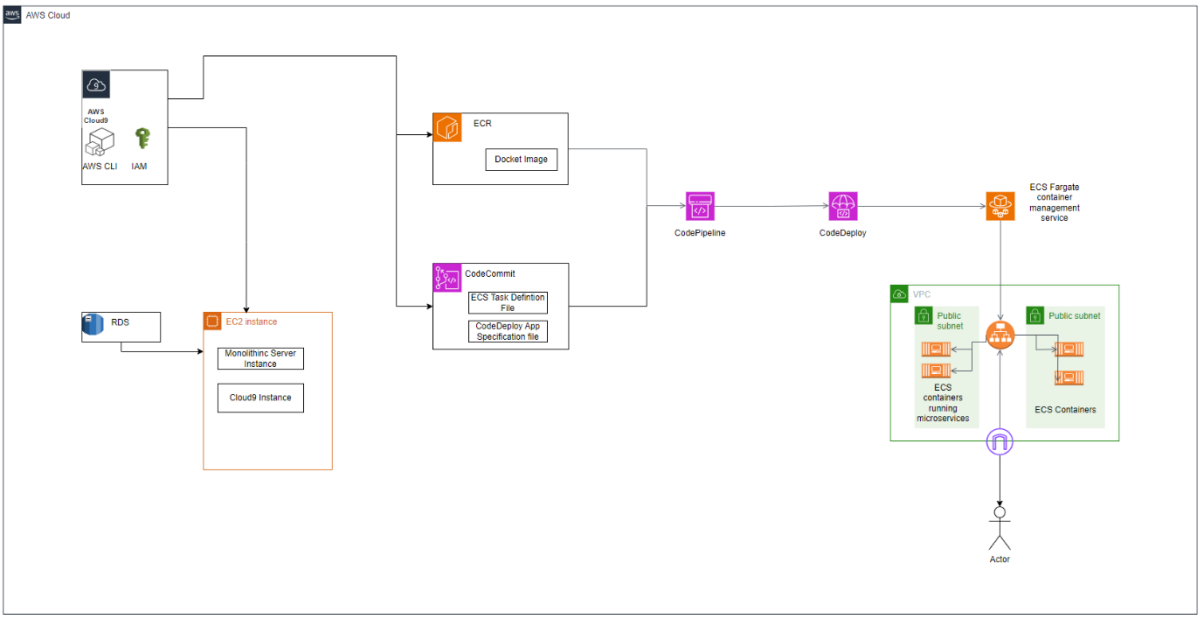
CS 524 – Course Project

Submitted By – Shweta Madhale

Project Topic - Building Microservices and a CI/CD Pipeline with AWS

Execution

Phase 1 - Planning the design and estimating cost

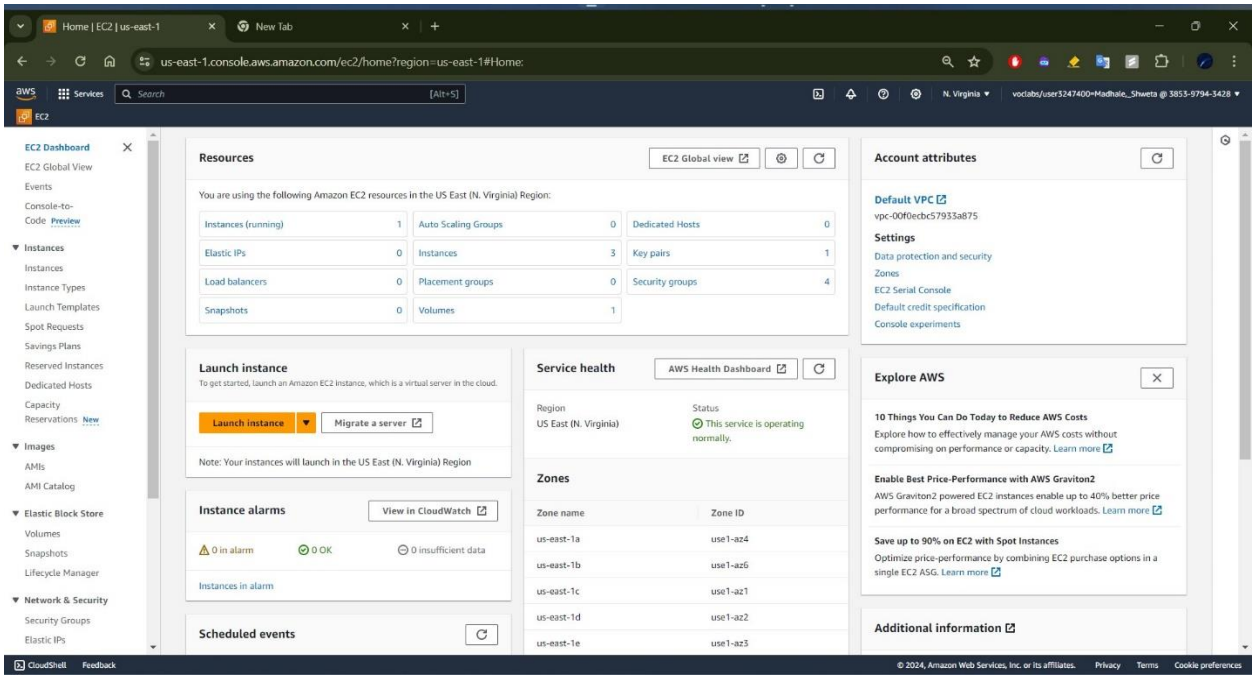


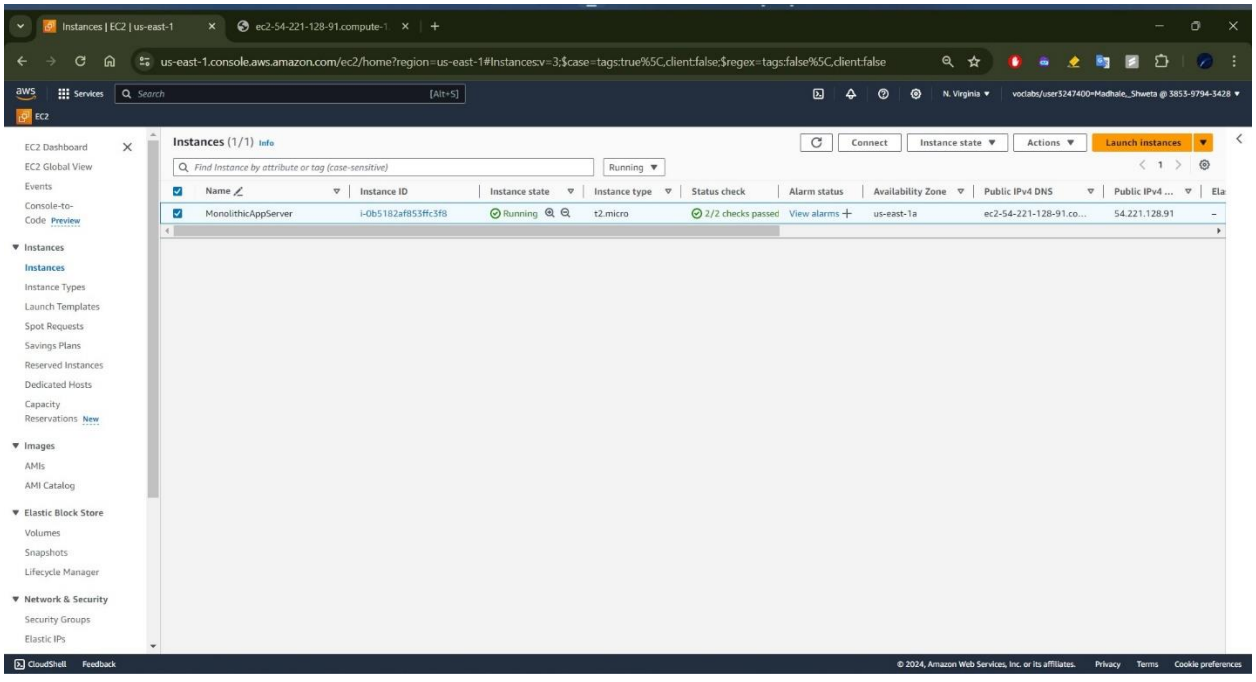
ECS is around $20/month, EC2 is $50/month, RDS is $30/month So deployment will take roughly $100/ month.

Phase 2 – Analyze the design of the monolithic application and test the application.

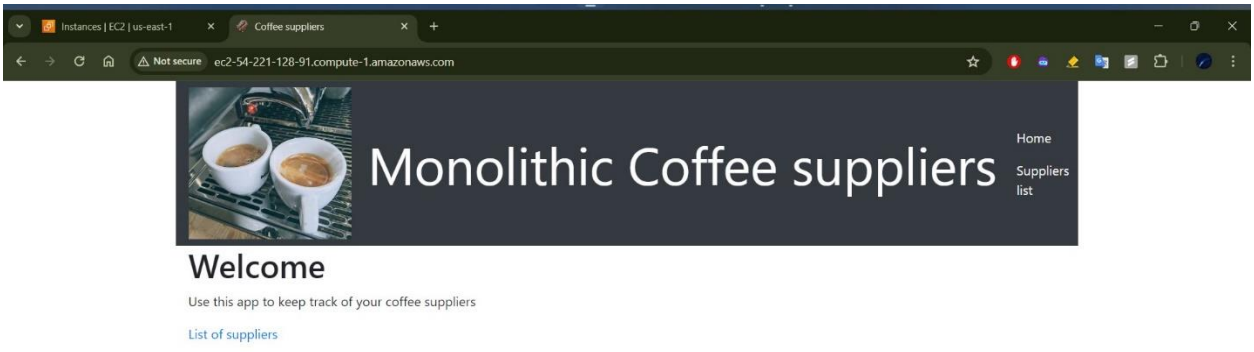
Task 2.1: Verify that the monolithic application is available

- Navigate to the Amazon EC2 console.



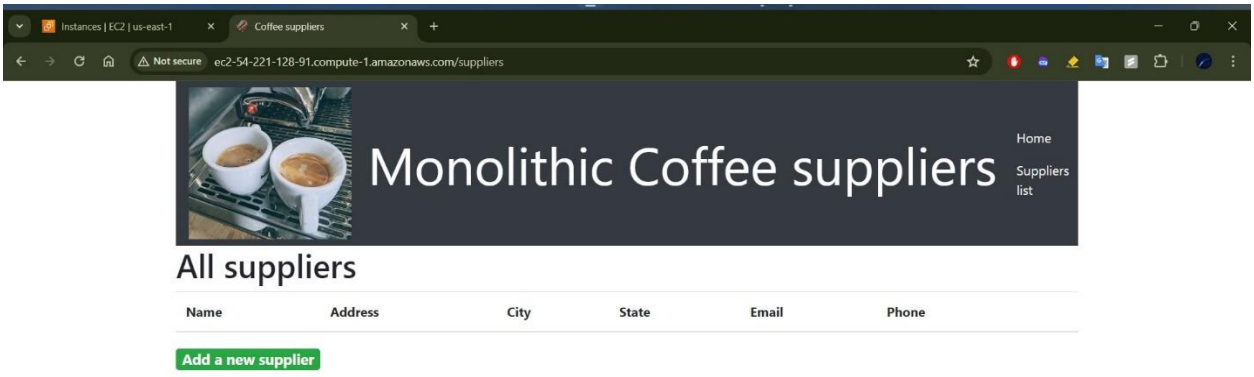


- Copy the Public IPv4 address of the MonolithicAppServer instance, and load it in a new browser tab. The coffee suppliers website displays.

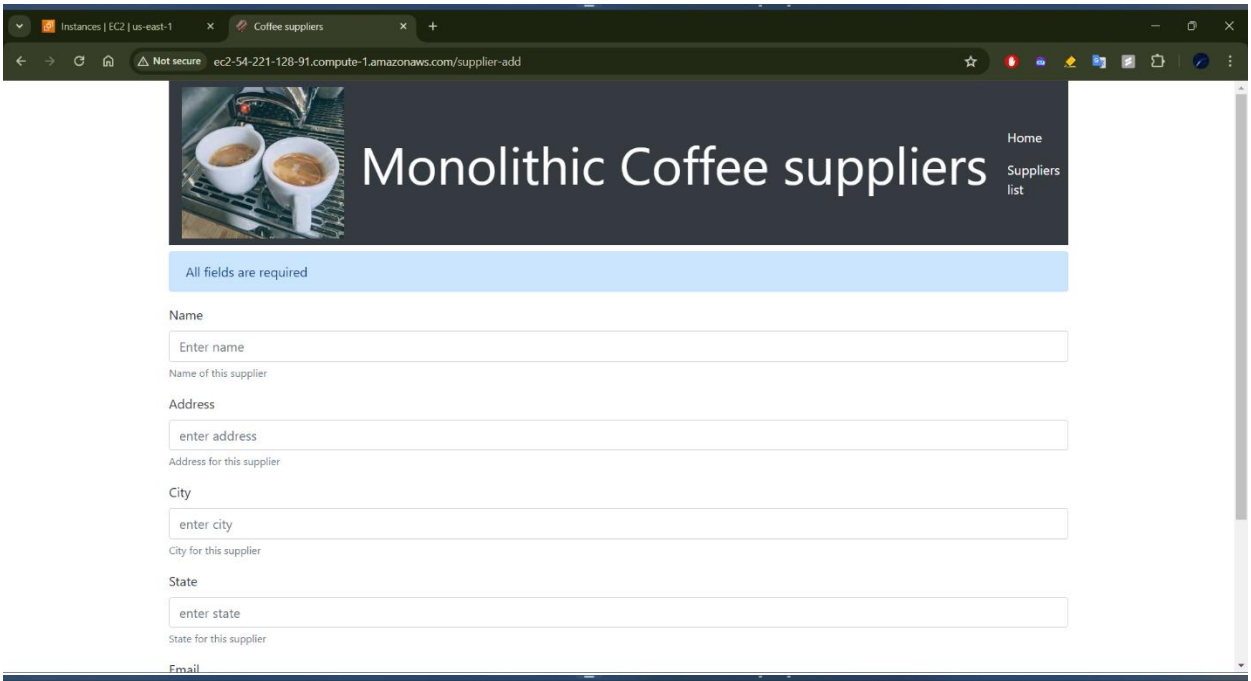


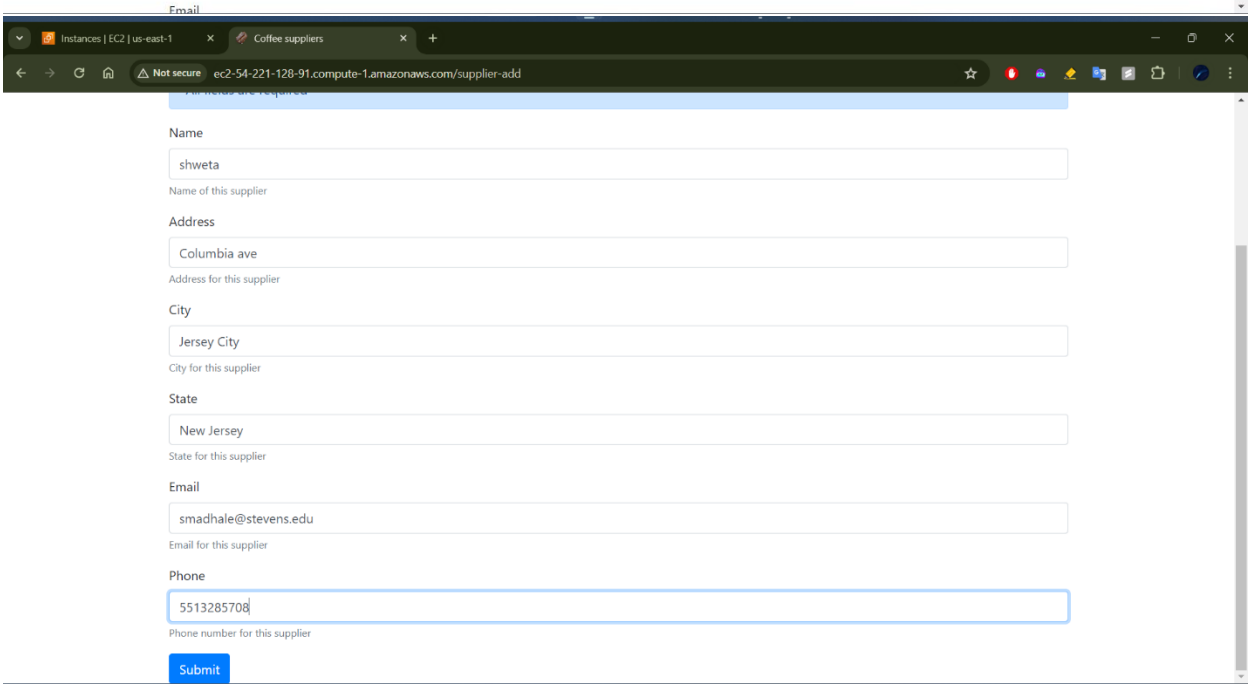
Task 2.2: Test the monolithic web application

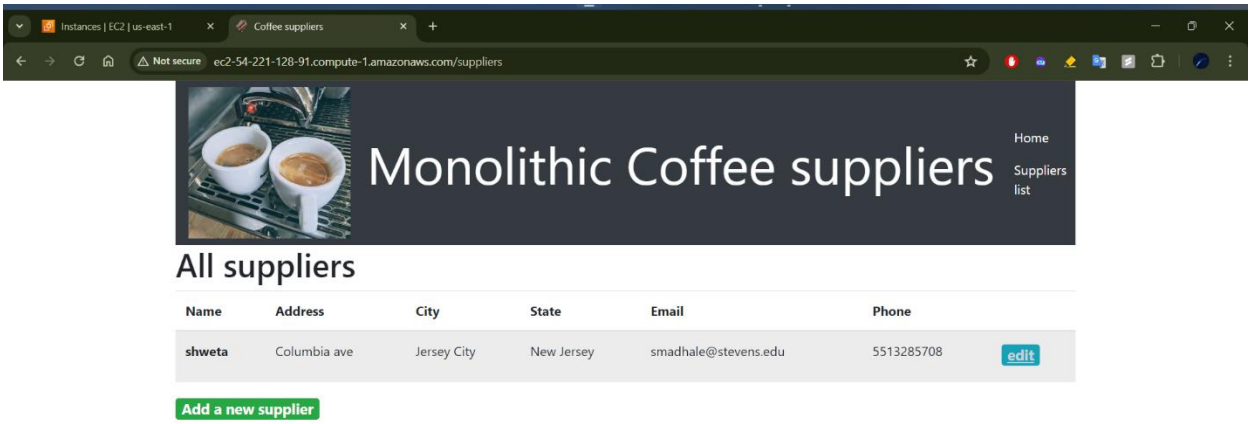
- Choose List of suppliers and notice that the URL path includes /suppliers.



- Add a new supplier, on the page where you add a new supplier, notice that the URL path includes /supplier-add. Fill in all of the fields to create a supplier entry.

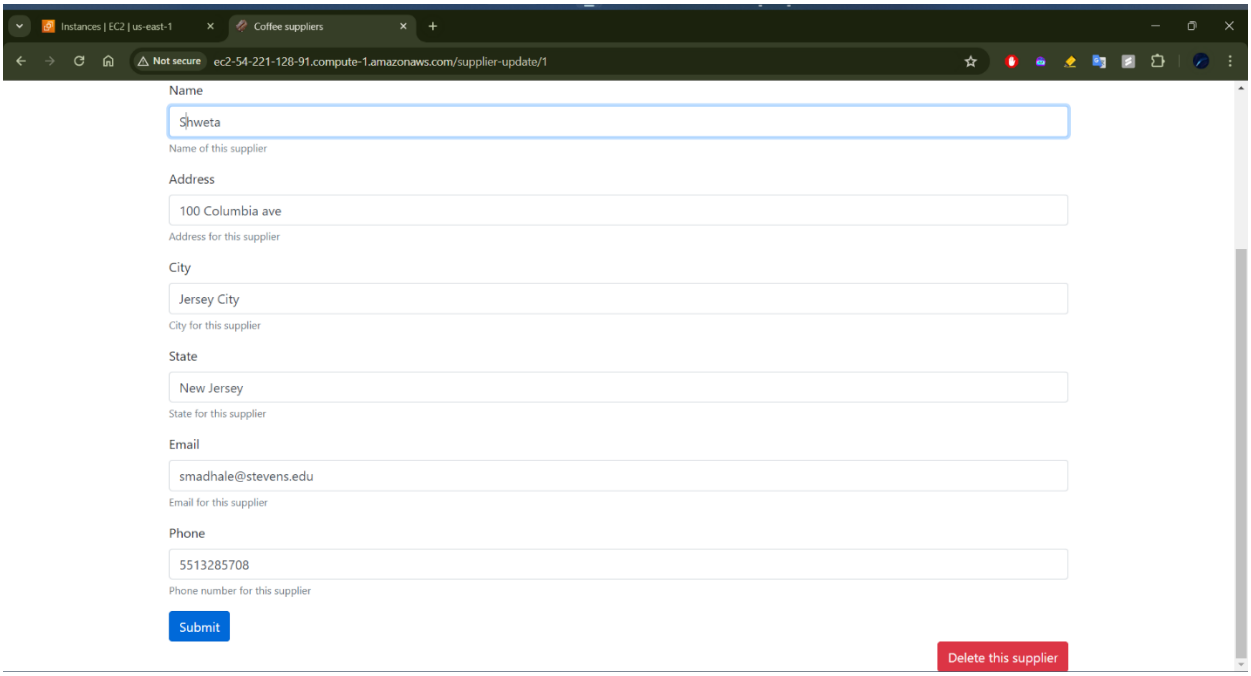

Instances | EC2 | us-east-1
Coffee suppliers
Not secure
ec2-54-221-128-91.compute-1.amazonaws.com/supplier-add
Monolithic Coffee suppliers
Home
Suppliers list
All fields are required
Name
Enter name
Name of this supplier
Address
enter address
Address for this supplier
City
enter city
City for this supplier
State
enter state
State for this supplier
Email

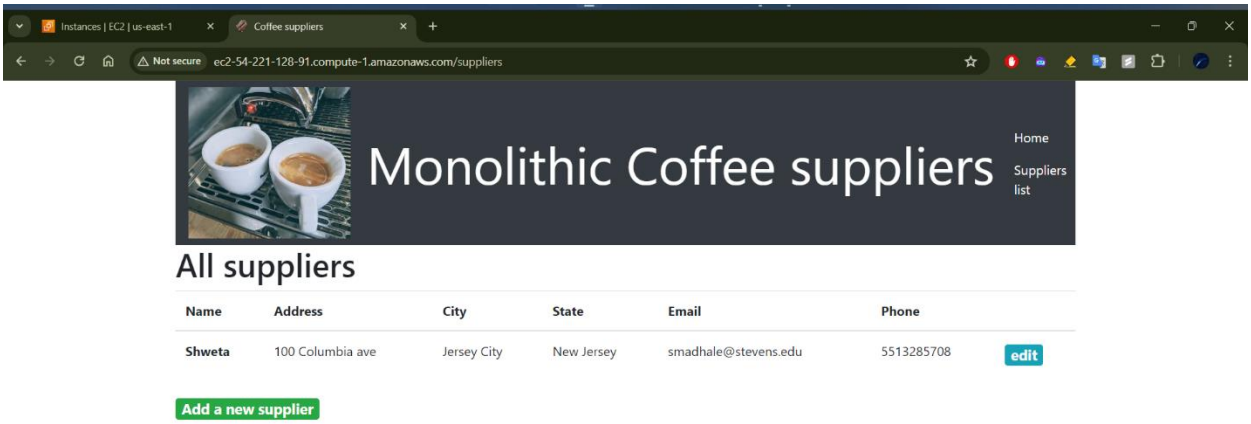

Instances | EC2 | us-east-1
Coffee suppliers
Not secure
ec2-54-221-128-91.compute-1.amazonaws.com/supplier-add
Name
shweta
Name of this supplier
Address
Columbia ave
Address for this supplier
City
Jersey City
City for this supplier
State
New Jersey
State for this supplier
Email
smadhale@stevens.edu
Email for this supplier
Phone
5513285708
Phone number for this supplier
Submit



ec2-54-221-128-91.compute-1.amazonaws.com/supplier-update/1

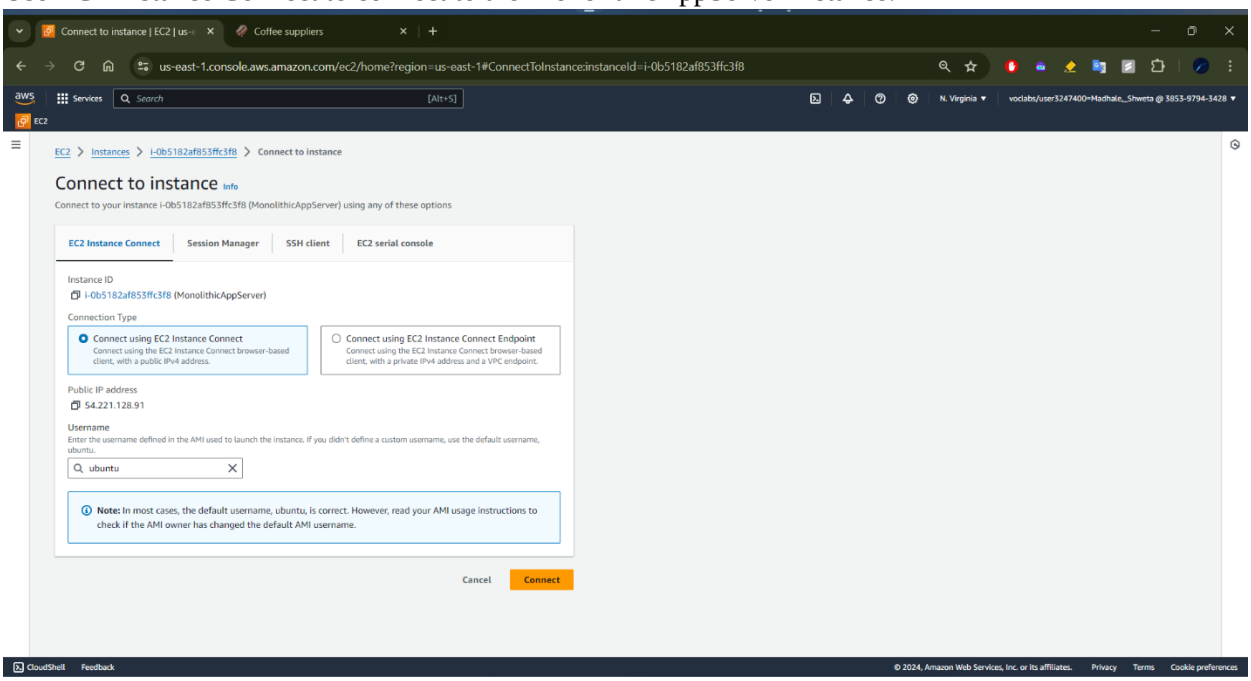
- Edit an entry. On the page where you edit a supplier entry, notice that the URL path now includes supplier-update/1. Modify the record in some way and save the change. Notice that the change was saved in the record.





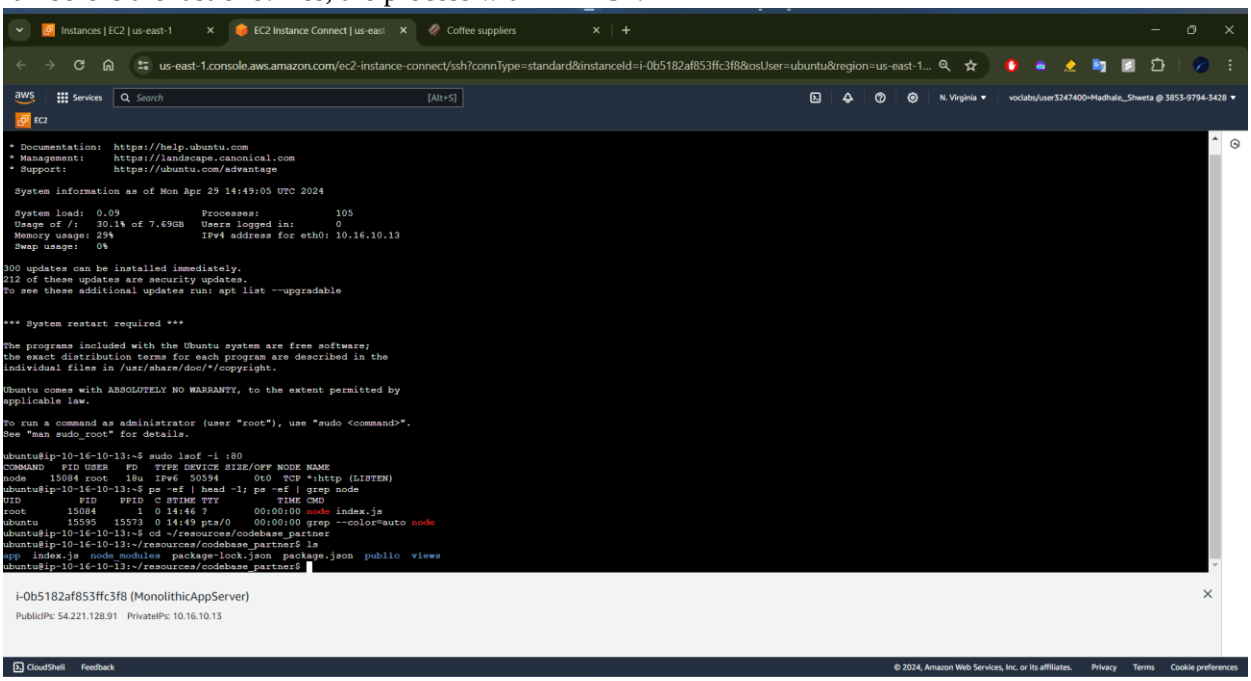
Task 2.3: Analyze how the monolithic application runs

- Use EC2 Instance Connect to connect to the MonolithicAppServer instance.

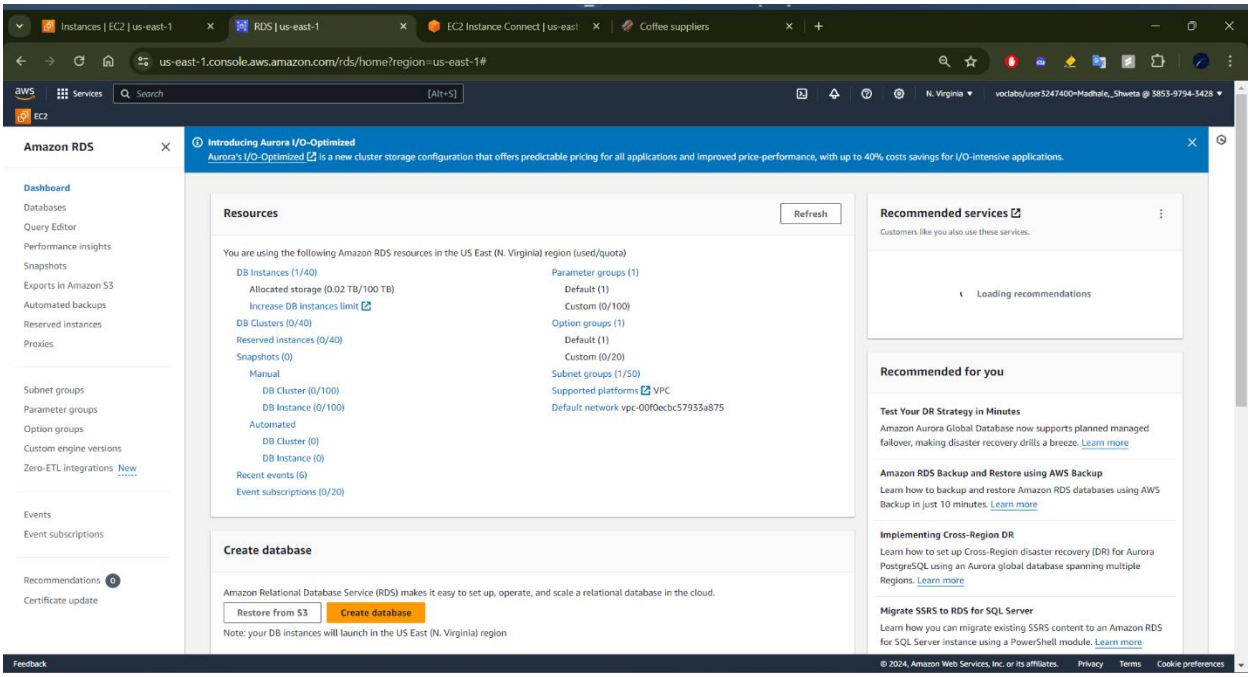


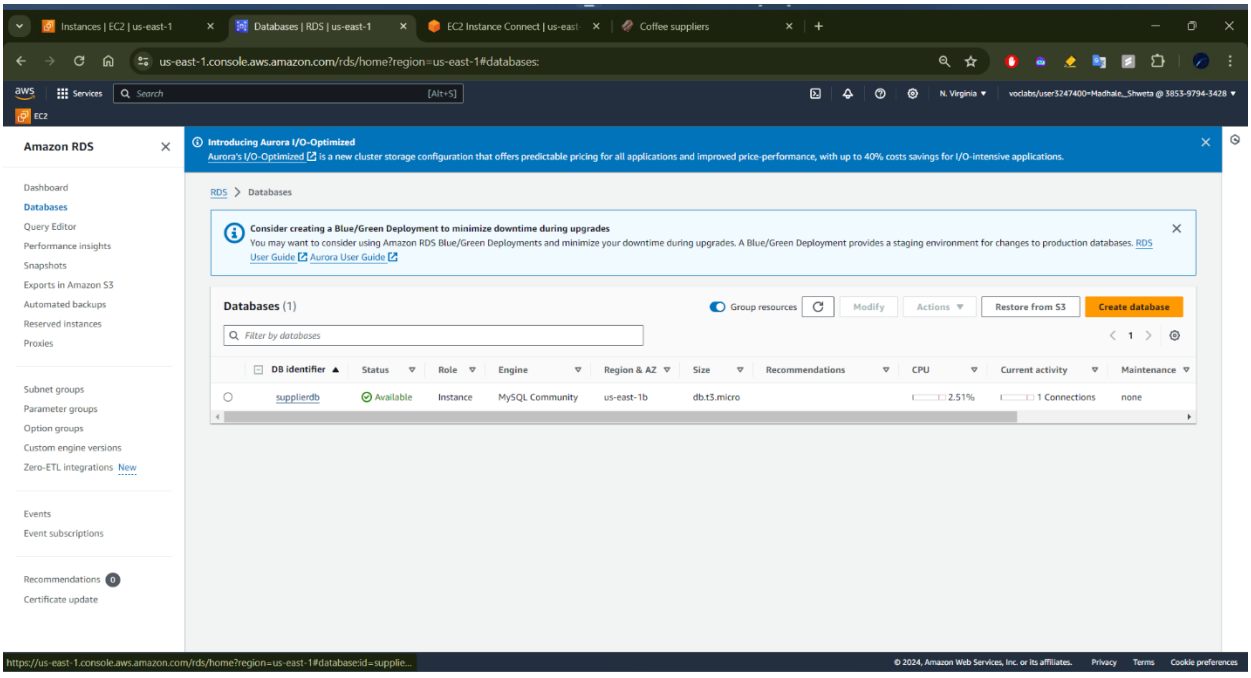
- Analyze how the application is running. In the terminal session, running the following command - **sudo lsof -i :80**
- What did you notice in the command output? What port and protocol are the node daemon using? Node.js daemon (process) is listening on port 80 using the TCP protocol.
- Next, running the following command - **ps -ef | head -1; ps -ef | grep node**.
- What did you notice in the command output? The output is list of currently running processes and it is filtered to show only the processes which have the node term.

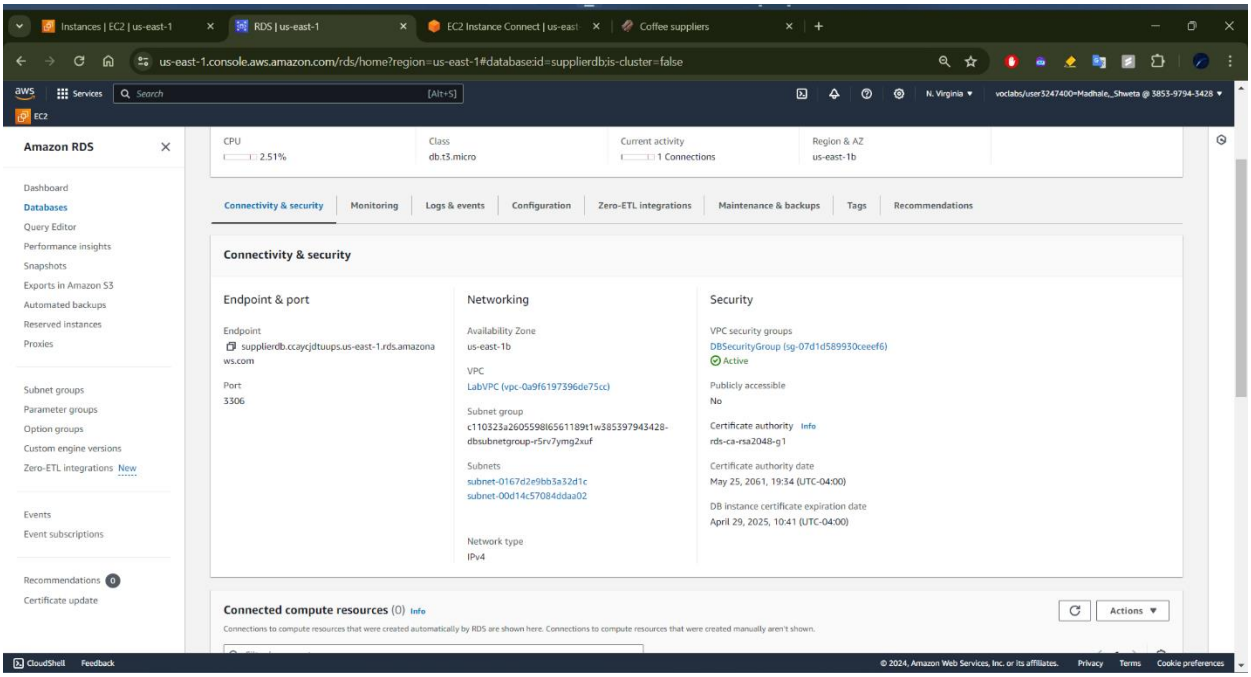
- Which user on this EC2 instance is running a node process? The root user is running the node process.
- Does the node process ID (PID) match any of the PIDs from the output of the command that you ran before the last one? Yes, the process with PID 432.



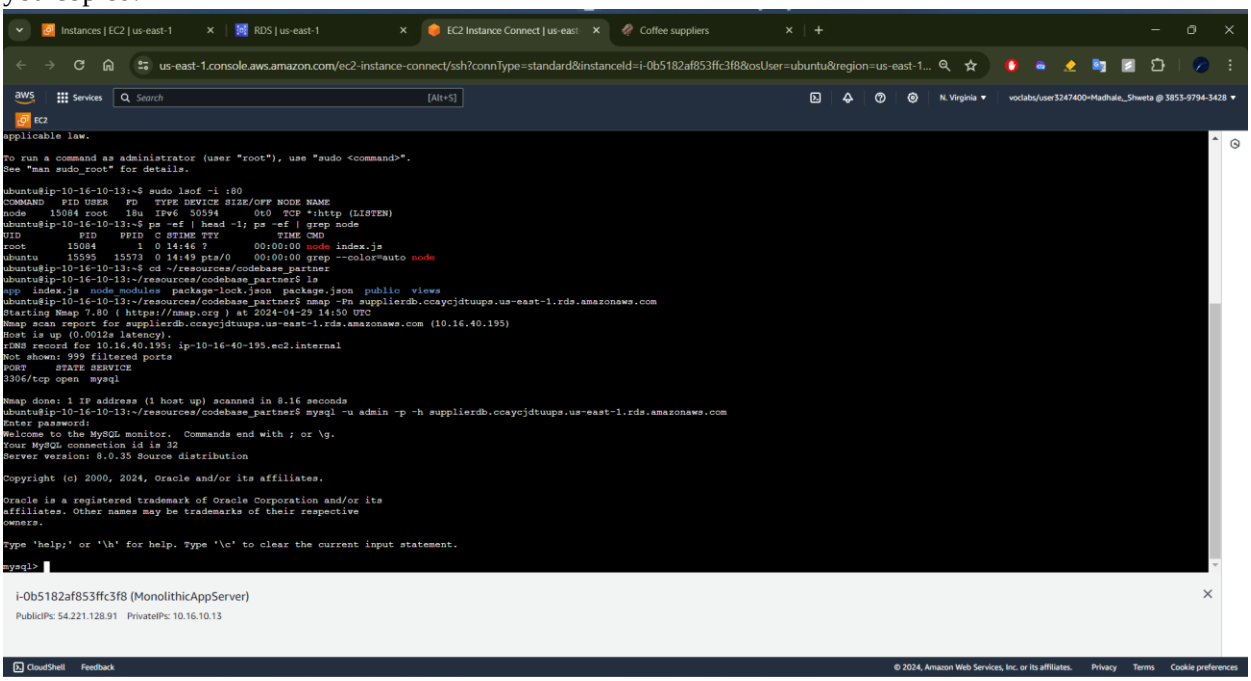
- To analyze the application structure, run the following commands: **cd ~/resources/codebase_partner, ls.** This is where the index.js file exists. It contains the base application logic.
- Questions for thought: Based on what you have observed, how and where this node application is running? resources/codebase_partner.
- Connect a MySQL client to the RDS database that the node application stores data in.
- Find and copy the endpoint of the RDS database that is running in the lab environment.


Instances | EC2 | us-east-1
RDS | us-east-1
EC2 Instance Connect | us-east
Coffee suppliers
us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Amazon RDS
Introducing Aurora I/O-Optimized
Aurora's I/O-Optimized is a new cluster storage configuration that offers predictable pricing for all applications and improved price-performance, with up to 40% costs savings for I/O-intensive applications.
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Events
Event subscriptions
Recommendations 0
Certificate update
Resources
Refresh
You are using the following Amazon RDS resources in the US East (N. Virginia) region (used/quota)
DB Instances (1/40)
Allocated storage (0.02 TB/100 TB)
Increase DB instances limit
DB Clusters (0/40)
Reserved instances (0/40)
Snapshots (0)
Manual
DB Cluster (0/100)
DB Instance (0/100)
Automated
DB Cluster (0)
DB Instance (0)
Recent events (6)
Event subscriptions (0/20)
Parameter groups (1)
Default (1)
Custom (0/100)
Option groups (1)
Default (1)
Custom (0/20)
Subnet groups (1/50)
Supported platforms VPC
Default network vpc-00f0ecbc57933a875
Recommended services
Customers like you also use these services.
Loading recommendations
Recommended for you
Test Your DR Strategy in Minutes
Amazon Aurora Global Database now supports planned managed failover, making disaster recovery drills a breeze. Learn more
Amazon RDS Backup and Restore using AWS Backup
Learn how to backup and restore Amazon RDS databases using AWS Backup in just 10 minutes. Learn more
Implementing Cross-Region DR
Learn how to set up Cross-Region disaster recovery (DR) for Aurora PostgreSQL using an Aurora global database spanning multiple Regions. Learn more
Migrate SSRS to RDS for SQL Server
Learn how you can migrate existing SSRS content to an Amazon RDS for SQL Server instance using a PowerShell module. Learn more
Create database
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud.
Restore from S3
Create database
Note: your DB instances will launch in the US East (N. Virginia) region
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

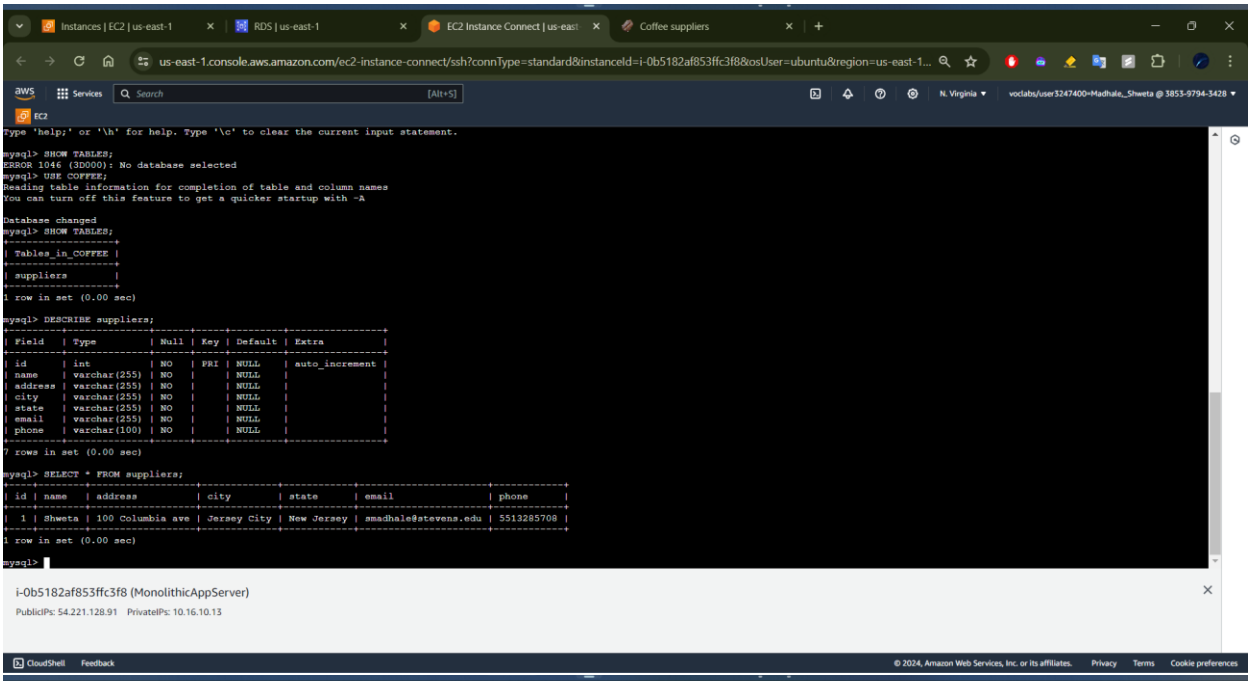

Instances | EC2 | us-east-1
Databases | RDS | us-east-1
EC2 Instance Connect | us-east
Coffee suppliers
us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#databases:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Amazon RDS
Introducing Aurora I/O-Optimized
Aurora's I/O-Optimized is a new cluster storage configuration that offers predictable pricing for all applications and improved price-performance, with up to 40% costs savings for I/O-intensive applications.
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Events
Event subscriptions
Recommendations 0
Certificate update
RDS > Databases
Consider creating a Blue/Green Deployment to minimize downtime during upgrades
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. RDS User Guide Aurora User Guide
Databases (1)
Group resources
Modify
Actions
Restore from S3
Create database
Filter by databases
1
DB identifier
Status
Role
Engine
Region & AZ
Size
Recommendations
CPU
Current activity
Maintenance
supplierdb
Available
Instance
MySQL Community
us-east-1b
db.t3.micro
2.51%
1 Connections
none
https://us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#database:id=supplie...
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

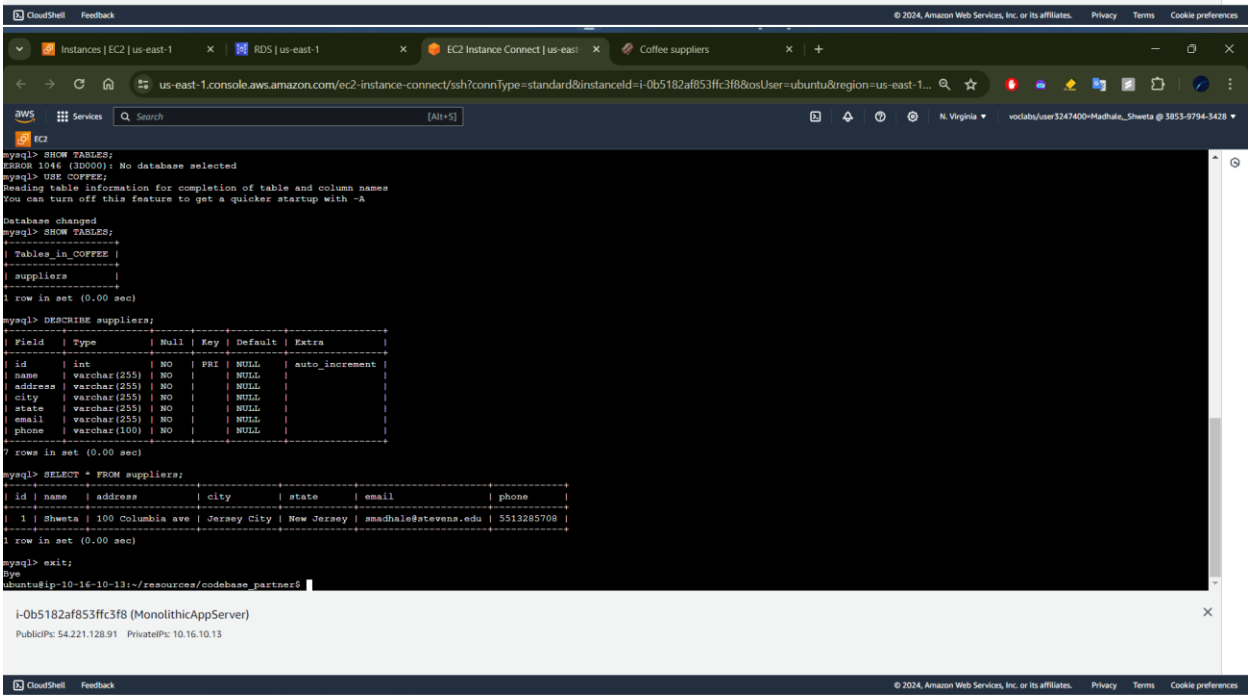


- To verify that the database can be reached from the MonolithicAppServer instance on the standard MySQL port number, use the nmap -Pn command with the RDS database endpoint that you copied.



- To connect to the database, use the MySQL client that is already installed on the MonolithicAppServer instance. Use the following values: Username: admin, Password: lab-password. Observe the data in the database. From the mysql> prompt, run SQL commands as appropriate to see that a database named COFFEE contains a table named suppliers. This table contains the supplier entry or entries that you added earlier when you tested the web application. Exit the MySQL client and then close the EC2 Instance Connect tab. Also close the coffee suppliers web application tab.

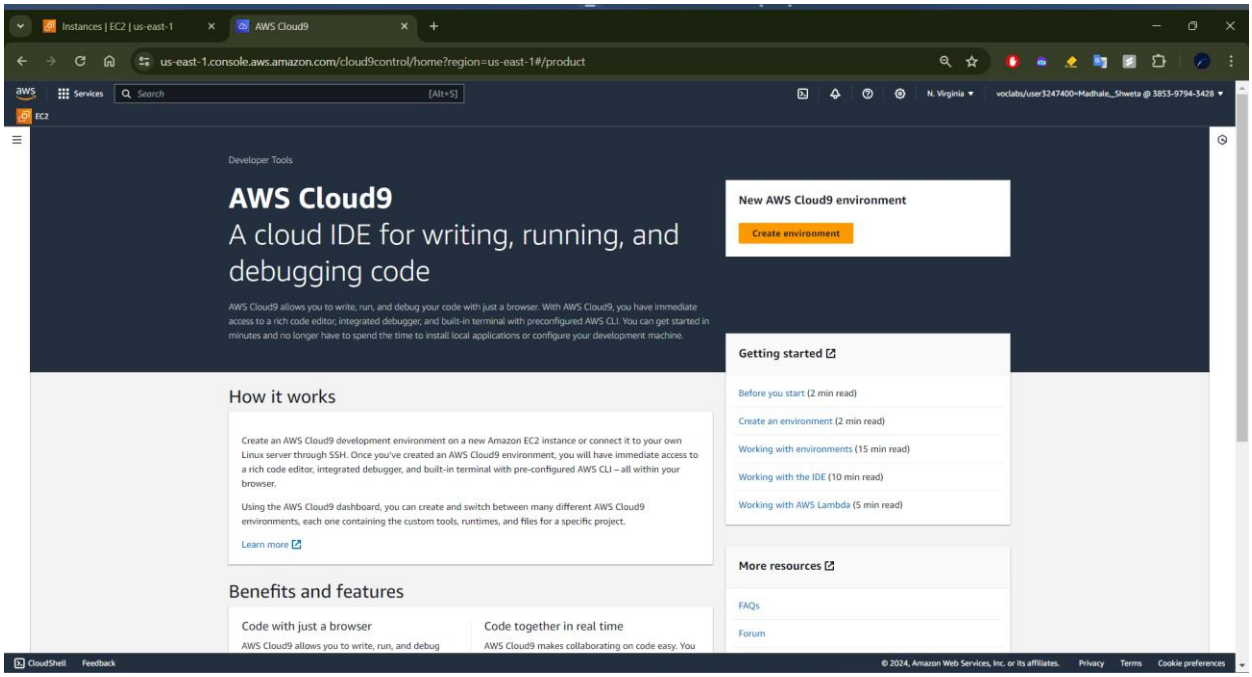


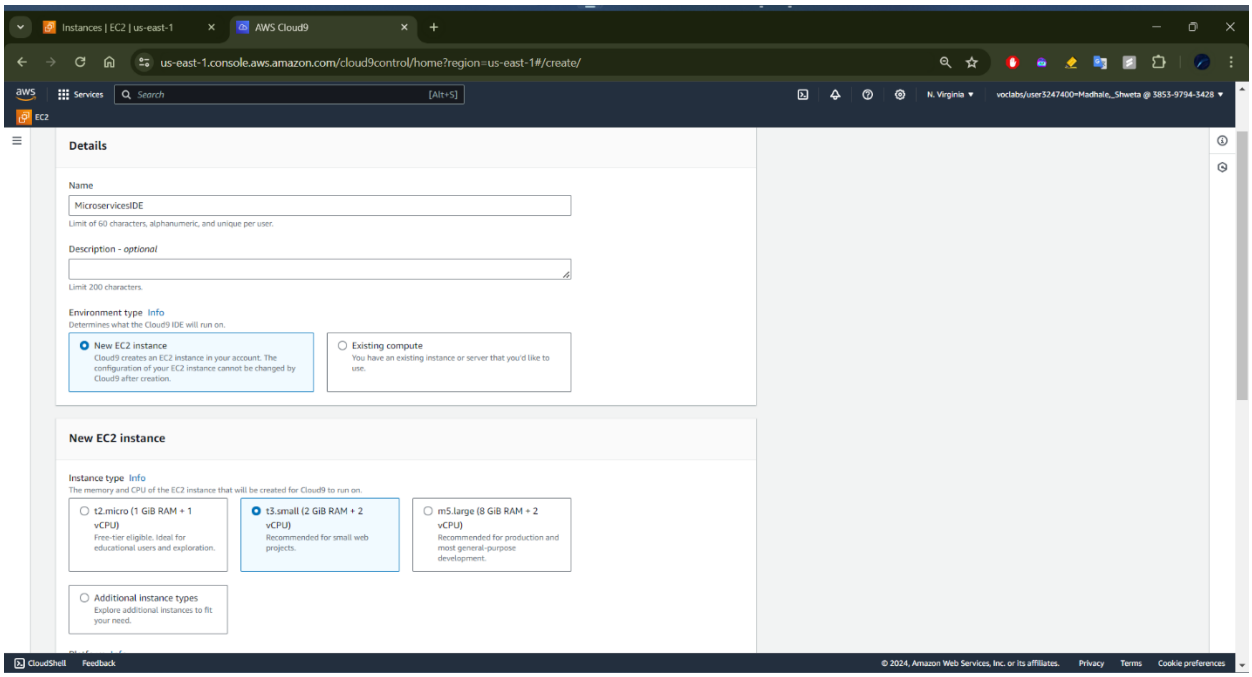


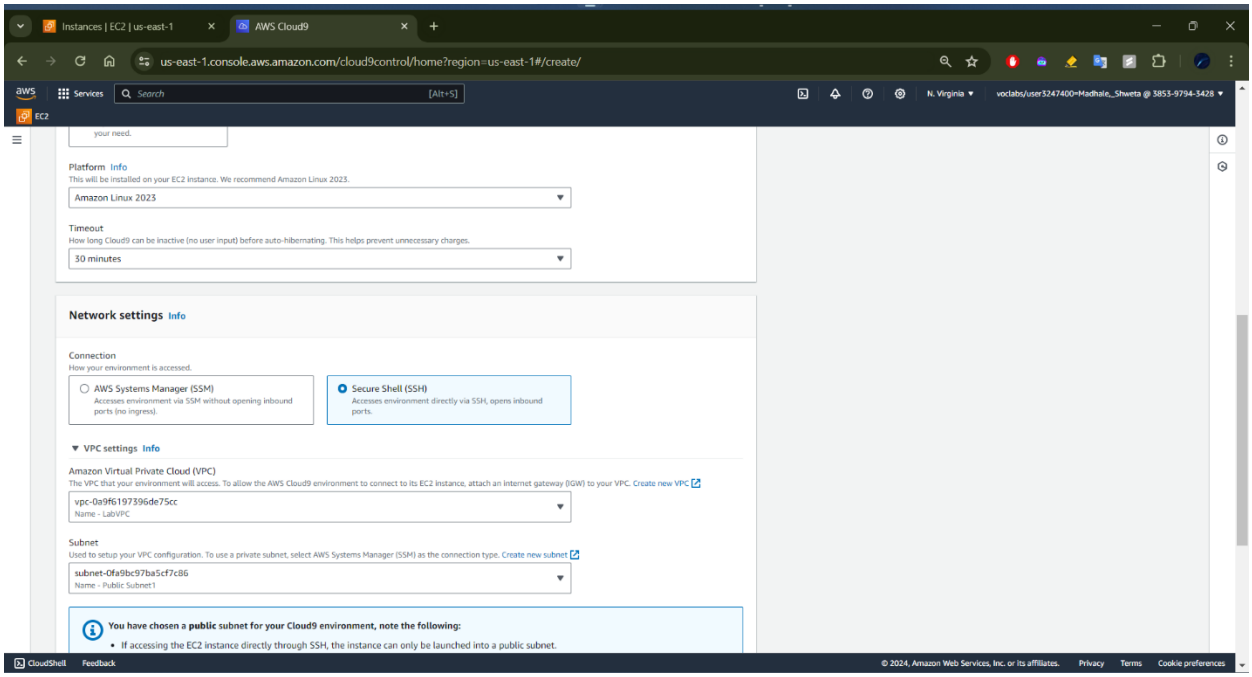
Phase 3 - Create a development environment on AWS Cloud9, and check the monolithic source code into CodeCommit.

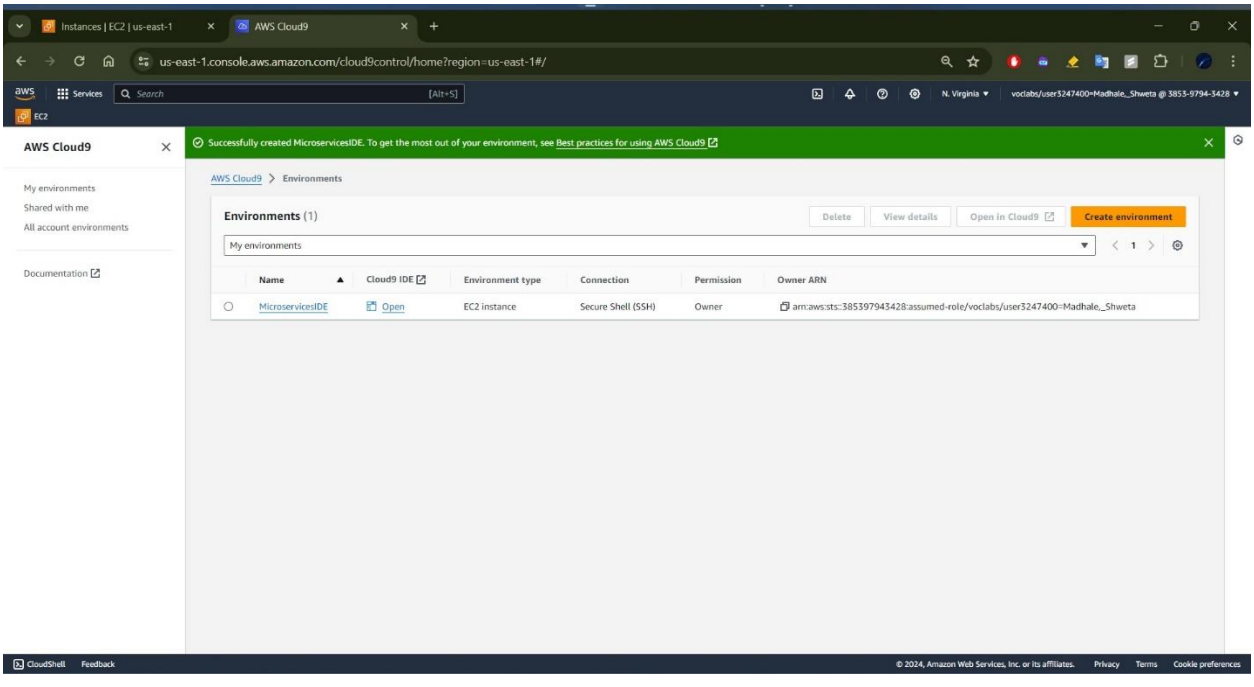
Task 3.1: Create an AWS Cloud9 IDE as your work environment

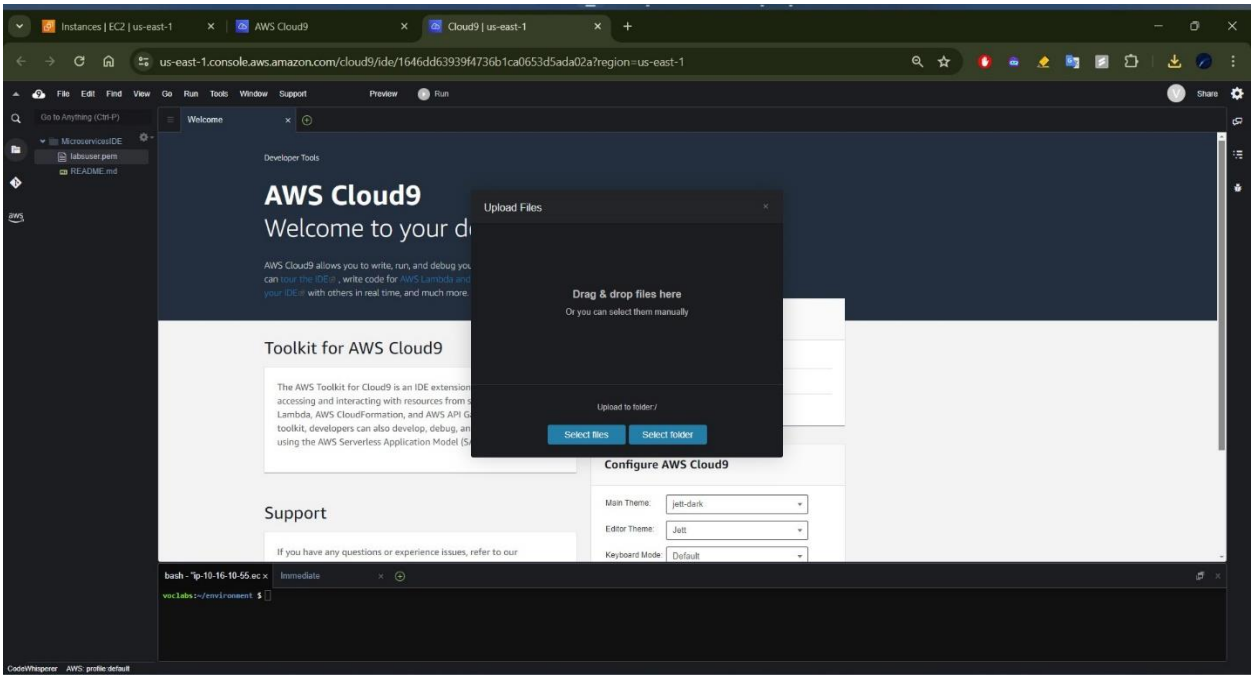
- Creating AWS Cloud9 instance named MicroservicesIDE, instance has size t3.small, runs on Amazon Linux 2, supports SSH connections and runs in the LabVPC in Public subnet1.


Instances | EC2 | us-east-1
AWS Cloud9
us-east-1.console.aws.amazon.com/cloud9control/home?region=us-east-1#/product
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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 AWS Cloud9 environments, each one containing the custom tools, runtimes, and files for a specific project.
Learn more
More resources
FAQs
Forum
Benefits and features
Code with just a browser
AWS Cloud9 allows you to write, run, and debug
Code together in real time
AWS Cloud9 makes collaborating on code easy. You
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | us-east-1
AWS Cloud9
us-east-1.console.aws.amazon.com/cloud9control/home?region=us-east-1#/create/
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
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.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

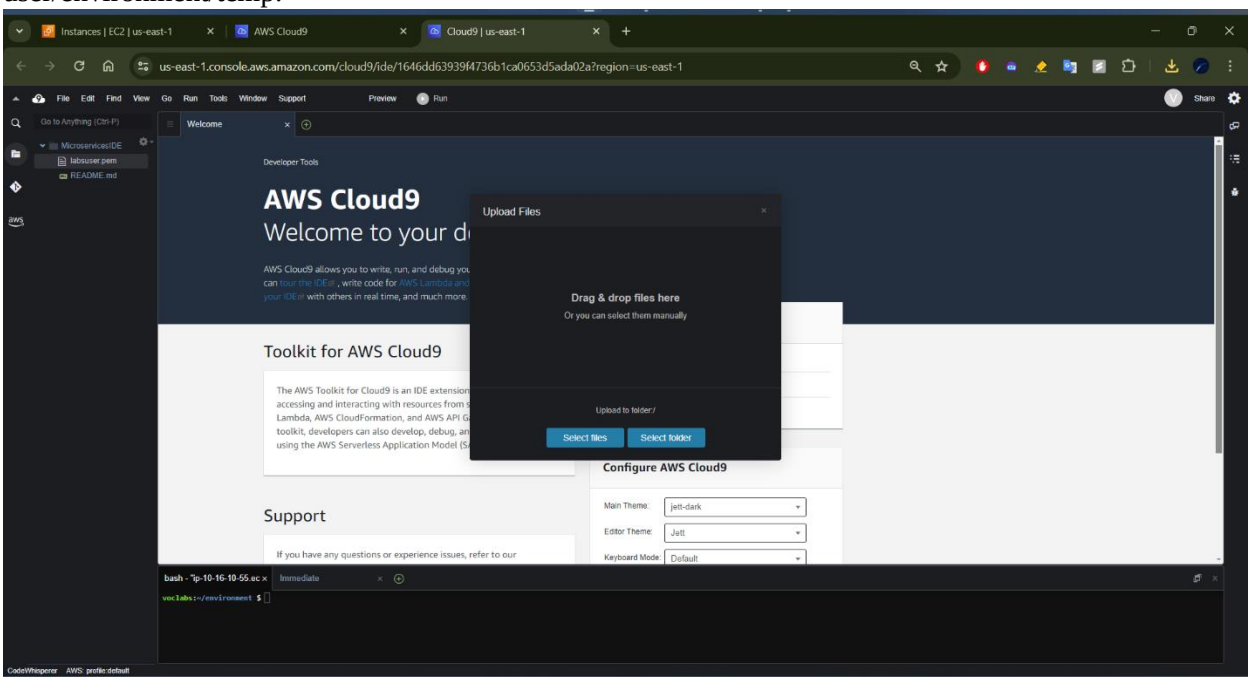

Instances | EC2 | us-east-1
AWS Cloud9
us-east-1.console.aws.amazon.com/cloud9control/home?region=us-east-1#/create/
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.
Amazon Linux 2023
Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes
Network settings Info
Connection
How your environment is accessed.
AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.
VPC settings Info
Amazon Virtual Private Cloud (VPC)
The VPC that your environment will access. To allow the AWS Cloud9 environment to connect to its EC2 instance, attach an internet gateway (IGW) to your VPC. Create new VPC
vpc-0a9f6197396de75cc
Name - LabVPC
Subnet
Used to setup your VPC configuration. To use a private subnet, select AWS Systems Manager (SSM) as the connection type. Create new subnet
subnet-0fa9bc97ba5cf7c86
Name - Public Subnet1
You have chosen a public subnet for your Cloud9 environment, note the following:
If accessing the EC2 instance directly through SSH, the instance can only be launched into a public subnet.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

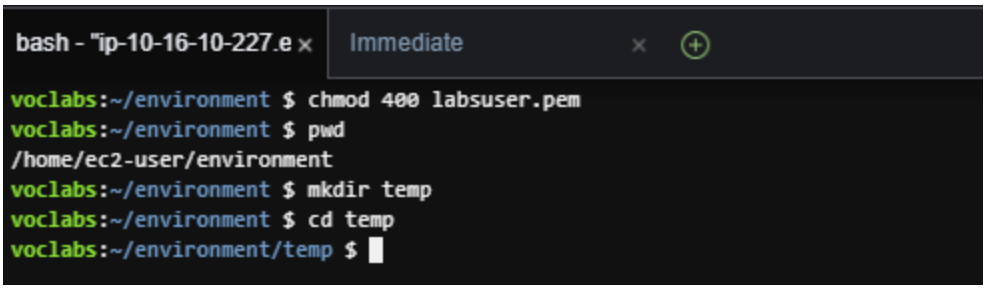

Instances | EC2 | us-east-1
AWS Cloud9
us-east-1.console.aws.amazon.com/cloud9control/home?region=us-east-1#/
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
AWS Cloud9
My environments
Shared with me
All account environments
Documentation
Successfully created MicroservicesIDE. To get the most out of your environment, see Best practices for using AWS Cloud9
AWS Cloud9 > Environments
Environments (1)
Delete
View details
Open in Cloud9
Create environment
My environments
Name
Cloud9 IDE
Environment type
Connection
Permission
Owner ARN
MicroservicesIDE
Open
EC2 instance
Secure Shell (SSH)
Owner
arn:aws:sts::38539794342 8:assumed-role/voclabs/user3247400=Madhale,_Shweta
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



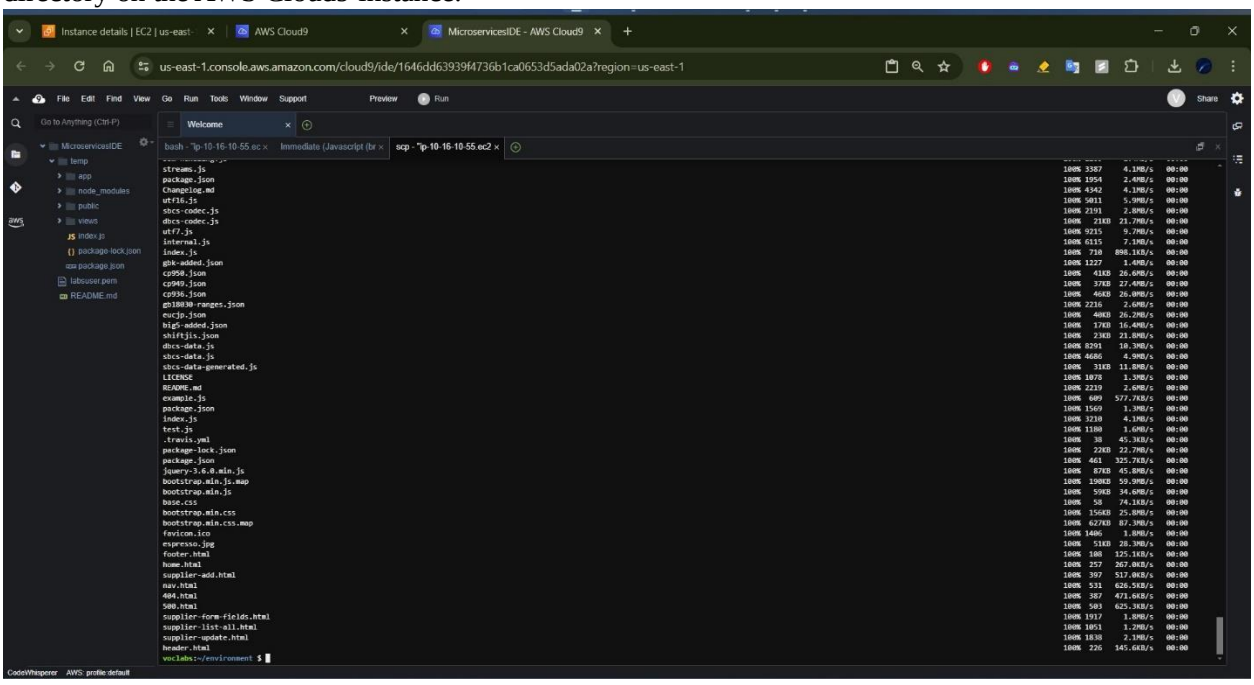
Task 3.2: Copy the application code to your IDE

- Download the labsuser.pem file and upload that to the AWS Cloud9 IDE and use chmod to set permissions. Create temp directory on the AWS Cloud9 instance at /home/ec2-user/environment/temp.



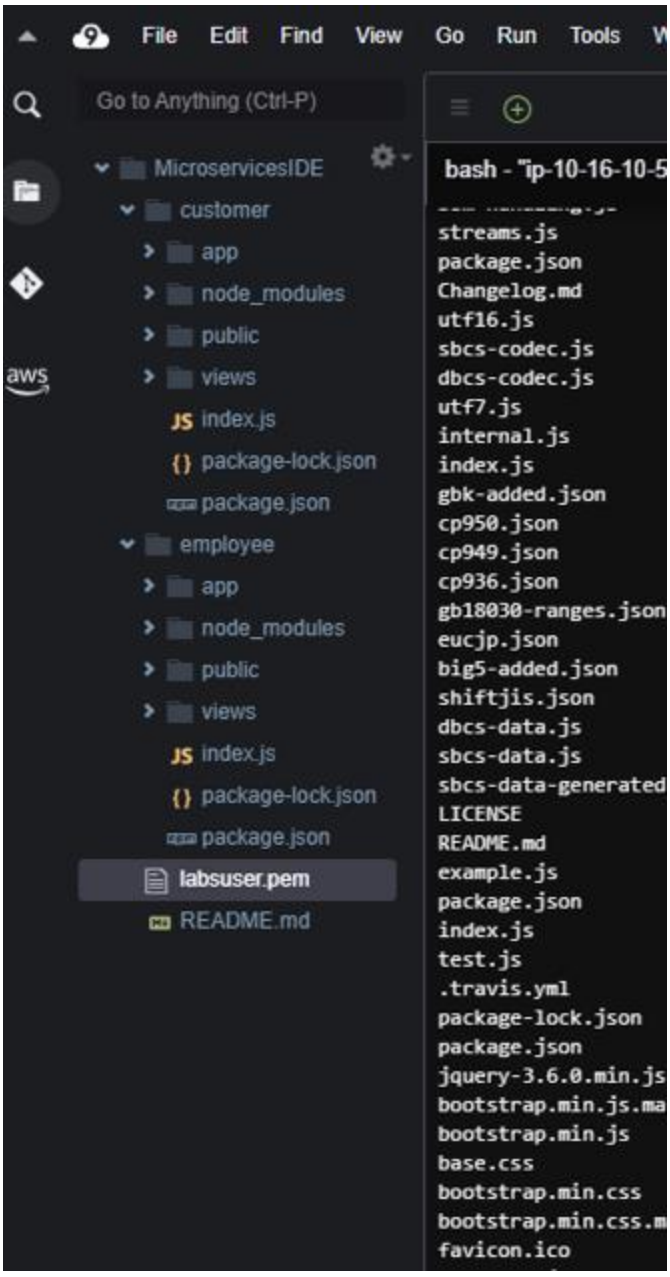


- From the Amazon EC2 console, retrieve the private IPv4 address of the MonolithicAppServer instance. Use scp -i ~/environment/labsuser.pem -r ubuntu@10.16.10.13:/home/ubuntu/resources/codebase_partner/* ~/environment/temp/ In the file browser of the IDE, verify that the source files for the application have been copied to the temp directory on the AWS Cloud9 instance.



