

# COURSE PROJECT

## Building Microservices and a CI/CD Pipeline with AWS

NAME: SUPREETH HASSAN RAVINDRA

CWID: 20020533

### PART-1: PROJECT EXECUTION

### PHASE-2: Analyzing the infrastructure of the monolithic application

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

The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' services: Simple Queue Service, CloudFormation, S3, Billing and Cost Management, Resource Groups & Tag Editor, and EC2. Below this is a 'View all services' link. In the center, the 'Applications' section is displayed, showing '0' applications under 'us-east-1 (Current Region)'. It includes a 'Create application' button and a 'Find applications' search bar. At the bottom of this section is a 'Create application' button and a 'Go to myApplications' link. The right side of the page has sections for 'Welcome to AWS', 'AWS Health', and 'Cost and usage'.

The screenshot shows the AWS EC2 Resources page. The top section displays EC2 resource statistics: 1 running instance, 0 auto scaling groups, 0 dedicated hosts, 0 elastic IPs, 2 instances, 1 key pair, 0 placement groups, 4 security groups, 0 load balancers, 0 snapshots, and 1 volume. To the right, the 'Account attributes' section shows the Default VPC (vpc-06d8cb13203fc000). Below this are sections for 'Settings', 'Data protection and security', 'Zones', 'EC2 Serial Console', 'Default credit specification', and 'Console experiments'. The bottom section contains links for 'Launch instance', 'Service health', 'Explore AWS', and promotional content for AWS Graviton2 and cost reduction.

S | Services | Search | [Alt+S] | N. Virginia | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

### Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive) | All states

Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP
<input checked="" type="checkbox"/> MonolithicApp...	i-0dd0ffc76c2f1809f	<span>Running</span>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms</a>	us-east-1a	ec2-3-82-198-105.com...	3.82.198.105

**i-0dd0ffc76c2f1809f (MonolithicAppServer)**

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

▼ Instance summary [Info](#)

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0dd0ffc76c2f1809f (MonolithicAppServer)	<a href="#">3.82.198.105   open address</a>	<a href="#">10.16.10.85</a>
IPv6 address	Instance state	Public IPv4 DNS
-	<span>Running</span>	<a href="#">ec2-3-82-198-105.compute-1.amazonaws.com   open address</a>
Hostname type	Private IP DNS name (IPv4 only)	
IP name: ip-10-16-10-85.ec2.internal	<a href="#">ip-10-16-10-85.ec2.internal</a>	

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

S | Services | Search | [Alt+S] | N. Virginia | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

### Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive) | All states

Instance state = running | Clear filters

Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP
i-0dd0ffc76c2f1809f	<span>Running</span>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms</a>	us-east-1a	ec2-3-82-198-105.compute-1.amazonaws.com	3.82.198.1

**i-0dd0ffc76c2f1809f (MonolithicAppServer)**

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

▼ Instance summary [Info](#)

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0dd0ffc76c2f1809f (MonolithicAppServer)	<a href="#">3.82.198.105   open address</a>	<a href="#">10.16.10.85</a>
IPv6 address	Instance state	Public IPv4 DNS
-	<span>Running</span>	<a href="#">ec2-3-82-198-105.compute-1.amazonaws.com   open address</a>
Hostname type	Private IP DNS name (IPv4 only)	
IP name: ip-10-16-10-85.ec2.internal	<a href="#">ip-10-16-10-85.ec2.internal</a>	

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1> | © 2024, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences

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



# Monolithic Coffee suppliers

Home  
Suppliers  
list

## Welcome

Use this app to keep track of your coffee suppliers

[List of suppliers](#)

The screenshot shows a browser window with multiple tabs open. The active tab is titled "ec2-3-82-198-105.compute-1.amazonaws.com/suppliers". The page content is identical to the first screenshot, displaying the "Monolithic Coffee suppliers" logo, navigation links, and the "All suppliers" heading.

## All suppliers

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

[Add a new supplier](#)

The screenshot shows a Windows desktop environment. The taskbar at the bottom includes icons for search, file explorer, and various applications. The system tray on the right shows the date and time (12:24 PM, 4/25/2024), battery level (11°C), and connectivity status.

The screenshot shows a browser window with the URL "ec2-3-82-198-105.compute-1.amazonaws.com/supplier-add". The page displays the "Monolithic Coffee suppliers" logo and navigation links. A message "All fields are required" is displayed above the form fields. The form consists of four text input fields: Name, Address, City, and State, each with a placeholder and a descriptive label below it.

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

The screenshot shows a Windows desktop environment. The taskbar at the bottom includes icons for search, file explorer, and various applications. The system tray on the right shows the date and time (12:25 PM, 4/25/2024), battery level (11°C), and connectivity status.



# Monolithic Coffee suppliers

[Home](#)  
[Suppliers list](#)

All fields are required

Name

Supreeth H R

Name of this supplier

Address

56 Poplar Street, Apt #2

Address for this supplier

City

Jersey City

City for this supplier

State

New Jersey

State for this supplier



# Monolithic Coffee suppliers

[Home](#)  
[Suppliers list](#)

## All suppliers

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

[Add a new supplier](#)



# Monolithic Coffee suppliers

[Home](#)  
[Suppliers list](#)

## All suppliers

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

[Add a new supplier](#)

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

The screenshot shows the AWS EC2 Instance Connect interface. At the top, there's a navigation bar with 'Services' selected, followed by a search bar and a 'Connect' button. Below the navigation bar, the path 'EC2 > Instances > i-0dd0ffc76c2f1809f > Connect to instance' is displayed. The main content area is titled 'Connect to instance' with an 'Info' link. It says 'Connect to your instance i-0dd0ffc76c2f1809f (MonolithicAppServer) using any of these options'. There are four tabs at the top of this section: 'EC2 Instance Connect' (selected), 'Session Manager', 'SSH client', and 'EC2 serial console'. Under 'Connection Type', two options are shown: 'Connect using EC2 Instance Connect' (selected) and 'Connect using EC2 Instance Connect Endpoint'. The 'EC2 Instance Connect' option is described as 'Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.' The 'EC2 Instance Connect Endpoint' option is described as 'Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.' Below these options, the 'Public IP address' is listed as '3.82.198.105'. The 'Username' field contains 'ubuntu'. A note below the username field states: 'Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.' At the bottom of the page, there are links for 'CloudShell', 'Feedback', '© 2024, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

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

The screenshot shows a terminal session on an Ubuntu system. The terminal window title is 'i-0dd0ffc76c2f1809f (MonolithicAppServer)'. The session starts with system information: 'System information as of Thu Apr 25 16:29:25 UTC 2024'. It lists system load (0.56), processes (104), usage of / (30.0% of 7.69GB), users logged in (0), memory usage (28%), and swap usage (0%). It then displays update information: '300 updates can be installed immediately. 212 of these updates are security updates. To see these additional updates run: apt list --upgradable'. Following this, there's a warning about system restart: '\*\*\* System restart required \*\*\*'. The next part of the terminal shows the copyright notice for the Ubuntu system. Below that, it states: 'Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.' It then provides instructions for running commands as root: 'To run a command as administrator (user "root"), use "sudo <command>". See "man sudo\_root" for details.' The final line shows the prompt 'ubuntu@ip-10-16-10-85:~\$'. At the bottom of the terminal window, there's a status bar with 'Public IPs: 3.82.198.105 Private IPs: 10.16.10.85'. The bottom of the screen also features the same navigation and footer links as the first screenshot.

```

aws | Services | Search | [Alt+S] | ⓘ ⓘ ⓘ ⓘ ⓘ ⓘ N. Virginia | vclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250 ▾
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1038-aws x86_64)

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

System information as of Thu Apr 25 23:32:26 UTC 2024

System load: 0.2 Processes: 105
Usage of /: 30.7% of 7.69GB Users logged in: 0
Memory usage: 26% IPv4 address for eth0: 10.16.10.85
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

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr 25 16:29:26 2024 from 18.206.107.28
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-16-10-85:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 454 root 18u IPv6 20957 0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-85:~$ 
```

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

```

aws | Services | Search | [Alt+S] | ⓘ ⓘ ⓘ ⓘ ⓘ ⓘ N. Virginia | vclabs/user3233567=Hassan_Ravindra,_Supreeth @ 5797-9441-6250 ▾
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

System information as of Thu Apr 25 23:32:26 UTC 2024

System load: 0.2 Processes: 105
Usage of /: 30.7% of 7.69GB Users logged in: 0
Memory usage: 26% IPv4 address for eth0: 10.16.10.85
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

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr 25 16:29:26 2024 from 18.206.107.28
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-16-10-85:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 454 root 18u IPv6 20957 0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-85:~$ ps -ef | head -1; ps -ef | grep node
UID PID PPID C STIME TTY TIME CMD
root 425 419 0 23:27 ? 00:00:00 sudo node index.js
root 454 425 0 23:27 ? 00:00:00 node index.js
ubuntu 1289 1269 0 23:34 pts/0 00:00:00 grep --color=auto node
ubuntu@ip-10-16-10-85:~$ 
```

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

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

System information as of Thu Apr 25 23:32:26 UTC 2024

```
System load: 0.2      Processes:          105
Usage of /: 30.7% of 7.69GB  Users logged in:    0
Memory usage: 26%        IPv4 address for eth0: 10.16.10.85
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

New release '22.04.3 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr 25 16:29:26 2024 from 18.206.107.28  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo\_root" for details.

