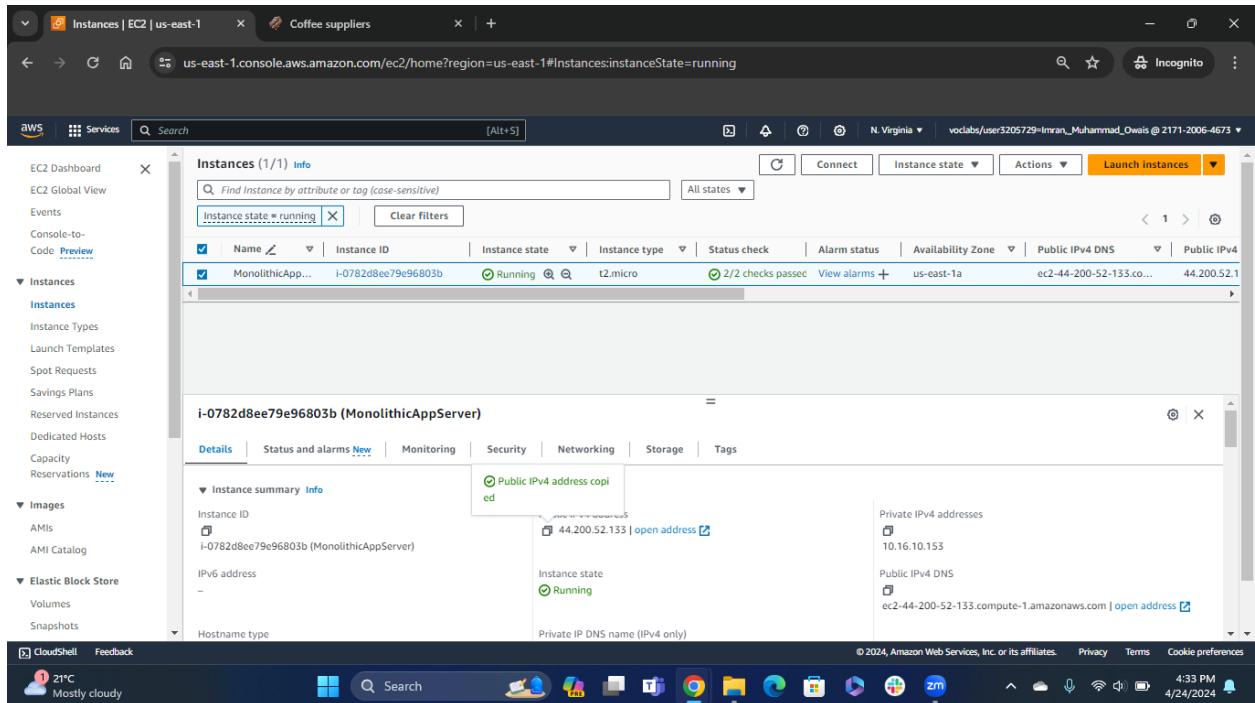


Phase 2: Analyzing the infrastructure of the monolithic application

Task 2.1: Verify that the monolithic application is available

1. Configuration of EC2 instance hosting the monolithic web application:



Task 2.2: Test the monolithic web application

1. List of Suppliers Web Page of monolithic web application:

The screenshot shows a web browser window titled "Instances | EC2 | us-east-1" with the URL "44.200.52.133/suppliers". The page has a header with two coffee cups and the text "Monolithic Coffee suppliers". It features a navigation bar with "Home" and "Suppliers list". Below the header, there's a section titled "All suppliers" with a table header row containing columns for Name, Address, City, State, Email, and Phone. A green button labeled "Add a new supplier" is visible. The taskbar at the bottom shows various icons and the date "4/24/2024".

2. Adding a New Supplier:

The screenshot shows a web browser window titled "Instances | EC2 | us-east-1" with the URL "44.200.52.133/supplier-add". The page displays a form with a blue header bar stating "All fields are required". The form fields are: Name (Muhammad Owais Imran), Address (110 Lake St), City (Jersey City), State (NJ), Email (mimran1@stevens.edu), and Phone (+1 516 675 7898). The taskbar at the bottom shows various icons and the date "4/24/2024".

3. Verifying if the supplier is added to the monolithic web application:



All suppliers

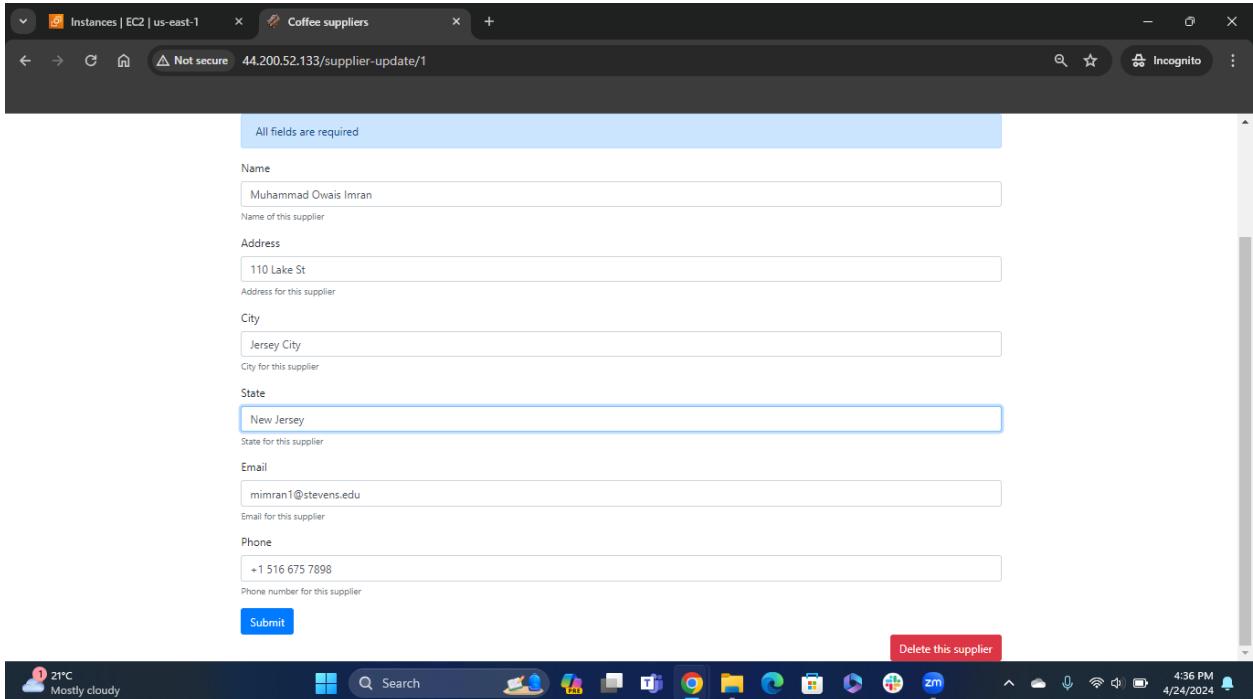
Name	Address	City	State	Email	Phone
Muhammad Owais Imran	110 Lake St	Jersey City	NJ	mimran1@stevens.edu	+1 516 675 7898

[edit](#)

[Add a new supplier](#)



4. Editing the newly added supplier state from NJ to New Jersey to check Edit Functionality of Monolithic Web Application:



5. Verifying if the edit is saved successfully:

Instances | EC2 | us-east-1

Coffee suppliers

Not secure 44.200.52.133/suppliers

Monolithic Coffee suppliers

All suppliers

Name	Address	City	State	Email	Phone
Muhammad Owais Imran	110 Lake St	Jersey City	New Jersey	mimran1@stevens.edu	+1 516 675 7898

Add a new supplier

Home

Suppliers list

21°C Mostly cloudy

Search

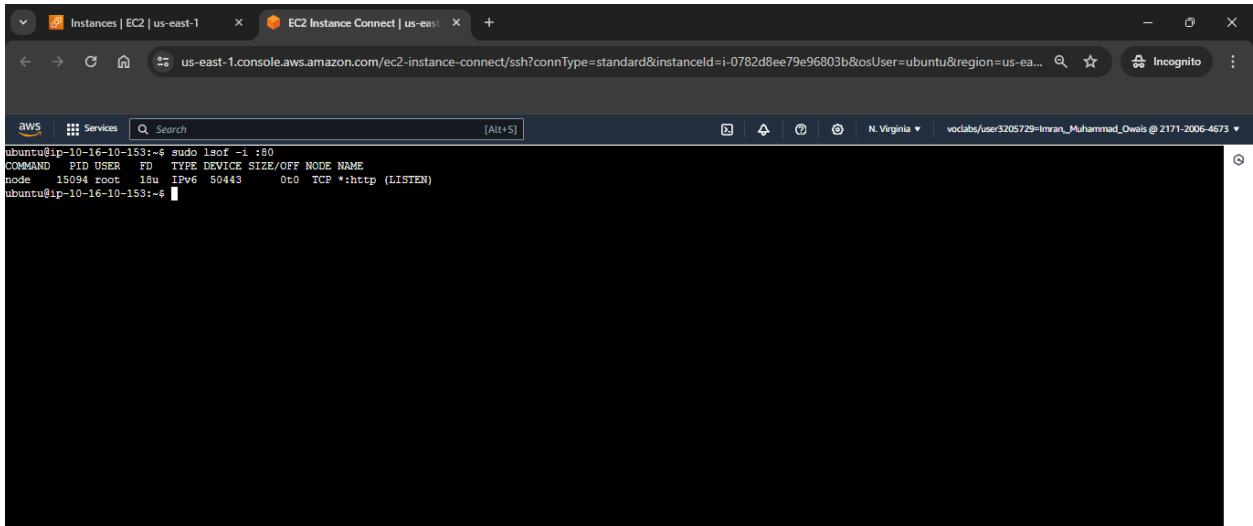
Snipping Tool

Screenshot copied to clipboard and saved
Select here to mark up and share.

4:36 PM 4/24/2024

Task 2.3: Analyze how the monolithic application runs

1. After connecting to the ec2 instance using AWS SSH Connect service.
2. Executing `sudo lsof -i :80` , this command shows us the information of the application process executing on port 80



```
ubuntu@ip-10-16-10-153:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 15094 root 18u IPv6 50443      0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-153:~$
```

i-0782d8ee79e96803b (MonolithicAppServer)
Public IPs: 44.200.52.133 Private IPs: 10.16.10.153



3. Executing `ps -ef | head -1; ps -ef | grep node` this command gives us the information of all pid involved with the process executing node, the PID of process being executed by root matches the PID returned in above step i.e. step 2 of task 2.3.

```
aws Services Search [Alt+5] Instances | EC2 | us-east-1 EC2 Instance Connect | us-east-1 N. Virginia v vocabs/user3205729-lmran_Muhammad_Owais @ 2171-2006-4673

ubuntu@ip-10-16-10-153:~$ ps -ef | head -1; ps -ef | grep node
UID      PID  PPID  C STIME  TT  TIME Cmd
root     1509     1  0 20:25 ?    00:00:00 node index.js
ubuntu   15642  15617  0 20:39 pts/0    00:00:00 grep --color=auto node
ubuntu@ip-10-16-10-153:~$
```

i-0782d8ee79e96803b (MonolithicAppServer)
PublicIPs: 44.200.52.133 PrivateIPs: 10.16.10.153

4. Analyzing the Monolithic Application file structure:

```
aws Services Search [Alt+5] Instances | EC2 | us-east-1 EC2 Instance Connect | us-east-1 N. Virginia v vocabs/user3205729-lmran_Muhammad_Owais @ 2171-2006-4673

ubuntu@ip-10-16-10-153:~$ ls ~/resources/codebase_partner/
app  index.js  node_modules  package-lock.json  package.json  public  views
ubuntu@ip-10-16-10-153:~$
```

i-0782d8ee79e96803b (MonolithicAppServer)
PublicIPs: 44.200.52.133 PrivateIPs: 10.16.10.153

Based on the current analysis, we can conclude that a system command is configured to be executed on the ec2 instance which runs the node application as root process on port 80 of the ec2 instance. The application is configured to store data into some remote database.

5. Connecting to MySQL instance to analyze the data by getting the RDS endpoint and exploring the contents present in the database.

Screenshot of the AWS CloudShell interface showing the results of an nmap scan on an RDS MySQL instance.

```

ubuntu@ip-10-16-10-153:~$ nmap -Pn
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-24 20:50 UTC
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.04 seconds
ubuntu@ip-10-16-10-153:~$ nmap -Pn supplierdb.cumg7fuqv4pa.us-east-1.rds.amazonaws.com
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-24 20:50 UTC
Nmap scan type: standard
Host is up (0.0005s latency).
rDNS record for 10.16.30.179: ip-10-16-30-179.ec2.internal
Not shown: 999 filtered ports
PORT      STATE SERVICE
3306/tcp  open  mysql

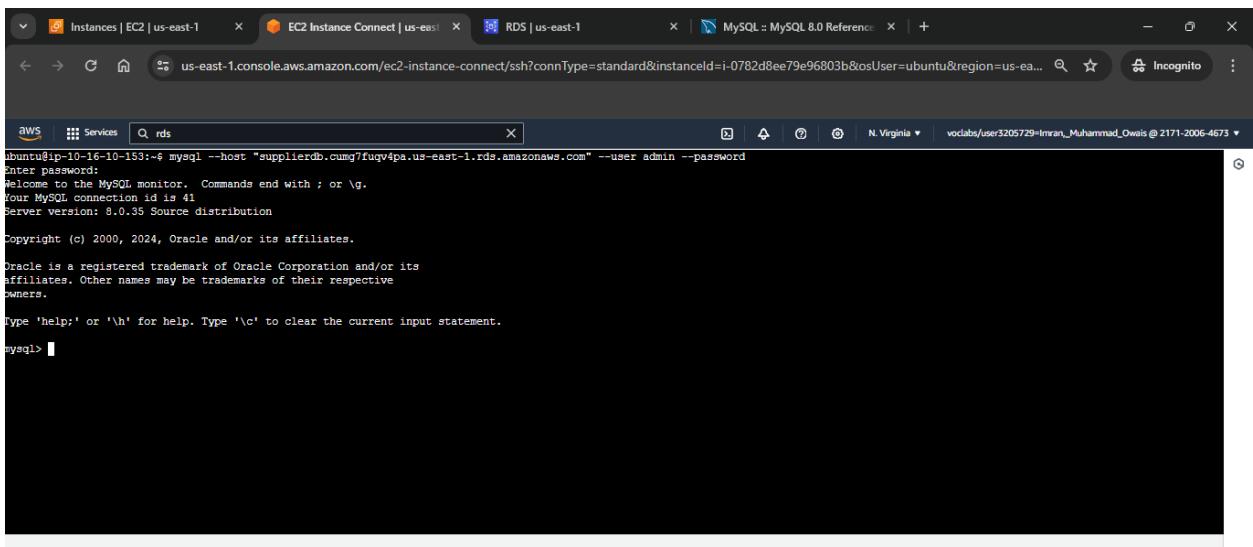
Nmap done: 1 IP address (1 host up) scanned in 6.55 seconds
ubuntu@ip-10-16-10-153:~$ 

```

The screenshot shows the AWS CloudShell interface with the following details:

- CloudShell** tab is active.
- Feedback** tab is visible.
- Instances | EC2 | us-east-1**, **EC2 Instance Connect | us-east-1**, and **RDS | us-east-1** tabs are open in the browser.
- Search** bar contains "rds".
- Summary** section of the RDS console shows the following details:

DB identifier	supplierdb	Status	Available	Role	Instance	Engine	MySQL Community	Recommendations
CPU	2.86%	Class	db.t3.micro	Current activity	1 Connections	Region & AZ	us-east-1a	
- Connectivity & security** tab is selected in the RDS console.
- Endpoint & port** section shows the endpoint as `supplierdb.cumg7fuqv4pa.us-east-1.rds.amazonaws.com` and port as 3306.
- Networking** section shows the availability zone as us-east-1a and VPC as LabVPC (vpc-0f075ab6fa226f64).
- Security** section shows the VPC security group as DBSecurityGroup (sg-0777dce6dde0d2779) and status as Active.
- Events** and **Event subscriptions** sections are listed in the sidebar.
- CloudShell** terminal window shows the execution of the nmap command.
- Feedback** window shows the results of the nmap scan, identifying the host as `i-0782d8ee79e96803b` (MonolithicAppServer) with public IP 44.200.52.133 and private IP 10.16.10.153.
- Bottom status bar shows the date and time as 4/24/2024 4:51 PM.



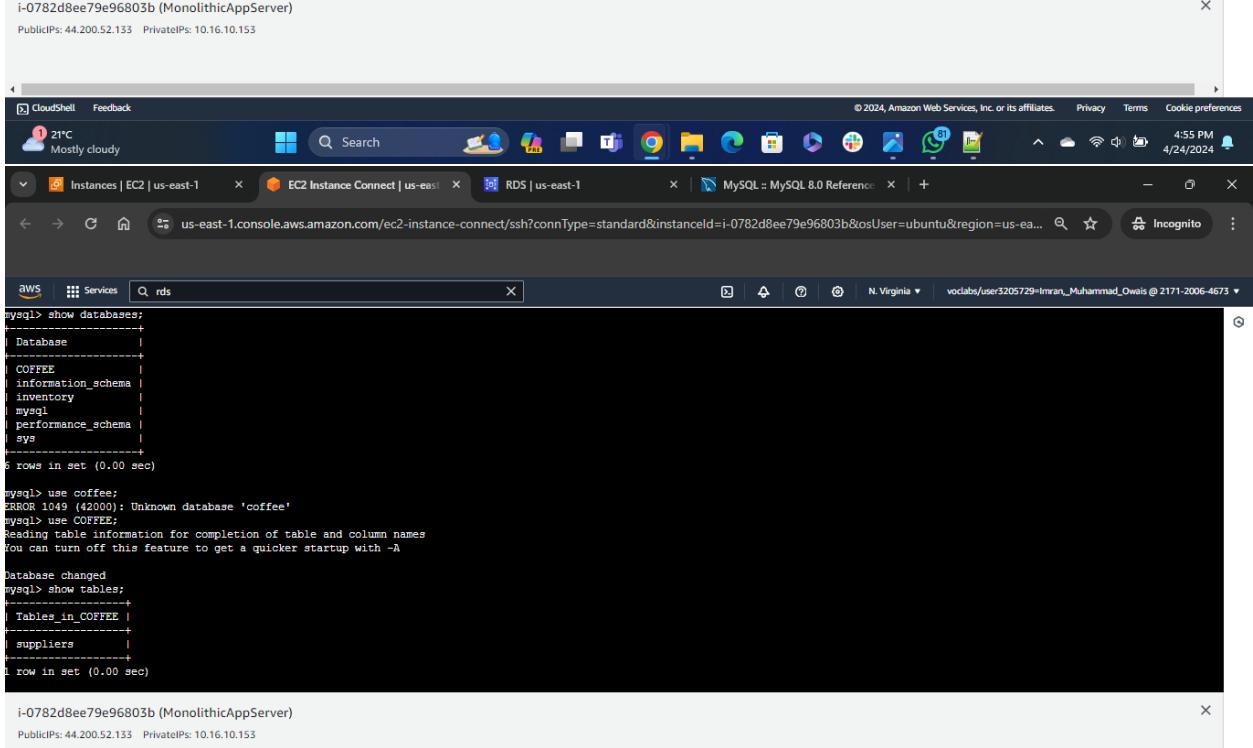
```
ubuntu@ip-10-16-10-153:~$ mysql --host "supplierdb.cung7fugv4pa.us-east-1.rds.amazonaws.com" --user admin --password
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server Version: 8.0.35 Source distribution

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```



```
i-0782d8ee79e96803b (MonolithicAppServer)
Public IPs: 44.200.52.133 Private IPs: 10.16.10.153

CloudShell Feedback
21°C Mostly cloudy Search
Instances | EC2 | us-east-1 EC2 Instance Connect | us-east-1 RDS | us-east-1 MySQL: MySQL 8.0 Reference
MySQL 8.0 Reference
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
4:55 PM 4/24/2024

us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0782d8ee79e96803b&osUser=ubuntu&region=us-ea...
Incognito

AWS Services Q rds

mysql> show databases;
+-----+
| Database |
+-----+
| COFFEE   |
| information_schema |
| inventory |
| mysql     |
| performance_schema |
| sys       |
+-----+
5 rows in set (0.00 sec)

mysql> use coffee;
ERROR 1049 (42000): Unknown database 'coffee'
mysql> use COFFEE;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables in COFFEE |
+-----+
| suppliers        |
+-----+
1 row in set (0.00 sec)

i-0782d8ee79e96803b (MonolithicAppServer)
Public IPs: 44.200.52.133 Private IPs: 10.16.10.153
```



Instances | EC2 | us-east-1 EC2 Instance Connect | us-east-1 RDS | us-east-1 MySQL: MySQL 8.0 Reference - X us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0782d8ee79e96803b&osUser=ubuntu®ion=us-ea... Incognito

AWS Services Q rds N. Virginia v vocabs/user3205729-lmran_Muhammad_Owais @ 2171-2006-4673

```
ERROR 1049 (42000): Unknown database 'coffee'
mysql> use COFFEE;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> show tables;
+-----+
| Tables_in_COFFEE |
+-----+
| suppliers |
+-----+
1 row in set (0.00 sec)

mysql> clear
mysql> cls
    -> :
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'cls' at line 1
mysql> clear;
mysql> select * from suppliers;
+----+-----+-----+-----+-----+-----+
| id | name           | address          | city            | state           | email           | phone          |
+----+-----+-----+-----+-----+-----+
| 1  | Muhammad Owais Imran | 110 Lake St     | Jersey City    | New Jersey     | mimrani@stevens.edu | +1 516 675 7898 |
+----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

i-0782d8ee79e96803b (MonolithicAppServer)
Public IPs: 44.200.52.133 Private IPs: 10.16.10.153



Significance & Importance of this Task

This task helps in determining:

- How the monolithic application works
- Possible ways to identify core services of the monolithic application
- Possible ways to break system into multiple microservices by knowing their dependencies on each other and their workflow pattern

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

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

1. Creating an AWS Cloud9 IDE, and observing the configuration of launched EC2 instance by the Cloud IDE.

AWS Cloud9

us-east-1.console.aws.amazon.com/cloud9control/home?region=us-east-1#/create

Details

Name
MircoservicesIDE
Limit of 60 characters, alphanumeric, and unique per user.