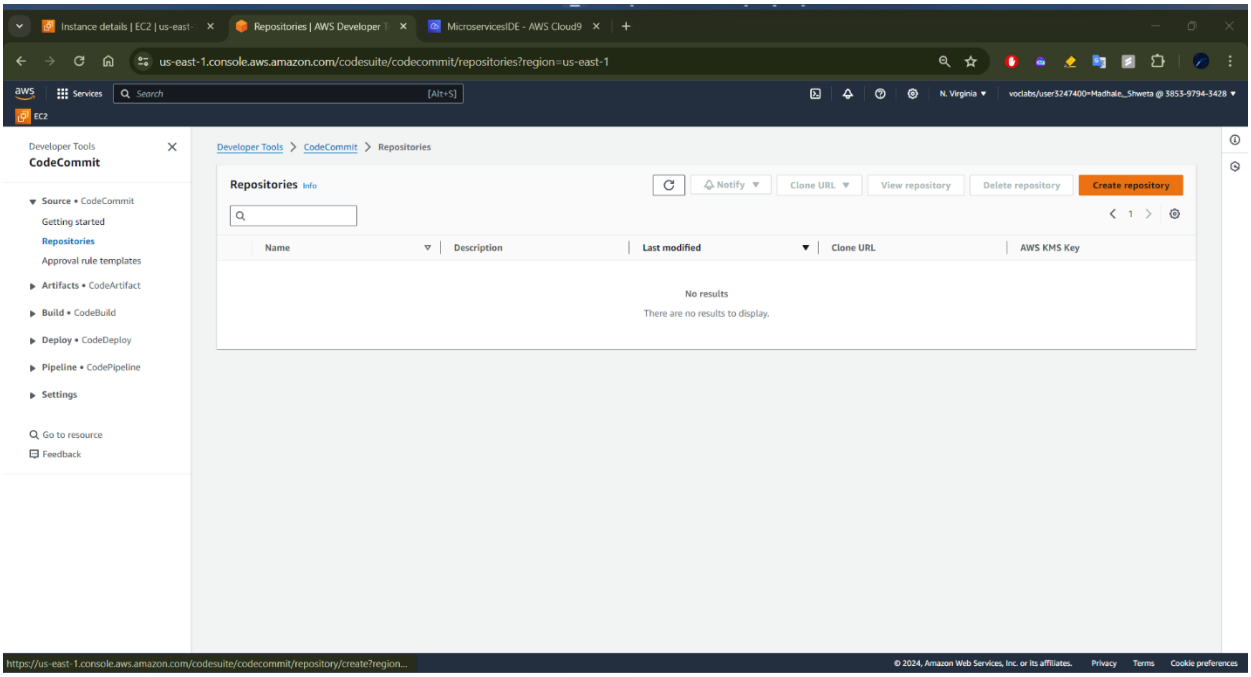
Task 3.3: Create working directories with starter code for the two microservices

- In the microservices directory, create new directories for customer and employee. Copy the source code from temp into these directories.

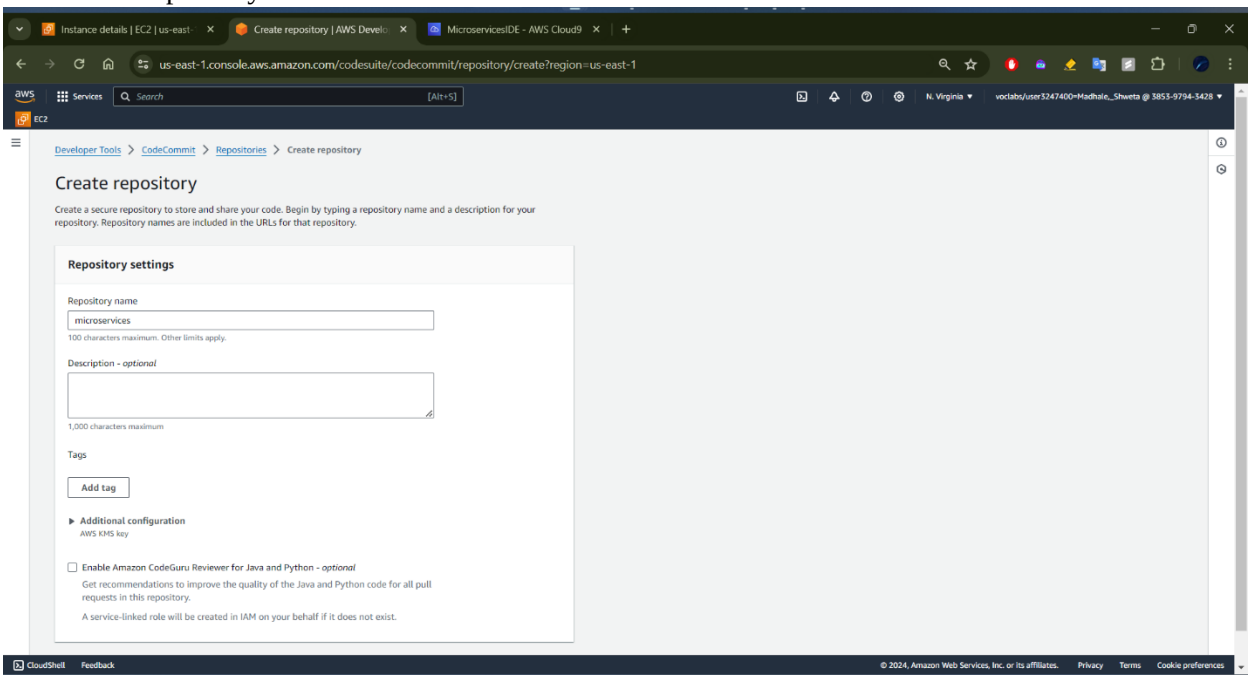


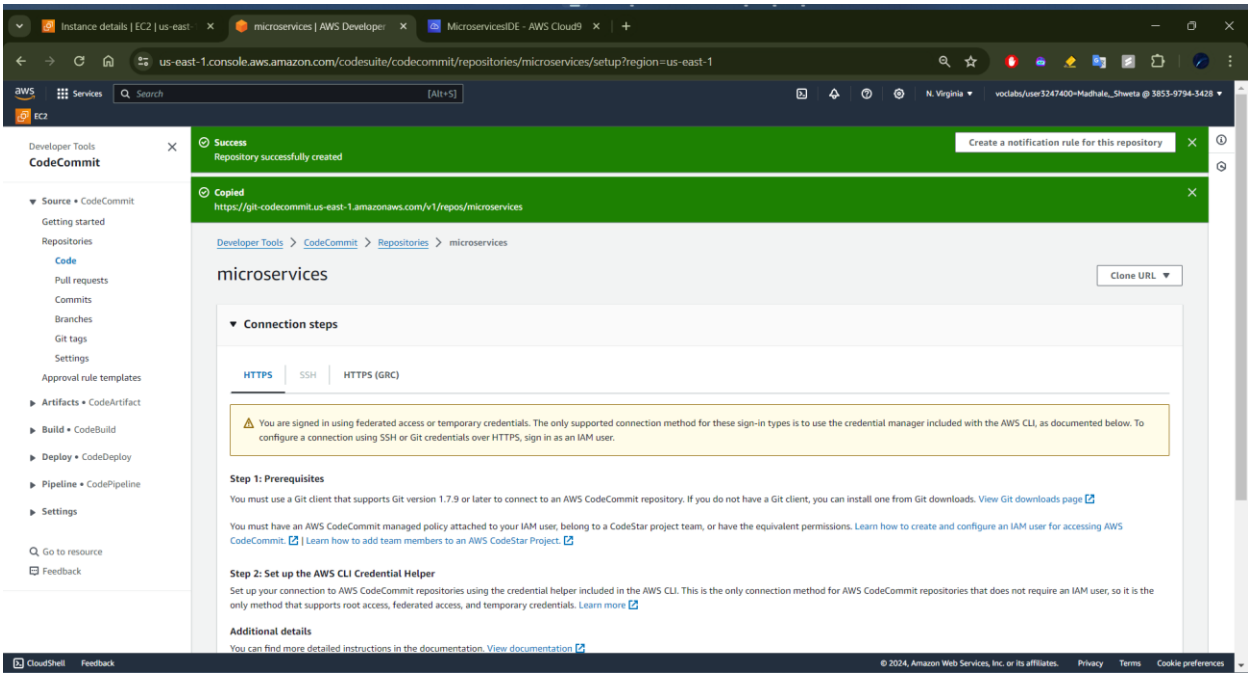
Task 3.4: Create a Git repository for the microservices code and push the code to CodeCommit

- Navigate to CodeCommit



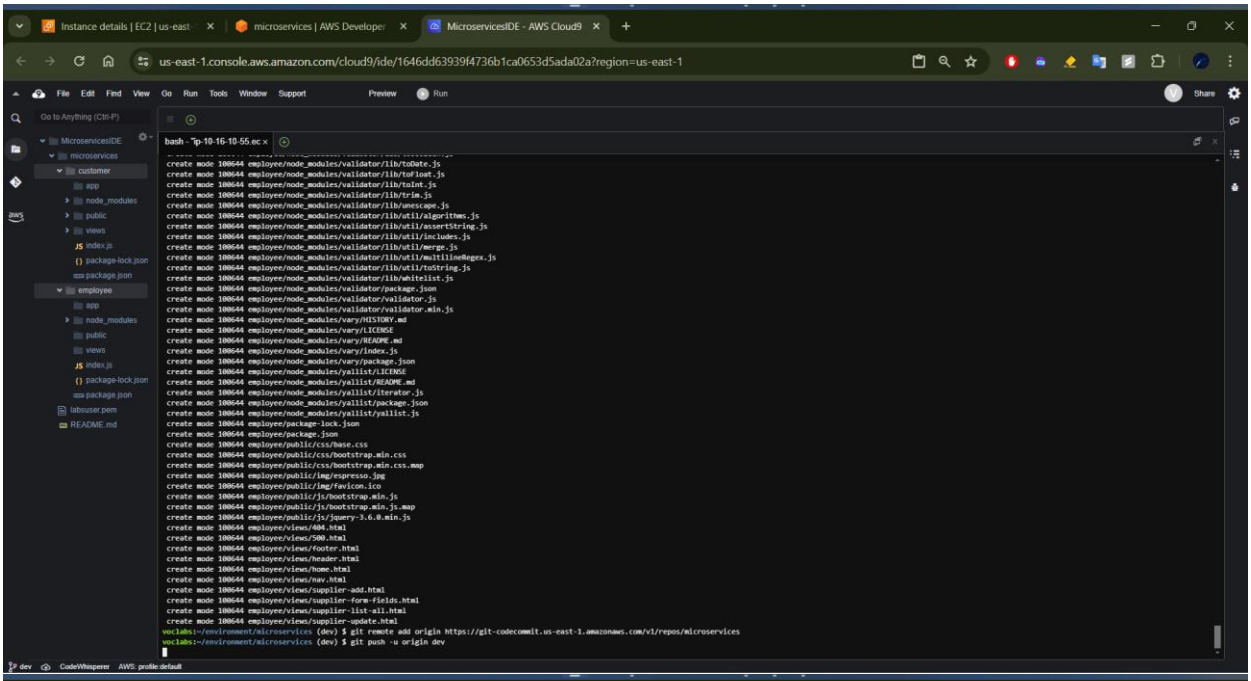
- Create new repository named microservices.

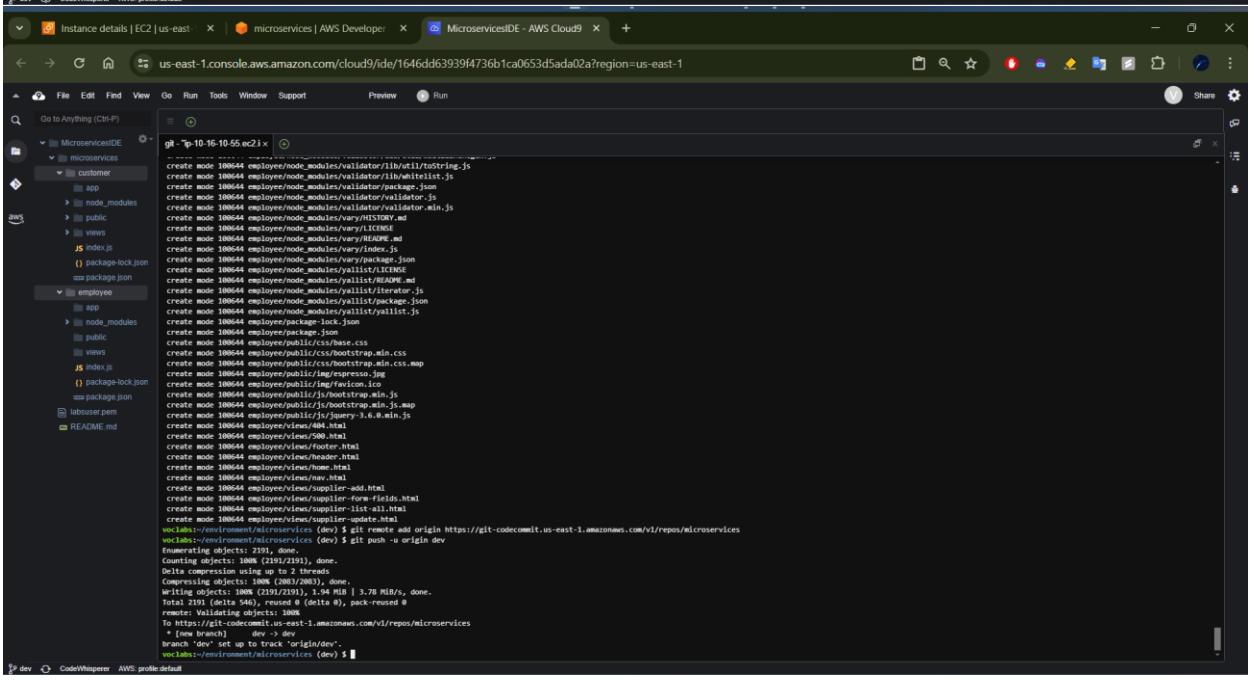




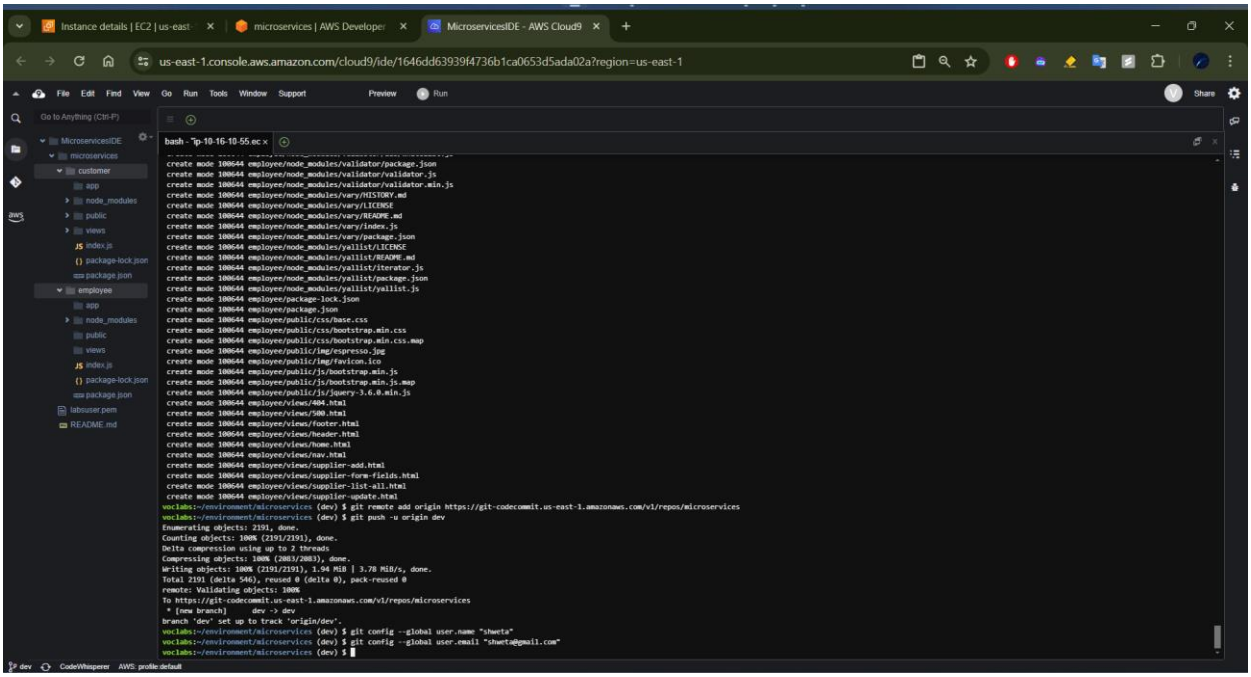
- Execute

```
cd ~/environment/microservices
git init
git branch -m dev
git add .
git commit -m 'two unmodified copies of the application code'
git remote add origin https://git-codecommit.us-east-
1.amazonaws.com/v1/repos/microservices
git push -u origin dev
```

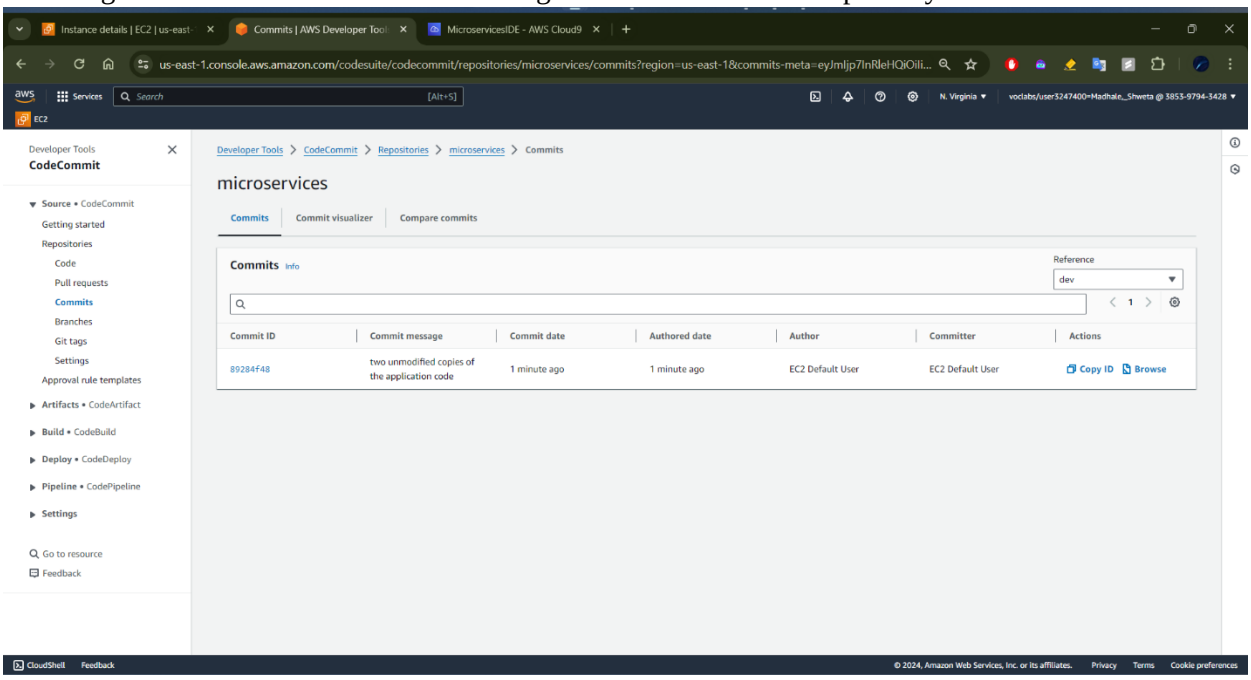




- Configure git client with username and email address.



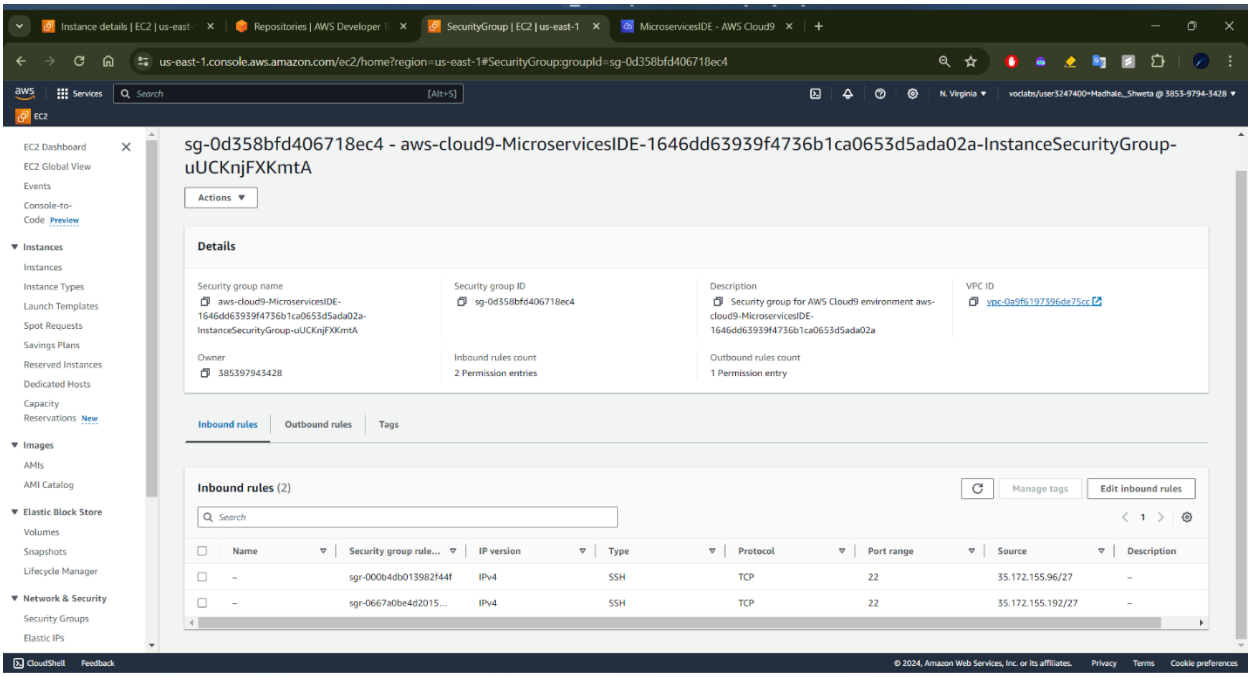
- Checking CodeCommit console and observing code checked into the repository.

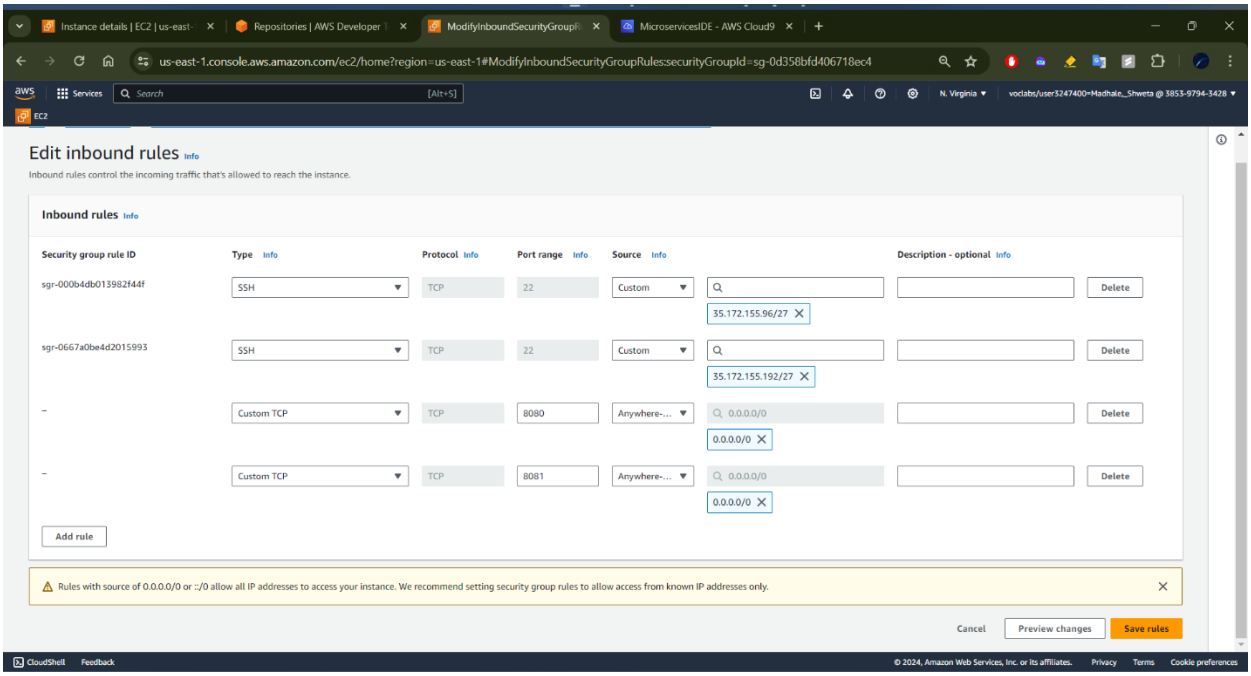


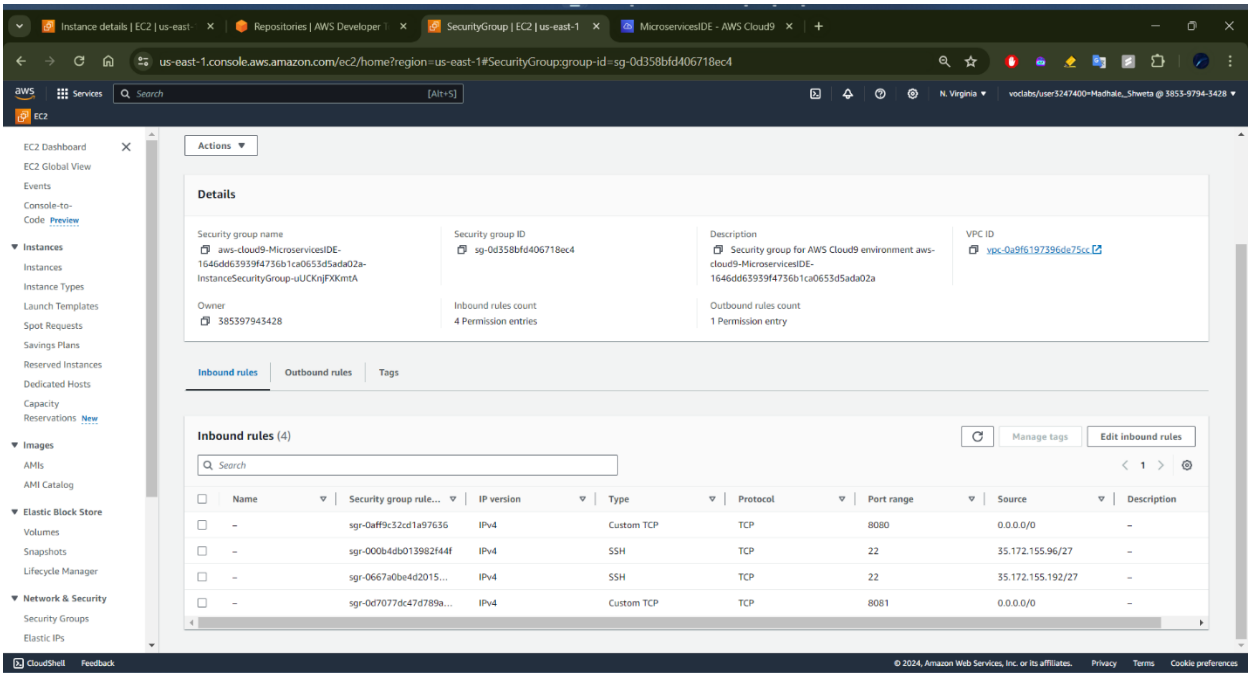
Phase 4 - Break the monolithic design into microservices, and launch test Docker containers.

Task 4.1: Adjust the AWS Cloud9 instance security group settings

- Edit the security group of AWS Cloud9 EC2 instance to allow inbound network traffic on TCP ports 8080 and 8081.

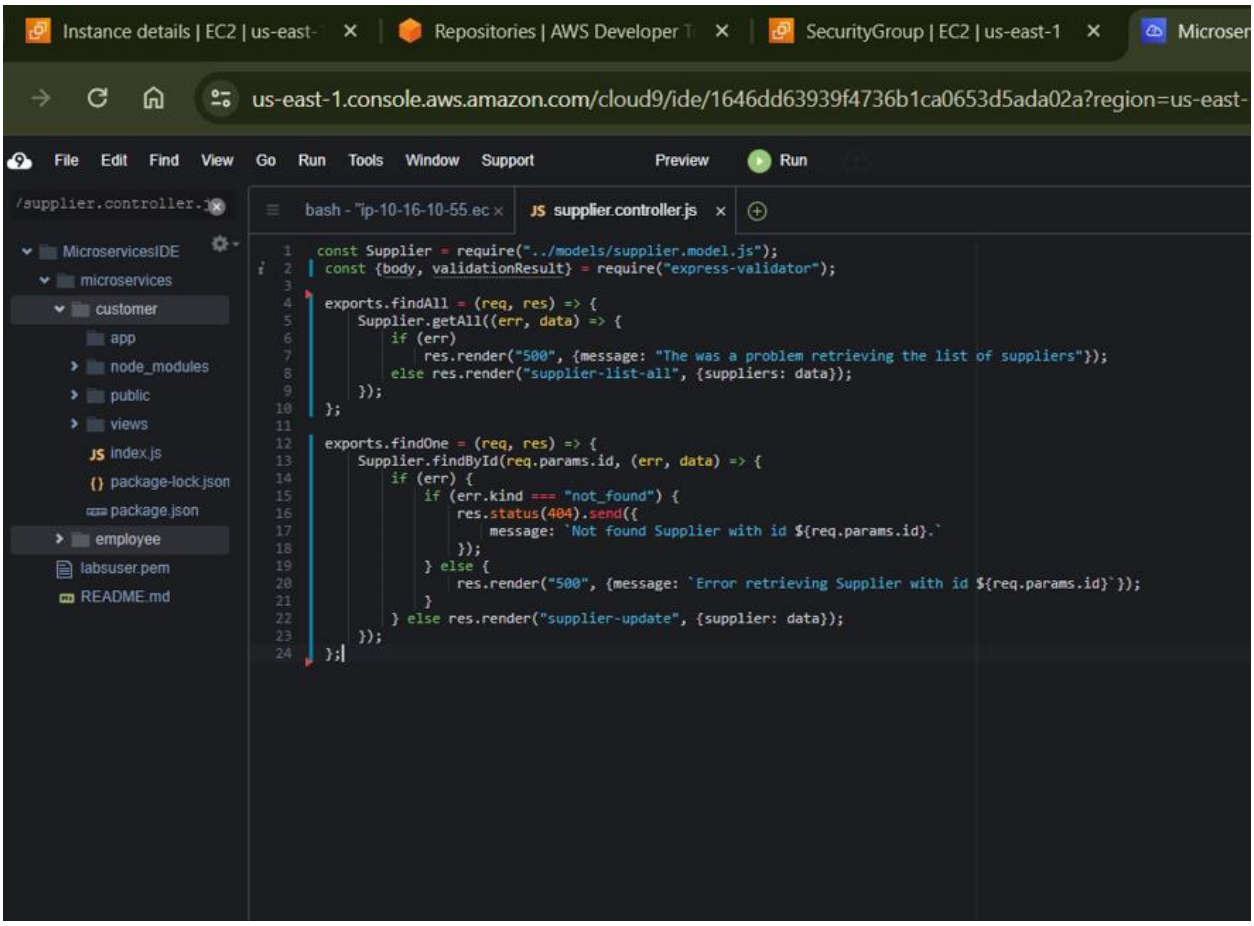

Instance details | EC2 | us-east-
Repositories | AWS Developer
SecurityGroup | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroup:groupId=sg-0d358bfd406718ec4
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
sg-0d358bfd406718ec4 - aws-cloud9-MicroservicesIDE-1646dd63939f4736b1ca0653d5ada02a-InstanceSecurityGroup-uUCKnjFXKmtA
Actions
Details
Security group name
aws-cloud9-MicroservicesIDE-1646dd63939f4736b1ca0653d5ada02a-InstanceSecurityGroup-uUCKnjFXKmtA
Security group ID
sg-0d358bfd406718ec4
Description
Security group for AWS Cloud9 environment aws-cloud9-MicroservicesIDE-1646dd63939f4736b1ca0653d5ada02a
VPC ID
vpc-0a9f6197396de75cc
Owner
385397943428
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
Source
Description
sgr-000b4db01398 2f44f
IPv4
SSH
TCP
22
35.172.155.96/27
sgr-0667a0be4d2015...
IPv4
SSH
TCP
22
35.172.155.192/27
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
Repositories | AWS Developer
ModifyInboundSecurityGroupR
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-0d358bfd406718ec4
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-000b4db01398 2f44f
SSH
TCP
22
Custom
35.172.155.96/27
Delete
sgr-0667a0be4d2015993
SSH
TCP
22
Custom
35.172.155.192/27
Delete
–
Custom TCP
TCP
8080
Anywhere-...
0.0.0.0/0
Delete
–
Custom TCP
TCP
8081
Anywhere-...
0.0.0.0/0
Delete
Add rule
Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.
Cancel
Preview changes
Save rules
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

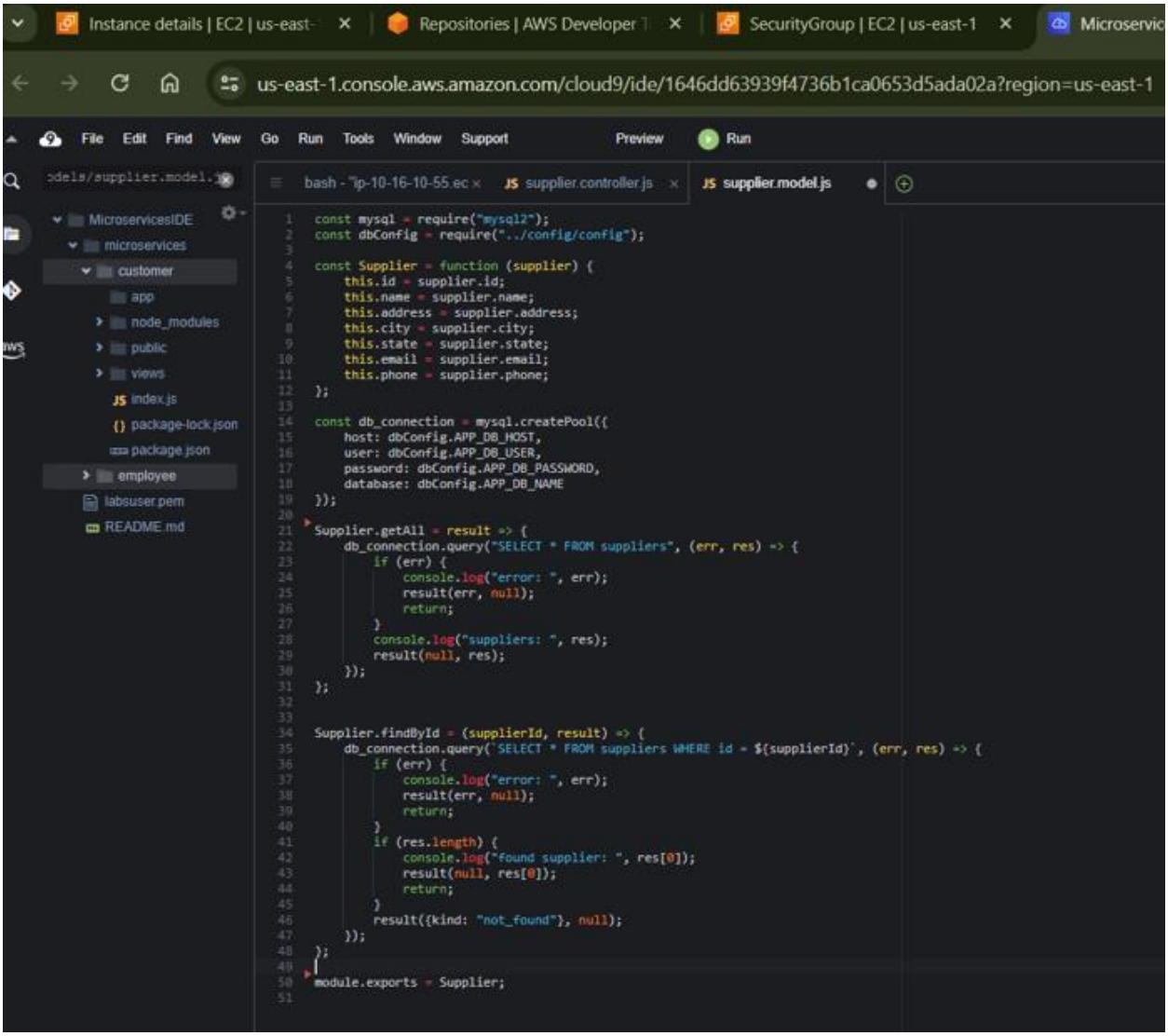


Task 4.2: Modify the source code of the customer microservice

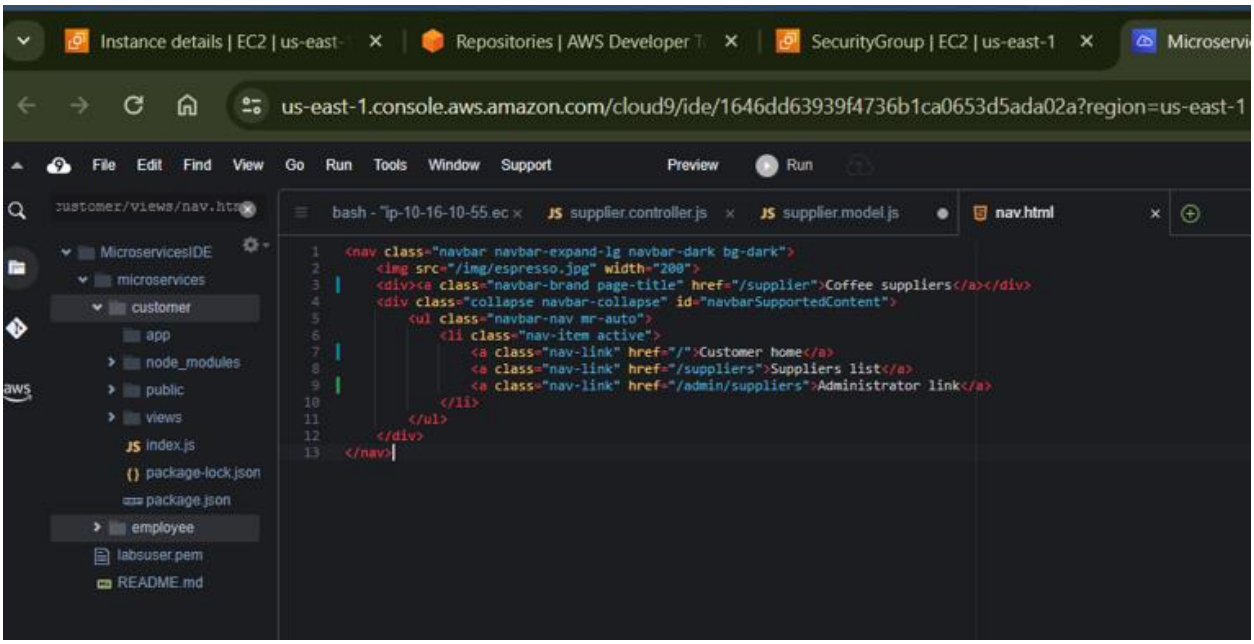
- Expand the customer directory. Edit the **customer/app/controller/supplier.controller.js** file so that the remaining functions provide only the read-only actions that you want customers to be able to perform



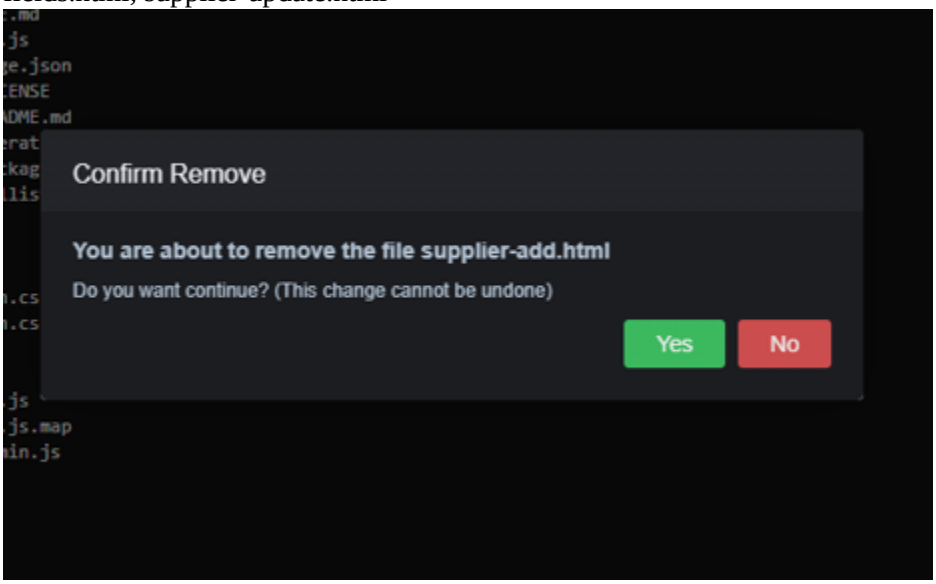
- Edit the **customer/app/models/supplier.model.js** file. Delete the unnecessary functions in it so that what remains are only read-only functions.

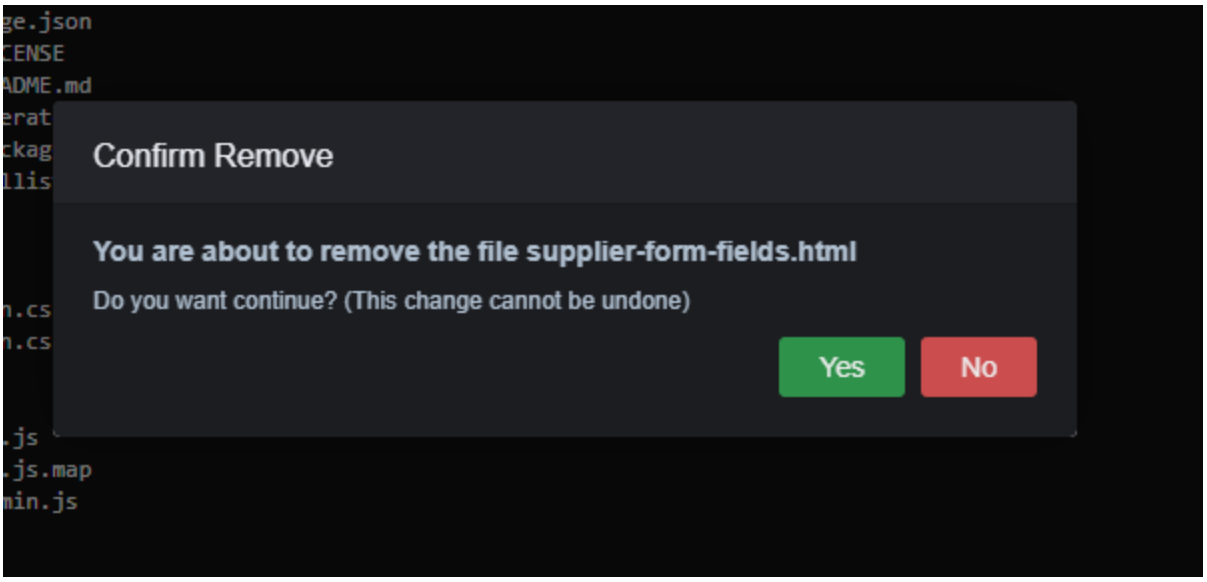


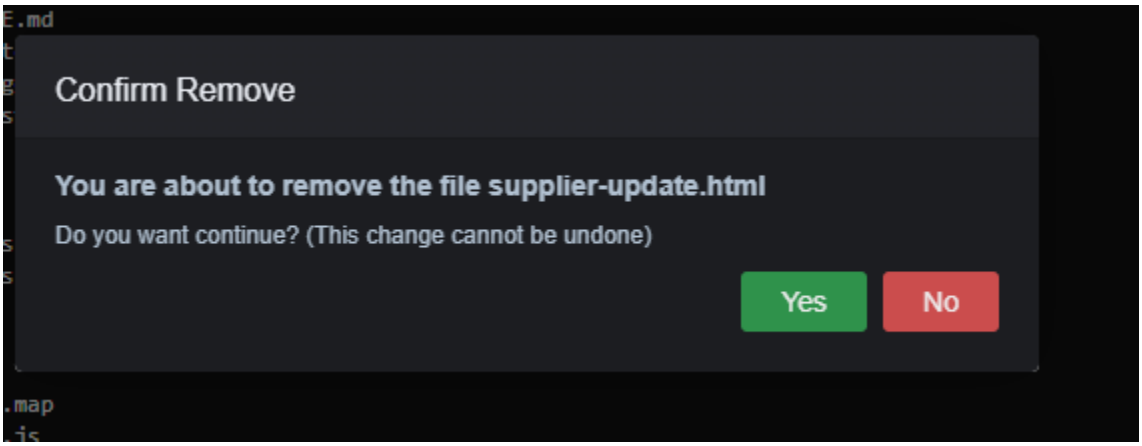
- Later in the project, when deploying the microservices behind an Application Load Balancer, the employees need to be able to navigate from the main customer page to the area of the web application where they can add, edit, or delete supplier entries. To support this, editing the **customer/views/nav.html** file:
  On line 3, change Monolithic Coffee suppliers to Coffee suppliers. On line 7, change Home to Customer home. Add a new line after line 8 that contains the following HTML: Administrator link



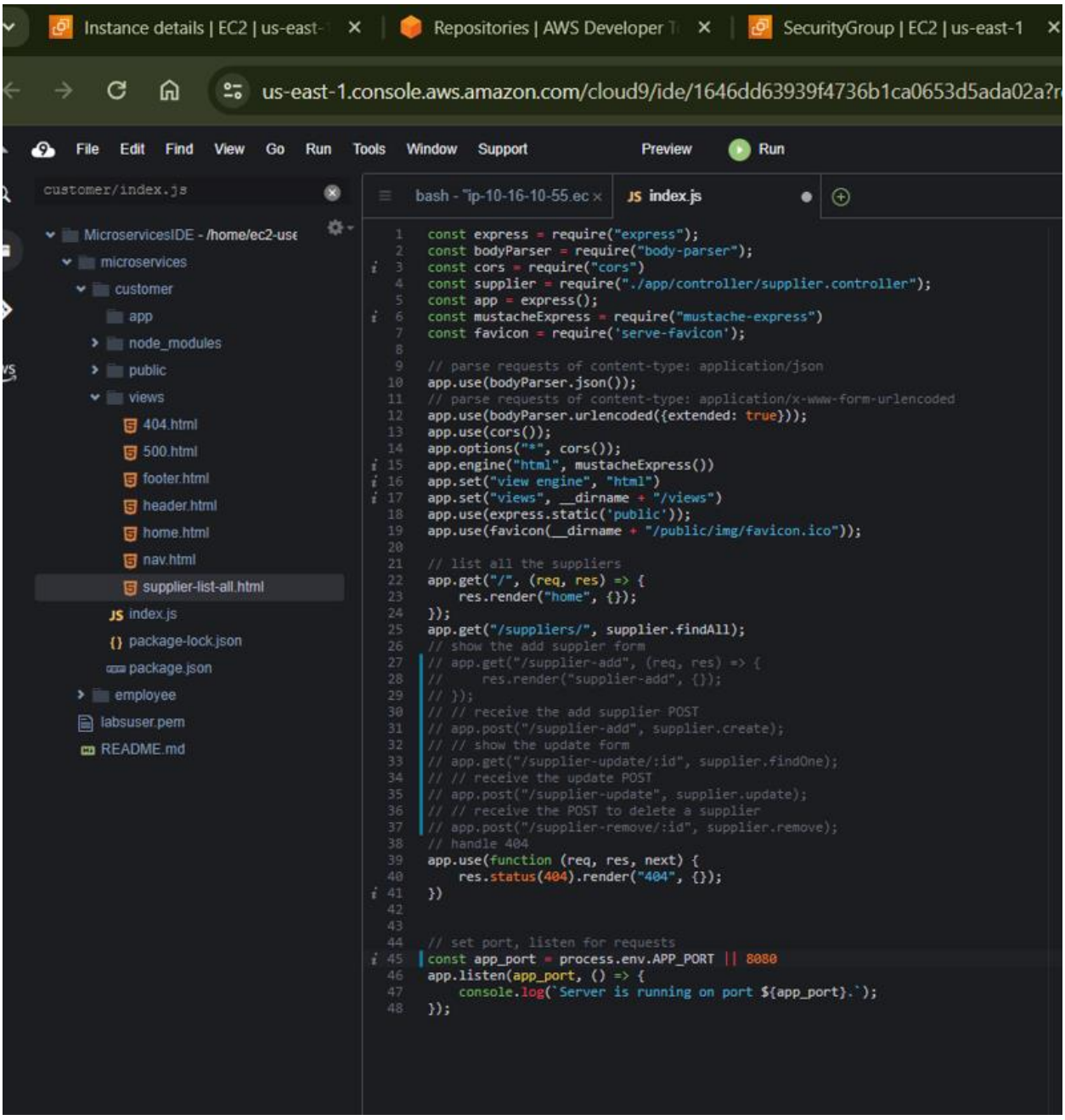
- Because the customer microservice doesn't need to support read-write actions, DELETE the following .html files from the customer/views directory: supplier-add.html, supplier-form-fields.html, supplier-update.html







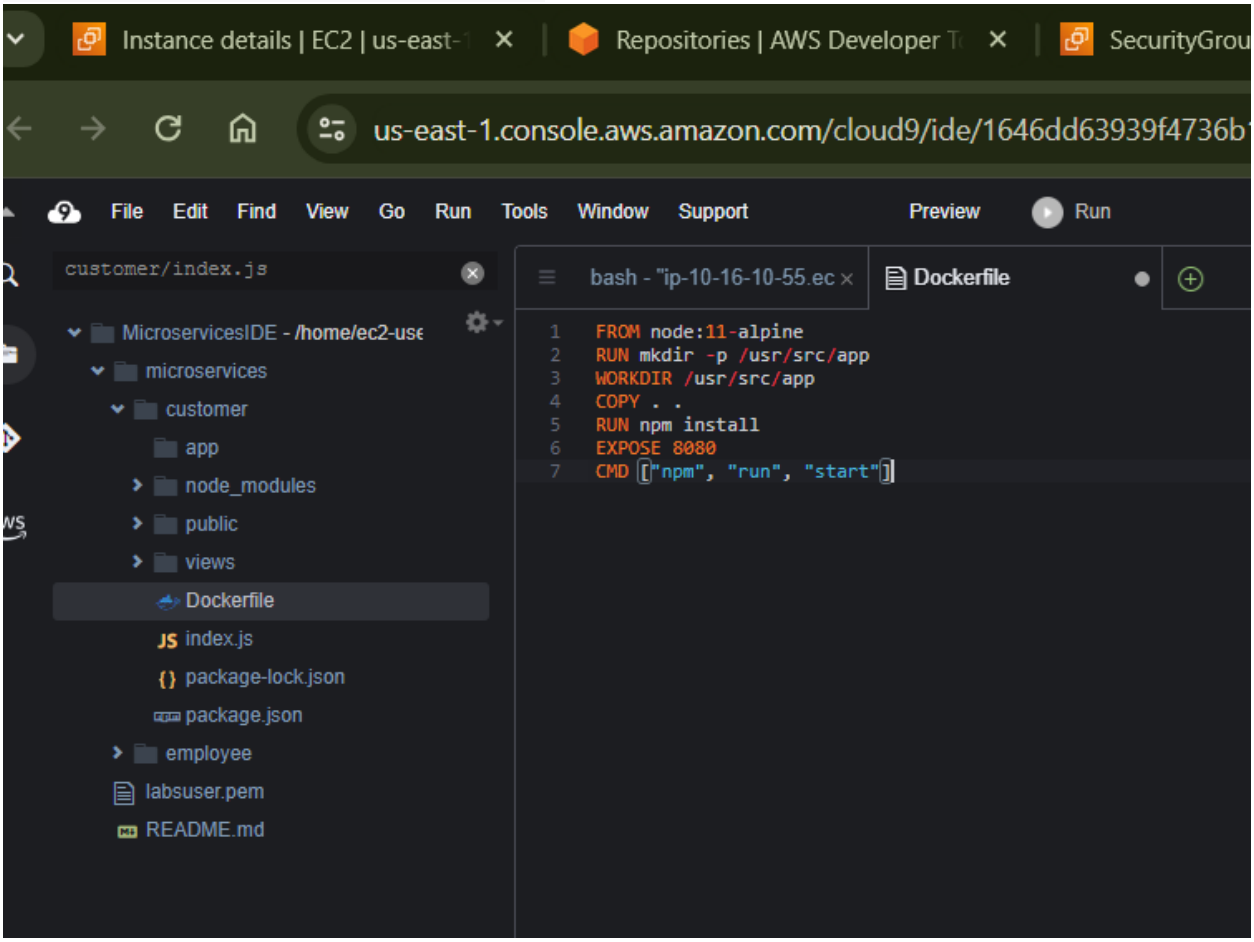
- Edit the customer/index.js file as needed to account for the fact that the node application will now run on Docker containers: Comment out lines 27 to 37 (ensure that each line starts with //). On line 45, change the port number to 8080



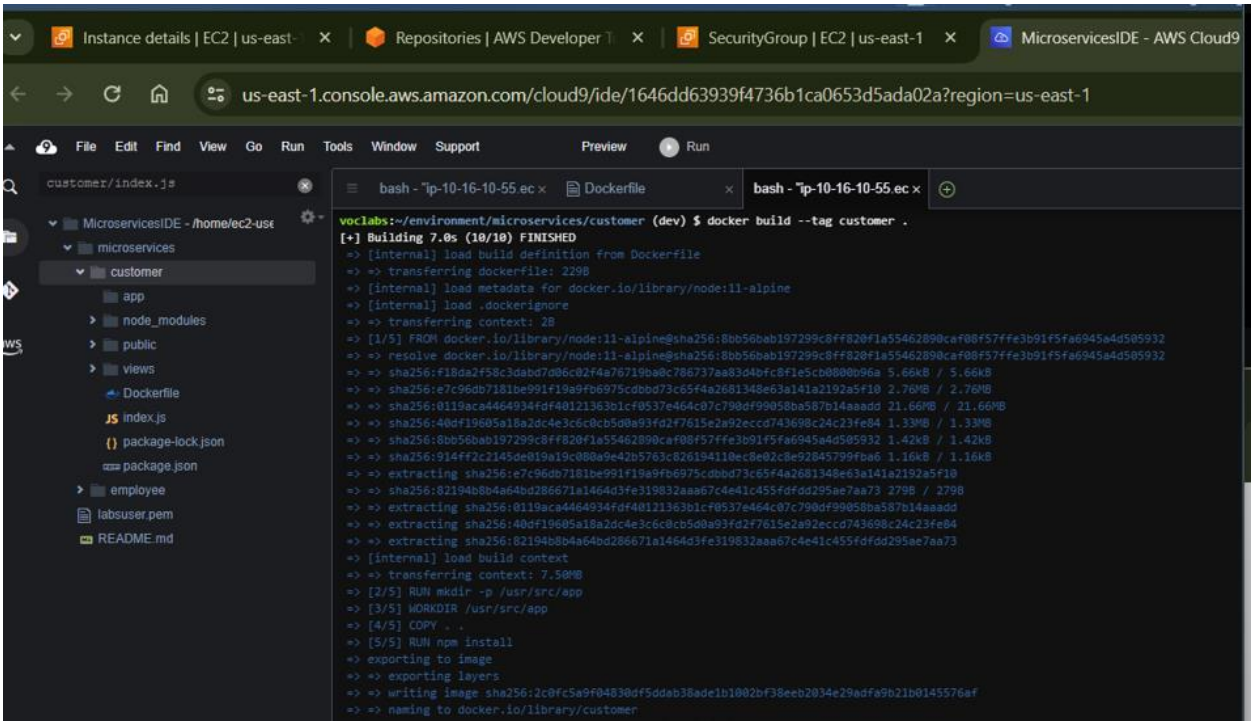
Task 4.3: Create the customer microservice Dockerfile and launch a test container

- In the customer directory, create a new file named Dockerfile that contains the following code

```
FROM node:11-alpine
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . .
RUN npm install
EXPOSE 8080
CMD ["npm", "run", "start"]
```



- Build an image from customer Dockerfile. In the AWS Cloud9 terminal, change to the customer directory. Run the following command: **docker build --tag customer .**



- Verify that the customer-labeled Docker image was created

```
voclabs:~/environment/microservices/customer (dev) $ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
customer     latest    2c0fc5a9f048   2 minutes ago    82.7MB
voclabs:~/environment/microservices/customer (dev) $
```

- Launch a Docker container that runs the customer microservice on port 8080. Set dbEndpoint variable in the terminal session.

```
dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
echo $dbEndpoint
```

```
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.ccaycjdtuups.us-east-1.rds.amazonaws.com
```

- Launch container from image with **docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint" customer**

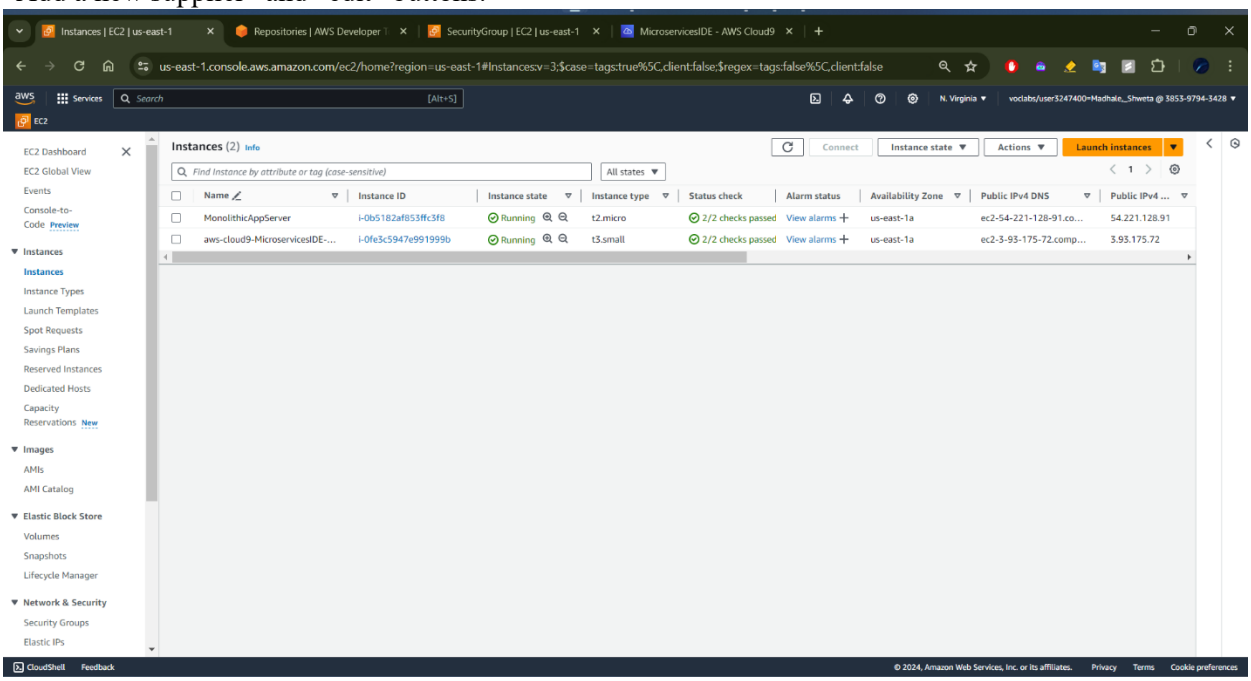
```
voclabs:~/environment/microservices/customer (dev) $ docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint" customer
93745d4a452f193688f5502e5251a8c0bf5ab9736ae26cf927241774c990b734
voclabs:~/environment/microservices/customer (dev) $
```

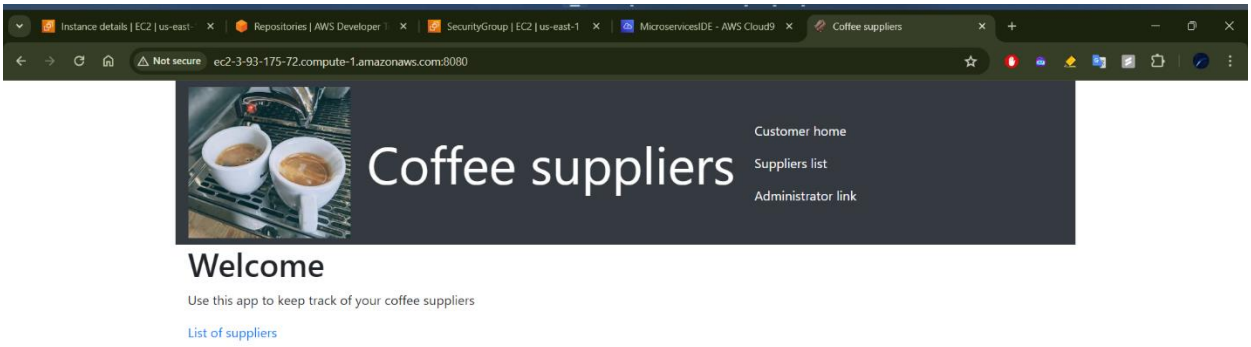
- Checking Docker containers currently running on the AWS Cloud9 Instance.

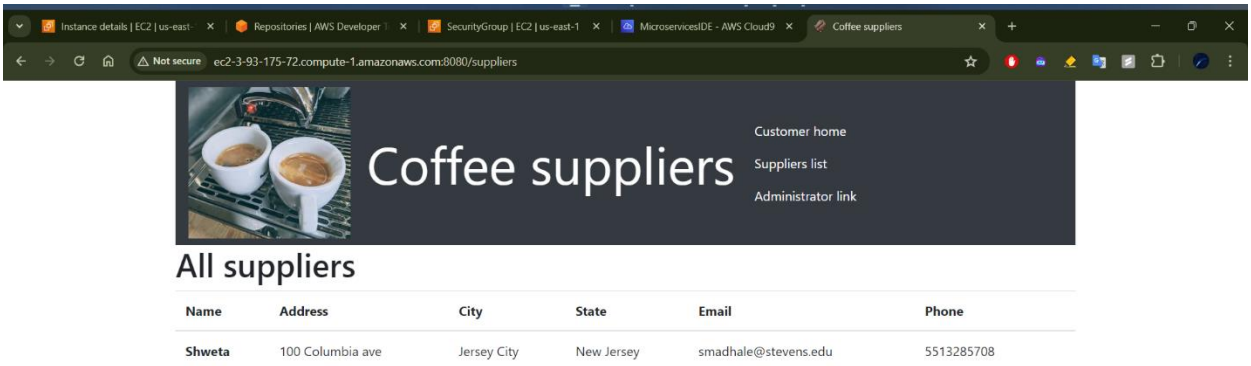
```
voclabs:~/environment/microservices/customer (dev) $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS                                       NAMES
93745d4a452f   customer   "docker-entrypoint.s…"   18 seconds ago   Up 17 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   customer_1
voclabs:~/environment/microservices/customer (dev) $
```

- Verifying that customer microservice is running the container and is working as intended. Loading new browser tab with http://<cloud-9-public-IPv4-address>:8080. Selecting the “List of

Suppliers” should show the entries added earlier. Confirming that suppliers page does not have “Add a new supplier” and “edit” buttons.

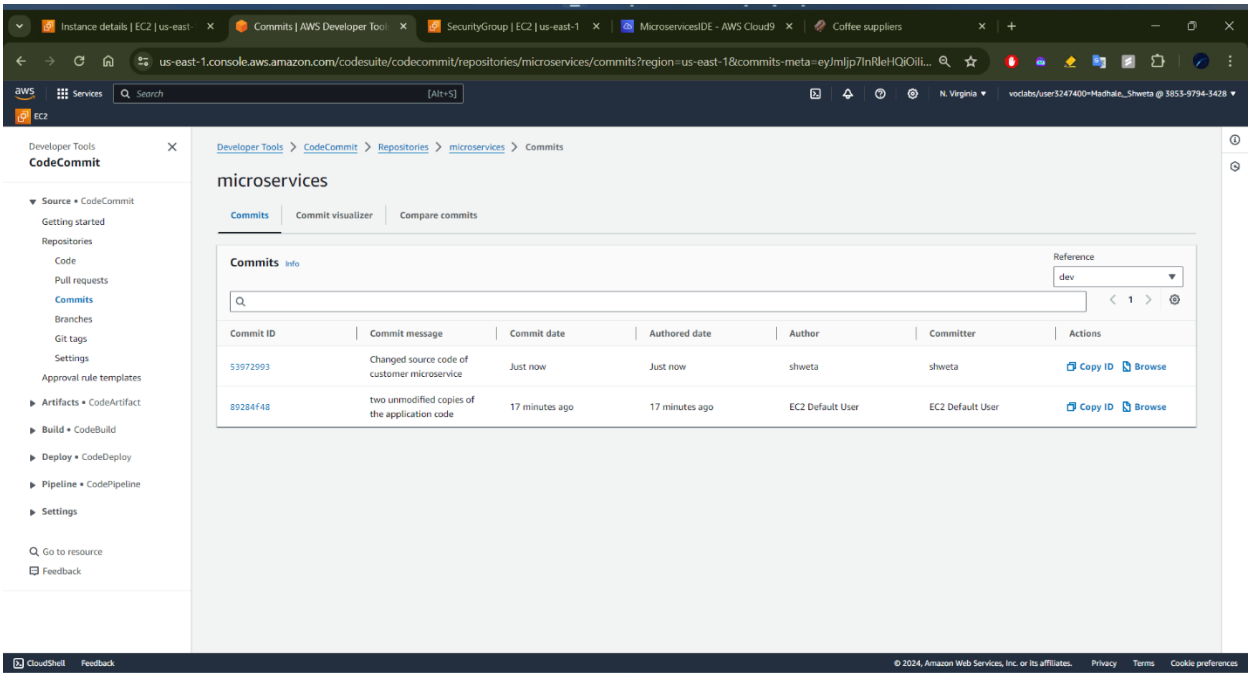






- Commit and push the code changes to CodeCommit.

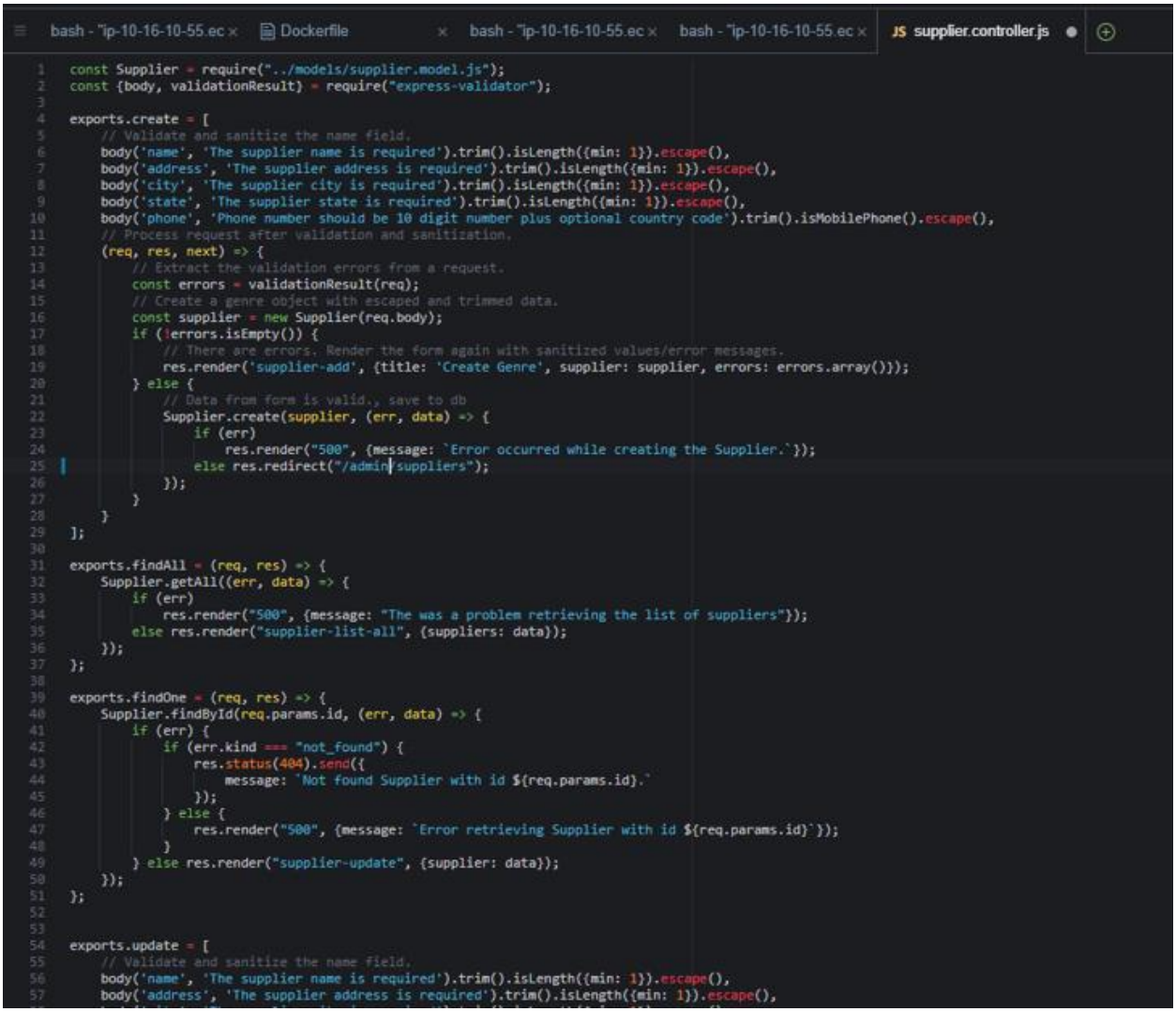
```
voclabs:~/environment/microservices/customer (dev) $ git add .
voclabs:~/environment/microservices/customer (dev) $ git commit -m "Changed source code of customer microservice"
[dev 5397299] Changed source code of customer microservice
 9 files changed, 45 insertions(+), 289 deletions(-)
 create mode 100644 customer/Dockerfile
 delete mode 100644 customer/views/supplier-add.html
 delete mode 100644 customer/views/supplier-form-fields.html
 delete mode 100644 customer/views/supplier-update.html
voclabs:~/environment/microservices/customer (dev) $ git push origin dev
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 2 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (13/13), 1.41 KiB | 724.00 KiB/s, done.
Total 13 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
   89284f4..5397299  dev -> dev
voclabs:~/environment/microservices/customer (dev) $
```

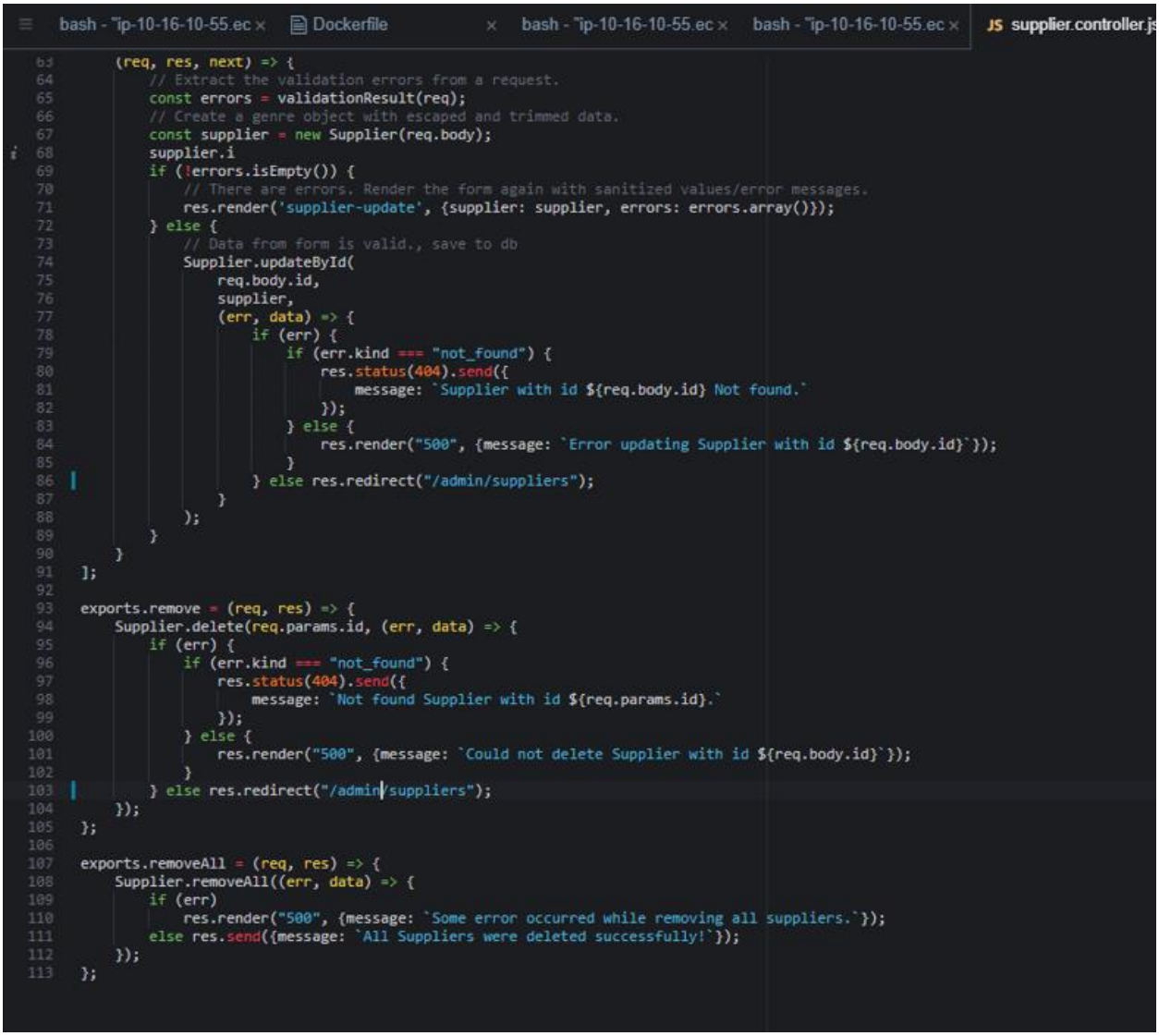


Task 4.4: Modify the source code of the employee microservice

- Expand the employee directory. In the employee/app/controller/supplier.controller.js file, for all the redirect calls, prepend /admin to the path. Find locations where to make changes with

```
cd ~/environment/microservices/employee
grep -n 'redirect' app/controller/supplier.controller.js
```


bash - "ip-10-16-10-55.ec
Dockerfile
bash - "ip-10-16-10-55.ec
bash - "ip-10-16-10-55.ec
supplier.controller.js
const Supplier = require("../models/supplier.model.js");
const {body, validationResult} = require("express-validator");
exports.create = [
// Validate and sanitize the name field.
body('name', 'The supplier name is required').trim().isLength({min: 1}).escape(),
body('address', 'The supplier address is required').trim().isLength({min: 1}).escape(),
body('city', 'The supplier city is required').trim().isLength({min: 1}).escape(),
body('state', 'The supplier state is required').trim().isLength({min: 1}).escape(),
body('phone', 'Phone number should be 10 digit number plus optional country code').trim().isMobilePhone().escape(),
// Process request after validation and sanitization.
(req, res, next) => {
// Extract the validation errors from a request.
const errors = validationResult(req);
// Create a genre object with escaped and trimmed data.
const supplier = new Supplier(req.body);
if (!errors.isEmpty()) {
// There are errors. Render the form again with sanitized values/error messages.
res.render('supplier-add', {title: 'Create Genre', supplier: supplier, errors: errors.array()});
} else {
// Data from form is valid., save to db
Supplier.create(supplier, (err, data) => {
if (err)
res.render("500", {message: `Error occurred while creating the Supplier.`});
else res.redirect("/admin/suppliers");
});
}
}
];
exports.findAll = (req, res) => {
Supplier.getAll((err, data) => {
if (err)
res.render("500", {message: "The was a problem retrieving the list of suppliers"});
else res.render("supplier-list-all", {suppliers: data});
});
};
exports.findOne = (req, res) => {
Supplier.findById(req.params.id, (err, data) => {
if (err) {
if (err.kind === "not_found") {
res.status(404).send({
message: `Not found Supplier with id ${req.params.id}.`
});
} else {
res.render("500", {message: `Error retrieving Supplier with id ${req.params.id}`});
}
} else res.render("supplier-update", {supplier: data});
});
};
exports.update = [
// Validate and sanitize the name field.
body('name', 'The supplier name is required').trim().isLength({min: 1}).escape(),
body('address', 'The supplier address is required').trim().isLength({min: 1}).escape(),



- In the employee/index.js file, update the app.get calls, app.post calls, and a port number. To find the seven lines that need to be updated, run the following command in the terminal:

```
grep -n 'app.get\|app.post' index.js
```

Set the port number to 8081,


bash - "ip-10-16-10-55.ec
Dockerfile
bash - "ip-10-16-10-
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors")
const supplier = require("./app/controller/supplier.controller");
const app = express();
const mustacheExpress = require("mustache-express")
const favicon = require('serve-favicon');
// parse requests of content-type: application/json
app.use(bodyParser.json());
// parse requests of content-type: application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({extended: true}));
app.use(cors());
app.options("*", cors());
app.engine("html", mustacheExpress())
app.set("view engine", "html")
app.set("views", __dirname + "/views")
app.use(express.static('public'));
app.use(favicon(__dirname + "/public/img/favicon.ico"));
// list all the suppliers
app.get("/admin", (req, res) => {
res.render("home", {});
});
app.get("/admin/suppliers/", supplier.findAll);
// show the add suppler form
app.get("/admin/supplier-add", (req, res) => {
res.render("supplier-add", {});
});
// receive the add supplier POST
app.post("/admin/supplier-add", supplier.create);
// show the update form
app.get("/admin/supplier-update/:id", supplier.findOne);
// receive the update POST
app.post("/admin/supplier-update", supplier.update);
// receive the POST to delete a supplier
app.post("/admin/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
res.status(404).render("404", {});
})
// set port, listen for requests
const app_port = process.env.APP_PORT || 80
app.listen(app_port, () => {
console.log(`Server is running on port ${app_port}.`);
});