```
ubuntu@ip-10-16-10-85:~$ sudo lsof -i :80
COMMAND PID USER FD   TYPE DEVICE SIZE/OFF NODE NAME
node    454 root  18u  IPv6  20957      0t0  TCP *:http (LISTEN)
ubuntu@ip-10-16-10-85:~$ ps -ef | head -1; ps -ef | grep node
UID      PID  PPID C STIME TTY      TIME CMD
root      425     1 23:27 ?    00:00:00 sudo node index.js
root      454     425 23:27 ?    00:00:00 node index.js
ubuntu   1289    1269 23:34 pts/0    00:00:00 grep --color=auto node
ubuntu@ip-10-16-10-85:~$ cd ~/resources/codebase_partner
ubuntu@ip-10-16-10-85:~/resources/codebase_partner$ ls
app index.js node_modules package-lock.json package.json public views
ubuntu@ip-10-16-10-85:~/resources/codebase_partner$ 
```

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

Amazon RDS

Introducing Aurora I/O-Optimized

Aurora's I/O-Optimized is a new cluster storage configuration that offers predictable pricing for all applications and improved price-performance, with up to 40% cost savings for I/O-intensive applications.

RDS > Databases

Consider creating a Blue/Green Deployment to minimize downtime during upgrades

You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. [RDS User Guide](#) [Aurora User Guide](#)

DB identifier	Status	Role	Engine	Region & AZ	Size	Recommendations	CPU
supplierdb	Available	Instance	MySQL Community	us-east-1b	db.t3.micro	5 Informational	

Events

https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1

Screenshot of the AWS RDS console showing the connectivity & security details for a MySQL database.

**Amazon RDS**

CPU: 2.74% | Class: db.t3.micro | Current activity: 0 Connections | Region & AZ: us-east-1b

**Connectivity & security**

Endpoint & port	Networking	Security
Endpoint: supplierdb.ccylnx0fhcyf.us-east-1.rds.amazonaws.com	Availability Zone: us-east-1b	VPC security groups: DBSecurityGroup (sg-08610c3836cbe7923)   Active
Port: 3306	VPC: LabVPC (vpc-0930b2fdf65345dee)	Publicly accessible: No
	Subnet group: c110323a2605598l6490453t1w579 794416250-dbsubnetgroup-fe4yti7tjcgt	Certificate authority: rds-ca-rsa2048-g1
	Subnets: subnet-0276819c80747b5c2 subnet-0a2b274d88b091aa5	Certificate authority date: May 25, 2061, 19:34 (UTC-04:00)
		DB instance certificate expiration date

Events

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

- Using the pre-installed MySQL client on the MonolithicAppServer instance, I connected to the RDS database with the following values:

**Username: admin**

**Password: lab-password**

- I ran SQL commands to check the data in the COFFEE database specifically the suppliers table where I found the supplier entries added earlier.

```
To see these additional updates run: apt list --upgradable

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr 25 16:29:26 2024 from 18.206.107.28
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-16-10-85:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 454 root 18u IPv6 20957 0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-85:~$ ps -ef | head -1; ps -ef | grep node
UID      PID  PPID C STIME TTY          TIME CMD
root      425     1  0 23:27 ?        00:00:00 sudo node index.js
root      454     425  0 23:27 ?        00:00:00 node index.js
ubuntu    1289    1269  0 23:34 pts/0   00:00:00 grep --color=auto node
ubuntu@ip-10-16-10-85:~$ cd ~/resources/codebase_partner
ubuntu@ip-10-16-10-85:~/resources/codebase_partner$ ls
app index.js node_modules package-lock.json package.json public views
ubuntu@ip-10-16-10-85:~/resources/codebase_partner$ nmap -Pn supplierdb.ccylnx0fhcyf.us-east-1.rds.amazonaws.com
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-25 23:45 UTC
Nmap scan report for supplierdb.ccylnx0fhcyf.us-east-1.rds.amazonaws.com (10.16.40.179)
Host is up (0.00055s latency).
rDNS record for 10.16.40.179: ip-10-16-40-179.ec2.internal
Not shown: 999 filtered ports
PORT      STATE SERVICE
3306/tcp  open  mysql

Nmap done: 1 IP address (1 host up) scanned in 8.16 seconds
ubuntu@ip-10-16-10-85:~/resources/codebase_partner$
```

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

System information as of Fri Apr 26 00:29:00 UTC 2024

```
System load: 0.0      Processes:          102
Usage of /: 30.7% of 7.69GB  Users logged in:    0
Memory usage: 27%        IPv4 address for eth0: 10.16.10.85
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

New release '22.04.3 LTS' available.  
 Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr 25 23:32:27 2024 from 18.206.107.29  
 ubuntu@ip-10-16-10-85:~\$ mysql -h supplierdb.ccylnx0fhcylf.us-east-1.rds.amazonaws.com -P 3306 -u admin -p  
 Enter password:  
 Welcome to the MySQL monitor. Commands end with ; or \g.  
 Your MySQL connection id is 124  
 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> [REDACTED]

[CloudShell](#) [Feedback](#) © 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

Last login: Fri Apr 26 00:33:26 2024 from 18.206.107.28  
 ubuntu@ip-10-16-10-85:~\$ mysql -h supplierdb.ccylnx0fhcylf.us-east-1.rds.amazonaws.com -P 3306 -u admin -p  
 Enter password:  
 Welcome to the MySQL monitor. Commands end with ; or \g.  
 Your MySQL connection id is 129  
 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> Select \* from suppliers;  
 ERROR 1046 (3D000): 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> Select \* from suppliers;  

id   name	address	city	state	email	phone
1   Supreeth Coffee   56 Poplar Street, Apt #2   Jersey City   New Jersey   shassanr@stevens.edu   2012047263					

 1 row in set (0.00 sec)

mysql> [REDACTED]

[CloudShell](#) [Feedback](#) © 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

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

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

AWS Cloud9

# AWS Cloud9

## A cloud IDE for writing, running, and debugging code

AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.

### How it works

Create an AWS Cloud9 development environment on a new Amazon EC2 instance or connect it to your own Linux server through SSH. Once you've created an AWS Cloud9 environment, you will have immediate access to a rich code editor, integrated debugger, and built-in terminal with pre-configured AWS CLI – all within your browser. Using the AWS Cloud9 dashboard, you can create and switch between many different environments.

### Getting started

- Before you start (2 min read)
- Create an environment (2 min read)
- Working with environments (15 min read)
- Working with the IDE (10 min read)
- Working with AWS Lambda (5 min read)

CloudShell Feedback

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

### Create environment

**Details**

Name: MicroservicesIDE

Description - optional:

Environment type: New EC2 instance

New EC2 instance

CloudShell Feedback

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

### New EC2 instance

Instance type: t3.small (2 GiB RAM + 2 vCPU)

Platform: Amazon Linux 2

Timeout: 30 minutes

CloudShell Feedback

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

S | Services | Q Search [Alt+S] | N. Virginia | vodlabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

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-0930b2fdf65345dee  
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-079b846921b332a35  
Name - Public Subnet1

**Info** You have chosen a public subnet for your Cloud9 environment, note the following:

- If accessing the EC2 instance directly through SSH, the instance can only be launched into a public subnet.

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

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

ALPMCPB... > Modules > AWS Acad...  
> Lab Instructions: Building Microservices and a CI/CD Pipeline with AWS

AWS | Used \$5 of \$40 | 11:51 | Start Lab | End Lab | AWS Details | Details | Reset | X

Home | Modules | Discussions | Grades

**Task 3.2: Copy the application code to your IDE**

[Return to table of contents](#)

In this task, you will copy the source code for the monolithic application to your development environment.

1. From the **AWS Details** panel on this lab instructions page, download the `labsuser.pem` file to your local computer.
2. Upload the `.pem` file to your AWS Cloud9 IDE, and use the Linux `chmod` command to set the proper permissions on the file so that you can use it to connect to an EC2 instance.
3. Create a temp directory on the AWS Cloud9 instance at `/home/ec2-user/environment/temp`.
4. From the Amazon EC2 console, retrieve the private IPv4 address of the `MonolithicAppServer` instance.
5. Use the Linux `scp` command in the Bash terminal on the AWS Cloud9 instance to copy

**Cloud Access**

AWS CLI: Show

CLOUD LABS

Remaining session time: 11:51:18(712 minutes)  
Session started at: 2024-04-26T09:08:27-0700  
Session to end at: 2024-04-26T21:09:14-0700

Accumulated lab time: 02:36:00 (156 minutes)

