

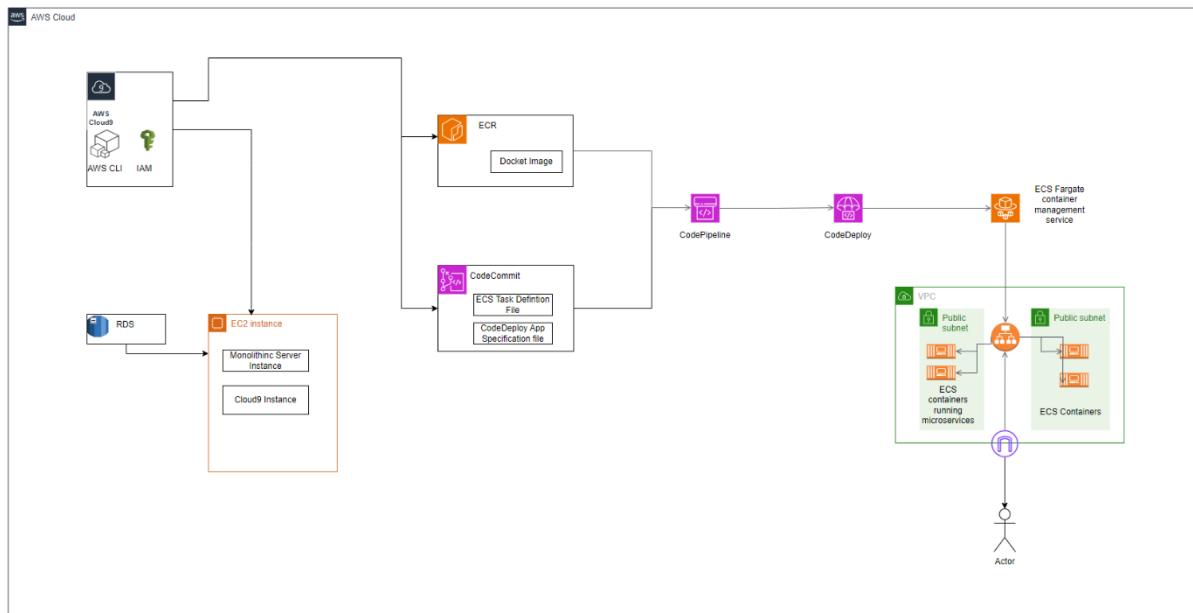
CS 524 – Course Project

Submitted By – Shweta Madhale

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

Execution

Phase 1 - Planning the design and estimating cost



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

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

Task 2.1: Verify that the monolithic application is available

- Navigate to the Amazon EC2 console.

The screenshot shows the AWS EC2 Home page for the us-east-1 region. The left sidebar contains navigation links for EC2 Dashboard, EC2 Global View, Events, Console-to-Code Preview, Instances, Images, Elastic Block Store, Network & Security, and CloudShell/Feedback. The main content area displays the following sections:

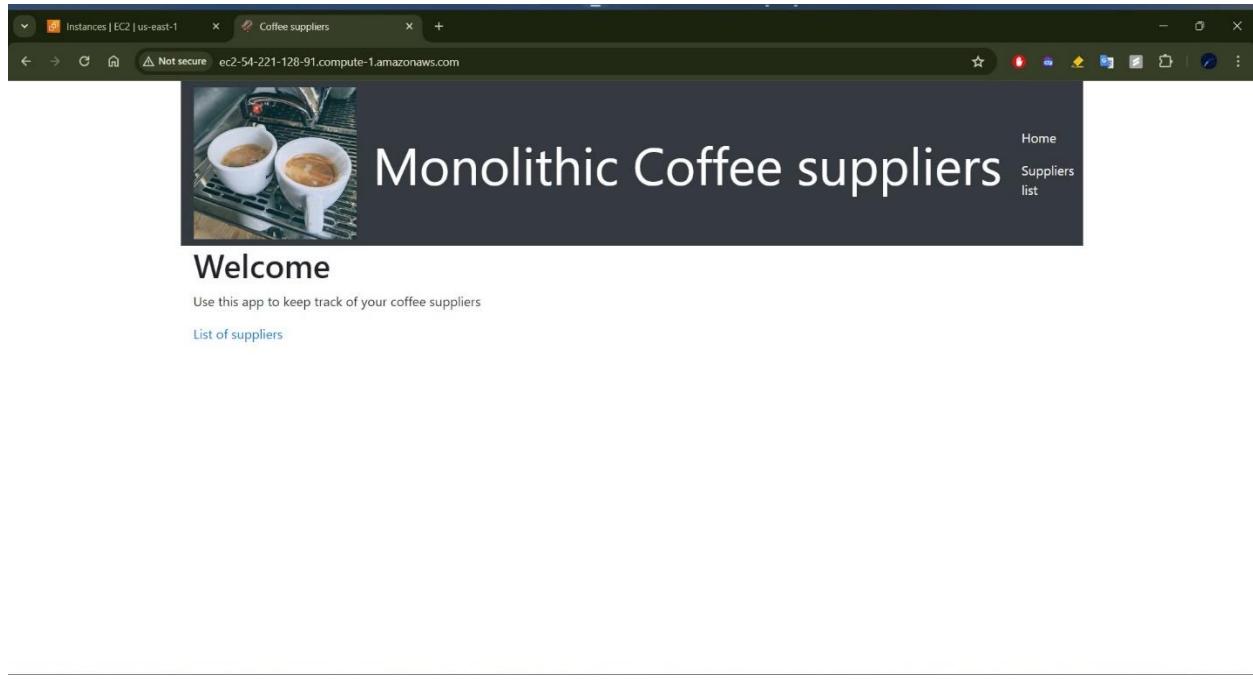
- Resources:** Shows a summary of Amazon EC2 resources in the US East (N. Virginia) Region. The table includes:

	Instances (running)	Auto Scaling Groups	Dedicated Hosts
Instances	1	0	0
Elastic IPs	0	Instances	3
Load balancers	0	Placement groups	0
Snapshots	0	Security groups	4
	Volumes		
	1		
- Launch instance:** A button to "Launch instance" and a link to "Migrate a server". A note states: "Note: Your instances will launch in the US East (N. Virginia) Region".
- Service health:** Shows the region as "US East (N. Virginia)" with status "This service is operating normally".
- Zones:** A table of availability zones:

Zone name	Zone ID
us-east-1a	use1-az4
us-east-1b	use1-az6
us-east-1c	use1-az1
us-east-1d	use1-az2
us-east-1e	use1-az3
- Account attributes:** Includes "Default VPC" (vpc-00f0ecb57933a875), "Settings" (Data protection and security, Zones, EC2 Serial Console, Default credit specification, Console experiments), and "Explore AWS" sections.

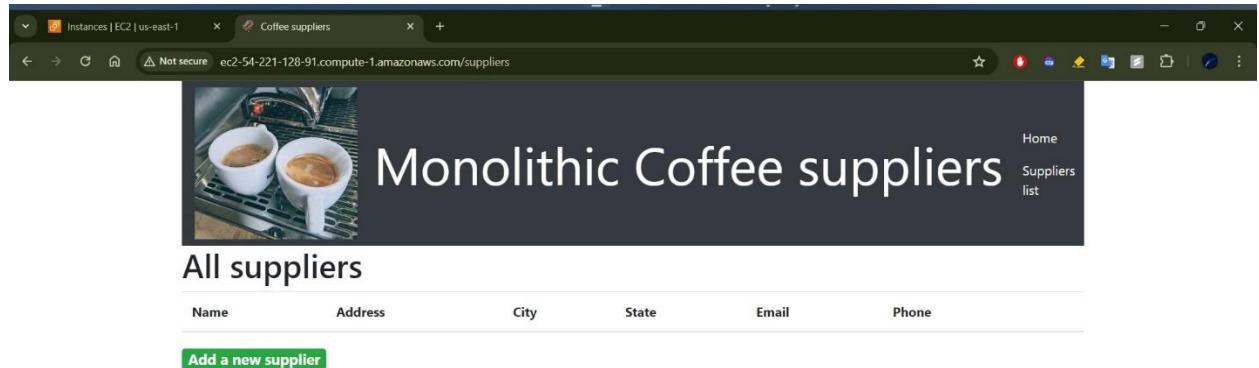
Instances (1/1) Info: A table showing one instance named "MonolithicAppServer" with details like Instance ID (i-0b5182af853fc5f8), Status (Running), Type (t2.micro), and Public IPv4 (54.221.128.91).

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



Task 2.2: Test the monolithic web application

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



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

Instances | EC2 | us-east-1 Coffee suppliers +

Not secure ec2-54-221-128-91.compute-1.amazonaws.com/supplier-add



Monolithic Coffee suppliers

Home
Suppliers list

All fields are required

Name
 Enter name
Name of this supplier

Address
 enter address
Address for this supplier

City
 enter city
City for this supplier

State
 enter state
State for this supplier

Email
 Email for this supplier

Name
 shweta
Name of this supplier

Address
 Columbia ave
Address for this supplier

City
 Jersey City
City for this supplier

State
 New Jersey
State for this supplier

Email
 smadhale@stevens.edu
Email for this supplier

Phone
 5513285708
Phone number for this supplier

The screenshot shows a web browser window with the URL <http://ec2-54-221-128-91.compute-1.amazonaws.com/suppliers>. The page has a dark header with the title "Monolithic Coffee suppliers". Below the header is a banner featuring two white cups of coffee on a coffee machine. To the right of the banner are links for "Home" and "Suppliers list". The main content area is titled "All suppliers" and contains a table with one row of data:

Name	Address	City	State	Email	Phone
shweta	Columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

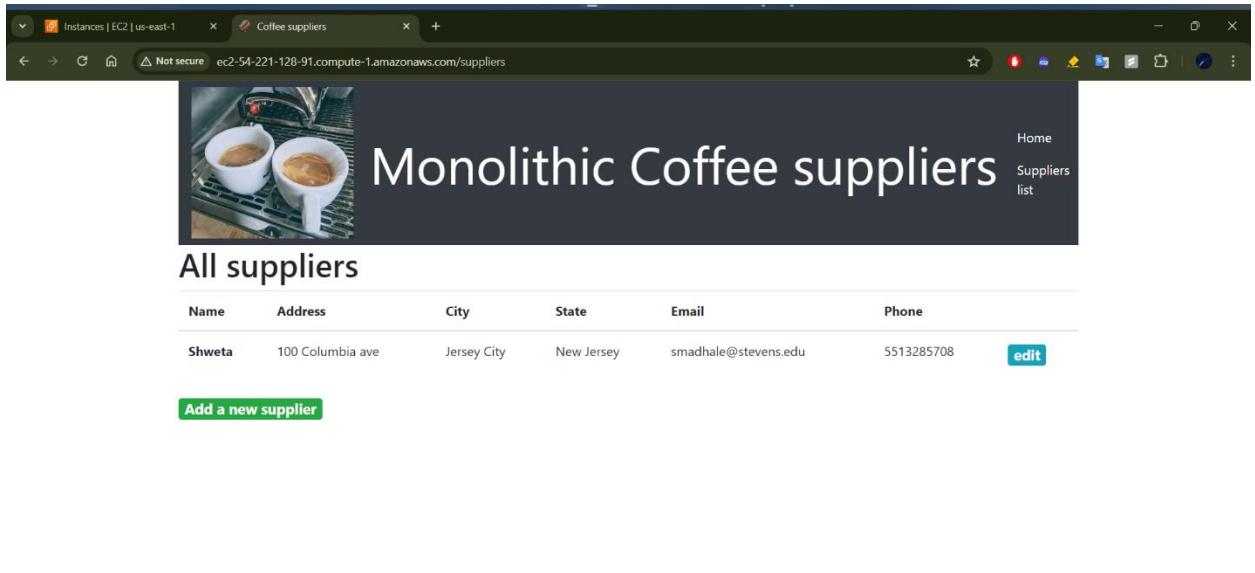
A green button labeled "Add a new supplier" is located at the bottom left of the table.

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

The screenshot shows a web browser window with the URL <http://ec2-54-221-128-91.compute-1.amazonaws.com/supplier-update/1>. The page displays a form for editing a supplier entry. The form fields are as follows:

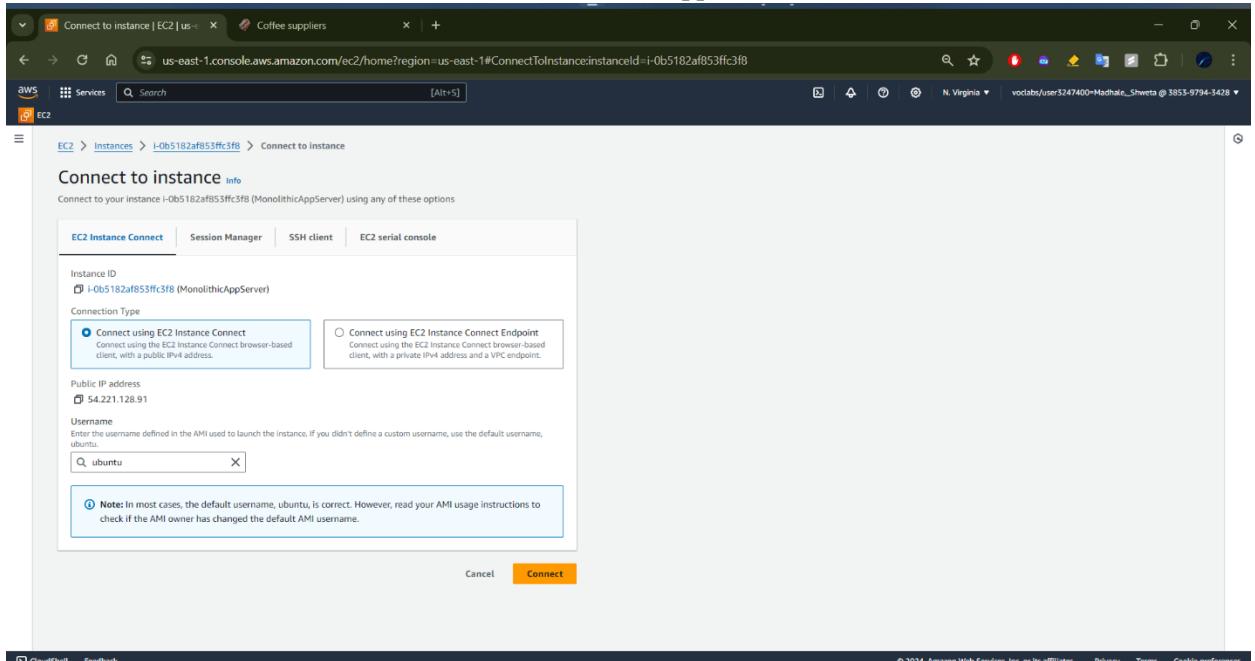
- Name:** shweta (placeholder: Name of this supplier)
- Address:** 100 Columbia ave (placeholder: Address for this supplier)
- City:** Jersey City (placeholder: City for this supplier)
- State:** New Jersey (placeholder: State for this supplier)
- Email:** smadhale@stevens.edu (placeholder: Email for this supplier)
- Phone:** 5513285708 (placeholder: Phone number for this supplier)

At the bottom of the form are a blue "Submit" button and a red "Delete this supplier" link.



Task 2.3: Analyze how the monolithic application runs

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



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

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

The screenshot shows a Windows taskbar with several open windows. From left to right, the visible titles are: Instances | EC2 | us-east-1, EC2 Instance Connect | us-east-1, Coffee suppliers, and vocabs/user3247400-Madhale_Shweta @ 3853-9794-3428*. Below the taskbar is a CloudShell window titled 'aws' with the URL 'us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0b5182af853ffcf3f8&osUser=ubuntu®ion=us-east-1'. The window displays system information, a package upgrade notice, and a list of running processes. At the bottom of the CloudShell window, it says 'i-0b5182af853ffcf3f8 (MonolithicAppServer)' and 'PublicIPs: 54.221.128.91 PrivateIPs: 10.16.10.13'.

```

aws Documentation: https://help.ubuntu.com
Management: https://landscape.canonical.com
Support: https://ubuntu.com/advantage

System information as of Mon Apr 29 14:49:05 UTC 2024
System load: 0.09      Processes: 105
Usage of /: 30.1% of 7.69GB Users logged in: 0
Memory usage: 29%      IPv4 address for eth0: 10.16.10.13
Swap usage: 0%
300 updates can be installed immediately.
212 of these updates are security updates.
To see these additional updates run: apt list --upgradable

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*-copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@i-0b5182af853ffcf3f8:~$ sudo lsof -i :80
COMMAND  PID USER   FD  TYPE DEVICE SIZE/OFF NODE NAME
node  15084 root    18u  IPv6  50594      0T0  TCP *:http (LISTEN)
ubuntu@i-0b5182af853ffcf3f8:~$ ps -ef | head -1; ps -ef | grep node
UID          PPID  C STIME   TT  CPU% MEM% COMMAND
root        15084      0 14:49 14:49 00:00:00 grep --color=auto node
ubuntu     15593  15573      0 14:49 pts/0    00:00:00 grep --color=auto node
ubuntu@i-0b5182af853ffcf3f8:~$ cd ~/resources/codebase_partner
ubuntu@i-0b5182af853ffcf3f8:~/resources/codebase_partner$ ls
app           package-lock.json  package.json  public  views
app           package-lock.json  package.json  public  views
ubuntu@i-0b5182af853ffcf3f8:~/resources/codebase_partner$ 

```

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

The screenshot shows the AWS RDS console for the US East (N. Virginia) region. The left sidebar includes links for Instances, Databases, Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL Integrations, Events, Event subscriptions, Recommendations (0), and Certificate update. The main content area displays a banner for 'Introducing Aurora I/O-Optimized' and a 'Resources' section showing usage statistics: DB Instances (1/40), DB Clusters (0/40), Reserved instances (0/40), Snapshots (0), and Recent events (6). It also lists Parameter groups (1), Option groups (1), Subnet groups (1/50), and Supported platforms (VPC). Below this is a 'Create database' section with 'Restore from S3' and 'Create database' buttons, and a note that DB instances will launch in the US East (N. Virginia) region. A 'Recommended services' section lists EC2 Instance Connect and EC2, and a 'Recommended for you' section lists Test Your DR Strategy in Minutes, Amazon RDS Backup and Restore using AWS Backup, Implementing Cross-Region DR, and Migrate SSRS to RDS for SQL Server.

The screenshot shows the 'Databases' page within the AWS RDS console. The left sidebar is identical to the previous screenshot. The main content area shows a 'Databases' table with one entry: 'supplierdb' (Status: Available, Instance: MySQL Community, Region & AZ: us-east-1b, Engine: db.t3.micro, Size: 2.51%, Current activity: 1 Connections, Maintenance: none). A tooltip suggests creating a Blue/Green Deployment to minimize downtime during upgrades. The bottom of the screen shows the URL https://us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#databaseid=supplierdb and the standard AWS footer with links for Privacy, Terms, and Cookie preferences.

The screenshot shows the AWS RDS console for the 'us-east-1' region. On the left, the sidebar lists various database-related options like Databases, Query Editor, and Performance insights. The main content area is titled 'Connectivity & security'. It displays the endpoint information (supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com, port 3306), networking details (VPC LabVPC, subnet group c1t0323a2e05598b65611891w385397943428-dbsubnetgroup-r5rv7ymg2xuf, subnet subnet-0167d2e9bb3a32d1c, network type IPv4), and security information (VPC security groups DBSecurityGroup (sg-07d1d589930ceef6) Active, Publicly accessible No, Certificate authority rds-ca-rsa2048-g1, Certificate authority date May 25, 2061, 19:34 (UTC-04:00), DB instance certificate expiration date April 29, 2025, 10:41 (UTC-04:00)).

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

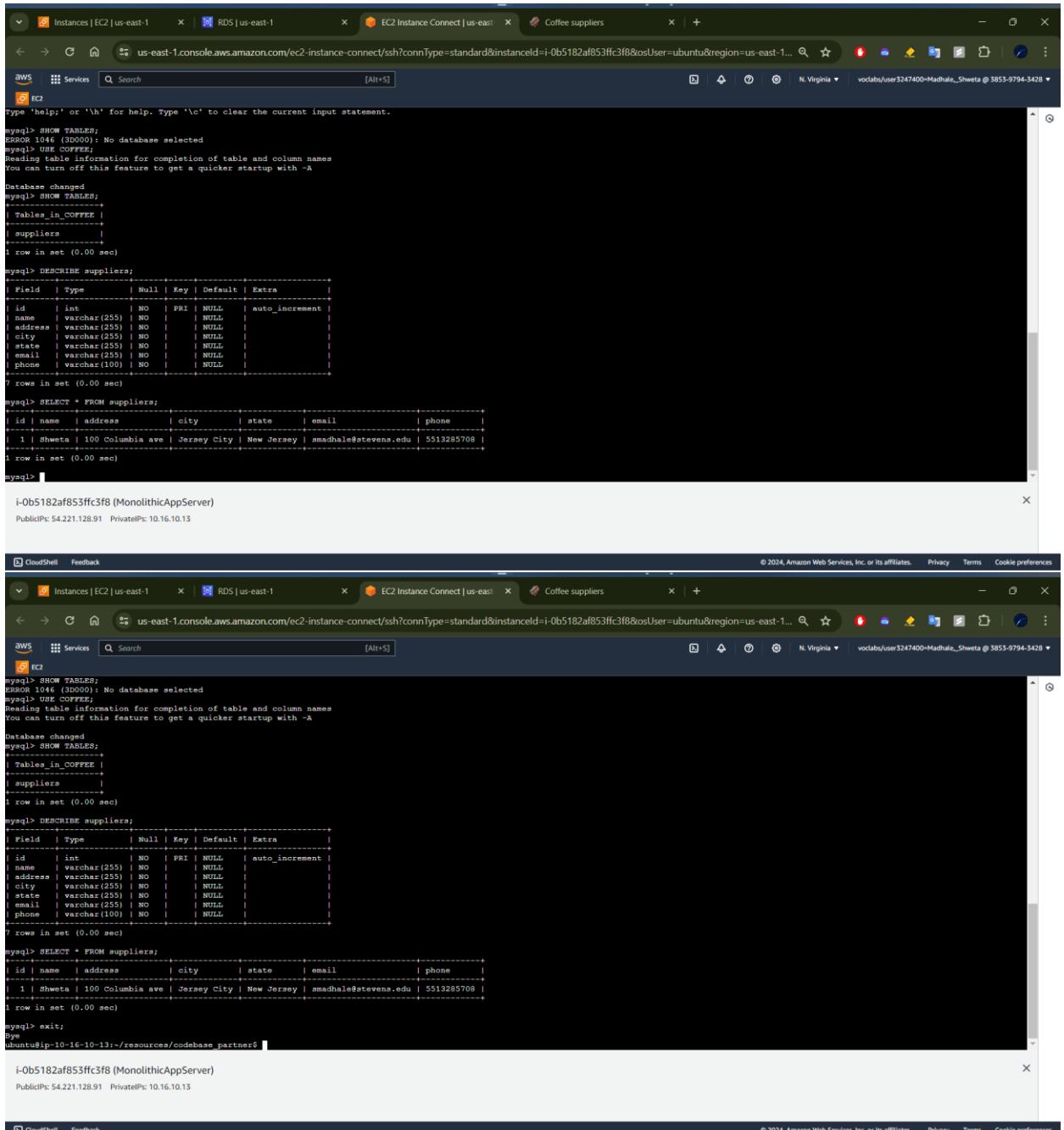
```

ubuntu#ip=10-16-10-13:$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 15684 root 18u IPv6 50594 0t0 TCP *:http (LISTEN)
ubuntu#ip=10-16-10-13:$ ps -ef | head -1; ps -ef | grep node
UID PID PPID C STIME TTY CMD
root 15684 1 0 2024-04-29 14:49 pts/0 00:00:00 grep --auto node
ubuntu 15593 15573 0 14:49 pts/0 00:00:00 grep --color=auto node
ubuntu#ip=10-16-10-13:$ cd ~/resources/database_partner
ubuntu#ip=10-16-10-13:/resources/database_partner$ ls
app package.json package-lock.json package.json public views
ubuntu#ip=10-16-10-13:/resources/database_partner$ nmap -Pn supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-29 14:50 UTC
Nmap scan report for supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com (10.16.40.195)
Host is up (0.0012s latency).
Nmap done: 1 IP address (1 host up) scanned in 8.16 seconds
PORT      STATE SERVICE
3306/tcp  open  mysql
$3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 8.16 seconds
ubuntu#ip=10-16-10-13:/resources/database_partner$ mysql -u admin -p -h supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 32
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>

```

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



```

Instances | EC2 | us-east-1      RDS | us-east-1      EC2 Instance Connect | us-east-1      Coffee suppliers
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0b5182af853fc3f8&osUser=ubuntu&region=us-east-1... Search N. Virginia vocabs/user3247400-Madhale_Shweta @ 3853-9794-3428

aws Services Search [Alt+S] EC2

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

mysql> SHOW TABLES;
ERROR 1046 (42000): No database selected
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> DESCRIBE suppliers;
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
| id   | int  | NO  | PRI | NULL    | auto_increment |
| name | varchar(255) | NO | NULL | NULL    |                |
| address | varchar(255) | NO | NULL | NULL    |                |
| city  | varchar(255) | NO | NULL | NULL    |                |
| state | varchar(255) | NO | NULL | NULL    |                |
| email | varchar(255) | NO | NULL | NULL    |                |
| phone | varchar(100) | NO | NULL | NULL    |                |
+-----+
7 rows in set (0.00 sec)

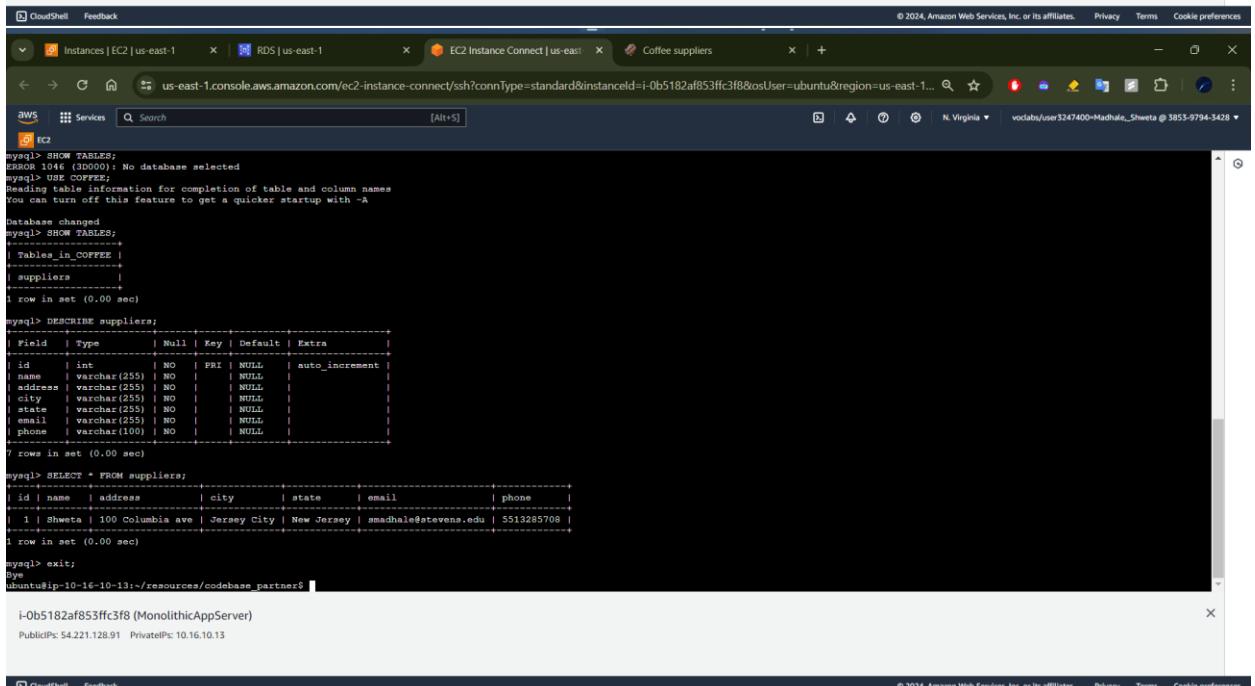
mysql> SELECT * FROM suppliers;
+-----+
| id   | name        | address     | city       | state      | email      | phone      |
+-----+
| 1    | Shweta      | 100 Columbia ave | Jersey City | New Jersey | smadhale@stevens.edu | 5513285708 |
+-----+
1 row in set (0.00 sec)

mysql> exit;
Bye
ubuntu@ip-10-16-10-13:/resources/codebase_partner$ 

```

i-0b5182af853fc3f8 (MonolithicAppServer)
PublicIP: 54.221.128.91 PrivateIP: 10.16.10.13

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



```

Instances | EC2 | us-east-1      RDS | us-east-1      EC2 Instance Connect | us-east-1      Coffee suppliers
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0b5182af853fc3f8&osUser=ubuntu&region=us-east-1... Search N. Virginia vocabs/user3247400-Madhale_Shweta @ 3853-9794-3428

aws Services Search [Alt+S] EC2

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

mysql> SHOW TABLES;
ERROR 1046 (42000): No database selected
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> DESCRIBE suppliers;
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
| id   | int  | NO  | PRI | NULL    | auto_increment |
| name | varchar(255) | NO | NULL | NULL    |                |
| address | varchar(255) | NO | NULL | NULL    |                |
| city  | varchar(255) | NO | NULL | NULL    |                |
| state | varchar(255) | NO | NULL | NULL    |                |
| email | varchar(255) | NO | NULL | NULL    |                |
| phone | varchar(100) | NO | NULL | NULL    |                |
+-----+
7 rows in set (0.00 sec)

mysql> SELECT * FROM suppliers;
+-----+
| id   | name        | address     | city       | state      | email      | phone      |
+-----+
| 1    | Shweta      | 100 Columbia ave | Jersey City | New Jersey | smadhale@stevens.edu | 5513285708 |
+-----+
1 row in set (0.00 sec)

mysql> exit;
Bye
ubuntu@ip-10-16-10-13:/resources/codebase_partner$ 

```

i-0b5182af853fc3f8 (MonolithicAppServer)
PublicIP: 54.221.128.91 PrivateIP: 10.16.10.13

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

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

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

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

The screenshot shows the AWS Cloud9 homepage. At the top right, there is a prominent orange button labeled "Create environment". Below this, there is a section titled "Getting started" with several links: "Before you start (2 min read)", "Create an environment (2 min read)", "Working with environments (15 min read)", "Working with the IDE (10 min read)", and "Working with AWS Lambda (5 min read)". On the left side, there is a "How it works" section and a "Benefits and features" section. The "Benefits and features" section includes two main points: "Code with just a browser" (described as allowing immediate access to a rich code editor, integrated debugger, and built-in terminal) and "Code together in real time" (described as making collaboration on code easy). The bottom of the page includes standard AWS navigation links like CloudShell and Feedback, and a footer with copyright information.

