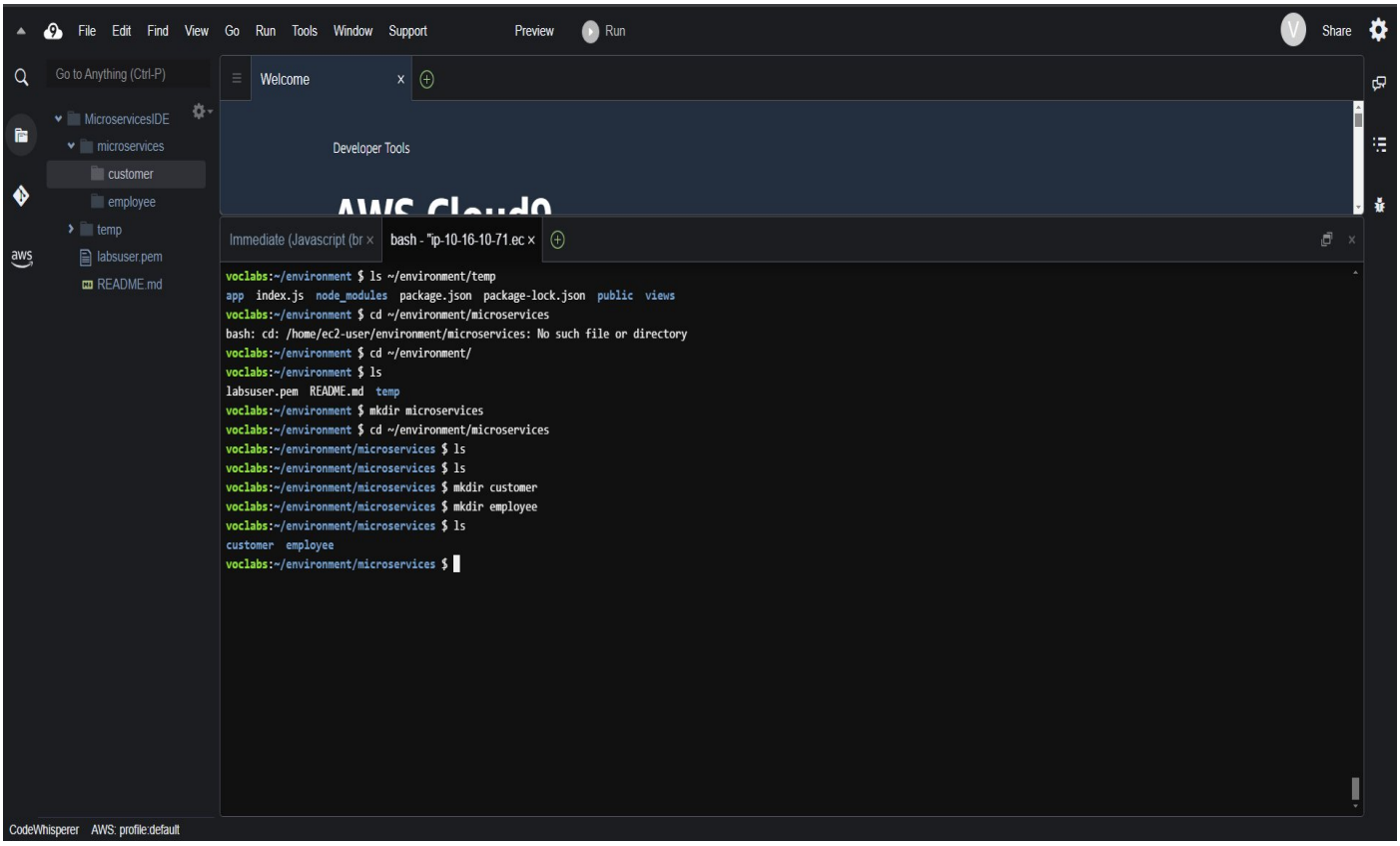
- I retrieved the private IPv4 address of the MonolithicAppServer instance from the Amazon EC2 console and used the Linux **scp** command to copy the source code from the MonolithicAppServer instance to the temp directory on the AWS Cloud9 instance using the proper SSH key. Finally, I verified through the IDE's file browser that the source files were successfully copied to the temp directory.

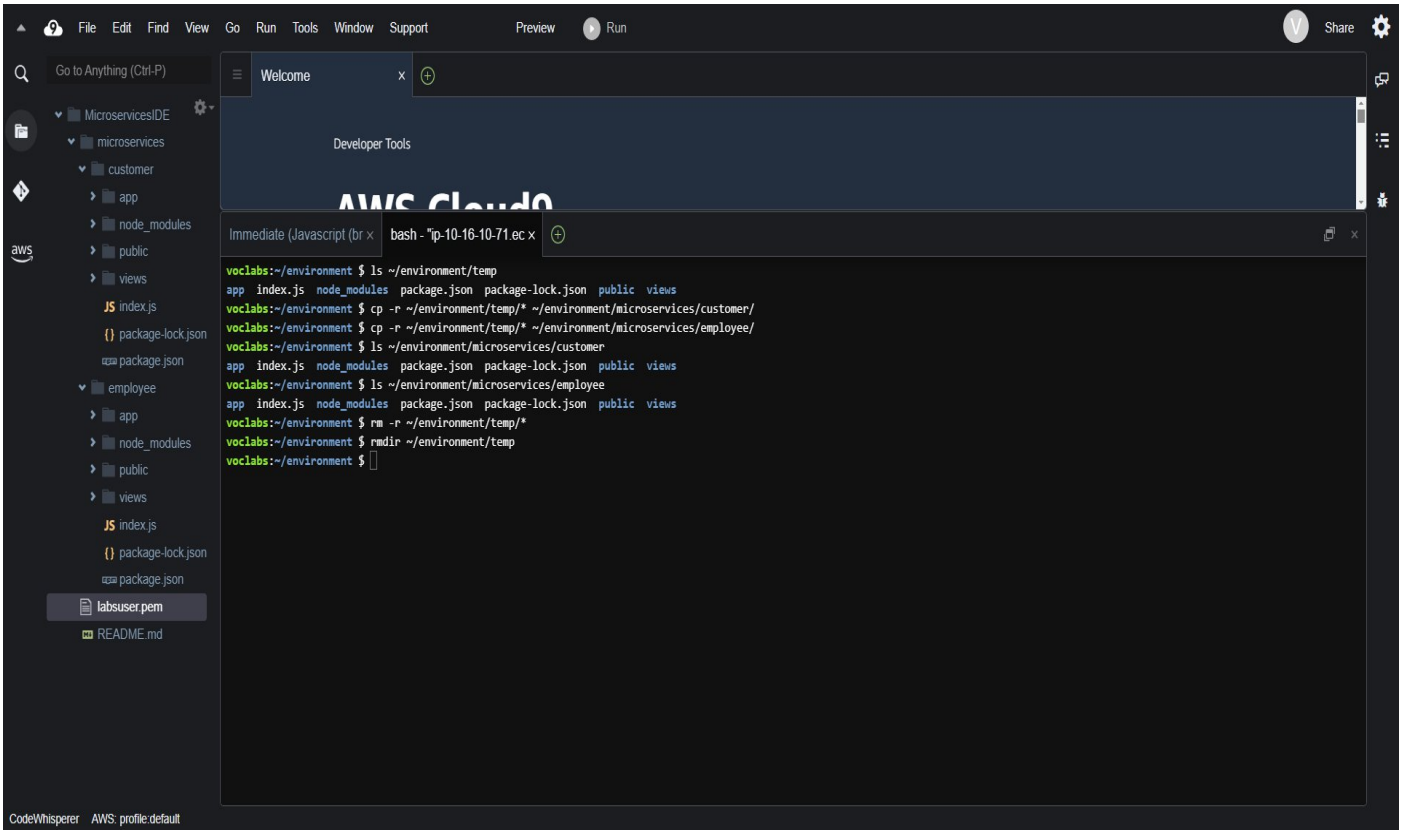




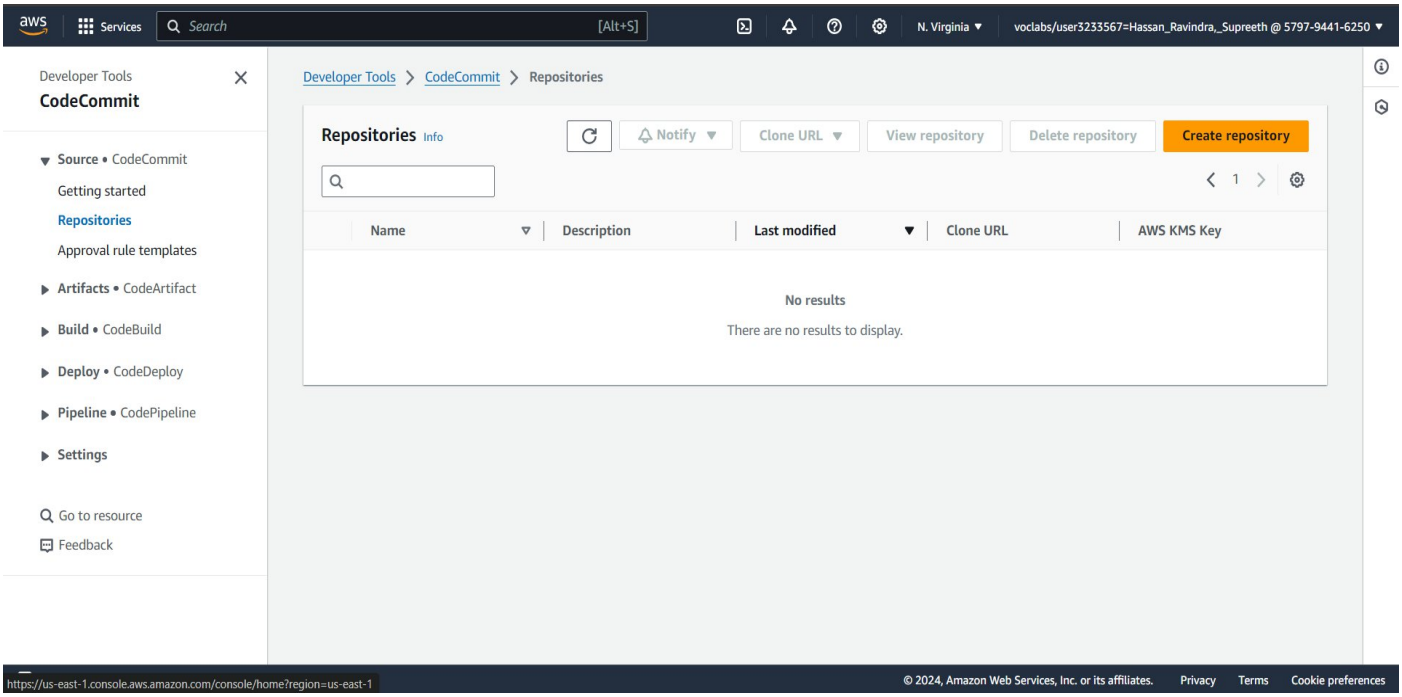


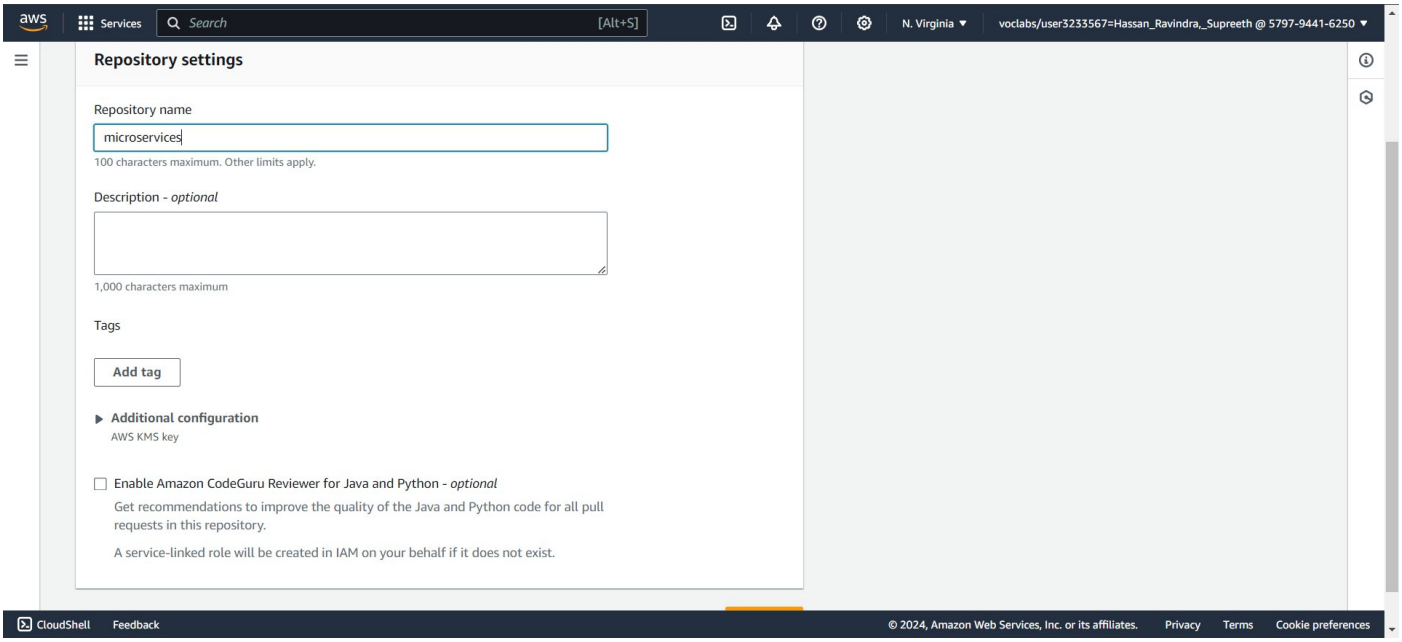
- Task 3.3: Creating two directories customer and employee. Copying the code of the monolithic application in each new directory and finally removing the temp directory from the tree structure.

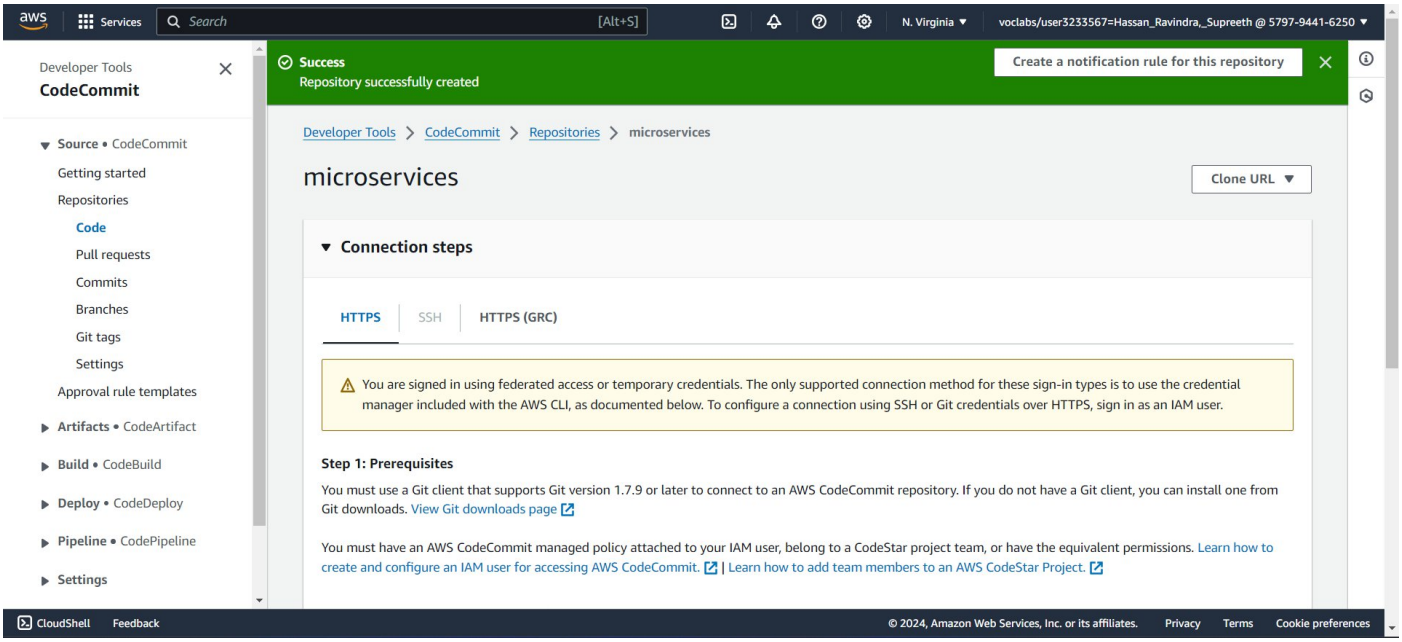


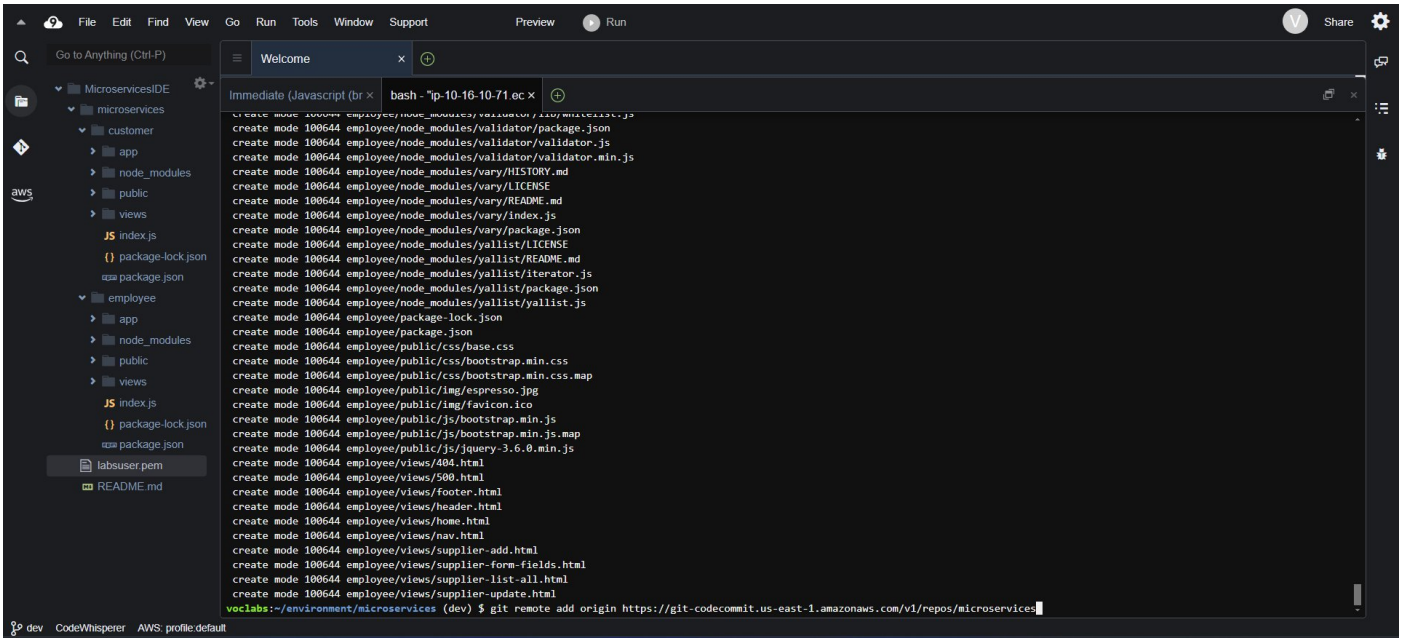


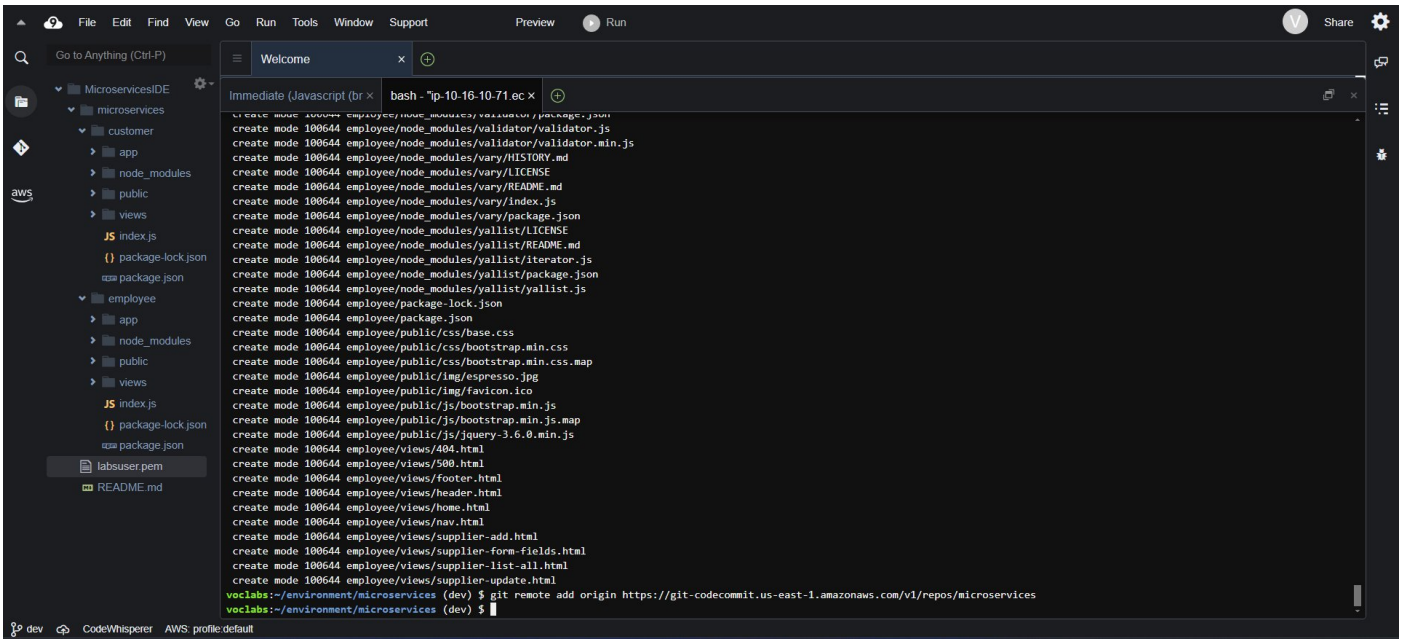
- Task 3.4: In this task, I am creating CodeCommit repository named microservices as my Git repository to be able to evolve the application fucntionality and to time the deployment of feature enhancements for these microservices separately.

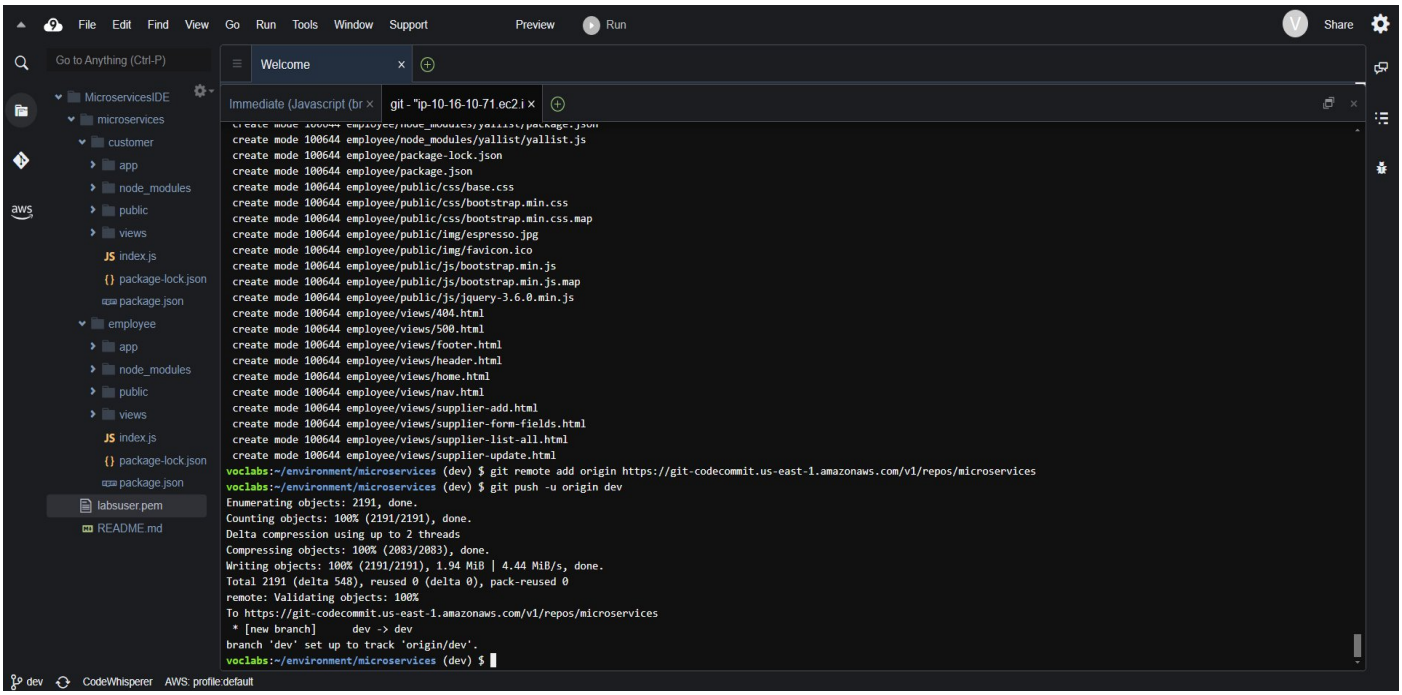

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go 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
No results
There are no results to display.
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

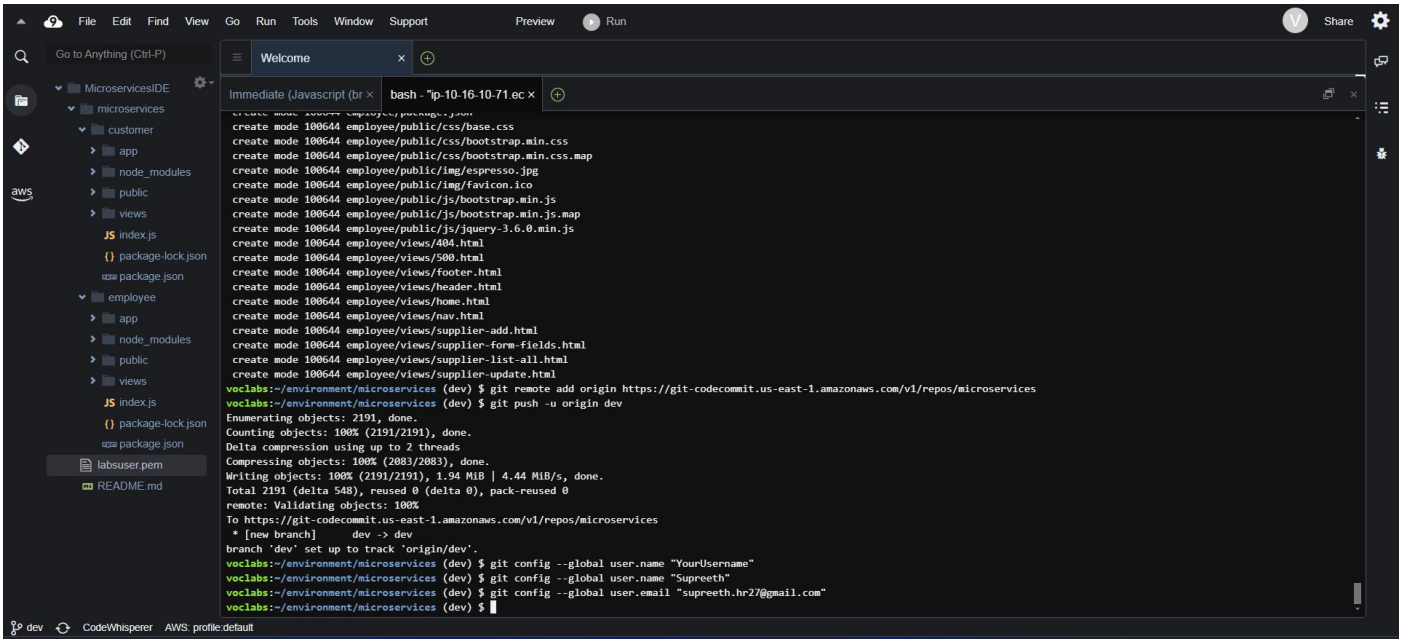

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Repository settings
Repository name
microservices
100 characters maximum. Other limits apply.
Description - optional
1,000 characters maximum
Tags
Add tag
Additional configuration
AWS KMS key
Enable Amazon CodeGuru Reviewer for Java and Python - optional
Get recommendations to improve the quality of the Java and Python code for all pull requests in this repository.
A service-linked role will be created in IAM on your behalf if it does not exist.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

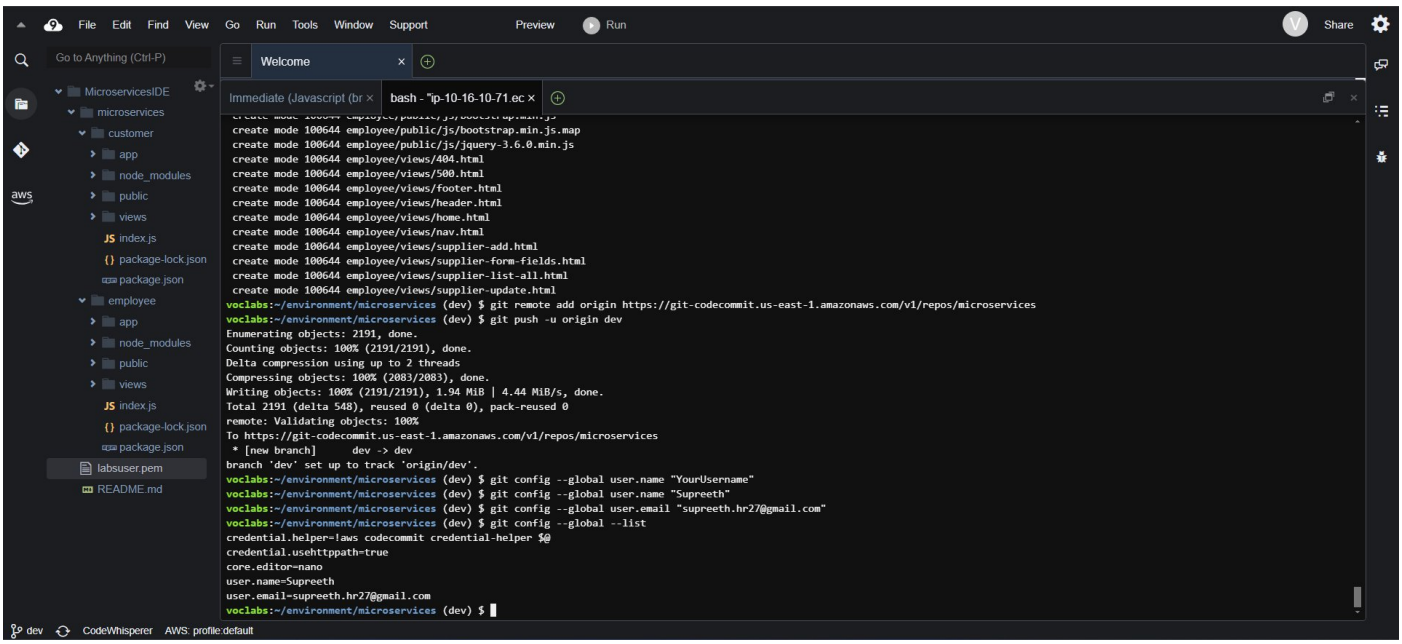

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Success
Repository successfully created
Create a notification rule for this repository
Developer Tools > CodeCommit > Repositories > microservices
microservices
Clone URL
Connection steps
HTTPS
SSH
HTTPS (GRC)
You are signed in using federated access or temporary credentials. The only supported connection method for these sign-in types is to use the credential manager included with the AWS CLI, as documented below. To configure a connection using SSH or Git credentials over HTTPS, sign in as an IAM user.
Step 1: Prerequisites
You must use a Git client that supports Git version 1.7.9 or later to connect to an AWS CodeCommit repository. If you do not have a Git client, you can install one from Git downloads. View Git downloads page
You must have an AWS CodeCommit managed policy attached to your IAM user, belong to a CodeStar project team, or have the equivalent permissions. Learn how to create and configure an IAM user for accessing AWS CodeCommit. | Learn how to add team members to an AWS CodeStar Project.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

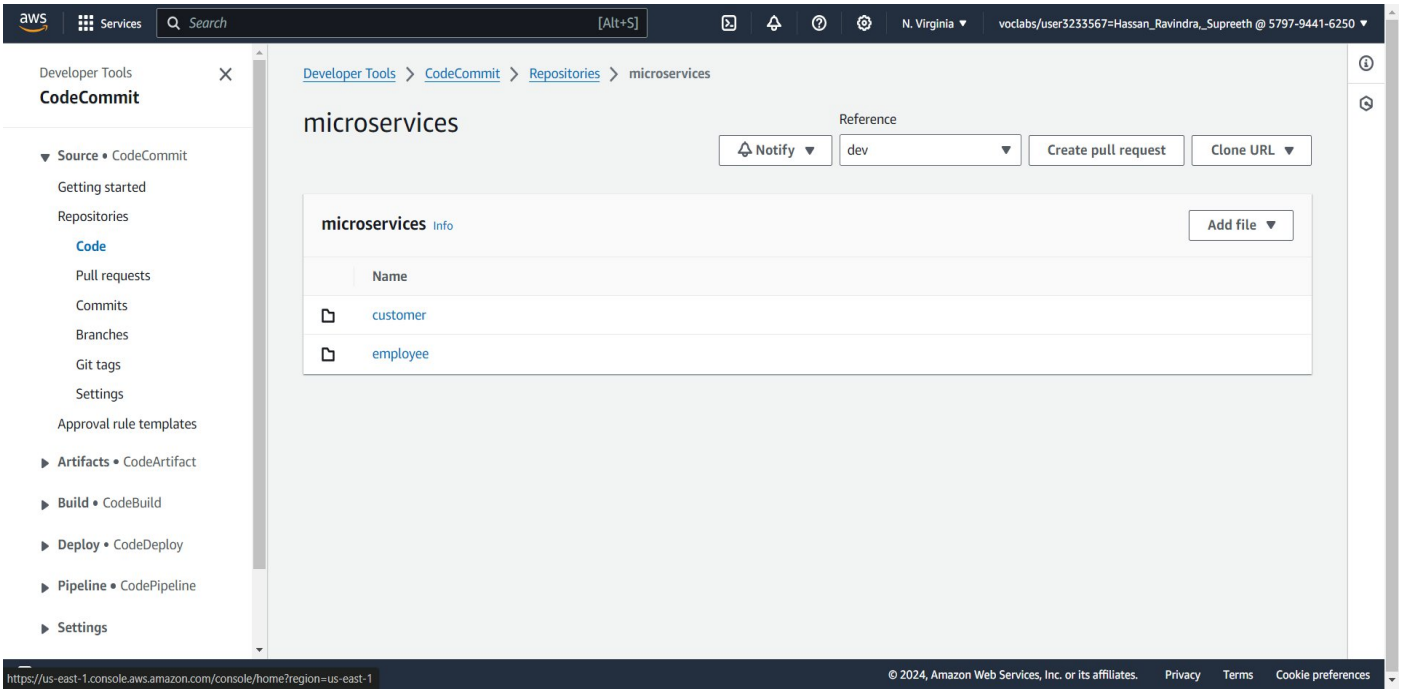

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
MicroservicesIDE
microservices
customer
app
node_modules
public
views
index.js
package-lock.json
package.json
employee
app
node_modules
public
views
index.js
package-lock.json
package.json
labsuser.pem
README.md
create mode 100644 employee/node_modules/validator/package.json
create mode 100644 employee/node_modules/validator/validator.js
create mode 100644 employee/node_modules/validator/validator.min.js
create mode 100644 employee/node_modules/vary/HISTORY.md
create mode 100644 employee/node_modules/vary/LICENSE
create mode 100644 employee/node_modules/vary/README.md
create mode 100644 employee/node_modules/vary/index.js
create mode 100644 employee/node_modules/vary/package.json
create mode 100644 employee/node_modules/yallist/LICENSE
create mode 100644 employee/node_modules/yallist/README.md
create mode 100644 employee/node_modules/yallist/iterator.js
create mode 100644 employee/node_modules/yallist/package.json
create mode 100644 employee/node_modules/yallist/yallist.js
create mode 100644 employee/package-lock.json
create mode 100644 employee/package.json
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
voclabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
dev CodeWhisperer AWS: profile:default

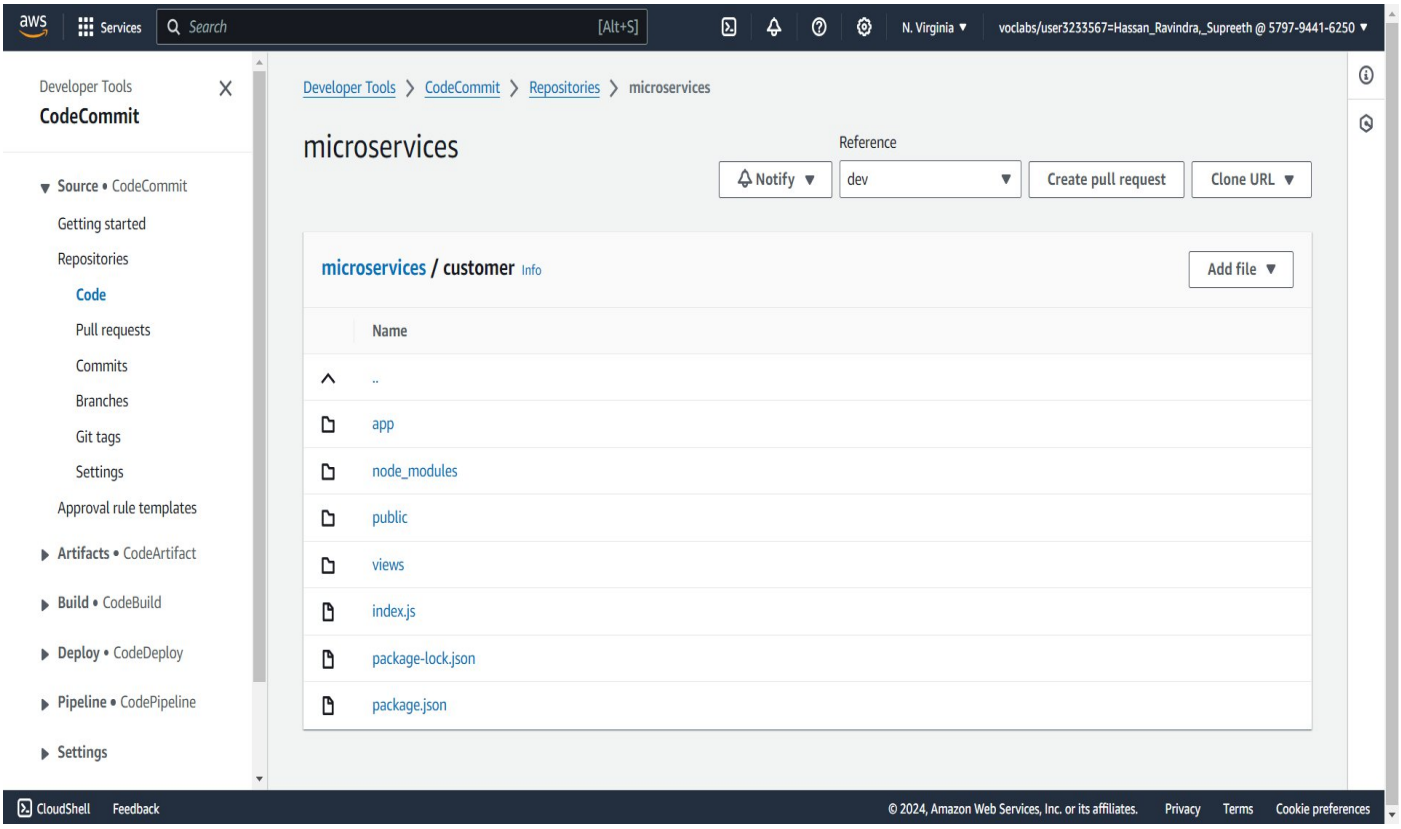

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
MicroservicesIDE
microservices
customer
app
node_modules
public
views
index.js
package-lock.json
package.json
employee
app
node_modules
public
views
index.js
package-lock.json
package.json
labsuser.pem
README.md
create mode 100644 employee/node_modules/validator/validator.js
create mode 100644 employee/node_modules/validator/validator.min.js
create mode 100644 employee/node_modules/vary/HISTORY.md
create mode 100644 employee/node_modules/vary/LICENSE
create mode 100644 employee/node_modules/vary/README.md
create mode 100644 employee/node_modules/vary/index.js
create mode 100644 employee/node_modules/vary/package.json
create mode 100644 employee/node_modules/yallist/LICENSE
create mode 100644 employee/node_modules/yallist/README.md
create mode 100644 employee/node_modules/yallist/iterator.js
create mode 100644 employee/node_modules/yallist/package.json
create mode 100644 employee/node_modules/yallist/yallist.js
create mode 100644 employee/package-lock.json
create mode 100644 employee/package.json
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
voclabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
voclabs:~/environment/microservices (dev) $
dev CodeWhisperer AWS: profile:default

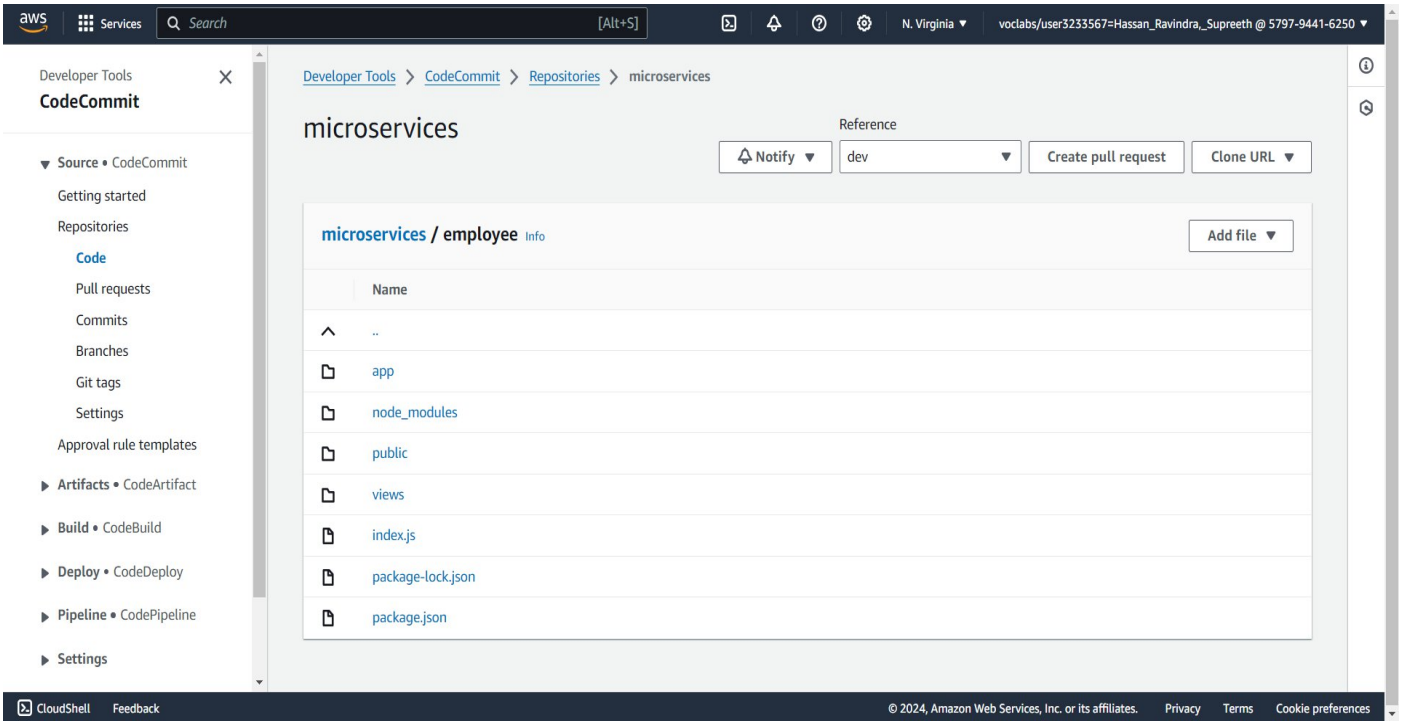

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Immediate (Javascript (br
git - "ip-10-16-10-71.ec2.i
MicroservicesIDE
microservices
customer
app
node_modules
public
views
index.js
package-lock.json
package.json
employee
app
node_modules
public
views
index.js
package-lock.json
package.json
labsuser.pem
README.md
create mode 100644 employee/node_modules/yallist/yallist.js
create mode 100644 employee/package-lock.json
create mode 100644 employee/package.json
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
voclabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
voclabs:~/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.44 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
* [new branch] dev -> dev
branch 'dev' set up to track 'origin/dev'.
voclabs:~/environment/microservices (dev) $
dev CodeWhisperer AWS: profile:default


File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
MicroservicesIDE
microservices
customer
app
node_modules
public
views
index.js
package-lock.json
package.json
employee
app
node_modules
public
views
index.js
package-lock.json
package.json
labsuser.pem
README.md
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
voclabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
voclabs:~/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.44 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
* [new branch] dev -> dev
branch 'dev' set up to track 'origin/dev'.
voclabs:~/environment/microservices (dev) $ git config --global user.name "YourUsername"
voclabs:~/environment/microservices (dev) $ git config --global user.name "Supreeth"
voclabs:~/environment/microservices (dev) $ git config --global user.email "supreeth.hr27@gmail.com"
voclabs:~/environment/microservices (dev) $
dev CodeWhisperer AWS: profile:default


File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
MicroservicesIDE
microservices
customer
app
node_modules
public
views
index.js
package-lock.json
package.json
employee
app
node_modules
public
views
index.js
package-lock.json
package.json
labsuser.pem
README.md
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
voclabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
voclabs:~/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.44 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
* [new branch] dev -> dev
branch 'dev' set up to track 'origin/dev'.
voclabs:~/environment/microservices (dev) $ git config --global user.name "YourUsername"
voclabs:~/environment/microservices (dev) $ git config --global user.name "Supreeth"
voclabs:~/environment/microservices (dev) $ git config --global user.email "supreeth.hr27@gmail.com"
voclabs:~/environment/microservices (dev) $ git config --global --list
credential.helper=!aws codecommit credential-helper $@
credential.usehttppath=true
core.editor=nano
user.name=Supreeth
user.email=supreeth.hr27@gmail.com
voclabs:~/environment/microservices (dev) $
dev CodeWhisperer AWS: profile:default

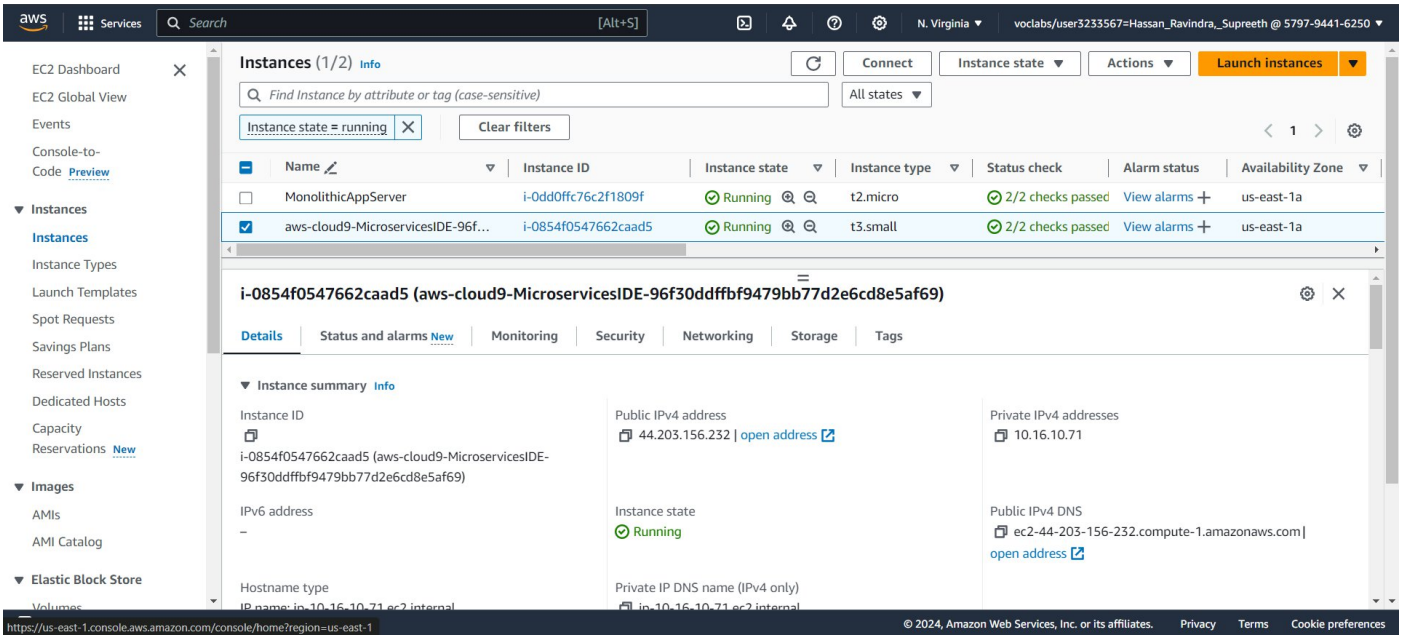

aws Services Search [Alt+S] N. Virginia voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • 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
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

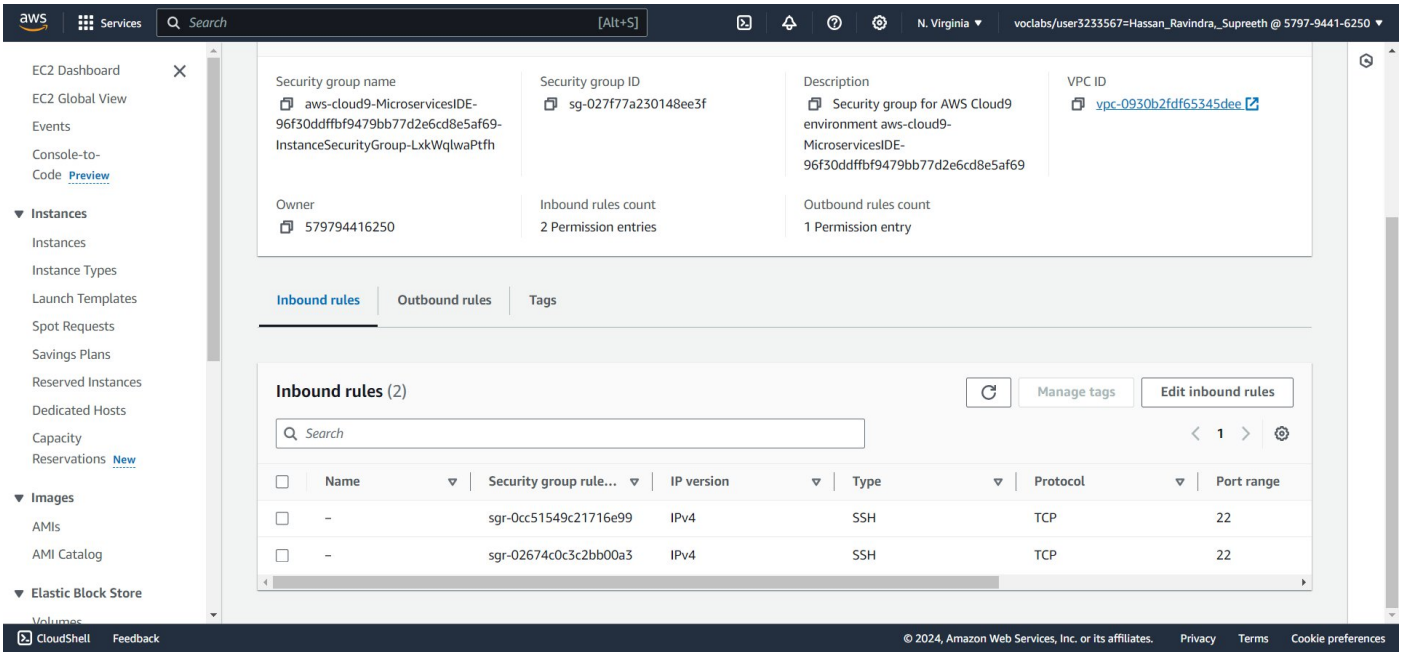


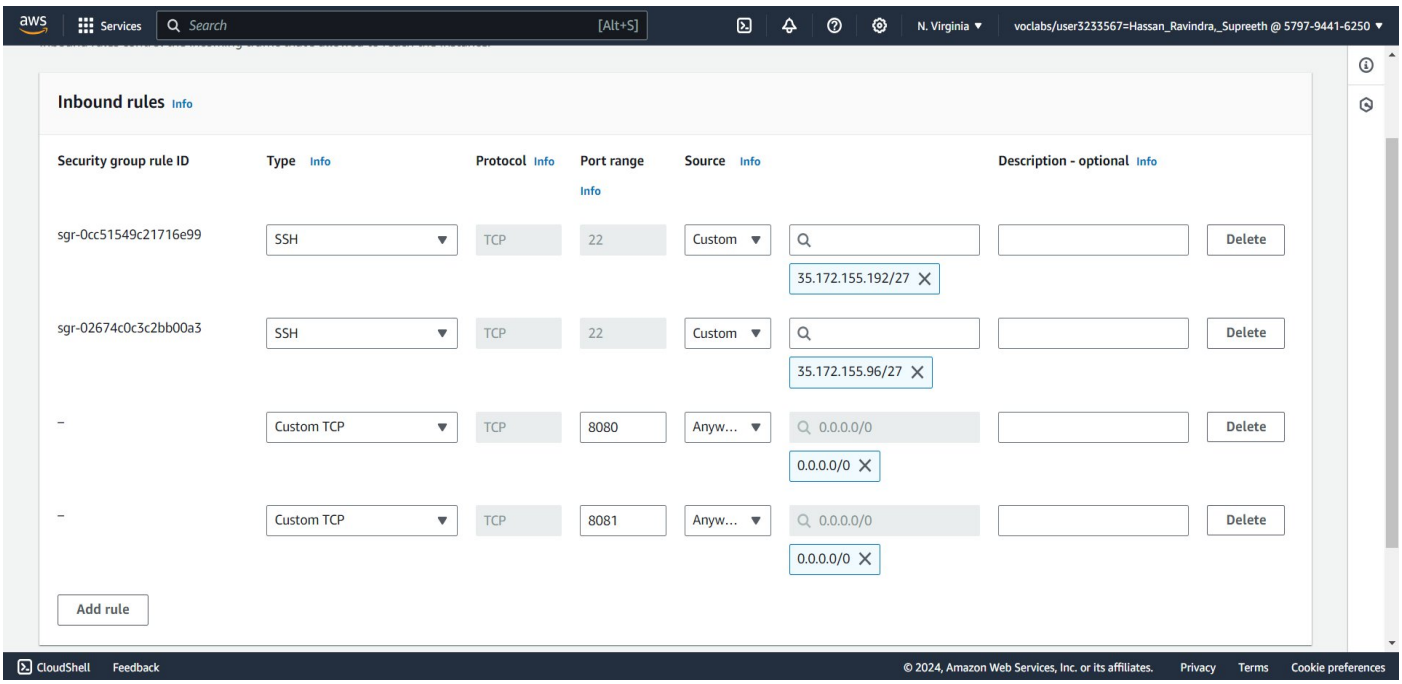


PHASE-4: Configuring the application as two microservices and testing them in Docker containers

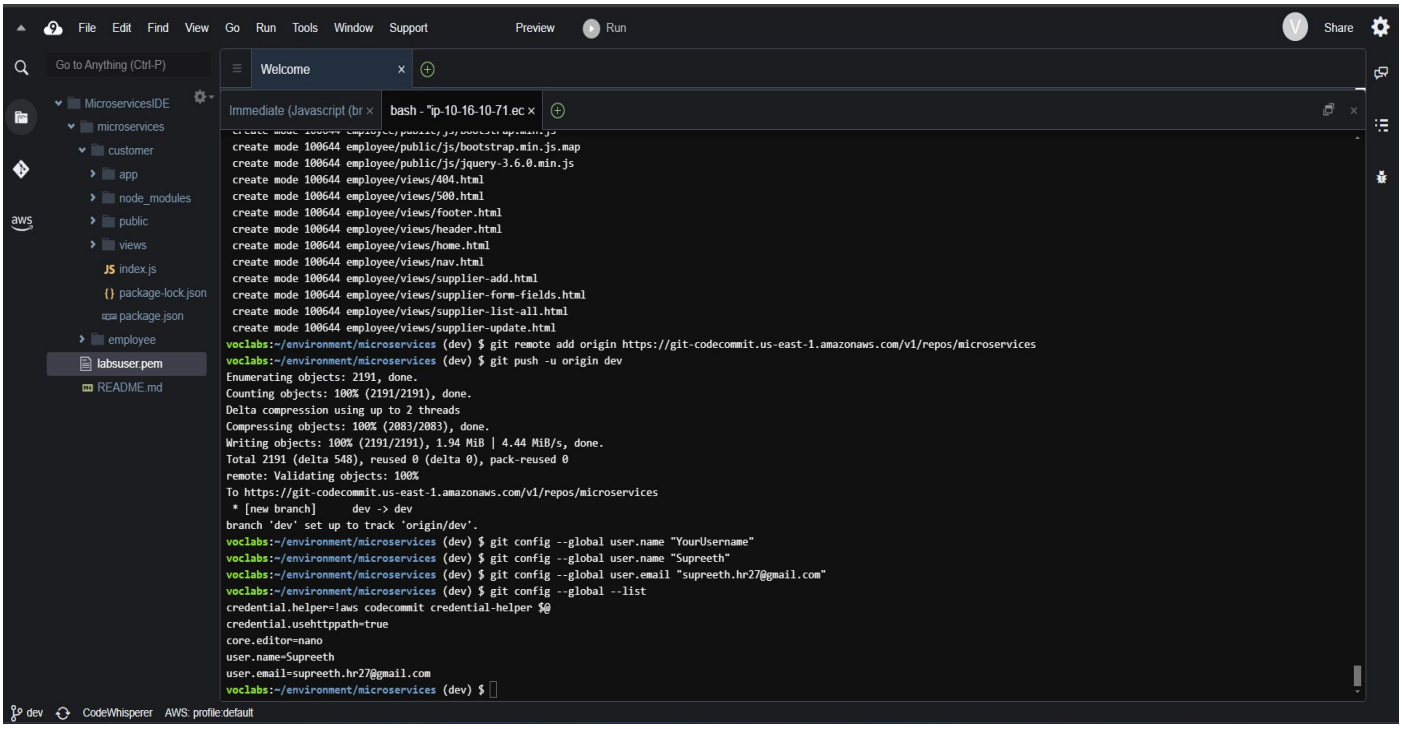
- Task 4.1: I adjusted the security group of the Cloud9 instance to allow inbound traffic on TCP ports 8080 and 8081.

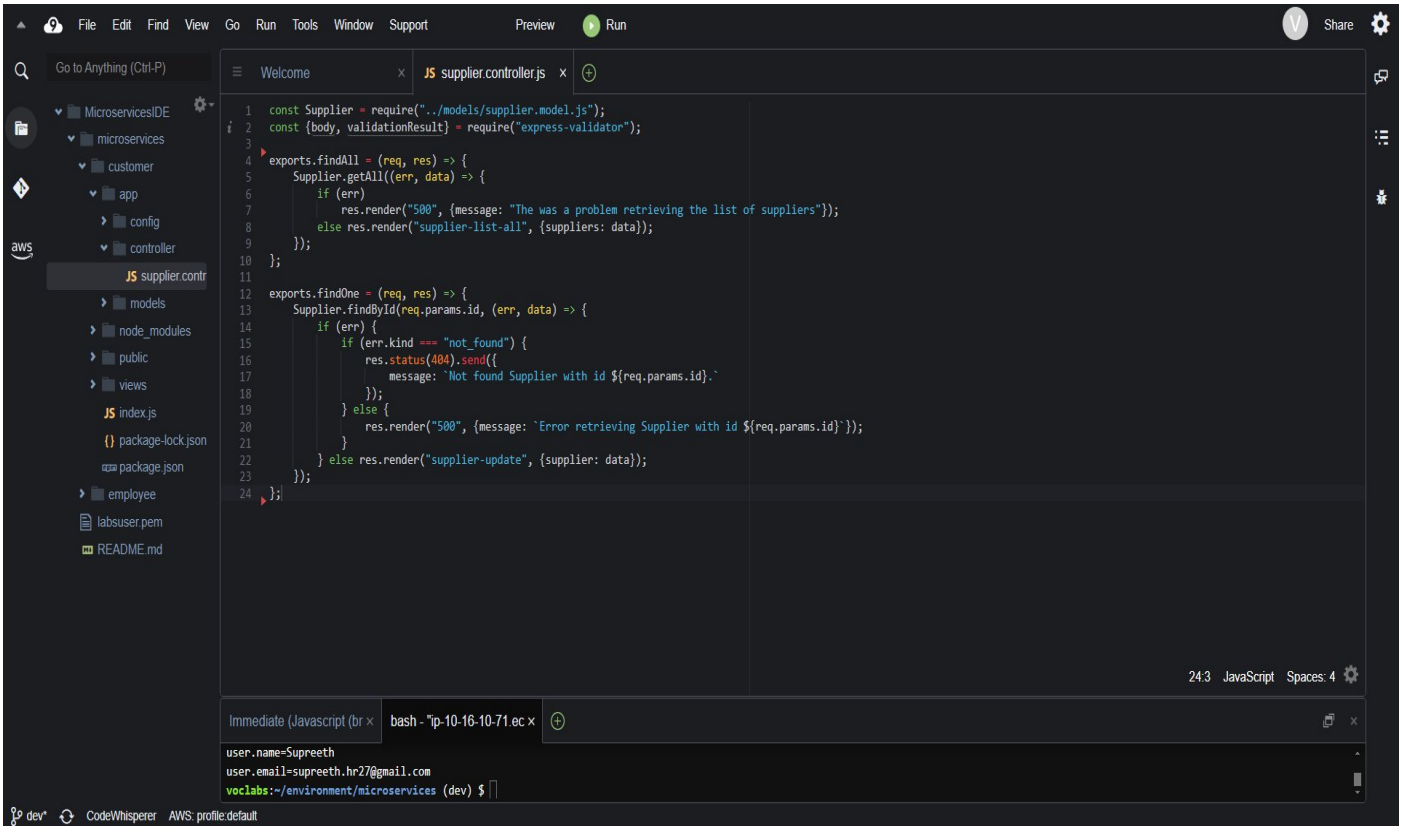

Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
MonolithicAppServer
i-0dd0ffc76c2f1809f
Running
t2.micro
2/2 checks passed
View alarms
us-east-1a
aws-cloud9-MicroservicesIDE-96f...
i-0854f0547662caad5
Running
t3.small
2/2 checks passed
View alarms
us-east-1a
i-0854f0547662caad5 (aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69)
Details
Status and alarms New
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-0854f0547662caad5 (aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69)
IPv6 address
–
Hostname type
Public IPv4 address
44.203.156.232 | open address
Instance state
Running
Private IP DNS name (IPv4 only)
Private IPv4 addresses
10.16.10.71
Public IPv4 DNS
ec2-44-203-156-232.compute-1.amazonaws.com | open address
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

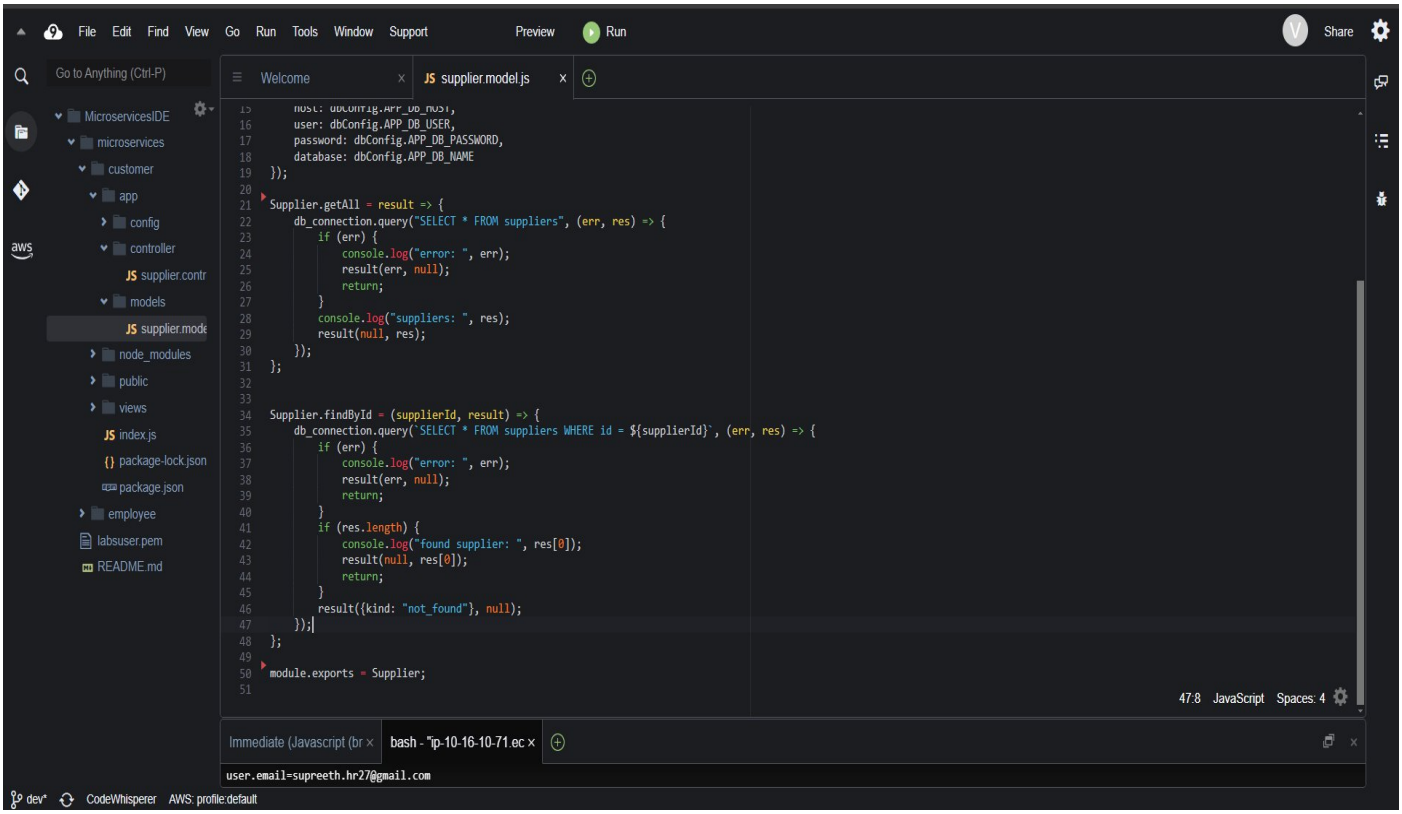

Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Security group name
aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69-InstanceSecurityGroup-LxkWqlwaPtfh
Security group ID
sg-027f77a230148ee3f
Description
Security group for AWS Cloud9 environment aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69
VPC ID
vpc-0930b2fdf65345dee
Owner
579794416250
Inbound rules count
2 Permission entries
Outbound rules count
1 Permission entry
Inbound rules
Outbound rules
Tags
Inbound rules (2)
Manage tags
Edit inbound rules
Search
Name
Security group rule...
IP version
Type
Protocol
Port range
sgr-0cc51549c21716e99
IPv4
SSH
TCP
22
sgr-02674c0c3c2bb00a3
IPv4
SSH
TCP
22
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

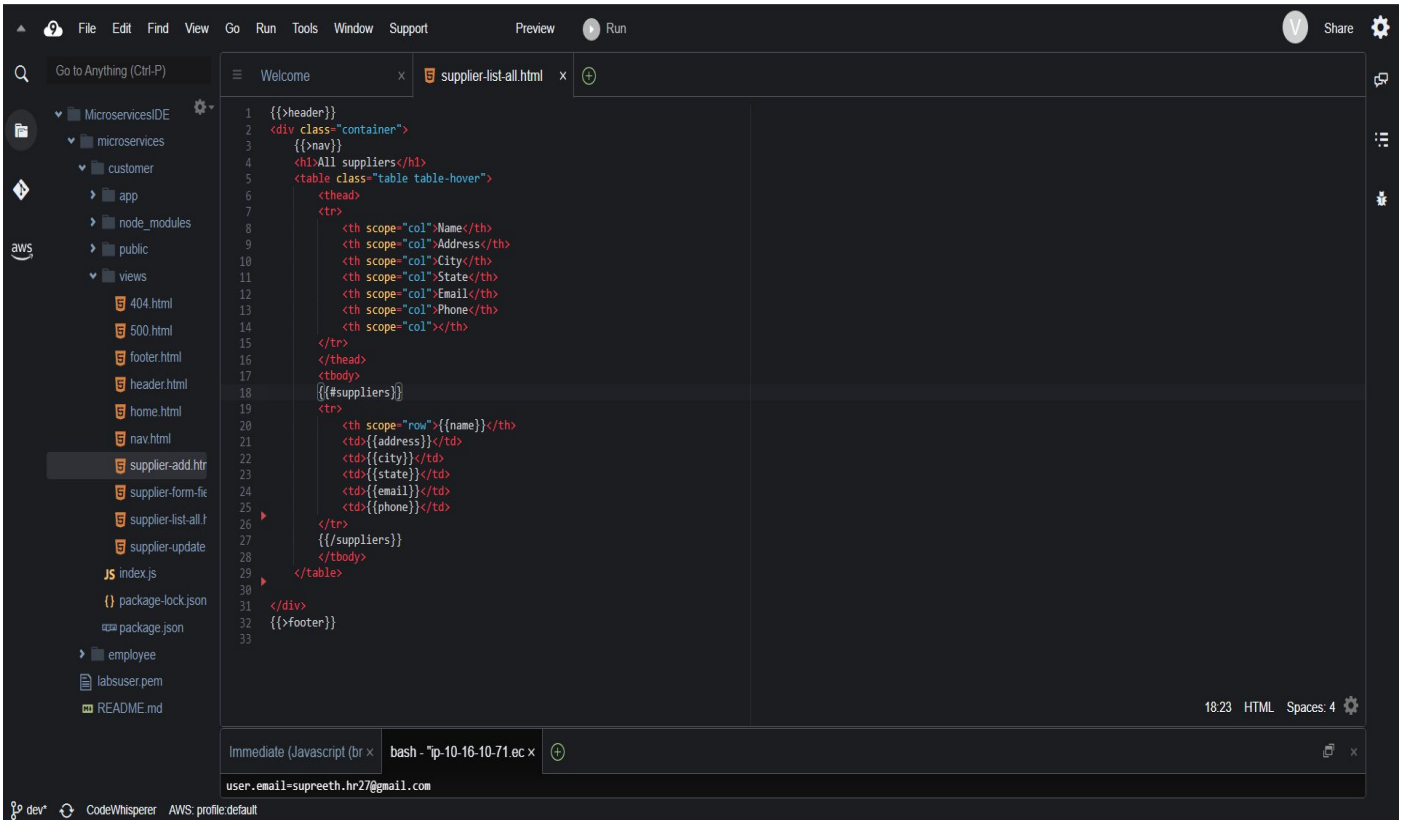

Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-0cc51549c21716e99
SSH
TCP
22
Custom
35.172.155.192/27
Delete
sgr-02674c0c3c2bb00a3
SSH
TCP
22
Custom
35.172.155.96/27
Delete
–
Custom TCP
TCP
8080
Anyw...
0.0.0.0/0
Delete
–
Custom TCP
TCP
8081
Anyw...
0.0.0.0/0
Delete
Add rule
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

- Task 4.2: To edit the source code for the customer microservice to remove features that are part of the employee microservice, I started by collapsing the "employee" directory in AWS Cloud9 and expanding the "customer" directory to access the relevant files. In the customer/app/controller/supplier.controller.js file I ensured only read-only functions remained, removing any that allowed adding, editing, or deleting suppliers. Similarly in the customer/app/models/supplier.model.js file, I deleted unnecessary functions keeping only Supplier.getAll and Supplier.findById, while retaining the last line (module.exports = Supplier).