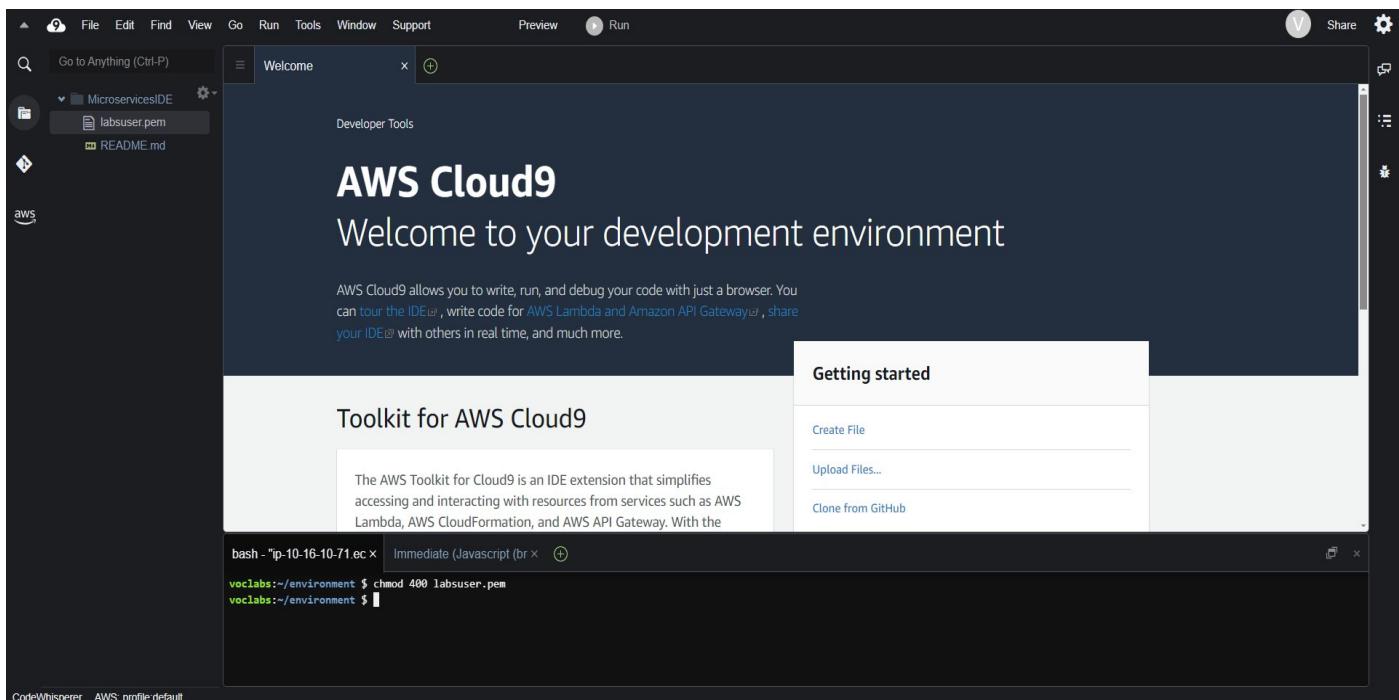
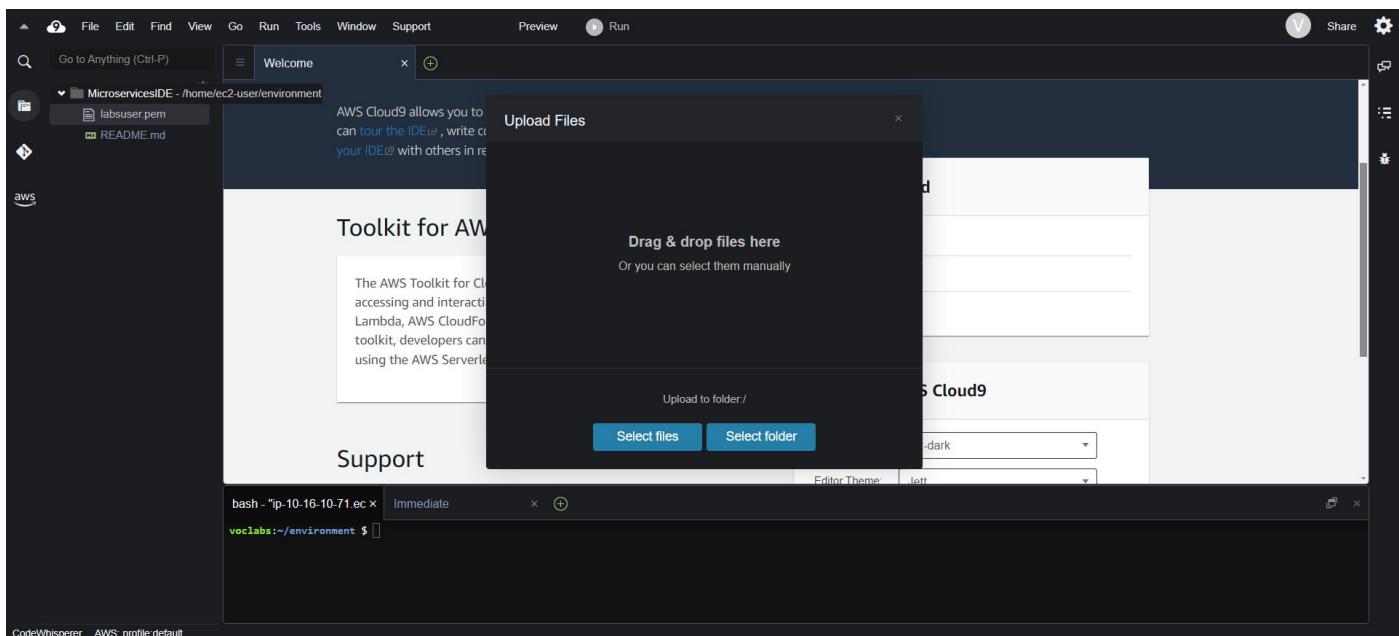
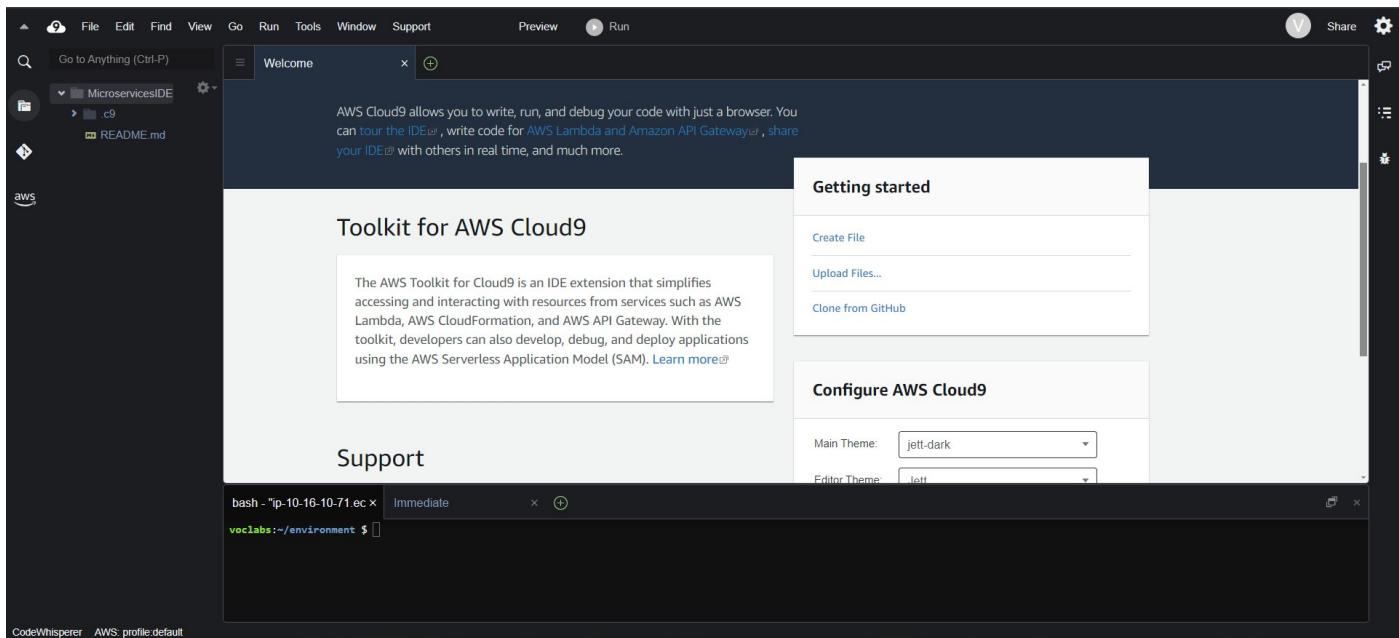
ips -- public:3.91.188.228, private:10.16.10.85