Description - optional
Project IDE For Muhammad Owais Imran
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.

Platform [Info](#)
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.

Amazon Linux 2

Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes

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-06f075ab6f2226f64
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-07dc729d57a3f80a2
Name - Public Subnet1

CloudShell Feedback

21°C Mostly cloudy

Search

CloudShell Feedback

21°C Mostly cloudy

Search

AWS Cloud9

Creating MircoservicesIDE. This can take several minutes. While you wait, see [Best practices for using AWS Cloud9](#).

Environments

Environments (1)

Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
MircoservicesIDE	Open	EC2 instance	Secure Shell (SSH)	Owner	arn:aws:sts::217120064673:assumed-role/vclabs/user3205729=Imran_Muhammad_Owais

CloudShell Feedback

21°C Mostly cloudy

AWS Cloud9

Instances | EC2 | us-east-1

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:search=85d1a49f8c4443b4a6bd9a89fa56f7fa;sort=tag:Name

Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 address
aws-cloud9-Mi...	i-0cc9010998ac48531	Running	t3.small	2/2 checks passed	View alarms	us-east-1a	ec2-44-199-218-166.co...	44.19

i-0cc9010998ac48531 (aws-cloud9-MircoservicesIDE-85d1a49f8c4443b4a6bd9a89fa56f7fa)

Details | Status and alarms [New](#) | Monitoring | Security | Networking | Storage | Tags

Instance summary

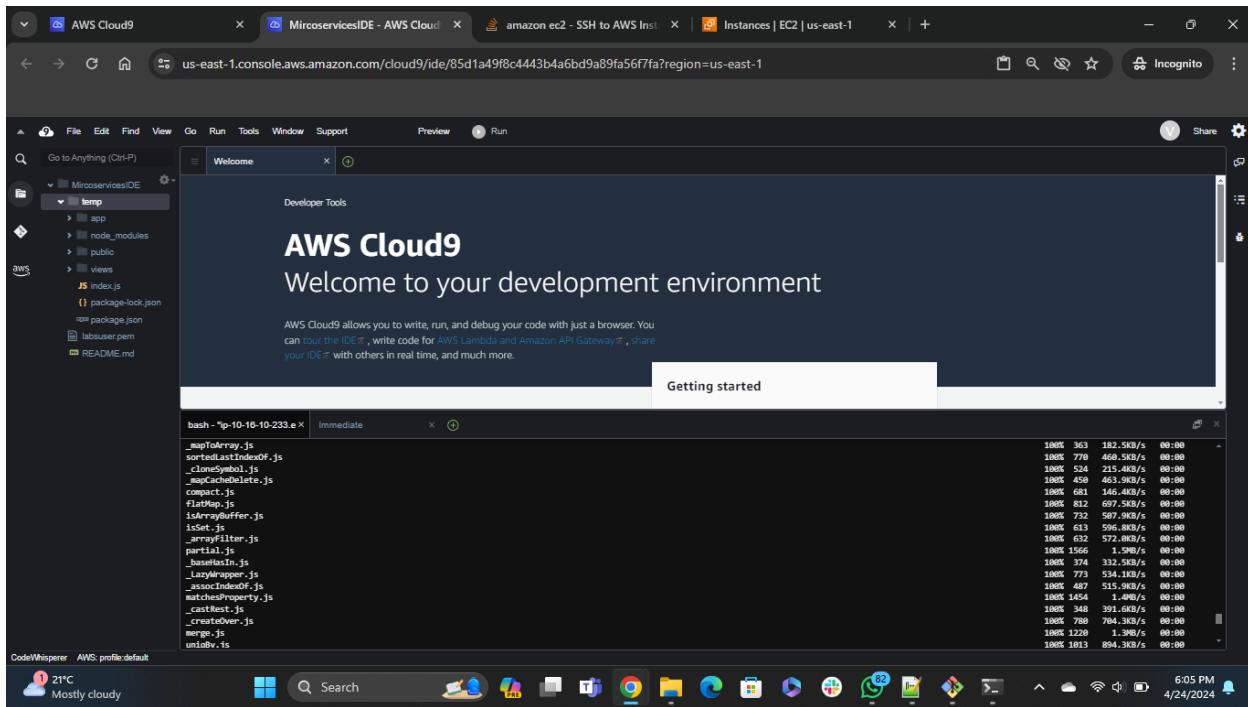
Instance ID	Public IPv4 address	Private IPv4 addresses
i-0cc9010998ac48531 (aws-cloud9-MircoservicesIDE-85d1a49f8c4443b4a6bd9a89fa56f7fa)	44.199.218.166 open address	10.16.10.233
IPv6 address	Instance state	Public IPv4 DNS
-	Running	ec2-44-199-218-166.compute-1.amazonaws.com open address

Task 3.2: Copy the application code to your IDE

1. Create a temp directory in Cloud9 IDE, and copying the codebase of Monolithic application by uploading the KeyPair for connecting with the monolithic application ec2 instance to the Cloud9 instance, and the use of scp to copy all the files.

Following commands were executed to complete the said task:

- Download the KeyPair to the local system and uploaded it to cloud9
- Change the permissions of KeyPair file to make it executable using `chmod 400 ./labsuser.pem`
- `scp -r -i ~/environment/labsuser.pem ubuntu@$appServerPrivIp:/home/ubuntu/resources/codebase_partner/* ~/environment/temp/`



All files of Monolithic Web Application are now copied to the temp directory of Cloud9 IDE.

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

Execute the following commands to complete this task:

- `cd ~/environment`
- `mkdir microservices && cd microservices`
- `mkdir customer, employee`
- `cd customer && cp -r ../../temp/* ./ && cd ../../`
- `cd employee && cp -r ../../temp/* ./ && cd ../../`
- `rm -rf temp`

The screenshot displays two separate sessions of the AWS Cloud9 IDE interface. Both sessions are titled "AWS Cloud9" and show the "Welcome" screen with the message "Welcome to your development environment".

Session 1 (Top):

- File Explorer:** Shows a project structure for "MicoservicesIDE" with a "microservices" folder containing "customer" and "employee" subfolders, and a "temp" folder.
- Terminal:** A bash shell window titled "bash - *p-10-10-10-233.e" with the "Immediate" tab selected. It shows the following command history:

```
veclabs:~/environment/microservices $ cp -r ../../temp/* ./
veclabs:~/environment/microservices $ rm -rf *
veclabs:~/environment/microservices $ mkdir customer && mkdir employee
veclabs:~/environment/microservices $ ls
customer employee
veclabs:~/environment/microservices $
```
- Getting Started:** A white box with the text "Getting started" and a "Get started" button.

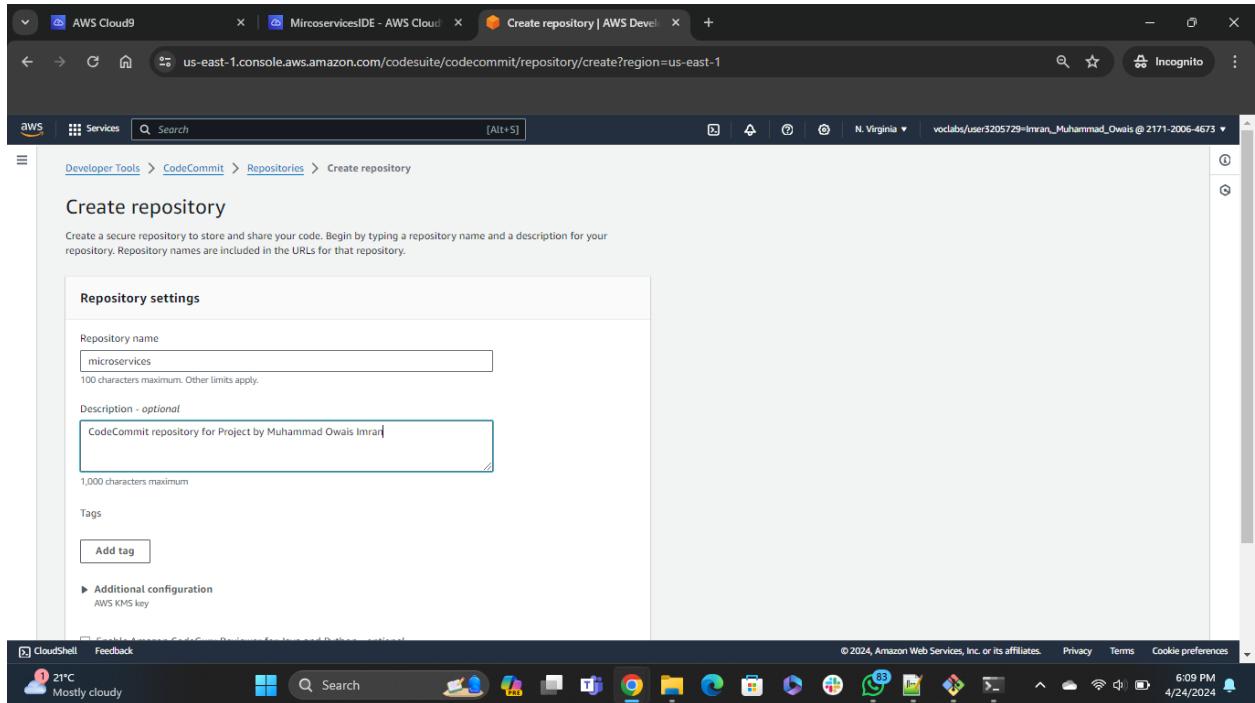
Session 2 (Bottom):

- File Explorer:** Shows a project structure for "MicoservicesIDE" with a "microservices" folder containing "customer" and "employee" subfolders, and a "temp" folder.
- Terminal:** A bash shell window titled "bash - *p-10-10-10-233.e" with the "Immediate" tab selected. It shows the following command history:

```
veclabs:~/environment/microservices $ ls
customer employee
veclabs:~/environment/microservices $ cd customer/
veclabs:~/environment/microservices/customer $ cp -r ../../temp/* ../
veclabs:~/environment/microservices/customer $ cd ..
veclabs:~/environment/microservices $ cd employee/
veclabs:~/environment/microservices/employee $ cp -r ../../temp/* ../
veclabs:~/environment/microservices/employee $ cd ..
veclabs:~/environment $ ls
labuser.pem microservices README.md temp
veclabs:~/environment $ rm -rf temp
veclabs:~/environment $
```
- Getting Started:** A white box with the text "Getting started" and a "Get started" button.

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

1. Create a code commit repository by the name of “microservices” and verifying if code commit repository is created.



The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a sidebar titled 'Developer Tools' with a 'CodeCommit' section expanded, showing options like 'Source', 'Artifacts', 'Build', 'Deploy', 'Pipeline', and 'Settings'. The main content area is titled 'Repositories | AWS Developer' and shows a list of repositories under 'CodeCommit > Repositories'. A single repository named 'microservices' is listed. The table has columns for Name, Description, Last modified, Clone URL, and AWS KMS Key. The 'Clone URL' row contains links for HTTPS, SSH, and HTTPS (GRC). The AWS KMS Key column shows a long ARN. At the bottom of the screen, there's a taskbar with icons for CloudShell, Feedback, and various application icons.

2. Executing the following commands to configure this as a git repository on Cloud9 IDE:

- `git init`
- `git branch -m dev`
- `git add .`
- `git commit -m "two unmodified copies of the application code"`
- `git remote add origin REMOTE_URL_COPY`
- `git push origin -u dev`
- `git config --global user.name "MUHAMMAD_OWAIS_IMRAN"`
- `git config --global user.email mimran1@stevens.edu`

The screenshot shows the AWS Cloud9 IDE interface. The terminal window displays a git push command being run against a repository named 'microservices'. The output shows the process of enumerating objects, calculating deltas, and pushing them to the 'origin' branch.

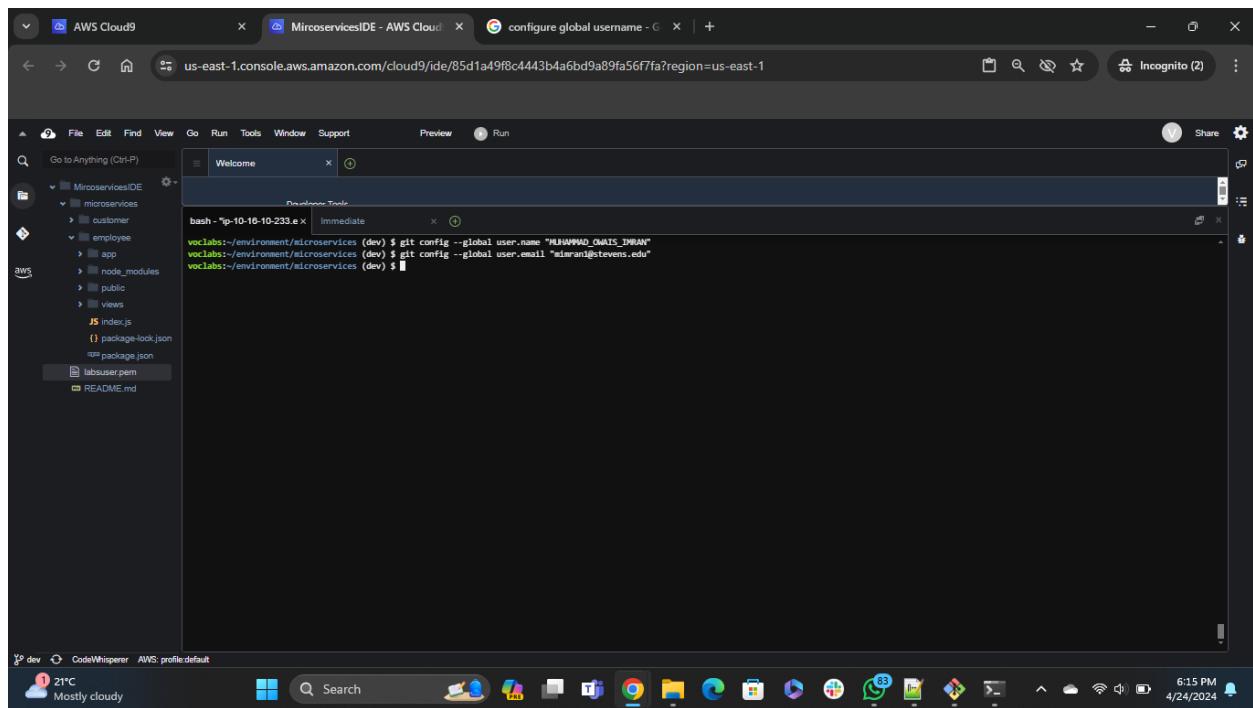
```
git - ip-10-16-10-233.ec2.x Immediate
create mode 100644 employee/node_modules/yallist/package.json
create mode 100644 employee/node_modules/yallist/yallist.js
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/json
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/js/base.js
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-update.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
vclabs:~/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.52 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
 ! [rejected]     dev -> dev
branch 'dev' set up to track 'origin/dev'.
vclabs:~/environment/microservices (dev) $
```

The status bar at the bottom indicates the environment is 'profile/default' and the weather is '21°C Mostly cloudy'.

The screenshot shows the AWS CodeCommit interface. It displays a list of commits for the 'microservices' repository. There is one commit listed:

Commit ID	Commit message	Commit date	Authored date	Author	Committer	Actions
0846e52d	two unmodified copies of the application code	1 minute ago	1 minute ago	EC2 Default User	EC2 Default User	Copy ID Browse

The status bar at the bottom indicates the environment is 'profile/default' and the weather is '21°C Mostly cloudy'.