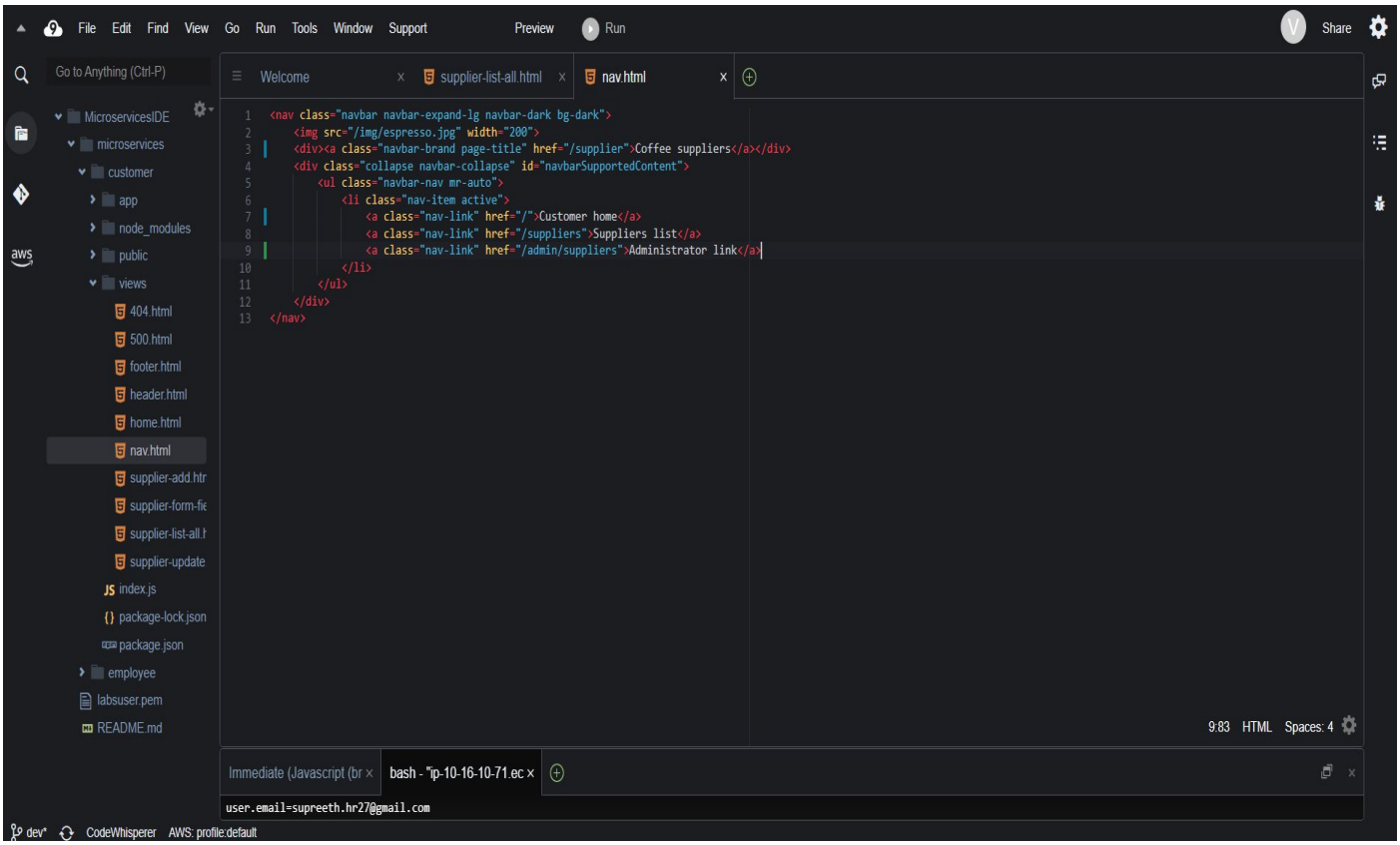


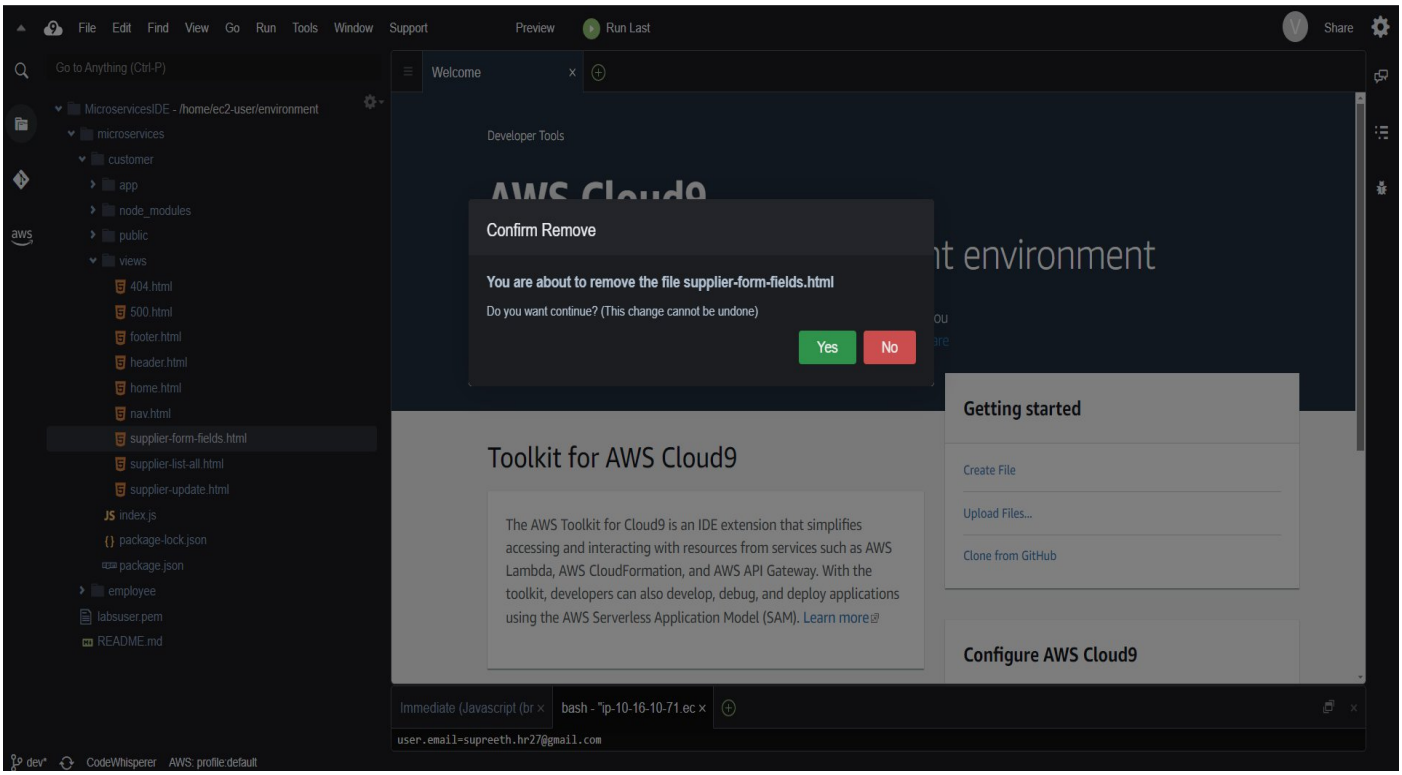


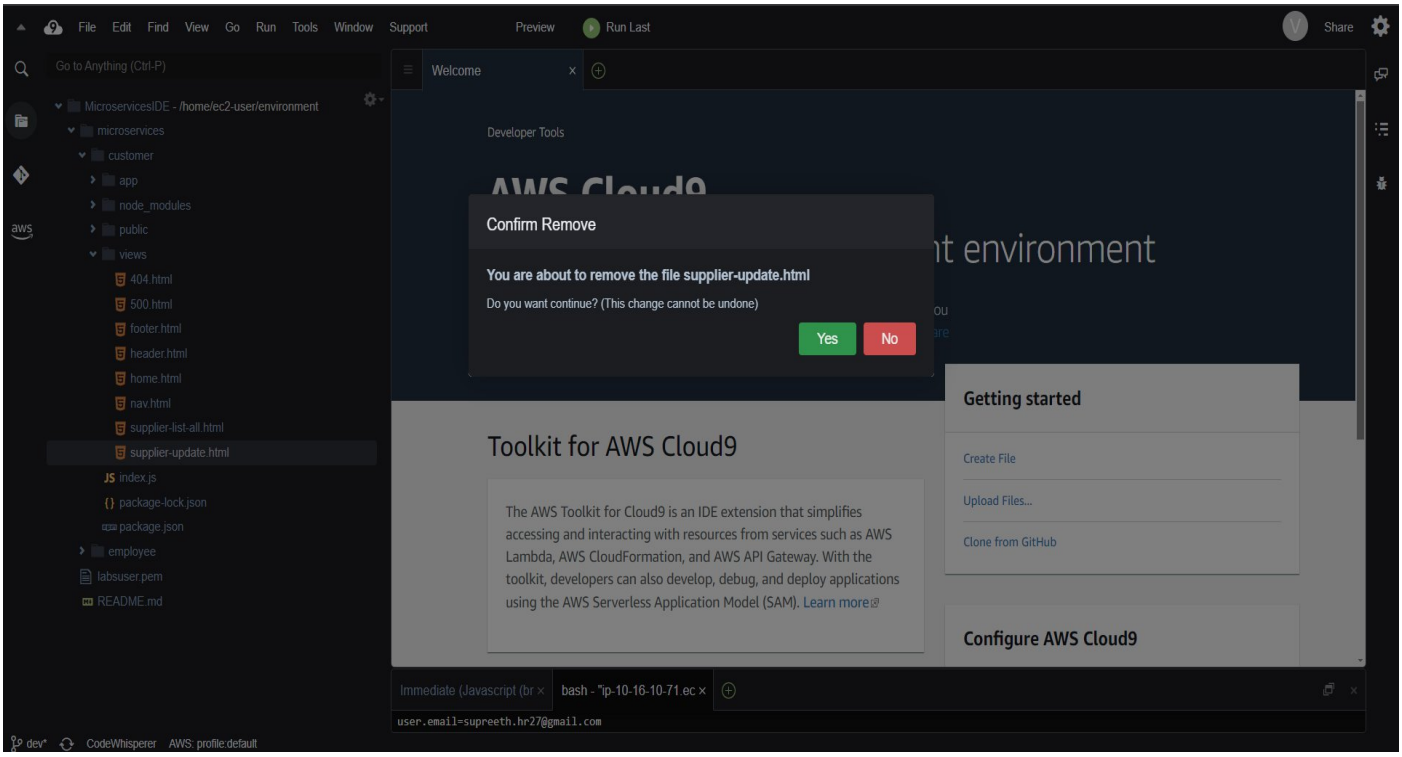


- To update the navigation, I edited the customer/views/nav.html file to change line 3 to "Coffee suppliers" and line 7 to "Customer home." I added a new line after line 8 to include an HTML link (Administrator link) to provide access to the administrative pages for employees.

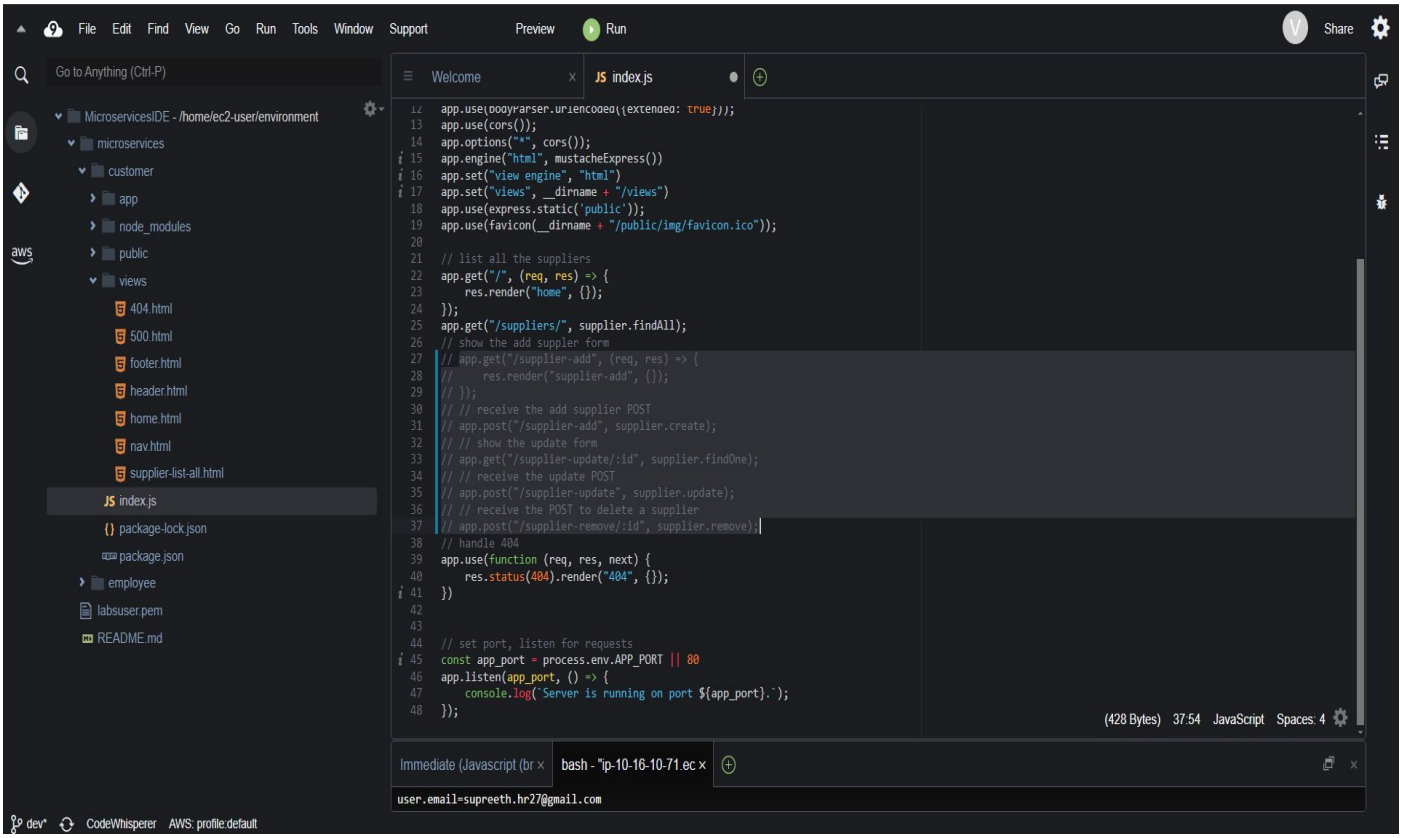
- In the customer/views/supplier-list-all.html file, I removed line 32, which had the "Add a new supplier" button, and lines 26 and 27, which had the "badge badge-info" and "supplier-update" options to prevent customers from editing supplier entries. Also, I deleted unnecessary .html files, including supplier-add.html, supplier-form-fields.html, and supplier-update.html, as these actions are not required for the customer microservice.

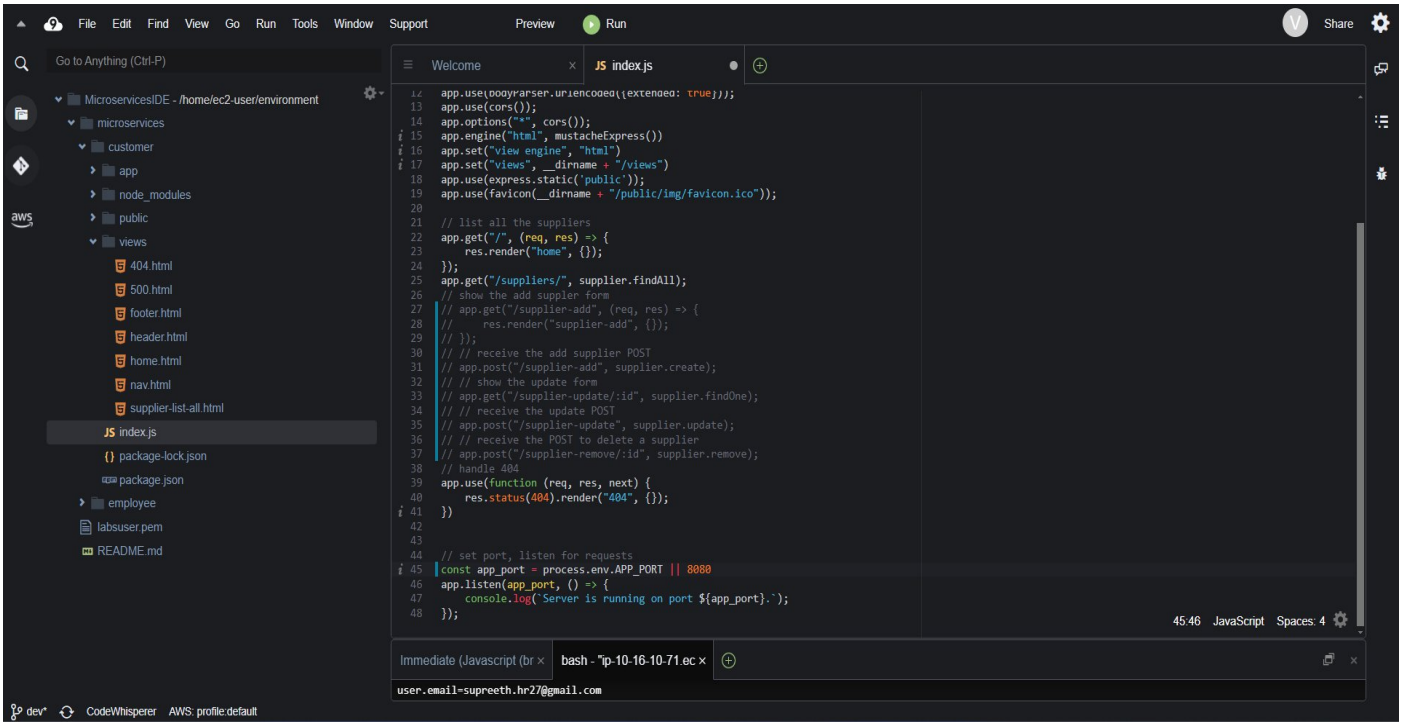




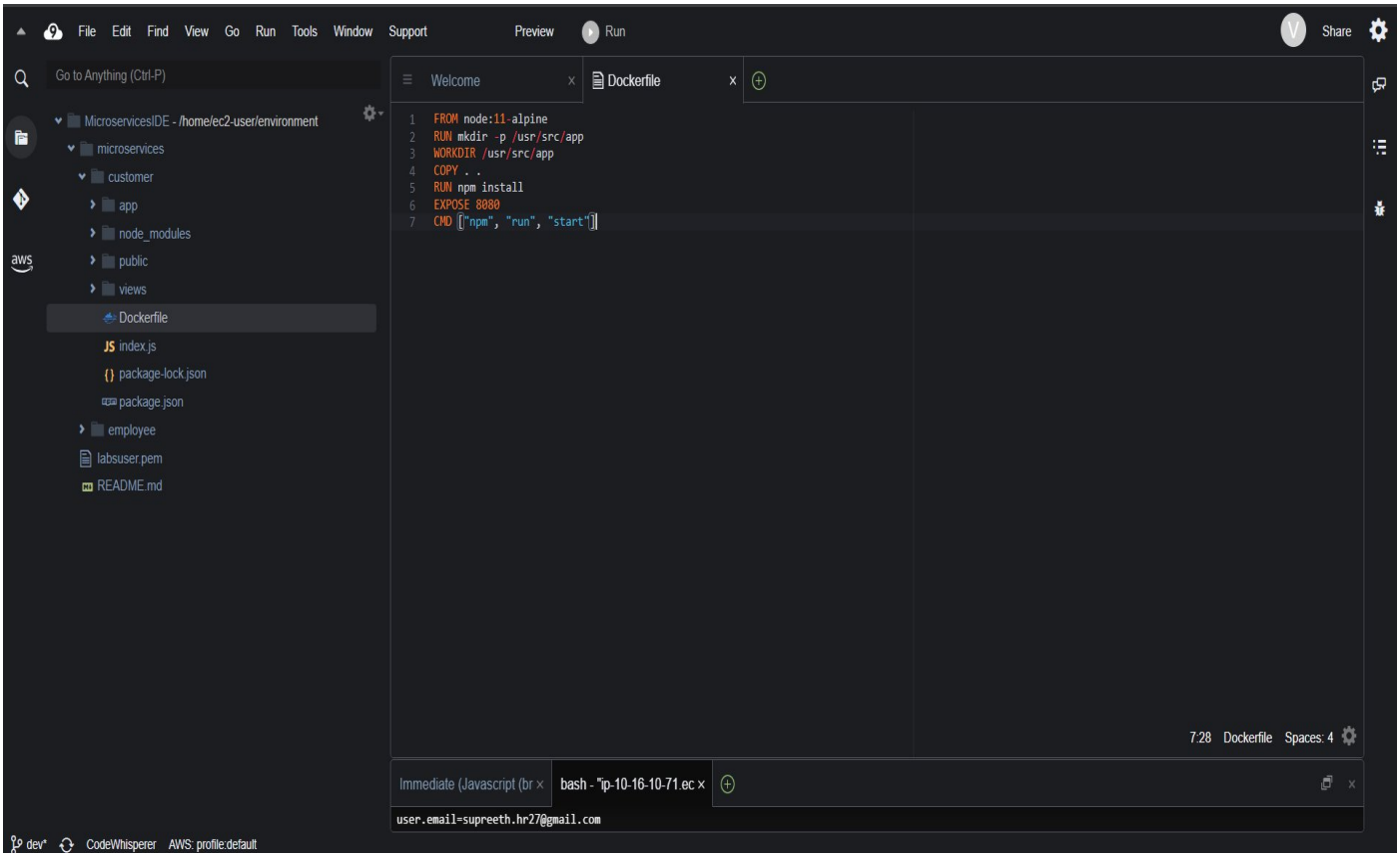


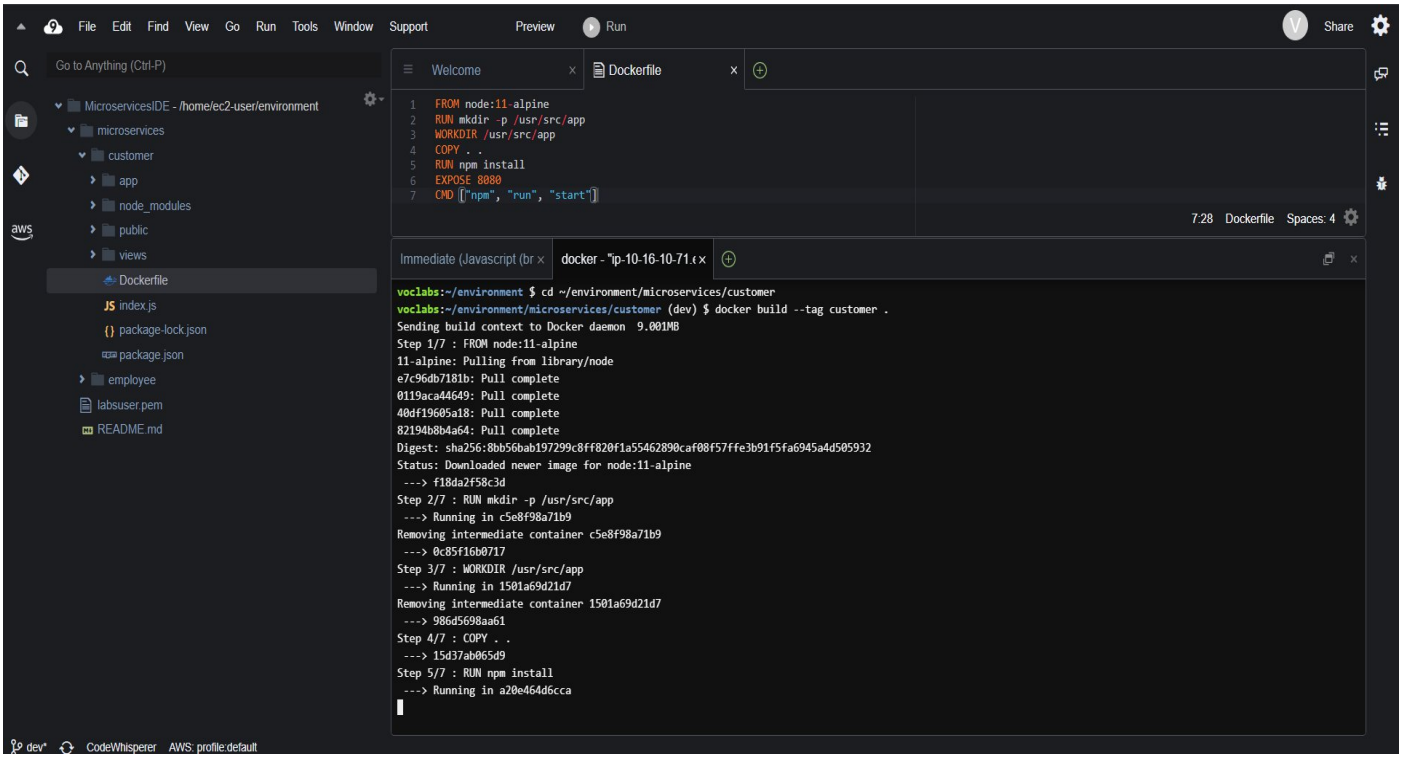
- In customer/index.js, I commented out lines 27 to 37 to adjust for Docker container execution, changing the port number on line 45 from 80 to 8080 as this is the standard port for Docker containers. These changes ensure the customer microservice is streamlined and secure focusing solely on read-only operations while allowing employees to access the required functionalities through dedicated links.

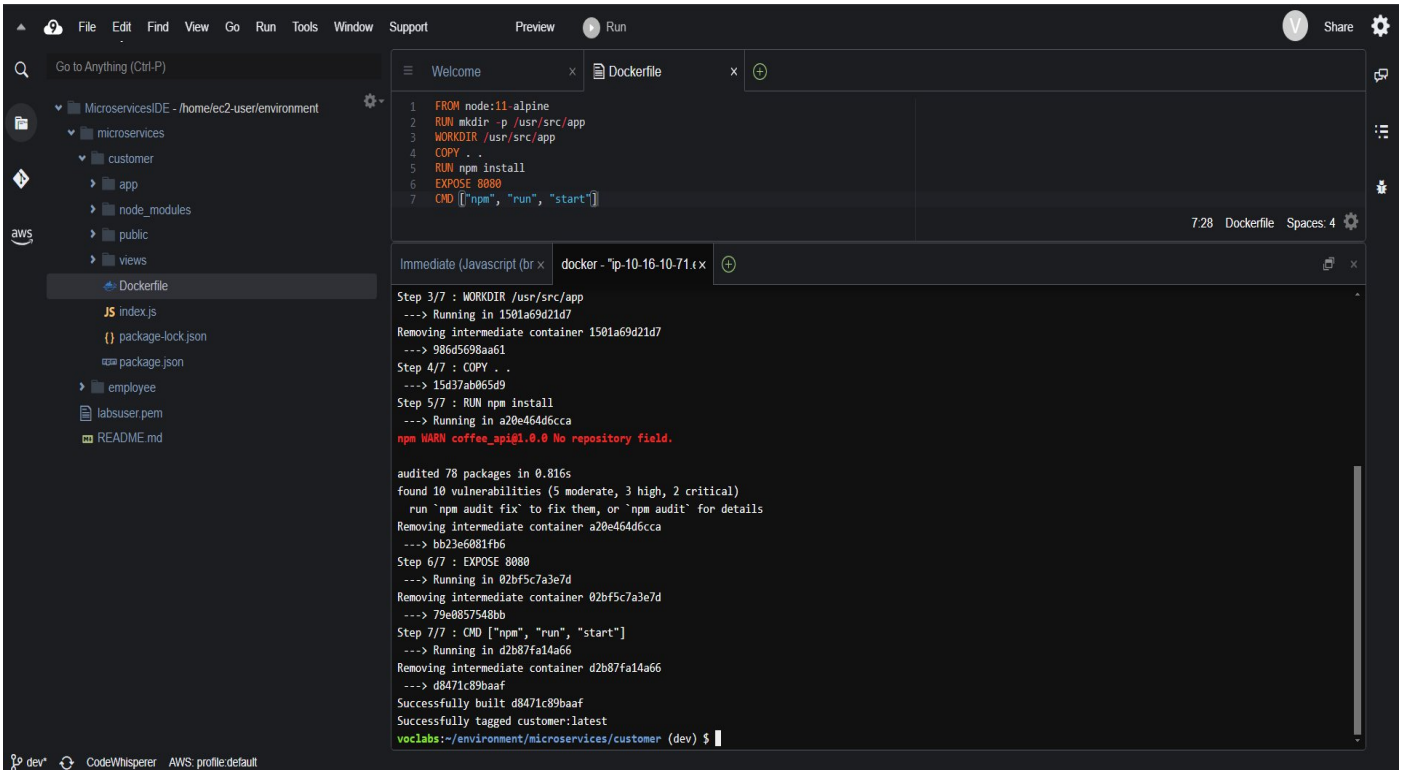




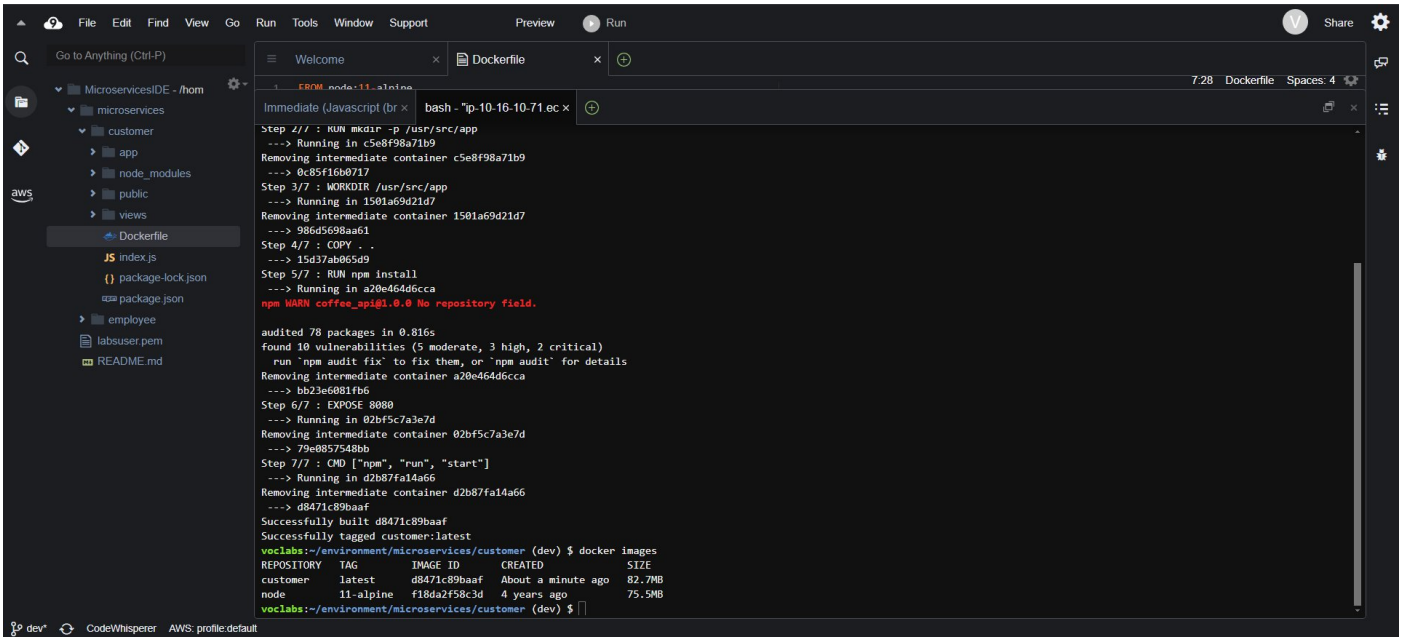
- Task 4.3: To create and test a Docker image for the customer microservice I began by adding a Dockerfile to the customer directory with code that uses the Alpine Linux distribution with Node.js to create the Docker image. I then built the Docker image by navigating to the customer directory in AWS Cloud9 and running **docker build --tag customer .**, which successfully created a Docker image labeled "customer."

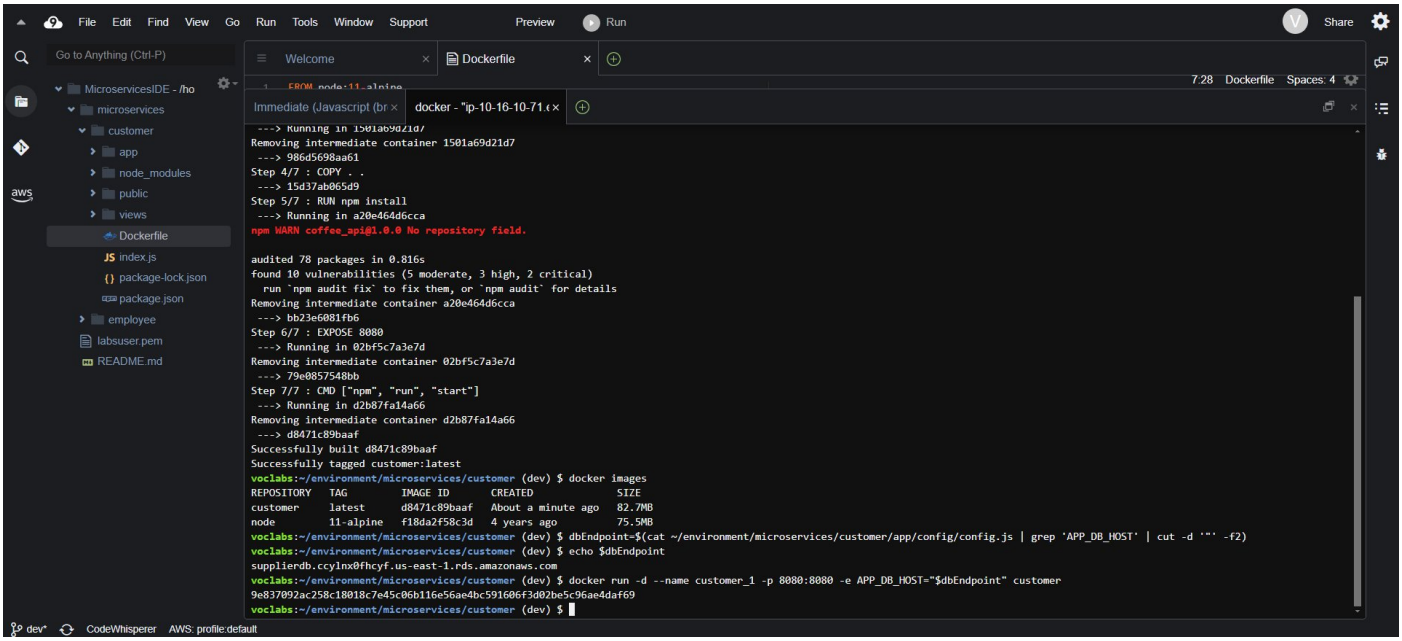


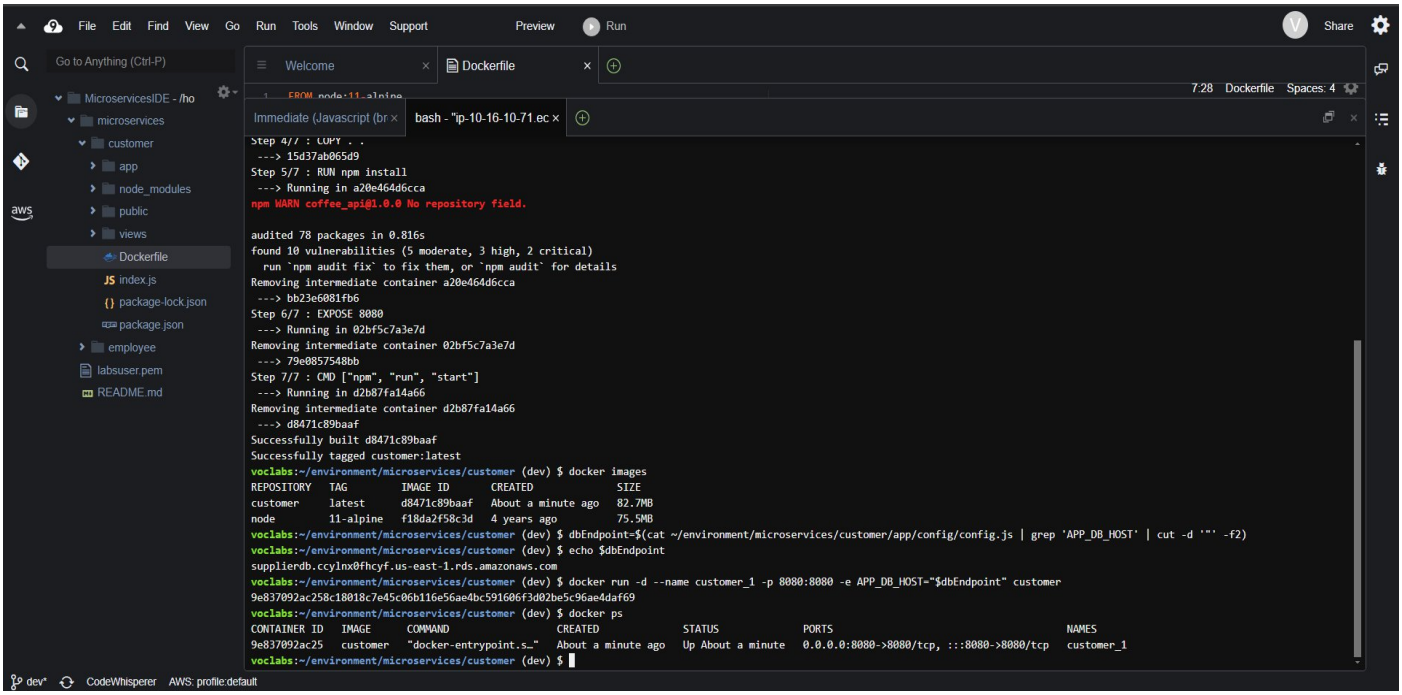




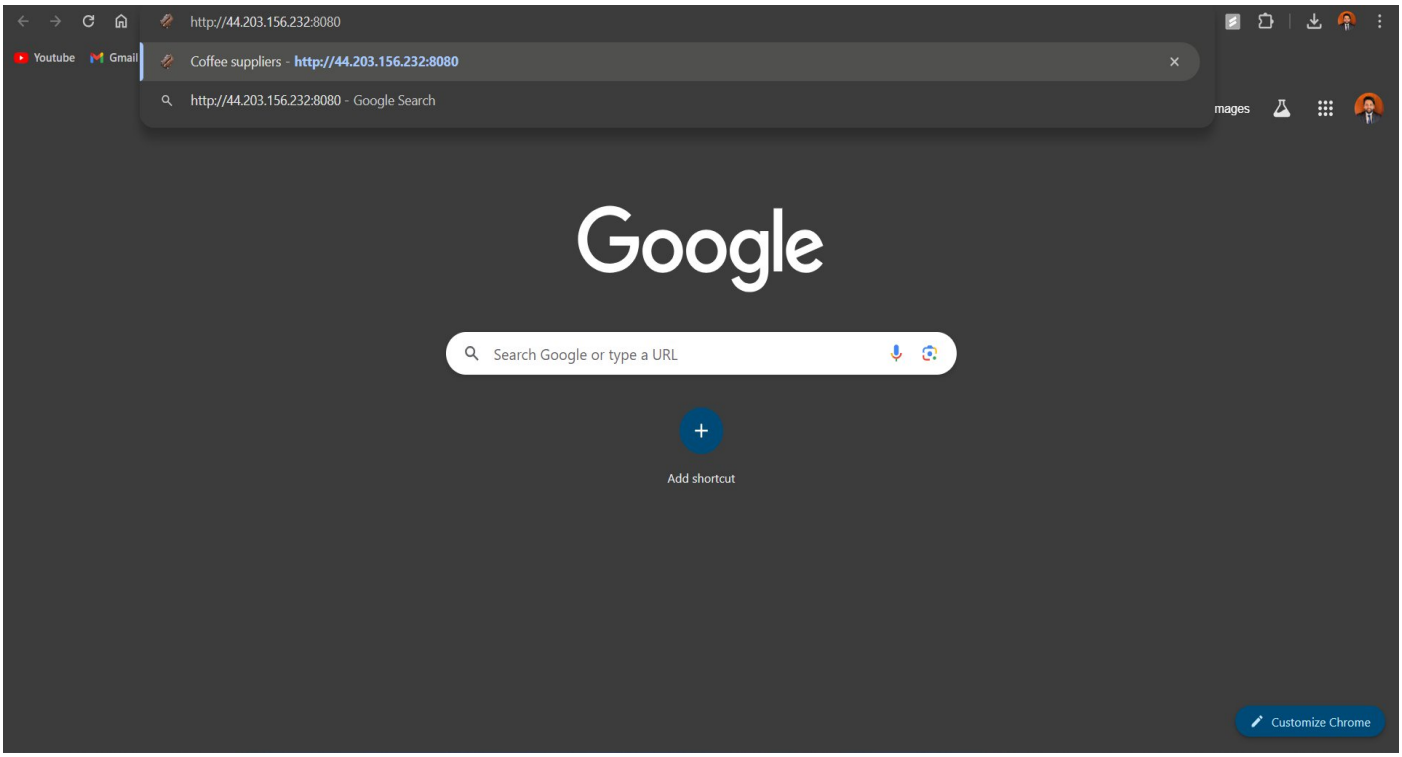
- I verified that the image was created by listing the Docker images with a command that showed the customer-labeled image alongside the base Alpine Linux node image. To launch the Docker container on port 8080 I set a dbEndpoint environment variable to point to the RDS database by extracting the value from config.js, then used **docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint"** customer to start the container. This command launched the container mapping port 8080 from the host to the container and passed the database endpoint as an environment variable.

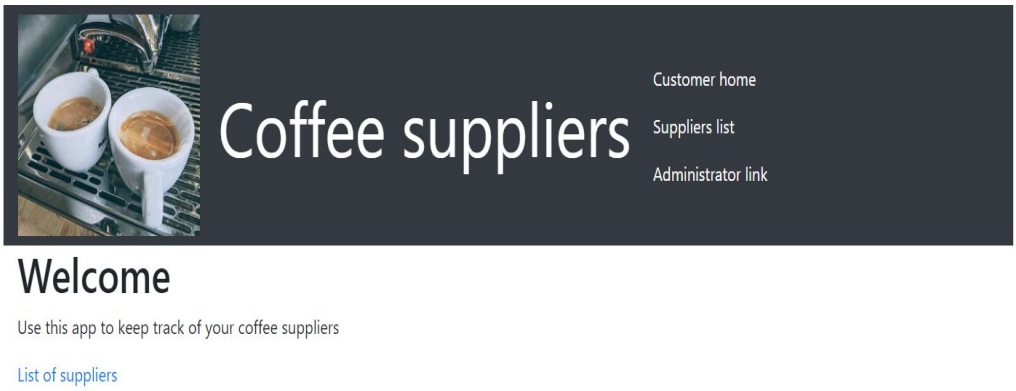

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Dockerfile
7:28 Dockerfile Spaces: 4
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
Step 2/7 : RUN mkdir -p /usr/src/app
---> Running in c5e8f98a71b9
Removing intermediate container c5e8f98a71b9
---> 0c85f16b0717
Step 3/7 : WORKDIR /usr/src/app
---> Running in 1501a69d21d7
Removing intermediate container 1501a69d21d7
---> 986d5698aa61
Step 4/7 : COPY . .
---> 15d37ab065d9
Step 5/7 : RUN npm install
---> Running in a20e464d6cca
npm WARN coffee_api@1.0.0 No repository field.
audited 78 packages in 0.816s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container a20e464d6cca
---> bb23e6081fb6
Step 6/7 : EXPOSE 8080
---> Running in 02bf5c7a3e7d
Removing intermediate container 02bf5c7a3e7d
---> 79e0857548bb
Step 7/7 : CMD ["npm", "run", "start"]
---> Running in d2b87fa14a66
Removing intermediate container d2b87fa14a66
---> d8471c89baaf
Successfully built d8471c89baaf
Successfully tagged customer:latest
voclabs:~/environment/microservices/customer (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
customer latest d8471c89baaf About a minute ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment/microservices/customer (dev) $
dev*
CodeWhisperer
AWS: profile:default

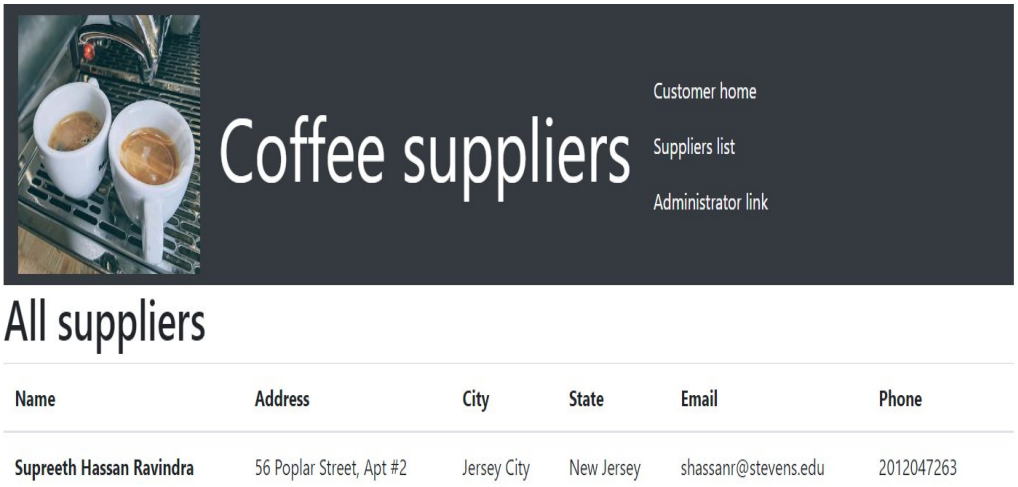

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Dockerfile
7:28 Dockerfile Spaces: 4
Immediate (Javascript (br
docker - "ip-10-16-10-71.e
---> Running in 1501a69d21d7
Removing intermediate container 1501a69d21d7
---> 986d5698aa61
Step 4/7 : COPY . .
---> 15d37ab065d9
Step 5/7 : RUN npm install
---> Running in a20e464d6cca
npm WARN coffee_api@1.0.0 No repository field.
audited 78 packages in 0.816s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container a20e464d6cca
---> bb23e6081fb6
Step 6/7 : EXPOSE 8080
---> Running in 02bf5c7a3e7d
Removing intermediate container 02bf5c7a3e7d
---> 79e0857548bb
Step 7/7 : CMD ["npm", "run", "start"]
---> Running in d2b87fa14a66
Removing intermediate container d2b87fa14a66
---> d8471c89baaf
Successfully built d8471c89baaf
Successfully tagged customer:latest
voclabs:~/environment/microservices/customer (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
customer latest d8471c89baaf About a minute ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment/microservices/customer (dev) $ dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment/microservices/customer (dev) $ echo $dbEndpoint
supplierdb.ccylnx0fhcyf.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/customer (dev) $ docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint" customer
9e837092ac258c18018c7e45c06b116e56ae4bc591606f3d02be5c96ae4daf69
voclabs:~/environment/microservices/customer (dev) $
dev*
CodeWhisperer
AWS: profile:default


File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
Welcome
Dockerfile
7:28 Dockerfile Spaces: 4
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
Step 4/7 : COPY . .
---> 15d37ab065d9
Step 5/7 : RUN npm install
---> Running in a20e464d6cca
npm WARN coffee_api@1.0.0 No repository field.
audited 78 packages in 0.816s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container a20e464d6cca
---> bb23e6081fb6
Step 6/7 : EXPOSE 8080
---> Running in 02bf5c7a3e7d
Removing intermediate container 02bf5c7a3e7d
---> 79e0857548bb
Step 7/7 : CMD ["npm", "run", "start"]
---> Running in d2b87fa14a66
Removing intermediate container d2b87fa14a66
---> d8471c89baaf
Successfully built d8471c89baaf
Successfully tagged customer:latest
voclabs:~/environment/microservices/customer (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
customer latest d8471c89baaf About a minute ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment/microservices/customer (dev) $ dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment/microservices/customer (dev) $ echo $dbEndpoint
supplierdb.ccylnx0fhcyf.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/customer (dev) $ docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint" customer
9e837092ac258c18018c7e45c06b116e56ae4bc591606f3d02be5c96ae4daf69
voclabs:~/environment/microservices/customer (dev) $ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9e837092ac25 customer "docker-entrypoint.s…" About a minute ago Up About a minute 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp customer_1
voclabs:~/environment/microservices/customer (dev) $
dev*
CodeWhisperer
AWS: profile:default

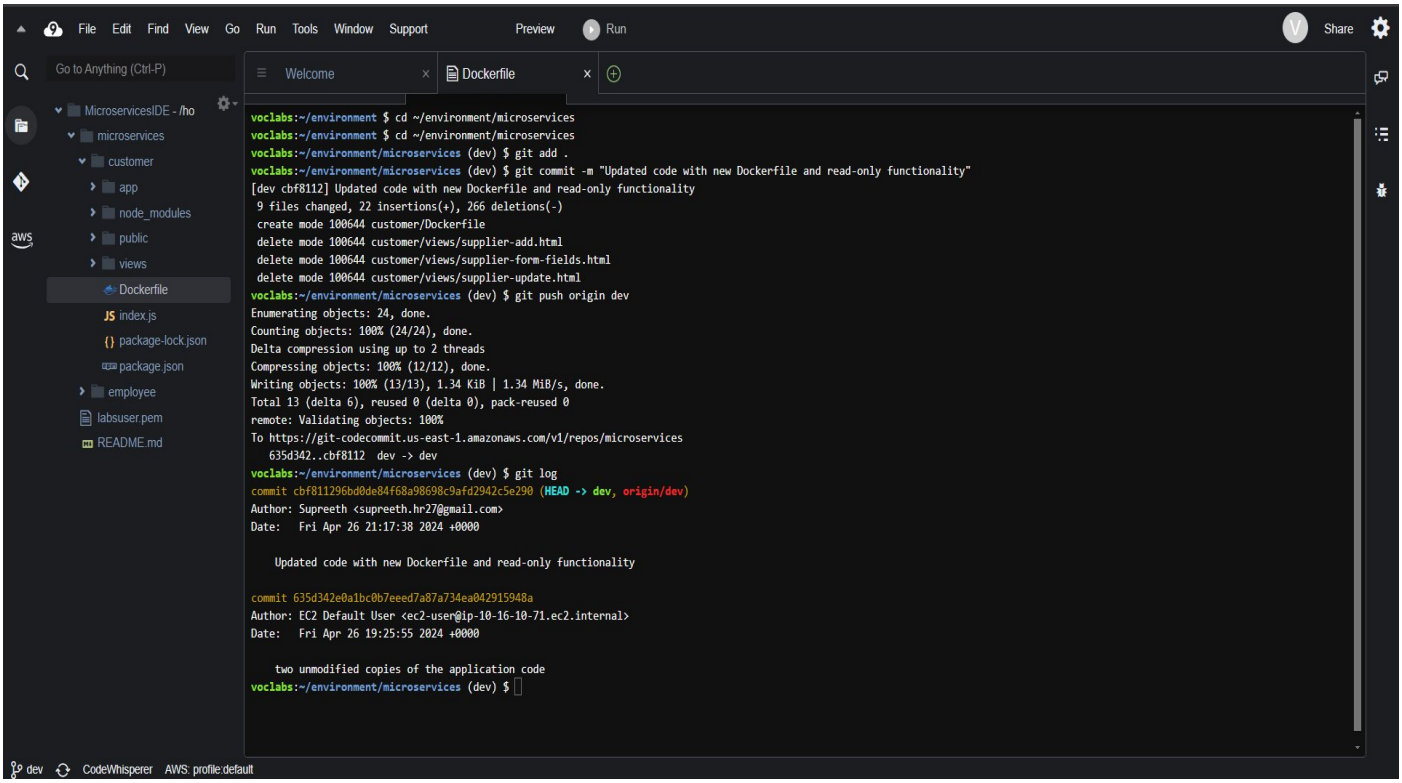
- To confirm that the customer microservice was running, I checked the currently running Docker containers, and then opened the web application in a browser by entering the public IPv4 address of the AWS Cloud9 instance followed by :8080. I verified that the application loaded and navigated to the "List of Suppliers" to confirm the presence of the supplier entries added earlier. I ensured that the "Add a new supplier" button and edit buttons were removed from the suppliers page.

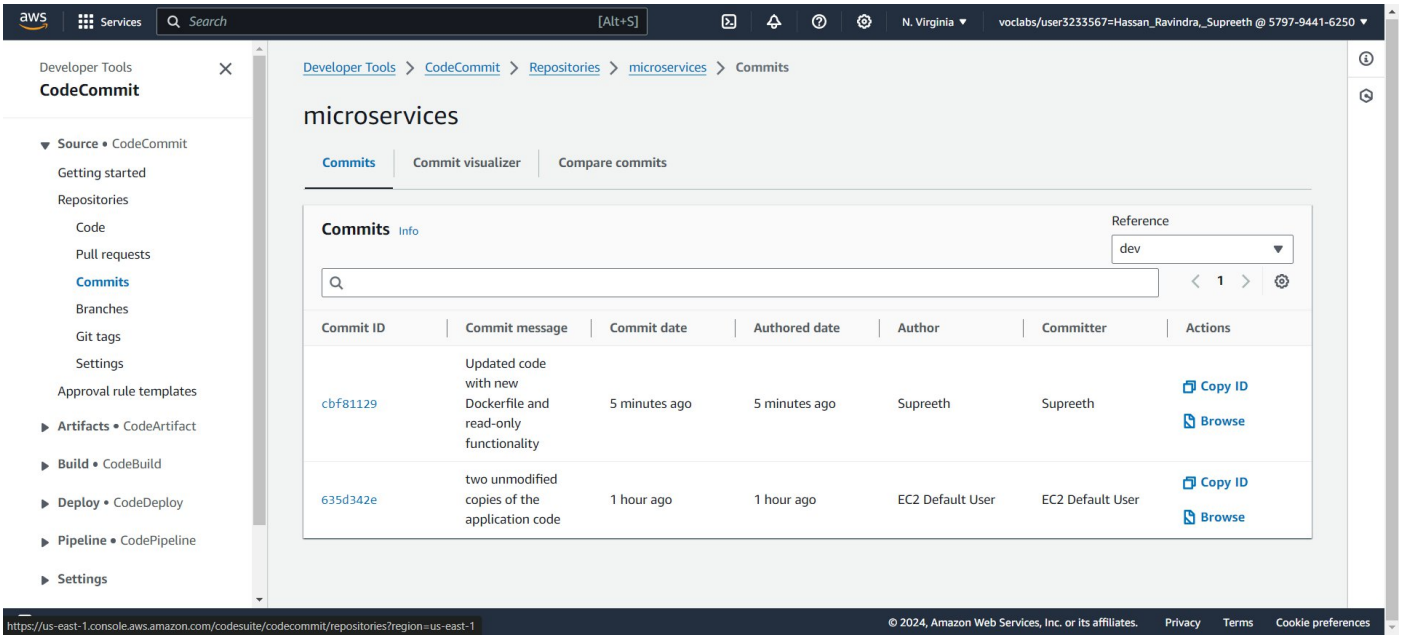


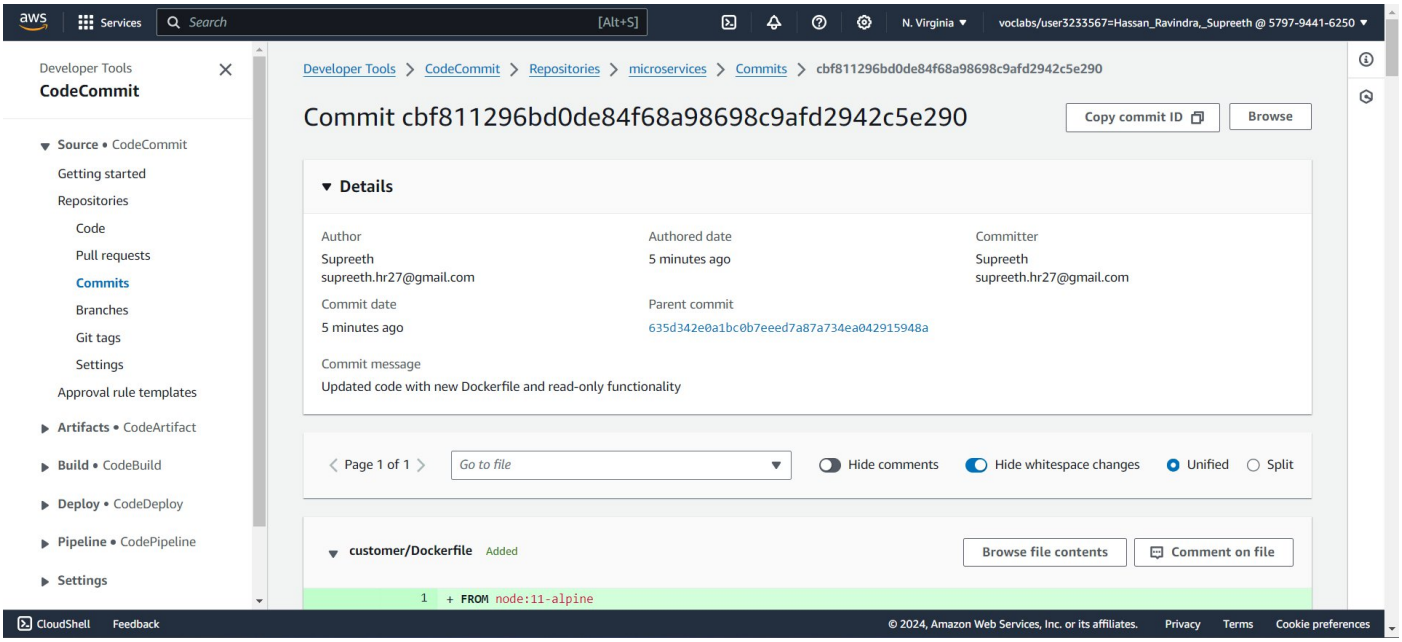


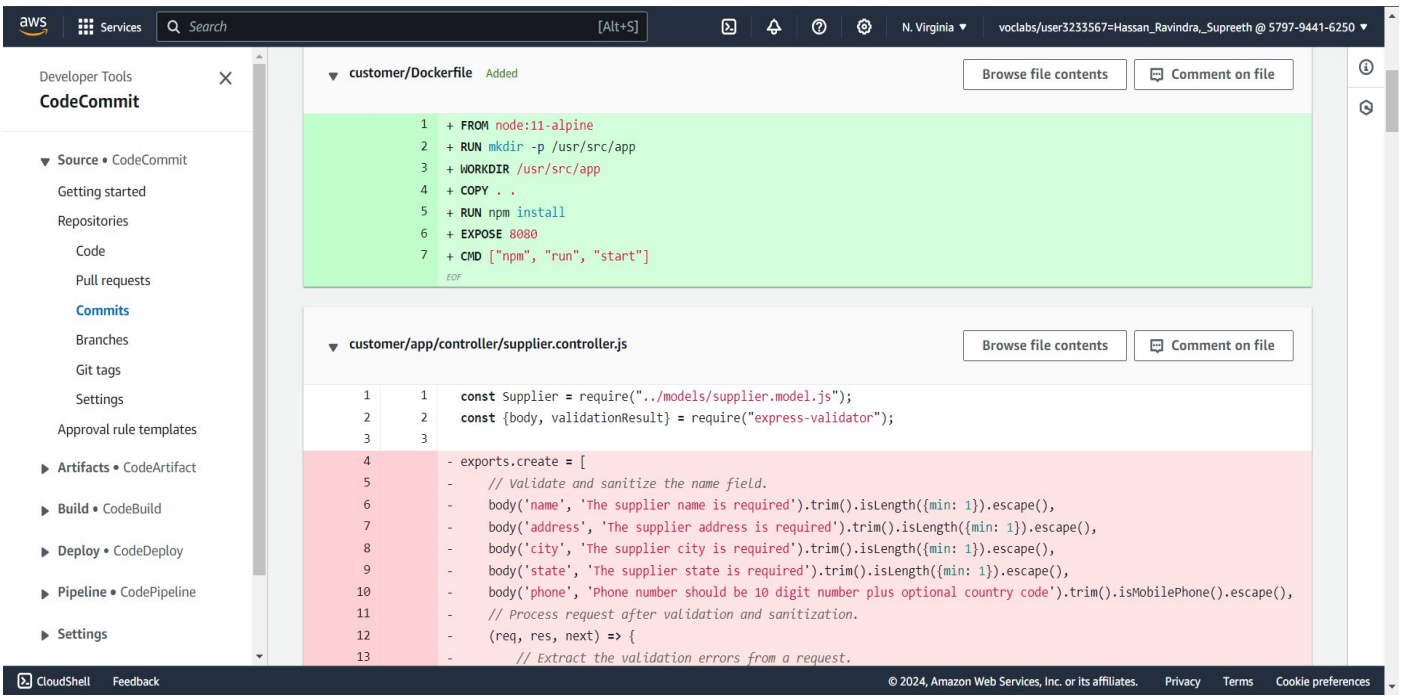


- I committed and pushed the source code changes to CodeCommit using either the Git source control panel in Cloud9 or the git commit and git push commands. Finally, I observed the commit details in the CodeCommit console where I examined the differences in the code, with deleted lines shown in red and added lines shown in green, allowing me to track all changes in the most recent commit.