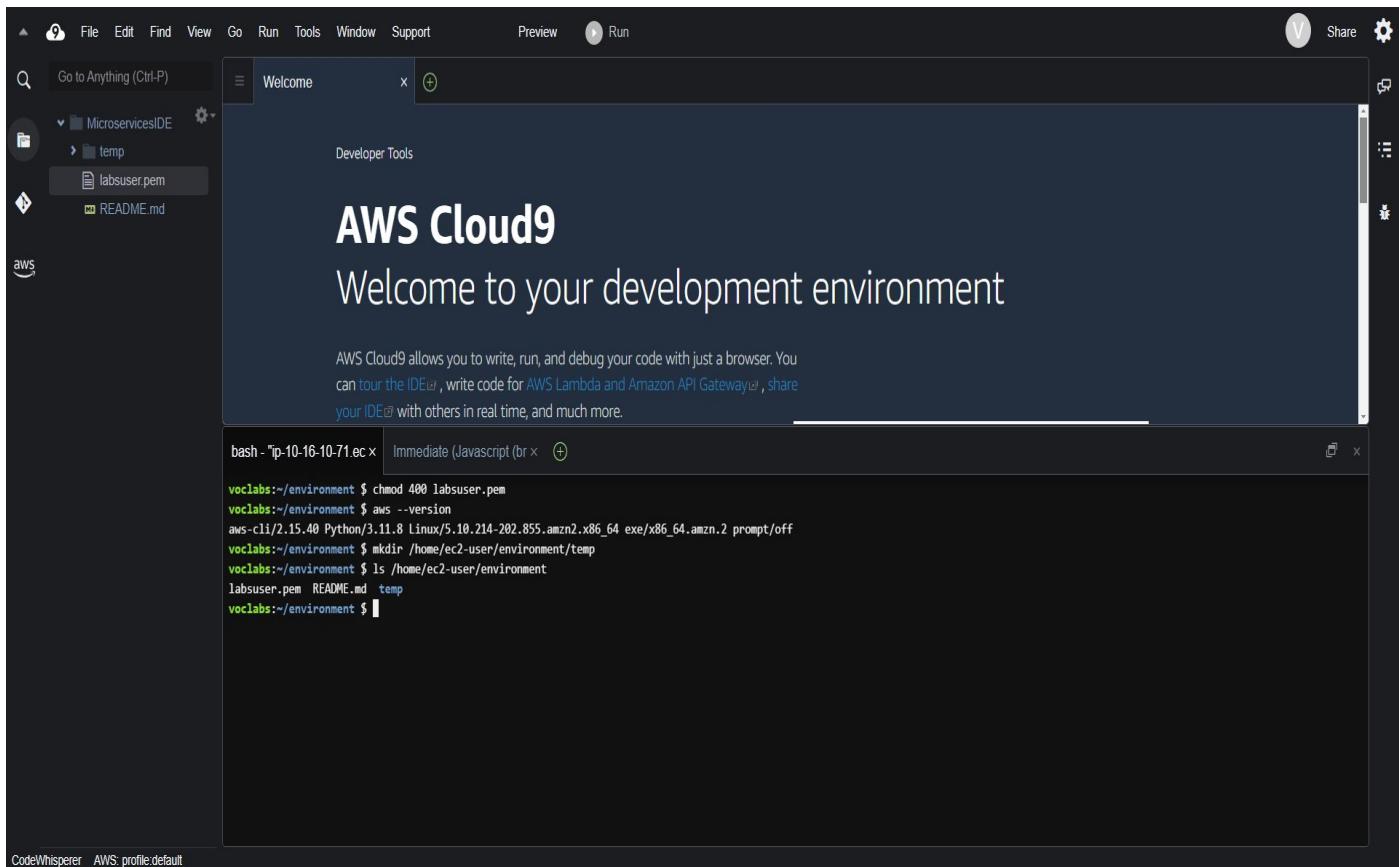
SSH key Show Download PEM

Download PPK AWS SSO Download URL

RDS Endpoint supplierdb.ccylnx0fhcf.us-east-1.rds.amazonaws.com

← Previous Next →





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

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
MonolithicApp...	i-0dd0ffcc76c2f1809f	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-3-91-188-228.compute-1.amazonaws.com
aws-cloud9-Mi...	i-0854f0547662caad5	Running	t3.small	2/2 checks passed	View alarms	us-east-1a	ec2-3-85-238-199.compute-1.amazonaws.com

AWS Cloud9 IDE interface. The terminal window shows the command: `scp -r -i ~/environment/labsuser.pem ubuntu@10.16.10.85:/home/ubuntu/resources/codebase_partner/* ~/environment/temp/`. The output of the command is visible below the command line.

AWS Cloud9 IDE interface. The terminal window shows the command: `ls ~/environment/temp`. The output of the command is visible below the command line, showing files: app, index.js, node\_modules, package.json, package-lock.json, public, views.

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

The screenshot shows the AWS Cloud9 IDE interface. On the left, the file tree displays a project named 'MicroservicesIDE' containing a 'microservices' folder with 'customer' and 'employee' subfolders, along with 'temp', 'labsuser.pem', and 'README.md'. The main workspace shows an 'Immediate (Javascript br x)' tab and a 'bash - "ip-10-16-10-71.ec x' tab. The terminal output in the bash tab shows the creation of a directory structure:

```
vclabs:~/environment $ ls ~/environment/temp  
app index.js node_modules package.json package-lock.json public views  
vclabs:~/environment $ cd ~/environment/microservices  
bash: cd: /home/ec2-user/environment/microservices: No such file or directory  
vclabs:~/environment $ cd ~/environment/  
vclabs:~/environment $ ls  
labsuser.pem README.md temp  
vclabs:~/environment $ mkdir microservices  
vclabs:~/environment $ cd ~/environment/microservices  
vclabs:~/environment/microservices $ ls  
vclabs:~/environment/microservices $ mkdir customer  
vclabs:~/environment/microservices $ mkdir employee  
vclabs:~/environment/microservices $ ls  
customer employee  
vclabs:~/environment/microservices $
```

This screenshot shows the AWS Cloud9 IDE interface again. The file tree is identical to the first one. The terminal output in the bash tab shows the user copying files from the temporary directory to the newly created 'customer' and 'employee' subdirectories:

```
vclabs:~/environment $ ls ~/environment/temp  
app index.js node_modules package.json package-lock.json public views  
vclabs:~/environment $ cp -r ~/environment/temp/* ~/environment/microservices/customer/  
vclabs:~/environment $ cp -r ~/environment/temp/* ~/environment/microservices/employee/  
vclabs:~/environment $ ls ~/environment/microservices/customer  
app index.js node_modules package.json package-lock.json public views  
vclabs:~/environment $ ls ~/environment/microservices/employee  
app index.js node_modules package.json package-lock.json public views  
vclabs:~/environment $ rm -r ~/environment/temp/*  
vclabs:~/environment $ rmdir ~/environment/temp  
vclabs:~/environment $
```

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

Developer Tools X

**CodeCommit**

Source • CodeCommit

- Getting started
- Repositories**
- Approval rule templates

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Go to resource Feedback

Repositories Info C Notify Clone URL View repository Delete repository Create repository

No results

There are no results to display.

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1>

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

Repository settings

Repository name  100 characters maximum. Other limits apply.

Description - optional

Tags

Additional configuration AWS KMS key

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

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

Developer Tools X

**CodeCommit**

Source • CodeCommit

- Getting started
- Repositories
- Code**
- Pull requests
- Commits
- Branches
- Git tags
- Settings

Approval rule templates

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Settings

Success Repository successfully created Create a notification rule for this repository

Developer Tools > CodeCommit > Repositories > microservices

Clone URL

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](#).

You must have an AWS CodeCommit managed policy attached to your IAM user, belong to a CodeStar project team, or have the equivalent permissions. [Learn how to create and configure an IAM user for accessing AWS CodeCommit](#). [Learn how to add team members to an AWS CodeStar Project](#).

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

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE

- microservices
- customer
- app
- node\_modules
- public
- views

JS index.js

{ package-lock.json

package.json

employee

- app
- node\_modules
- public
- views

JS index.js

{ package-lock.json

package.json

labsuser.pem

README.md

Immediate (Javascript br x) bash - "ip-10-16-10-71.ec x +

```
CREATE MODE 100644 employee/node_modules/validator/package.json
create mode 100644 employee/node_modules/validator/validator.js
create mode 100644 employee/node_modules/validator/validator.min.js
create mode 100644 employee/node_modules/vary/HISTORY.md
create mode 100644 employee/node_modules/vary/LICENSE
create mode 100644 employee/node_modules/vary/README.md
create mode 100644 employee/node_modules/vary/index.js
create mode 100644 employee/node_modules/vary/package.json
create mode 100644 employee/node_modules/vary/yallist/LICENSE
create mode 100644 employee/node_modules/vary/yallist/README.md
create mode 100644 employee/node_modules/vary/yallist/iterator.js
create mode 100644 employee/node_modules/vary/yallist/package.json
create mode 100644 employee/node_modules/vary/yallist/yallist.js
create mode 100644 employee/package.lock.json
create mode 100644 employee/package.json
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
vocabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
```

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE

- microservices
- customer
- app
- node\_modules
- public
- views

JS index.js

{ package-lock.json

package.json

employee

- app
- node\_modules
- public
- views

JS index.js

{ package-lock.json

package.json

labsuser.pem

README.md

Immediate (Javascript br x) bash - "ip-10-16-10-71.ec x +

```
CREATE MODE 100644 employee/node_modules/validator/package.json
create mode 100644 employee/node_modules/validator/validator.js
create mode 100644 employee/node_modules/validator/validator.min.js
create mode 100644 employee/node_modules/vary/HISTORY.md
create mode 100644 employee/node_modules/vary/LICENSE
create mode 100644 employee/node_modules/vary/README.md
create mode 100644 employee/node_modules/vary/index.js
create mode 100644 employee/node_modules/vary/package.json
create mode 100644 employee/node_modules/vary/yallist/LICENSE
create mode 100644 employee/node_modules/vary/yallist/README.md
create mode 100644 employee/node_modules/vary/yallist/iterator.js
create mode 100644 employee/node_modules/vary/yallist/package.json
create mode 100644 employee/node_modules/vary/yallist/yallist.js
create mode 100644 employee/package.lock.json
create mode 100644 employee/package.json
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
vocabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
vocabs:~/environment/microservices (dev) $
```