The screenshot shows the "Create environment" wizard in progress. The first step, "Details", is displayed. It requires entering a "Name" (set to "MicroservicesIDE") and an optional "Description". Under "Environment type", the "New EC2 instance" option is selected, which is highlighted with a blue border. This option allows Cloud9 to create a new EC2 instance in the user's account. Below this, there is a "New EC2 instance" section where the "Instance type" is chosen. The "t3.small (2 GB RAM + 2 vCPU)" option is selected, also highlighted with a blue border. Other options shown include "t2.micro (1 GB RAM + 1 vCPU)" and "m5.large (8 GB RAM + 2 vCPU)". There is also a link to "Additional instance types". The bottom of the page includes CloudShell and Feedback links, and a footer with copyright information.

The screenshot shows the 'Create' configuration page for AWS Cloud9. It includes fields for Platform (Amazon Linux 2023), Timeout (30 minutes), and Network settings (Secure Shell (SSH) selected). A note indicates that the instance can only be launched into a public subnet if accessed via SSH.

Platform: Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.
Amazon Linux 2023

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

Network settings: Info

Connection
How your environment is accessed.
 AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
 Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.

VPC settings: Info
Amazon Virtual Private Cloud (VPC)
The VPC that your environment will access. To allow the AWS Cloud9 environment to connect to its EC2 instance, attach an internet gateway (IGW) to your VPC. Create new VPC
vpc-096197396de75cc
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-0f9b9c97ba5cf7c86
Name - Public Subnet1

Note: You have chosen a public subnet for your Cloud9 environment, note the following:
• If accessing the EC2 instance directly through SSH, the instance can only be launched into a public subnet.

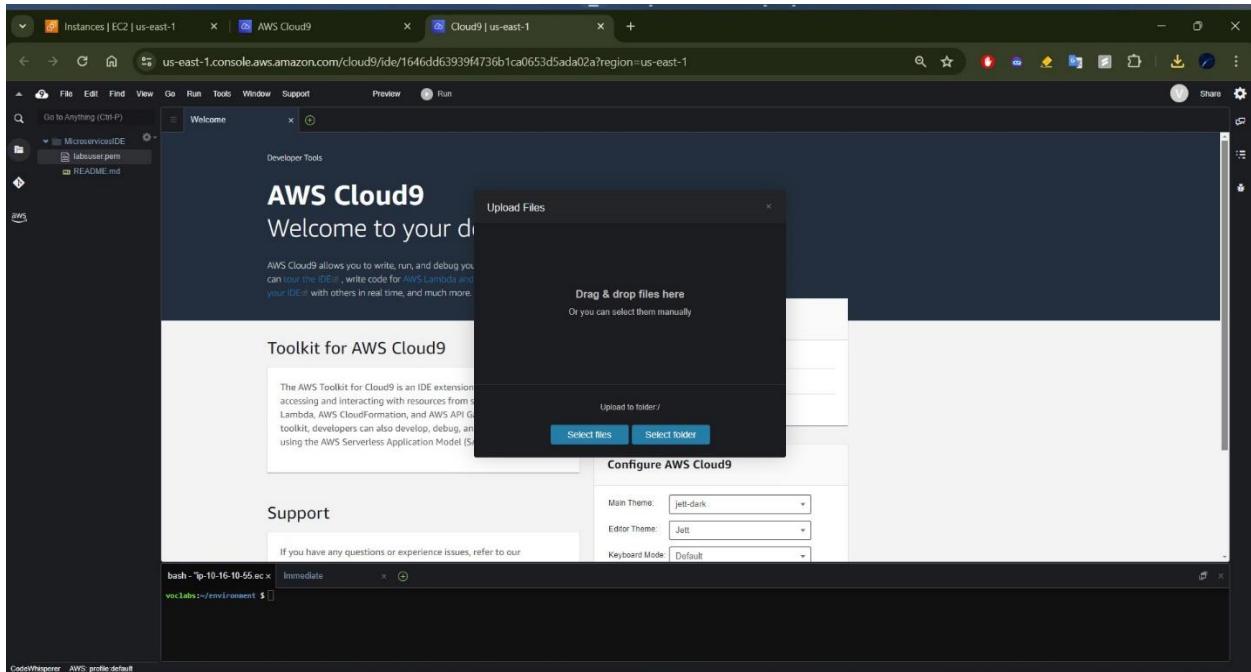
The screenshot shows the 'Environments' page for AWS Cloud9, displaying a single environment named 'MicroservicesIDE'. The page includes a 'Create environment' button and a table with columns for Name, Cloud9 IDE, Environment type, Connection, Permission, and Owner ARN.

AWS Cloud9

Successfully created MicroservicesIDE. To get the most out of your environment, see [Best practices for using AWS Cloud9](#).

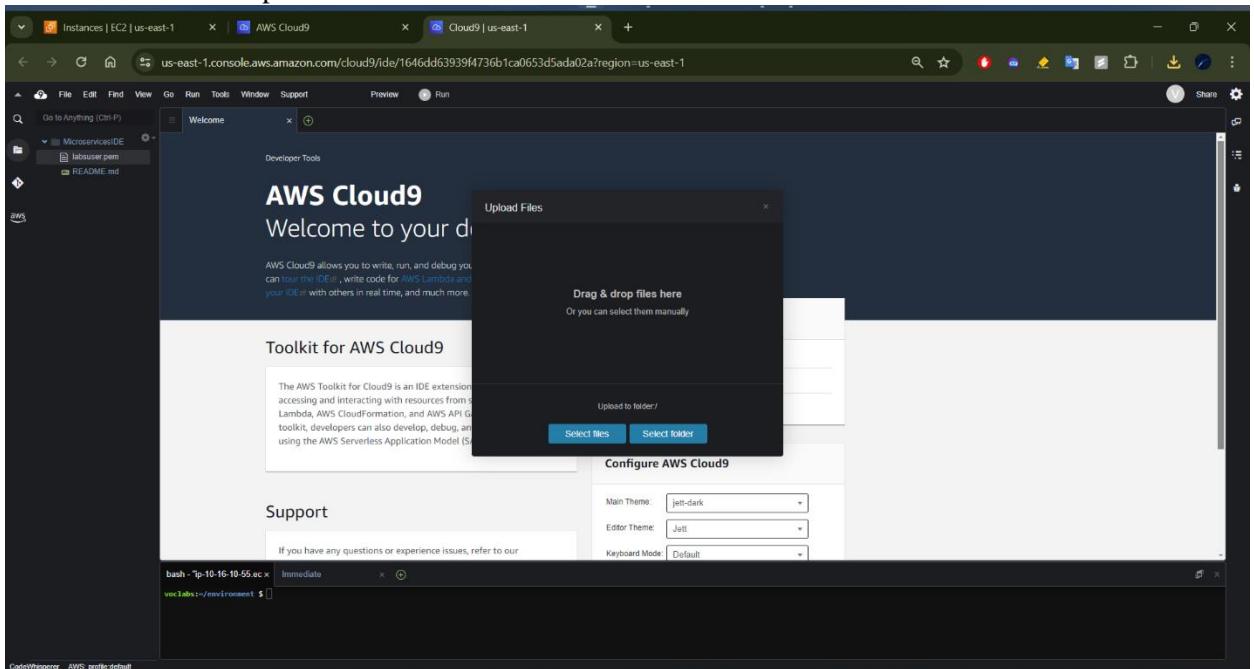
Environments (1)

Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
MicroservicesIDE	Open	EC2 instance	Secure Shell (SSH)	Owner	arn:aws:sts::385397943428:assumed-role/voclabs/user3247400=Madhale_Shweta



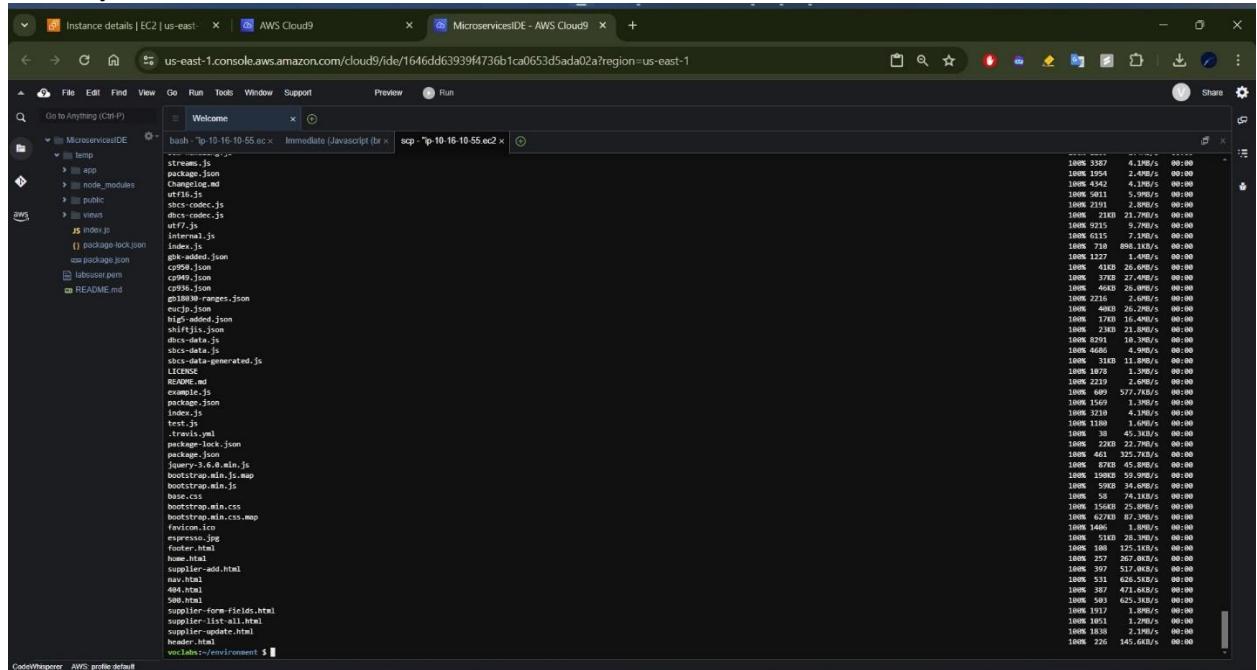
Task 3.2: Copy the application code to your IDE

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



```
bash - "ip-10-16-10-227.e x  Immediate  ×  +  
voclabs:~/environment $ chmod 400 labsuser.pem  
voclabs:~/environment $ pwd  
/home/ec2-user/environment  
voclabs:~/environment $ mkdir temp  
voclabs:~/environment $ cd temp  
voclabs:~/environment/temp $ █
```

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



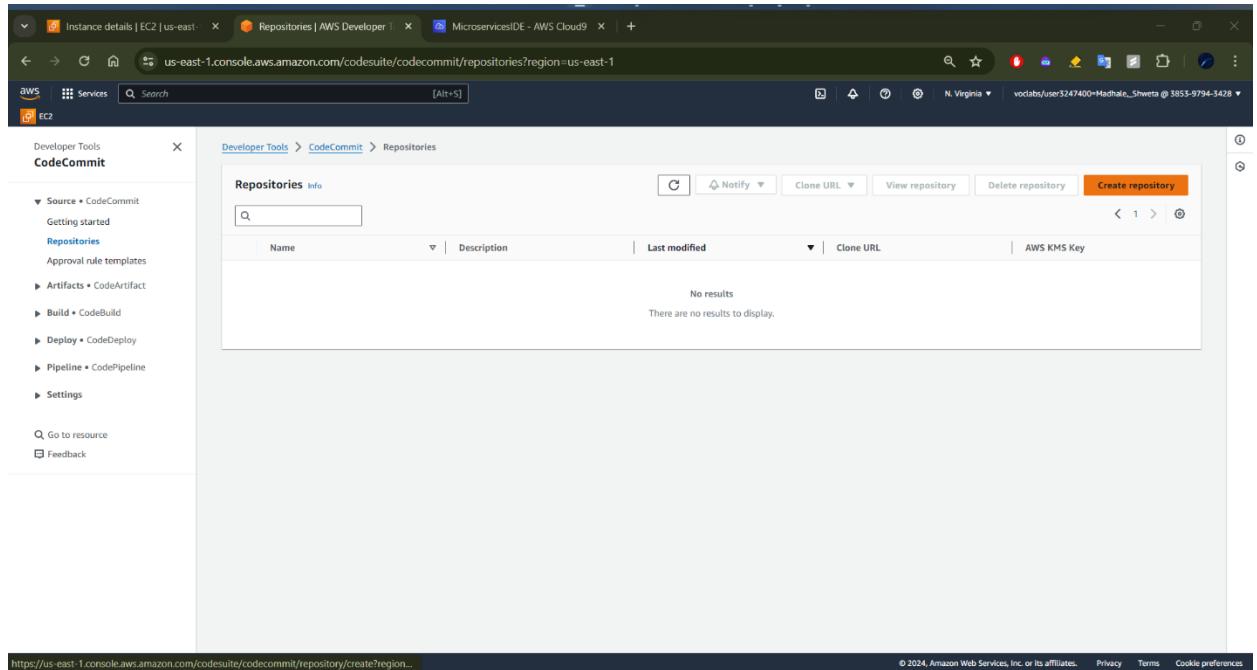
Task 3.3: Create working directories with starter code for the two microservices

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

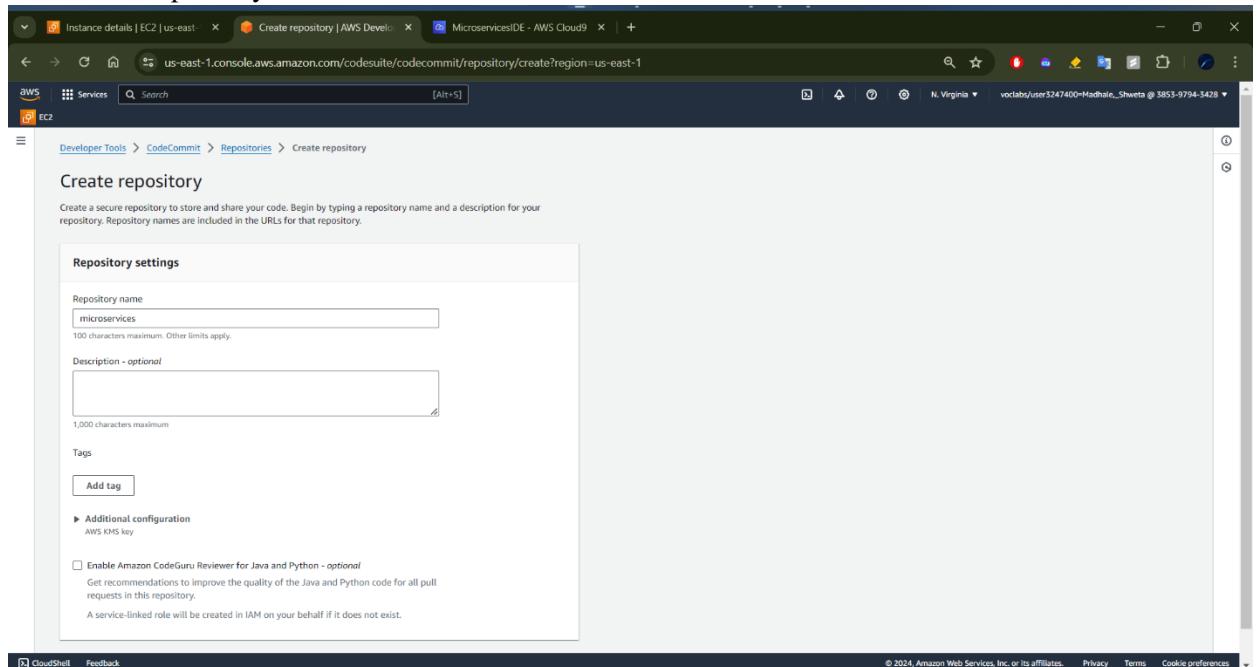
The screenshot shows a terminal window with a file tree on the left and a list of files on the right. The file tree is for a directory named 'MicroservicesIDE' containing two main subfolders: 'customer' and 'employee'. Each subfolder contains an 'app', 'node_modules', 'public', and 'views' folder, along with an 'index.js' file, a 'package-lock.json' file, and a 'package.json' file. The 'customer' folder also contains a 'streams.js' file. The 'employee' folder contains a 'Changelog.md' file. The right side of the terminal shows a list of files from a 'bash - "ip-10-16-10-5"' session, including 'streams.js', 'package.json', 'Changelog.md', 'utf16.js', 'sbcs-codec.js', 'dbc5-codec.js', 'utf7.js', 'internal.js', 'index.js', 'gbk-added.json', 'cp950.json', 'cp949.json', 'cp936.json', 'gb18030-ranges.json', 'eucjp.json', 'big5-added.json', 'shiftjis.json', 'dbc5-data.js', 'sbcs-data.js', 'sbcs-data-generated', 'LICENSE', 'README.md', 'example.js', 'package.json', 'index.js', 'test.js', '.travis.yml', 'package-lock.json', 'package.json', 'jquery-3.6.0.min.js', 'bootstrap.min.js.map', 'bootstrap.min.js', 'base.css', 'bootstrap.min.css', 'bootstrap.min.css.map', and 'favicon.ico'. The file 'labsuser.pem' is highlighted with a dark gray background.

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

- Navigate to CodeCommit



- Create new repository named microservices.



The screenshot shows a browser window with three tabs open: 'Instance details | EC2 | us-east-1', 'microservices | AWS Developer', and 'MicroservicesIDE - AWS Cloud9'. The main content area is a 'CodeCommit' repository setup page for 'microservices'. A green banner at the top says 'Success' and 'Repository successfully created'. Below it, a 'Copied' message shows the URL: <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices>. The page includes sections for 'Connection steps' (HTTPS, SSH, HTTPS (GRPC)), 'Step 1: Prerequisites', 'Step 2: Set up the AWS CLI Credential Helper', and 'Additional details'. On the left, a sidebar lists various AWS services like EC2, Lambda, S3, etc. At the bottom, there are links for 'CloudShell' and 'Feedback'.

- Execute

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

```

bash - 'ip-10-16-10-55.ec2.us-east-1.compute.amazonaws.com:80'
us-east-1.console.aws.amazon.com/cloud9/ide/1646dd63939f4736b1ca0653d5ada02a?region=us-east-1

File Edit Find Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)
MicroservicesIDE
└── microservices
    ├── customer
    │   ├── app
    │   ├── node_modules
    │   ├── public
    │   └── views
    ├── index.js
    ├── package-lock.json
    └── employee
        ├── app
        ├── node_modules
        ├── public
        └── views
    └── index.js
    ├── package-lock.json
    └── package.json
    labuser.pem
    README.md

aws

bash - 'ip-10-16-10-55.ec2.us-east-1.compute.amazonaws.com:80'
us-east-1.console.aws.amazon.com/cloud9/ide/1646dd63939f4736b1ca0653d5ada02a?region=us-east-1

File Edit Find Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)
MicroservicesIDE
└── microservices
    ├── customer
    │   ├── app
    │   ├── node_modules
    │   ├── public
    │   └── views
    ├── index.js
    ├── package-lock.json
    └── employee
        ├── app
        ├── node_modules
        ├── public
        └── views
    └── index.js
    ├── package-lock.json
    └── package.json
    labuser.pem
    README.md

aws

create mode 106444 employee/node_modules/validator/lib/validate.js
create mode 106444 employee/node_modules/validator/lib/isloat.js
create mode 106444 employee/node_modules/validator/lib/isolat.js
create mode 106444 employee/node_modules/validator/lib/isint.js
create mode 106444 employee/node_modules/validator/lib/isint32.js
create mode 106444 employee/node_modules/validator/lib/isint64.js
create mode 106444 employee/node_modules/validator/lib/isnull.js
create mode 106444 employee/node_modules/validator/lib/isstring.js
create mode 106444 employee/node_modules/validator/lib/isarray.js
create mode 106444 employee/node_modules/validator/lib/ismerge.js
create mode 106444 employee/node_modules/validator/lib/isnullRegexp.js
create mode 106444 employee/node_modules/validator/lib/isnullToString.js
create mode 106444 employee/node_modules/validator/lib/iswhitelist.js
create mode 106444 employee/node_modules/validator/validator.js
create mode 106444 employee/node_modules/validator/validator.min.js
create mode 106444 employee/node_modules/validator/package.json
create mode 106444 employee/node_modules/validator/README.md
create mode 106444 employee/node_modules/validator/index.js
create mode 106444 employee/node_modules/validator/jshon
create mode 106444 employee/node_modules/validator/yallist.js
create mode 106444 employee/node_modules/validator/yallist/ICRCN3
create mode 106444 employee/node_modules/validator/yallist/README.md
create mode 106444 employee/node_modules/validator/yallist/iterator.js
create mode 106444 employee/node_modules/validator/yallist/queue.js
create mode 106444 employee/node_modules/validator/yallist/yallist.js
create mode 106444 employee/package-lock.json
create mode 106444 employee/public/css/bootstrap.css
create mode 106444 employee/public/css/bootstrap.min.css
create mode 106444 employee/public/css/bootstrap.min.css.map
create mode 106444 employee/public/img/favicon.ico
create mode 106444 employee/public/js/bootstrap.min.js
create mode 106444 employee/public/js/bootstrap.min.js.map
create mode 106444 employee/public/js/jquery-3.6.0.min.js
create mode 106444 employee/views/404.html
create mode 106444 employee/views/500.html
create mode 106444 employee/views/footer.html
create mode 106444 employee/views/header.html
create mode 106444 employee/views/home.html
create mode 106444 employee/views/nav.html
create mode 106444 employee/views/supplier-add.html
create mode 106444 employee/views/supplier-form-fields.html
create mode 106444 employee/views/supplier-list-all.html
create mode 106444 employee/views/supplier-update.html
vclabs:/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
vclabs:/environment/microservices (dev) $ git push -u origin dev

git - 'ip-10-16-10-55.ec2.us-east-1.compute.amazonaws.com:80'
us-east-1.console.aws.amazon.com/cloud9/ide/1646dd63939f4736b1ca0653d5ada02a?region=us-east-1

File Edit Find Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)
MicroservicesIDE
└── microservices
    ├── customer
    │   ├── app
    │   ├── node_modules
    │   ├── public
    │   └── views
    ├── index.js
    ├── package-lock.json
    └── employee
        ├── app
        ├── node_modules
        ├── public
        └── views
    └── index.js
    ├── package-lock.json
    └── package.json
    labuser.pem
    README.md

aws

create mode 106444 employee/node_modules/validator/lib/validate.js
create mode 106444 employee/node_modules/validator/lib/isloat.js
create mode 106444 employee/node_modules/validator/lib/isolat.js
create mode 106444 employee/node_modules/validator/lib/isint.js
create mode 106444 employee/node_modules/validator/lib/isint32.js
create mode 106444 employee/node_modules/validator/lib/isint64.js
create mode 106444 employee/node_modules/validator/lib/isnull.js
create mode 106444 employee/node_modules/validator/lib/isstring.js
create mode 106444 employee/node_modules/validator/lib/isarray.js
create mode 106444 employee/node_modules/validator/lib/ismerge.js
create mode 106444 employee/node_modules/validator/lib/isnullRegexp.js
create mode 106444 employee/node_modules/validator/lib/isnullToString.js
create mode 106444 employee/node_modules/validator/lib/iswhitelist.js
create mode 106444 employee/node_modules/validator/validator.js
create mode 106444 employee/node_modules/validator/validator.min.js
create mode 106444 employee/node_modules/validator/package.json
create mode 106444 employee/node_modules/validator/README.md
create mode 106444 employee/node_modules/validator/index.js
create mode 106444 employee/node_modules/validator/jshon
create mode 106444 employee/node_modules/validator/yallist.js
create mode 106444 employee/node_modules/validator/yallist/ICRCN3
create mode 106444 employee/node_modules/validator/yallist/README.md
create mode 106444 employee/node_modules/validator/yallist/iterator.js
create mode 106444 employee/node_modules/validator/yallist/queue.js
create mode 106444 employee/node_modules/validator/yallist/yallist.js
create mode 106444 employee/package-lock.json
create mode 106444 employee/public/css/bootstrap.css
create mode 106444 employee/public/css/bootstrap.min.css
create mode 106444 employee/public/img/espresso.jpg
create mode 106444 employee/public/js/bootstrap.min.js
create mode 106444 employee/public/js/bootstrap.min.js.map
create mode 106444 employee/public/js/jquery-3.6.0.min.js
create mode 106444 employee/public/js/jquery-3.6.0.min.js.map
create mode 106444 employee/views/500.html
create mode 106444 employee/views/footer.html
create mode 106444 employee/views/header.html
create mode 106444 employee/views/nav.html
create mode 106444 employee/views/supplier-add.html
create mode 106444 employee/views/supplier-form-fields.html
create mode 106444 employee/views/supplier-list-all.html
create mode 106444 employee/views/supplier-update.html
vclabs:/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
vclabs:/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 1064 (2191/2191), done.
Counting objects: 1064 (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (532/532), done.
Writing objects: 100% (2191/2191), 1.94 MB | 3.78 MB/s, done.
Total 2191 (delta 546), reused 0 (delta 0), pack-reused 0
Updating objects: 100% (2191/2191), done.
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
 * [new branch] dev -> dev
branch 'dev' set up to track 'origin/dev'.
vclabs:/environment/microservices (dev) $ 

```

- Configure git client with username and email address.

```
bash -t ip-10-16-10-55.ec2.us-east-1.compute.internal ~
```

```
create mode 100644 employee/node_modules/validator/package.json
create mode 100644 employee/node_modules/validator/validator.js
create mode 100644 employee/node_modules/validator/validator.min.js
create mode 100644 employee/node_modules/validator/HISTORY.md
create mode 100644 employee/node_modules/vary/LICENSE
create mode 100644 employee/node_modules/vary/README.md
create mode 100644 employee/node_modules/vary/index.js
create mode 100644 employee/node_modules/vary/package.json
create mode 100644 employee/node_modules/vaillist/LICENSE
create mode 100644 employee/node_modules/vaillist/README.md
create mode 100644 employee/node_modules/vaillist/iterator.js
create mode 100644 employee/node_modules/vaillist/parser.json
create mode 100644 employee/node_modules/vaillist/vaillist.js
create mode 100644 employee/package-lock.json
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.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/bootstrap.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/fontawesome-free-5.6.0.min.js
create mode 100644 employee/public/js/fontawesome-free.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-create-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
vscodeshell:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/vi/repos/microservices
vscodeshell:~/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% (1978/1978), done.
Writing objects: 100% (2191/2191), 1.94 MB | 3.78 MB/s, done.
Total 2191 (delta 546), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/vi/repos/microservices
 * [new branch]    dev -> dev
branch 'dev' set up to track 'origin/dev'.
vscodeshell:~/environment/microservices (dev) $ git config --global user.name "sheeta"
vscodeshell:~/environment/microservices (dev) $ git config --global user.email "sheeta@gmail.com"
vscodeshell:~/environment/microservices (dev) $
```

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

The screenshot shows the AWS CodeCommit interface. The left sidebar navigation includes 'Developer Tools', 'CodeCommit' (selected), 'Source', 'CodeCommit' (under Source), 'Getting started', 'Repositories', 'Code', 'Pull requests', 'Commits' (selected), 'Branches', 'Git tags', 'Settings', 'Approval rule templates', 'Artifacts', 'Build', 'Deploy', 'Pipeline', and 'Settings'. A search bar at the top has the placeholder 'Search' and an 'Alt+S' keyboard shortcut. The main content area shows the 'microservices' repository's commit history. The breadcrumb navigation is 'Developer Tools > CodeCommit > Repositories > microservices > Commits'. The commit list table has columns: Commit ID, Commit message, Commit date, Authored date, Author, Committer, and Actions. One commit is listed: '89284f48 two unmodified copies of the application code' (Authored and Committed 1 minute ago by 'EC2 Default User'). There are buttons for 'Copy ID' and 'Browse'. The top right corner shows 'N. Virginia' and a user profile for 'vocabs/user3247400+Madhale_Shweta @ 3853-9794-3428'.