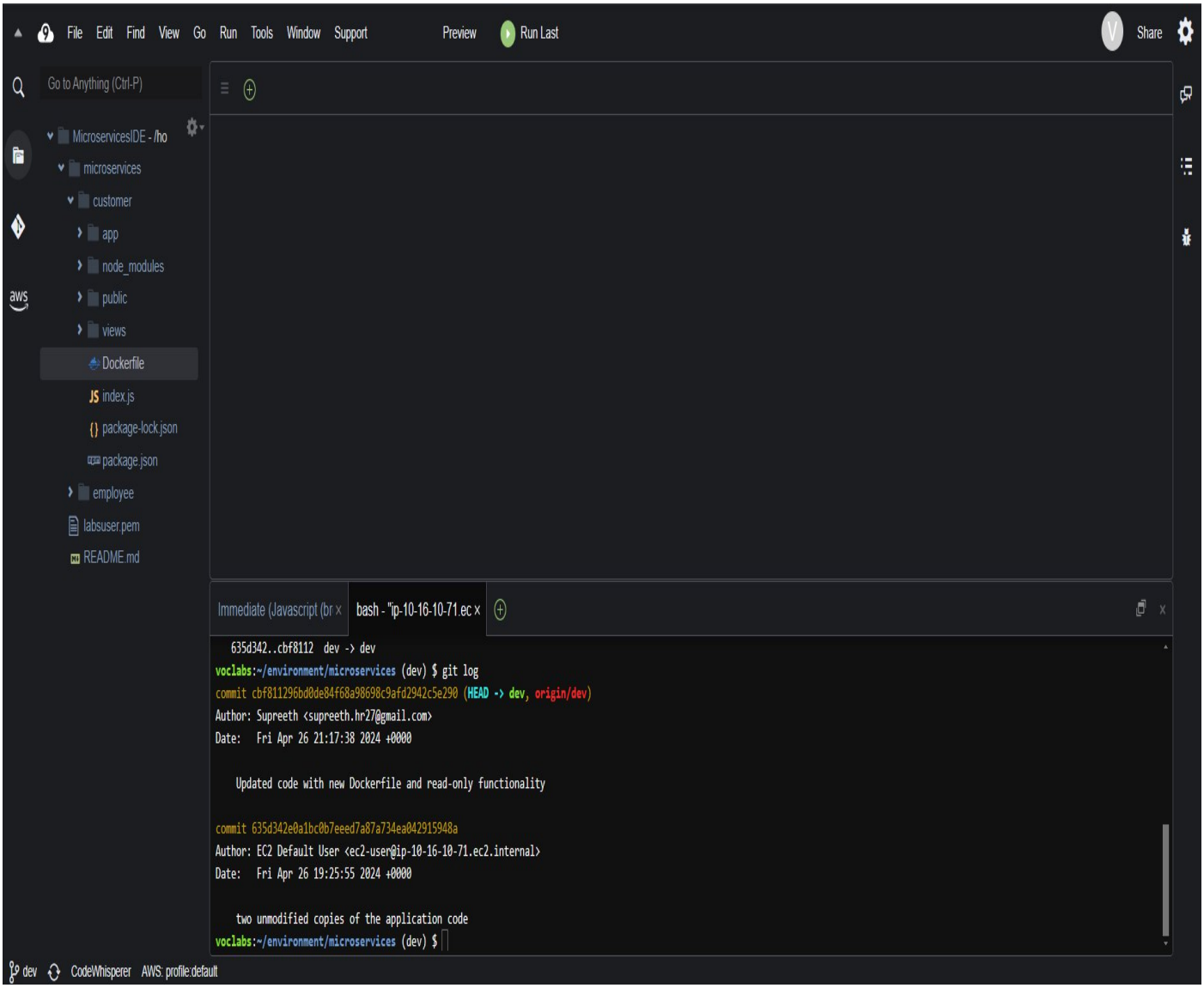


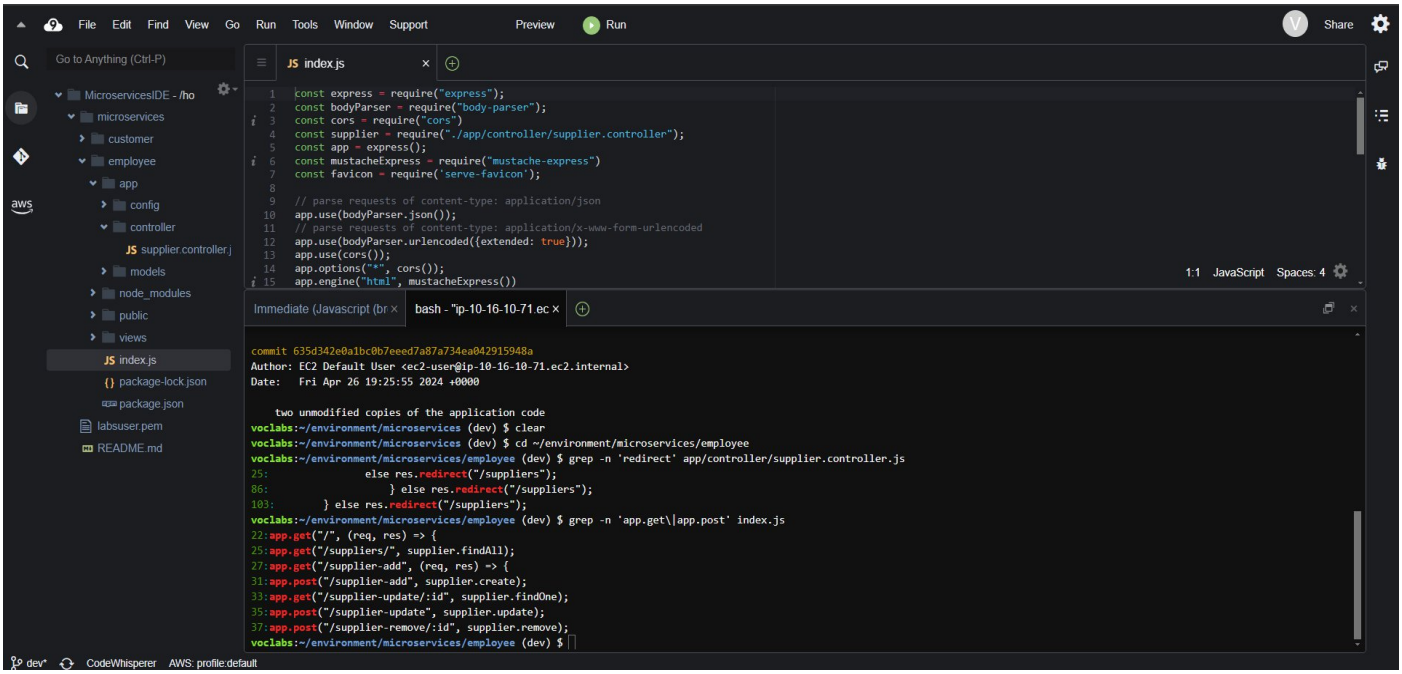

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • 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
cbf81129
Updated code with new Dockerfile and read-only functionality
5 minutes ago
5 minutes ago
Supreeth
Supreeth
Copy ID
Browse
635d342e
two unmodified copies of the application code
1 hour ago
1 hour ago
EC2 Default User
EC2 Default User
Copy ID
Browse
https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories?region=us-east-1
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

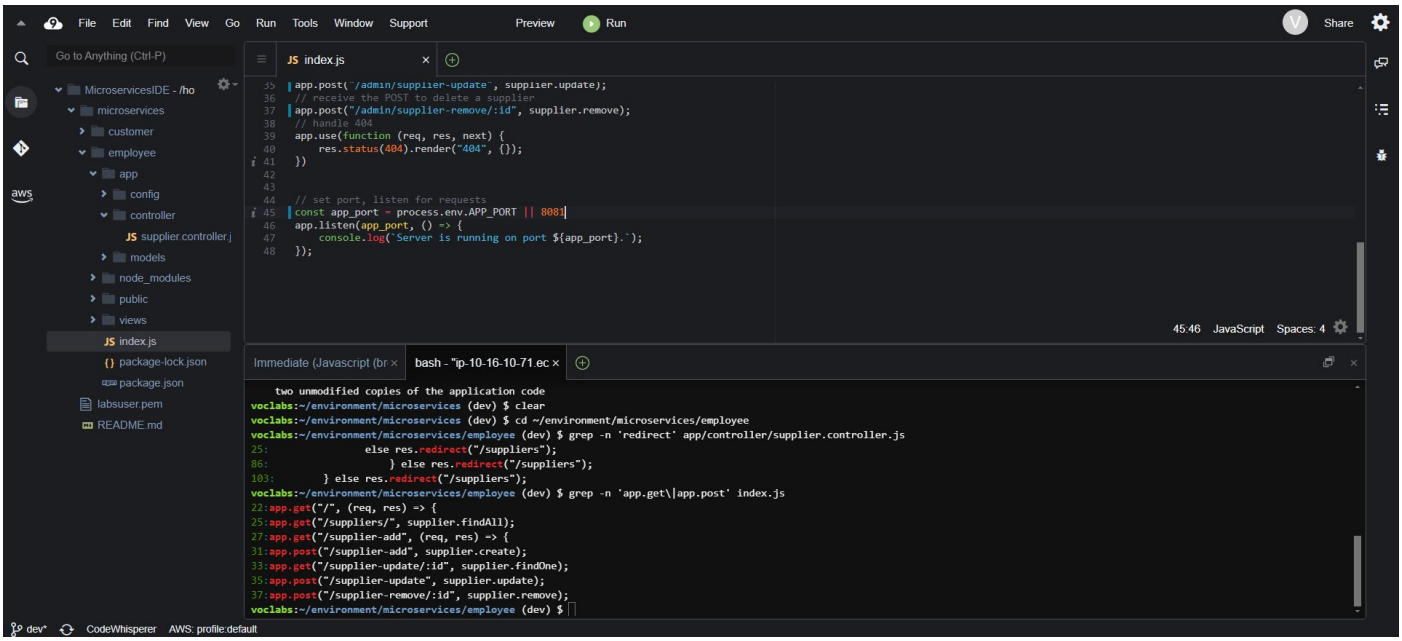

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Developer Tools > CodeCommit > Repositories > microservices > Commits > cbf811296bd0de84f68a98698c9afd2942c5e290
Commit cbf811296bd0de84f68a98698c9afd2942c5e290
Copy commit ID
Browse
Details
Author
Supreeth
supreeth.hr27@gmail.com
Authored date
5 minutes ago
Committer
Supreeth
supreeth.hr27@gmail.com
Commit date
5 minutes ago
Parent commit
635d342e0a1bc0b7eeed7a87a734ea042915948a
Commit message
Updated code with new Dockerfile and read-only functionality
Page 1 of 1
Go to file
Hide comments
Hide whitespace changes
Unified
Split
customer/Dockerfile
Added
Browse file contents
Comment on file
1 + FROM node:11-alpine
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

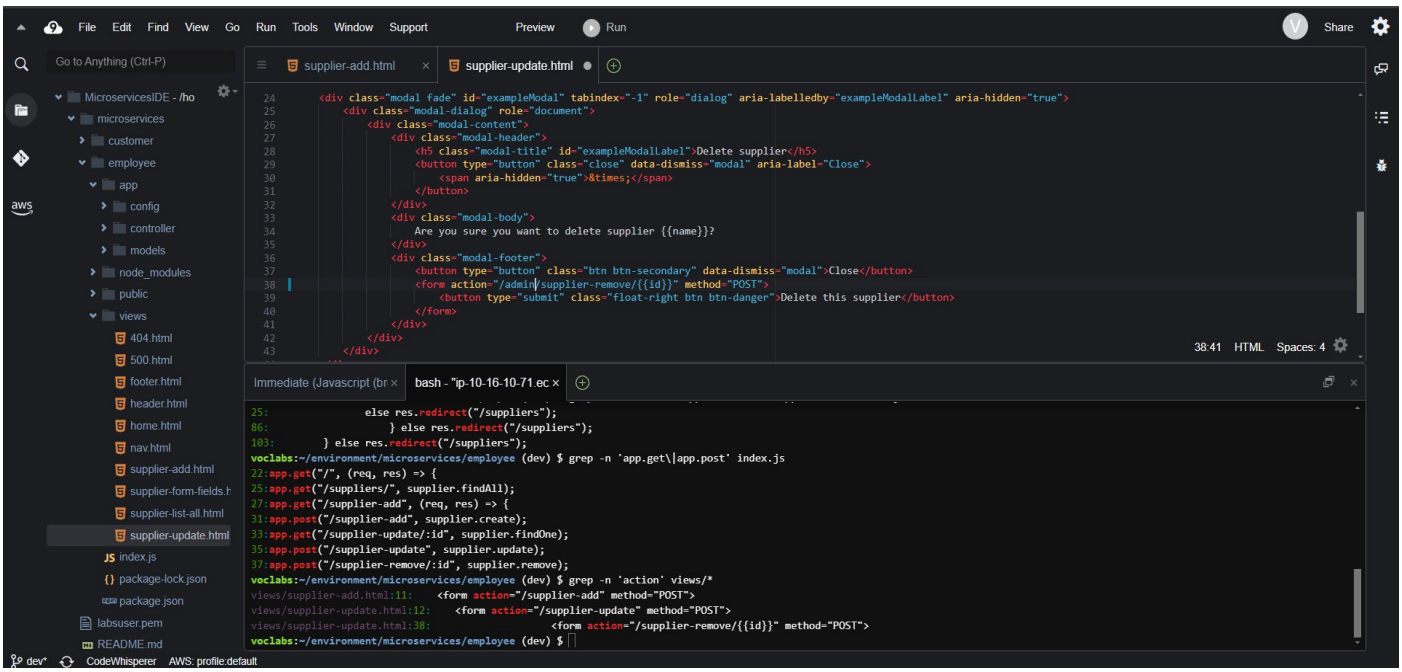

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
customer/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
customer/app/controller/supplier.controller.js
Browse file contents
Comment on file
1 1 const Supplier = require("../models/supplier.model.js");
2 2 const {body, validationResult} = require("express-validator");
3 3
4 - exports.create = [
5 - // Validate and sanitize the name field.
6 - body('name', 'The supplier name is required').trim().isLength({min: 1}).escape(),
7 - body('address', 'The supplier address is required').trim().isLength({min: 1}).escape(),
8 - body('city', 'The supplier city is required').trim().isLength({min: 1}).escape(),
9 - body('state', 'The supplier state is required').trim().isLength({min: 1}).escape(),
10 - body('phone', 'Phone number should be 10 digit number plus optional country code').trim().isMobilePhone().escape(),
11 - // Process request after validation and sanitization.
12 - (req, res, next) => {
13 - // Extract the validation errors from a request.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

- Task 4.4: To configure the employee microservice for adding new entries or modifying existing ones- I made several changes to the source code to ensure the microservice serves content under the /admin/ path, allowing it to work with an Application Load Balancer for traffic routing.
- I accessed the employee directory in the AWS Cloud9 IDE and updated the employee/app/controller/supplier.controller.js file where I prepended /admin to the redirect paths to ensure the correct URL structure for routing. This was identified using the grep -n 'redirect' command to locate the relevant lines.
- I modified employee/index.js to prepend /admin to the first parameter of all app.get and app.post calls. I also changed the port number to 8081 on line 45 to avoid conflicts with other microservices running on the same host.
- In the employee/views/supplier-add.html and employee/views/supplier-update.html files, I prepended /admin to the form action paths ensuring that submission goes to the correct endpoint. Similarly, in employee/views/supplier-list-all.html and employee/views/home.html, I updated the HTML paths ensuring they start with /admin. This was confirmed using the grep -n 'href' command to identify the correct lines to change.




File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /ho
microservices
customer
employee
app
config
controller
supplier.controller.j
models
node_modules
public
views
index.js
package-lock.json
package.json
labsuser.pem
README.md
index.js
1 const express = require("express");
2 const bodyParser = require("body-parser");
3 const cors = require("cors")
4 const supplier = require("./app/controller/supplier.controller");
5 const app = express();
6 const mustacheExpress = require("mustache-express")
7 const favicon = require('serve-favicon');
8
9 // parse requests of content-type: application/json
10 app.use(bodyParser.json());
11 // parse requests of content-type: application/x-www-form-urlencoded
12 app.use(bodyParser.urlencoded({extended: true}));
13 app.use(cors());
14 app.options("*", cors());
15 app.engine("html", mustacheExpress())
1:1 JavaScript Spaces: 4
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
commit 635d342e0a1bc0b7eeed7a87a734ea042915948a
Author: EC2 Default User <ec2-user@ip-10-16-10-71.ec2.internal>
Date: Fri Apr 26 19:25:55 2024 +0000
two unmodified copies of the application code
voclabs:~/environment/microservices (dev) $ clear
voclabs:~/environment/microservices (dev) $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ grep -n 'redirect' app/controller/supplier.controller.js
25: else res.redirect("/suppliers");
86: } else res.redirect("/suppliers");
103: } else res.redirect("/suppliers");
voclabs:~/environment/microservices/employee (dev) $ grep -n 'app.get\|app.post' index.js
22:app.get("/", (req, res) => {
25:app.get("/suppliers/", supplier.findAll);
27:app.get("/supplier-add", (req, res) => {
31:app.post("/supplier-add", supplier.create);
33:app.get("/supplier-update/:id", supplier.findOne);
35:app.post("/supplier-update", supplier.update);
37:app.post("/supplier-remove/:id", supplier.remove);
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)
MicroservicesIDE - /ho
microservices
customer
employee
app
config
controller
supplier.controller.j
models
node_modules
public
views
index.js
package-lock.json
package.json
labsuser.pem
README.md
index.js
app.post(`/admin/suppiier-update`, suppiier.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", {});
})
// set port, listen for requests
const app_port = process.env.APP_PORT || 8081
app.listen(app_port, () => {
console.log(`Server is running on port ${app_port}.`);
});
45:46 JavaScript Spaces: 4
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
two unmodified copies of the application code
voclabs:~/environment/microservices (dev) $ clear
voclabs:~/environment/microservices (dev) $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ grep -n 'redirect' app/controller/supplier.controller.js
25: else res.redirect("/suppliers");
86: } else res.redirect("/suppliers");
103: } else res.redirect("/suppliers");
voclabs:~/environment/microservices/employee (dev) $ grep -n 'app.get\|app.post' index.js
22:app.get("/", (req, res) => {
25:app.get("/suppliers/", supplier.findAll);
27:app.get("/supplier-add", (req, res) => {
31:app.post("/supplier-add", supplier.create);
33:app.get("/supplier-update/:id", supplier.findOne);
35:app.post("/supplier-update", supplier.update);
37:app.post("/supplier-remove/:id", supplier.remove);
voclabs:~/environment/microservices/employee (dev) $
dev*
CodeWhisperer
AWS: profile:default

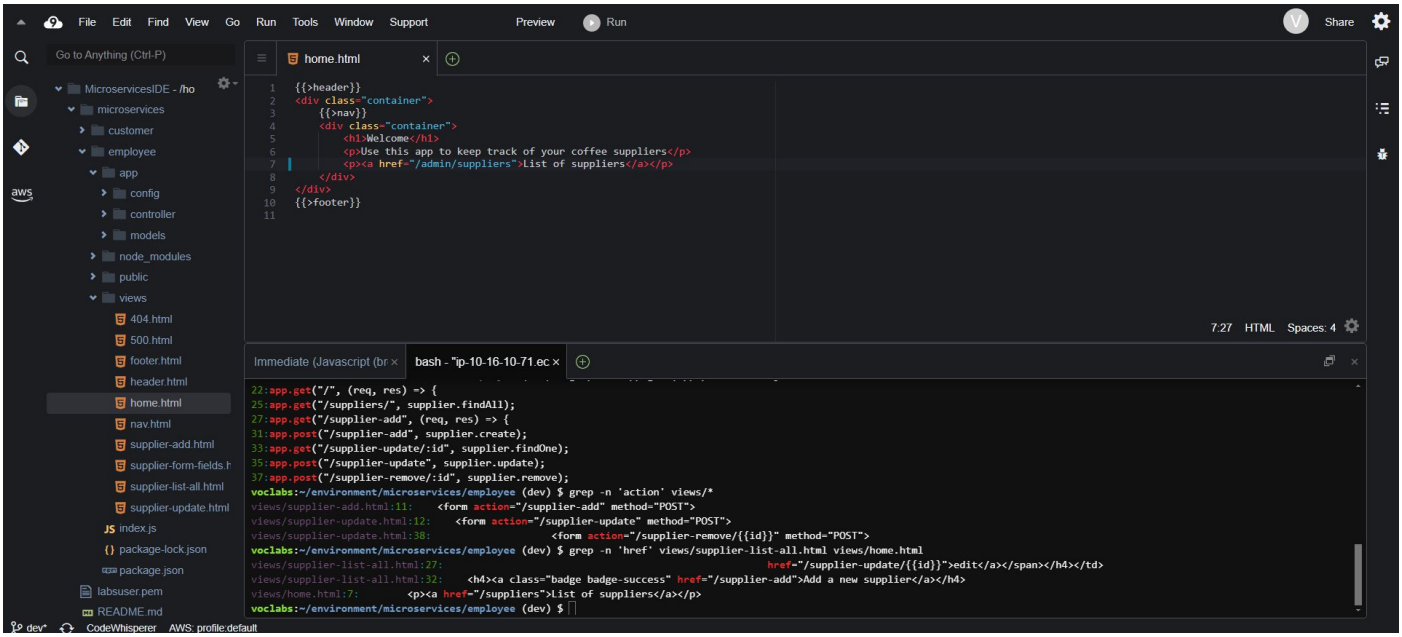

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /ho
microservices
customer
employee
app
config
controller
models
node_modules
public
views
404.html
500.html
footer.html
header.html
home.html
nav.html
supplier-add.html
supplier-form-fields.h
supplier-list-all.html
supplier-update.html
index.js
package-lock.json
package.json
labsuser.pem
README.md
supplier-add.html
supplier-update.html

<h5 class="modal-title" id="exampleModalLabel">Delete supplier</h5>
<button type="button" class="close" data-dismiss="modal" aria-label="Close">
×
</button>

Are you sure you want to delete supplier {{name}}?

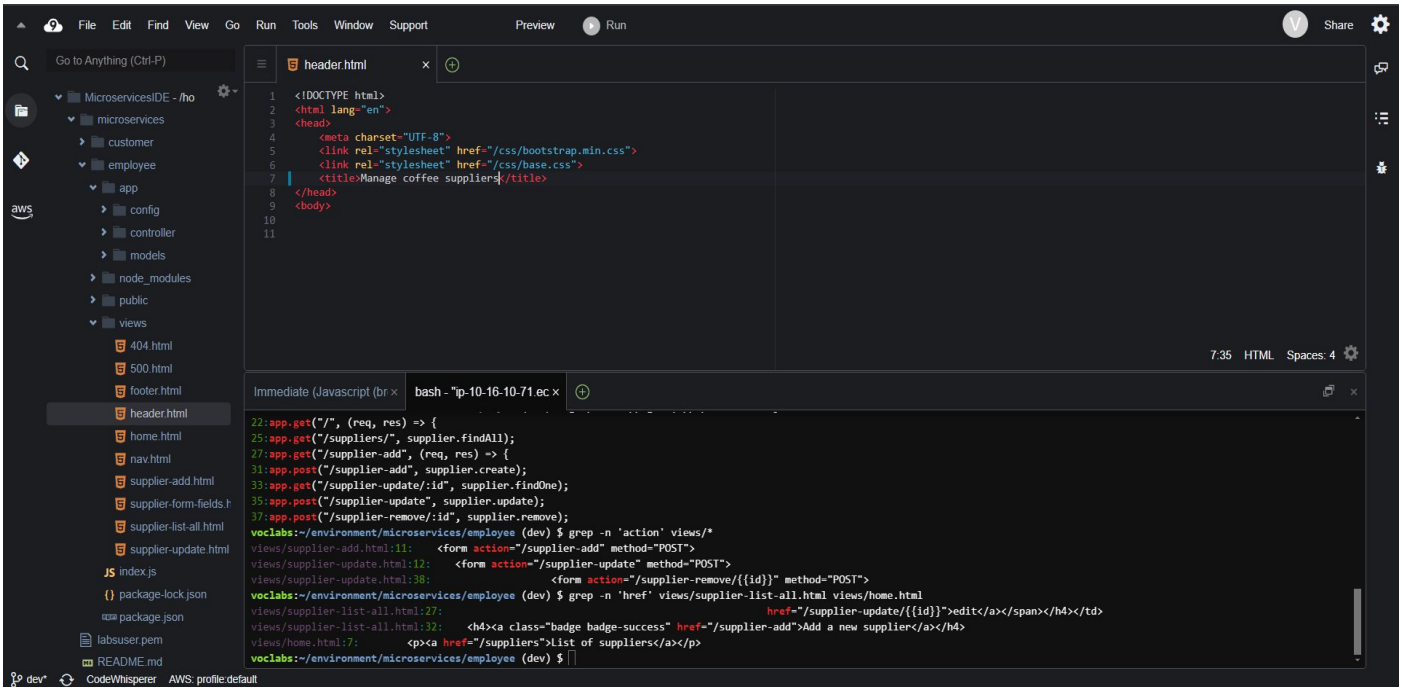
<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
<form action="/admin/supplier-remove/{{id}}" method="POST">
<button type="submit" class="float-right btn btn-danger">Delete this supplier</button>
</form>

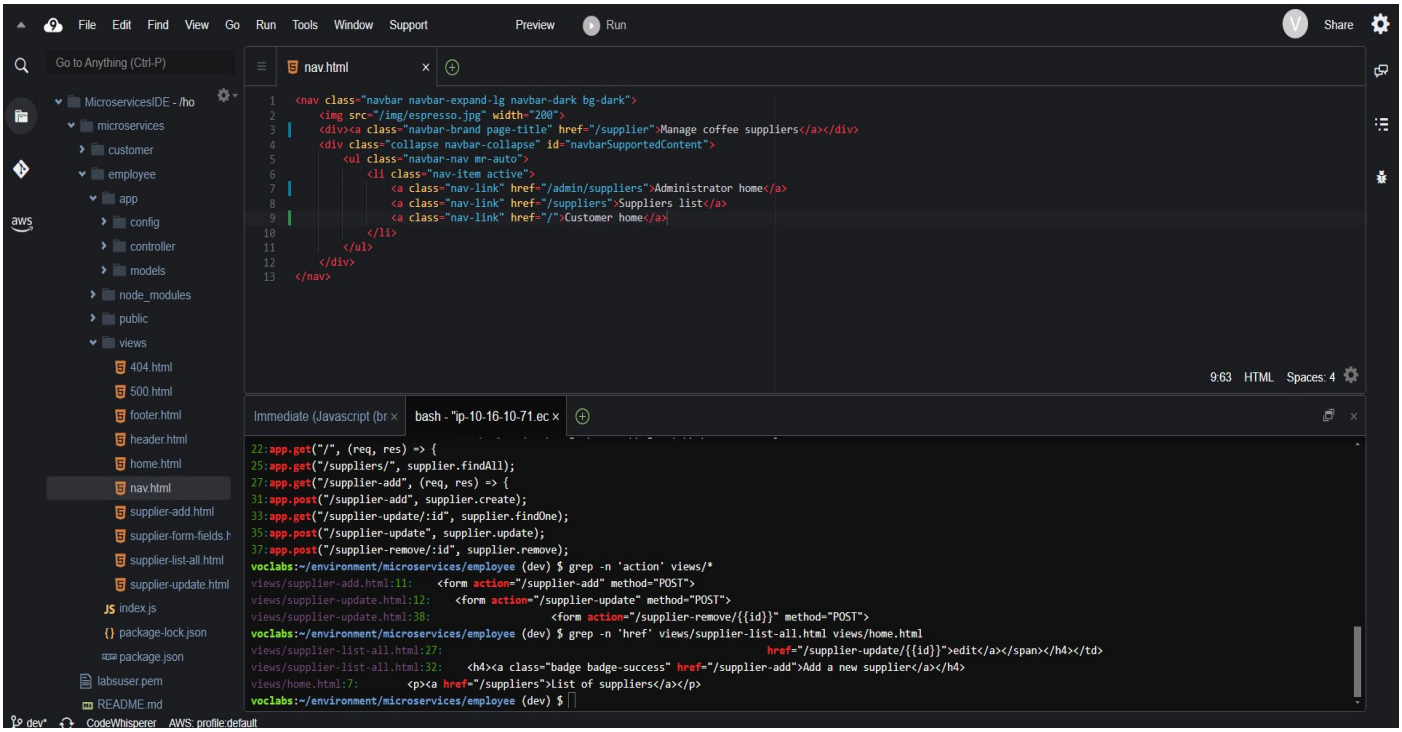
38:41 HTML Spaces: 4
Immediate (Javascript (br
bash - "ip-10-16-10-71.ec
25: else res.redirect("/suppliers");
86: } else res.redirect("/suppliers");
103: } else res.redirect("/suppliers");
voclabs:~/environment/microservices/employee (dev) $ grep -n 'app.get\|app.post' index.js
22:app.get("/", (req, res) => {
25:app.get("/suppliers/", supplier.findAll);
27:app.get("/supplier-add", (req, res) => {
31:app.post("/supplier-add", supplier.create);
33:app.get("/supplier-update/:id", supplier.findOne);
35:app.post("/supplier-update", supplier.update);
37:app.post("/supplier-remove/:id", supplier.remove);
voclabs:~/environment/microservices/employee (dev) $ grep -n 'action' views/*
views/supplier-add.html:11: <form action="/supplier-add" method="POST">
views/supplier-update.html:12: <form action="/supplier-update" method="POST">
views/supplier-update.html:38: <form action="/supplier-remove/{{id}}" method="POST">
voclabs:~/environment/microservices/employee (dev) $
dev*
CodeWhisperer
AWS: profile:default



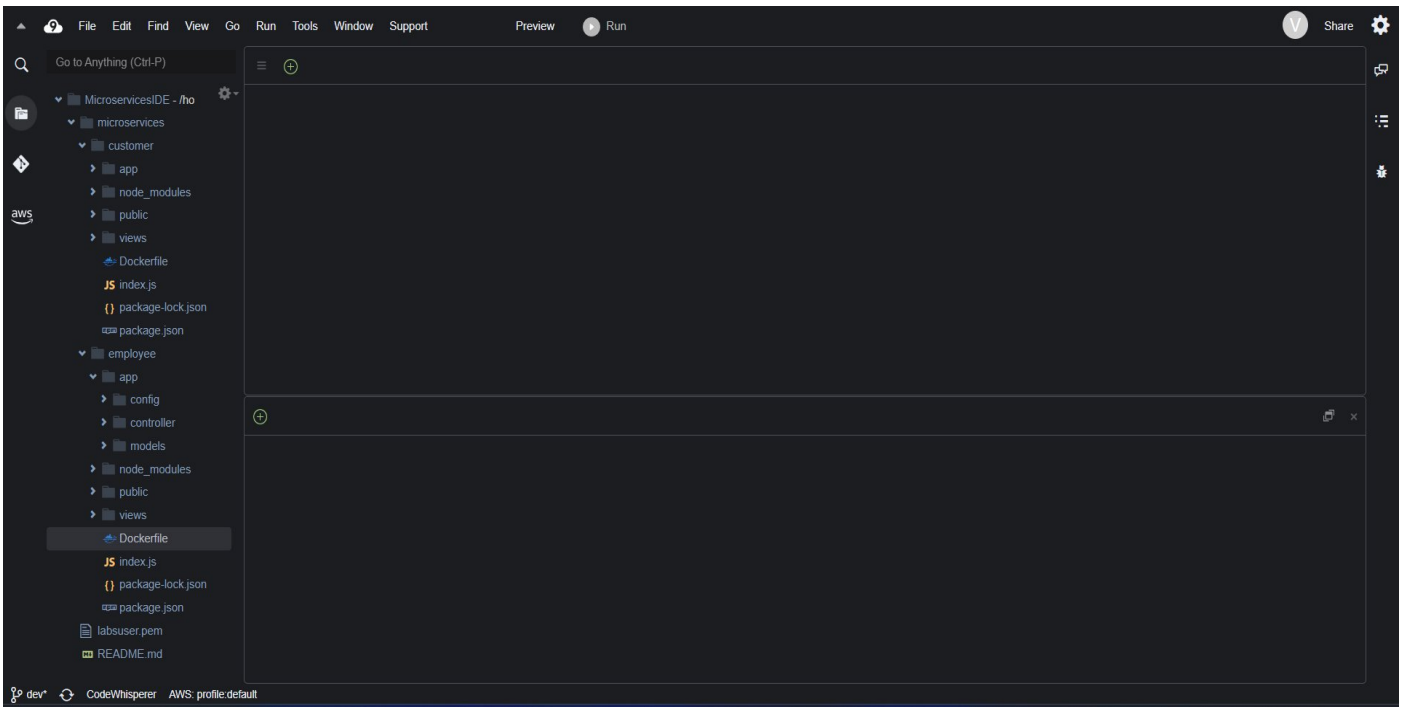
- For the page headers, I updated employee/views/header.html to have the title "Manage coffee suppliers." In the employee/views/nav.html file, I changed line 3 to "Manage coffee suppliers," updated the link on line 7 to /admin/suppliers labeled "Administrator home," and added a new link after line 8 for navigation back to the customer area (Customer home).

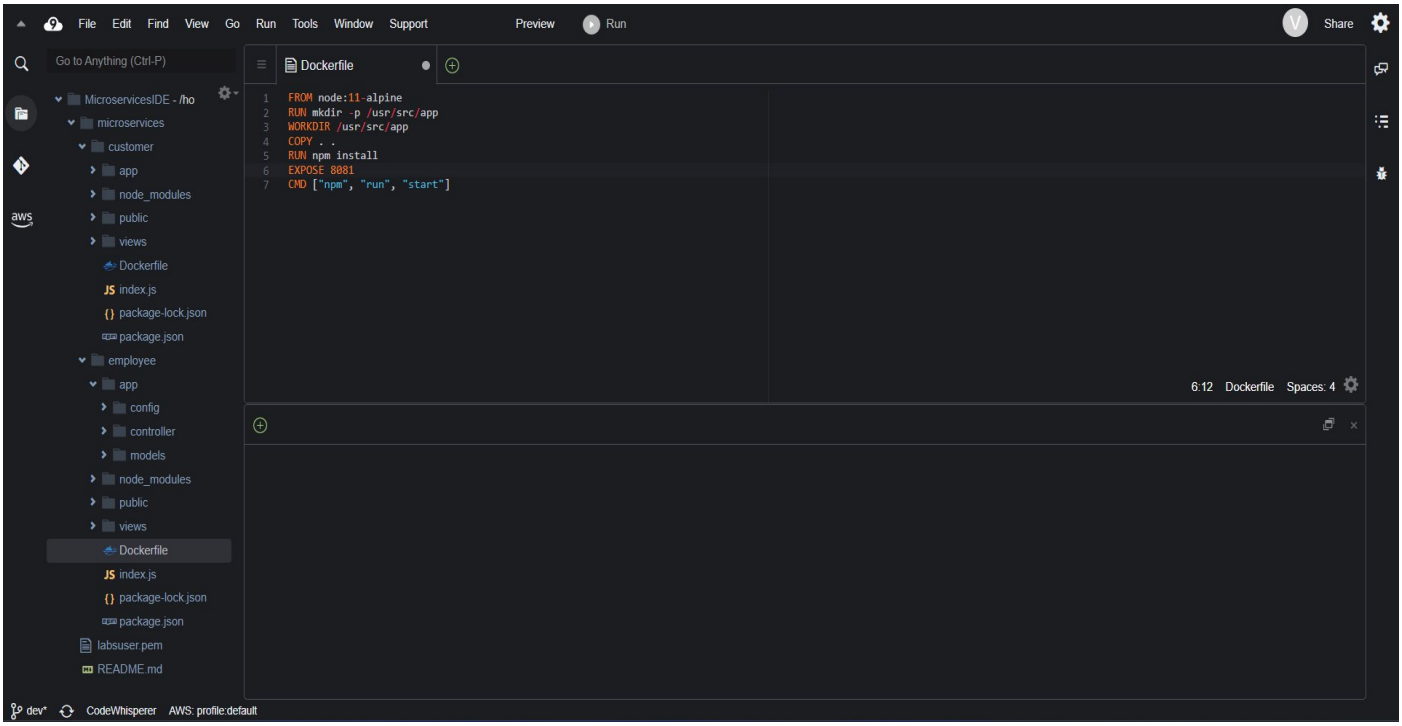
- These changes ensure that the employee microservice operates with a distinct path structure for administrative tasks, allowing employees to manage coffee suppliers effectively, while also providing navigation to the customer microservice. This setup is designed to work with the planned Application Load Balancer and support a clear separation of responsibilities within the microservices architecture.

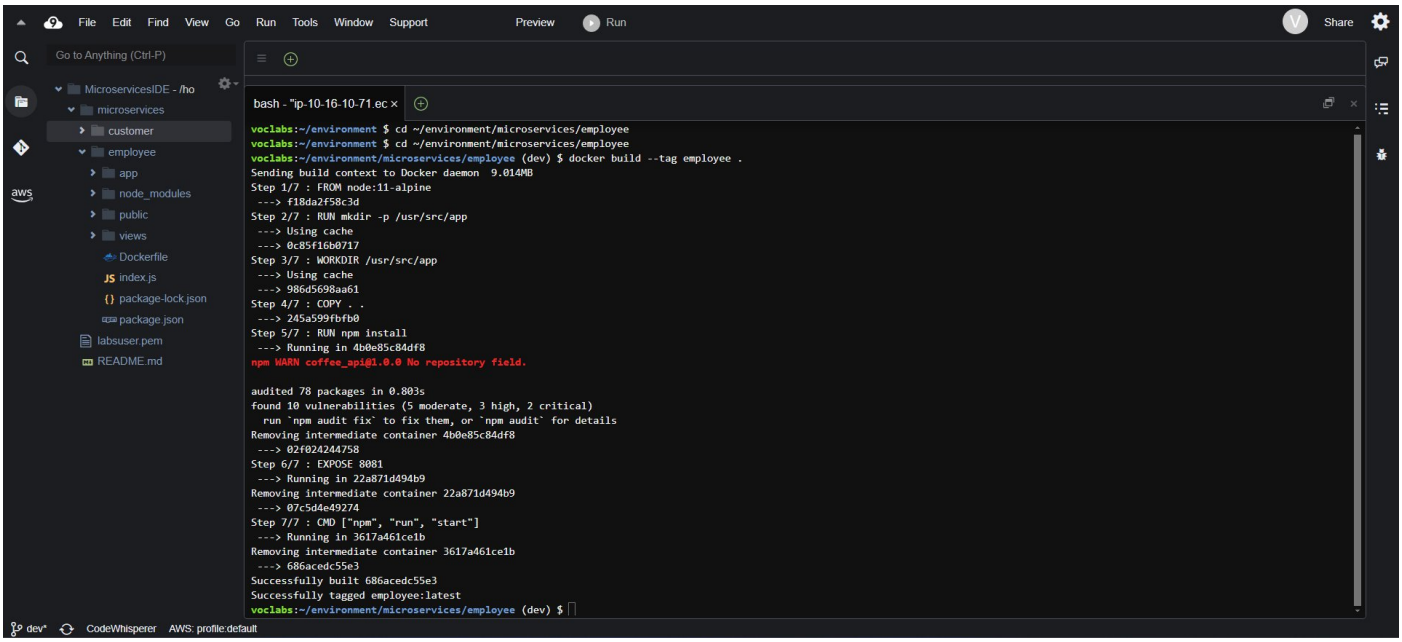


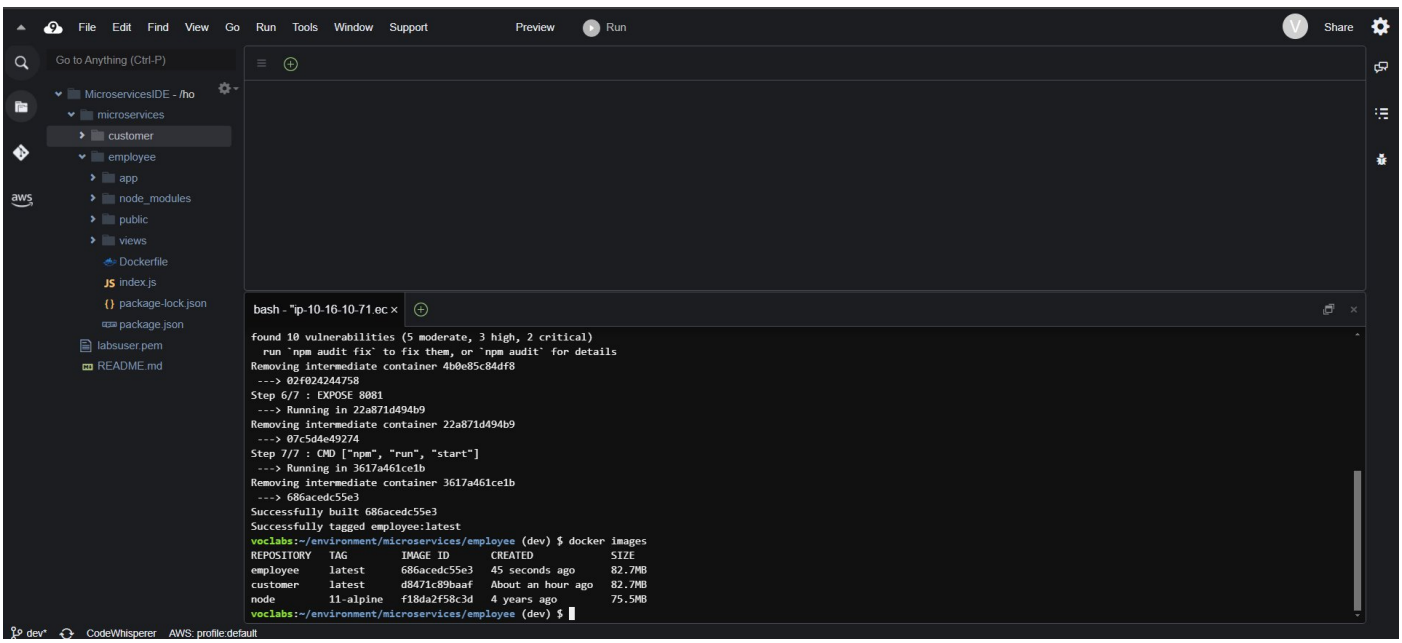


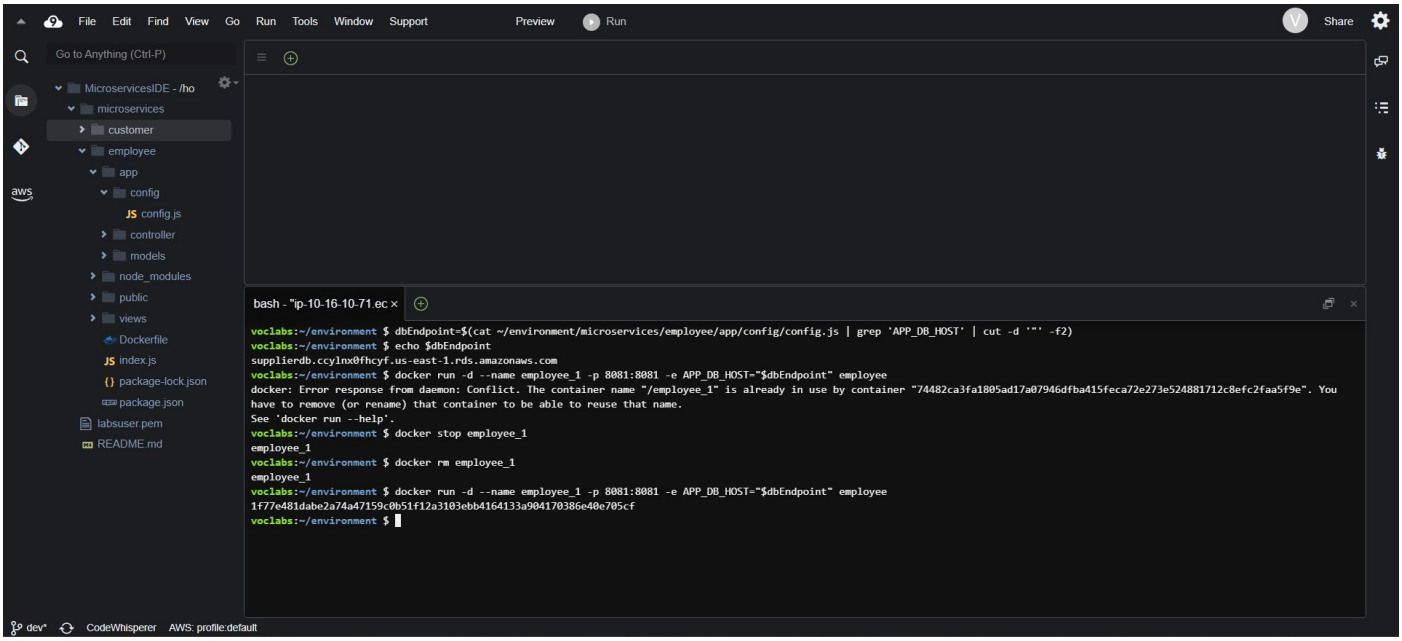
- Task 4.5: To create a Docker image for the employee microservice and test its functionality, I first duplicated the Dockerfile from the customer microservice into the employee microservice directory. I then edited the copied Dockerfile to set the EXPOSE line to port 8081 to match the unique port assigned to this microservice.
- I built the Docker image for the employee microservice specifying "employee" as the tag by running the **docker build --tag employee .** command in the terminal from the employee directory. After building the Docker image, I launched a container named "employee_1" on port 8081, ensuring I passed the database endpoint using the -e parameter to establish connectivity with the RDS database. The command looked like this: **docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint"** employee.




File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /ho
microservices
customer
app
node_modules
public
views
Dockerfile
index.js
package-lock.json
package.json
employee
app
config
controller
models
node_modules
public
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
Dockerfile
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 8081
7 CMD ["npm", "run", "start"]
6:12 Dockerfile Spaces: 4
dev*
CodeWhisperer
AWS: profile:default


File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /ho
microservices
customer
employee
app
node_modules
public
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
bash - "ip-10-16-10-71.ec
voclabs:~/environment $ cd ~/environment/microservices/employee
voclabs:~/environment $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
Sending build context to Docker daemon 9.014MB
Step 1/7 : FROM node:11-alpine
---> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
---> Using cache
---> 0c85f16b0717
Step 3/7 : WORKDIR /usr/src/app
---> Using cache
---> 986d5698aa61
Step 4/7 : COPY . .
---> 245a599fbfb0
Step 5/7 : RUN npm install
---> Running in 4b0e85c84df8
npm WARN coffee_api@1.0.0 No repository field.
audited 78 packages in 0.803s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 4b0e85c84df8
---> 02f024244758
Step 6/7 : EXPOSE 8081
---> Running in 22a871d494b9
Removing intermediate container 22a871d494b9
---> 07c5d4e49274
Step 7/7 : CMD ["npm", "run", "start"]
---> Running in 3617a461ce1b
Removing intermediate container 3617a461ce1b
---> 686acedc55e3
Successfully built 686acedc55e3
Successfully tagged employee:latest
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)
MicroservicesIDE - /ho
microservices
customer
employee
app
node_modules
public
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
bash - "ip-10-16-10-71.ec
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 4b0e85c84df8
---> 02f024244758
Step 6/7 : EXPOSE 8081
---> Running in 22a871d494b9
Removing intermediate container 22a871d494b9
---> 07c5d4e49274
Step 7/7 : CMD ["npm", "run", "start"]
---> Running in 3617a461ce1b
Removing intermediate container 3617a461ce1b
---> 686acedc55e3
Successfully built 686acedc55e3
Successfully tagged employee:latest
voclabs:~/environment/microservices/employee (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
employee latest 686acedc55e3 45 seconds ago 82.7MB
customer latest d8471c89baaf About an hour ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment/microservices/employee (dev) $
dev*
CodeWhisperer
AWS: profile:default



- To verify that the microservice was running I opened the application in a new browser tab using the public IP address of the AWS Cloud9 instance, followed by :8081/admin/suppliers. I checked that the application loaded successfully and confirmed that it displayed the expected functionality for employees, including buttons to add new suppliers and edit existing ones.

- I tested adding a new supplier and verified that it appeared on the suppliers' page. Then, I tested editing an existing supplier to ensure the updated information was displayed correctly. Additionally, I tested deleting an existing supplier by choosing "Edit," then "Delete this supplier," to ensure the supplier no longer appeared on the suppliers' page after deletion.

Manage coffee suppliers

Administrator home

Suppliers list

Customer home

All suppliers

Name	Address	City	State	Email	Phone	
Supreeth Hassan Ravindra	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263	edit

Add a new supplier

Manage coffee suppliers

Administrator home

Suppliers list

Customer home

All suppliers

Name	Address	City	State	Email	Phone	
Supreeth Hassan Ravindra	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263	edit
Yash Deshpande	1 World Trade Centre, Manhattan	New York City	New york	yashd@gmail.com	1234567890	edit

Add a new supplier

Manage coffee suppliers

Administrator home

Suppliers list

Customer home

All suppliers

Name	Address	City	State	Email	Phone	
Supreeth Hassan Ravindra	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263	edit
Yash Deshpande	100 Hopkins Ave	New York City	New york	yashd@gmail.com	1234567890	edit

Add a new supplier

Manage coffee suppliers

Administrator home

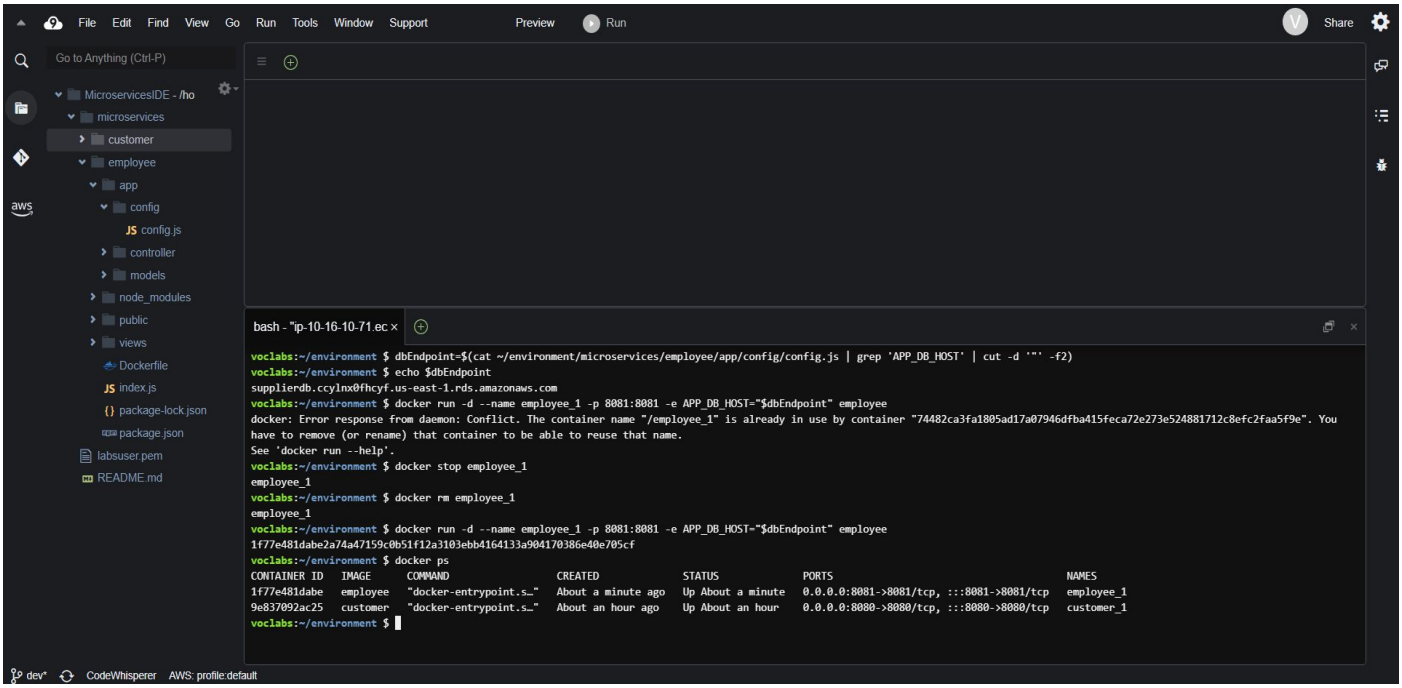
Suppliers list

Customer home

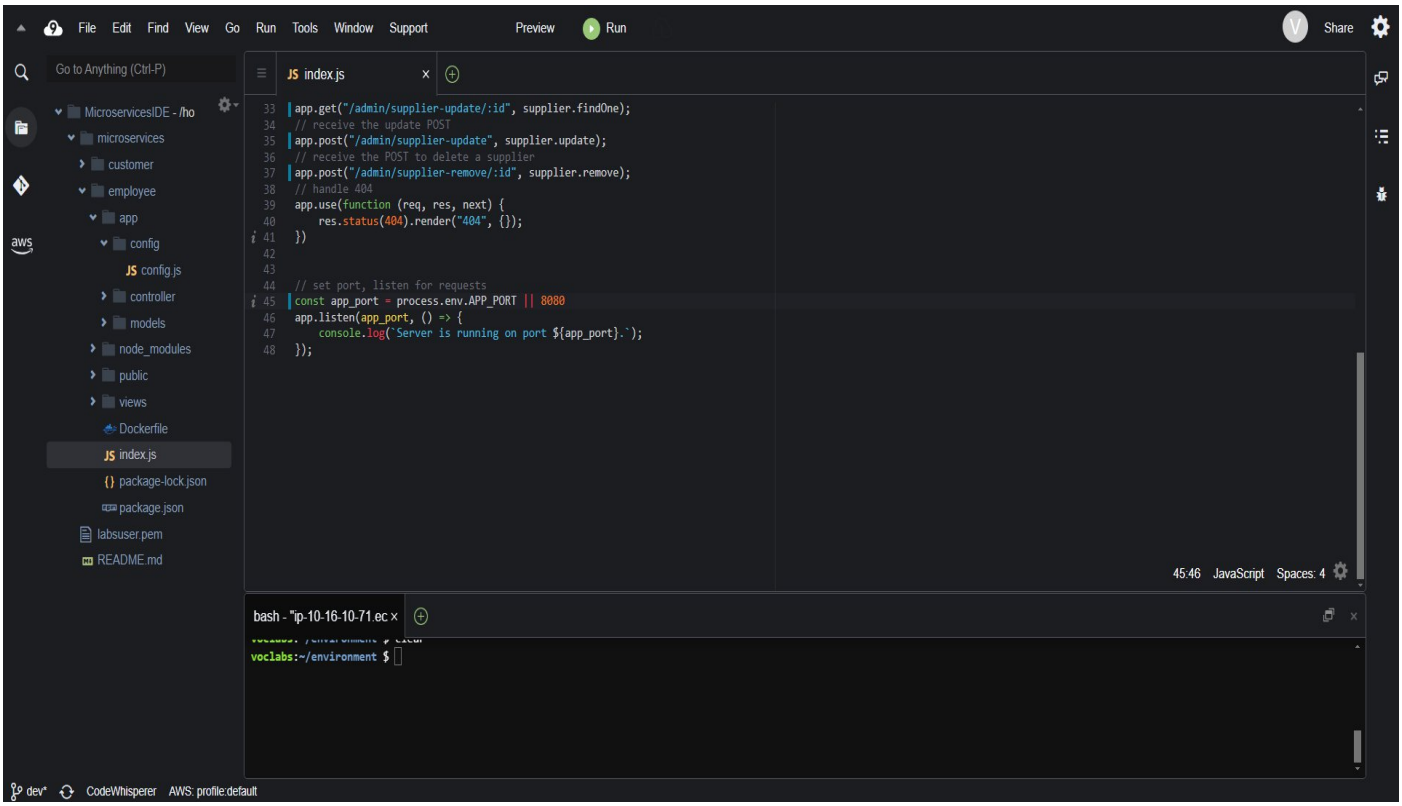
All suppliers

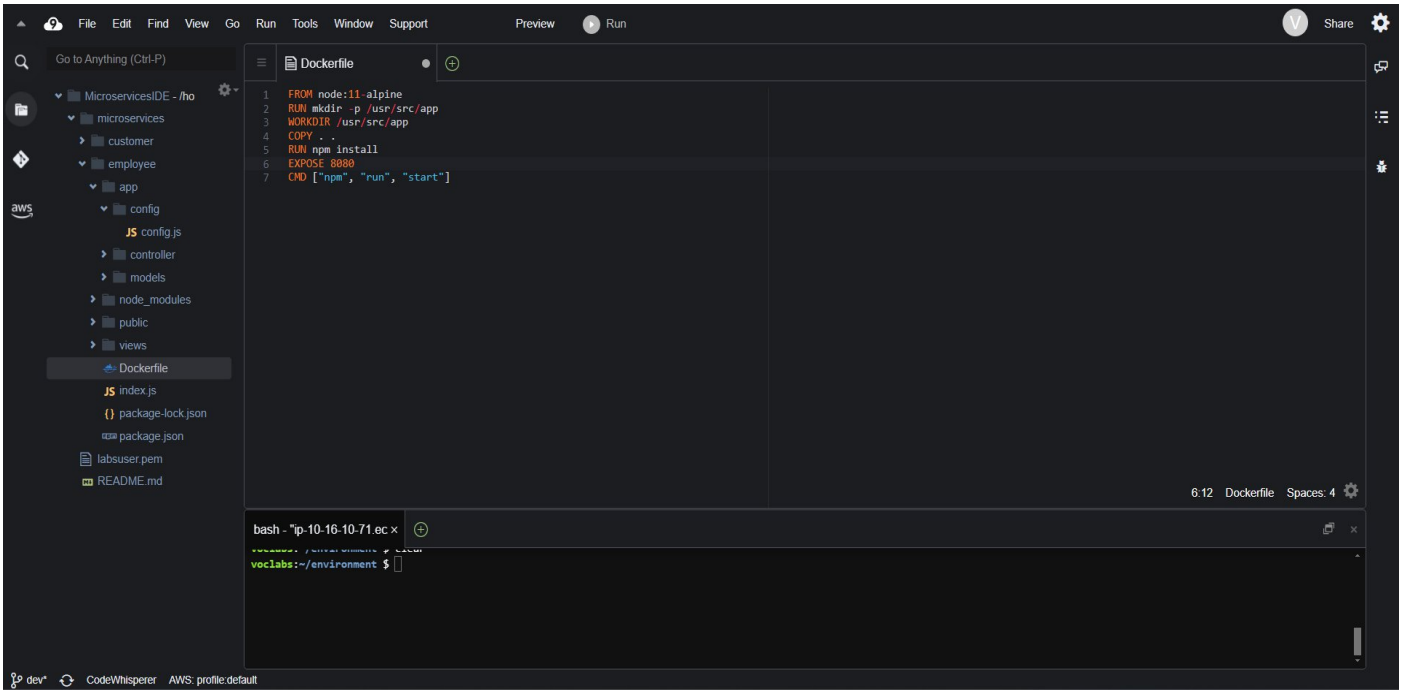
Name	Address	City	State	Email	Phone	
Supreeth Hassan Ravindra	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263	edit

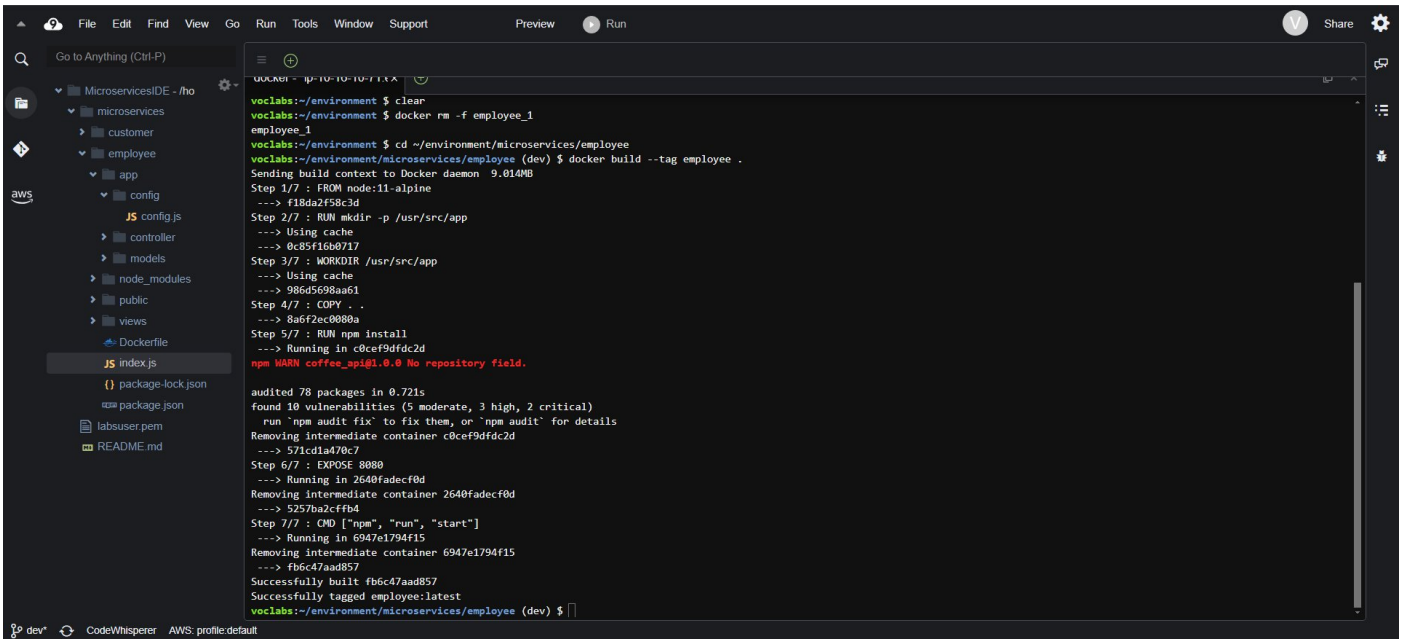
Add a new supplier

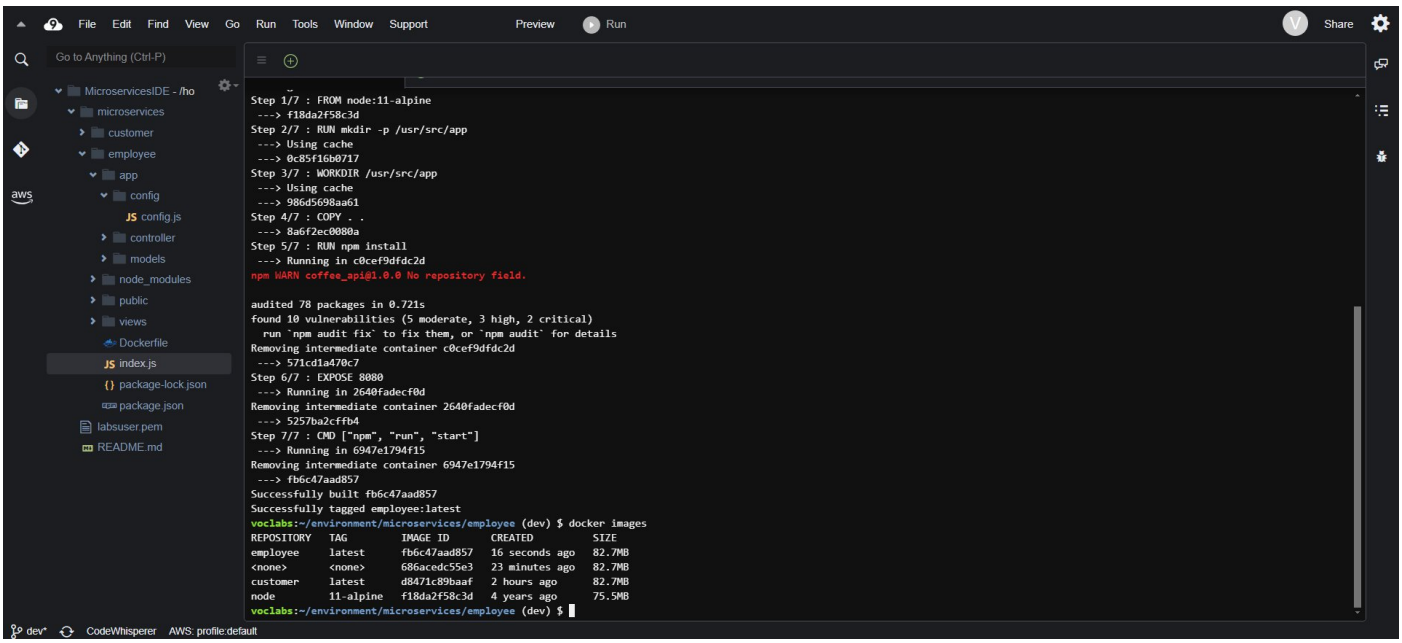


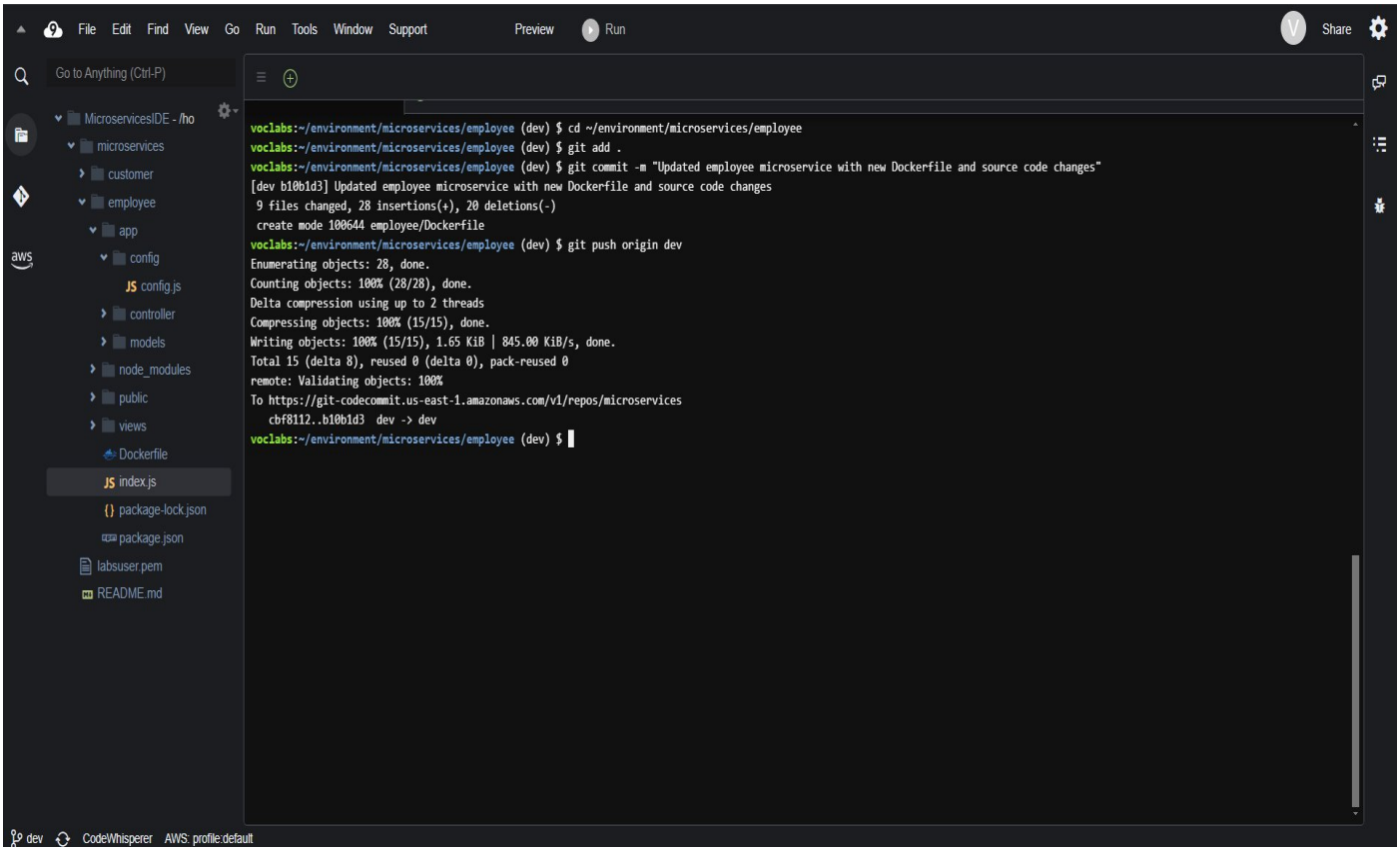
- Task 4.6: To adjust the port number for the employee microservice to prepare for deployment on Amazon ECS I started by editing two files: employee/index.js and employee/Dockerfile, changing the port from 8081 to 8080. This adjustment aligns with the desired port for ECS deployment.
- After updating the files, I rebuilt the Docker image for the employee microservice to incorporate the port change. I first stopped and deleted the existing container using **docker rm -f employee_1.** Then, ensuring my terminal was in the employee directory I ran **docker build --tag employee .** to rebuild the Docker image with the new port setting.
- This ensured the Docker image for the employee microservice was updated with the correct port, preparing it for deployment on Amazon ECS where it will run on port 8080.