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE

- microservices
- customer
- app
- node\_modules
- public
- views

JS index.js

{ package-lock.json

package.json

employee

- app
- node\_modules
- public
- views

JS index.js

{ package-lock.json

package.json

labsuser.pem

README.md

Immediate (Javascript br x) git - "ip-10-16-10-71.ec2 i x +

```
CREATE MODE 100644 employee/node_modules/yallist/yallist.js
create mode 100644 employee/package-lock.json
create mode 100644 employee/package.json
create mode 100644 employee/public/css/base.css
create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
vocabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
vocabs:~/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.44 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
 * [new branch]    dev -> dev
branch 'dev' set up to track 'origin/dev'.
vocabs:~/environment/microservices (dev) $
```

```

Go to Anything (Ctrl-P)
File Edit Find View Go Run Tools Window Support Preview Run
Welcome + Immediate (Javascript br x bash -ip-10-16-10-71.ec x +)

create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/jquery-3.6.0.min.js.map
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
voclabs:-/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
voclabs:-/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.44 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
 * [new branch]      dev -> dev
branch 'dev' set up to track 'origin/dev'.
voclabs:-/environment/microservices (dev) $ git config --global user.name "YourUsername"
voclabs:-/environment/microservices (dev) $ git config --global user.name "Supreeth"
voclabs:-/environment/microservices (dev) $ git config --global user.email "supreeth.hr27@gmail.com"
voclabs:-/environment/microservices (dev) $ 

```

dev CodeWhisperer AWS profile:default

```

Go to Anything (Ctrl-P)
File Edit Find View Go Run Tools Window Support Preview Run
Welcome + Immediate (Javascript br x bash -ip-10-16-10-71.ec x +)

create mode 100644 employee/public/css/bootstrap.min.css
create mode 100644 employee/public/css/bootstrap.min.css.map
create mode 100644 employee/public/img/espresso.jpg
create mode 100644 employee/public/img/favicon.ico
create mode 100644 employee/public/js/bootstrap.min.js
create mode 100644 employee/public/js/jquery-3.6.0.min.js.map
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
voclabs:-/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
voclabs:-/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.44 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
 * [new branch]      dev -> dev
branch 'dev' set up to track 'origin/dev'.
voclabs:-/environment/microservices (dev) $ git config --global user.name "YourUsername"
voclabs:-/environment/microservices (dev) $ git config --global user.name "Supreeth"
voclabs:-/environment/microservices (dev) $ git config --global user.email "supreeth.hr27@gmail.com"
voclabs:-/environment/microservices (dev) $ git config --global credential.helper=aws codecommit credential-helper $#
credential.useHttpPath=true
core.editor=nano
user.name=Supreeth
user.email=supreeth.hr27@gmail.com
voclabs:-/environment/microservices (dev) $ 

```

dev CodeWhisperer AWS profile:default

AWS Services Search [Alt+S]

Developer Tools > CodeCommit > Repositories > microservices

## microservices

Reference

Notify dev Create pull request Clone URL

**microservices info**

Name
customer
employee

Add file ▾

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

https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1

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

The screenshot shows the AWS CodeCommit interface for the 'customer' microservice. The left sidebar is collapsed, and the main area displays the repository structure:

```

microservices / customer
+-- app
|   +-- node_modules
|   +-- public
|   +-- views
|   +-- index.js
|   +-- package-lock.json
|   +-- package.json

```

At the top right, there are buttons for 'Notify' (dropdown), 'Reference' (dropdown set to 'dev'), 'Create pull request', and 'Clone URL'.

The screenshot shows the AWS CodeCommit interface for the 'employee' microservice. The left sidebar is collapsed, and the main area displays the repository structure:

```

microservices / employee
+-- app
|   +-- node_modules
|   +-- public
|   +-- views
|   +-- index.js
|   +-- package-lock.json
|   +-- package.json

```

At the top right, there are buttons for 'Notify' (dropdown), 'Reference' (dropdown set to 'dev'), 'Create pull request', and 'Clone URL'.

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

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

**Instances (1/2) Info**

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
MonolithicAppServer	i-0dd0fffc76c2f1809f	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a
aws-cloud9-MicroservicesIDE-96f...	i-0854f0547662caad5	Running	t3.small	2/2 checks passed	View alarms +	us-east-1a

**i-0854f0547662caad5 (aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69)**

**Details** | Status and alarms New | Monitoring | Security | Networking | Storage | Tags

**Instance summary**

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0854f0547662caad5 (aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69)	44.203.156.232   open address	10.16.10.71
IPv6 address	Instance state	Public IPv4 DNS
-	Running	ec2-44-203-156-232.compute-1.amazonaws.com   open address
Hostname type	Private IP DNS name (IPv4 only)	
IP name: ip-10-16-10-71.ec2.internal	ip-10-16-10-71.ec2.internal	

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

**Security group name**

aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69-InstanceSecurityGroup-LxkWqlwaPtfh

**Security group ID**

sg-027f77a230148ee3f

**Description**

Security group for AWS Cloud9 environment aws-cloud9-MicroservicesIDE-96f30ddffbf9479bb77d2e6cd8e5af69

**VPC ID**

vpc-0930b2fdf65345dee

**Owner**

579794416250

**Inbound rules count**

2 Permission entries

**Outbound rules count**

1 Permission entry

**Inbound rules (2)**

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0cc51549c21716e99	IPv4	SSH	TCP	22
-	sgr-02674c0c3c2bb00a3	IPv4	SSH	TCP	22

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

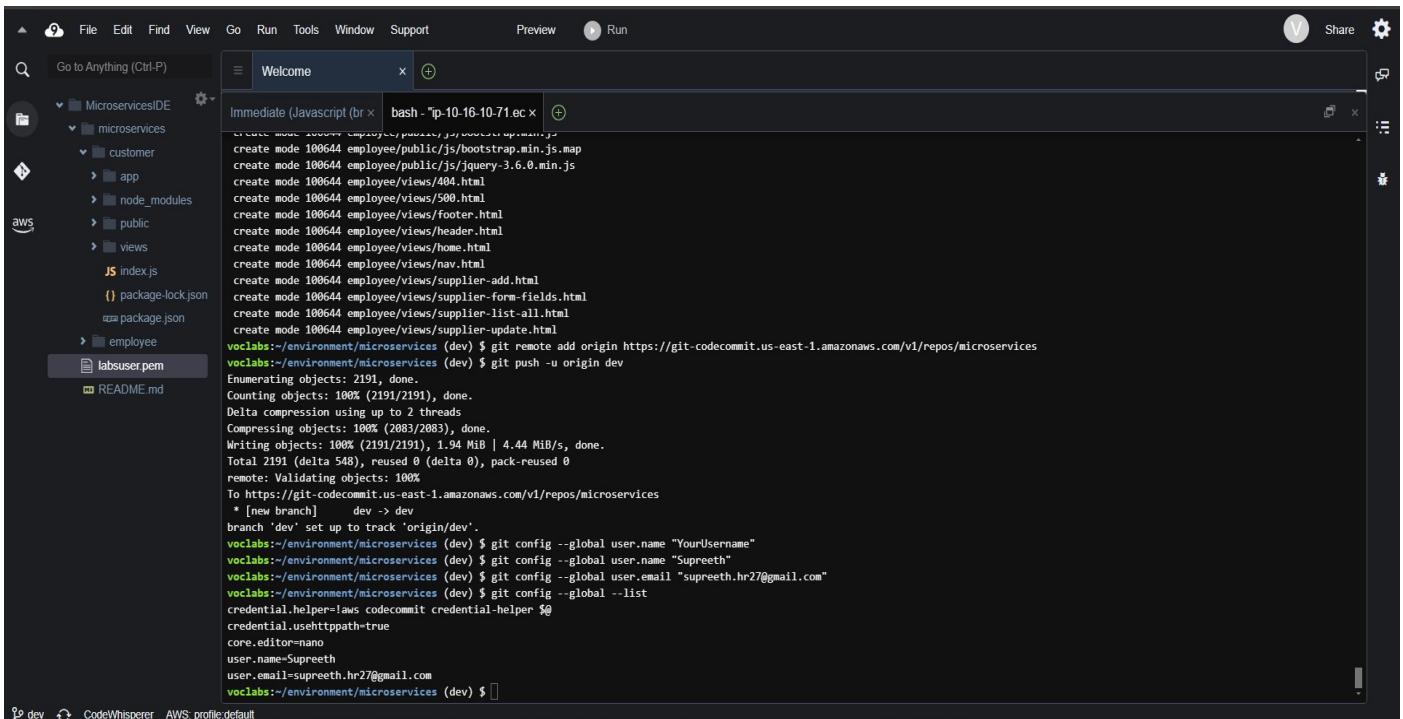
**Inbound rules**

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0cc51549c21716e99	SSH	TCP	22	Custom	35.172.155.192/27
sgr-02674c0c3c2bb00a3	SSH	TCP	22	Custom	35.172.155.96/27
-	Custom TCP	TCP	8080	Anyw...	0.0.0.0/0
-	Custom TCP	TCP	8081	Anyw...	0.0.0.0/0