- In the employee/views/supplier-add.html and employee/views/supplier-update.html files, for the form action paths, prepend /admin to the path. To find the three lines that need to be updated in the two files, run the following command in the terminal:

```
grep -n 'action' views/*
```

```
{{>header}}
<div class="container">
    {{>nav}}
    <div class="mt-2">
        {{#errors}}
        <div class="alert alert-warning" role="alert">
            {{msg}}
        </div>
        {{/errors}}
    </div>
    <form action="/admin/supplier-add" method="POST">
        {{>supplier-form-fields}}
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>

</div>
{{>footer}}
```

```
{{>header}}
<div class="container">
    {{>nav}}
    <div class="mt-2">
        {{#errors}}
        <div class="alert alert-warning" role="alert">
            {{msg}}
        </div>
        {{/errors}}
    </div>
    {{#supplier}}
    <form action="/admin/supplier-update" method="POST">
        <input type="hidden" id="id" name="id" value="{{id}}">
        {{>supplier-form-fields}}
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>

    <!-- Button trigger modal -->
    <button type="button" class="float-right btn btn-danger" data-toggle="modal" data-target="#exampleModal">
        Delete this supplier
    </button>

    <!-- Modal -->
    <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">Delete supplier</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    Are you sure you want to delete supplier {{name}}?
                </div>
                <div class="modal-footer">
                    <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>
                </div>
            </div>
        </div>
    </div>


    {{/supplier}}

</div>
{{>footer}}
```

- In the employee/views/supplier-list-all.html and employee/views/home.html files, for the HTML paths, prepend /admin to the path. To find the three lines that need to be updated in the two files, run the following command in the terminal:

```
grep -n 'href' views/supplier-list-all.html views/home.html
```

```
{{>header}}
<div class="container">
    {{>nav}}
    <h1>All suppliers</h1>
    <table class="table table-hover">
        <thead>
        <tr>
            <th scope="col">Name</th>
            <th scope="col">Address</th>
            <th scope="col">City</th>
            <th scope="col">State</th>
            <th scope="col">Email</th>
            <th scope="col">Phone</th>
            <th scope="col"></th>
        </tr>
        </thead>
        <tbody>
        {{#suppliers}}
        <tr>
            <th scope="row">{{name}}</th>
            <td>{{address}}</td>
            <td>{{city}}</td>
            <td>{{state}}</td>
            <td>{{email}}</td>
            <td>{{phone}}</td>
            <td><h4><span class="badge badge-info"><a class="text-light"
                                                      href="/admin/supplier-update/{{id}}">edit</a></span></h4></td>
        </tr>
        {{/suppliers}}
        </tbody>
    </table>
    <h4><a class="badge badge-success" href="/admin/supplier-add">Add a new supplier</a></h4>

</div>
{{>footer}}

```

```
{{>header}}
<div class="container">
    {{>nav}}
    <div class="container">
        <h1>Welcome</h1>
        <p>Use this app to keep track of your coffee suppliers</p>
        <p><a href="/admin/suppliers">List of suppliers</a></p>
    </div>
</div>
{{>footer}}

```

- In the employee/views/**header.html** file, modify the title to be `Manage coffee 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>Manage coffee suppliers</title>
</head>
<body>


```

- Edit the employee/views/nav.html file. On line 3, change Monolithic Coffee suppliers to Manage coffee suppliers. On line 7, replace the existing line of code with the following:

```
<a class="nav-link" href="/admin/suppliers">Administrator home</a>
```

Add a new line after line 8 that contains the following HTML:

```
<a class="nav-link" href="/">Customer home</a>
```

```
1  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
2      <img src="/img/espresso.jpg" width="200">
3      <div><a class="navbar-brand page-title" href="/supplier">Manage coffee suppliers</a></div>
4      <div class="collapse navbar-collapse" id="navbarSupportedContent">
5          <ul class="navbar-nav mr-auto">
6              <li class="nav-item active">
7                  <a class="nav-link" href="/admin/suppliers">Administrator home</a>
8                  <a class="nav-link" href="/suppliers">Suppliers list</a>
9                  <a class="nav-link" href="/">Customer home</a>
10             </li>
11         </ul>
12     </div>
13 </nav>
```

Task 4.5: Create the employee microservice Dockerfile and launch a test container

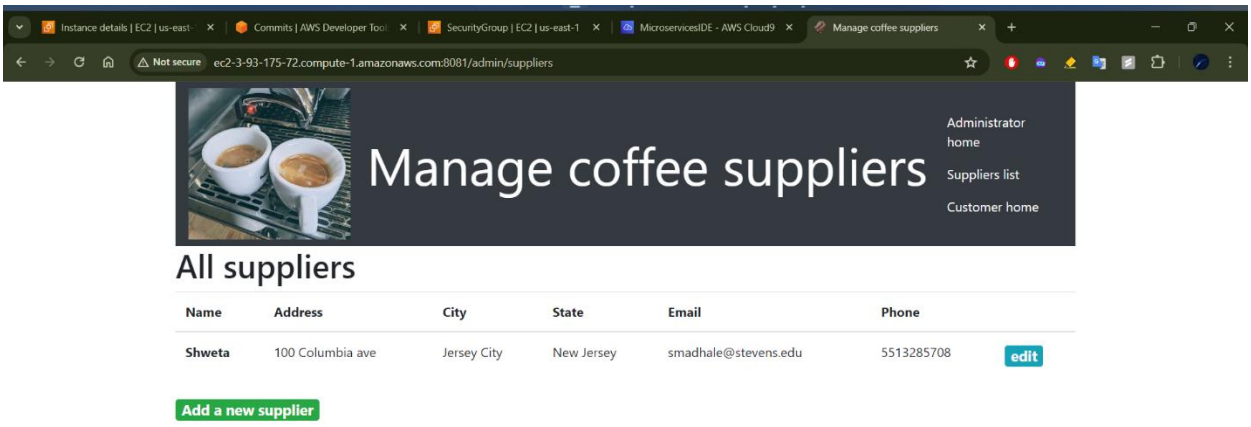
- Create Dockerfile for employee microservice by duplicating Dockerfile from customer microservice. Editing employee dockerfile to change the port number to 8081

```
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"]
```

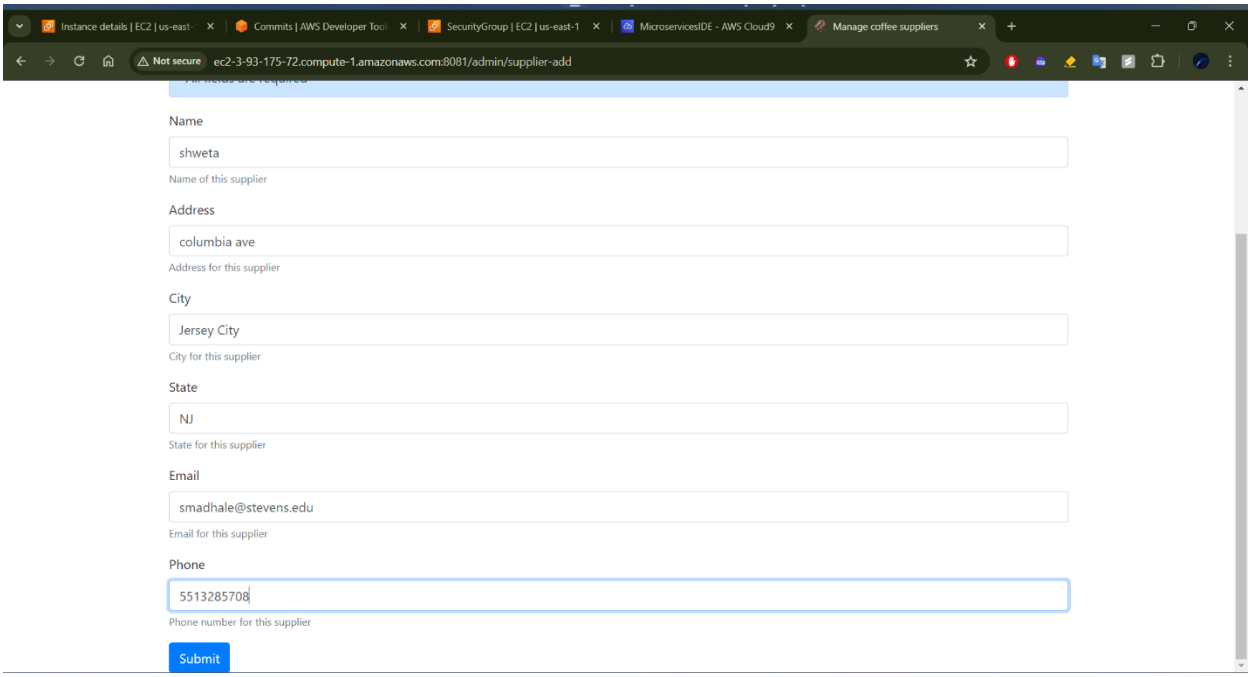
- Build the Docker image for the employee microservice. Specify employee as the tag. Run a container named employee_1 based on the employee image. Run it on port 8081 and be sure to pass in the database endpoint. Verify that the employee microservice is running in the container and that the microservice functions as intended. Load the microservice web page in a new browser tab at http://<cloud9-public-ip-address>:8081/admin/suppliers. Verify that this view shows buttons to edit existing suppliers and to add a new supplier.

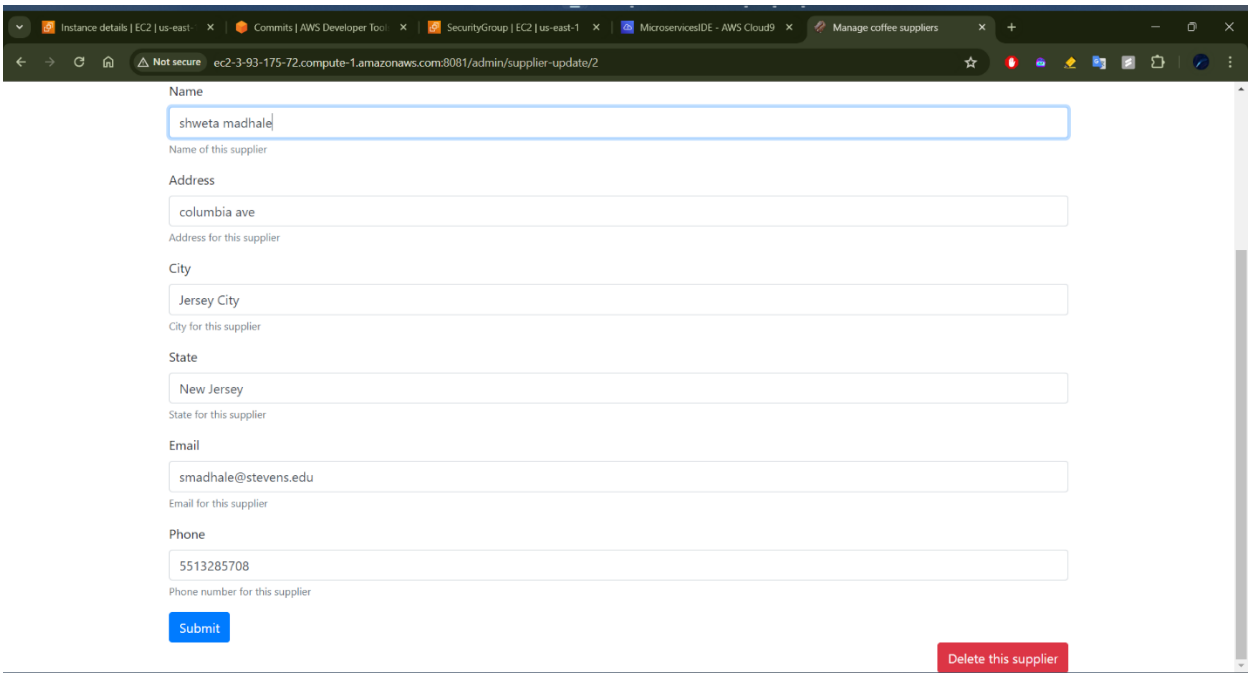
```
voclabs:~/environment/microservices/employee (dev) $ grep -n 'redirect' app/controller/supplier.controller.js
25:                 else res.redirect("/suppliers");
86:                     } else res.redirect("/suppliers");
103:        } else res.redirect("/suppliers");
voclabs:~/environment/microservices/employee (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) $ 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) $ grep -n 'href' views/supplier-list-all.html views/home.html
views/supplier-list-all.html:27:                                                    href="/supplier-update/{{id}}">edit</a></span></h4></td>
views/supplier-list-all.html:32:    <h4><a class="badge badge-success" href="/supplier-add">Add a new supplier</a></h4>
views/home.html:7:        <p><a href="/suppliers">List of suppliers</a></p>
```

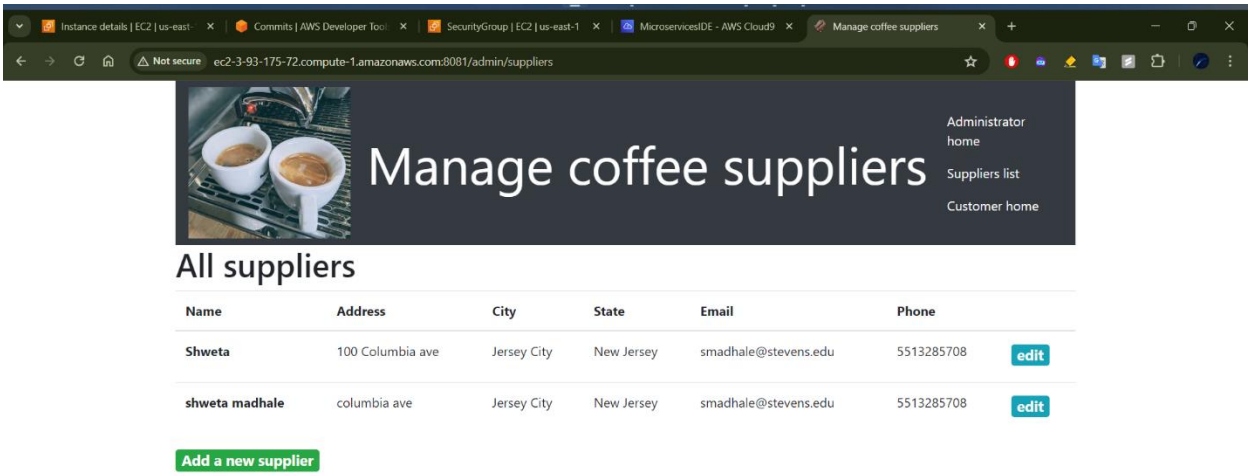
```
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
[+] Building 0.3s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 229B
 => [internal] load metadata for docker.io/library/node:11-alpine
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932
 => [internal] load build context
 => => transferring context: 263.85kB
 => CACHED [2/5] RUN mkdir -p /usr/src/app
 => CACHED [3/5] WORKDIR /usr/src/app
 => CACHED [4/5] COPY . .
 => CACHED [5/5] RUN npm install
 => exporting to image
 => => exporting layers
 => => writing image sha256:efab4e40fe60dc6c8f507844ab32787f3f49f17364cfe078f36ac741bd1af315
 => => naming to docker.io/library/employee
voclabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint
supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/employee (dev) $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee
71021a663be33dfda883780df2b55e4d7d6b8af9751f4ae9e79c65eab1357121
voclabs:~/environment/microservices/employee (dev) $
```



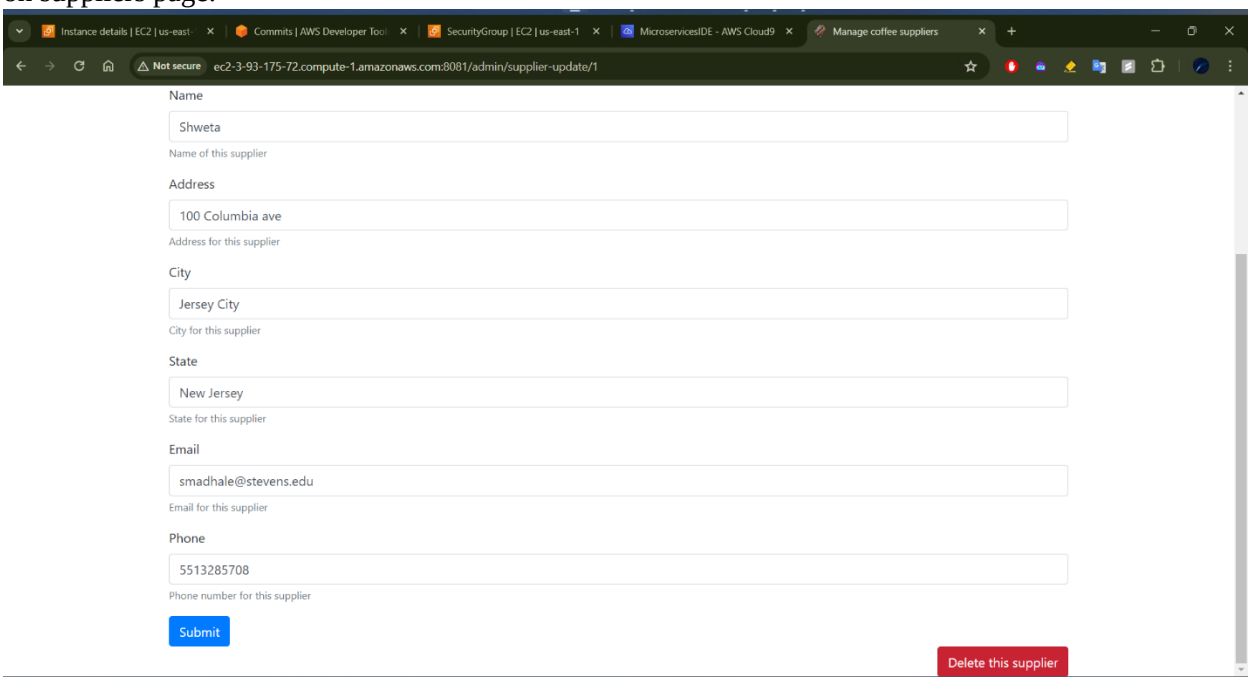
- Test adding a new supplier and verify that it appears on the suppliers page.

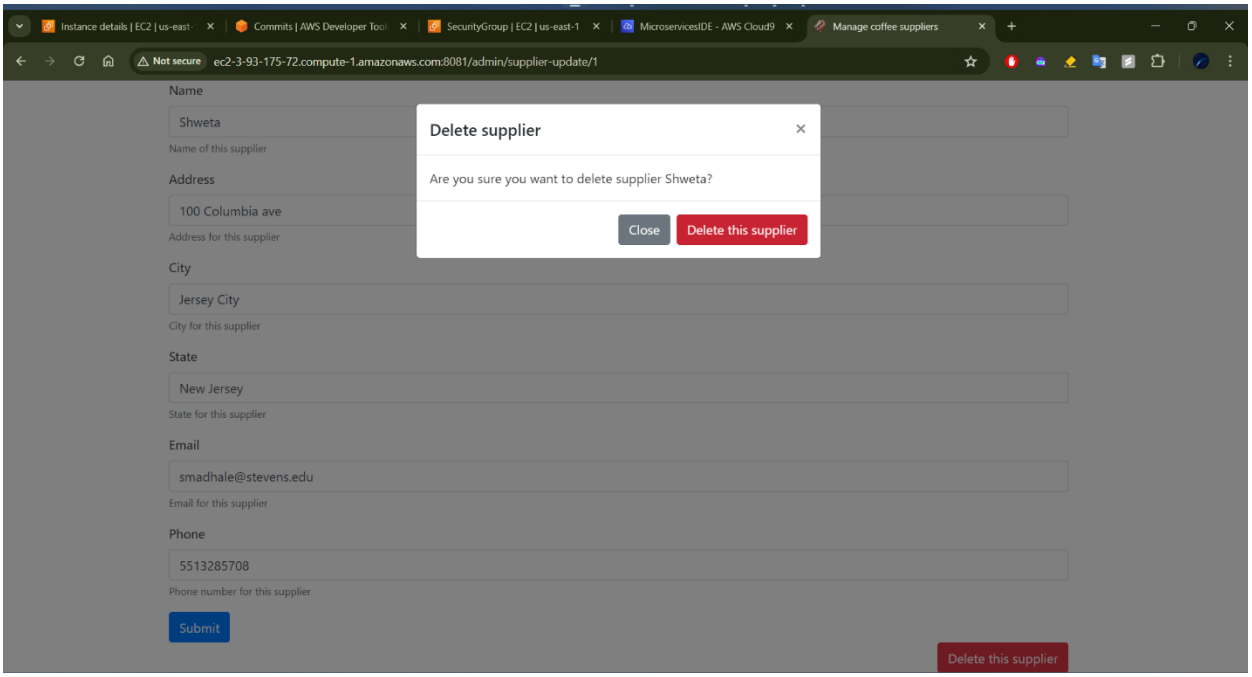

Instance details | EC2 | us-east-
Commits | AWS Developer Tool
SecurityGroup | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
Manage coffee suppliers
Not secure
ec2-3-93-175-72.compute-1.amazonaws.com:8081/admin/supplier-add
Name
shweta
Name of this supplier
Address
columbia ave
Address for this supplier
City
Jersey City
City for this supplier
State
NJ
State for this supplier
Email
smadhale@stevens.edu
Email for this supplier
Phone
5513285708
Phone number for this supplier
Submit


Instance details | EC2 | us-east-
Commits | AWS Developer Tool
SecurityGroup | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
Manage coffee suppliers
Not secure
ec2-3-93-175-72.compute-1.amazonaws.com:8081/admin/supplier-update/2
Name
shweta madhale
Name of this supplier
Address
columbia ave
Address for this supplier
City
Jersey City
City for this supplier
State
New Jersey
State for this supplier
Email
smadhale@stevens.edu
Email for this supplier
Phone
5513285708
Phone number for this supplier
Submit
Delete this supplier



- Test deleting the existing supplier, edit the entry, the supplier and verify that it no longer appears on suppliers page.





- To observe details about both running test containers, run the following command:

```
docker ps
```

```
voclabs:~/environment/microservices/employee (dev) $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED              STATUS              PORTS                                       NAMES
71021a663be3   employee   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp   employee_1
93745d4a452f   customer   "docker-entrypoint.s…"   29 minutes ago       Up 29 minutes       0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   customer_1
voclabs:~/environment/microservices/employee (dev) $
```

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

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

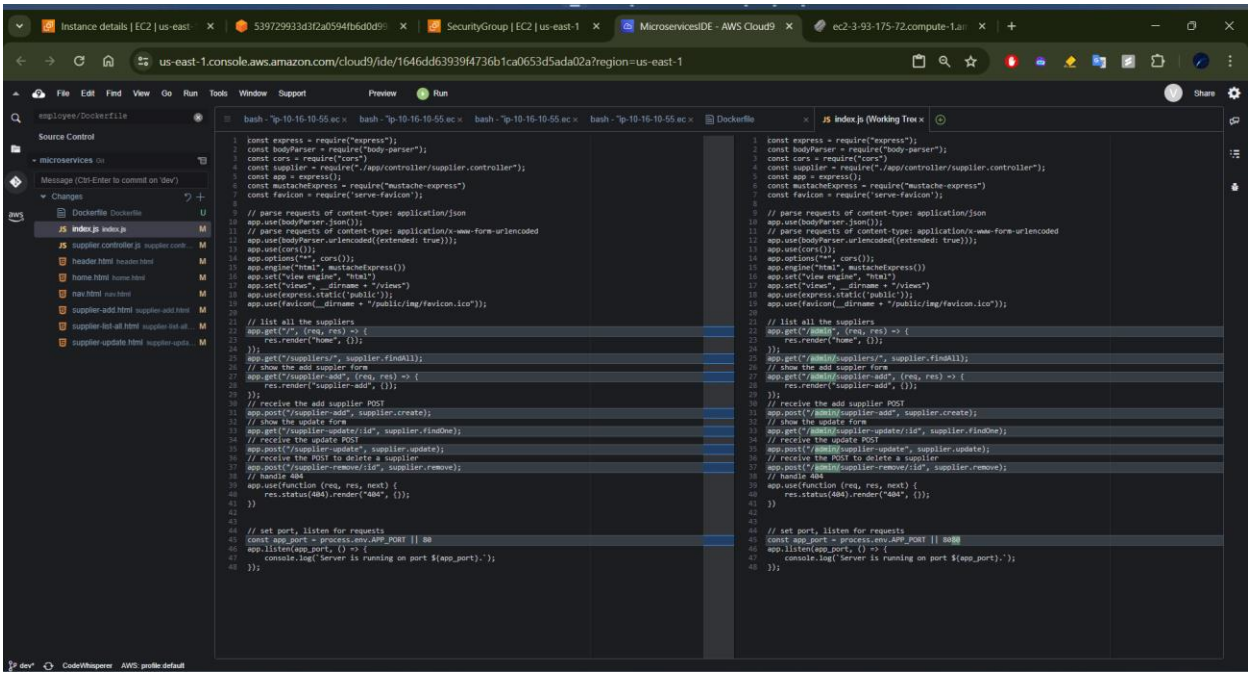
```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors")
const supplier = require("./app/controller/supplier.controller");
const app = express();
const mustacheExpress = require("mustache-express")
const favicon = require('serve-favicon');

// parse requests of content-type: application/json
app.use(bodyParser.json());
// parse requests of content-type: application/x-www-form-urlencod
app.use(bodyParser.urlencoded({extended: true}));
app.use(cors());
app.options("*", cors());
app.engine("html", mustacheExpress())
app.set("view engine", "html")
app.set("views", __dirname + "/views")
app.use(express.static('public'));
app.use(favicon(__dirname + "/public/img/favicon.ico"));

// list all the suppliers
app.get("/admin", (req, res) => {
    res.render("home", {});
});
app.get("/admin/suppliers/", supplier.findAll);
// show the add suppler form
app.get("/admin/supplier-add", (req, res) => {
    res.render("supplier-add", {});
});
// receive the add supplier POST
app.post("/admin/supplier-add", supplier.create);
// show the update form
app.get("/admin/supplier-update/:id", supplier.findOne);
// receive the update POST
app.post("/admin/supplier-update", supplier.update);
// receive the POST to delete a supplier
app.post("/admin/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
    res.status(404).render("404", {});
})


// set port, listen for requests
const app_port = process.env.APP_PORT || 8080
app.listen(app_port, () => {
    console.log(`Server is running on port ${app_port}.`);
});
```

- Rebuild the Docker image for the employee microservice.

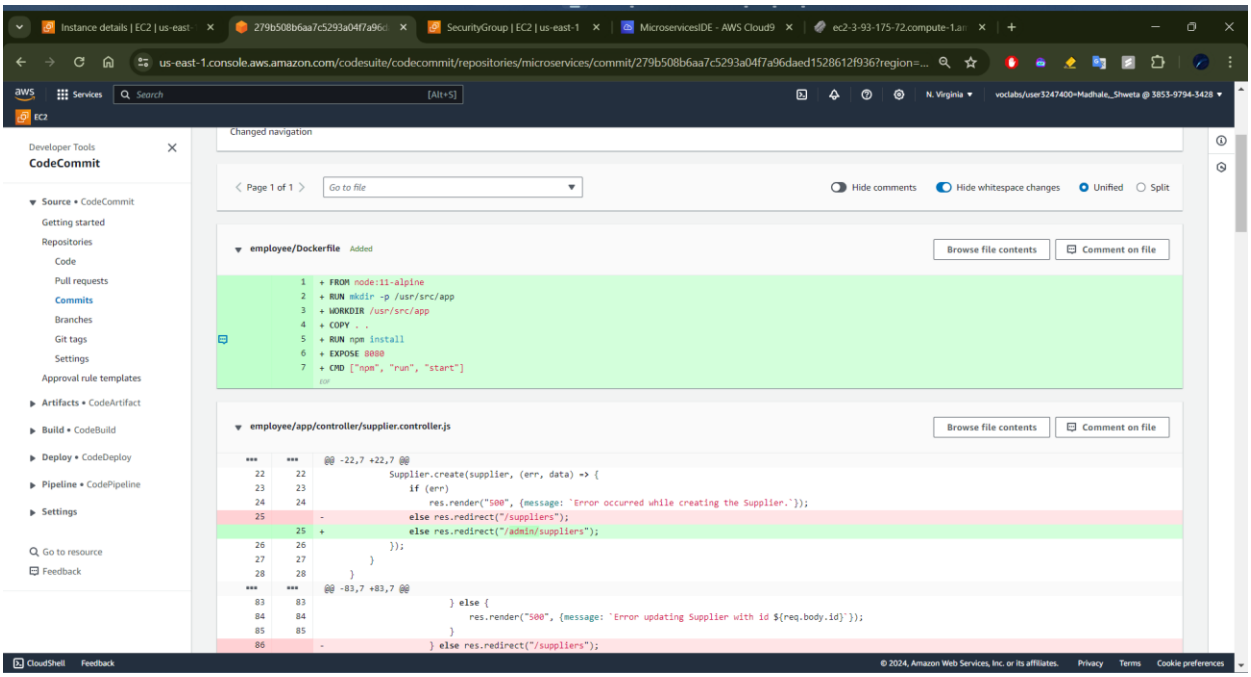
```
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
[+] Building 0.3s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 229B
 => [internal] load metadata for docker.io/library/node:11-alpine
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932
 => [internal] load build context
 => => transferring context: 263.85kB
 => CACHED [2/5] RUN mkdir -p /usr/src/app
 => CACHED [3/5] WORKDIR /usr/src/app
 => CACHED [4/5] COPY . .
 => CACHED [5/5] RUN npm install
 => exporting to image
 => => exporting layers
 => => writing image sha256:efab4e40fe60dc6c8f507844ab32787f3f49f17364cfe078f36ac741bd1af315
 => => naming to docker.io/library/employee
voclabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint
supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/employee (dev) $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee
71021a663be33dfda883780df2b55e4d7d6b8af9751f4ae9e79c65eab1357121
voclabs:~/environment/microservices/employee (dev) $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED              STATUS              PORTS                                            NAMES
71021a663be3   employee   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp        employee_1
93745d4a452f   customer   "docker-entrypoint.s…"   29 minutes ago       Up 29 minutes       0.0.0.0:8080->8080/tcp, :::8080->8080/tcp        customer_1
voclabs:~/environment/microservices/employee (dev) $ docker rm -f employee_1
employee_1
voclabs:~/environment/microservices/employee (dev) $
```

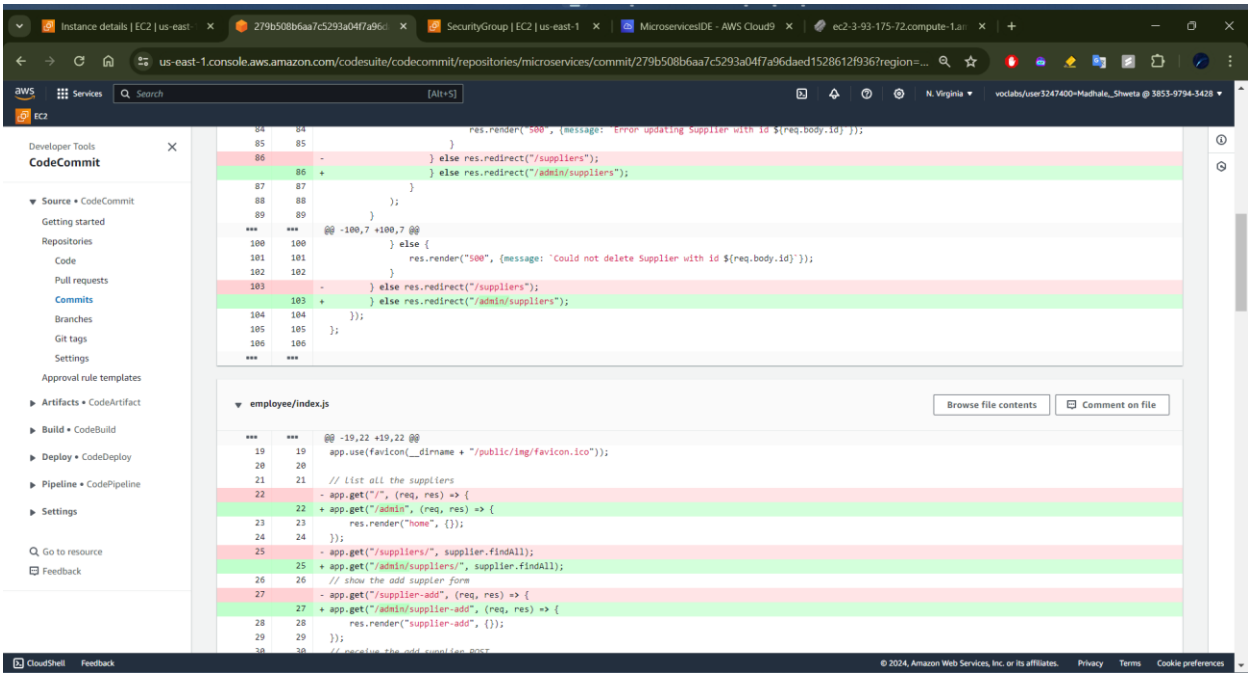
Task 4.7: Check code into CodeCommit

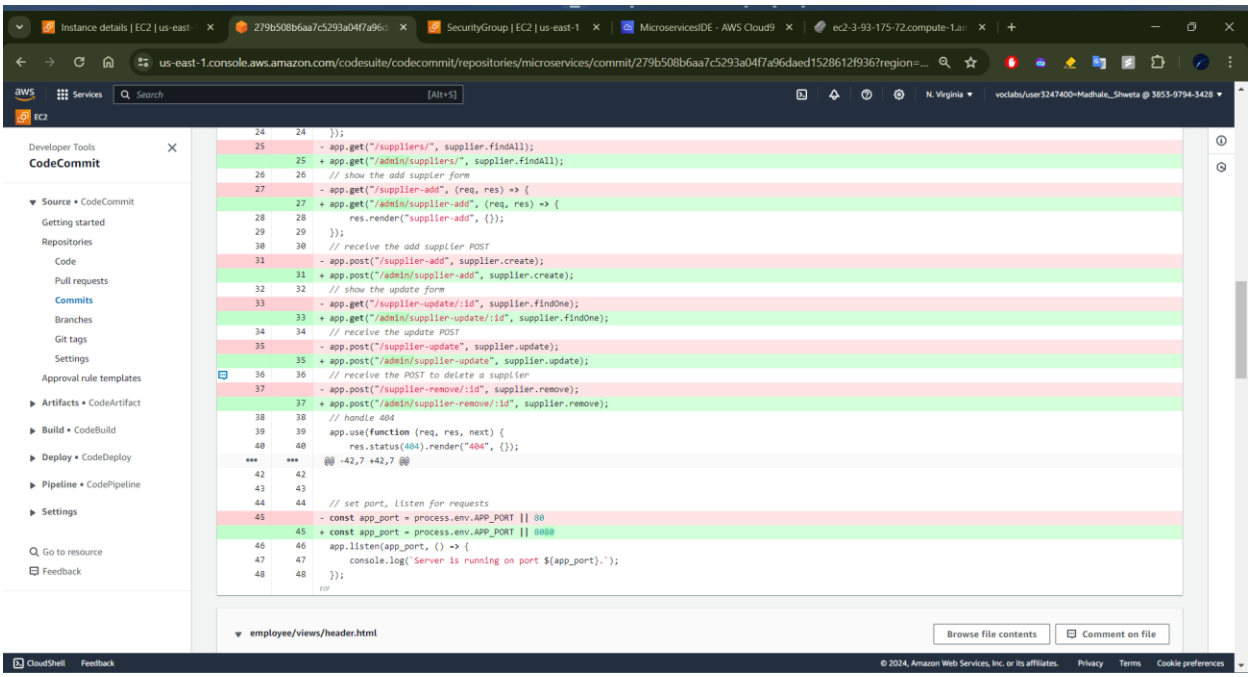
- For reviewing the updates made to the source code, going to the source control in AWS Cloud9 IDE, review changes and check changes into CodeCommit.

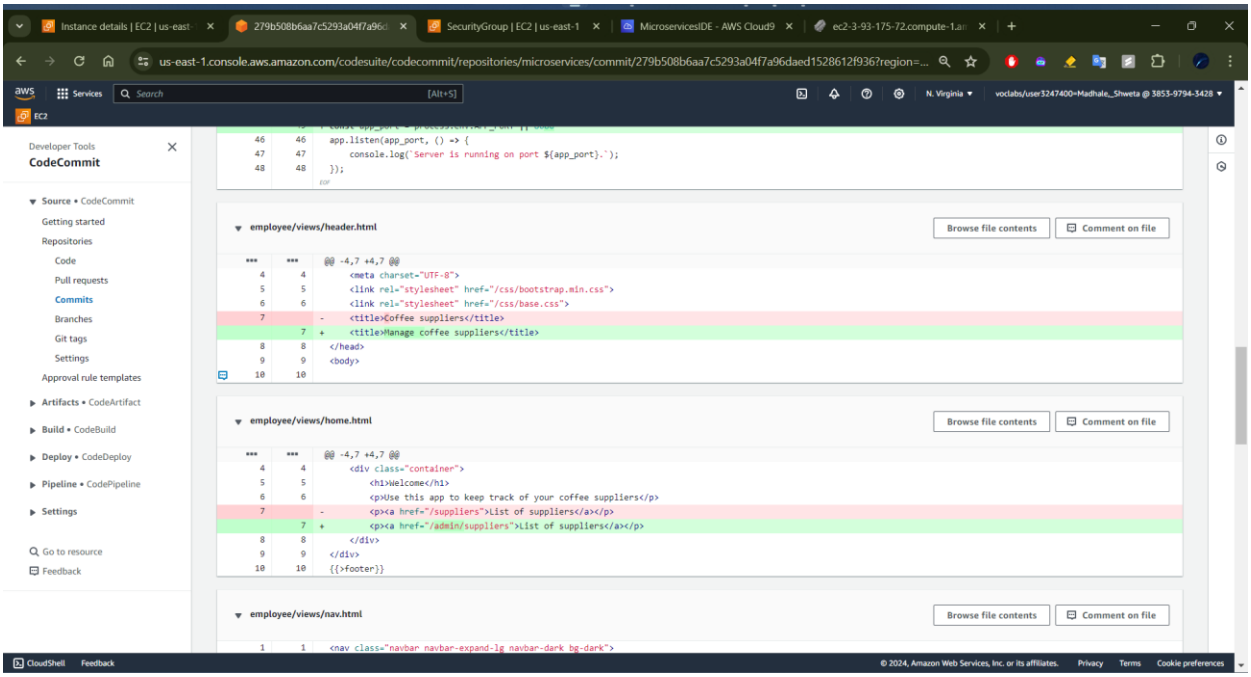


```
voclabs:~/environment/microservices (dev) $ git push origin dev
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 1.60 KiB | 821.00 KiB/s, done.
Total 15 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
   5397299..279b508  dev -> dev
voclabs:~/environment/microservices (dev) $
```



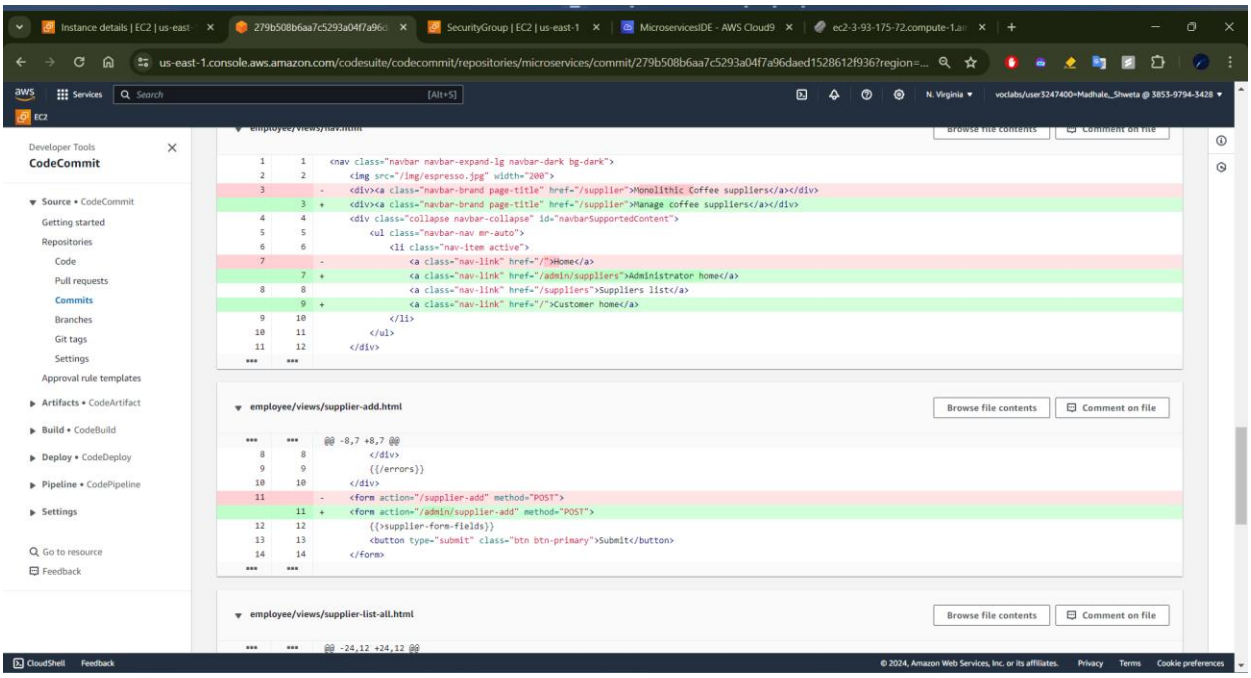

Instance details | EC2 | us-east-1
279b508b6aa7c5293a04f7a96d
SecurityGroup | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
ec2-3-93-175-72.compute-1.am
us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
Go to resource
Feedback
res.render("500", {message: `Error updating Supplier with id ${req.body.id}`});
} else res.redirect("/suppliers");
} else res.redirect("/admin/suppliers");
} else res.redirect("/suppliers");
} else res.redirect("/admin/suppliers");
employee/index.js
Browse file contents
Comment on file
@@ -19,22 +19,22 @@
app.use(favicon(__dirname + "/public/img/favicon.ico"));
// list all the suppliers
- app.get("/", (req, res) => {
+ app.get("/admin", (req, res) => {
res.render("home", {});
});
- app.get("/suppliers/", supplier.findAll);
+ app.get("/admin/suppliers/", supplier.findAll);
// show the add supplier form
- app.get("/supplier-add", (req, res) => {
+ app.get("/admin/supplier-add", (req, res) => {
res.render("supplier-add", {});
});
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-1
279b508b6aa7c5293a04f7a96d
SecurityGroup | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
ec2-3-93-175-72.compute-1.am
us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
Go to resource
Feedback
});
- app.get("/suppliers/", supplier.findAll);
+ app.get("/admin/suppliers/", supplier.findAll);
// show the add supplier form
- app.get("/supplier-add", (req, res) => {
+ app.get("/admin/supplier-add", (req, res) => {
res.render("supplier-add", {});
});
// receive the add supplier POST
- app.post("/supplier-add", supplier.create);
+ app.post("/admin/supplier-add", supplier.create);
// show the update form
- app.get("/supplier-update/:id", supplier.findOne);
+ app.get("/admin/supplier-update/:id", supplier.findOne);
// receive the update POST
- app.post("/supplier-update", supplier.update);
+ app.post("/admin/supplier-update", supplier.update);
// receive the POST to delete a supplier
- app.post("/supplier-remove/:id", supplier.remove);
+ app.post("/admin/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
res.status(404).render("404", {});
@@ -42,7 +42,7 @@
// set port, listen for requests
- const app_port = process.env.APP_PORT || 80
+ const app_port = process.env.APP_PORT || 8080
app.listen(app_port, () => {
console.log(`Server is running on port ${app_port}.`);
});
EOF
employee/views/header.html
Browse file contents
Comment on file
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-1
279b508b6aa7c5293a04f7a96d
SecurityGroup | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
ec2-3-93-175-72.compute-1.am
us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale._Shweta @ 3853-9794-3428
EC2
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
Go to resource
Feedback
app.listen(app_port, () => {
console.log(`Server is running on port ${app_port}.`);
});
employee/views/header.html
Browse file contents
Comment on file
@@ -4,7 +4,7 @@
<meta charset="UTF-8">
<link rel="stylesheet" href="/css/bootstrap.min.css">
<link rel="stylesheet" href="/css/base.css">
<title>Coffee suppliers</title>
<title>Manage coffee suppliers</title>
</head>
<body>
employee/views/home.html
Browse file contents
Comment on file
@@ -4,7 +4,7 @@

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

{{>footer}}
employee/views/nav.html
Browse file contents
Comment on file
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

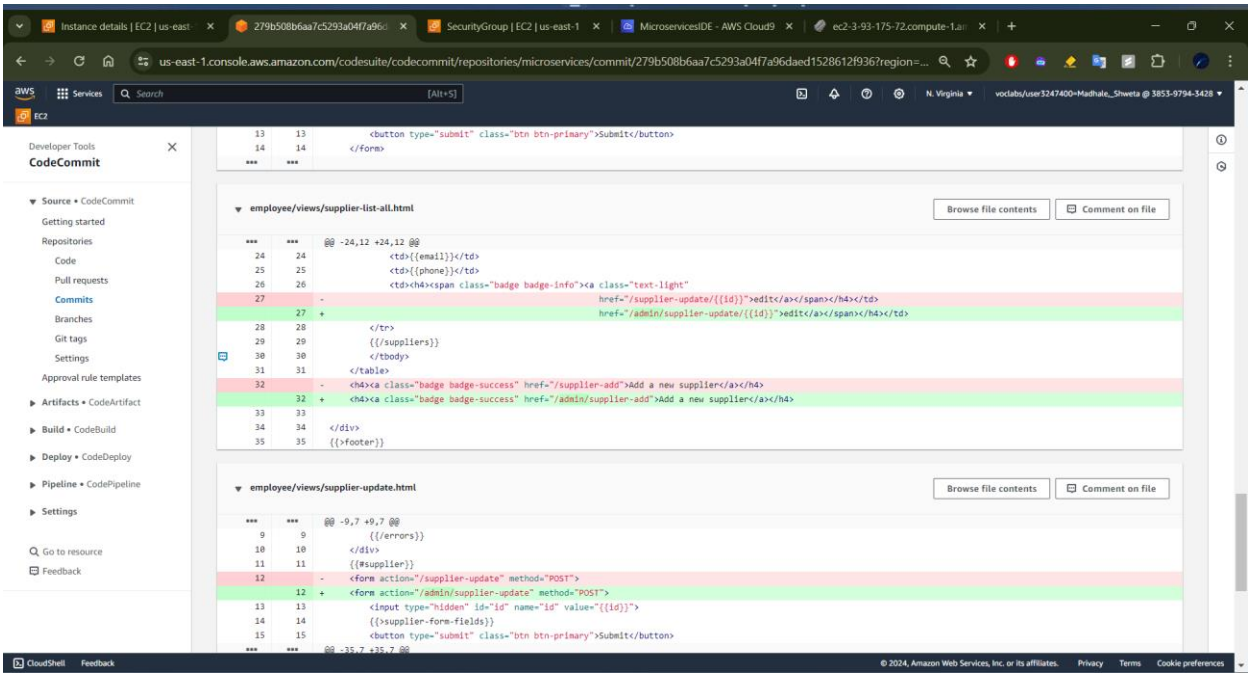

Instance details | EC2 | us-east-1
279b508b6aa7c5293a04f7a96d
SecurityGroup | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
ec2-3-93-175-72.compute-1.am
us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale._Shweta @ 3853-9794-3428
EC2
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
Go to resource
Feedback
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<img src="/img/espresso.jpg" width="200">
Monolithic Coffee suppliers
Manage coffee suppliers

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

employee/views/supplier-add.html
Browse file contents
Comment on file
@@ -8,7 +8,7 @@

{{/errors}}

<form action="/supplier-add" method="POST">
<form action="/admin/supplier-add" method="POST">
{{>supplier-form-fields}}
<button type="submit" class="btn btn-primary">Submit</button>
</form>
employee/views/supplier-list-all.html
Browse file contents
Comment on file
@@ -24,12 +24,12 @@
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Phase 5 – Create ECR repositories to store Docker images. Create an ECS cluster, ECS task definitions, and CodeDeploy application specification files.

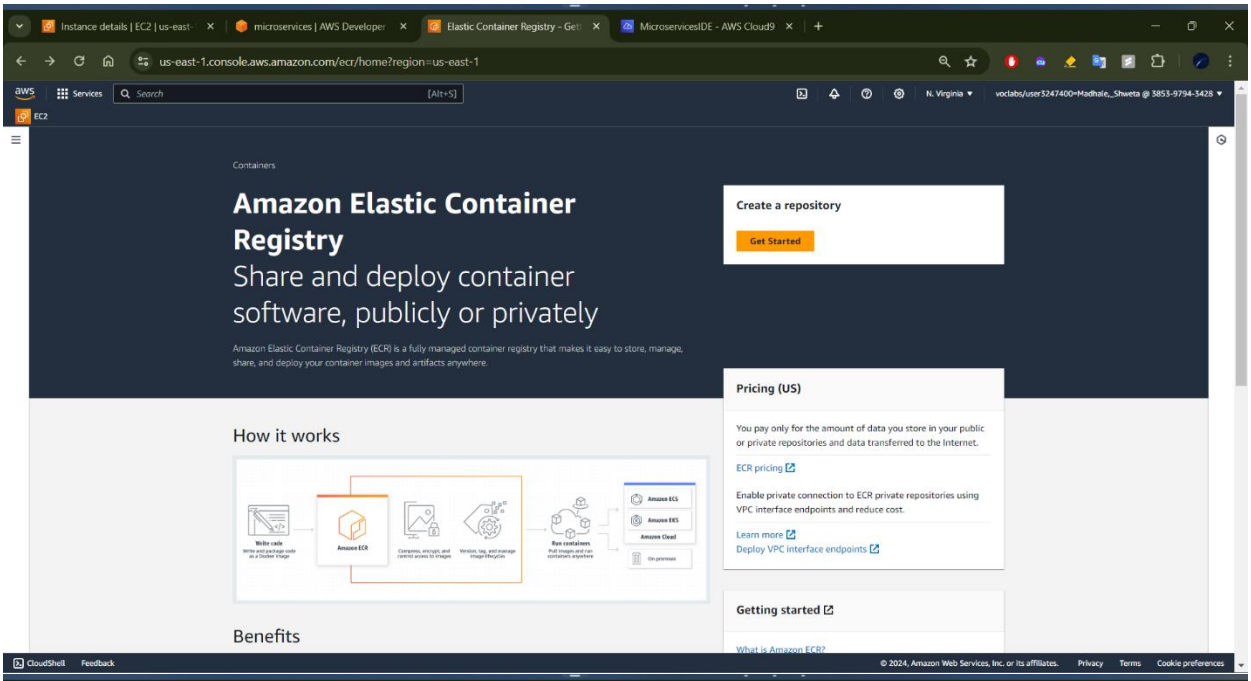
Task 5.1: Create ECR repositories and upload the Docker images

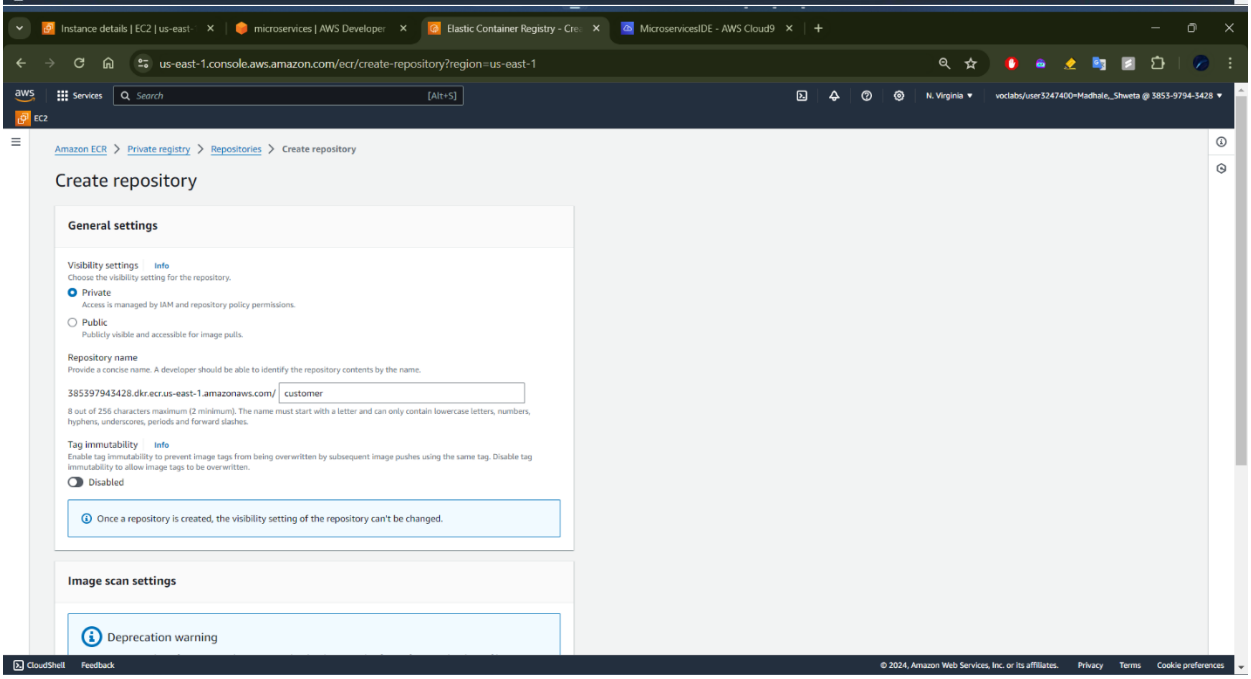
- Authorize the docker client to connect to the Amazon ECR service,

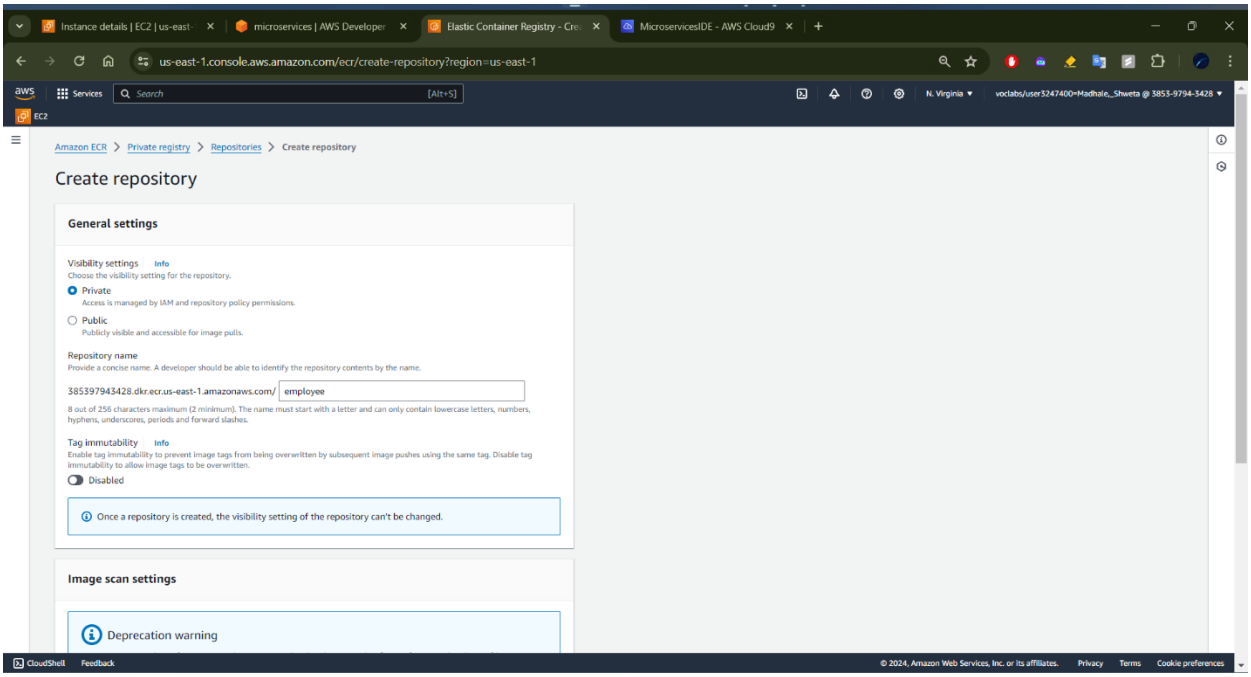
```
account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)
echo $account_id
aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
```

```
voclabs:~/environment/microservices (dev) $ 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/microservices (dev) $
```

- Create separate private ECR repositories customer and employee.

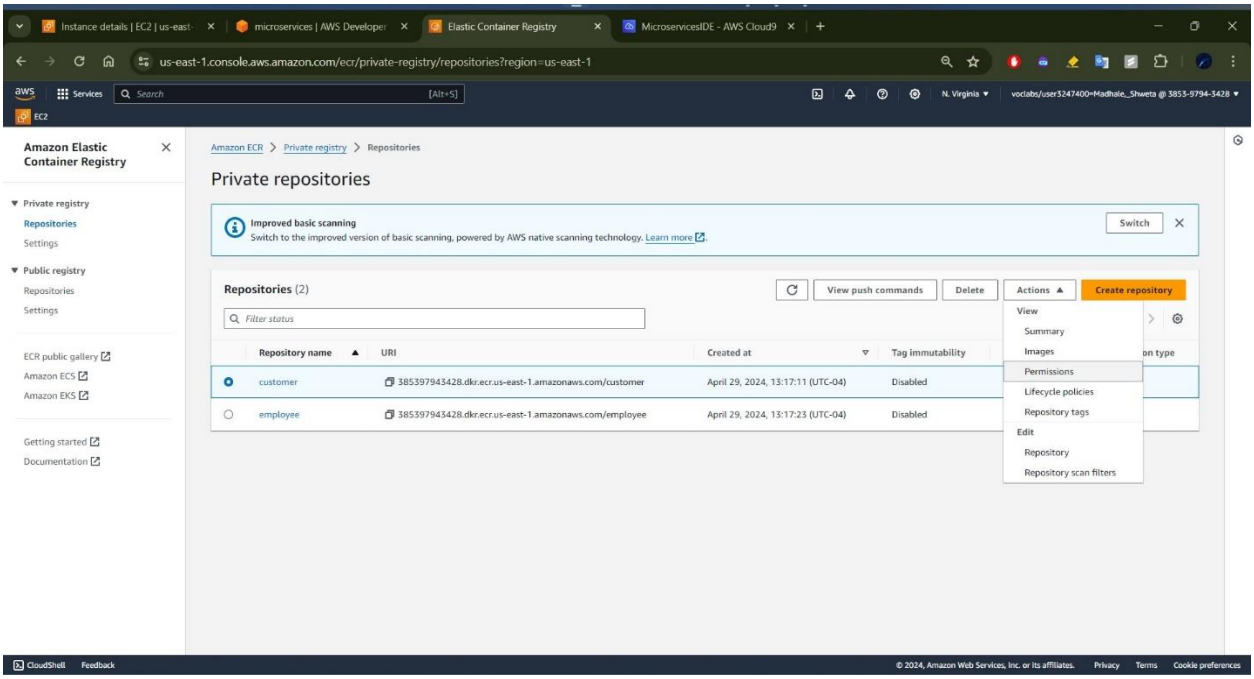

Instance details | EC2 | us-east-
microservices | AWS Developer
Elastic Container Registry - Get
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ecr/home?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Containers
Amazon Elastic Container Registry
Share and deploy container software, publicly or privately
Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, manage, share, and deploy your container images and artifacts anywhere.
Create a repository
Get Started
Pricing (US)
You pay only for the amount of data you store in your public or private repositories and data transferred to the Internet.
ECR pricing
Enable private connection to ECR private repositories using VPC interface endpoints and reduce cost.
Learn more
Deploy VPC interface endpoints
Getting started
What is Amazon ECR?
How it works
Write code
Amazon ECR
Amazon ECS
Amazon EKS
Amazon Cloud
Benefits
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

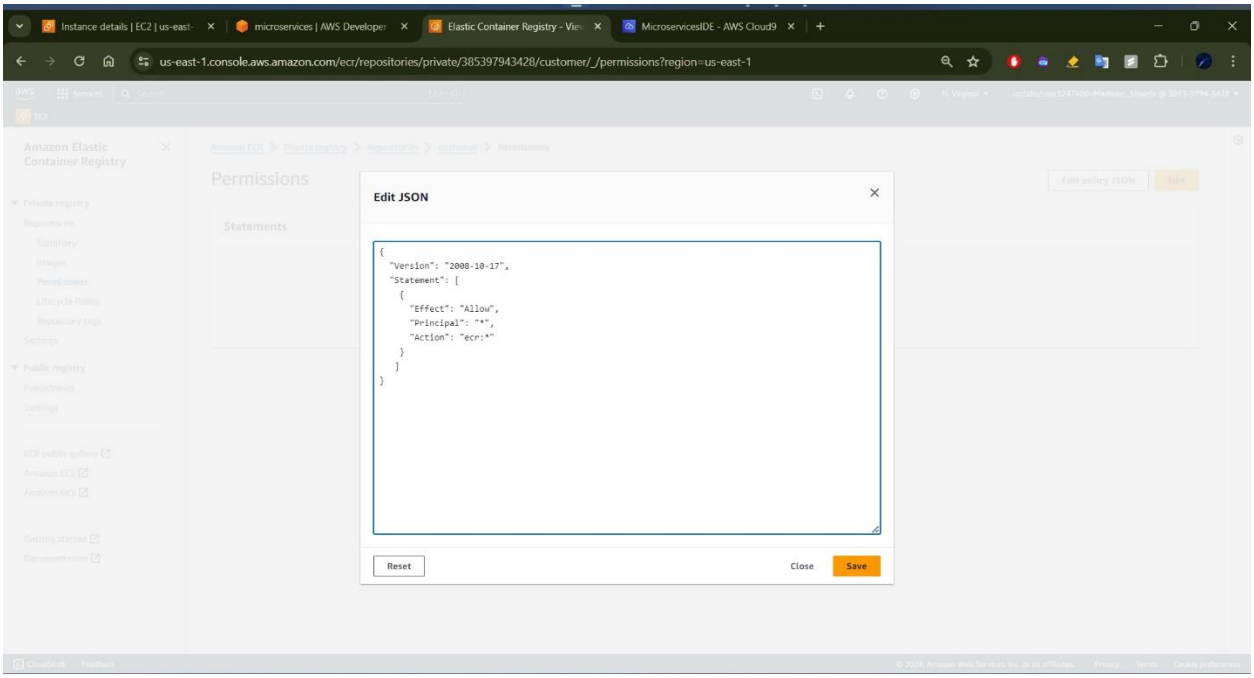

Instance details | EC2 | us-east-
microservices | AWS Developer
Elastic Container Registry - Cre
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ecr/create-repository?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Amazon ECR > Private registry > Repositories > Create repository
Create repository
General settings
Visibility settings Info
Choose the visibility setting for the repository.
Private
Access is managed by IAM and repository policy permissions.
Public
Publicly visible and accessible for image pulls.
Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.
385397943428.dkr.ecr.us-east-1.amazonaws.com/
customer
8 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.
Tag immutability Info
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.
Disabled
Once a repository is created, the visibility setting of the repository can't be changed.
Image scan settings
Deprecation warning
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

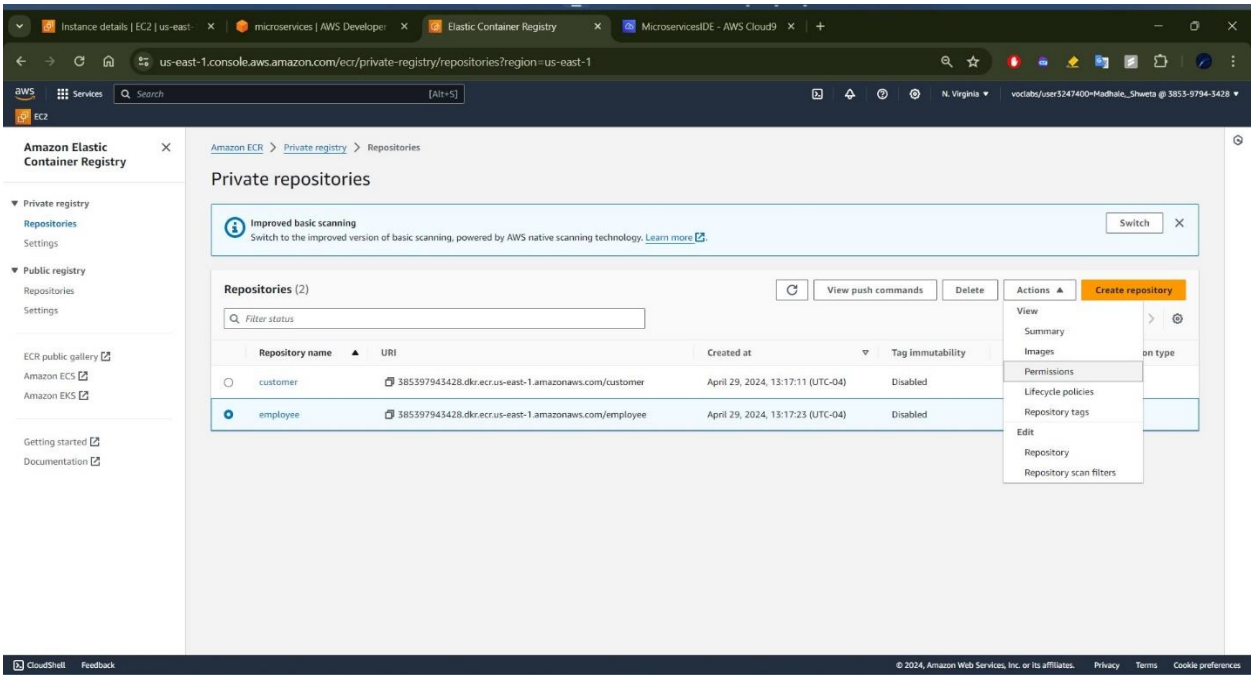


- Set permissions on the customer ECR repository by replacing the existing lines with given code.

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


Instance details | EC2 | us-east-
microservices | AWS Developer
Elastic Container Registry
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ecr/private-registry/repositories?region=us-east-1
Amazon Elastic Container Registry
Private registry
Repositories
Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
Amazon ECR > Private registry > Repositories
Private repositories
Improved basic scanning
Switch to the improved version of basic scanning, powered by AWS native scanning technology. Learn more.
Switch
Repositories (2)
View push commands
Delete
Actions
Create repository
Filter status
Repository name
URI
Created at
Tag immutability
customer
385397943428.dkr.ecr.us-east-1.amazonaws.com/customer
April 29, 2024, 13:17:11 (UTC-04)
Disabled
employee
385397943428.dkr.ecr.us-east-1.amazonaws.com/employee
April 29, 2024, 13:17:23 (UTC-04)
Disabled
View
Summary
Images
Permissions
Lifecycle policies
Repository tags
Edit
Repository
Repository scan filters
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
microservices | AWS Developer
Elastic Container Registry - View
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ecr/repositories/private/385397943428/customer/_/permissions?region=us-east-1
Permissions
Statements
Edit JSON
{
"Version": "2008-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": "*",
"Action": "ecr:*"
}
]
}
Reset
Close
Save



- Tagging the Docker images with account ID, running following commands.

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

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

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

# Tag the employee image
docker tag employee:latest $account_id.dkr.ecr.us-east-
1.amazonaws.com/employee:latest
```

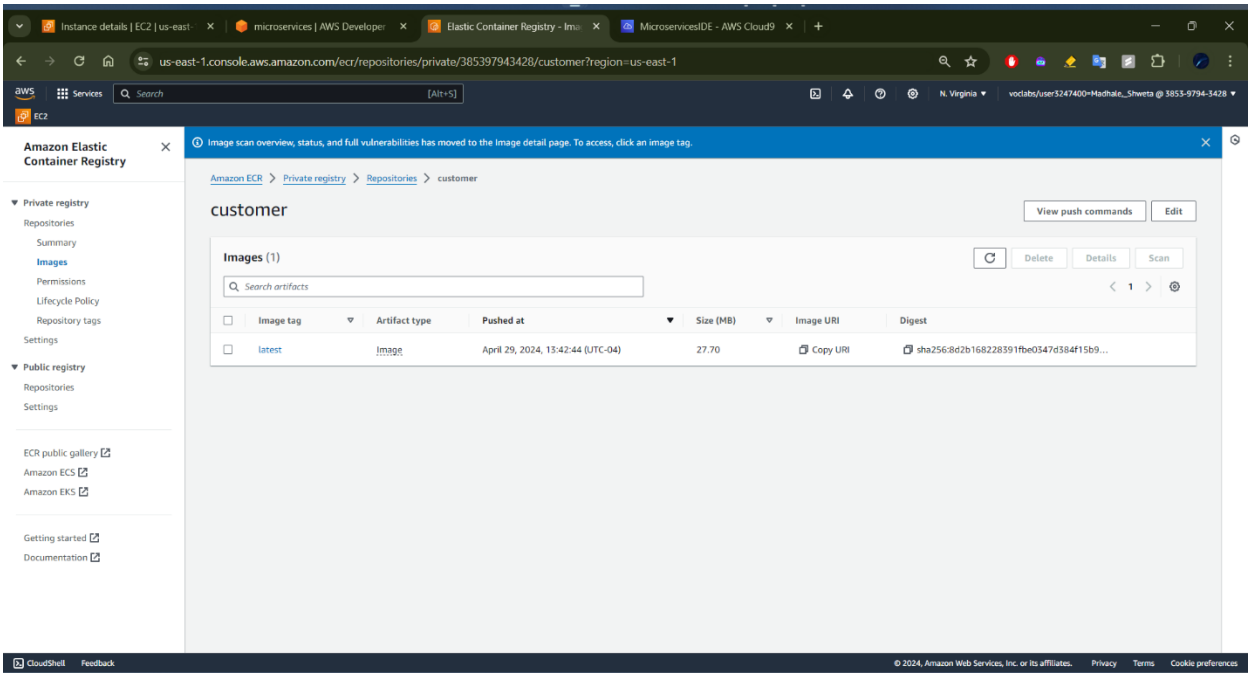
```
voclabs:~/environment $ account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)
voclabs:~/environment $ echo $account_id
385397943428
voclabs:~/environment $ docker tag customer:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
voclabs:~/environment $ docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
voclabs:~/environment $ docker images
REPOSITORY                                                TAG       IMAGE ID       CREATED              SIZE
385397943428.dkr.ecr.us-east-1.amazonaws.com/employee     latest    fa0c6341f894   54 minutes ago       82.7MB
employee                                                  latest    fa0c6341f894   54 minutes ago       82.7MB
<none>                                                    <none>    efab4e40fe60   About an hour ago    82.7MB
385397943428.dkr.ecr.us-east-1.amazonaws.com/customer     latest    2c0fc5a9f048   About an hour ago    82.7MB
customer                                                  latest    2c0fc5a9f048   About an hour ago    82.7MB
voclabs:~/environment $
```

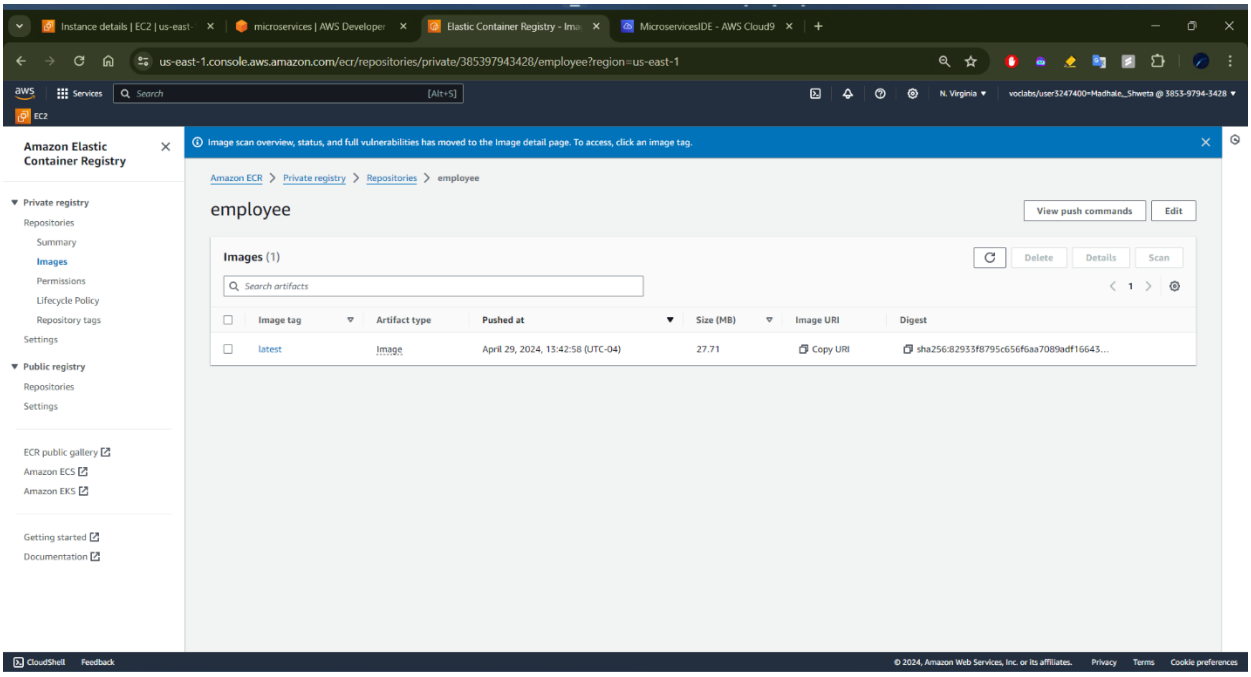
- Run appropriate docker command to push each of Docker images to Amazon ECR.

```
account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)
docker REPLACE_ME $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
docker REPLACE_ME $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
```

```
voclabs:~/environment $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
The push refers to repository [385397943428.dkr.ecr.us-east-1.amazonaws.com/customer]
216f113d8053: Pushed
8645e45774f0: Pushed
5f70bf18a086: Pushed
6f8e0edab334: Pushed
d81d715330b7: Pushed
1dc7f3bb09a4: Pushed
dcaceb729824: Pushed
f1b5933fe4b5: Pushed
latest: digest: sha256:8d2b168228391fbe0347d384f15b9bb10d3e9ba94e43e9034d8978ca655bcc20 size: 1989
voclabs:~/environment $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
The push refers to repository [385397943428.dkr.ecr.us-east-1.amazonaws.com/employee]
bcfa07be99c3: Pushed
18fe5390168d: Pushed
5f70bf18a086: Pushed
6f8e0edab334: Pushed
d81d715330b7: Pushed
1dc7f3bb09a4: Pushed
dcaceb729824: Pushed
f1b5933fe4b5: Pushed
latest: digest: sha256:82933f8795c656f6aa7089adf166433f254bc0503df39dccf708c4ebfdb72201 size: 1989
voclabs:~/environment $ docker images
REPOSITORY                                                TAG       IMAGE ID       CREATED             SIZE
385397943428.dkr.ecr.us-east-1.amazonaws.com/employee     latest    fa0c6341f894   55 minutes ago      82.7MB
employee                                                  latest    fa0c6341f894   55 minutes ago      82.7MB
<none>                                                    <none>    efab4e40fe60   About an hour ago   82.7MB
385397943428.dkr.ecr.us-east-1.amazonaws.com/customer     latest    2c0fc5a9f048   About an hour ago   82.7MB
customer                                                  latest    2c0fc5a9f048   About an hour ago   82.7MB
voclabs:~/environment $
```

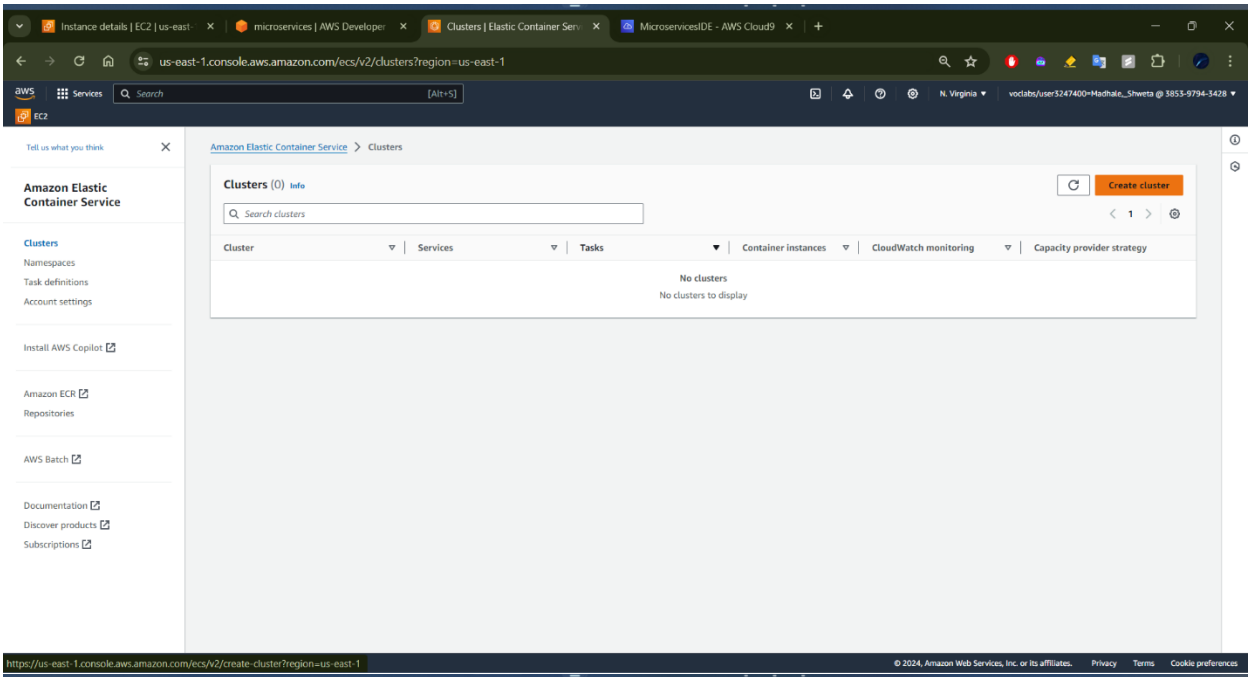
- Confirm that the two images are now stored in Amazon ECR and that each has the latest label applied.