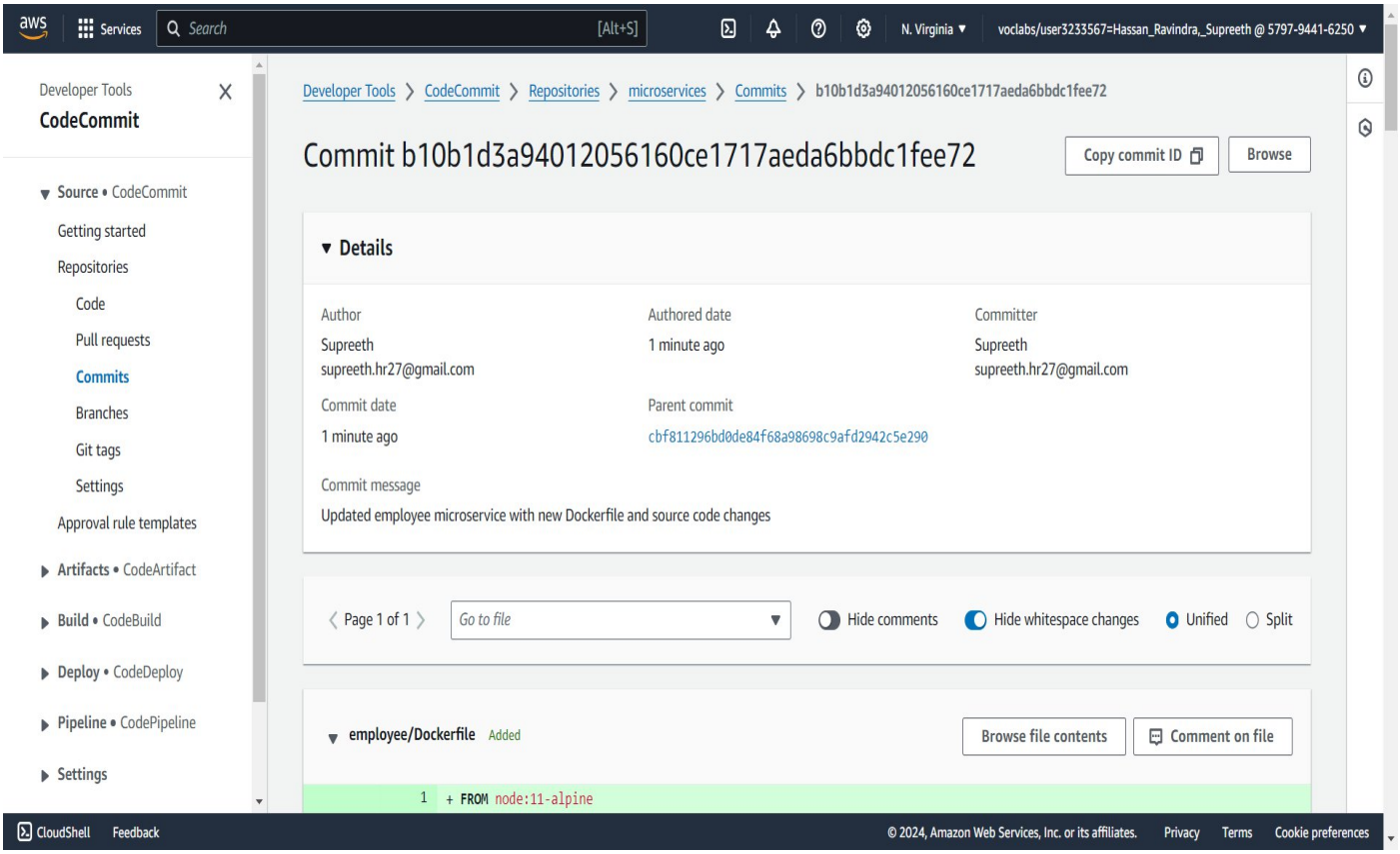

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /ho
microservices
customer
employee
app
config
config.js
controller
models
node_modules
public
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
Dockerfile
1 FROM 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"]
6:12 Dockerfile Spaces: 4
bash - "ip-10-16-10-71.ec
voclabs:~/environment $
dev* CodeWhisperer AWS: profile:default

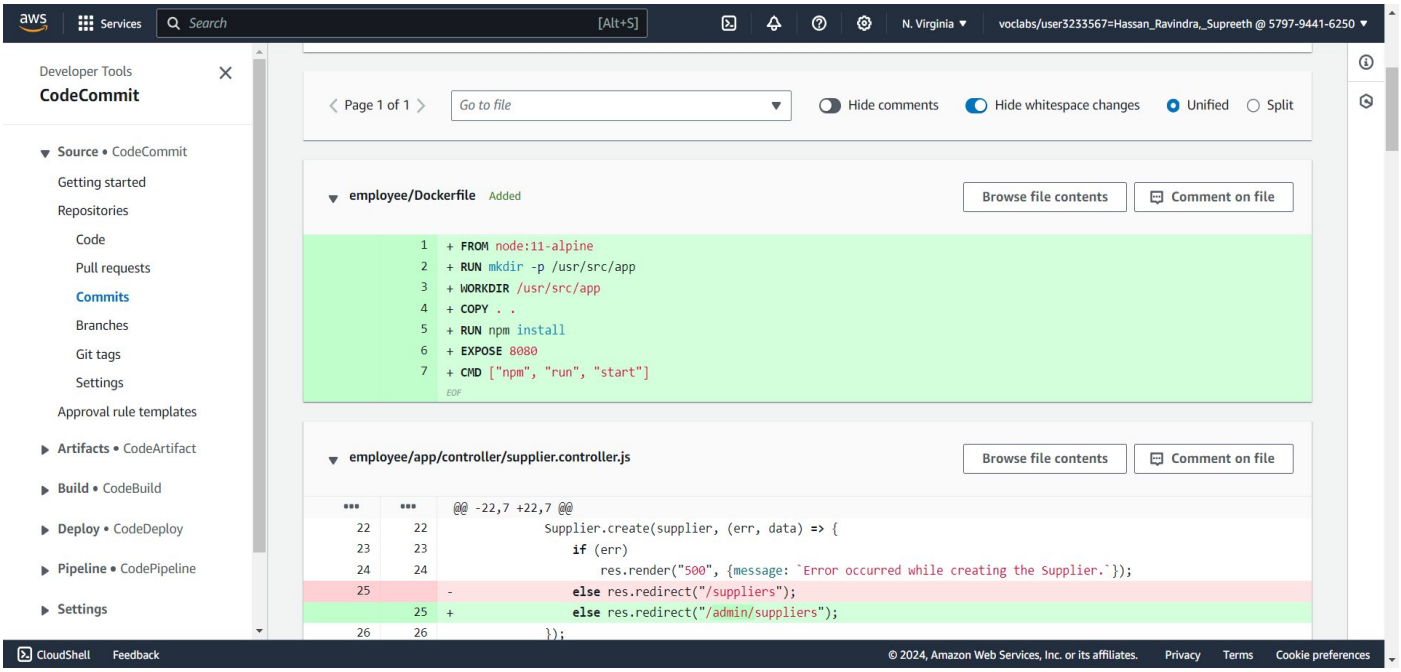

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /ho
voclabs:~/environment $ clear
voclabs:~/environment $ docker rm -f employee_1
employee_1
voclabs:~/environment $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
Sending build context to Docker daemon 9.014MB
Step 1/7 : FROM node:11-alpine
---> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
---> Using cache
---> 0c85f16b0717
Step 3/7 : WORKDIR /usr/src/app
---> Using cache
---> 986d5698aa61
Step 4/7 : COPY . .
---> 8a6f2ec0080a
Step 5/7 : RUN npm install
---> Running in c0cef9dfdc2d
npm WARN coffee_api@1.0.0 No repository field.
audited 78 packages in 0.721s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container c0cef9dfdc2d
---> 571cd1a470c7
Step 6/7 : EXPOSE 8080
---> Running in 2640fadecf0d
Removing intermediate container 2640fadecf0d
---> 5257ba2cffb4
Step 7/7 : CMD ["npm", "run", "start"]
---> Running in 6947e1794f15
Removing intermediate container 6947e1794f15
---> fb6c47aad857
Successfully built fb6c47aad857
Successfully tagged employee:latest
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)
MicroservicesIDE - /ho
Step 1/7 : FROM node:11-alpine
---> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
---> Using cache
---> 0c85f16b0717
Step 3/7 : WORKDIR /usr/src/app
---> Using cache
---> 986d5698aa61
Step 4/7 : COPY . .
---> 8a6f2ec0080a
Step 5/7 : RUN npm install
---> Running in c0cef9dfdc2d
npm WARN coffee_api@1.0.0 No repository field.
audited 78 packages in 0.721s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container c0cef9dfdc2d
---> 571cd1a470c7
Step 6/7 : EXPOSE 8080
---> Running in 2640fadecf0d
Removing intermediate container 2640fadecf0d
---> 5257ba2cffb4
Step 7/7 : CMD ["npm", "run", "start"]
---> Running in 6947e1794f15
Removing intermediate container 6947e1794f15
---> fb6c47aad857
Successfully built fb6c47aad857
Successfully tagged employee:latest
voclabs:~/environment/microservices/employee (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
employee latest fb6c47aad857 16 seconds ago 82.7MB
<none> <none> 686acedc55e3 23 minutes ago 82.7MB
customer latest d8471c89baaf 2 hours ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment/microservices/employee (dev) $
dev* CodeWhisperer AWS: profile:default



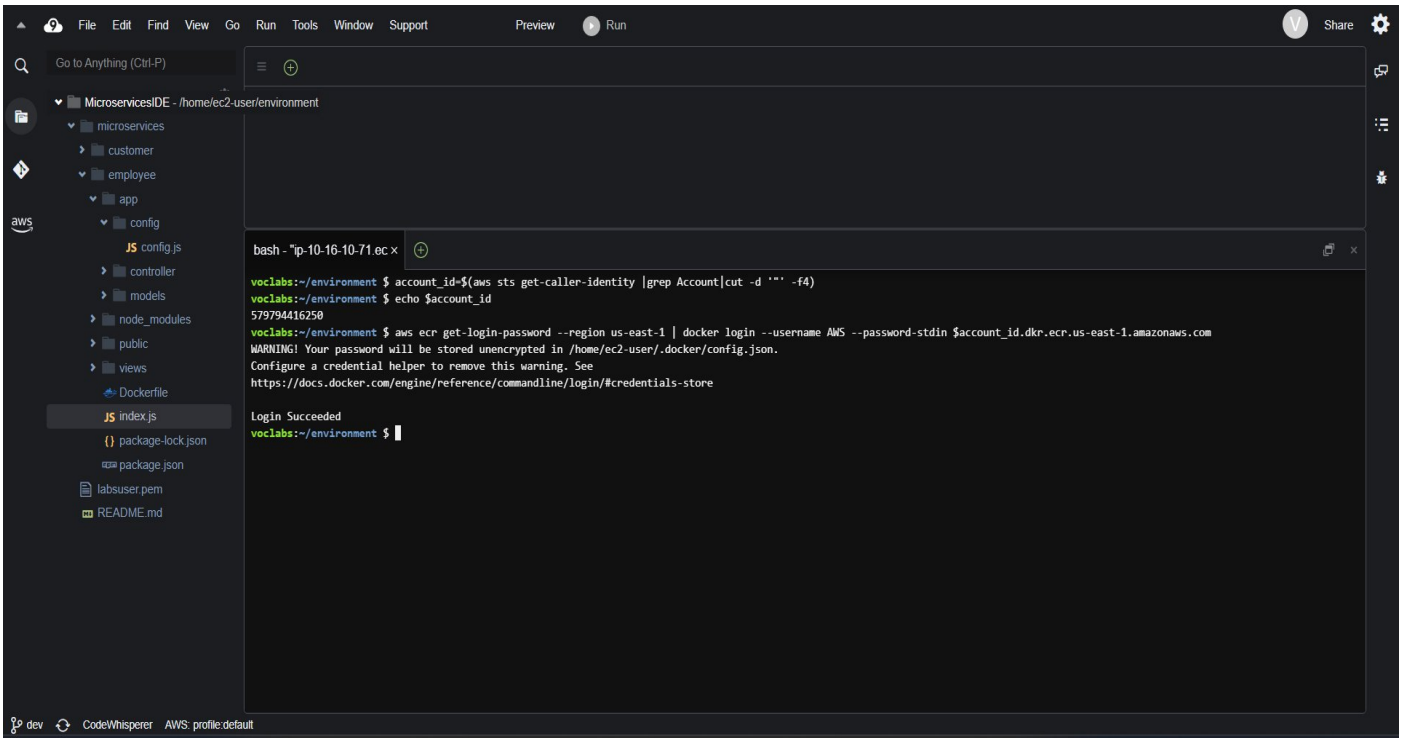
Task 4.7: Checking the code changes into CodeCommit.





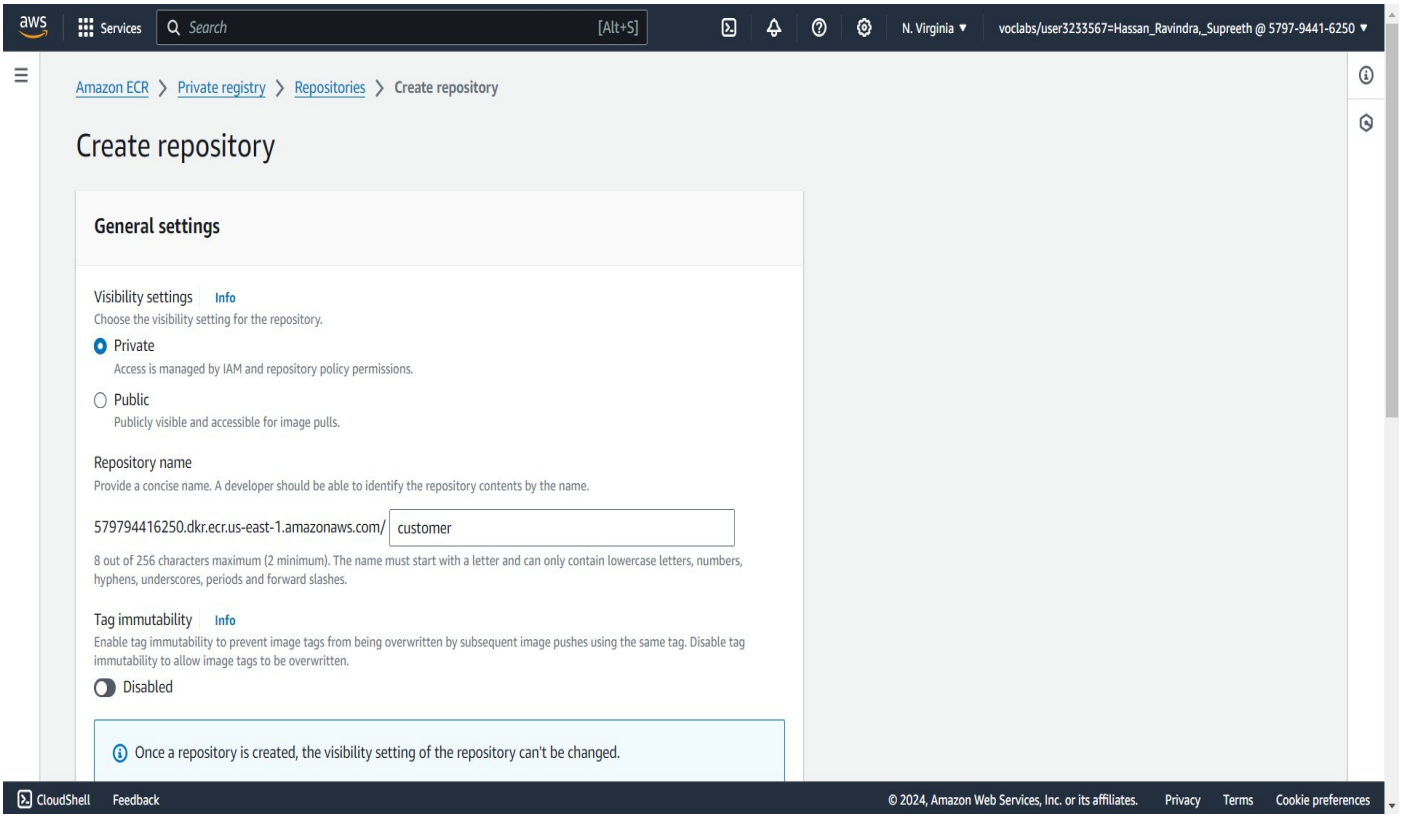
PHASE-5: Creating ECR repositories, an ECS cluster, task definitions, and AppSpec files

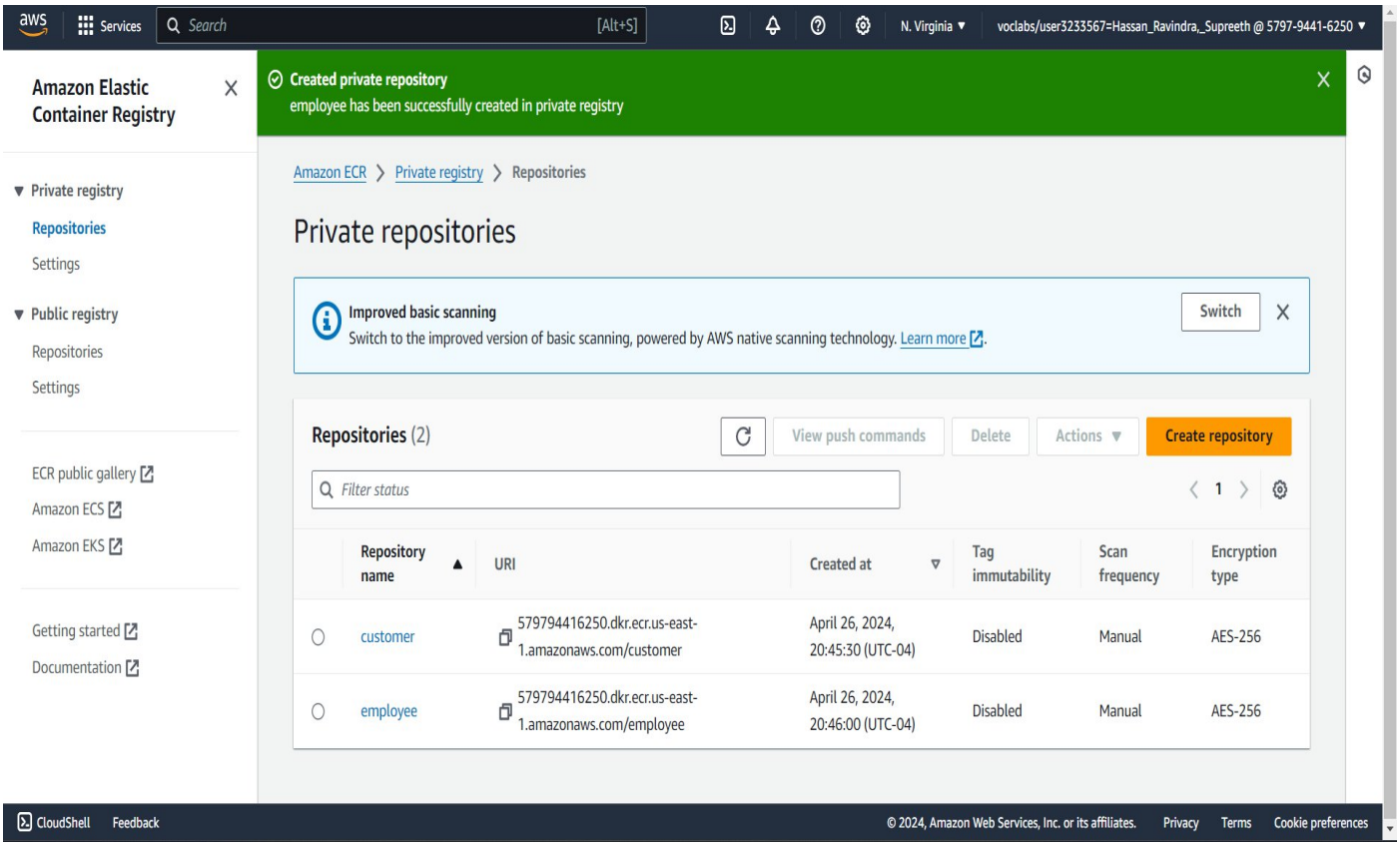
- Task 5.1: To upload the Docker images for the customer and employee microservices to Amazon ECR I started by authorizing my Docker client to connect to Amazon ECR by retrieving my AWS account ID and using it to log in to the ECR service. After confirming a successful login I created two separate private ECR repositories, naming them "customer" and "employee."

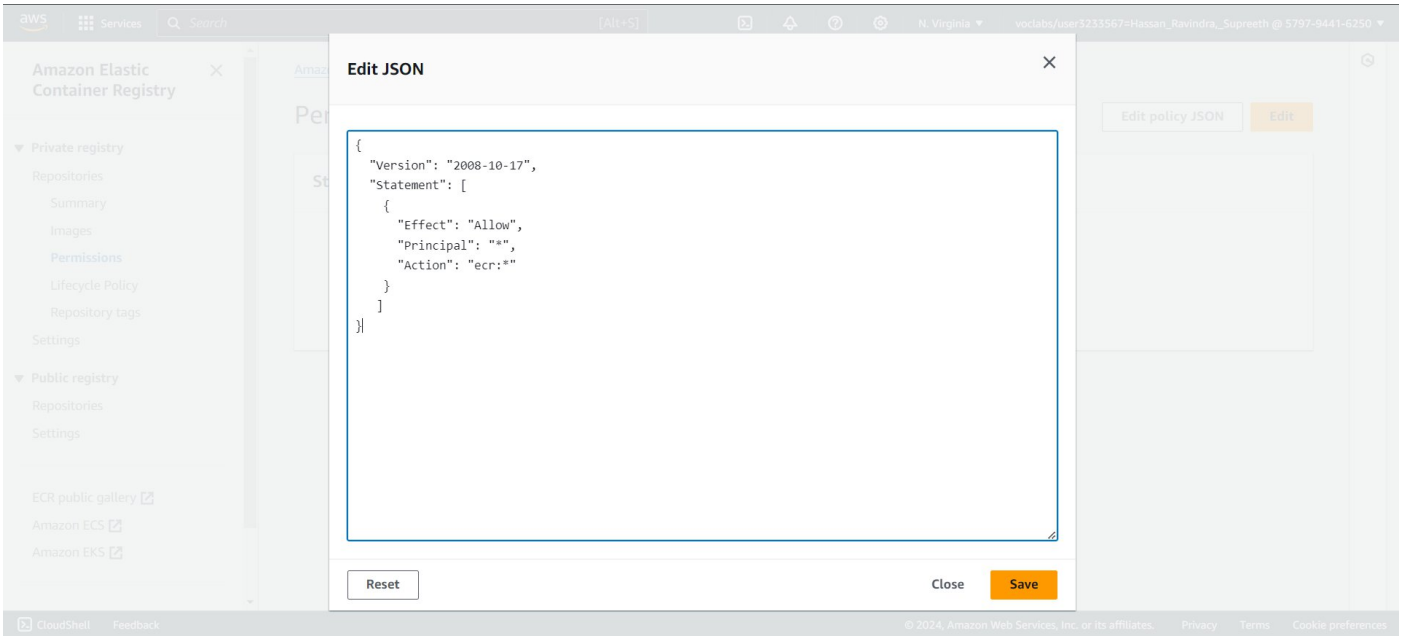


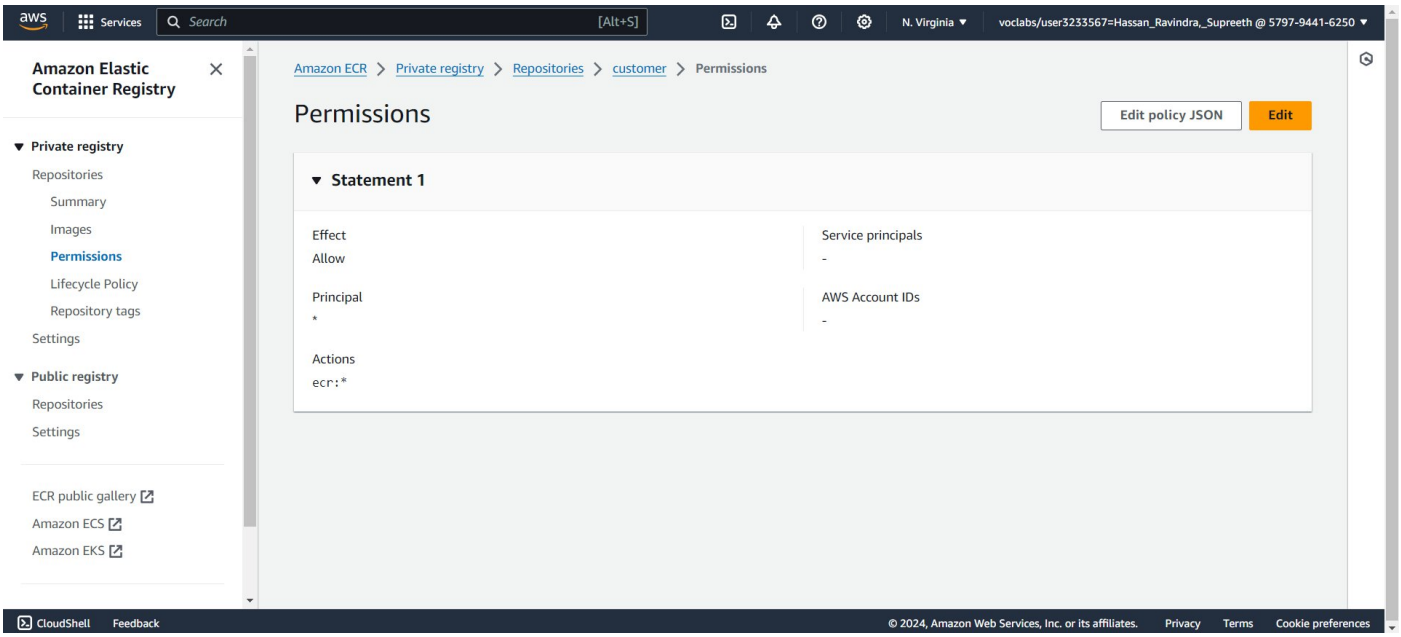
- I set permissions on both created repositories to allow all actions on the ECR service by updating the JSON policy.

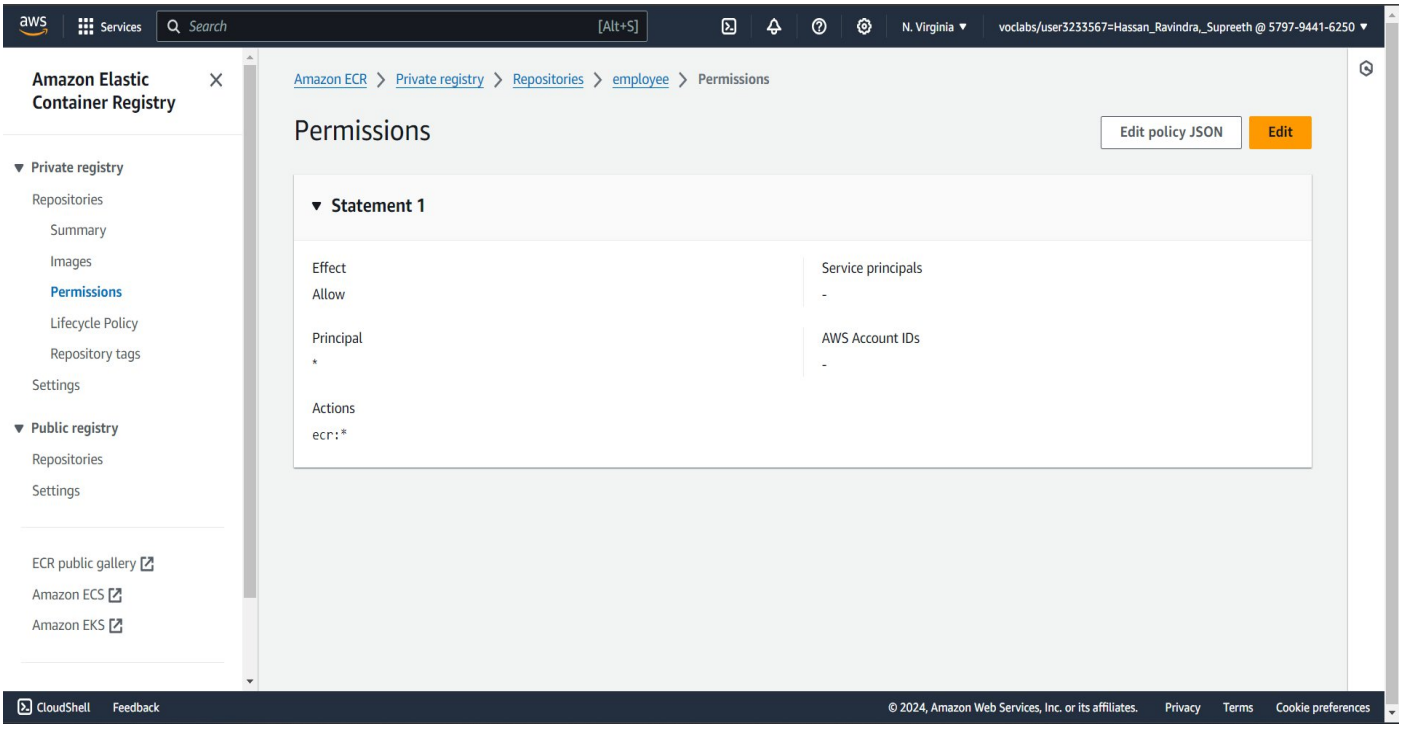
- I then tagged the Docker images with my AWS account ID to maintain a unique registry identifier. I tagged the customer image with **docker tag customer:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest** and the employee image with **docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest**. These tags indicated the destination repositories for the images.





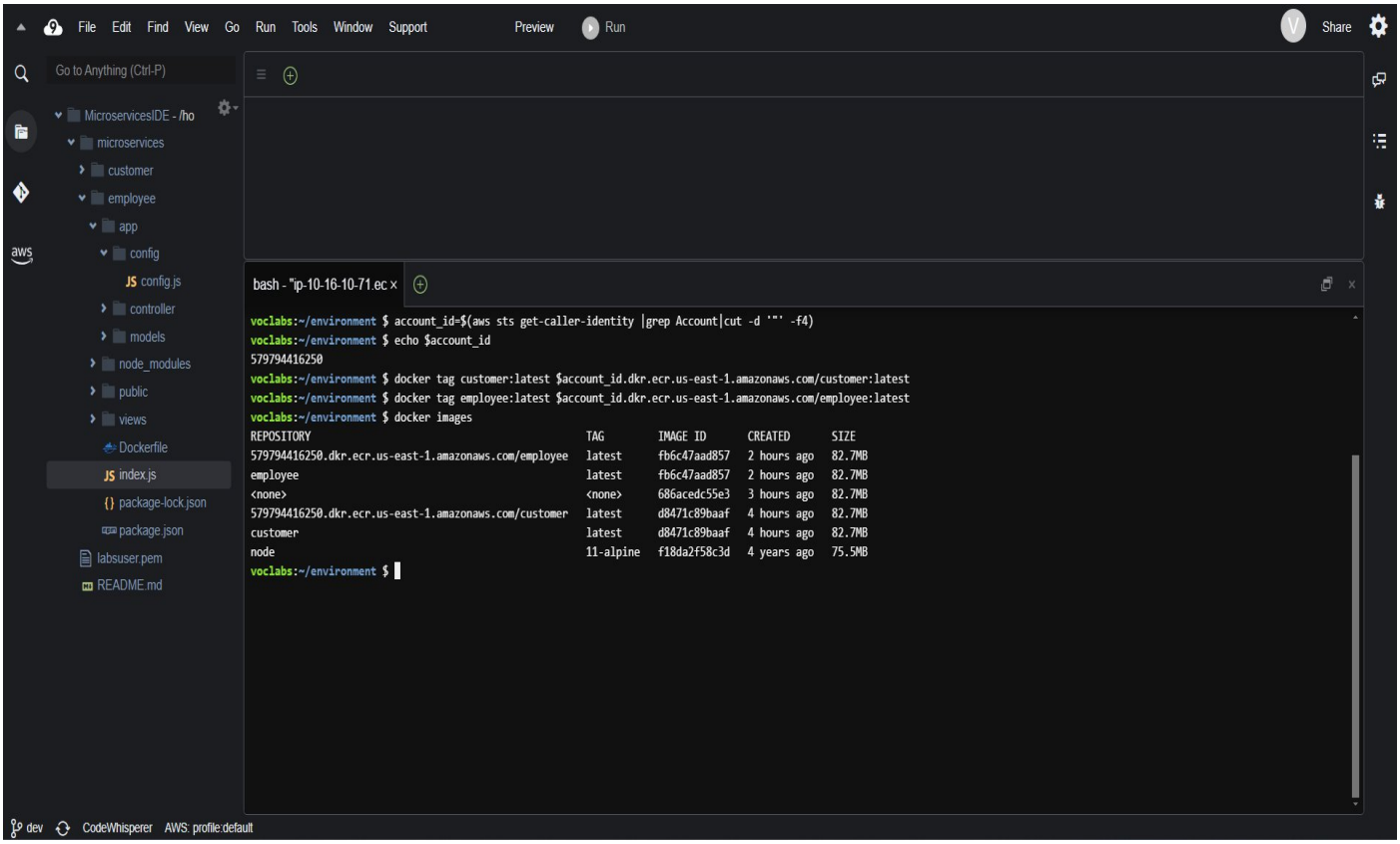

Edit JSON
{
"Version": "2008-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": "*",
"Action": "ecr:*"
}
]
}
Reset
Close
Save

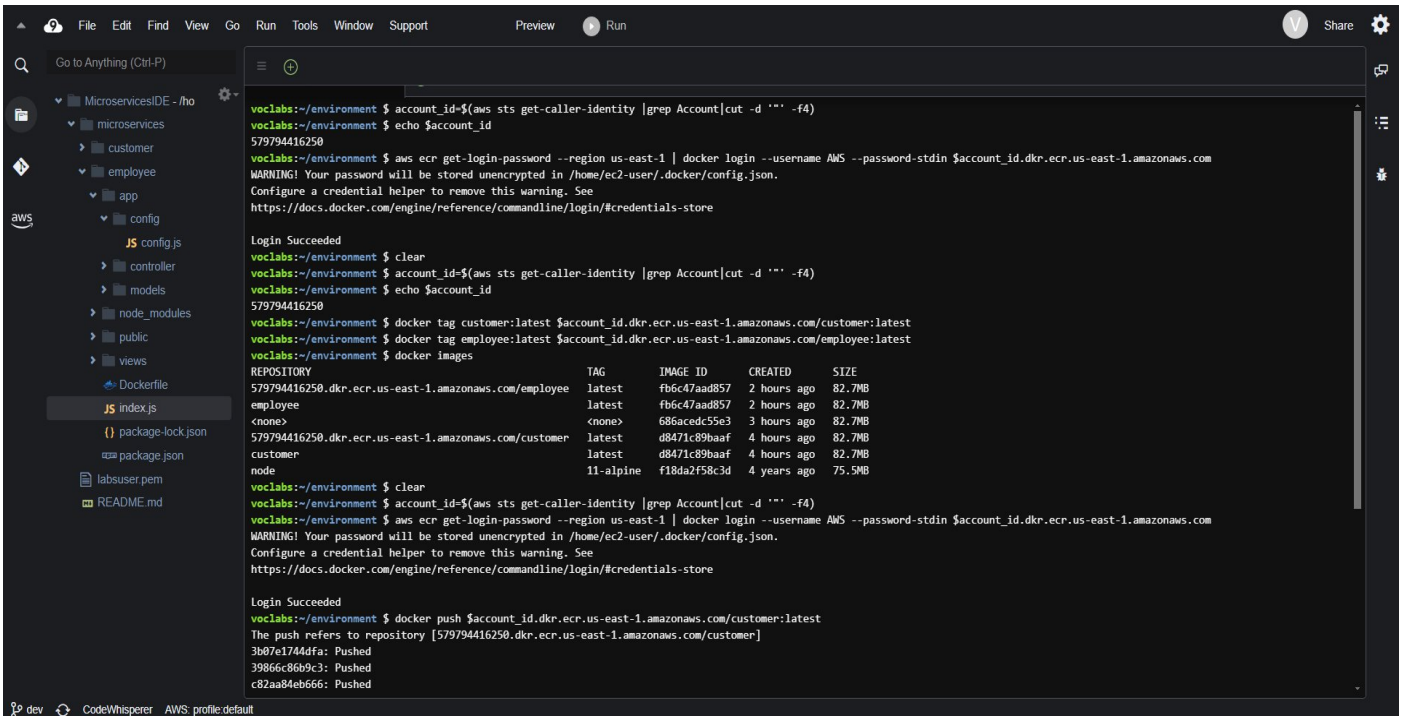

Amazon Elastic Container Registry
Amazon ECR > Private registry > Repositories > customer > Permissions
Permissions
Edit policy JSON
Edit
Statement 1
Effect
Allow
Principal
*
Actions
ecr:*
Service principals
-
AWS Account IDs
-
Private registry
Repositories
Summary
Images
Permissions
Lifecycle Policy
Repository tags
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

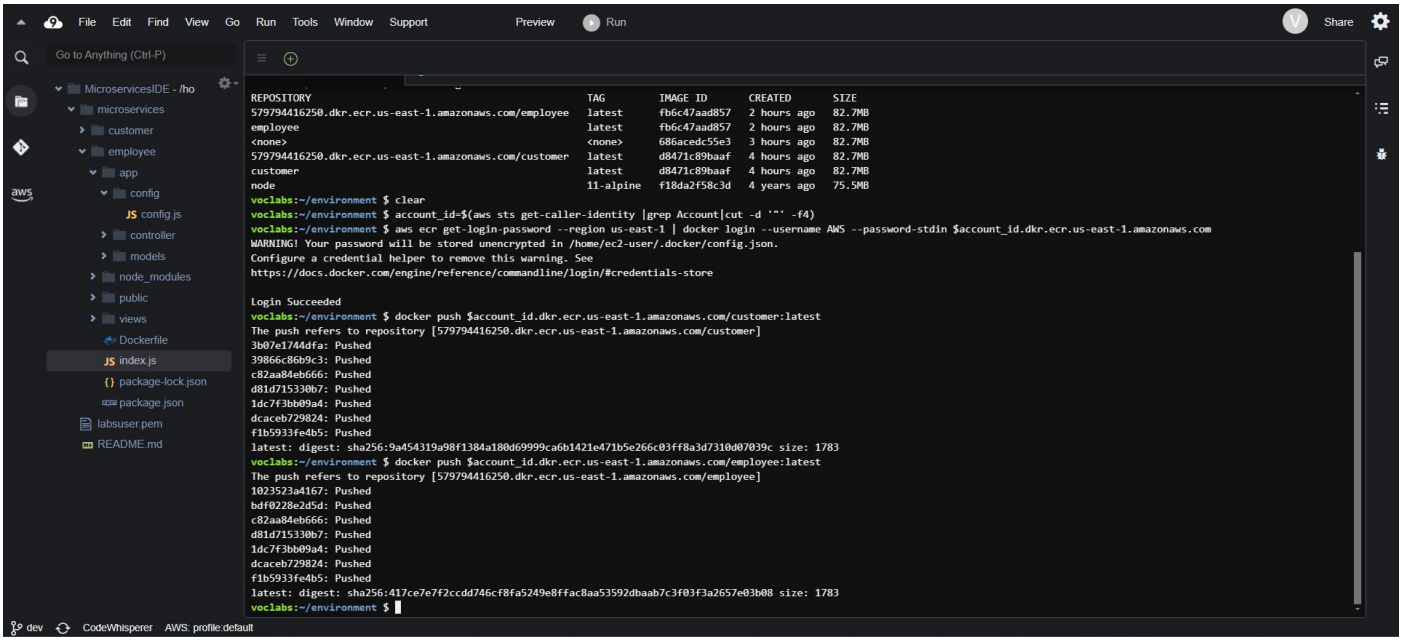

Amazon Elastic Container Registry
Amazon ECR > Private registry > Repositories > employee > Permissions
Permissions
Edit policy JSON
Edit
Statement 1
Effect
Allow
Principal
*
Actions
ecr:*
Service principals
-
AWS Account IDs
-
Private registry
Repositories
Summary
Images
Permissions
Lifecycle Policy
Repository tags
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

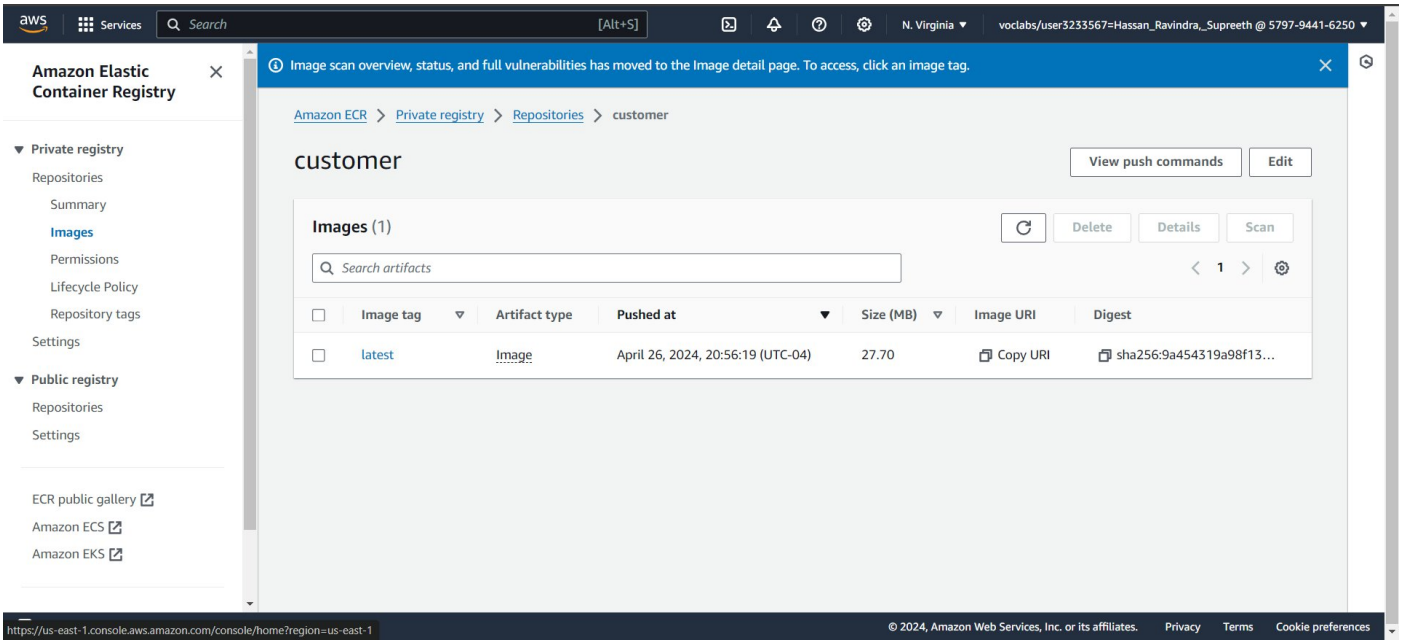
- After applying the tags I ran the Docker command to list the images to ensure the tags were correctly applied and the image names now included the remote repository names.

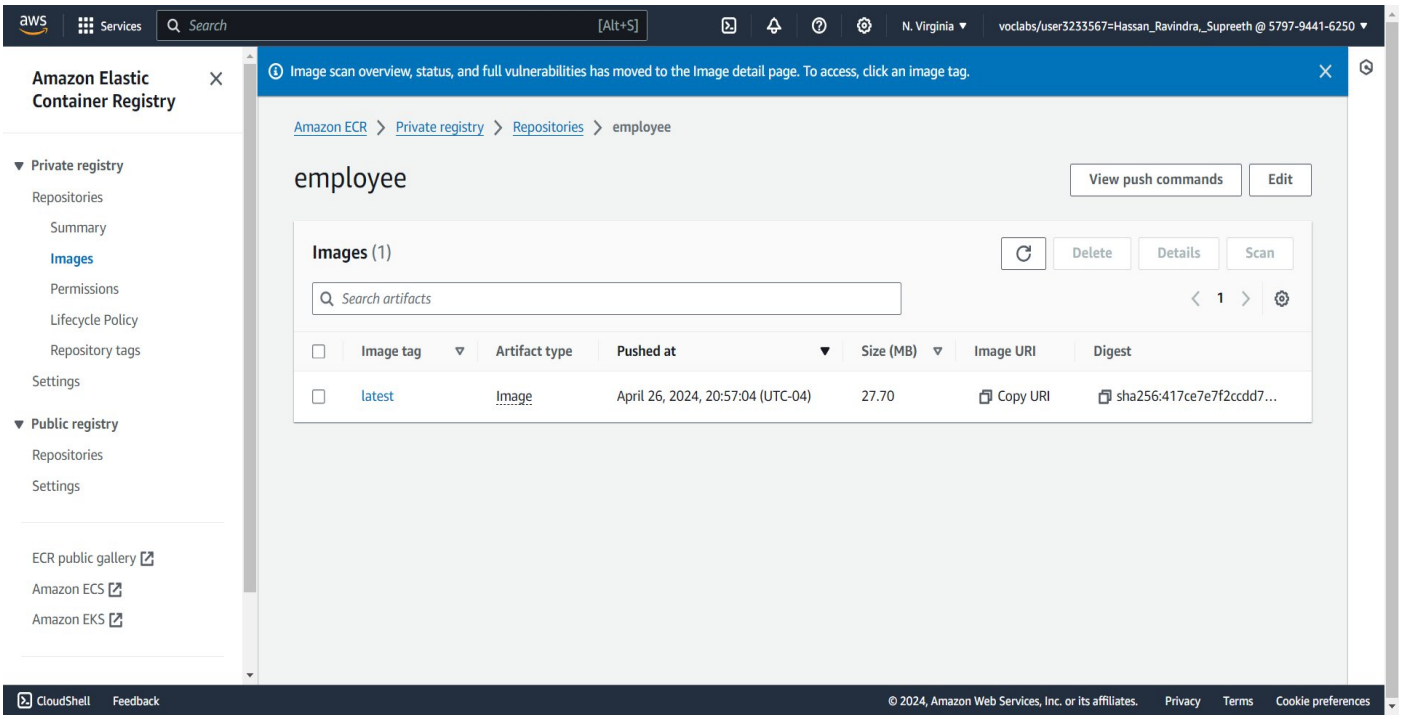
- With the images tagged I pushed them to their respective Amazon ECR repositories. Using the correct Docker command with the proper repository path I pushed the customer image and the employee image to Amazon ECR, confirming they were successfully stored with the "latest" label applied. This process ensured the Docker images for the customer and employee microservices were uploaded and ready for deployment in Amazon ECR.





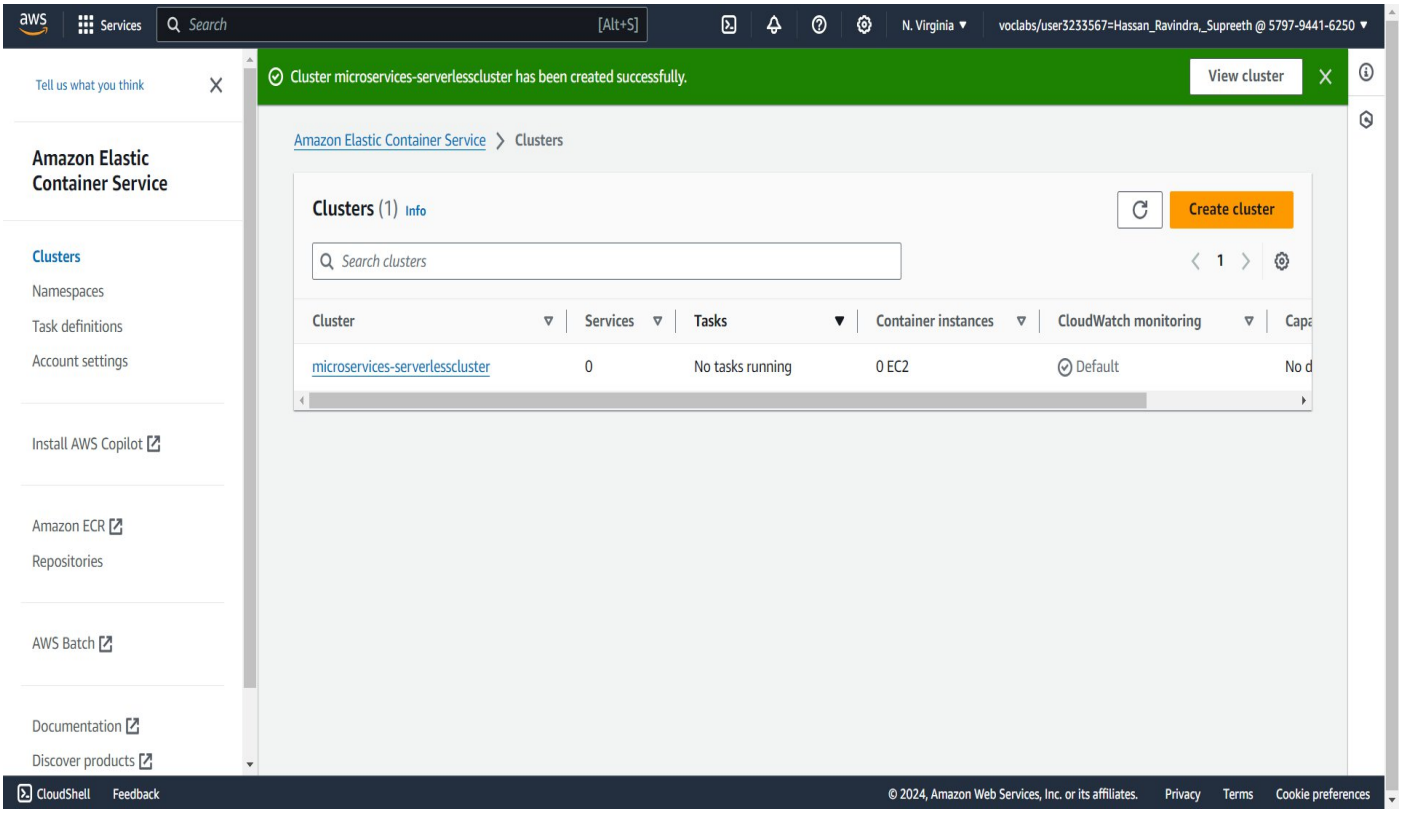

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE - /ho
microservices
customer
employee
app
config
config.js
controller
models
node_modules
public
views
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
REPOSITORY TAG IMAGE ID CREATED SIZE
579794416250.dkr.ecr.us-east-1.amazonaws.com/employee latest fb6c47aad857 2 hours ago 82.7MB
employee latest fb6c47aad857 2 hours ago 82.7MB
<none> <none> 686acedc55e3 3 hours ago 82.7MB
579794416250.dkr.ecr.us-east-1.amazonaws.com/customer latest d8471c89baaf 4 hours ago 82.7MB
customer latest d8471c89baaf 4 hours ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment $ clear
voclabs:~/environment $ account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)
voclabs:~/environment $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
voclabs:~/environment $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
The push refers to repository [579794416250.dkr.ecr.us-east-1.amazonaws.com/customer]
3b07e1744dfa: Pushed
39866c86b9c3: Pushed
c82aa84eb666: Pushed
d81d715330b7: Pushed
1dc7f3bb09a4: Pushed
dcaceb729824: Pushed
f1b5933fe4b5: Pushed
latest: digest: sha256:9a454319a98f1384a180d69999ca6b1421e471b5e266c03ff8a3d7310d07039c size: 1783
voclabs:~/environment $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
The push refers to repository [579794416250.dkr.ecr.us-east-1.amazonaws.com/employee]
1023523a4167: Pushed
bdf0228e2d5d: Pushed
c82aa84eb666: Pushed
d81d715330b7: Pushed
1dc7f3bb09a4: Pushed
dcaceb729824: Pushed
f1b5933fe4b5: Pushed
latest: digest: sha256:417ce7e7f2ccdd746cf8fa5249e8ffac8aa53592dbaab7c3f03f3a2657e03b08 size: 1783
voclabs:~/environment $
dev CodeWhisperer AWS: profile:default

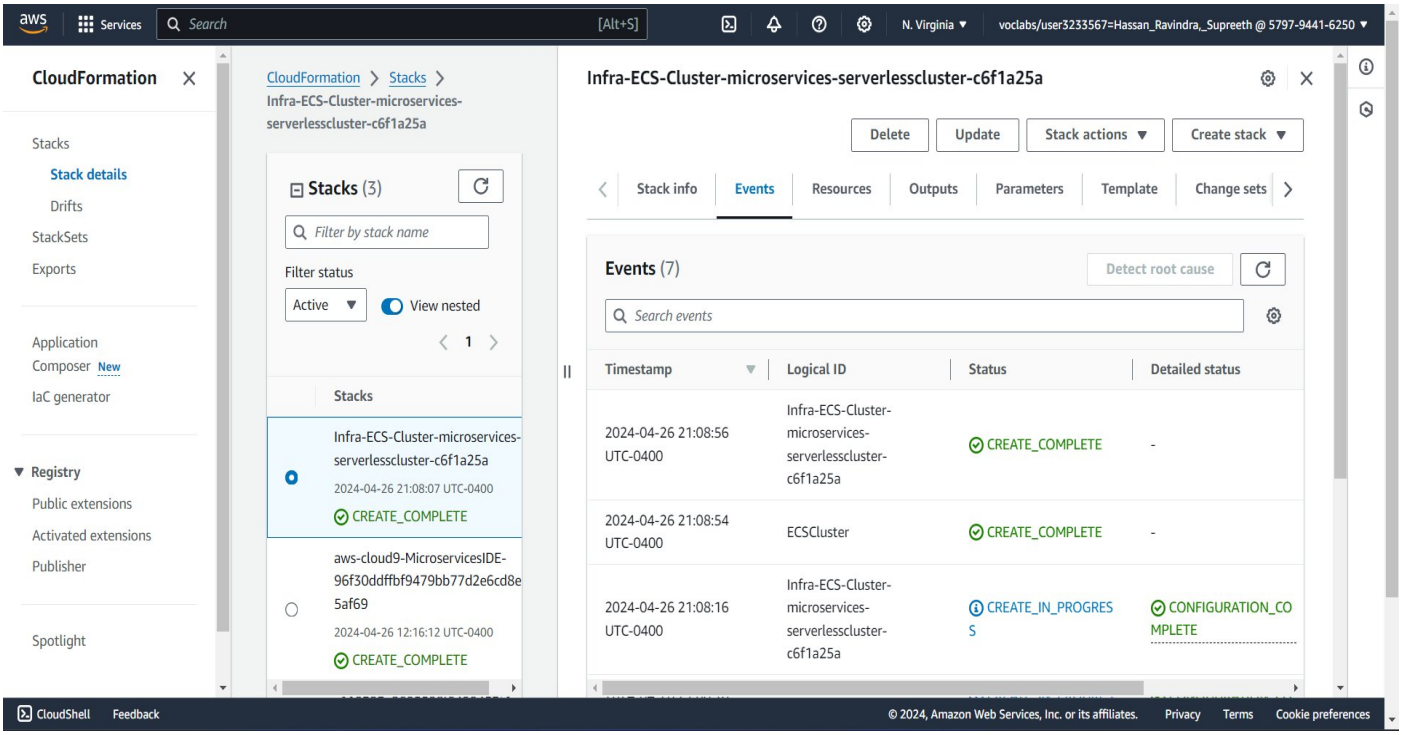

aws Services Search [Alt+S] N. Virginia voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Amazon Elastic Container Registry
Private registry
Repositories
Summary
Images
Permissions
Lifecycle Policy
Repository tags
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.
Amazon ECR > Private registry > Repositories > customer
customer
View push commands
Edit
Images (1)
Delete
Details
Scan
Search artifacts
Image tag
Artifact type
Pushed at
Size (MB)
Image URI
Digest
latest
Image
April 26, 2024, 20:56:19 (UTC-04)
27.70
Copy URI
sha256:9a454319a98f13...
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

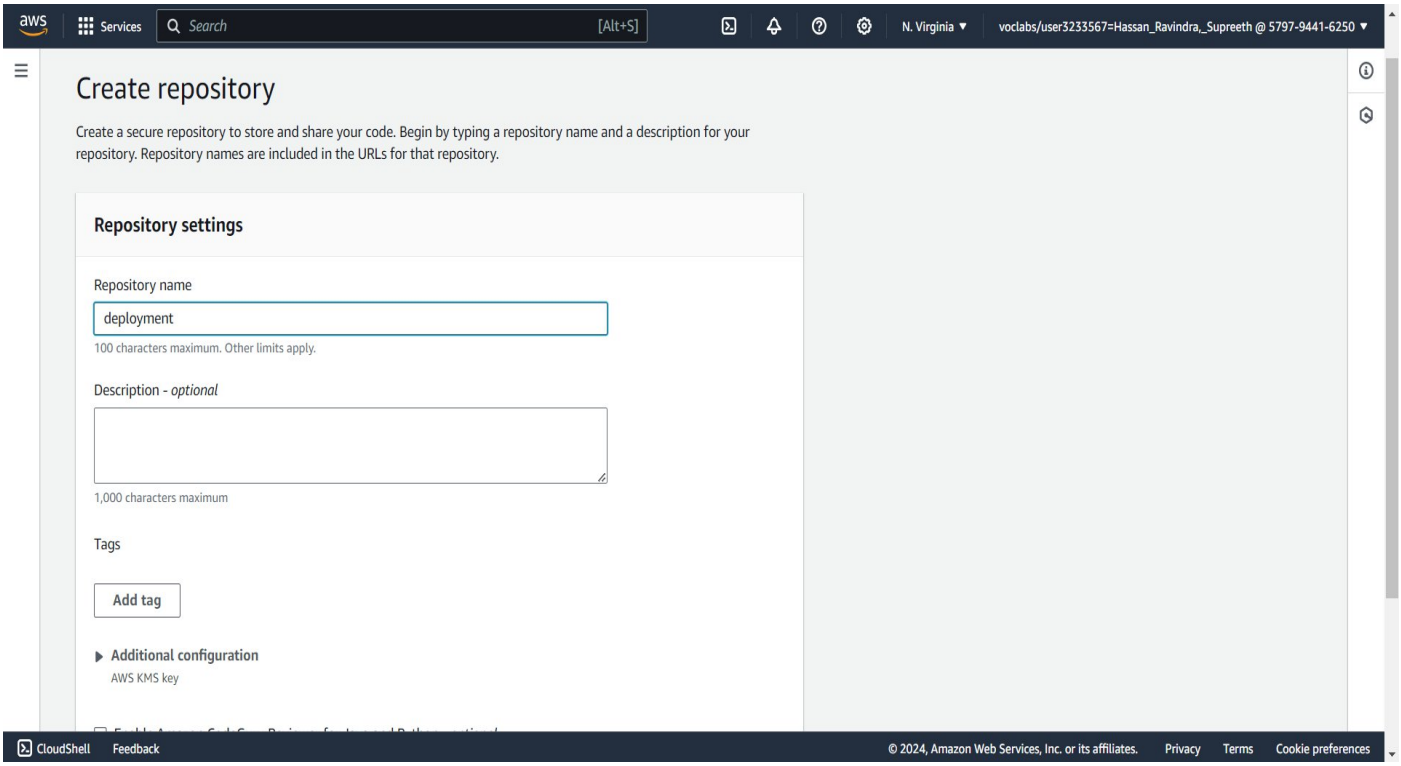

aws Services Search [Alt+S] N. Virginia voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Amazon Elastic Container Registry
Private registry
Repositories
Summary
Images
Permissions
Lifecycle Policy
Repository tags
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.
Amazon ECR > Private registry > Repositories > employee
employee
View push commands
Edit
Images (1)
Delete
Details
Scan
Search artifacts
Image tag
Artifact type
Pushed at
Size (MB)
Image URI
Digest
latest
Image
April 26, 2024, 20:57:04 (UTC-04)
27.70
Copy URI
sha256:417ce7e7f2ccdd7...
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

- Task 5.2: To create an Amazon ECS cluster I started by creating a serverless AWS Fargate cluster named "microservices-serverlesscluster." During the setup I configured the cluster to use LabVPC, PublicSubnet1, and PublicSubnet2, ensuring no other subnets were included in the configuration. I made sure not to select Amazon EC2 instances or ECS Anywhere adhering to the serverless Fargate architecture.

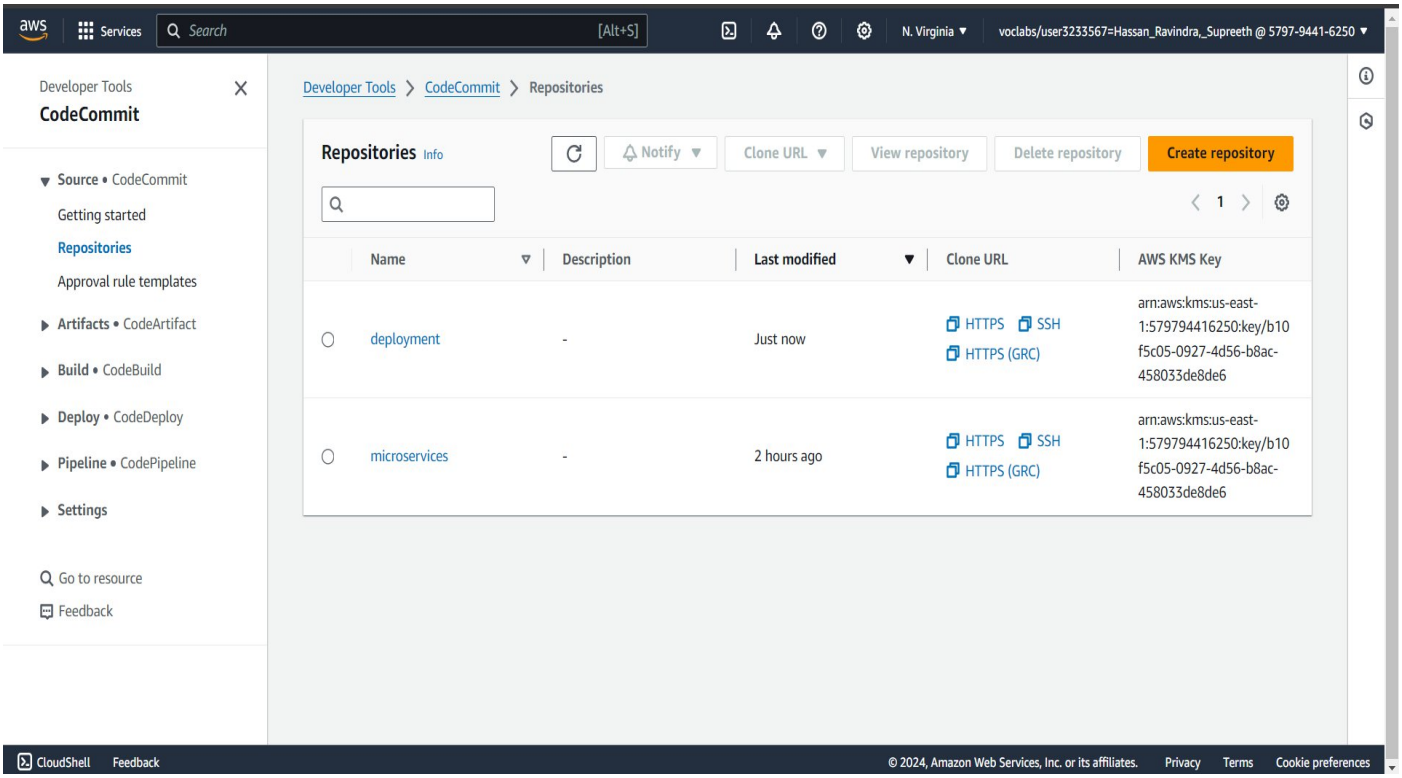
- After choosing to create the cluster I used the "View in CloudFormation" button to monitor the status of the stack that creates the cluster. I waited until the status changed to CREATE_COMPLETE before proceeding, ensuring the cluster was successfully created.

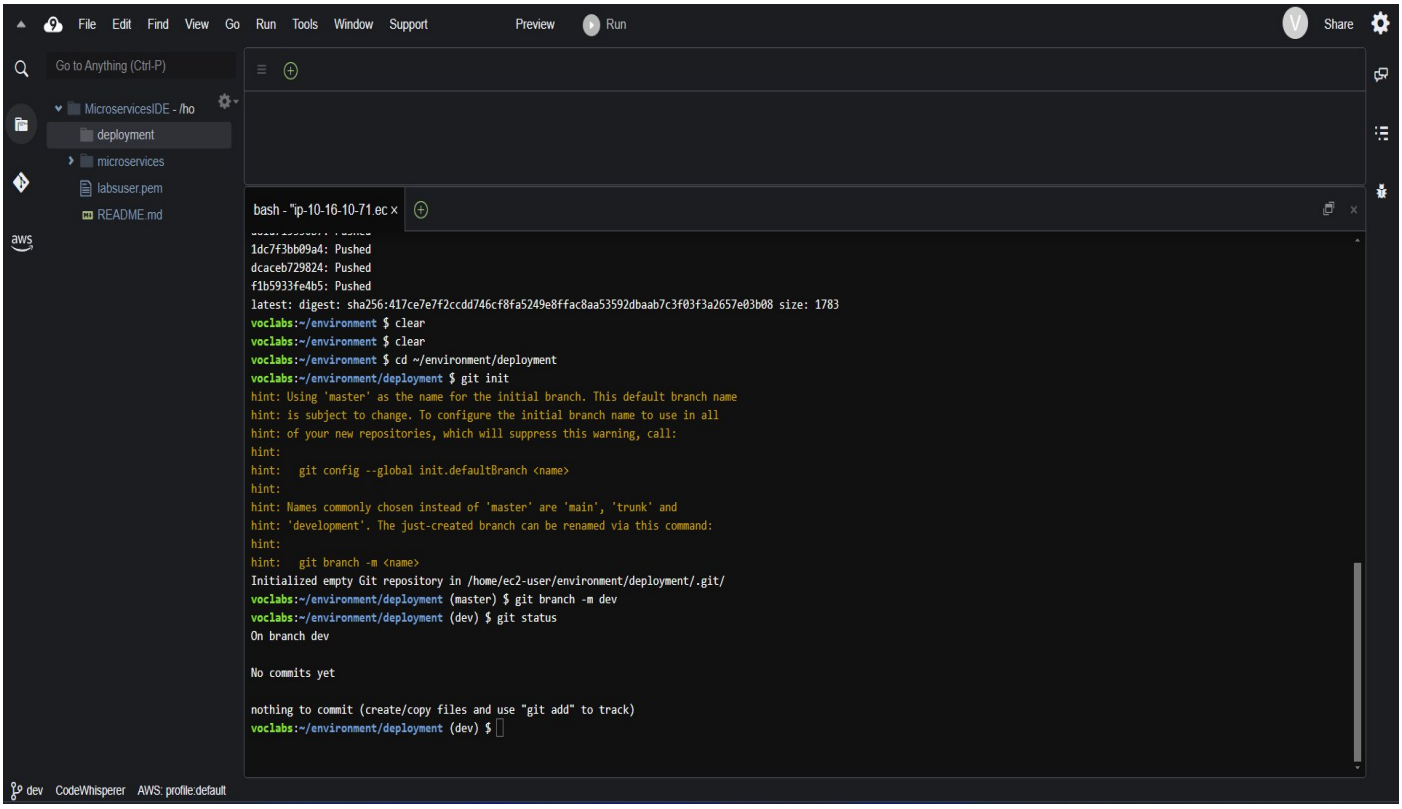




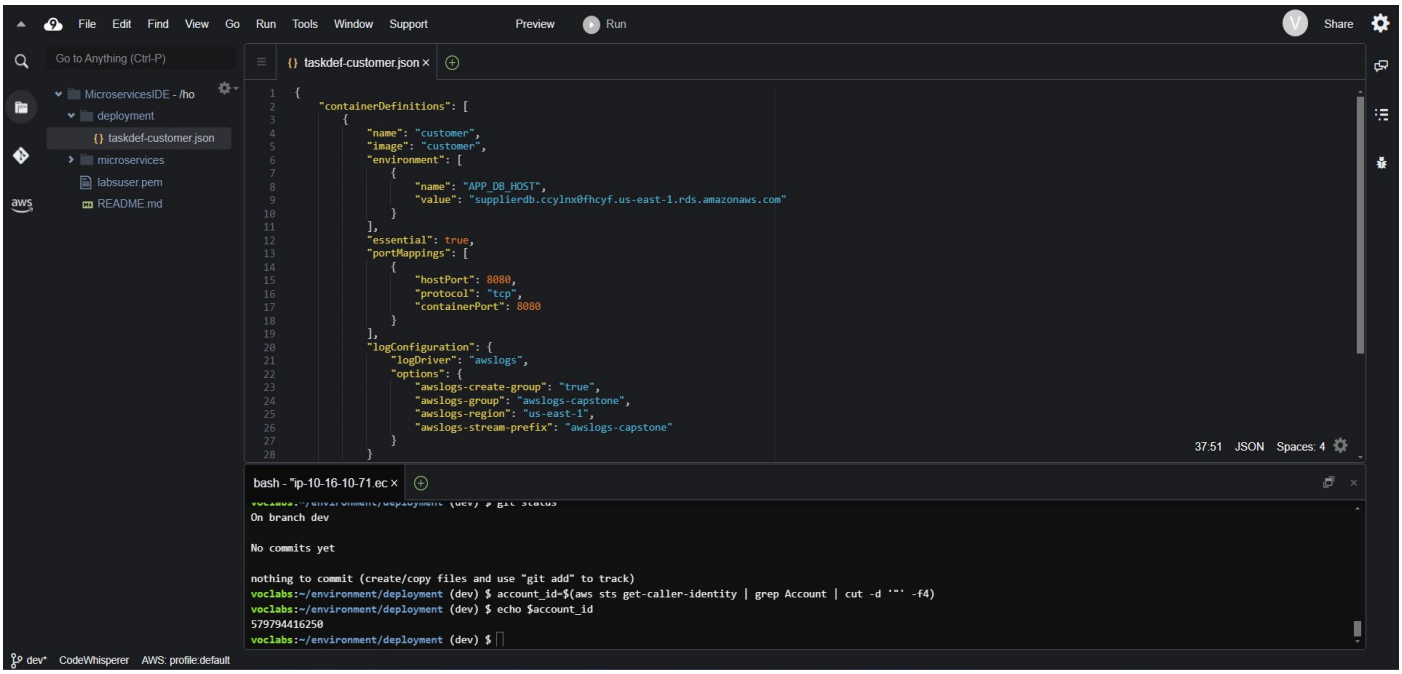


- Task 5.3: To create a new CodeCommit repository for storing deployment configuration files I began by creating a CodeCommit repository named "deployment." This repository will house the task configuration specification files for Amazon ECS and the AppSpec specification files for CodeDeploy
- In the AWS Cloud9 environment, I created a new directory named "deployment" to hold these configuration files locally. Afterwards I initialized this new directory as a Git repository and created a branch named "dev" to manage the deployment-related code. This setup provides a structured environment for handling deployment configurations and integrates with the new CodeCommit repository.





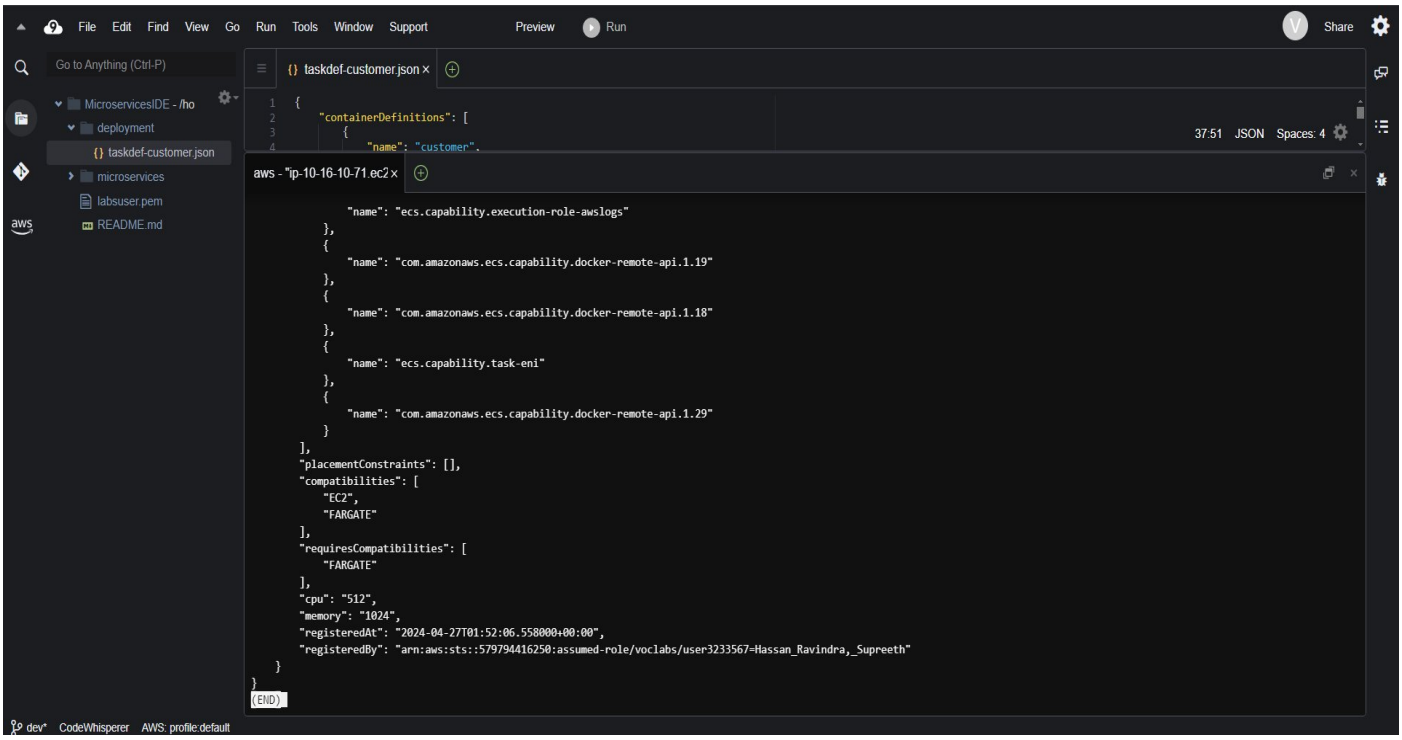
- Task 5.4: To create and register task definition files for each microservice with Amazon ECS I started by creating an empty file named `taskdef-customer.json` in the new deployment directory. I then edited the file and pasted the provided JSON code, which specifies the task definition for the customer microservice. I updated line 37 with my AWS account ID and line 9 with the actual RDS endpoint to replace the placeholder values.