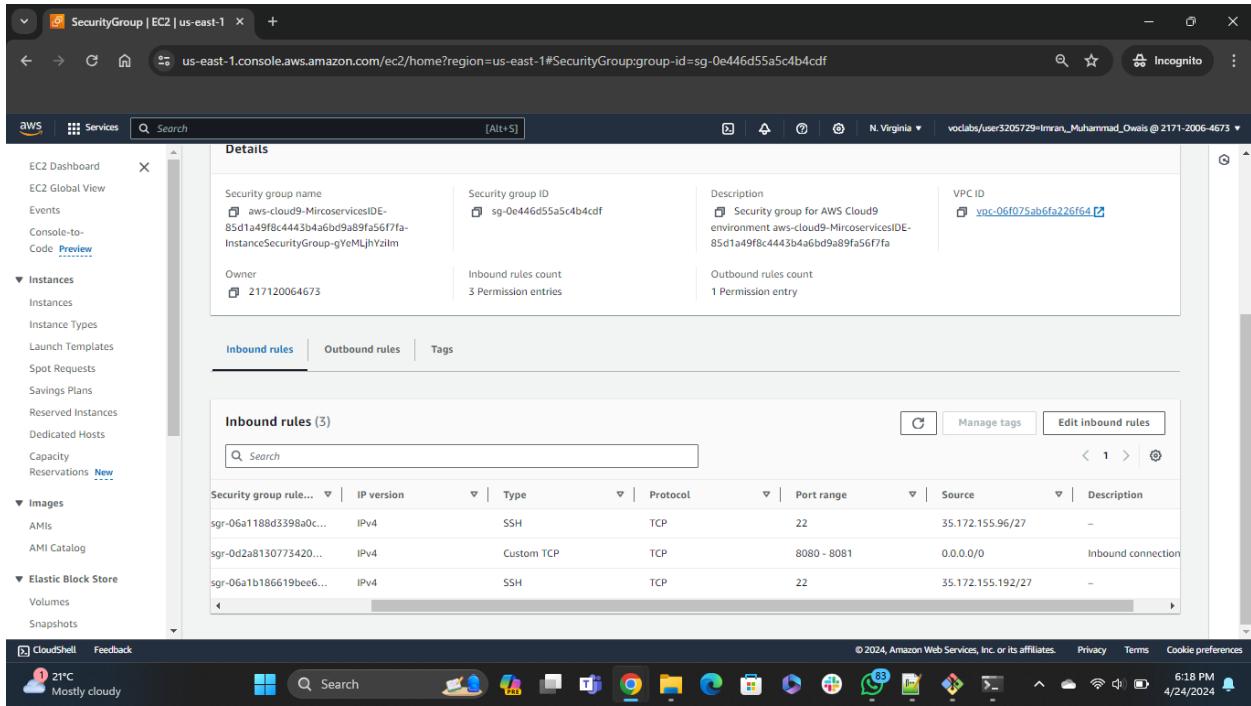
Significance & Importance of this Task

This task helps us to determine:

- How cloud-based IDE can help in code sharing
- How to use Git for version control and code control
- Use case for AWS Cloud9 and CodeCommit

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

Task 4.1: Adjust the AWS Cloud9 instance security group settings



The screenshot shows the AWS Cloud9 Security Group configuration page. The URL is us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroup:group-id=sg-0e446d55a5c4b4cdf. The page displays the following details:

Security group name	sg-0e446d55a5c4b4cdf	Description	VPC ID
aws-cloud9-MircoservicesIDE-85d1a49f8c4443b4a6bd9a89fa56f7fa-InstanceSecurityGroup-gYeMujhYzilm		Security group for AWS Cloud9 Environment aws-cloud9-MircoservicesIDE-85d1a49f8c4443b4a6bd9a89fa56f7fa	vpc-06f075ab6fa226f64
Owner	217120064673	Inbound rules count	3 Permission entries
		Outbound rules count	1 Permission entry

The Inbound rules section shows three entries:

Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-06a1188d3398a0c...	IPv4	SSH	TCP	22	35.172.155.96/27	-
sgr-0d2a8130773420...	IPv4	Custom TCP	TCP	8080 - 8081	0.0.0.0/0	Inbound connection
sgr-06a1b186619bee6...	IPv4	SSH	TCP	22	35.172.155.192/27	-

Task 4.2: Modify the source code of the *customer* microservice

1. 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.
2. Edit the customer/app/models/supplier.model.js file. Delete the unnecessary functions in it so that what remains are only read-only functions.
3. Later in the project, when you deploy the microservices behind an Application Load Balancer, you will want employees 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, edit the customer/views/nav.html file
4. You don't want customers to see the Add a new supplier button or any edit buttons next to supplier rows. To implement these changes, edit the customer/views/supplier-list-all.html file
5. Because the customer microservice doesn't need to support read-write actions, DELETE the following .html files from the customer/views directory
6. Edit the customer/index.js file as needed to account for the fact that the node application will now run on Docker containers

Screenshots of all steps attached:

supplier.controller.js

```
const Supplier = require("../models/supplier.model.js");
const { body, validationResult } = require("express-validator");

exports.findAll = (req, res) => {
  Supplier.getAll((err, data) => {
    if (err)
      res.render("500", { message: "There was a problem retrieving the list of suppliers"});
    else res.render("supplier-list-all", { suppliers: data });
  });
};

exports.findById = (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 });
  });
};
```

supplier.model.js

```
host: dbConfig.APP_DB_HOST,
user: dbConfig.APP_DB_USER,
password: dbConfig.APP_DB_PASSWORD,
database: dbConfig.APP_DB_NAME
});

Supplier.getAll = result => {
  db_connection.query("SELECT * FROM suppliers", (err, res) => {
    if (err) {
      console.log("Error: ", err);
      result(null, null);
      return;
    }
    console.log("Suppliers: ", res);
    result(null, res);
  });
};

Supplier.findById = (supplierId, result) => {
  db_connection.query("SELECT * FROM suppliers WHERE id = ${supplierId}", (err, res) => {
    if (err) {
      console.log("Error: ", err);
      result(null, null);
      return;
    }
    if (res.length) {
      console.log(`Found supplier: ${res[0]}`);
      result(null, res[0]);
      return;
    }
    result({ kind: "not_found", null });
  });
};

module.exports = Supplier;
```

The screenshot displays the AWS Cloud9 IDE interface with two code editors open:

- Supplier Controller:** A JavaScript file containing logic for handling supplier requests. It includes methods for finding all suppliers, adding a new supplier, updating an existing supplier, and removing a supplier.
- Supplier Model:** A JavaScript file defining the supplier model, which includes properties for name, address, city, state, email, phone, and id.
- Supplier List All:** An HTML file containing a table with columns for Name, Address, City, State, Email, and Phone, used for displaying a list of suppliers.
- Nav:** An HTML file containing the navigation bar for the application.

Below the code editors is a terminal window showing a bash session on a Linux instance (ip-10-16-10-233.e). The terminal output includes:

```
bash - *p-10-16-10-233.e* Immediate vclabs~/environment/microservices (dev) $ |
```

The system status bar at the bottom indicates the following:

- Temperature: 21°C
- Weather: Mostly cloudy
- Date: 4/24/2024
- Time: 6:26 PM

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

1. Create a Dockerfile for customer, Build it, run to launch a test container, and test the application if customer microservice is working as expected.

The screenshot shows two instances of the AWS Cloud9 IDE interface. Both instances have the same workspace structure:

- Root directory: MicroservicesIDE
- Sub-directory: microservices
- Sub-sub-directory: customer
- Sub-sub-sub-directory: app
- Files in app directory:
 - config
 - controller
 - supplier.controller.js
 - supplier.model.js
 - models
 - node_modules
 - public
 - views
 - nav.html
 - supplier-list-all.html
 - index.js
 - Dockerfile
 - index.js
 - package-lock.json
 - package.json
 - employee
 - subuser.pem
 - README.md

In the bottom terminal window of both instances, the Dockerfile content is displayed:

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

Below the Dockerfile, the command `touch Dockerfile` is run in the Immediate tab of the terminal.

In the second instance (bottom), the terminal output shows the Dockerfile being built and run:

```
Step 0/7 : FROM node:11-alpine
Step 1/7 : RUN mkdir -p /usr/src/app
Step 2/7 : WORKDIR /usr/src/app
Step 3/7 : CMD ["npm", "run", "start"]
Step 4/7 : EXPOSE 8080
Step 5/7 : Running in e8310f2045c0
Step 6/7 : Removing intermediate container e8310f2045c0
Step 7/7 : CMD ["npm", "run", "start"]
Step 8/8 : Running in 1a0a2cefef15
Step 9/9 : Removing intermediate container 1a0a2cefef15
Successfully built 26fb8ab1e9743
Successfully tagged customer:latest
voclabs:-/environment/microservices/customer (dev) $
```

The status bar at the bottom of the interface indicates the current environment and system status.

AWS Cloud9 | MircoservicesIDE - AWS Cloud9

us-east-1.console.aws.amazon.com/cloud9/ide/85d1a49f8c4443b4a6bd9a89fa56f7fa?region=us-east-1

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl+P)

MircoservicesIDE

- microservices
 - customer
 - app
 - config
 - controller
 - models
 - Dockerfile

JS supplier.controller.js JS supplier.model.js nav.html supplier-list-all.html JS index.js Dockerfile

```

1 FROM node:11-alpine
2 RUN apk add --no-cache curl
3 WORKDIR /usr/src/app
4 COPY . .
5 RUN npm install
6 REMOVE apk
7 CMD ["npm", "run", "start"]

```

bash - *p-10-16-10-233.e x Immediate

Step 4/7 : COPY .
--> acf411eef66
Step 5/7 : RUN npm install
--> Running in c76756525081
npm WARN coffee_spider@1.0.0 No repository field.
audited 78 packages in 0.68s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
 run 'npm audit fix' to fix them, or 'npm audit' for details
 Removing intermediate container c76756525081
--> Step 6/7 : EXPOSE 8080
--> Running in e831bf2845c0
Removing intermediate container e831bf2845c0
--> 5d5860f3539
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in 26f8a1e9743
Removing intermediate container 1a9a2cfef15f
--> 26f8a1e9743
Successfully built 26f8a1e9743
Successfully tagged customer:latest
vocla:~/environment/microservices/customer (dev) \$ docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
customer latest 26f8a1e9743 29 seconds ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.9MB
vocla:~/environment/microservices/customer (dev) \$

CodeWhisperer AWS profile/default

21°C Mostly cloudy

Search

6:12 Dockerfile Spaces: 4

AWS Cloud9 | Cloud9 | us-east-1

us-east-1.console.aws.amazon.com/cloud9/ide/85d1a49f8c4443b4a6bd9a89fa56f7fa?region=us-east-1

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl+P)

MircoservicesIDE

- microservices
 - customer
 - app
 - config
 - controller
 - models
 - Dockerfile

JS supplier.controller.js JS supplier.model.js nav.html supplier-list-all.html JS index.js Dockerfile

```

1 FROM node:11-alpine
2 RUN apk add --no-cache curl
3 WORKDIR /usr/src/app
4 COPY . .
5 RUN npm install
6 REMOVE apk
7 CMD ["npm", "run", "start"]

```

bash - *p-10-16-10-233.e x Immediate

vocla:~/environment/customer_1 \$ docker run -d --name customer_1 -p 8888:8888 -e APP_DB_HOST="\$dbEndpoint" customer_1
2de30c411a52-92:b61e563535713223186kecc2ed6437e1205a442951a
vocla:~/environment \$ docker ls

docker: 'ls' is not a docker command.
See 'docker --help'.
vocla:~/environment \$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2de30c411a52	customer_1	"docker-entrypoint..."	50 seconds ago	Up 49 seconds	0.0.0.0:8888->8888/tcp, :::8888->8888/tcp	customer_1

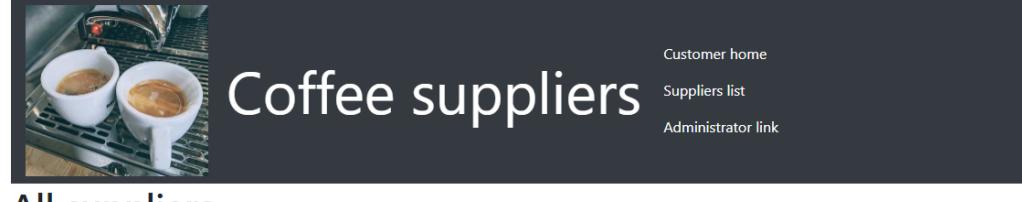
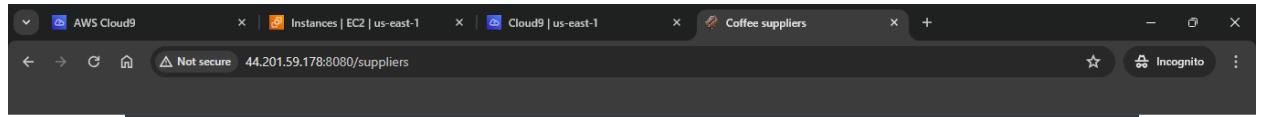
vocla:~/environment \$

CodeWhisperer AWS profile/default

21°C Mostly cloudy

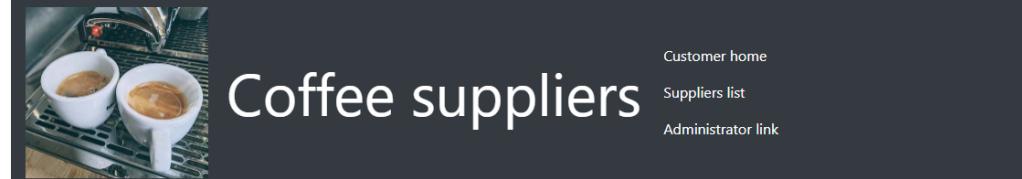
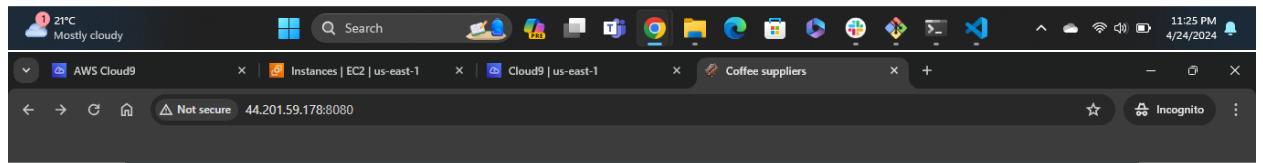
Search

11:24 PM 4/24/2024



All suppliers

Name	Address	City	State	Email	Phone
Muhammad Owais Imran	110 Lake St	Jersey City	New Jersey	mimran1@stevens.edu	+1 516 675 7898



Welcome

Use this app to keep track of your coffee suppliers

[List of suppliers](#)



2. Commit the code onto Github and verify if it is updated on Codecommit

The screenshot shows a browser window with multiple tabs open. The active tab is titled "Commit 9825e30aab12cd1331e706088bc1b0d3decb4580". The left sidebar of the AWS Cloud9 interface is visible, showing navigation links for "Source", "CodeCommit", "Getting started", "Repositories", "Code", "Pull requests", "Commits", "Branches", "Git tags", "Settings", "Approval rule templates", "Artifacts", "Build", "Deploy", "Pipeline", and "Settings". The main content area displays the commit details, including the author (MUHAMMAD_OWAIS_IMRAN), authored date (Just now), committer (MUHAMMAD_OWAIS_IMRAN), commit message (Modified Customer Microservice), and the Dockerfile content:

```
1 + FROM node:11-alpine
2 + RUN mkdir -p /usr/src/app
3 + WORKDIR /usr/src/app
4 + COPY .
5 + RUN npm install
6 + EXPOSE 8080
```

Task 4.4: Modify the source code of the *employee* microservice

1. In the employee/app/controller/supplier.controller.js file, for all the redirect calls, prepend /admin to the path
2. In the employee/index.js file, update the app.get calls, app.post calls, and a port number
3. In the employee/views/supplier-add.html and employee/views/supplier-update.html files, for the form action paths, prepend /admin to the path
4. In the employee/views/supplier-list-all.html and employee/views/home.html files, for the HTML paths, prepend /admin to the path.
5. In the employee/views/header.html file, modify the title to be Manage coffee suppliers
6. Edit the employee/views/nav.html file.

Screenshots of all steps attached below:

Supplier Controller Logic (supplier.controller.js):

```
95     if (err.kind === "not_found") {
96       res.status(404).send({
97         message: `Not found Supplier with id ${req.params.id}`});
98     } else {
99       res.render("500", {message: `Could not delete Supplier with id ${req.body.id}`});
100    }
101  });
102}
103
104 exports.removeAll = (req, res) => {
105   res.redirect("/admin/suppliers");
106}
107
108 module.exports = supplierController;
```

Terminal Output (supplier.controller.js):

```
bash -cp-10-16-10-233.x | Immediate
voelabs:/environment/microservices/employee (dev) $ grep -n 'redirect' app/controller/supplier.controller.js
51:           else res.redirect("/admin/suppliers");
52:     } else res.redirect("/admin/suppliers");
53:   } else res.redirect("/admin/suppliers");
54:   else res.redirect("/admin/suppliers");
voelabs:/environment/microservices/employee (dev) $
```

Index Logic (index.js):

```
26 // show the add supplier form
27 app.get('/admin/supplier-add', (req, res) => {
28   res.render('supplier-add', {});
29 });
30 // receive the add supplier POST
31 app.post('/admin/supplier-add', supplier.create);
32 // show the update form
33 app.get('/admin/supplier-update/:id', supplier.findOne);
34 // receive the update POST
35 app.post('/admin/supplier-update/:id', supplier.update);
36 // receive the POST to delete a supplier
37 app.post('/admin/supplier-remove/:id', supplier.remove);
38 // handle add
39
40 module.exports = indexController;
```

Terminal Output (index.js):

```
bash -cp-10-16-10-233.x | Immediate
voelabs:/environment/microservices/employee (dev) $ grep -n 'app.get' index.js
22 app.get('/', (req, res) => {
23   res.render('home');
24 });
25 app.get('/suppliers', supplier.findAll);
26 app.get('/supplier-add', (req, res) => {
27   res.render('supplier-add', {});
28 });
29 app.post('/supplier-add', supplier.create);
30 app.get('/supplier-update/:id', supplier.findOne);
31 app.post('/supplier-update/:id', supplier.update);
32 app.post('/supplier-remove/:id', supplier.remove);
33 app.get('/admin/supplier-add', supplier.create);
34 app.get('/admin/supplier-update/:id', supplier.update);
35 app.get('/admin/supplier-remove/:id', supplier.remove);
36 module.exports = indexController;
voelabs:/environment/microservices/employee (dev) $ grep -n 'app.get' index.js
22 app.get('/admin', (req, res) => {
23   app.get('/admin/suppliers', supplier.findAll);
24   app.get('/admin/supplier-add', (req, res) => {
25     res.render('admin-supplier-add', {});
26   });
27   app.post('/admin/supplier-add', supplier.create);
28   app.get('/admin/supplier-update/:id', supplier.findOne);
29   app.post('/admin/supplier-update/:id', supplier.update);
30   app.get('/admin/supplier-remove/:id', supplier.remove);
31   app.post('/admin/supplier-remove/:id', supplier.remove);
32   module.exports = indexController;
voelabs:/environment/microservices/employee (dev) $
```

AWS Cloud9 | 9825e30aab12cd1331e706088b | MircoservicesIDE - AWS Cloud9 | Coffee suppliers | Incognito

us-east-1.console.aws.amazon.com/cloud9/ide/85d1a49f8c4443b4a6bd9a89fa56f7fa?region=us-east-1

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

Mircoservices:DE

- └─ microservices
 - └─ customer
 - └─ employee
 - └─ app
 - └─ config
 - └─ controller
 - JS supplier contr
 - └─ models
 - └─ node_modules
 - └─ public
 - └─ views
 - 404.html
 - 500.html
 - footer.html
 - header.html
 - home.html
 - nav.html
 - supplier-add.htm
 - supplier-form-fix.htm
 - supplier-list-all.htm
 - supplier-update.htm

aws

bash - *p-10-16-10-233.e x Immediate

```
views->/environment/microservices/employees (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-remove.html:8:    <form action="/supplier-remove/{{id}}" method="POST">
views->/environment/microservices/employee (dev) $ grep -n 'action' views/*
views/supplier-add.html:11:    <form action="/admin/supplier-add" method="POST">
views/supplier-update.html:12:    <form action="/admin/supplier-update" method="POST">
views/supplier-remove.html:8:    <form action="/admin/supplier-remove/{{id}}" method="POST">
views->/environment/microservices/employee (dev) $ |
```

CodeWisperer AWS: profile:default

21°C Mostly cloudy

AWS Cloud9 | 9825e30aab12cd1331e706088b | MircoservicesIDE - AWS Cloud9 | Coffee suppliers | Incognito

us-east-1.console.aws.amazon.com/cloud9/ide/85d1a49f8c4443b4a6bd9a89fa56f7fa?region=us-east-1

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

Mircoservices:DE

- └─ microservices
 - └─ customer
 - └─ employee
 - └─ app
 - └─ config
 - └─ controller
 - JS supplier contr
 - └─ models
 - └─ node_modules
 - └─ public
 - └─ views
 - 404.html
 - 500.html
 - footer.html
 - header.html
 - home.html
 - nav.html
 - supplier-add.htm
 - supplier-form-fix.htm
 - supplier-list-all.htm
 - supplier-update.htm

aws

bash - *p-10-16-10-233.e x Immediate

```
views->/environment/microservices/employees (dev) $ grep -n 'href' views/supplier-list-all.html views/home.html
views/supplier-list-all.html:1:    <a href="#">List of suppliers</a>
views/supplier-list-all.html:5:    <button class="badge badge-success" href="/admin/supplier-add">Add a new supplier</button>
views/home.html:7:    <p><a href="#">List of suppliers</a></p>
views->/environment/microservices/employee (dev) $ grep -n 'href' views/supplier-list-all.html views/home.html
views/supplier-list-all.html:2:    <button class="badge badge-success" href="/admin/supplier-update/{{id}}">Edit</button>
views/supplier-list-all.html:3:    <button class="badge badge-success" href="/admin/supplier-remove/{{id}}">Delete</button>
views/home.html:7:    <p><a href="#">List of suppliers</a></p>
views->/environment/microservices/employee (dev) $ |
```