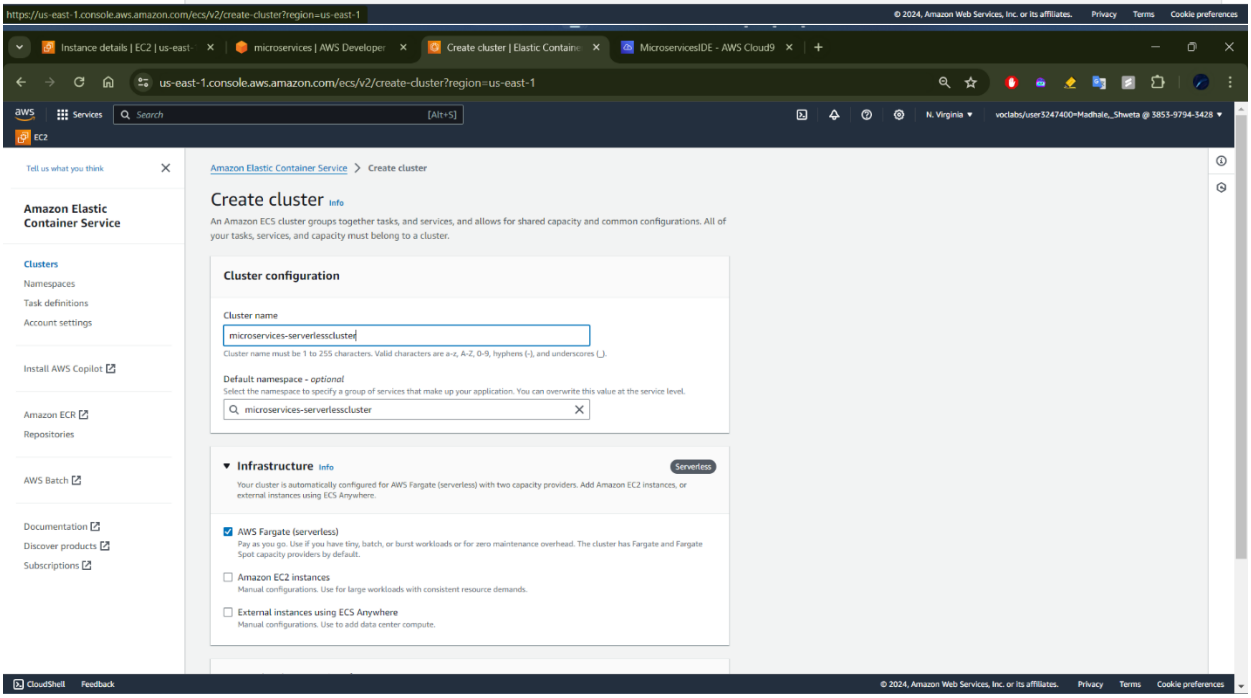


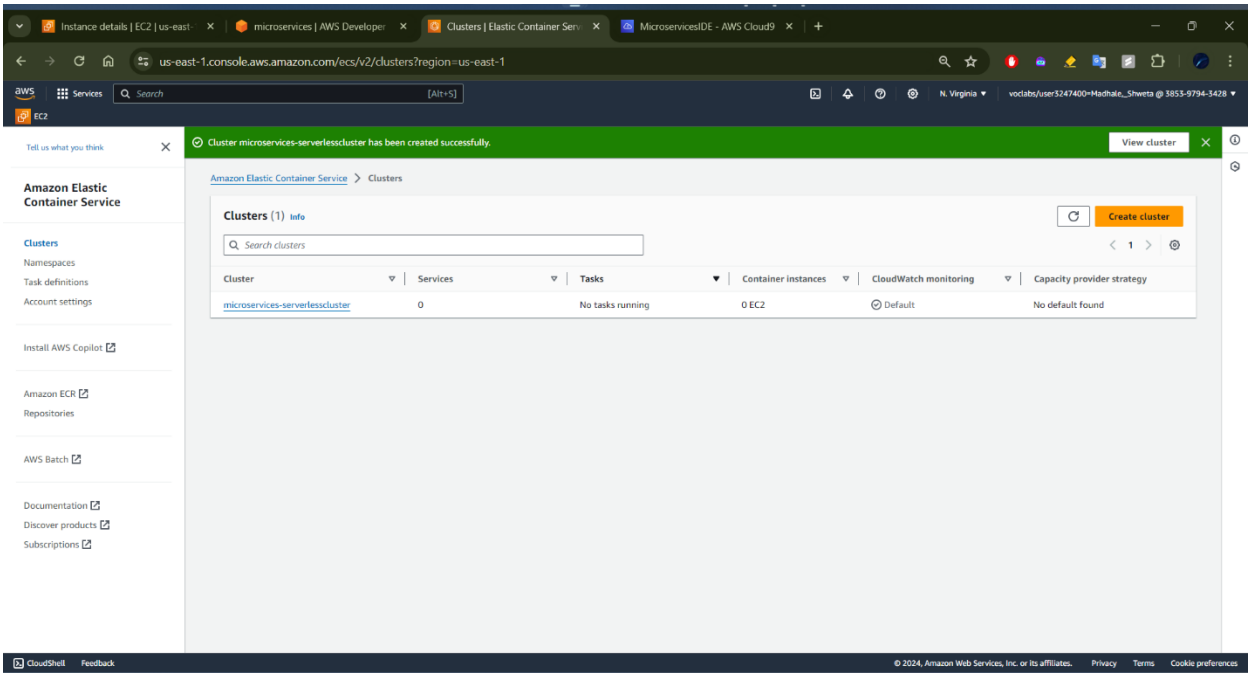


Task 5.2: Create an ECS cluster

- Create a serverless AWS Fargate cluster that is named microservices-serverlesscluster configured to use LabVPC, PublicSubnet1, and PublicSubnet2

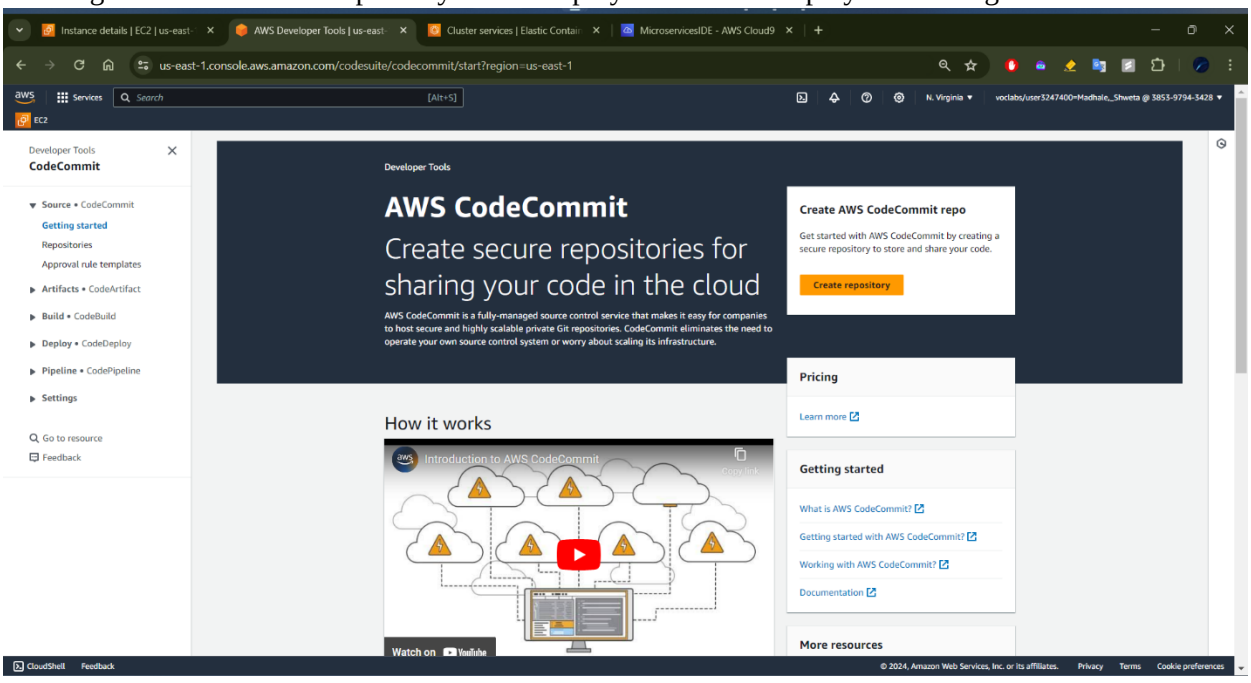

Instance details | EC2 | us-east-
microservices | AWS Developer
Clusters | Elastic Container Serv
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ecs/v2/clusters?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Tell us what you think
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Install AWS Copilot
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Amazon Elastic Container Service > Clusters
Clusters (0) Info
Search clusters
Create cluster
Cluster
Services
Tasks
Container instances
CloudWatch monitoring
Capacity provider strategy
No clusters
No clusters to display

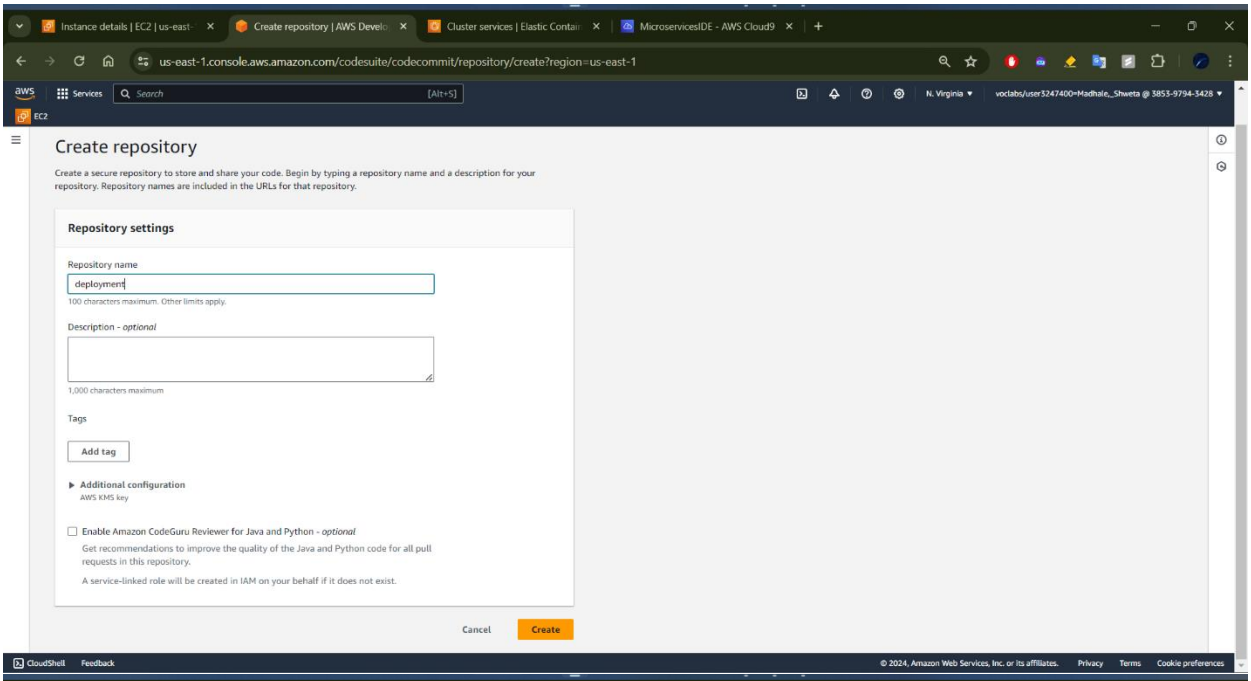

https://us-east-1.console.aws.amazon.com/ecs/v2/create-cluster?region=us-east-1
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Instance details | EC2 | us-east-
microservices | AWS Developer
Create cluster | Elastic Containe
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ecs/v2/create-cluster?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
Subscriptions
Amazon Elastic Container Service > Create cluster
Create cluster Info
An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.
Cluster configuration
Cluster name
microservices-serverlesscluster
Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).
Default namespace - optional
Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.
microservices-serverlesscluster
Infrastructure Info
Serverless
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.
AWS Fargate (serverless)
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.
Amazon EC2 instances
Manual configurations. Use for large workloads with consistent resource demands.
External instances using ECS Anywhere
Manual configurations. Use to add data center compute.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

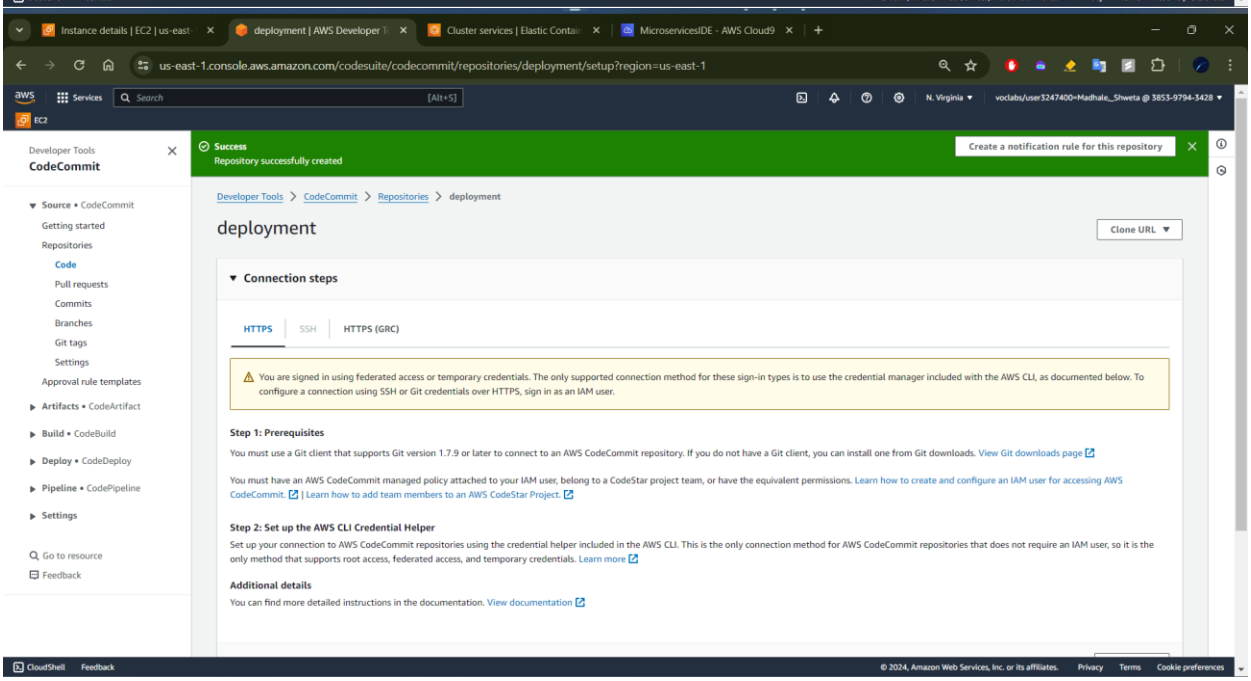


Task 5.3: Create a CodeCommit repository to store deployment files

- Creating new CodeCommit repository named deploymentto store deployment configuration files.







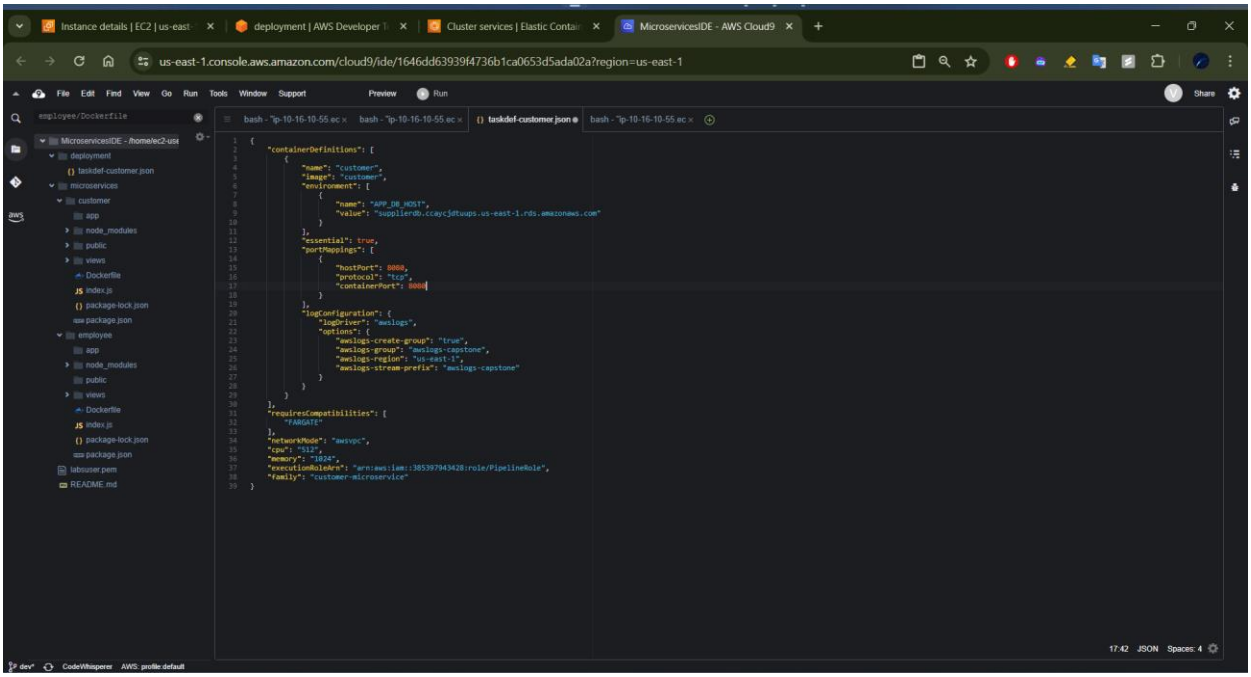
- In AWS Cloud9, in the environment directory, create a new directory that is named deployment. Initialize the directory as a Git repository with a branch named dev.

```
voclabs:~/environment/deployment $ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ec2-user/environment/deployment/.git/
voclabs:~/environment/deployment (master) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
voclabs:~/environment/deployment (master) $ git checkout -b dev
Switched to a new branch 'dev'
voclabs:~/environment/deployment (dev) $
```

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

- In deployment directory, create an empty file named taskdef-customer.json with following code

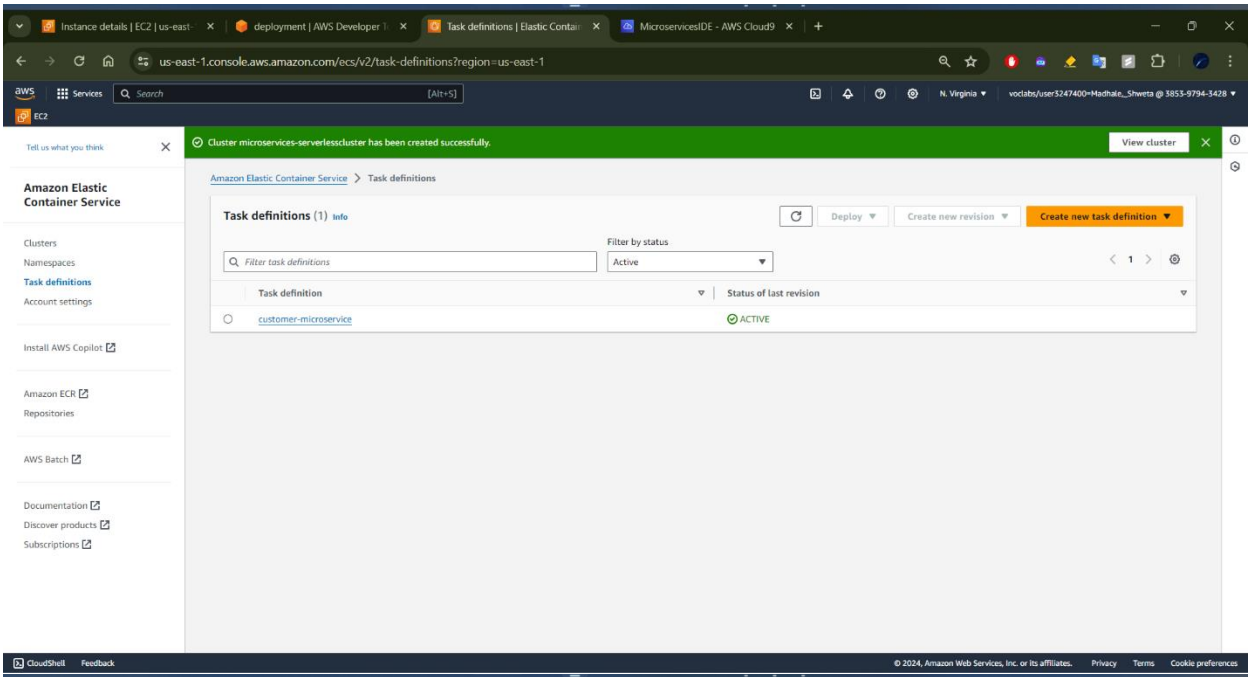
```
{
    "containerDefinitions": [
        {
            "name": "customer",
            "image": "customer",
            "environment": [
                {
                    "name": "APP_DB_HOST",
                    "value": "<RDS-ENDPOINT>"
                }
            ],
            "essential":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

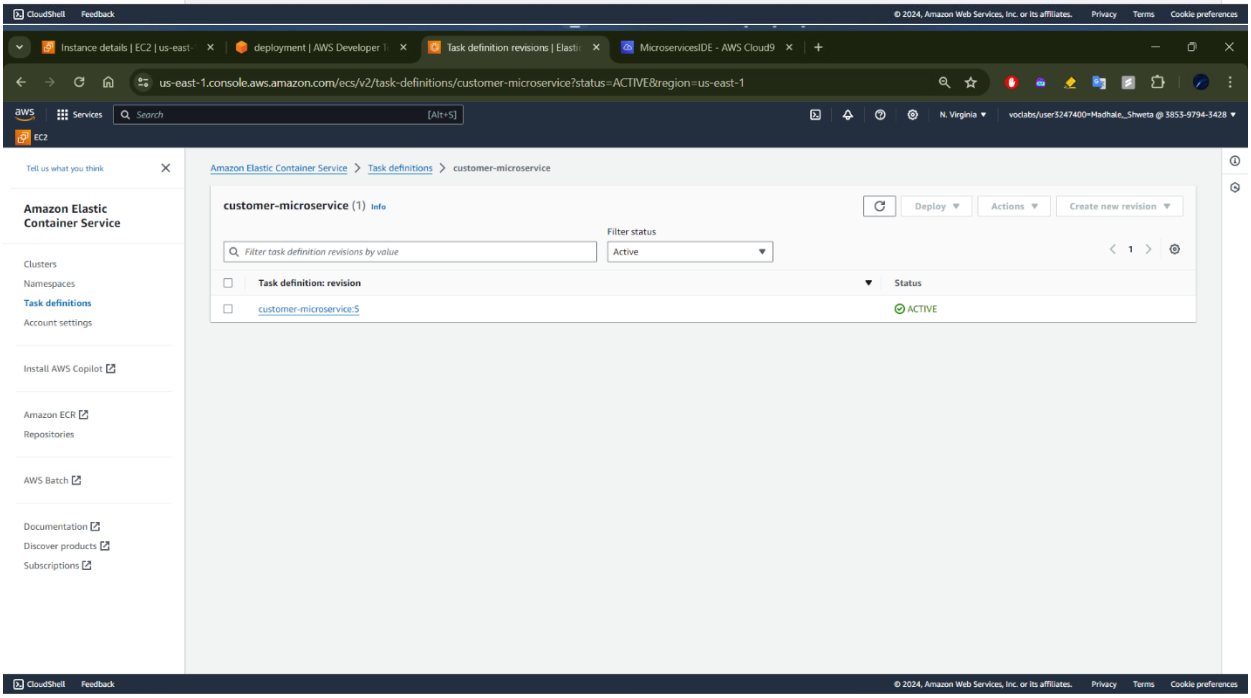


- To register the customer microservice task definition in Amazon ECS, run the following command:

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

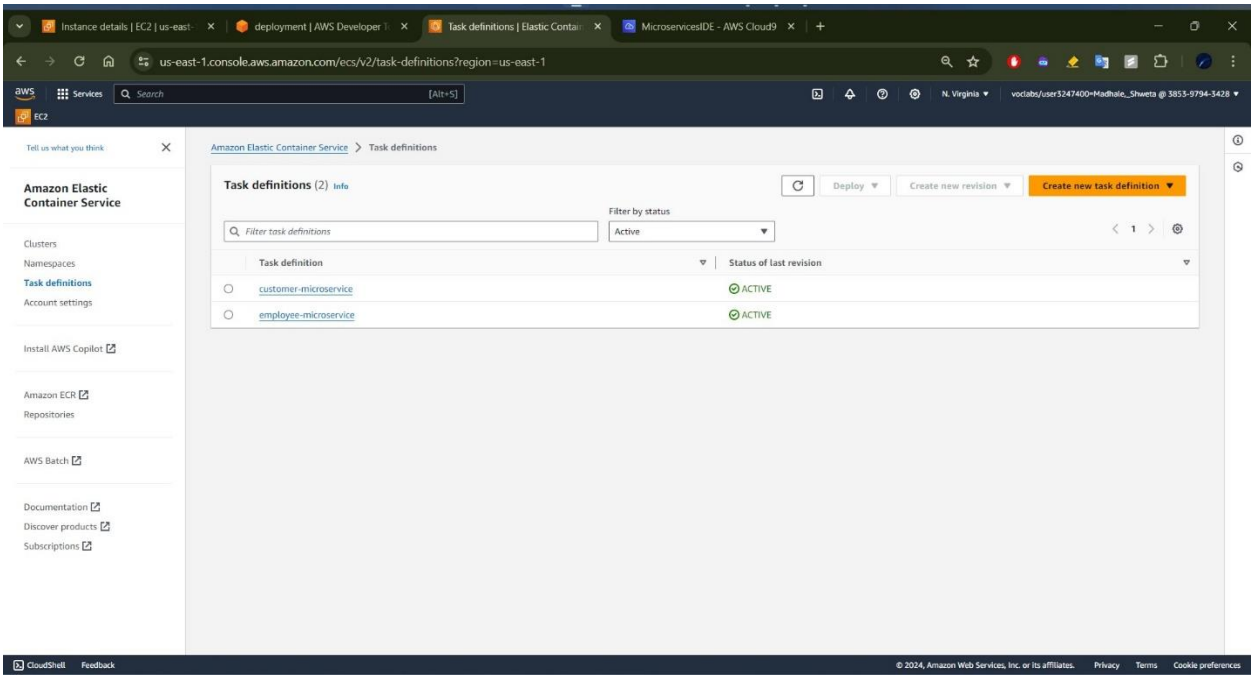
- Verify that the customer-microservice task definition now appears in the Task definitions pane

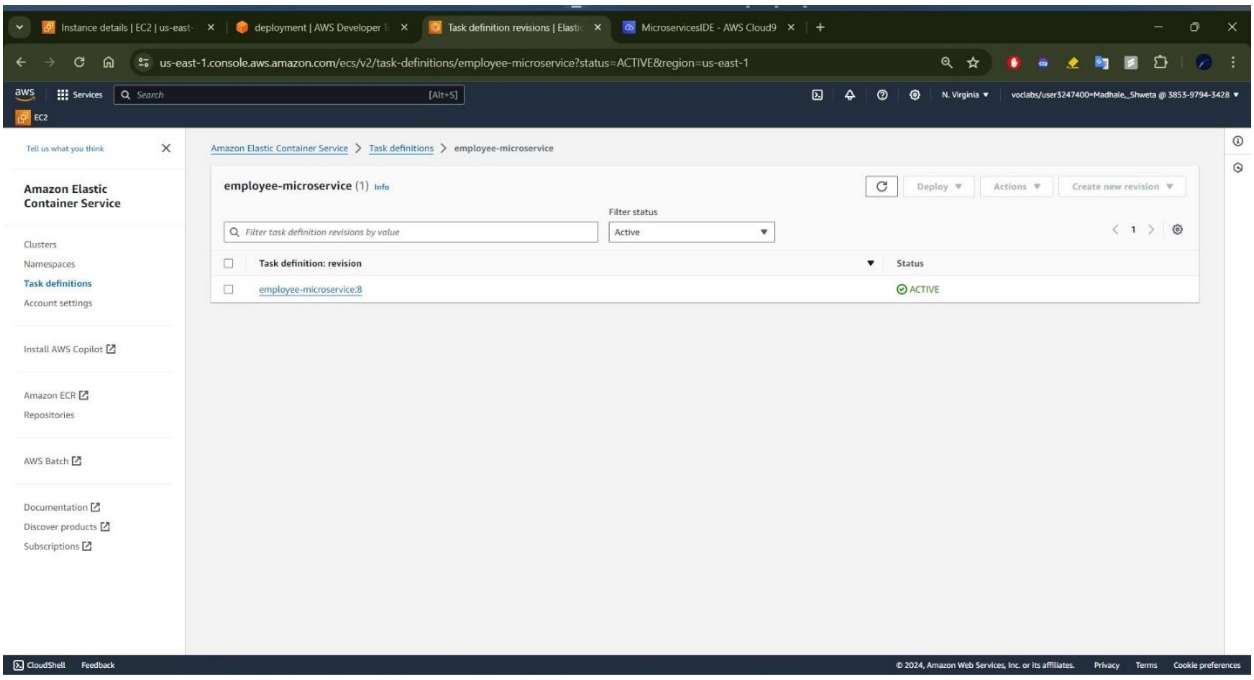




- Doing the same for taskdef-employee.json

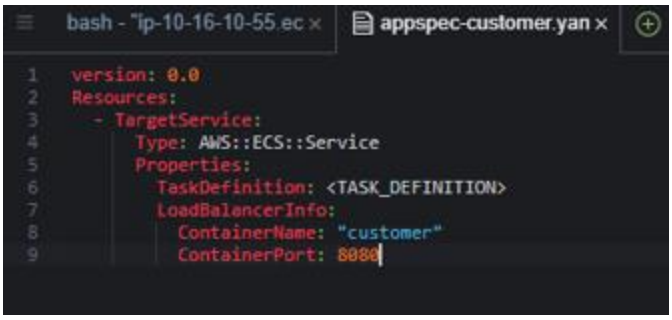
```
{
    "taskDefinition": {
        "taskDefinitionArn": "arn:aws:ecs:us-east-1:385397943428:task-definition/employee-microservice:8",
        "containerDefinitions": [
            {
                "name": "employee",
                "image": "employee",
                "cpu": 0,
                "portMappings": [
                    {
                        "containerPort": 8080,
                        "hostPort": 8080,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "environment": [
                    {
                        "name": "APP_DB_HOST",
                        "value": "supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com"
                    }
                ],
                "mountPoints": [],
                "volumesFrom": [],
                "logConfiguration": {
                    "logDriver": "awslogs",
                    "options": {
                        "awslogs-create-group": "true",
                        "awslogs-group": "awslogs-capstone",
                        "awslogs-region": "us-east-1",
                        "awslogs-stream-prefix": "awslogs-capstone"
                    }
                },
                "systemControls": []
            }
        ],
        "family": "employee-microservice",
        "executionRoleArn": "arn:aws:iam::385397943428:role/PipelineRole",
        "networkMode": "awsvpc",
        "revision": 8,
        "volumes": [],
        "status": "ACTIVE",
        "requiresAttributes": [
            {
                "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
            },
:
```



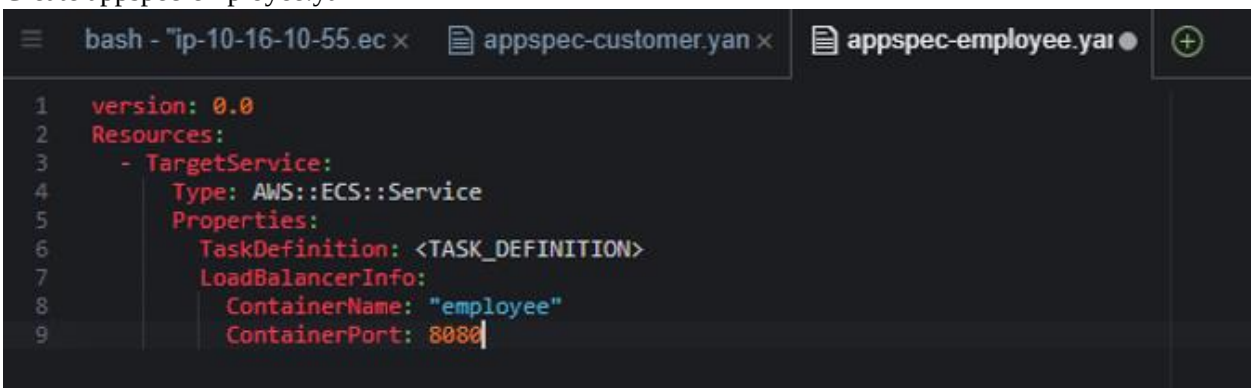


Task 5.5: Create AppSpec files for CodeDeploy for each microservice

- Create appspec-customer.yaml



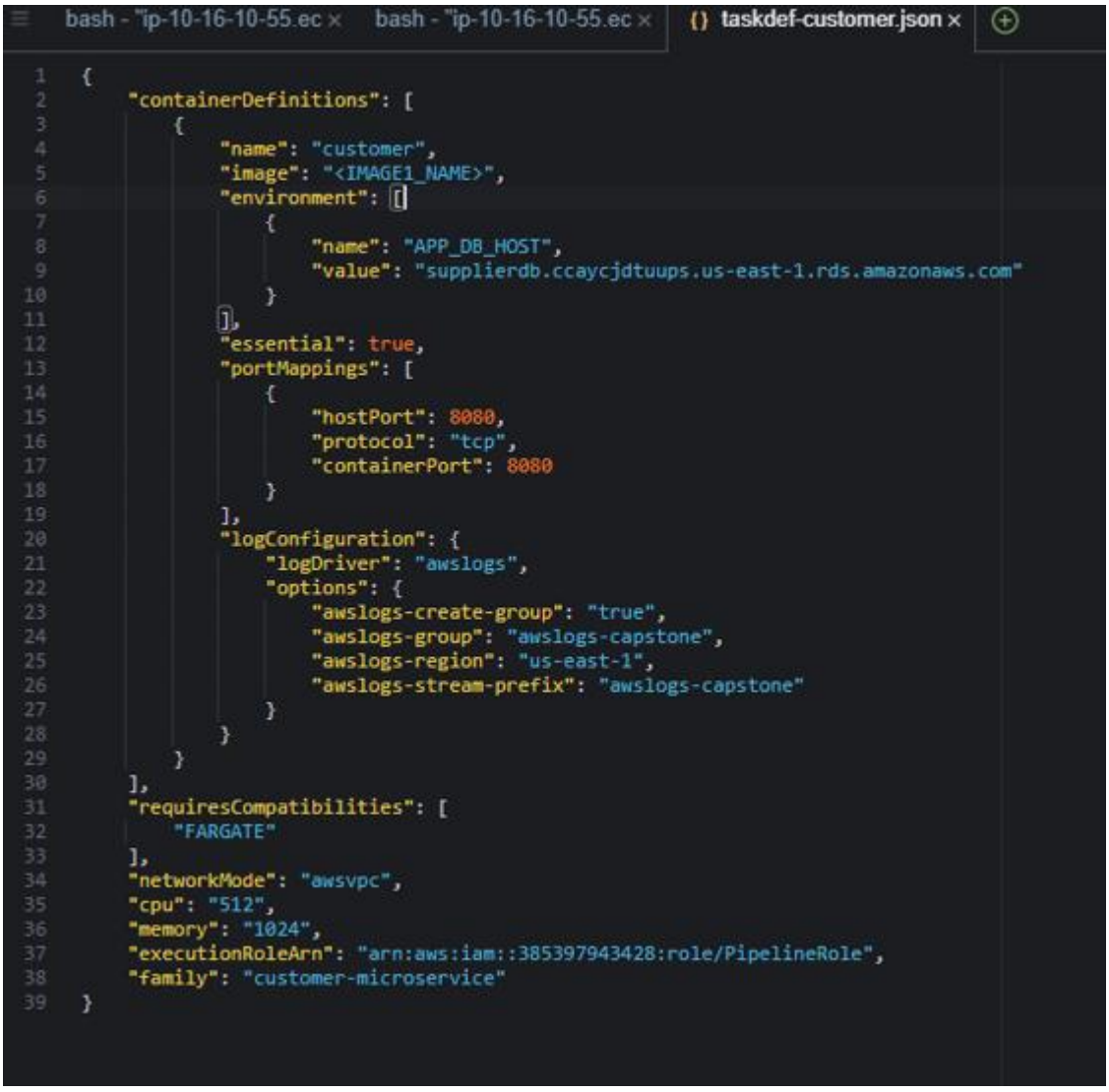
- Create appspec-employee.yaml



Task 5.6: Update files and check them into CodeCommit

- Edit the taskdef-customer.json file. Modify line 5 to match the following line: "image": "<IMAGE1_NAME>",

bash - "ip-10-16-10-55.ec × bash - "ip-10-16-10-55.ec × {} taskdef-customer.json ×



```
{
    "containerDefinitions": [
        {
            "name": "customer",
            "image": "<IMAGE1_NAME>",
            "environment": [
                {
                    "name": "APP_DB_HOST",
                    "value": "supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com"
                }
            ],
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 8080,
                    "protocol": "tcp",
                    "containerPort": 8080
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-capstone",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "awslogs-capstone"
                }
            }
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "512",
    "memory": "1024",
    "executionRoleArn": "arn:aws:iam::385397943428:role/PipelineRole",
    "family": "customer-microservice"
}
```

- Edit the taskdef-employee.json file. Modify line 5 to match the following line: "image": "<IMAGE1_NAME>",

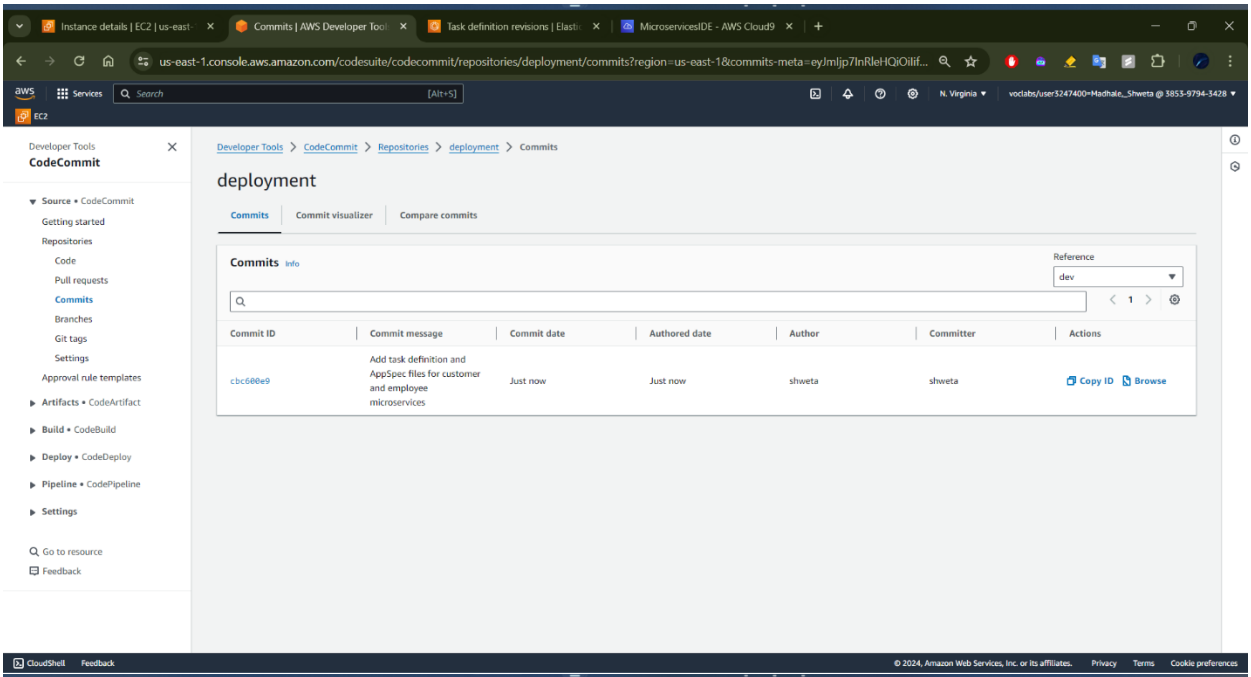
bash - "ip-10-16-10-55.ec × bash - "ip-10-16-10-55.ec × {} taskdef-customer.json × {} taskdef-employee.jsor ●

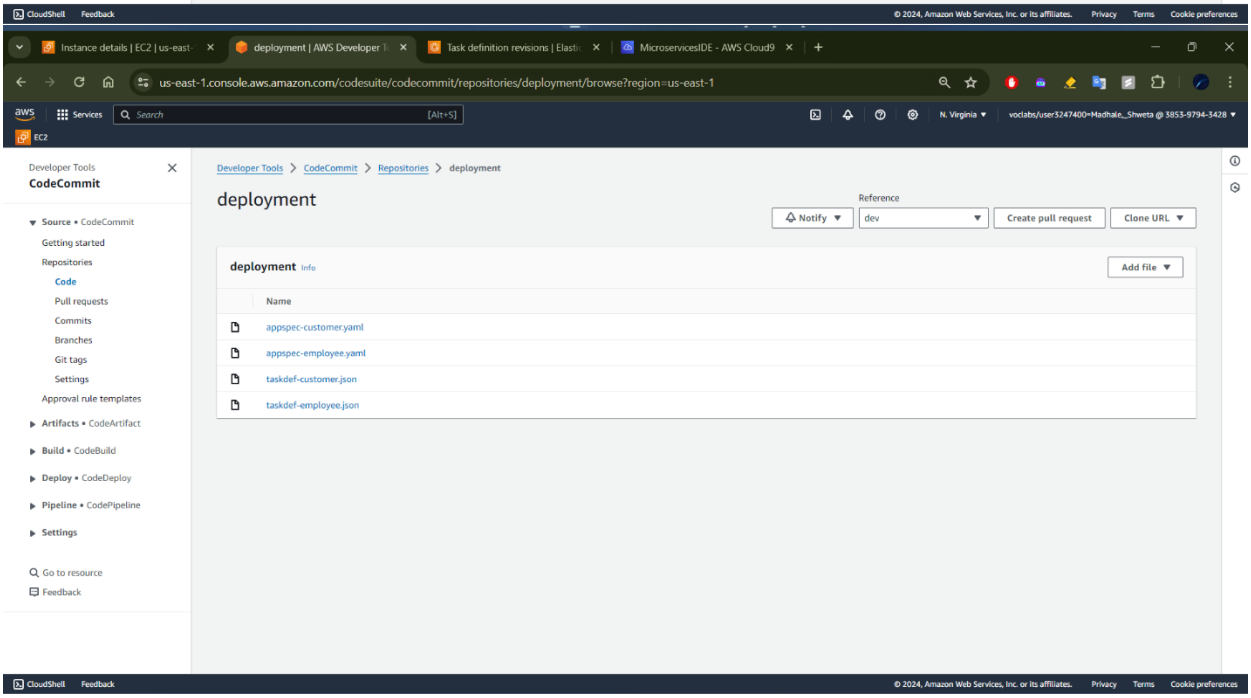
```
{
    "containerDefinitions": [
        {
            "name": "employee",
            "image": "<IMAGE1_NAME>",
            "environment": [
                {
                    "name": "APP_DB_HOST",
                    "value": "supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com"
                }
            ],
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 8080,
                    "protocol": "tcp",
                    "containerPort": 8080
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-capstone",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "awslogs-capstone"
                }
            }
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "512",
    "memory": "1024",
    "executionRoleArn": "arn:aws:iam::385397943428:role/PipelineRole",
    "family": "employee-microservice"
}
```

- Push the files.

bash - "ip-10-16-10-55.ec × git - "ip-10-16-10-55.ec2.i ×