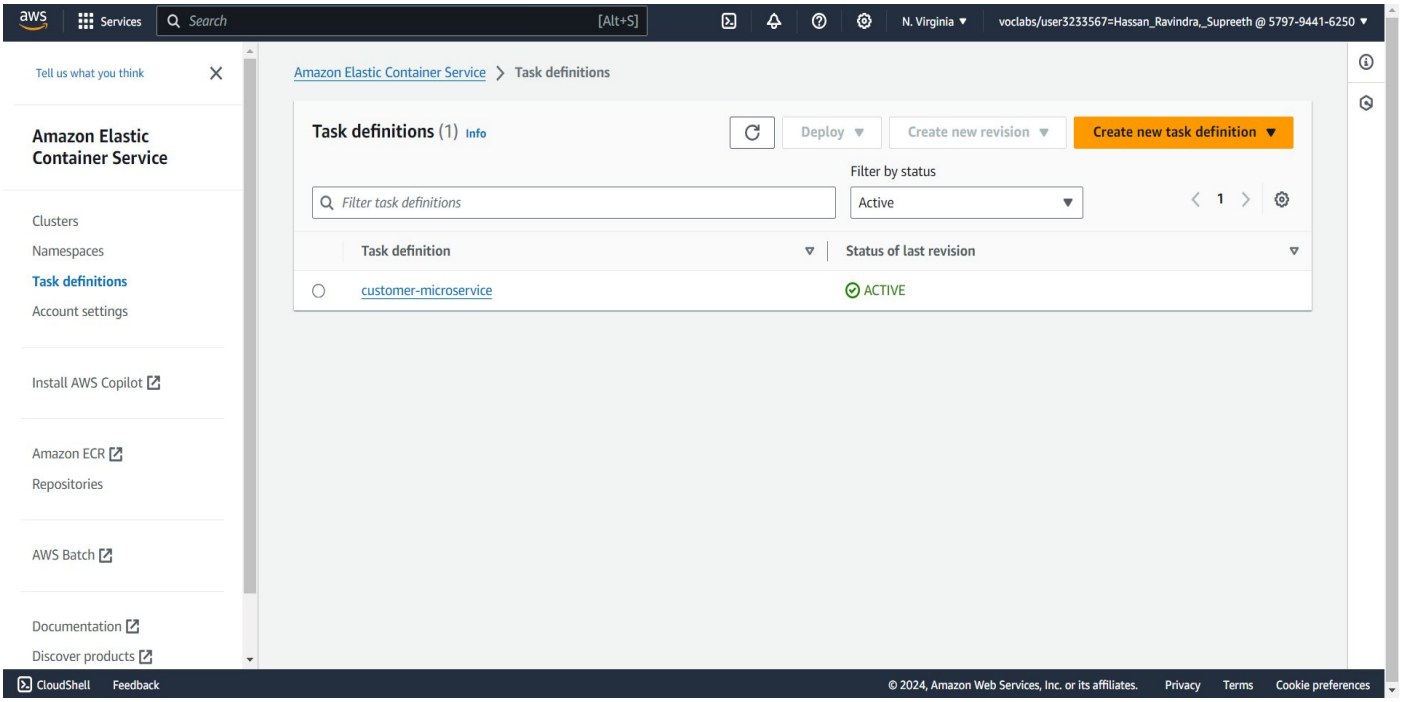


- I registered the customer microservice task definition with Amazon ECS by running the following command:

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

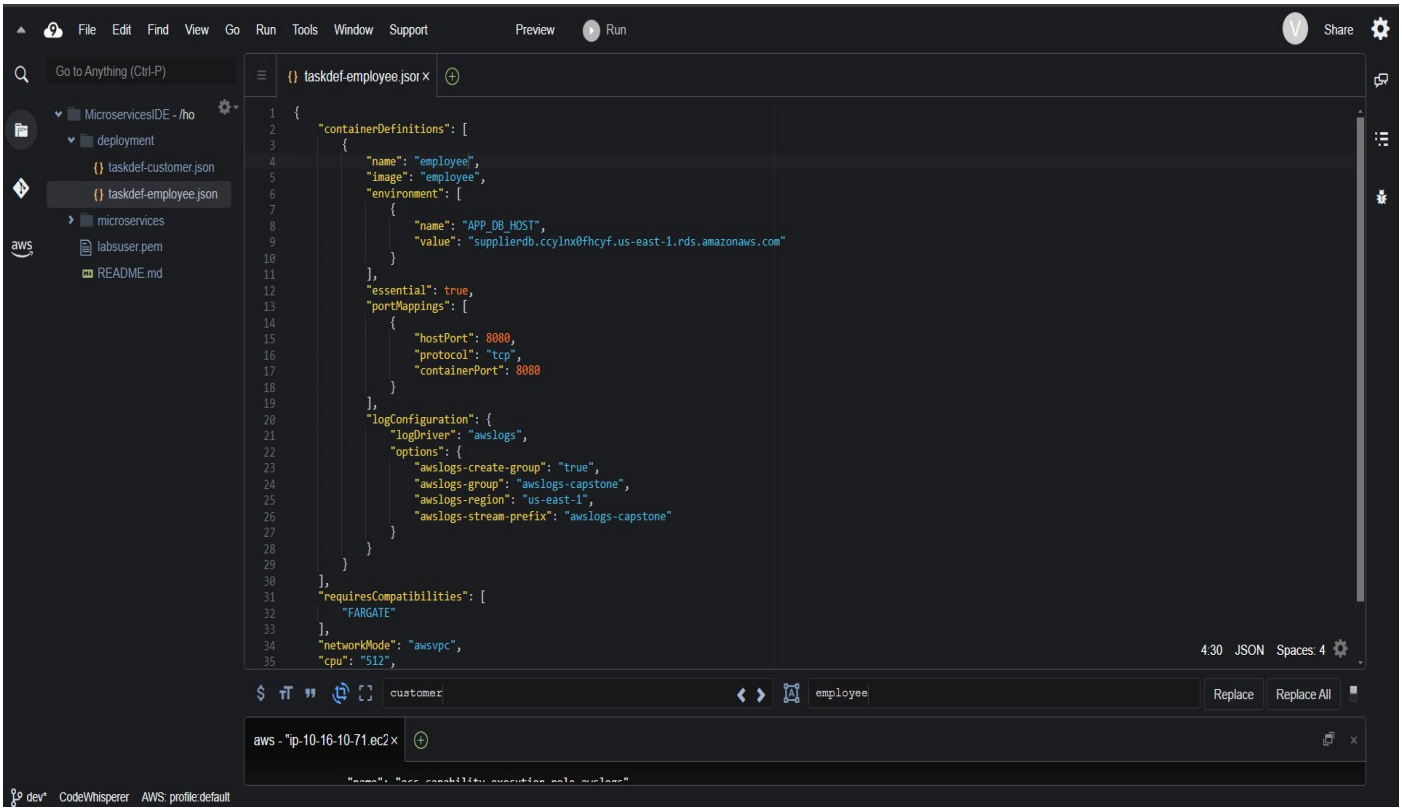
Afterwards I verified in the Amazon ECS console that the customer-microservice task definition appeared in the Task definitions pane, along with its revision number.

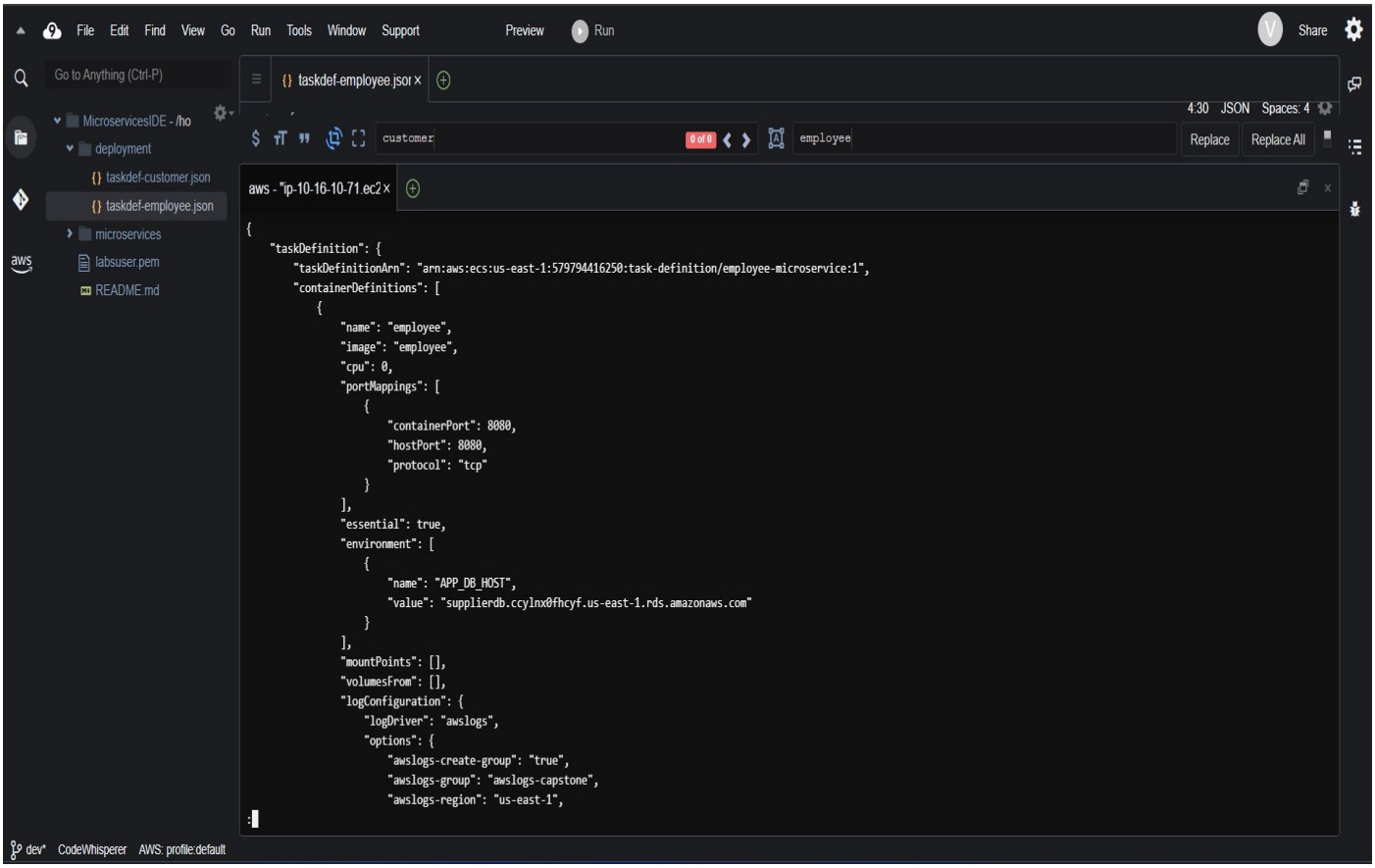


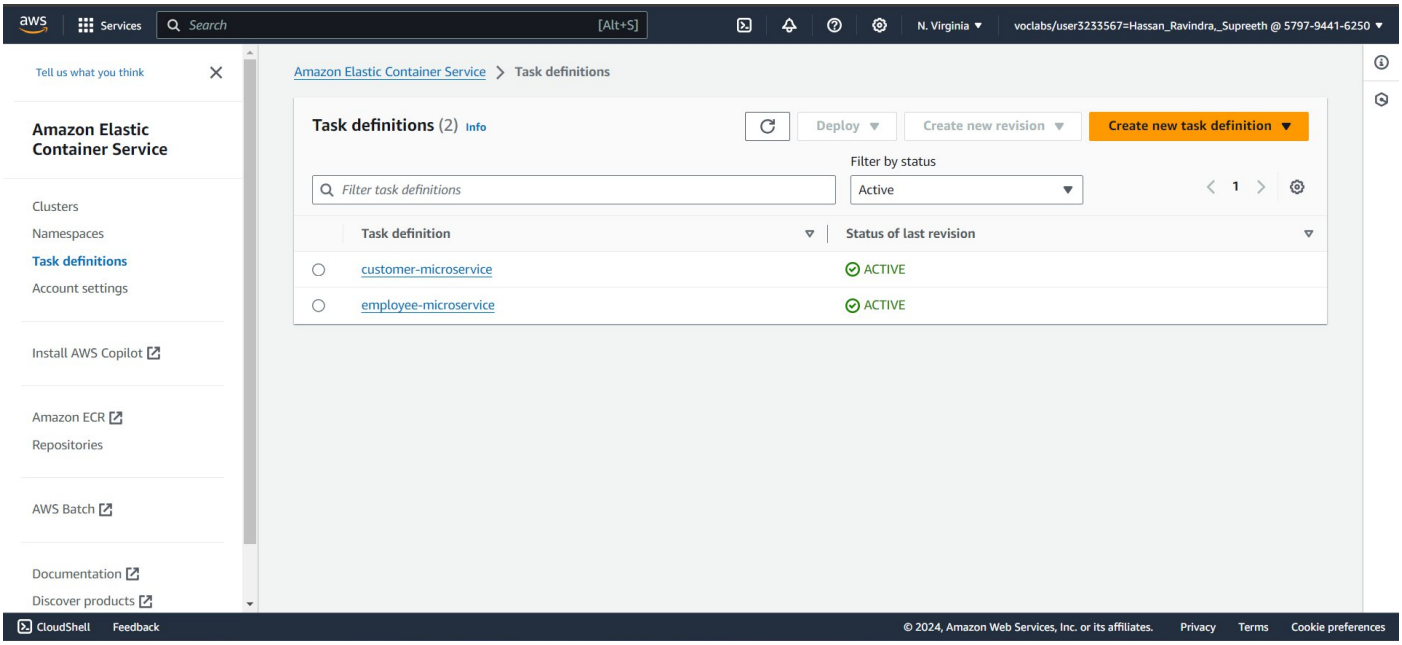


- I created a `taskdef-employee.json` specification file in the deployment directory, using the same JSON code from `taskdef-customer.json`. After pasting the code, I changed three occurrences of "customer" to "employee" to tailor the task definition to the employee microservice. I registered

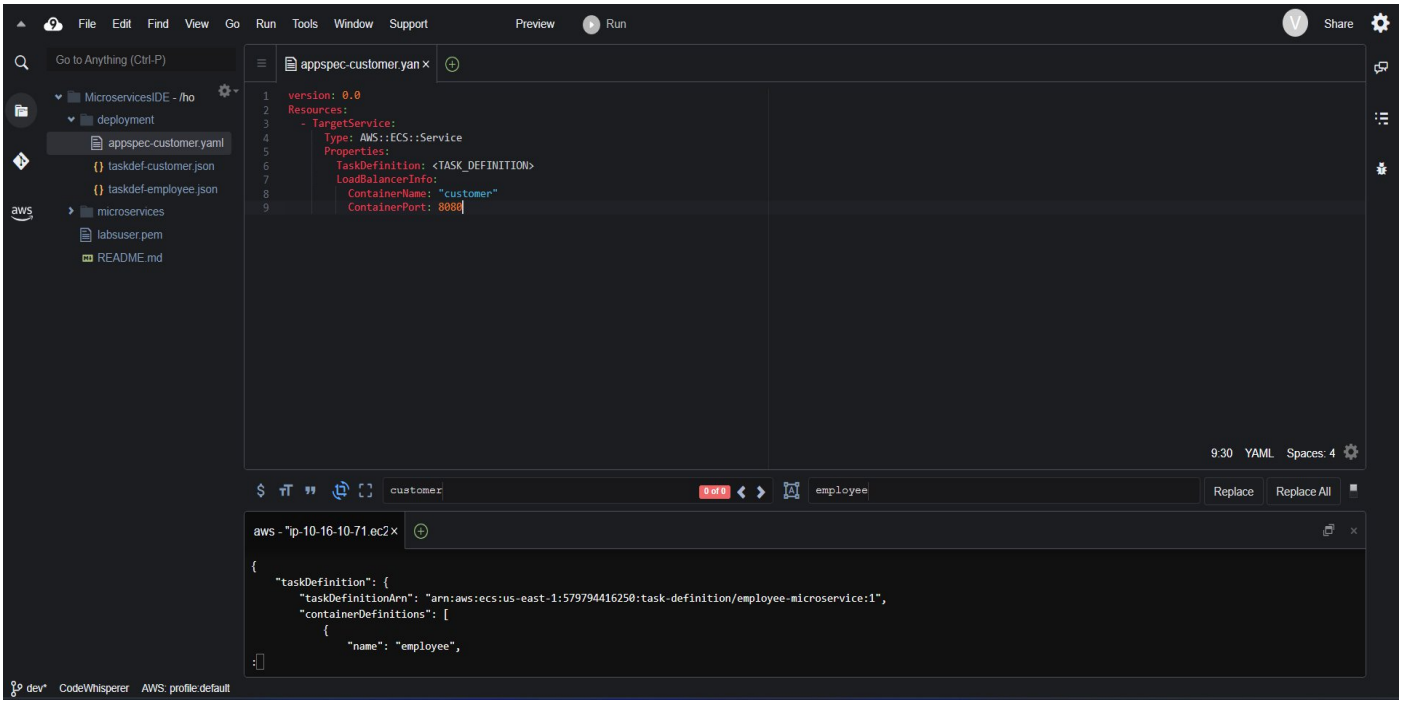
the employee task definition with Amazon ECS using the AWS CLI command and verified its presence in the Task definitions pane in the ECS console, confirming that the employee-microservice task definition was registered, along with its revision number. These steps ensure that both microservices have their task definitions set up and registered with Amazon ECS for deployment.





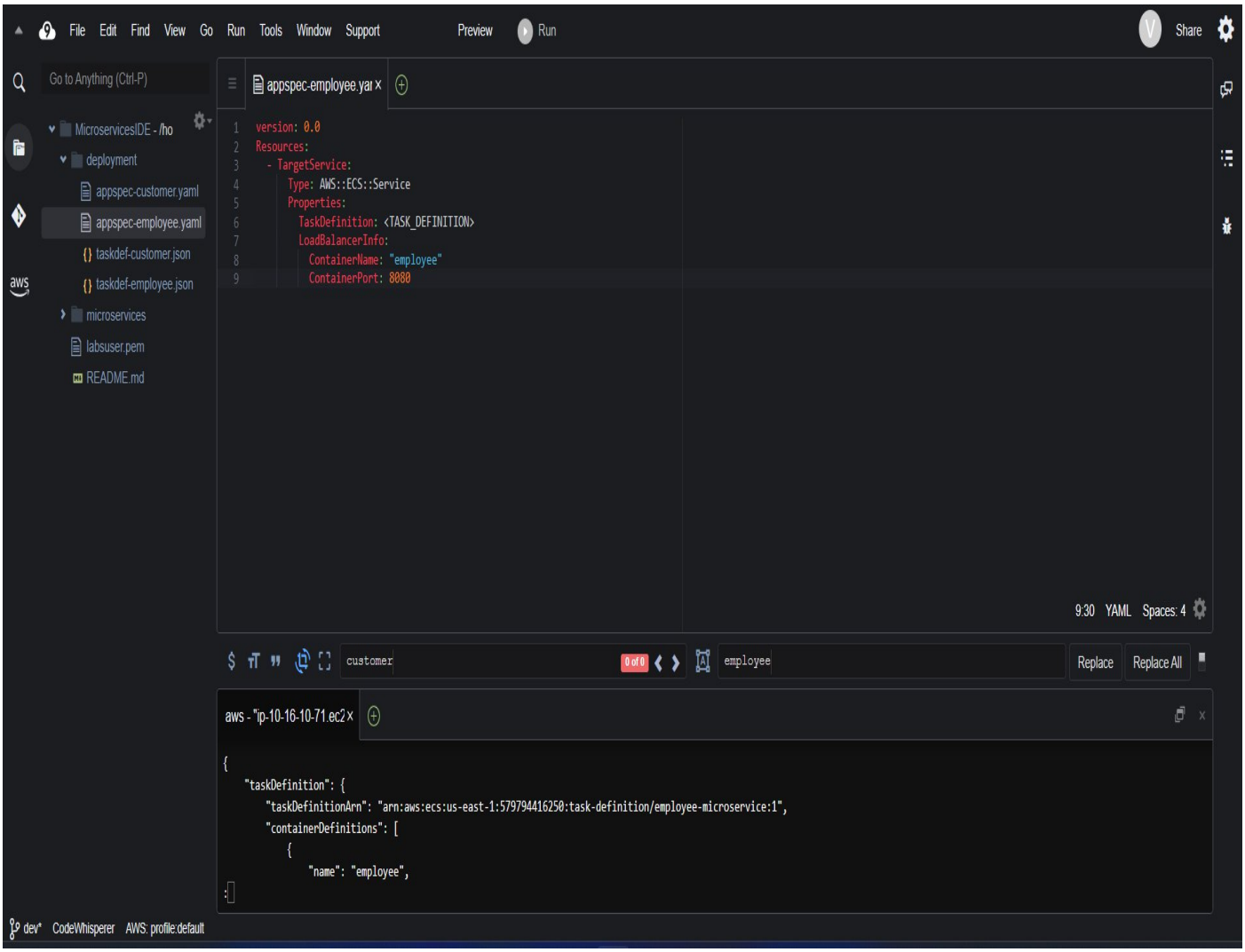


- Task 5.5: To create AppSpec files for CodeDeploy for each microservice, I began by creating an AppSpec file for the customer microservice in the deployment directory. I created a new file named `appspec-customer.yaml` and pasted the provided YAML code which specifies deployment instructions for CodeDeploy on Amazon ECS with Fargate infrastructure. The code outlines the target service, load balancer information, and task definition with a placeholder for `<TASK_DEFINITION>`, which will be updated automatically when the CI/CD pipeline runs. I ensured the YAML formatting including indentation was correct then saved the file.

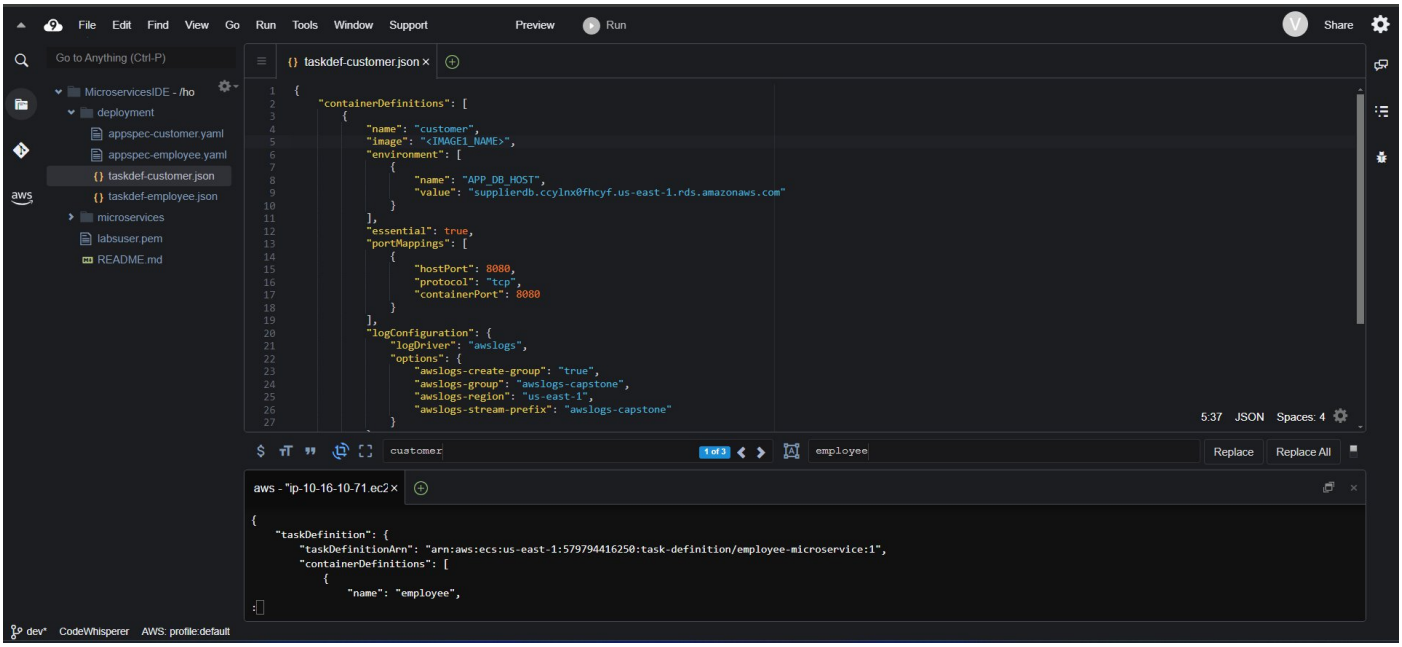


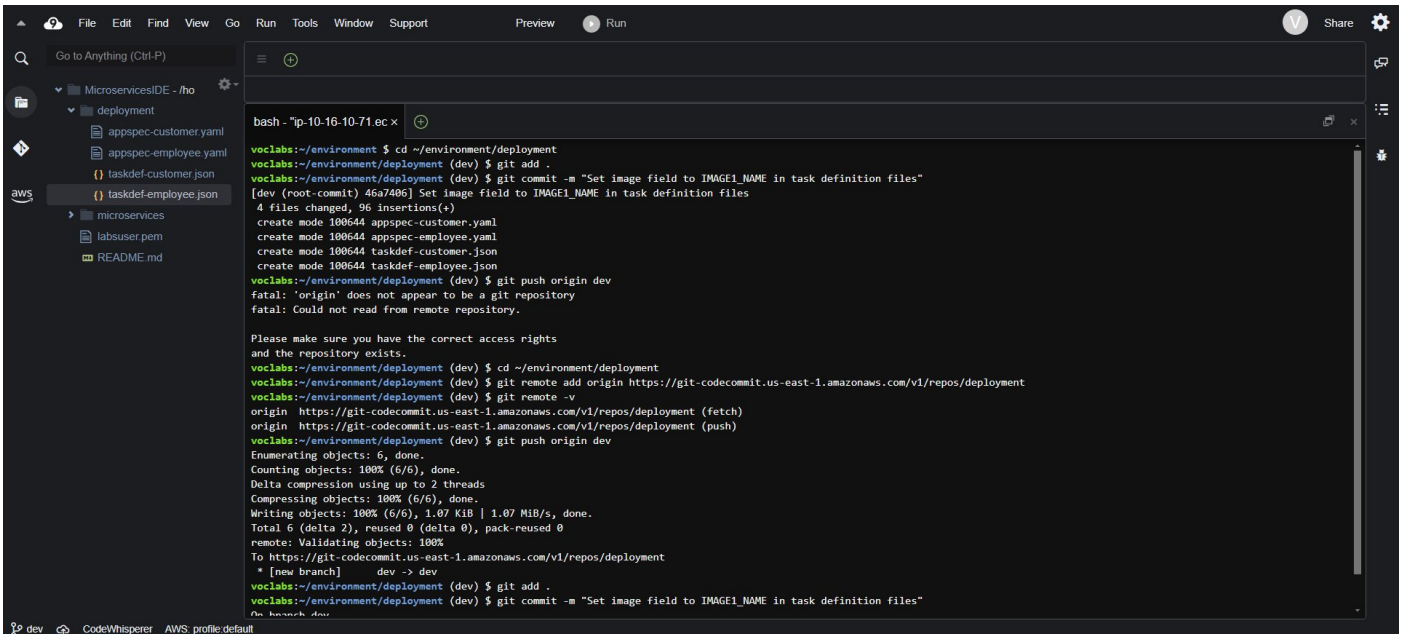
- I created an AppSpec file for the employee microservice in the same deployment directory, naming it `appspec-employee.yaml`. I used the same YAML code from `appspec-customer.yaml`, but changed the `ContainerName` from "customer" to "employee." This adjustment ensures the

file instructs CodeDeploy to target the employee microservice during deployment. After confirming the contents and proper indentation, I saved the changes, completing the creation of the AppSpec files for both microservices to guide CodeDeploy's deployment process in the ECS cluster.



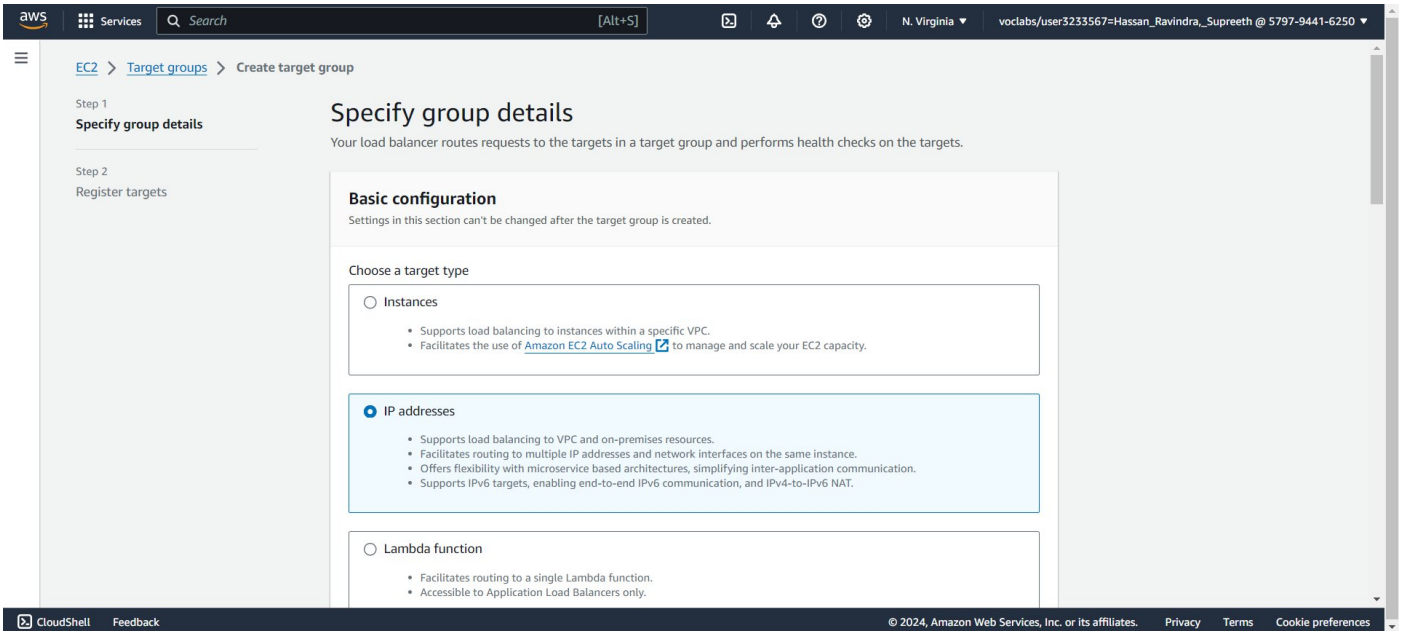
- Task 5.6: To update the task definition files and check them into CodeCommit, I started by editing the `taskdef-customer.json` file. I modified line 5 to set the `"image"` value to `"<IMAGE1_NAME>"` to use as a placeholder text for the image name, allowing dynamic updates later in the project. After saving the change I repeated the same step for the `taskdef-employee.json` file, also modifying line 5 to use `"<IMAGE1_NAME>"` as the placeholder text for the image name.
- With these changes, I ensured that the task definition files are set up for future integration with CodePipeline, where the correct image names will be dynamically updated at runtime. After saving the modifications I pushed the four files (including the two AppSpec files and the two updated task definition files) to the CodeCommit repository to store the latest deployment configurations.
- This process ensures that the updated task definition files and AppSpec files are securely checked into CodeCommit, preparing them for later use in the CI/CD pipeline to support deploying the microservices-based application to Amazon ECS.

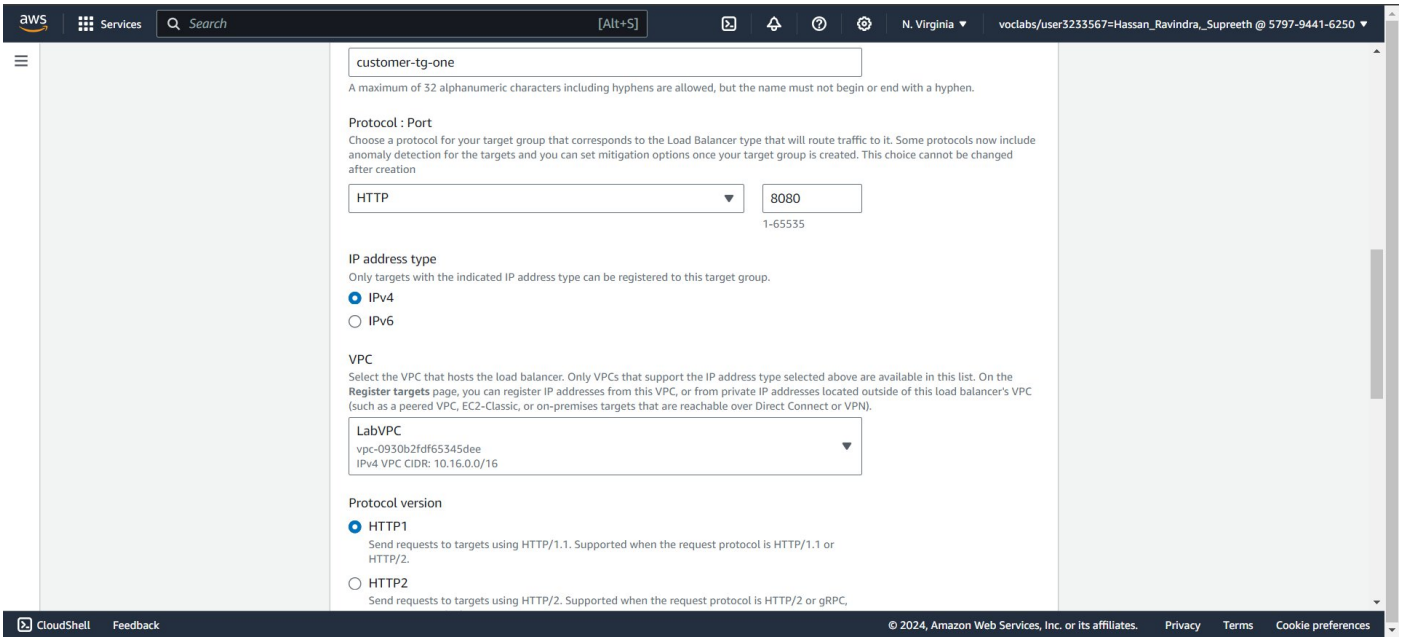


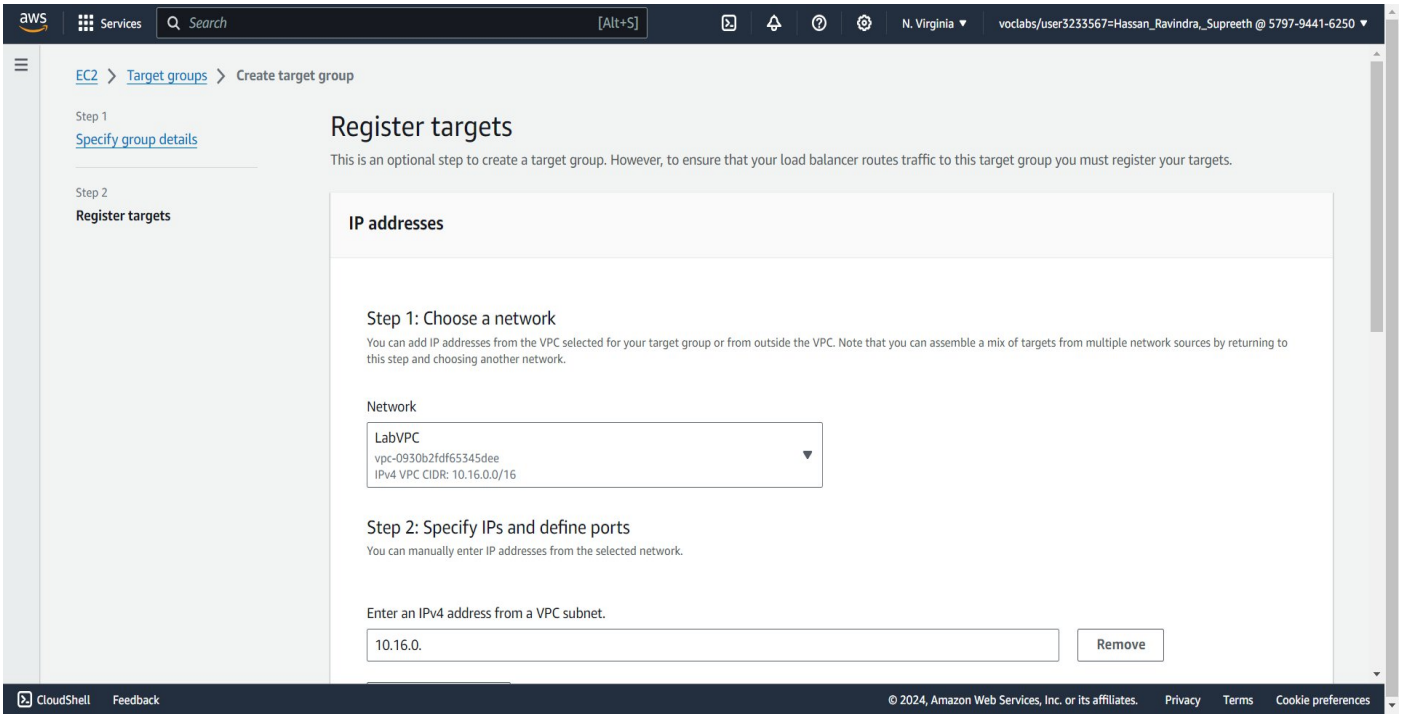


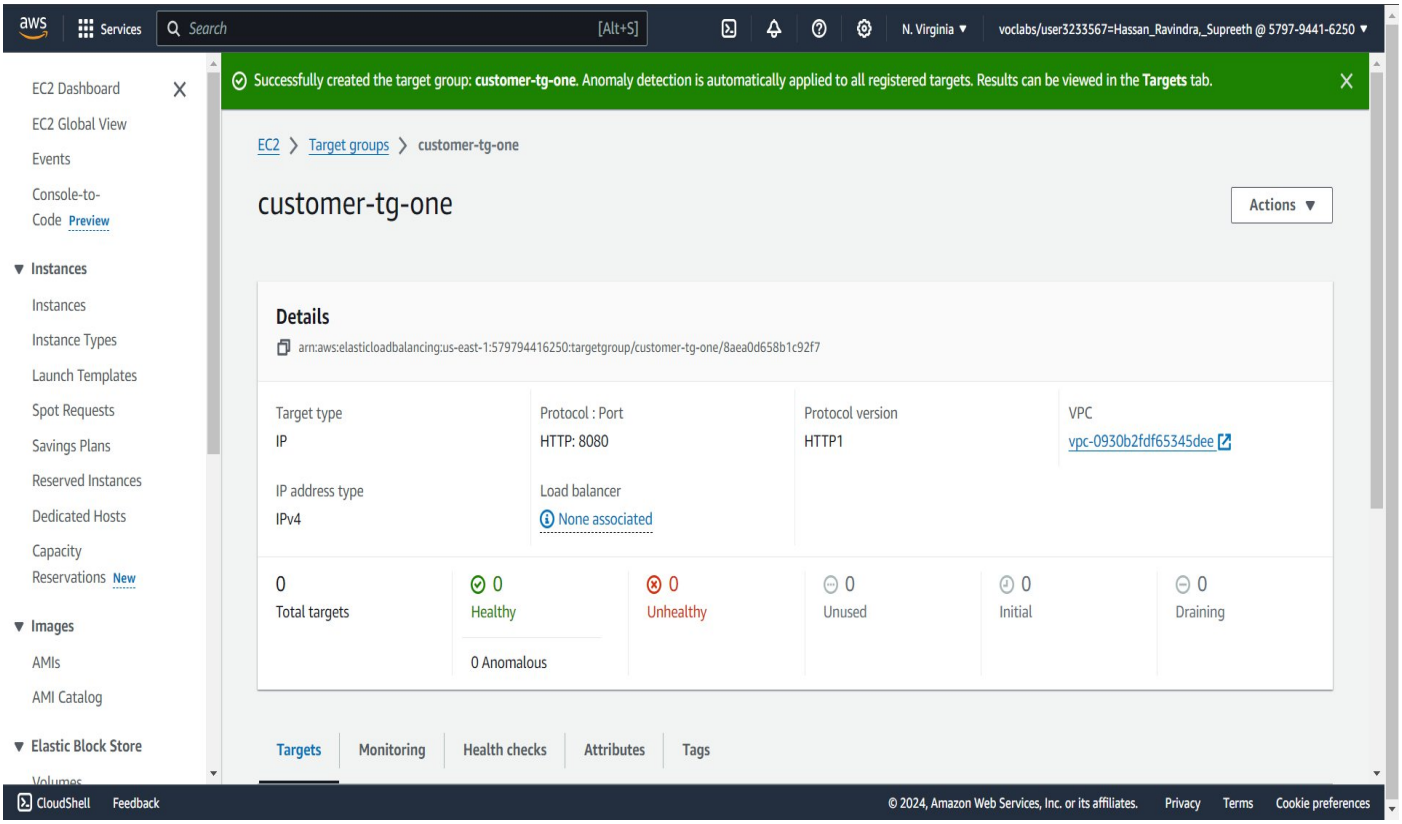
PHASE-6: Creating target groups and an Application Load Balancer

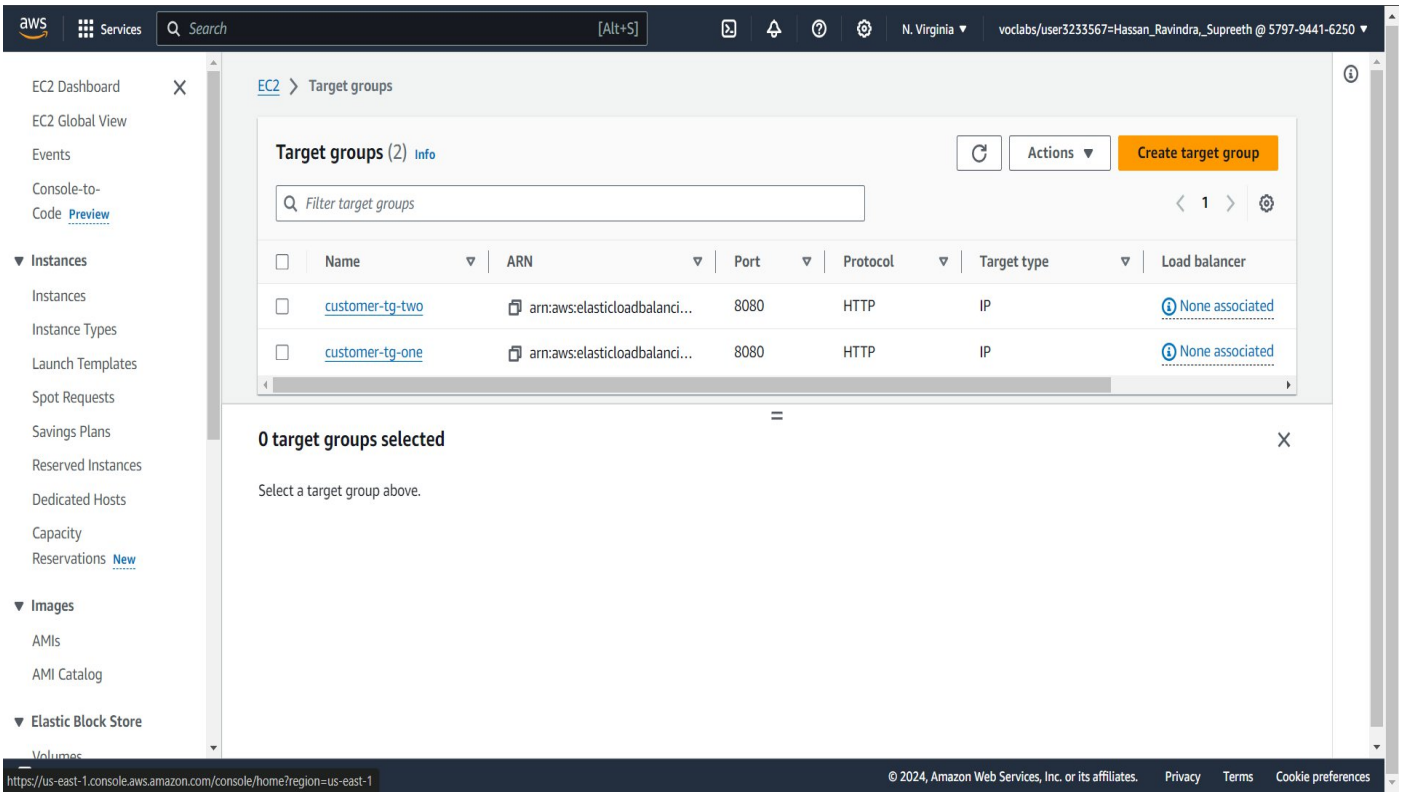
- Task 6.1: To create four target groups for a blue/green deployment in Amazon EC2 I started by creating two target groups for the customer microservice and two for the employee microservice.
- I navigated to the Amazon EC2 console, selected "Target Groups," and then chose "Create target group." For the first customer microservice target group, I configured it according to the lab instructions. I left all defaults unchanged on the "Register targets" page (without registering any targets) and created the target group.
- For the second customer microservice target group I used the same settings but with the target group name `customer-tg-two`.


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
EC2 > Target groups > Create target group
Step 1
Specify group details
Step 2
Register targets
Specify group details
Your load balancer routes requests to the targets in a target group and performs health checks on the targets.
Basic configuration
Settings in this section can't be changed after the target group is created.
Choose a target type
Instances
Supports load balancing to instances within a specific VPC.
Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.
IP addresses
Supports load balancing to VPC and on-premises resources.
Facilitates routing to multiple IP addresses and network interfaces on the same instance.
Offers flexibility with microservice based architectures, simplifying inter-application communication.
Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
Lambda function
Facilitates routing to a single Lambda function.
Accessible to Application Load Balancers only.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

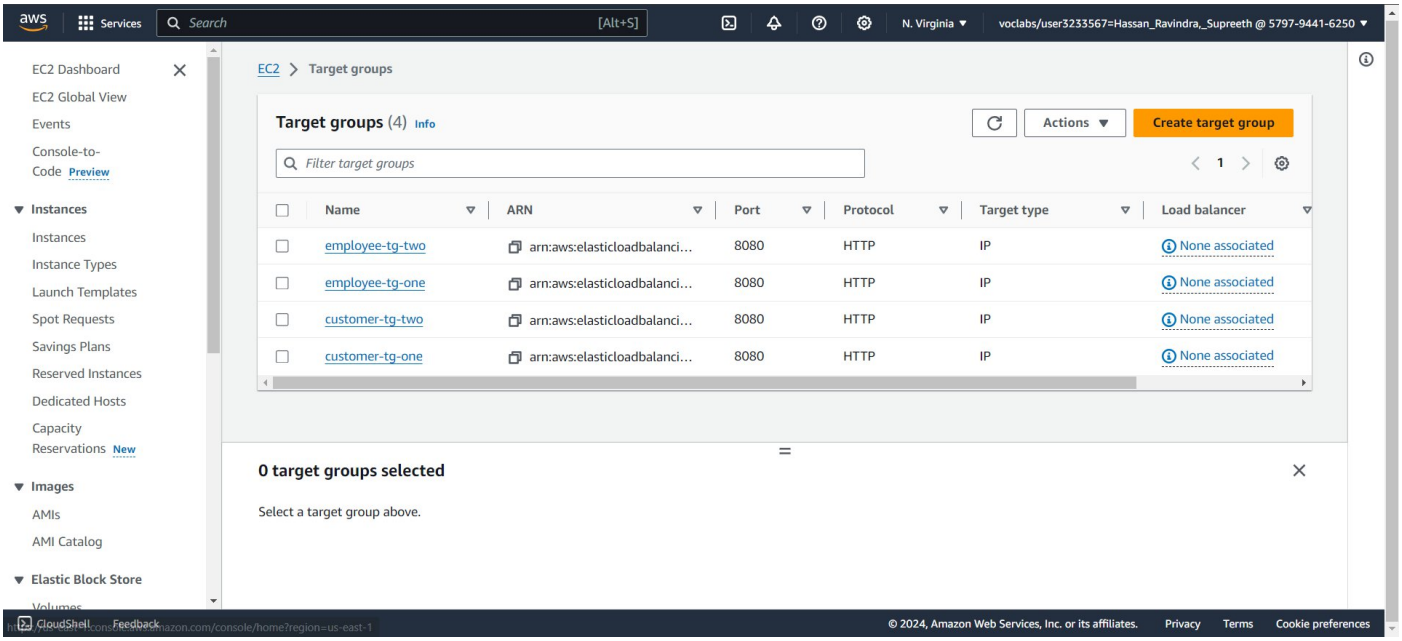

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
customer-tg-one
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.
Protocol : Port
Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation
HTTP
8080
1-65535
IP address type
Only targets with the indicated IP address type can be registered to this target group.
IPv4
IPv6
VPC
Select the VPC that hosts the load balancer. Only VPCs that support the IP address type selected above are available in this list. On the Register targets page, you can register IP addresses from this VPC, or from private IP addresses located outside of this load balancer's VPC (such as a peered VPC, EC2-Classic, or on-premises targets that are reachable over Direct Connect or VPN).
LabVPC
vpc-0930b2fdf65345dee
IPv4 VPC CIDR: 10.16.0.0/16
Protocol version
HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC,
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
EC2 > Target groups > Create target group
Step 1
Specify group details
Step 2
Register targets
Register targets
This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.
IP addresses
Step 1: Choose a network
You can add IP addresses from the VPC selected for your target group or from outside the VPC. Note that you can assemble a mix of targets from multiple network sources by returning to this step and choosing another network.
Network
LabVPC
vpc-0930b2fdf65345dee
IPv4 VPC CIDR: 10.16.0.0/16
Step 2: Specify IPs and define ports
You can manually enter IP addresses from the selected network.
Enter an IPv4 address from a VPC subnet.
10.16.0.
Remove
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

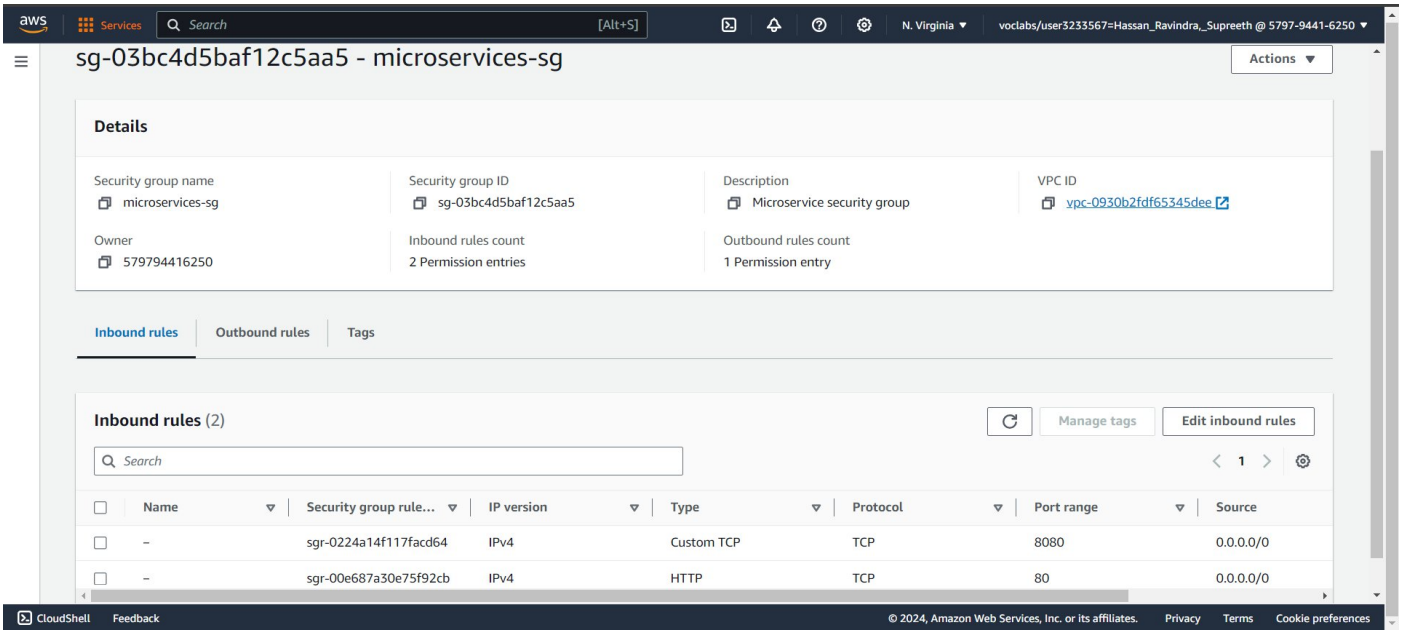


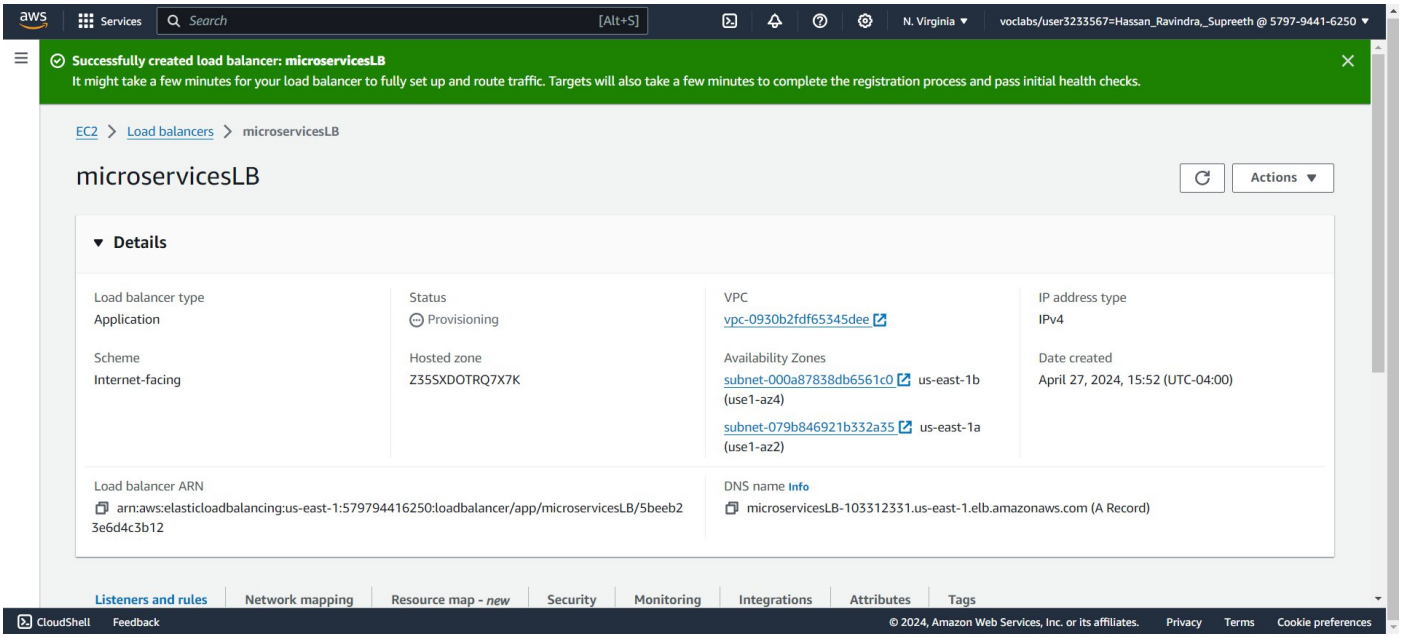


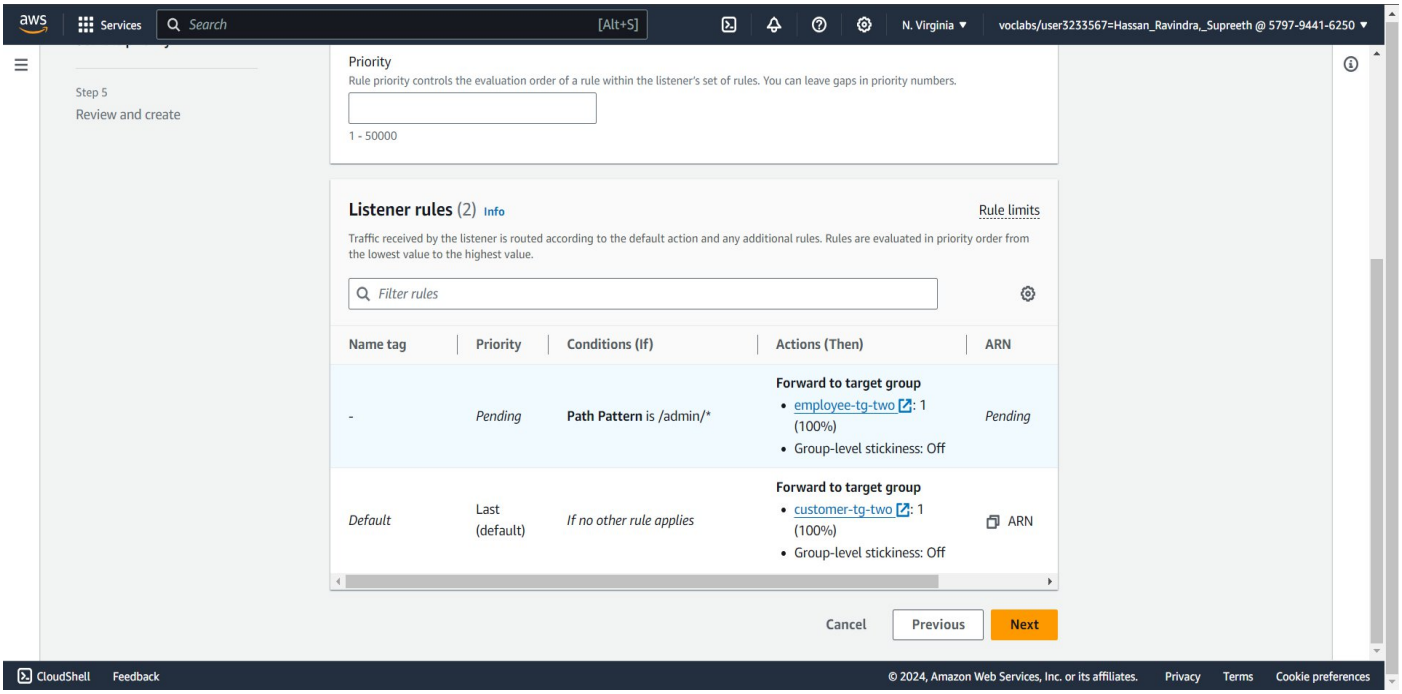
- Similarly, I created a target group for the employee microservice using the same initial configuration, with some exceptions:
- For the second target group for the employee microservice, I used the same settings, naming it `employee-tg-two`, with the same health check path `/admin/suppliers`.
- By creating these four target groups, I set the stage for blue/green deployments, allowing for a smooth transition between different versions of the microservices without downtime. These target groups are used by CodeDeploy for managing traffic during deployments in an Amazon ECS cluster.

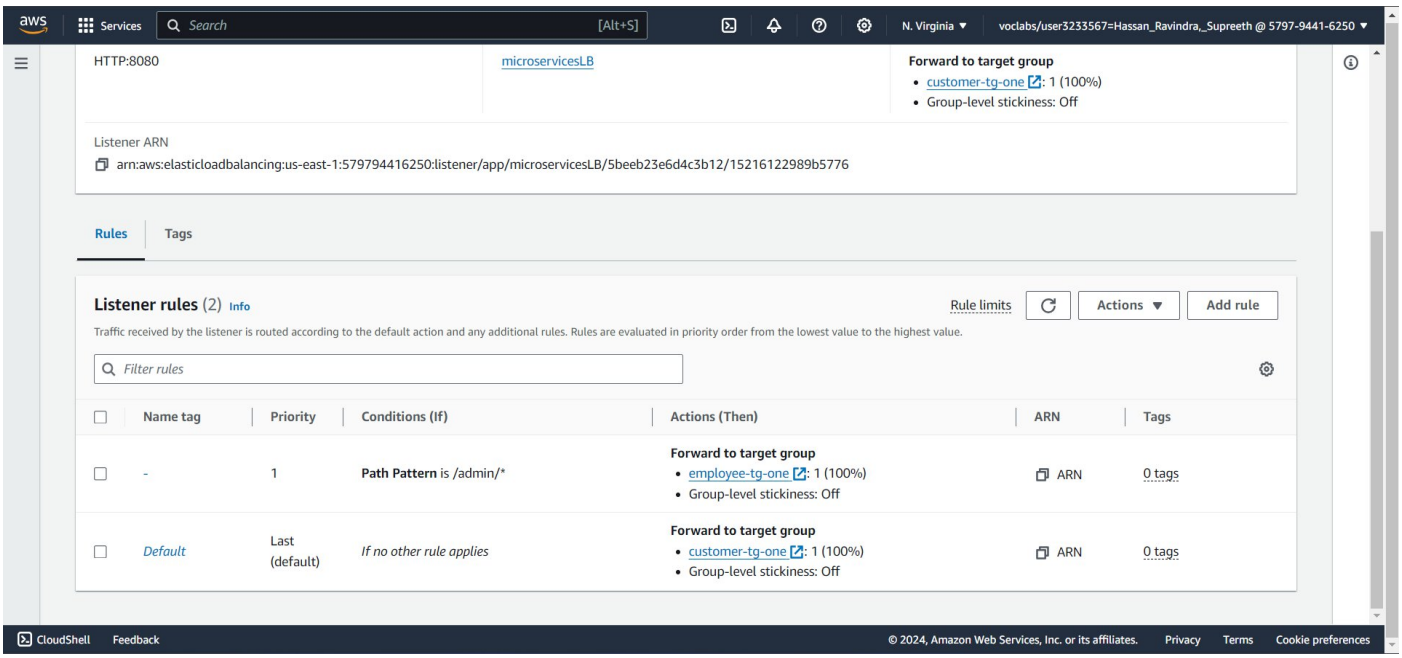


- Task 6.2: To set up an Application Load Balancer (ALB) with path-based routing rules I first created a new EC2 security group named `microservices-sg` for LabVPC, allowing inbound TCP traffic on ports 80 and 8080 from any IPv4 address. Then, in the Amazon EC2 console, I created an Application Load Balancer named `microservicesLB` configured as internet-facing for IPv4 addresses, with LabVPC, Public Subnet1, Public Subnet2, and the newly created `microservices-sg` security group.
- Next I configured two listeners on the load balancer: one listening on HTTP:80 and forwarding traffic to the `customer-tg-two` target group by default, and another on HTTP:8080 with a default forwarding to `customer-tg-one`.
- For the HTTP:80 listener I added a path-based routing rule: if the URL path starts with `/admin/*`, traffic is routed to the `employee-tg-two` target group. Similarly, for the HTTP:8080 listener, I added a routing rule that forwards traffic to `employee-tg-one` when the path includes `/admin/*`.
- These configurations set up the ALB to handle traffic based on specified paths ensuring correct routing to target groups depending on whether the request is intended for customer or employee microservices, as part of the blue/green deployment strategy.




Successfully created load balancer: microservicesLB
It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.
EC2 > Load balancers > microservicesLB
microservicesLB
Actions
Details
Load balancer type
Application
Status
Provisioning
VPC
vpc-0930b2fdf65345dee
IP address type
IPv4
Scheme
Internet-facing
Hosted zone
Z35SXDOTRQ7X7K
Availability Zones
subnet-000a87838db6561c0 us-east-1b (use1-az4)
subnet-079b846921b332a35 us-east-1a (use1-az2)
Date created
April 27, 2024, 15:52 (UTC-04:00)
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:579794416250:loadbalancer/app/microservicesLB/5beeb23e6d4c3b12
DNS name Info
microservicesLB-103312331.us-east-1.elb.amazonaws.com (A Record)
Listeners and rules
Network mapping
Resource map - new
Security
Monitoring
Integrations
Attributes
Tags
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Step 5
Review and create
Priority
Rule priority controls the evaluation order of a rule within the listener's set of rules. You can leave gaps in priority numbers.
1 - 50000
Listener rules (2) Info
Rule limits
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
-
Pending
Path Pattern is /admin/*
Forward to target group
employee-tg-two: 1 (100%)
Group-level stickiness: Off
Pending
Default
Last (default)
If no other rule applies
Forward to target group
customer-tg-two: 1 (100%)
Group-level stickiness: Off
ARN
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

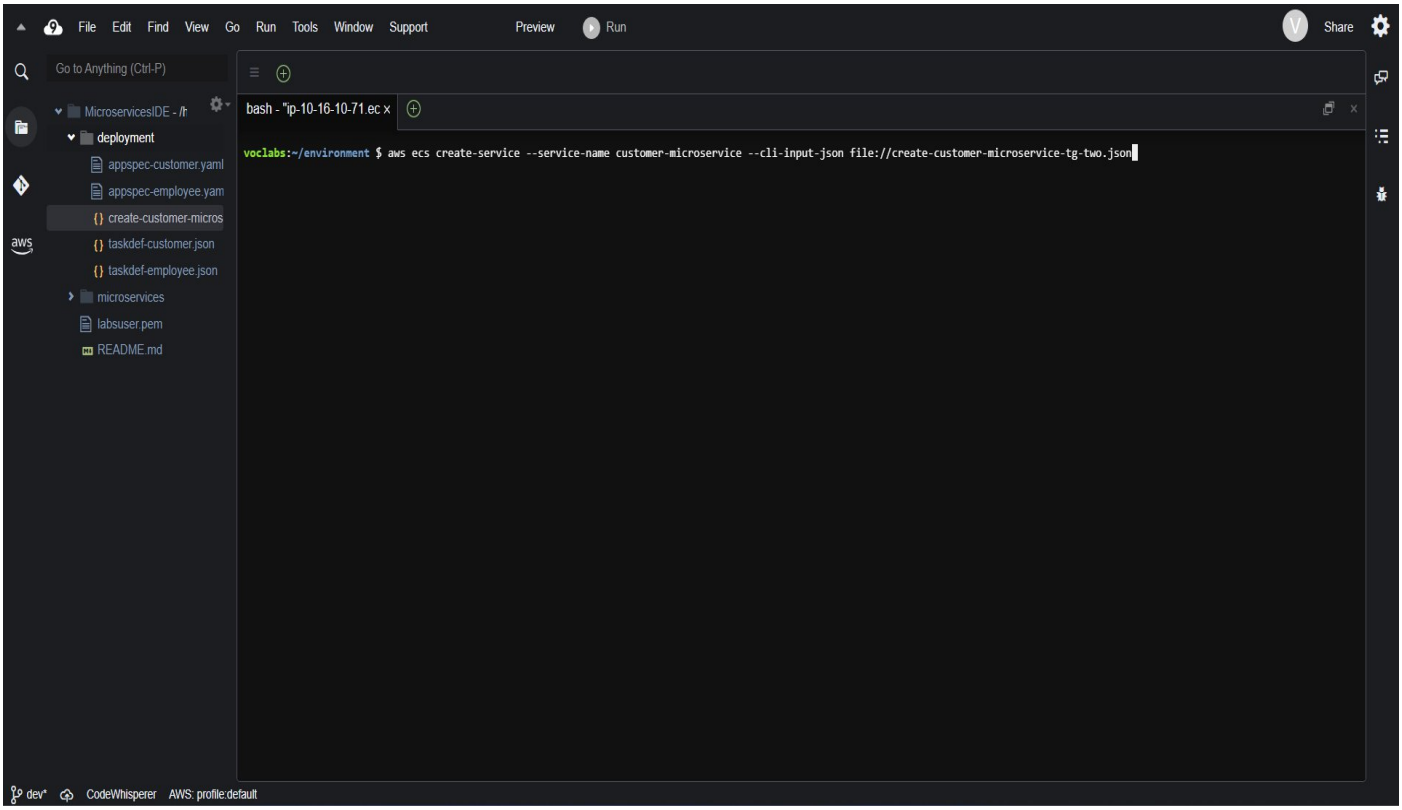

HTTP:8080
microservicesLB
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
Listener ARN
arn:aws:elasticloadbalancing:us-east-1:579794416250:listener/app/microservicesLB/5beeb23e6d4c3b12/15216122989b5776
Rules
Tags
Listener rules (2) Info
Rule limits
Actions
Add rule
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
Tags
-
1
Path Pattern is /admin/*
Forward to target group
employee-tg-one: 1 (100%)
Group-level stickiness: Off
ARN
0 tags
Default
Last (default)
If no other rule applies
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
ARN
0 tags
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

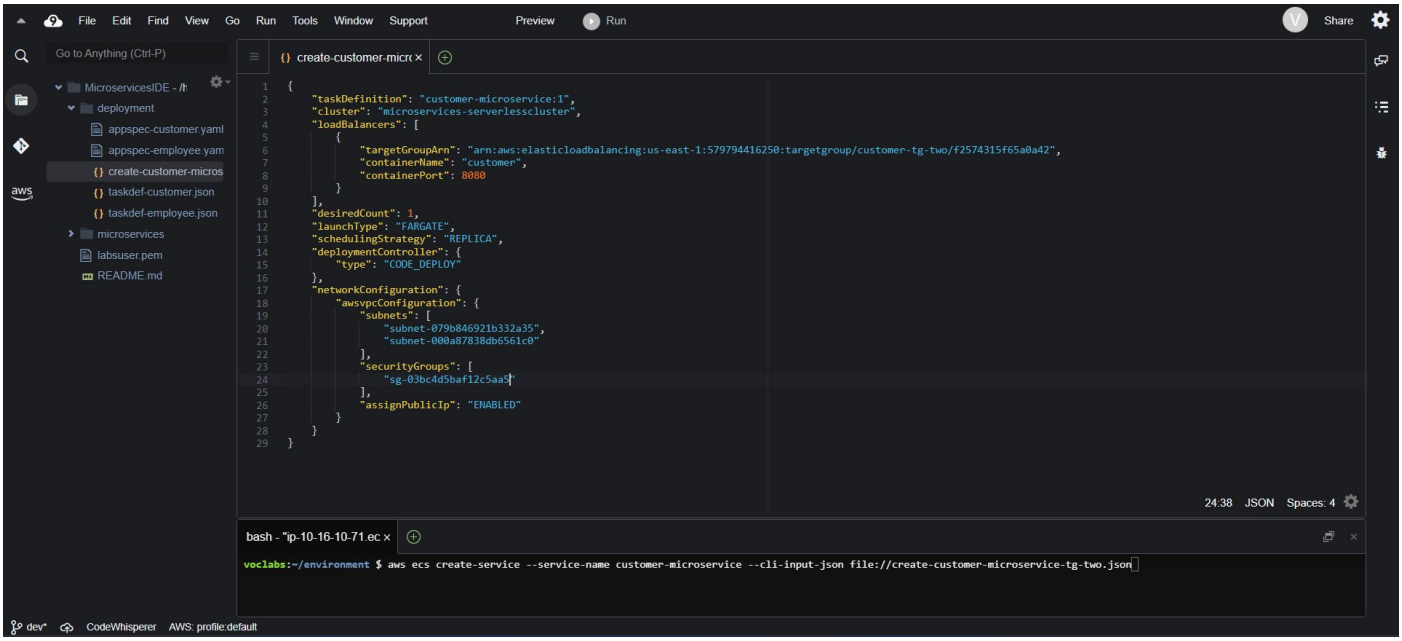
PHASE-7: Creating two Amazon ECS services

- Task 7.1: To create an ECS service for the customer microservice I started by creating a new file named `create-customer-microservice-tg-two.json` in the deployment directory within AWS Cloud9. I pasted the provided JSON code into this file, which contains the ECS service configuration including the task definition, cluster name, load balancer information, desired count of instances, launch type, network configuration, and other parameters.
- Next I edited the JSON code to customize it with specific details:
- I replaced "REVISION-NUMBER" with the latest revision number of the `customer-microservice` task definition. If this is the first time running the step, the revision number is 1. Otherwise, I checked the revision number in the Amazon ECS console by selecting "Task definitions" and then choosing `customer-microservice`.
- I replaced "MICROSERVICE-TG-TWO-ARN" with the actual ARN of the `customer-tg-two` target group.
- I replaced "PUBLIC-SUBNET-1-ID" and "PUBLIC-SUBNET-2-ID" with the respective subnet IDs for Public Subnet1 and Public Subnet2.
- I replaced "SECURITY-GROUP-ID" with the ID of the `microservices-sg` security group.
- After updating the JSON configuration, I saved the changes and navigated to the deployment directory. To create the ECS service for the customer microservice, I ran the following command:

  `aws ecs create-service --service-name customer-microservice --cli-input-json file://create-customer-microservice-tg-two.json`.