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

Task 4.1: Adjust the AWS Cloud9 instance security group settings

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

Instance details | EC2 | us-east-1 | Repositories | AWS Developer | SecurityGroup | EC2 | us-east-1 | MicroservicesIDE - AWS Cloud9

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroup:groupId=sg-0d358bfd406718ec4

N. Virginia vocabs/user3247400+Madhale_Shweta @ 5853-9794-3428

EC2 Dashboard EC2 Global View Events Console-to-Code [Preview](#)

Instances Instances Instance Types Launch Templates Spot Requests Savings Plans Reserved Instances Dedicated Hosts Capacity Reservations [New](#)

Images AMIs AMI Catalog

Elastic Block Store Volumes Snapshots Lifecycle Manager

Network & Security Security Groups Elastic IPs

CloudShell Feedback

sg-0d358bfd406718ec4 - aws-cloud9-MicroservicesIDE-1646dd63939f4736b1ca0653d5ada02a-InstanceSecurityGroup-uUCKnjFXKmtA

Actions

Details

Security group name aws-cloud9-MicroservicesIDE-1646dd63939f4736b1ca0653d5ada02a-InstanceSecurityGroup-uUCKnjFXKmtA	Security group ID sg-0d358bfd406718ec4	Description Security group for AWS Cloud9 environment aws-cloud9-MicroservicesIDE-1646dd63939f4736b1ca0653d5ada02a
Owner 385397943428	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry

Inbound rules Outbound rules Tags

Inbound rules (2)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-000b4db013982f44f	IPv4	SSH	TCP	22	35.172.155.96/27	-
-	sgr-0667a0be4d2015...	IPv4	SSH	TCP	22	35.172.155.192/27	-

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

This screenshot shows the AWS EC2 Security Group details page for a specific security group. It displays basic information like the security group name, ID, owner, and rule counts. Below this, the 'Inbound rules' tab is selected, showing two existing rules for SSH traffic on port 22 from specific IP ranges. The interface includes standard AWS navigation and search features.

Instance details | EC2 | us-east-1 | Repositories | AWS Developer | ModifyInboundSecurityGroupRules | MicroservicesIDE - AWS Cloud9

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-0d358bfd406718ec4

N. Virginia vocabs/user3247400+Madhale_Shweta @ 3853-9794-3428

CloudShell Feedback

Edit inbound rules [Info](#)

Inbound rules [Info](#)

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-000b4db013982f44f	SSH	TCP	22	Custom	35.172.155.96/27
sgr-0667a0be4d2015993	SSH	TCP	22	Custom	35.172.155.192/27
-	Custom TCP	TCP	8080	Anywhere...	0.0.0.0/0
-	Custom TCP	TCP	8081	Anywhere...	0.0.0.0/0

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

This screenshot shows the 'Edit inbound rules' page for a security group. It lists the current rules and allows for modification. A warning message at the bottom advises against using 0.0.0.0/0 or ::/0 as sources. The interface includes a 'Save rules' button and standard AWS navigation elements.

The screenshot shows the AWS Cloud9 Security Group Details page. The security group name is "aws-cloud9-MicroservicesIDE-1646dd639394736b1ca0653d5ada02a-InstanceSecurityGroup-uUCKnJFXXmtA". It has a security group ID of "sg-0d358bfd406718ec4", an owner of "385397943428", and 4 permission entries. The VPC ID is "vpc-0a9f6197396de75cc". The Inbound rules section lists four rules:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0af9c32cd1a97636	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	-
-	sgr-000b4db013982f144	IPv4	SSH	TCP	22	35.172.155.96/27	-
-	sgr-0667a0be4d2015...	IPv4	SSH	TCP	22	35.172.155.192/27	-
-	sgr-0e7077dc47d789a...	IPv4	Custom TCP	TCP	8081	0.0.0.0/0	-

Task 4.2: Modify the source code of the customer microservice

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

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are three tabs: "Instance details | EC2 | us-east-", "Repositories | AWS Developer Tools", and "SecurityGroup | EC2 | us-east-1". Below the tabs, the URL is "us-east-1.console.aws.amazon.com/cloud9/ide/1646dd63939f4736b1ca0653d5ada02a?region=us-east-1". The main area has a navigation bar with File, Edit, Find, View, Go, Run, Tools, Window, Support, Preview, and Run buttons. On the left, a file tree shows the project structure: "MicroservicesIDE", "microservices", "customer" (selected), "app", "node_modules", "public", "views", "index.js", "package-lock.json", "package.json", "employee", "labsuser.pem", and "README.md". The right side contains a code editor with the file "supplier.controller.js" open. The code is as follows:

```
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.findOne = (req, res) => {
    Supplier.findById(req.params.id, (err, data) => {
        if (err) {
            if (err.kind === "not_found") {
                res.status(404).send({
                    message: `Not found Supplier with id ${req.params.id}.`
                });
            } else {
                res.render("500", {message: `Error retrieving Supplier with id ${req.params.id}`});
            }
        } else res.render("supplier-update", {supplier: data});
    });
};
```

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

The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes tabs for 'Instance details | EC2 | us-east-1', 'Repositories | AWS Developer Tools', 'SecurityGroup | EC2 | us-east-1', and 'Microservices'. Below the tabs, the URL is 'us-east-1.console.aws.amazon.com/cloud9/ide/1646dd63939f4736b1ca0653d5ada02a?region=us-east-1'. The main area has a dark theme with a file tree on the left and two code editors on the right.

File Tree:

```

└─ MicroservicesIDE
    └─ microservices
        └─ customer
            └─ app
                └─ node_modules
                    └─ models
                        └─ supplier.model.js
                └─ public
                └─ views
                    └─ index.js
                └─ package-lock.json
                └─ package.json
            └─ employee
                └─ labuser.pem
            └─ README.md

```

Code Editor (supplier.controller.js):

```

1  const mysql = require("mysql2");
2  const dbConfig = require("../config/config");
3
4  const Supplier = function (supplier) {
5      this.id = supplier.id;
6      this.name = supplier.name;
7      this.address = supplier.address;
8      this.city = supplier.city;
9      this.state = supplier.state;
10     this.email = supplier.email;
11     this.phone = supplier.phone;
12 };
13
14 const db_connection = mysql.createPool({
15     host: dbConfig.APP_DB_HOST,
16     user: dbConfig.APP_DB_USER,
17     password: dbConfig.APP_DB_PASSWORD,
18     database: dbConfig.APP_DB_NAME
19 });
20
21 Supplier.getAll = result => {
22     db_connection.query("SELECT * FROM suppliers", (err, res) => {
23         if (err) {
24             console.log("error: ", err);
25             result(err, null);
26             return;
27         }
28         console.log("suppliers: ", res);
29         result(null, res);
30     });
31 }
32
33
34 Supplier.findById = (supplierId, result) => {
35     db_connection.query(' SELECT * FROM suppliers WHERE id = ?', [supplierId], (err, res) => {
36         if (err) {
37             console.log("error: ", err);
38             result(err, null);
39             return;
40         }
41         if (res.length) {
42             console.log("found supplier: ", res[0]);
43             result(null, res[0]);
44             return;
45         }
46         result({kind: "not_found"}, null);
47     });
48 }
49
50 module.exports = Supplier;
51

```

- Later in the project, when deploying the microservices behind an Application Load Balancer, the employees need to be able to navigate from the main customer page to the area of the web application where they can add, edit, or delete supplier entries. To support this, editing the **customer/views/nav.html** file:

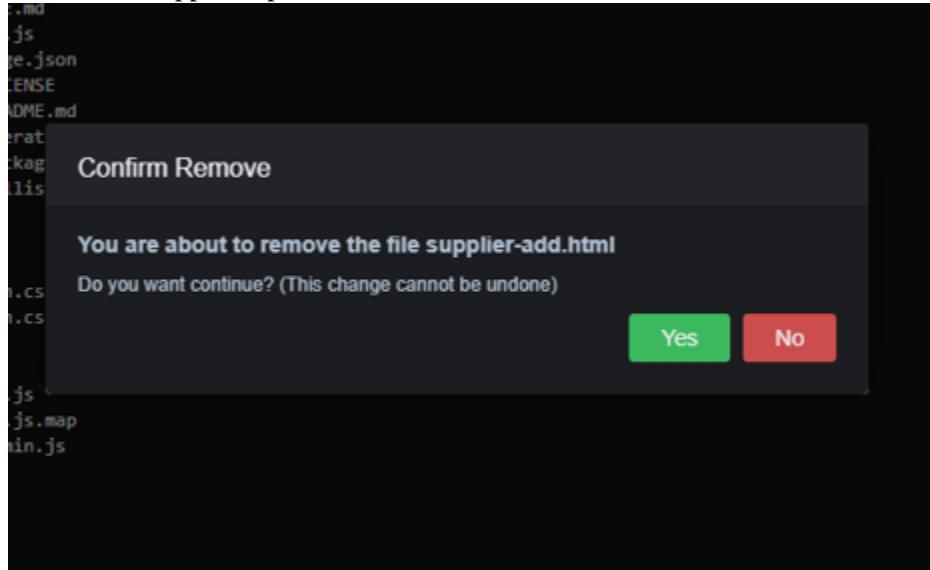
On line 3, change Monolithic Coffee suppliers to Coffee suppliers. On line 7, change Home to Customer home. Add a new line after line 8 that contains the following HTML: Administrator link

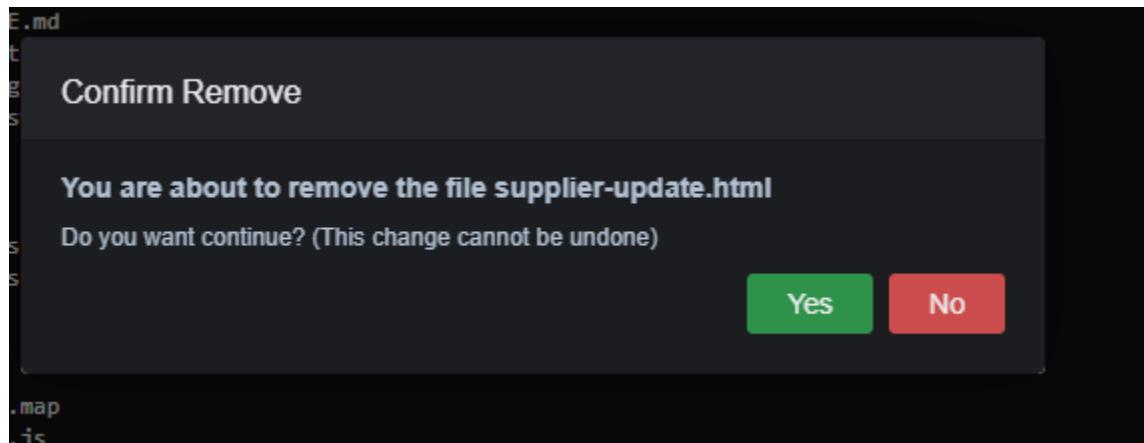
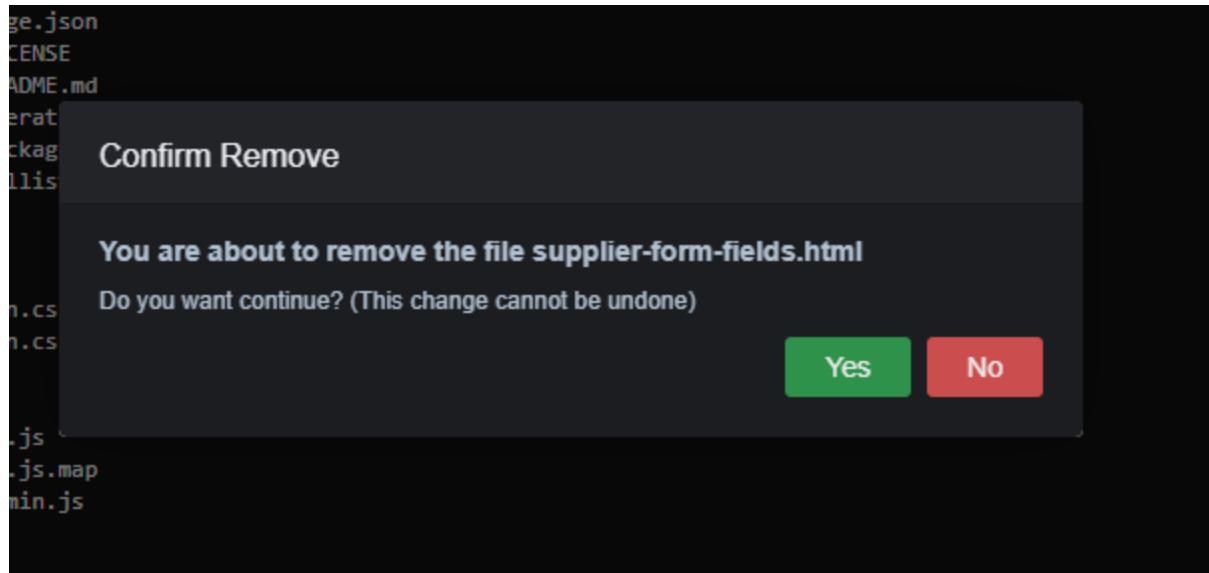
```
customer/views/nav.html
bash - tlp-10-16-10-55.ec2 ~ JS supplier.controller.js x JS supplier.model.js . nav.html x +
```

```
customer
  └── views
    └── nav.html
```

```
1 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
2   
3   <div><a class="navbar-brand page-title" 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</a>
8       <a class="nav-link" href="/suppliers">Suppliers list</a>
9       <a class="nav-link" href="/admin/suppliers">Administrator link</a>
10      </li>
11    </ul>
12  </div>
13 </nav>
```

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





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

The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays a file tree for a project named 'MicroservicesIDE'. The 'customer' directory contains several files: index.js, package-lock.json, package.json, employee.js, labsuser.pem, and README.md. The right pane shows the content of the 'index.js' file.

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

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

// list all the suppliers
app.get("/", (req, res) => {
    res.render("home", {});
});

app.get("/suppliers/", supplier.findAll);
// show the add supplier form
// app.get("/supplier-add", (req, res) => {
//     res.render("supplier-add", {});
// });

// receive the add supplier POST
// app.post("/supplier-add", supplier.create);
// show the update Form
// app.get("/supplier-update/:id", supplier.findOne);
// receive the update POST
// app.post("/supplier-update", supplier.update);
// receive the POST to delete a supplier
// app.post("/supplier-remove/:id", supplier.remove);
// handle 404
app.use(function (req, res, next) {
    res.status(404).render("404", {});
});

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

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

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

```

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

```

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are tabs for 'Instance details | EC2 | us-east-1', 'Repositories | AWS Developer Tools', and 'Security Groups'. Below the tabs, the URL 'us-east-1.console.aws.amazon.com/cloud9/ide/1646dd63939f4736b' is displayed. The main area has a dark theme with a navigation bar at the top containing 'File', 'Edit', 'Find', 'View', 'Go', 'Run', 'Tools', 'Window', 'Support', 'Preview', and 'Run' buttons. On the left, a file tree shows a directory structure under 'customer/index.js': 'MicroservicesIDE - /home/ec2-user/microservices/customer/app/node_modules/public/views'. Inside 'customer', there are files: 'Dockerfile', 'index.js', 'package-lock.json', 'package.json', 'employee', 'labsuser.pem', and 'README.md'. The 'Dockerfile' is selected and shown in the central code editor window.

```

1 FROM node:11-alpine
2 RUN mkdir -p /usr/src/app
3 WORKDIR /usr/src/app
4 COPY . .
5 RUN npm install
6 EXPOSE 8080
7 CMD ["npm", "run", "start"]

```

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

```

customer/index.js
└─ MicroservicesIDE - /home/ec2-user
    └─ microservices
        └─ customer
            └─ app
                └─ node_modules
            └─ public
            └─ views
            └─ Dockerfile
            └─ index.js
            └─ package-lock.json
            └─ package.json
        └─ employee
    └─ labsuser.pem
    └─ README.md

bash - "ip-10-16-10-55.ec2.us-east-1.compute.amazonaws.com" bash - "ip-10-16-10-55.ec2.us-east-1.compute.amazonaws.com"
voclabs:~/environment/microservices/customer (dev) $ docker build --tag customer .
[+] Building 7.0s (10/10) FINISHED
--> [internal] load build definition from Dockerfile
--> => transferring dockerfile: 229B
--> [internal] load metadata for docker.io/library/node:11-alpine
--> [internal] load .dockerignore
--> [internal] transferring context: 2B
--> [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56ab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932
--> => resolve docker.io/library/node:11-alpine@sha256:8bb56ab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932
--> => sha256:f18daaf58c3dabd7086c02f6a76719ba0c78673aa83d4bfbc8f1e5cb0000096a 5.66kB / 5.66kB
--> => sha256:e7c96db7181be991f19a9fb6975cd8bd73c85f4a2681348e63a141a2192a5f10 2.76kB / 2.76kB
--> => sha256:8119aca4464934fd401213631cf0537464c07c98d9990580587b14aaadd 21.66MB / 21.66MB
--> => sha256:40df19605a18a2dc4e3cc0cb508a93fd2f7615e2a92eccd743698c24c23fe84 1.33MB / 1.33MB
--> => sha256:8bb56ab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932 1.42kB / 1.42kB
--> => sha256:914ff7cc1454e019a19c080a9e42b5763c826194110e8e902c8e92845799fb6 1.16kB / 1.16kB
--> => extracting sha256:e7c96db7181be991f19a9fb6975cd8bd73c85f4a2681348e63a141a2192a5f10
--> => sha256:821940804a64b286671a46403fe19802aa87c4e41c4551fd0d295ee7ea73 279B / 279B
--> => extracting sha256:8119aca4464934fd401213631cf0537464c07c799df998580587b14aaadd
--> => extracting sha256:40df19605a18a2dc4e3cc0cb508a93fd2f7615e2a92eccd743698c24c23fe84
--> => extracting sha256:82194bb04a4b4bd286671a1464d3fe319832aaa67c4e41c455fdff2d295ae7aa73
--> [internal] load build context
--> [internal] transferring context: 7.50kB
--> [2/5] RUN mkdir -p /usr/src/app
--> [3/5] WORKDIR /usr/src/app
--> [4/5] COPY . .
--> [5/5] RUN npm install
--> => exporting to image
--> => exporting layers
--> => writing image sha256:2c0fc5a9f04830df5ddab38adelb1002bf38eeb2034e29adfa9b21b0145576af
--> => naming to docker.io/library/customer

```

- Verify that the customer-labeled Docker image was created

```

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

```

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

```

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

```

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

```

voclabs:~/environment/microservices/customer (dev) $ docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint" customer
93745d4a452f193688f5502e5251a8c0bf5ab9736ae26cf92724174c990b734
voclabs:~/environment/microservices/customer (dev) $

```

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

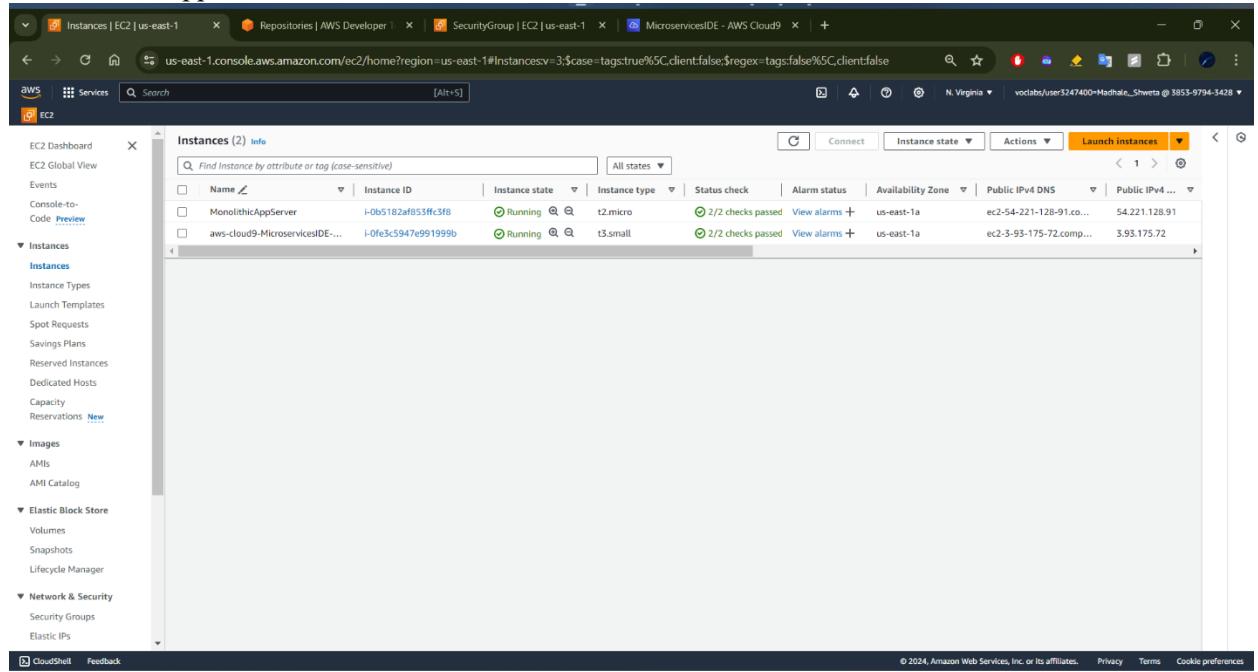
```

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

```

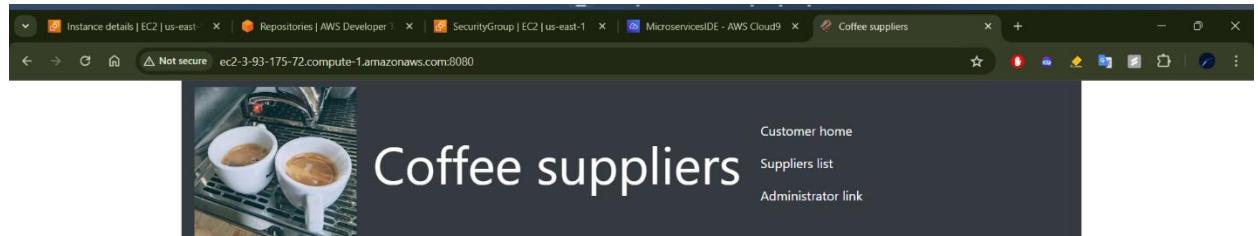
- Verifying that customer microservice is running the container and is working as intended.
Loading new browser tab with <http://<cloud-9-public-IPv4-address>:8080>. Selecting the “List of

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



The screenshot shows the AWS EC2 Instances page. The left sidebar has sections like EC2 Dashboard, Instances (selected), Images, Elastic Block Store, Network & Security, and CloudShell. The main area shows a table titled "Instances (2) Info" with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public IPv4 IP. Two instances are listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP
MonolithicAppServer	i-0b5182af853ffcf8	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-54-221-128-91.co...	54.221.128.91
aws-cloud9-MicroservicesIDE-...	i-0fe3c5947e991999b	Running	t3.small	2/2 checks passed	View alarms	us-east-1a	ec2-3-93-175-72.com...	3.93.175.72



The screenshot shows the "Coffee suppliers" application. The top navigation bar includes links for Instance details, SecurityGroup, MicroservicesIDE - AWS Cloud9, and Coffee suppliers. The page title is "Coffee suppliers". On the left, there's a small image of two cups of coffee on a espresso machine. The main content area has a dark background with white text. It says "Welcome" and "Use this app to keep track of your coffee suppliers". Below that is a link "List of suppliers". On the right side, there are links for "Customer home", "Suppliers list", and "Administrator link".

Name	Address	City	State	Email	Phone
Shweta	100 Columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

- Commit and push the code changes to CodeCommit.

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

The screenshot shows the AWS CodeCommit interface. The left sidebar has a 'CodeCommit' section with 'Commits' selected. The main area shows a list of commits for the 'microservices' repository. There are two commits listed:

Commit ID	Commit message	Commit date	Authored date	Author	Committer	Actions
53972993	Changed source code of customer microservice	Just now	Just now	shweta	shweta	Copy ID Browse
89284f48	two unmodified copies of the application code	17 minutes ago	17 minutes ago	EC2 Default User	EC2 Default User	Copy ID Browse

Task 4.4: Modify the source code of the employee microservice

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

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

```
 1  const Supplier = require("../models/supplier.model.js");
 2  const {body, validationResult} = require("express-validator");
 3
 4  exports.create = [
 5      // Validate and sanitize the name field,
 6      body('name', 'The supplier name is required').trim().isLength({min: 1}).escape(),
 7      body('address', 'The supplier address is required').trim().isLength({min: 1}).escape(),
 8      body('city', 'The supplier city is required').trim().isLength({min: 1}).escape(),
 9      body('state', 'The supplier state is required').trim().isLength({min: 1}).escape(),
10      body('phone', 'Phone number should be 10 digit number plus optional country code').trim().isMobilePhone().escape(),
11      // Process request after validation and sanitization.
12      (req, res, next) => {
13          // Extract the validation errors from a request.
14          const errors = validationResult(req);
15          // Create a genre object with escaped and trimmed data.
16          const supplier = new Supplier(req.body);
17          if (!errors.isEmpty()) {
18              // There are errors. Render the form again with sanitized values/error messages,
19              res.render('supplier-add', {title: 'Create Supplier', supplier, errors: errors.array()});
20          } else {
21              // Data from form is valid., save to db
22              Supplier.create(supplier, (err, data) => {
23                  if (err)
24                      res.render("500", {message: "Error occurred while creating the Supplier."});
25                  else res.redirect("/admin/suppliers");
26              });
27          }
28      }
29  ];
30
31  exports.findAll = (req, res) => {
32      Supplier.getAll((err, data) => {
33          if (err)
34              res.render("500", {message: "There was a problem retrieving the list of suppliers"});
35          else res.render("supplier-list-all", {suppliers: data});
36      });
37  };
38
39  exports.findOne = (req, res) => {
40      Supplier.findById(req.params.id, (err, data) => {
41          if (err) {
42              if (err.kind === "not_found") {
43                  res.status(404).send({
44                      message: "Not found Supplier with id ${req.params.id}."
45                  });
46              } else {
47                  res.render("500", {message: "Error retrieving Supplier with id ${req.params.id}"});
48              }
49          } else res.render("supplier-update", {supplier: data});
50      });
51  };
52
53
54  exports.update = [
55      // Validate and sanitize the name field,
56      body('name', 'The supplier name is required').trim().isLength({min: 1}).escape(),
57      body('address', 'The supplier address is required').trim().isLength({min: 1}).escape(),
58      // Process request after validation and sanitization.
59      (req, res, next) => {
60          // Extract the validation errors from a request.
61          const errors = validationResult(req);
62          // Create a genre object with escaped and trimmed data.
63          const supplier = new Supplier(req.body);
64          if (!errors.isEmpty()) {
65              // There are errors. Render the form again with sanitized values/error messages,
66              res.render('supplier-update', {title: 'Update Supplier', supplier, errors: errors.array()});
67          } else {
68              Supplier.findByIdAndUpdate(req.params.id, supplier, (err, data) => {
69                  if (err)
70                      res.render("500", {message: "Error occurred while updating the Supplier."});
71                  else res.redirect("/admin/suppliers");
72              });
73          }
74      }
75  ];
76
```

```

  63     (req, res, next) => {
  64       // Extract the validation errors from a request.
  65       const errors = validationResult(req);
  66       // Create a genre object with escaped and trimmed data.
  67       const supplier = new Supplier(req.body);
  68       supplier.i
  69       if (!errors.isEmpty()) {
  70         // There are errors. Render the form again with sanitized values/error messages.
  71         res.render('supplier-update', {supplier: supplier, errors: errors.array()});
  72       } else {
  73         // Data from form is valid., save to db
  74         Supplier.updateById(
  75           req.body.id,
  76           supplier,
  77           (err, data) => {
  78             if (err) {
  79               if (err.kind === "not_found") {
  80                 res.status(404).send({
  81                   message: `Supplier with id ${req.body.id} Not found.`);
  82                 });
  83               } else {
  84                 res.render("500", {message: `Error updating Supplier with id ${req.body.id}`});
  85               }
  86             } else res.redirect("/admin/suppliers");
  87           );
  88         );
  89       }
  90     ];
  91   };
  92 
  93   exports.remove = (req, res) => {
  94     Supplier.delete(req.params.id, (err, data) => {
  95       if (err) {
  96         if (err.kind === "not_found") {
  97           res.status(404).send({
  98             message: `Not found Supplier with id ${req.params.id}.`);
  99           });
  100         } else {
  101           res.render("500", {message: `Could not delete Supplier with id ${req.body.id}`});
  102         }
  103       } else res.redirect("/admin/suppliers");
  104     );
  105   };
  106 
  107   exports.removeAll = (req, res) => {
  108     Supplier.removeAll((err, data) => {
  109       if (err)
  110         res.render("500", {message: `Some error occurred while removing all suppliers.`});
  111       else res.send({message: `All Suppliers were deleted successfully!`});
  112     );
  113   };