```
voclabs:~/environment/deployment (dev) $ git add taskdef-customer.json taskdef-employee.json appspec-customer.yaml appspec-employee.yaml
voclabs:~/environment/deployment (dev) $ git commit -m "Add task definition and AppSpec files for customer and employee microservices"
[dev (root-commit) cbc600e] Add task definition and AppSpec files for customer and employee microservices
 4 files changed, 96 insertions(+)
 create mode 100644 appspec-customer.yaml
 create mode 100644 appspec-employee.yaml
 create mode 100644 taskdef-customer.json
 create mode 100644 taskdef-employee.json
voclabs:~/environment/deployment (dev) $ git push origin dev
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.07 KiB | 1.07 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
 * [new branch]      dev -> dev
voclabs:~/environment/deployment (dev) $
```



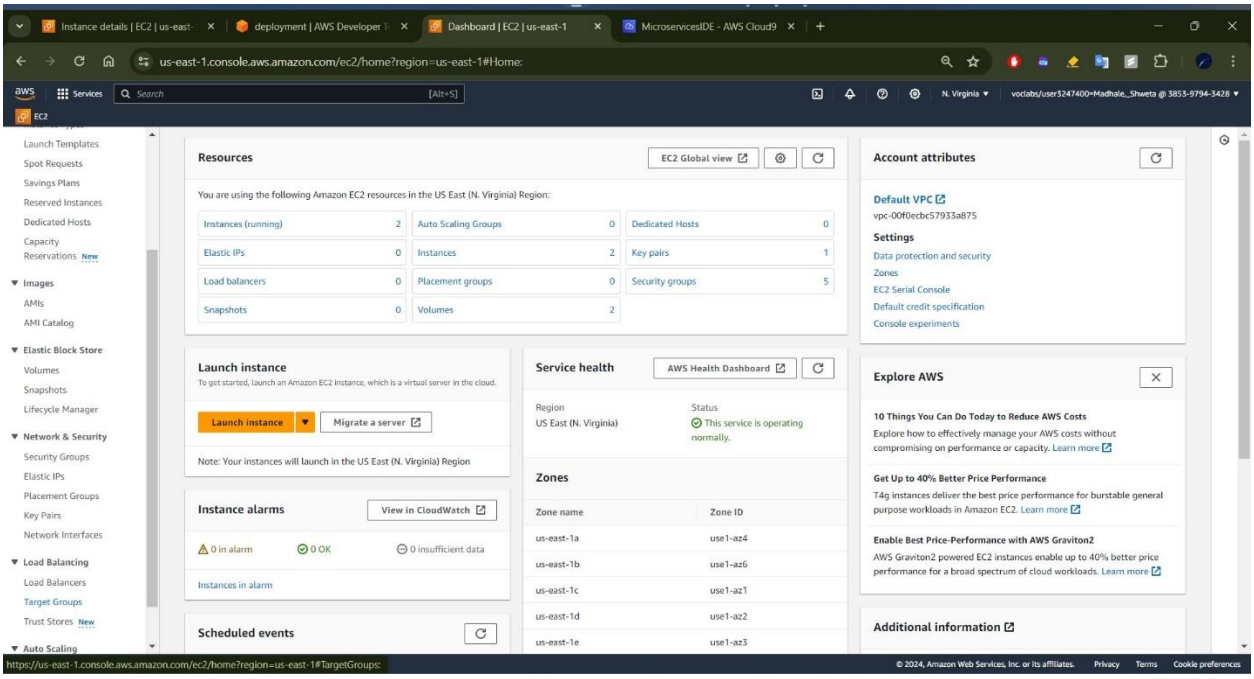


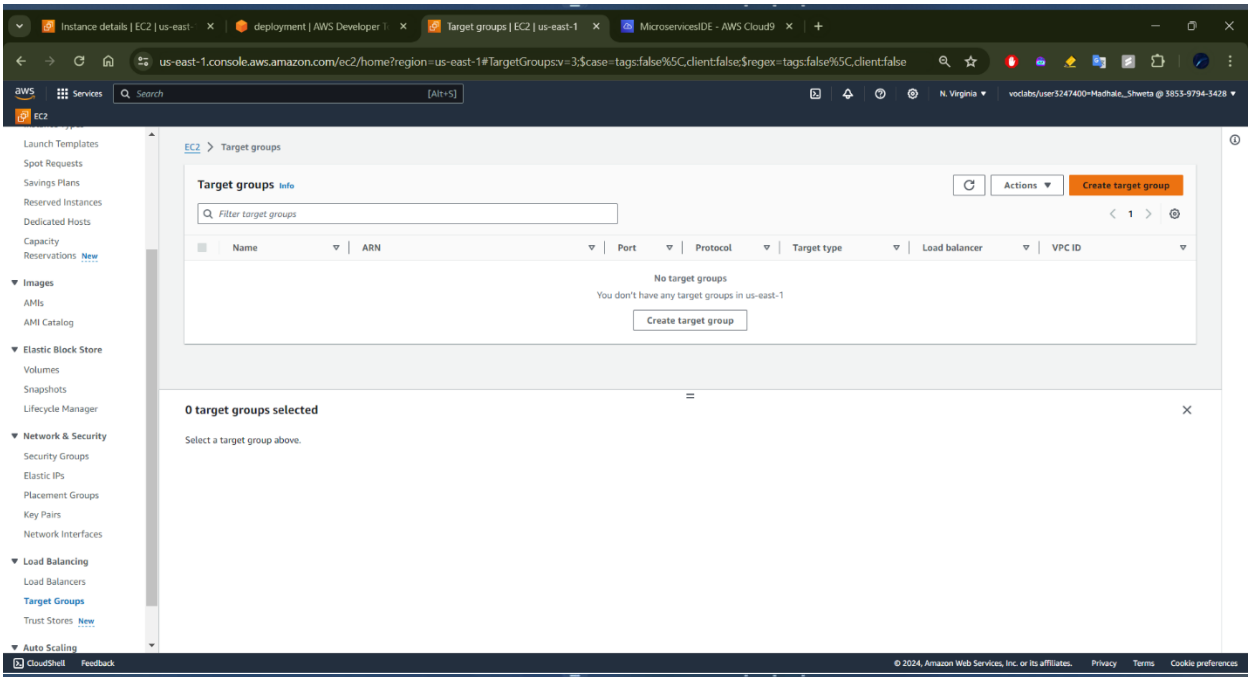
Phase 6 - Create target groups and an Application Load Balancer that routes web traffic to them.

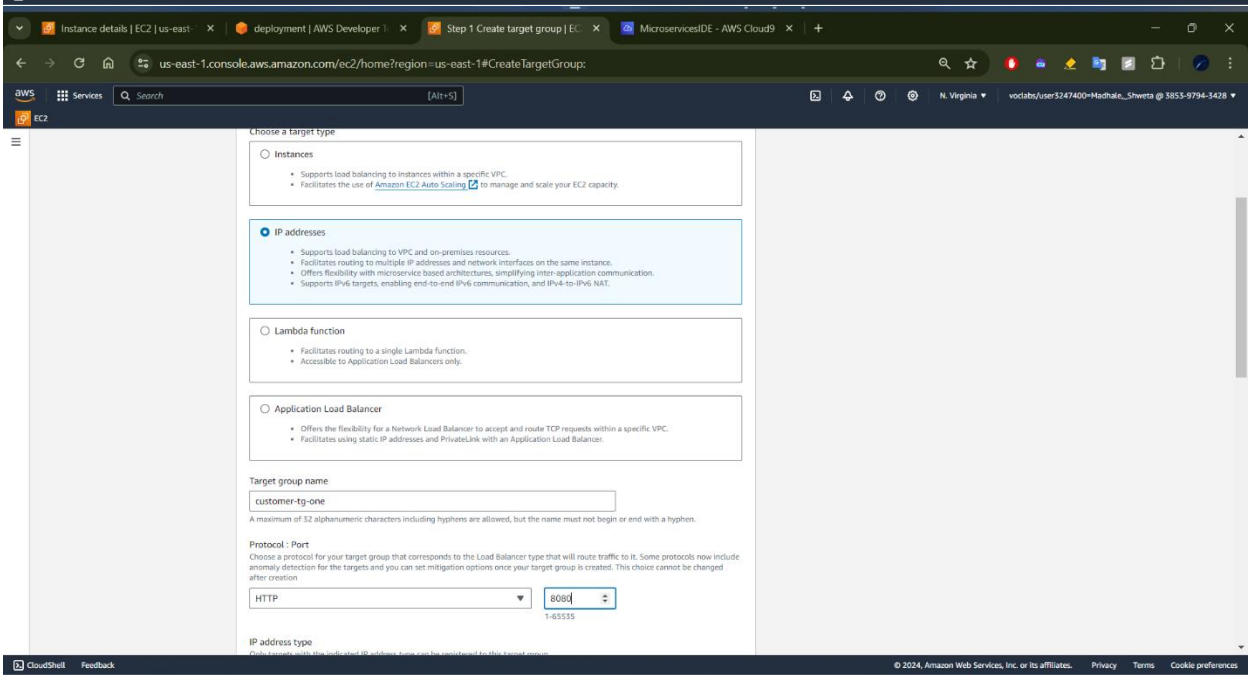
Task 6.1: Create four target groups

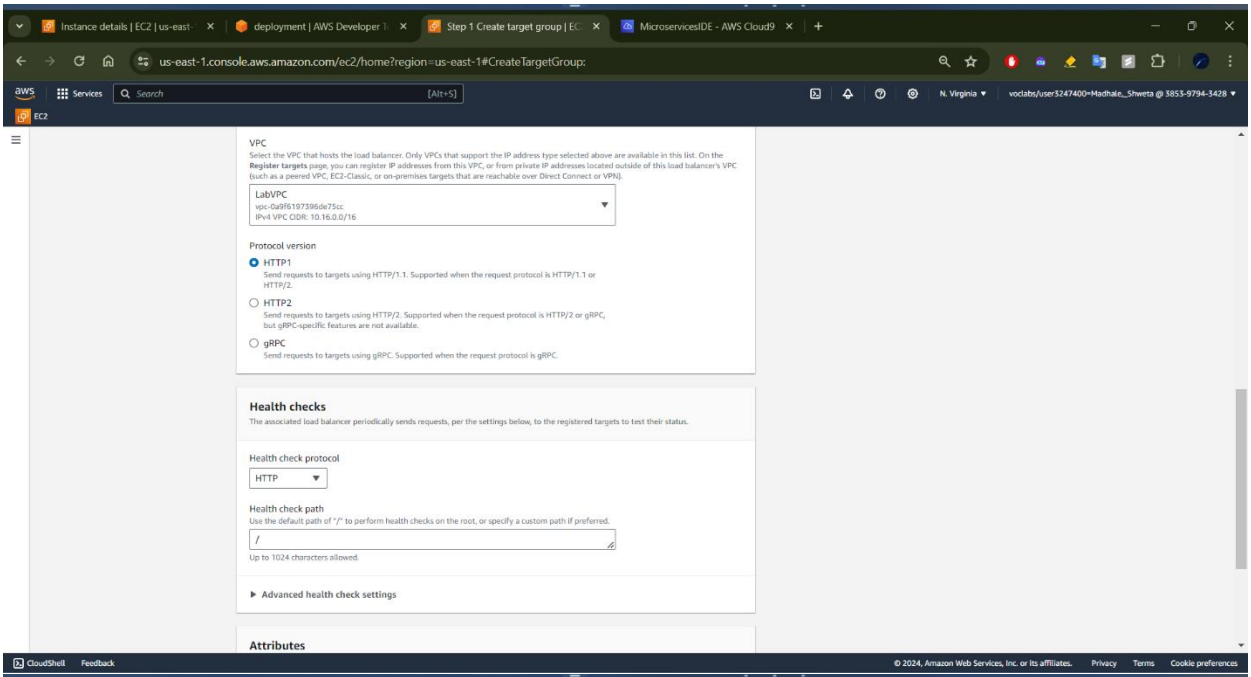
- Create target groups with

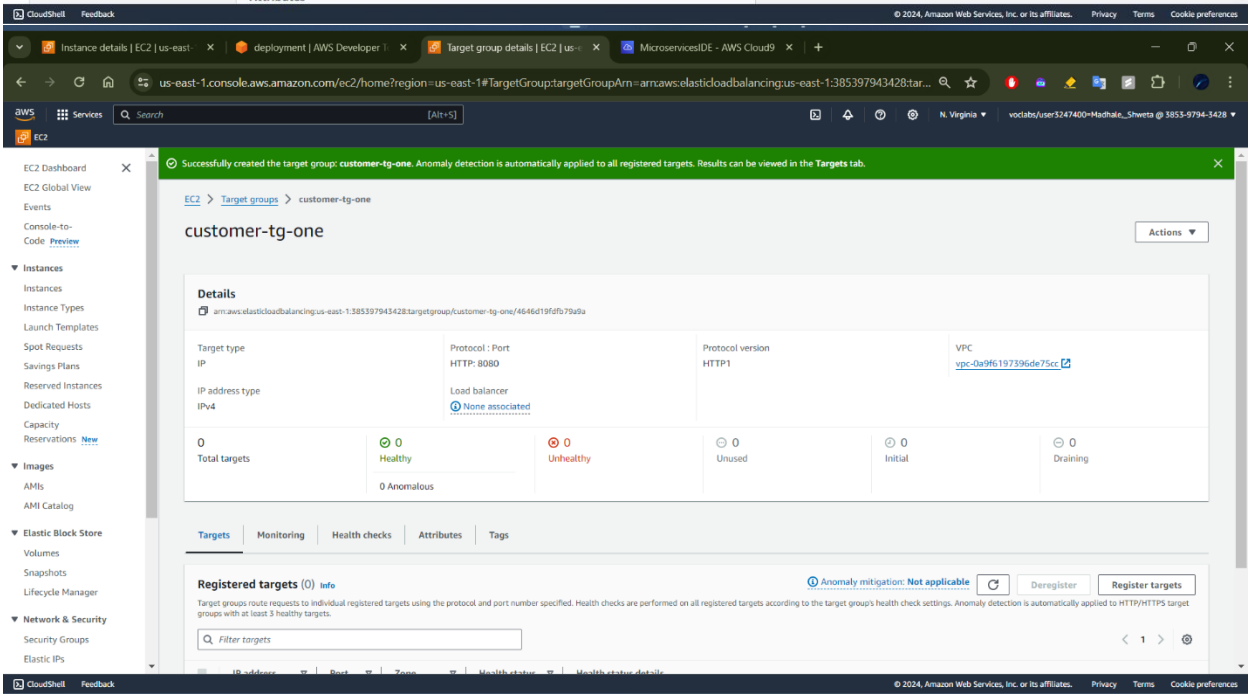
- Choose **Create target group** and configure the following:
  - **Choose a target type:** Choose **IP addresses**.
  - **Target group name:** Enter `customer-tg-one`
  - **Protocol:** Choose **HTTP**.
  - **Port:** Enter `8080`
  - **VPC:** Choose **LabVPC**.
  - **Health check path:** Enter `/`
- Choose **Next**.
- On the **Register targets** page, accept all defaults (don't register any targets), and choose **Create target group**.



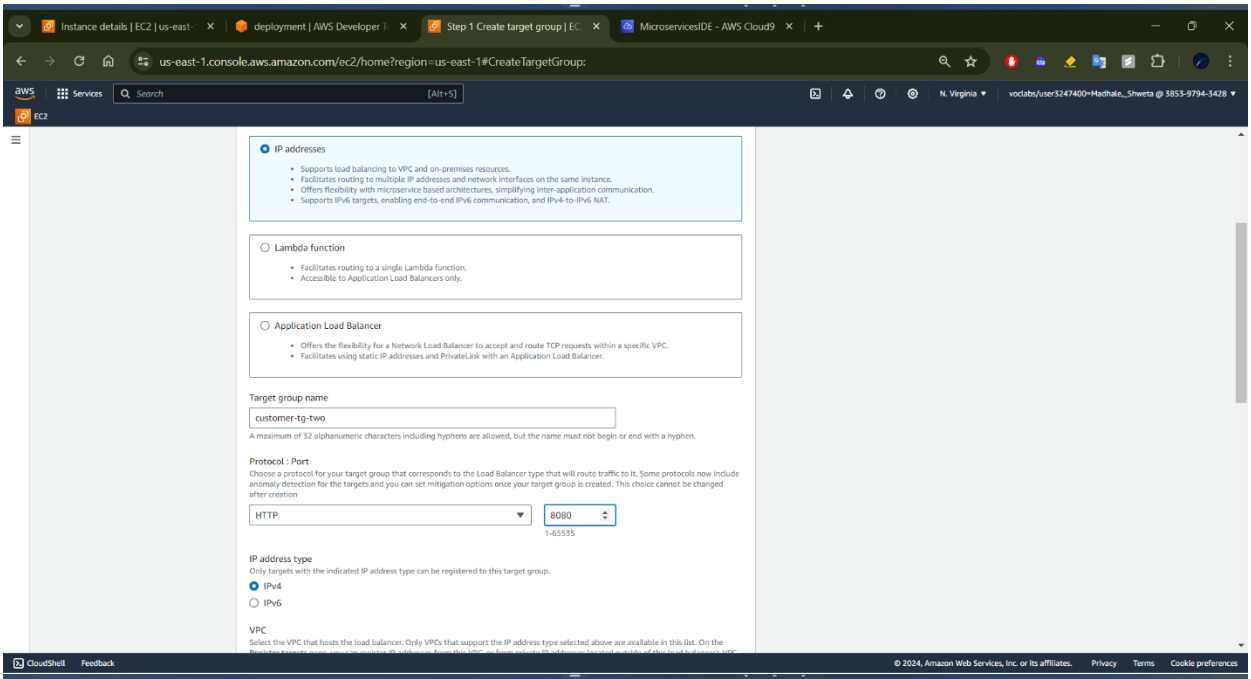

Instance details | EC2 | us-east-
deployment | AWS Developer
Target groups | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:v=3;$case=tags:false%5C,client:false;$regex=tags:false%5C,client:false
Services
Search
[Alt+S]
N. Virginia
EC2
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
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
EC2 > Target groups
Target groups Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
VPC ID
No target groups
You don't have any target groups in us-east-1
Create target group
0 target groups selected
Select a target group above.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

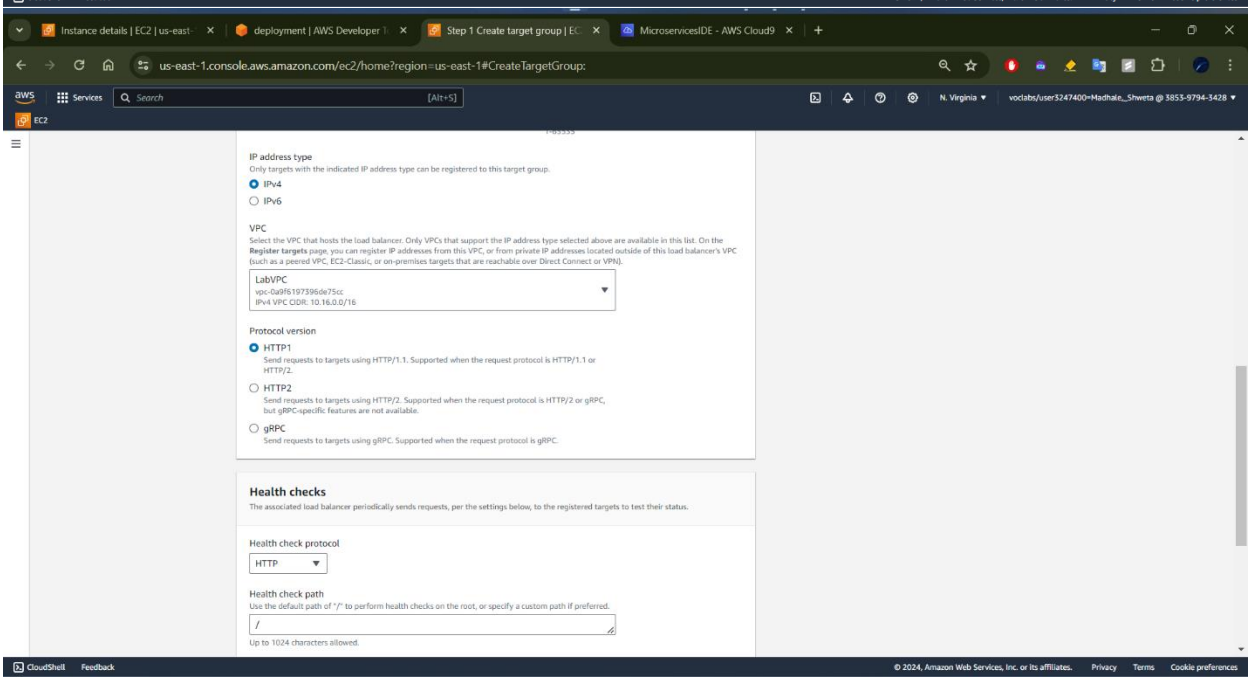

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 1 Create target group | EC
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:
Services
Search
[Alt+S]
N. Virginia
EC2
Choose a target type
Instances
Supports load balancing to instances 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.
Application Load Balancer
Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.
Target group name
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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

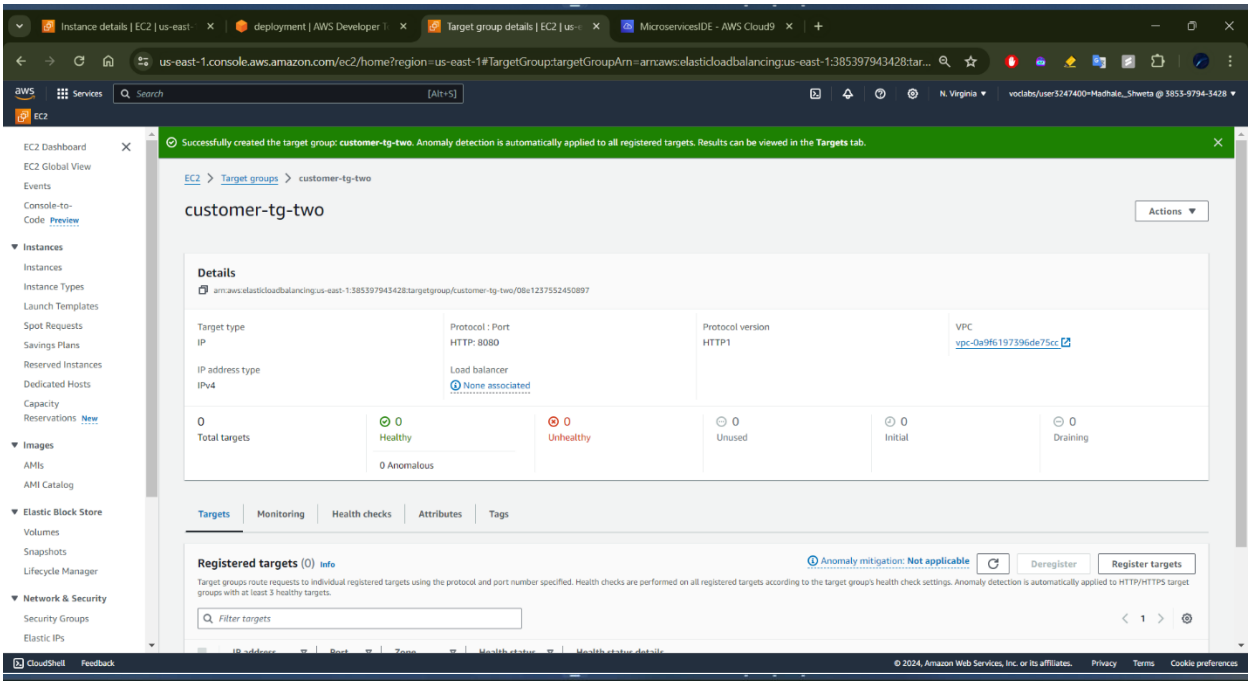


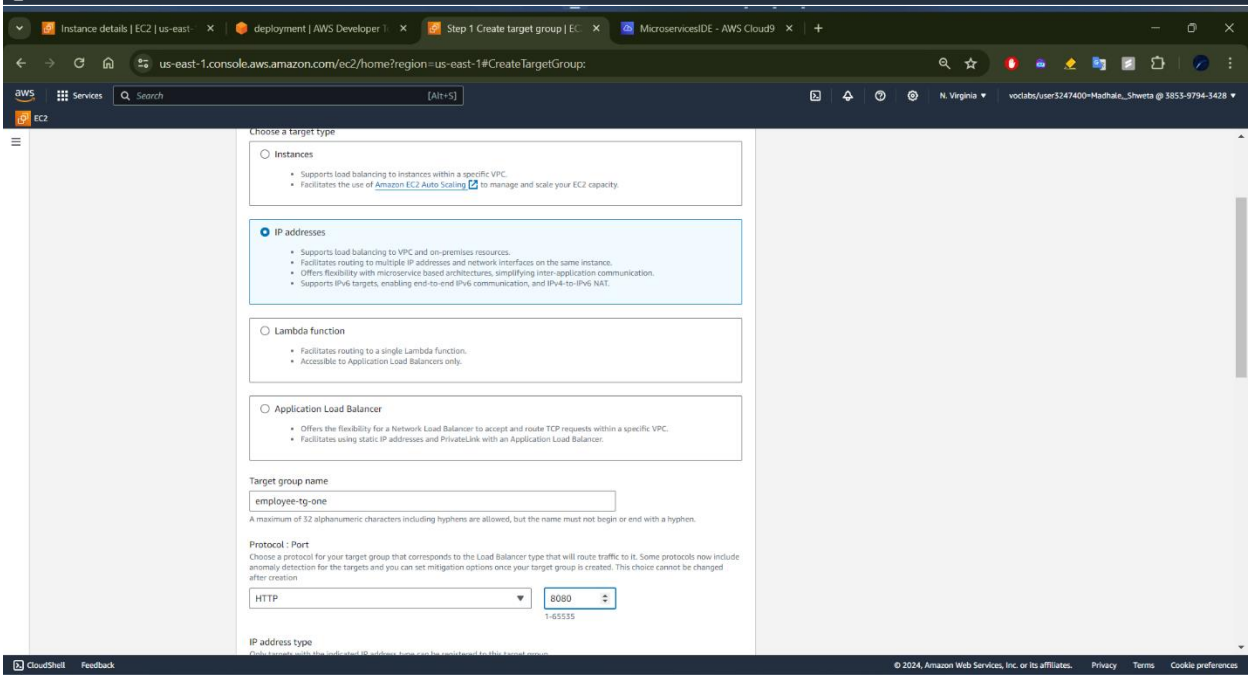


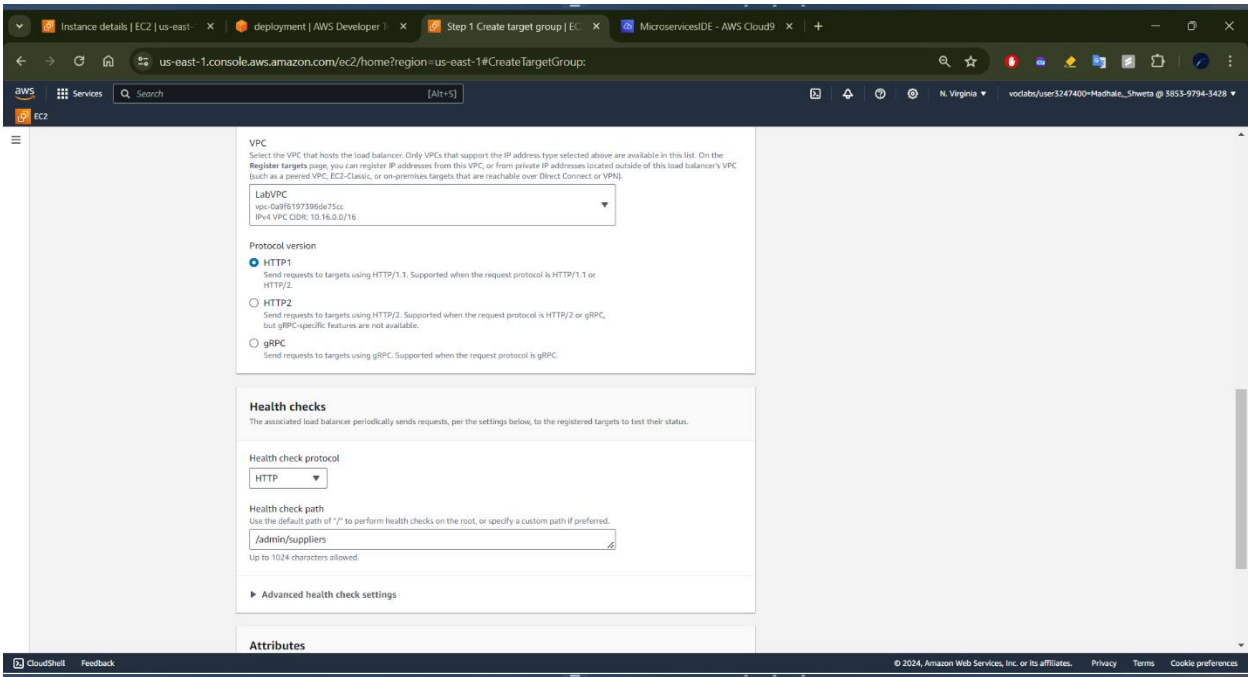
- Create similar groups customer-tg-two, employee-tg-one, employee-tg-two

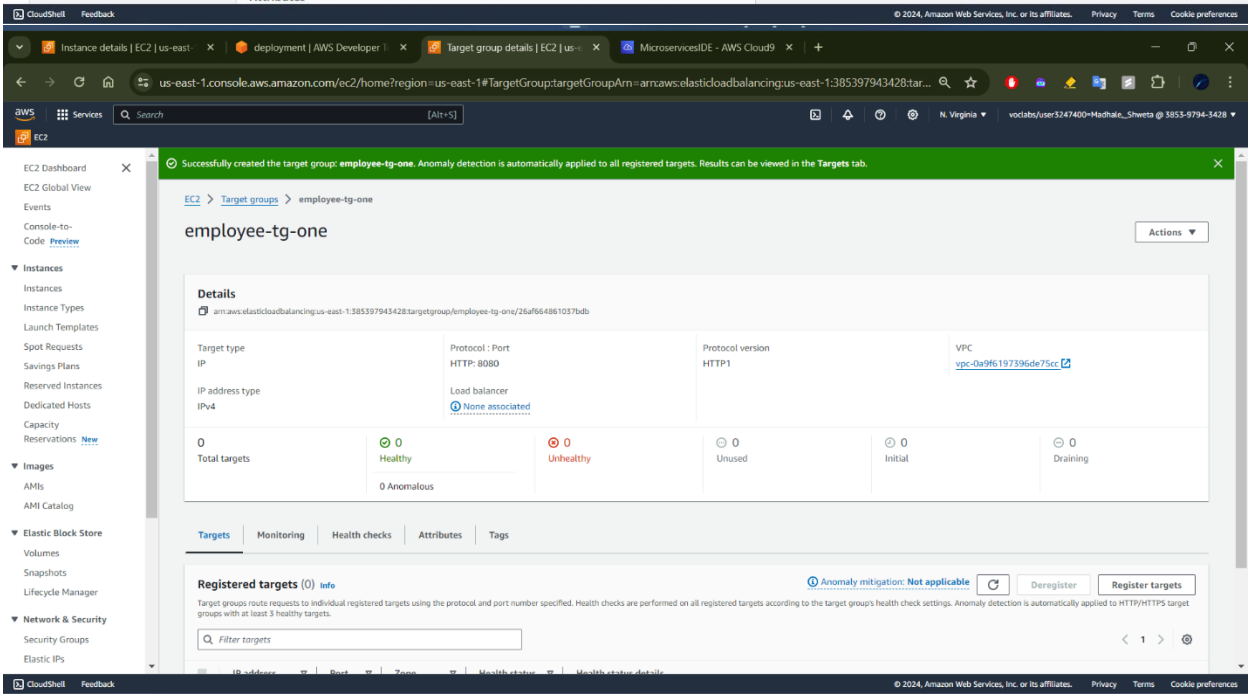

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 1 Create target group | EC
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
IP addresses
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.
Application Load Balancer
Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.
Target group name
customer-tg-two
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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

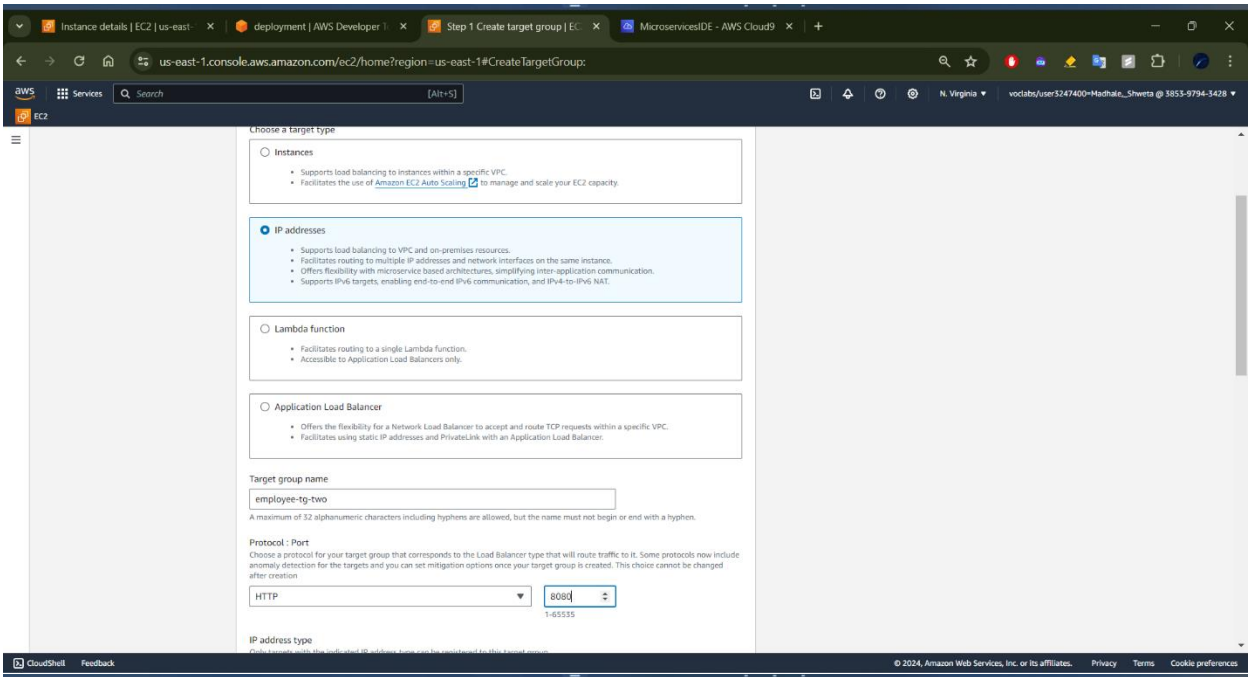

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 1 Create target group | EC
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
IP address 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-0a9f6197396de75cc
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, but gRPC-specific features are not available.
gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.
Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.
Health check protocol
HTTP
Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/
Up to 1024 characters allowed.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

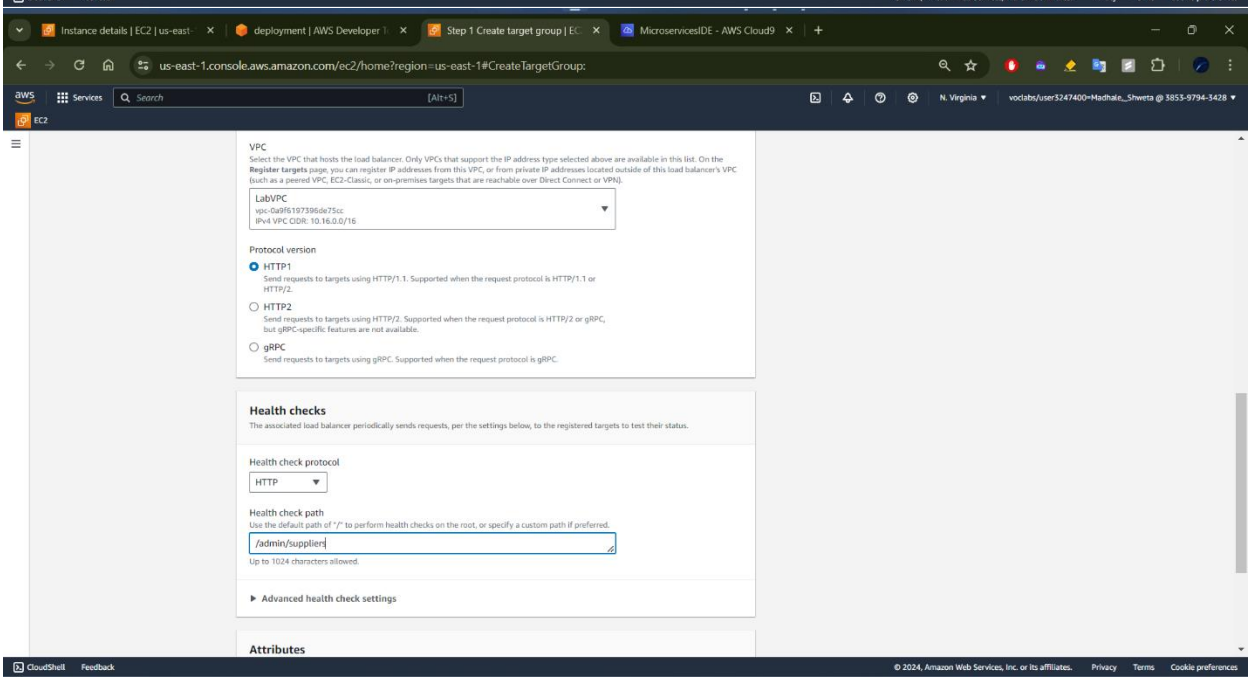

Instance details | EC2 | us-east-
deployment | AWS Developer
Target group details | EC2 | us-
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroup:targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:tar...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Successfully created the target group: customer-tg-two. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab.
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
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
EC2 > Target groups > customer-tg-two
customer-tg-two
Actions
Details
arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/customer-tg-two/08e1237552450897
Target type
IP
Protocol : Port
HTTP: 8080
Protocol version
HTTP1
VPC
vpc-0a9f6197396de75cc
IP address type
IPv4
Load balancer
None associated
0
Total targets
0
Healthy
0 Anomalous
0
Unhealthy
0
Unused
0
Initial
0
Draining
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (0) Info
Anomaly mitigation: Not applicable
Deregister
Register targets
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.
Filter targets
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

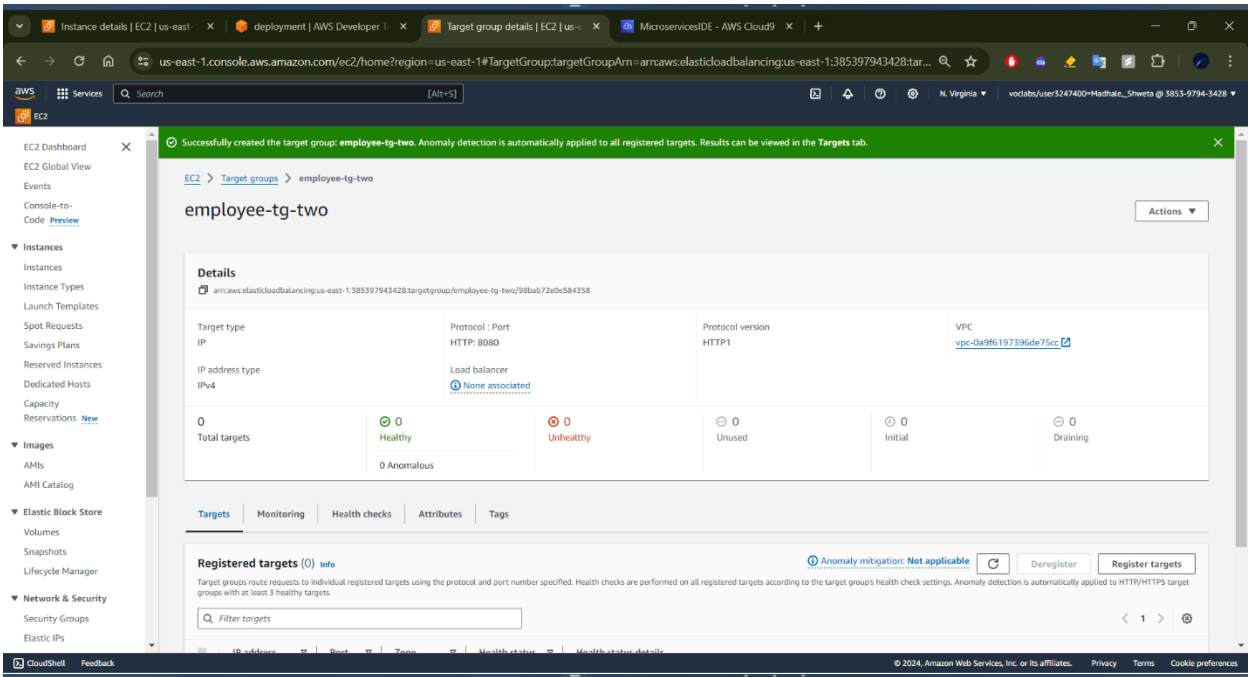

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 1 Create target group | EC
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Choose 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.
Application Load Balancer
Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.
Target group name
employee-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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

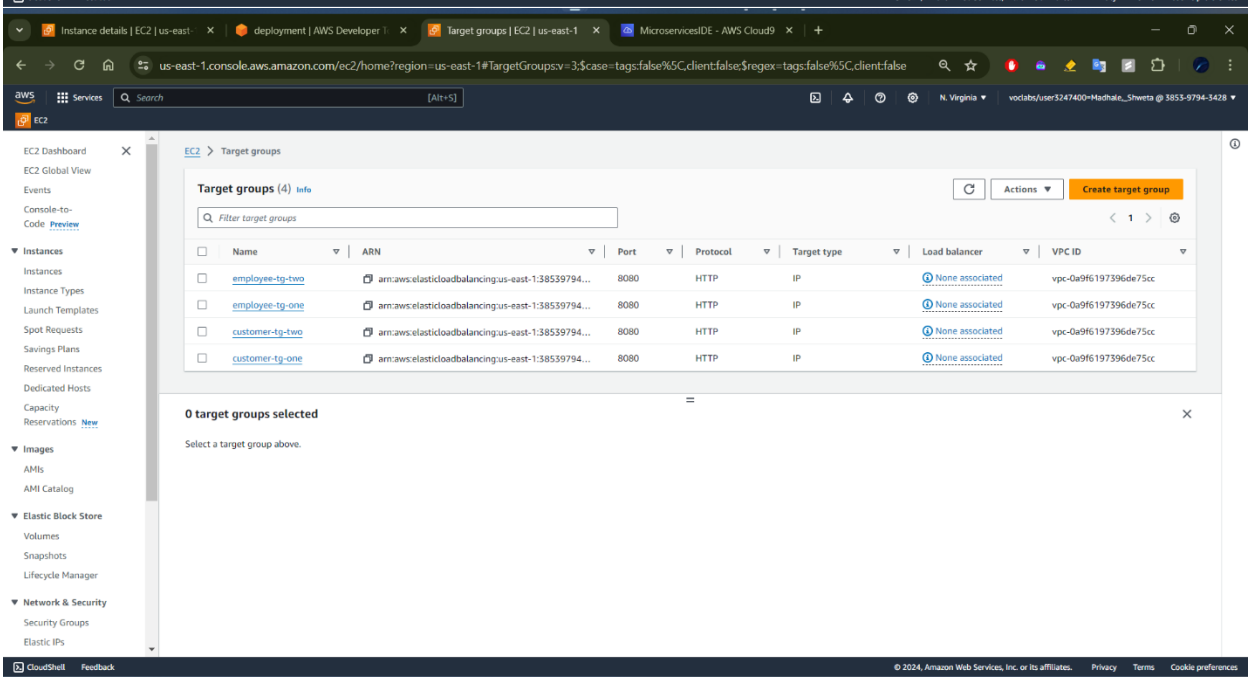

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 1 Create target group | EC
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
VPC
Select 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-0a9f6197396de75cc
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, but gRPC-specific features are not available.
gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.
Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.
Health check protocol
HTTP
Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/admin/suppliers
Up to 1024 characters allowed.
Advanced health check settings
Attributes
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
deployment | AWS Developer
Target group details | EC2 | us-
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroup:targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:tar...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Successfully created the target group: employee-tg-one. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab.
EC2 > Target groups > employee-tg-one
employee-tg-one
Actions
Details
arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/employee-tg-one/26af664861037bdb
Target type
IP
Protocol : Port
HTTP: 8080
Protocol version
HTTP1
VPC
vpc-0a9f6197396de75cc
IP address type
IPv4
Load balancer
None associated
0
Total targets
0
Healthy
0 Anomalous
0
Unhealthy
0
Unused
0
Initial
0
Draining
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (0) Info
Anomaly mitigation: Not applicable
Deregister
Register targets
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.
Filter targets
1
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
deployment | AWS Developer
Step 1 Create target group | EC
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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.
Application Load Balancer
Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.
Target group name
employee-tg-two
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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

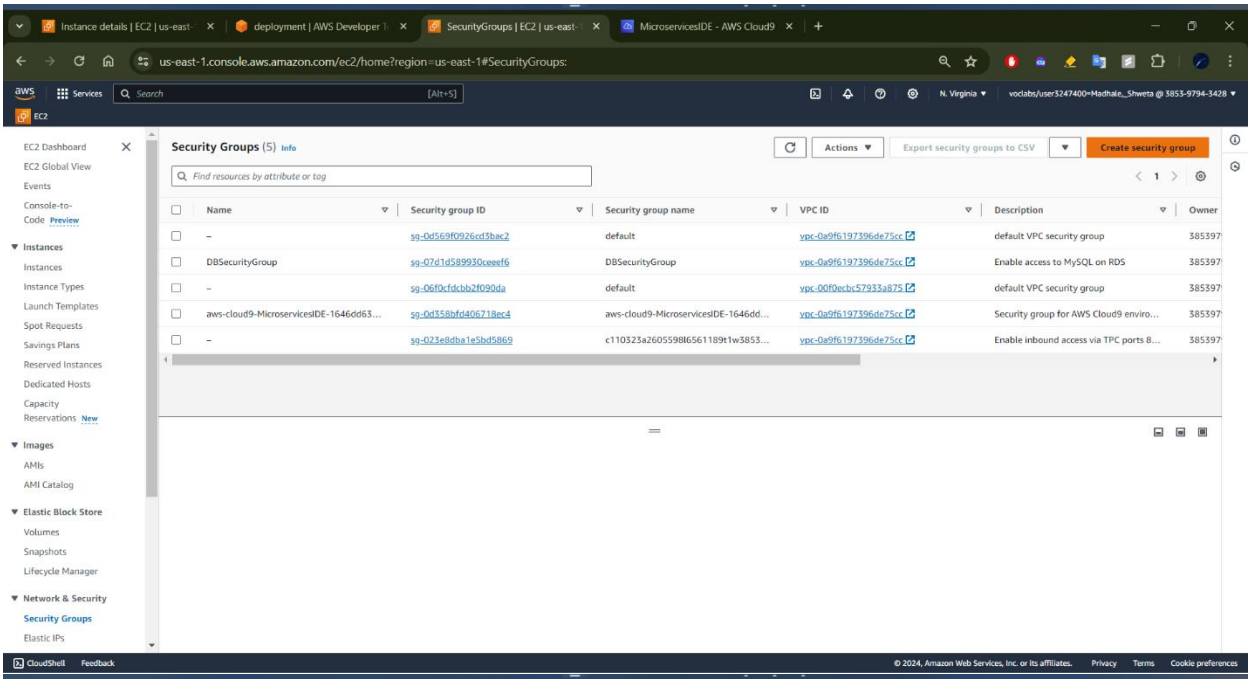

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 1 Create target group | EC
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
VPC
Select 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-0a9ff6197396de75cc
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, but gRPC-specific features are not available.
gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.
Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.
Health check protocol
HTTP
Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/admin/supplier
Up to 1024 characters allowed.
Advanced health check settings
Attributes
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

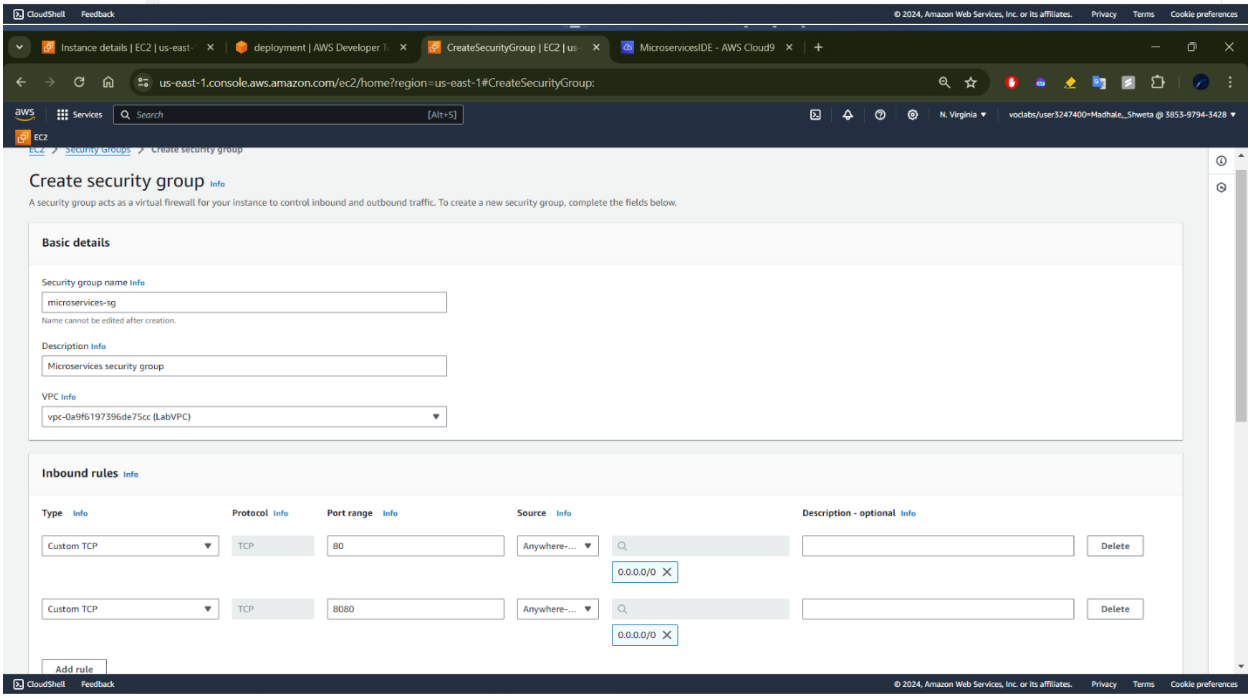


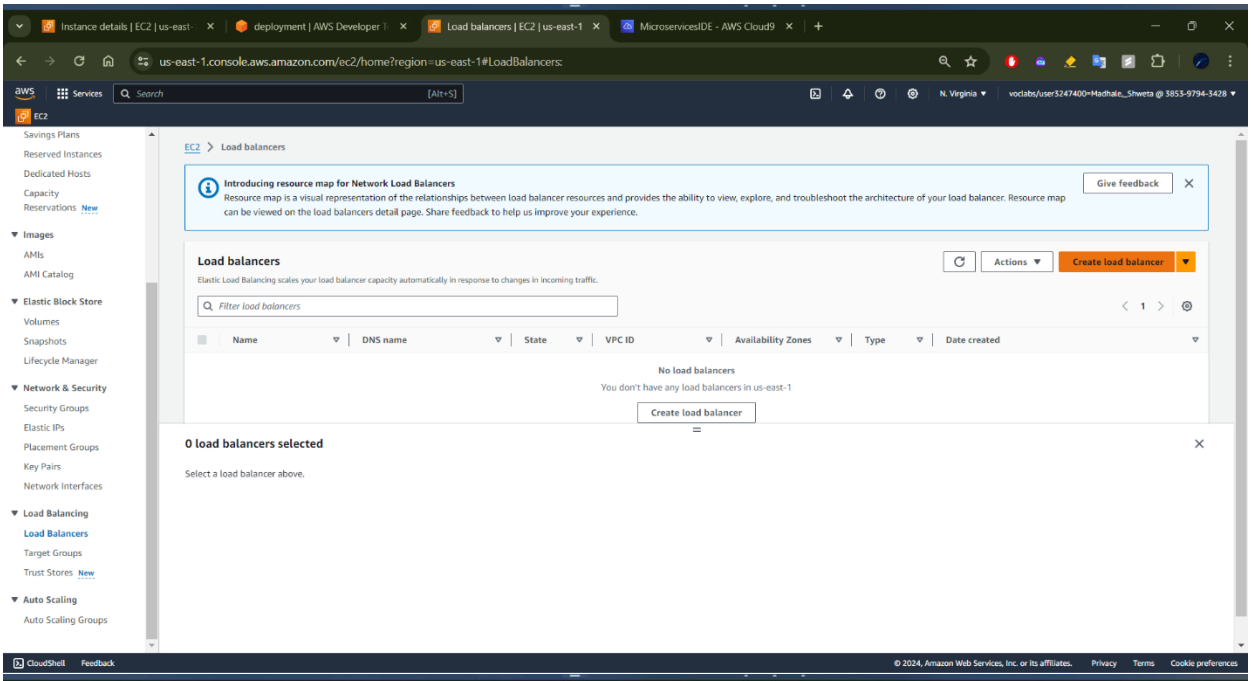


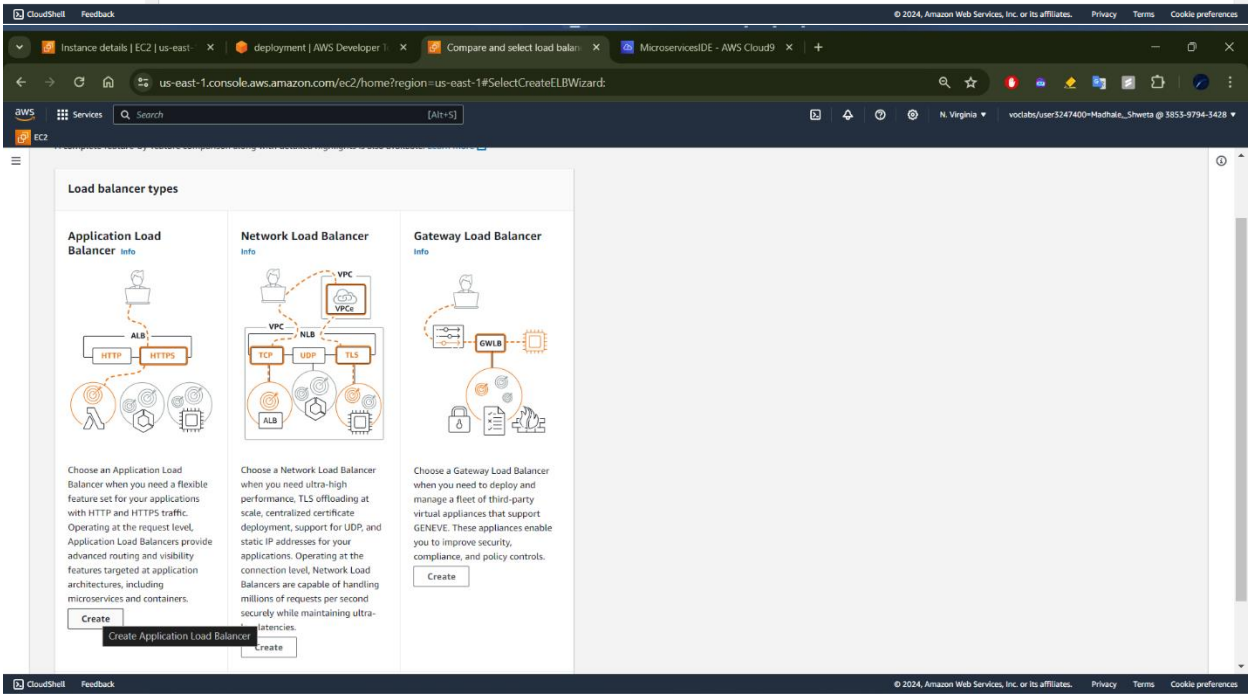
Task 6.2: Create a security group and an Application Load Balancer, and configure rules to route traffic

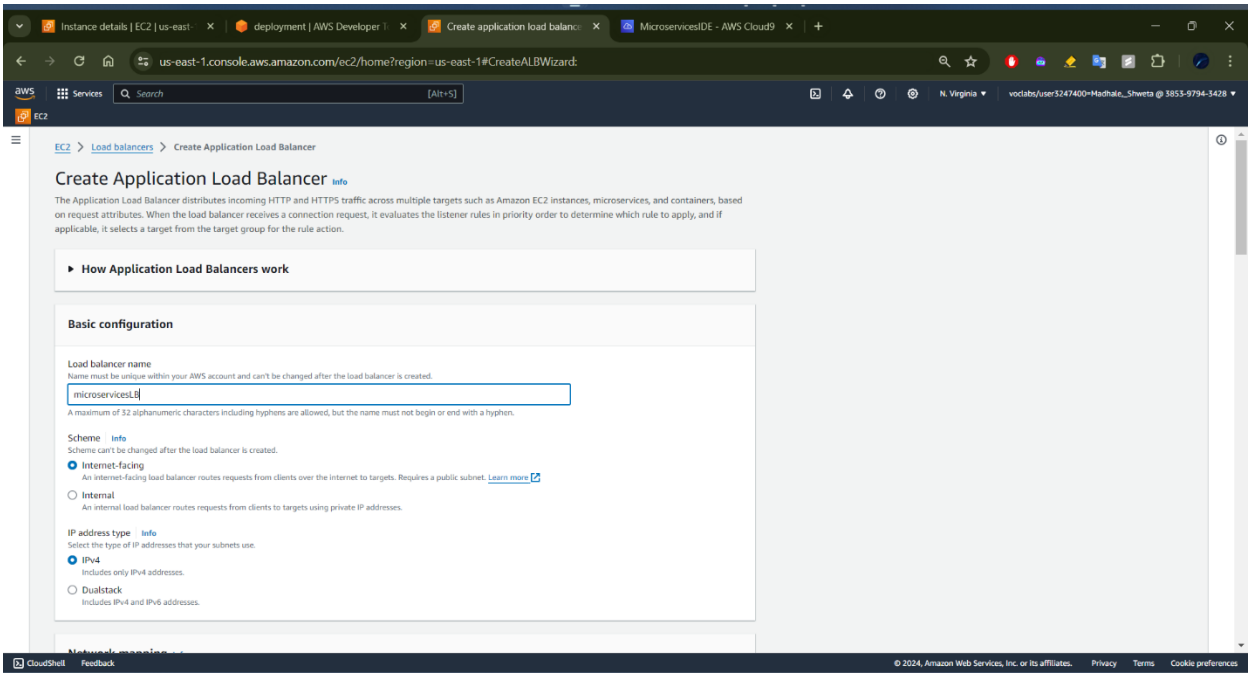
- Create a new EC2 security group named microservices-sg to use in LabVPC. Add inbound rules that allow TCP traffic from any IPv4 address on ports 80 and 8080. In the Amazon EC2 console, create an Application Load Balancer named microservicesLB. Make it internet facing for IPv4 addresses. Use LabVPC, Public Subnet1, Public Subnet2, and the microservices-sg security group. Configure two listeners on it. The first should listen on HTTP:80 and forward traffic to customer-tg-two by default. The second should listen on HTTP:8080 and forward traffic to customer-tg-one by default.

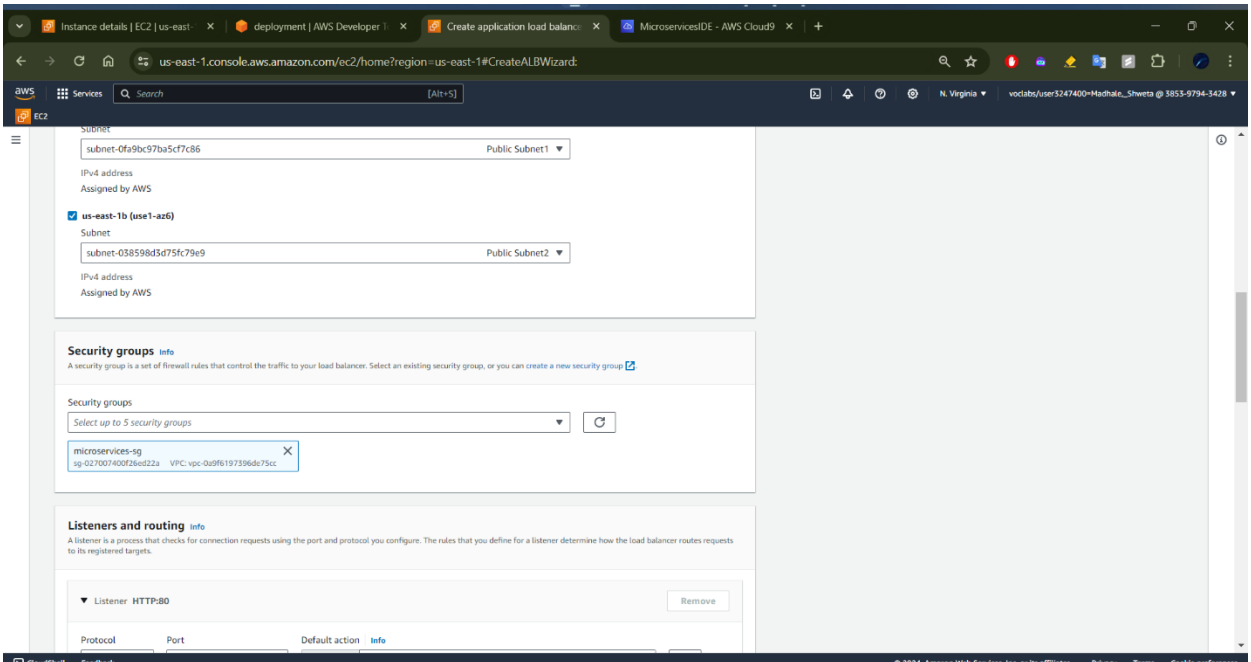

Instance details | EC2 | us-east-
deployment | AWS Developer
SecurityGroups | EC2 | us-east-
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroups:
Services
Search
[Alt+S]
N. Virginia
EC2
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Security Groups (5) Info
Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag
Name
Security group ID
Security group name
VPC ID
Description
Owner
sg-0d569f0926cd3bac2
default
vpc-0a9f6197396de75cc
default VPC security group
sg-07d1d589930ceeef6
DBSecurityGroup
Enable access to MySQL on RDS
sg-06f0cfdcbb2f090da
vpc-00f0ecbc57933a875
sg-0d358bfd406718ec4
aws-cloud9-MicroservicesIDE-1646dd...
Security group for AWS Cloud9 enviro...
sg-023e8dba1e5bd5869
Enable inbound access via TPC ports 8...
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

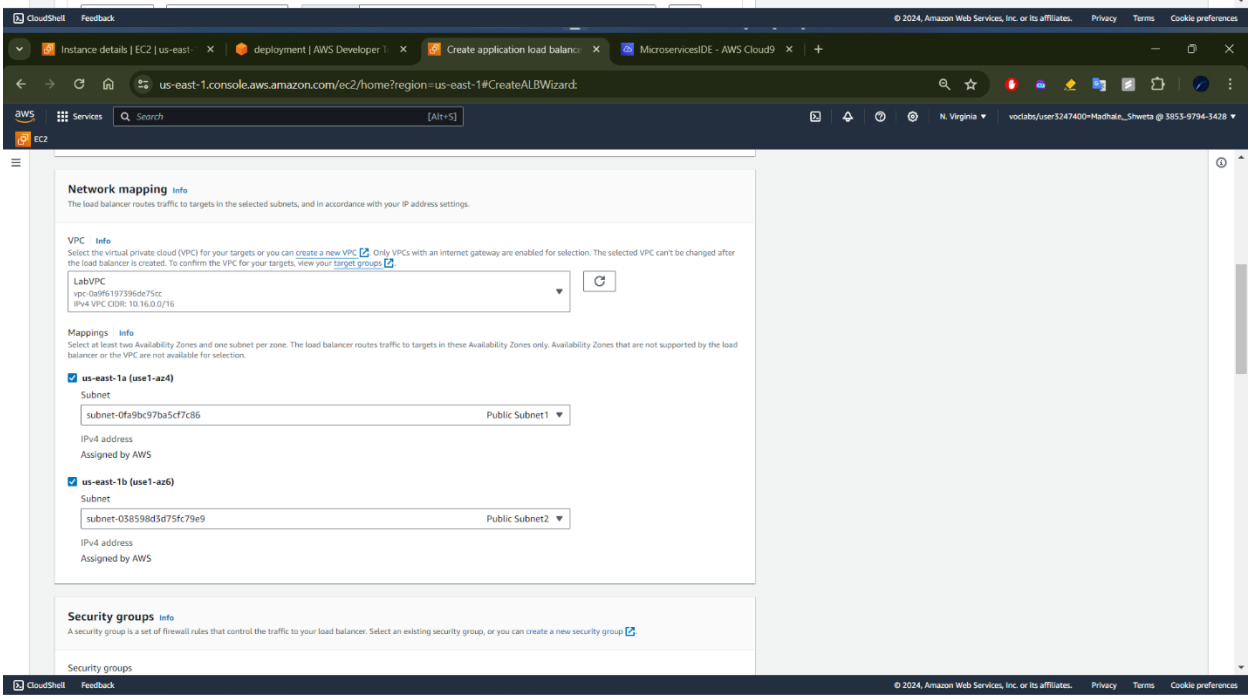

Instance details | EC2 | us-east-
deployment | AWS Developer
CreateSecurityGroup | EC2 | us-
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateSecurityGroup:
Services
Search
[Alt+S]
N. Virginia
EC2
EC2 > Security Groups > Create security group
Create security group Info
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.
Basic details
Security group name Info
microservices-sg
Name cannot be edited after creation.
Description Info
Microservices security group
VPC Info
vpc-0a9f6197396de75cc (LabVPC)
Inbound rules Info
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
Custom TCP
TCP
80
Anywhere-...
0.0.0.0/0
Delete
Custom TCP
TCP
8080
Anywhere-...
0.0.0.0/0
Delete
Add rule
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

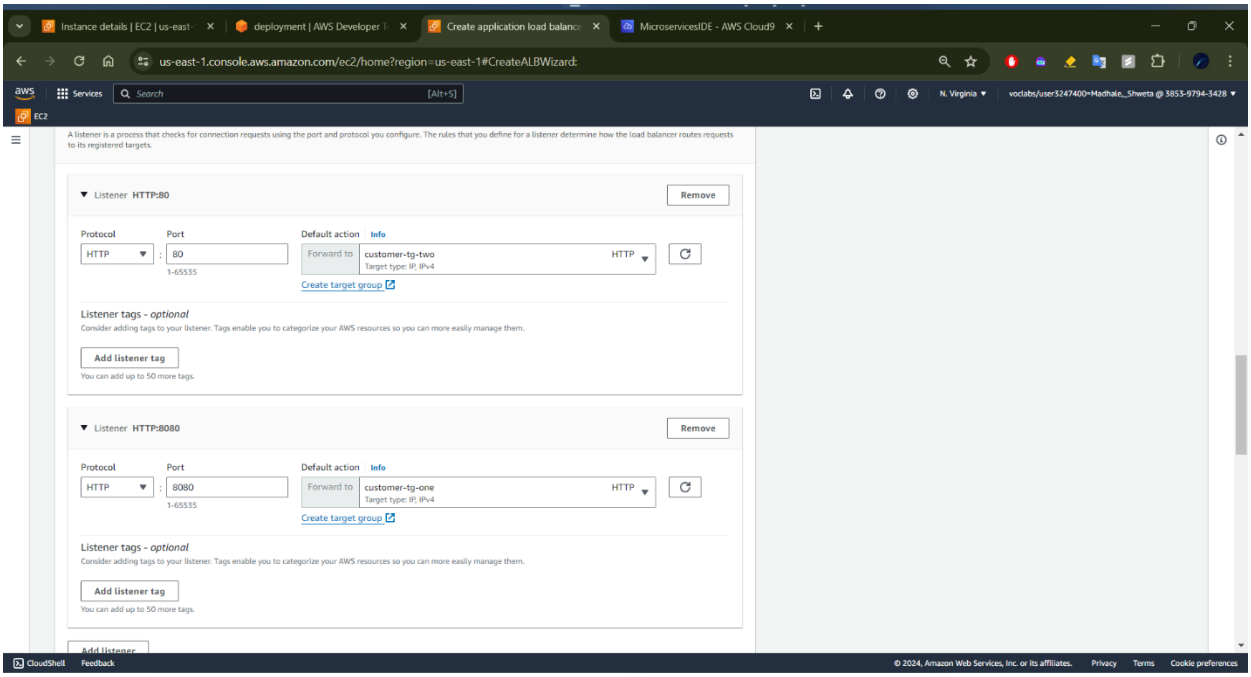

Instance details | EC2 | us-east-
deployment | AWS Developer
Load balancers | EC2 | us-east-1
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:
Services
Search
[Alt+S]
N. Virginia
EC2
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
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
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
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
Date created
No load balancers
You don't have any load balancers in us-east-1
Create load balancer
0 load balancers selected
Select a load balancer above.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
deployment | AWS Developer
Compare and select load balan
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SelectCreateELBWizard:
Services
Search
[Alt+S]
N. Virginia
EC2
Load balancer types
Application Load Balancer Info
ALB
HTTP
HTTPS
Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.
Create
Create Application Load Balancer
Network Load Balancer Info
VPC
VPCs
NLB
TCP
UDP
TLS
ALB
Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-
latencies.
Create
Gateway Load Balancer Info
GWLB
Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.
Create
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

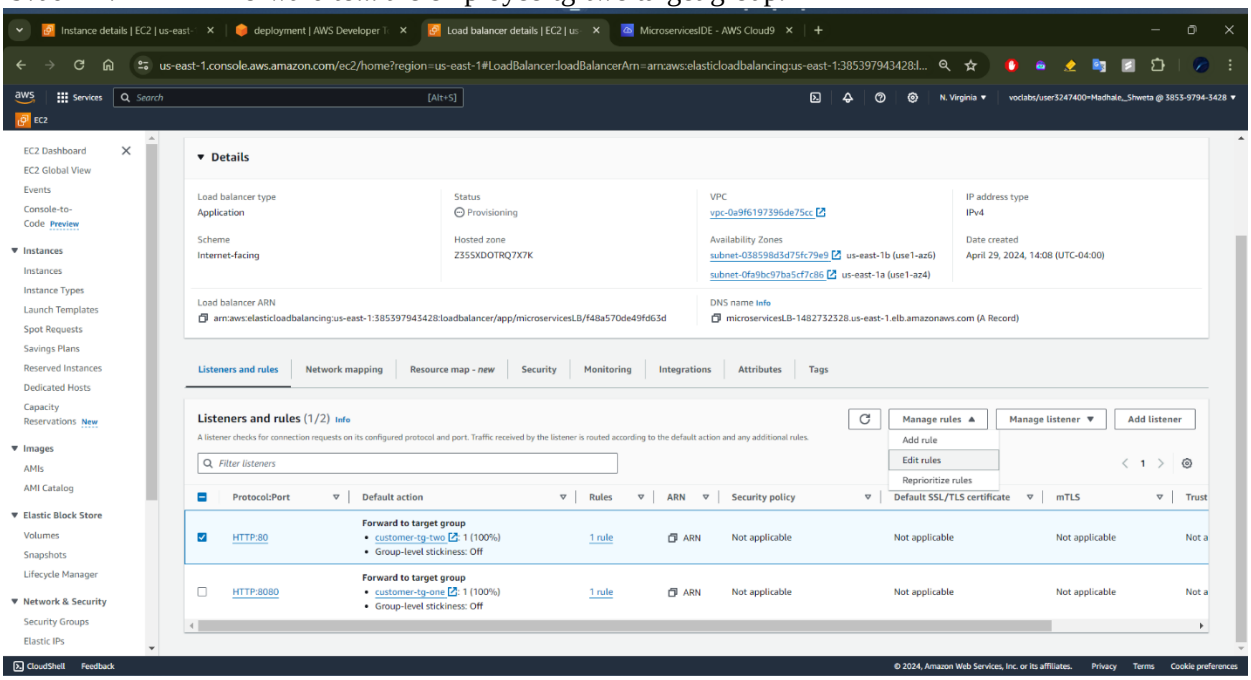

Instance details | EC2 | us-east-
deployment | AWS Developer
Create application load balanc
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateALBWizard:
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
EC2 > Load balancers > Create Application Load Balancer
Create Application Load Balancer Info
The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.
How Application Load Balancers work
Basic configuration
Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
microservicesLB
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.
Scheme Info
Scheme can't be changed after the load balancer is created.
Internet-facing
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more
Internal
An internal load balancer routes requests from clients to targets using private IP addresses.
IP address type Info
Select the type of IP addresses that your subnets use.
IPv4
Includes only IPv4 addresses.
Dualstack
Includes IPv4 and IPv6 addresses.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

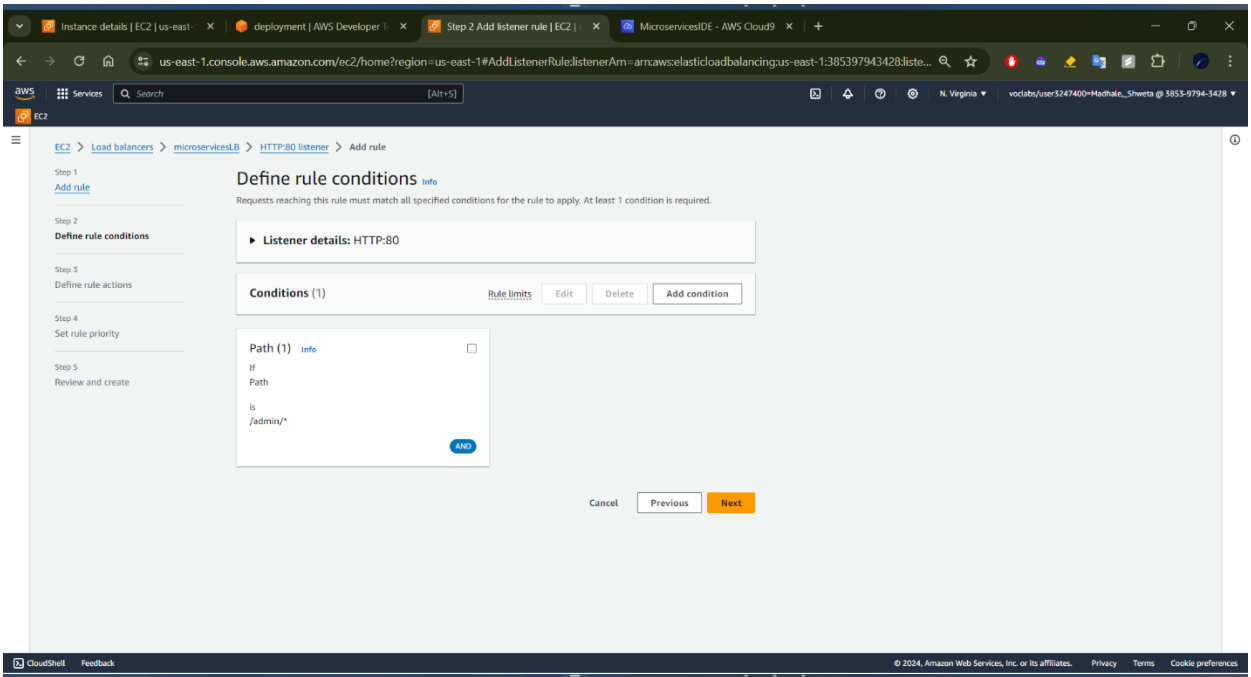

Instance details | EC2 | us-east-
deployment | AWS Developer
Create application load balanc
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateALBWizard:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Subnet
subnet-0fa9bc97ba5cf7c86
Public Subnet1
IPv4 address
Assigned by AWS
us-east-1b (use1-az6)
Subnet
subnet-038598d3d75fc79e9
Public Subnet2
IPv4 address
Assigned by AWS
Security groups Info
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group
Security groups
Select up to 5 security groups
microservices-sg
sg-02700740f26ed22a VPC: vpc-0a9f6197396de75cc
Listeners and routing Info
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.
Listener HTTP:80
Remove
Protocol
Port
Default action Info
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

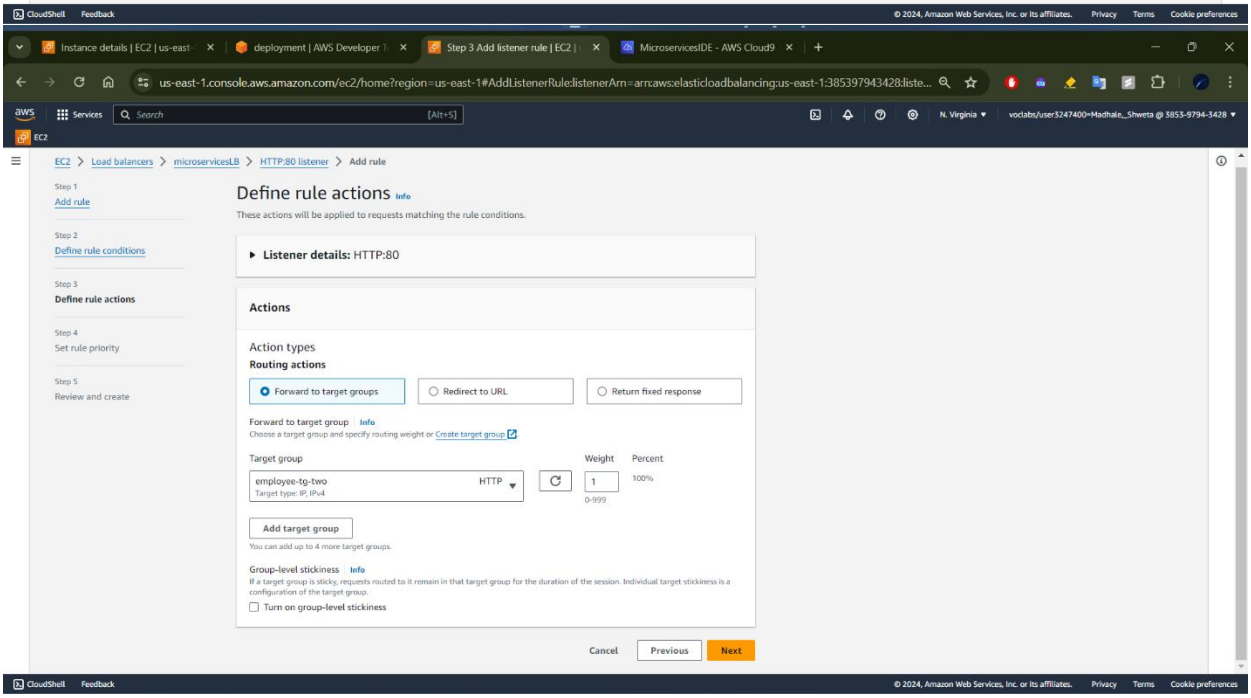

Instance details | EC2 | us-east-
deployment | AWS Developer
Create application load balanc
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateALBWizard:
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Network mapping Info
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.
VPC Info
Select the virtual private cloud (VPC) for your targets or you can create a new VPC. Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your target groups
LabVPC
vpc-0a9f6197396de75cc
IPv4 VPC CIDR: 10.16.0.0/16
Mappings Info
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.
us-east-1a (use1-az4)
Subnet
subnet-0fa9bc97ba5cf7c86
Public Subnet1
IPv4 address
Assigned by AWS
us-east-1b (use1-az6)
Subnet
subnet-038598d3d75fc79e9
Public Subnet2
IPv4 address
Assigned by AWS
Security groups Info
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group
Security groups
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

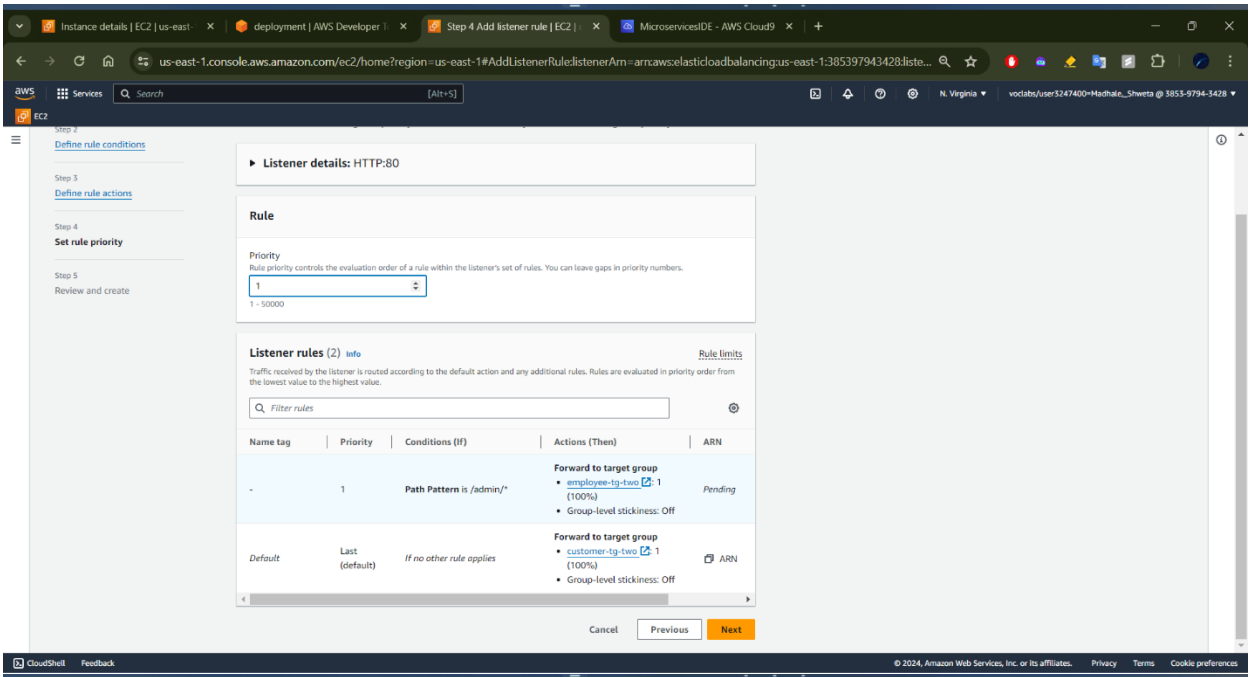


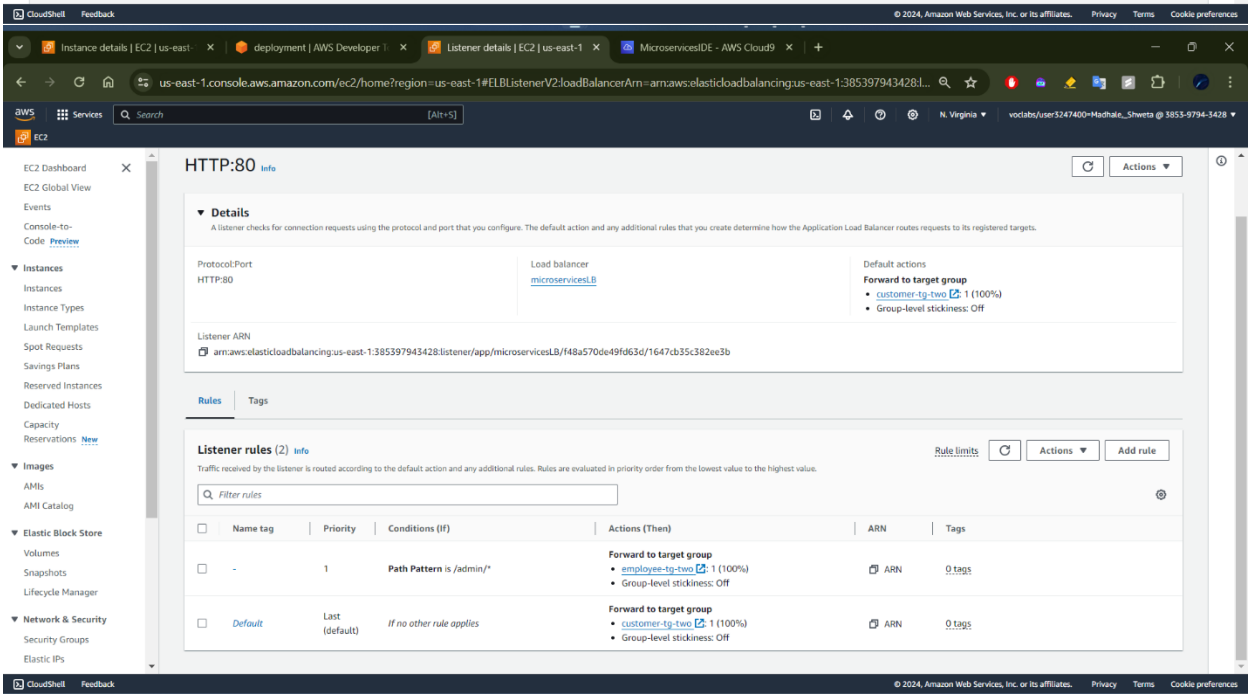
- Add a second rule for the HTTP:80 listener. Define the following logic for this new rule: IF Path is /admin/* THEN Forward to... the employee-tg-two target group.



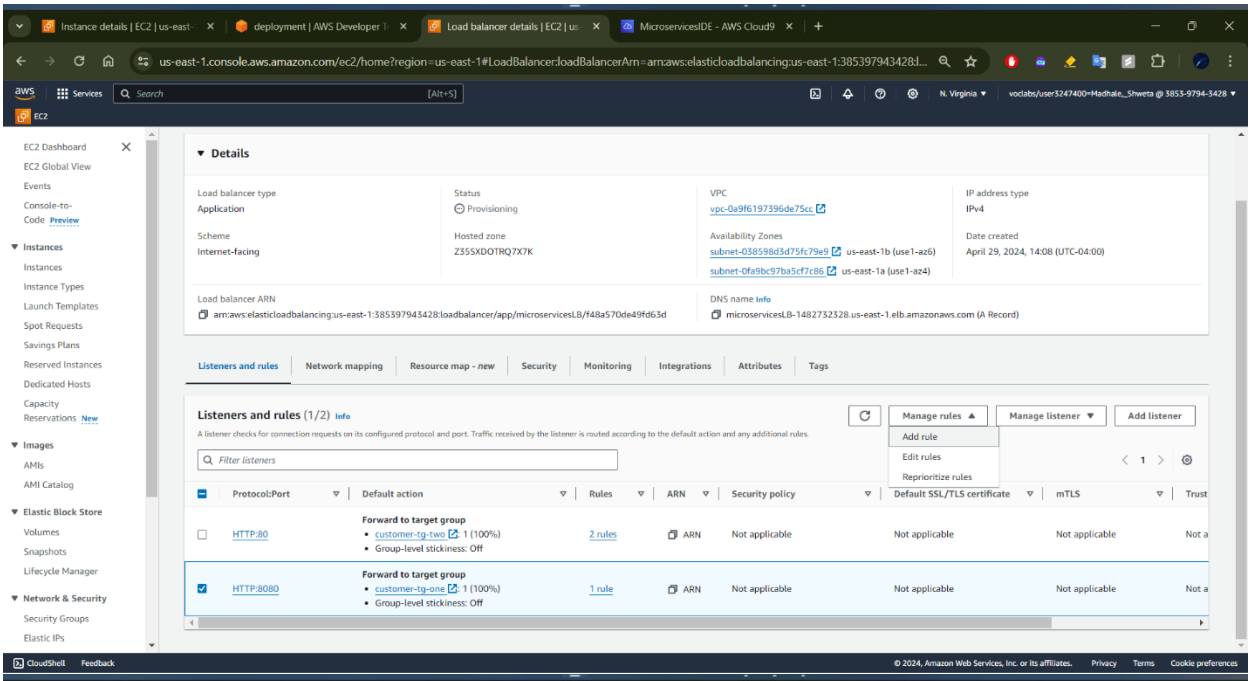

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 2 Add listener rule | EC2 |
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
EC2 > Load balancers > microservicesLB > HTTP:80 listener > Add rule
Step 1
Add rule
Step 2
Define rule conditions
Step 3
Define rule actions
Step 4
Set rule priority
Step 5
Review and create
Define rule conditions Info
Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.
Listener details: HTTP:80
Conditions (1)
Rule limits
Edit
Delete
Add condition
Path (1) Info
If
Path
is
/admin/*
AND
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

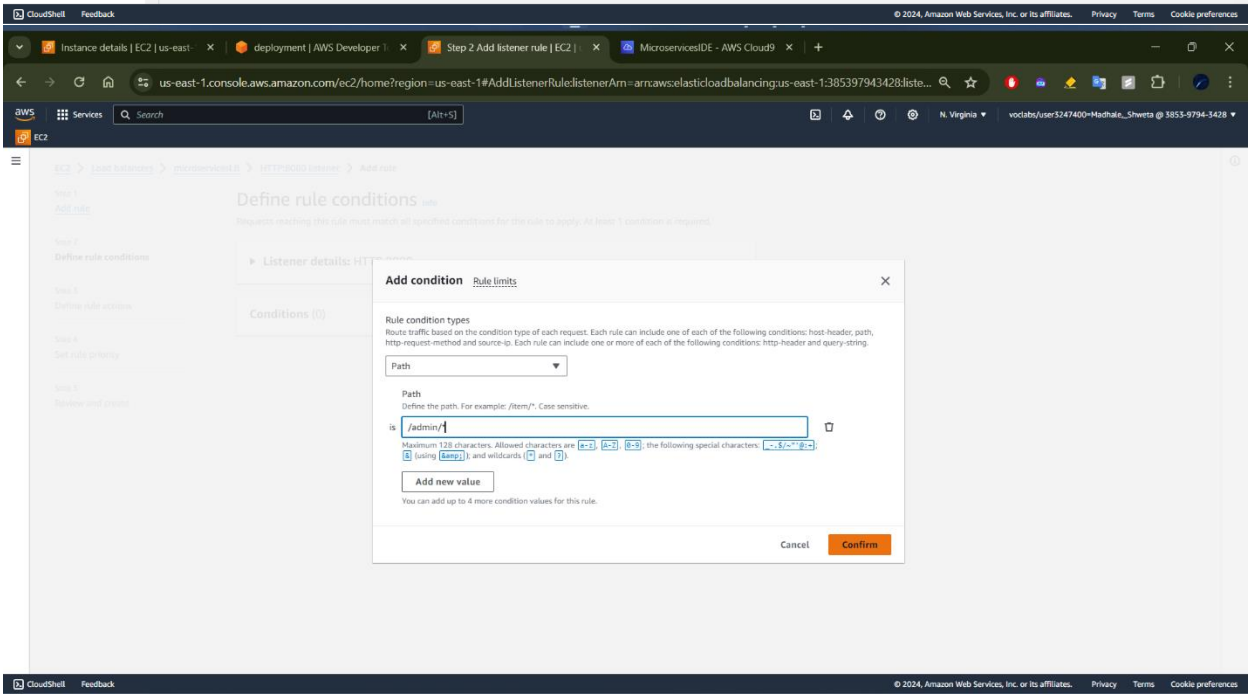

Instance details | EC2 | us-east-
deployment | AWS Developer
Step 3 Add listener rule | EC2 |
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
EC2 > Load balancers > microservicesLB > HTTP:80 listener > Add rule
Step 1
Add rule
Step 2
Define rule conditions
Step 3
Define rule actions
Step 4
Set rule priority
Step 5
Review and create
Define rule actions Info
These actions will be applied to requests matching the rule conditions.
Listener details: HTTP:80
Actions
Action types
Routing actions
Forward to target groups
Redirect to URL
Return fixed response
Forward to target group Info
Choose a target group and specify routing weight or Create target group
Target group
Weight
Percent
employee-tg-two
Target type: IP, IPv4
HTTP
1
100%
0-999
Add target group
You can add up to 4 more target groups.
Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.
Turn on group-level stickiness
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

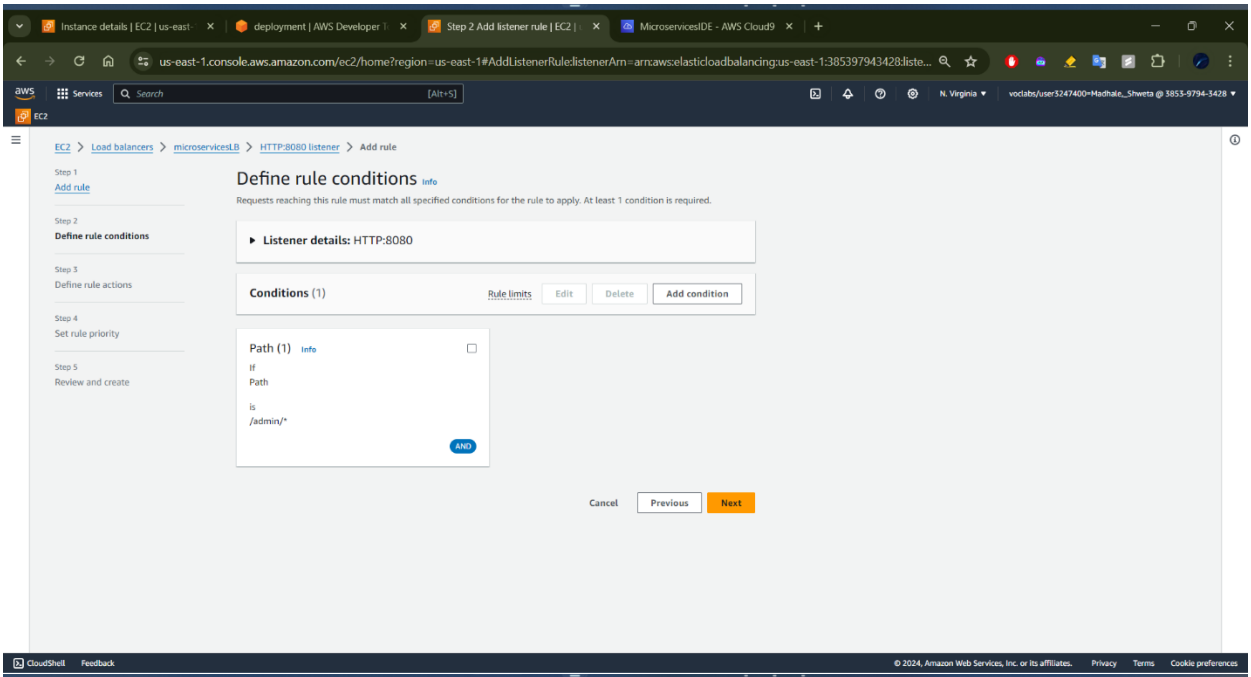


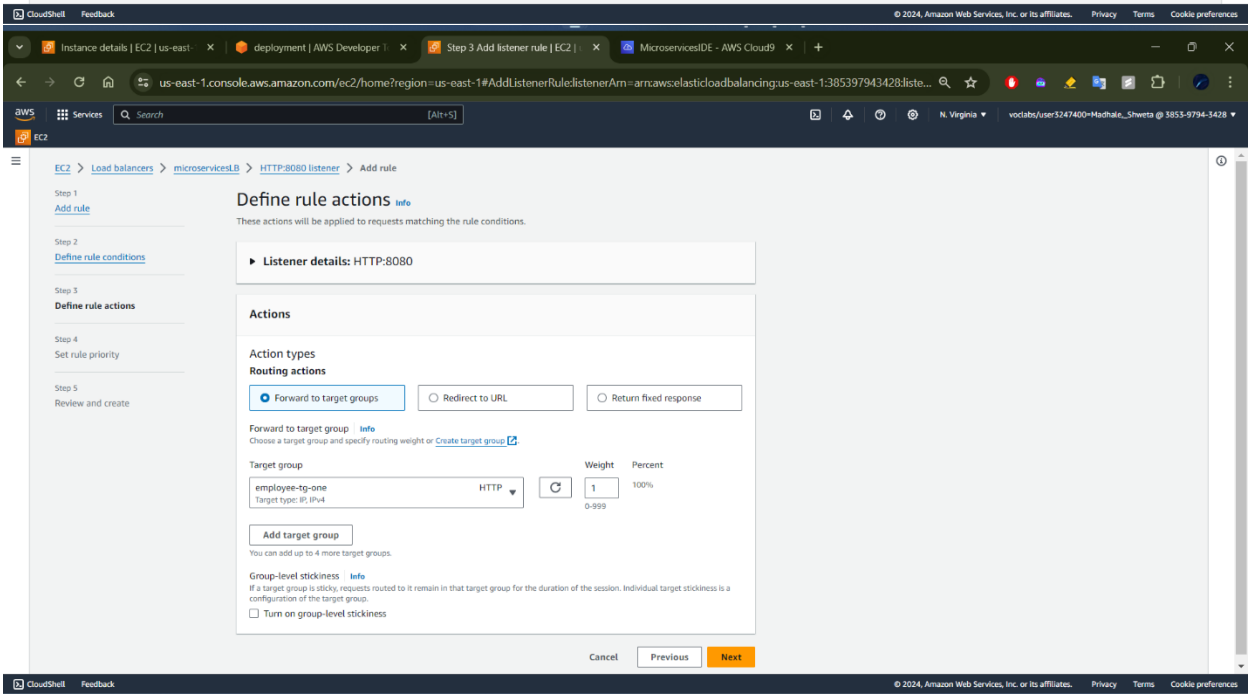


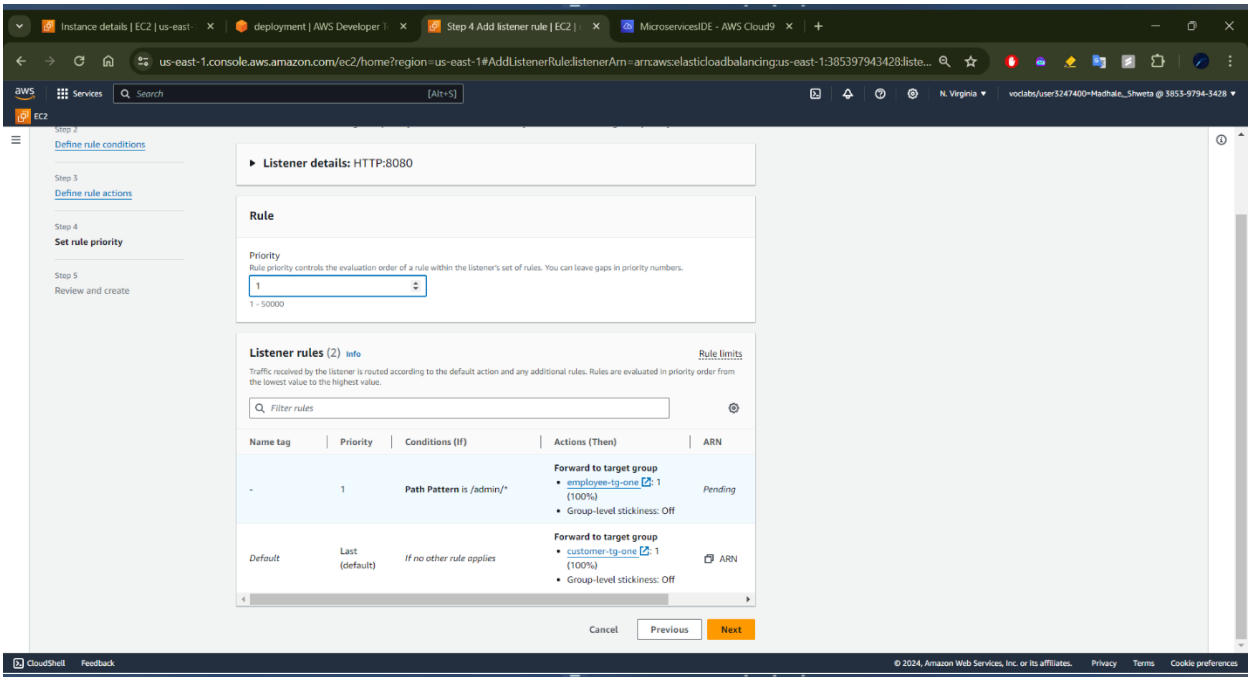
- Add a second rule for the HTTP:8080 listener. Define the following logic for this new rule: IF Path is /admin/* THEN Forward to the employee-tg-one target group.

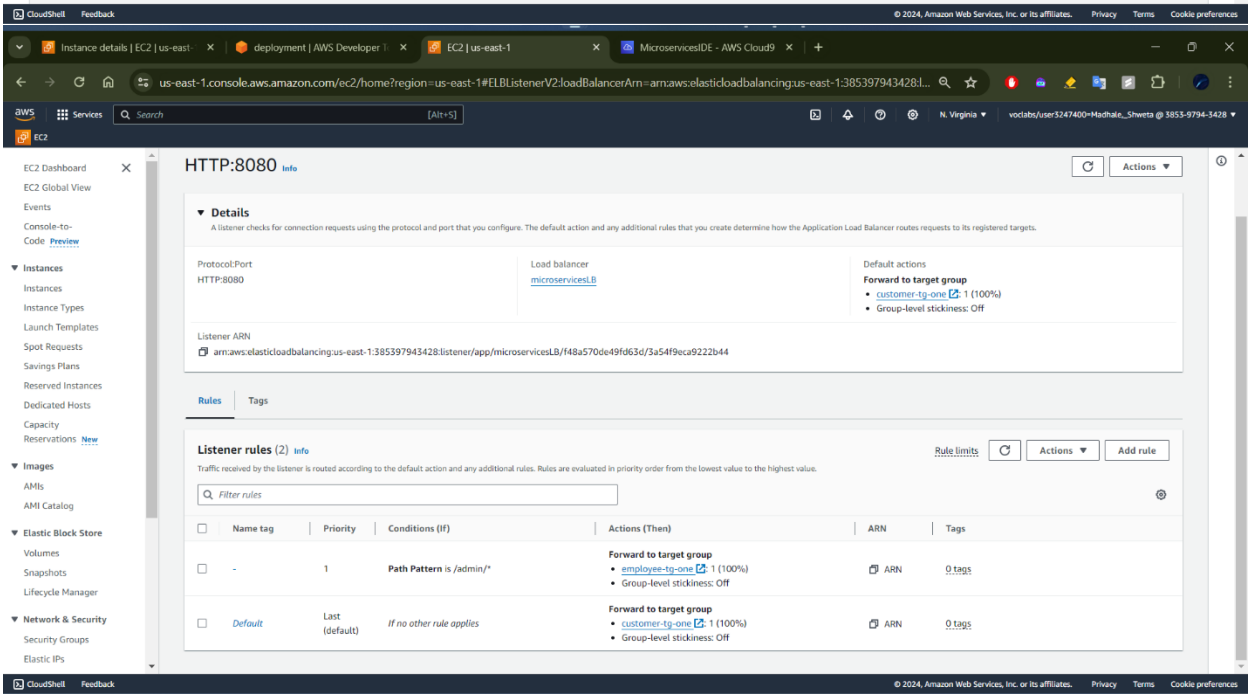

Instance details | EC2 | us-east-
deployment | AWS Developer
Load balancer details | EC2 | us
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:l...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Details
Load balancer type
Application
Status
Provisioning
VPC
vpc-0a9f6197396de75cc
IP address type
IPv4
Scheme
Internet-facing
Hosted zone
Z35SXDOTRQ7X7K
Availability Zones
subnet-038598d3d75fc79e9 us-east-1b (use1-az6)
subnet-0fa9bc97ba5cf7c86 us-east-1a (use1-az4)
Date created
April 29, 2024, 14:08 (UTC-04:00)
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:385397943428:loadbalancer/app/microservicesLB/f48a570de49fd63d
DNS name Info
microservicesLB-1482732328.us-east-1.elb.amazonaws.com (A Record)
Listeners and rules
Network mapping
Resource map - new
Security
Monitoring
Integrations
Attributes
Tags
Listeners and rules (1/2) Info
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.
Manage rules
Manage listener
Add listener
Add rule
Edit rules
Reprioritize rules
Filter listeners
Protocol:Port
Default action
Rules
ARN
Security policy
Default SSL/TLS certificate
mTLS
Trust
HTTP:80
Forward to target group
customer-tg-two 1 (100%)
Group-level stickiness: Off
2 rules
ARN
Not applicable
Not applicable
Not applicable
HTTP:8080
Forward to target group
customer-tg-one 1 (100%)
Group-level stickiness: Off
1 rule
ARN
Not applicable
Not applicable
Not applicable
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
deployment | AWS Developer
Step 2 Add listener rule | EC2 |
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Define rule conditions Info
Add condition
Rule limits
Rule condition types
Route traffic based on the condition type of each request. Each rule can include one of each of the following conditions: host-header, path, http-request-method and source-ip. Each rule can include one or more of each of the following conditions: http-header and query-string.
Path
Path
Define the path. For example: /item/*. Case sensitive.
is
/admin/*
Maximum 128 characters. Allowed characters are a-z, A-Z, 0-9; the following special characters: _-.$/~"':@+=, & (using &); and wildcards (* and ?).
Add new value
You can add up to 4 more condition values for this rule.
Cancel
Confirm
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
deployment | AWS Developer
Step 2 Add listener rule | EC2 |
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
EC2 > Load balancers > microservicesLB > HTTP:8080 listener > Add rule
Step 1
Add rule
Step 2
Define rule conditions
Step 3
Define rule actions
Step 4
Set rule priority
Step 5
Review and create
Define rule conditions Info
Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.
Listener details: HTTP:8080
Conditions (1)
Rule limits
Edit
Delete
Add condition
Path (1) Info
If
Path
is
/admin/*
AND
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instance details | EC2 | us-east-
deployment | AWS Developer
Step 3 Add listener rule | EC2 |
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...
Services
Search
[Alt+S]
N. Virginia
voclabs/user5247400=Madhale,_Shweta @ 3853-9794-3428
EC2
EC2 > Load balancers > microservicesLB > HTTP:8080 listener > Add rule
Step 1
Add rule
Step 2
Define rule conditions
Step 3
Define rule actions
Step 4
Set rule priority
Step 5
Review and create
Define rule actions Info
These actions will be applied to requests matching the rule conditions.
Listener details: HTTP:8080
Actions
Action types
Routing actions
Forward to target groups
Redirect to URL
Return fixed response
Forward to target group Info
Choose a target group and specify routing weight or Create target group
Target group
Weight
Percent
employee-tg-one
HTTP
Target type: IP, IPv4
1
100%
0-999
Add target group
You can add up to 4 more target groups.
Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.
Turn on group-level stickiness
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



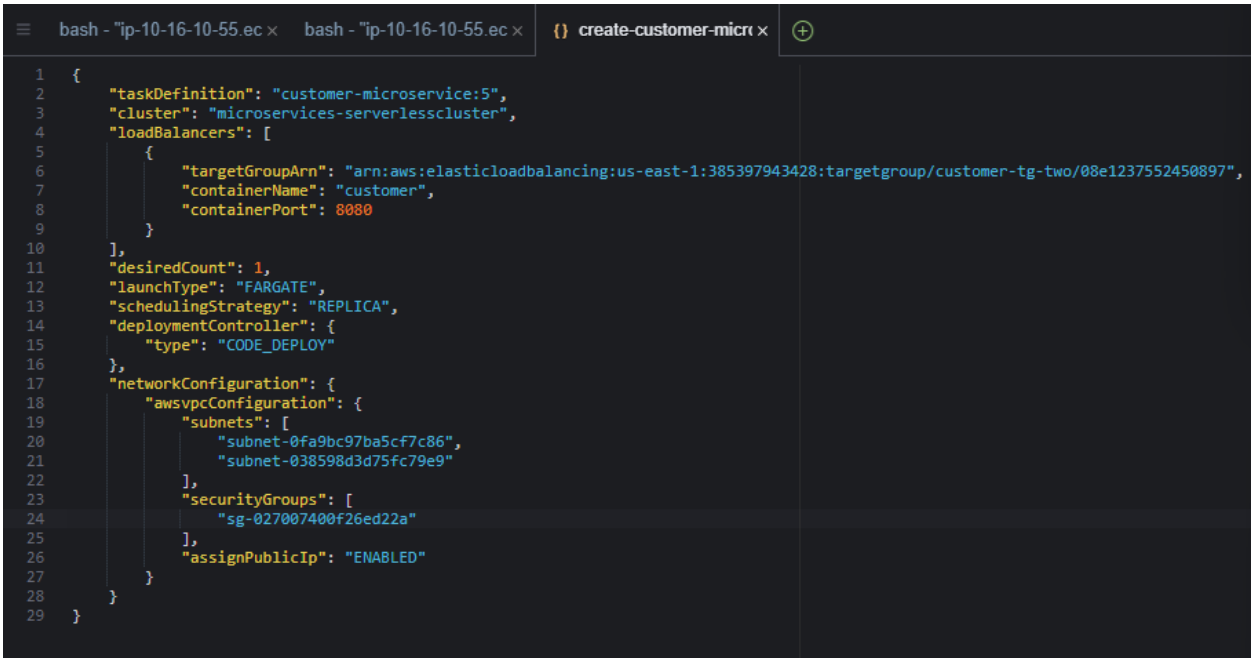


Phase 7 - Create ECS services.

Task 7.1: Create the ECS service for the customer microservice

- In AWS Cloud9, create a new file named create-customer-microservice-tg-two.json in the deployment directory
- Edit the create-customer-microservice-tg-two.json file:
    - Replace REVISION-NUMBER with the number of the latest revision of the customer-microservice task definition that is registered with Amazon ECS.
        - If this is the first time that you are completing this step, the revision number should be 1.

        - If you are repeating this step, find the latest revision number in the Amazon ECS console by choosing Task definitions, and then choosing customer-microservice.
    - Replace MICROSERVICE-TG-TWO-ARN with the actual ARN of the customer-tg-two target group.
    - Replace PUBLIC-SUBNET-1-ID with the actual subnet ID of Public Subnet1.
    - Replace PUBLIC-SUBNET-2-ID with the actual subnet ID of Public Subnet2.
    - Replace SECURITY-GROUP-ID with the actual security group ID of microservices-sg.
    - Save the changes.