- This step establishes the ECS service for the customer microservice enabling deployment and traffic routing based on the defined configuration.

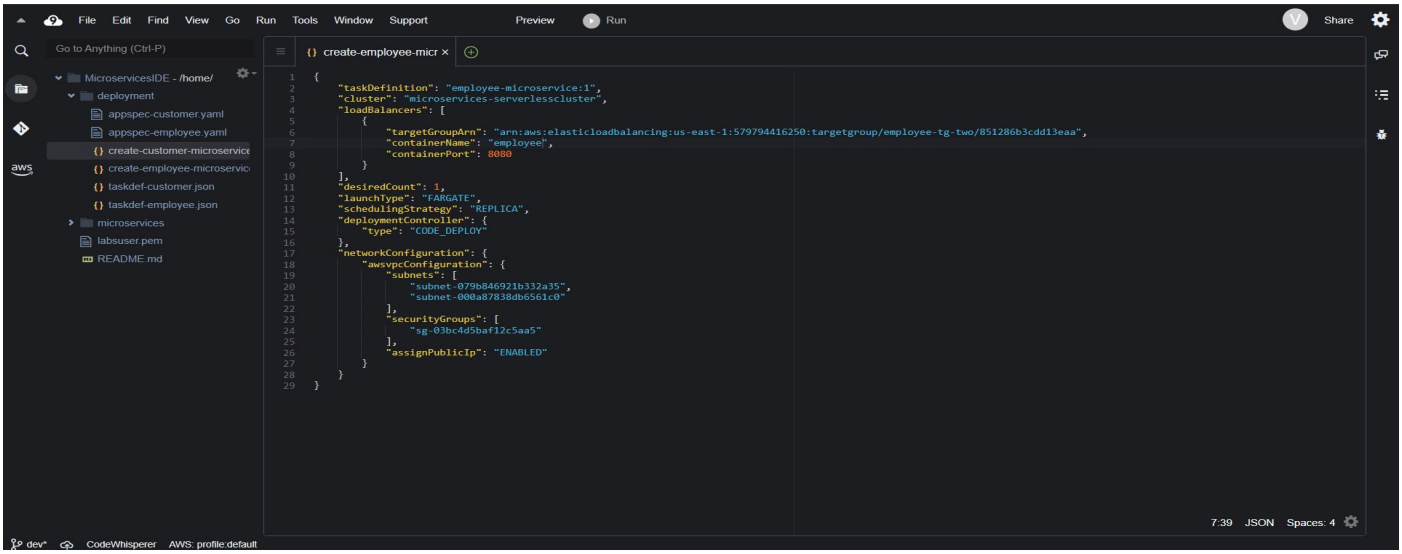


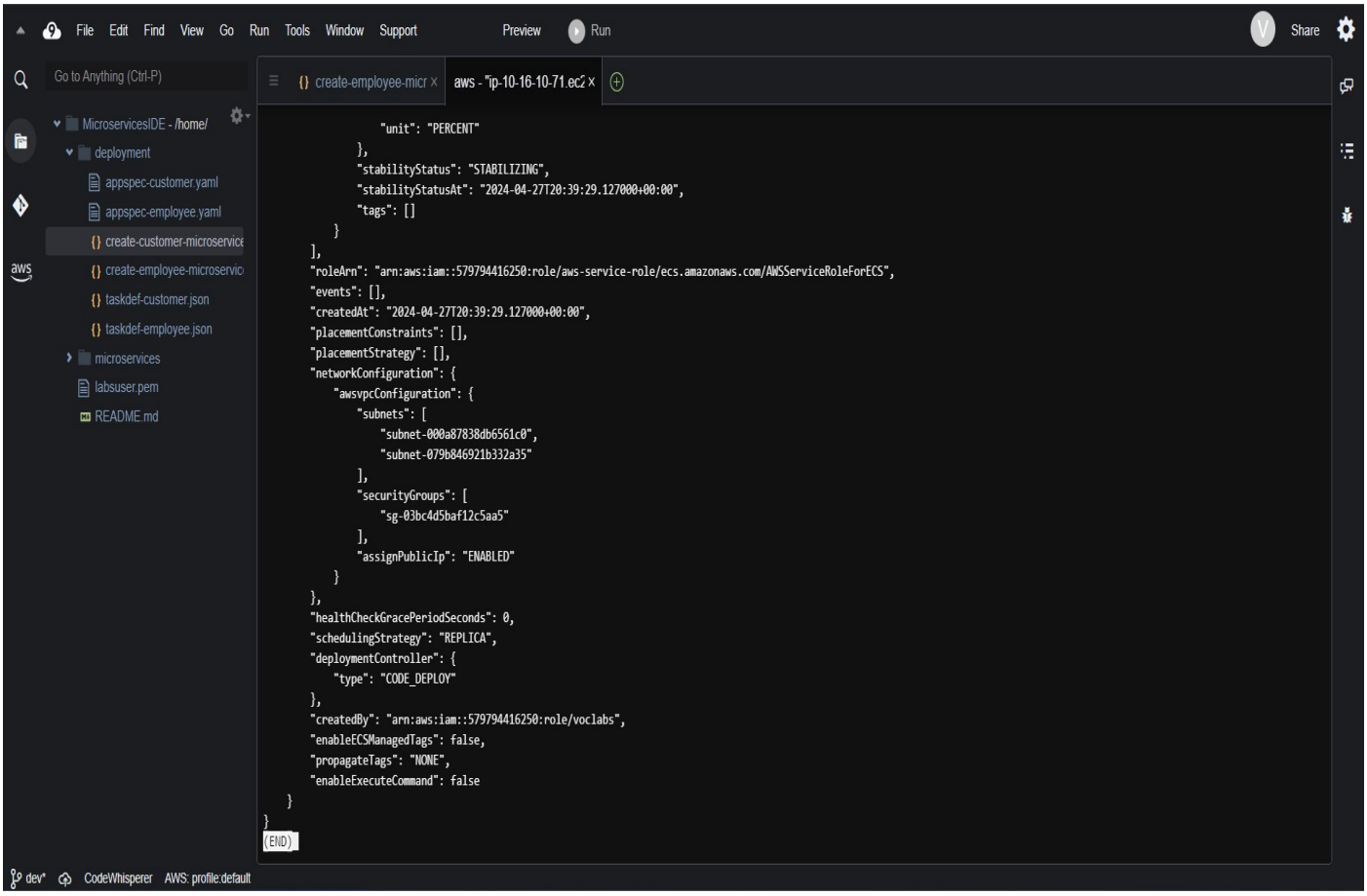


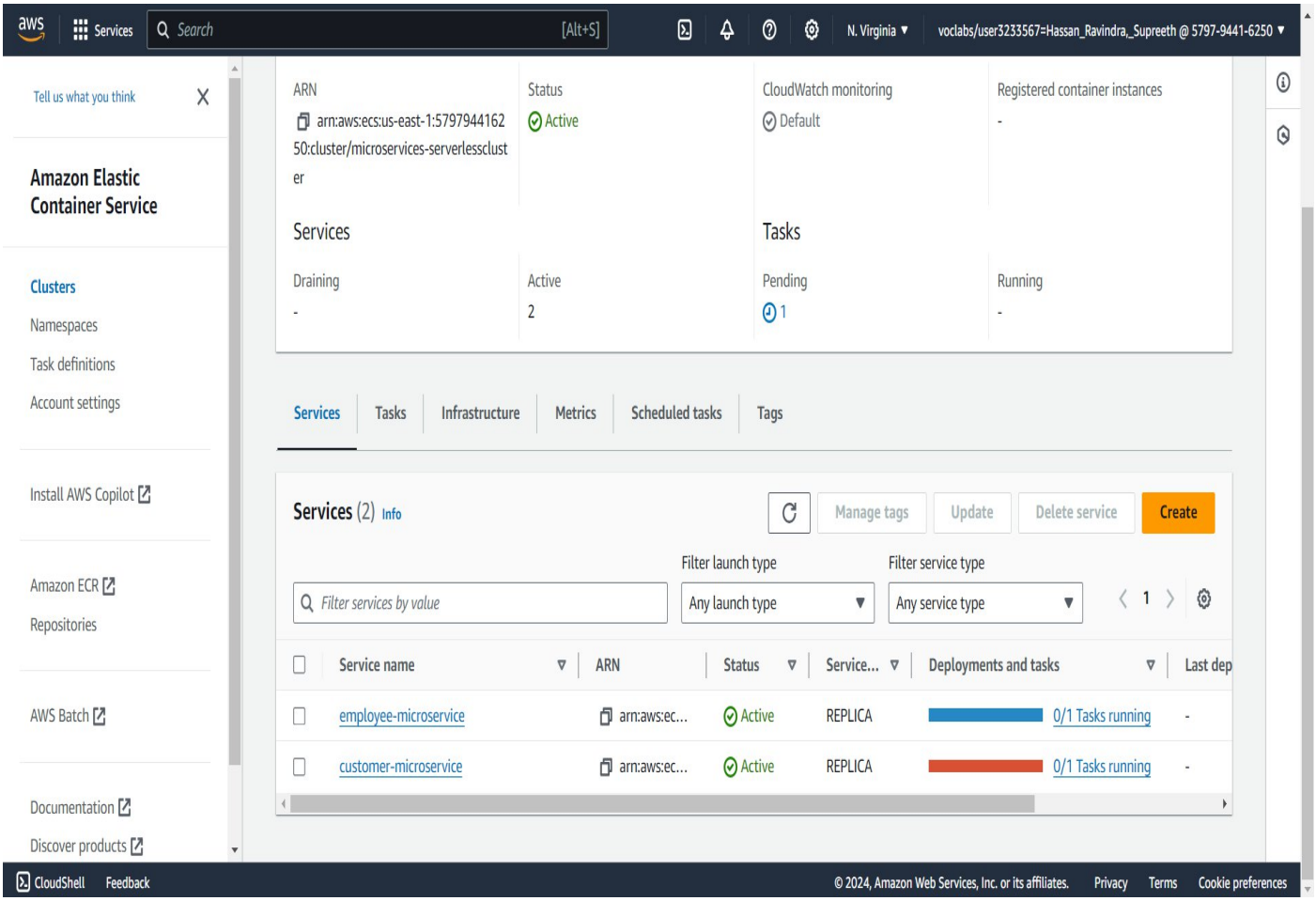
- Task 7.2: I copied the JSON file created for the customer microservice and named it `create-employee-microservice-tg-two.json`, saving it in the same deployment directory.
- In the `create-employee-microservice-tg-two.json` file I made the following modifications:
- On line 2, I changed `customer-microservice` to `employee-microservice` updating the revision number to reflect the latest for the employee task definition.
- On line 6, I entered the ARN of the `employee-tg-two` target group. (It's crucial to use the correct ARN, not just replace "customer" with "employee.")
- On line 7, I changed `customer` to `employee`.
- After saving the changes, I navigated to the deployment directory and ran the appropriate AWS CLI command to create the ECS service for the employee microservice. The command to create the ECS service was:

  `aws ecs create-service --service-name employee-microservice --cli-input-json file://create-employee-microservice-tg-two.json`.

- These steps created the ECS service for the employee microservice, allowing for deployment and traffic routing to the correct target group in a blue/green deployment environment.

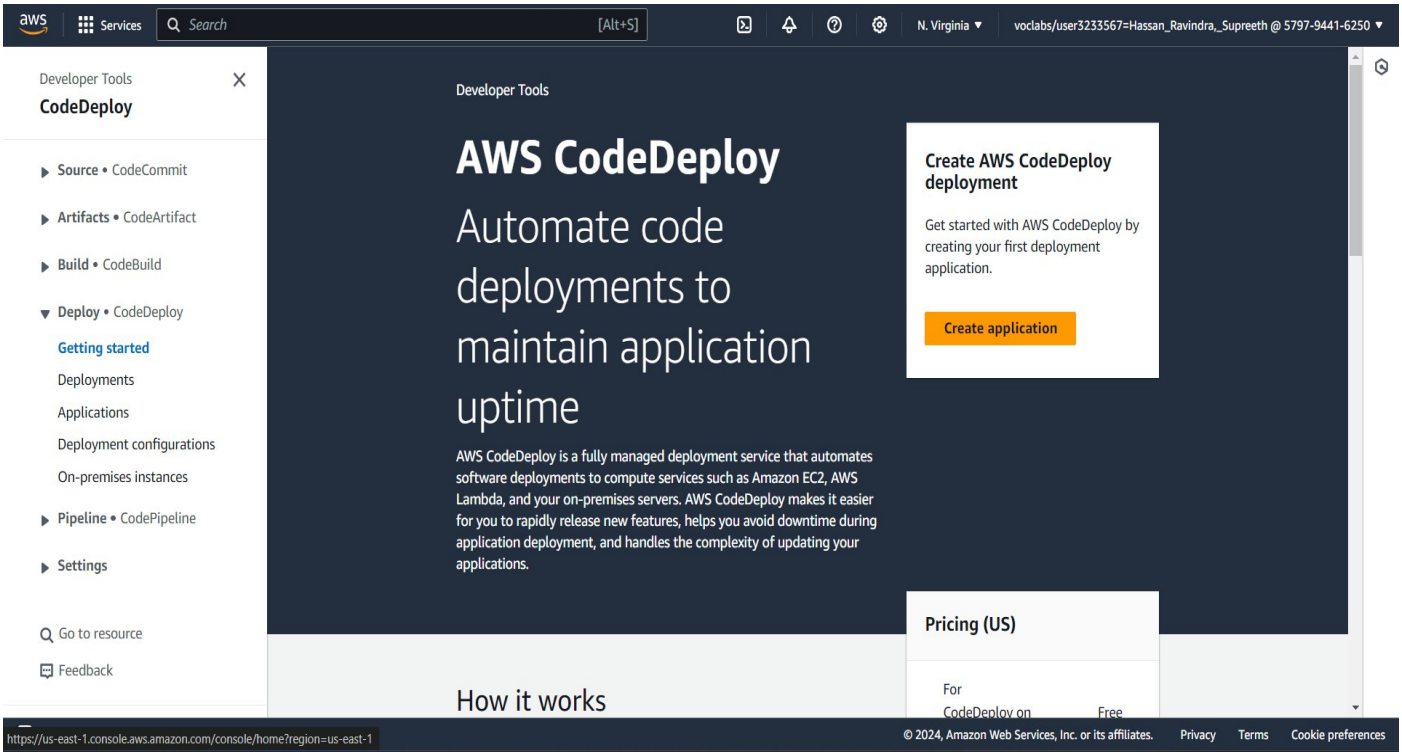


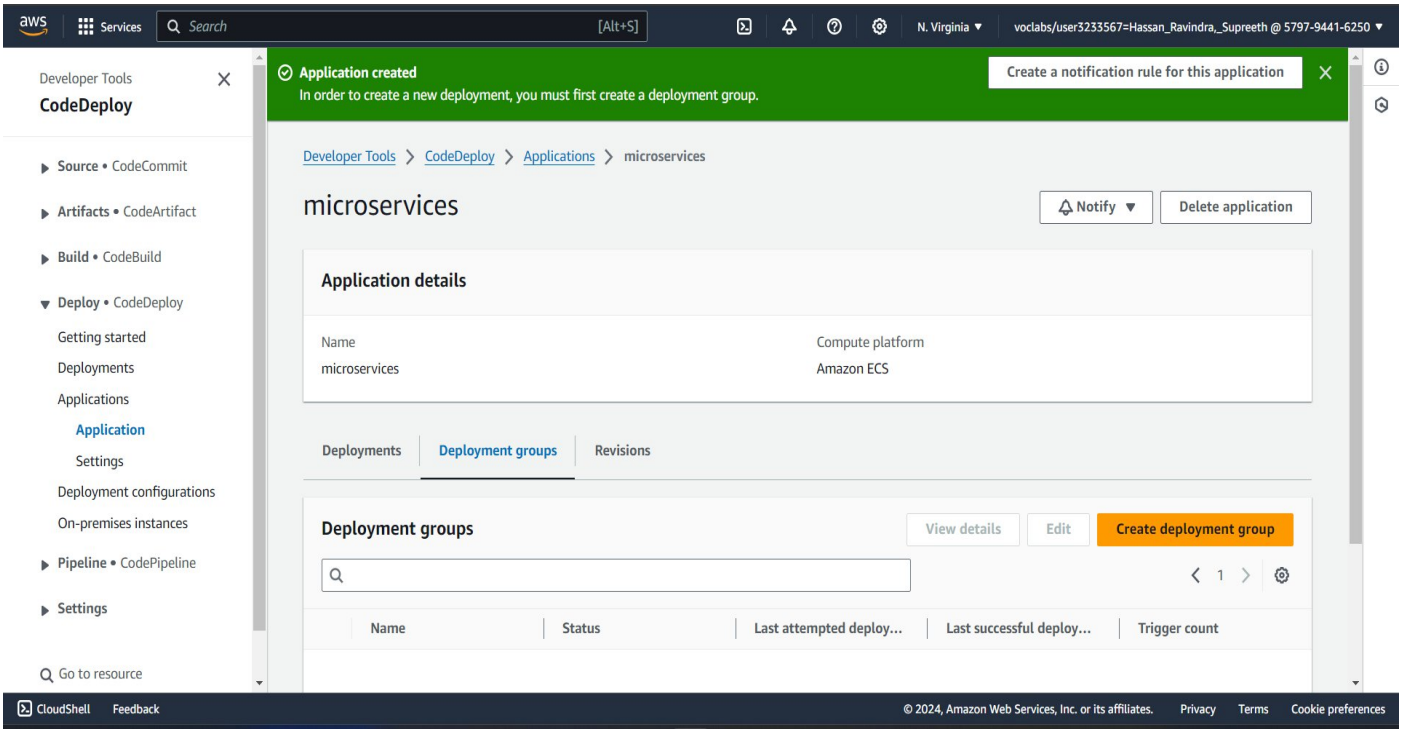

File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
create-employee-micr
aws - "ip-10-16-10-71.ec2
MicroservicesIDE - /home/
deployment
appspec-customer.yaml
appspec-employee.yaml
create-customer-microservice
create-employee-microservic
taskdef-customer.json
taskdef-employee.json
microservices
labsuser.pem
README.md
"unit": "PERCENT"
"stabilityStatus": "STABILIZING",
"stabilityStatusAt": "2024-04-27T20:39:29.127000+00:00",
"tags": []
"roleArn": "arn:aws:iam::579794416250:role/aws-service-role/ecs.amazonaws.com/AWSServiceRoleForECS",
"events": [],
"createdAt": "2024-04-27T20:39:29.127000+00:00",
"placementConstraints": [],
"placementStrategy": [],
"networkConfiguration": {
"awsvpcConfiguration": {
"subnets": [
"subnet-000a87838db6561c0",
"subnet-079b846921b332a35"
"securityGroups": [
"sg-03bc4d5baf12c5aa5"
"assignPublicIp": "ENABLED"
"healthCheckGracePeriodSeconds": 0,
"schedulingStrategy": "REPLICA",
"deploymentController": {
"type": "CODE_DEPLOY"
"createdBy": "arn:aws:iam::579794416250:role/voclabs",
"enableECSManagedTags": false,
"propagateTags": "NONE",
"enableExecuteCommand": false
(END)
dev*
CodeWhisperer
AWS: profile:default

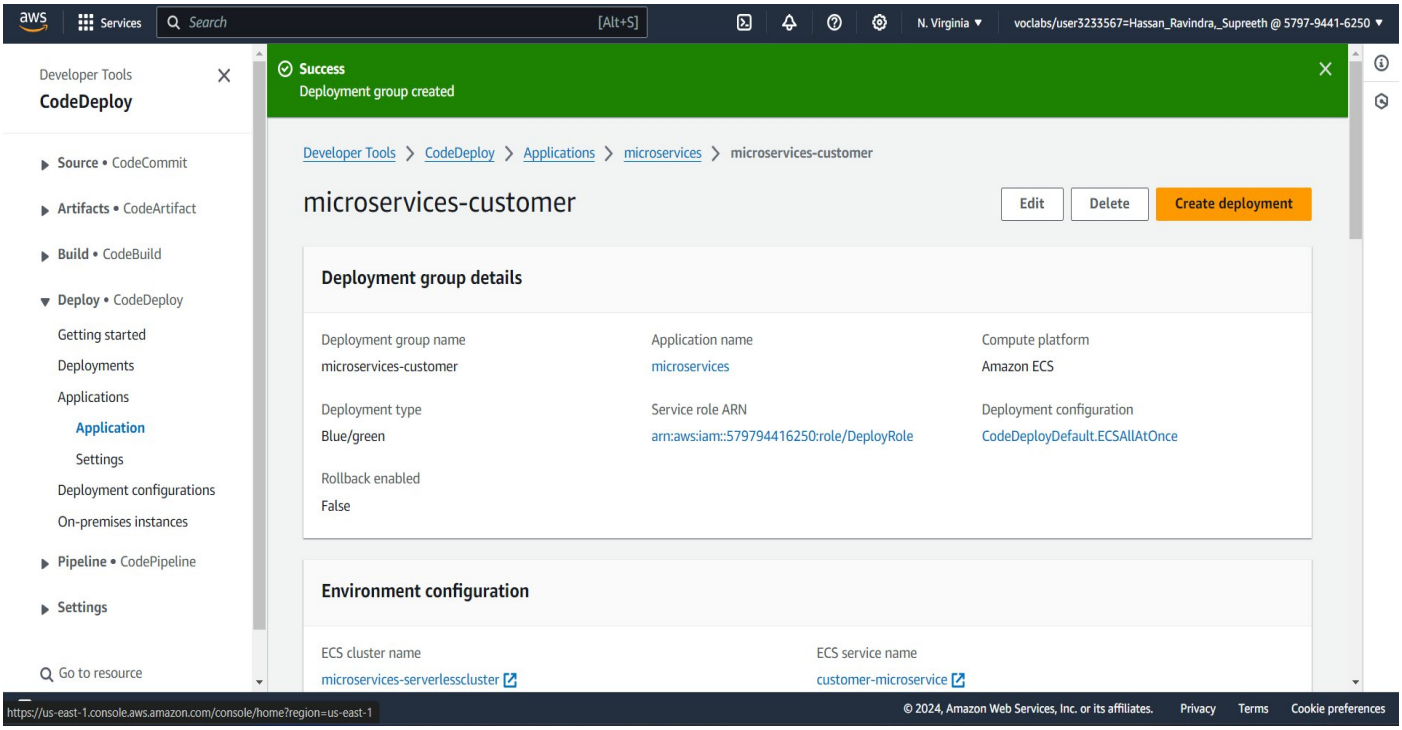

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Tell us what you think
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Install AWS Copilot
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
ARN
arn:aws:ecs:us-east-1:579794416250:cluster/microservices-serverlesscluster
Status
Active
CloudWatch monitoring
Default
Registered container instances
-
Services
Draining
-
Active
2
Tasks
Pending
1
Running
-
Services
Tasks
Infrastructure
Metrics
Scheduled tasks
Tags
Services (2) Info
Manage tags
Update
Delete service
Create
Filter launch type
Any launch type
Filter service type
Any service type
Filter services by value
Service name
ARN
Status
Service...
Deployments and tasks
Last dep
employee-microservice
arn:aws:ec...
Active
REPLICA
0/1 Tasks running
customer-microservice
arn:aws:ec...
Active
REPLICA
0/1 Tasks running
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

PHASE-8: Configuring CodeDeploy and CodePipeline

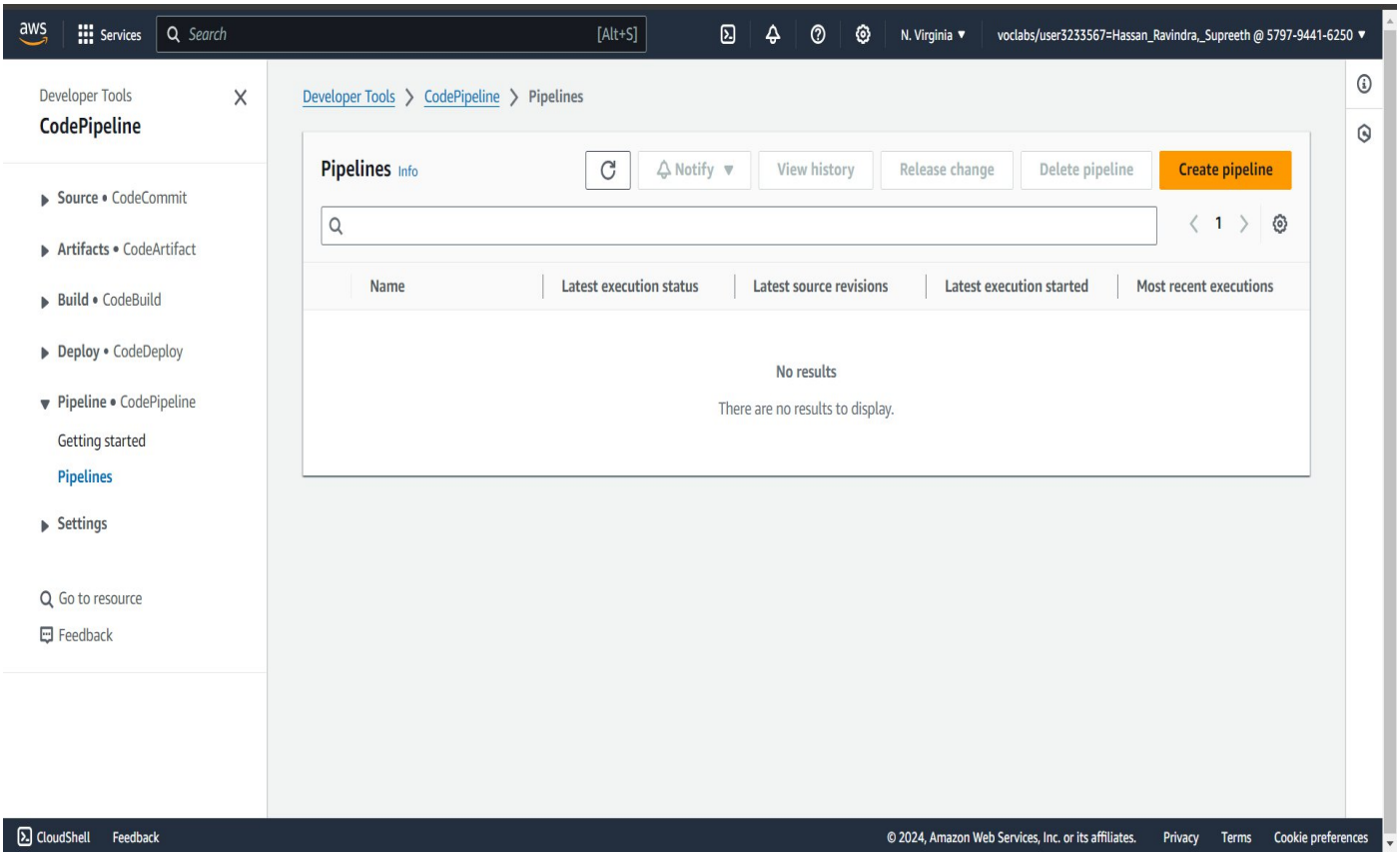
Task 8.1: I used the CodeDeploy console to create a CodeDeploy application with the name microservices that uses Amazon ECS as the compute platform and created a CodeDeploy deployment group for the customer and employee microservices with the configurations given in the lab instructions.

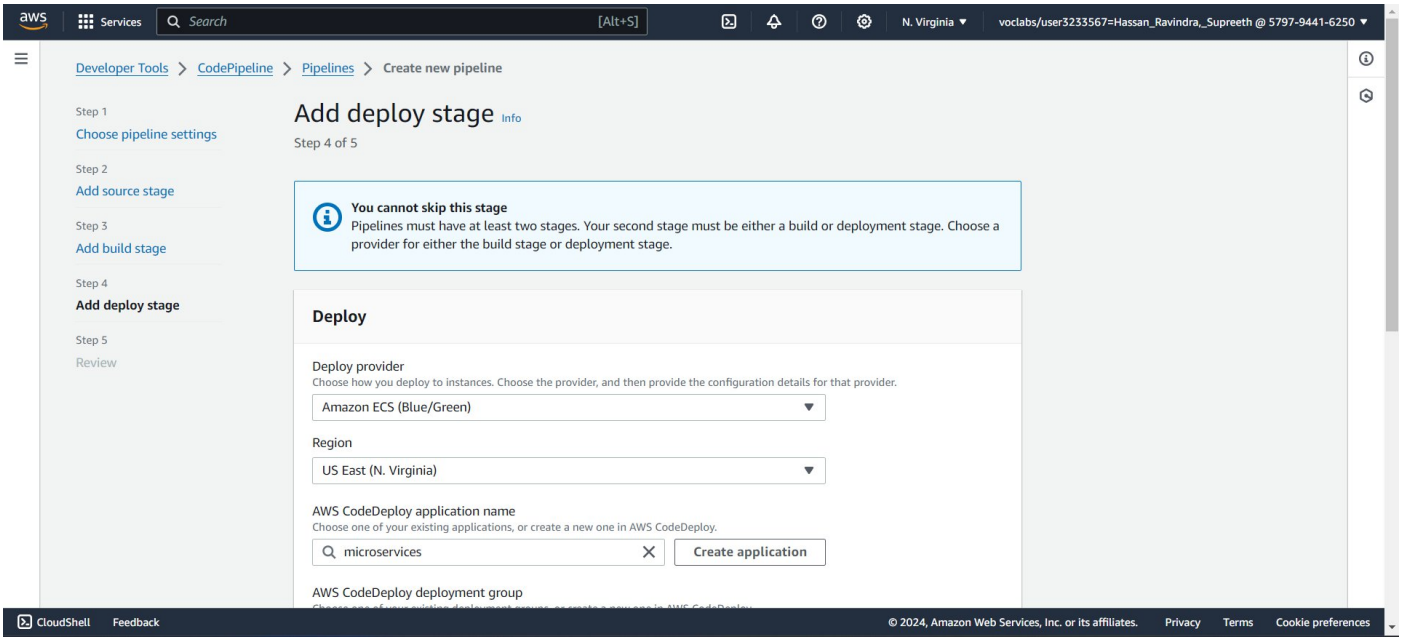


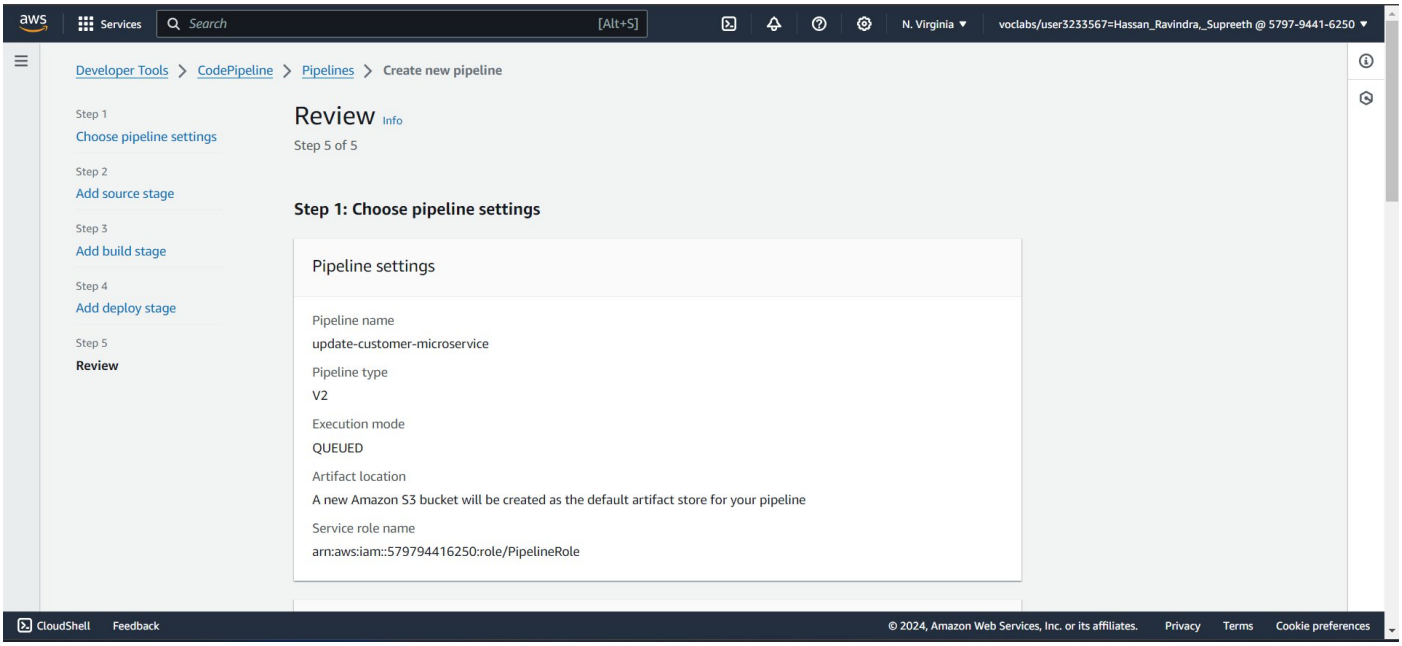


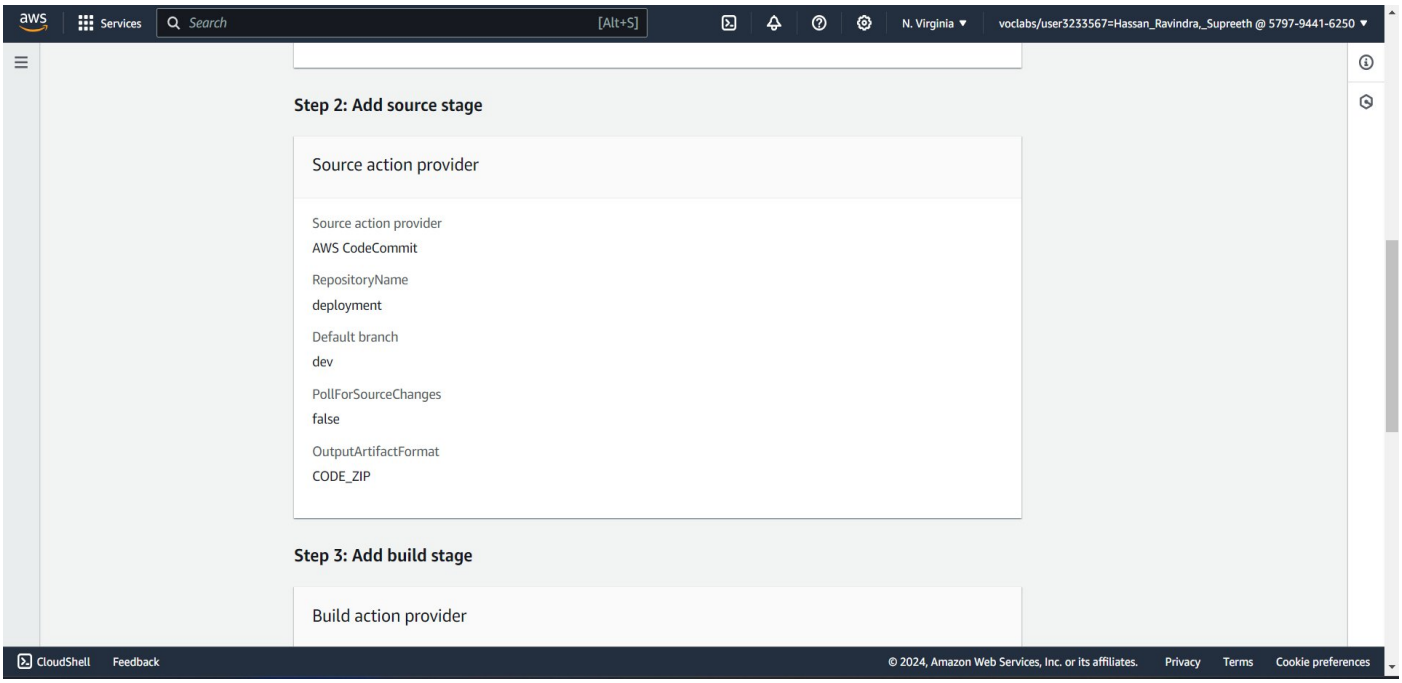


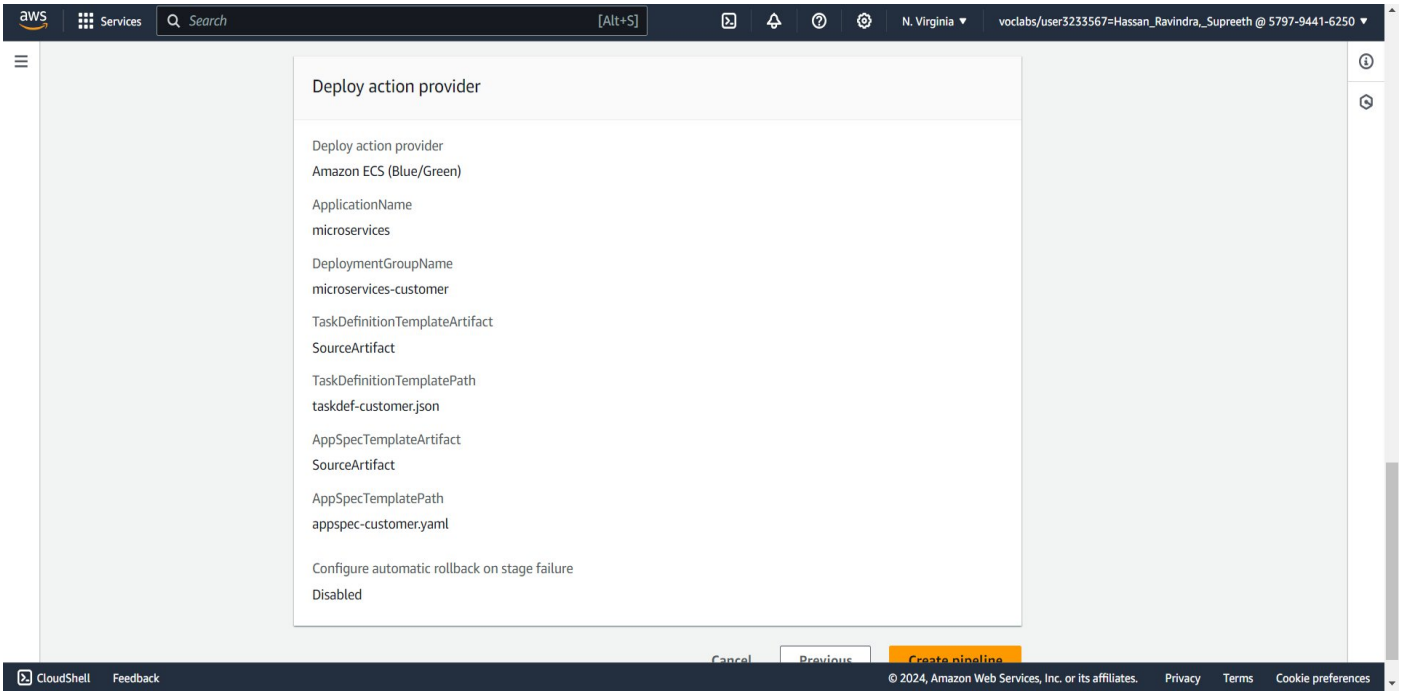
- Task 8.2: To create a pipeline for the customer microservice I started by creating a CodePipeline named "update-customer-microservice" with the settings given in the instruction. After creating the pipeline, I edited it to add another source action for Amazon ECR with the settings given in the instruction. I edited the deploy action to link the Amazon ECR source to the pipeline. These steps set up a CodePipeline for the customer microservice with source, deploy, and image artifact configurations to enable Amazon ECS blue/green deployments.

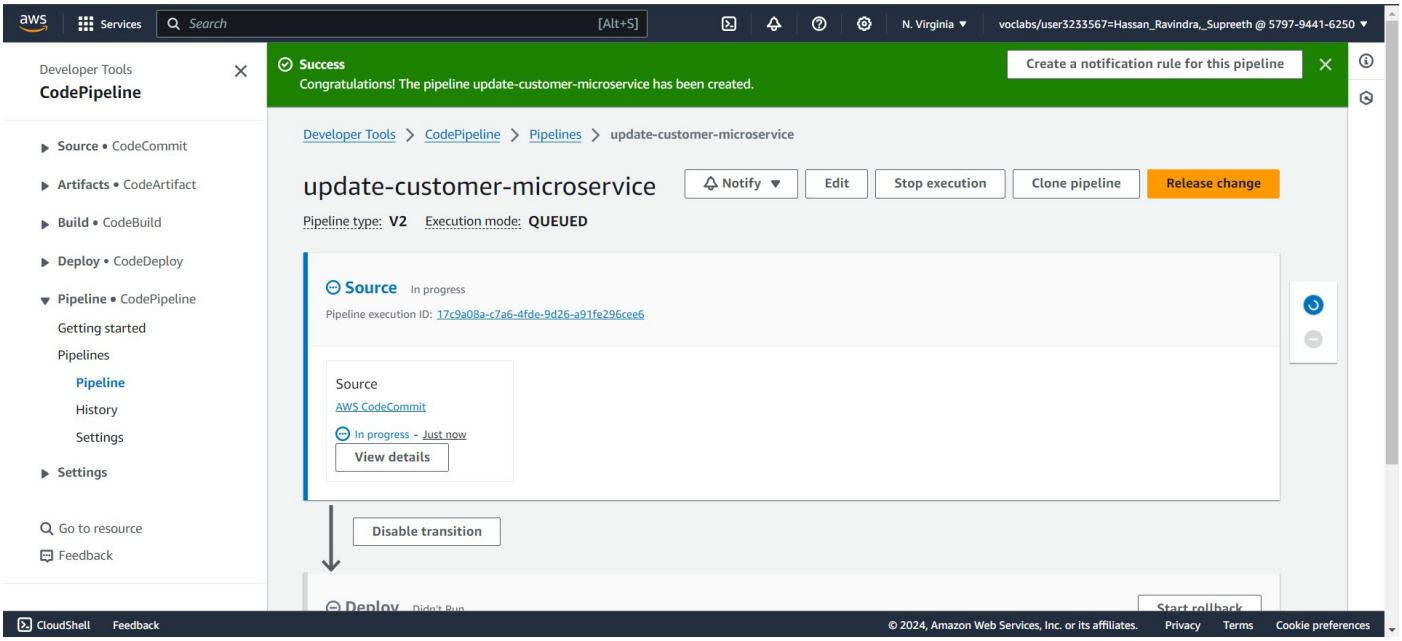


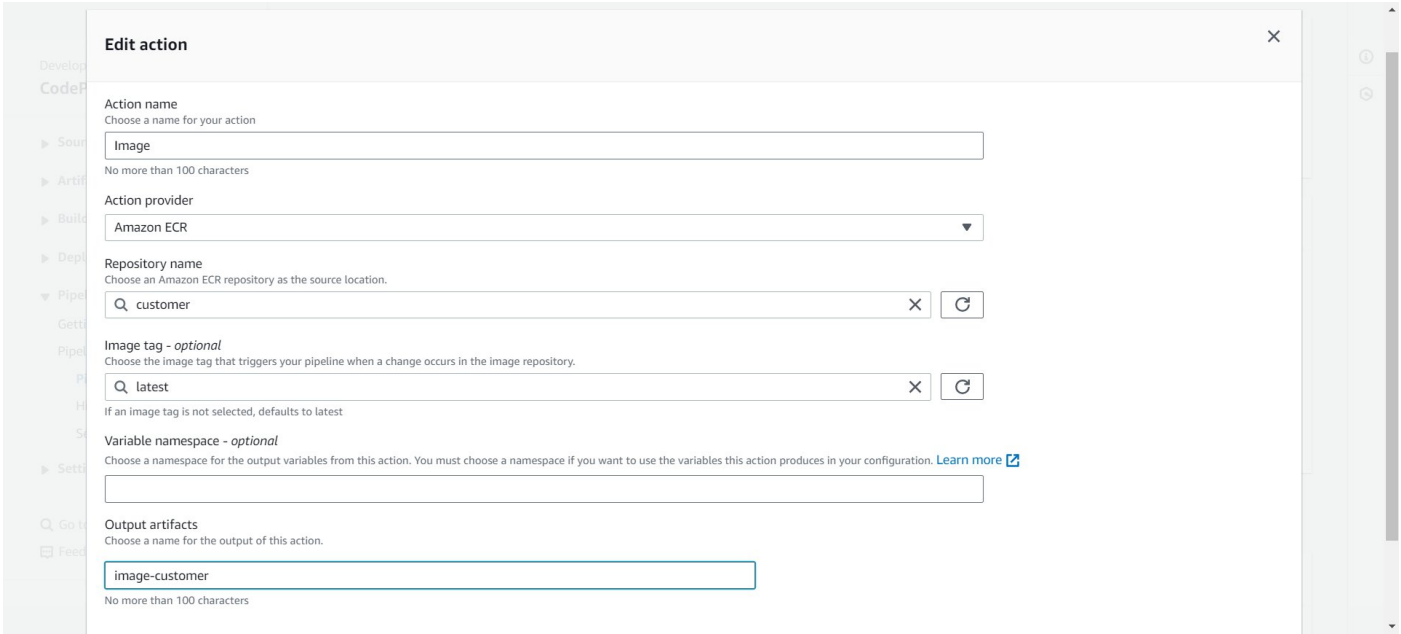

Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Add deploy stage Info
Step 4 of 5
You cannot skip this stage
Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.
Deploy
Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.
Amazon ECS (Blue/Green)
Region
US East (N. Virginia)
AWS CodeDeploy application name
Choose one of your existing applications, or create a new one in AWS CodeDeploy.
microservices
Create application
AWS CodeDeploy deployment group
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

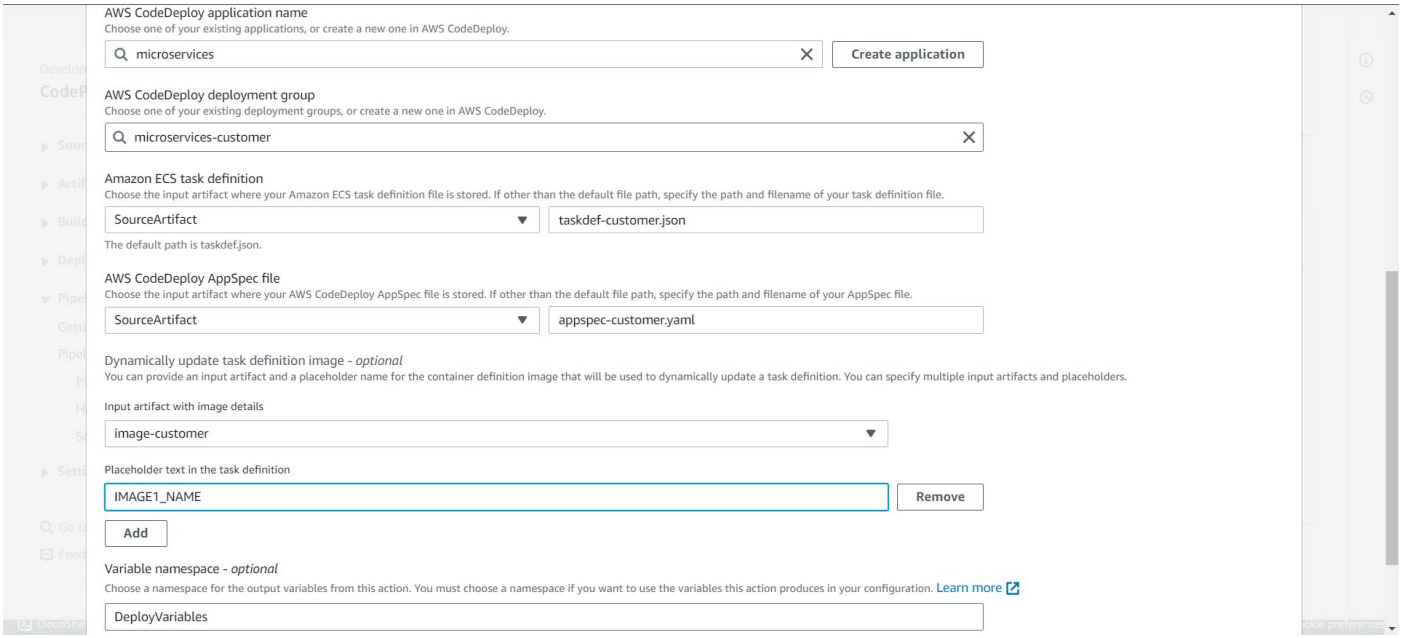

Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Review Info
Step 5 of 5
Step 1: Choose pipeline settings
Pipeline settings
Pipeline name
update-customer-microservice
Pipeline type
V2
Execution mode
QUEUED
Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name
arn:aws:iam::579794416250:role/PipelineRole
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Step 2: Add source stage
Source action provider
Source action provider
AWS CodeCommit
RepositoryName
deployment
Default branch
dev
PollForSourceChanges
false
OutputArtifactFormat
CODE_ZIP
Step 3: Add build stage
Build action provider
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

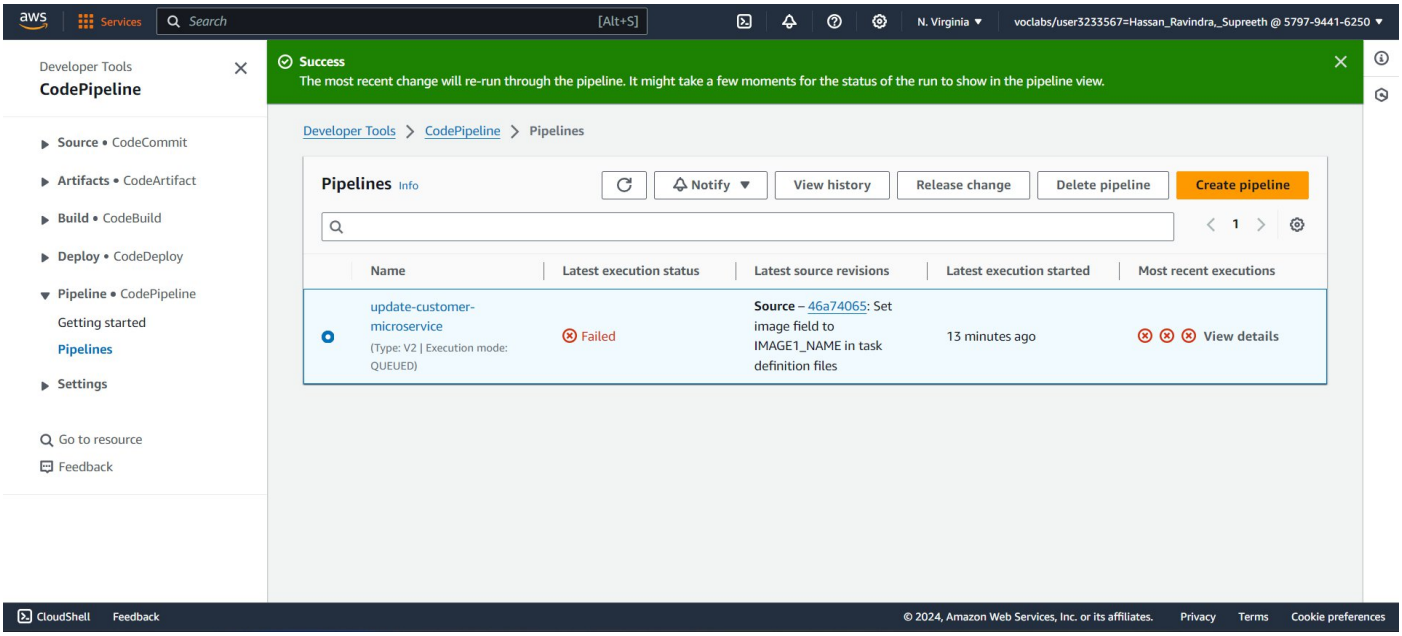

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Deploy action provider
Deploy action provider
Amazon ECS (Blue/Green)
ApplicationName
microservices
DeploymentGroupName
microservices-customer
TaskDefinitionTemplateArtifact
SourceArtifact
TaskDefinitionTemplatePath
taskdef-customer.json
AppSpecTemplateArtifact
SourceArtifact
AppSpecTemplatePath
appspec-customer.yaml
Configure automatic rollback on stage failure
Disabled
Cancel
Previous
Create pipeline
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

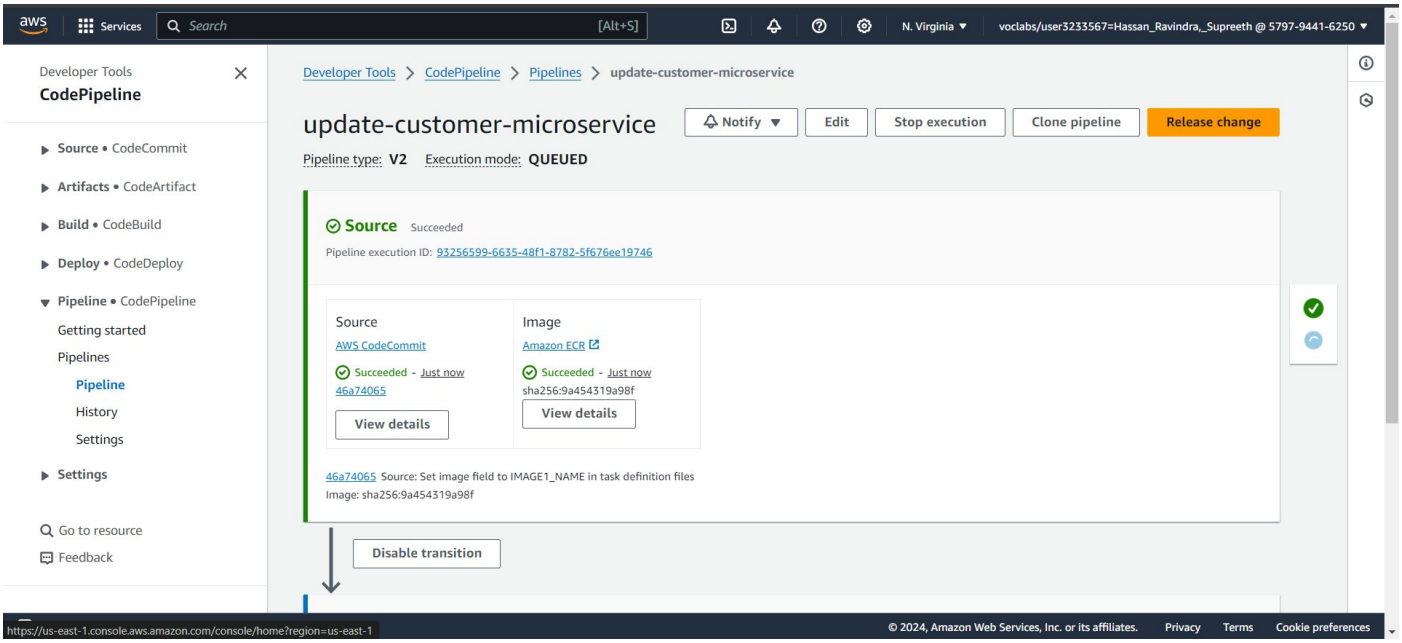

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Success
Congratulations! The pipeline update-customer-microservice has been created.
Create a notification rule for this pipeline
Developer Tools > CodePipeline > Pipelines > update-customer-microservice
update-customer-microservice
Notify
Edit
Stop execution
Clone pipeline
Release change
Pipeline type: V2
Execution mode: QUEUED
Source
In progress
Pipeline execution ID: 17c9a08a-c7a6-4fde-9d26-a91fe296cee6
Source
AWS CodeCommit
In progress - Just now
View details
Disable transition
Deploy
Didn't Run
Start rollback
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

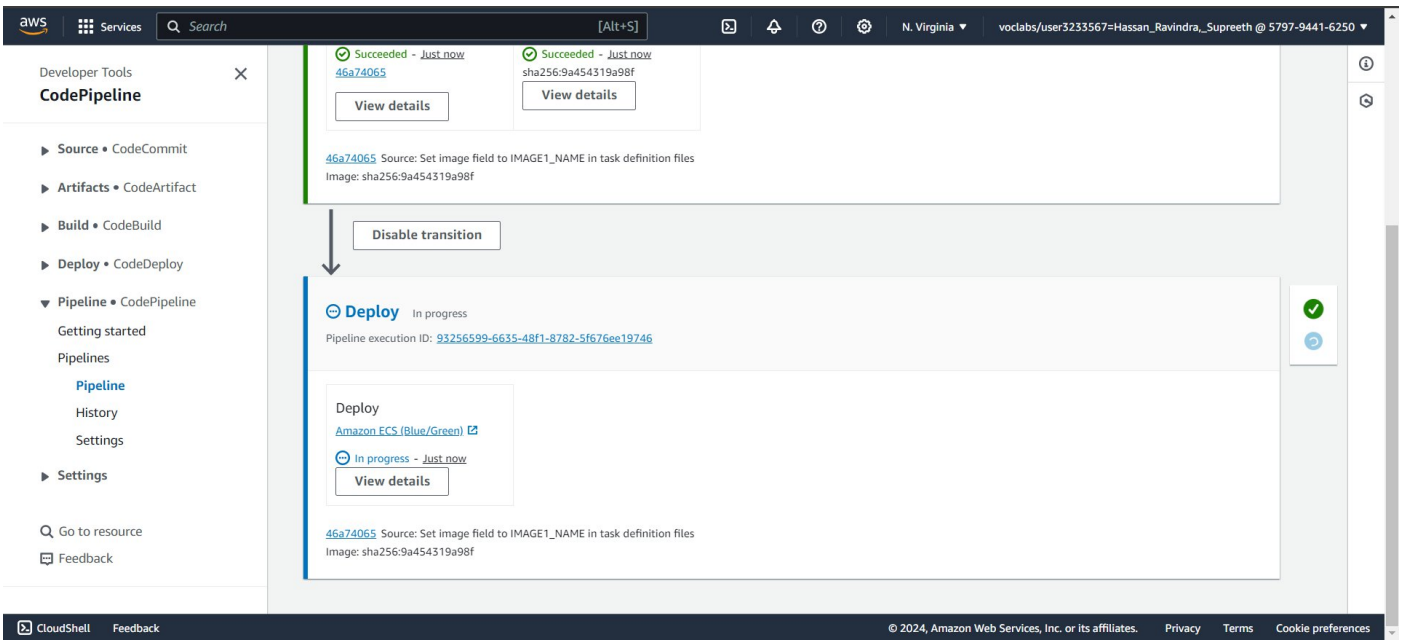

Edit action
Action name
Choose a name for your action
Image
No more than 100 characters
Action provider
Amazon ECR
Repository name
Choose an Amazon ECR repository as the source location.
customer
Image tag - optional
Choose the image tag that triggers your pipeline when a change occurs in the image repository.
latest
If an image tag is not selected, defaults to latest
Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. Learn more
Output artifacts
Choose a name for the output of this action.
image-customer
No more than 100 characters

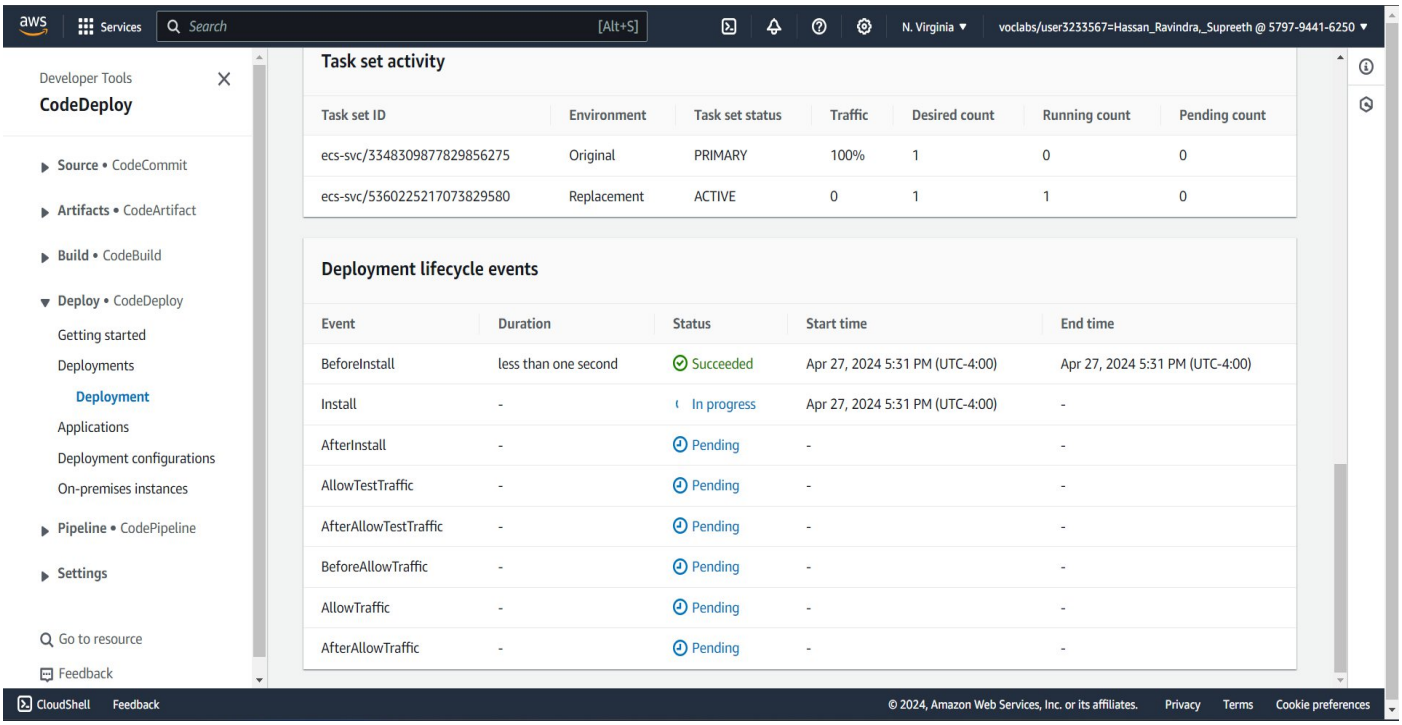


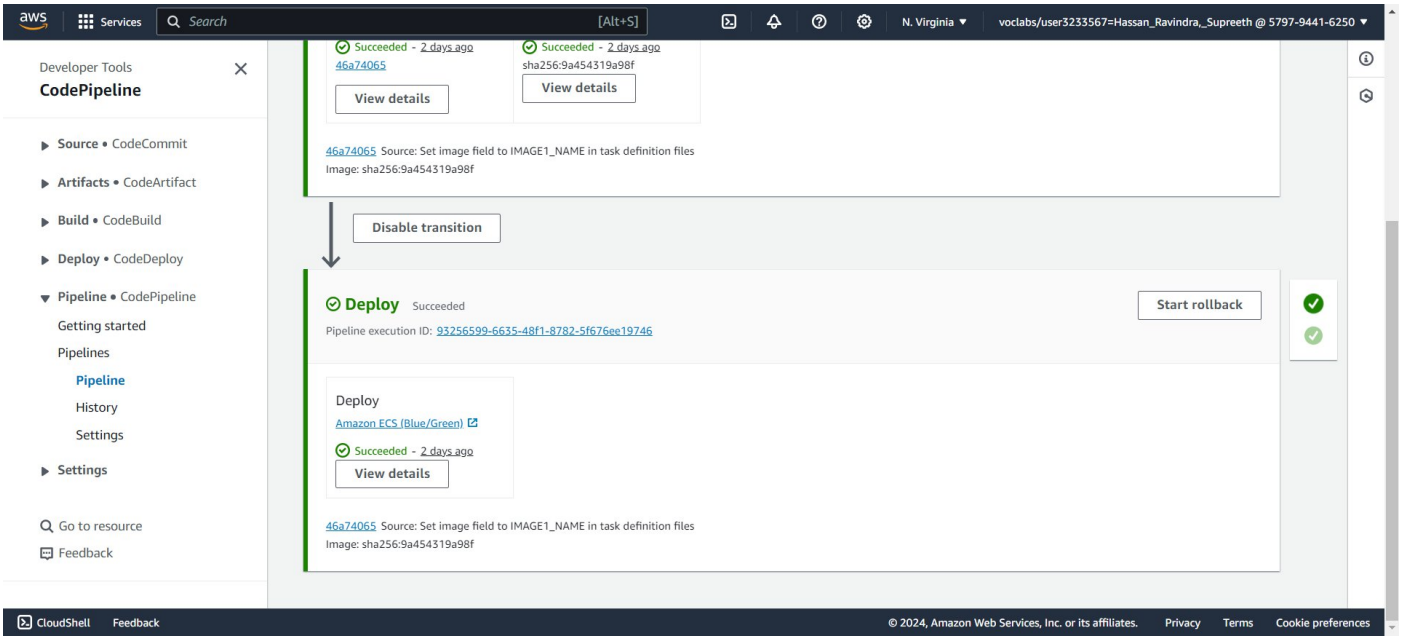
- Task 8.3: To test the CI/CD pipeline for the customer microservice, I launched a deployment on Amazon ECS with Fargate. In CodePipeline, I forced a test on the "update-customer-microservice" pipeline by selecting "Release change," triggering a new revision.
- After the source tasks showed "Succeeded," I checked the deployment progress on the CodeDeploy page and ensured that all deployment lifecycle events succeeded.
- I found the load balancer's DNS name, pasted it into a browser, and tested the microservice. If it didn't load, I added `:8080` to the URL, considering blue/green deployment transitions. I tested the suppliers list to ensure it lacked "edit" or "add" buttons, confirming it was read-only.
- In Amazon ECS, I checked the customer-microservice's status, running tasks, and container details. In Amazon EC2, I verified that the load balancer and target groups were properly configured, ensuring the HTTP:80 and HTTP:8080 listener rules routed correctly.
- These steps confirmed that the CI/CD pipeline for the customer microservice successfully deployed and functioned as expected on Amazon ECS with Fargate.