CodeWisperer AWS: profile:default

21°C Mostly cloudy

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are three tabs: "AWS Cloud9", "9825e3aab12cd133le706088b", and "MircoservicesIDE - AWS Cloud9". The main area has a file browser on the left showing a directory structure for "MircoservicesIDE" containing "microservices", "customer", "employee", "app", "config", and "controller" subfolders. Below the file browser are two code editors: one for "supplier-list-all.html" and another for "header.html". A terminal window titled "bash" is open, showing a command-line session with several grep and curl commands. The terminal output includes URLs like "/supplier-list-all.html", "/supplier-add", and "/admin/supplier-add". The bottom right corner shows the system tray with the date and time (4/24/2024, 11:42 PM).

```
supplier-list-all.html
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <link rel="stylesheet" href="/css/bootstrap.min.css" />
6     <link rel="stylesheet" href="/css/base.css" />
7     <title>Manage coffee suppliers</title>
8   </head>
9   <body>
10
11
12
13
14
15
16
17
18
19
20
21

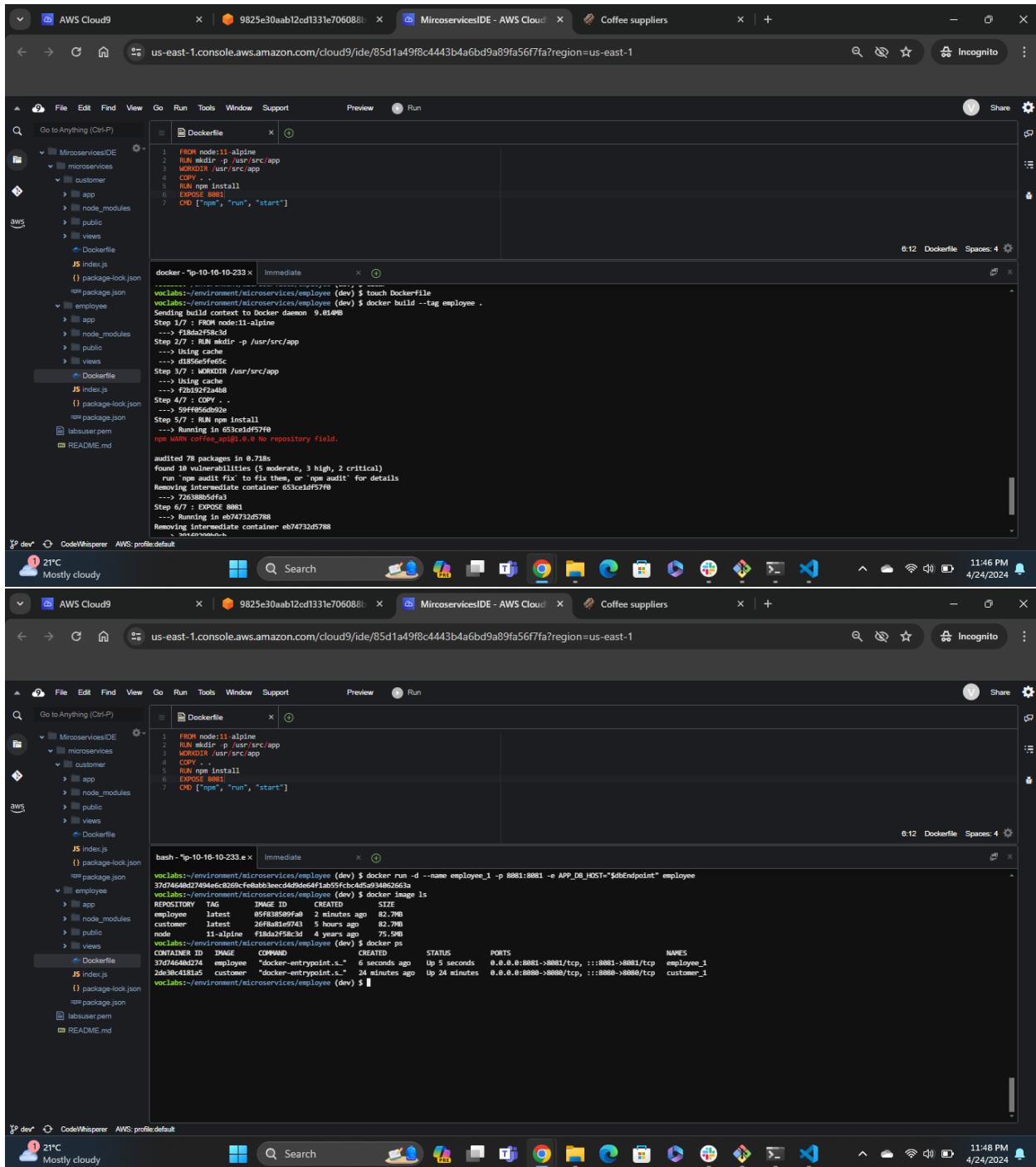
home.html
header.html

bash *-ip-10-16-10-233.e x | Immediate x

curlabs:~/environment/microservices/employee (dev) $ grep -n 'href' views/supplier-list-all.html views/home.html
views/supplier-list-all.html:27:           <a href="#">Edit</a></td>
views/supplier-list-all.html:32:           <h4><a class="badge badge-success" href="/supplier-add">Add a new supplier</a></h4>
```

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

Create a Dockerfile for employee, Build it, run to launch a test container, and test the application if employee microservice is working as expected.



The screenshot shows the AWS Cloud9 IDE interface with two windows open. The left window displays the file structure of a project named 'MicroservicesIDE' containing several subfolders like 'customer', 'employee', 'node_modules', 'public', 'views', and 'aws'. Inside the 'aws' folder, there is a 'Dockerfile' and a 'Dockerfile.js'. The right window shows the terminal output of a build process for the 'employee' service. The Dockerfile content is:

```
FROM node:11-alpine
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . .
RUN npm install
EXPOSE 8081
CMD ["npm", "run", "start"]
```

The terminal output shows the build command being run, followed by a detailed log of the Docker build steps:

```
docker - *ip-10-10-10-233.x* Immediate
.
.
.
Step 1/7 : FROM node:11-alpine
--> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
--> f2b592f2a2a8
Step 3/7 : WORKDIR /usr/src/app
--> f2b592f2a2a8
Step 4/7 : COPY . .
--> f2b592f2a2a8
Step 5/7 : RUN npm install
--> Running in 653e3df5790
npm WARN coffee_api@1.0.0 No repository field.

audited 78 packages in 0.71s
Found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
Run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 653e3df5790
--> 726388b5df3
Step 6/7 : EXPOSE 8081
--> Running in eb74732d5788
Removing intermediate container eb74732d5788
.
.
.

bash - *ip-10-10-10-233.x* Immediate
.
.
.
REPOSITORY TAG IMAGE ID CREATED SIZE
employee latest 0f5f8380f9a9 2 minutes ago 82.7MB
customer latest 20f0a2ff7f43 2 minutes ago 82.7MB
node:11-alpine f18da2f58c3d 4 years ago 25.5MB
.
.
.

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
37d7464dd274 employee "docker-entrypoint.s..." 6 seconds ago Up 5 seconds 0.0.0.0:8081->8081/tcp employee_1
2de3b4c181a5 customer "docker-entrypoint.s..." 24 minutes ago Up 24 minutes 0.0.0.0:8080->8080/tcp, ::1:8080->8080/tcp customer_1
.
.
.
```

A screenshot of a web browser window titled "Manage coffee suppliers". The page displays a header with a coffee machine image and the title "Manage coffee suppliers". On the right, there is a sidebar with links: "Administrator home", "Suppliers list", and "Customer home". The main content area is titled "All suppliers" and lists one supplier entry:

Name	Address	City	State	Email	Phone
Muhammad Owais Imran	110 Lake St	Jersey City	New Jersey	mimran1@stevens.edu	+1 516 675 7898

At the bottom left, there is a green button labeled "Add a new supplier".

A screenshot of a web browser window titled "Manage coffee suppliers" showing the "supplier-add" form. A blue banner at the top states "All fields are required". The form consists of several input fields:

- Name: Muhammad Owais Imran - Microservice (placeholder)
- Address: 110 Lake St (placeholder)
- City: Jersey City (placeholder)
- State: NJ (placeholder)
- Email: mimran1@stevens.edu (placeholder)
- Phone: +1 516 675 7898 (placeholder)

At the bottom left of the form is a blue "Submit" button.

The screenshot displays two consecutive screenshots of a web application interface, likely a microservices-based platform, showing the management of coffee suppliers.

Screenshot 1: Supplier Update Form

The top screenshot shows a "Customer home" page with a banner stating "All fields are required". The form fields are as follows:

- Name:** Muhammad Owais Imran - Microservice
- Address:** 110 Lake St
- City:** Jersey City
- State:** New Jersey
- Email:** mimran1@stevens.edu
- Phone:** +1 516 675 7898

Screenshot 2: Delete Confirmation Dialog

The bottom screenshot shows the same form with a modal dialog titled "Delete supplier". The dialog contains the message: "Are you sure you want to delete supplier Muhammad Owais Imran?". It features "Close" and "Delete this supplier" buttons. The "Delete this supplier" button is highlighted in red.

Both screenshots show a standard Windows taskbar at the bottom with various pinned icons and system status indicators like battery level and signal strength.

Verifying the status of all the containers with the command `docker ps`:

```

Dockerfile
FROM node:11-alpine
WORKDIR /usr/src/app
COPY . .
RUN npm install
EXPOSE 8888
CMD ["npm", "run", "start"]

bash -> ip-10-16-10-233.e | Immediate
vclabs:~/environment/microservices/employees (dev) $ docker run -d --name employee_1 -p 8881:8881 -e APP_DB_ENDPOINT=$dbEndpoint employee
37d74646d27494ec0259cfebab33ec4d494ed4f1ab55fc4d55e34d626563a
vclabs:~/environment/microservices/employees (dev) $ docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED          SIZE
employee           latest   45f83359fa00  1 minute ago   82.7MB
customer           latest   26f8a1e1d743  5 hours ago   82.7MB
node               11-alpine f18da2f58c3d  4 years ago   75.9MB
vclabs:~/environment/microservices/employees (dev) $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
37d74646d274...    "docker-entrypoint..."   6 seconds ago     Up 5 seconds       0.0.0.0:8881->8881/tcp, ::1:8881->8881/tcp   employee_1
26f8a1e1d743...   "docker-entrypoint..."   24 minutes ago    Up 24 minutes      0.0.0.0:8880->8880/tcp, ::1:8880->8880/tcp   customer_1
vclabs:~/environment/microservices/employees (dev) $ [ ]

```

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

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

The screenshot shows the AWS Cloud9 IDE interface. On the left, the file tree displays the project structure for 'MicroservicesIDE' with 'employee' and 'customer' subfolders. In the center, the 'index.js' file is open, showing the following code:

```
37 app.post('/admin/supplier-remove/:id', supplier.remove);
38 // handle 404
39 app.use(function (req, res, next) {
40   res.status(404).render("404");
41 })
42
43 // set port. listen for requests
44 const app_port = process.env.APP_PORT || 8080
45 app.listen(app_port, () => {
46   console.log(`Server is running on port ${app_port}`);
47 });
48 }
```

Below the code editor is a terminal window titled 'bash *ip-10-16-10-233.x' showing Docker commands being run:

```
vecTab:~/environment/microservices/employee (dev)$ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="sqlite:///db.sqlite3" employee
37d74640d27404e6c0269fedabb3ec4d8de64f1ab55fc934062663a
vecTab:~/environment/microservices/employee (dev)$ docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
employee latest 05f83599f9d0 2 minutes ago 82.7MB
customer latest 202d8d55743 2 hours ago 32.7MB
node 11-alpine f18dca2f58c3d 4 hours ago 75.5MB
vecTab:~/environment/microservices/employee (dev)$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
37d74640d274 employee "docker-entrypoint.s..." 6 seconds ago Up 5 seconds 0.0.0.0:8081>8081/tcp employee_1
2de30c4181a5 customer "docker-entrypoint.s..." 24 minutes ago Up 24 minutes 0.0.0.0:8080>8080/tcp, 0.0.0.0:8080>8080/tcp customer_1
vecTab:~/environment/microservices/employee (dev)$
```

A status bar at the bottom indicates the weather as '21°C Mostly cloudy' and the date/time as '4/24/2024 11:52 PM'.

2. Rebuild the Docker image for the employee microservice

The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes tabs for 'AWS Cloud9', '9825e30aab12cd1331e706088...', 'MircoservicesIDE - AWS Cloud...', and 'Manage coffee suppliers'. The main workspace has a 'File' menu with options like 'Edit', 'Find', 'View', 'Go', 'Run', 'Tools', 'Window', 'Support', and 'Preview'. A 'Run' button is visible in the top right.

The left sidebar displays a file tree for a project named 'MircoservicesIDE'. The tree includes 'aws', 'customer', 'employee', 'node_modules', 'public', and 'views' directories, along with 'Dockerfile', 'index.js', 'package-lock.json', and 'package.json' files.

The central workspace contains two tabs: 'Dockerfile' and 'JS index.js'. The 'index.js' tab shows the following code:

```
37 | app.post("/admin/supplier-remove/:id", supplier.remove);
38 |
39 | app.use(function (req, res, next) {
40 |   res.status(404).render("404", {});
41 |
42 |
43 |
44 |   // set port, listen for requests
45 |   const app_port = process.env.APP_PORT || 8080;
46 |   app.listen(app_port, () => {
47 |     console.log(`Server is running on port ${app_port}`);
48 |   });
49 |
50 | }
```

The bottom workspace shows a terminal window titled 'docker - *ip-10-16-10-233 x'. The terminal output is as follows:

```
Step 3/7 : WORKDIR /usr/src/app
--> Using cache
--> F2b392f24b8
Step 4/7 : COPY .
--> 81ed6bccdd1d
Step 5/7 : RUN npm install
--> Running in e602925fb0f4
npm WARN coffee_uglify@0.0.1 No repository field.

audited 78 packages in 0.834s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
  run `npm audit --force` to completely update your package(s)
--> 72156a77b057
Step 6/7 : EXPOSE 8080
--> Running in 3ea12c698a2d
Removing intermediate container 3ea12c698a2d
--> 45570a314149
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in 4eb05b513534
Removing intermediate container 4eb05b513534
--> ebb4d887961
Successfully built ebb4d887961
Successfully tagged employee:latest
vocTab:-/environment/microservices/employee (dev) $
```

The status bar at the bottom shows 'dev', 'CodeWhisperer', 'AWS.profile.default', '21°C', 'Mostly cloudy', '11:53 PM', and '4/24/2024'.

Task 4.7: Check code into CodeCommit

Push the updated code on to Code Commit

The screenshot shows the AWS Cloud9 IDE interface. On the left, a file tree displays the project structure: MircoservicesIDE, microservices (customer, app, node_modules, public, views), Dockerfile, index.js, package-lock.json, package.json, employee, README.md, labuser.pem. The main editor window shows a portion of index.js with several changes highlighted. Below the editor is a terminal window showing the output of a git push command:

```
git -> git add .  
modified: employee/views/index.html  
modified: employee/views/supplier-add.html  
modified: employee/views/supplier-list-all.html  
modified: employee/views/supplier-update.html  
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
employee/index.html  
no changes added to commit (use "git add" and/or "git commit -a")  
veclabs:~/environment/microservices (dev) $ git add .  
veclabs:~/environment/microservices (dev) $ git commit -m "Completed Employee Microservice"  
[dev c7f1ee] Completed Employee Microservice  
9 files changed, 28 insertions(+), 20 deletions(-)  
create mode 100644 employee/Dockerfile  
veclabs:~/environment/microservices (dev) $ git push origin  
Enumerating objects: 28, done.  
Counting objects: 28, done.  
Delta compression using up to 2 threads.  
Compressing objects: 100% (15/15), done.  
Writing objects: 100% (15/15), 1.64 KiB | 839.00 KiB/s, done.  
Total 15 (delta 8), reused 0 (delta 0), pack-reused 0  
remote: Validating objects: 100%  
remote: To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices  
9825e30a..c7f1ee dev -> dev  
veclabs:~/environment/microservices (dev) $
```

Verifying the code update on CodeCommit

The screenshot shows the AWS CodeCommit interface. The left sidebar navigation includes Developer Tools, CodeCommit, Source (Getting started, Repositories, Pull requests, Commits, Branches, Git tags, Settings, Approval rule templates), Artifacts (CodeArtifact, Build, CodeBuild), Deploy (CodeDeploy), Pipeline (CodePipeline), and Settings. The main content area shows the 'Commits' tab for the 'microservices' repository. The commit list table displays three entries:

Commit ID	Commit message	Commit date	Authored date	Author	Committer	Actions
c7f1ee8	Completed Employee Microservice	Just now	Just now	MUHAMMAD_OWAIS_I MRAN	MUHAMMAD_OWAIS_I MRAN	Copy ID Browse
9825e30a	Modified Customer Microservice	26 minutes ago	26 minutes ago	MUHAMMAD_OWAIS_I MRAN	MUHAMMAD_OWAIS_I MRAN	Copy ID Browse
0a46e52d	two unmodified copies of the application code	5 hours ago	5 hours ago	EC2 Default User	EC2 Default User	Copy ID Browse

Significance & Importance of this Task

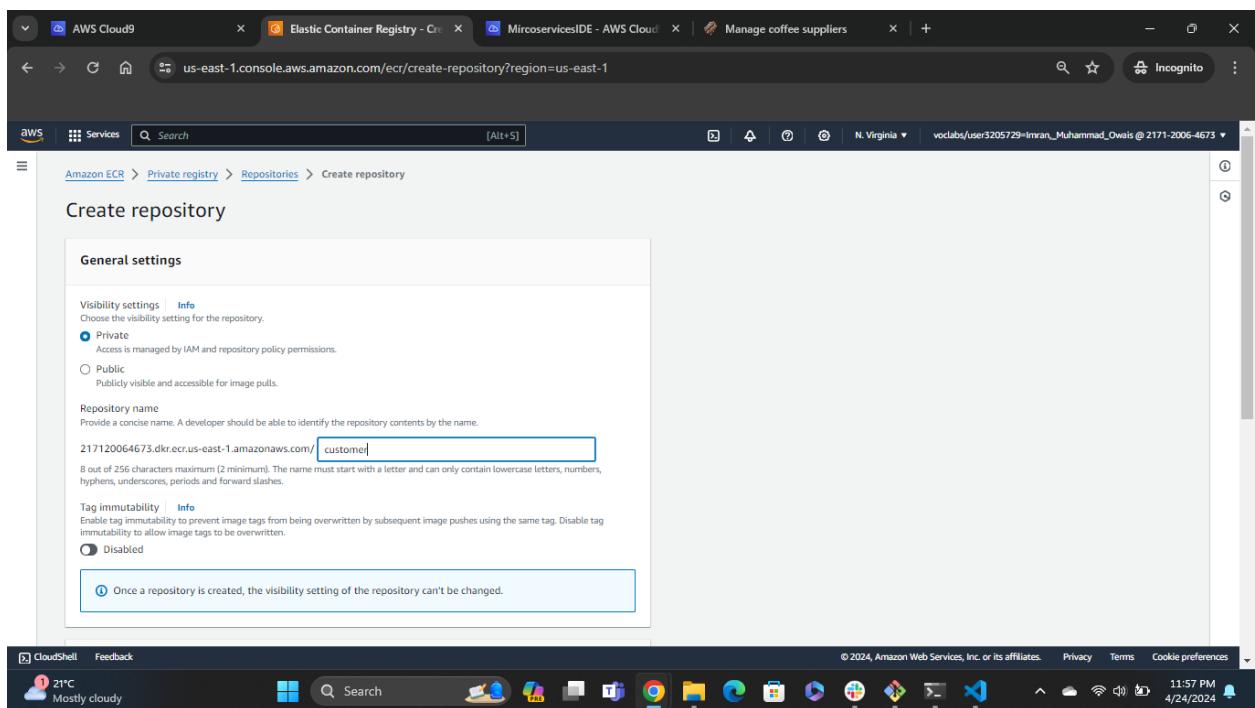
This task helps us in learning:

- How to use the EC2 instance as a source for running docker containers and use various port of the EC2 instance for hosting different parts of application traffic.
- How Security groups can help in controlling traffic access to/from EC2 instance.
- Basic strategy of how microservice should work and communicate with other microservices.

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

Task 5.1: Create ECR repositories and upload the Docker images

1. Create ECR Repositories for storing docker image of employee and customer microservices, screenshot of creating ECR for customer attached, same procedure followed for employee as well



2. Verifying if the ECR are created.

The screenshot shows the AWS Cloud9 IDE interface. In the top navigation bar, there are tabs for AWS Cloud9, Elastic Container Registry, MircoservicesIDE - AWS Cloud, and Manage coffee suppliers. The main content area displays a success message: "Created private repository employee has been successfully created in private registry". Below this, the "Private repositories" section is shown, featuring an "Improved basic scanning" notification with a "Switch" button. A table lists two repositories: "customer" and "employee".