```
{
    "taskDefinition": "customer-microservice:5",
    "cluster": "microservices-serverlesscluster",
    "loadBalancers": [
        {
            "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/customer-tg-two/08e1237552450897",
            "containerName": "customer",
            "containerPort": 8080
        }
    ],
    "desiredCount": 1,
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                "subnet-0fa9bc97ba5cf7c86",
                "subnet-038598d3d75fc79e9"
            ],
            "securityGroups": [
                "sg-027007400f26ed22a"
            ],
            "assignPublicIp": "ENABLED"
        }
    }
}
```

- Create the Amazon ECS service for the customer microservice, run the following commands

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

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:385397943428:service/microservices-serverlesscluster/customer-microservice",
        "serviceName": "customer-microservice",
        "clusterArn": "arn:aws:ecs:us-east-1:385397943428:cluster/microservices-serverlesscluster",
        "loadBalancers": [
            {
                "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/customer-tg-two/08e1237552450897",
                "containerName": "customer",
                "containerPort": 8080
            }
        ],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "1.4.0",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-east-1:385397943428:task-definition/customer-microservice:5",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "taskSets": [
            {
                "id": "ecs-svc/0140707528900518217",
                "taskSetArn": "arn:aws:ecs:us-east-1:385397943428:task-set/microservices-serverlesscluster/customer-microservice/ecs-svc/0140707528900518217",
                "serviceArn": "arn:aws:ecs:us-east-1:385397943428:service/microservices-serverlesscluster/customer-microservice",
                "clusterArn": "arn:aws:ecs:us-east-1:385397943428:cluster/microservices-serverlesscluster",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-east-1:385397943428:task-definition/customer-microservice:5",
                "computedDesiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "createdAt": "2024-04-29T18:14:11.117000+00:00",
                "updatedAt": "2024-04-29T18:14:11.117000+00:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-038598d3d75fc79e9",
                            "subnet-0fa9bc97ba5cf7c86"
:
```


Task 7.2: Create the Amazon ECS service for the employee microservice

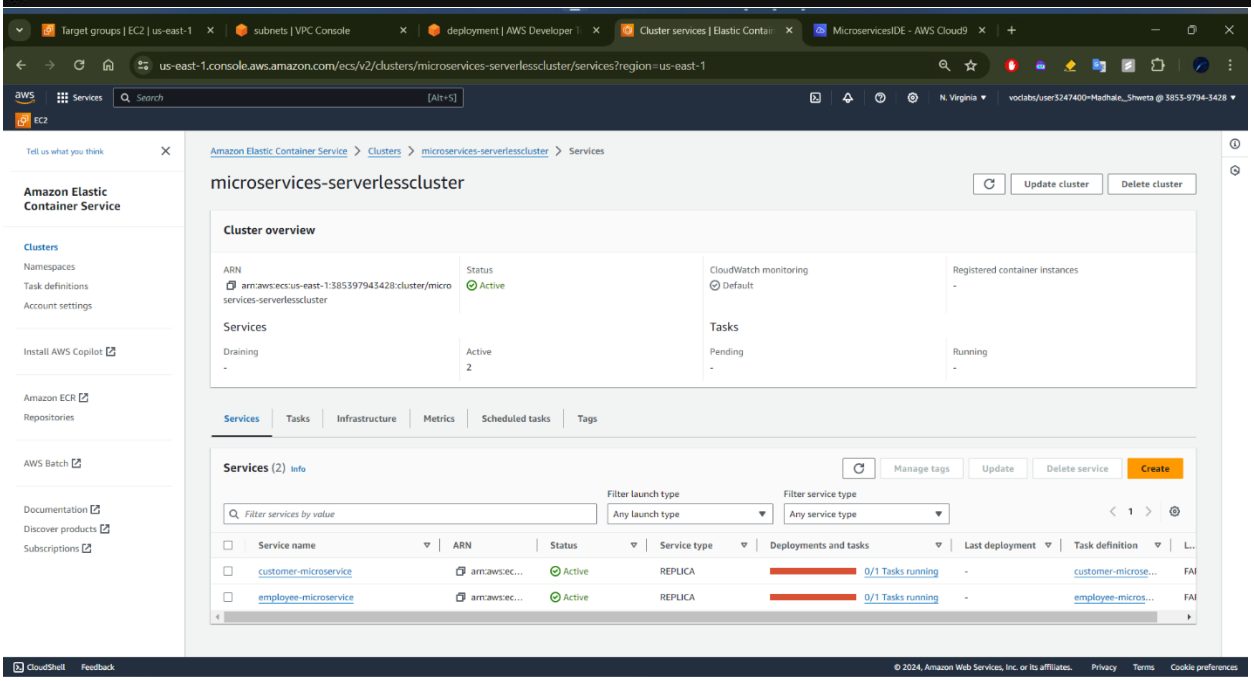
- Create an Amazon ECS service for the employee microservice


```
{
    "taskDefinition": "employee-microservice:5",
    "cluster": "microservices-serverlesscluster",
    "loadBalancers": [
        {
            "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/employee-tg-two/98bab72e0e584358",
            "containerName": "employee",
            "containerPort": 8080
        }
    ],
    "desiredCount": 1,
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                "subnet-0fa9bc97ba5cf7c86",
                "subnet-038598d3d75fc79e9"
            ],
            "securityGroups": [
                "sg-027007400f26ed22a"
            ],
            "assignPublicIp": "ENABLED"
        }
    }
}
```


- Run AWS CLI Command to create the service

```
voclabs:~/environment/deployment (dev) $ aws ecs create-service --service-name employee-microservice --cli-input-json file://create-employee-microservice-tg-two.json
```