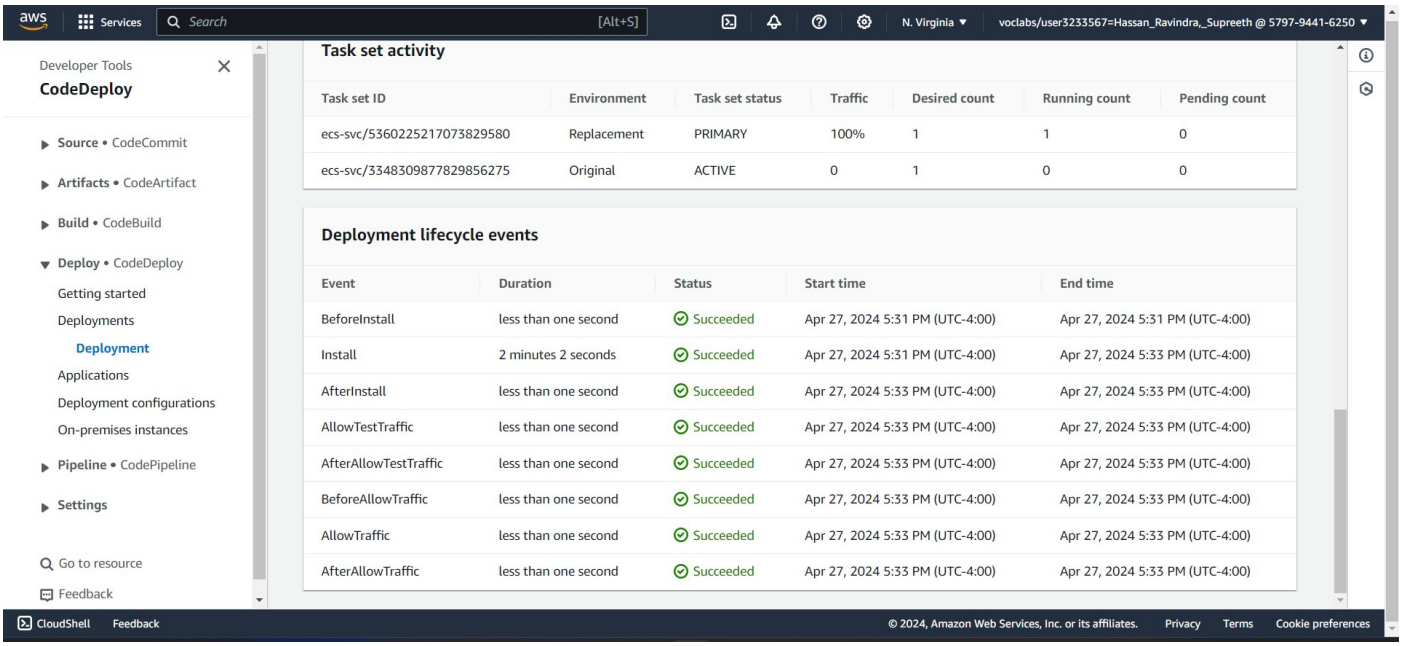


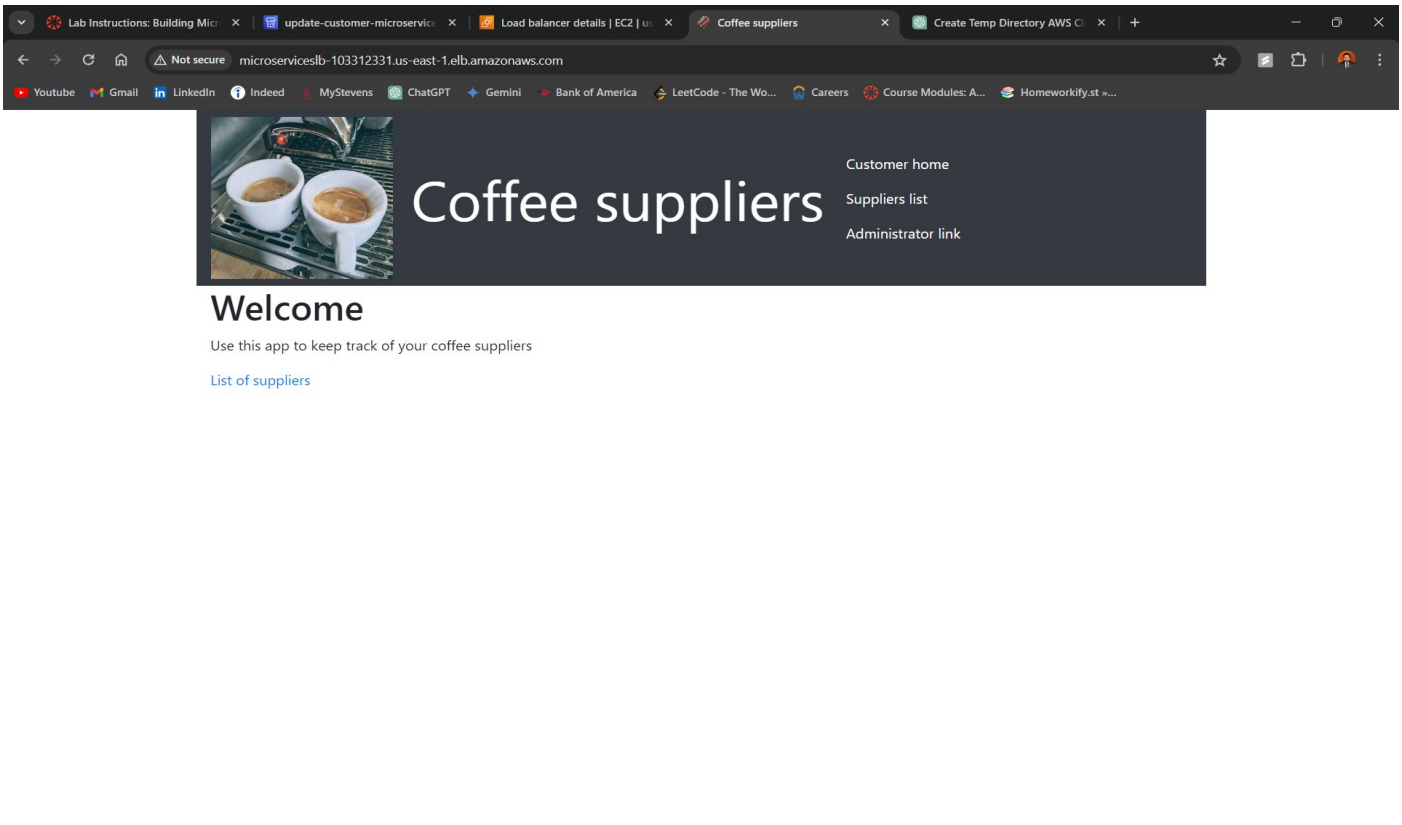

Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Developer Tools > CodePipeline > Pipelines > update-customer-microservice
update-customer-microservice
Notify
Edit
Stop execution
Clone pipeline
Release change
Pipeline type: V2
Execution mode: QUEUED
Source Succeeded
Pipeline execution ID: 93256599-6635-48f1-8782-5f676ee19746
Source
AWS CodeCommit
Succeeded - Just now
46a74065
View details
Image
Amazon ECR
Succeeded - Just now
sha256:9a454319a98f
View details
46a74065 Source: Set image field to IMAGE1_NAME in task definition files
Image: sha256:9a454319a98f
Disable transition
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

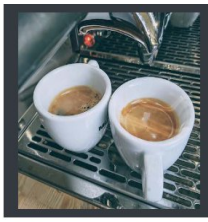

Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Succeeded - Just now
46a74065
View details
Succeeded - Just now
sha256:9a454319a98f
View details
46a74065 Source: Set image field to IMAGE1_NAME in task definition files
Image: sha256:9a454319a98f
Disable transition
Deploy In progress
Pipeline execution ID: 93256599-6635-48f1-8782-5f676ee19746
Deploy
Amazon ECS (Blue/Green)
In progress - Just now
View details
46a74065 Source: Set image field to IMAGE1_NAME in task definition files
Image: sha256:9a454319a98f
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Deployment
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Task set activity
Task set ID | Environment | Task set status | Traffic | Desired count | Running count | Pending count
ecs-svc/3348309877829856275 | Original | PRIMARY | 100% | 1 | 0 | 0
ecs-svc/5360225217073829580 | Replacement | ACTIVE | 0 | 1 | 1 | 0
Deployment lifecycle events
Event | Duration | Status | Start time | End time
BeforeInstall | less than one second | Succeeded | Apr 27, 2024 5:31 PM (UTC-4:00) | Apr 27, 2024 5:31 PM (UTC-4:00)
Install | - | In progress | Apr 27, 2024 5:31 PM (UTC-4:00) | -
AfterInstall | - | Pending | - | -
AllowTestTraffic | - | Pending | - | -
AfterAllowTestTraffic | - | Pending | - | -
BeforeAllowTraffic | - | Pending | - | -
AllowTraffic | - | Pending | - | -
AfterAllowTraffic | - | Pending | - | -
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Succeeded - 2 days ago
46a74065
View details
Succeeded - 2 days ago
sha256:9a454319a98f
View details
46a74065 Source: Set image field to IMAGE1_NAME in task definition files
Image: sha256:9a454319a98f
Disable transition
Deploy Succeeded
Start rollback
Pipeline execution ID: 93256599-6635-48f1-8782-5f676ee19746
Deploy
Amazon ECS (Blue/Green)
Succeeded - 2 days ago
View details
46a74065 Source: Set image field to IMAGE1_NAME in task definition files
Image: sha256:9a454319a98f
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Deployment
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Task set activity
Task set ID
Environment
Task set status
Traffic
Desired count
Running count
Pending count
ecs-svc/5360225217073829580
Replacement
PRIMARY
100%
1
1
0
ecs-svc/3348309877829856275
Original
ACTIVE
0
1
0
0
Deployment lifecycle events
Event
Duration
Status
Start time
End time
BeforeInstall
less than one second
Succeeded
Apr 27, 2024 5:31 PM (UTC-4:00)
Apr 27, 2024 5:31 PM (UTC-4:00)
Install
2 minutes 2 seconds
Succeeded
Apr 27, 2024 5:31 PM (UTC-4:00)
Apr 27, 2024 5:33 PM (UTC-4:00)
AfterInstall
less than one second
Succeeded
Apr 27, 2024 5:33 PM (UTC-4:00)
Apr 27, 2024 5:33 PM (UTC-4:00)
AllowTestTraffic
less than one second
Succeeded
Apr 27, 2024 5:33 PM (UTC-4:00)
Apr 27, 2024 5:33 PM (UTC-4:00)
AfterAllowTestTraffic
less than one second
Succeeded
Apr 27, 2024 5:33 PM (UTC-4:00)
Apr 27, 2024 5:33 PM (UTC-4:00)
BeforeAllowTraffic
less than one second
Succeeded
Apr 27, 2024 5:33 PM (UTC-4:00)
Apr 27, 2024 5:33 PM (UTC-4:00)
AllowTraffic
less than one second
Succeeded
Apr 27, 2024 5:33 PM (UTC-4:00)
Apr 27, 2024 5:33 PM (UTC-4:00)
AfterAllowTraffic
less than one second
Succeeded
Apr 27, 2024 5:33 PM (UTC-4:00)
Apr 27, 2024 5:33 PM (UTC-4:00)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Not secure
microserviceslb-103312331.us-east-1.elb.amazonaws.com
Coffee suppliers
Customer home
Suppliers list
Administrator link
Welcome
Use this app to keep track of your coffee suppliers
List of suppliers



Coffee suppliers

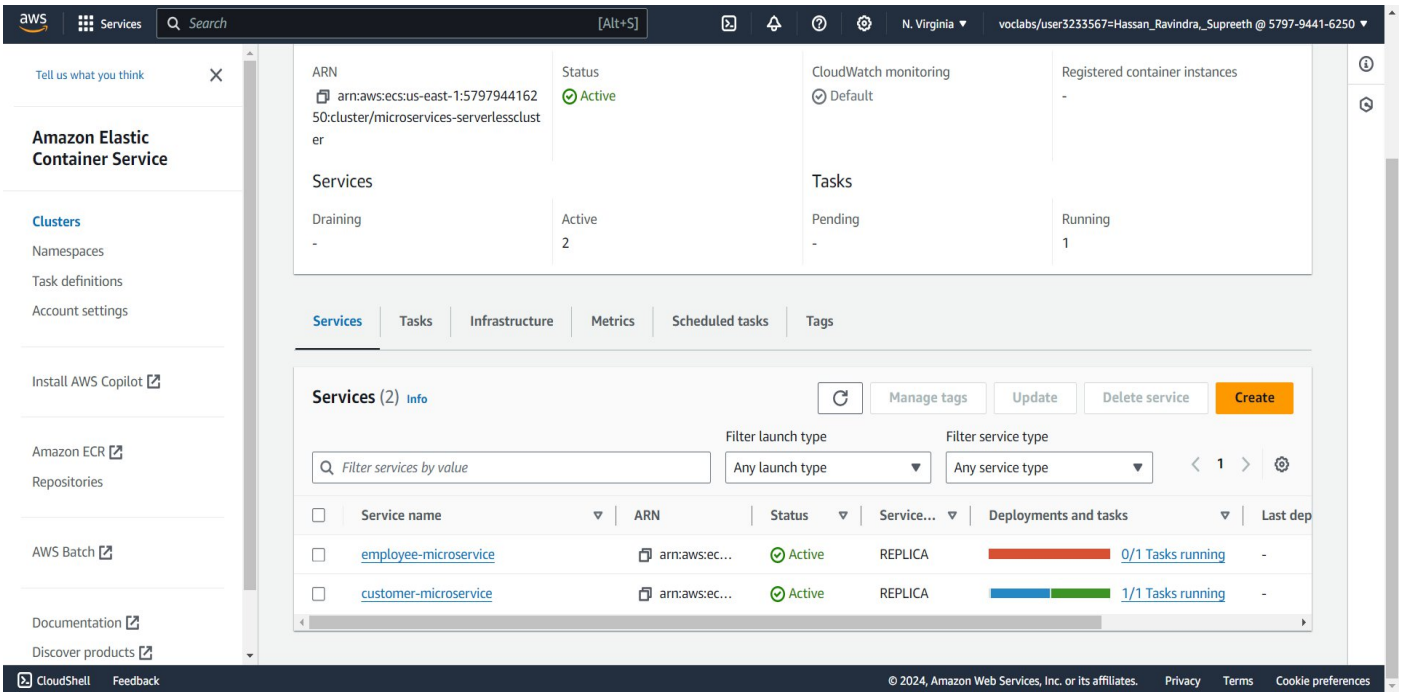
Customer home

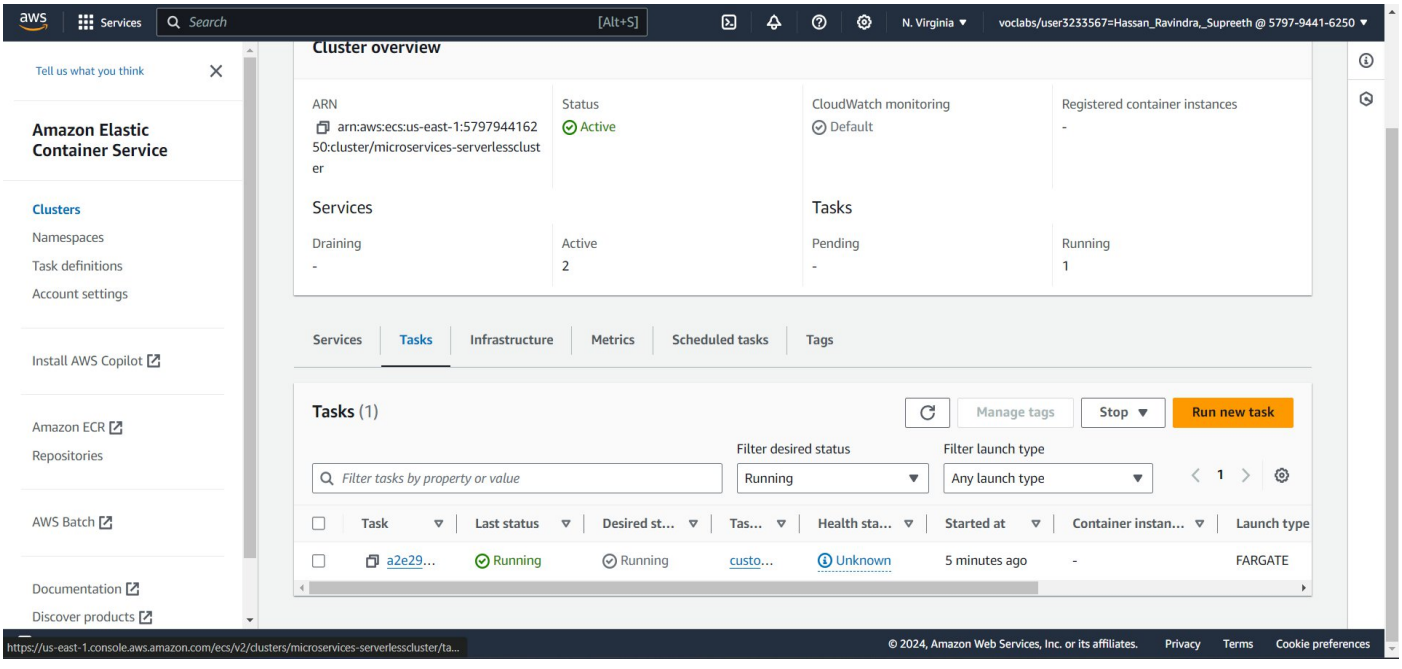
Suppliers list

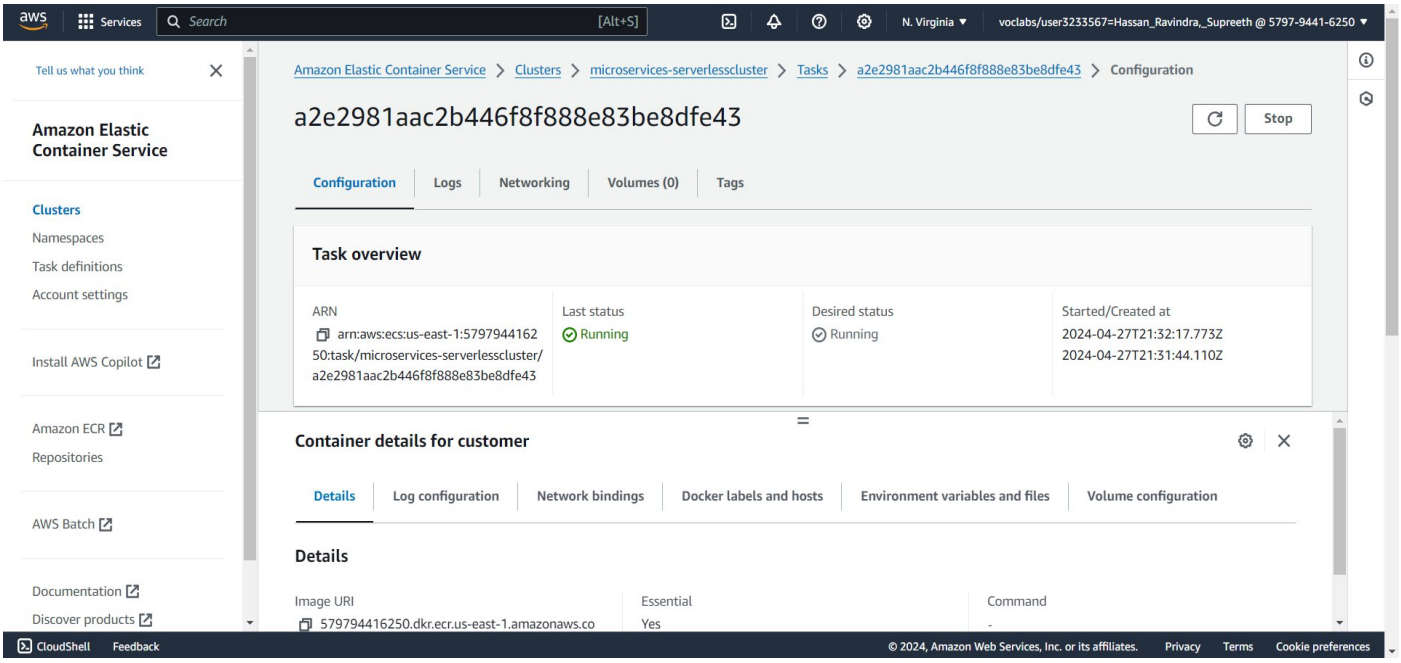
Administrator link

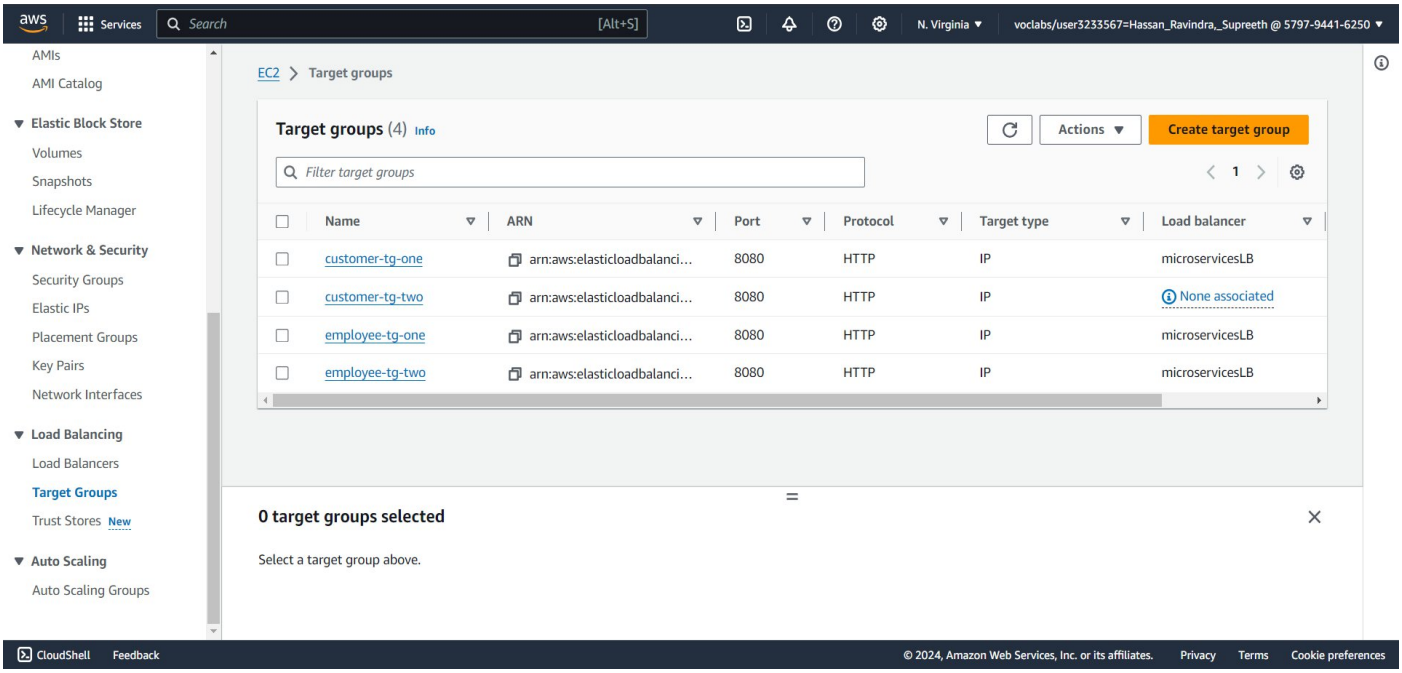
All suppliers

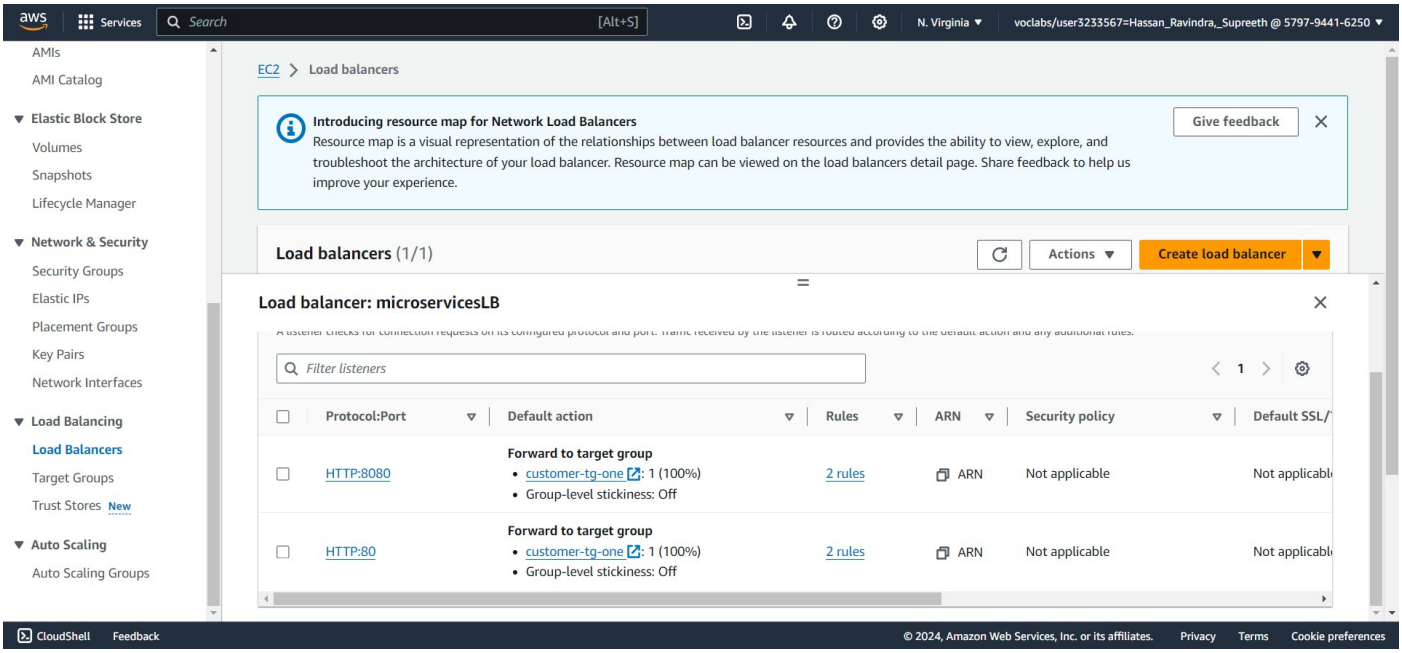
Name	Address	City	State	Email	Phone
Supreeth Hassan Ravindra	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263



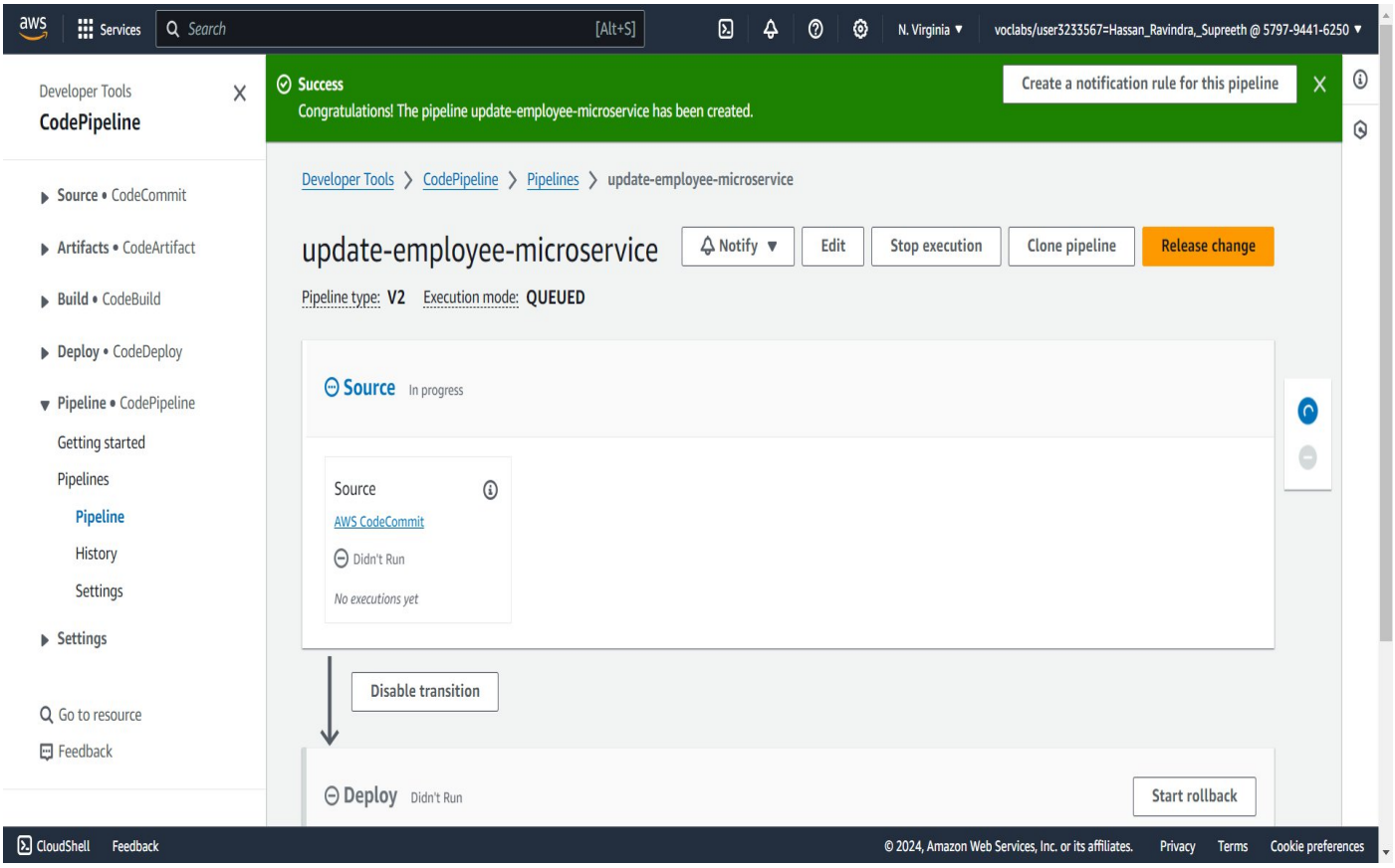


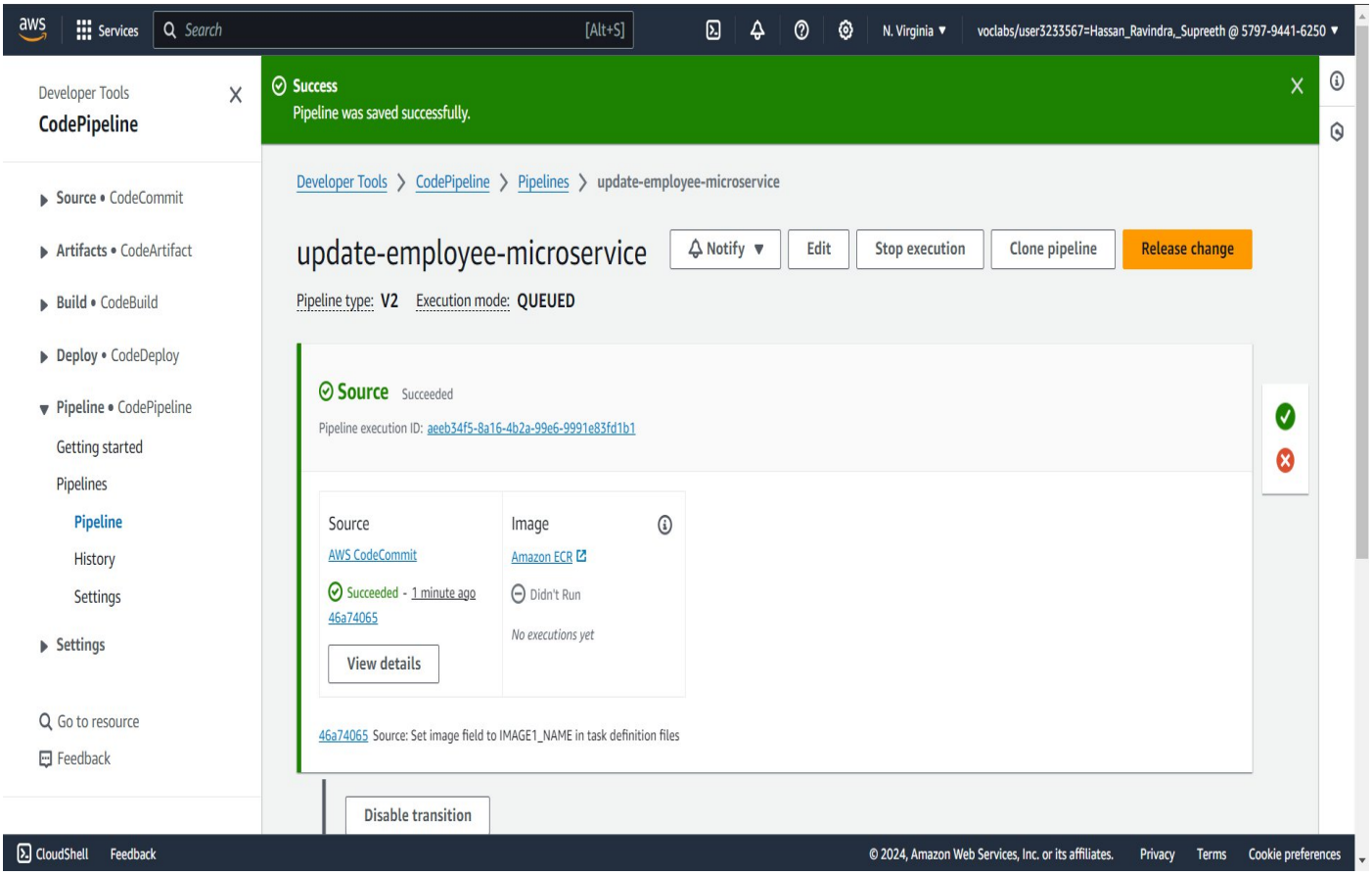

Amazon Elastic Container Service > Clusters > microservices-serverlesscluster > Tasks > a2e2981aac2b446f8f888e83be8dfe43 > Configuration
a2e2981aac2b446f8f888e83be8dfe43
Stop
Configuration
Logs
Networking
Volumes (0)
Tags
Task overview
ARN
arn:aws:ecs:us-east-1:579794416250:task/microservices-serverlesscluster/a2e2981aac2b446f8f888e83be8dfe43
Last status
Running
Desired status
Running
Started/Created at
2024-04-27T21:32:17.773Z
2024-04-27T21:31:44.110Z
Container details for customer
Details
Log configuration
Network bindings
Docker labels and hosts
Environment variables and files
Volume configuration
Image URI
579794416250.dkr.ecr.us-east-1.amazonaws.co
Essential
Yes
Command


EC2 > Target groups
Target groups (4) Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
customer-tg-one
arn:aws:elasticloadbalanci...
8080
HTTP
IP
microservicesLB
customer-tg-two
arn:aws:elasticloadbalanci...
8080
HTTP
IP
None associated
employee-tg-one
arn:aws:elasticloadbalanci...
8080
HTTP
IP
microservicesLB
employee-tg-two
arn:aws:elasticloadbalanci...
8080
HTTP
IP
microservicesLB
0 target groups selected
Select a target group above.

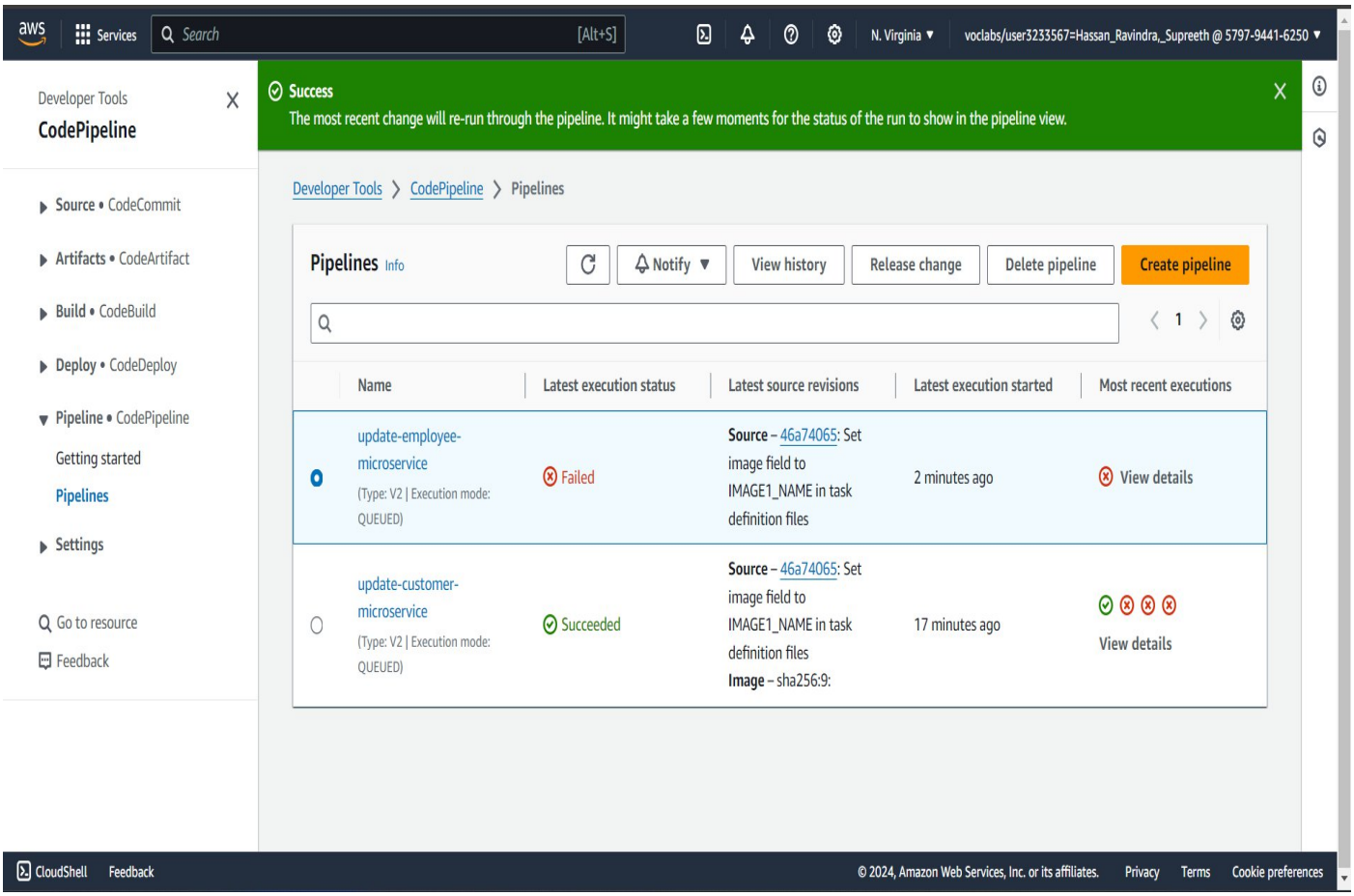

EC2 > Load balancers
Introducing resource map for Network Load Balancers
Resource map is a visual representation of the relationships between load balancer resources and provides the ability to view, explore, and troubleshoot the architecture of your load balancer. Resource map can be viewed on the load balancers detail page. Share feedback to help us improve your experience.
Give feedback
Load balancers (1/1)
Actions
Create load balancer
Load balancer: microservicesLB
Filter listeners
Protocol:Port
Default action
Rules
ARN
Security policy
HTTP:8080
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
2 rules
ARN
Not applicable
HTTP:80
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
2 rules
ARN
Not applicable

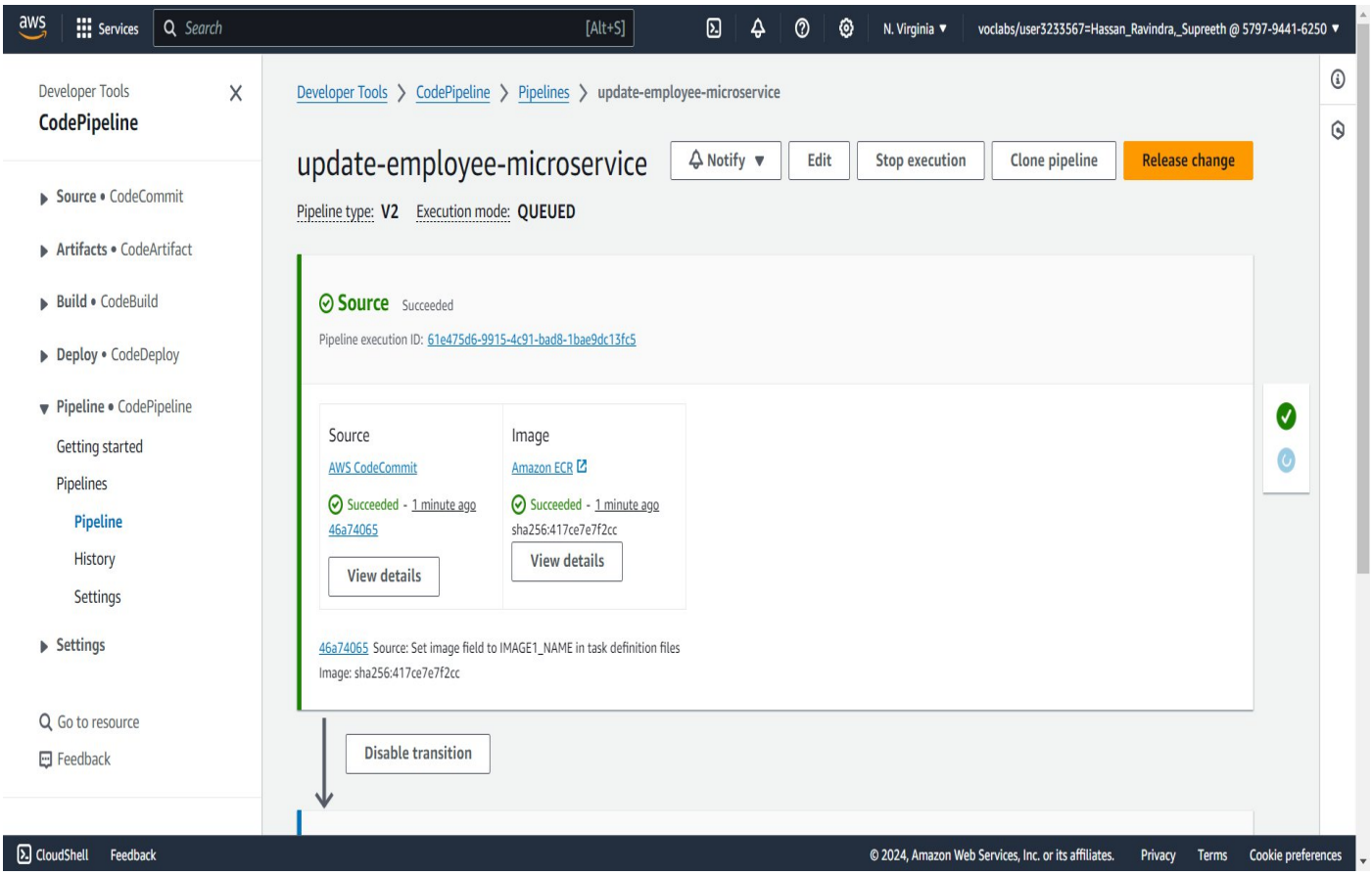
- Task 8.4: Created a pipeline for the employee microservice I configured it with the setting given in the lab instruction in CodePipeline. Named "update-employee-microservice," using PipelineRole ARN and AWS CodeCommit as the source provider (repository: "deployment," branch: "dev").

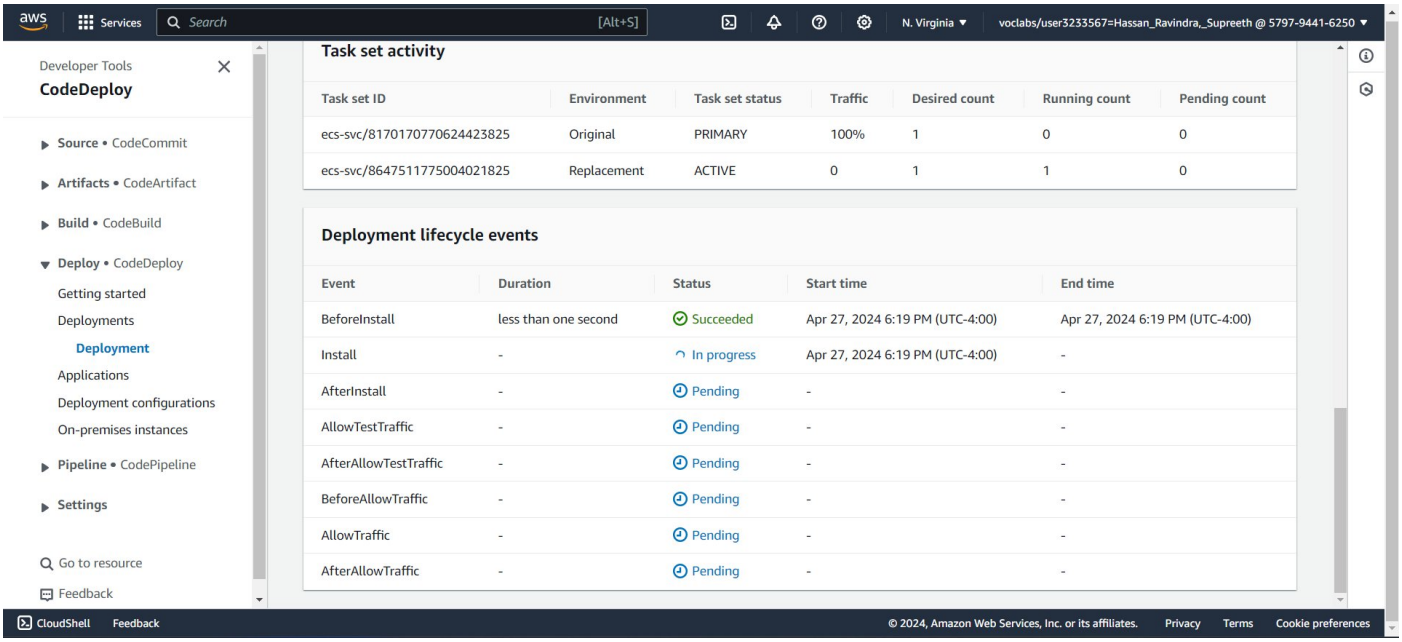


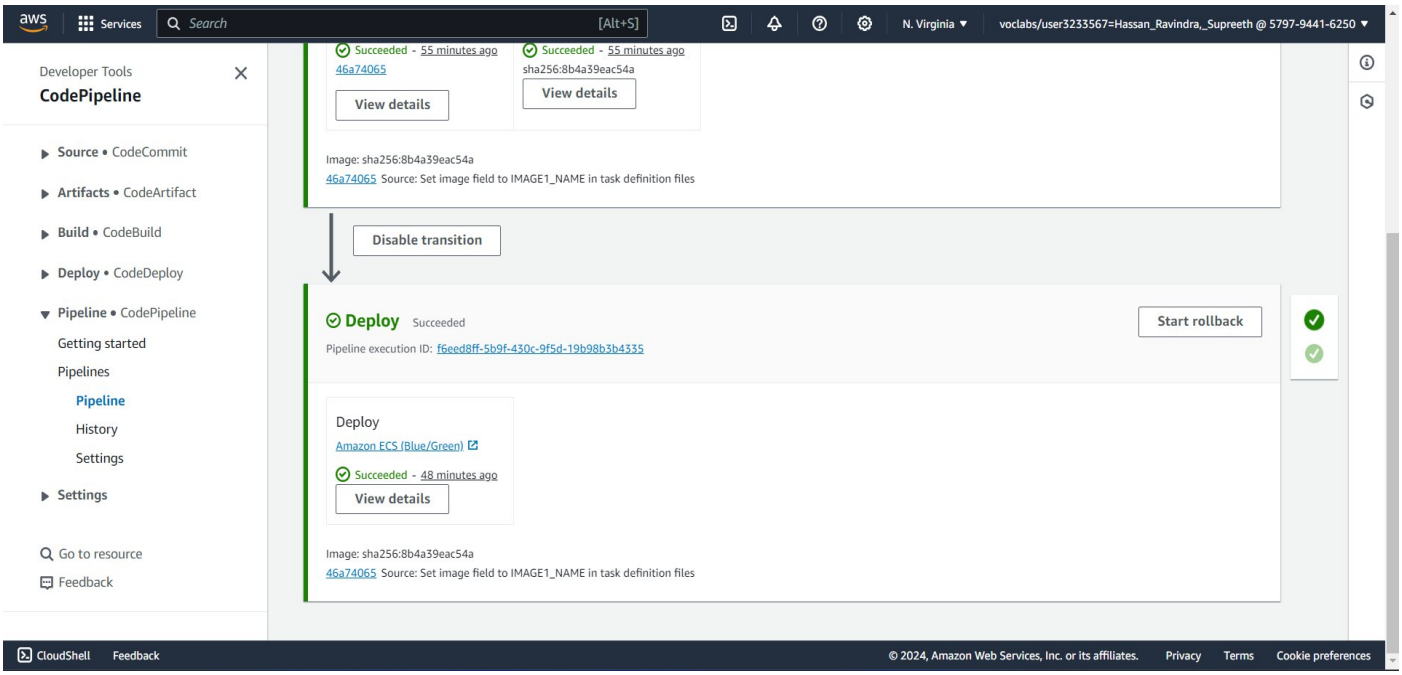


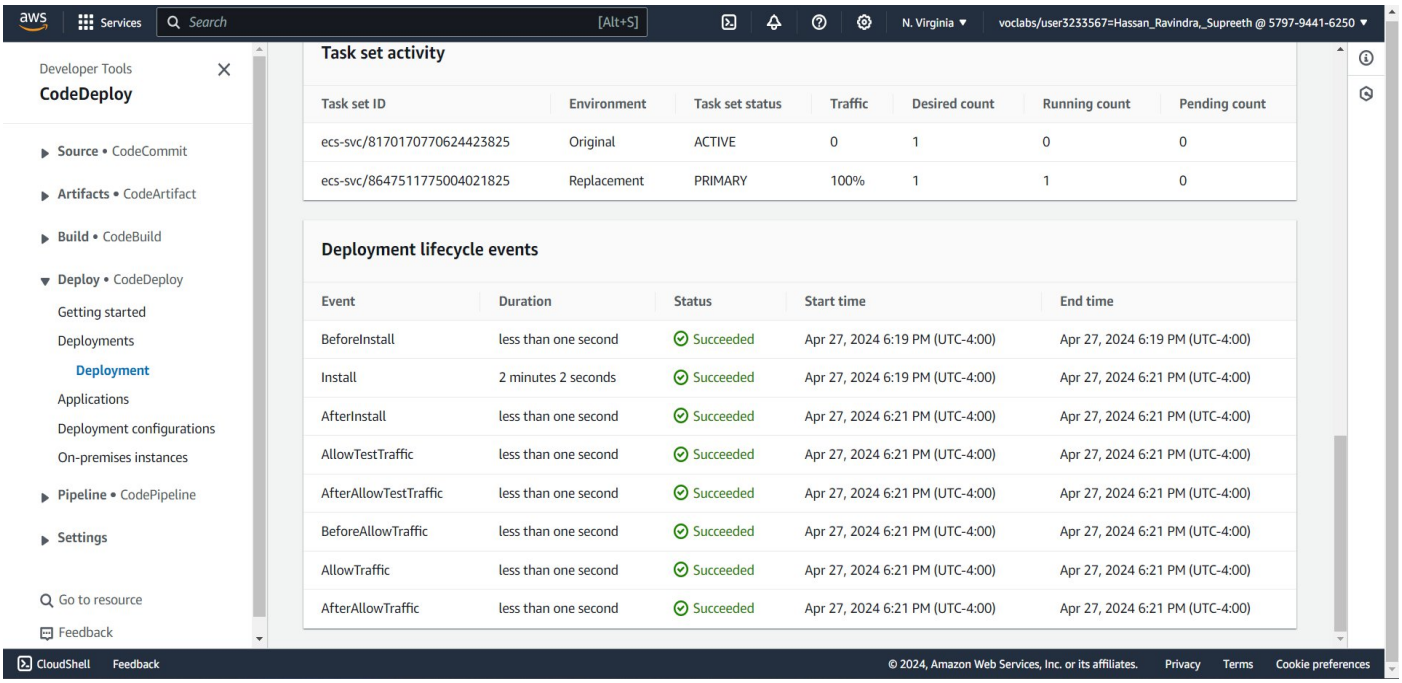
- Task 8.5: Tested the CI/CD pipeline for the employee microservice similar to the customer microservice.

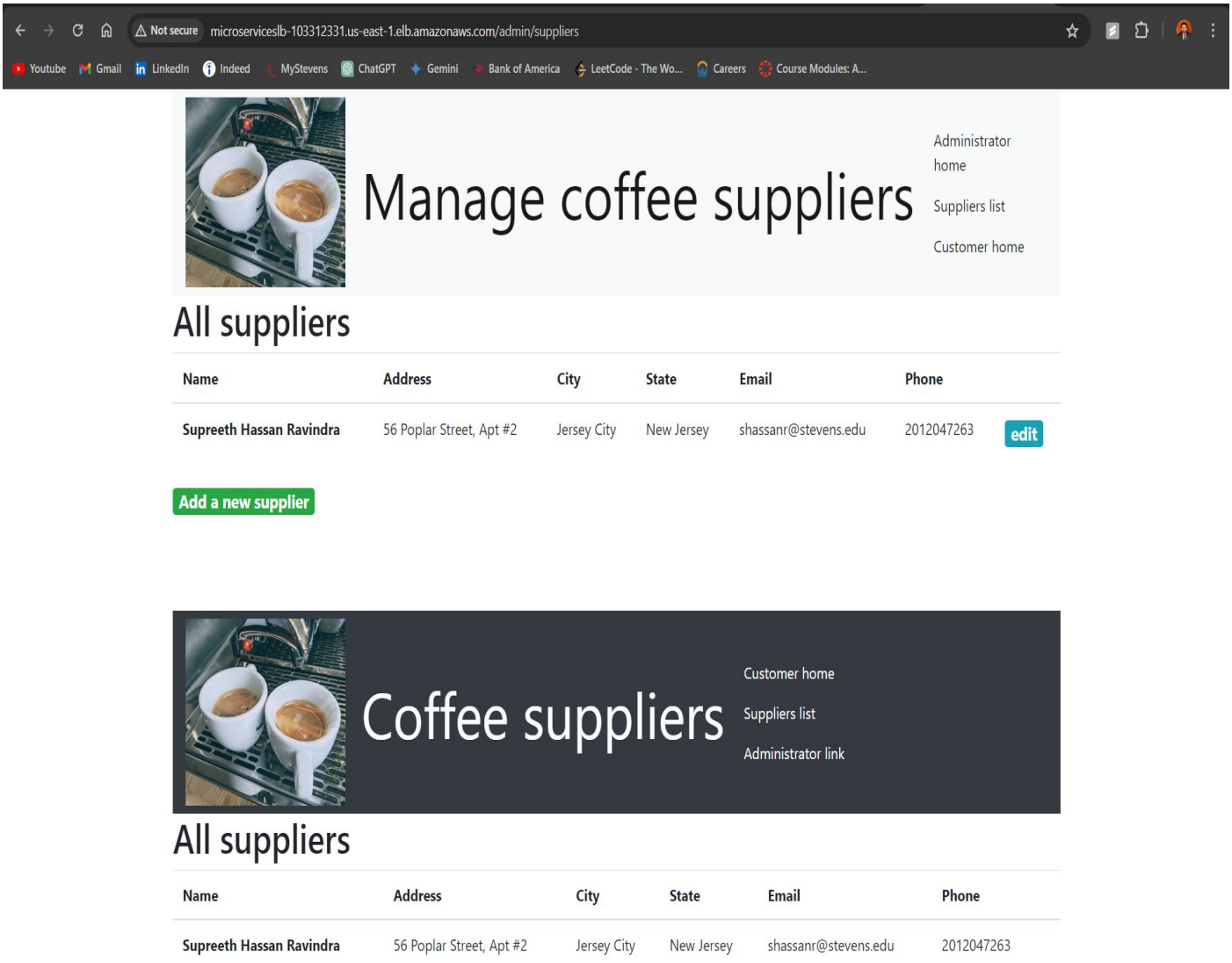




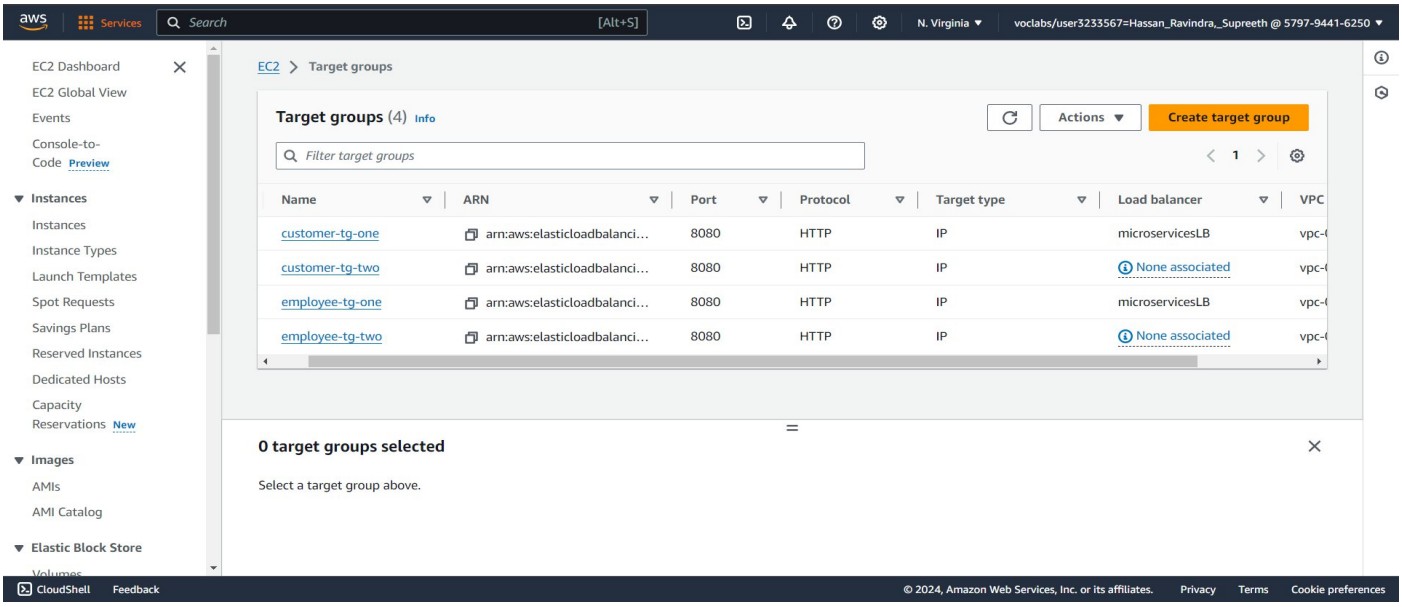

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Deployment
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Task set activity
Task set ID
Environment
Task set status
Traffic
Desired count
Running count
Pending count
ecs-svc/8170170770624423825
Original
PRIMARY
100%
1
0
0
ecs-svc/8647511775004021825
Replacement
ACTIVE
0
1
1
0
Deployment lifecycle events
Event
Duration
Status
Start time
End time
BeforeInstall
less than one second
Succeeded
Apr 27, 2024 6:19 PM (UTC-4:00)
Apr 27, 2024 6:19 PM (UTC-4:00)
Install
-
In progress
Apr 27, 2024 6:19 PM (UTC-4:00)
-
AfterInstall
-
Pending
-
-
AllowTestTraffic
-
Pending
-
-
AfterAllowTestTraffic
-
Pending
-
-
BeforeAllowTraffic
-
Pending
-
-
AllowTraffic
-
Pending
-
-
AfterAllowTraffic
-
Pending
-
-
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

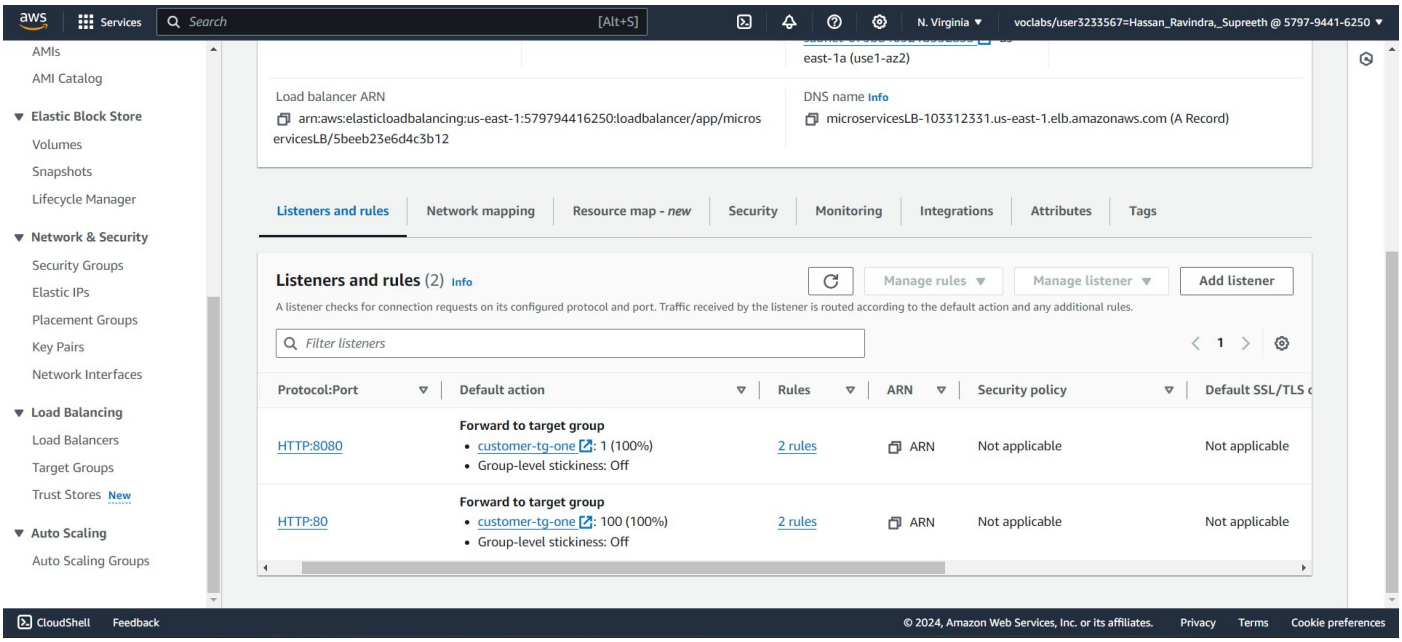

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Succeeded - 55 minutes ago
46a74065
View details
Succeeded - 55 minutes ago
sha256:8b4a39eac54a
View details
Image: sha256:8b4a39eac54a
46a74065 Source: Set image field to IMAGE1_NAME in task definition files
Disable transition
Deploy
Succeeded
Start rollback
Pipeline execution ID: f6eed8ff-5b9f-430c-9f5d-19b98b3b4335
Deploy
Amazon ECS (Blue/Green)
Succeeded - 48 minutes ago
View details
Image: sha256:8b4a39eac54a
46a74065 Source: Set image field to IMAGE1_NAME in task definition files
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

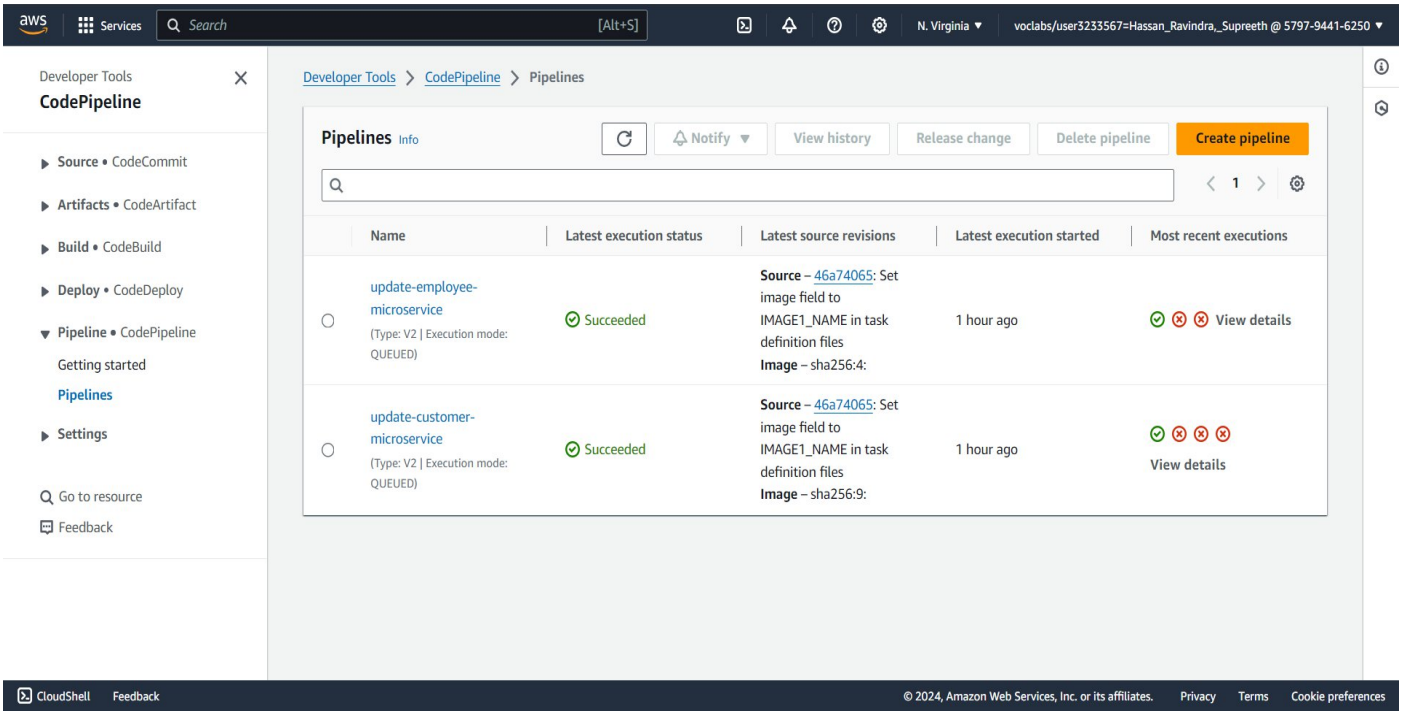

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Deployment
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Task set activity
Task set ID
Environment
Task set status
Traffic
Desired count
Running count
Pending count
ecs-svc/8170170770624423825
Original
ACTIVE
0
1
0
0
ecs-svc/8647511775004021825
Replacement
PRIMARY
100%
1
1
0
Deployment lifecycle events
Event
Duration
Status
Start time
End time
BeforeInstall
less than one second
Succeeded
Apr 27, 2024 6:19 PM (UTC-4:00)
Apr 27, 2024 6:19 PM (UTC-4:00)
Install
2 minutes 2 seconds
Succeeded
Apr 27, 2024 6:19 PM (UTC-4:00)
Apr 27, 2024 6:21 PM (UTC-4:00)
AfterInstall
less than one second
Succeeded
Apr 27, 2024 6:21 PM (UTC-4:00)
Apr 27, 2024 6:21 PM (UTC-4:00)
AllowTestTraffic
less than one second
Succeeded
Apr 27, 2024 6:21 PM (UTC-4:00)
Apr 27, 2024 6:21 PM (UTC-4:00)
AfterAllowTestTraffic
less than one second
Succeeded
Apr 27, 2024 6:21 PM (UTC-4:00)
Apr 27, 2024 6:21 PM (UTC-4:00)
BeforeAllowTraffic
less than one second
Succeeded
Apr 27, 2024 6:21 PM (UTC-4:00)
Apr 27, 2024 6:21 PM (UTC-4:00)
AllowTraffic
less than one second
Succeeded
Apr 27, 2024 6:21 PM (UTC-4:00)
Apr 27, 2024 6:21 PM (UTC-4:00)
AfterAllowTraffic
less than one second
Succeeded
Apr 27, 2024 6:21 PM (UTC-4:00)
Apr 27, 2024 6:21 PM (UTC-4:00)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



- Task 8.6: To observe the load balancer and target group settings I did the following in the Amazon EC2 console:
- I navigated to the "Target Groups" page and refreshed it. I noticed that the customer-tg-two target group was no longer associated with the load balancer. This happened because CodeDeploy manages the load balancer listener rules, adjusting associations as needed.
- The default "if no other rule applies" rule previously pointed to customer-tg-two but it now points to customer-tg-one. This change reflects the updated configuration due to CodeDeploy's management.
- Both rules still forward traffic to the "one" target groups, indicating stability in these listener rules.
- These observations highlight changes in load balancer associations and listener rules, demonstrating CodeDeploy's role in managing the deployment process and dynamically adjusting the load balancer settings.