```

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

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

Set the port number to 8081,

```
≡ bash - "ip-10-16-10-55.ec" x Dockerfile x bash - "ip-10-16-10-55.ec"
1 const express = require("express");
2 const bodyParser = require("body-parser");
3 const cors = require("cors")
4 const supplier = require("./app/controller/supplier.controller");
5 const app = express();
6 const mustacheExpress = require("mustache-express")
7 const favicon = require('serve-favicon');
8
9 // parse requests of content-type: application/json
10 app.use(bodyParser.json());
11 // parse requests of content-type: application/x-www-form-urlencoded
12 app.use(bodyParser.urlencoded({extended: true}));
13 app.use(cors());
14 app.options("", cors());
15 app.engine("html", mustacheExpress())
16 app.set("view engine", "html")
17 app.set("views", __dirname + "/views")
18 app.use(express.static('public'));
19 app.use(favicon(__dirname + "/public/img/favicon.ico"));
20
21 // list all the suppliers
22 app.get("/admin", (req, res) => {
23   res.render("home", {});
24 });
25 app.get("/admin/suppliers/", supplier.findAll);
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", supplier.update);
36 // receive the POST to delete a supplier
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 || 80
45 app.listen(app_port, () => {
46   console.log(`Server is running on port ${app_port}.`);
47 });


```

```
 1  const express = require("express");
 2  const bodyParser = require("body-parser");
 3  const cors = require("cors")
 4  const supplier = require("./app/controller/supplier.controller");
 5  const app = express();
 6  const mustacheExpress = require("mustache-express")
 7  const favicon = require('serve-favicon');
 8
 9  // parse requests of content-type: application/json
10  app.use(bodyParser.json());
11  // parse requests of content-type: application/x-www-form-urlencoded
12  app.use(bodyParser.urlencoded({extended: true}));
13  app.use(cors());
14  app.options("*", cors());
15  app.engine("html", mustacheExpress())
16  app.set("view engine", "html")
17  app.set("views", __dirname + "/views")
18  app.use(express.static('public'));
19  app.use(favicon(__dirname + "/public/img/favicon.ico"));
20
21  // list all the suppliers
22  app.get("/admin", (req, res) => {
23    res.render("home", {});
24  });
25  app.get("/admin/suppliers/", supplier.findAll);
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", supplier.update);
36  // receive the POST to delete a supplier
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
44  // set port, listen for requests
45  const app_port = process.env.APP_PORT || 8081
46  app.listen(app_port, () => {
47    console.log(`Server is running on port ${app_port}.`);
48  });

```

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

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

```

1  {{>header}}
2  <div class="container">
3    {{>nav}}
4    <div class="mt-2">
5      {{#errors}}
6        <div class="alert alert-warning" role="alert">
7          {{msg}}
8        </div>
9      {{/errors}}
10     </div>
11     <form action="/admin/supplier-add" method="POST">
12       {{>supplier-form-fields}}
13       <button type="submit" class="btn btn-primary">Submit</button>
14     </form>
15
16   </div>
17 {{>footer}}
18

```

```

1  {{>header}}
2  <div class="container">
3    {{>nav}}
4    <div class="mt-2">
5      {{#errors}}
6        <div class="alert alert-warning" role="alert">
7          {{msg}}
8        </div>
9      {{/errors}}
10     </div>
11     {{#supplier}}
12       <form action="/admin/supplier-update" method="POST">
13         <input type="hidden" id="id" name="id" value="{{id}}>
14         {{>supplier-form-fields}}
15         <button type="submit" class="btn btn-primary">Submit</button>
16       </form>
17
18      <!-- Button trigger modal -->
19      <button type="button" class="float-right btn btn-danger" data-toggle="modal" data-target="#exampleModal">
20        Delete this supplier
21      </button>
22
23      <!-- Model -->
24      <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
25        <div class="modal-dialog" role="document">
26          <div class="modal-content">
27            <div class="modal-header">
28              <h5 class="modal-title" id="exampleModalLabel">Delete supplier</h5>
29              <button type="button" class="close" data-dismiss="modal" aria-label="Close">
30                <span aria-hidden="true">&times;</span>
31              </button>
32            </div>
33            <div class="modal-body">
34              Are you sure you want to delete supplier {{name}}?
35            </div>
36            <div class="modal-footer">
37              <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
38              <form action="/admin/supplier-remove/{{id}}" method="POST">
39                <button type="submit" class="float-right btn btn-danger">Delete this supplier</button>
40              </form>
41            </div>
42          </div>
43        </div>
44      </div>
45
46    {{/supplier}}
47
48  </div>
49 {{>footer}}
50

```

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

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

```

1  {{>header}}
2  <div class="container">
3    {{>nav}}
4    <h1>All suppliers</h1>
5    <table class="table table-hover">
6      <thead>
7        <tr>
8          <th scope="col">Name</th>
9          <th scope="col">Address</th>
10         <th scope="col">City</th>
11         <th scope="col">State</th>
12         <th scope="col">Email</th>
13         <th scope="col">Phone</th>
14         <th scope="col"></th>
15       </tr>
16     </thead>
17     <tbody>
18       {{#suppliers}}
19         <tr>
20           <th scope="row">{{name}}</th>
21           <td>{{address}}</td>
22           <td>{{city}}</td>
23           <td>{{state}}</td>
24           <td>{{email}}</td>
25           <td>{{phone}}</td>
26           <td><h4><span class="badge badge-info"><a class="text-light" href="/admin/supplier-update/{{id}}">edit</a></span></h4></td>
27         </tr>
28       {{/suppliers}}
29     </tbody>
30   </table>
31   <h4><a class="badge badge-success" href="/admin/supplier-add">Add a new supplier</a></h4>
32
33
34 </div>
35 {{>footer}}
36

```

```

1  {{>header}}
2  <div class="container">
3    {{>nav}}
4    <div class="container">
5      <h1>Welcome</h1>
6      <p>Use this app to keep track of your coffee suppliers</p>
7      <p><a href="/admin/suppliers">List of suppliers</a></p>
8    </div>
9  </div>
10 {{>footer}}
11

```

- In the employee/views/header.html file, modify the title to be `Manage coffee suppliers`

```

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

```

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

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

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

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

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

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

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

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

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

```
vocabs:~/environment/microservices/employee (dev) $ grep -n 'redirect' app/controller/supplier.controller.js
25:           else res.redirect('/suppliers');
86:             } else res.redirect('/suppliers');
103:       } else res.redirect('/suppliers');
vocabs:~/environment/microservices/employee (dev) $ grep -n 'app.get|app.post' index.js
22:app.get("/", (req, res) => {
25:app.get("/suppliers/", supplier.findAll);
27:app.get("/supplier-add", (req, res) => {
31:app.post("/supplier-add", supplier.create);
33:app.get("/supplier-update/:id", supplier.findOne);
35:app.post("/supplier-update", supplier.update);
37:app.post("/supplier-remove/:id", supplier.remove);
vocabs:~/environment/microservices/employee (dev) $ grep -n 'action' views/*
views/supplier-add.html:11:   <form action="/supplier-add" method="POST">
views/supplier-update.html:12:   <form action="/supplier-update" method="POST">
views/supplier-update.html:38:                               <form action="/supplier-remove/{:id}" method="POST">
vocabs:~/environment/microservices/employee (dev) $ grep -n 'href' views/supplier-list-all.html views/home.html
views/supplier-list-all.html:27:   href="/supplier-update/{:id}"/>edit</a></span></h4></td>
views/supplier-list-all.html:32:   <h4><a class="badge badge-success" href="/supplier-add">Add a new supplier</a></h4>
views/home.html:7:   <p><a href="/suppliers">List of suppliers</a></p>
```

```

voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
[+] Building 0.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 229B
=> [internal] load metadata for docker.io/library/node:11-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6045a4d505932
=> [internal] load build context
=> => transferring context: 263.85kB
=> CACHED [2/5] RUN mkdir -p /usr/src/app
=> CACHED [3/5] WORKDIR /usr/src/app
=> CACHED [4/5] COPY . .
=> CACHED [5/5] RUN npm install
=> exporting to image
=> => exporting layers
=> => writing image sha256:efab4e40fe60dc6c8f507844ab32787f3f49f17364fce078f36ac741bd1ef315
=> => naming to docker.io/library/employee
voclabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint
supplierdb.ccaycjdtups.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/employee (dev) $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee
71021a663be33dfda883780df2b55e4d7d6b8af9751f4ae9e79c65eab1357121
voclabs:~/environment/microservices/employee (dev) $ 

```

Administrator home
Suppliers list
Customer home

All suppliers

Name	Address	City	State	Email	Phone
Shweta	100 Columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

Add a new supplier

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

Not secure ec2-3-93-175-72.compute-1.amazonaws.com:8081/admin/supplier-add

Name

Name of this supplier

Address

Address for this supplier

City

City for this supplier

State

State for this supplier

Email

Email for this supplier

Phone

Phone number for this supplier

Not secure ec2-3-93-175-72.compute-1.amazonaws.com:8081/admin/supplier-update/2

Name

Name of this supplier

Address

Address for this supplier

City

City for this supplier

State

State for this supplier

Email

Email for this supplier

Phone

Phone number for this supplier

Administrator home
Suppliers list
Customer home

Manage coffee suppliers

All suppliers

Name	Address	City	State	Email	Phone
Shweta	100 Columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708
shweta madhale	columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

[Add a new supplier](#)

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

Name
Shweta
Name of this supplier

Address
100 Columbia ave
Address for this supplier

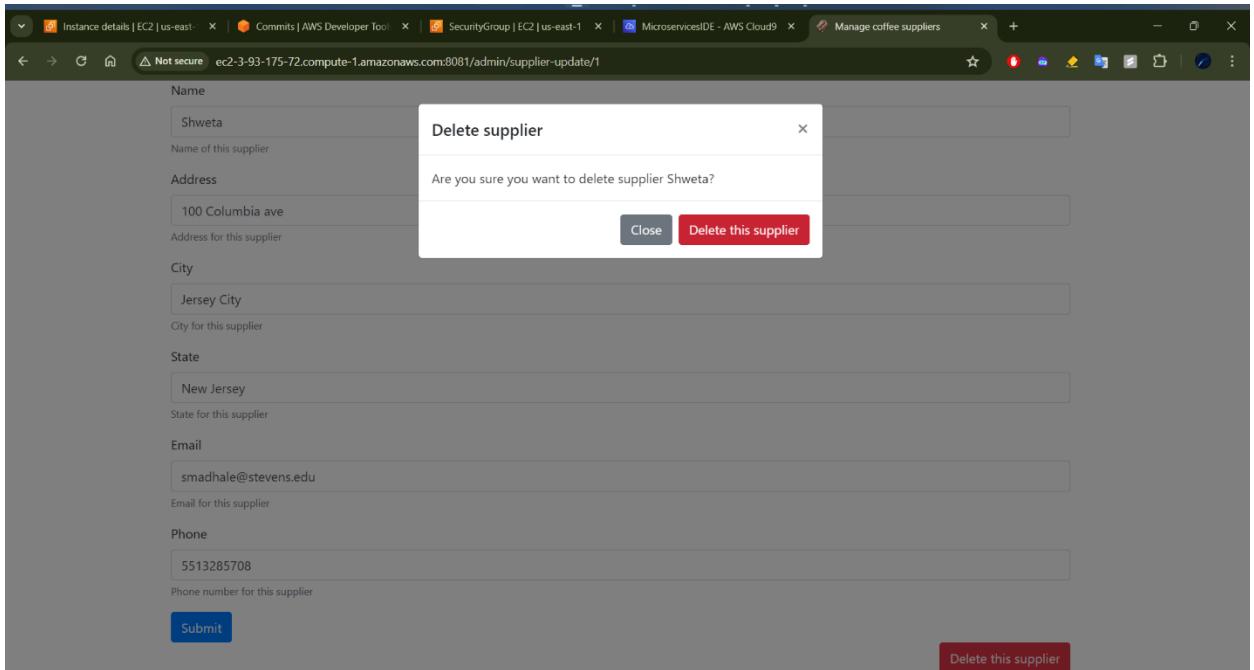
City
Jersey City
City for this supplier

State
New Jersey
State for this supplier

Email
smadhale@stevens.edu
Email for this supplier

Phone
5513285708
Phone number for this supplier

[Submit](#) [Delete this supplier](#)



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

```
docker ps
```

```
vocabs:~/environment/microservices/employee (dev) $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS                         NAMES
71021a663be3        employee            "docker-entrypoint.s_"
93745d4a452f        customer           "docker-entrypoint.s_"
vocabs:~/environment/microservices/employee (dev) $
```

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

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

```

1  const express = require("express");
2  const bodyParser = require("body-parser");
3  const cors = require("cors")
4  const supplier = require("./app/controller/supplier.controller");
5  const app = express();
6  const mustacheExpress = require("mustache-express")
7  const favicon = require('serve-favicon');
8
9  // parse requests of content-type: application/json
10 app.use(bodyParser.json());
11 // parse requests of content-type: application/x-www-form-urlencoded
12 app.use(bodyParser.urlencoded({extended: true}));
13 app.use(cors());
14 app.options("*", cors());
15 app.engine("html", mustacheExpress())
16 app.set("view engine", "html")
17 app.set("views", __dirname + "/views")
18 app.use(express.static('public'));
19 app.use(favicon(__dirname + "/public/img/favicon.ico"));
20
21 // list all the suppliers
22 app.get("/admin", (req, res) => {
23   res.render("home", {});
24 });
25 app.get("/admin/suppliers/", supplier.findAll);
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", supplier.update);
36 // receive the POST to delete a supplier
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
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 });

```

- Rebuild the Docker image for the employee microservice.

```

voctabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
[+] Building 0.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 229B
=> [internal] load metadata for docker.io/library/node:11-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fe6945a4d505932
=> [internal] load build context
=> => transferring context: 263.85kB
=> CACHED [2/5] RUN mkdir -p /usr/src/app
=> CACHED [3/5] WORKDIR /usr/src/app
=> CACHED [4/5] COPY . .
=> CACHED [5/5] RUN npm install
=> exporting to image
=> => exporting layers
=> => writing image sha256:cfab4e40fe60dc6c8f507844ab32787f3f49f17364cfef078f36ac741bd1af315
=> => naming to docker.io/library/employee
voctabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voctabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint
supplierdb.caycjdutups.us-east-1.amazonaws.com
voctabs:~/environment/microservices/employee (dev) $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST=$dbEndpoint employee
71021a663be3d3fd8a833780df2b5e4d7d6b8a9751f44ae9e79c65eab1357121
voctabs:~/environment/microservices/employee (dev) $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
71021a663be3    employee   "docker-entrypoint.s..."   About a minute ago   Up About a minute   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp   employee_1
93745d4a452f    customer   "docker-entrypoint.s..."   29 minutes ago     Up 29 minutes      0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   customer_1
voctabs:~/environment/microservices/employee (dev) $ docker rm -f employee_1
employee_1
voctabs:~/environment/microservices/employee (dev) $ 

```

Task 4.7: Check code into CodeCommit

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

```

employee/Dockerfile
Source Control
- microservices
  - Changes
    - Dockerfile Dockerfile
    - index.js M
    - supplier controller.js supplier.controller
    - header.html header.html
    - home.html home.html
    - nav.html nav.html
    - supplier-add.html supplier-add.html
    - supplier-list-all.html supplier-list-all.html
    - supplier-list-all.html supplier-list-all.html
    - supplier-update.html supplier-update.html
    - supplier-update.html supplier-upda...
  - Dockerfile
  - index.js (Working Tree) D
    1 const express = require('express');
    2 const bodyParser = require('body-parser');
    3 const cors = require('cors');
    4 const supplier = require('../app/controller/supplier.controller');
    5 const mustacheExpress = require('mustache-express');
    6 const favicon = require('serve-favicon');
    7
    8 // parse requests of content-type: application/json
    9 app.use(bodyParser.json());
    10 // parse requests of content-type: application/x-www-form-urlencoded
    11 app.use(bodyParser.urlencoded({extended: true}));
    12
    13 app.options('*', cors());
    14 app.get('*', cors());
    15 app.engine('html', mustacheExpress())
    16 app.set('views', __dirname + '/views')
    17 app.set('view engine', 'html');
    18 app.use(express.static('public'));
    19 app.use(favicon(__dirname + '/public/img/favicon.ico'));
    20
    21 // list all the suppliers
    22 app.get('/suppliers', supplier.findAll);
    23 res.render('home', {});
    24
    25 app.get('/suppliers/:id', supplier.findById);
    26 // show the add supplier form
    27 app.get('/supplier-add', (req, res) => {
    28   res.render('supplier-add', {});
    29 });
    30 // receive the add supplier POST
    31 app.post('/supplier-add', supplier.create);
    32 // show the update form
    33 app.get('/supplier-update/:id', supplier.findById);
    34 // receive the update POST
    35 app.post('/supplier-update', supplier.update);
    36 // handle the delete
    37 app.post('/supplier-remove/:id', supplier.remove);
    38 // handle the function
    39 app.get(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 });

```

```

voclabs:~/environment/microservices (dev) $ git push origin dev
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 1.60 KiB | 821.00 KiB/s, done.
Total 15 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
  5397299..279b508b6aa7c5293a04f7a96daed1528612f936fregion=us-east-1
    dev -> dev
voclabs:~/environment/microservices (dev) $

```

The screenshot shows the AWS CodeCommit interface. On the left, there's a sidebar with navigation links like 'Source • CodeCommit', 'Commits', 'Branches', 'Git tags', 'Settings', and 'Approval rule templates'. The main area displays two code diff panels. The first panel shows a Dockerfile with changes from line 1 to 7. The second panel shows a controller.js file with changes from line 22 to 86. The interface includes standard browser controls (back, forward, search, etc.) and AWS-specific features like 'Comment on file' and 'Browse file contents' buttons.

Instance details | EC2 | us-east-1 | 279b508b6aa7c5293a04f7a96d | SecurityGroup | EC2 | us-east-1 | MicroservicesIDE - AWS Cloud9 | ec2-3-93-175-72.compute-1.amazonaws.com | +

us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=... N. Virginia vocabs/use3247400-Madhale_Shweta @ 3853-9794-3428

Developer Tools **CodeCommit**

Source • CodeCommit

Getting started

Repositories

Code

Pull requests

Commits

Branches

Git tags

Settings

Approval rule templates

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Go to resource

Feedback

employee/index.js

```
84 84             res.render('500', { message: 'Error updating Supplier with id ${req.body.id}' });
85 85         }
86 86     -     } else res.redirect('/suppliers');
87 87     } else res.redirect('/admin/suppliers');
88 88     );
89 89 }
*** *** @@ -100,7 +100,7 @@
100 100     } else {
101 101         res.render("500", { message: 'Could not delete Supplier with id ${req.body.id}' });
102 102     }
103 103     -     } else res.redirect('/suppliers');
104 104     +     } else res.redirect('/admin/suppliers');
105 105     );
106 106
*** ***
```

Browse file contents Comment on file

CloudShell Feedback

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

Instance details | EC2 | us-east-1 | 279b508b6aa7c5293a04f7a96d | SecurityGroup | EC2 | us-east-1 | MicroservicesIDE - AWS Cloud9 | ec2-3-93-175-72.compute-1.amazonaws.com | +

us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=... N. Virginia vocabs/use3247400-Madhale_Shweta @ 3853-9794-3428

Developer Tools **CodeCommit**

Source • CodeCommit

Getting started

Repositories

Code

Pull requests

Commits

Branches

Git tags

Settings

Approval rule templates

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

employee/views/header.html

```
24 24     });
25 25     - app.get('/suppliers/:id', supplier.findAll);
26 26     + app.get('/admin/suppliers/:id', supplier.findAll);
27 27     // show the add supplier form
28 28     - app.get('/suppliers/add', (req, res) => {
29 29     + app.get('/admin/supplier-add', (req, res) => {
30 30         res.render("supplier-add", {});
31 31     // receive the add supplier POST
32 32     - app.post('/supplier-add', supplier.create);
33 33     + app.post('/admin/supplier-add', supplier.create);
34 34     // show the update form
35 35     - app.get('/suppliers/:id', supplier.findOne);
36 36     + app.get('/admin/supplier-update/:id', supplier.findOne);
37 37     // receive the POST to update a supplier
38 38     - app.post('/suppliers/:id', supplier.update);
39 39     + app.post('/admin/supplier-update/:id', supplier.update);
40 40     // receive the POST to delete a supplier
41 41     - app.post('/suppliers/:id', supplier.remove);
42 42     + app.post('/admin/supplier-remove/:id', supplier.remove);
43 43     // handle 404
44 44     app.use(function (req, res, next) {
45 45         const app_port = process.env.APP_PORT || 80
46 46         + const app_port = process.env.APP_PORT || 8080
47 47         app.listen(app_port, () => {
48 48             console.log(`Server is running on port ${app_port}.`);
        });
    }

    @@ -42,7 +42,7 @@
    42
    43
    44
    45
    46
    47
    48
```

Browse file contents Comment on file

CloudShell Feedback

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

Instance details | EC2 | us-east-1 | 279b508b6aa7c5293a04f7a96d | SecurityGroup | EC2 | us-east-1 | MicroservicesIDE - AWS Cloud9 | ec2-3-93-175-72.compute-1.amazonaws.com | +

us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=... N. Virginia vocabs/user3247400-Madhale_Shweta @ 3853-9794-3428

Developer Tools **CodeCommit**

Source • CodeCommit

Getting started

Repositories

Code

Pull requests

Commits

Branches

Git tags

Settings

Approval rule templates

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Go to resource

Feedback

CloudShell Feedback

employee/views/header.html

```
46 46 app.listen(app_port, () => {
47 47   console.log(`Server is running on port ${app_port}.`);
48 48 })

```

employee/views/home.html

```
*** *** @@ -4,7 +4,7 @@
4 4 <div class="container">
5 5   <h1>Welcome</h1>
6 6   <p>Use this app to keep track of your coffee suppliers</p>
7 7   - <p><a href="/suppliers">List of suppliers</a></p>
7 7 + <p><a href="/admin/suppliers">List of suppliers</a></p>
8 8 </div>
9 9 </div>
10 10 [>Footer]
```

employee/views/nav.html

```
1 1 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
```

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

This screenshot shows a comparison of three commits in the 'header.html' file. The first commit (line 7) contains a link to '/suppliers'. The second commit (line 7) changes this to '/admin/suppliers'. The third commit (line 7) reverts it back to '/suppliers'.

Instance details | EC2 | us-east-1 | 279b508b6aa7c5293a04f7a96d | SecurityGroup | EC2 | us-east-1 | MicroservicesIDE - AWS Cloud9 | ec2-3-93-175-72.compute-1.amazonaws.com | +

us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/microservices/commit/279b508b6aa7c5293a04f7a96daed1528612f936?region=... N. Virginia vocabs/user3247400-Madhale_Shweta @ 3853-9794-3428

Developer Tools **CodeCommit**

Source • CodeCommit

Getting started

Repositories

Code

Pull requests

Commits

Branches

Git tags

Settings

Approval rule templates

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Go to resource

Feedback

CloudShell Feedback

employee/views/nav.html

```
1 1 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
2 2   
3 3   - <div><a class="navbar-brand page-title" href="/supplier">Monolithic Coffee suppliers</a></div>
3 3 + <div><a class="navbar-brand page-title" href="/supplier">Manage coffee suppliers</a></div>
4 4   <div class="collapse navbar-collapse" id="navbarSupportedContent">
5 5     <ul class="navbar-nav mr-auto">
6 6       <li class="nav-item active">
7 7         - <a class="nav-link" href="/">Home</a>
7 7 +         <a class="nav-link" href="/admin/suppliers">Administrator home</a>
8 8         <a class="nav-link" href="/suppliers">Suppliers list</a>
9 9         + <a class="nav-link" href="/">Customer home</a>
10 10      </li>
11 11    </ul>
12 12  </div>
*** *** @@ -8,7 +8,7 @@
8 8   </div>
9 9   [>errors]
10 10 </div>
11 11 - <form action="/supplier-add" method="POST">
11 11 + <form action="/admin/supplier-add" method="POST">
12 12   [>supplier-form-fields]
13 13   <button type="submit" class="btn btn-primary">Submit</button>
14 14 </form>
```

employee/views/supplier-add.html

```
*** *** @@ -24,12 +24,12 @@

```

employee/views/supplier-list-all.html

```
*** *** @@ -24,12 +24,12 @@
```

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

This screenshot shows a comparison of two commits in the 'nav.html' file. The first commit (line 3) changes the page title from 'Monolithic Coffee suppliers' to 'Manage coffee suppliers'. The second commit (line 3) reverts it back to 'Monolithic Coffee suppliers'.

The screenshot shows the AWS CodeCommit interface with two code diff panels. The left sidebar lists repositories, branches, pull requests, and other tools. The top panel displays the diff for `employee/views/supplier-list-all.html`, and the bottom panel displays the diff for `employee/views/supplier-update.html`. Both panels show code snippets with line numbers and color-coded changes (red for deleted, green for added).