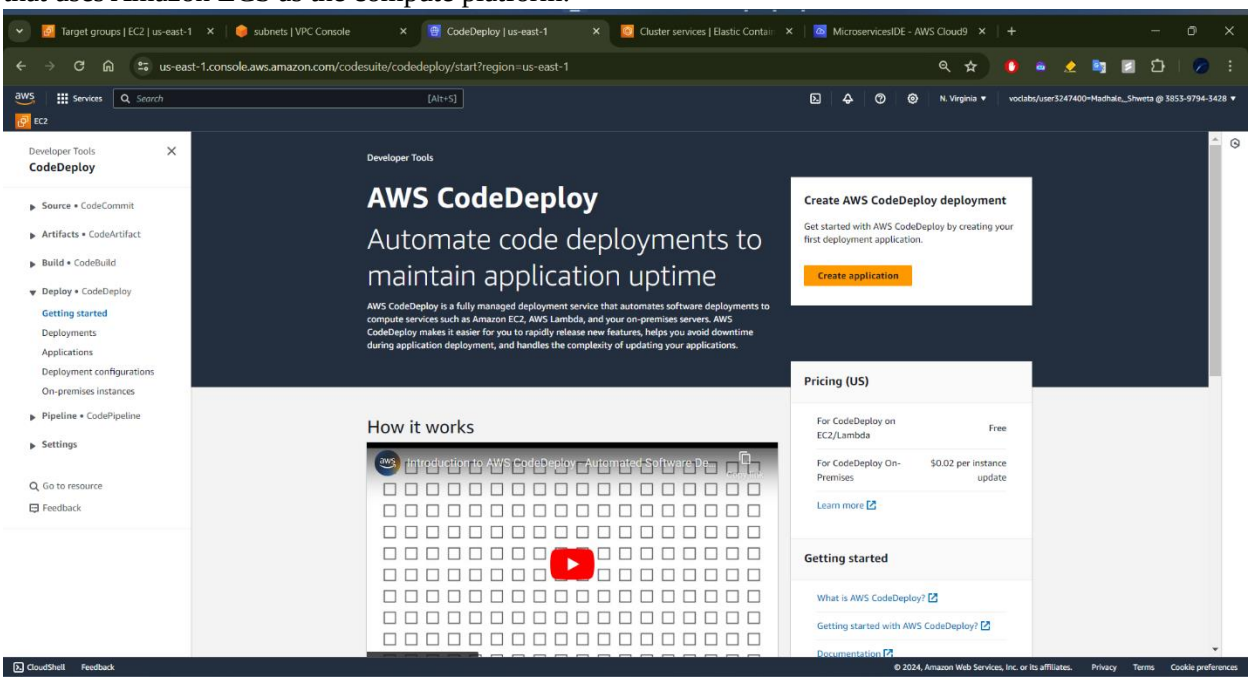

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:385397943428:service/microservices-serverlesscluster/employee-microservice",
        "serviceName": "employee-microservice",
        "clusterArn": "arn:aws:ecs:us-east-1:385397943428:cluster/microservices-serverlesscluster",
        "loadBalancers": [
            {
                "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/employee-tg-two/98bab72e0e584358",
                "containerName": "employee",
                "containerPort": 8080
            }
        ],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "1.4.0",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-east-1:385397943428:task-definition/employee-microservice:8",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "taskSets": [
            {
                "id": "ecs-svc/4349528712062614841",
                "taskSetArn": "arn:aws:ecs:us-east-1:385397943428:task-set/microservices-serverlesscluster/employee-microservice/ecs-svc/4349528712062614841",
                "serviceArn": "arn:aws:ecs:us-east-1:385397943428:service/microservices-serverlesscluster/employee-microservice",
                "clusterArn": "arn:aws:ecs:us-east-1:385397943428:cluster/microservices-serverlesscluster",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-east-1:385397943428:task-definition/employee-microservice:8",
                "computedDesiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "createdAt": "2024-04-29T18:16:45.687000+00:00",
                "updatedAt": "2024-04-29T18:16:45.687000+00:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-038598d3d75fc79e9",
                            "subnet-0fa9bc97ba5cf7c86"
:
```

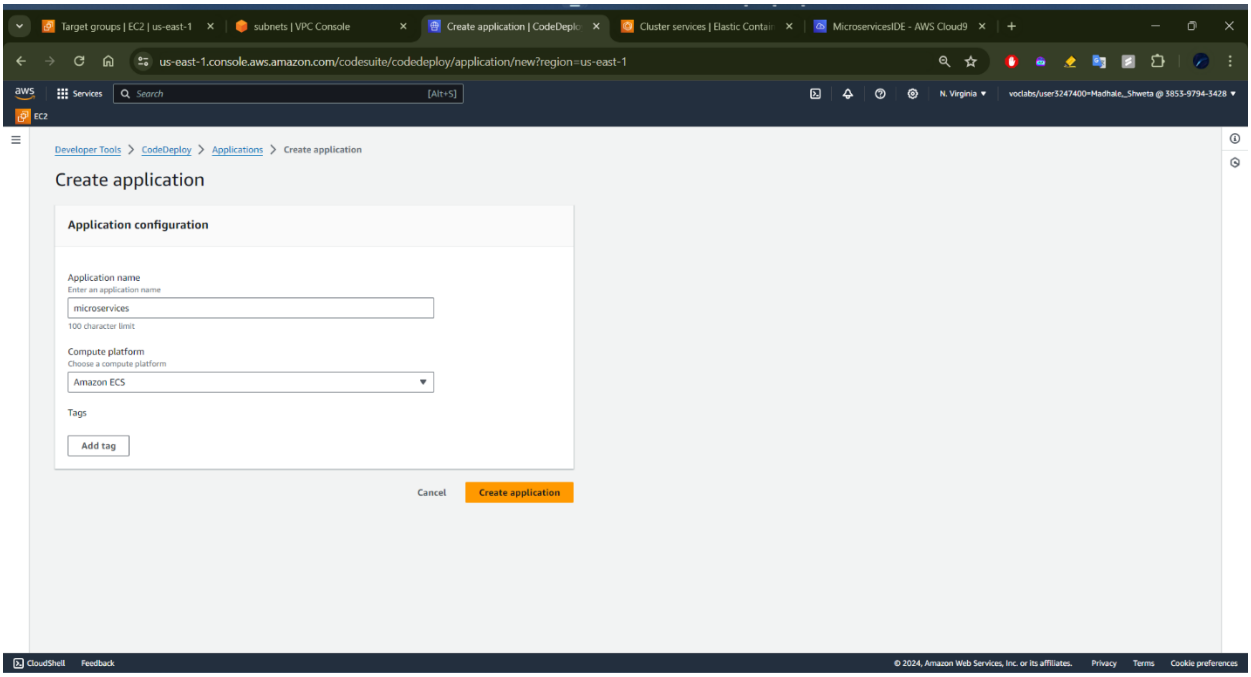
Phase 8 – Configure applications and deployments groups in CodeDeploy, and create two CI/CD pipelines by using CodePipeline.

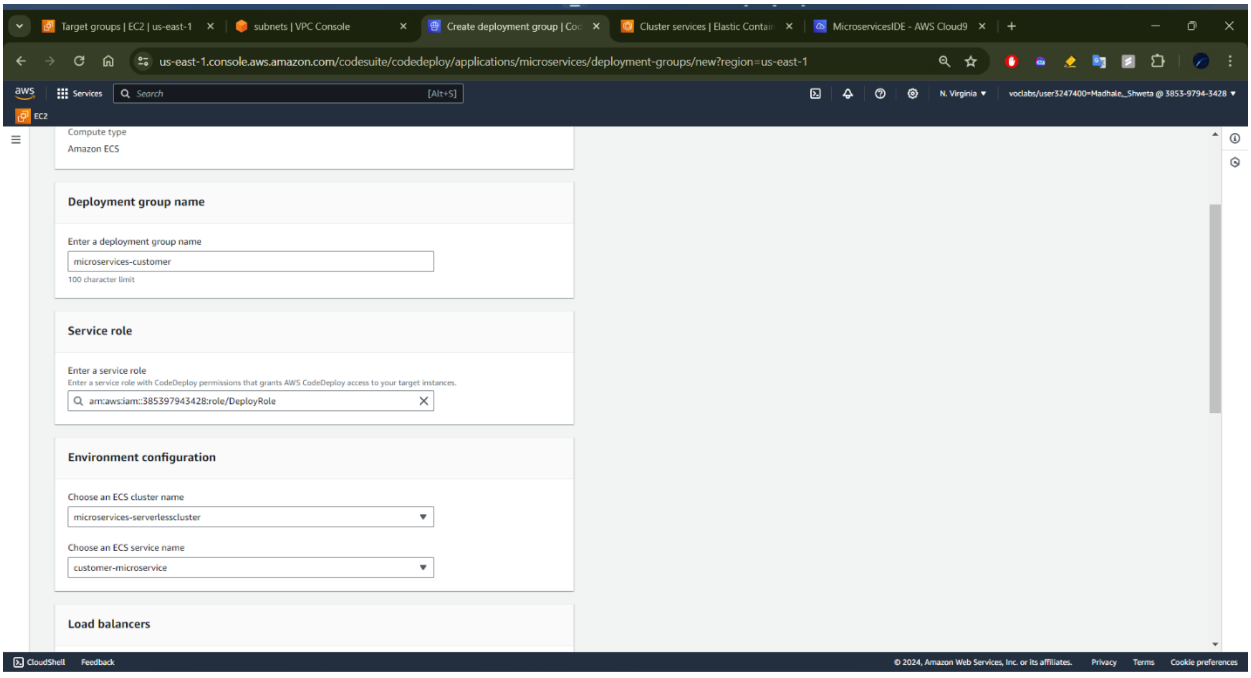
Task 8.1: Create a CodeDeploy application and deployment groups

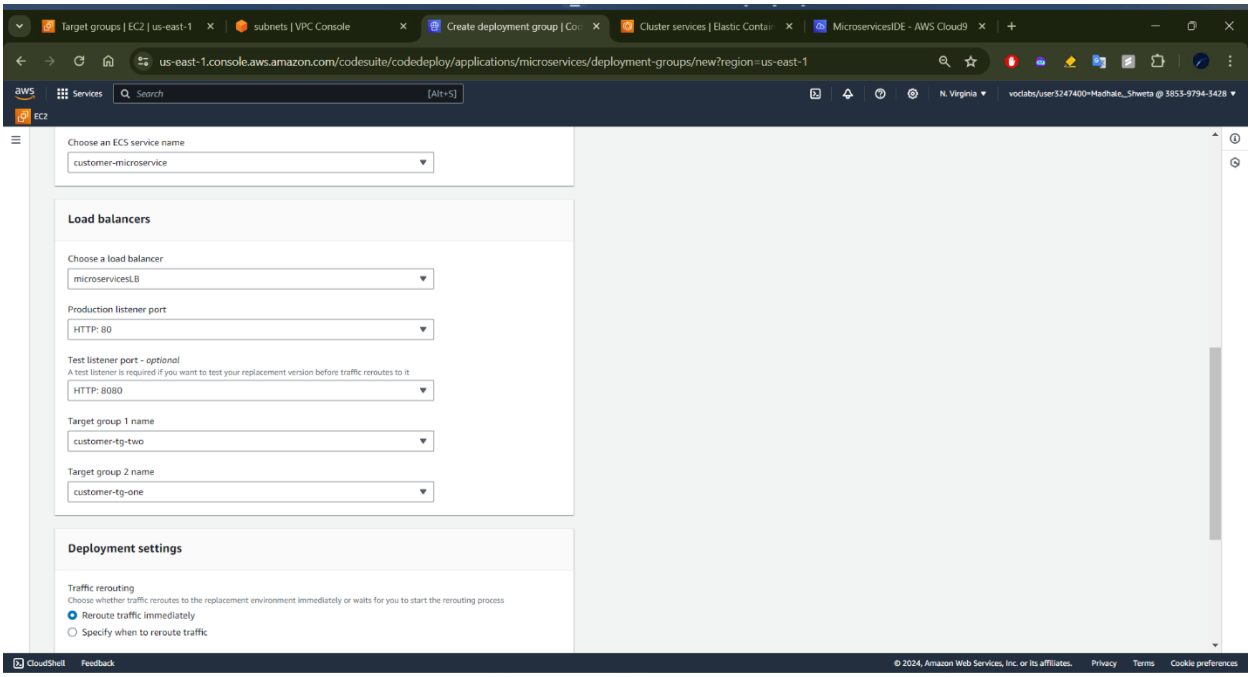
- Use the CodeDeploy console to create a CodeDeploy application with the name microservices that uses Amazon ECS as the compute platform.

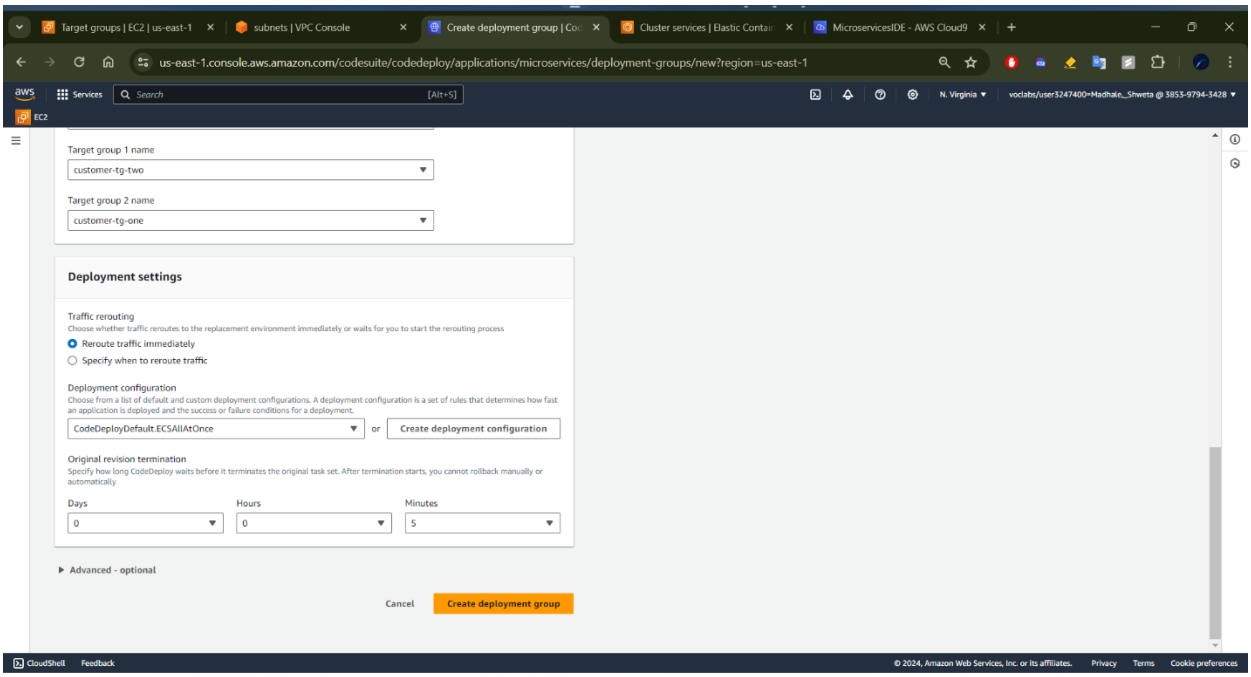


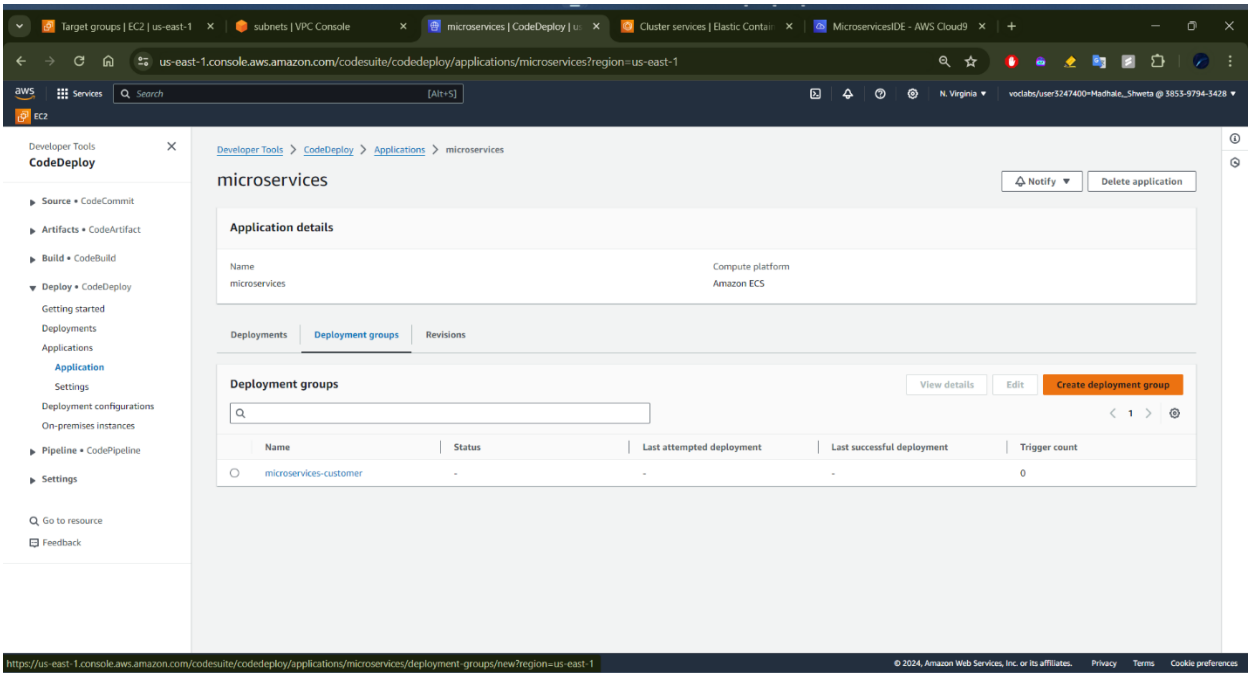
- Create a CodeDeploy deployment group for the customer microservice.
    - On the microservices application detail page, choose the Deployment groups tab.
    - Choose Create deployment group and configure the following:
        - Deployment group name: Enter microservices-customer
        - Service role: Place your cursor in the search box, and choose the ARN for DeployRole.
        - In the Environment configuration section:
        - ECS cluster name: Choose microservices-serverlesscluster.
        - ECS service name: Choose customer-microservice.
        - In the Load balancers section:
        - Load balancer: Choose microservicesLB.
        - Production listener port: Choose HTTP:80.
        - Test listener port: Choose HTTP:8080.
        - Target group 1 name: Choose customer-tg-two.
        - Target group 2 name: Choose customer-tg-one.
        - In the Deployment settings section:
        - Traffic rerouting: Choose Reroute traffic immediately.
        - Deployment configuration: Choose CodeDeployDefault.ECSAllAtOnce.
        - Original revision termination: Days: 0, Hours: 0, Minutes: 5
    - Choose Create deployment group.


Target groups | EC2 | us-east-1
subnets | VPC Console
Create application | CodeDeploy
Cluster services | Elastic Contai
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codedeploy/application/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
EC2
Developer Tools > CodeDeploy > Applications > Create application
Create application
Application configuration
Application name
Enter an application name
microservices
100 character limit
Compute platform
Choose a compute platform
Amazon ECS
Tags
Add tag
Cancel
Create application
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

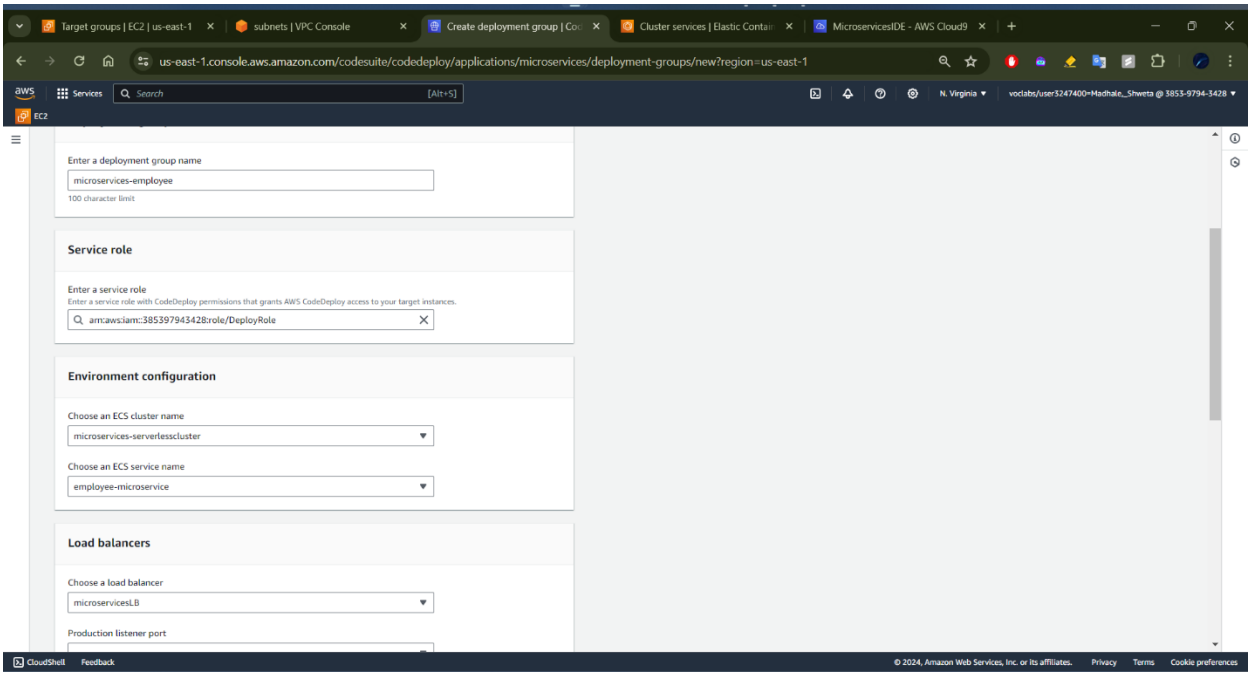

Target groups | EC2 | us-east-1
subnets | VPC Console
Create deployment group | Co
Cluster services | Elastic Contai
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
EC2
Compute type
Amazon ECS
Deployment group name
Enter a deployment group name
microservices-customer
100 character limit
Service role
Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
arn:aws:iam::385397943428:role/DeployRole
Environment configuration
Choose an ECS cluster name
microservices-serverlesscluster
Choose an ECS service name
customer-microservice
Load balancers
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

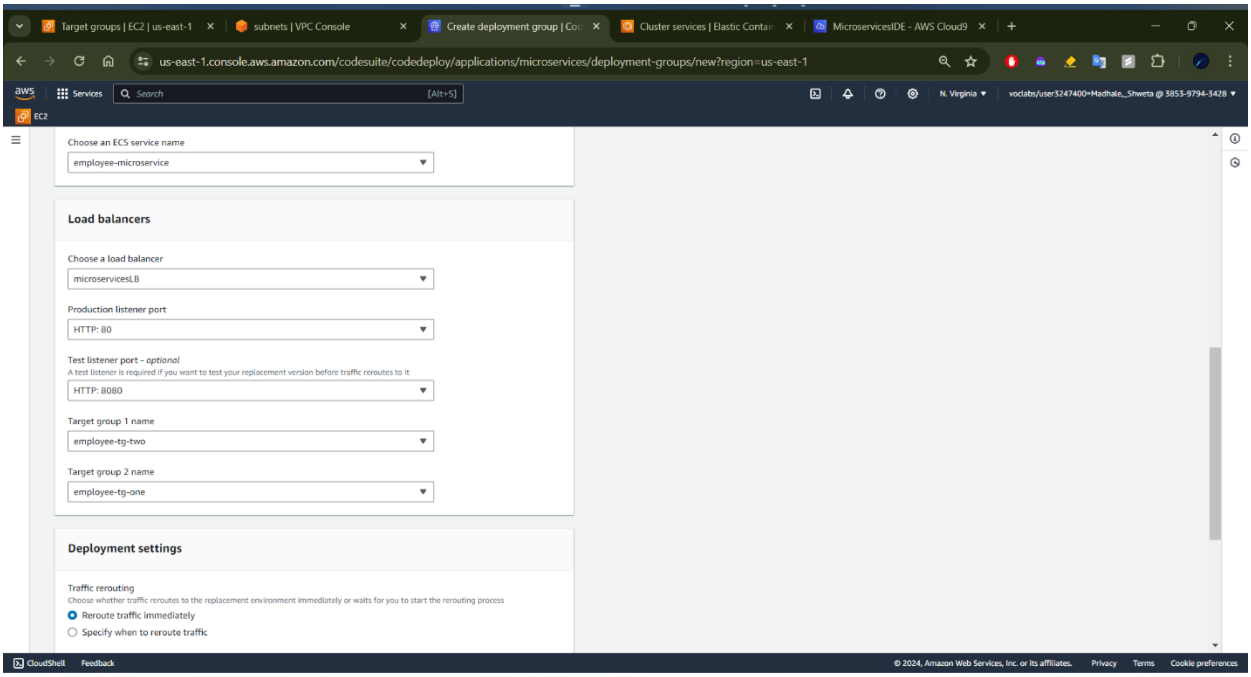

Target groups | EC2 | us-east-1
subnets | VPC Console
Create deployment group | Co
Cluster services | Elastic Contai
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale_Shweta @ 3853-9794-3428
EC2
Choose an ECS service name
customer-microservice
Load balancers
Choose a load balancer
microservicesLB
Production listener port
HTTP: 80
Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it
HTTP: 8080
Target group 1 name
customer-tg-two
Target group 2 name
customer-tg-one
Deployment settings
Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
Reroute traffic immediately
Specify when to reroute traffic
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

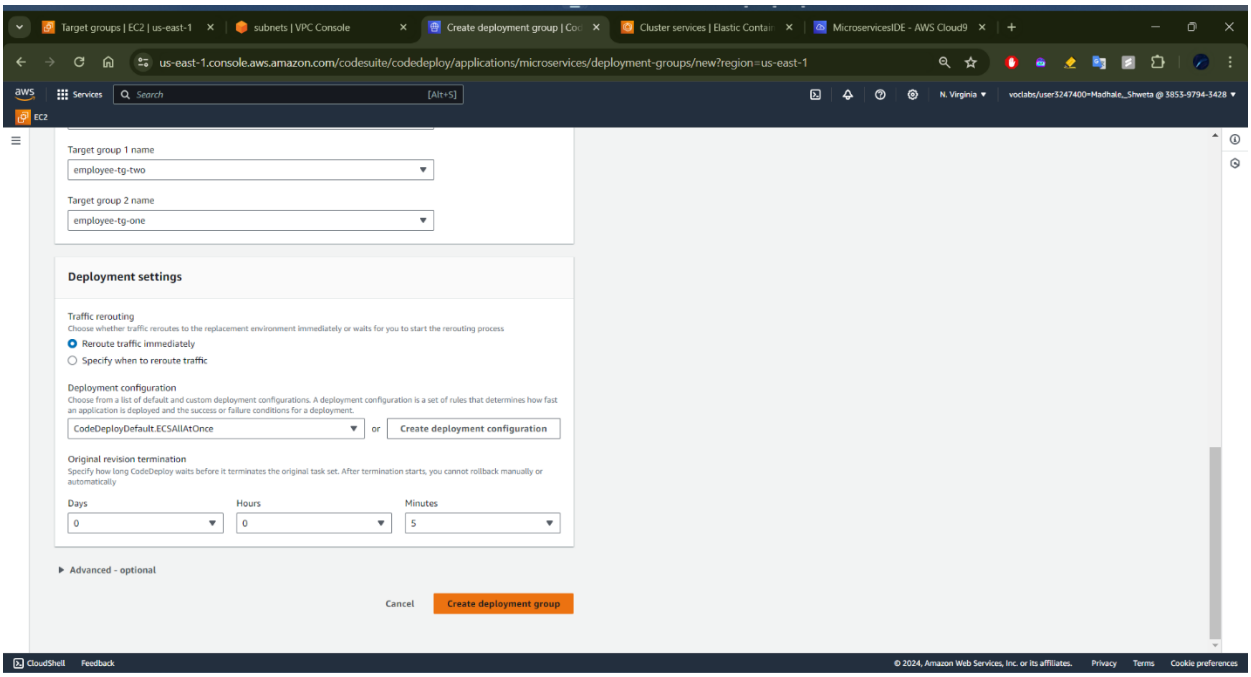

Target groups | EC2 | us-east-1
subnets | VPC Console
Create deployment group | Co
Cluster services | Elastic Contai
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale_Shweta @ 3853-9794-3428
EC2
Target group 1 name
customer-tg-two
Target group 2 name
customer-tg-one
Deployment settings
Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
Reroute traffic immediately
Specify when to reroute traffic
Deployment configuration
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.
CodeDeployDefault.ECSAllAtOnce
or
Create deployment configuration
Original revision termination
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically
Days
Hours
Minutes
0
0
5
Advanced - optional
Cancel
Create deployment group
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

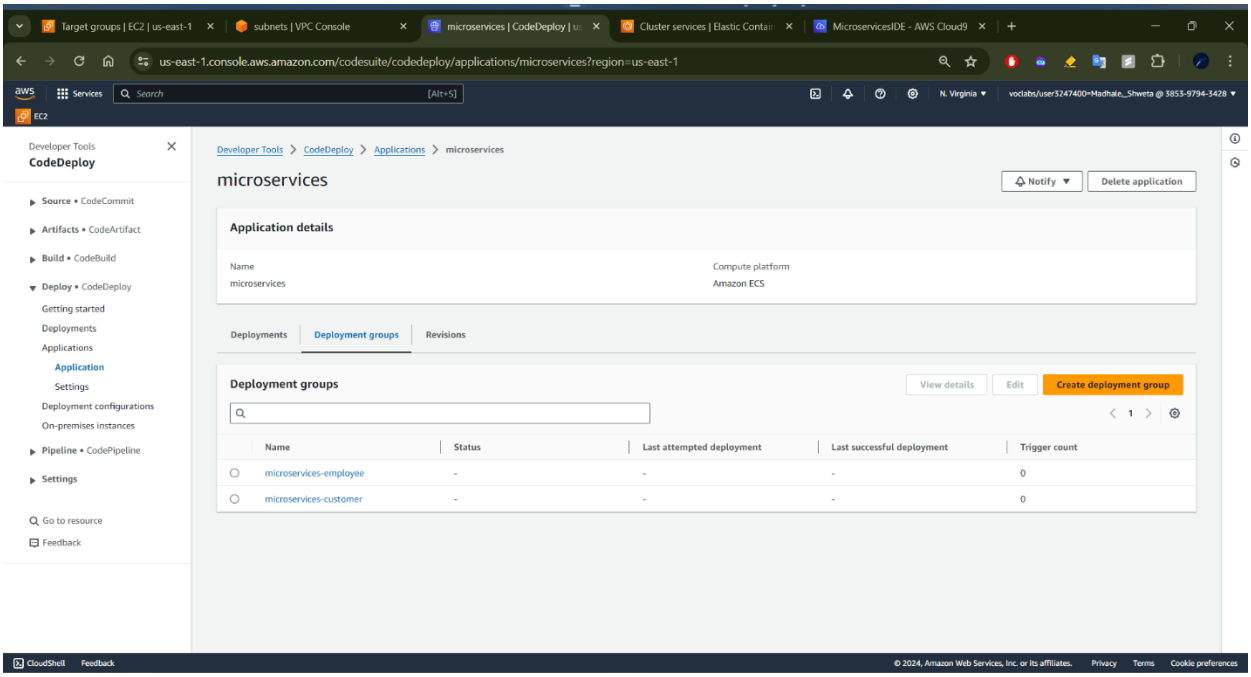


- Create a CodeDeploy deployment group for the employee microservice. Specify the same settings that you did in the prior step, except for the following:
    - Deployment group name: Enter microservices-employee
    - ECS service name: Choose employee-microservice.
    - Target group 1 name: Choose employee-tg-two.
    - Target group 2 name: Choose employee-tg-one.



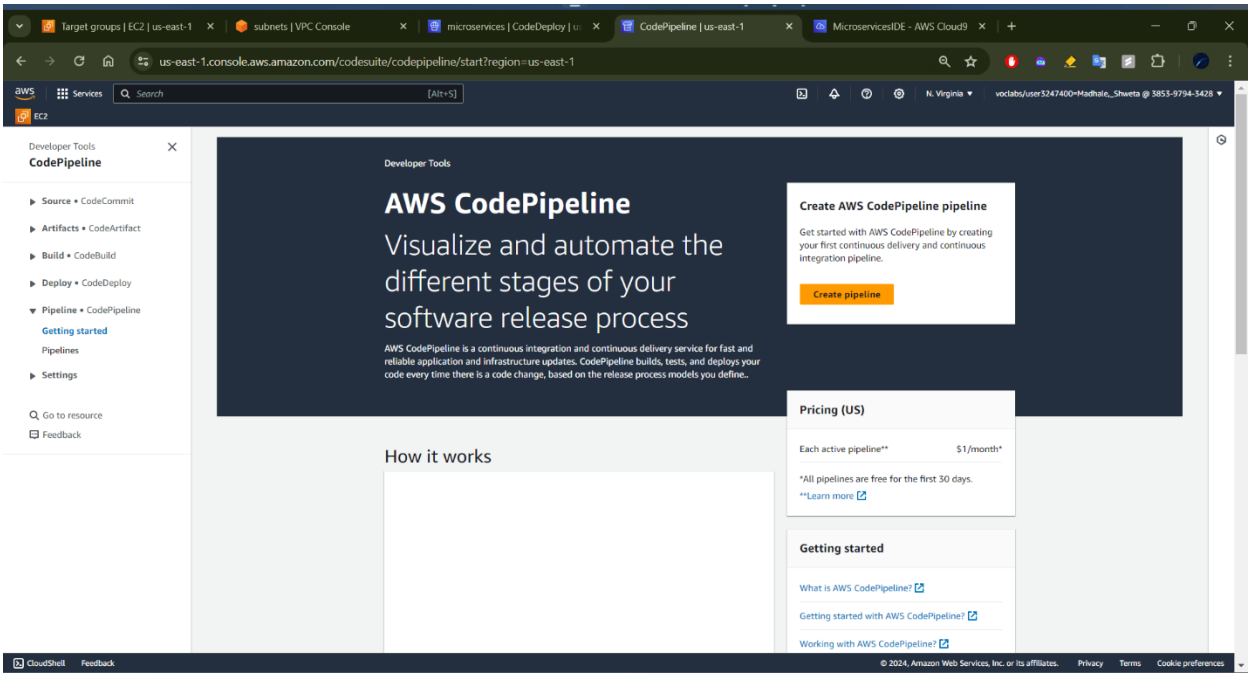

Target groups | EC2 | us-east-1
subnets | VPC Console
Create deployment group | Co
Cluster services | Elastic Contai
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale._Shweta @ 3853-9794-3428
EC2
Choose an ECS service name
employee-microservice
Load balancers
Choose a load balancer
microservicesLB
Production listener port
HTTP: 80
Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it
HTTP: 8080
Target group 1 name
employee-tg-two
Target group 2 name
employee-tg-one
Deployment settings
Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
Reroute traffic immediately
Specify when to reroute traffic
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

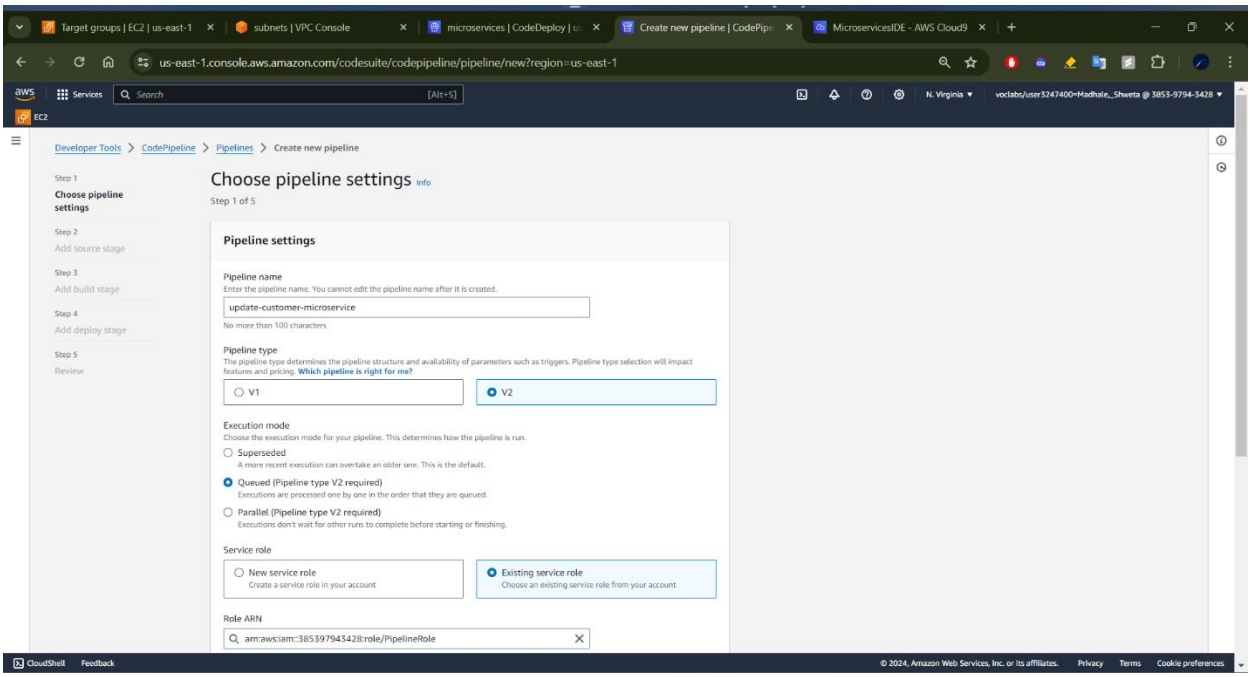

Target groups | EC2 | us-east-1
subnets | VPC Console
Create deployment group | Co
Cluster services | Elastic Contai
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale._Shweta @ 3853-9794-3428
EC2
Target group 1 name
employee-tg-two
Target group 2 name
employee-tg-one
Deployment settings
Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
Reroute traffic immediately
Specify when to reroute traffic
Deployment configuration
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.
CodeDeployDefault.ECSAllAtOnce
or
Create deployment configuration
Original revision termination
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically
Days
Hours
Minutes
0
0
5
Advanced - optional
Cancel
Create deployment group
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

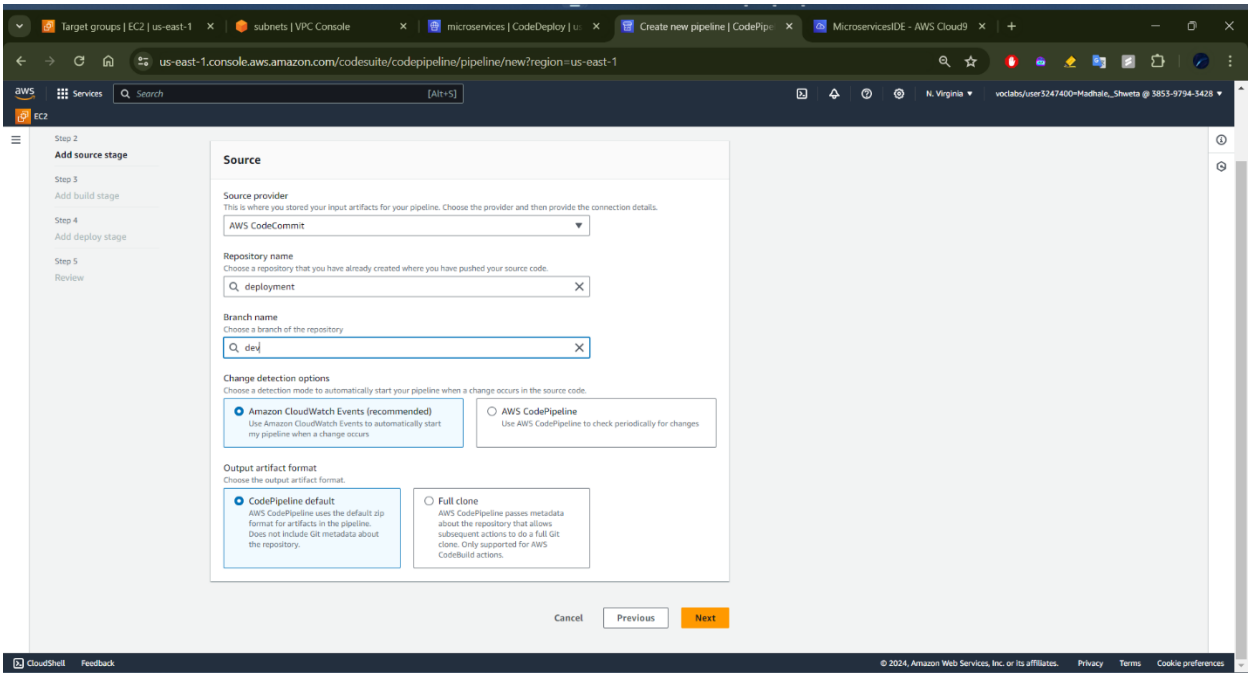


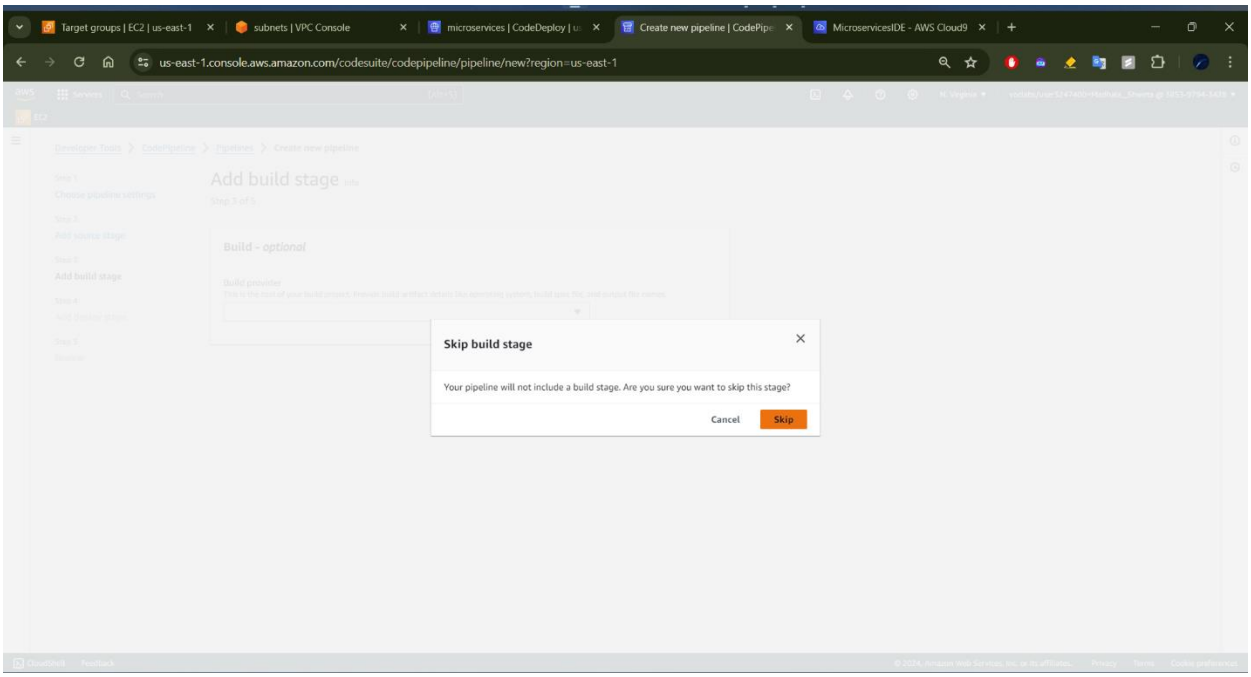
Task 8.2: Create a pipeline for the customer microservice

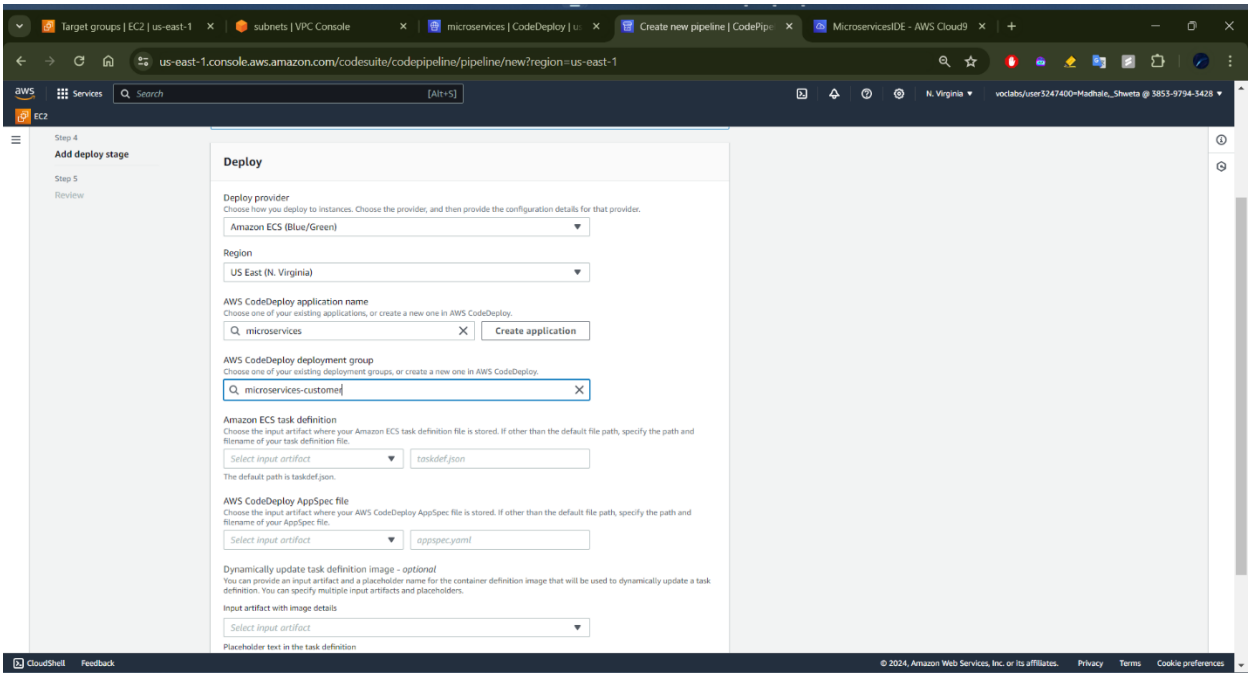
- In the CodePipeline console, create a customer pipeline with the following settings:
    - Pipeline name: Enter update-customer-microservice
    - Service role: Choose the ARN for PipelineRole.
    - Source provider: Choose AWS CodeCommit.
    - Repository name: Choose deployment.
    - Note: You have defined two CodeCommit repositories. The deployment repository contains the Amazon ECS task definition files and CodeDeploy AppSpec files that your pipeline will need, so that is the one you choose here.
    - Branch name: Choose dev.
    - Note: Skip the build stage.
    - Deploy provider: Amazon ECS (Blue/Green)
    - Region: US East (N. Virginia).
    - AWS CodeDeploy application name: microservices
    - AWS CodeDeploy deployment group: microservices-customer
    - Under Amazon ECS task definition:
    - Set a SourceArtifact with a value of taskdef-customer.json
    - Under AWS CodeDeploy AppSpec file:
    - Set a SourceArtifact with a value of appspec-customer.yaml
    - Note: Leave the Dynamically update task definition image fields blank for now.

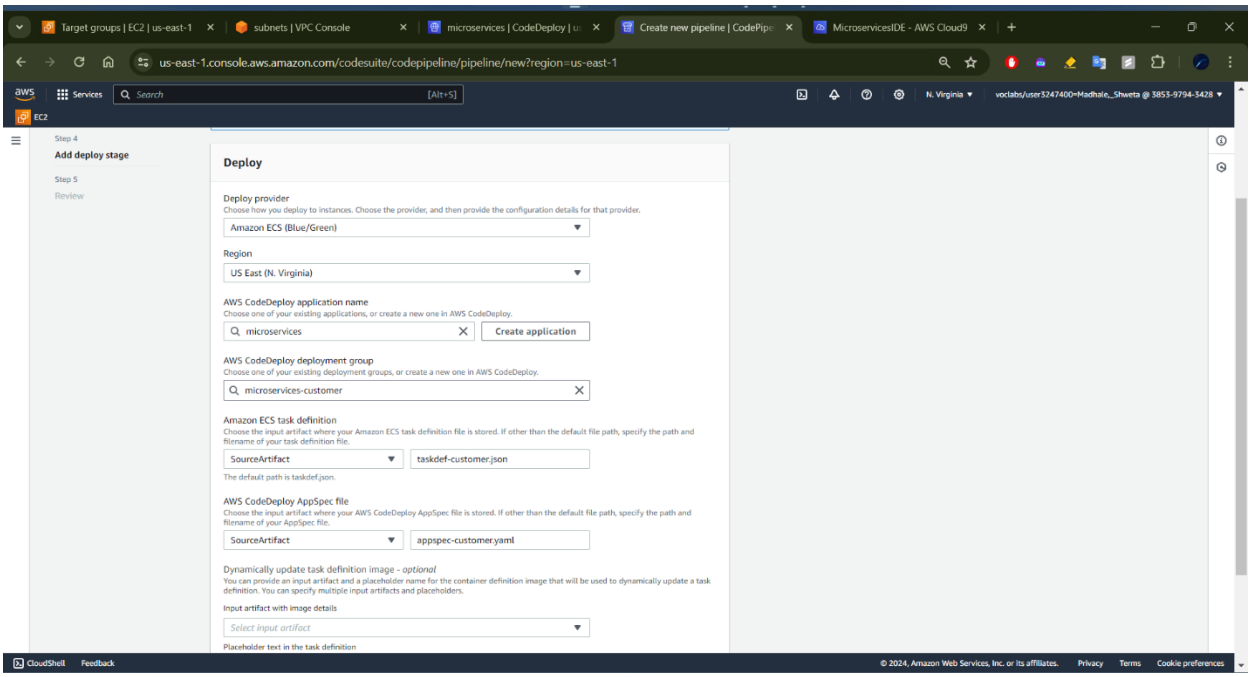

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
CodePipeline | us-east-1
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/start?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Settings
Go to resource
Feedback
Developer Tools
AWS CodePipeline
Visualize and automate the different stages of your software release process
AWS CodePipeline is a continuous integration and continuous delivery service for fast and reliable application and infrastructure updates. CodePipeline builds, tests, and deploys your code every time there is a code change, based on the release process models you define..
Create AWS CodePipeline pipeline
Get started with AWS CodePipeline by creating your first continuous delivery and continuous integration pipeline.
Create pipeline
Pricing (US)
Each active pipeline**
$1/month*
*All pipelines are free for the first 30 days.
**Learn more
Getting started
What is AWS CodePipeline?
Getting started with AWS CodePipeline?
Working with AWS CodePipeline?
How it works
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

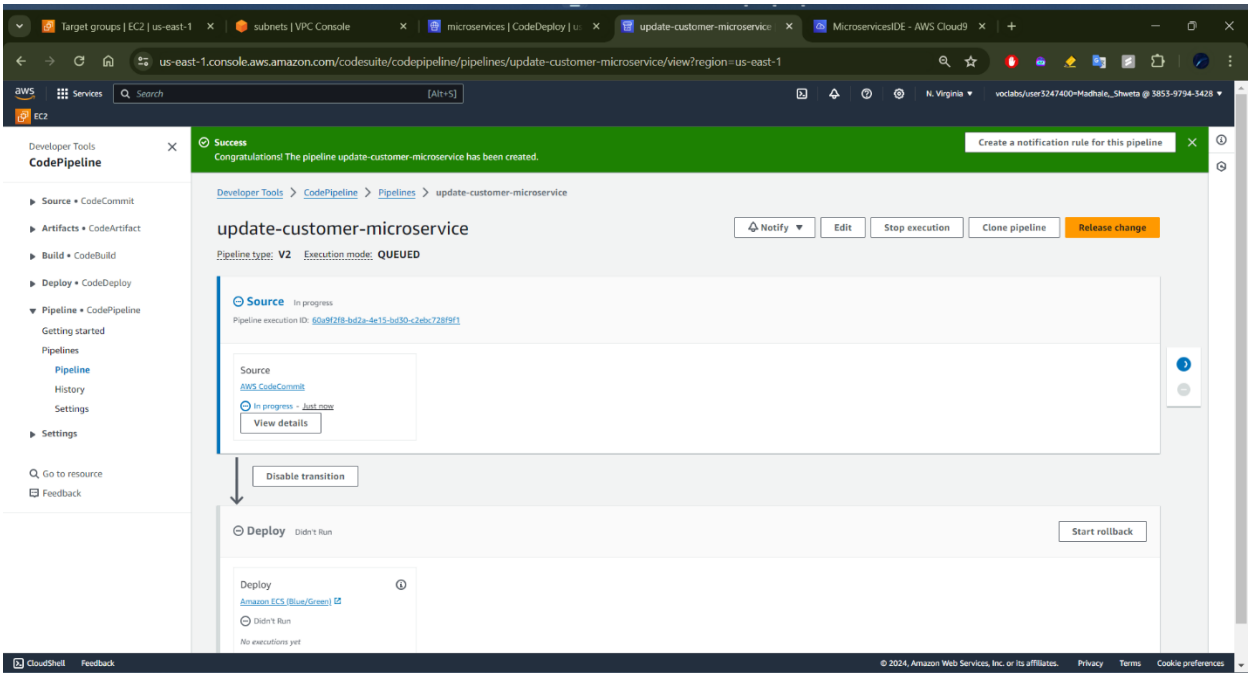

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Create new pipeline | CodePipe
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
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
Choose pipeline settings Info
Step 1 of 5
Pipeline settings
Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
update-customer-microservice
No more than 100 characters
Pipeline type
The pipeline type determines the pipeline structure and availability of parameters such as triggers. Pipeline type selection will impact features and pricing. Which pipeline is right for me?
V1
V2
Execution mode
Choose the execution mode for your pipeline. This determines how the pipeline is run.
Superseded
A more recent execution can overtake an older one. This is the default.
Queued (Pipeline type V2 required)
Executions are processed one by one in the order that they are queued.
Parallel (Pipeline type V2 required)
Executions don't wait for other runs to complete before starting or finishing.
Service role
New service role
Create a service role in your account
Existing service role
Choose an existing service role from your account
Role ARN
arn:aws:iam::385397943428:role/PipelineRole
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Create new pipeline | CodePipe
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
AWS CodeCommit
Repository name
Choose a repository that you have already created where you have pushed your source code.
deployment
Branch name
Choose a branch of the repository
dev
Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.
Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs
AWS CodePipeline
Use AWS CodePipeline to check periodically for changes
Output artifact format
Choose the output artifact format.
CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.
Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

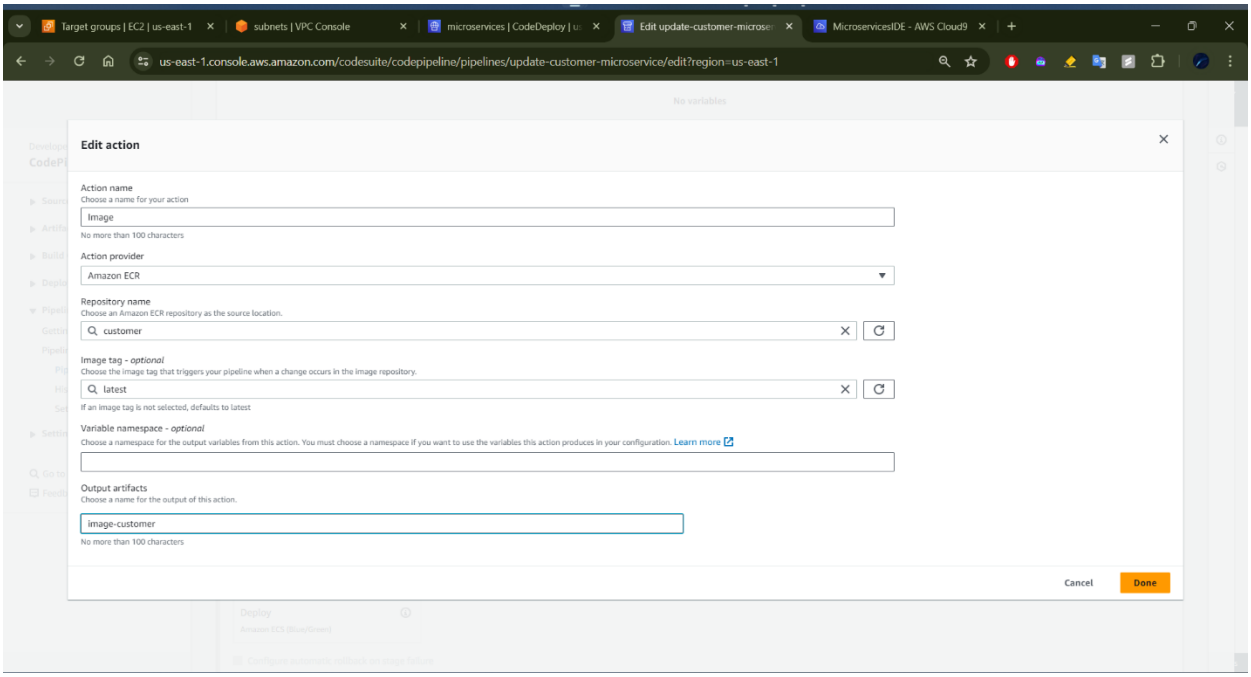

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Create new pipeline | CodePipe
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1
Add build stage
Build - optional
Skip build stage
Your pipeline will not include a build stage. Are you sure you want to skip this stage?
Cancel
Skip

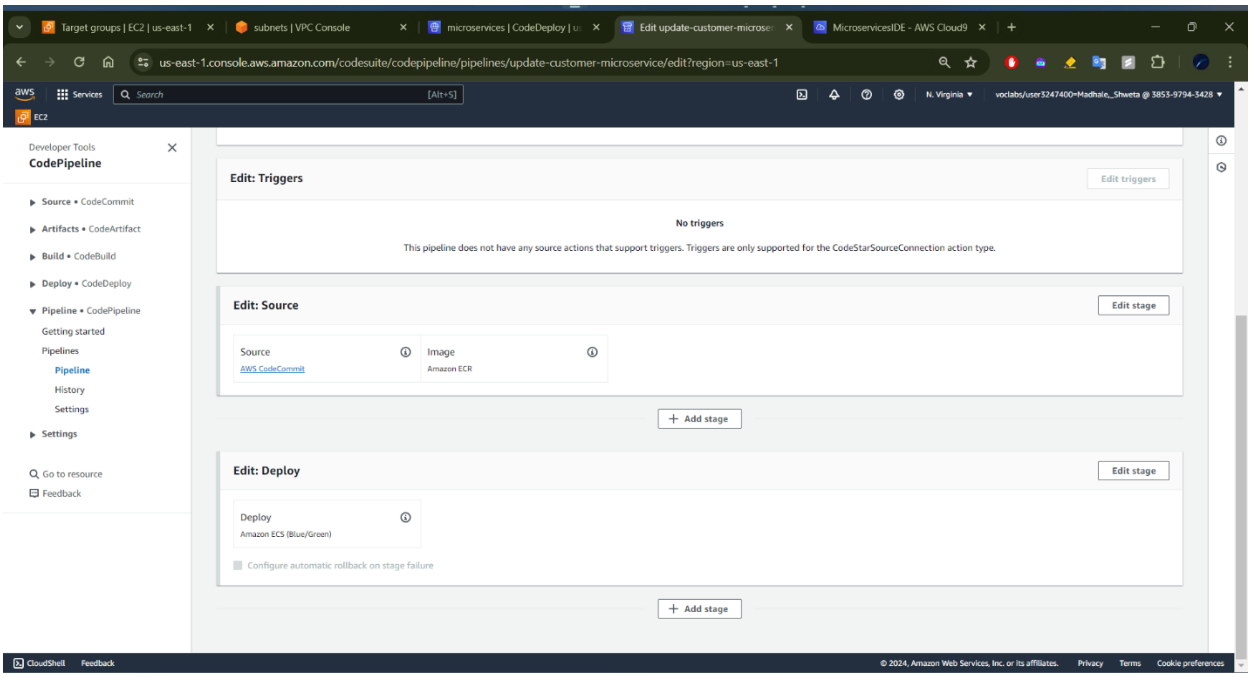

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Create new pipeline | CodePipe
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
EC2
Step 4
Add deploy stage
Step 5
Review
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
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.
microservices-customer
Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.
Select input artifact
taskdef.json
The default path is taskdef.json.
AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.
Select input artifact
appspec.yaml
Dynamically update task definition image - optional
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.
Input artifact with image details
Select input artifact
Placeholder text in the task definition
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

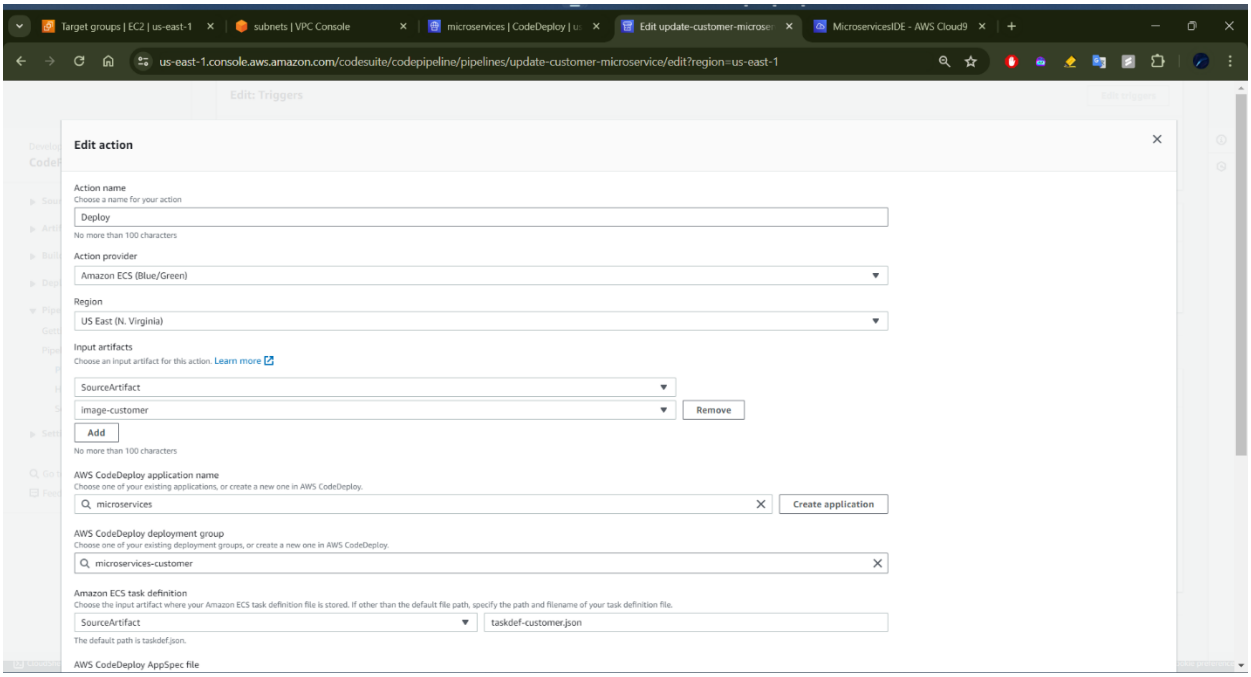

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Create new pipeline | CodePipe
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1
Services
Search
[Alt+S]
N. Virginia
EC2
Step 4
Add deploy stage
Step 5
Review
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
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.
microservices-customer
Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.
SourceArtifact
taskdef-customer.json
The default path is taskdef.json.
AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.
SourceArtifact
appspec-customer.yaml
Dynamically update task definition image - optional
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.
Input artifact with image details
Select input artifact
Placeholder text in the task definition
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

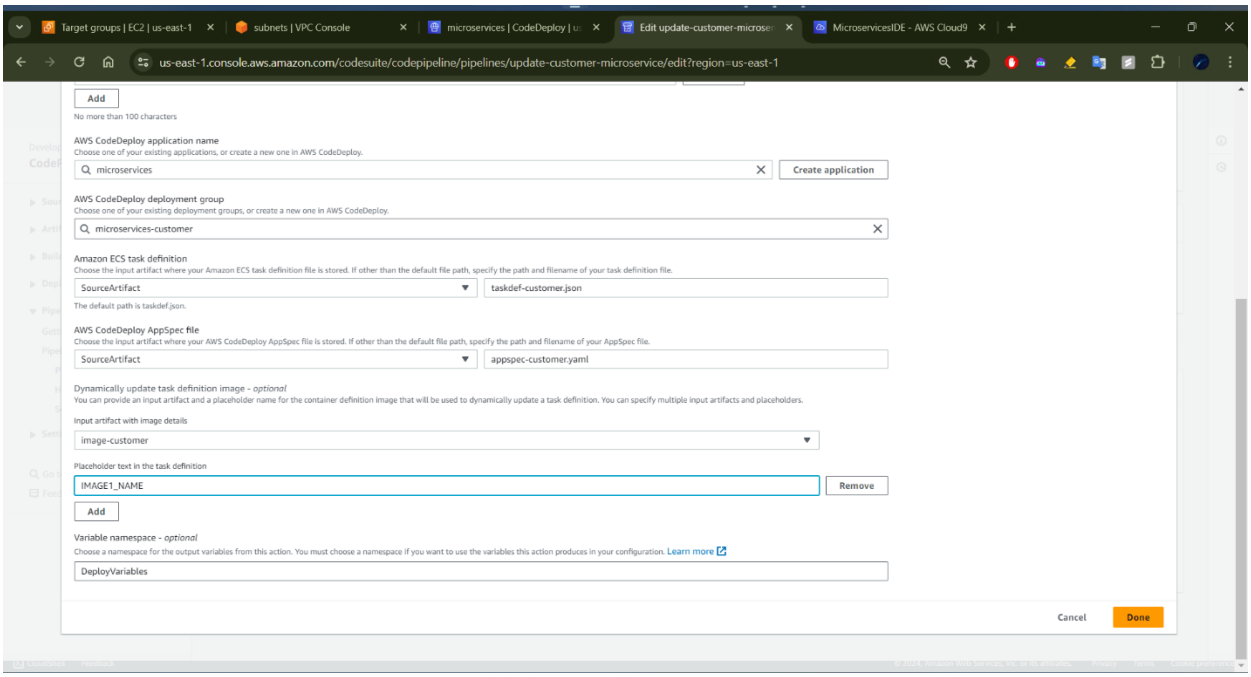


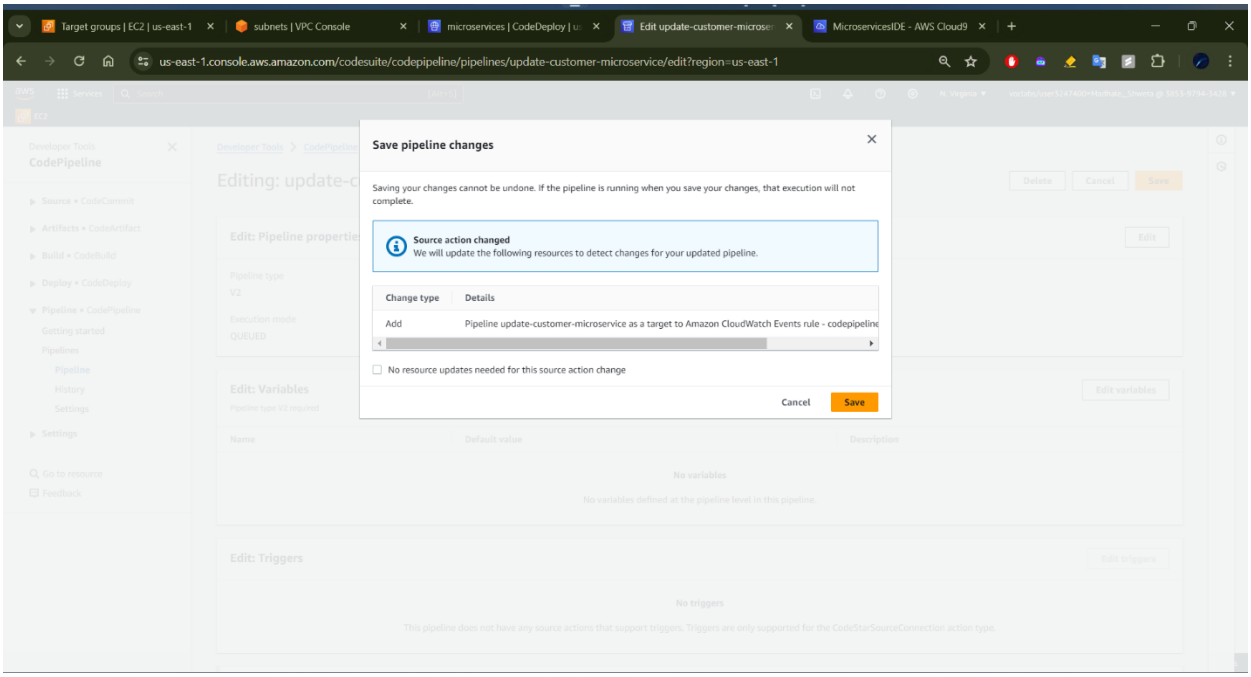
- Edit the update-customer-microservice pipeline to add another source.
    - In the Edit: Source section, choose Edit stage, then add an action with these details:
        - Action name: Image
        - Action provider: Amazon ECR
        - Repository name: customer
        - Image tag: latest
        - Output artifacts: image-customer
- Edit the deploy action of the update-customer-microservice pipeline.
    - Edit the update-customer-microservice pipeline
        - In the Edit: Deploy section, choose Edit stage, then add an input artifact as described below:
        - On the Deploy Amazon ECS (Blue/Green) card, choose the edit (pencil) icon.
        - Under Input artifacts, choose Add and then choose image-customer.
        - Note: You should now have SourceArtifact and image-customer as listed input artifacts.
        - Under Dynamically update task definition image, for Input artifact with image details, choose image-customer.
        - For Placeholder text in the task definition, enter IMAGE1_NAME

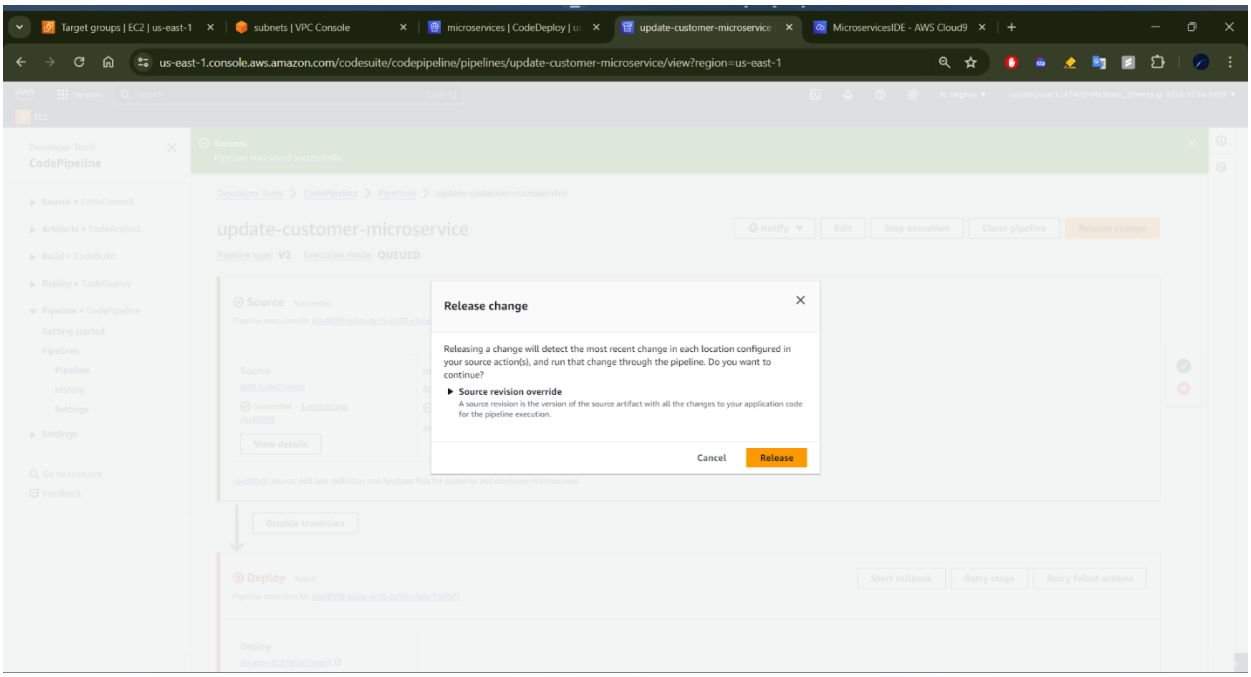

Edit action
Action name
Choose a name for your action
Image
No more than 100 characters
Action provider
Amazon ECR
Repository name
Choose an Amazon ECR repository as the source location.
customer
Image tag - optional
Choose the image tag that triggers your pipeline when a change occurs in the image repository.
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
Cancel
Done

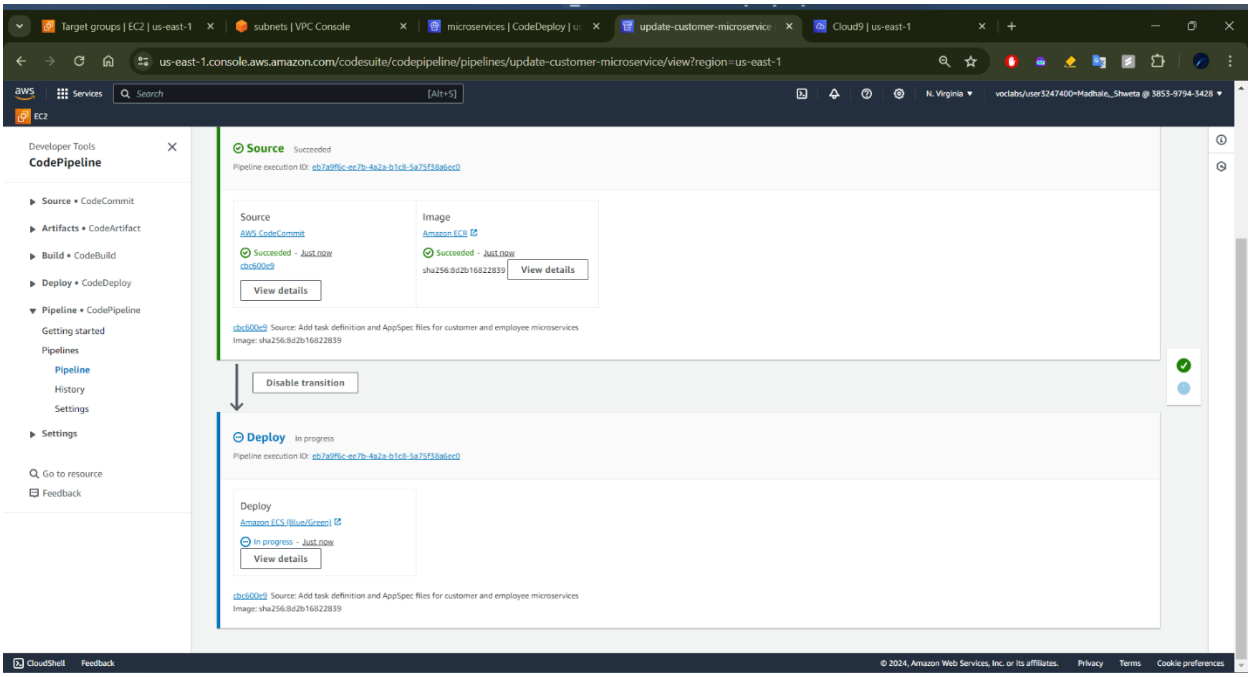

Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Edit: Triggers
Edit triggers
No triggers
This pipeline does not have any source actions that support triggers. Triggers are only supported for the CodeStarSourceConnection action type.
Edit: Source
Edit stage
Source
AWS CodeCommit
Image
Amazon ECR
Add stage
Edit: Deploy
Edit stage
Deploy
Amazon ECS (Blue/Green)
Configure automatic rollback on stage failure
Add stage
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

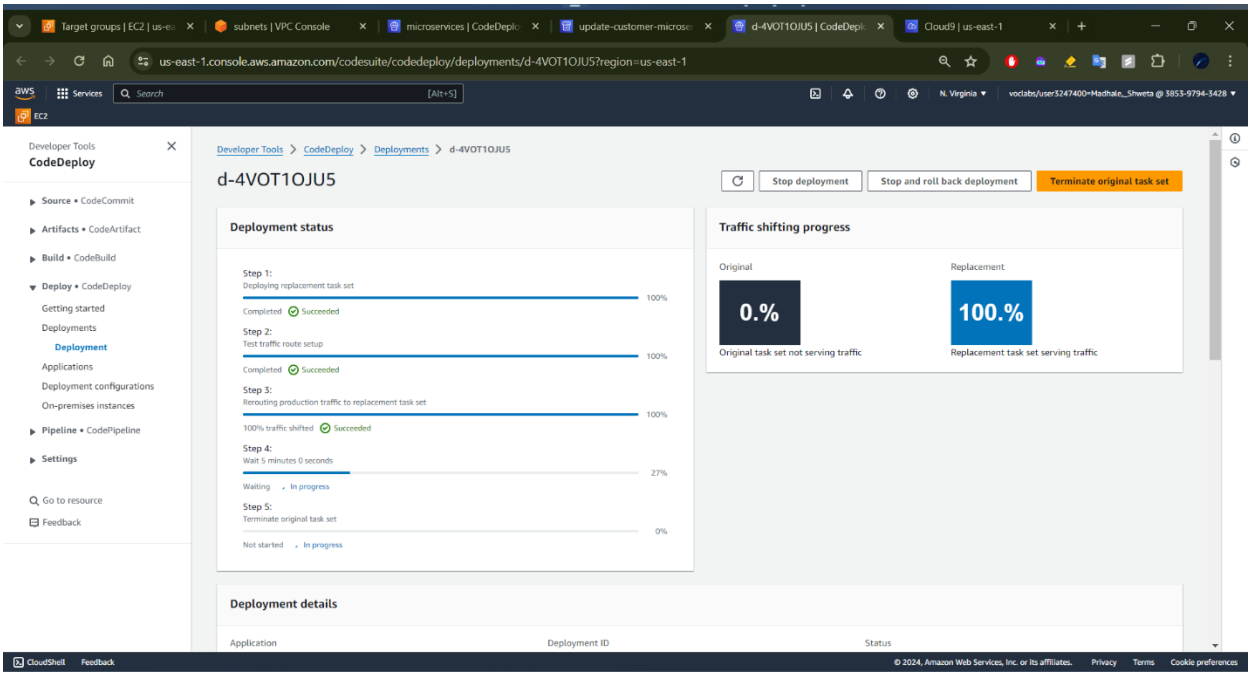

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Edit update-customer-microse
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/edit?region=us-east-1
Edit: Triggers
Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
Amazon ECS (Blue/Green)
Region
US East (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
image-customer
Remove
Add
No more than 100 characters
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
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.
microservices-customer
Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.
SourceArtifact
taskdef-customer.json
The default path is taskdef.json.
AWS CodeDeploy AppSpec file

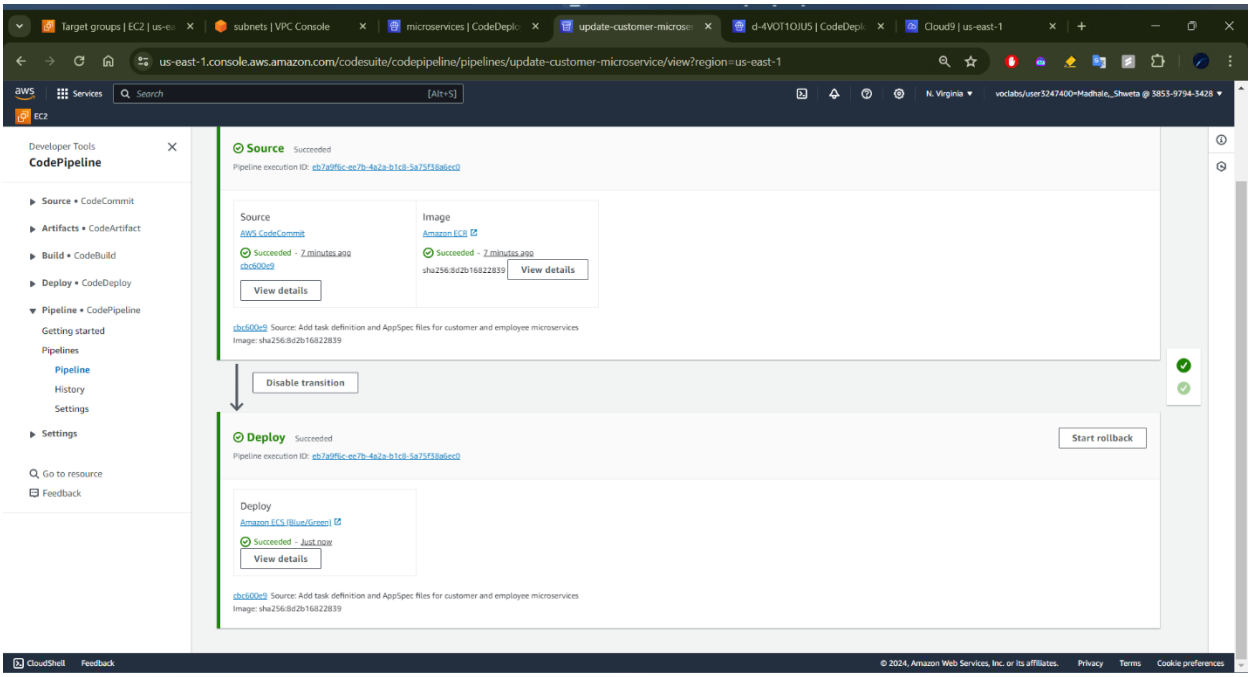

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Edit update-customer-microse
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/edit?region=us-east-1
Add
No more than 100 characters
AWS CodeDeploy application name
Choose one of your existing applications, or create a new one in AWS CodeDeploy.
microservices
Create application
AWS CodeDeploy deployment group
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.
microservices-customer
Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.
SourceArtifact
taskdef-customer.json
The default path is taskdef.json.
AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.
SourceArtifact
appspec-customer.yaml
Dynamically update task definition image - optional
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.
Input artifact with image details
image-customer
Placeholder text in the task definition
IMAGE1_NAME
Remove
Add
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
DeployVariables
Cancel
Done

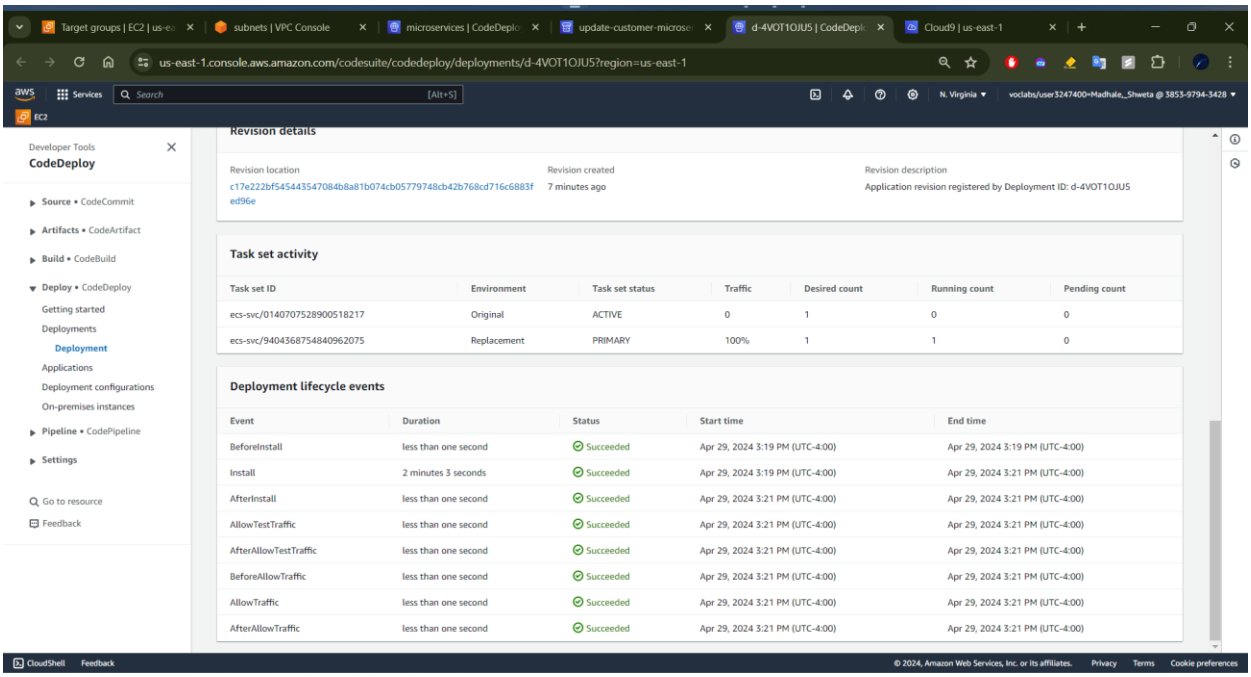

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
Edit update-customer-microse
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/edit?region=us-east-1
Save pipeline changes
Saving your changes cannot be undone. If the pipeline is running when you save your changes, that execution will not complete.
Source action changed
We will update the following resources to detect changes for your updated pipeline.
Change type
Details
Add
Pipeline update-customer-microservice as a target to Amazon CloudWatch Events rule - codepipeline
No resource updates needed for this source action change
Cancel
Save


Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
update-customer-microservice
MicroservicesIDE - AWS Cloud9
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/view?region=us-east-1
Success
Pipeline was saved successfully.
update-customer-microservice
Release change
Releasing a change will detect the most recent change in each location configured in your source action(s), and run that change through the pipeline. Do you want to continue?
Source revision override
A source revision is the version of the source artifact with all the changes to your application code for the pipeline execution.
Cancel
Release

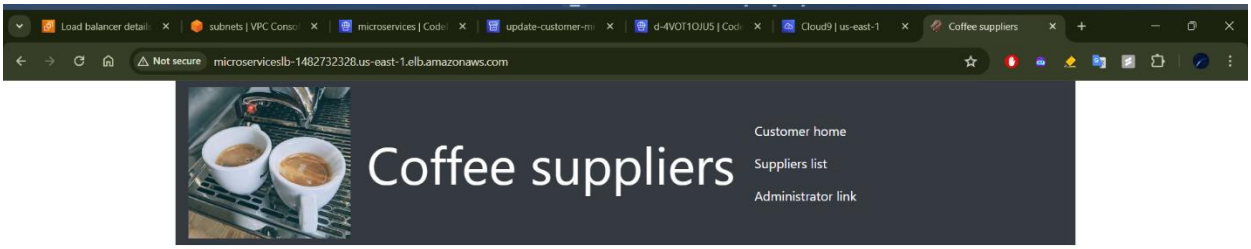

Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
update-customer-microservice
Cloud9 | us-east-1
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/view?region=us-east-1
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Source Succeeded
Pipeline execution ID: eb7a9f6c-ee7b-4a2a-b1c8-5a75f38a6ec0
Source
AWS CodeCommit
Succeeded - Just now
cbc600e9
View details
Image
Amazon ECR
Succeeded - Just now
sha256:8d2b16822839
View details
cbc600e9 Source: Add task definition and AppSpec files for customer and employee microservices
Image: sha256:8d2b16822839
Disable transition
Deploy In progress
Pipeline execution ID: eb7a9f6c-ee7b-4a2a-b1c8-5a75f38a6ec0
Deploy
Amazon ECS (Blue/Green)
In progress - Just now
View details
cbc600e9 Source: Add task definition and AppSpec files for customer and employee microservices
Image: sha256:8d2b16822839
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Target groups | EC2 | us-east-1
subnets | VPC Console
microservices | CodeDeploy
update-customer-microservice
d-4VOT1OJU5 | CodeDeploy
Cloud9 | us-east-1
us-east-1.console.aws.amazon.com/codesuite/codedeploy/deployments/d-4VOT1OJU5?region=us-east-1
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
Developer Tools > CodeDeploy > Deployments > d-4VOT1OJU5
d-4VOT1OJU5
Stop deployment
Stop and roll back deployment
Terminate original task set
Deployment status
Step 1:
Deploying replacement task set
100%
Completed Succeeded
Step 2:
Test traffic route setup
100%
Completed Succeeded
Step 3:
Rerouting production traffic to replacement task set
100%
100% traffic shifted Succeeded
Step 4:
Wait 5 minutes 0 seconds
27%
Waiting In progress
Step 5:
Terminate original task set
0%
Not started In progress
Traffic shifting progress
Original
0.%
Original task set not serving traffic
Replacement
100.%
Replacement task set serving traffic
Deployment details
Application
Deployment ID
Status
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

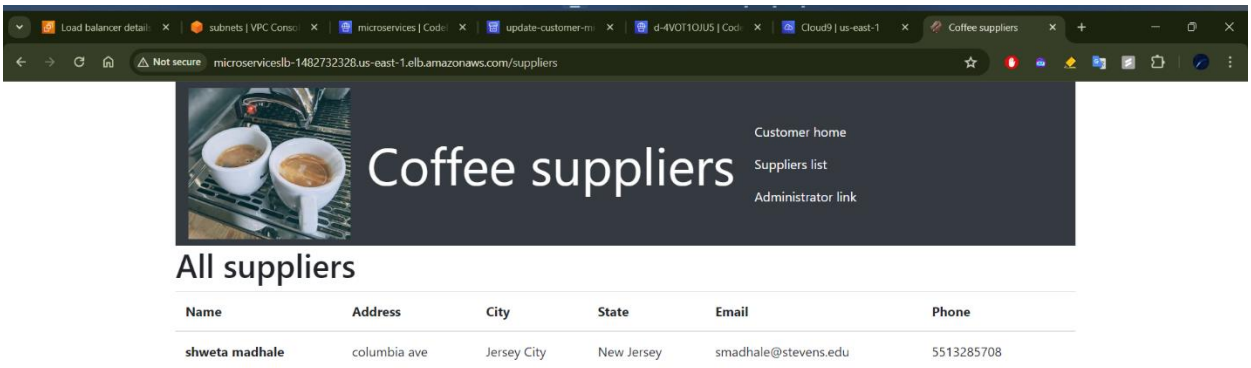


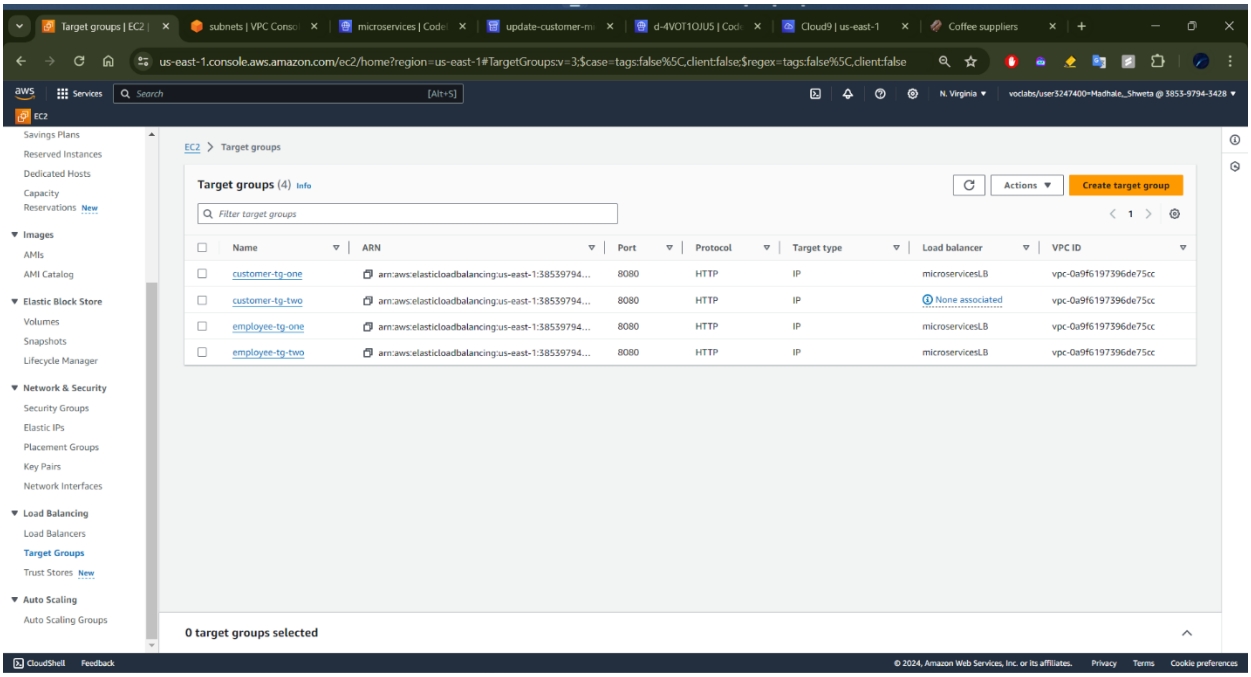


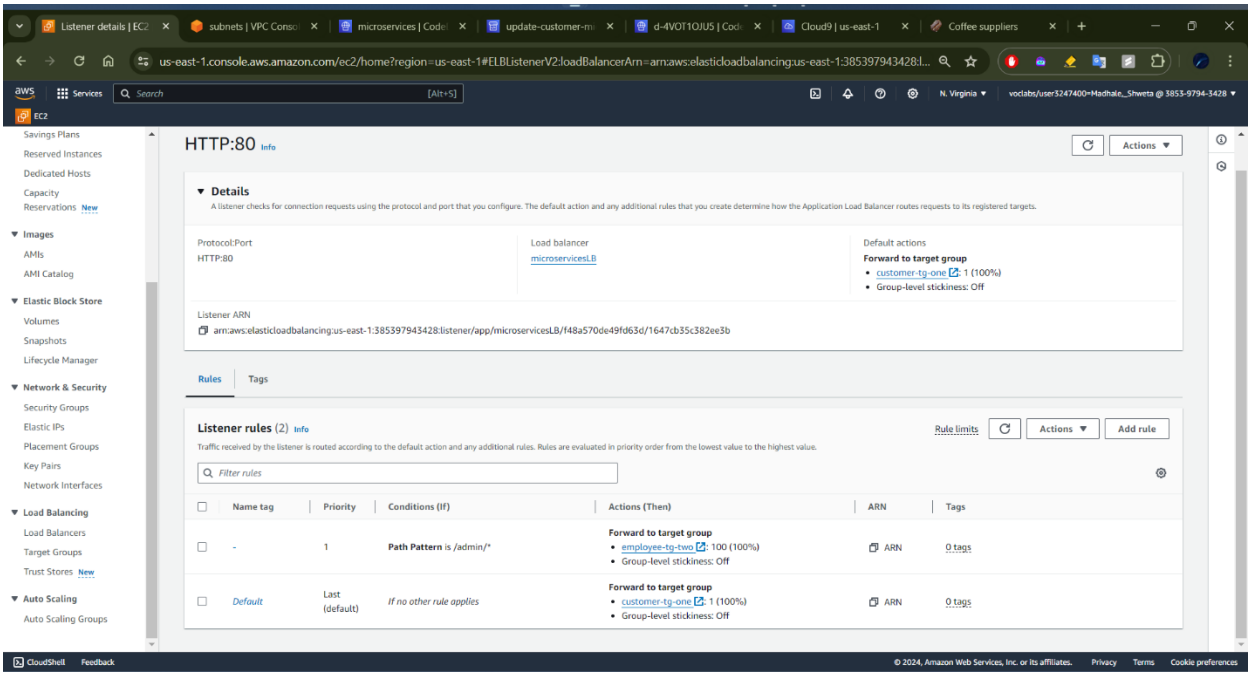
Task 8.3: Test the CI/CD pipeline for the customer microservice

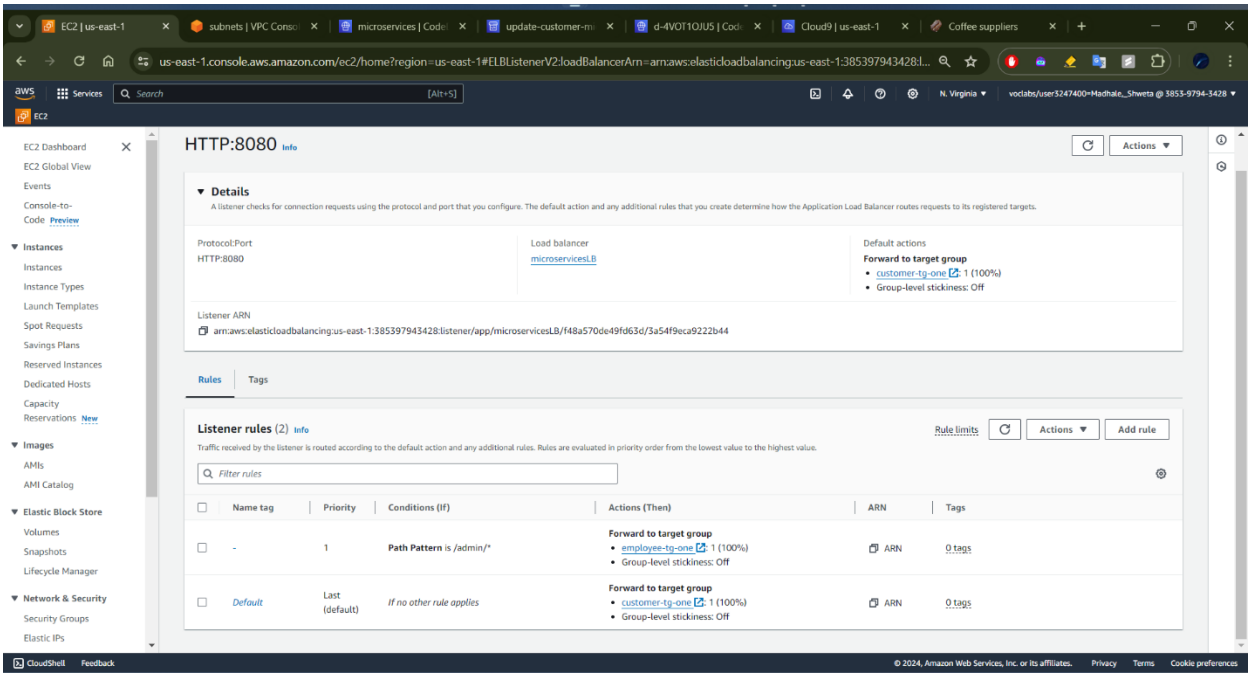


microserviceslb-1482732328.us-east-1.elb.amazonaws.com



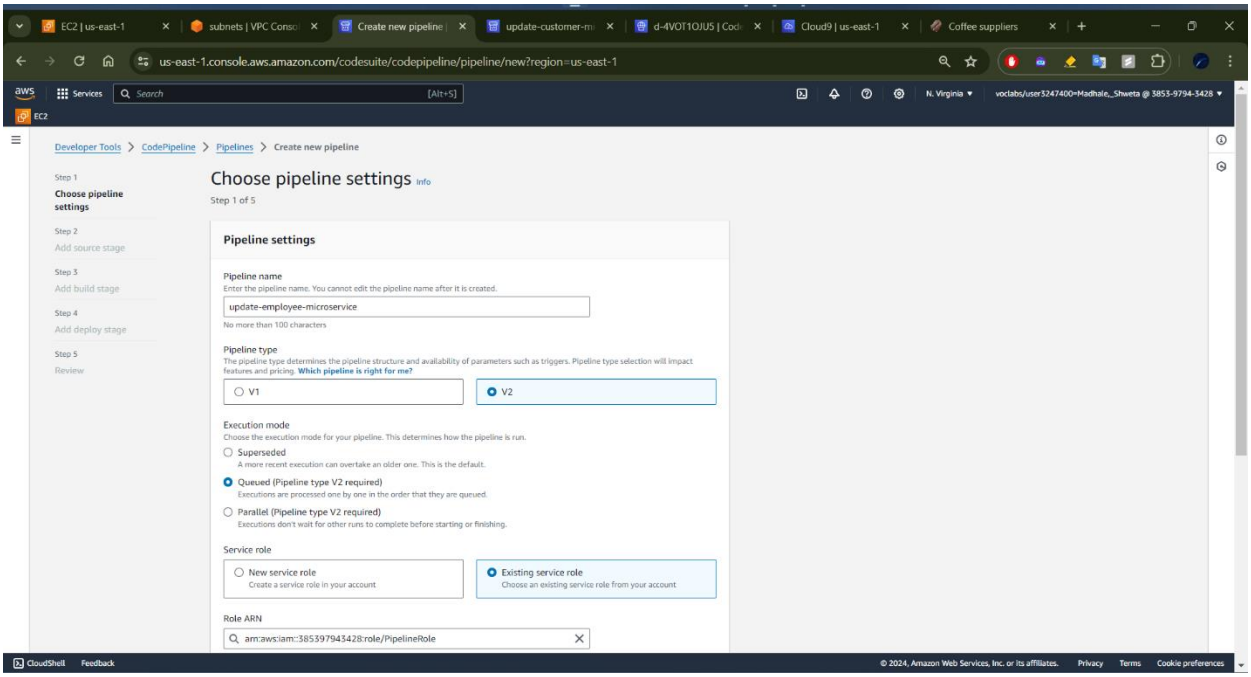

Target groups | EC2
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:v=3;$case=tags:false%5C,client:false;$regex=tags:false%5C,client:false
EC2 > Target groups
Target groups (4) Info
Actions
Create target group
Filter target groups
Name | ARN | Port | Protocol | Target type | Load balancer | VPC ID
customer-tg-one | arn:aws:elasticloadbalancing:us-east-1:38539794... | 8080 | HTTP | IP | microservicesLB | vpc-0a9f6197396de75cc
customer-tg-two | arn:aws:elasticloadbalancing:us-east-1:38539794... | 8080 | HTTP | IP | None associated | vpc-0a9f6197396de75cc
employee-tg-one | arn:aws:elasticloadbalancing:us-east-1:38539794... | 8080 | HTTP | IP | microservicesLB | vpc-0a9f6197396de75cc
employee-tg-two | arn:aws:elasticloadbalancing:us-east-1:38539794... | 8080 | HTTP | IP | microservicesLB | vpc-0a9f6197396de75cc
0 target groups selected

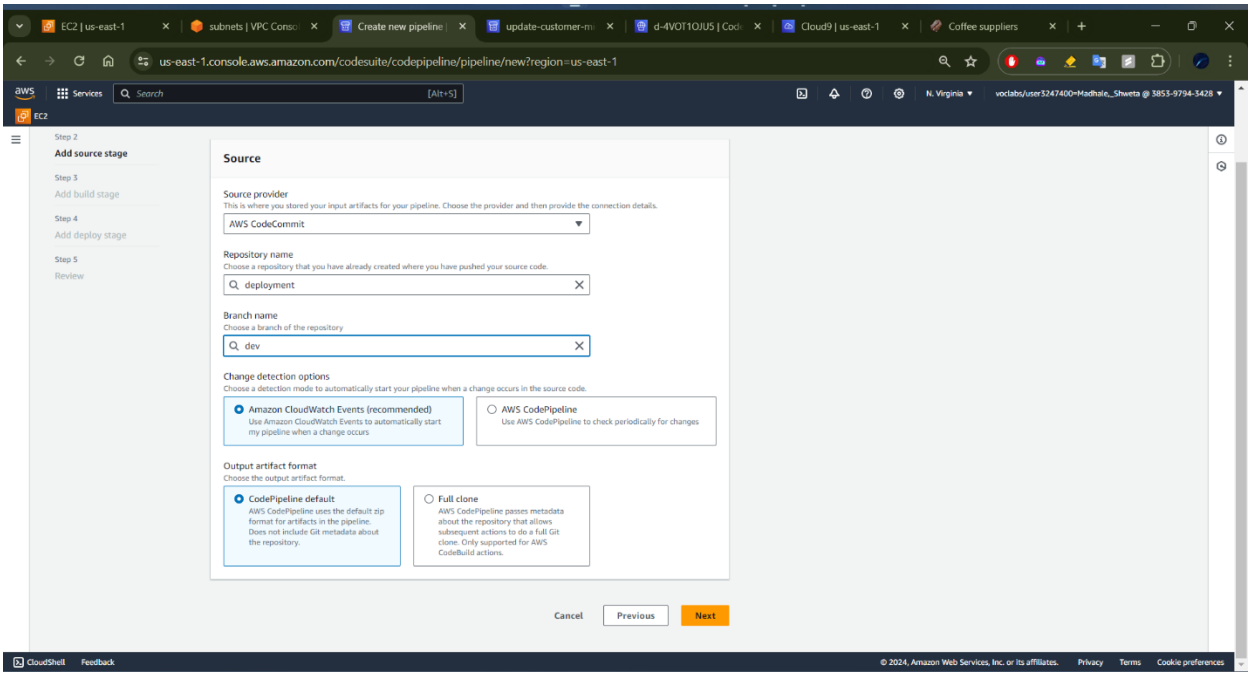

Listener details | EC2
HTTP:80 Info
Actions
Details
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.
Protocol:Port
HTTP:80
Load balancer
microservicesLB
Default actions
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
Listener ARN
arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd63d/1647cb35c382ee3b
Rules
Tags
Listener rules (2) Info
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.
Rule limits
Actions
Add rule
Filter rules
Name tag | Priority | Conditions (If) | Actions (Then) | ARN | Tags
- | 1 | Path Pattern is /admin/* | Forward to target group: employee-tg-two: 100 (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

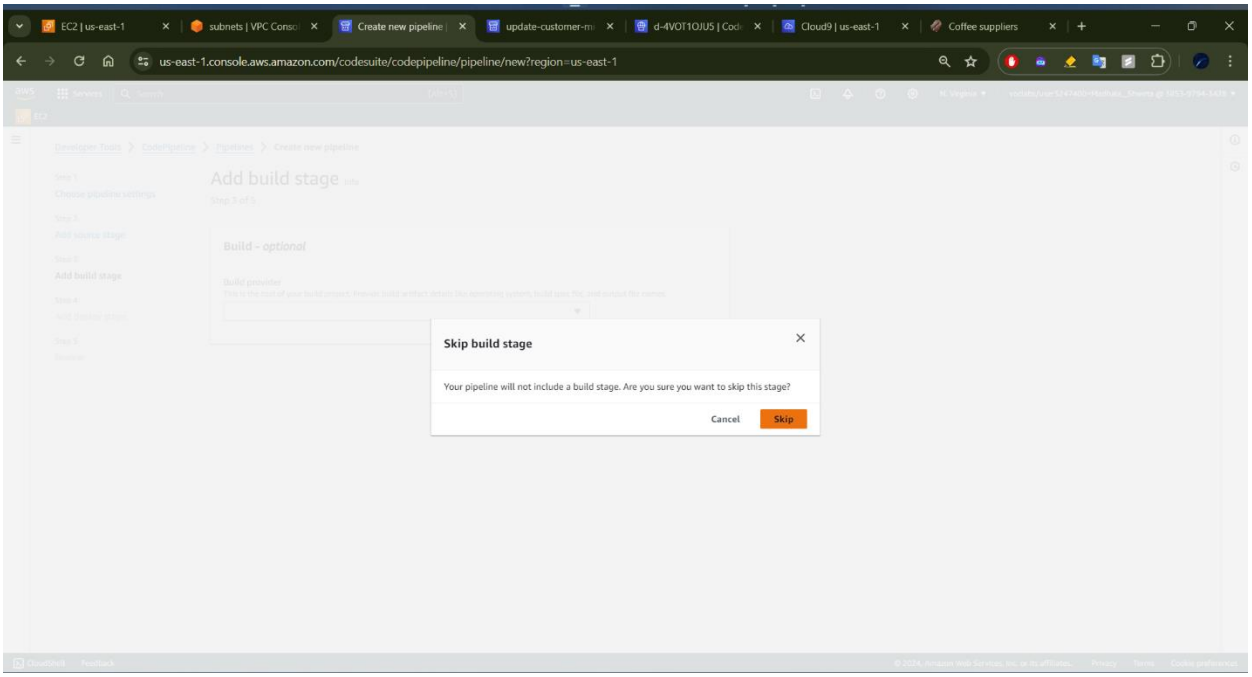


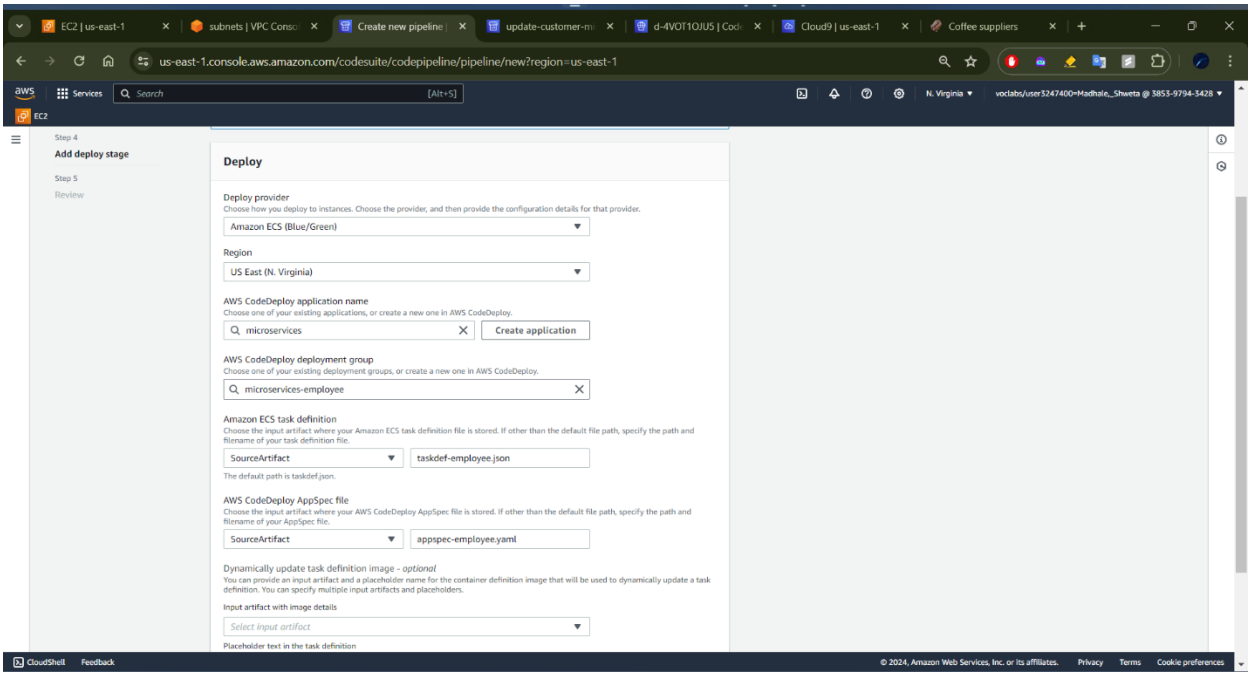
Task 8.4: Create a pipeline for the employee microservice

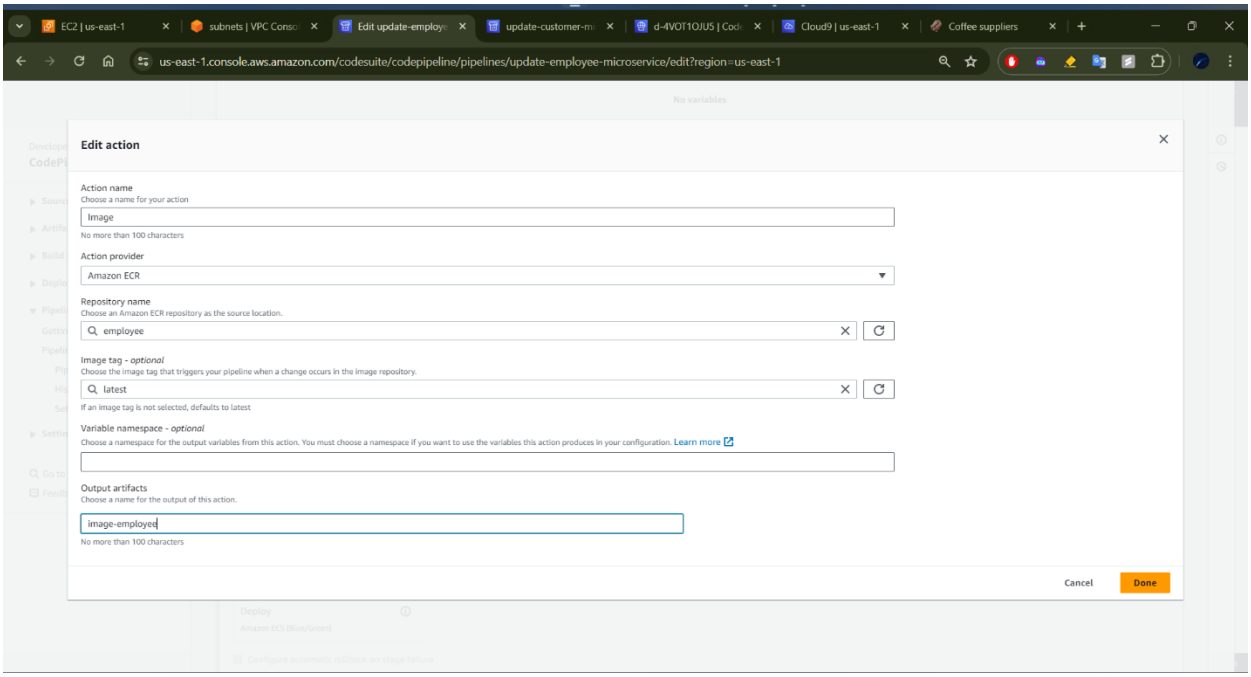
- Create a pipeline for the employee microservice with the following specifications:
  - Pipeline name: update-employee-microservice
  - Role ARN: PipelineRole
  - Source provider: AWS CodeCommit
  - Repository name: deployment
  - Branch name: dev
  - Deploy provider: Amazon ECS (Blue/Green)
  - AWS CodeDeploy application name: microservices
  - AWS CodeDeploy deployment group: microservices-employee
  - Amazon ECS task definition: SourceArtifact
  - Path: taskdef-employee.json
  - AWS CodeDeploy AppSpec file: SourceArtifact
  - Path: appspec-employee.yaml
- Add another source to the employee microservice pipeline. Add an action with the following details:
  - Action name: Image
  - Action provider: Amazon ECR
  - Repository name: employee
  - Image tag: latest
  - Output artifacts: image-employee
  - Edit the Amazon ECS (Blue/Green) action in the deploy stage:
- Add another input artifact and choose image-employee.
  - Under Dynamically update task definition image, for Input artifact with image details, choose image-employee.
  - For Placeholder text in the task definition, enter IMAGE1_NAME

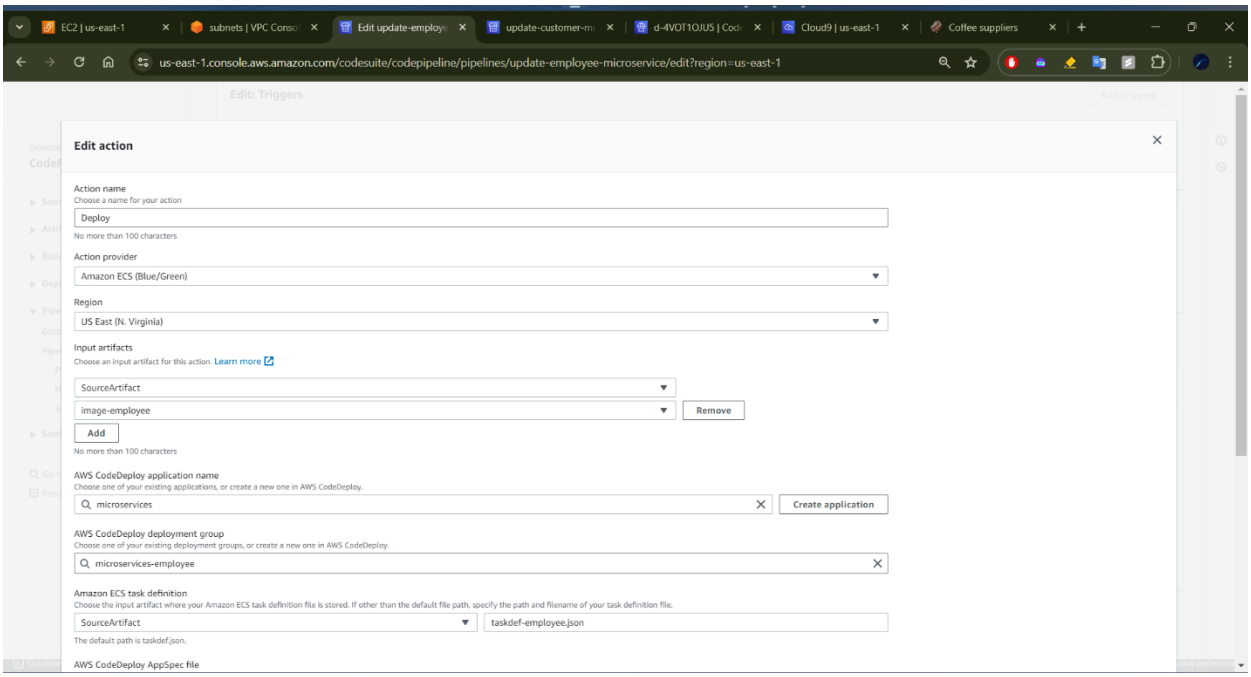

Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Choose pipeline settings Info
Step 1 of 5
Pipeline settings
Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
update-employee-microservice
No more than 100 characters
Pipeline type
The pipeline type determines the pipeline structure and availability of parameters such as triggers. Pipeline type selection will impact features and pricing. Which pipeline is right for me?
V1
V2
Execution mode
Choose the execution mode for your pipeline. This determines how the pipeline is run.
Superseded
A more recent execution can overtake an older one. This is the default.
Queued (Pipeline type V2 required)
Executions are processed one by one in the order that they are queued.
Parallel (Pipeline type V2 required)
Executions don't wait for other runs to complete before starting or finishing.
Service role
New service role
Create a service role in your account
Existing service role
Choose an existing service role from your account
Role ARN
arn:aws:iam::385397943428:role/PipelineRole

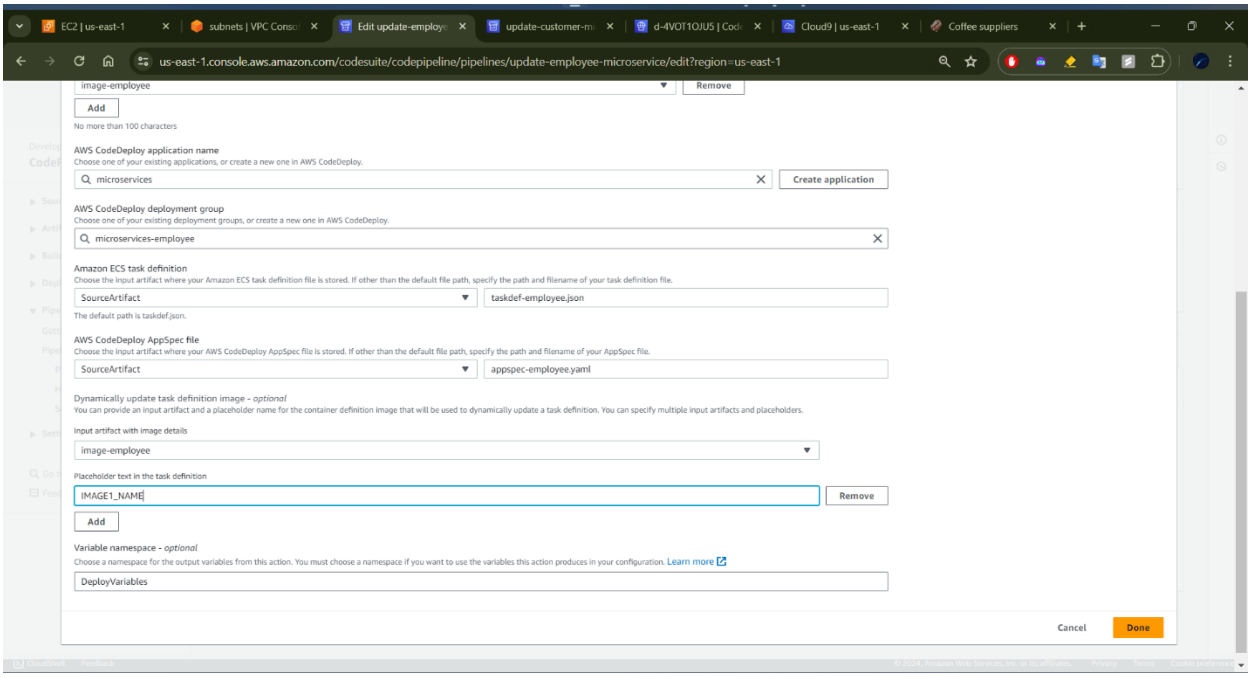

Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
AWS CodeCommit
Repository name
Choose a repository that you have already created where you have pushed your source code.
deployment
Branch name
Choose a branch of the repository
dev
Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.
Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs
AWS CodePipeline
Use AWS CodePipeline to check periodically for changes
Output artifact format
Choose the output artifact format.
CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.
Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.
Cancel
Previous
Next

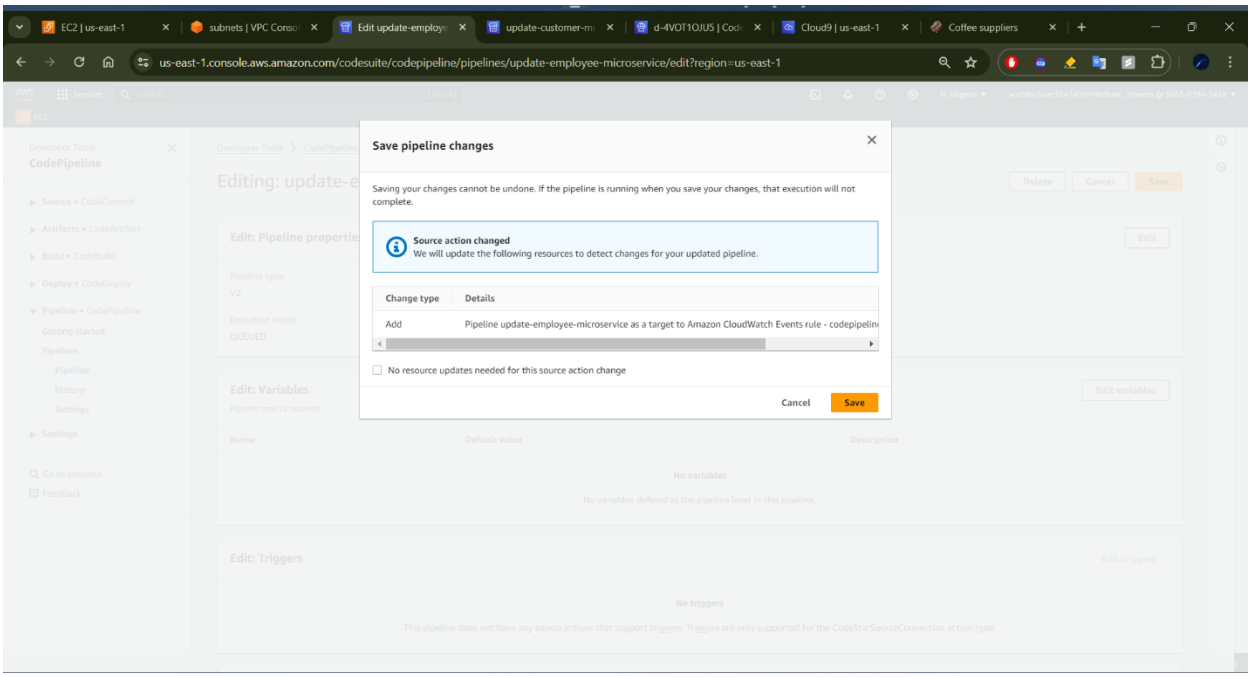

Add build stage
Build - optional
Skip build stage
Your pipeline will not include a build stage. Are you sure you want to skip this stage?
Cancel
Skip


Add deploy stage
Deploy
Deploy provider
Amazon ECS (Blue/Green)
Region
US East (N. Virginia)
AWS CodeDeploy application name
microservices
Create application
AWS CodeDeploy deployment group
microservices-employee
Amazon ECS task definition
SourceArtifact
taskdef-employee.json
AWS CodeDeploy AppSpec file
SourceArtifact
appspec-employee.yaml
Dynamically update task definition image - optional
Input artifact with image details
Select input artifact
CloudShell
Feedback

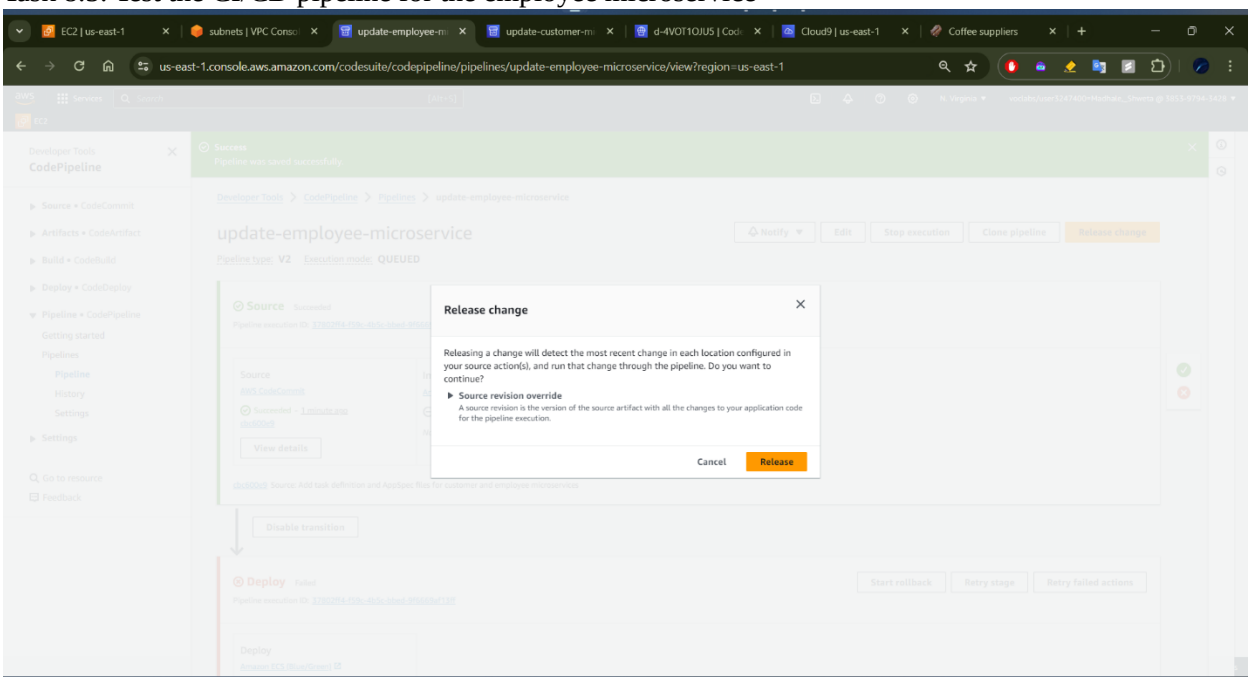

EC2 | us-east-1
subnets | VPC Conso
Edit update-employe
update-customer-m
d-4VOT1OJU5 | Cod
Cloud9 | us-east-1
Coffee suppliers
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-employee-microservice/edit?region=us-east-1
Edit action
Action name
Choose a name for your action
Image
No more than 100 characters
Action provider
Amazon ECR
Repository name
Choose an Amazon ECR repository as the source location.
employee
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-employee
No more than 100 characters
Cancel
Done

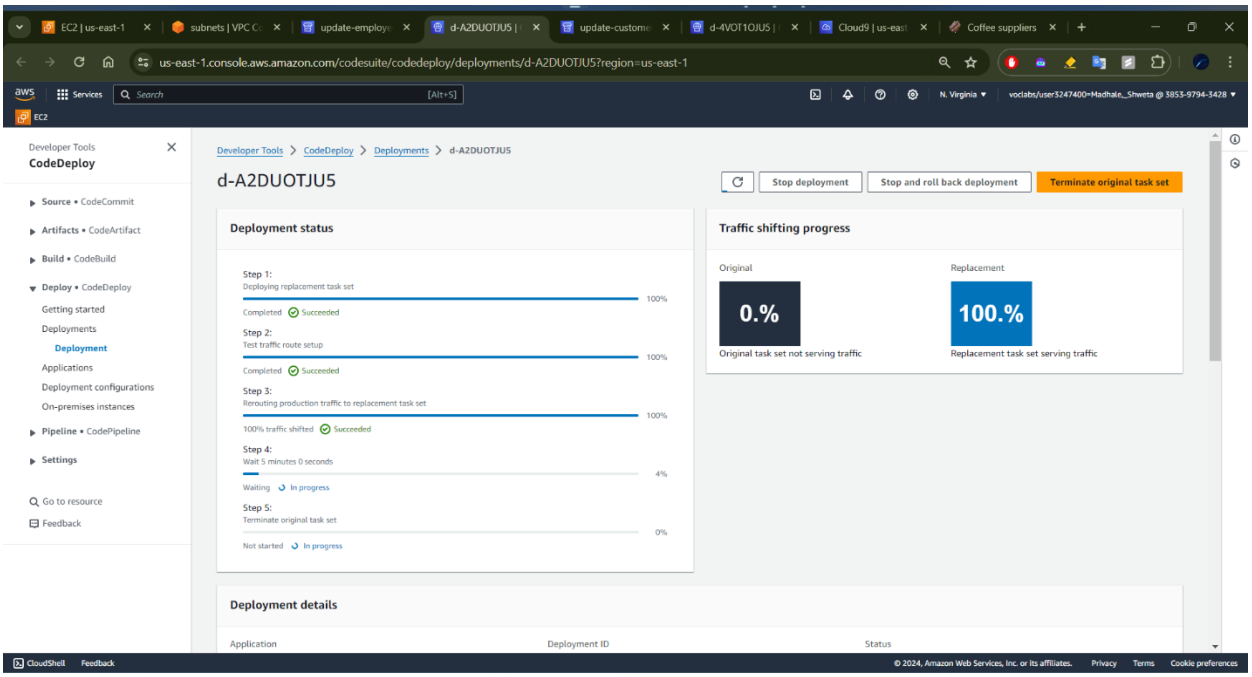

EC2 | us-east-1
subnets | VPC Conso
Edit update-employe
update-customer-m
d-4VOT1OJU5 | Cod
Cloud9 | us-east-1
Coffee suppliers
us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-employee-microservice/edit?region=us-east-1
Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
Amazon ECS (Blue/Green)
Region
US East (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
image-employee
Remove
Add
No more than 100 characters
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
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.
microservices-employee
Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.
SourceArtifact
taskdef-employee.json
The default path is taskdef.json.
AWS CodeDeploy AppSpec file

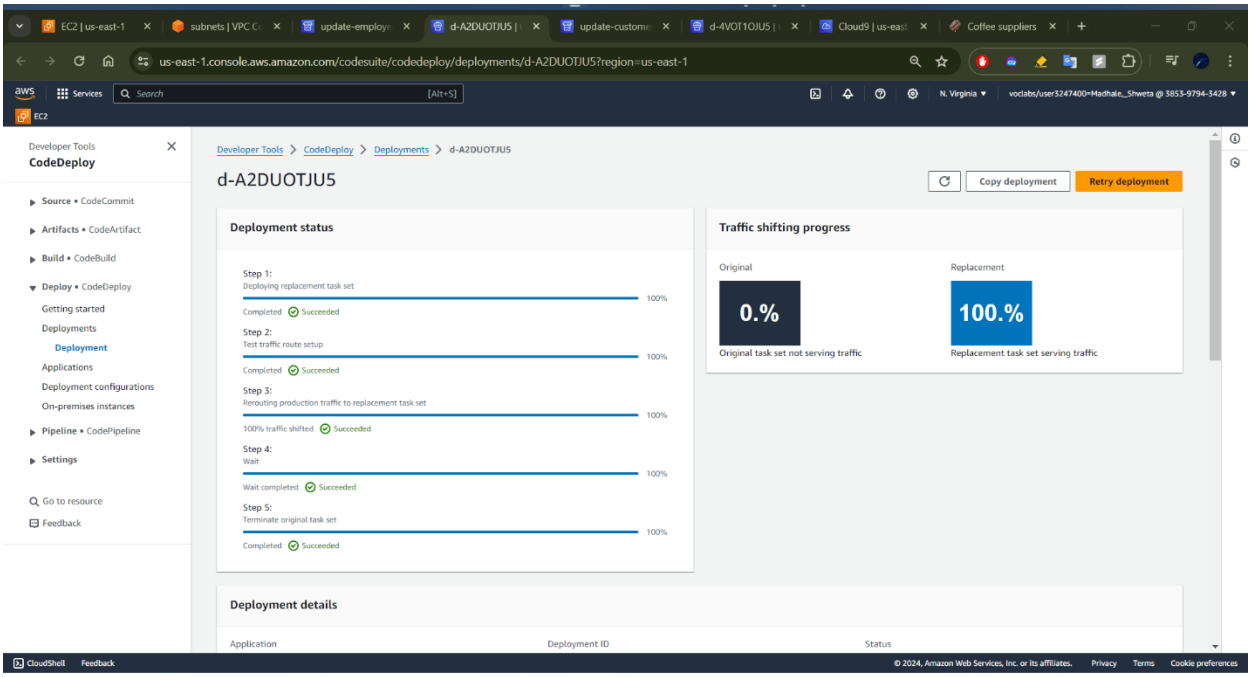

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-employee-microservice/edit?region=us-east-1
image-employee
Remove
Add
No more than 100 characters
AWS CodeDeploy application name
Choose one of your existing applications, or create a new one in AWS CodeDeploy.
microservices
Create application
AWS CodeDeploy deployment group
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.
microservices-employee
Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.
SourceArtifact
taskdef-employee.json
The default path is taskdef.json.
AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.
SourceArtifact
appspec-employee.yaml
Dynamically update task definition image - optional
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.
Input artifact with image details
image-employee
Placeholder text in the task definition
IMAGE1_NAME
Remove
Add
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
DeployVariables
Cancel
Done

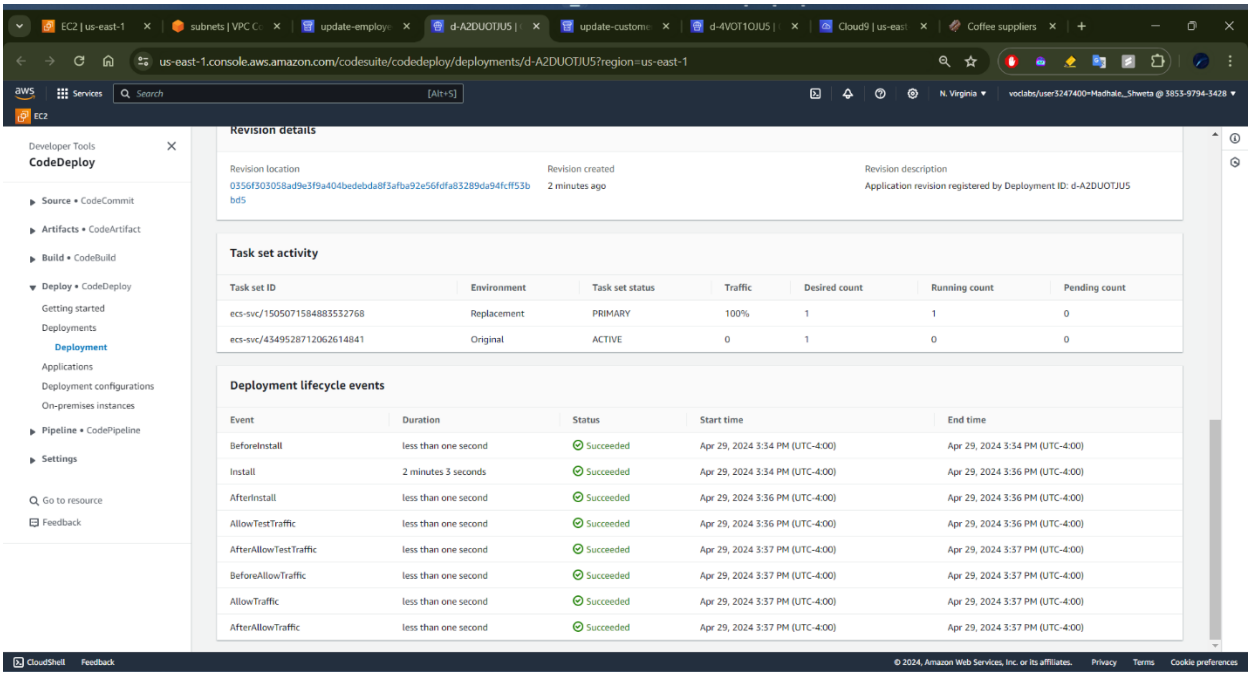

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-employee-microservice/edit?region=us-east-1
Save pipeline changes
Saving your changes cannot be undone. If the pipeline is running when you save your changes, that execution will not complete.
Source action changed
We will update the following resources to detect changes for your updated pipeline.
Change type
Details
Add
Pipeline update-employee-microservice as a target to Amazon CloudWatch Events rule - codepipelin
No resource updates needed for this source action change
Cancel
Save

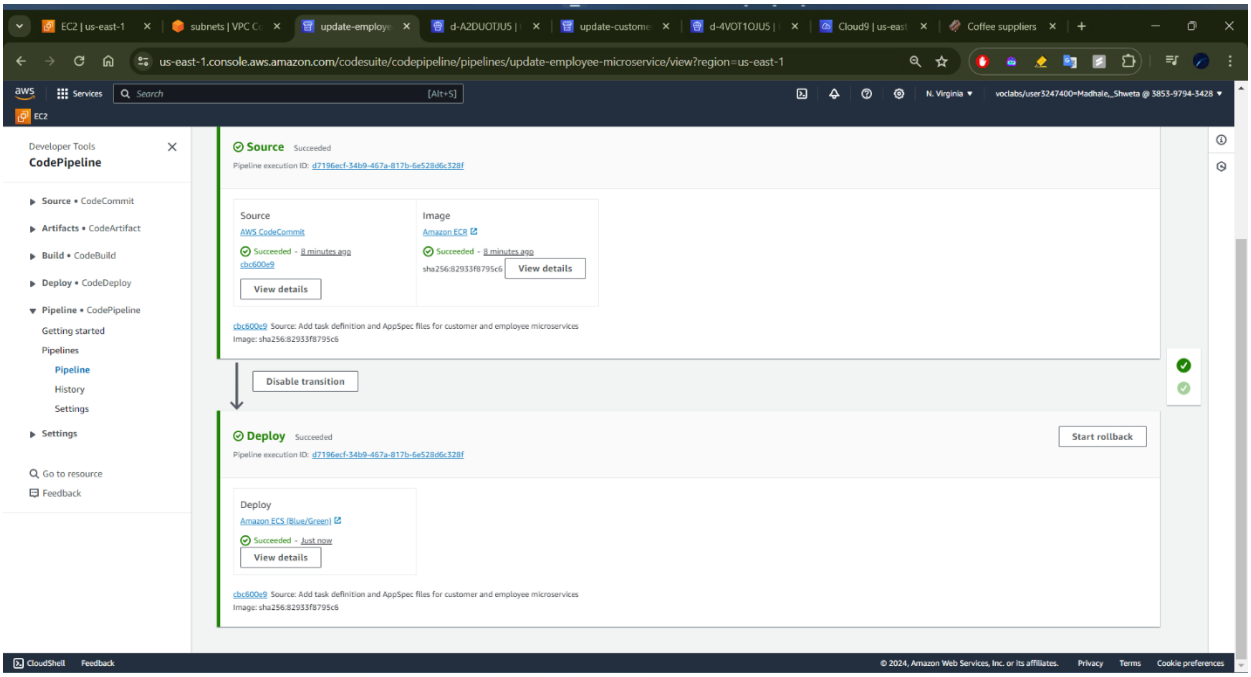
Task 8.5: Test the CI/CD pipeline for the employee microservice

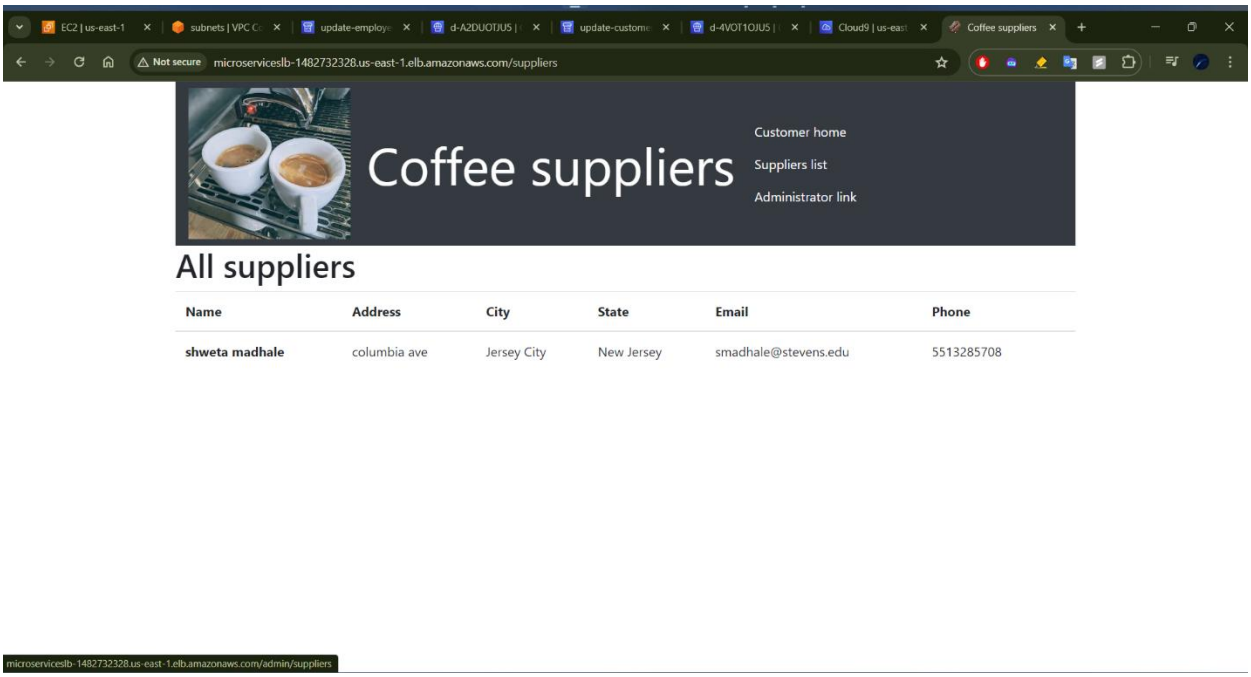


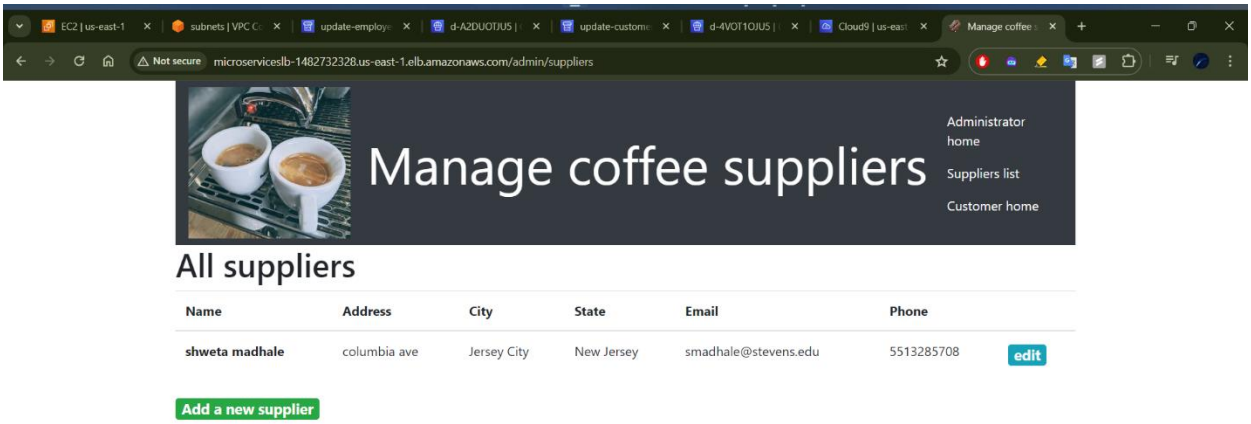


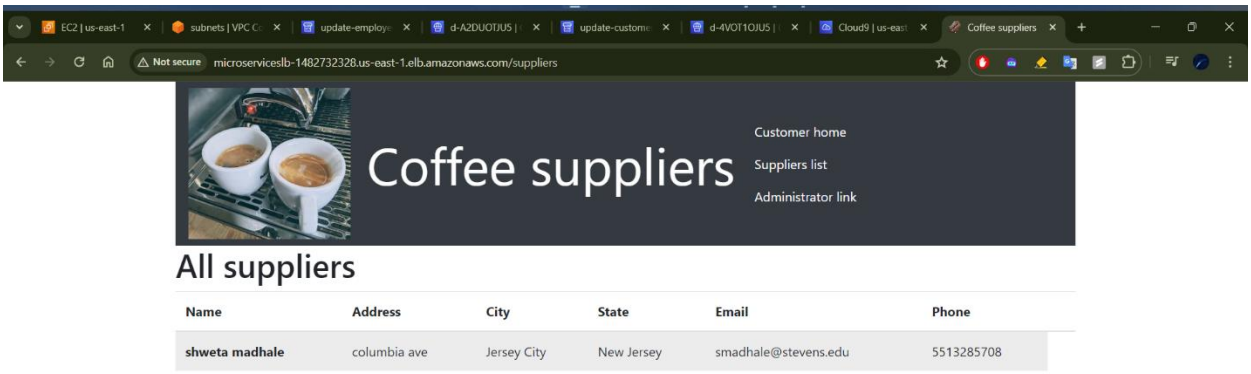

Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Deployment
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Developer Tools > CodeDeploy > Deployments > d-A2DUOTJU5
d-A2DUOTJU5
Copy deployment
Retry deployment
Deployment status
Step 1:
Deploying replacement task set
100%
Completed Succeeded
Step 2:
Test traffic route setup
100%
Completed Succeeded
Step 3:
Rerouting production traffic to replacement task set
100%
100% traffic shifted Succeeded
Step 4:
Wait
100%
Wait completed Succeeded
Step 5:
Terminate original task set
100%
Completed Succeeded
Traffic shifting progress
Original
0.%
Original task set not serving traffic
Replacement
100.%
Replacement task set serving traffic
Deployment details
Application
Deployment ID
Status
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Revision details
Revision location
0356f303058ad9e3f9a404bedebda8f3afba92e56fdfa83289da94fcff53bbd5
Revision created
2 minutes ago
Revision description
Application revision registered by Deployment ID: d-A2DUOTJU5
Task set activity
Task set ID | Environment | Task set status | Traffic | Desired count | Running count | Pending count
ecs-svc/1505071584883532768 | Replacement | PRIMARY | 100% | 1 | 1 | 0
ecs-svc/4349528712062614841 | Original | ACTIVE | 0 | 1 | 0 | 0
Deployment lifecycle events
Event | Duration | Status | Start time | End time
BeforeInstall | less than one second | Succeeded | Apr 29, 2024 3:34 PM (UTC-4:00) | Apr 29, 2024 3:34 PM (UTC-4:00)
Install | 2 minutes 3 seconds | Succeeded | Apr 29, 2024 3:34 PM (UTC-4:00) | Apr 29, 2024 3:36 PM (UTC-4:00)
AfterInstall | less than one second | Succeeded | Apr 29, 2024 3:36 PM (UTC-4:00) | Apr 29, 2024 3:36 PM (UTC-4:00)
AllowTestTraffic | less than one second | Succeeded | Apr 29, 2024 3:36 PM (UTC-4:00) | Apr 29, 2024 3:36 PM (UTC-4:00)
AfterAllowTestTraffic | less than one second | Succeeded | Apr 29, 2024 3:37 PM (UTC-4:00) | Apr 29, 2024 3:37 PM (UTC-4:00)
BeforeAllowTraffic | less than one second | Succeeded | Apr 29, 2024 3:37 PM (UTC-4:00) | Apr 29, 2024 3:37 PM (UTC-4:00)
AllowTraffic | less than one second | Succeeded | Apr 29, 2024 3:37 PM (UTC-4:00) | Apr 29, 2024 3:37 PM (UTC-4:00)
AfterAllowTraffic | less than one second | Succeeded | Apr 29, 2024 3:37 PM (UTC-4:00) | Apr 29, 2024 3:37 PM (UTC-4:00)

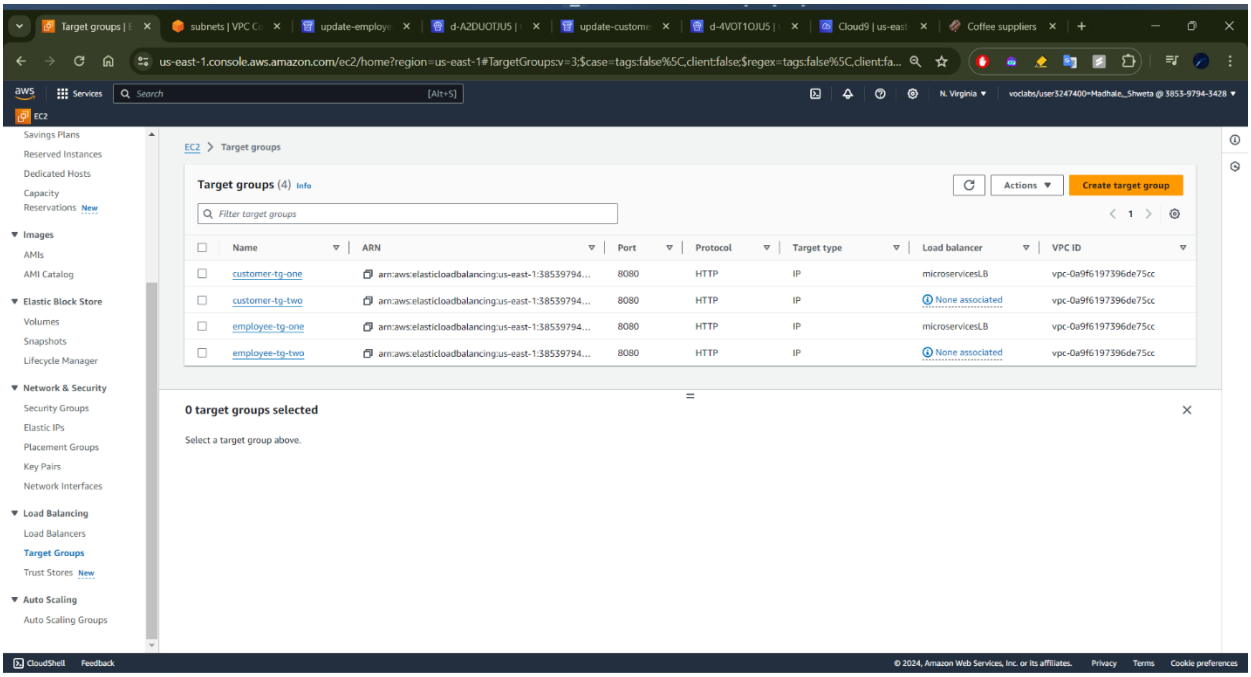

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-employee-microservice/view?region=us-east-1
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Source Succeeded
Pipeline execution ID: d7196ecf-34b9-467a-817b-6e528d6c328f
Source
AWS CodeCommit
Succeeded - 8 minutes ago
cbc600e9
View details
Image
Amazon ECR
Succeeded - 8 minutes ago
sha256:82933f8795c6
View details
cbc600e9 Source: Add task definition and AppSpec files for customer and employee microservices
Image: sha256:82933f8795c6
Disable transition
Deploy Succeeded
Pipeline execution ID: d7196ecf-34b9-467a-817b-6e528d6c328f
Start rollback
Deploy
Amazon ECS (Blue/Green)
Succeeded - Just now
View details
cbc600e9 Source: Add task definition and AppSpec files for customer and employee microservices
Image: sha256:82933f8795c6
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

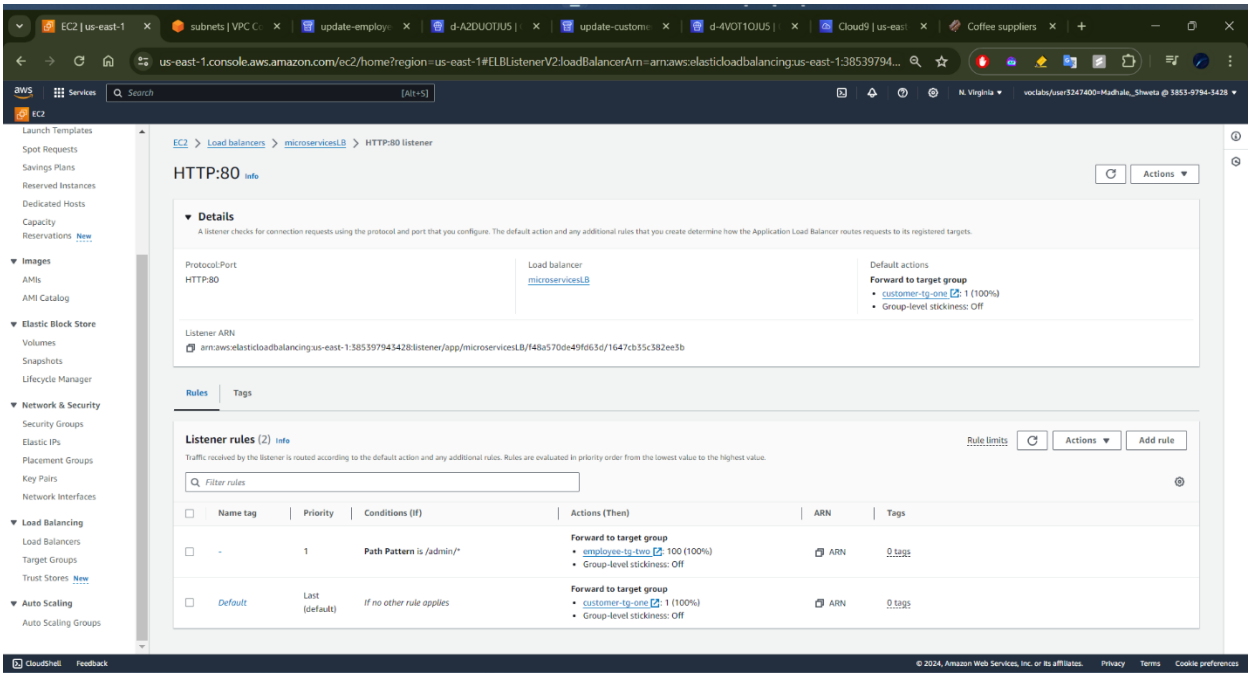

Not secure
microserviceslb-1482732328.us-east-1.elb.amazonaws.com/suppliers
Coffee suppliers
Customer home
Suppliers list
Administrator link
All suppliers
Name
Address
City
State
Email
Phone
shweta madhale
columbia ave
Jersey City
New Jersey
smadhale@stevens.edu
5513285708
microserviceslb-1482732328.us-east-1.elb.amazonaws.com/admin/suppliers

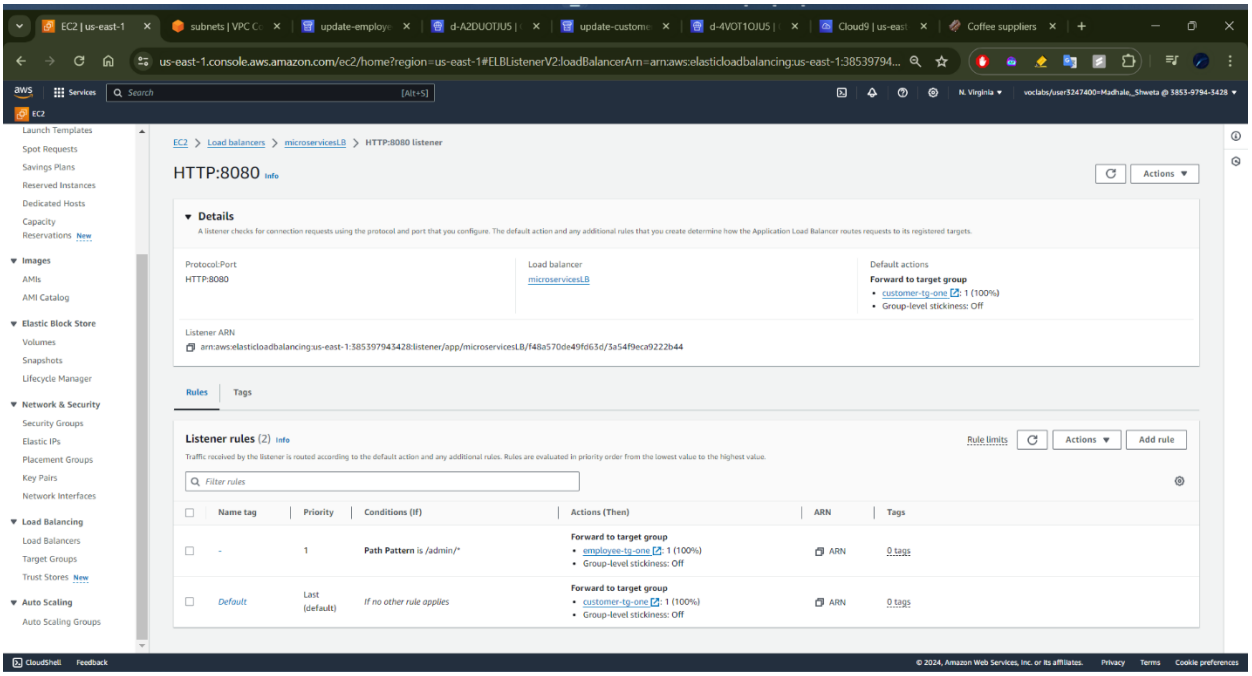




Task 8.6: Observe how CodeDeploy modified the load balancer listener rules

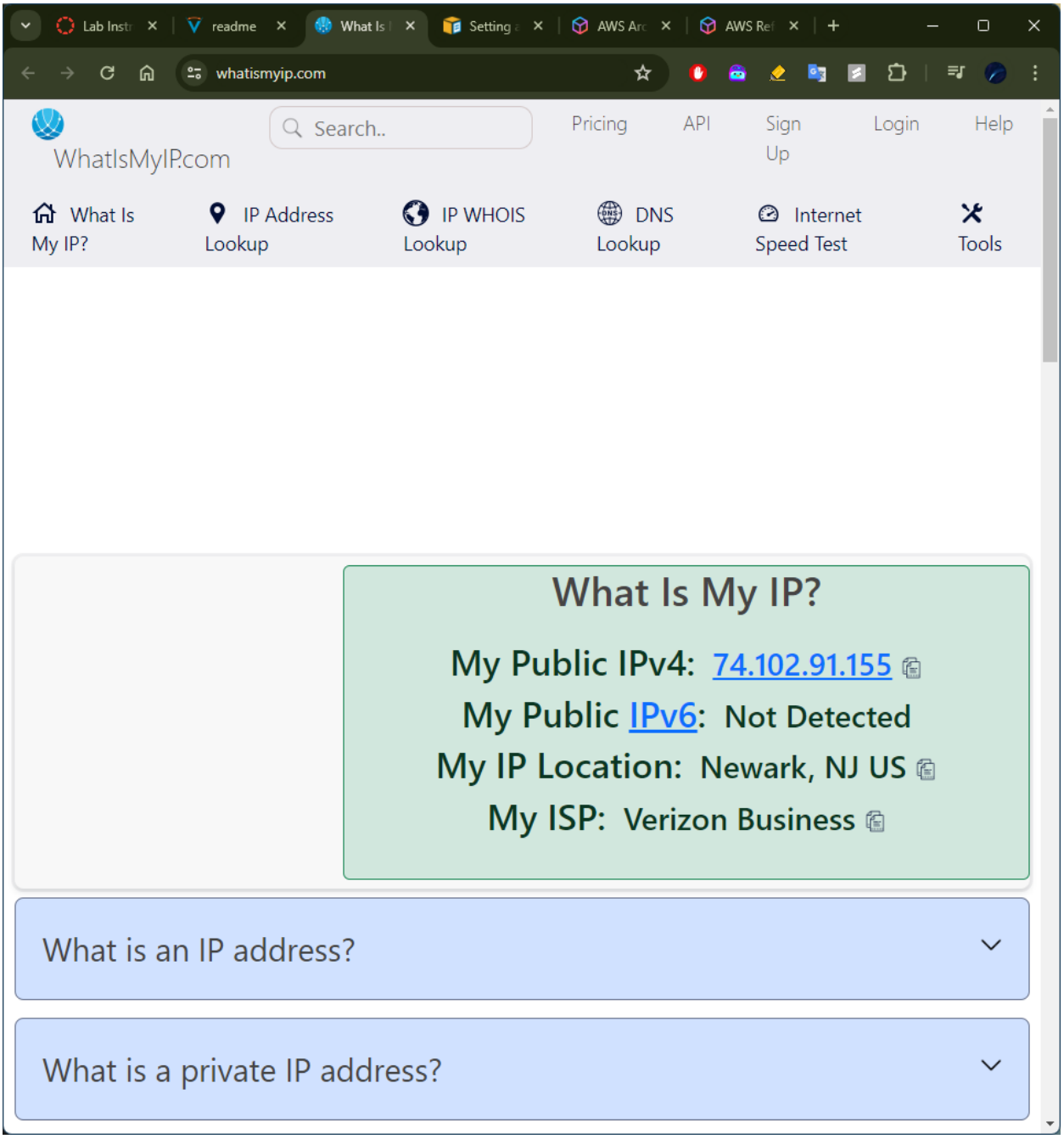

Target groups
Target groups (4) Info
Actions
Create target group
Filter target groups
Name
ARN
Port
Protocol
Target type
Load balancer
VPC ID
customer-tg-one
customer-tg-two
employee-tg-one
employee-tg-two
8080
HTTP
IP
microservicesLB
None associated
vpc-0a9f6197396de75cc
0 target groups selected
Select a target group above.

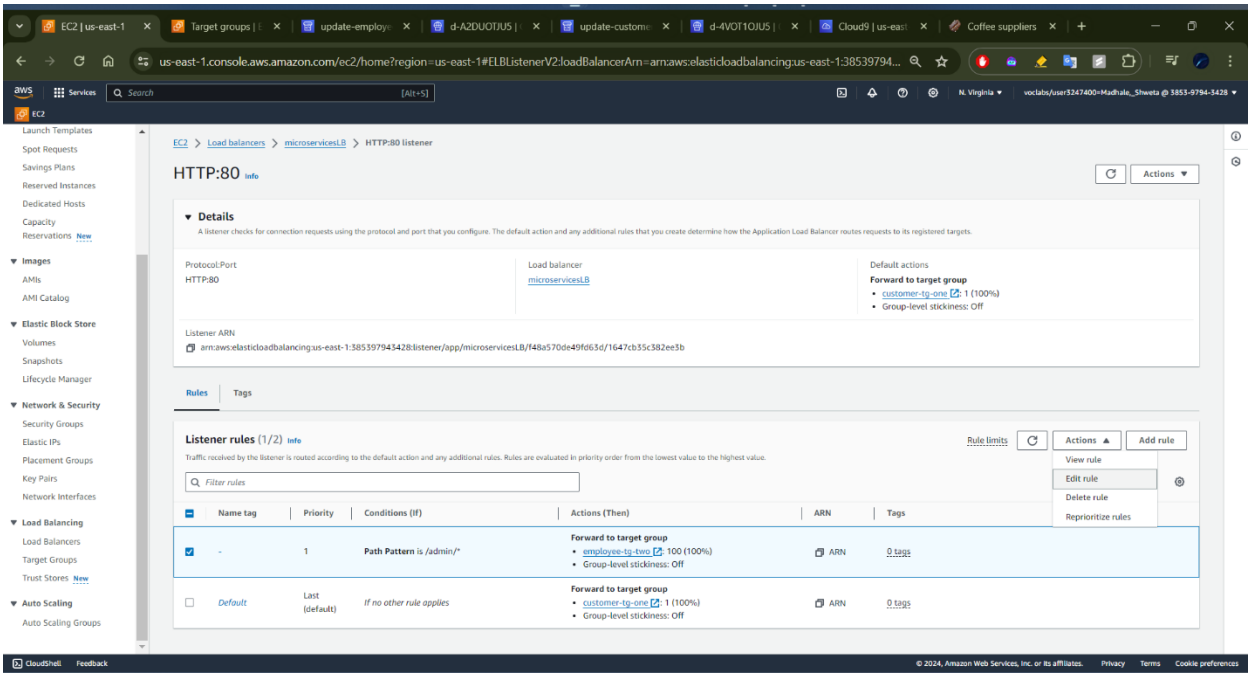


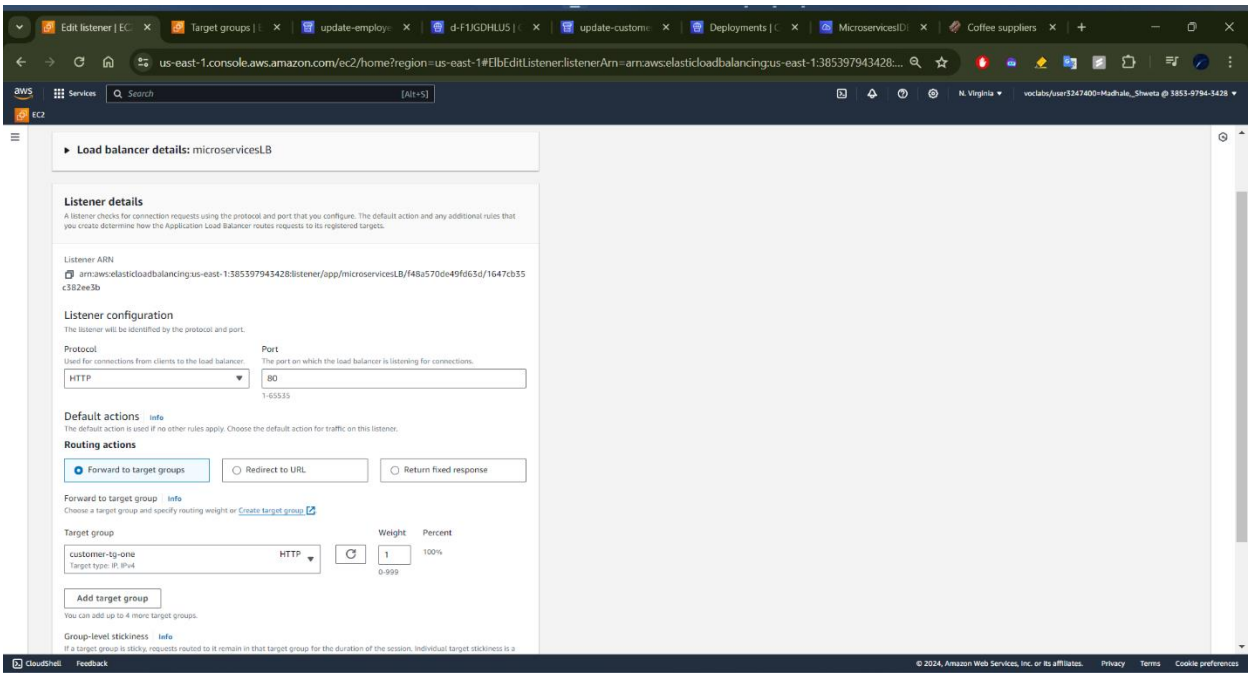


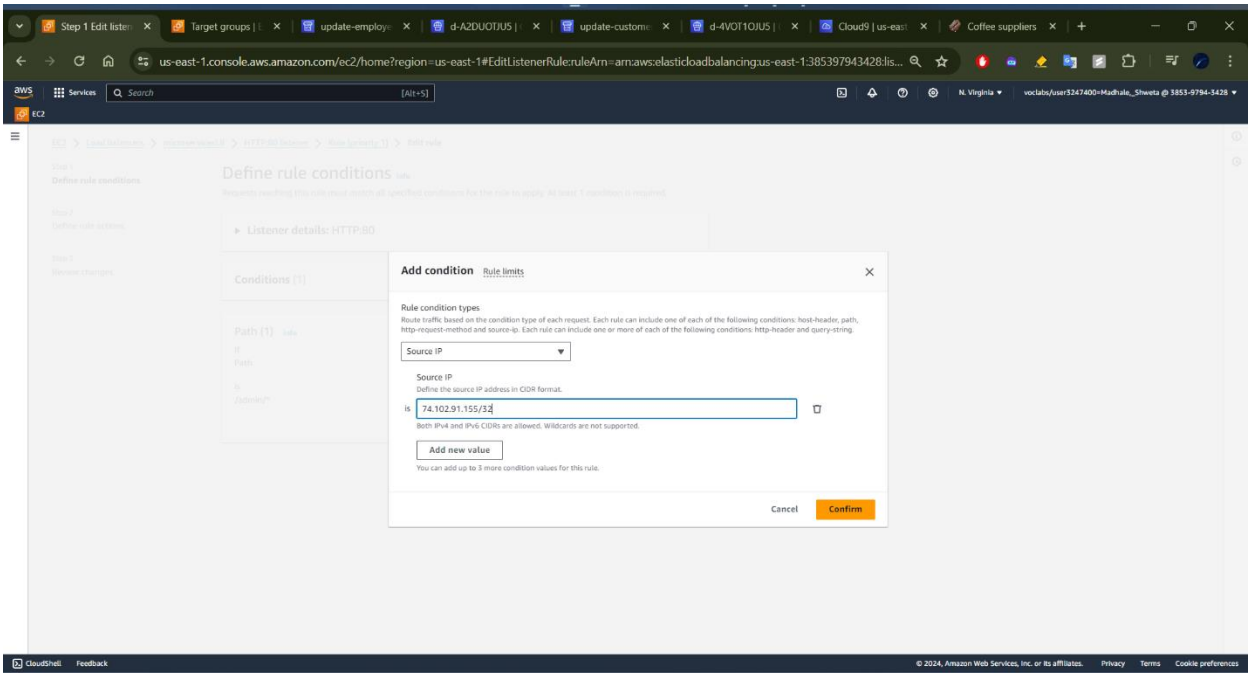
Phase 9 - Modify your microservices and scale capacity and use the pipelines that you created to deploy iterative improvements to production by using a blue/green deployment strategy.

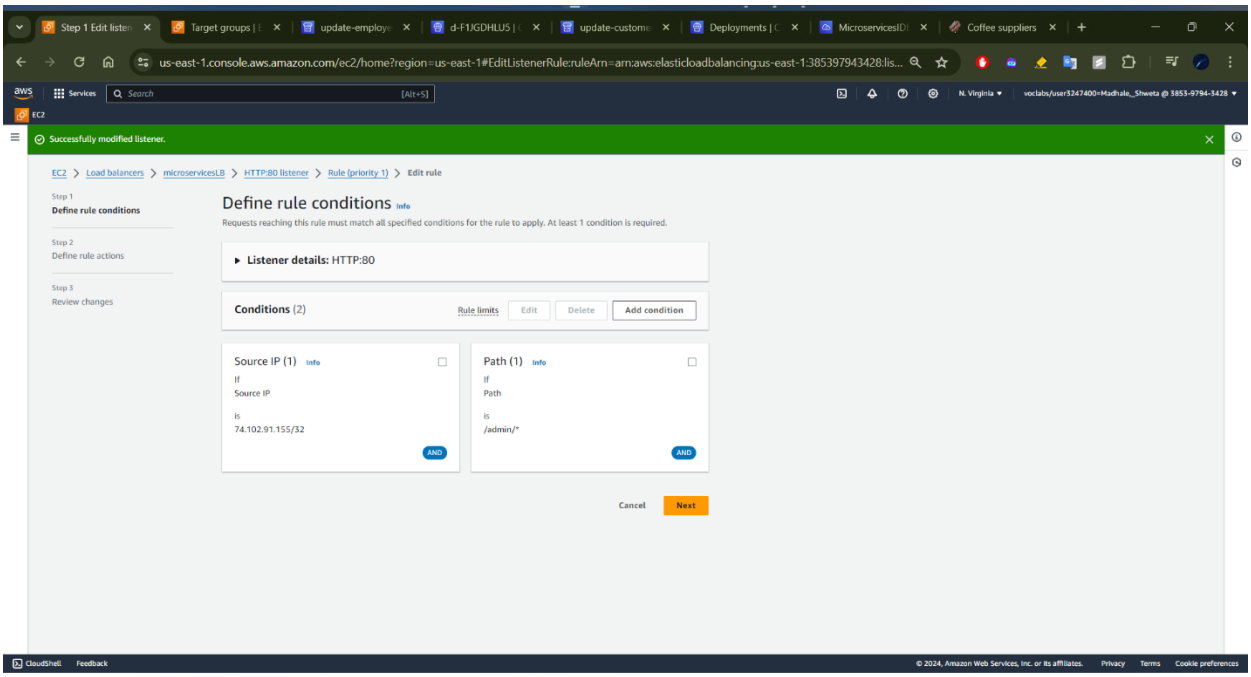
Task 9.1: Limit access to the employee microservice

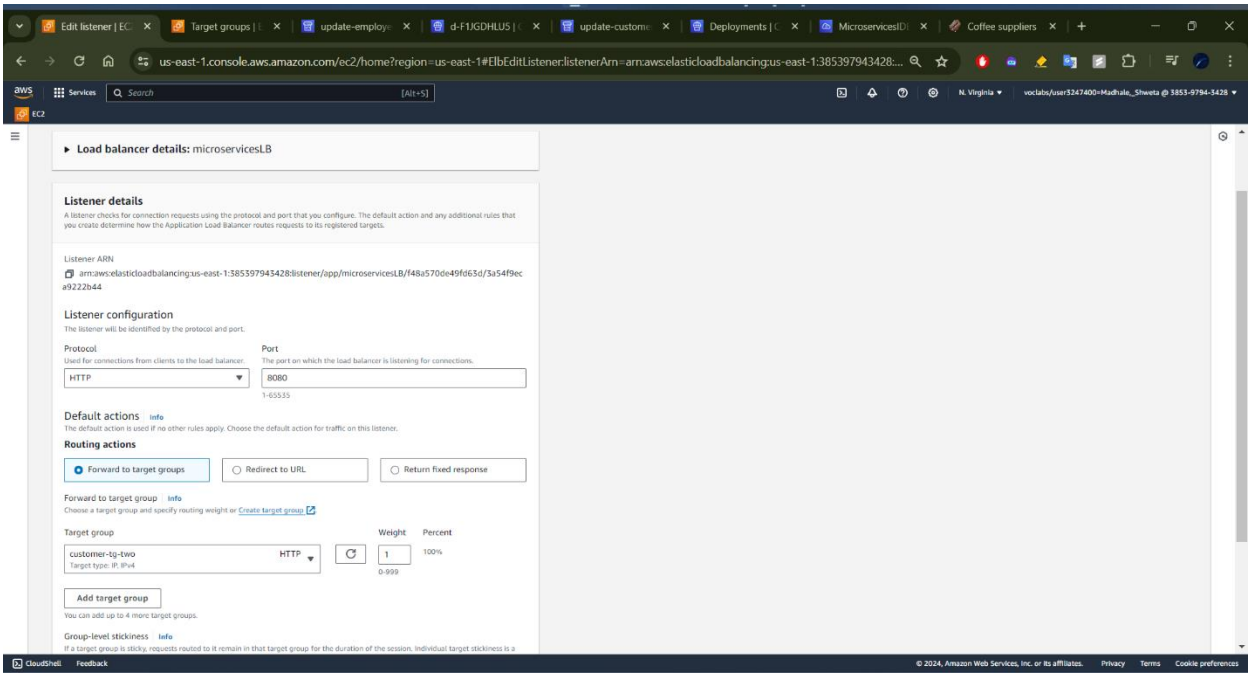

Lab Instr
readme
What Is
Setting
AWS Arc
AWS Ref
whatismyip.com
WhatIsMyIP.com
Search..
Pricing
API
Sign Up
Login
Help
What Is My IP?
IP Address Lookup
IP WHOIS Lookup
DNS Lookup
Internet Speed Test
Tools
What Is My IP?
My Public IPv4: 74.102.91.155
My Public IPv6: Not Detected
My IP Location: Newark, NJ US
My ISP: Verizon Business
What is an IP address?
What is a private IP address?

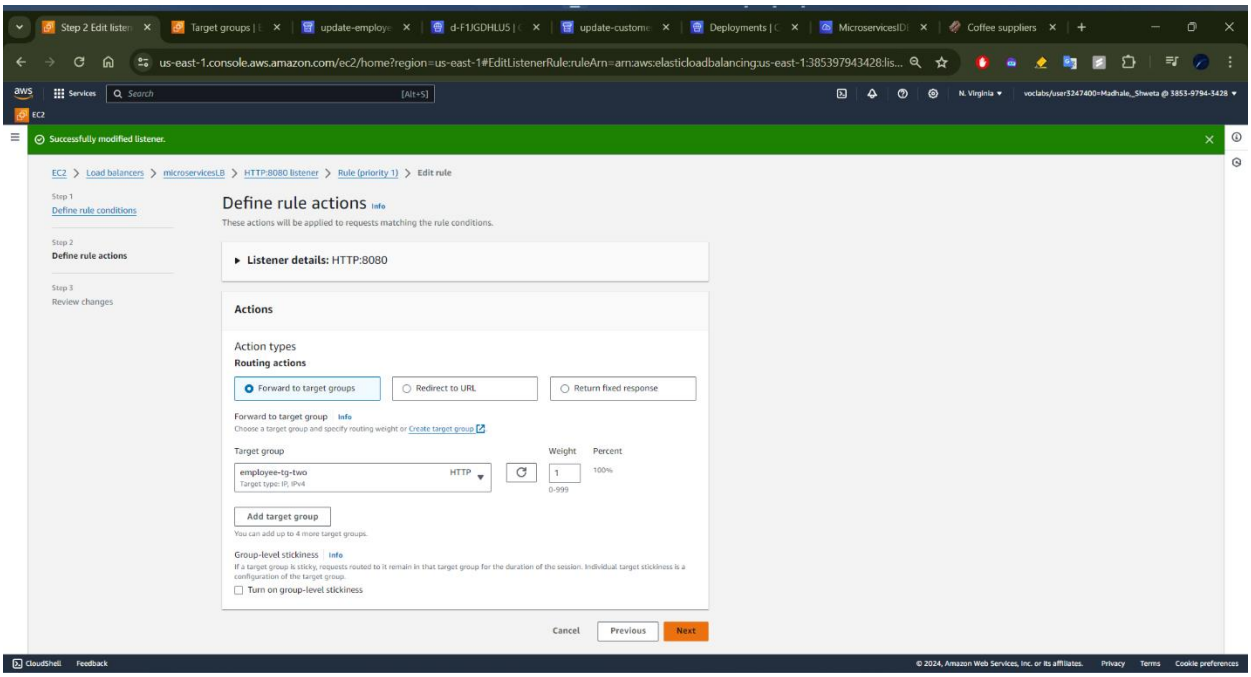

EC2 | us-east-1
Target groups
update-employe
d-A2DUOTJU5
update-custome
d-4VOT1OJU5
Cloud9 | us-east
Coffee suppliers
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ELBListenerV2:loadBalancerArn=arn:aws:elasticloadbalancing:us-east-1:38539794...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
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
EC2 > Load balancers > microservicesLB > HTTP:80 listener
HTTP:80 Info
Actions
Details
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.
Protocol:Port
HTTP:80
Load balancer
microservicesLB
Default actions
Forward to target group
customer-tg-one: 1 (100%)
Group-level stickiness: Off
Listener ARN
arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd63d/1647cb35c382ee3b
Rules
Tags
Listener rules (1/2) Info
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.
Rule limits
Actions
Add rule
View rule
Edit rule
Delete rule
Reprioritize rules
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
Tags
-
1
Path Pattern is /admin/*
Forward to target group
employee-tg-two: 100 (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

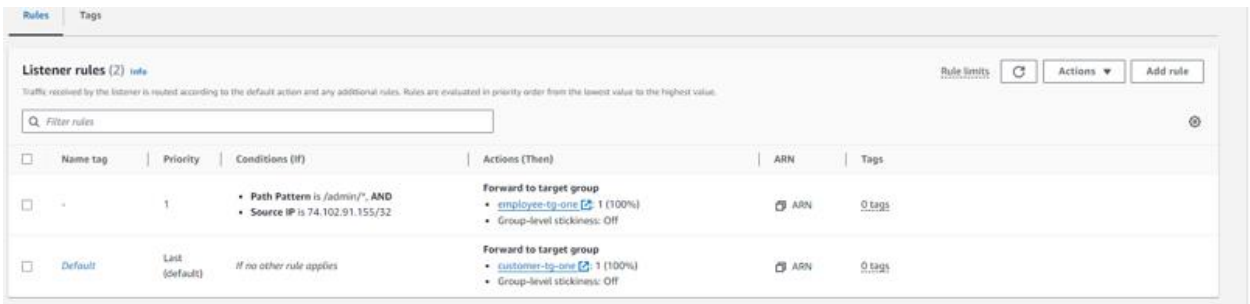

Edit listener | EC
Target groups
update-employe
d-F1JGDHLU5
update-custome
Deployments
MicroservicesID
Coffee suppliers
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ElbEditListener:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Load balancer details: microservicesLB
Listener details
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.
Listener ARN
arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd63d/1647cb35c382ee3b
Listener configuration
The listener will be identified by the protocol and port.
Protocol
Used for connections from clients to the load balancer.
HTTP
Port
The port on which the load balancer is listening for connections.
80
1-65535
Default actions Info
The default action is used if no other rules apply. Choose the default action for traffic on this listener.
Routing actions
Forward to target groups
Redirect to URL
Return fixed response
Forward to target group Info
Choose a target group and specify routing weight or Create target group
Target group
Weight
Percent
customer-tg-one
Target type: IP, IPv4
HTTP
1
0-999
100%
Add target group
You can add up to 4 more target groups.
Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

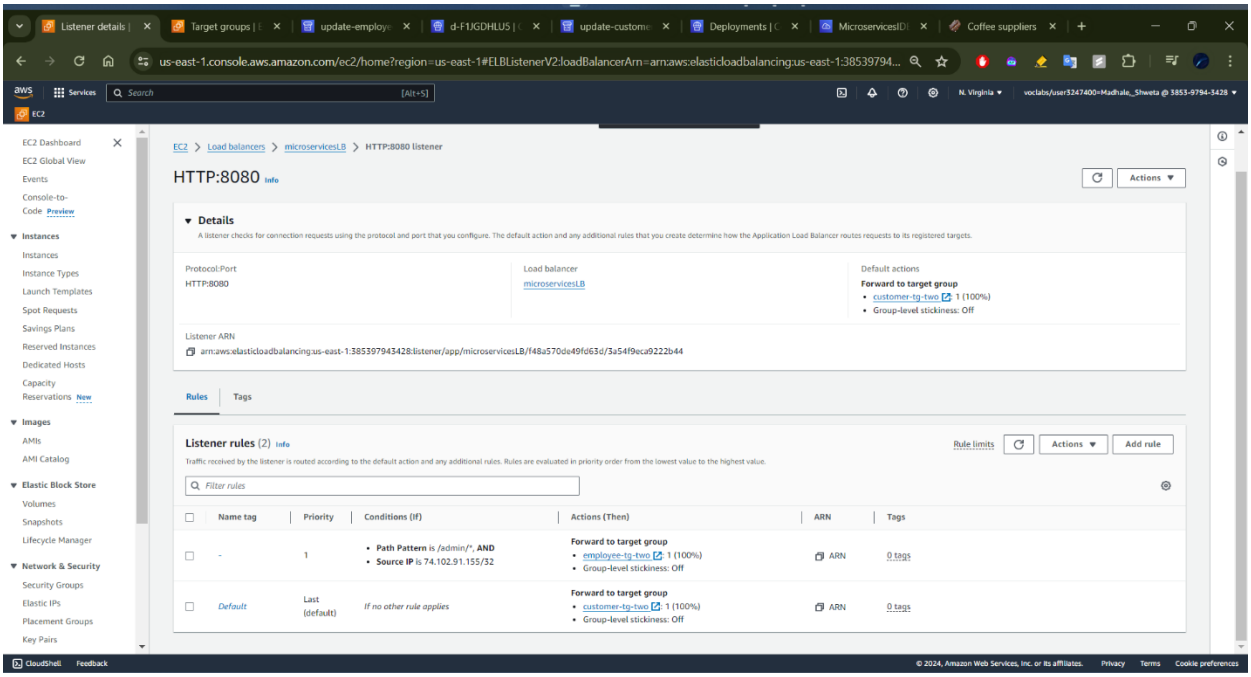

Define rule conditions
Add condition
Rule limits
Rule condition types
Source IP
Source IP
Define the source IP address in CIDR format.
is
74.102.91.155/32
Both IPv4 and IPv6 CIDRs are allowed. Wildcards are not supported.
Add new value
You can add up to 3 more condition values for this rule.
Cancel
Confirm


Successfully modified listener.
EC2 > Load balancers > microservicesLB > HTTP:80 listener > Rule (priority 1) > Edit rule
Define rule conditions
Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.
Listener details: HTTP:80
Conditions (2)
Rule limits
Edit
Delete
Add condition
Source IP (1)
If
Source IP
is
74.102.91.155/32
AND
Path (1)
If
Path
is
/admin/*
AND
Cancel
Next


Edit listener | EC
Target groups |
update-employe
d-F1JGDHLU5 |
update-custome
Deployments |
MicroservicesID
Coffee suppliers
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ElbEditListener:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Load balancer details: microservicesLB
Listener details
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.
Listener ARN
arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd63d/3a54f9eca9222b44
Listener configuration
The listener will be identified by the protocol and port.
Protocol
Used for connections from clients to the load balancer.
HTTP
Port
The port on which the load balancer is listening for connections.
8080
1-65535
Default actions Info
The default action is used if no other rules apply. Choose the default action for traffic on this listener.
Routing actions
Forward to target groups
Redirect to URL
Return fixed response
Forward to target group Info
Choose a target group and specify routing weight or Create target group
Target group
Weight
Percent
customer-tg-two
HTTP
Target type: IP, IPv4
1
0-999
100%
Add target group
You can add up to 4 more target groups.
Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Step 2 Edit listen
Target groups |
update-employe
d-F1JGDHLU5 |
update-custome
Deployments |
MicroservicesID
Coffee suppliers
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#EditListenerRule:ruleArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:lis...
Services
Search
[Alt+S]
N. Virginia
voclabs/user3247400=Madhale,_Shweta @ 3853-9794-3428
EC2
Successfully modified listener.
EC2 > Load balancers > microservicesLB > HTTP:8080 listener > Rule (priority 1) > Edit rule
Step 1
Define rule conditions
Step 2
Define rule actions
Step 3
Review changes
Define rule actions Info
These actions will be applied to requests matching the rule conditions.
Listener details: HTTP:8080
Actions
Action types
Routing actions
Forward to target groups
Redirect to URL
Return fixed response
Forward to target group Info
Choose a target group and specify routing weight or Create target group
Target group
Weight
Percent
employee-tg-two
HTTP
Target type: IP, IPv4
1
0-999
100%
Add target group
You can add up to 4 more target groups.
Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.
Turn on group-level stickiness
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Rules
Tags
Listener rules (2) Info
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.
Rule limits
Actions
Add rule
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
Tags
1
Path Pattern is /admin/*, AND
Source IP is 74.102.91.155/32
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



Task 9.2: Adjust the UI for the employee microservice and push the updated image to Amazon ECR

- Edit the employee/views/nav.html file.
  - On line 1, change navbar-dark bg-dark to navbar-light bg-light
  - Save the change.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <img src="/img/espresso.jpg" width="200">
    <div><a class="navbar-brand page-title" href="/supplier">Manage coffee suppliers</a></div>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="/admin/suppliers">Administrator home</a>
                <a class="nav-link" href="/suppliers">Suppliers list</a>
                <a class="nav-link" href="/">Customer home</a>
            </li>
        </ul>
    </div>
</nav>
```