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
EC2 > Target groups
Target groups (4) Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
VPC
customer-tg-one
arn:aws:elasticloadbalanci...
8080
HTTP
IP
microservicesLB
customer-tg-two
arn:aws:elasticloadbalanci...
8080
HTTP
IP
None associated
employee-tg-one
arn:aws:elasticloadbalanci...
8080
HTTP
IP
microservicesLB
employee-tg-two
arn:aws:elasticloadbalanci...
8080
HTTP
IP
None associated
0 target groups selected
Select a target group above.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

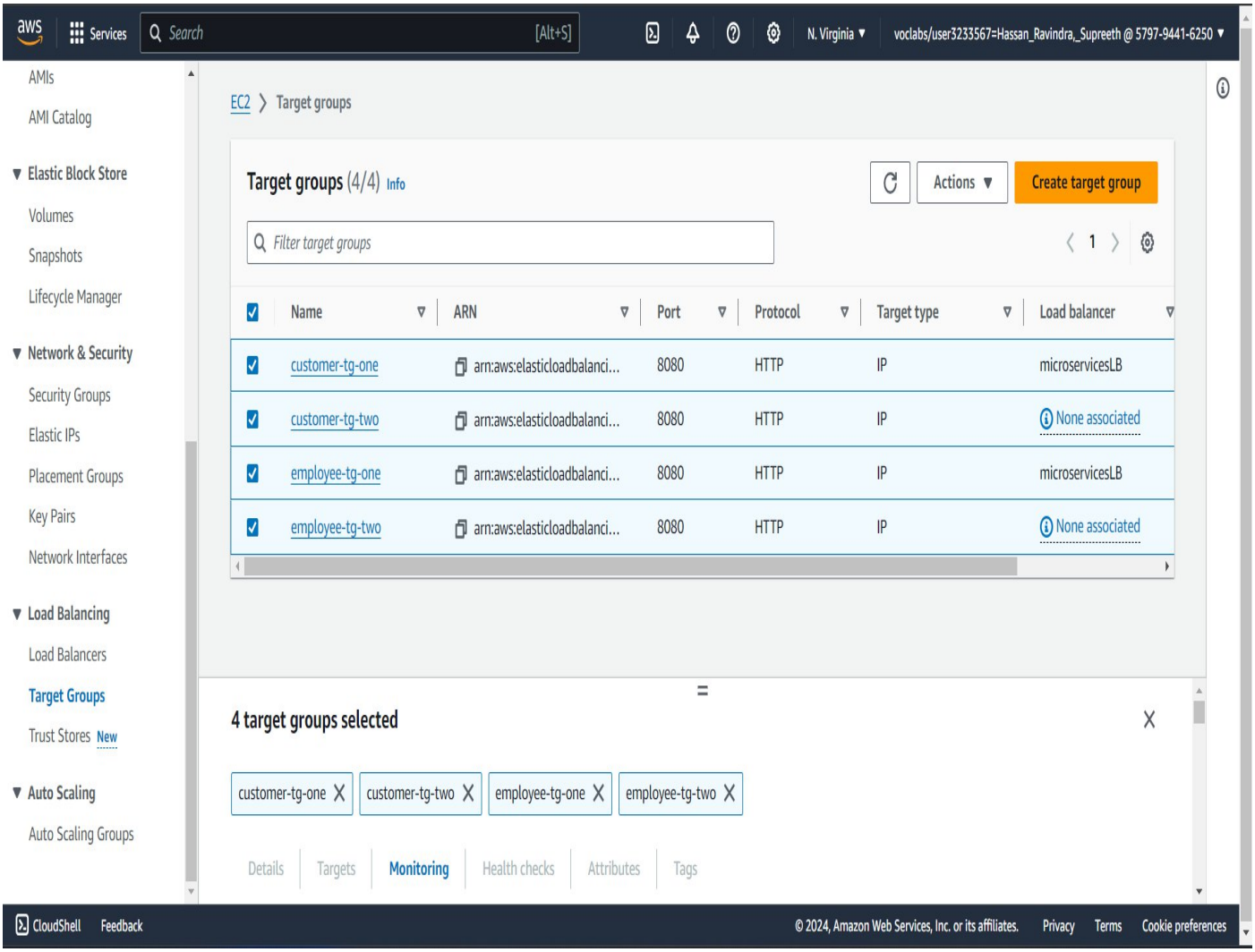

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
east-1a (use1-az2)
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:579794416250:loadbalancer/app/microservicesLB/5beeb23e6d4c3b12
DNS name Info
microservicesLB-103312331.us-east-1.elb.amazonaws.com (A Record)
Listeners and rules
Network mapping
Resource map - new
Security
Monitoring
Integrations
Attributes
Tags
Listeners and rules (2) Info
Manage rules
Manage listener
Add listener
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.
Filter listeners
Protocol:Port
Default action
Rules
ARN
Security policy
Default SSL/TLS
HTTP:8080
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
2 rules
ARN
Not applicable
Not applicable
HTTP:80
Forward to target group
customer-tg-one: 100 (100%)
Group-level stickiness: Off
2 rules
ARN
Not applicable
Not applicable
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Settings
Go to resource
Feedback
Developer Tools > CodePipeline > Pipelines
Pipelines Info
Notify
View history
Release change
Delete pipeline
Create pipeline
Name
Latest execution status
Latest source revisions
Latest execution started
Most recent executions
update-employee-microservice
(Type: V2 | Execution mode: QUEUED)
Succeeded
Source – 46a74065: Set image field to IMAGE1_NAME in task definition files
Image – sha256:4:
1 hour ago
View details
update-customer-microservice
(Type: V2 | Execution mode: QUEUED)
Succeeded
Source – 46a74065: Set image field to IMAGE1_NAME in task definition files
Image – sha256:9:
1 hour ago
View details
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

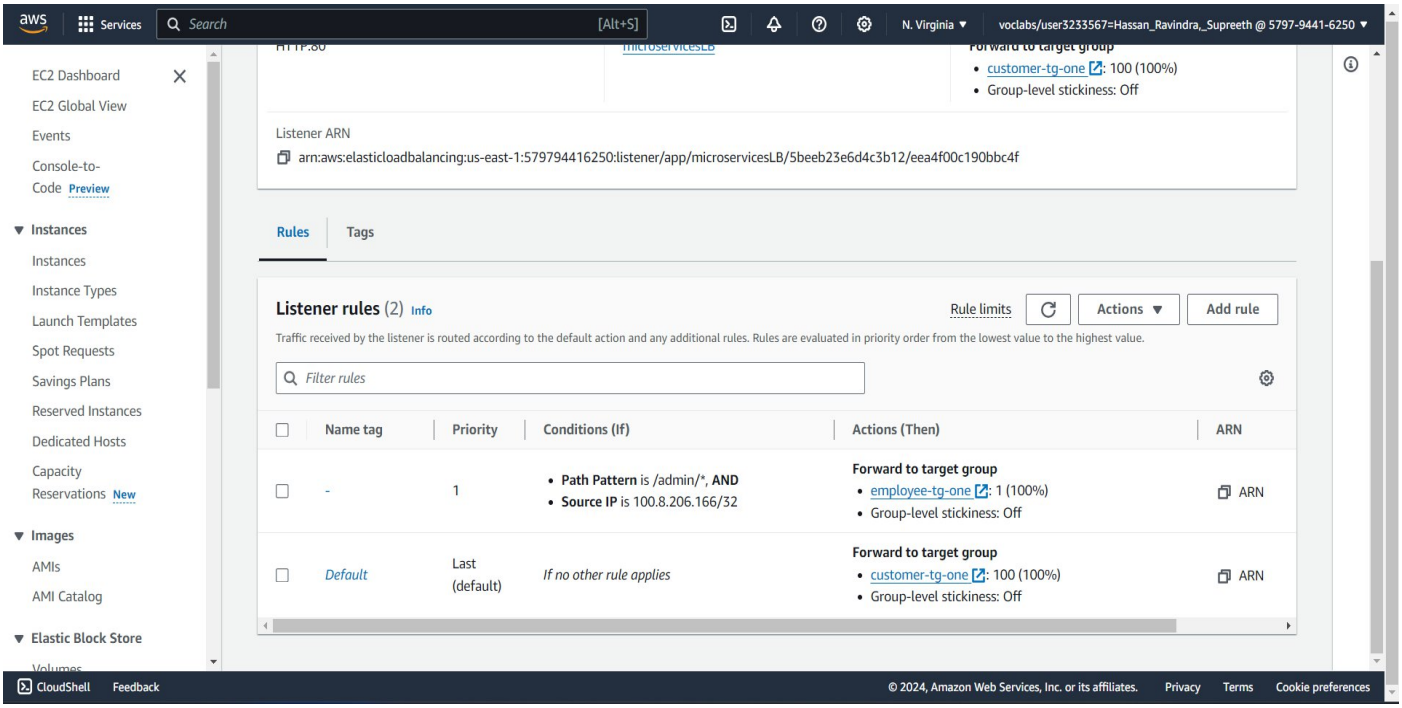
PHASE-9: Adjusting the microservice code to cause a pipeline to run again

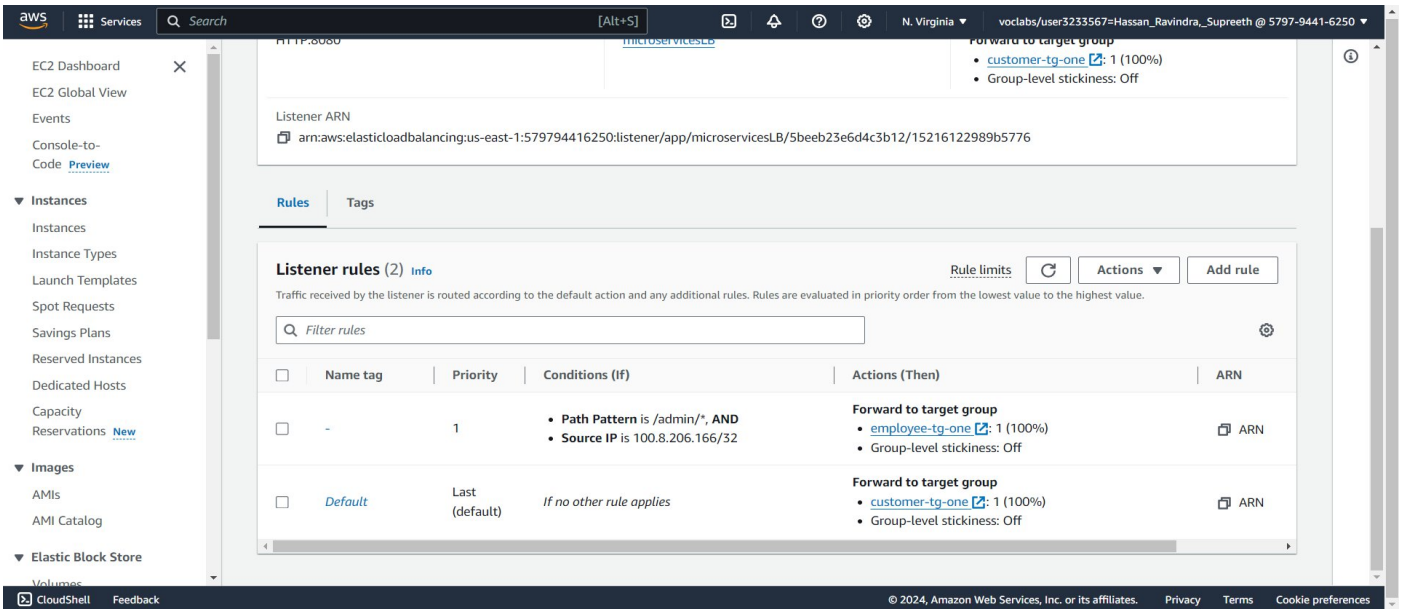
- Task 9.1: To limit access to the employee microservice to a specific IP address I completed the following steps:

- In the Amazon EC2 console, I verified that all four target groups were still associated with the Application Load Balancer. If necessary, I reassociated any missing target groups.
- I used an online resource (like whatismyip.com) to find my public IPv4 address.
- I added a condition to the rule with "IF Path is /admin/*" to limit access to a specific IP address. This condition allowed routing to employee target groups only if the source IP matched my public IPv4 address with a /32 subnet mask.
- Similarly, I added the same source IP condition to the corresponding rule ensuring that the employee microservice could only be accessed from my specific IP address.
- By implementing these changes, I restricted access to the employee microservice to users connecting from a designated IP address providing an additional layer of security and preventing unauthorized users from modifying or deleting supplier entries.

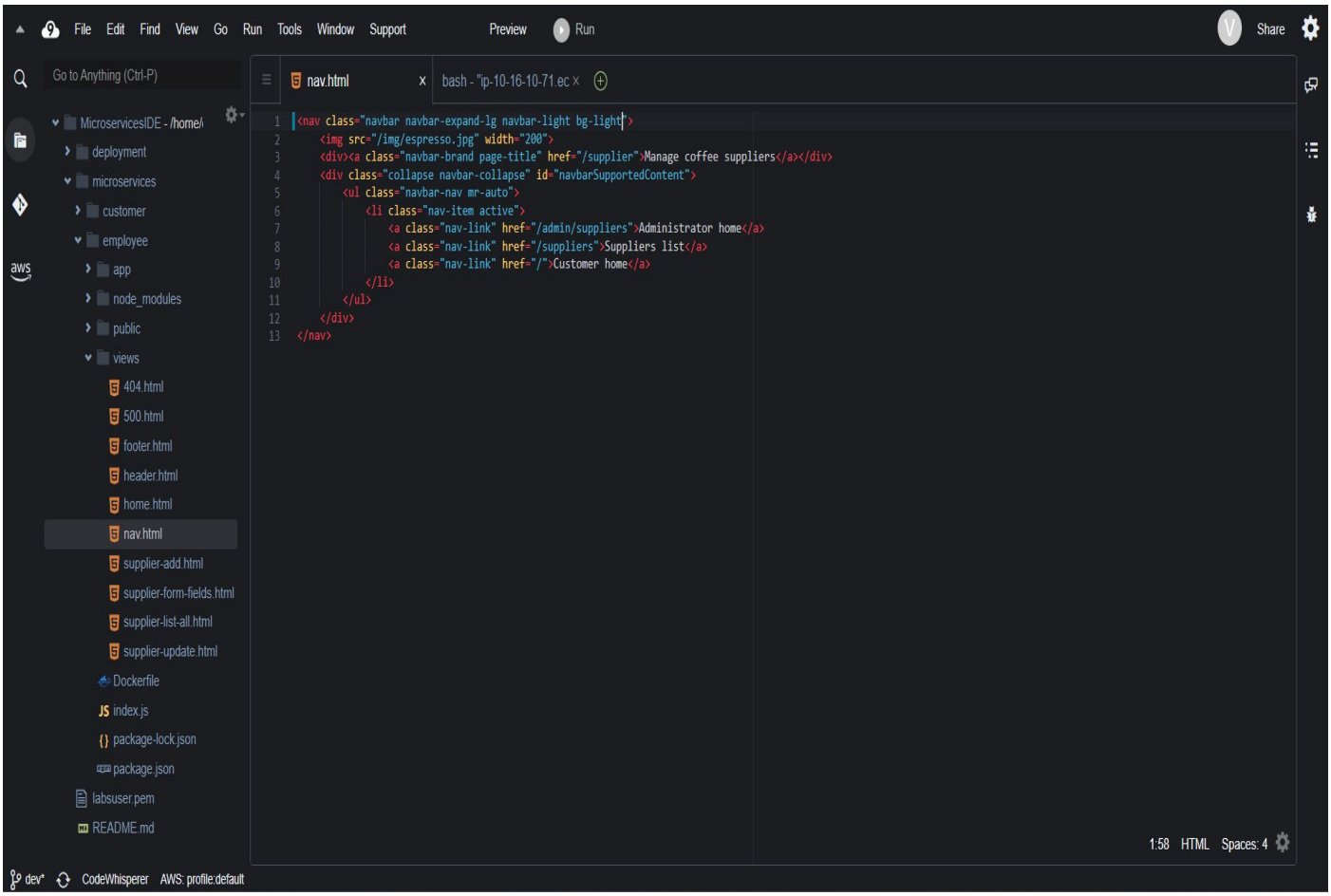


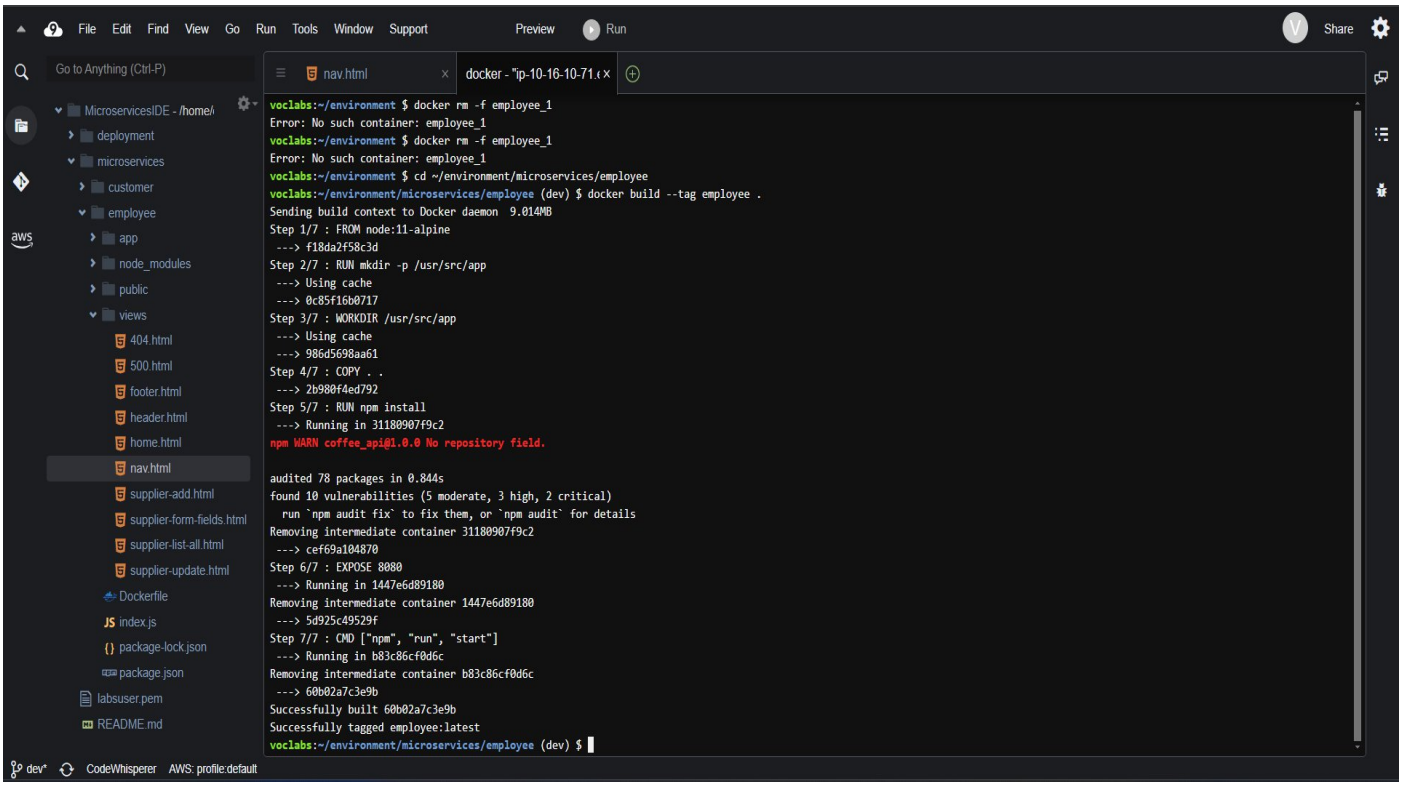

WhatIsMyIP.com
Search..
Pricing
API
Sign Up
Login
Help
What Is My IP?
IP Address Lookup
IP WHOIS Lookup
DNS Lookup
Internet Speed Test
Tools
What Is My IP?
My Public IPv4: 100.8.206.166
My Public IPv6: Not Detected
My IP Location: Newark, NJ US
My ISP: Verizon Business
What is an IP address?

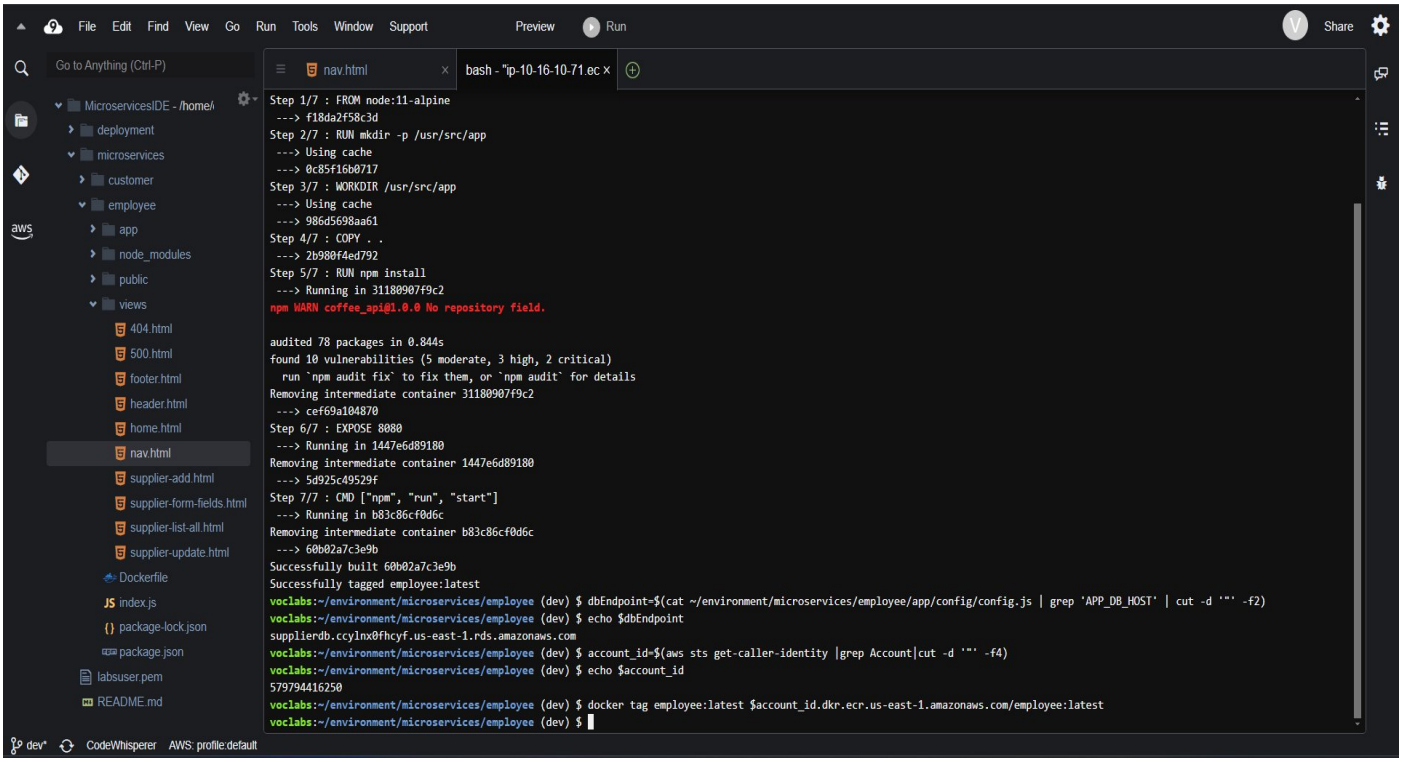

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
customer-tg-one: 100 (100%)
Group-level stickiness: Off
Listener ARN
arn:aws:elasticloadbalancing:us-east-1:579794416250:listener/app/microservicesLB/5beeb23e6d4c3b12/eea4f00c190bbc4f
Rules
Tags
Listener rules (2) Info
Rule limits
Actions
Add rule
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
-
1
Path Pattern is /admin/*, AND
Source IP is 100.8.206.166/32
Forward to target group
employee-tg-one: 1 (100%)
Group-level stickiness: Off
Default
Last (default)
If no other rule applies
Forward to target group
customer-tg-one: 100 (100%)
Group-level stickiness: Off
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

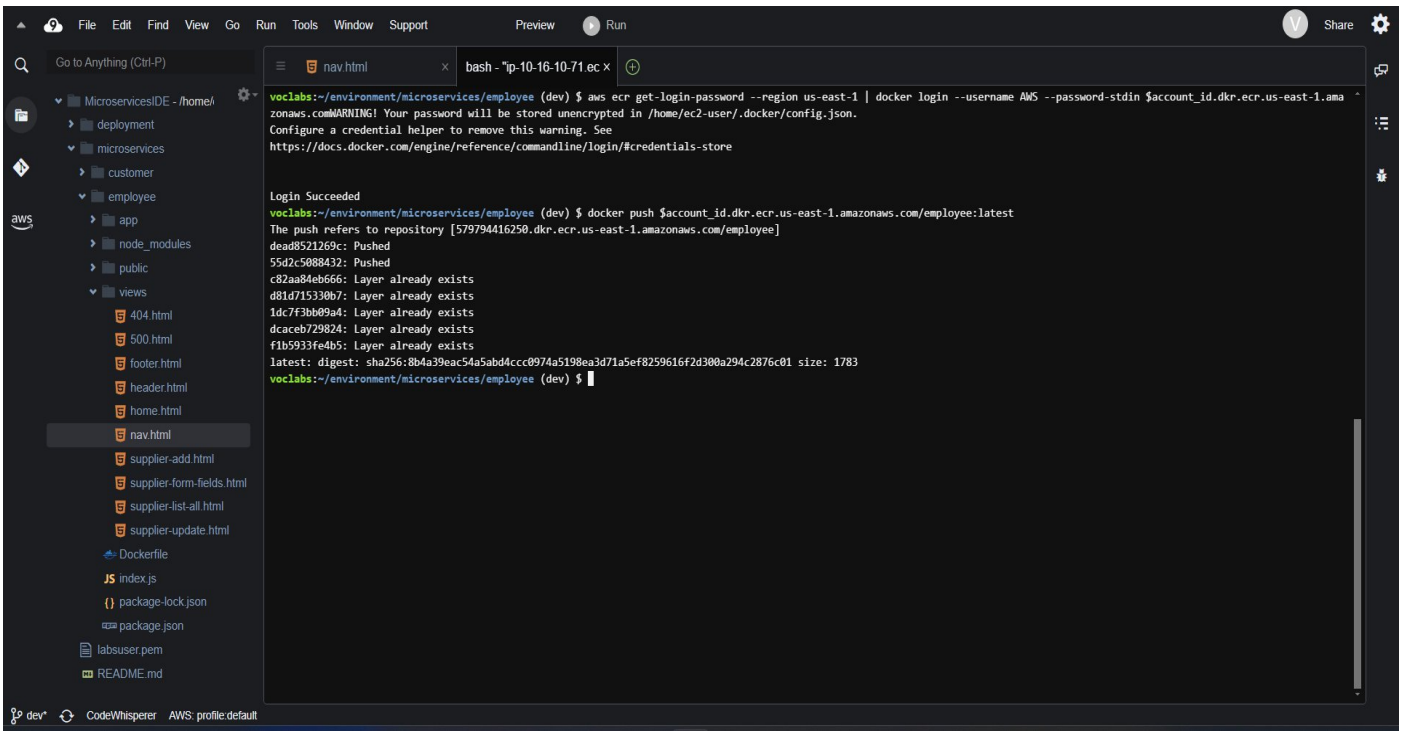

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
customer-tg-one: 1 (100%)
Group-level stickiness: Off
Listener ARN
arn:aws:elasticloadbalancing:us-east-1:579794416250:listener/app/microservicesLB/5beeb23e6d4c3b12/15216122989b5776
Rules
Tags
Listener rules (2) Info
Rule limits
Actions
Add rule
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
-
1
Path Pattern is /admin/*, AND
Source IP is 100.8.206.166/32
Forward to target group
employee-tg-one: 1 (100%)
Group-level stickiness: Off
Default
Last (default)
If no other rule applies
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

- Task 9.2: I Adjusted the UI for the employee microservice and push the updated image to Amazon ECR. These steps ensured that the UI adjustment was reflected in a new Docker image and the updated image was successfully pushed to Amazon ECR for use in the CI/CD pipeline for the employee microservice.

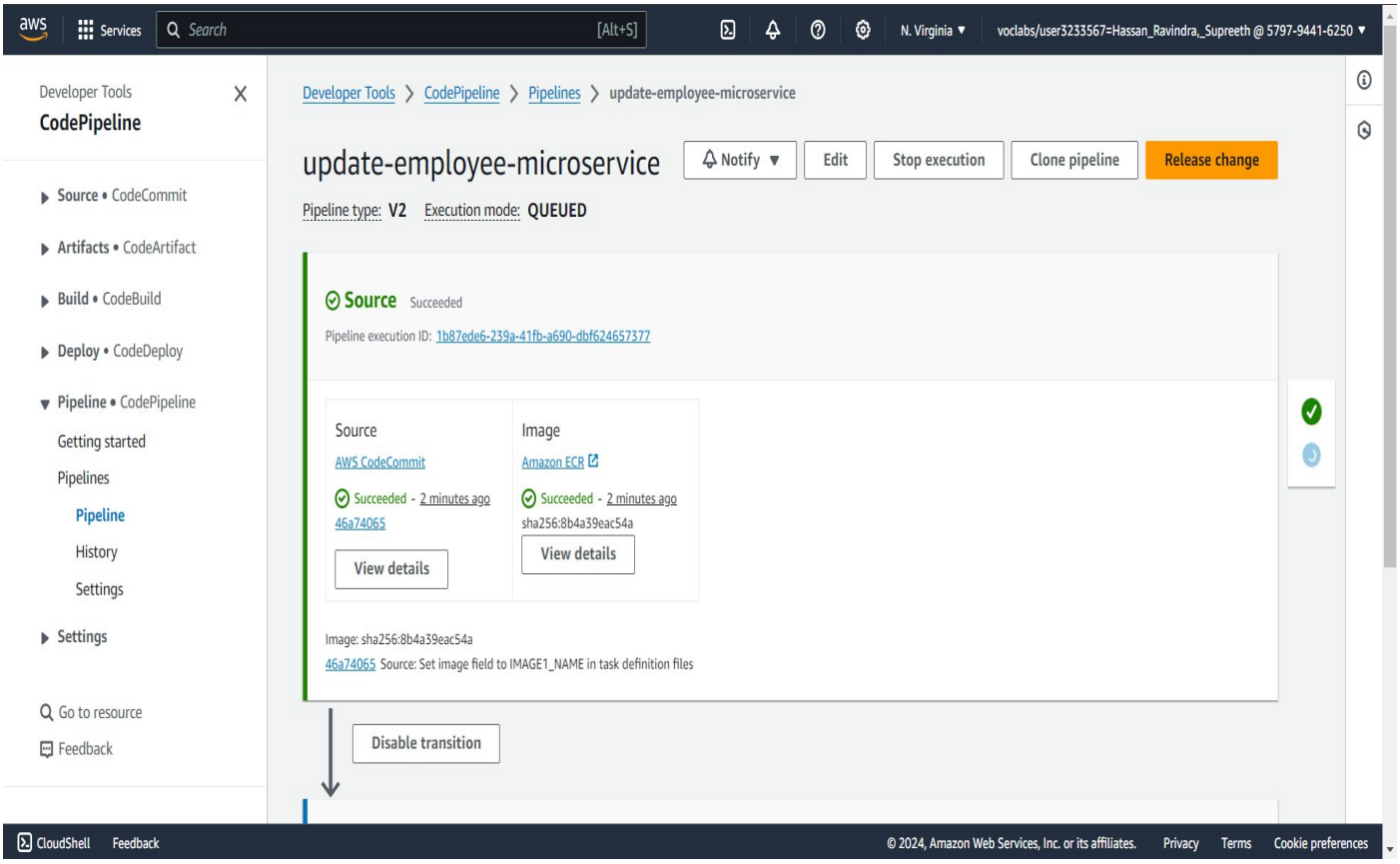


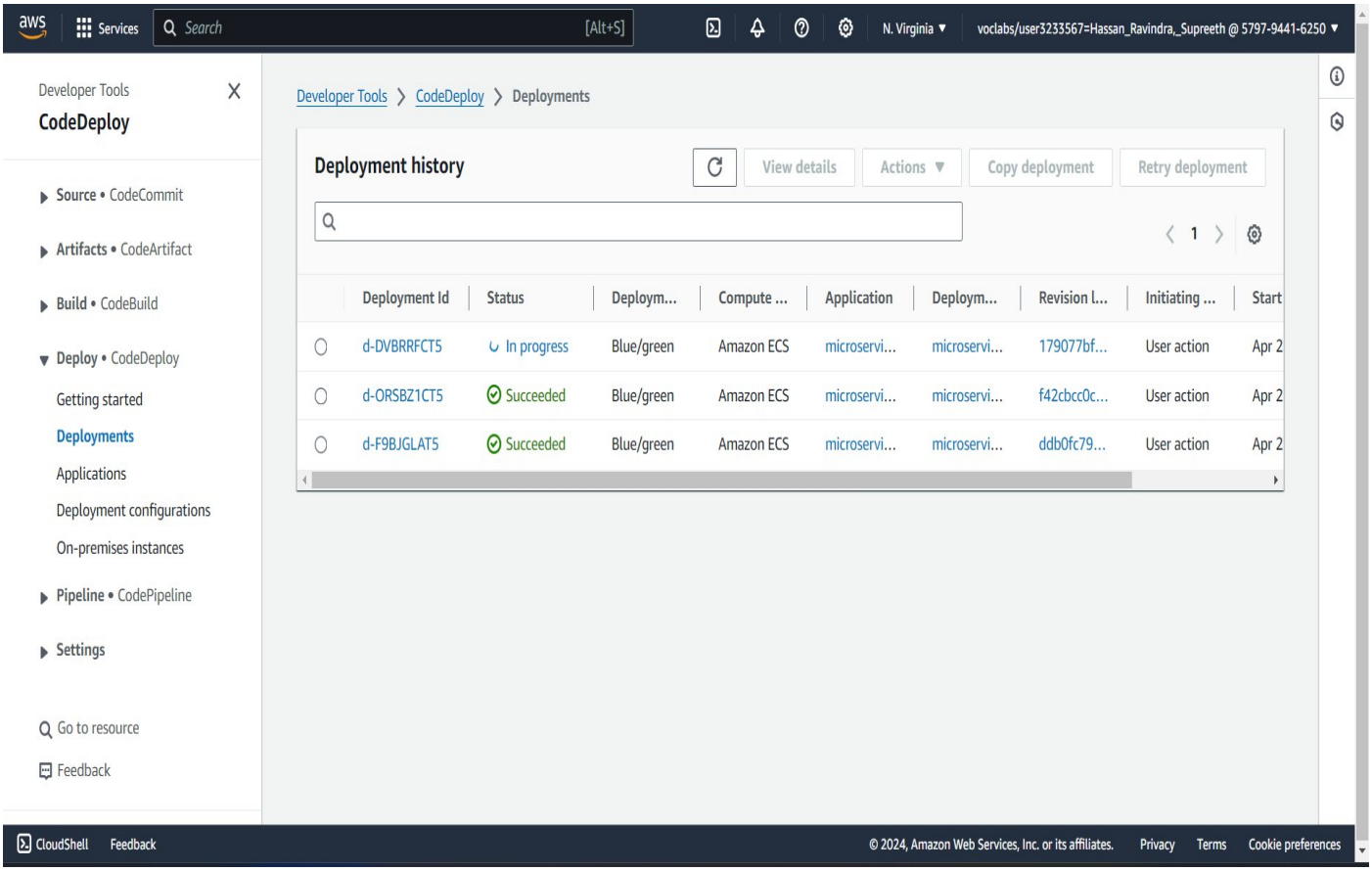




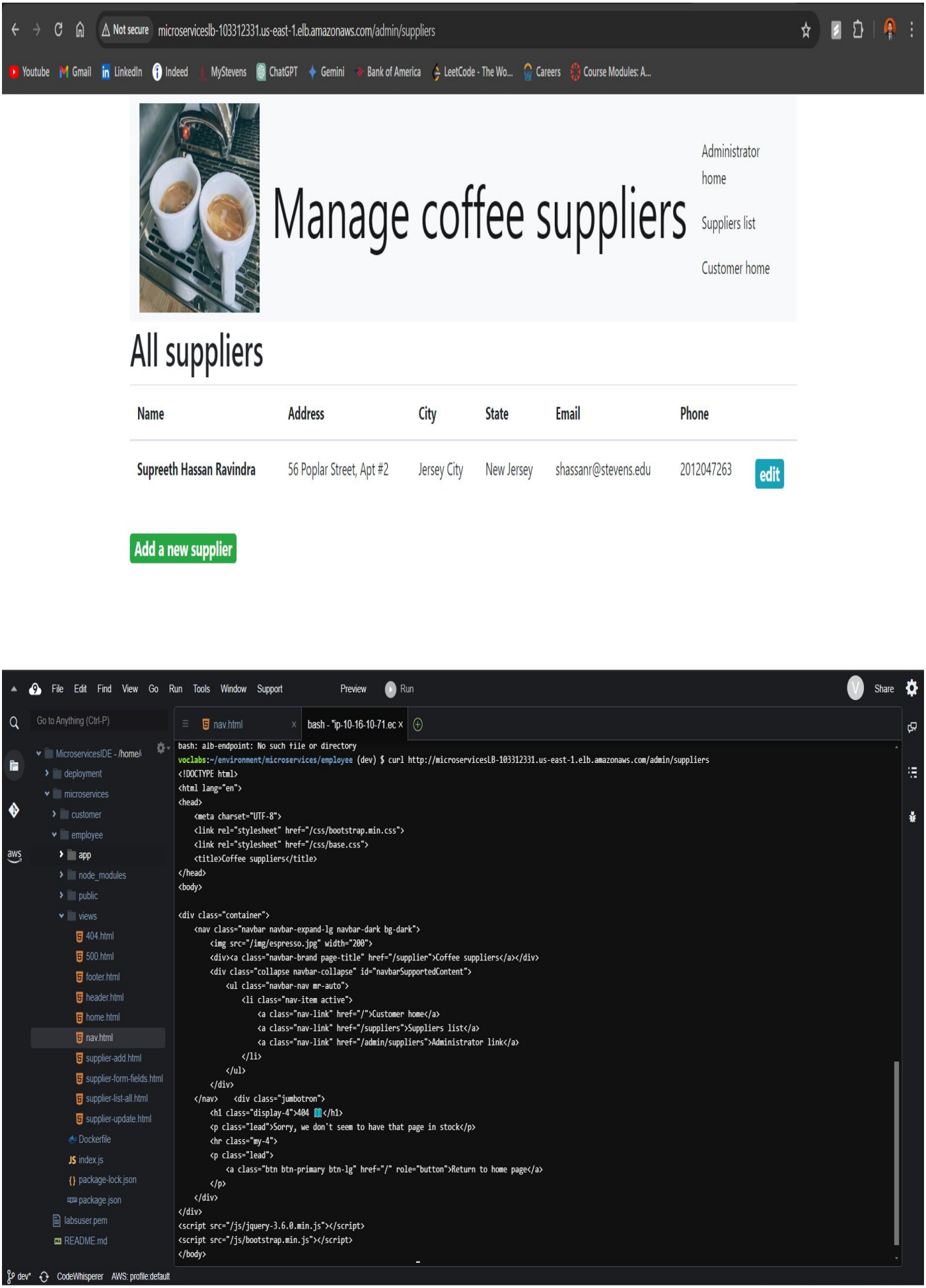


- Task 9.3: I confirmed that the employee pipeline ran and the microservice was updated. In the CodePipeline console I navigated to the "update-employee-microservice" pipeline details. I noticed that when I uploaded the new Docker image to Amazon ECR, the pipeline was invoked and began running. It can take a minute or two for the pipeline to detect that the Docker image has been updated.
- I checked the CodeDeploy console to monitor the deployment progress and ensure that the microservice update was being applied correctly. I verified that the deployment lifecycle events succeeded indicating that the updated employee microservice was being deployed as expected.
- These steps ensured that the pipeline was triggered by the updated Docker image and the deployment process was proceeding as intended.





- Task 9.4: I tested access to the employee microservice using the instructions given.


Not secure
microserviceslb-103312331.us-east-1.elb.amazonaws.com/admin/suppliers
Youtube
Gmail
LinkedIn
Indeed
MyStevens
ChatGPT
Gemini
Bank of America
LeetCode - The Wo...
Careers
Course Modules: A...
Manage coffee suppliers
Administrator home
Suppliers list
Customer home
All suppliers
Name
Address
City
State
Email
Phone
Supreeth Hassan Ravindra
56 Poplar Street, Apt #2
Jersey City
New Jersey
shassanr@stevens.edu
2012047263
edit
Add a new supplier
File
Edit
Find
View
Go
Run
Tools
Window
Support
Preview
Run
Share
Go to Anything (Ctrl-P)
nav.html
bash - "ip-10-16-10-71.ec
MicroservicesIDE - /home/
deployment
microservices
customer
employee
app
node_modules
public
views
404.html
500.html
footer.html
header.html
home.html
nav.html
supplier-add.html
supplier-form-fields.html
supplier-list-all.html
supplier-update.html
Dockerfile
index.js
package-lock.json
package.json
labsuser.pem
README.md
bash: alb-endpoint: No such file or directory
voclabs:~/environment/microservices/employee (dev) $ curl http://microservicesLB-103312331.us-east-1.elb.amazonaws.com/admin/suppliers
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="/css/bootstrap.min.css">
<link rel="stylesheet" href="/css/base.css">
<title>Coffee suppliers</title>
</head>
<body>

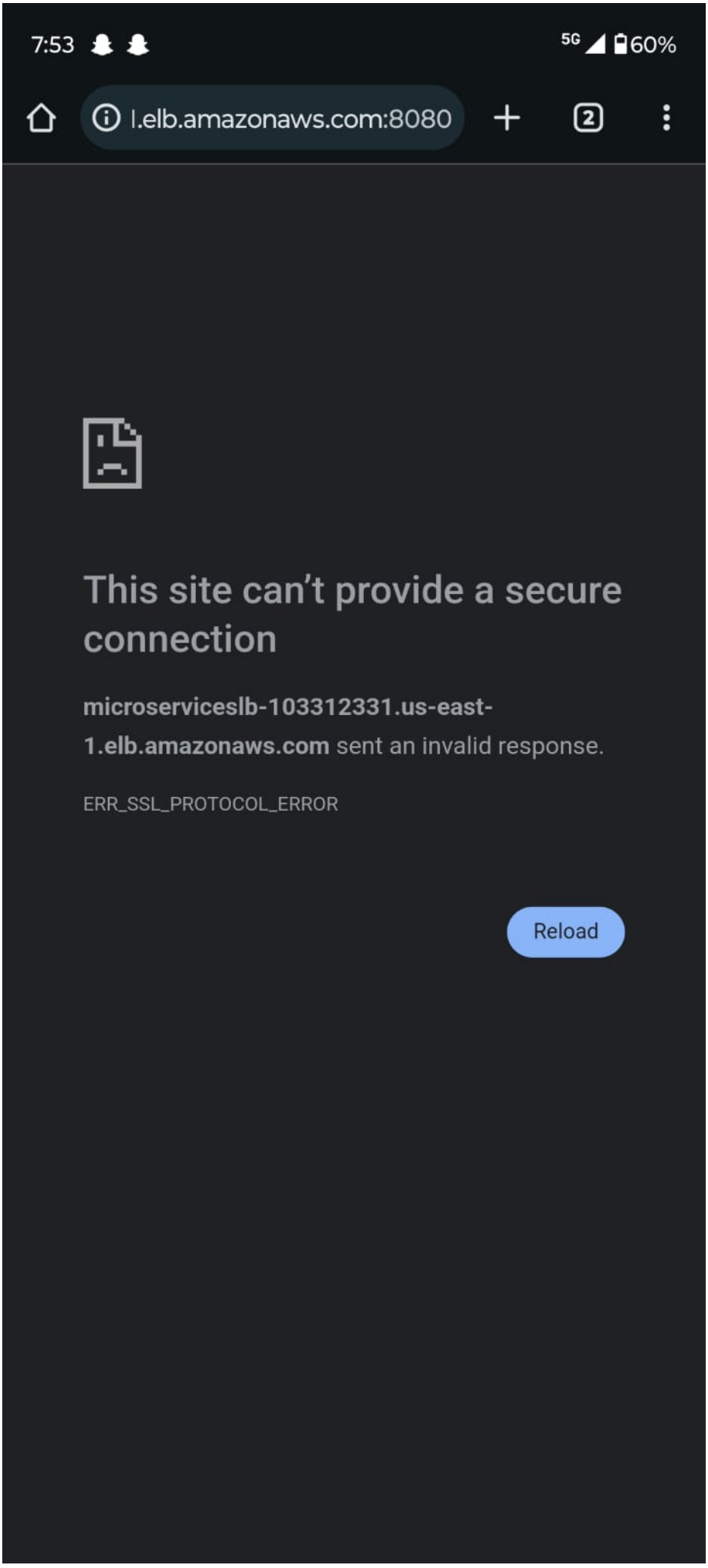
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<img src="/img/espresso.jpg" width="200">
Coffee suppliers

<ul class="navbar-nav mr-auto">
<li class="nav-item active">
Customer home
Suppliers list
Administrator link

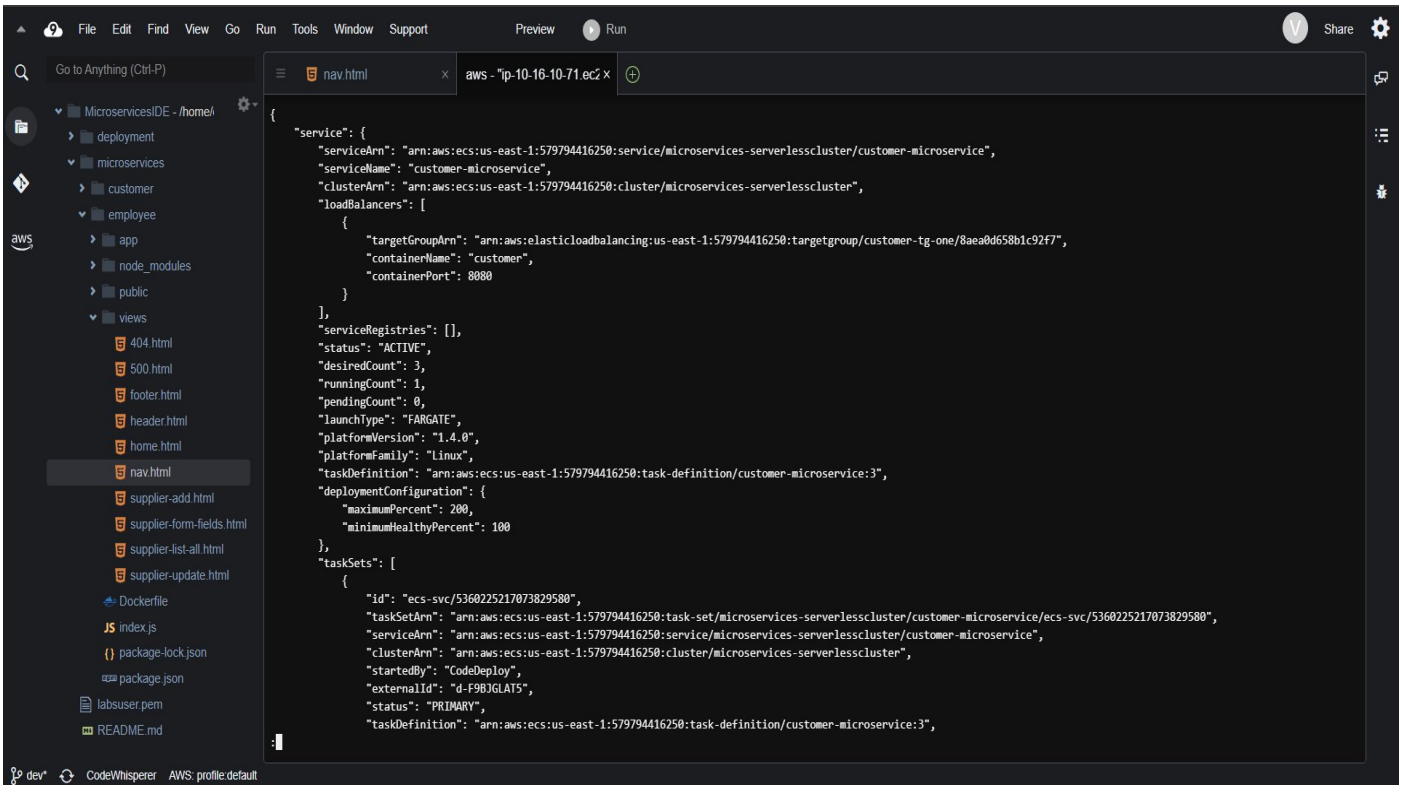
</nav>
<h1 class="display-4">404</h1>
Sorry, we don't seem to have that page in stock
<hr class="my-4">

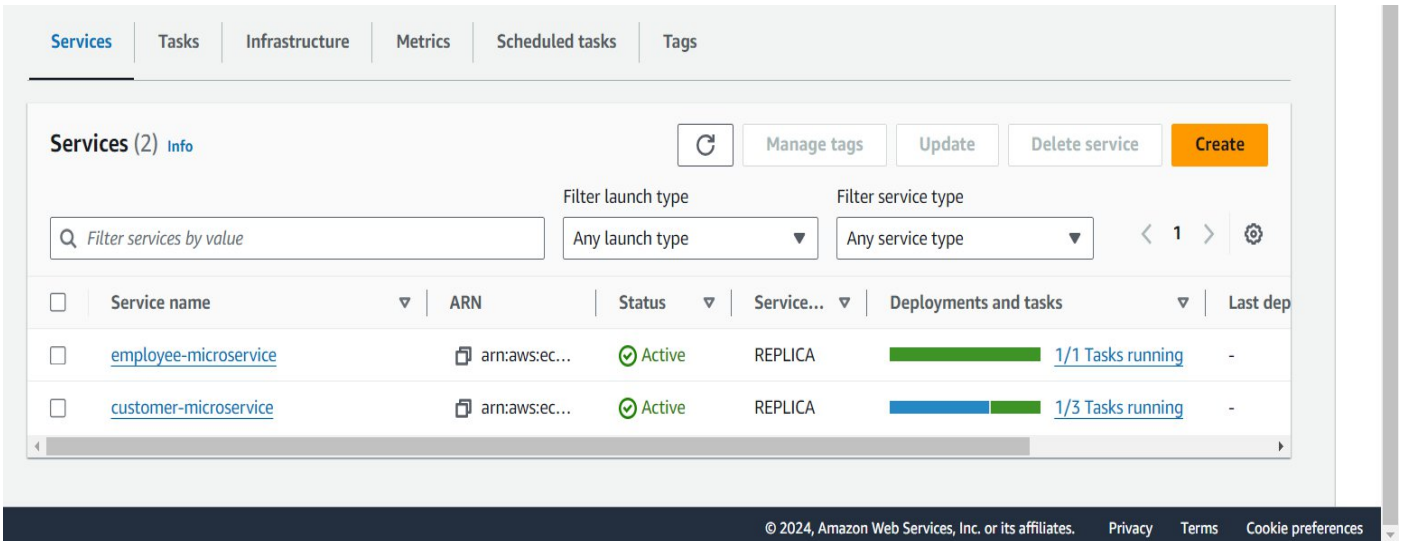
Return to home page

<script src="/js/jquery-3.6.0.min.js"></script>
<script src="/js/bootstrap.min.js"></script>
</body>
dev*
CodeWhisperer
AWS: profile:default


7:53
5G 60%
l.elb.amazonaws.com:8080
2
This site can’t provide a secure connection
microserviceslb-103312331.us-east-1.elb.amazonaws.com sent an invalid response.
ERR_SSL_PROTOCOL_ERROR
Reload

- Task 9.5: To scale the customer microservice I increased the number of containers that run to support it. These steps illustrate how to scale the customer microservice in Amazon ECS, allowing more containers to run and support the microservice without affecting the CI/CD pipeline or other microservices.





PART-2: ANALYSIS

PHASE-2: Analyzing the infrastructure of the monolithic application

- In this initial phase, I analyzed the infrastructure of the monolithic application and its operations.
- The monolithic application is hosted on an Amazon EC2 instance and is accessible via a Public IPv4 address. However it is accessible via HTTP instead of HTTPS indicating a lack of SSL/TLS security. This poses security risks for production environments and shows the need for SSL/TLS certificates to ensure encrypted communication.
- By interacting with the web application where I started to add my name “Supreeth Hassan Ravindra” as a supplier and edit information regarding the supplier, I saw that the application follows a simple URL structure. This suggests a basic RESTful approach with URLs indicating different actions such as listing, adding and updating suppliers. The application's ability to add and update data demonstrates that it's functional. However this functionality must be preserved during a transition to microservices.
- Using the command `**sudo lsof -i :80**` it shows that the application uses port 80 with the HTTP protocol. By examining the processes with `**ps -ef | grep node**` it appears that a Node.js-based server is running. The process ID can be matched with the output of the `**lsof**` command, confirming that this is the process serving the web application.
- The presence of `index.js` in the specified directory shows that the Node.js application has a basic structure with `index.js` being the main entry point for the application logic.This structure might show that the application is installed through Node.js with necessary prerequisites and is likely managed through `npm`.
- The application stores its data in a MySQL database that is hosted on **Amazon RDS**. The use of a separate database service for storage demonstrates a typical architecture where the application logic and data storage are decoupled.
- Connection to the RDS database from the EC2 instance indicates that the monolithic application can communicate with external resources. Using `**nmap -Pn**`, the standard MySQL port 3306 was verified to be open which confirms the connectivity. Observing the data within the COFFEE database and suppliers table further tells us that the application's data layer is functional and interacts with the web application as expected.
- These observations in the Phase 2 of project execution provides a comprehensive analysis of the current monolithic application infrastructure and offer insight into potential challenges and considerations for transitioning to a microservices architecture. It sets the foundation for identifying dependencies and understanding communication pathways and preparing for any re-architecture tasks in the next project phases.

PHASE-3: Creating a development environment and checking code into a Git repository

- In this phase, I started by creating an AWS Cloud9 IDE where I was establishing a cloud-based development environment to support the project. The Cloud9 instance should run on an Amazon Linux 2 EC2 instance with SSH support providing a consistent development and debugging experience.
- The instance size and type are crucial with t3.small being a suitable choice for a moderate workload in a lab or development context. Ensuring the environment is connected to the correct VPC and subnet is also essential for communication with other AWS services and resources.
- I set permissions for the SSH key file (labsuser.pem) using the **chmod** command ensuring secure access to the EC2 instance and used **scp** command to copy source code from the MonolithicAppServer instance to the Cloud9 IDE temp directory demonstrates the process of securely transferring files across instances in a development environment. After copying and verifying the presence of source code in the temp directory ensures the data transfer was successful.
- I created a working directories for microservices by creating directories for customer and employee microservices sets the foundation for decoupling the monolithic application. This step involves replicating the original source code into these directories to maintain a separate working environment for each microservice.
- The distinction between customer and employee microservices represents different functionalities and access levels. Customers require a read-only view of supplier contact information and while employees need read-write access to manage suppliers. Moving source code to the microservices directories and deleting the temp directory helps maintain a clean and organized development environment.
- I also created a Git Repository and Pushing Code to CodeCommit. Creating a Git repository in the Cloud9 environment (**git init**) is the initial step in version control, allowing you to track changes and collaborate.
- In the next step, creating a development branch (git branch -m dev) and defining the remote origin (git remote add origin) connects the local repository in my PC to AWS CodeCommit which is a Git-compatible repository for centralized code management.
- Adding all files to the repository and committing them with a descriptive message (**git commit -m 'two unmodified copies of the application code'**) marks a significant point in the project's development. Pushing to the remote origin ensures that the code is backed up and accessible from other environments or by other team members.
- Setting the Git username and email address ensures proper attribution and tracking of changes in the repository. This is crucial for collaboration and code reviews.
- All these steps in Phase 3 focuses on establishing a development environment, transferring source code, creating a structure for microservices and setting up version control with Git and AWS CodeCommit. These steps are foundational for future development which allows to maintain, update and evolve the codebase with proper version control and a clear path towards microservices architecture. This phase also highlights the importance of secure connections, organized directory structures and proper Git configurations to ensure a smooth development process.

Phase 4: Configuring the application as two microservices and testing them in Docker containers

- The project execution in Phase 4 involves creating and configuring microservices for a previously monolithic application testing these microservices using Docker containers and establishing a strong Git workflow with CodeCommit.
- I started off in Opening Ports for Docker Containers by allowing inbound network traffic on TCP ports 8080 and 8081 ensuring that the Docker containers running on your AWS Cloud9 instance can be accessed from the internet. This setup is crucial for testing the microservices independently.
- I modified the customer microservice as it should only provide read-only access to the supplier data. By removing functions related to adding, editing or deleting suppliers we ensure that customers have the correct level of access. By deleting unnecessary files and functions, it maintains a clean codebase, reducing the risk of unintended behavior or security vulnerabilities. Adjusting navigation links in `nav.html` to provide appropriate paths for customer and admin (employee) access creates a clear separation between the two microservices.
- In the next steps, I started Creating and Testing the Customer Microservice Docker Container. The Dockerfile specifies how to build the Docker image for the customer microservice.
- Next, I Built the Docker Image which compiles the Docker image from the source code and Dockerfile creating a template for launching containers. By running a container on port 8080 verifies that the customer microservice works as intended. Testing key functionalities like listing suppliers and verifying the absence of edit and add buttons ensures the microservice is correctly configured.
- I then Modifying the Employee Microservice by updating the code to prepend `/admin` to various URL paths ensuring that the employee microservice operates separately from the customer microservice. The employee microservice requires functionality to add, edit and delete suppliers. This is achieved by maintaining the appropriate functions and adjusting navigation and form action paths. Similar to the customer microservice navigation links are updated to reflect the correct structure and ensure proper navigation between customer and employee microservices.
- I Created and Tested the Employee Microservice Docker Container following the same process as the customer microservice but with the port set to 8081 creating a Docker image for the employee microservice and launch a container to test it.
- Regarding the Port Configuration for ECS Deployment Changing the port to 8080 prepares the employee microservice for deployment on Amazon ECS which typically uses this port for service endpoints. This step ensures that the image reflects the updated port configuration, ready for deployment in later phases.
- Committing and pushing changes to CodeCommit ensures that the source code is version-controlled and accessible. The Git-compatible workflow provides a robust method to track changes, collaborate, and maintain code integrity.
- Reviewing changes in AWS Cloud9 and then committing to CodeCommit reinforces best practices for code management. This step allows me to track modifications and ensure a clean and organized codebase.
- All these steps in Phase 4 establishes the foundation for running and testing microservices in a Dockerized environment. It involves significant code changes, Docker configuration and Git operations. These steps are crucial for decoupling the monolithic application, verifying microservices functionality and maintaining a robust code repository for future development and deployment.

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

- The tasks in Phase 5 focuses on building and configuring the necessary infrastructure to deploy and manage microservices at scale. This phase includes creating Amazon Elastic Container Registry (ECR) repositories, setting up an Amazon Elastic Container Service (ECS) cluster, defining task definitions and preparing AppSpec files for CodeDeploy integration.
- I started by Creating ECR Repositories and Upload Docker Images. Authorizing my Docker client to connect to Amazon ECR ensures secure access for uploading Docker images. This step is crucial for maintaining security and compliance.
- I created separate repositories for each microservice (customer and employee) and organize the Docker images in a scalable way allowing for independent version control and deployment. The JSON policy for each ECR repository must allow specific actions ensuring proper access and security.Tagging Docker images with my account ID and uploading them to ECR creates a centralized storage location. This process is foundational for deploying to ECS and managing containerized applications.
- In the next task, I created a serverless ECS cluster using AWS Fargate allows for flexible scaling and reduces infrastructure management overhead. By selecting specific subnets I can control where the containers will run. Ensuring the ECS cluster is created successfully requires monitoring the CloudFormation stack creation process.
- I Created a new CodeCommit repository named deployment for task definitions and AppSpec files provides a central location for deployment-related files. This step aligns with best practices for maintaining deployment configurations in version control.
- After that, I Initialized the deployment directory as a Git repository with a dev branch and set the stage for tracking changes and collaborating on deployment-related configurations.
- Task definitions specify how containers should run in ECS. By creating separate task definitions for customer and employee ensures each microservice has its unique configuration.Registering task definitions with ECS establishes a blueprint for deploying containers. This step is critical for scaling, resource allocation and load balancing. Adjusting details like the image name, environment variables such as the RDS endpoint and log configuration provides the necessary instructions for ECS to deploy and manage the microservices.
- The AppSpec Files created after will guide CodeDeploy in deploying microservices to ECS. They include information about task definitions, load balancing and service configuration. By creating individual AppSpec files for customer and employee ensures that CodeDeploy has the correct instructions for deploying and updating each microservice. The AppSpec files include load balancer details indicating how traffic should be routed to the appropriate microservice. This information is crucial for scaling and ensuring proper service routing.
- Modifying the taskdef-customer.json and taskdef-employee.json files to use placeholder text for the image name allows for dynamic updates in a CI/CD pipeline. This flexibility is key for automated deployment processes.
- By pushing the updated files to CodeCommit ensures that the latest deployment configurations are stored in version control. This step is crucial for integrating with CI/CD pipelines and maintaining a robust deployment workflow.
- All these steps in Phase 5 lays the groundwork for deploying microservices to a scalable ECS cluster with CodeDeploy support. It involves creating and configuring ECR repositories, defining ECS task definitions and preparing AppSpec files for automated deployment. These steps are critical for achieving a scalable, resilient and flexible microservices architecture.

Phase 6: Creating target groups and an Application Load Balancer

- In Phase 6, I created an Application Load Balancer with path-based routing to ensure that requests are forwarded to the appropriate target groups depending on the URL path.
- The concept of blue/green deployment ensures a smooth transition when deploying new versions of microservices. By creating two target groups for each microservice I can set the groundwork for deploying updates with minimal downtime and rollback capability.
- Regarding the target group configurations, choosing IP addresses as the target type allows flexibility in routing traffic to running containers. This approach is particularly useful when deploying on ECS with Fargate. Setting the port and protocol to HTTP on specific ports (8080 for customer microservices, 8081 for employee microservices) aligns with the port configurations in previous phases. Configuring health checks ensures that the load balancer can determine the health of each target group. This is crucial for maintaining high availability and reliability in a microservices architecture.
- Giving each target group a distinct name (**`customer-tg-one`, `customer-tg-two`, `employee-tg-one`, `employee-tg-two`**) allows for easy identification and management during deployment and troubleshooting.
- In the next task, I created a Security Group and an Application Load Balancer and Configure Rules to Route Traffic
- Creating a security group with rules allowing TCP traffic from any IPv4 address on ports 80 and 8080 provides the necessary access for public-facing applications. This setup is crucial for an internet-facing load balancer.
- An internet-facing Application Load Balancer allows customers and employees to access the application from the internet. This setup is suitable for a scalable and publicly accessible microservices architecture. By selecting the correct VPC and subnets ensures that the load balancer is configured to interact with your ECS cluster and other resources.
- Setting up listeners on ports 80 and 8080 provides the entry points for web traffic. The listeners forward traffic to the appropriate target groups based on the path rules.
- Setting the default forward targets ensures that general traffic is directed to the correct microservice. For example, routing traffic on port 80 to `customer-tg-two` and on port 8080 to `customer-tg-one`. Adding a rule to forward traffic with `/admin/*` in the path to the correct employee microservice target group ensures that administrative functions are directed to the appropriate service.
- All these tasks in Phase 6 focuses on creating the infrastructure needed to manage and route traffic in a microservices architecture. By setting up an Application Load Balancer with path-based routing and creating multiple target groups for blue/green deployments ensures that the application can handle traffic efficiently while supporting smooth transitions during deployments and updates.
- The configuration and routing rules established in this phase are critical for achieving high availability, scalability and resilience in a production-ready environment. Properly setting up these components lays the foundation for a robust and scalable microservices architecture.

Phase 7: Creating two Amazon ECS services

- The tasks in Phase 7 involves creating individual Amazon ECS services for each microservice to ensure independent deployment, management and scaling. This phase focuses on setting up the ECS services with the correct configurations which is crucial for a scalable microservices architecture.
- I started off by Creating the ECS Service for the Customer Microservice. The `create-customer-microservice-tg-two.json` file contains the necessary configuration to create an ECS service for the customer microservice.
- The task definition refers to the specific revision of the customer-microservice that ECS uses to create the service. The revision number should match the latest registered version.
- The configuration includes the ARN for the customer-tg-two target group and specifies the container name (`customer`) and port (8080). By setting the correct subnets, security groups, and enabling public IP assignment ensures proper connectivity and security.Using the blue/green deployment strategy with CodeDeploy allows for safer updates with minimal downtime and rollback capabilities.
- Running the `**aws ecs create-service**` command with the configured JSON file creates the ECS service for the customer microservice. This step is critical for deploying the microservice on ECS with the specified configuration.
- In the second task, I created the Amazon ECS Service for the Employee Microservice. By copying the `create-customer-microservice-tg-two.json` file and updating it for the employee microservice, you ensure consistency in the configuration process.
- The employee microservice should have its own task definition with the correct revision number. The ARN for the employee-tg-two target group is unique and the container name changes to `employee`. Running the appropriate AWS CLI command with the modified JSON file creates the ECS service for the employee microservice. This step is critical for deploying the employee microservice on ECS with the specified configuration.
- All these steps in Phase 7 establishes the ECS services for the microservices which allowing for independent management and deployment. Creating separate services for `customer` and `employee` ensures flexibility and scalability enabling you to scale each microservice as needed without affecting the other.
- By setting the correct task definitions, load balancers, target groups, network configurations and deployment strategies I can lay the groundwork for a robust and scalable ECS deployment. The blue/green deployment strategy and CodeDeploy integration offer a safer approach to rolling out changes, reducing downtime and providing a rollback mechanism if needed.

Phase 8: Configuring CodeDeploy and CodePipeline

- Phase 8 focuses on configuring CodeDeploy and CodePipeline to automate deploying microservices to Amazon ECS using blue/green deployment strategies for smooth updates.
- I created a CodeDeploy application named "microservices" with Amazon ECS as the compute platform. This application manages deployment groups and orchestrates deployments. I defined groups for both `customer-microservice` and `employee-microservice`. These groups specify ECS services, load balancers, target groups and traffic rerouting configurations for CodeDeploy to execute blue/green deployments.
- In the next task, I set up a pipeline for **customer microservice** with CodeCommit as the source and CodeDeploy as the deploy provider. This creates a CI/CD pipeline for continuous integration and deployment. Adding Amazon ECR as a source allows pipeline triggers when new Docker images are pushed to ECR. The pipeline is configured to dynamically update the ECS task definition with the correct image name.
- Next, I Manually released a test deployment to ensure proper configuration and validate the CI/CD pipeline. This step verifies that CodeDeploy manages the deployment correctly and checking how CodeDeploy dynamically adjusts load balancer rules during deployment. The traffic rerouting process should work as expected.
- Similarly, I created a pipeline for **employee-microservice** with similar settings as the customer pipeline. This independent pipeline allows flexible deployment and scaling. Adding Amazon ECR as a Source include the ECR repository for the employee microservice to trigger the pipeline when Docker images are updated.
- Similar to the customer pipeline, manually releasing a test deployment and validating the process. Ensuring the employee microservice deploys correctly with blue/green strategy. I then tested the deployed microservice in a browser, observed running tasks in ECS, and confirmed proper traffic rerouting in CodeDeploy.
- CodeDeploy actively manages the load balancer listener rules during blue/green deployments. Monitor the reassociation of target groups to ensure smooth transitions.Lastly, verifying that CodeDeploy manages load balancer rules effectively, ensuring traffic is routed to the correct microservice.
- All these tasks in Phase 8 established CI/CD pipelines for automated deployment of microservices to Amazon ECS. CodeDeploy's blue/green strategy ensures safe updates while CodePipeline provides a flexible framework for continuous integration and deployment. Proper configuration and testing are crucial for a robust, scalable and resilient microservices architecture.

Phase 9: Adjusting the microservice code to cause a pipeline to run again

- The tasks of Phase 9 demonstrates the flexibility and scalability of microservices by updating the load balancer listener rules, modifying the source code of the employee microservice, causing the CI/CD pipeline to run and scaling the number of containers for the customer microservice.
- Firstly, I Limit the Access to the Employee Microservice by Modifying the rules for the HTTP:80 and HTTP:8080 listeners to limit access to the employee microservice to a specific IP address. This enhances security by allowing only authorized users to access the employee microservice's admin pages. Adding a condition to route traffic to the employee microservice only if the source IP matches the specified address. This restriction helps secure sensitive areas of the application.
- Next, I Adjusted the UI for the Employee Microservice and Pushed the Updated Image to Amazon ECR. I Changed the navbar style in `employee/views/nav.html` to update the appearance of the employee microservice. This demonstrates how we can independently modify microservices without affecting others.
- After making the code change, I built a new Docker image and pushed it to Amazon ECR. This action triggers the CodePipeline causing the update-employee-microservice pipeline to run and deploy the updated microservice.The pipeline automatically updates the running deployment with the new Docker image illustrating the benefits of automation and continuous integration.
- After pushing the updated Docker image, observe the pipeline in CodePipeline. It should automatically trigger and proceed through the deployment stages.We can also monitor the deployment in the CodeDeploy console to ensure successful completion. This step validates that the pipeline and deployment process function as intended.
- I Tested access to the employee microservice from the authorized IP address Verifying that the page loads correctly and displays the updated UI.Also,I attempted to access the employee microservice from a different IP address cellular network from my phone (the screenshot is given in the project execution phase). It resulted in a 404 error, indicating that the load balancer rules effectively restrict access.
- Lastly, I updated the desired count for the customer microservice to increase the number of running containers. This step demonstrates the scalability of ECS and the ability to adjust the capacity of individual microservices.
- In addition, ccaling the customer microservice independently from the employee microservice shows the benefits of a microservices architecture allowing me to respond to varying workloads without impacting other services.
- All these tasks in Phase 9 showcases the benefits of a microservices architecture, enabling to update, scale and control access to individual microservices with minimal impact on the overall system. By leveraging CodePipeline and CodeDeploy, I achieved automated deployment and a robust CI/CD pipeline. The ability to scale services independently and rollback deployments as needed provides flexibility and resilience, key aspects of a successful microservices-based application.