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type
customer	217120064673.dkr.ecr.us-east-1.amazonaws.com/customer	April 24, 2024, 23:57:15 (UTC-04)	Disabled	Manual	AES-256
employee	217120064673.dkr.ecr.us-east-1.amazonaws.com/employee	April 24, 2024, 23:57:29 (UTC-04)	Disabled	Manual	AES-256

3. Updating Push policy for ECR repositories so that CodePipeline can push new build onto ECR.

The image consists of three vertically stacked screenshots of the AWS Cloud9 IDE interface. Each screenshot shows the 'Amazon Elastic Container Registry' service in the AWS Cloud9 sidebar. The main content area displays the 'Permissions' tab for two different repositories: 'customer' and 'employee'. Both screens show a single policy statement with the following details:

- Statement 1:**
- Effect:** Allow
- Principal:** *
- Actions:** ecr:*
- Service principals:** -
- AWS Account IDs:** -

The top and middle screenshots are identical, showing the 'customer' repository. The bottom screenshot shows the 'employee' repository. The AWS Cloud9 interface includes a toolbar at the top with various icons, a search bar, and a status bar at the bottom indicating the date and time.

4. Tag the images created for customer (Task4.3) and employee (Task4.6), and push them onto relevant ECR repositories, and verify if image is pushed onto ECR with latest tag.

The screenshot displays a browser window with the AWS Cloud9 IDE interface. The top navigation bar includes tabs for 'AWS Cloud9', 'Elastic Container Registry', and 'MircoservicesIDE - AWS Cloud9'. Below the tabs, the address bar shows the URL: `us-east-1.console.aws.amazon.com/cloud9/ide/85d1a49f8c4443b4a6bd9a89fa56f7fa?region=us-east-1`.

The main content area features a code editor with two tabs: 'Dockerfile' and 'index.js'. The 'index.js' tab contains the following code:

```
37 app.post('/admin/supplier-remove/:id', supplier.remove);
38 // ...
39 app.use(function (req, res, next) {
40   res.status(404).render('404', {});
41 })
42
43
44 // set port, listen for requests
45 const app_port = process.env.APP_PORT || 8080
46 app.listen(app_port, () => {
47   console.log(`Server is running on port: ${app_port}.`);
48 });

bash - *p-10-16-10-233.e x Immediate x ⓘ
```

The terminal window below the code editor shows the following commands and output:

```
veclabs:~/environment/microservices (dev)$ account_id=$(aws sts get-caller-identity |grep Account|cut -d '-' -f4)
veclabs:~/environment/microservices (dev)$ echo $account_id
217120084673
veclabs:~/environment/microservices (dev)$ 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
veclabs:~/environment/microservices (dev)$ docker tag customer:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
veclabs:~/environment/microservices (dev)$ docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
veclabs:~/environment/microservices (dev)$
```

After performing the login and tagging steps, the user runs the command `docker image ls` to list the available Docker images:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
217120084673.dkr.ecr.us-east-1.amazonaws.com/employee	latest	eb84d4087061	12 minutes ago	82.7MB
employee	latest	eb84d4087061	12 minutes ago	82.7MB
<none>	<none>	65f638599fa9	19 minutes ago	82.7MB
customer	latest	26fb8a1e9743	6 hours ago	82.7MB
217120084673.dkr.ecr.us-east-1.amazonaws.com/customer	latest	26fb8a1e9743	6 hours ago	82.7MB
node	11-alpine	f18da2158c3d	4 years ago	75.9MB

The terminal concludes with the prompt `veclabs:~/environment/microservices (dev)$`.

The screenshot shows a browser window with multiple tabs open, illustrating the deployment process of a microservices application.

AWS Cloud9 IDE Tab:

- The main pane displays the `Dockerfile` and `index.js` files. The `Dockerfile` includes commands for building the Docker image and pushing it to ECR. The `index.js` file contains logic for handling supplier removal requests.
- The terminal pane shows the command history for pushing the Docker image to the ECR repository. It lists several pushes for different layers of the image, each with its corresponding digest and size.

ECR Repository Tab:

- The tab URL is `us-east-1.console.aws.amazon.com/ecr/repositories/private/217120064673/customer?region=us-east-1`.
- The page displays the `customer` repository details, specifically the `Images (1)` section.
- A single image entry is shown: `latest` (Image type, pushed on April 25, 2024, at 00:07:40 UTC-04, size 27.70 MB).

System Status Bar:

- Temperature: 21°C
- Weather: Mostly cloudy
- Date: 4/25/2024
- Time: 12:07 AM

AWS Cloud9 Elastic Container Registry - Im... MircoservicesIDE - AWS Cloud... Manage coffee suppliers

us-east-1.console.aws.amazon.com/ecr/repositories/private/217120064673/employee?region=us-east-1

Incognito

AWS Services Search [Alt+S]

N. Virginia v vocabs/user3205729=Imran_Muhammad_Owais @ 2171-2006-4673

Amazon Elastic Container Registry

Private registry

- Repositories
- Summary
- Images**
- Permissions
- Lifecycle Policy
- Repository tags
- Settings

Public registry

- Repositories
- Settings

ECR public gallery

- Amazon ECS
- Amazon EKS

Getting started

Documentation

CloudShell Feedback

Search

Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.

Amazon ECR > Private registry > Repositories > employee

employee

View push commands Edit

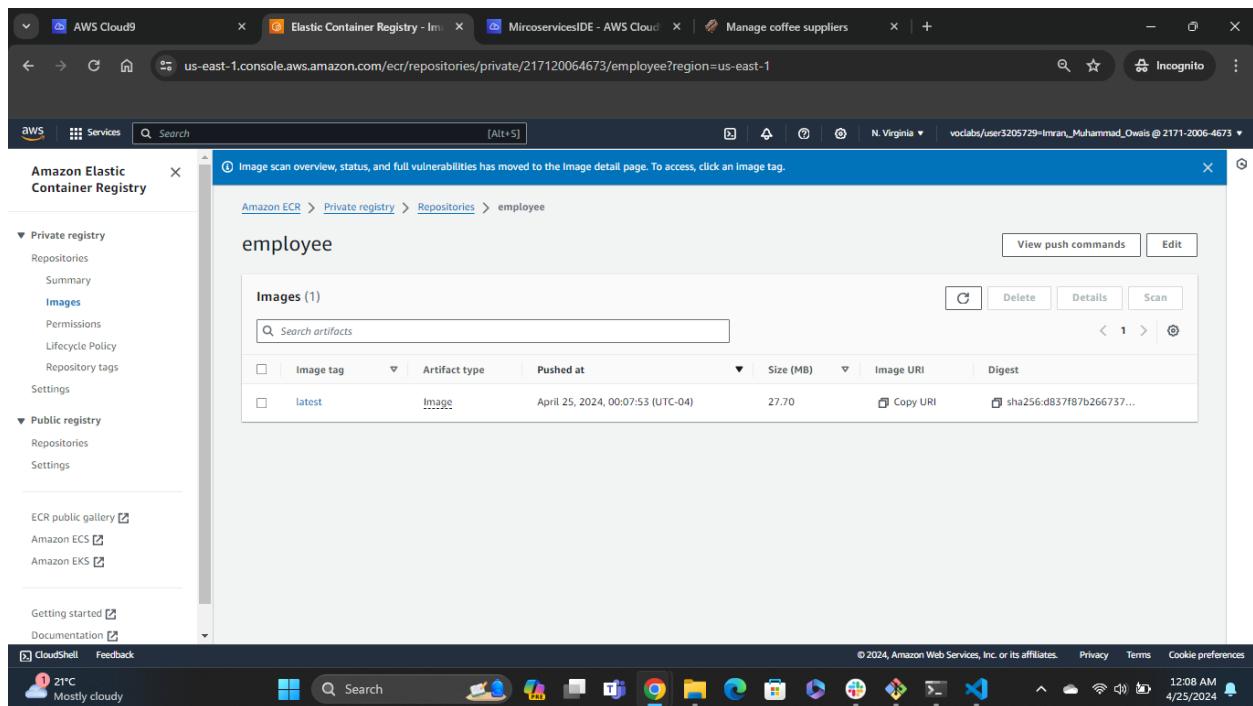
Images (1)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	April 25, 2024, 00:07:53 (UTC-04)	27.70	Copy URI	sha256:d837f87b266737...

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12:08 AM 4/25/2024

21°C Mostly cloudy



Task 5.2: Create an ECS cluster

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

The screenshot shows the 'Create cluster' wizard for the Amazon Elastic Container Service (ECS). The left sidebar lists 'Clusters', 'Namespaces', 'Task definitions', and 'Account settings'. The main area is titled 'Create cluster' and contains two sections: 'Cluster configuration' and 'Infrastructure'. In the 'Cluster configuration' section, the 'Cluster name' is set to 'microservices-serverlesscluster' and the 'Default namespace' is also set to 'microservices-serverlesscluster'. In the 'Infrastructure' section, the 'AWS Fargate (serverless)' checkbox is checked, and the 'Amazon EC2 instances' checkbox is unchecked. The status bar at the bottom indicates it's a serverless cluster.

Task 5.3: Create a CodeCommit repository to store deployment files

1. Create a code commit repository by the name of deployment

The screenshot shows the 'Create repository' page in the AWS CodeCommit console. The 'Repository name' field contains 'deployment'. The 'Description - optional' field contains 'CodeCommit repository to store deployment files for Muhammad Owais Imran Project'. The 'Tags' section has a 'Add tag' button. The 'Additional configuration' section includes an 'AWS KMS key' dropdown set to 'None'. The URL in the browser is us-east-1.console.aws.amazon.com/codesuite/codecommit/repository/create?region=us-east-1.

2. Verifying if the repository is created.

The screenshot shows the 'Repositories' page in the AWS CodeCommit console. The 'deployment' repository is listed with the following details:

Name	Description	Last modified	Clone URL	AWS KMS Key
deployment	CodeCommit repository to store deployment files for Muhammad Owais Imran Project	Just now	HTTPS SSH HTTPS (GRC)	arn:aws:kms:us-east-1:217120064673:key/14dbd07b-623c-43e8-9907-57b99033f22d
microservices	CodeCommit repository for Project by Muhammad Owais Imran	16 hours ago	HTTPS SSH HTTPS (GRC)	arn:aws:kms:us-east-1:217120064673:key/14dbd07b-623c-43e8-9907-57b99033f22d

The URL in the browser is us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories?region=us-east-1.

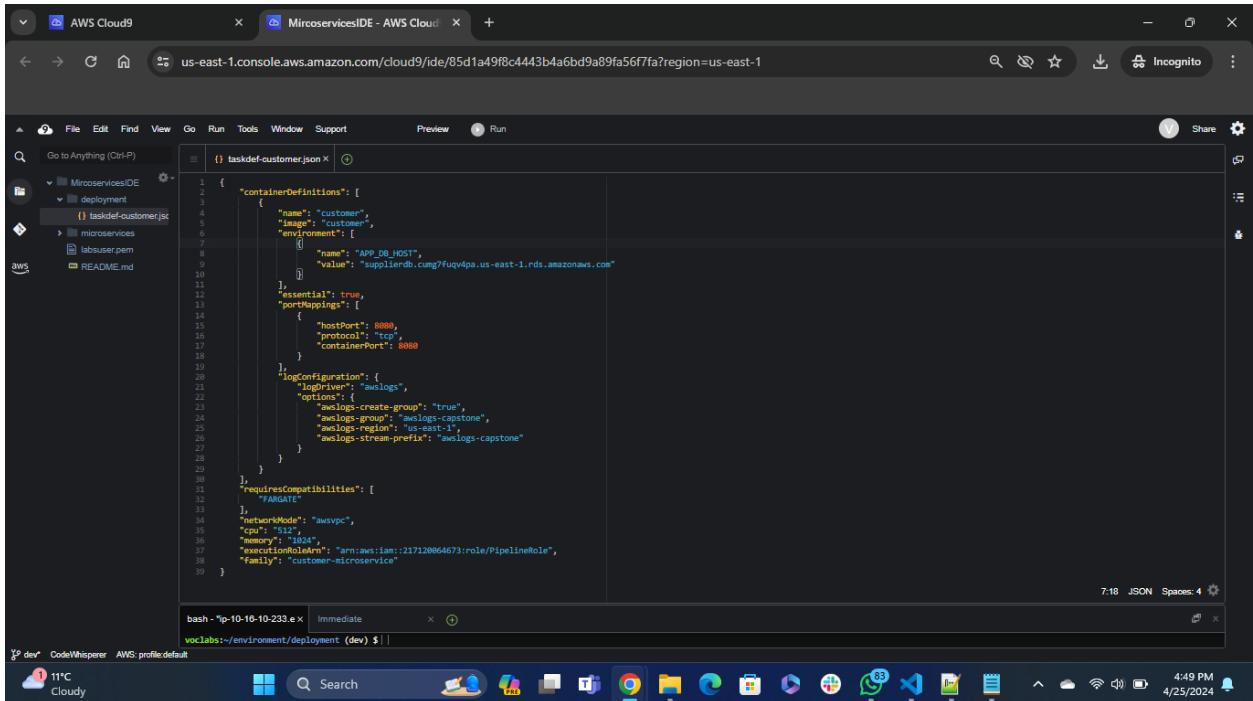
3. Initialize an empty git repository and creating a branch named dev.

The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays a file tree for a project named 'MicoservicesIDE' containing files like Dockerfile, index.js, deployment, microservices, libuser.pem, and README.md. The main workspace has two tabs open: 'Dockerfile' and 'JS index.js'. The 'index.js' tab contains code for a Node.js application, including a POST endpoint for removing a supplier and a port configuration. Below the code editor is a terminal window titled 'bash -> p-10-16-10-233.e' showing the command to initialize a git repository and create a new branch named 'dev'. The terminal output includes hints about using 'master' instead of 'main' and renaming the branch. The bottom of the screen shows the Windows taskbar with various pinned icons and the date/time (4/25/2024, 4:47 PM).

```
37 app.post("/admin/supplier-remove/:id", supplier.remove);
38 // handle 404
39 app.use(function (req, res, next) {
40   res.status(404).render("404", {});
41 })
42
43 // set port, listen for requests
44 const app_port = process.env.APP_PORT || 8080
45 app.listen(app_port, () => {
46   console.log(`Server is running on port ${app_port}.`);
47 });
vocabs:/environment $ mkdir deployment
vocabs:/environment $ cd deployment/
vocabs:/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: 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/
vocabs:/environment/deployment (master) $ git branch -m dev
vocabs:/environment/deployment (dev) $
```

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

1. In the AWS Cloud9 IDE, create a new directory by the name of deployment on ~environment
2. Create a file taskdef-customer.json



```
1  {
2     "containerDefinitions": [
3         {
4             "name": "customer",
5             "image": "customer",
6             "environment": [
7                 {
8                     "name": "APP_DB_HOST",
9                     "value": "supplierdb.cugf4pa.us-east-1.rds.amazonaws.com"
10                }
11            ],
12            "essential": true,
13            "portMappings": [
14                {
15                    "hostPort": 8080,
16                    "protocol": "tcp",
17                    "containerPort": 8080
18                }
19            ],
20            "logConfiguration": {
21                "logDriver": "awslogs",
22                "options": {
23                    "awslogs-create-group": "true",
24                    "awslogs-group": "awslogs-capstone",
25                    "awslogs-region": "us-east-1",
26                    "awslogs-stream-prefix": "awslogs-capstone"
27                }
28            }
29        },
30        "requiresCompatibilities": [
31            "#AWS_ECS"
32        ],
33        "networkMode": "awsvpc",
34        "cpu": "512",
35        "memory": "128",
36        "executionRoleArn": "arn:aws:iam::217120064673:role/PipelineRole",
37        "family": "customer-microservice"
38    }
39 }
```

3. Register this service onto ECR, and verify.

The screenshot shows two browser windows. The top window is 'Cluster tasks | Elastic Container Service - AWS Cloud9' with the URL 'us-east-1.console.aws.amazon.com/cloud9/ide/85d1a49f8c4443b4a6bd9a89fa56f7fa?region=us-east-1'. It displays the contents of 'taskdef-customer.json' in a code editor. The bottom window is 'Task definition json | Elastic Container Service - AWS Cloud9' with the URL 'us-east-1.console.aws.amazon.com/ecs/v2/task-definitions/customer-microservice/1/json?region=us-east-1'. It shows the 'customer-microservice:1' task definition in the AWS console, including its ARN, status, and configuration details.

```

{
  "cpu": "1G",
  "memory": "1024",
  "executionRoleArn": "arn:aws:iam::217120064673:role/PipelineRole",
  "family": "customer-microservice"
}

{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
},
{
  "name": "ecs.capability.task-eni"
},
{
  "name": "com.amazonaws.ecs.capability.docker-remote-api.1.29"
},
"placementConstraints": [],
"compatibilities": [
  "FARGATE"
],
"requiresCompatibilities": [
  "FARGATE"
],
"cpu": "1G",
"memory": "1024",
"registeredAt": "2024-04-25T20:58:03.853000+00:00",
"registeredBy": "arn:aws:sts::217120064673:assumed-role/voclabs/user3205729-Imran_Muhammad_Owais"
}

```

4. Create a file for taskdef-employee.json and copy the contents of taskdef-customer. After you paste in the code, change the three occurrences of customer to employee, and register the service onto ECS. Copy and replace is done using the following commands:
 - **cat taskdef-customer.json > taskdef-employee.json**
 - **sed -i “s/customer/employee/g” ./taskdef-employee.json**

```

1 {
2     "containerDefinitions": [
3         {
4             "name": "employee",
5             "image": "employee",
6             "environment": [
7                 {
8                     "name": "APP_DB_HOST",
9                     "value": "supplierdb.cmg7fuqv4pa.us-east-1.rds.amazonaws.com"
10                }
11            ],
12            "essential": true,
13            "portMappings": [
14                {
15                    "hostPort": 8080,
16                    "protocol": "tcp",
17                    "containerPort": 8080
18                }
19            ],
20            "logConfiguration": {
21                "logDriver": "awslogs",
22                "options": {
23                    "awslogs-create-group": "true",
24                    "awslogs-group": "awslogs-capstone",
25                    "awslogs-region": "us-east-1",
26                    "awslogs-stream-prefix": "awslogs-capstone"
27                }
28            }
29        ],
30        "requiresCompatibilities": [
31            "FARGATE"
32        ]
33    ]
34 }

```

bash -p 10-16-10-233.x Immediate

```

vocabs:/environment/deployment (dev) $ ls
vocabs:/environment/deployment (dev) $ cat taskdef-customer.json > taskdef-employee.json
vocabs:/environment/deployment (dev) $ sed -i "s/customer/employee/g" ./taskdef-employee.json
vocabs:/environment/deployment (dev) $ aws ecs register-task-definition --cli-input-json "file:///home/ec2-user/environment/deployment/taskdef-employee.json"

```

Verify if the service was registered successfully onto ECR.