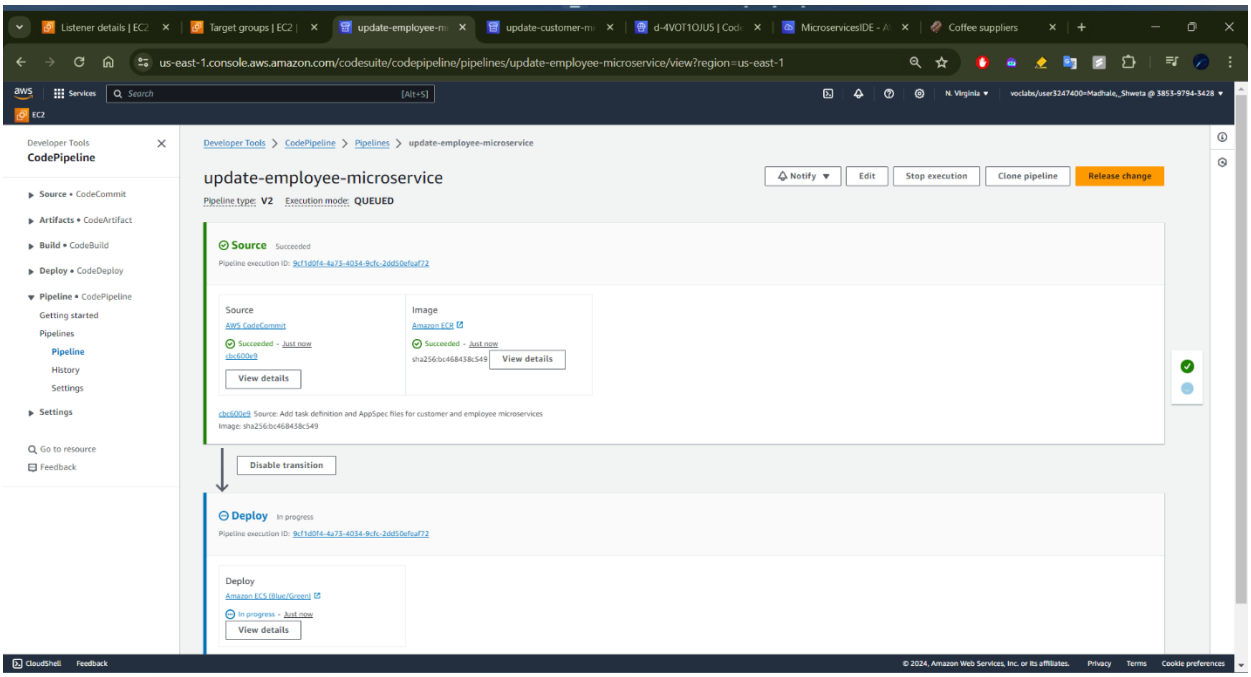
- To generate a new Docker image from the employee microservice source files that you modified and to label the image, run the following commands:

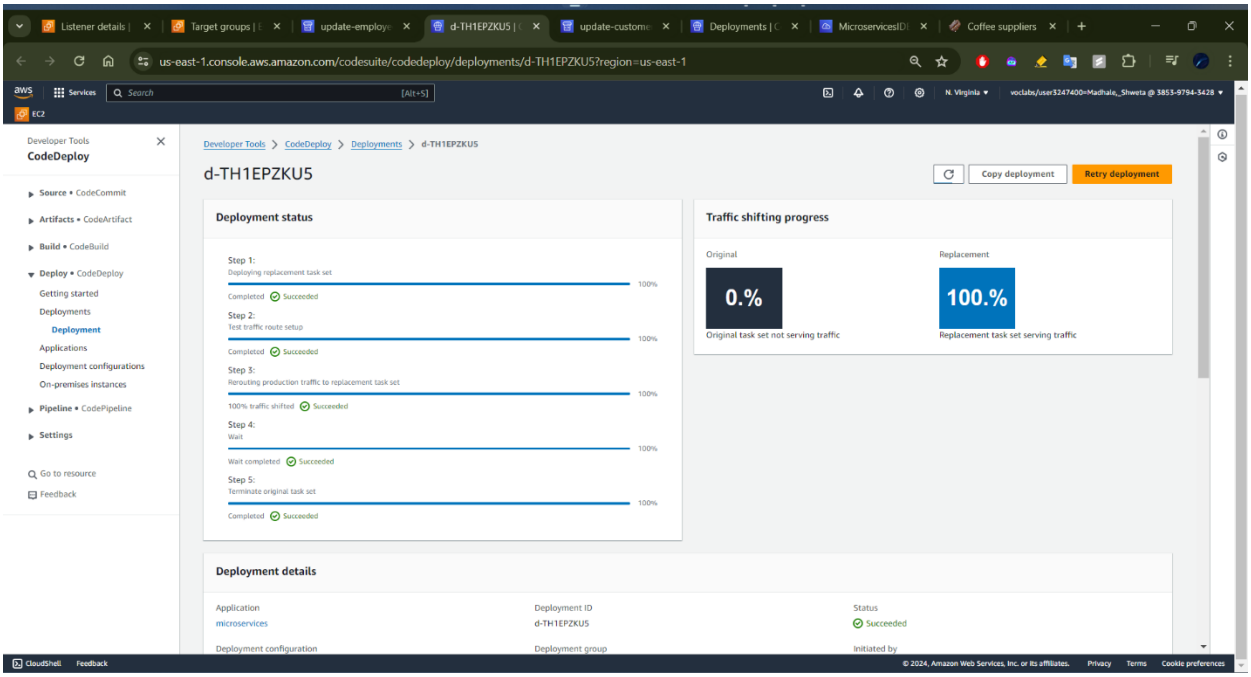
```
docker rm -f employee_1
cd ~/environment/microservices/employee
docker build --tag employee .
dbEndpoint=$(cat ~/environment/microservices/employee/app/config/config.js | grep 'APP_DB_HOST' | cut -d
'"' -f2)
echo $dbEndpoint
account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)
echo $account_id
docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
```

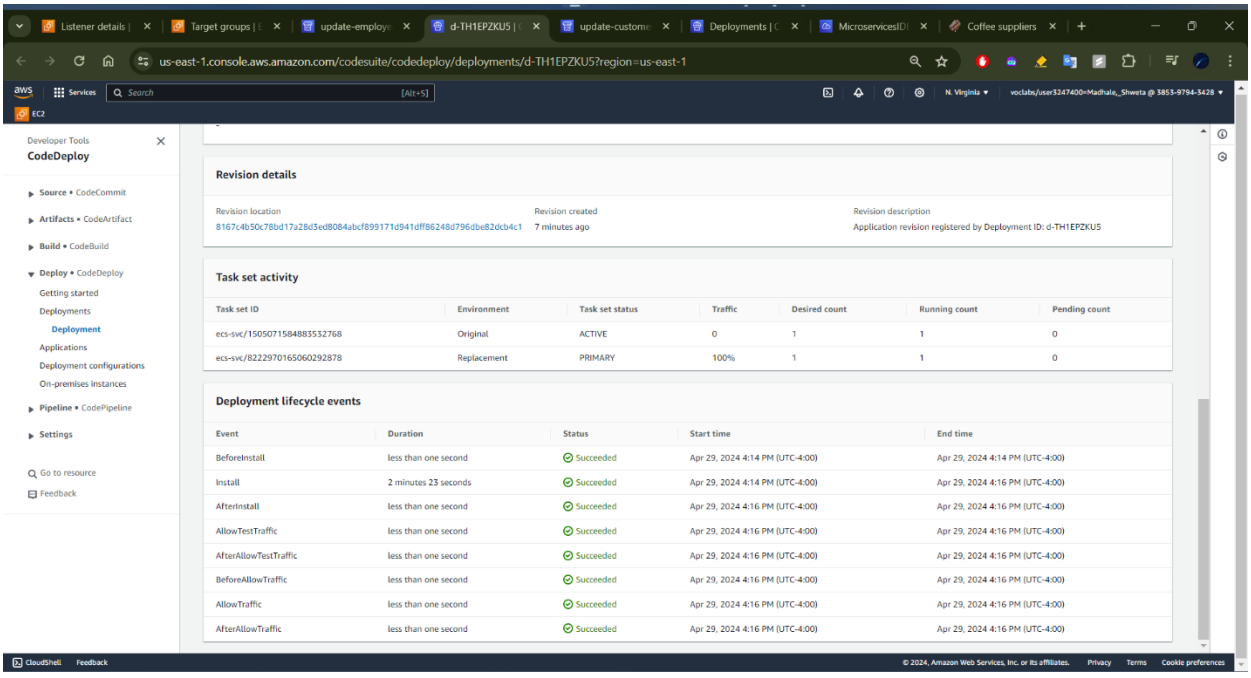
```
voclabs:~/environment $ docker rm -f employee_1
employee_1
voclabs:~/environment $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
[+] Building 3.0s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 229B
 => [internal] load metadata for docker.io/library/node:11-alpine
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf80f57ffe3b91f5fa6945a4d505932
 => [internal] load build context
 => => transferring context: 264.48kB
 => CACHED [2/5] RUN mkdir -p /usr/src/app
 => CACHED [3/5] WORKDIR /usr/src/app
 => [4/5] COPY . .
 => [5/5] RUN npm install
 => exporting to image
 => => exporting layers
 => => writing image sha256:e526550dc16bcc2c07d77c8be66dc395544953844180d174ba77872131e66a54
 => => naming to docker.io/library/employee
voclabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/employee/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint
supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/employee (dev) $ account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)
voclabs:~/environment/microservices/employee (dev) $ echo $account_id
385397943428
voclabs:~/environment/microservices/employee (dev) $ docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
voclabs:~/environment/microservices/employee (dev) $
```

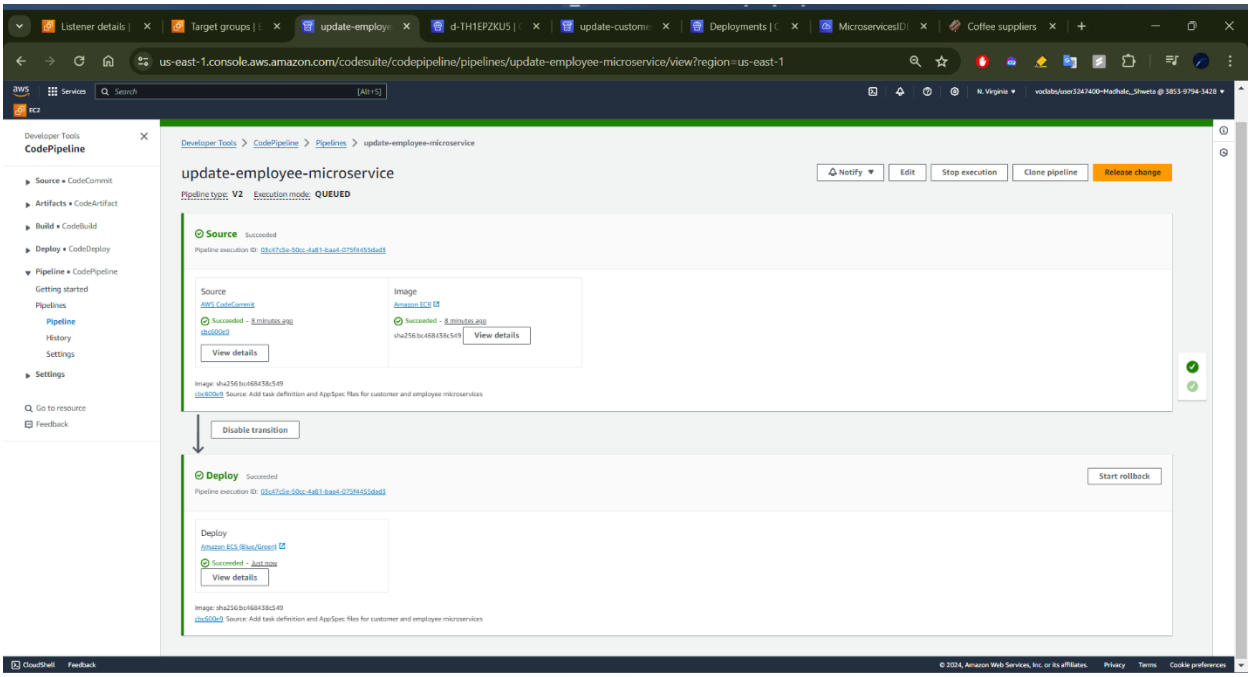
```
voclabs:~/environment/microservices/employee (dev) $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclabs:~/environment/microservices/employee (dev) $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
The push refers to repository [385397943428.dkr.ecr.us-east-1.amazonaws.com/employee]
2100f3e094c4: Pushed
385e11831471: Pushed
5f70bf18a086: Layer already exists
6f8e0edab334: Layer already exists
d81d715330b7: Layer already exists
1dc7f3bb09a4: Layer already exists
dcaceb729824: Layer already exists
f1b5933fe4b5: Layer already exists
latest: digest: sha256:bc468438c549daa302ff72a107e424af0fb6afa9bc75db55d3e32f9f981fa618 size: 1989
voclabs:~/environment/microservices/employee (dev) $
```

Task 9.3: Confirm that the employee pipeline ran and the microservice was updated



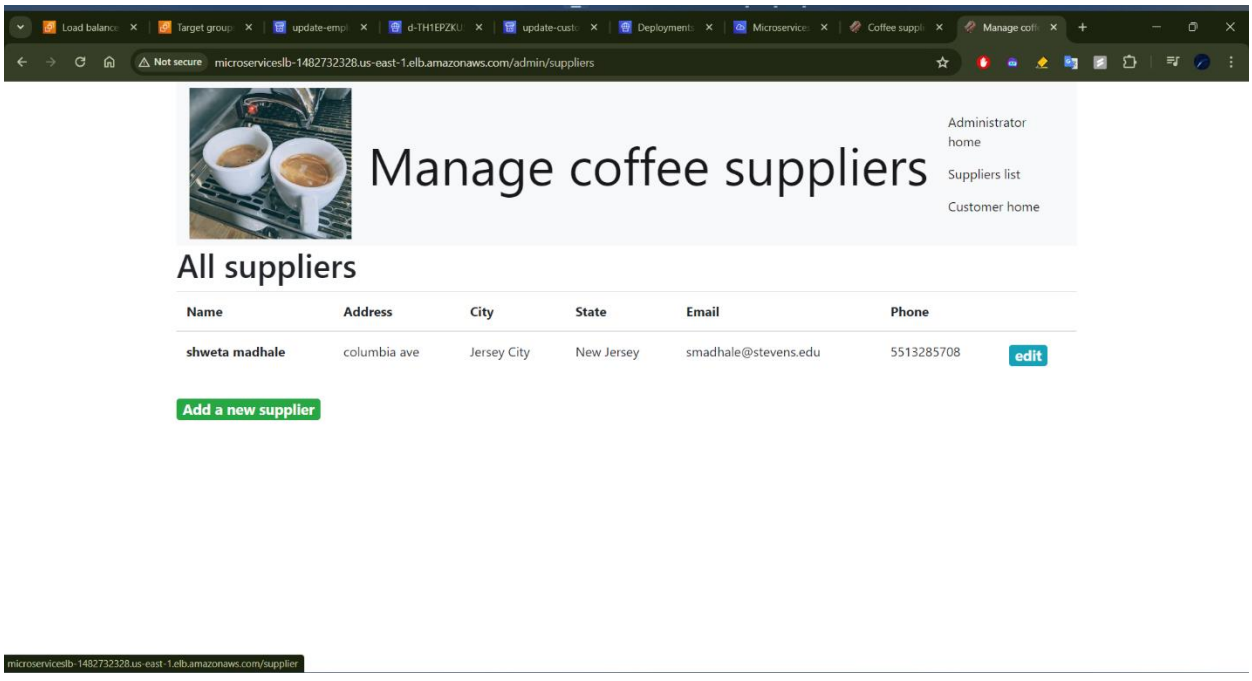

Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Deployment
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Developer Tools > CodeDeploy > Deployments > d-TH1EPZKU5
d-TH1EPZKU5
Copy deployment
Retry deployment
Deployment status
Step 1:
Deploying replacement task set
100%
Completed Succeeded
Step 2:
Test traffic route setup
100%
Completed Succeeded
Step 3:
Rerouting production traffic to replacement task set
100%
100% traffic shifted Succeeded
Step 4:
Wait
100%
Wait completed Succeeded
Step 5:
Terminate original task set
100%
Completed Succeeded
Traffic shifting progress
Original
0.%
Original task set not serving traffic
Replacement
100.%
Replacement task set serving traffic
Deployment details
Application
microservices
Deployment ID
d-TH1EPZKU5
Status
Succeeded
Deployment configuration
Deployment group
Initiated by
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

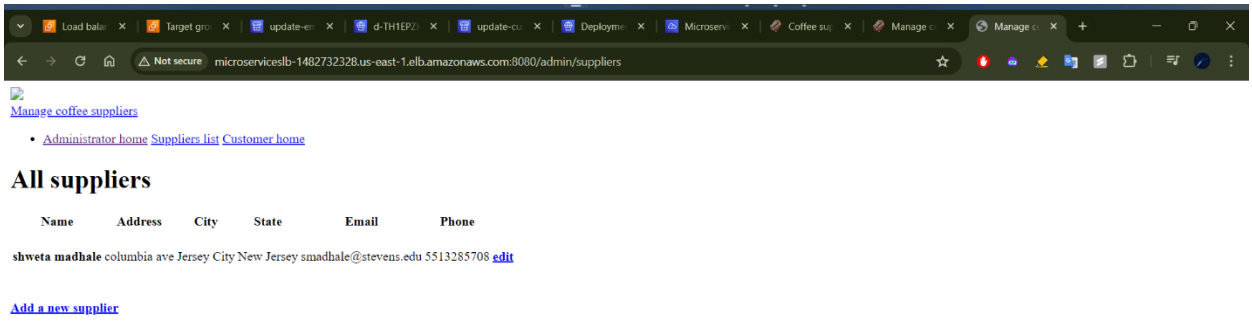

Revision details
Revision location
8167c4b50c78bd17a28d3ed8084abcf899171d941dff86248d796dbe82dcb4c1
Revision created
7 minutes ago
Revision description
Application revision registered by Deployment ID: d-TH1EPZKU5
Task set activity
Task set ID | Environment | Task set status | Traffic | Desired count | Running count | Pending count
ecs-svc/1505071584883532768 | Original | ACTIVE | 0 | 1 | 1 | 0
ecs-svc/8222970165060292878 | Replacement | PRIMARY | 100% | 1 | 1 | 0
Deployment lifecycle events
Event | Duration | Status | Start time | End time
BeforeInstall | less than one second | Succeeded | Apr 29, 2024 4:14 PM (UTC-4:00) | Apr 29, 2024 4:14 PM (UTC-4:00)
Install | 2 minutes 23 seconds | Succeeded | Apr 29, 2024 4:14 PM (UTC-4:00) | Apr 29, 2024 4:16 PM (UTC-4:00)
AfterInstall | less than one second | Succeeded | Apr 29, 2024 4:16 PM (UTC-4:00) | Apr 29, 2024 4:16 PM (UTC-4:00)
AllowTestTraffic | less than one second | Succeeded | Apr 29, 2024 4:16 PM (UTC-4:00) | Apr 29, 2024 4:16 PM (UTC-4:00)
AfterAllowTestTraffic | less than one second | Succeeded | Apr 29, 2024 4:16 PM (UTC-4:00) | Apr 29, 2024 4:16 PM (UTC-4:00)
BeforeAllowTraffic | less than one second | Succeeded | Apr 29, 2024 4:16 PM (UTC-4:00) | Apr 29, 2024 4:16 PM (UTC-4:00)
AllowTraffic | less than one second | Succeeded | Apr 29, 2024 4:16 PM (UTC-4:00) | Apr 29, 2024 4:16 PM (UTC-4:00)
AfterAllowTraffic | less than one second | Succeeded | Apr 29, 2024 4:16 PM (UTC-4:00) | Apr 29, 2024 4:16 PM (UTC-4:00)



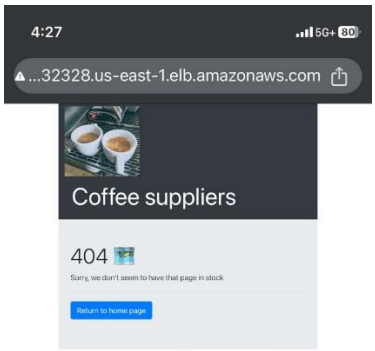
Task 9.4: Test access to the employee microservice

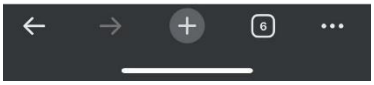
- Test access to the employee microservice pages at http://<alb-endpoint>/admin/suppliers and http://<alb-endpoint>:8080/admin/suppliers from the same device that you have used for this project so far. Replace <alb-endpoint> with the DNS name of the microservicesLB load balancer.





- Test access to the same employee microservice pages from a different device.

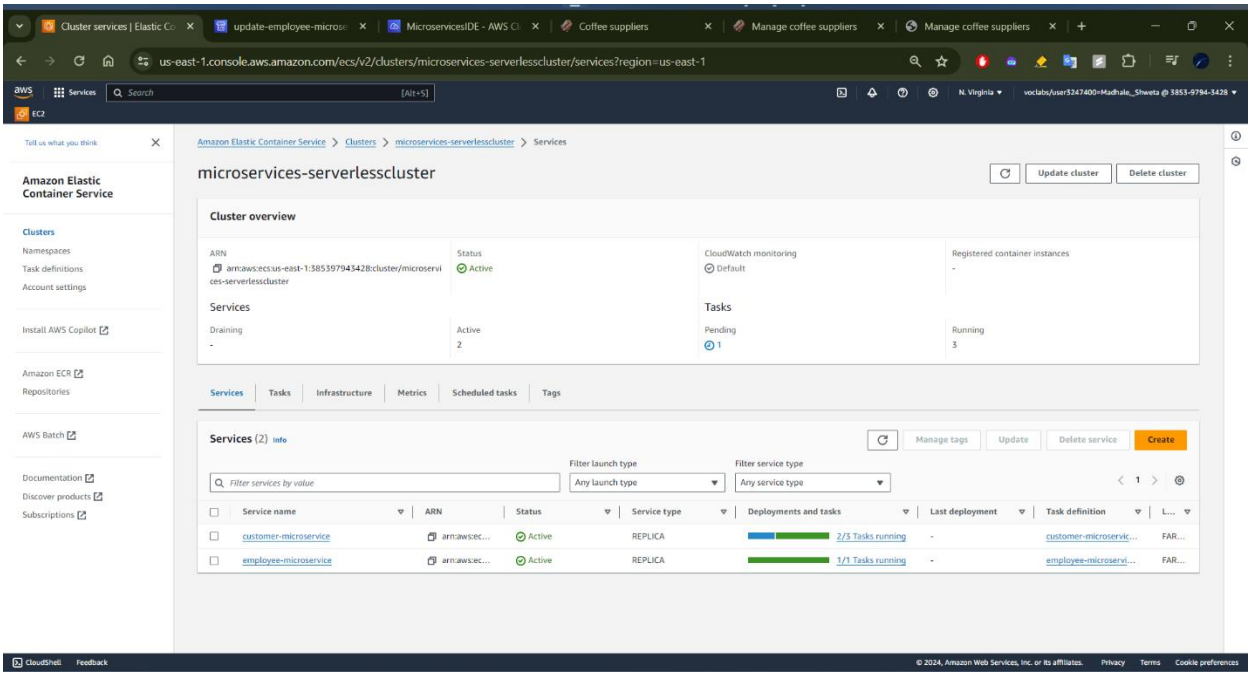


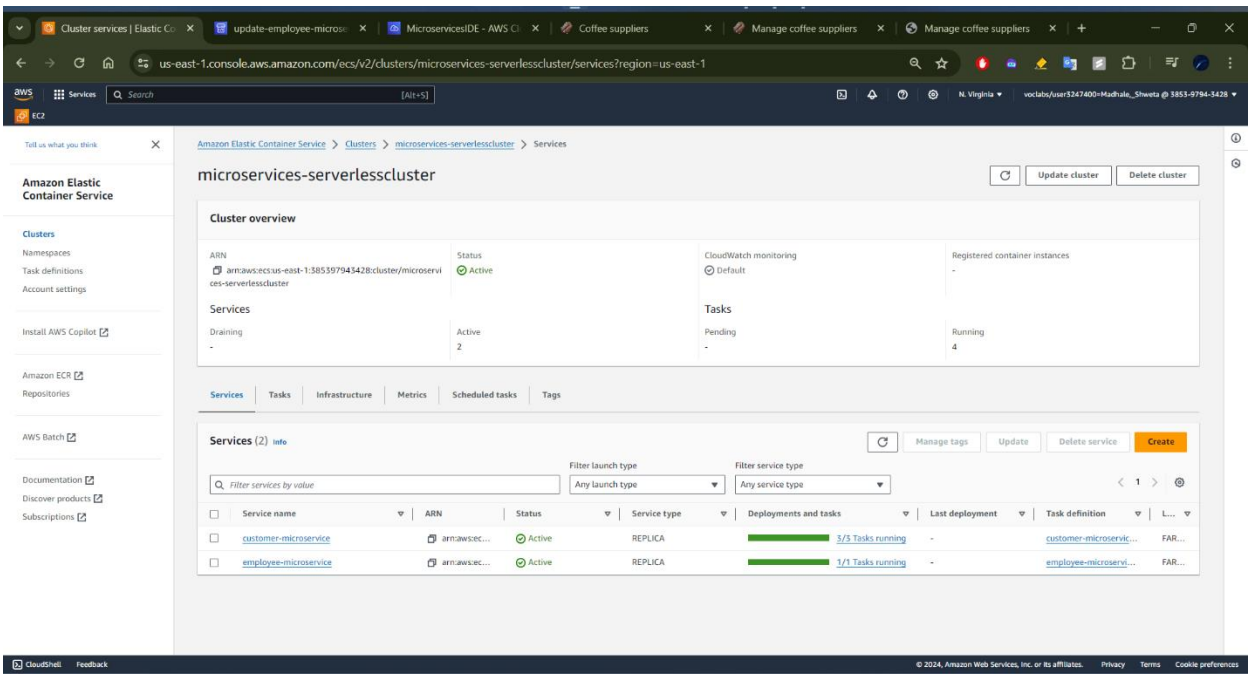


Task 9.5: Scale the customer microservice

- Update the customer service in Amazon ECS.
  - Run the following command:

- aws ecs update-service --cluster microservices-serverlesscluster --service customer-microservice --desired-count 3
- A large JSON-formatted response is returned.





Analysis

This project was aimed to build microservices and continuous integration and continuous development (CI/CD) solution. The project involved a scenario with a coffee suppliers application which runs as a monolithic application. This monolithic application has some reliability and performances issues so some tasks which will mitigate these issues were executed. The major task was to split this monolithic application into microservices which are scalable independently. Also, the microservices can allocate extra resources when necessary to services with high demand. This also helped mitigate the problem of single point of failure and dependency of microservices. Also, a CI/CD pipeline was automatically deployed different updates to the production clusters which run the containers and use of blue/green deployment strategy. The various phases involved with this project are as follows:

Phase 1. Create an architecture diagram and cost estimate for the solution.

- Analysis
  This phase involves creating an architecture diagram and cost estimate. This ensures a better understanding of the infrastructure for everyone involved. This can facilitate analysis of the infrastructure before implementing and make necessary changes where required. Also, the cost estimate can help evaluate the financial scope of the project and budgeting around it. This ensures that informed decisions are made regarding aspects of the project like resource allocation. This phase sets up a direction for the entire project and helps ensure that the solution is technically and financially feasible and well-defined to get the desired outcome.
- Significance
  The architecture diagram is significant in the project to have a visual representation of the proposed project implementation. It helps in understanding the various components within the project, their connections, how they work together to come to the solution. Whereas cost estimation is important for setting budgets and other planning. The architecture and cost estimation will ensure that the project will give desired outcomes within the proposed infrastructure efficiently and within the budget.
- Outcome
  The outcome of this phase involves the planning before the implementation of the project. This phase will give an architecture diagram which will help understand the infrastructure involving various resources and services within AWS. The cost estimate gives an estimate of the expenses which will be incurred for this project's execution.

Phase 2. Analyze the design of the monolithic application and test the application. R3

- Analysis
  This task was aimed at analyzing the current infrastructure of the monolithic application. It also involved testing the web application. The availability of the web application was verified and its different features and pages were tested which included viewing data, adding suppliers, editing data, etc. The underlying infrastructure was analyzed along with various processes. The commands lsof and ps helped identify the various processes which were running on the instance. This command also identified that the application was hosted on port 80 and was managed by node process. The dependency of the project on the RDS database was also viewed to better understand the structure. Also, the application codebase was examined.
- Significance
  This phase plays significant role in understanding the current infrastructure and functionality of the monolithic application. It helped in understanding the various issues involving this

current structure and how can it be improved. It also provided significant insights into the current architecture, codebase, dependencies, and database.

- Outcome
  The outcome of this phase was to verify the validity of the current application. Checking of the accessibility of the application from its public IPv4 address was done. Various features of the application were tested as well to check the UI. Also, information regarding the hosting environment, processes, dependencies and data storage was found. This data will help in later implementation stages of the project.

Phase 3. Create a development environment on AWS Cloud9, and check the monolithic source code into CodeCommit.

- Analysis
  This phase involved creating a development environment for the project using the AWS Cloud9 Service. Also, using the AWS CodeCommit which is a git-compatible repository to check the application code into. A development environment named MicroservicesIDE was created and configured with an EC2 instance which runs Amazon Linux2 in the specified VPC and subnet. Then the source code of the monolithic application was copied into the Cloud9 environment. Also, various directories for the customer and employee microservices were configured which prepared the environment to split the monolithic application into microservices. Finally, a git repository for these microservices was created using the CodeCommit. The original source code along with the modified code were pushed to this repository.
- Significance
  This phase of development helped in creating a development environment and setting up a git repository which will help in version control. The development environment was created with the Cloud9 service which provided a workspace for the project development. The CodeCommit service was used to set up the repository which helped in version control and in general code management throughout the project.
- Outcome
  This phase establishes a development environment and sets up a version control for the application code which will be the foundation of the upcoming tasks.

Phase 4. Break the monolithic design into microservices and launch test Docker containers.

- Analysis
  In this phase the microservices originating from splitting the monolithic application were configured with a starter code. The application functionality is implemented as two separate microservices. For initial testing, the containers were run on the same EC2 instance of the Cloud9 environment. The IDE was used to build the docker images and for launching the docker containers. Firstly, the security group for Cloud9 EC2 instance was modified to allow inbound traffic on the TCP ports 8080 and 8081 which is used for running the docker containers. Then the source code for the customer microservice was modified to remove functions supporting employee actions. A docker file was created for this microservice, a docker images was built and a test container was launched. The docker container was configured to run on the port 8080 and was connect to the RDS database endpoint. Also, the source code for the employee microservice was modified to enable read write operations. Again a docker file was created, docker image was built and a test container was launched on port 8081. Later the port for employee microservice was configured to port 8080. Lastly, the modified codes were pushed into the CodeCommit repository.
- Significance

This phase mainly played a significant role in splitting the monolithic application into employee and customer microservices and testing these microservices within the docker containers. Microservices play important roles for portability, dependability, for maintaining and scaling. The docker containers provide a lightweight portable environment for running the applications. This ensured consistent and efficient deployment of the microservices across the environments,

- Outcome

  In this phase the monolithic application was successfully split into two microservices employee and customer. Then docker containers were created for both the microservices and and were tested which help in checking the feasibility of running the application in a containerized environment. Also the port configurations were adjusted as per the deployment requirements and all the changes were successfully committed and pushed to CodeCommit.

Phase 5. Create ECR repositories to store Docker images. Create an ECS cluster, ECS task definitions, and CodeDeploy application specification files.

- Analysis

  In this phase, deployment of the microservices-based web application to Amazon ECS (Elastic Container Service) with Fargate, integration with CI/CD (Continuous Integration/Continuous Deployment) pipelines was done. With microservices, the deployment process becomes simpler and more efficient. Amazon ECS was used with fargate which made it easier to implement the underlying infrastructure with a serverless container. The integration of CI/CD pipelines automated the building, testing, and deployment processes which ensured consistent and reliable deployment. Firstly the latest docker images of the two microservices were uploaded to two separate Amazon ECR repositories. Then an ECS cluster was created which was serverless AWS fargate cluster configured for the specified VPC and subnets. Also, another code repository was created just for the deployment process. Then task definition for each of the microservices was created and registered to the Amazon ECS. Also, application specific files were created for each microservice to provide instructions regarding deployment of the microservice to the Amaozn ECS on fargate infrastructure which were provided to CodeDeploy.

- Significance

  This phase plays a very crucial stage in the project because of the deployment of the microservices based application with help of Amazon ECS with fargate. By breaking down the monolithic application into smaller independently deployable services the deployment became more flexible and easier. The containerization with docker also ensured consistency with deployment of each microservices and helped to streamline the entire process.

- Outcome

  The docker clients were connected to the Amazon ECR service. Separate driver repositories were created for each micro service and appropriate permissions were set. The docker images for each microservice were pushed to Amazon ECR. A serverless AWS fargate cluster was created. Code commit repository was created to store the deployment files. Task definition files were created for each microservice, and they were registered. Application specification files were also created for each of the microservice. In this phase, an efficient deployment pipeline supported by the AWS services was created. This includes creation and configuration of Amazon ECR repositories for storing the docker images of the micro services. Tagging and uploading of the docker images to the ECR to make them available for deployment was done. Also, an ECS cluster was created which enabled managing the containerized application, enhancing the scalability and resource utilization.

Phase 6. Create target groups and an Application Load Balancer that routes web traffic to them.

- Analysis
  This task involved creating an application load balancer which eventually provided an endpoint URL. This URL will subsequently behave as the HTTPS entry point for the users – customers and employees. This load balancer was configured with listeners who were in turn configured with routing and access rules for different target groups for directing requests. With this phase the web application can handle a lot of traffic without crashing. Services like EC2LB and security groups are well used to implement scalable and highly available architecture. The use of path-based routing rules enhanced flexibility and control over traffic routing.
- Significance
  in this phase the application load balancer and the target groups were created. This step is very pivotal as the ALB will serve as the entry point for the incoming traffic and it provides a single HTTPS endpoint for the customers and the employees to access the application. Also, traffic is distributed across the multiple target groups which will ensure that the ALB will have high availability and will direct request to healthy instances. Also, use of the blue-green deployment strategies with multiple target groups will help and seamless update and rollback capabilities and will minimize the impact on the user during the deployment.
- Outcome
  in this phase four target groups were created which will aid in facilitating efficient routing of traffic. Each of the target groups is configured with specific health checks and routing rules. The load balancer was provisioned with two listeners, one on the port 80 and another on port 8080 which will help handle incoming HTTP traffic. This will help in routing the request to the appropriate target group and will ensure distributing the workload across various instances.

Phase 7. Create ECS services.

- Analysis
  This phase involves creating Amazon ECS for each of the microservices which makes it easier to manage the microservices independently as each is deployed to its own ECS service. This will enhance the maintenance and scalability of the application. This also will decrease the dependence between each of the components. Also, with the use of ECS load balancers and task definitions, it is easier to manage the applications and scale while maintaining optimal performance and resource utilization.
- Significance
  This phase is very significant because it helps in managing and scaling the microservices independently. By having two dedicated services for each of the micro services it is ensured that changing or updating one microservice will not affect the other. This also is useful in the case of troubleshooting or debugging so each of the micro services is independent and can be managed on its own.
- Outcome
  In this phase Amazon ECS services, one for the customer microservice and another for the employee microservice were created. Each of the servers was configured with its own task definition, some network settings, and load balancer configuration. Because of separate services it is possible to deploy scale and monitor each of the microservice independently

Phase 8. Configure applications and deployments groups in CodeDeploy, and create two CI/CD pipelines by using CodePipeline.

- Analysis
  In this task, a CI/CD pipeline was defined which was used to deploy the application. Configuring code deploy and code pipeline the development and deployment workflow is automated. With the blue/green deployment the deployment of updates is possible seamlessly and has minimal impact on the production environment. Integration of the AWS services used previously like ECS, code commit, an ECR provided scalable solution for managing the entire software lifecycle.
- Significance
  In this phase the configuration of the code deployment code pipeline was done which played a significant role in establishing the CI/CD pipeline for deploying and managing each of the microservices. Because of the automation of the deployment process the deployment cycle became very consistent and easier. Also the code deploys blue-green deployment strategy enhanced the reliability by minimizing the downtime and by enabling an easy rollback if there are any deployment issues.
- Outcome
  A CodeDeploy application was created which is a collection of deployment groups and revisions. This application was created by specifying the earlier created loadbalancer, ECS services, listener, target groups. Two deployment groups for customer and employee were created in this application with the specified configuration. A pipeline was created, one each for employee and customer with CodeCommit as source and CodeDeploy responsible for deployment. Amazon ECS blue/green deployment was used to test the new application version and to send production traffic to it. Each of the CI/CD pipelines were tested as well. Also, changes to the load balancer listener rules from CodeDeploy were observed.

Phase 9. Modify your microservices and scale capacity, and use the pipelines that you created to deploy iterative improvements to production by using a blue/green deployment strategy.

- Analysis
  In this phase, the benefits for the microservices environment were understood. There were some changes in configuration of the listener rules and source code. Also, a new docker image was generated and pushed to the ECR which automatically caused the pipeline to run and update the deployment. Also, scaling the number of containers was possible to support the customer microservices. Limiting the IP addresses the security and control over sensitive features voice insured. Also there were some changes in the UI of just one microservice which didn't affect the other microservice. Also automation of the docker image update and the deployment via the CI CD pipeline was possible. Finally the ability to scale the customer microservice independently was done which demonstrated the elasticity benefits and the scalability benefits of the micro services which allowed efficient resource utilization and responds to a changing or dynamic workload demand.
- Significance
  This phase played a significant role in understanding the practical application of the microservices architectures and how the CI/CD pipelines play a role in it. The microservice source code was reconfigured and its independence with each other was understood. Also, the trigger of the automated pipeline execution from docker image was observed. This phase helped understand the benefits of micro services to enable independence in deployment,

development, and scaling of various components which leads to better maintenance of the system.

- Outcome
  In this phase, the access to the employee microservice was limited by changing the listener rules, In this case it was limited to my IP address. The UI for the employee microservice was also updated and a new docker image was generated and pushed to the ECR. This triggered the deployment from the pipeline created previously. Testing of the new changes was done. The customer microservice was updated to handle up to 3 containers.
-