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

Task 5.1: Create ECR repositories and upload the Docker images

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

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

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

Login Succeeded
voclabs:/environment/microservices (dev) $
```

- Create separate private ECR repositories customer and employee.

Amazon Elastic Container Registry

Share and deploy container software, publicly or privately

Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, manage, share, and deploy your container images and artifacts anywhere.

How it works

```

graph LR
    A[Write code] --> B[Push code to a Docker image]
    B --> C[Amazon ECR]
    C --> D[Compress, encrypt, and control access to images]
    D --> E[Delete, tag, and manage image tags]
    E --> F[Run containers]
    F --> G[Pull images and run container instances]
    G --> H[Amazon ECS, Amazon EKS, Amazon Lambda, On-premises]
  
```

Benefits

- You pay only for the amount of data you store in your public or private repositories and data transferred to the Internet.
- ECR pricing
- Enable private connection to ECR private repositories using VPC interface endpoints and reduce cost.
- Learn more
- Deploy VPC interface endpoints

Create a repository

General settings

Visibility settings [Info](#)
Choose the visibility setting for the repository.
 Private
Access is managed by IAM and repository policy permissions.
 Public
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.
385397943428.dkr.ecr.us-east-1.amazonaws.com/ customer
8 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, underscores, periods and forward slashes.

Tag immutability [Info](#)
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.
 Disabled

Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

Deprecation warning

The screenshot shows the 'Create repository' page in the AWS ECR console. The 'General settings' section is active, displaying visibility settings (Private selected), a repository name ('employee'), and tag immutability (Disabled). A note states that visibility cannot be changed once a repository is created. The 'Image scan settings' section is shown below, containing a 'Deprecation warning' message.

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

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

Amazon Elastic Container Registry

Private registry

Repositories

Settings

Public registry

Repositories

Settings

ECR public gallery

Amazon ECS

Amazon EKS

Getting started

Documentation

Private repositories

Improved basic scanning

Repositories (2)

customer

employee

View push commands

Delete

Create repository

Actions

View

Summary

Images

Permissions

Lifecycle policies

Repository tags

Edit

Repository

Repository scan filters

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

Permissions

customer

customer

Permissions

Edit JSON

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

Reset

Close

Save

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

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

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

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

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

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

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

```

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

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

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

```

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

The screenshot shows the Amazon ECR interface. On the left, a sidebar menu for 'Amazon Elastic Container Registry' includes sections for 'Private registry' (Repositories, Summary, Images, Permissions, Lifecycle Policy, Repository tags, Settings), 'Public registry' (Repositories, Settings), and links to 'ECR public gallery', 'Amazon ECS', and 'Amazon EKS'. The main content area displays the 'customer' repository under 'Amazon ECR > Private registry > Repositories > customer'. A message at the top states: 'Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.' Below this, a table titled 'Images (1)' lists one image entry:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	April 29, 2024, 13:42:44 (UTC-04)	27.70	Copy URI	sha256:8d2b168228391fbe0347d384f15b9...

Buttons for 'View push commands', 'Edit', 'Delete', 'Details', and 'Scan' are located at the top right of the table. The bottom of the page includes standard AWS navigation links like CloudShell and Feedback, and a footer with copyright information.

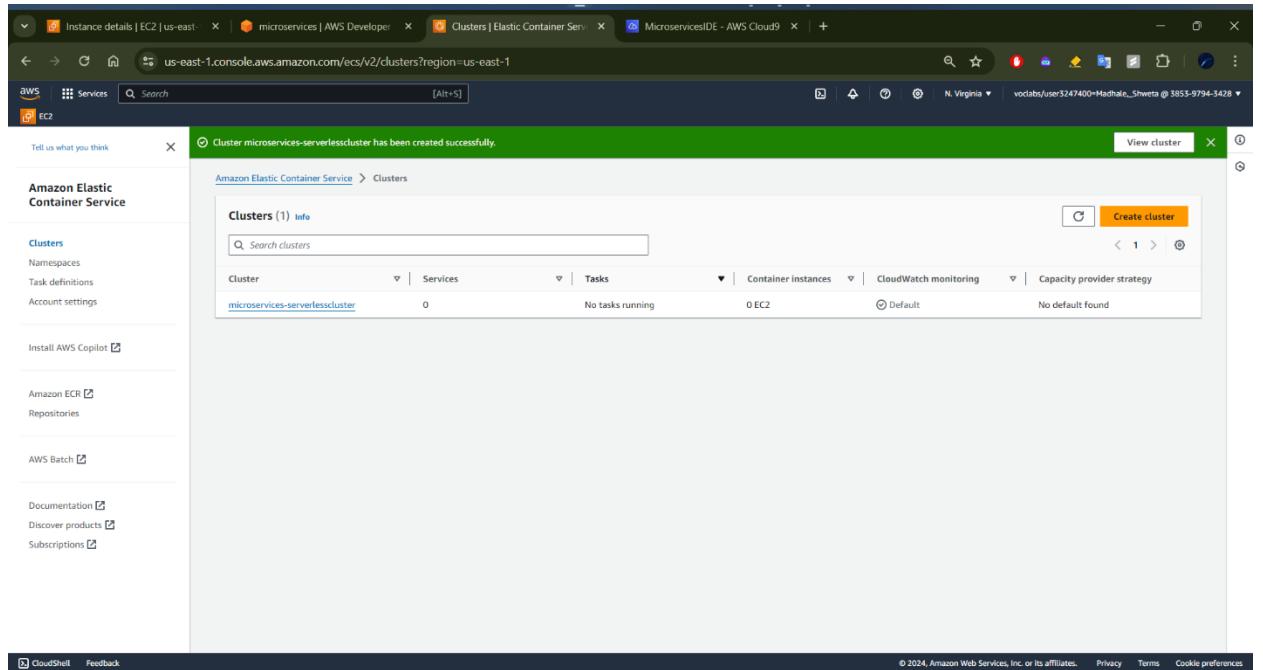
This screenshot is identical to the one above, showing the 'customer' repository details. It features the same sidebar, the same message about image scans, and the same table listing one image entry for the 'customer' repository. The bottom of the page includes standard AWS navigation links like CloudShell and Feedback, and a footer with copyright information.

Task 5.2: Create an ECS cluster

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

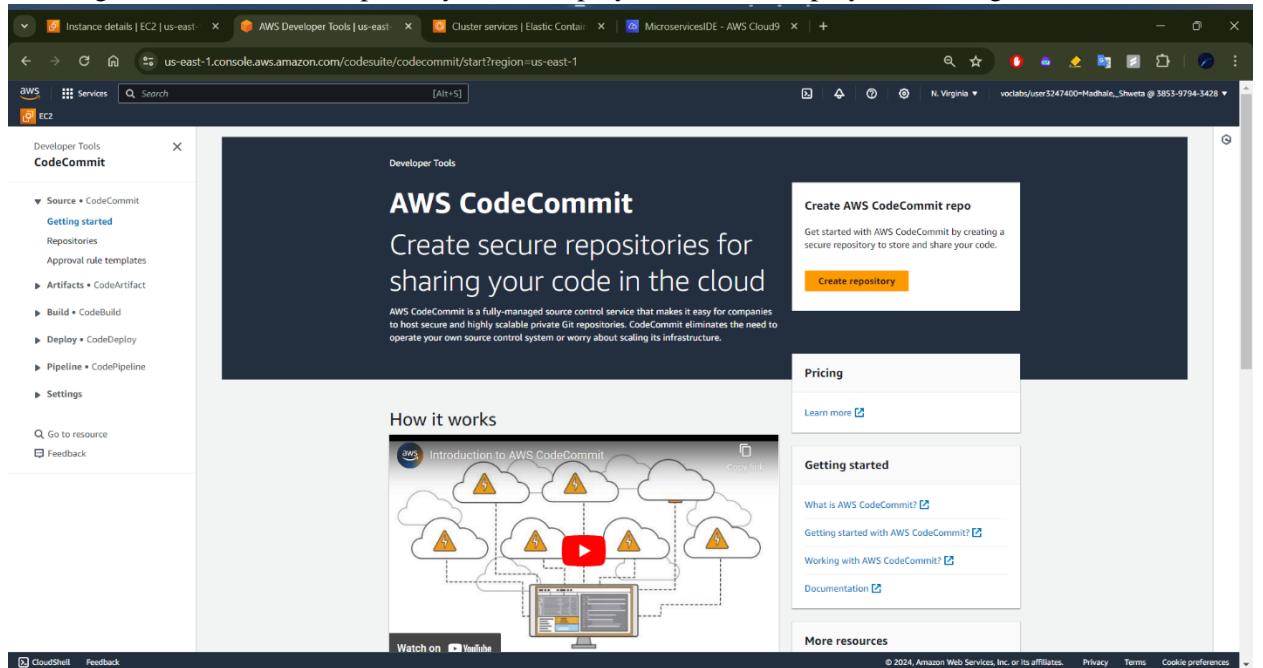
The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar has tabs for "Instance details | EC2 | us-east-1", "microservices | AWS Developer", "Clusters | Elastic Container Service", and "MicroservicesIDE - AWS Cloud9". The main content area displays the "Clusters (0) info" page for the Amazon Elastic Container Service. A search bar at the top says "Search clusters". Below it is a table with columns for Cluster, Services, Tasks, Container instances, CloudWatch monitoring, and Capacity provider strategy. A message at the bottom states "No clusters to display". On the left sidebar, under the "Amazon Elastic Container Service" heading, there are links for Clusters, Namespaces, Task definitions, Account settings, Install AWS Copilot, Amazon ECR, AWS Batch, Documentation, Discover products, and Subscriptions.

The screenshot shows the AWS Cloud9 IDE interface with the "Create cluster" tab open. The top navigation bar has tabs for "Instance details | EC2 | us-east-1", "microservices | AWS Developer", "Clusters | Elastic Container Service", and "MicroservicesIDE - AWS Cloud9". The main content area displays the "Create cluster" page for the Amazon Elastic Container Service. It starts with a "Cluster configuration" section where the "Cluster name" is set to "microservices-serverlesscluster". Below this is a "Default namespace - optional" field containing "microservices-serverlesscluster". Under the "Infrastructure" section, the "Serverless" tab is selected. It shows three options: "AWS Fargate (serverless)" (selected), "Amazon EC2 instances" (unchecked), and "External instances using ECS Anywhere" (unchecked). The "AWS Fargate (serverless)" option is described as "Pay as you go. Use if you have long, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.". The "Amazon EC2 instances" option is described as "Manual configurations. Use for large workloads with consistent resource demands.". The "External instances using ECS Anywhere" option is described as "Manual configurations. Use to add data center compute.". On the left sidebar, under the "Amazon Elastic Container Service" heading, there are links for Clusters, Namespaces, Task definitions, Account settings, Install AWS Copilot, Amazon ECR, AWS Batch, Documentation, Discover products, and Subscriptions.



Task 5.3: Create a CodeCommit repository to store deployment files

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



The screenshot shows the 'Create repository' wizard in AWS CodeCommit. The 'Repository settings' step is active, displaying fields for 'Repository name' (set to 'deployment'), 'Description - optional' (empty), and 'Tags' (empty). Advanced configuration options like AWS KMS key and Amazon CodeGuru Reviewer are shown as optional. A note indicates a service-linked role will be created in IAM.

Create repository

Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.

Repository settings

Repository name
deployment
100 characters maximum. Other limits apply.

Description - optional

Tags

Add tag

► Additional configuration
AWS KMS key

Enable Amazon CodeGuru Reviewer for Java and Python - optional
Get recommendations to improve the quality of the Java and Python code for all pull requests in this repository.
A service-linked role will be created in IAM on your behalf if it does not exist.

Cancel Create

The screenshot shows the 'deployment' repository setup page. It displays a success message: 'Repository successfully created'. The 'Connection steps' section shows the 'HTTPS' tab selected, with a note about using federated access or temporary credentials. It also lists 'Step 1: Prerequisites' and 'Step 2: Set up the AWS CLI Credential Helper'. A sidebar on the left provides navigation for developer tools and repositories.

Success
Repository successfully created

Developer Tools > CodeCommit > Repositories > deployment

deployment

Connection steps

HTTPS SSH HTTPS (GRC)

Step 1: Prerequisites
You must use a Git client that supports Git version 1.7.9 or later to connect to an AWS CodeCommit repository. If you do not have a Git client, you can install one from Git downloads page [View Git downloads page](#).

Step 2: Set up the AWS CLI Credential Helper
Set up your connection to AWS CodeCommit repositories using the credential helper included in the AWS CLI. This is the only connection method for AWS CodeCommit repositories that does not require an IAM user, so it is the only method that supports root access, federated access, and temporary credentials. [Learn more](#)

Additional details
You can find more detailed instructions in the documentation. [View documentation](#)

Clone URL

Developer Tools Feedback

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

```

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:
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 remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/deployment
vocabs:~/environment/deployment (master) $ git checkout -b dev
Switched to a new branch 'dev'
vocabs:~/environment/deployment (dev) $ 

```

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

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

```

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

```

```
1  {
2     "containerDefinitions": [
3         {
4             "name": "customer",
5             "image": "customer",
6             "environment": [
7                 {
8                     "name": "APP_DB_HOST",
9                     "value": "supplierdb.ccaycjsups.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        },
21        {
22            "logConfiguration": {
23                "logDriver": "awslogs",
24                "options": {
25                    "awslogs-create-group": "true",
26                    "awslogs-region": "us-east-1",
27                    "awslogs-group": "task-1",
28                    "awslogs-stream-prefix": "awslogs-captone"
29                }
30            }
31        }
32    ],
33    "requiresCompatibilities": [
34        "FARGATE"
35    ],
36    "networkMode": "awsvpc",
37    "memory": "1024",
38    "executionRoleArn": "arn:aws:iam::38597943428:role/PipelineRole",
39    "family": "customer-microservice"
40 }
```

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

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

```
{  
    "taskDefinition": {  
        "taskDefinitionArn": "arn:aws:ecs:us-east-1:385397943428:task-definition/customer-microservice:5",  
        "containerDefinitions": [  
            {  
                "name": "customer",  
                "image": "customer",  
                "cpu": 0,  
                "portMappings": [  
                    {  
                        "containerPort": 8080,  
                        "hostPort": 8080,  
                        "protocol": "tcp"  
                    }  
                ],  
                "essential": true,  
                "environment": [  
                    {  
                        "name": "APP_DB_HOST",  
                        "value": "supplierdb.ccaycjdtuups.us-east-1.rds.amazonaws.com"  
                    }  
                ],  
                "mountPoints": [],  
                "volumesFrom": [],  
                "logConfiguration": {  
                    "logDriver": "awslogs",  
                    "options": {  
                        "awslogs-create-group": "true",  
                        "awslogs-group": "awslogs-capstone",  
                        "awslogs-region": "us-east-1",  
                        "awslogs-stream-prefix": "awslogs-capstone"  
                    }  
                },  
                "systemControls": []  
            }  
        ],  
        "family": "customer-microservice",  
        "executionRoleArn": "arn:aws:iam::385397943428:role/PipelineRole",  
        "networkMode": "awsvpc",  
        "revision": 5,  
        "volumes": [],  
        "status": "ACTIVE",  
        "requiresAttributes": [  
            {  
                "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"  
            },  
            {  
                "name": "ecs.capability.execution-role-awslogs"  
            },  
            :|  
        ]  
    }  
}
```

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

The screenshot shows the AWS Cloud9 IDE interface. The browser tab is titled "Task definitions | Elastic Container Service" and the URL is "us-east-1.console.aws.amazon.com/ecs/v2/task-definitions?region=us-east-1". The page displays a success message: "Cluster microservices-serverlesscluster has been created successfully." Below this, the "Task definitions (1) info" section is shown, with a table listing one task definition named "customer-microservice" which is marked as "ACTIVE".

This screenshot shows the detailed view of the "customer-microservice" task definition. The URL is "us-east-1.console.aws.amazon.com/ecs/v2/task-definitions/customer-microservice?status=ACTIVE®ion=us-east-1". The page title is "Amazon Elastic Container Service > Task definitions > customer-microservice". The table shows the task definition revision, which is "customer-microservice:5", and its status is "ACTIVE".

- Doing the same for taskdef-employee.json

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

The screenshot shows the AWS Cloud Console interface for the Amazon Elastic Container Service (ECS). The left sidebar navigation bar includes links for Clusters, Namespaces, Task definitions (which is selected), Account settings, Install AWS Copilot, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, and Subscriptions. The main content area is titled "Task definitions (2) Info". It features a search bar and a filter dropdown set to "Active". Below this, there is a table with two rows:

Task definition	Status of last revision
customer-microservice	ACTIVE
employee-microservice	ACTIVE

At the bottom right of the main content area, there are buttons for "Create new task definition", "Deploy", and "Actions". The footer of the page includes links for CloudShell, Feedback, and copyright information: © 2024, Amazon Web Services, Inc. or its affiliates.

This screenshot shows the "Task definition revisions" page for the "employee-microservice" task definition. The left sidebar and main navigation bar are identical to the first screenshot. The main content area is titled "employee-microservice (1) Info". It features a search bar and a filter dropdown set to "Active". Below this, there is a table with one row:

Task definition: revision	Status
employee-microservice:8	ACTIVE

At the bottom right of the main content area, there are buttons for "Actions" and "Create new revision". The footer of the page includes links for CloudShell, Feedback, and copyright information: © 2024, Amazon Web Services, Inc. or its affiliates.

Task 5.5: Create AppSpec files for CodeDeploy for each microservice

- Create appspec-customer.yaml

```
bash - "ip-10-16-10-55.ec x  appspec-customer.yaml x +  
1 version: 0.0  
2 Resources:  
3   - TargetService:  
4     Type: AWS::ECS::Service  
5     Properties:  
6       TaskDefinition: <TASK_DEFINITION>  
7       LoadBalancerInfo:  
8         ContainerName: "customer"  
9         ContainerPort: 8080
```

- Create appspec-employee.yaml

```
bash - "ip-10-16-10-55.ec x  appspec-customer.yaml x  appspec-employee.yaml +  
1 version: 0.0  
2 Resources:  
3   - TargetService:  
4     Type: AWS::ECS::Service  
5     Properties:  
6       TaskDefinition: <TASK_DEFINITION>  
7       LoadBalancerInfo:  
8         ContainerName: "employee"  
9         ContainerPort: 8080
```

Task 5.6: Update files and check them into CodeCommit

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

```
 1  {
 2      "containerDefinitions": [
 3          {
 4              "name": "customer",
 5              "image": "<IMAGE1_NAME>",
 6              "environment": [
 7                  {
 8                      "name": "APP_DB_HOST",
 9                      "value": "supplierdb.ccaycjdtuups.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      ],
31      "requiresCompatibilities": [
32          "FARGATE"
33      ],
34      "networkMode": "awsvpc",
35      "cpu": "512",
36      "memory": "1024",
37      "executionRoleArn": "arn:aws:iam::385397943428:role/PipelineRole",
38      "family": "customer-microservice"
39  }
```

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

```

1   {
2     "containerDefinitions": [
3       {
4         "name": "employee",
5         "image": "<IMAGE1_NAME>",
6         "environment": [
7           {
8             "name": "APP_DB_HOST",
9             "value": "supplierdb.ccaycjdtups.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      "networkMode": "awsvpc",
34      "cpu": "512",
35      "memory": "1024",
36      "executionRoleArn": "arn:aws:iam::385397943428:role/PipelineRole",
37      "family": "employee-microservice"
38    }
39  }

```

- Push the files.