ARN	Status	Time created	App environment
arn:aws:ecs:us-east-1:217120064673:task-definition/employee-microservice:1	ACTIVE	2024-04-25T20:54:05.342Z	FARGATE

JSON

```

1 {
2     "taskDefinitionArn": "arn:aws:ecs:us-east-1:217120064673:task-definition/employee-microservice:1",
3     "containerDefinitions": [
4         {
5             "name": "employee",
6             ...
7         }
8     ],
9     ...
10 }

```

Task 5.5: Create AppSpec files for CodeDeploy for each microservice

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

The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes 'File', 'Edit', 'Find', 'Go', 'Run', 'Tools', 'Window', 'Support', 'Preview', and 'Run'. The left sidebar shows a file tree with 'MircoservicesIDE', 'deployment', 'appspec-customer', 'taskdef-customer.js', 'taskdef-employee.js', 'microservices', 'labuser.pem', and 'README.md'. The main editor window displays the 'appspec-customer.yaml' file:

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:
          ContainerName: "customer"
          ContainerPort: 8080
```

Below the editor, a terminal window titled 'aws - *ip-10-10-10-233.exe' shows the command: 'aws - *ip-10-10-10-233.exe \$ touch appspec-customer.yaml'. The bottom status bar indicates 'dev' profile, 'CodeWhisperer', 'AWS profile default', and the date/time '4/25/2024 4:56 PM'.

2. Create an AppSpec file for the employee microservice
 - a. Name the file appspec-employee.yaml
 - b. The contents of the file should be the same as the appspec-customer.yaml file. However, change customer` ` on the containerNameline to be employee

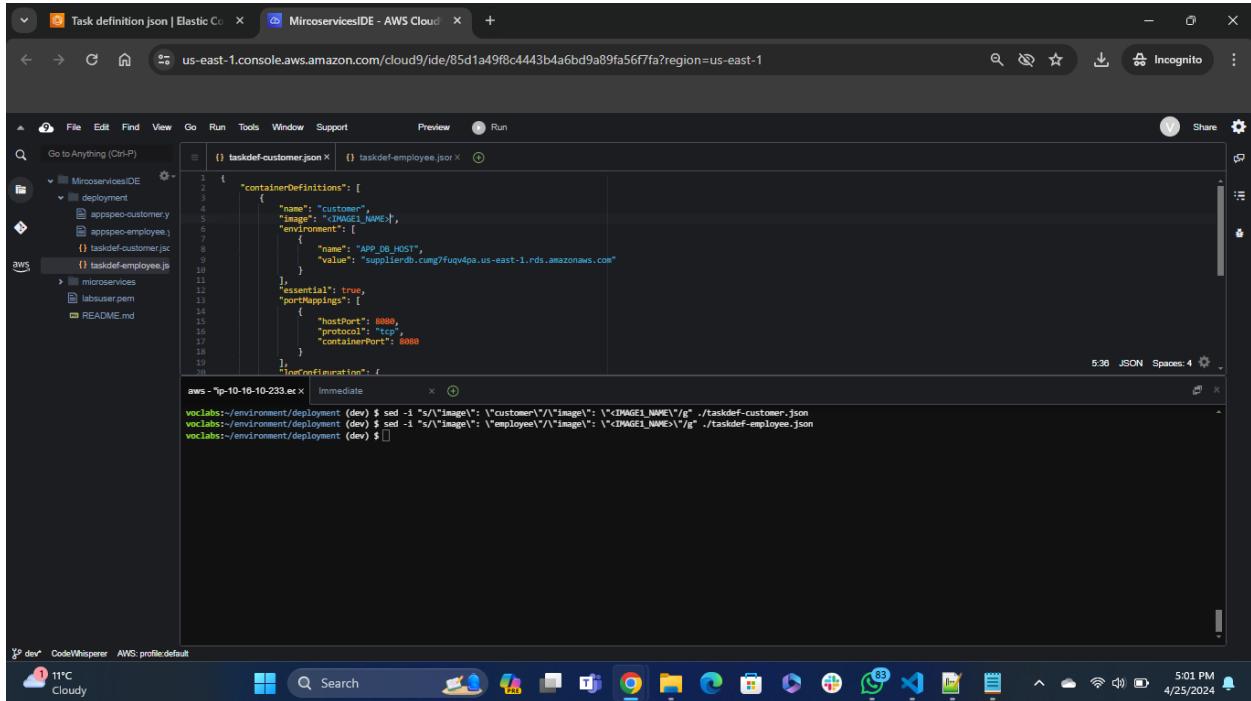
The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes tabs for 'Task definition json | Elastic Co' and 'MircoservicesIDE - AWS Cloud'. The address bar displays the URL: 'us-east-1.console.aws.amazon.com/cloud9/ide/85d1a49f8c4443b4a6bd9a89fa56f7fa?region=us-east-1'. The main workspace contains two tabs: 'appspec-employee.yaml' and 'aws - "ip-10-16-10-233.ex"'. The 'appspec-employee.yaml' tab shows the following YAML code:

```
version: 0.6
Resources:
  TargetService:
    Type: AWS::ECS::Service
    Properties:
      TaskDefinition: !@TASK_DEFINITION!
      LoadBalancerInfo:
        ContainerName: "employee"
        ContainerPort: 8080
```

The 'aws - "ip-10-16-10-233.ex"' tab shows the command: 'aws - "ip-10-16-10-233.ex" \$ cat appspec-customer.yaml > appspec-employee.yaml && sed -i "s/customer/employee/g" ./appspec-employee.yaml'. The bottom of the screen shows the Windows taskbar with various pinned icons and the system tray.

Task 5.6: Update files and check them into CodeCommit

Edit the file taskdef-customer.json and taskdef-employee.json and replace image field value with “IMAGE1_NAME”. This value will be updated automatically by the codepipeline which will be created in the later steps.



The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a sidebar with project navigation. In the center, two JSON files are open: "taskdef-customer.json" and "taskdef-employee.json". The "taskdef-customer.json" file contains the following code:

```
1 {
2     "containerDefinitions": [
3         {
4             "name": "customer",
5             "image": "<IMAGE1_NAME>",
6             "environment": [
7                 {
8                     "name": "APP_DB_HOST",
9                     "value": "supplierdb.cug7fuqv4ps.us-east-1.rds.amazonaws.com"
10                }
11            ],
12            "essential": true,
13            "portMappings": [
14                {
15                    "hostPort": 8080,
16                    "protocol": "tcp",
17                    "containerPort": 8080
18                }
19            ],
20            "memory": 128
21        }
22    ],
23    "volumes": []
24}
```

The "taskdef-employee.json" file is similar but with different environment variables. Below the code editor, a terminal window shows the command used to update the files:

```
aws -ip=10-16-10-233.ec2.internal environment/deployment (dev) $ sed -i "s/\"image\": \"customer\"/\"image\": \"<IMAGE1_NAME>\"/g" ./taskdef-customer.json
aws -ip=10-16-10-233.ec2.internal environment/deployment (dev) $ sed -i "s/\"image\": \"employee\"/\"image\": \"<IMAGE1_NAME>\"/g" ./taskdef-employee.json
aws -ip=10-16-10-233.ec2.internal environment/deployment (dev) $
```

Commit the code onto code commit.

The screenshot shows a Windows desktop environment with two windows open. The top window is a web browser displaying the AWS CodeCommit 'Commits' page for the 'deployment' repository. It lists a single commit (1f274a78) with the message 'Add TaskDefinition and AppSpec files for employee & customer microservices'. The bottom window is a 'CloudShell' terminal window showing a terminal session. The user has navigated to the 'environment/deployment' directory and run several commands:

```
bash - *p-10-16-10-233.e x Immediate x +  
veclabs:~/environment/deployment (dev) $ git add .  
veclabs:~/environment/deployment (dev) $ git commit -am "Add TaskDefinition and AppSpec files for employee & customer microservices"  
[dev (root-commit) 1f274a7] Add TaskDefinition and AppSpec files for employee & customer 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  
veclabs:~/environment/deployment (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment  
veclabs:~/environment/deployment (dev) $ git push origin -u dev  
Enumerating objects: 6, done.  
Counting objects: 6 (delta 0), done.  
Delta compression using up to 2 threads  
Compressing objects: 100% (6/6), done.  
Writing objects: 100% (6/6), 1.09 KB | 1.09 MiB/s, done.  
Total 6 (delta 0), 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  
branch 'dev' set up to track 'origin/dev'.  
veclabs:~/environment/deployment (dev) $
```

Significance & Importance of this Task

This task has the following outcomes:

- Learning the use case of ECR
- How to create an ECS cluster
- Difference between Task and Service of an ECS
- How to Define AppSpec for CodeDeployment, and Task Definition for ECS
- How CodeDeployment helps in achieving CI/CD in an application Development to Deployment lifecycle

Phase 6: Creating target groups and an Application Load Balancer

Task 6.1: Create four target groups

1. For customer-microservice create target group with this config

Target Group Type: IP address

Target Group Name: customer-tg-one/customer-tg-two

Protocol: HTTP

Port: 8080

VPC: LabVPC

Health check path: /

2. For employee-microservice create target group with this config

Target Group Type: IP address

Target Group Name: employee-tg-one/employee-tg-two

Protocol: HTTP

Port: 8081

VPC: LabVPC

Health check path: /admin/suppliers

Step 1 Create target group | EC2 +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:

aws Services Search [Alt+S]

N. Virginia v vodlabs/user3205729-Imran_Muhammad_Owais @ 2171-2006-4673

EC2 > Target groups > Create target group

Step 1 Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Step 2 Register targets

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

CloudShell Feedback 11°C Cloudy Search N. Virginia v vodlabs/user3205729-Imran_Muhammad_Owais @ 2171-2006-4673 © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 5:07 PM 4/25/2024

Step 1 Create target group | EC2 +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:

aws Services Search [Alt+S]

N. Virginia v vodlabs/user3205729-Imran_Muhammad_Owais @ 2171-2006-4673

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

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-06f075abdfaf226f64
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.

CloudShell Feedback 11°C Cloudy Search N. Virginia v vodlabs/user3205729-Imran_Muhammad_Owais @ 2171-2006-4673 © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 5:07 PM 4/25/2024

Step 1 Create target group | EC2

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:

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.
/

Attributes
Certain default attributes will be applied to your target group. You can view and edit them after creating the target group.

Tags - optional
Consider adding tags to your target group. Tags enable you to categorize your AWS resources so you can more easily manage them.

Next

Step 2 Create target group | EC2

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:

Register targets
This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

IP addresses

Step 1: Choose a network
You can add IP addresses from the VPC selected for your target group or from outside the VPC. Note that you can assemble a mix of targets from multiple network sources by returning to this step and choosing another network.

Network
LabVPC
vpc-040f7cddbf2256d4
[IPv4 CIDR: 10.16.0.0/16]

Step 2: Specify IPs and define ports
You can manually enter IP addresses from the selected network.

Enter an IPv4 address from a VPC subnet.
10.16.0.
Add IPv4 address
You can add up to 4 more IP addresses.

Ports
Ports for routing to this target.
8080
1-65535 (separate multiple ports with commas)

Next

Target groups | EC2 | us-east-1

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:

EC2 > Target groups

Target groups (4) Info

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
employee-tg-two	arn:aws:elasticloadbalancing:us-east-1:217120064673:targetgroup/customer-tg-one/b310624cafa6501	8080	HTTP	IP	None associated	vpc-06f075ab6fa226f64
employee-tg-one	arn:aws:elasticloadbalancing:us-east-1:217120064673:targetgroup/customer-tg-one/b310624cafa6501	8080	HTTP	IP	None associated	vpc-06f075ab6fa226f64
customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:217120064673:targetgroup/customer-tg-one/b310624cafa6501	8080	HTTP	IP	None associated	vpc-06f075ab6fa226f64
customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:217120064673:targetgroup/customer-tg-one/b310624cafa6501	8080	HTTP	IP	None associated	vpc-06f075ab6fa226f64

0 target groups selected

Select a target group above.

Target group details | EC2 | us-east-1

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroup:targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:217120064673:targetgroup/customer-tg-one/b310624cafa6501

customer-tg-one

Details

arn:aws:elasticloadbalancing:us-east-1:217120064673:targetgroup/customer-tg-one/b310624cafa6501

Target type IP	Protocol : Port HTTP: 8080	Protocol version HTTP1	VPC vpc-06f075ab6fa226f64	
IP address type IPv4	Load balancer None associated			
0 Total targets	0 Healthy	0 Unhealthy	0 Unused	
	0 Anomalous			
Targets	Monitoring	Health checks	Attributes	Tags

Health check settings

Protocol HTTP	Path /	Port Traffic port	Healthy threshold 5 consecutive health check successes
Unhealthy threshold 2 consecutive health check failures	Timeout 5 seconds	Interval 30 seconds	Success codes 200

CloudShell Feedback 11°C Cloudy 5:12 PM 4/25/2024

Target group details | EC2 | us

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroup:targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:21712006473:targetgroup/employee-tg-two@2171-2006-4673

EC2 > Target groups > employee-tg-two

employee-tg-two

Details

arn:aws:elasticloadbalancing:us-east-1:21712006473:targetgroup/employee-tg-two@ea56e77a9ab927

Target type	Protocol : Port	Protocol version
IP	HTTP: 8080	HTTP1
IP address type	Load balancer IPv4 <small>None associated</small>	

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
0	0	0	0	0	0
	0 Anomalous				

Targets | Monitoring | **Health checks** | Attributes | Tags

Health check settings

Protocol	Path	Port	Healthy threshold
HTTP	/admin/suppliers	Traffic port	5 consecutive health check successes

Unhealthy threshold	Timeout	Interval	Success codes
2 consecutive health check failures	5 seconds	30 seconds	200

CloudShell Feedback

Cloud Shell icon | Search icon | Task Manager icon | File icon | Task View icon | Taskbar icons | Network icon | Battery icon | Weather icon | Date and time: 5:13 PM | Date: 4/25/2024

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

1. New EC2 security group microservice-sg, and allow inbound traffic on port 80 and 8080 from anywhere IPv4.

The screenshot shows the 'Create security group' wizard on the AWS Management Console. In the 'Basic details' section, the security group name is 'microservices-sg' and the description is 'Allow Access on Port80 and Port8080 for microservices Access'. The VPC is set to 'vpc-06f075ab6fa226f64 (LabVPC)'. In the 'Inbound rules' section, there are two rules: one for port 80 (Custom TCP, TCP protocol, port range 80, source Anywhere..., description 'Allow traffic from anywhere IPv4 on Port 80') and one for port 8080 (Custom TCP, TCP protocol, port range 8080, source Anywhere..., description 'Allow traffic from anywhere IPv4 on Port 8080').

2. Create a LoadBalancer microservicesLB, making it internet facing for IPv4 address, in LabVPC, PublicSubnet1, and PublicSubnet2, and use the sg created in above step. Also configure two listeners, each listener will have two rules.

The screenshot shows three vertically stacked windows from the AWS CloudShell interface, each displaying a different step in the process of creating an Application Load Balancer (ALB).

Top Window: "Create Application Load Balancer" (Info)

The "Basic configuration" section is visible. Under "Load balancer name", the value "microservicesLB" is entered. A note states: "Name must be unique within your AWS account and can't be changed after the load balancer is created." Below this, the "Scheme" dropdown is set to "Internet-facing". The "IP address type" dropdown is set to "IPv4".

Middle Window: "Mappings" (Info)

The "Mappings" section shows two subnets assigned by AWS: "Public Subnet1" (subnet-07dc729d57a3f80a2) and "Public Subnet2" (subnet-0433f0345b6e456e2). Both subnets are selected with checked checkboxes.

Bottom Window: "Security groups" (Info)

The "Security groups" section shows one security group selected: "microservices-sg" (sg-0f44f0be7186022a2). It also lists the VPC and subnet information: "VPC: vpc-06f075ab6fa226f64" and "Subnet: subnet-0433f0345b6e456e2".

Screenshot of the AWS CloudShell interface showing the creation of an Application Load Balancer (ALB). The user is navigating through the 'Listeners and routing' configuration step.

The 'Listeners and routing' section displays two listeners:

- Listener HTTP:80**: Protocol: HTTP, Port: 80, Default action: Forward to target group `customer-tg-two`. Target type: IP, IPv4. Subnets: 1-65535.
- Listener HTTP:8080**: Protocol: HTTP, Port: 8080, Default action: Forward to target group `customer-tg-one`. Target type: IP, IPv4. Subnets: 1-65535.

Below the listeners, there is a section for **Listener tags - optional** where the user can add tags to categorize the resource.

At the bottom of the page, the 'Review' section summarizes the configurations:

- Summary**: Basic configuration includes `microservicesLB` (Internet-facing, IPv4). Security groups include `microservices-sg` (VPC: `vpc-06f075ab6fa226f64`, LabVPC). Network mapping includes `us-east-1a` (subnet: `c7d6-729d57a3f80a2`, Public Subnet1) and `us-east-1b` (subnet: `d435f0345b6e456e2`, Public Subnet2). Listeners and routing shows `HTTP:80` defaulting to `customer-tg-two` and `HTTP:8080` defaulting to `customer-tg-one`.
- Service integrations**: AWS WAF: None, AWS Global Accelerator: None.
- Tags**: None.
- Attributes**: A note states that certain default attributes will be applied to the load balancer.

The CloudShell interface at the bottom shows the AWS logo, weather (11°C, Cloudy), search bar, and various application icons.

Step 5 Add listener rule | EC2 | +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:217120064... 🔍 ⌂ Incognito

EC2 Services Search [Alt+S]

N. Virginia vocabs/user3205729-imran_Muhammad_Owais @ 2171-2006-4673

EC2 > Load balancers > microservicesLB > HTTP:80 listener > Add rule

Review and create

Step 1 Add rule

Step 2 Define rule conditions

Step 3 Define rule actions

Step 4 Set rule priority

Step 5 Review and create

Listener details: HTTP:80

Protocol:Port HTTP:80	Load balancer microservicesLB	Default actions Forward to target group • customer-tg-two 1 (100%) • Group-level stickiness: Off
Listener ARN arn:aws:elasticloadbalancing:us-east-1:217120064673:listener/app/microservicesLB/d5f090c54137b5e9/58ec15db6cadb7d0		

Rule details

Priority 1	Conditions (If) If request matches all: Path Pattern is /admin/*	Actions (Then) Forward to target group • employee-tg-one 1 (100%) • Group-level stickiness: Off
Rule ARN Pending		

Rule tags (0)

CloudShell Feedback 11°C Cloudy Search 5:24 PM 4/25/2024

Step 5 Add listener rule | EC2 | +

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:217120064... 🔍 ⌂ Incognito

EC2 Services Search [Alt+S]

N. Virginia vocabs/user3205729-imran_Muhammad_Owais @ 2171-2006-4673

EC2 > Load balancers > microservicesLB > HTTP:8080 listener > Add rule

Review and create

Step 1 Add rule

Step 2 Define rule conditions

Step 3 Define rule actions

Step 4 Set rule priority

Step 5 Review and create

Listener details: HTTP:8080

Protocol:Port HTTP:8080	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-1:217120064673:listener/app/microservicesLB/d5f090c54137b5e9/6811e64242b76572		

Rule details

Priority 1	Conditions (If) If request matches all: Path Pattern is /admin/*	Actions (Then) Forward to target group • employee-tg-one 1 (100%) • Group-level stickiness: Off
Rule ARN Pending		

Rule tags (0)

CloudShell Feedback 11°C Cloudy Search 5:25 PM 4/25/2024

Load balancer details | EC2 | us-east-1

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:us-east-1:217120064673:loadbalancer/app/microservicesLB/d5f09bc54137b5e9

N. Virginia

voclabs/user3205729-imran_Muhammad_Owais @ 2171-2006-4673

Scheme: Internet-facing Hosted zone: Z35SXDOTRQ7X7K Availability Zones: subnet-0433f0345b6e456e2 us-east-1b (use1-az2) subnet-07dc729d57a3fb0a2 us-east-1a (use1-az1) Date created: April 25, 2024, 17:21 (UTC-04:00)

Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:217120064673:loadbalancer/app/microservicesLB/d5f09bc54137b5e9 DNS name: microservicesLB-581467332.us-east-1.elb.amazonaws.com (A Record)

Listeners and rules | Network mapping | Resource map - new | Security | Monitoring | Integrations | Attributes | Tags

Listeners and rules (2) info

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.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS
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	2 rules	ARN	Not applicable	Not applicable	Not applicable

CloudShell Feedback 11°C Cloudy Search 5:26 PM 4/25/2024 © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Significance & Importance of this Task

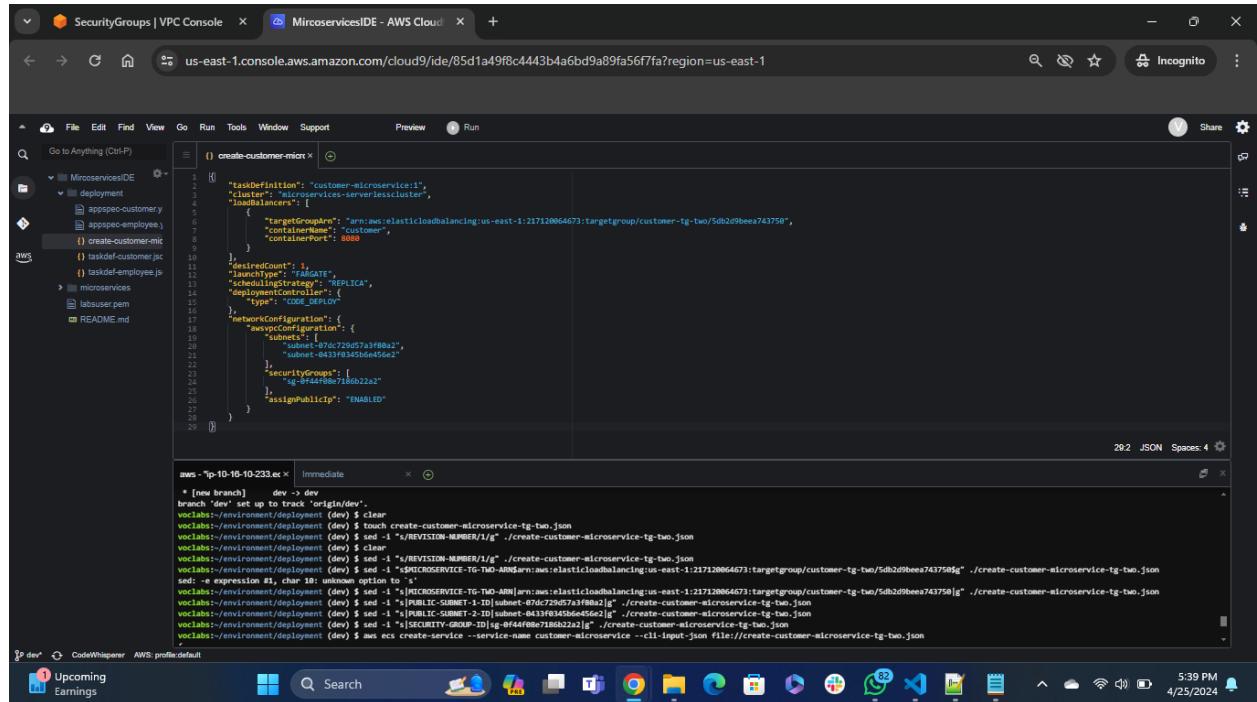
This task helps us in implementing:

- How target groups work and the way they can be used to route traffic between production and test environments based on pre-defined host port and application logic
- The reason of health check
- How load balancer control and decides that on which request, the traffic should be sent to which application host port.

Phase 7: Creating two Amazon ECS services

Task 7.1: Create the ECS service for the *customer* microservice

1. In the deployment directory in Cloud9, create a new file named create-customer-microservice-tg-two.json



The screenshot shows a Cloud9 IDE interface with the following details:

- File Explorer:** Shows a folder structure for "MicoservicesIDE" containing "deployment", "appspec-customer", "appspec-employee", "create-customer-mic", "taskdef-customer.json", "taskdef-employee.json", "micoservices", "labusec.pem", and "README.md".
- Code Editor:** An open JSON file named "create-customer-mic" with the following content:

```
1 {  
2     "taskDefinition": "customer-microservice:1",  
3     "cluster": "micoservices-serverlesscluster",  
4     "loadBalancers": [  
5         {  
6             "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:217120864673:targetgroup/customer-tg-two/5db2d9beea743754",  
7             "containerPort": 8080  
8         }  
9     ],  
10    "desiredCount": 1,  
11    "maximumCapacity": 1,  
12    "schedulingStrategy": "REPLICAS",  
13    "deploymentController": {  
14        "type": "CODE_DEPLOY"  
15    },  
16    "networkConfiguration": {  
17        "awsvpcConfiguration": {  
18            "subnets": [  
19                "subnet-0f7290c73980a2",  
20                "subnet-043193456edc556e2",  
21            ],  
22            "securityGroups": [  
23                "sg-0f4248fb718b22a2",  
24            ],  
25            "assignPublicIp": "ENABLED"  
26        }  
27    }  
28 }  
29 }
```

- Terminal:** An open terminal window titled "aws - [ip-10-10-10-233.eu]" showing AWS CLI commands for creating a service:

```
* [new branch]  dev -> dev  
branch 'dev' set up to track 'origin/dev'.  
vocabs:-environment/deployment $ clear  
vocabs:-environment/deployment $ touch create-customer-microservice-tg-two.json  
vocabs:-environment/deployment $ sed -i "/REVISION-NUMBER/1/g" ./create-customer-microservice-tg-two.json  
vocabs:-environment/deployment (dev) $ clear  
vocabs:-environment/deployment (dev) $ sed -i "/REVISION-NUMBER/1/g" ./create-customer-microservice-tg-two.json  
vocabs:-environment/deployment (dev) $ sed -i "/SERVICE-NAME/1/g" ./create-customer-microservice-tg-two.json  
sed: can't open file #1, check 10: invalid option to 'sed'  
vocabs:-environment/deployment (dev) $ sed -i "/[MICROSERVICE-TG-TWO-ARN]/r aws:elasticloadbalancing:us-east-1:217120864673:targetgroup/customer-tg-two/5db2d9beea7437585g" ./create-customer-microservice-tg-two.json  
vocabs:-environment/deployment (dev) $ sed -i "/[PUBLIC-SUBNET-1-ID]/s/substr-0fd2729d73a3f88a/g" ./create-customer-microservice-tg-two.json  
vocabs:-environment/deployment (dev) $ sed -i "/[PUBLIC-SUBNET-2-ID]/s/substr-043193456edc556e2/g" ./create-customer-microservice-tg-two.json  
vocabs:-environment/deployment (dev) $ sed -i "/[SECURITY-GROUP-ID]/s/ig-0f4460e071ab02a2a1/g" ./create-customer-microservice-tg-two.json  
vocabs:-environment/deployment (dev) $ max ec2 create-service --service-name customer-microservice -cli-input-json file://./create-customer-microservice-tg-two.json
```

- Bottom Status Bar:** Shows "P dev", "CodeWhisperer", "AWS: profile default", "Upcoming Earnings", and system icons for date, time, and notifications.

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

1. In the deployment directory in Cloud9, create a new file named create-employee-microservice-tg-two.json

The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes tabs for 'Target groups | EC2 | us-east-1', 'MicroservicesIDE - AWS Cloud', and 'Cluster services | Elastic Conta...'. The main workspace contains two code editors. The left editor shows a JSON file named 'create-employee-microservice-tg-two.json' with the following content:

```
1 {
  "taskDefinition": "employee-microservice:1",
  "clusterArn": "arn:aws:lambda:us-east-1:217120864673:cluster/microservices-serverlesscluster",
  "loadBalancers": [
    {
      "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:217120864673:targetgroup/employee-tg-two/e55e777a9ab8927",
      "containerName": "employee",
      "containerPort": 8080
    }
  ],
  "desiredCount": 1,
  "launchType": "TAGGED",
  "schedulingStrategy": "REPLICAS",
  "deployConfiguration": {
    "type": "CODE_DEPLOY"
  },
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "subnet-07c72057a3f88a2",
        "subnet-0433f0345b0e556e2"
      ],
      "securityGroups": [
        "sg-0444fb0871b021a2"
      ],
      "assignPublicIp": "ENABLED"
    }
  }
}
```

The right editor shows a terminal window with the command: `wsc-labs~/environment/deployment (dev) $ aws ecs create-service --service-name employee-microservice --cli-input-json file:///create-employee-microservice-tg-two.json`. The terminal output shows the service creation process:

```
{
  "service": {
    "serviceName": "arn:aws:ecs:us-east-1:217120864673:service/microservices-serverlesscluster/employee-microservice",
    "serviceArn": "arn:aws:lambda:us-east-1:217120864673:service/employee-microservice",
    "clusterArn": "arn:aws:lambda:us-east-1:217120864673:cluster/microservices-serverlesscluster",
    "loadBalancers": [
      {
        "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:217120864673:targetgroup/employee-tg-two/e55e777a9ab8927",
        "containerName": "employee",
        "containerPort": 8080
      }
    ],
    "serviceRegistries": []
  }
}
```

Verifying task in AWS ECS

The screenshot shows the AWS Cloud Console interface for managing an Amazon Elastic Container Service (ECS) cluster named "microservices-serverlesscluster".

Cluster overview:

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:217120064673:cluster/microservices-serverlesscluster	Active	Default	-

Services:

Draining	Active	Pending	Running
-	2	0	-

Services List:

Service name	ARN	Status	Service type	Deployments and tasks	Last deploy...	Task definit...
employee-microservice	arn:aws:sec...	Active	REPLICA	0/1 Tasks running	-	employee-mic...
customer-microservice	arn:aws:sec...	Active	REPLICA	0/1 Tasks running	-	customer-mic...

Significance & Importance of this Task

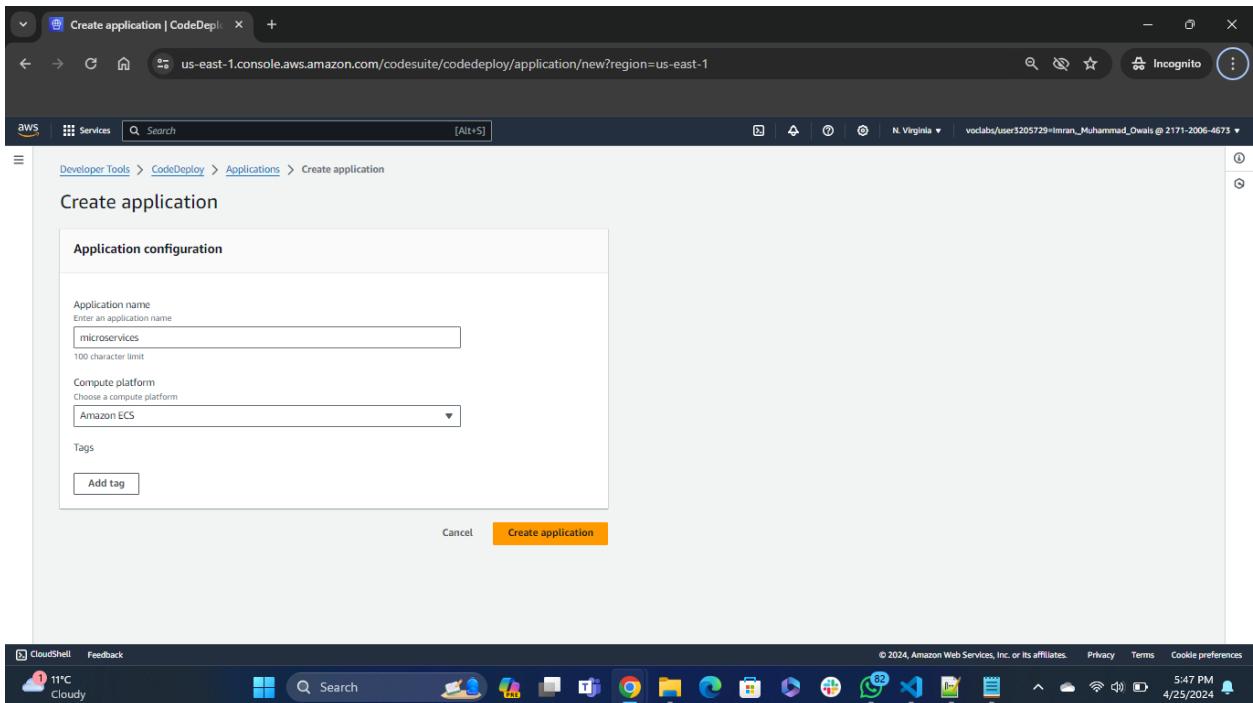
This task helps us in implementing:

- How Create Services on AWS ECS to deploy different microservices by pulling their images from an AWS ECR repository.

Phase 8: Configuring CodeDeploy and CodePipeline

Task 8.1: Create a CodeDeploy application and deployment groups

1. Create AWS CodeDeploy application by the name of microservices.



2. Create a Deployment Group for the microservices-customer with the below config

Deployment Group name: **microservices-customer**

Service Role: ARN for DeployRole

Environment Configuration:

ECS Cluster Name: **microservices-serverlesscluster**

ECS Service Name: **customer-microservice**

Load Balancer Section:

Load balancer name: **microservicesLB**

Production Listener Port: **HTTP: 80**

Test Listener Port: **HTTP: 8080**

Target group 1 name: **customer-tg-two**

Target group 2 name: **customer-tg-one**

Deployment Settings:

Traffic Rerouting: **Reroute traffic immediately**

Deployment Configuration: **CodeDeployDefault.ECSAllAtOnce**

Original Revision termination: Days: 0, Hours: 0, Minutes: 5

microservices | CodeDeploy | +

us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices?region=us-east-1

Application created
In order to create a new deployment, you must first create a deployment group.

Microservices

Application details

Name: microservices Compute platform: Amazon ECS

Deployments Deployment groups Revisions

Deployment groups

No deployment groups
Before you can deploy your application using CodeDeploy, you must create a deployment group.

Create deployment group

CloudShell Feedback

11°C Cloudy

Search

aws Services Search [Alt+S]

CloudShell Feedback

11°C Cloudy

Search

CloudShell Feedback

11°C Cloudy

Search

N. Virginia vocabs/user3205729-imran_Muhammad_Owais@2171-2006-4673

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 5:47 PM 4/25/2024

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 5:48 PM 4/25/2024

Create deployment group | CodeDeploy | +

us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1

aws Services Search [Alt+S]

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. armawsiam:217120064673:role/DeployRole

Environment configuration

Choose an ECS cluster name: microservices-serverlesscluster

Choose an ECS service name: customer-microservice

Load balancers

Create deployment group | Co

us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1

Incognito

aws Services Search [Alt+S]

Developer Tools CodeDeploy

Source • CodeCommit Artifacts • CodeArtifact Build • CodeBuild Deploy • CodeDeploy

Getting started Deployments Applications Application Settings Deployment configurations On-premises instances Pipeline • CodePipeline Settings

Q Go to resource Feedback

CloudShell Feedback

CloudShell 11°C Cloudy

Create deployment group | Co

us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups/new?region=us-east-1

Incognito

aws Services Search [Alt+S]

Developer Tools CodeDeploy

Source • CodeCommit Artifacts • CodeArtifact Build • CodeBuild Deploy • CodeDeploy

Getting started Deployments Applications Application Settings Deployment configurations On-premises instances Pipeline • CodePipeline Settings

Q Go to resource Feedback

CloudShell Feedback

CloudShell 11°C Cloudy

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 routes 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 routes 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.ECSALIASatOnce 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: 0 Hours: 0 Minutes: 5

Advanced... optional

CloudShell Feedback

CloudShell 11°C Cloudy

The screenshot shows the AWS CodeDeploy console with the URL us-east-1.console.aws.amazon.com/codesuite/codedeploy/applications/microservices/deployment-groups?region=us-east-1. The left sidebar is collapsed, showing the navigation path: Developer Tools > CodeDeploy > Applications > microservices. The main content area displays the 'Application details' for the 'microservices' application, which has a name of 'microservices' and a compute platform of 'Amazon ECS'. Below this, the 'Deployment groups' tab is selected, showing a table with two entries:

Name	Status	Last attempted deployment	Last successful deployment	Trigger count
microservices-employee	-	-	-	0
microservices-customer	-	-	-	0

At the bottom of the page, there is a toolbar with icons for CloudShell, Feedback, and various system status indicators like weather and system time.

3. Create a Deployment Group for the microservices-employee using the above step and change
 - Deployment group name to microservice-employee
 - ECS Service Name to employee-microservice
 - Target group 1 Name to employee-tg-two
 - Target group 2 Name to employee-tg-one

Task 8.2: Create a pipeline for the *customer* microservice

1. Create a Pipeline for Customer Microservice using the name update-customer-microservice.
2. Edit the *update-customer-microservice* pipeline to add another *source*.
 - a. In the **Edit: Source** section, choose **Edit stage**, then add an action with these details:
 - i. **Action name:** Image
 - ii. **Action provider:** Amazon ECR
 - iii. **Repository name:** customer
 - iv. **Image tag:** latest
 - v. **Output artifacts:** image-customer
3. Edit the *deploy* action of the *update-customer-microservice* pipeline.
 - a. Edit the **update-customer-microservice** pipeline
 - i. In the **Edit: Deploy** section, choose **Edit stage**, then add an input artifact as described below:
 1. On the **Deploy Amazon ECS (Blue/Green)** card, choose the edit (pencil) icon.
 2. Under **Input artifacts**, choose **Add** and then choose **image-customer**.

Note: You should now have *SourceArtifact* and *image-customer* as listed input artifacts.

- ii. Under **Dynamically update task definition image**, for **Input artifact with image details**, choose **image-customer**.
- iii. For **Placeholder text in the task definition**, enter IMAGE1_NAME

Create new pipeline | CodePipeline

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1

Step 1 of 5

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
 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

CloudShell Feedback

11°C Cloudy Search

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

5:53 PM 4/25/2024

Create new pipeline | CodePipeline

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1

Step 2

Add source stage

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

Repository name
Choose a repository that you have already created where you have pushed your source code.

Branch name
Choose a branch of the repository

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.

CloudShell Feedback

11°C Cloudy Search

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

5:55 PM 4/25/2024

Create new pipeline | CodePipeline

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1

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: microservices

AWS CodeDeploy deployment group: microservices-customer

Amazon ECS task definition: SourceArtifact: taskdef-customer.json

AWS CodeDeploy AppSpec file: SourceArtifact: appspec-customer.yaml

Dynamically update task definition image - optional: Select input artifact: IMAGE

CloudShell Feedback

11°C Cloudy

Search

CloudShell Feedback

Edit update-customer-microservice

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/edit?region=us-east-1

Edit action

Action name: Image

Action provider: Amazon ECR

Repository name: customer

Image tag - optional: latest

Variable namespace - optional:

Output artifacts: Image-custome

Cancel Done

Edit update-customer-microservice

us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/edit?region=us-east-1

Choose an input artifact for this action. [Learn more](#)

Image-customer

Add

No more than 100 characters

AWS CodeDeploy application name

Choose one of your existing applications, or create a new one in AWS CodeDeploy.

Q. microservices

Create application

AWS CodeDeploy deployment group

Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.

Q. 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-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.

Select input artifact

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

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](#)

Cloudy 11°C

Search

6:05 PM 4/25/2024

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

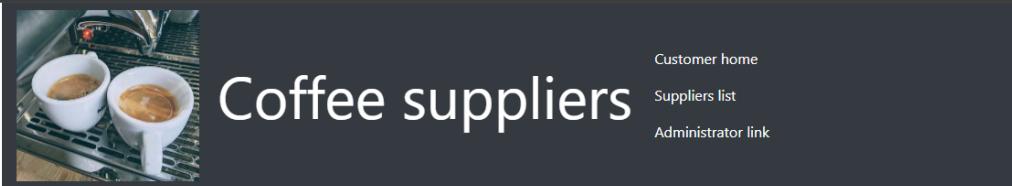
1. Launch a deployment of the customer microservice on Amazon ECS on Fargate by Clicking on Release Change on Codepipeline

The screenshot shows the AWS CodePipeline console with the pipeline 'update-customer-microservice'. The pipeline is currently in the 'QUEUED' state. It consists of two stages: 'Source' and 'Deploy'. The 'Source' stage is marked as 'Succeeded' and has a pipeline execution ID of '84845cca-a3bb-4fa7-ae96-669ec7765dd'. The 'Deploy' stage is marked as 'Failed' and has a pipeline execution ID of '1f27da78'. A tooltip for the 'Deploy' stage indicates the failure reason: 'Source: Add TaskDefinition and AppSpec files for employee & customer microservices'. On the right side of the pipeline, there is a vertical toolbar with a green checkmark icon and a red X icon. The browser address bar shows 'us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-customer-microservice/view?region=us-east-1'.

2. Observe the progress in CodeDeploy.

The screenshot shows the AWS CodeDeploy console interface. On the left, there's a sidebar with navigation links for Source, Artifacts, Build, Deploy, Pipeline, Settings, and Go to resource. The main content area has two tabs: 'Task set activity' and 'Deployment lifecycle events'. The 'Task set activity' tab displays two task sets: 'ecs-svc/8938186913323259518' (Replacement, ACTIVE, 0 desired, 1 running, 1 pending) and 'ecs-svc/1215782903995355988' (Original, PRIMARY, 100% desired, 0 running, 1 pending). The 'Deployment lifecycle events' tab lists several events with their status and timestamps: BeforeInstall (Succeeded), Install (In progress), AfterInstall (Pending), AllowTestTraffic (Pending), AfterAllowTestTraffic (Pending), BeforeAllowTraffic (Pending), AllowTraffic (Pending), and AfterAllowTraffic (Pending). The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, and various icons.

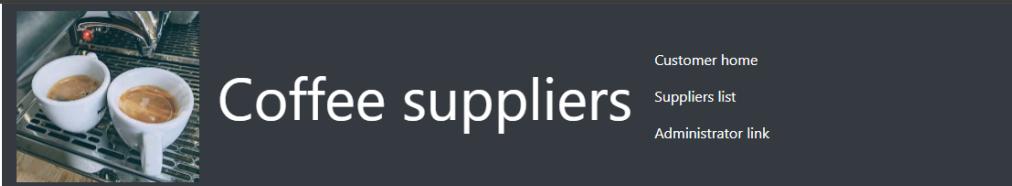
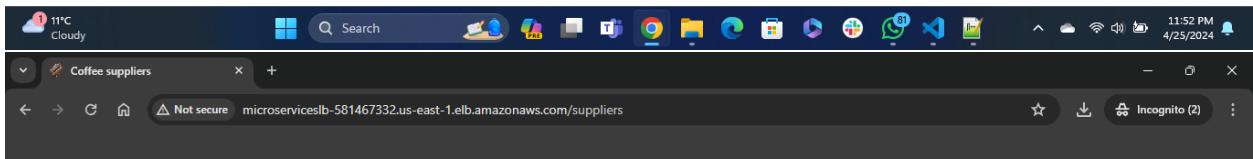
- Load the customer microservice in a new tab and see if it is working properly by copying the DNS name of LoadBalancer.



Welcome

Use this app to keep track of your coffee suppliers

[List of suppliers](#)



All suppliers

Name	Address	City	State	Email	Phone
Muhammad Owais Imran - Microservice	110 Lake St	Jersey City	New Jersey	mimran1@stevens.edu	+1 516 675 7898



4. Observe the ECS cluster services and load balancer target group setting (we can see that the forward to option of both listener rule is now same due to traffic forwarding and test deployment successful)

Screenshot of the AWS Cloud Console showing the Cluster tasks | Elastic Container Service page for the microservices-serverlesscluster task.

Cluster overview

ARN arn:aws:ecs:us-east-1:217120064673:cluster/microservices-serverlesscluster	Status Active	CloudWatch monitoring Default	Registered container instances -
---	---	---	-------------------------------------

Services

Draining -	Active 2	Pending -	Running 1
---------------	-------------	--------------	--------------

Tasks

Task	Last status	Desired state	Task ...	Health state	Started at	Container instant...	Launch type	Platform ...
db0d9...	Pending	Running	customer...	Unknown	-	-	FARGATE	1.4.0
dc01d...	Running	Running	customer...	Unknown	3 minutes ago	-	FARGATE	1.4.0

Tasks (2)

Filter tasks by property or value: Running

Load balancers (1/1)

elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
microservicesLB	microservicesLB-5814673...	Active	vpc-06f075ab6fa226f64	2 Availability Zones	application	April 25, 2024, 17:21 (...

Load balancer: microservicesLB

ProtocolPort	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate
HTTP:80	Forward to target group • customer-tg-one [1] (100%) • Group-level stickiness: Off	2 rules	ARN	Not applicable	Not applicable
HTTP:8080	Forward to target group • customer-tg-one [1] (100%) • Group-level stickiness: Off	2 rules	ARN	Not applicable	Not applicable

Task 8.4: Create a pipeline for the *employee* microservice

1. Create a Pipeline for Employee Microservice using the name update-employee-microservice.
2. Edit the *update-employee-microservice* pipeline to add another *source*.
 - a. In the **Edit: Source** section, choose **Edit stage**, then add an action with these details:
 - i. **Action name:** Image
 - ii. **Action provider:** Amazon ECR
 - iii. **Repository name:** employee
 - iv. **Image tag:** latest
 - v. **Output artifacts:** image-employee
3. Edit the *deploy* action of the *update-employee-microservice* pipeline.
 - a. Edit the **update-employee-microservice** pipeline
 - i. In the **Edit: Deploy** section, choose **Edit stage**, then add an input artifact as described below:
 1. On the **Deploy Amazon ECS (Blue/Green)** card, choose the edit (pencil) icon.
 2. Under **Input artifacts**, choose **Add** and then choose **image-employee**.

Note: You should now have *SourceArtifact* and *image-employee* as listed input artifacts.

- ii. Under **Dynamically update task definition image**, for **Input artifact with image details**, choose **image-employee**.
- iii. For **Placeholder text in the task definition**, enter IMAGE1_NAME

Pipeline settings

Pipeline name: update-employee-microservice

Pipeline type: V2

Execution mode: Queued (Pipeline type V2 required)

Service role: Existing service role

Role ARN: arn:aws:iam::217120064673:role/PipelineRole

Source

Source provider: AWS CodeCommit

Repository name: deployment

Branch name: dev

Change detection options:

- 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:

- 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.

The screenshot shows two separate browser windows side-by-side, both connected to the AWS CodePipeline service.

Top Window (Create new pipeline):

- Tab: Create new pipeline | CodePipeline
- Tab: Listener details | EC2 | us-east-1
- Tab: Task configuration | Elastic Container Registry

URL: us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipeline/new?region=us-east-1

Content: A modal dialog titled "Deploy action provider" is open. It lists several configuration parameters:

- Deploy action provider: Amazon ECS (Blue/Green)
- ApplicationName: microservices
- DeploymentGroupName: microservices-employee
- TaskDefinitionTemplateArtifact: SourceArtifact
- TaskDefinitionTemplatePath: taskdef-employee.json
- AppSpecTemplateArtifact: SourceArtifact
- AppSpecTemplatePath: appspec-employee.yaml
- Image1ArtifactName: SourceArtifact
- Image1ContainerName: IMAGE1_NAME

Bottom Window (Edit update-employee-microservice):

- Tab: Edit update-employee-microservice
- Tab: Listener details | EC2 | us-east-1
- Tab: Task configuration | Elastic Container Registry

URL: us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-employee-microservice/edit?region=us-east-1

Content: An "Edit action" configuration dialog is open. The "Action provider" is set to "Amazon ECR". The "Repository name" is "Q_employee". The "Image tag - optional" field contains "latest". The "Variable namespace - optional" field is empty. The "Output artifacts" field contains "image-employee".

Screenshot of the AWS CodePipeline console showing the configuration of a new pipeline step.

The pipeline step is titled "Edit update-employee-microservice".

Configuration details:

- AWS CodeDeploy application name:** microservices
- 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: image-employee
- Placeholder text in the task definition:** IMAGE1_NAME
- Variable namespace - optional:** DeployVariables

The browser address bar shows: us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/update-employee-microservice/edit?region=us-east-1

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

1. Launch a deployment of the employee microservice on Amazon ECS on Fargate by Clicking on Release Change on Codepipeline

The image consists of three vertically stacked screenshots of the AWS CodePipeline console, showing the execution of a pipeline named "update-employee-microservice".

Screenshot 1: Pipeline execution ID: `e1fd203b-58d2-449c-b7b8-9826c46c2a8d`. The "Source" stage is listed as "Succeeded". The "Deploy" stage is shown as "In progress". A "Disable transition" button is visible between the stages.

Screenshot 2: Pipeline execution ID: `e1fd203b-58d2-449c-b7b8-9826c46c2a8d`. The "Source" stage is listed as "Succeeded". The "Deploy" stage is shown as "In progress". A "Disable transition" button is visible between the stages.

Screenshot 3: Pipeline execution ID: `e1fd203b-58d2-449c-b7b8-9826c46c2a8d`. The "Source" stage is listed as "Succeeded". The "Deploy" stage is shown as "In progress" using the "Amazon ECS (Blue/Green)" provider. A "Disable transition" button is visible between the stages.

The screenshot shows the AWS CodeDeploy console with the following tabs open: "update-employee-microservice", "d-HND5IUS55 | CodeDeploy", "Listener details | EC2 | us-east-", and "Task configuration | Elastic Container". The main view displays deployment details for a task set activity and deployment lifecycle events.

Task set activity

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/7349544165585108593	Original	ACTIVE	0	1	0	0
ecs-svc/4141746588667480125	Replacement	PRIMARY	100%	1	1	0

Deployment lifecycle events

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 26, 2024 12:09 AM (UTC-4:00)	Apr 26, 2024 12:09 AM (UTC-4:00)
Install	2 minutes 3 seconds	Succeeded	Apr 26, 2024 12:09 AM (UTC-4:00)	Apr 26, 2024 12:11 AM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 26, 2024 12:11 AM (UTC-4:00)	Apr 26, 2024 12:11 AM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 26, 2024 12:11 AM (UTC-4:00)	Apr 26, 2024 12:11 AM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 26, 2024 12:11 AM (UTC-4:00)	Apr 26, 2024 12:11 AM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 26, 2024 12:11 AM (UTC-4:00)	Apr 26, 2024 12:11 AM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 26, 2024 12:11 AM (UTC-4:00)	Apr 26, 2024 12:11 AM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 26, 2024 12:11 AM (UTC-4:00)	Apr 26, 2024 12:11 AM (UTC-4:00)

CloudShell Feedback 11°C Cloudy Search © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 12:13 AM 4/26/2024

2. Check if admin link is working properly

The screenshot shows a web browser window with the title "Manage coffee suppliers". The URL in the address bar is "microserviceslb-581467332.us-east-1.elb.amazonaws.com/admin/suppliers". The page features a header with a coffee machine image and the text "Manage coffee suppliers". On the right side of the header are links for "Administrator home", "Suppliers list", and "Customer home". Below the header, the text "All suppliers" is displayed. A table lists a single supplier entry:

Name	Address	City	State	Email	Phone
Muhammad Owais Imran - Microservice	110 Lake St	Jersey City	New Jersey	mimran1@stevens.edu	+1 516 675 7898

A green button labeled "Add a new supplier" is located at the bottom left of the table.

The screenshot shows a web browser window with multiple tabs open, including "update-employee-microservice", "d-HND5IUS55 | CodeDeploy", "Listener details | EC2 | us-east-", and "Cluster tasks | Elastic Container Service". The active tab is "Cluster tasks | Elastic Container Service". The URL in the address bar is "us-east-1.console.aws.amazon.com/ecs/v2/clusters/microservices-serverlesscluster/tasks?region=us-east-1". The page displays the "Cluster overview" and the "Tasks" section. The "Tasks" section shows two tasks:

Task	Last status	Desired state	Task ID	Health state	Started at	Container instance	Launch type	Platform
40389242dad...	Running	Running	employee...	Unknown	5 minutes ago	-	FARGATE	1.4.0
dc01d7591dbf...	Running	Running	customer...	Unknown	25 minutes ago	-	FARGATE	1.4.0

At the bottom of the page, there are links for "CloudShell" and "Feedback". The browser's toolbar and status bar are visible at the top and bottom of the window.

Significance & Importance of this Task

This task helps us in implementing:

- How to automate this entire application deployment process with zero interaction of the developers with the production environment.
- The infrastructure and workflow of AWS CodePipeline and its integration with AWS CodeDeploy, AWS ECR, and AWS CodeCommit.

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

Task 9.1: Limit access to the *employee* microservice

1. Edit the rules for HTTP: 80 and HTTP:8080 listener rules to restrict access to /admin/* routes to your IP address only.

The screenshot shows the AWS CloudShell interface with the following details:

- Step 1 Edit listener rule | EC2**: The current step.
- Cluster tasks | Elastic Container**: A task in progress.
- What Is My IP? Best Way To C**: A search result.
- us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#EditListenerRule:ruleArn=arn:aws:elasticloadbalancing:us-east-1:2171200646... Incognito (2)**: The browser tab.

The main content area displays the "Define rule conditions" step of the listener rule editor. It shows two conditions defined:

- Path (1)**: If Path is /admin/*.
- Source IP (1)**: If Source IP is 73.29.51.70/32.

The conditions are joined by an **AND** operator. The "Next" button is visible at the bottom right.

The screenshot shows the AWS CloudShell interface with the following details:

- Step 3 Edit listener rule | EC2**: The current step.
- Cluster tasks | Elastic Container**: A task in progress.
- What Is My IP? Best Way To C**: A search result.
- us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#EditListenerRule:ruleArn=arn:aws:elasticloadbalancing:us-east-1:2171200646... Incognito (2)**: The browser tab.

The main content area displays the "Review changes" step of the listener rule editor. It shows the rule details:

Priority	Conditions (If)	Actions (Then)
1	If request matches all: <ul style="list-style-type: none">Path Pattern is /admin/*, ANDSource IP is 73.29.51.70/32	Forward to target group <ul style="list-style-type: none">employee-tg-one [2]: 1 (100%)Group-level stickiness: Off

The Rule ARN is listed as arn:aws:elasticloadbalancing:us-east-1:217120064673:listener-rule/app/microservicesLB/d5f09bc54137b5c9/6811e64242b76572/aa3936f5c335a04c. The "Save changes" button is visible at the bottom right.

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

1. Adjust the navigation bar theme of employee microservice from light to dark, and create a new docker image, and push the newly created docker image onto ECR to detect this change.

2.