```

bash - "ip-10-16-10-55.ec2.i" git - "ip-10-16-10-55.ec2.i" + 

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

```

The screenshot shows two instances of the AWS CodeCommit interface. The top instance displays a list of deployment commits for a repository named 'deployment'. A single commit is listed:

Commit ID	Commit message	Commit date	Authored date	Author	Committer	Actions
cbc600e9	Add task definition and AppSpec files for customer and employee microservices	Just now	Just now	shweta	shweta	Copy ID Browse

The bottom instance shows the contents of the 'deployment' repository, specifically the 'appspec-*' files and 'taskdef-*' JSON files.

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

Task 6.1: Create four target groups

- Create target groups with

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

The screenshot shows the AWS EC2 Dashboard for the US East (N. Virginia) Region. The left sidebar contains navigation links for Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, Load Balancers, Target Groups, Trust Stores, and Auto Scaling.

Resources: You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	2	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	0	Instances	2	Key pairs	1
Load balancers	0	Placement groups	0	Security groups	5
Snapshots	0	Volumes	2		

Launch instance: To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Service health: AWS Health Dashboard

Region	Status
US East (N. Virginia)	This service is operating normally.

Zones:

Zone name	Zone ID
us-east-1a	use1-az4
us-east-1b	use1-az6
us-east-1c	use1-az1
us-east-1d	use1-az2
us-east-1e	use1-az3

Additional information:

- Default VPC:** `vpc-00f0ecbc5793aa875`
- Settings:** Data protection and security, Zones, EC2 Serial Console, Default credit specification, Console experiments.
- Explore AWS:** 10 Things You Can Do Today to Reduce AWS Costs, Get Up to 40% Better Price Performance, Enable Best Price-Performance with AWS Graviton2.

The screenshot shows two AWS Cloud9 environments running on EC2 instances in the us-east-1 region. The top window displays the EC2 Target Groups page, which is currently empty. The bottom window shows the 'Step 1 Create target group' configuration screen.

EC2 Target Groups Page:

- Shows the 'Target groups info' section with a search bar and a 'Create target group' button.
- Table headers: Name, ARN, Port, Protocol, Target type, Load balancer, VPC ID.
- Message: "No target groups" and "You don't have any target groups in us-east-1".
- Link: "Create target group".

Create Target Group Step 1:

- Section: "Choose a target type".
 - Instances
 - Supports load balancing to instances within a specific VPC.
 - Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.
 - IP addresses
 - Supports load balancing to VPC and on-premises resources.
 - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
 - Offers flexibility with microservice-based architectures, simplifying inter-application communication.
 - Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
 - Lambda function
 - Facilitates routing to a single Lambda function.
 - Accessible to Application Load Balancers only.
 - Application Load Balancer
 - Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
 - Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.
- Section: "Target group name". Input field: "customer-tg-one". Note: "A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen."
- Section: "Protocol : Port".
 - Protocol dropdown: "HTTP".
 - Port dropdown: "8080". Note: "1-65535".
- Section: "IP address type". Note: "Only applicable with the IP address target type." Options: "Static IP" and "AWS Lambda function".

Step 1 Create target group

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 Requester pays load balancers, you can register IP addresses from this VPC, or from private IP addresses located outside of this load balancer's VPC (such as a general VPC, EC2-Classic, or on-premises targets that are reachable over Direct Connect or VPN).

Protocol version

- HTTP1**
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
- HTTP2**
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
- gRPC**
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol
HTTP

Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/

Attributes

customer-tg-one

Details

Target type	Protocol : Port	Protocol version
IP IPv4	HTTP: 8080 Load balancer None associated	HTTP1 VPC vpc-0a96197396de75cc
0 Total targets	0 Healthy 0 Unhealthy 0 Anomalous	0 Unused 0 Initial 0 Draining

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

Registered targets (0)

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Anomaly mitigation: Not applicable

Deregister targets **Register targets**

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

Instance details | EC2 | us-east-1 | deployment | AWS Developer Tools | Step 1 Create target group | EC2 | MicroservicesIDE - AWS Cloud9 | +

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

EC2 Services Search [Alt+S]

N. Virginia vocabs/user\$247400-Madhale_Shweta @ 3853-9794-3428

IP addresses

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

Lambda function

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

Application Load Balance

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name: customer-tg-two

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol: Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation.

HTTP 8080 T-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-0a9f91973996de75cc
IPv4 CIDR: 10.16.0.0/16

Protocol version

HTTP 1.1

Sends requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

HTTP2

Sends requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

gRPC

Sends requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol: HTTP

Health check path: /

Up to 1024 characters allowed.

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

Screenshot of the AWS Cloud9 IDE showing the creation of a target group named "customer-tg-two".

EC2 Target Group Details:

- Name:** customer-tg-two
- Protocol:** HTTP
- Port:** 8080
- Protocol Version:** HTTP1
- VPC:** vpc-0a9f6197396de75cc
- Target Type:** IP
- IP Address Type:** IPv4
- Health Status:** 0 Healthy, 0 Unhealthy, 0 Unused, 0 Initial, 0 Draining
- Targets:** 0 registered targets.

Step 1 Create target group:

The screenshot shows the configuration of a new target group:

- Target type:** IP addresses (selected)
- Protocol:** HTTP
- Port:** 8080
- Target group name:** employee-tg-one
- Protocol:** Port (HTTP)
- IP address type:** IPv4

Step 1 Create target group

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 Requester pays load balancers, register IP addresses from this VPC, or from private IP addresses located outside of this load balancer's VPC (such as a general VPC, EC2-Classic, or on-premises targets that are reachable over Direct Connect or VPN).

LabVPC
vpc-0a96197396de75cc
IPv4 VPC CIDR: 10.16.0.0/16

Protocol version
 HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
 HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
 gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol

Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.

Up to 1024 characters allowed.

Attributes

employee-tg-one

Details
arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/employee-tg-one/26af664861037bdb

Target type	Protocol : Port	Protocol version
IP IPv4	HTTP: 8080 Load balancer None associated	HTTP1 VPC vpc-0a96197396de75cc
0 Total targets	0 Healthy 0 Unhealthy 0 Anomalous	0 Unused 0 Initial 0 Draining

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

Registered targets (0) Info
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Anomaly mitigation: Not applicable

CloudShell **Feedback**

Instance details | EC2 | us-east-1 | deployment | AWS Developer Tools | Step 1 Create target group | EC2 | MicroservicesIDE - AWS Cloud9 | +

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

Choose a target type

Instances

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

IP addresses

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

Lambda function

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

Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name

employee-tg-two

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol : Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation

HTTP 8080 1-65535

IP address type

Only targets with the indicated IP address type can be included in this target group.

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-0af9f197996de75cc
IPv4 CIDR: 10.16.0.0/16

Protocol version

HTTP1
Send requests using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTPS.

HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol

HTTP

Health check path

Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.
/admin/supplier

Advanced health check settings

Attributes

CloudShell Feedback

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

The screenshot shows two separate instances of the AWS EC2 Target Groups console.

Top Window: Shows the details for the target group 'employee-tg-two'. It has 0 total targets, 0 healthy, 0 unhealthy, 0 unused, 0 initial, and 0 draining. The VPC is vpc-0a9f6197396de75cc.

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

Bottom Window: Shows a list of target groups. There are four entries: 'employee-tg-two', 'employee-tg-one', 'customer-tg-two', and 'customer-tg-one'. All are associated with the VPC vpc-0a9f6197396de75cc.

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
employee-tg-two	arn:aws:elasticloadbalancing:us-east-1:38539794... arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	None associated	vpc-0a9f6197396de75cc
employee-tg-one	arn:aws:elasticloadbalancing:us-east-1:38539794... arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	None associated	vpc-0a9f6197396de75cc
customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:38539794... arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	None associated	vpc-0a9f6197396de75cc
customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:38539794... arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	None associated	vpc-0a9f6197396de75cc

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

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

Screenshot of the AWS EC2 Security Groups page showing a list of existing security groups.

Security Groups [5] Info

Name	Security group ID	Security group name	VPC ID	Description	Owner
-	sg-0d569f0926cd5bac2	default	vpc-0a9f6197396de75cc	default VPC security group	385397
DBSecurityGroup	sg-07e1d5b9930reeef6	DBSecurityGroup	vpc-0a9f6197396de75cc	Enable access to MySQL on RDS	385397
-	sg-06f0cfdcb2f090da	default	vpc-00f0ecb57933a875	default VPC security group	385397
aws-cloud9-MicroservicesIDE-1646dd63...	sg-0d158bf4d06718ec4	aws-cloud9-MicroservicesIDE-1646dd63...	vpc-0a9f6197396de75cc	Security group for AWS Cloud9 enviro...	385397
-	sg-023e8dba1e5bd5869	c110523a2605598b6561189t1w3853...	vpc-0a9f6197396de75cc	Enable inbound access via TPC ports 8...	385397

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name [Info](#)
microservices-tg
Name cannot be edited after creation.

Description [Info](#)
Microservices security group

VPC [Info](#)
vpc-0a9f6197396de75cc (LabVPC)

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
Custom TCP	TCP	80	Anywhere... ▾	<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>
Custom TCP	TCP	8080	Anywhere... ▾	<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>

The screenshot shows two separate AWS Cloud9 environments side-by-side, both connected to the same AWS account and region.

Left Cloud9 Environment:

- EC2 Services:** The user is in the EC2 Load Balancers section. A modal window titled "Introducing resource map for Network Load Balancers" is open, explaining the new feature. Below it, the "Load balancers" table shows no results.
- Navigation:** The sidebar includes links for Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling (Auto Scaling Groups).

Right Cloud9 Environment:

- EC2 Services:** The user is in the EC2 SelectCreateELBWizard section, comparing and selecting load balancer types.
- Load Balancer Types:** Three options are shown:
 - Application Load Balancer:** Handles HTTP and HTTPS traffic, operating at the request level. It supports Lambda functions and containers.
 - Network Load Balancer:** Handles TCP, UDP, and TLS traffic, operating at the connection level. It supports VPCs and centralized certificate deployment.
 - Gateway Load Balancer:** Handles traffic for third-party virtual appliances supporting GEN1V2. It supports GWLB and various security features.
- Create Buttons:** Buttons for "Create Application Load Balancer" and "Create" are visible at the bottom of each section.

The screenshot shows a browser window with multiple tabs open, including 'Instance details | EC2 | us-east-' and 'deployment | AWS Developer Tools'. The main content area is titled 'Create application load balancer' and shows the 'Basic configuration' step of the 'Create Application Load Balancer' wizard. The 'Load balancer name' field is filled with 'microservicesLB'. The 'Scheme' section is set to 'Internet-facing'. The 'IP address type' section is set to 'IPv4'. The page includes a sidebar with 'How Application Load Balancers work' and a 'Next Step' button at the bottom right.

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
microservicesLB

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme [Info](#)
Scheme can't be changed after the load balancer is created.

Internet-facing
An internet-facing load balancer routes requests from clients over the Internet to targets. Requires a public subnet. [Learn more](#)

Internal
An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type [Info](#)
Select the type of IP addresses that your subnets use.

IPv4
Includes only IPv4 addresses.

Dualstack
Includes IPv4 and IPv6 addresses.

Instance details | EC2 | us-east- | deployment | AWS Developer | Create application load balance | MicroservicesIDE - AWS Cloud9

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

Services Search [Alt+S]

N. Virginia vocabs/user3247400-Madhale_Shweta @ 3853-9794-3428

EC2

Subnets

subnet-0fa9bc97ba5cf7c86 Public Subnet1

IPv4 address Assigned by AWS

us-east-1a (use1-az1)

Subnet

subnet-038598d3d75fc79e9 Public Subnet2

IPv4 address Assigned by AWS

Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group [\[?\]](#)

Security groups

Select up to 5 security groups

micservices-sg sg-027007400f26ed22a VPC vpc-0a9f6197396de75cc

Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80

Protocol Port Default action [Info](#)

CloudShell Feedback

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

Instance details | EC2 | us-east- | deployment | AWS Developer | Create application load balance | MicroservicesIDE - AWS Cloud9

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

Services Search [Alt+S]

N. Virginia vocabs/user3247400-Madhale_Shweta @ 3853-9794-3428

EC2

Network mapping [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC [Info](#)

Select the virtual private cloud (VPC) for your targets or you can create a new VPC [\[?\]](#) Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your target groups [\[?\]](#)

LabVPC

vpc-0a9f6197396de75cc IPv4 VPC CIDR: 10.16.0.0/16

Mappings [Info](#)

Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

us-east-1a (use1-az1)

Subnet

subnet-0fa9bc97ba5cf7c86 Public Subnet1

IPv4 address Assigned by AWS

us-east-1b (use1-az2)

Subnet

subnet-038598d3d75fc79e9 Public Subnet2

IPv4 address Assigned by AWS

Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group [\[?\]](#)

Security groups

CloudShell Feedback

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

The screenshot shows the AWS Cloud9 IDE interface with multiple tabs open. The main content area displays the configuration of an Application Load Balancer (ALB) listener settings. There are two listeners defined:

- Listener HTTP:80:** Protocol: HTTP, Port: 80, Default action: Forward to target group customer-tg-two (Target type: IP, IPv4). A 'Create target group' button is visible.
- Listener HTTP:8080:** Protocol: HTTP, Port: 8080, Default action: Forward to target group customer-tg-one (Target type: IP, IPv4). A 'Create target group' button is visible.

Below each listener section, there is a 'Listener tags - optional' section with an 'Add listener tag' button and a note about adding up to 50 more tags.

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

The screenshot shows the AWS Cloud9 IDE interface with multiple tabs open. The main content area displays the detailed configuration of the Application Load Balancer (ALB). The 'Listeners and rules' tab is active, showing the following configuration:

- Listeners and rules (1/2)**: A table lists two listeners:
 - HTTP:80**: Protocol: HTTP, Port: 80, Default action: Forward to target group customer-tg-two (1 rule, ARN, Not applicable, Not applicable, Not applicable, Not applicable).
 - HTTP:8080**: Protocol: HTTP, Port: 8080, Default action: Forward to target group customer-tg-one (1 rule, ARN, Not applicable, Not applicable, Not applicable, Not applicable).
- Listeners and rules (2/2)**: A table lists two rules:
 - HTTP:80**: Forward to target group customer-tg-two (1 rule, ARN, Not applicable, Not applicable, Not applicable, Not applicable).
 - HTTP:8080**: Forward to target group customer-tg-one (1 rule, ARN, Not applicable, Not applicable, Not applicable, Not applicable).

The sidebar on the left shows the EC2 dashboard and various navigation links for instances, images, and elastic block store.

Instance details | EC2 | us-east-1 | deployment | AWS Developer Tools | Step 2 Add listener rule | EC2 | MicroservicesIDE - AWS Cloud9

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...

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

Step 1 Add rule

Step 2 Define rule conditions

Step 3 Define rule actions

Step 4 Set rule priority

Step 5 Review and create

Define rule conditions Info

Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.

▶ Listener details: HTTP:80

Conditions (1)

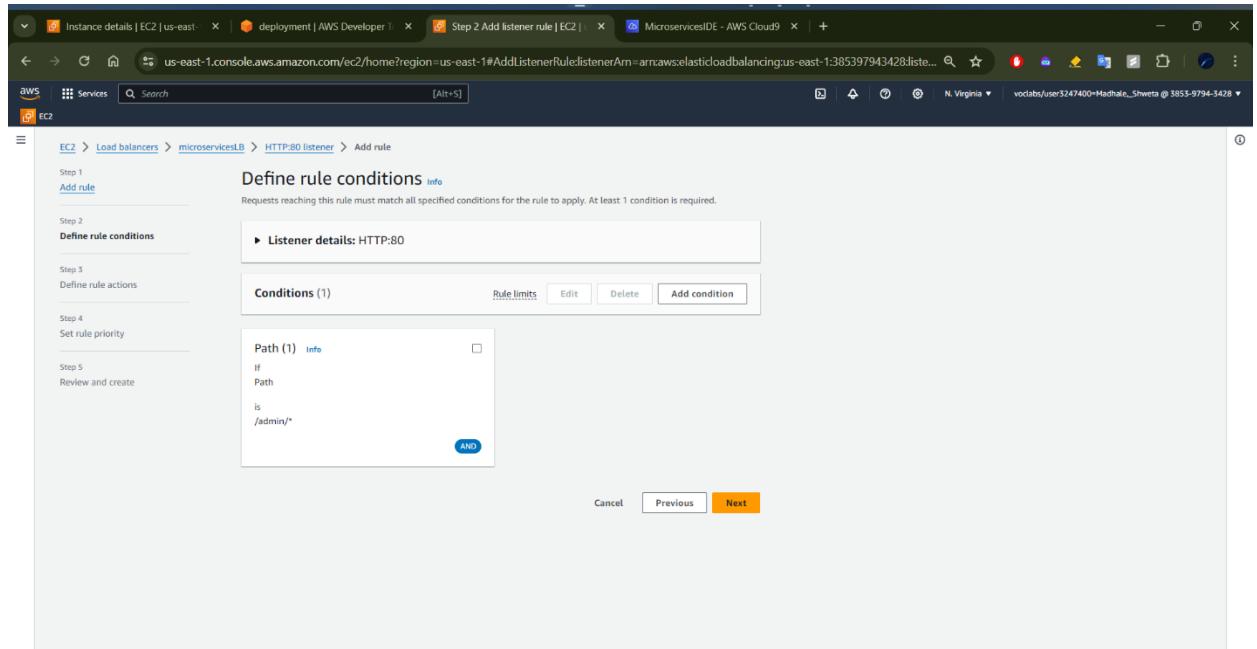
Rule limits Edit Delete Add condition

Path (1) Info

If Path
is /admin/*

AND

Cancel Previous Next



CloudShell Feedback

Instance details | EC2 | us-east-1 | deployment | AWS Developer Tools | Step 3 Add listener rule | EC2 | MicroservicesIDE - AWS Cloud9

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...

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

Step 1 Add rule

Step 2 Define rule conditions

Step 3 Define rule actions

Step 4 Set rule priority

Step 5 Review and create

Define rule actions Info

These actions will be applied to requests matching the rule conditions.

▶ Listener details: HTTP:80

Actions

Action types

Routing actions

Forward to target groups Redirect to URL Return fixed response

Forward to target group Info
Choose a target group and specify routing weight or [Create target group](#).

Target group

Target group	HTTP	Weight	Percent
employee-tg-two	HTTP	1	100%
0-999			

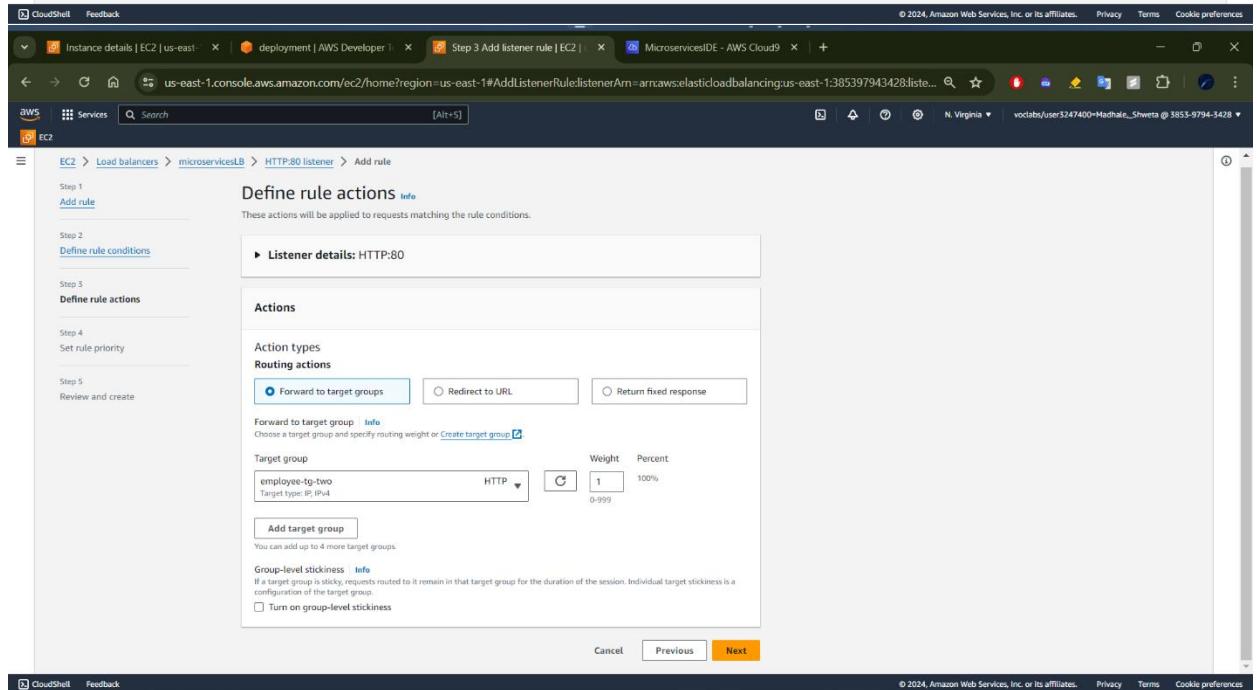
Add target group

You can add up to 4 more target groups.

Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.

Turn on group-level stickiness

Cancel Previous Next



The screenshot shows the AWS Cloud9 IDE interface with multiple tabs open. The main focus is on the 'Listener details: HTTP:80' step of creating a new listener rule. The 'Rule' section shows a priority of 1 selected. The 'Listener rules (2) info' table lists two rules:

Name tag	Priority	Conditions (If)	Actions (Then)	ARN
-	1	Path Pattern is <code>/admin/*</code>	Forward to target group • employee-tg-one 1 (100%) • Group-level stickiness: Off	Pending
Default	Last (default)	<i>If no other rule applies</i>	Forward to target group • customer-tg-two 1 (100%) • Group-level stickiness: Off	ARN

Below this, the 'HTTP:80' details page provides a summary of the listener configuration, including the protocol, port, load balancer, and rules.

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

Instance details | EC2 | us-east-1 | **deployment | AWS Developer Tools** | **Load balancer details | EC2 | us-east-1** | **MicroservicesIDE - AWS Cloud9**

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:loadbalancer/app/microservicesLB/f48a570de49fd65d... N. Virginia vocabs/user3247400=Madhale_Shweta @ 3853-9794-3428

EC2 Dashboard

Details

Load balancer type	Status	VPC	IP address type
Application	Provisioning	vpc-0a9f6197396de75cc	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z355XDTRQ7X7K	subnet-038598d3d75fc79e9 us-east-1b (use1-az6)	April 29, 2024, 14:08 (UTC-04:00)
Load balancer ARN	DNS name info	subnet-0fa9bc97ba5ef786 us-east-1a (use1-az4)	
arn:aws:elasticloadbalancing:us-east-1:385397943428:loadbalancer/app/microservicesLB/f48a570de49fd65d	microservicesLB-14827328.us-east-1.elb.amazonaws.com (A Record)		

Listeners and rules (1/2) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust
<input type="checkbox"/> HTTP:80	Forward to target group • customer-tg-two 1 (100%) • Group-level stickiness: Off	2 rules	<input type="checkbox"/> ARN	Not applicable	Not applicable	Not applicable	Not applicable
<input checked="" type="checkbox"/> HTTP:8080	Forward to target group • customer-tg-one 1 (100%) • Group-level stickiness: Off	1 rule	<input type="checkbox"/> ARN	Not applicable	Not applicable	Not applicable	Not applicable

CloudShell Feedback

Step 2 Add Listener rule | EC2 | us-east-1 | **MicroservicesIDE - AWS Cloud9**

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:loadbalancer/app/microservicesLB/f48a570de49fd65d... N. Virginia vocabs/user3247400=Madhale_Shweta @ 3853-9794-3428

Define rule conditions

Requires matching this rule must match all specified conditions for this rule to apply. At least 1 condition is required.

Listener details: HTTP:8080

Add condition

Rule condition types

Routes traffic based on the condition type of each request. Each rule can include one of each of the following conditions: host-header, path, http-request-method and source-ip. Each rule can include one or more of each of the following conditions: http-header and query-string.

Path

Path
Define the path. For example: /item/. Case sensitive.

is `/admin/`

Maximum 128 characters. Allowed characters are `[a-zA-Z][a-zA-Z][a-zA-Z]`, the following special characters `[-_.$^~!@#&=]`, and wildcards `(?)` and `(*)`.

Add new value

You can add up to 4 more condition values for this rule.

Cancel **Confirm**

Instance details | EC2 | us-east-1 | deployment | AWS Developer Tools | Step 2 Add listener rule | EC2 | MicroservicesIDE - AWS Cloud9

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...

EC2 Services Search [Alt+S]

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

Step 1 Add rule

Step 2 Define rule conditions

Step 3 Define rule actions

Step 4 Set rule priority

Step 5 Review and create

Define rule conditions Info

Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.

▶ Listener details: HTTP:8080

Conditions (1)

Rule limits Edit Delete Add condition

Path (1) Info

If Path
is /admin/*

AND

Cancel Previous Next

CloudShell Feedback

Instance details | EC2 | us-east-1 | deployment | AWS Developer Tools | Step 3 Add listener rule | EC2 | MicroservicesIDE - AWS Cloud9

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AddListenerRule:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:liste...

EC2 Services Search [Alt+S]

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

Step 1 Add rule

Step 2 Define rule conditions

Step 3 Define rule actions

Step 4 Set rule priority

Step 5 Review and create

Define rule actions Info

These actions will be applied to requests matching the rule conditions.

▶ Listener details: HTTP:8080

Actions

Action types Routing actions

Forward to target groups Redirect to URL Return fixed response

Forward to target group Info
Choose a target group and specify routing weight or [Create target group](#).

Target group: employee-tg-one (HTTP) Target type: IP, IPv4

	Weight	Percent
employee-tg-one	1	100%
0-999		

Add target group
You can add up to 4 more target groups.

Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.

Turn on group-level stickiness

Cancel Previous Next

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

Listener details: HTTP:8080

Rule

Priority
Rule priority controls the evaluation order of a rule within the listener's set of rules. You can leave gaps in priority numbers.

1	1 - 50000
---	-----------

Listener rules (2) Info

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Name tag	Priority	Conditions (If)	Actions (Then)	ARN
-	1	Path Pattern is /admin/*	Forward to target group • employee-tg-one 1 (100%) • Group-level stickiness: Off	Pending
Default	Last (default)	If no other rule applies	Forward to target group • customer-tg-one 1 (100%) • Group-level stickiness: Off	ARN

HTTP:8080 Info

Details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol:Port	Load balancer	Default actions
HTTP:8080	microservicesLB	Forward to target group • customer-tg-one 1 (100%) • Group-level stickiness: Off

Listener ARN

arn:aws:elasticloadbalancing:us-east-1:138539794342:listener/app/microservicesLB/f4ba570de49fd63d3a54f9eca922bb44

Rules | **Tags**

Listener rules (2) Info

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

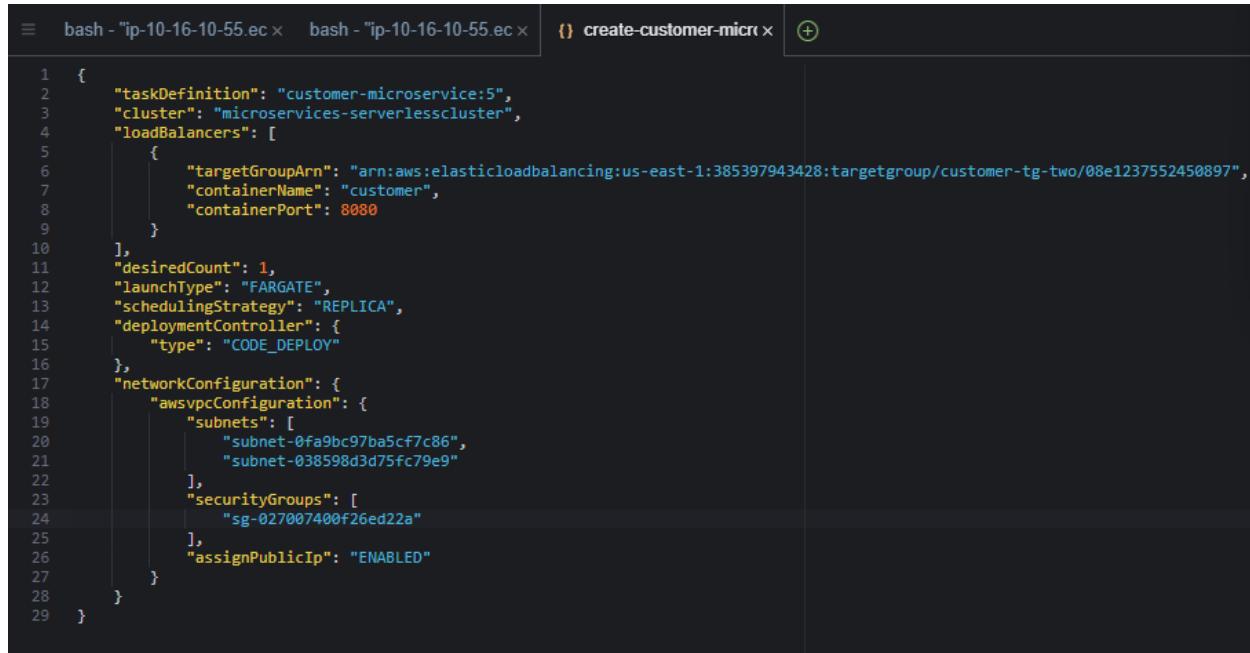
Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
-	1	Path Pattern is /admin/*	Forward to target group • employee-tg-one 1 (100%) • Group-level stickiness: Off	ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group • customer-tg-one 1 (100%) • Group-level stickiness: Off	ARN	0 tags

Phase 7 - Create ECS services.

Task 7.1: Create the ECS service for the customer microservice

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

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



```

1  {
2      "taskDefinition": "customer-microservice:5",
3      "cluster": "microservices-serverlesscluster",
4      "loadBalancers": [
5          {
6              "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/customer-tg-two/08e1237552450897",
7              "containerName": "customer",
8              "containerPort": 8080
9          }
10     ],
11     "desiredCount": 1,
12     "launchType": "FARGATE",
13     "schedulingStrategy": "REPLICAS",
14     "deploymentController": {
15         "type": "CODE_DEPLOY"
16     },
17     "networkConfiguration": {
18         "awsvpcConfiguration": {
19             "subnets": [
20                 "subnet-0fa9bc97ba5cf7c86",
21                 "subnet-038598d3d75fc79e9"
22             ],
23             "securityGroups": [
24                 "sg-027007400f26ed22a"
25             ],
26             "assignPublicIp": "ENABLED"
27         }
28     }
29 }

```

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

```

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

```

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

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

- Create an Amazon ECS service for the employee microservice