```

1 <nav class="navbar navbar-expand-lg navbar-light bg-light">
2   <img alt="img/expresso.jpg" width="200"/>
3   <div><a href="/supplier">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="/Customer home">Customer home</a>
8       <li class="nav-item">
9         <a class="nav-link" href="/suppliers">Suppliers list</a>
10      <a class="nav-link" href="/admin/suppliers">Administrator link</a>
11    </ul>
12  </div>
13 </nav>

```

```

bash ->ip-10-10-10-233.e X Immediate
voclabs:-/environment $ 

```

```

1 <nav class="navbar navbar-expand-lg navbar-light bg-light">
2   <img alt="img/expresso.jpg" width="200"/>
3   <div><a href="/supplier">Manage coffee supplies</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       <li class="nav-item">
9         <a class="nav-link" href="/suppliers">Suppliers list</a>
10      <a class="nav-link" href="/">Customer home</a>
11    </ul>
12  </div>
13 </nav>

```

```

docker ->ip-10-10-10-233 X Immediate
voclabs:-/environment $ 

```

```

found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
  Removing intermediate container cb3d4172ba3
  --> 259540ec64af: EXPOSE 8080
  Step 6/7 : EXPOSE 8080
  --> Running in 463f90773097
  Removing intermediate container 463f90773097
  --> 59c64fab8e67
Step 7/7 : CMD ["npm", "run", "start"]
  --> Running in 7c32d6151a1e
  Removing intermediate container 7c32d6151a1e
Success! Built in 1512ms
Successfully tagged employee:latest
(reverse-i-search)@: sed -i "s/[MICROSERVICE-TG-TWO-ARN]arn:aws:elasticloadbalancing:us-east-1:217120864673:targetgroup/customer-tg-two/5db2*Chees743758|g" ./create-customer-microservice-tg-two.json
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) $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
The push refers to repository [217120864673.dkr.ecr.us-east-1.amazonaws.com/employee]
  The push refers to repository [217120864673.dkr.ecr.us-east-1.amazonaws.com/employee]
  9844d3bc6c97: Pushed
  7c9b21cfcc3cb: Layer already exists
  d8d715330b7: Layer already exists
  1dc7f3bb0944: Layer already exists
  dcac67293044: Layer already exists
  f395466d5: Layer already exists
  latest: digest: sha256:34cf3bdff673697252735338bb70316923ccf14056c3977084ecc1347a510debd6 size: 1783
voclabs:-/environment/microservices/employee (dev) $ 

```

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

1. Observe that the employee pipeline run was executed.

The screenshot shows the AWS CodePipeline console. On the left, a sidebar lists pipeline stages: Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Deploy (CodeDeploy), Pipeline (CodePipeline), Getting started, Pipelines, Settings, Go to resource, and Feedback. The main area displays the 'Pipelines' page with two entries:

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
update-employee-microservice (Type: V2 Execution mode: QUEUED)	In progress	Source - 1f274a7b: Add TaskDefinition and AppSpec files for employee & customer microservices Image - sha256:3	Just now	View details
update-customer-microservice (Type: V2 Execution mode: QUEUED)	Succeeded	Image - sha256:b: Source - 1f274a7b: Add TaskDefinition and AppSpec files for employee & customer microservices	39 minutes ago	View details

2. Verify after successful run, image was uploaded onto ECR.

The screenshot shows the Amazon Elastic Container Registry (ECR) console. On the left, a sidebar lists Private registry (Repositories, Summary, Images, Permissions, Lifecycle Policy, Repository tags, Settings) and Public registry (Repositories, Settings). The main area displays the 'employee' repository under 'Images (2)'. The table shows:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	April 26, 2024, 00:28:58 (UTC-04)	27.70	Copy URI	sha256:3acf3bdf6736972...
-	Image	April 25, 2024, 00:07:53 (UTC-04)	27.70	Copy URI	sha256:d837f87b266737...

Task 9.4: Test access to the *employee* microservice

1. Access website with the same machine IP

All suppliers

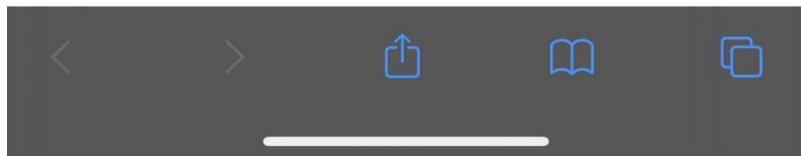
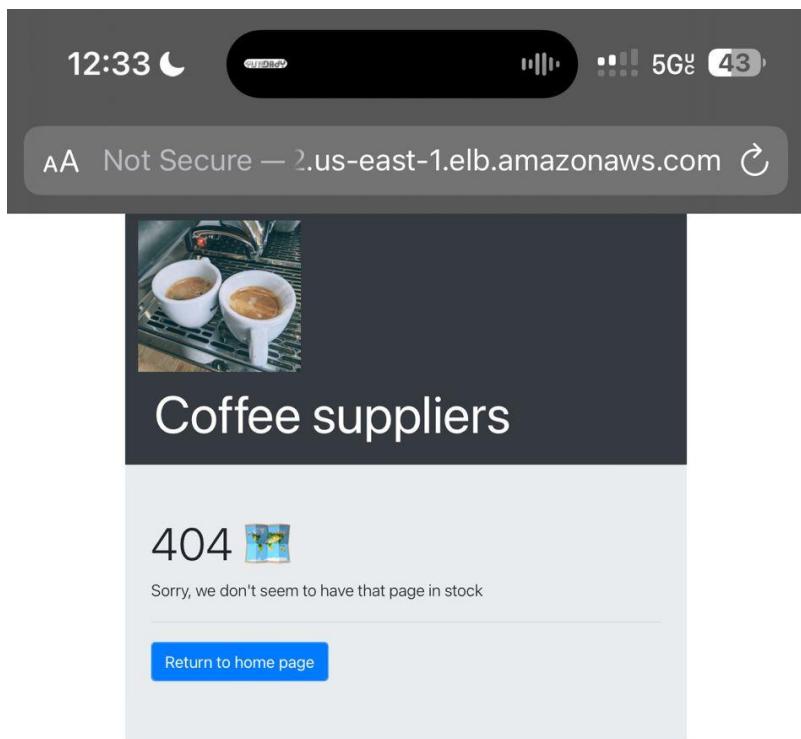
Name	Address	City	State	Email	Phone
Muhammad Owais Imran - Microservice	110 Lake St	Jersey City	New Jersey	mimran1@stevens.edu	+1 516 675 7898

Add a new supplier

Administrator home
Suppliers list
Customer home



2. Access website with another device allocated a different Public IP



Task 9.5: Scale the customer microservice

1. Update the customer microservice to run 3 containers

```
aws --region us-east-1 ecs update-service --cluster microservices-serverlesscluster --service customer-microservice --desired-count 3
```

The screenshot shows a browser window with multiple tabs. The main tab displays a code editor with HTML and CSS snippets for a navigation bar. Below the code editor is a terminal window showing the AWS CLI command to update the customer microservice's task count. The terminal output indicates the command was successful.

2. Check the change in ECS Tasks, we can see for customer service, running task are now x/3

The screenshot shows the AWS CloudWatch Metrics console. On the left, there is a sidebar with various AWS services like Amazon ECR, AWS Batch, Documentation, and Subscriptions. The main area is titled "Cluster overview" and shows the ARN of the cluster and its status as Active. Below this, there is a table for "Tasks" showing 2 Active tasks and 1 Pending task. At the bottom, there is a detailed view for the "customer-microservice" service, showing two active tasks with the ARN arn:aws:ecs:us-east-1:217120064673:task-definition/customer-microservice:3. The tasks are labeled "2/1 Tasks running" and "1/3 Tasks running".

Significance & Importance of this Task

This phase prepares us and gives us an overview of the immense power of AWS ECS and its services in terms of scalability and flexibility. It helps in realizing how easily we can upscale an application without doing any difficult server-side configuration. We just need to run a simple command, and within a few moments, our application can be scaled easily.

Moreover, we also learnt how microservice can help in upscaling a specific part of application as compared to monolithic architecture.

Example, the supplier part is something where the admin is responsible for adding/updating/deleting data, this part of application won't be used by most of the application users, however the customer facing side can have a huge user base. So, if we were to use a monolithic architecture, we have to upscale the entire application, which would upscale the admin part of application as well, which won't have any positive impact on resource consumption.