```

1  {
2    "taskDefinition": "employee-microservice:5",
3    "cluster": "microservices-serverlesscluster",
4    "loadBalancers": [
5      {
6        "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385397943428:targetgroup/employee-tg-two/98bab72e0e584358",
7        "containerName": "employee",
8        "containerPort": 8080
9      }
10   ],
11   "desiredCount": 1,
12   "launchType": "FARGATE",
13   "schedulingStrategy": "REPLICA",
14   "deploymentController": {
15     "type": "CODE_DEPLOY"
16   },
17   "networkConfiguration": {
18     "awsvpcConfiguration": {
19       "subnets": [
20         "subnet-0fa9bc97ba5cf7c86",
21         "subnet-038598d3d75fc79e9"
22       ],
23       "securityGroups": [
24         "sg-027007400f26ed22a"
25       ],
26       "assignPublicIp": "ENABLED"
27     }
28   }
29 }
```

- Run AWS CLI Command to create the service

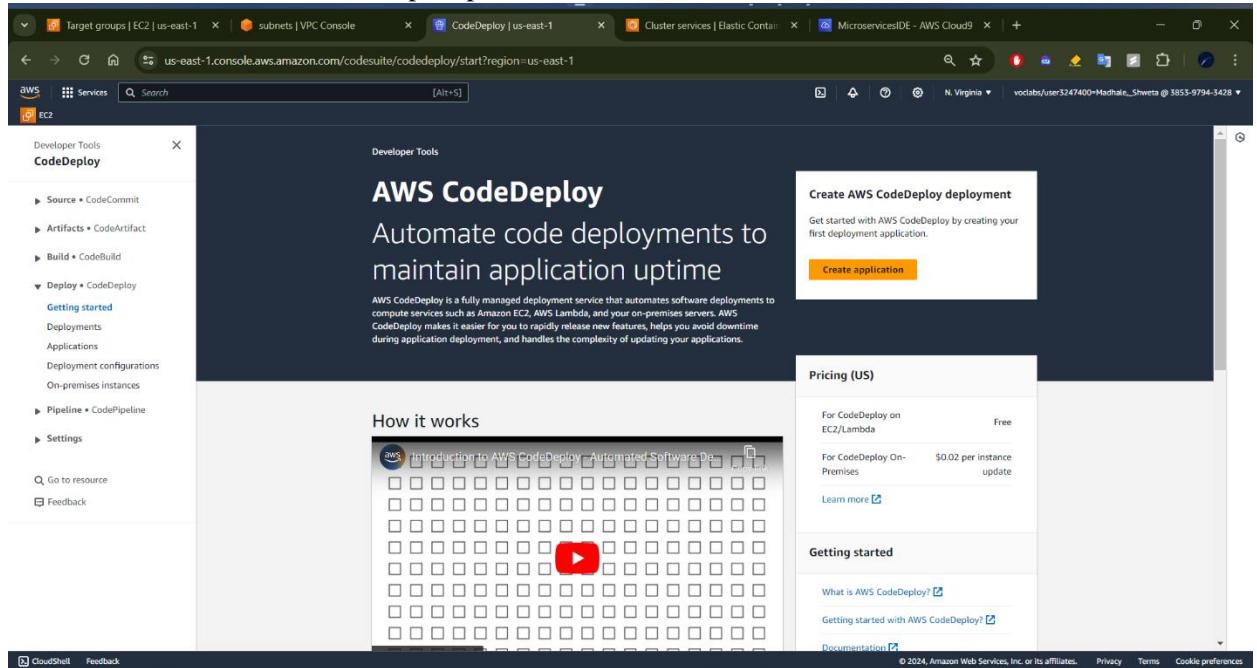
```
vocabs:~/environment/deployment (dev) $ aws ecs create-service --service-name employee-microservice --cli-input-json file://create-employee-microservice-tg-two.json
```

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

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

Task 8.1: Create a CodeDeploy application and deployment groups

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



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

Target groups | EC2 | us-east-1 | subnets | VPC Console | Create application | CodeDeploy | Cluster services | Elastic Container | MicroservicesIDE - AWS Cloud9 | +

us-east-1.console.aws.amazon.com/codesuite/codedeploy/application/new?region=us-east-1

Developer Tools > CodeDeploy > Applications > Create application

Create application

Application configuration

Application name
Enter an application name
 100 character limit

Compute platform
Choose a compute platform

Tags

CloudShell Feedback

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

Target groups | EC2 | us-east-1 | subnets | VPC Console | Create deployment group | CodeDeploy | Cluster services | Elastic Container | MicroservicesIDE - AWS Cloud9 | +

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

Services Search [Alt+S]

Compute-type
Amazon ECS

Deployment group name
Enter a deployment group name
 100 character limit

Service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

Environment configuration
Choose an ECS cluster name
 Choose an ECS service name

Load balancers

CloudShell Feedback

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

Create deployment group

Choose an ECS service name
customer-microservice

Load balancers

Choose a load balancer
microservicesLB

Production listener port
HTTP: 80

Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it
HTTP: 8080

Target group 1 name
customer-tg-two

Target group 2 name
customer-tg-one

Deployment settings

Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
 Reroute traffic immediately
 Specify when to reroute traffic

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

Create deployment group

Target group 1 name
customer-tg-two

Target group 2 name
customer-tg-one

Deployment settings

Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
 Reroute traffic immediately
 Specify when to reroute traffic

Deployment configuration
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.
CodeDeployDefault.ECSAllAtOnce

Original revision termination
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically.

Days	Hours	Minutes
0	0	5

► Advanced - optional

Cancel Create deployment group

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

The screenshot shows the AWS CodeDeploy application details page for 'microservices'. The 'Deployment groups' tab is selected. A table lists a single deployment group:

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

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

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

The screenshot shows the 'Create deployment group' wizard. Step 1: Enter a deployment group name (microservices-employee). Step 2: Service role (awsiam:385397943428role/DeployRole). Step 3: Environment configuration (ECS cluster: microservices-serverlesscluster, service: employee-microservice). Step 4: Load balancers (microservicesLB).

Create deployment group

Choose an ECS service name
employee-microservice

Load balancers

Choose a load balancer
microservicesLB

Production listener port
HTTP: 80

Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it
HTTP: 8080

Target group 1 name
employee-tg-two

Target group 2 name
employee-tg-one

Deployment settings

Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
 Reroute traffic immediately
 Specify when to reroute traffic

Create deployment group

Target group 1 name
employee-tg-two

Target group 2 name
employee-tg-one

Deployment settings

Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
 Reroute traffic immediately
 Specify when to reroute traffic

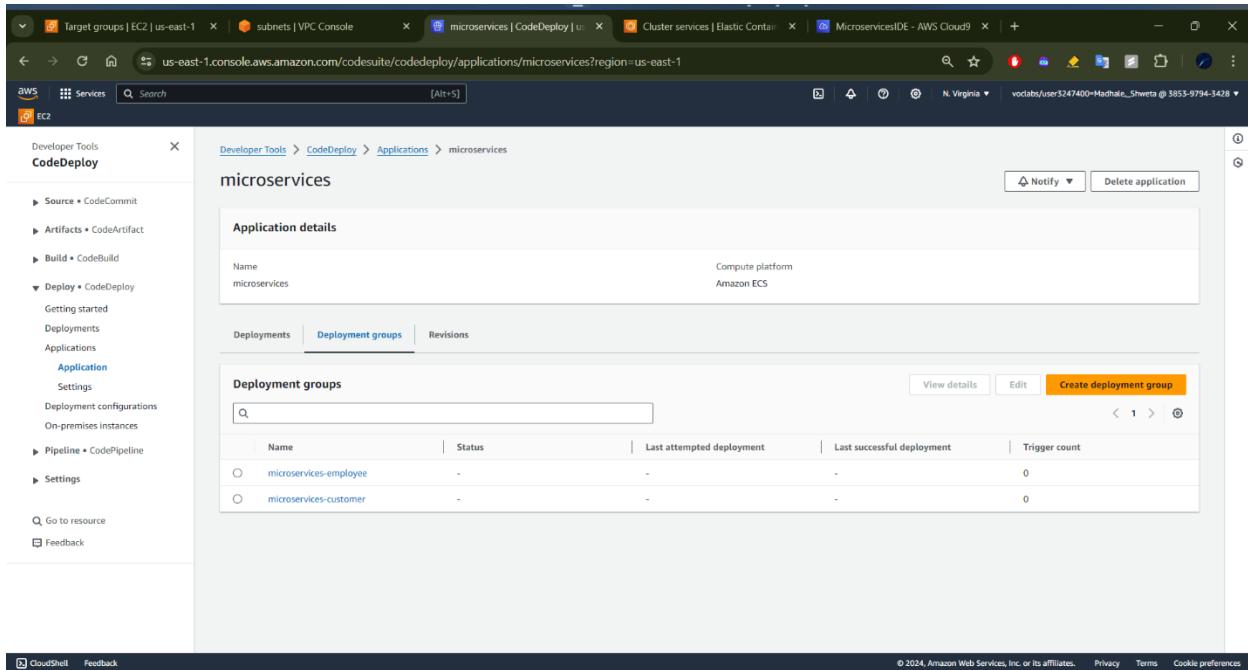
Deployment configuration
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.
CodeDeployDefault.ECSAllAtOnce

Original revision termination
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically.

Days	Hours	Minutes
0	0	5

► Advanced - optional

Cancel **Create deployment group**



Task 8.2: Create a pipeline for the customer microservice

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

The screenshot shows the AWS CodePipeline landing page. On the left, there is a sidebar with navigation links: Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Deploy (CodeDeploy), Pipeline (CodePipeline), Getting started, Pipelines, and Settings. Below the sidebar, there is a "How it works" section with a large image. To the right of the image, there is a "Create AWS CodePipeline pipeline" button. A modal window titled "Pricing (US)" is open, showing the cost of \$1/month for each active pipeline. The modal also includes terms and conditions and a "Create pipeline" button.

The screenshot shows the "Choose pipeline settings" step of the AWS CodePipeline creation wizard. The left sidebar lists steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review). The main area shows the "Pipeline settings" configuration. It includes fields for "Pipeline name" (set to "update-customer-microservice"), "Pipeline type" (set to "V2"), "Execution mode" (set to "Queued (Pipeline type V2 required)"), "Service role" (set to "Existing service role" with ARN "Q_arnawsiam:385597945428:role/PipelineRole"), and a "Role ARN" input field containing "Q_arnawsiam:385597945428:role/PipelineRole".

The screenshot shows the 'Source' configuration step in the AWS CodePipeline pipeline creation wizard. The left sidebar lists steps: Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), Step 5 (Review). The main panel is titled 'Source' and contains the following fields:

- Source provider:** AWS CodeCommit
- Repository name:** Q_deployment
- Branch name:** dev
- Change detection options:** Amazon CloudWatch Events (recommended) is selected.
- Output artifact format:** CodePipeline default is selected.

At the bottom are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted.

The screenshot shows the 'Add build stage' configuration step in the AWS CodePipeline pipeline creation wizard. The left sidebar lists steps: Step 1 (Choose pipeline settings), Step 2 (Add build stage), Step 3 (Add deploy stage), Step 4 (Review). The main panel is titled 'Add build stage' and contains the following fields:

- Build provider:** A dropdown menu is open, showing 'Skip build stage' as the selected option.

A modal dialog box titled 'Skip build stage' is displayed in the center:

Your pipeline will not include a build stage. Are you sure you want to skip this stage?

Buttons: Cancel, Skip (highlighted).

At the bottom are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted.

Target groups | EC2 | us-east-1 | subnets | VPC Console | microservices | CodeDeploy | Create new pipeline | CodePipe | MicroservicesIDE - AWS Cloud9

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

Step 4 Add deploy stage Step 5 Review

Deploy

Deploy provider Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Amazon ECS (Blue/Green)

Region US East (N. Virginia)

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

Q microservices

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.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 appspecyaml

Dynamically update task definition image - optional You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details Select input artifact

Placeholder text in the task definition

CloudShell Feedback

Target groups | EC2 | us-east-1 | subnets | VPC Console | microservices | CodeDeploy | Create new pipeline | CodePipe | MicroservicesIDE - AWS Cloud9

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

Step 4 Add deploy stage Step 5 Review

Deploy

Deploy provider Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Amazon ECS (Blue/Green)

Region US East (N. Virginia)

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

Q microservices

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.

SourceArtifact taskdef-customer.json

The default path is taskdef.json.

AWS CodeDeploy AppSpec file Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.

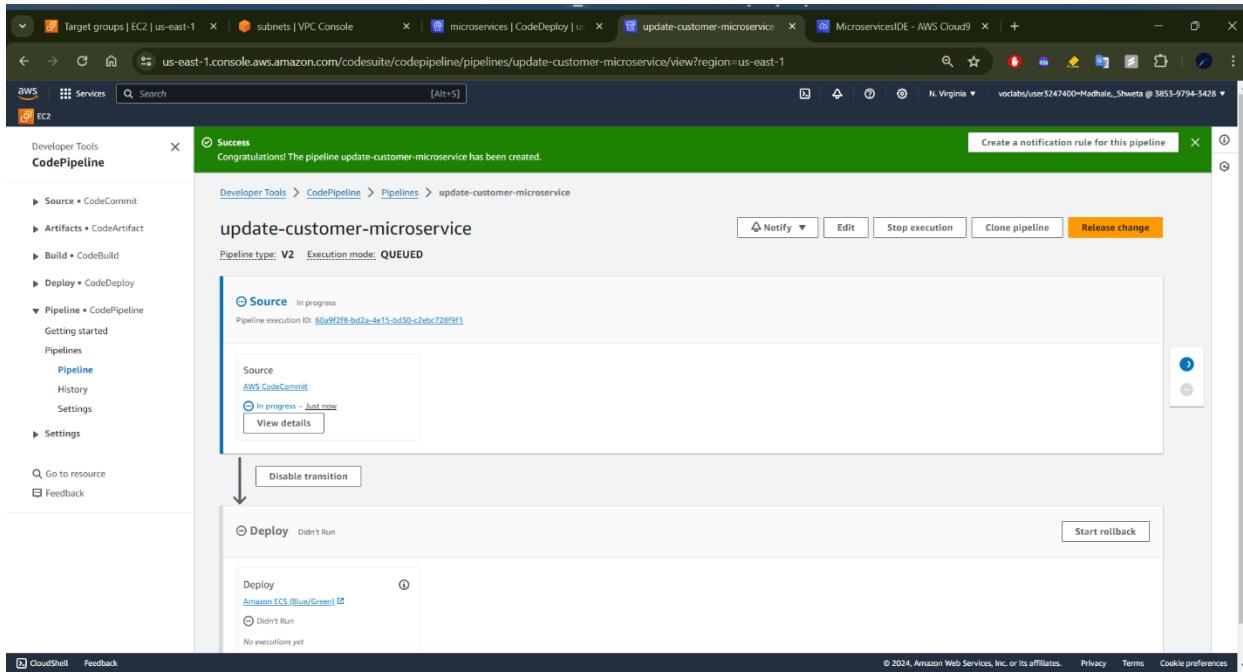
SourceArtifact appspec-customer.yaml

Dynamically update task definition image - optional You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details Select input artifact

Placeholder text in the task definition

CloudShell Feedback



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

Target groups | EC2 | us-east-1 | subnets | VPC Console | microservices | CodeDeploy | Edit update-customer-microservice | MicroservicesIDE - AWS Cloud9

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: Q customer

Image tag - optional: Q latest

Variable namespace - optional:

Output artifacts: image-customer

Cancel Done

Deploy: Amazon ECS (Blue/Green)

Configure automatic rollback on stage failure

Target groups | EC2 | us-east-1 | subnets | VPC Console | microservices | CodeDeploy | Edit update-customer-microservice | MicroservicesIDE - AWS Cloud9

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

aws Services Search [Alt+S]

N. Virginia vodabas/user3247400-Madhale_Shweta @ 3855-9794-3428

Developer Tools CodePipeline

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
- Getting started
- Pipelines
 - Pipeline**
- History
- Settings
- Settings

Go to resource Feedback

Edit: Triggers

No triggers

This pipeline does not have any source actions that support triggers. Triggers are only supported for the CodeStarSourceConnection action type.

Edit: Source

Source: AWS CodeCommit

Image: Amazon ECR

+ Add stage

Edit: Deploy

Deploy: Amazon ECS (Blue/Green)

Configure automatic rollback on stage failure

+ Add stage

CloudShell Feedback

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

Edit action

Action name
Choose a name for your action
Deploy

Action provider
Amazon ECS (Blue/Green)

Region
US East (N. Virginia)

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

SourceArtifact
image-customer

Add

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

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

Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact
taskdef-customer.json

The default path is taskdef.json.

AWS CodeDeploy AppSpec file

Add

No more than 100 characters

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

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

Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact
taskdef-customer.json

The default path is taskdef.json.

AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec File is stored. If other than the default file path, specify the path and filename of your AppSpec file.

SourceArtifact
appspec-customer.yaml

Dynamically update task definition image - optional
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details
image-customer

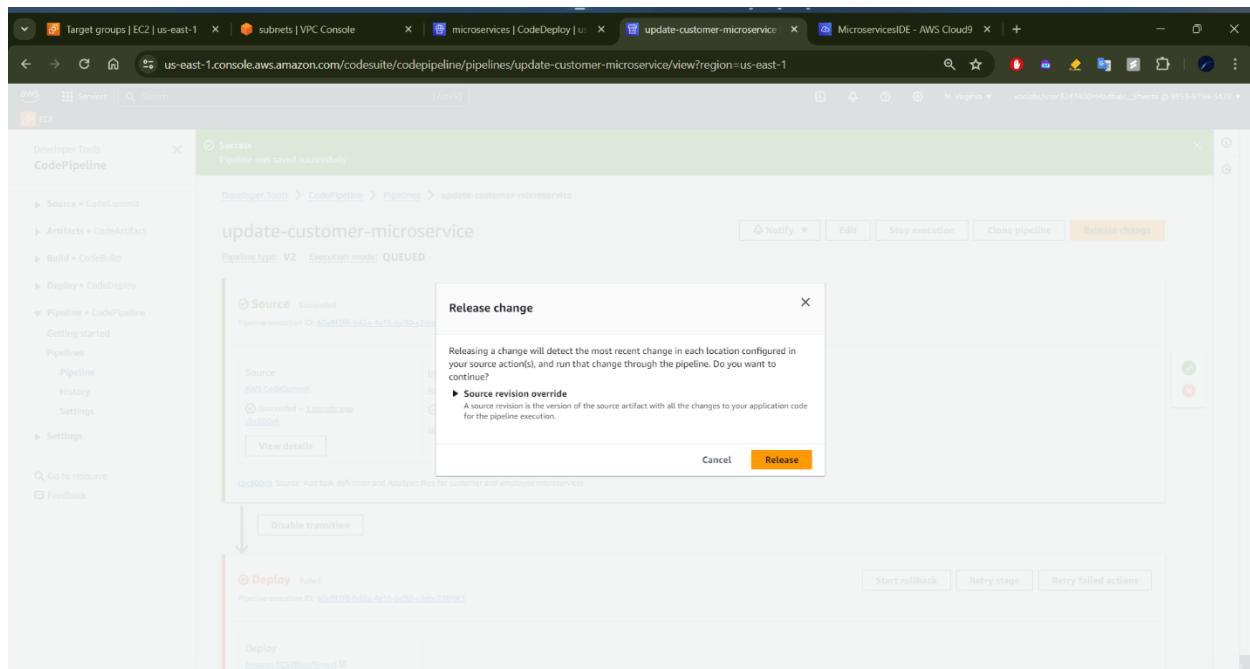
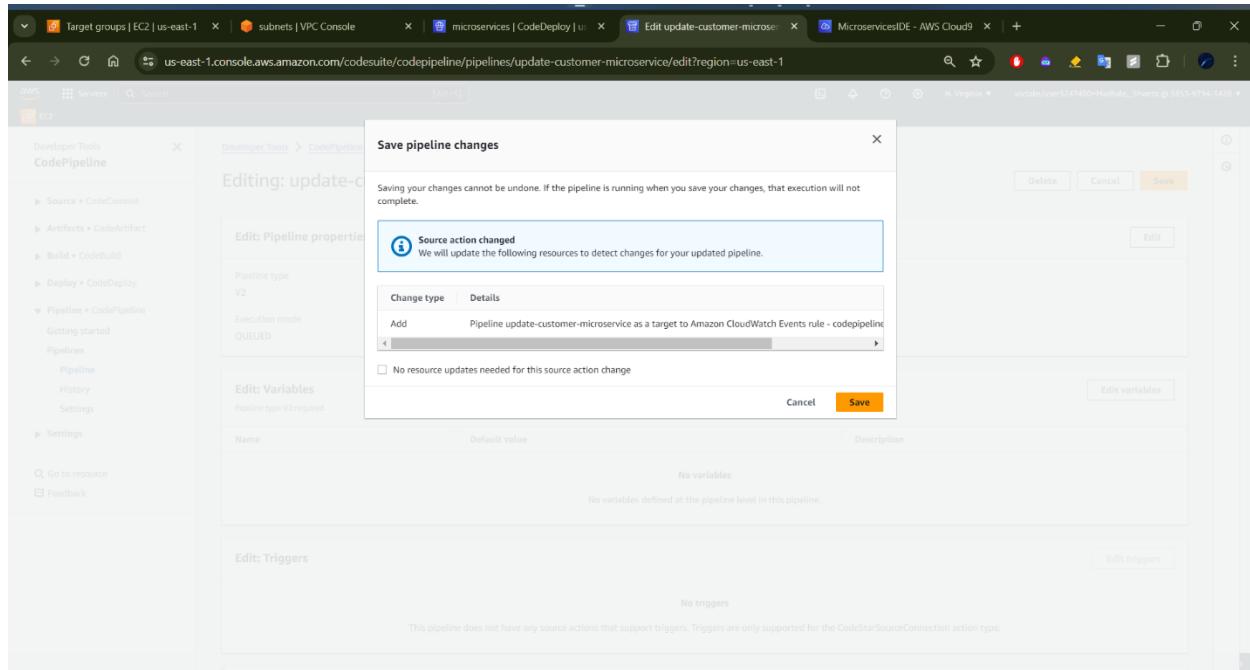
Placeholder text in the task definition
IMAGE1_NAME

Add

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

DeployVariables

Cancel Done



Screenshot of the AWS CodePipeline console showing a pipeline execution. The pipeline consists of two stages: Source and Deploy.

Source Stage: Status: Succeeded. Pipeline execution ID: eb7a0f6c-ea7b-4a2a-b1cb-5a75f318a6e0. Task details: AWS CodeCommit (Source) - Just now (View details). Image: Amazon ECS (View details). Status: Succeeded - Just now (View details). Description: chb600ad Source: Add task definition and AppSpec files for customer and employee microservices. Image: sha256:8d2b16822839.

Deploy Stage: Status: In progress. Pipeline execution ID: eb7a0f6c-ea7b-4a2a-b1cb-5a75f318a6e0. Task details: Amazon ECS (Blue/Green) (View details). Status: In progress - Just now (View details). Description: chb600ad Source: Add task definition and AppSpec files for customer and employee microservices. Image: sha256:8d2b16822839.

Screenshot of the AWS CodeDeploy console showing a deployment named d-4VOT1OJU5. The deployment status is displayed in a timeline.

Deployment status:

- Step 1: Deploying replacement task set (Completed, Succeeded)
- Step 2: Test traffic route setup (Completed, Succeeded)
- Step 3: Remove production traffic to replacement task set (100% traffic shifted, Succeeded)
- Step 4: Wait 5 minutes 0 seconds (Waiting, In progress)
- Step 5: Terminate original task set (Not started, In progress)

Traffic shifting progress:

Original	Replacement
0%	100%
Original task set not serving traffic	Replacement task set serving traffic

The screenshot shows the AWS CodePipeline console with a successful pipeline execution. The pipeline consists of two stages: Source and Deploy.

- Source Stage:** Pipeline execution ID: [d4VOT1OJUS](#). Status: Succeeded. Task: AWS CodeCommit. Sub-task: Successed - 7 minutes ago. Image: Amazon ECB. Sha256:8d2b16822839.
- Deploy Stage:** Pipeline execution ID: [d4VOT1OJUS](#). Status: Succeeded. Task: Amazon ECS (Blue/Green). Sub-task: Successed - Just now. Image: sha256:8d2b16822839.

A large green arrow points from the Source stage to the Deploy stage, indicating the flow of the pipeline. A "Disable transition" button is located between the stages.

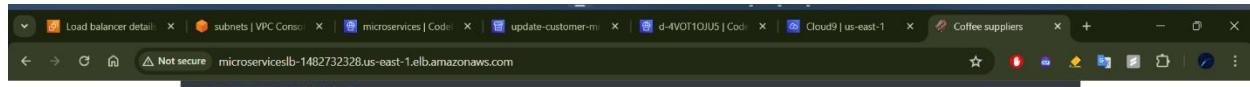
The screenshot shows the AWS CodeDeploy console for deployment [d-4VOT1OJUS](#).

- Revision details:** Revision location: [c17e222bf545443547084b8a1b074cb05779748cb42b768cd716c6883f](#), Revision created: 7 minutes ago, Revision description: Application revision registered by Deployment ID: d-4VOT1OJUS ed9fe.
- Task set activity:**

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/0140707528900518217	Original	ACTIVE	0	1	0	0
ecs-svc/9404368754840962075	Replacement	PRIMARY	100%	1	1	0
- Deployment lifecycle events:**

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 29, 2024 3:19 PM (UTC-4:00)	Apr 29, 2024 3:19 PM (UTC-4:00)
Install	2 minutes 3 seconds	Succeeded	Apr 29, 2024 3:19 PM (UTC-4:00)	Apr 29, 2024 3:21 PM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 29, 2024 3:21 PM (UTC-4:00)	Apr 29, 2024 3:21 PM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 3:21 PM (UTC-4:00)	Apr 29, 2024 3:21 PM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 3:21 PM (UTC-4:00)	Apr 29, 2024 3:21 PM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:21 PM (UTC-4:00)	Apr 29, 2024 3:21 PM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:21 PM (UTC-4:00)	Apr 29, 2024 3:21 PM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:21 PM (UTC-4:00)	Apr 29, 2024 3:21 PM (UTC-4:00)

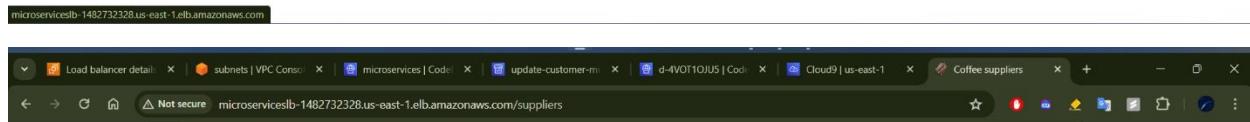
Task 8.3: Test the CI/CD pipeline for the customer microservice



Welcome

Use this app to keep track of your coffee suppliers

[List of suppliers](#)



All suppliers

Name	Address	City	State	Email	Phone
shweta madhale	columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

Target groups | EC2 | subnets | VPC Console | microservices | CodeDeploy | update-customer-microservice | d-4VOT1OJUS | Cloud9 | us-east-1 | Coffee suppliers | + | N. Virginia | vodlabs/user5247400-Madhale_Shweta @ 3855-9794-3428

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroupsv=3;\$case=false\$tags=false\$regex=false\$tags=false\$case=false

EC2 > Target groups

Target groups (4) Info

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	microservicesLB	vpc-0a9f6197396de75cc
customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	None associated	vpc-0a9f6197396de75cc
employee-tg-one	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	microservicesLB	vpc-0a9f6197396de75cc
employee-tg-two	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	microservicesLB	vpc-0a9f6197396de75cc

Actions Create target group

0 target groups selected

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

Listener details | EC2 | subnets | VPC Console | microservices | CodeDeploy | update-customer-microservice | d-4VOT1OJUS | Cloud9 | us-east-1 | Coffee suppliers | + | N. Virginia | vodlabs/user5247400-Madhale_Shweta @ 3855-9794-3428

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ELBLListenerV2:loadBalancerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428:l...

EC2 > Listener details

HTTP:80 Info

▼ Details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol:Port	Load balancer	Default actions
HTTP:80	microservicesLB	Forward to target group • customer-tg-one 1 (100%) • Group-level stickiness: Off

Listener ARN

arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd63d/1647cb35c382ee3b

Rules Tags

Listener rules (2) Info

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
-	1	Path Pattern is /admin/*	Forward to target group • employee-tg-two 100 (100%) • Group-level stickiness: Off	ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group • customer-tg-one 1 (100%) • Group-level stickiness: Off	ARN	0 tags

Rule limits Actions Add rule

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

Task 8.4: Create a pipeline for the employee microservice

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

EC2 | us-east-1 | subnets | VPC Console | Create new pipeline | update-customer-m... | d-4VOT1OJUS | CodePipeline | Cloud9 | us-east-1 | Coffee suppliers | + | N. Virginia | vclabs/user3247400-Madhale_Shweta @ 3855-9794-3428

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1
Choose pipeline settings Info

Step 1 of 5

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
 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 © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 | us-east-1 | subnets | VPC Console | Create new pipeline | update-customer-m... | d-4VOT1OJUS | CodePipeline | Cloud9 | us-east-1 | Coffee suppliers | + | N. Virginia | vclabs/user3247400-Madhale_Shweta @ 3855-9794-3428

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 2
Add source stage

Step 3
Add build stage

Step 4
Add deploy stage

Step 5
Review

Source

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

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

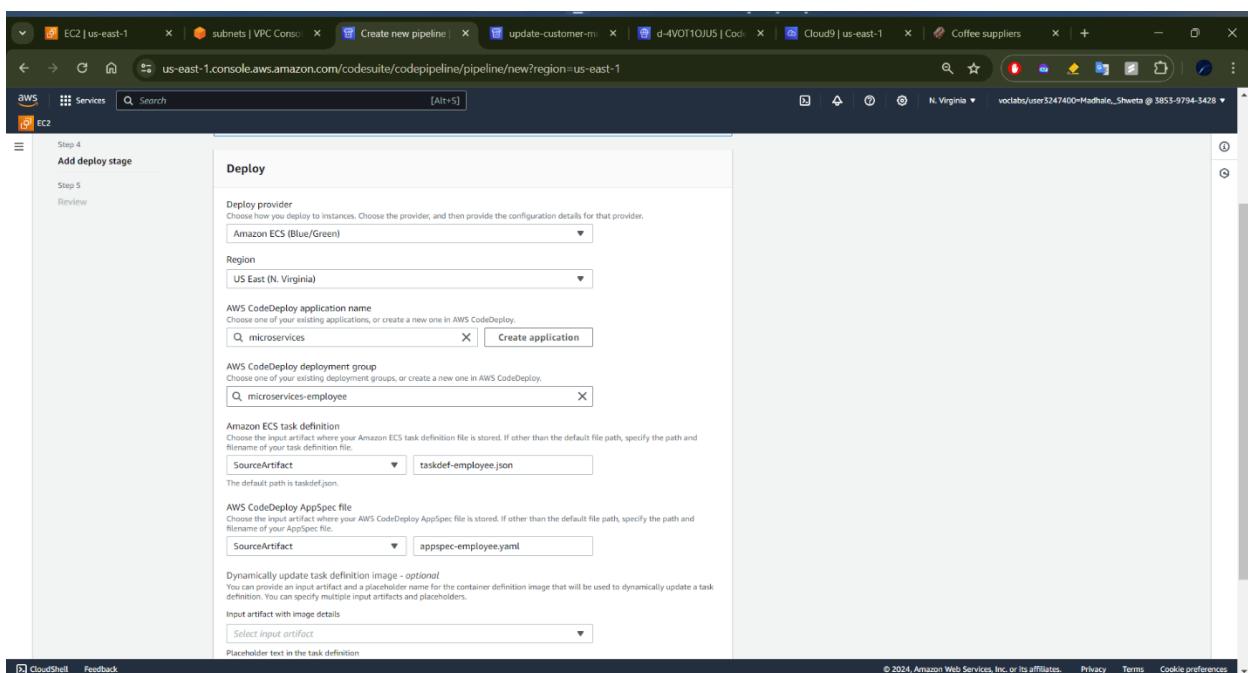
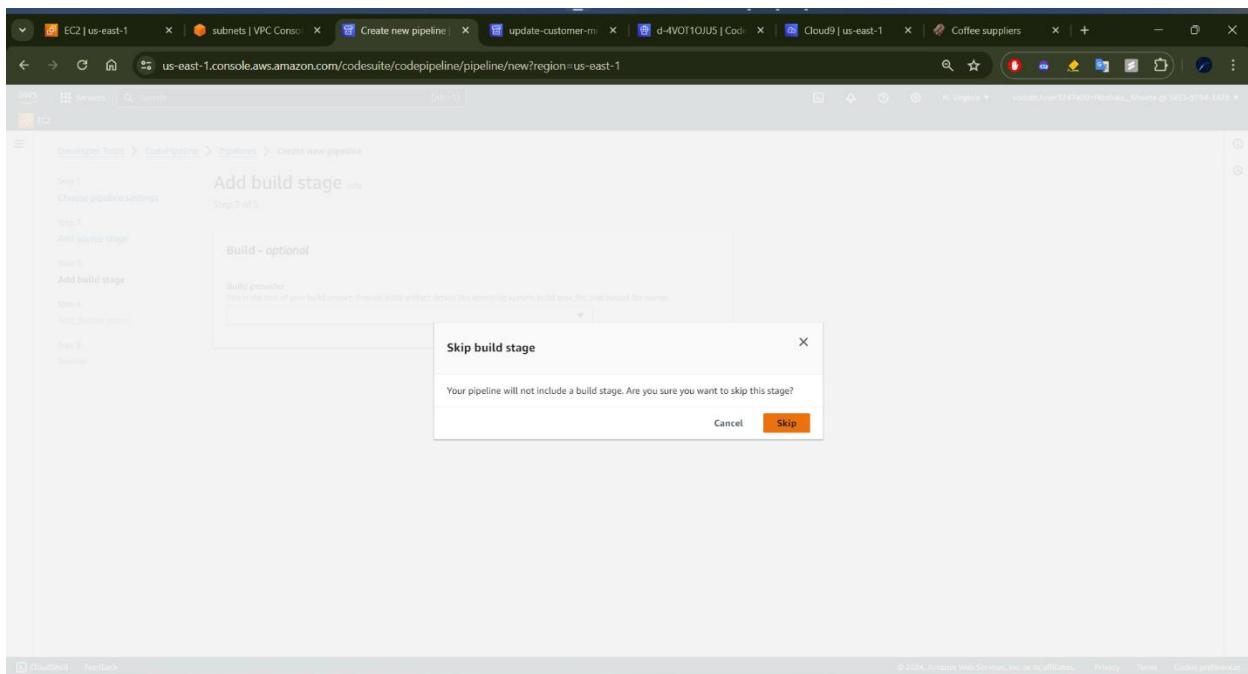
Output artifact format
Choose the output artifact format.

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.

Cancel Previous Next

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



EC2 | us-east-1 | subnets | VPC Console | Edit update-employee | update-customer-m... | d-4VOT1OJUS | CodeDeploy | Cloud9 | us-east-1 | Coffee suppliers | + | - | ×

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

No variables

Edit action

Action name
Choose a name for your action

Action provider
Amazon ECR

Repository name
Choose an Amazon ECR repository as the source location.

Image tag - optional
Choose the image tag that triggers your pipeline when a change occurs in the image repository.

If an image tag is not selected, defaults to latest

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Output artifacts
Choose a name for the output of this action.

No more than 100 characters

Cancel Done

Deploy
Amazon ECS (Blue/Green)

Deploy the latest version of your code

EC2 | us-east-1 | subnets | VPC Console | Edit update-employee | update-customer-m... | d-4VOT1OJUS | CodeDeploy | Cloud9 | us-east-1 | Coffee suppliers | + | - | ×

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

Edit: Triggers

Edit action

Action name
Choose a name for your action

Action provider
Amazon ECS (Blue/Green)

Region
US East (N. Virginia)

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

SourceArtifact

Add

No more than 100 characters

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

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

Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact

The default path is taskdef.json.

AWS CodeDeploy AppSpec file

The screenshot shows the AWS CodePipeline 'Edit update-employee' pipeline configuration page. The pipeline is named 'update-employee-microservice'. The configuration includes:

- Source Action:** AWS Lambda function 'image-employee'.
- Processor Actions:**
 - AWS CodeDeploy application name:** 'microrieservices'.
 - AWS CodeDeploy deployment group:** 'microrieservices-employee'.
 - Amazon ECS task definition:** 'taskdef-employee.json'.
 - AWS CodeDeploy AppSpec file:** 'appspec-employee.yaml'.
- Dynamically update task definition image - optional:** Placeholder text 'IMAGE1_NAME'.
- Variable namespace - optional:** 'DeployVariables'.

Save pipeline changes

Saving your changes cannot be undone. If the pipeline is running when you save your changes, that execution will not complete.

Source action changed

We will update the following resources to detect changes for your updated pipeline.

Change type	Details
Add	Pipeline update-employee-microservice as a target to Amazon CloudWatch Events rule - codepipeline

No resource updates needed for this source action change

Cancel **Save**

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

The screenshot shows the AWS CodePipeline console. A success message at the top indicates "Pipeline was saved successfully". Below it, the pipeline "update-employee-microservice" is listed with a "Pipeline type: V2" and "Execution mode: QUEUED". A "Release change" dialog box is open, prompting the user to release a change. It contains instructions: "Releasing a change will detect the most recent change in each location configured in your source action(s), and run that change through the pipeline. Do you want to continue?". It also includes a note: "► Source revision override A source revision is the version of the source artifact with all the changes to your application code for the pipeline execution." There are "Cancel" and "Release" buttons. The pipeline status shows a "Source" step succeeded and a "Deploy" step failed.

The screenshot shows the AWS CodeDeploy console. A deployment named "d-A2DUOTJUS" is in progress. The "Deployment status" section shows five steps: Step 1 (Deploying replacement task set) completed at 100% (Succeeded); Step 2 (Test traffic route setup) completed at 100% (Succeeded); Step 3 (Rerouting production traffic to replacement task set) completed at 100% (Succeeded); Step 4 (Wait 5 minutes 0 seconds) waiting at 4% (In progress); and Step 5 (Terminate original task set) not started (In progress). The "Traffic shifting progress" section shows "Original" at 0.% and "Replacement" at 100%. The "Deployment details" section lists the deployment ID and application information.

Screenshot of the AWS CodeDeploy console showing the deployment status and details for deployment ID d-A2DUOTJUS.

Deployment Status

The deployment has completed successfully with the following steps:

- Step 1:** Deploying replacement task set (Completed: Succeeded)
- Step 2:** Test traffic route setup (Completed: Succeeded)
- Step 3:** Rerouting production traffic to replacement task set (Completed: Succeeded)
- Step 4:** Wait (Wait completed: Succeeded)
- Step 5:** Terminate original task set (Completed: Succeeded)

Traffic shifting progress:

Original	Replacement
0.%	100.%
Original task set not serving traffic	Replacement task set serving traffic

Deployment Details

Application	Deployment ID	Status
update-employee	d-A2DUOTJUS	Success

Revision Details

Revision location: 0356f303058ad9e3f9a404bedebda8f3afba92e56fdfa83289da94fcff53b
bd5

Revision created: 2 minutes ago

Revision description: Application revision registered by Deployment ID: d-A2DUOTJUS

Task Set Activity

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/1505071584883532768	Replacement	PRIMARY	100%	1	1	0
ecs-svc/4349528712062614841	Original	ACTIVE	0	1	0	0

Deployment Lifecycle Events

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 29, 2024 3:34 PM (UTC-4:00)	Apr 29, 2024 3:34 PM (UTC-4:00)
Install	2 minutes 3 seconds	Succeeded	Apr 29, 2024 3:34 PM (UTC-4:00)	Apr 29, 2024 3:36 PM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 29, 2024 3:36 PM (UTC-4:00)	Apr 29, 2024 3:36 PM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 3:36 PM (UTC-4:00)	Apr 29, 2024 3:36 PM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 3:37 PM (UTC-4:00)	Apr 29, 2024 3:37 PM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:37 PM (UTC-4:00)	Apr 29, 2024 3:37 PM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:37 PM (UTC-4:00)	Apr 29, 2024 3:37 PM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:37 PM (UTC-4:00)	Apr 29, 2024 3:37 PM (UTC-4:00)

Screenshot of the AWS CodePipeline console showing a pipeline named "update-employee-microservice". The pipeline consists of two stages: "Source" and "Deploy".

Source Stage: Pipeline execution ID: d7196ecf-34b9-467a-817b-6e529d6c328f

- Source: AWS CodeCommit (Status: Succeeded)
- Image: Amazon ECR (Status: Succeeded - 8 minutes ago)

Deploy Stage: Pipeline execution ID: d7196ecf-34b9-467a-817b-6e529d6c328f

- Deploy: Amazon ECS (Blue/Green) (Status: Succeeded - Just now)

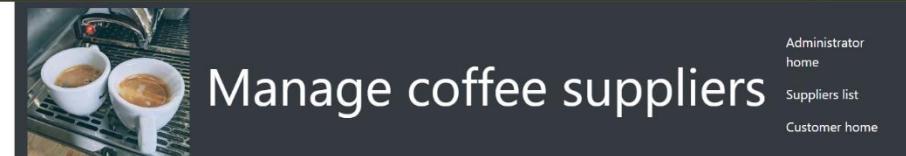
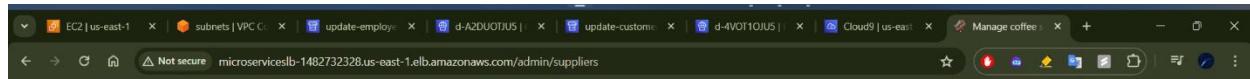
Details for both stages indicate they used the same task definition and AppSpec files for customer and employee microservices, with the image sha256:82933fb795cb.

Screenshot of a web browser showing the URL microserviceslb-1482732328.us-east-1.elb.amazonaws.com/suppliers. The page displays a header image of two coffee cups and the title "Coffee suppliers".

All suppliers

Name	Address	City	State	Email	Phone
shweta madhale	columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

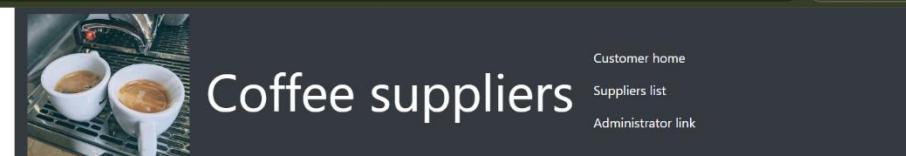
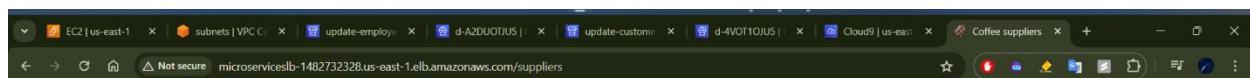
microserviceslb-1482732328.us-east-1.elb.amazonaws.com/admin/suppliers



All suppliers

Name	Address	City	State	Email	Phone
shweta madhale	columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

Add a new supplier



All suppliers

Name	Address	City	State	Email	Phone
shweta madhale	columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708

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

Screenshot of the AWS Cloud Console showing the EC2 Target groups page.

The URL in the address bar is: us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups

The sidebar navigation includes:

- Services
- Search [Alt+S]
- N. Virginia
- voclabs/user3247400-Madhale_Shweta @ 3853-9794-3428

The main content area displays the "Target groups (4) Info" section. A table lists four target groups:

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	microservicesLB	vpc-0a9f6197396de75cc
customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	None associated	vpc-0a9f6197396de75cc
employee-tg-one	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	microservicesLB	vpc-0a9f6197396de75cc
employee-tg-two	arn:aws:elasticloadbalancing:us-east-1:38539794...	8080	HTTP	IP	None associated	vpc-0a9f6197396de75cc

Below the table, a message states: "0 target groups selected". It also says "Select a target group above."

At the bottom right of the page, there are links for "CloudShell", "Feedback", "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

HTTP:80

Details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol: Port
HTTP:80

Load balancer
[microservicesLB](#)

Default actions

Forward to target group

- customer-tg-one 1 (100%)
- Group-level stickiness: Off

Listener ARN
[arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd65d/1647cb55c382ee3b](#)

Rules

Listener rules (2)

Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
-	1	Path Pattern is /admin/*	Forward to target group <ul style="list-style-type: none"> employee-tg-two 100 (100%) Group-level stickiness: Off 	ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> customer-tg-one 1 (100%) Group-level stickiness: Off 	ARN	0 tags

HTTP:8080

Details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol: Port
HTTP: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-1:385397943428:listener/app/microservicesLB/f48a570de49fd65d/5a54f9eca9222b44](#)

Rules

Listener rules (2)

Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
-	1	Path Pattern is /admin/*	Forward to target group <ul style="list-style-type: none"> employee-tg-one 1 (100%) Group-level stickiness: Off 	ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> customer-tg-one 1 (100%) Group-level stickiness: Off 	ARN	0 tags

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

Task 9.1: Limit access to the employee microservice

WhatIsMyIP.com

Search..

Pricing API Sign Up Login Help

What Is My IP? IP Address Lookup IP WHOIS Lookup DNS Lookup Internet Speed Test Tools

What Is My IP?

My Public IPv4: [74.102.91.155](#) ↗

My Public IPv6: Not Detected

My IP Location: Newark, NJ US ↗

My ISP: Verizon Business ↗

What is an IP address? ↘

What is a private IP address? ↘

EC2 | us-east-1 | **Target groups** | **update-employee** | **d-A2D0UJ05** | **update-customer** | **d-4V0T10J5** | **Cloud9 | us-east-1** | **Coffee suppliers**

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ELBListenerV2:loadBalancerArn=arn:aws:elasticloadbalancing:us-east-1:38539794... N. Virginia vocabs/user\$247400+Madhale_Shweta @ 3853-9794-3428

EC2 Services Search [Alt+S]

Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New

Images AMIs AMI Catalog

Elastic Block Store Volumes Snapshots Lifecycle Manager

Network & Security Security Groups Elastic IPs Placement Groups Key Pairs Network Interfaces

Load Balancing Load Balancers Target Groups Trust Stores New

Auto Scaling Auto Scaling Groups

CloudShell Feedback

HTTP:80 Info

Details
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol: Port: HTTP:80 Load balancer: microservicesLB Default actions
Forward to target group
customer-tg-one 1 (100%)
Group-level stickiness: Off

Listener ARN: arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd63d/1647cb55c382ee3b

Rules Tags

Listener rules (1/2) Info
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
-	1	Path Pattern is /admin/*	Forward to target group employee-tg-two 100 (100%) Group-level stickiness: Off	ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group customer-tg-one 1 (100%) Group-level stickiness: Off	ARN	0 tags

Rule limits Actions Add rule
View rule Edit rule Delete rule Reprioritize rules

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

Edit listener | EC2 | **Target groups** | **update-employee** | **d-f1G0H1US** | **update-customer** | **d-4V0T10J5** | **Deployments** | **Microservices** | **Coffee suppliers**

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ElbEditListener:listenerArn=arn:aws:elasticloadbalancing:us-east-1:385397943428... N. Virginia vocabs/user\$247400+Madhale_Shweta @ 3853-9794-3428

EC2 Services Search [Alt+S]

Load balancer details: microservicesLB

Listener details
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Listener ARN: arn:aws:elasticloadbalancing:us-east-1:385397943428:listener/app/microservicesLB/f48a570de49fd63d/1647cb55c382ee3b

Listener configuration
The listener will be identified by the protocol and port.

Protocol: HTTP Port: 80
1-65535

Default actions Info
The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Routing actions
 Forward to target groups Redirect to URL Return fixed response

Forward to target group Info
Choose a target group and specify routing weight or [Create target group](#).

Target group
customer-tg-one Target type: IP, IPv4 **HTTP** **Weight**: **1** **Percent**: **100%**
0-999
[Add target group](#)
You can add up to 4 more target groups.

Group-level stickiness Info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a

CloudShell Feedback

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

Screenshot of the AWS CloudFront console showing the "Define rule conditions" step for a listener rule. The rule conditions are being configured to match requests from a specific IP address (74.102.91.155/32) and path (admin/*).

Define rule conditions

Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.

Listener details: HTTP-80

Add condition

Rule condition types: Route traffic based on the condition type of each request. Each rule can include one or each of the following conditions: host-header, path, http-request-method and source-ip. Each rule can include one or more of each of the following conditions: http-header and query-string.

Source IP

Is 74.102.91.155/32

Add new value

You can add up to 3 more condition values for this rule.

Cancel **Confirm**

Screenshot of the AWS CloudFront console showing the successful modification of the listener rule. The rule now includes two conditions: Source IP (74.102.91.155/32) and Path (/admin/*).

Successfully modified listener.

Step 1 Define rule conditions

Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.

Listener details: HTTP-80

Conditions (2)

Source IP (1)

If
Source IP
Is
74.102.91.155/32

Path (1)

If
Path
Is
/admin/*

Cancel **Next**

Load balancer details: microservicesLB

Listener details
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Listener ARN
arn:aws:elasticloadbalancing:us-east-1:1385397943428:listener/app/microservicesLB/148a570de49fd65d/3a54f9ec9222b44

Listener configuration
The listener will be identified by the protocol and port.

Protocol HTTP
Port 8080
Used for connections from clients to the load balancer.
1-65535

Default actions info
The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Routing actions
 Forward to target groups Redirect to URL Return fixed response

Forward to target group info
Choose a target group and specify routing weight or [Create target group](#).

Target group	Weight	Percent
customer-tg-two	1	100%
Target type: IP, IPv4	0.999	

[Add target group](#)
You can add up to 4 more target groups.

Group-level stickiness info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.

[Turn on group-level stickiness](#)

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

Successfully modified listener.

Step 1 Define rule conditions
These actions will be applied to requests matching the rule conditions.

Step 2 Define rule actions

Step 3 Review changes

Define rule actions info
Actions

Action types
Routing actions

Forward to target groups Redirect to URL Return fixed response

Forward to target group info
Choose a target group and specify routing weight or [Create target group](#).

Target group	Weight	Percent
employee-tg-two	1	100%
Target type: IP, IPv4	0.999	

[Add target group](#)
You can add up to 4 more target groups.

Group-level stickiness info
If a target group is sticky, requests routed to it remain in that target group for the duration of the session. Individual target stickiness is a configuration of the target group.

Turn on group-level stickiness

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

Rules Tags

Listener rules (2) info
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Filter rules

Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
-	1	<ul style="list-style-type: none"> Path Pattern is /admin/*, AND Source IP is 74.102.91.155/32 	Forward to target group <ul style="list-style-type: none"> employee-tg-one 1 (100%) Group-level stickiness: Off 	Edit ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> customer-tg-one 1 (100%) Group-level stickiness: Off 	Edit ARN	0 tags

Rule limits Actions Add rule © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS Cloud Console interface for managing an Application Load Balancer (ALB). The left sidebar navigation includes 'EC2 Dashboard', 'Instances', 'Images', 'Elastic Block Store', and 'Network & Security'. The main content area displays the 'HTTP:8080' listener configuration under 'Load balancers > microservicesLB > HTTP:8080 listener'. The 'Details' tab shows the protocol as 'HTTP:8080' and the load balancer as 'microservicesLB'. The 'Default actions' section indicates traffic is forwarded to the 'customer-tg-two' target group at 100% weight. The 'Listener rules (2)' tab lists two rules: one for '/admin/*' with conditions 'Path Pattern is /admin/*, AND Source IP is 74.102.91.155/32' and another for 'Default' (last, default) with the condition 'If no other rule applies'. Both rules forward traffic to the 'customer-tg-two' target group at 100% weight.

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

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

```

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

```

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

```

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

```

```

voclabs:~/environment $ docker rm -f employee_1
employee_1
voclabs:~/environment $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
[+] Building 3.0s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 229B
=> [internal] load metadata for docker.io/library/node:11-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c0ff820f1a55462890caf80f57fe3b91f5f86945a4d505932
=> [internal] load build context
=> => transferring context: 264.4kB
=> CACHED [2/5] RUN mkdir -p /usr/src/app
=> CACHED [3/5] WORKDIR /usr/src/app
=> [4/5] COPY . .
=> [5/5] RUN npm install
=> exporting to image
=> => exporting layers
=> writing image sha256:e52e6550dc16bcc2c07d77c8be66dc395544953844180d174ba77872131e66a54
=> naming to docker.io/library/employee
voclabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/employee/app/config/config.js | grep 'APP_DB_HOST' | cut -d '' -f2)
voclabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint
supplierdb.ccaycjdtups.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/employee (dev) $ account_id=$(aws sts get-caller-identity |grep Account|cut -d '' -f4)
voclabs:~/environment/microservices/employee (dev) $ echo $account_id
385397943428
voclabs:~/environment/microservices/employee (dev) $ docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
voclabs:~/environment/microservices/employee (dev) $ 

```

```

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

Login Succeeded
voclabs:~/environment/microservices/employee (dev) $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
The push refers to repository [385397943428.dkr.ecr.us-east-1.amazonaws.com/employee]
2100f3e094c4: Pushed
385e11831471: Pushed
5f70bf18a066: Layer already exists
6f8e0edab334: Layer already exists
d41d71533007: Layer already exists
1dc7f3bbb994: Layer already exists
dcaceb729824: Layer already exists
fb5b933fe405: Layer already exists
latest: digest: sha256:bc468438c549daa302ff72a107e424af0fb6afa9bc75db55d3e32f9f901fa618 size: 1989
voclabs:~/environment/microservices/employee (dev) $ 

```

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

The screenshot shows the AWS CodePipeline console with the pipeline named "update-employee-microservice". The pipeline type is V2 and the execution mode is QUEUED. The pipeline has two stages: Source and Deploy. The Source stage is configured to use AWS CodeCommit and Amazon ECR, with a recent execution showing a success status. The Deploy stage is configured to use Amazon ECS (Blue/Green), also showing a success status. A green checkmark icon is present on the right side of the pipeline interface.

Listener details | Target groups | update-employee | d-TH1EPZKU5 | update-customer | Deployments | MicroservicesID | Coffee suppliers | + us-east-1.console.aws.amazon.com/codesuite/codedeploy/deployments/d-TH1EPZKU5?region=us-east-1

AWS Services Search [Alt+S] N. Virginia v voblasts/user\$247400-Madhale_Shweta @ 3853-9794-3428

Developer Tools CodeDeploy

Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments Deployment
Applications
Deployment configurations
On-premises Instances
Pipeline • CodePipeline
Settings

Go to resource Feedback

d-TH1EPZKU5

Deployment status

Step 1: Deploying replacement task set
Completed Succeeded 100%

Step 2: Test traffic route setup
Completed Succeeded 100%

Step 3: Rerouting production traffic to replacement task set
100% traffic shifted Succeeded 100%

Step 4: Wait
Wait completed Succeeded 100%

Step 5: Terminate original task set
Completed Succeeded 100%

Traffic shifting progress

Original: 0% Original task set not serving traffic

Replacement: 100% Replacement task set serving traffic

Copy deployment Retry deployment

Deployment details

Application: microservices Deployment ID: d-TH1EPZKU5 Status: Succeeded
Deployment configuration: Deployment group: initiated by:

CloudShell Feedback

Listener details | Target groups | update-employee | d-TH1EPZKU5 | update-customer | Deployments | MicroservicesID | Coffee suppliers | + us-east-1.console.aws.amazon.com/codesuite/codedeploy/deployments/d-TH1EPZKU5?region=us-east-1

AWS Services Search [Alt+S] N. Virginia v voblasts/user\$247400-Madhale_Shweta @ 3853-9794-3428

Developer Tools CodeDeploy

Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments Deployment
Applications
Deployment configurations
On-premises Instances
Pipeline • CodePipeline
Settings

Go to resource Feedback

Revision details

Revision location: 8167c4b50c780d17a28d3ed8084abc899171d941dff86248d796dbe82ccb4c1 Revision created: 7 minutes ago Revision description: Application revision registered by Deployment ID: d-TH1EPZKU5

Task set activity

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/1505071584883532768	Original	ACTIVE	0	1	1	0
ecs-svc/8222970165060292878	Replacement	PRIMARY	100%	1	1	0

Deployment lifecycle events

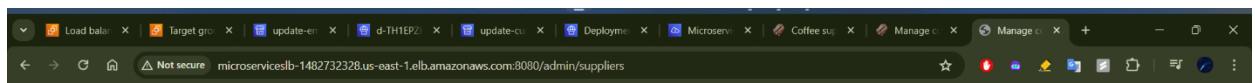
Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 29, 2024 4:14 PM (UTC-4:00)	Apr 29, 2024 4:14 PM (UTC-4:00)
Install	2 minutes 23 seconds	Succeeded	Apr 29, 2024 4:14 PM (UTC-4:00)	Apr 29, 2024 4:16 PM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 29, 2024 4:16 PM (UTC-4:00)	Apr 29, 2024 4:16 PM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 4:16 PM (UTC-4:00)	Apr 29, 2024 4:16 PM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 4:16 PM (UTC-4:00)	Apr 29, 2024 4:16 PM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 29, 2024 4:16 PM (UTC-4:00)	Apr 29, 2024 4:16 PM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 29, 2024 4:16 PM (UTC-4:00)	Apr 29, 2024 4:16 PM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 29, 2024 4:16 PM (UTC-4:00)	Apr 29, 2024 4:16 PM (UTC-4:00)

CloudShell Feedback

Task 9.4: Test access to the employee microservice

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

Name	Address	City	State	Email	Phone
shweta madhale	columbia ave	Jersey City	New Jersey	smadhale@stevens.edu	5513285708



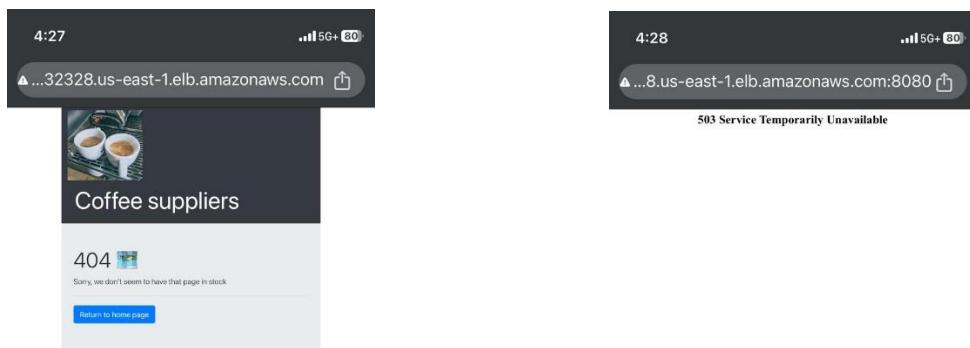
All suppliers

Name	Address	City	State	Email	Phone
------	---------	------	-------	-------	-------

shweta madhale columbia ave Jersey City New Jersey smadhale@stevens.edu 5513285708 [edit](#)

[Add a new supplier](#)

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



Task 9.5: Scale the customer microservice

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

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

The screenshot shows the AWS Elastic Container Service (ECS) Cluster overview page for the cluster `microservices-serverlesscluster`. The cluster ARN is listed as `arn:aws:ecs:us-east-1:1585597943428:cluster/microservices-serverlesscluster`. The status is Active, and CloudWatch monitoring is set to Default. There are no registered container instances. Under the Services section, there are two entries: `customer-microservice` and `employee-microservice`, both of which are active and have a service type of REPLICA. The `customer-microservice` has 2/3 tasks running, while the `employee-microservice` has 1/1 tasks running. The Tasks section shows pending tasks for the `customer-microservice`.

This screenshot shows the same AWS ECS Cluster overview page as the previous one, but it includes a third service entry in the Services list. The new service is listed as `customer-microservice` with an ARN of `arn:aws:ecs:us-east-1:1585597943428:service/customer-microservice`, status Active, and service type REPLICA. It has 3/3 tasks running. The other two services, `customer-microservice` and `employee-microservice`, remain active with their respective task counts of 2/3 and 1/1.

Analysis

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

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

- Analysis

This phase involves creating an architecture diagram and cost estimate. This ensures a better understanding of the infrastructure for everyone involved. This can facilitate analysis of the infrastructure before implementing and make necessary changes where required. Also, the cost estimate can help evaluate the financial scope of the project and budgeting around it.

This ensures that informed decisions are made regarding aspects of the project like resource allocation. This phase sets up a direction for the entire project and helps ensure that the solution is technically and financially feasible and well-defined to get the desired outcome.

- Significance

The architecture diagram is significant in the project to have a visual representation of the proposed project implementation. It helps in understanding the various components within the project, their connections, how they work together to come to the solution. Whereas cost estimation is important for setting budgets and other planning. The architecture and cost estimation will ensure that the project will give desired outcomes within the proposed infrastructure efficiently and within the budget.

- Outcome

The outcome of this phase involves the planning before the implementation of the project. This phase will give an architecture diagram which will help understand the infrastructure involving various resources and services within AWS. The cost estimate gives an estimate of the expenses which will be incurred for this project's execution.

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

- Analysis

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

- Significance

This phase plays significant role in understanding the current infrastructure and functionality of the monolithic application. It helped in understanding the various issues involving this

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

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

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

- Analysis

This phase involved creating a development environment for the project using the AWS Cloud9 Service. Also, using the AWS CodeCommit which is a git-compatible repository to check the application code into. A development environment named MicroservicesIDE was created and configured with an EC2 instance which runs Amazon Linux2 in the specified VPC and subnet. Then the source code of the monolithic application was copied into the Cloud9 environment. Also, various directories for the customer and employee microservices were configured which prepared the environment to split the monolithic application into microservices. Finally, a git repository for these microservices was created using the CodeCommit. The original source code along with the modified code were pushed to this repository.
- Significance

This phase of development helped in creating a development environment and setting up a git repository which will help in version control. The development environment was created with the Cloud9 service which provided a workspace for the project development. The CodeCommit service was used to set up the repository which helped in version control and in general code management throughout the project.
- Outcome

This phase establishes a development environment and sets up a version control for the application code which will be the foundation of the upcoming tasks.

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

- Analysis

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

- This phase mainly played a significant role in splitting the monolithic application into employee and customer microservices and testing these microservices within the docker containers. Microservices play important roles for portability, dependability, for maintaining and scaling. The docker containers provide a lightweight portable environment for running the applications. This ensured consistent and efficient deployment of the microservices across the environments,
- Outcome
In this phase the monolithic application was successfully split into two microservices employee and customer. Then docker containers were created for both the microservices and were tested which help in checking the feasibility of running the application in a containerized environment. Also the port configurations were adjusted as per the deployment requirements and all the changes were successfully committed and pushed to CodeCommit.

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

- Analysis
In this phase, deployment of the microservices-based web application to Amazon ECS (Elastic Container Service) with Fargate, integration with CI/CD (Continuous Integration/Continuous Deployment) pipelines was done. With microservices, the deployment process becomes simpler and more efficient. Amazon ECS was used with fargate which made it easier to implement the underlying infrastructure with a serverless container. The integration of CI/CD pipelines automated the building, testing, and deployment processes which ensured consistent and reliable deployment. Firstly the latest docker images of the two microservices were uploaded to two separate Amazon ECR repositories. Then an ECS cluster was created which was serverless AWS fargate cluster configured for the specified VPC and subnets. Also, another code repository was created just for the deployment process. Then task definition for each of the microservices was created and registered to the Amazon ECS. Also, application specific files were created for each microservice to provide instructions regarding deployment of the microservice to the Amaozen ECS on fargate infrastructure which were provided to CodeDeploy.
- Significance
This phase plays a very crucial stage in the project because of the deployment of the microservices based application with help of Amazon ECS with fargate. By breaking down the monolithic application into smaller independently deployable services the deployment became more flexible and easier. The containerization with docker also ensured consistency with deployment of each microservices and helped to streamline the entire process.
- Outcome
The docker clients were connected to the Amazon ECR service. Separate driver repositories were created for each micro service and appropriate permissions were set. The docker images for each microservice were pushed to Amazon ECR. A serverless AWS fargate cluster was created. Code commit repository was created to store the deployment files. Task definition files were created for each microservice, and they were registered. Application specification files were also created for each of the microservice. In this phase, an efficient deployment pipeline supported by the AWS services was created. This includes creation and configuration of Amazon ECR repositories for storing the docker images of the micro services. Tagging and uploading of the docker images to the ECR to make them available for deployment was done. Also, an ECS cluster was created which enabled managing the containerized application, enhancing the scalability and resource utilization.

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

- Analysis

This task involved creating an application load balancer which eventually provided an endpoint URL. This URL will subsequently behave as the HTTPS entry point for the users – customers and employees. This load balancer was configured with listeners who were in turn configured with routing and access rules for different target groups for directing requests. With this phase the web application can handle a lot of traffic without crashing. Services like EC2LB and security groups are well used to implement scalable and highly available architecture. The use of path-based routing rules enhanced flexibility and control over traffic routing.

- Significance

In this phase the application load balancer and the target groups were created. This step is very pivotal as the ALB will serve as the entry point for the incoming traffic and it provides a single HTTPS endpoint for the customers and the employees to access the application. Also, traffic is distributed across the multiple target groups which will ensure that the ALB will have high availability and will direct request to healthy instances. Also, use of the blue-green deployment strategies with multiple target groups will help and seamless update and rollback capabilities and will minimize the impact on the user during the deployment.

- Outcome

In this phase four target groups were created which will aid in facilitating efficient routing of traffic. Each of the target groups is configured with specific health checks and routing rules. The load balancer was provisioned with two listeners, one on the port 80 and another on port 8080 which will help handle incoming HTTP traffic. This will help in routing the request to the appropriate target group and will ensure distributing the workload across various instances.

Phase 7. Create ECS services.

- Analysis

This phase involves creating Amazon ECS for each of the microservices which makes it easier to manage the microservices independently as each is deployed to its own ECS service. This will enhance the maintenance and scalability of the application. This also will decrease the dependence between each of the components. Also, with the use of ECS load balancers and task definitions, it is easier to manage the applications and scale while maintaining optimal performance and resource utilization.

- Significance

This phase is very significant because it helps in managing and scaling the microservices independently. By having two dedicated services for each of the micro services it is ensured that changing or updating one microservice will not affect the other. This also is useful in the case of troubleshooting or debugging so each of the micro services is independent and can be managed on its own.

- Outcome

In this phase Amazon ECS services, one for the customer microservice and another for the employee microservice were created. Each of the servers was configured with its own task definition, some network settings, and load balancer configuration. Because of separate services it is possible to deploy scale and monitor each of the microservice independently

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

- Analysis

In this task, a CI/CD pipeline was defined which was used to deploy the application.

Configuring code deploy and code pipeline the development and deployment workflow is automated. With the blue/green deployment the deployment of updates is possible seamlessly and has minimal impact on the production environment. Integration of the AWS services used previously like ECS, code commit, an ECR provided scalable solution for managing the entire software lifecycle.

- Significance

In this phase the configuration of the code deployment code pipeline was done which played a significant role in establishing the CI/CD pipeline for deploying and managing each of the microservices. Because of the automation of the deployment process the deployment cycle became very consistent and easier. Also the code deploys blue-green deployment strategy enhanced the reliability by minimizing the downtime and by enabling an easy rollback if there are any deployment issues.

- Outcome

A CodeDeploy application was created which is a collection of deployment groups and revisions. This application was created by specifying the earlier created loadbalancer, ECS services, listener, target groups. Two deployment groups for customer and employee were created in this application with the specified configuration. A pipeline was created, one each for employee and customer with CodeCommit as source and CodeDeploy responsible for deployment. Amazon ECS blue/green deployment was used to test the new application version and to send production traffic to it. Each of the CI/CD pipelines were tested as well. Also, changes to the load balancer listener rules from CodeDeploy were observed.

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

- Analysis

In this phase, the benefits for the microservices environment were understood. There were some changes in configuration of the listener rules and source code. Also, a new docker image was generated and pushed to the ECR which automatically caused the pipeline to run and update the deployment. Also, scaling the number of containers was possible to support the customer microservices. Limiting the IP addresses the security and control over sensitive features voice insured. Also there were some changes in the UI of just one microservice which didn't affect the other microservice. Also automation of the docker image update and the deployment via the CI CD pipeline was possible. Finally the ability to scale the customer microservice independently was done which demonstrated the elasticity benefits and the scalability benefits of the micro services which allowed efficient resource utilization and responds to a changing or dynamic workload demand.

- Significance

This phase played a significant role in understanding the practical application of the microservices architectures and how the CI/CD pipelines play a role in it. The microservice source code was reconfigured and its independence with each other was understood. Also, the trigger of the automated pipeline execution from docker image was observed. This phase helped understand the benefits of micro services to enable independence in deployment,

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