**Add rule**

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

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



The screenshot shows the AWS Cloud9 IDE interface with the terminal window open. The terminal output is as follows:

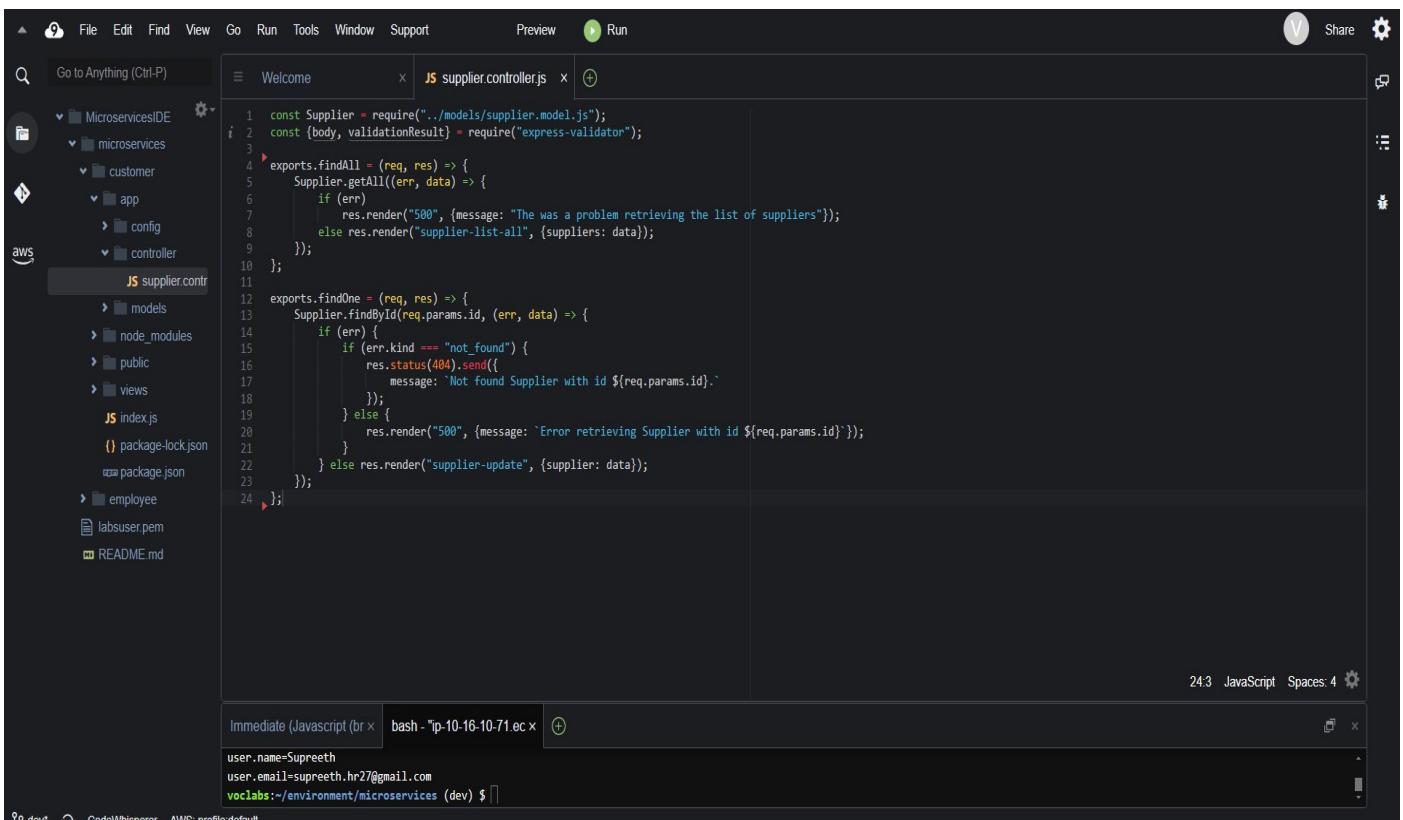
```

File Edit Find View Go Run Tools Window Support Preview Run

Welcome bash - ip-10-16-10-71.ec.x

create mode 100644 employee/public/js/bootstrap.min.js.map
create mode 100644 employee/public/js/jquery-3.6.0.min.js
create mode 100644 employee/views/404.html
create mode 100644 employee/views/500.html
create mode 100644 employee/views/footer.html
create mode 100644 employee/views/header.html
create mode 100644 employee/views/home.html
create mode 100644 employee/views/nav.html
create mode 100644 employee/views/supplier-add.html
create mode 100644 employee/views/supplier-form-fields.html
create mode 100644 employee/views/supplier-list-all.html
create mode 100644 employee/views/supplier-update.html
vocabs:~/environment/microservices (dev) $ git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
vocabs:~/environment/microservices (dev) $ git push -u origin dev
Enumerating objects: 2191, done.
Counting objects: 100% (2191/2191), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2083/2083), done.
Writing objects: 100% (2191/2191), 1.94 MiB | 4.44 MiB/s, done.
Total 2191 (delta 548), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
 * [new branch]      dev -> dev
branch 'dev' set up to track 'origin/dev'.
vocabs:~/environment/microservices (dev) $ git config --global user.name "YourUsername"
vocabs:~/environment/microservices (dev) $ git config --global user.name "Supreeth"
vocabs:~/environment/microservices (dev) $ git config --global user.email "supreeth.hr27@gmail.com"
vocabs:~/environment/microservices (dev) $ git config --global --list
credential.helper=laws codemcommit credential-helper @@
credential.usehttppath=true
core.editor=nano
user.name=Supreeth
user.email=supreeth.hr27@gmail.com
vocabs:~/environment/microservices (dev) $

```



The screenshot shows the AWS Cloud9 IDE interface with the code editor window open. The file being edited is supplier.controller.js. 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});
    });
};

```

The bottom terminal window shows the user's environment variables:

```

user.name=Supreeth
user.email=supreeth.hr27@gmail.com
vocabs:~/environment/microservices (dev) $

```

The screenshot shows the AWS Lambda Function Editor interface. The left sidebar displays the project structure for a 'MicroservicesIDE' function, including 'customer', 'app', 'config', 'controller', 'models', and 'node\_modules'. The main editor area contains the 'supplier.model.js' file:

```

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(null, null);
26         return;
27       }
28       console.log("suppliers: ", res);
29       result(null, res);
30     });
31   };
32
33   Supplier.findById = (supplierId, result) => {
34     db_connection.query(`SELECT * FROM suppliers WHERE id = ${supplierId}`, (err, res) => {
35       if (err) {
36         console.log("error: ", err);
37         result(null, null);
38         return;
39       }
40       if (res.length) {
41         console.log("found supplier: ", res[0]);
42         result(null, res[0]);
43         return;
44       }
45       result({kind: "not_found"}, null);
46     });
47   };
48
49   module.exports = Supplier;
50 
```

The bottom navigation bar shows 'Immediate (Javascript)' and 'bash - ip-10-16-10-71.ec2.us-west-2.amazonaws.com' with the command 'user.email=supreeth.hr27@gmail.com'.

The screenshot shows the AWS Lambda Function Editor interface. The left sidebar displays the project structure for a 'MicroservicesIDE' function, including 'customer', 'app', 'node\_modules', 'public', and 'views'. The 'views' folder contains files like '404.html', '500.html', 'footer.html', 'header.html', 'home.html', 'nav.html', 'supplier-add.htm', 'supplier-form-fie', 'supplier-list-all.htm', 'supplier-update.htm', 'index.js', 'package-lock.json', and 'package.json'. The main editor area contains the 'supplier-list-all.html' file:

```

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        </tr>
27      {{/suppliers}}
28    </tbody>
29  </table>
30 </div>
31 {{>footer}}
32 
```

The bottom navigation bar shows 'Immediate (Javascript)' and 'bash - ip-10-16-10-71.ec2.us-west-2.amazonaws.com' with the command 'user.email=supreeth.hr27@gmail.com'.

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

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

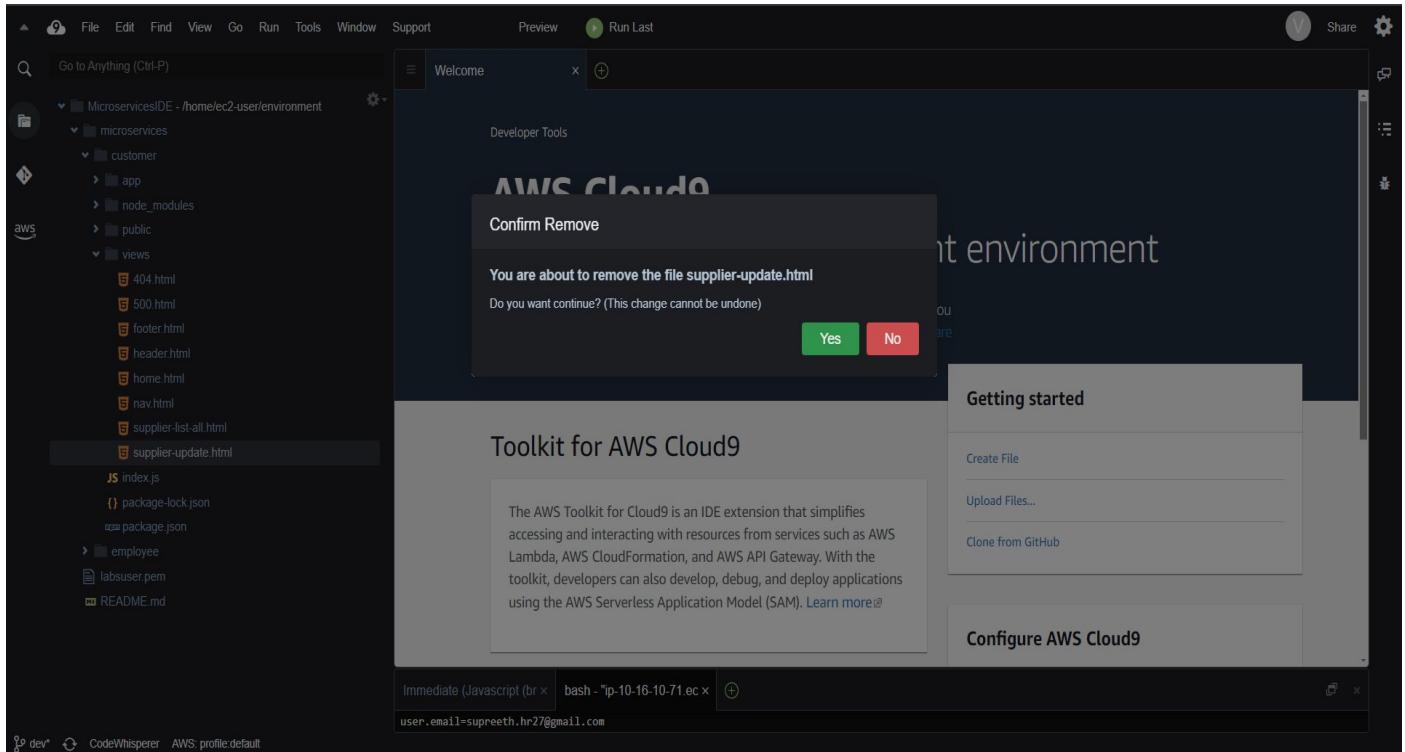
```

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>

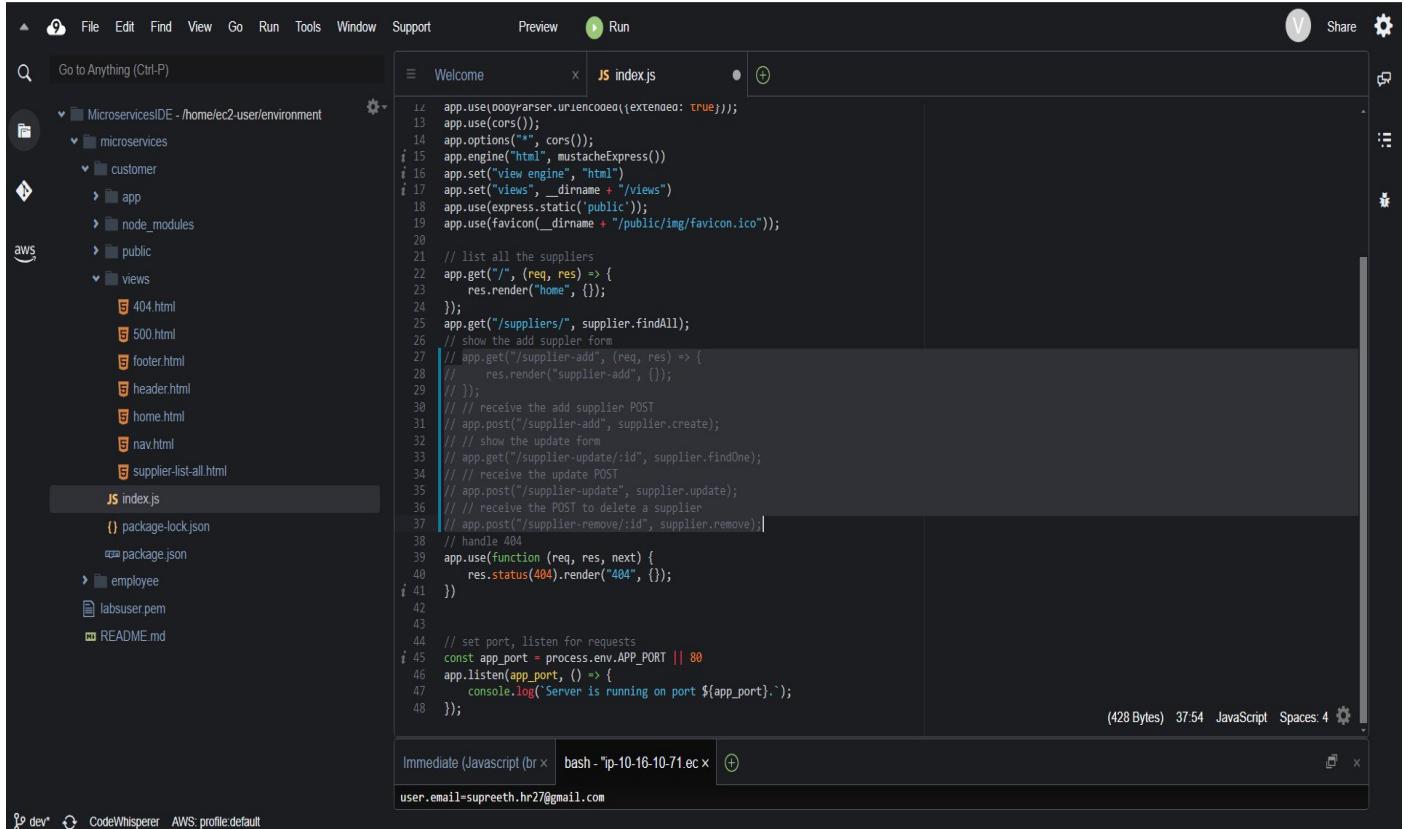
```

The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays the project structure under 'MicroservicesIDE'. The main editor window shows the content of 'nav.html'. Below the editor is a terminal window titled 'bash - ip-10-16-10-71.ec2.us-west-2.compute.amazonaws.com' with the command 'user.email=supreeth.hr27@gmail.com'. The status bar at the bottom indicates '9:33 HTML Spaces: 4'.

A confirmation dialog box titled 'Confirm Remove' is displayed in the center of the screen. It asks, 'You are about to remove the file supplier-form-fields.html. Do you want to continue? (This change cannot be undone)'. There are two buttons: 'Yes' (green) and 'No' (red). The background of the IDE shows the same project structure and terminal window as the previous screenshot.



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



The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays a file tree for a project named 'MicroservicesIDE' located at '/home/ec2-user/environment'. The 'customer' directory is expanded, showing sub-directories 'app', 'node\_modules', 'public', and 'views', along with files like '404.html', '500.html', 'footer.html', 'header.html', 'home.html', 'nav.html', and 'supplier-list-all.html'. The current file being edited is 'index.js' in the 'customer/app' directory. The code in 'index.js' is as follows:

```

1 app.use(bodyParser.urlencoded({extended: true}));
2 app.use(cors());
3 app.options('*', cors());
4 app.engine('html', mustacheExpress())
5 app.set('view engine', "html")
6 app.set('views', __dirname + '/views')
7 app.use(express.static('public'));
8 app.use(favicon(__dirname + "/public/img/favicon.ico"));
9
10 // list all the suppliers
11 app.get('/', (req, res) => {
12   res.render('home', {});
13 });
14 app.get('/suppliers', supplier.findAll);
15 // show the add supplier form
16 // app.get('/supplier-add', (req, res) => {
17 //   res.render("supplier-add", {});
18 // });
19 // receive the add supplier POST
20 // app.post("/supplier-add", supplier.create);
21 // show the update form
22 // app.get("/supplier-update/:id", supplier.findOne);
23 // receive the update POST
24 // app.post("/supplier-update", supplier.update);
25 // receive the POST to delete a supplier
26 // app.post("/supplier-remove/:id", supplier.remove);
27 // handle 404
28 app.use(function (req, res, next) {
29   res.status(404).render("404", {});
30 })
31
32 // set port, listen for requests
33 const app_port = process.env.APP_PORT || 8080
34 app.listen(app_port, () => {
35   console.log(`Server is running on port ${app_port}.`);
36 });

```

The status bar at the bottom right indicates '4.46 JavaScript Spaces: 4'.

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

The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays a file tree for a project named 'MicroservicesIDE' located at '/home/ec2-user/environment'. The 'customer' directory is expanded, showing sub-directories 'app', 'node\_modules', 'public', and 'views', along with files like '404.html', '500.html', 'footer.html', 'header.html', 'home.html', 'nav.html', and 'supplier-list-all.html'. The current file being edited is 'Dockerfile' in the 'customer/app' directory. The code in 'Dockerfile' is as follows:

```

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" ]

```

The status bar at the bottom right indicates '7:28 Dockerfile Spaces: 4'.

The screenshot shows the AWS Lambda Microservices IDE interface. On the left, a file tree displays a project structure with files like Dockerfile, index.js, package.json, and README.md. The Dockerfile content is:

```

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"]

```

The right pane shows the build logs for an immediate build:

```

voclabs:/environment $ cd ~/environment/microservices/customer
voclabs:/environment/microservices/customer (dev) $ docker build --tag customer .
Sending build context to Docker daemon 9.001MB
Step 1/7 : FROM node:11-alpine
11-alpine: Pulling from library/node
e7c96db7181b: Pull complete
0119aca44649: Pull complete
40df19605a18: Pull complete
82194b8b4a64: Pull complete
Digest: sha256:b8b56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932
Status: Downloaded newer image for node:11-alpine
--> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
--> Running in c5e8f98a71b9
Removing intermediate container c5e8f98a71b9
--> 0c85f1609717
Step 3/7 : WORKDIR /usr/src/app
--> Running in 1501a69d21d7
Removing intermediate container 1501a69d21d7
--> 986d5698aa61
Step 4/7 : COPY . .
--> 15d37ab065d9
Step 5/7 : RUN npm install
--> Running in a20e464d6cca

```

This screenshot is identical to the one above, but the build logs show additional output related to npm audit:

```

Step 3/7 : WORKDIR /usr/src/app
--> Running in 1501a69d21d7
Removing intermediate container 1501a69d21d7
--> 986d5698aa61
Step 4/7 : COPY . .
--> 15d37ab065d9
Step 5/7 : RUN npm install
--> Running in a20e464d6cca
npm WARN coffee_api@1.0.0 No repository field.

audited 78 packages in 0.816s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container a20e464d6cca
--> bb23e6081fb6
Step 6/7 : EXPOSE 8080
--> Running in 02bf5c7a3e7d
Removing intermediate container 02bf5c7a3e7d
--> 79e085754abb
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in d2b87fa14a66
Removing intermediate container d2b87fa14a66
--> 08471c89baaf
Successfully built d8471c89baaf
Successfully tagged customer:latest
voclabs:/environment/microservices/customer (dev) $

```

- I verified that the image was created by listing the Docker images with a command that showed the customer-labeled image alongside the base Alpine Linux node image. To launch the Docker container on port 8080 I set a dbEndpoint environment variable to point to the RDS database by extracting the value from config.js, then used **docker run -d --name customer\_1 -p 8080:8080 -e APP\_DB\_HOST="\$dbEndpoint" customer** to start the container. This command launched the container mapping port 8080 from the host to the container and passed the database endpoint as an environment variable.

The screenshot shows the AWS Lambda CodeWhisperer interface. The left sidebar lists project files: Dockerfile, index.js, package-lock.json, package.json, employee, labsuser.pem, and README.md. The Dockerfile tab is active, displaying a Dockerfile for a Node.js application. The right pane shows the Dockerfile code with several red annotations indicating potential issues or suggestions. A terminal window at the bottom shows the command `vocabs:~/environment/microservices/customer (dev) \$ docker images` followed by a list of images.

```
FROM node:11_alpine
bash - "ip-10-16-10-71.ec
Step 2/2 : RUN mkdir -p /usr/src/app
--> Running in c5e8f98a71b9
Removing intermediate container c5e8f98a71b9
--> 0c85f16b0717
Step 3/7 : WORKDIR /usr/src/app
--> Running in 1501a69d21d7
Removing intermediate container 1501a69d21d7
--> 986d5698aa61
Step 4/7 : COPY . .
--> 15d37ab065d9
Step 5/7 : RUN npm install
--> Running in a20e464d6cca
--> npm WARN coffee_apigl1.0.0 No repository field.

audited 78 packages in 0.816s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
Removing intermediate container a20e464d6cca
--> bb23e6081fb6
Step 6/7 : EXPOSE 8080
--> Running in 02bf5c7a3e7d
Removing intermediate container 02bf5c7a3e7d
--> 79e0857548bb
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in d2b87fa14a66
Removing intermediate container d2b87fa14a66
--> d8471c89baaf
Successfully built d8471c89baaf
Successfully tagged customer:latest
vocabs:~/environment/microservices/customer (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
customer latest d8471c89baaf About a minute ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
vocabs:~/environment/microservices/customer (dev) $
```

This screenshot is nearly identical to the one above, showing the same AWS Lambda CodeWhisperer interface. The left sidebar and Dockerfile content are the same. The right pane shows the Dockerfile code with red annotations and a terminal window at the bottom showing the command `vocabs:~/environment/microservices/customer (dev) \$ docker images` followed by a list of images.

```
FROM node:11_alpine
bash - "ip-10-16-10-71.ec
--> Running in 1501a69d21d7
Removing intermediate container 1501a69d21d7
--> 986d5698aa61
Step 4/7 : COPY . .
--> 15d37ab065d9
Step 5/7 : RUN npm install
--> Running in a20e464d6cca
--> npm WARN coffee_apigl1.0.0 No repository field.

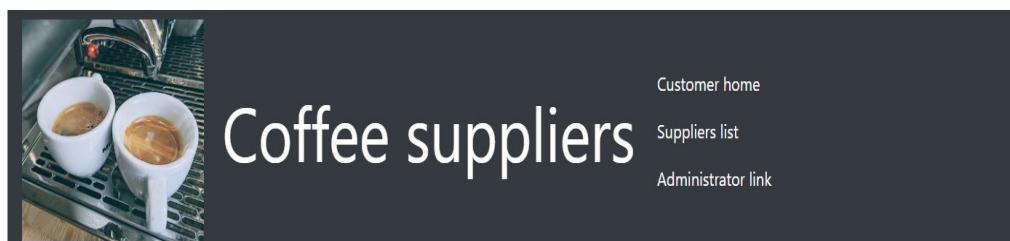
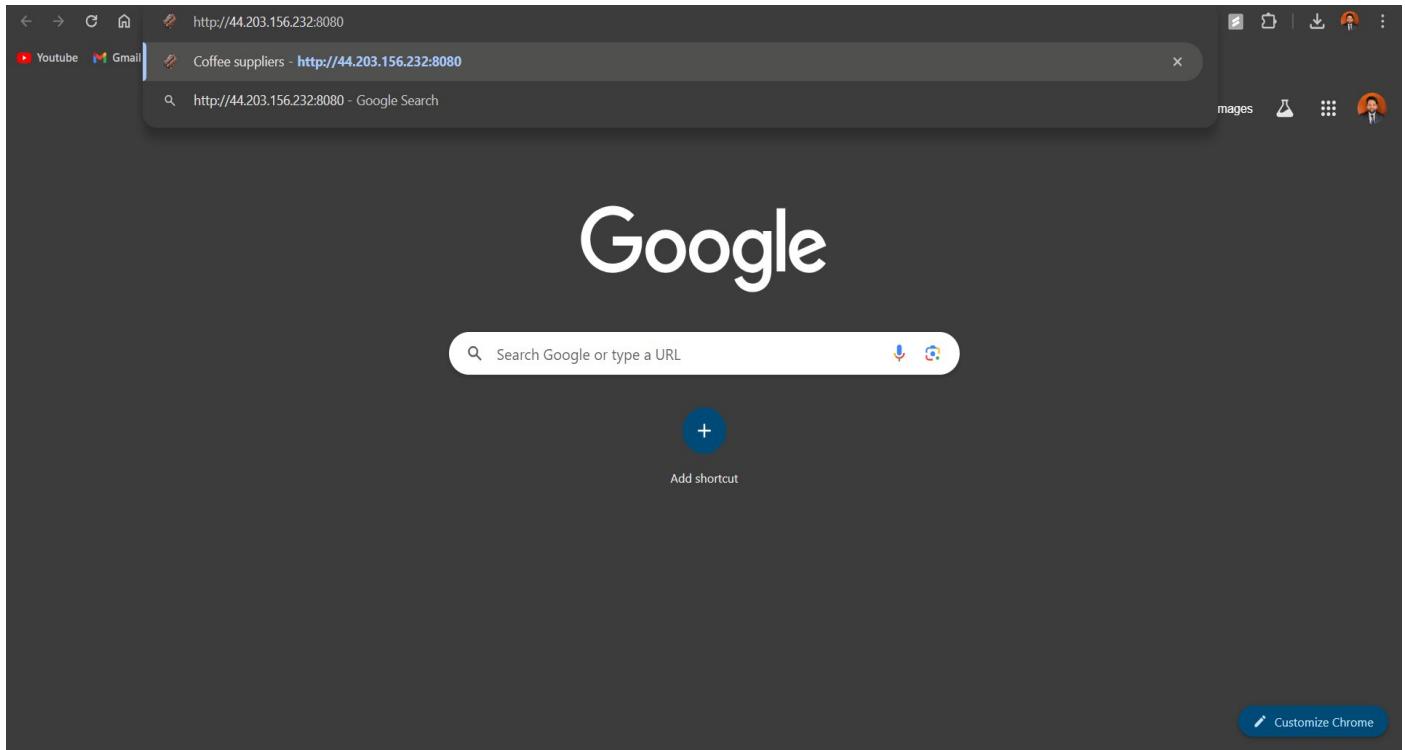
audited 78 packages in 0.816s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
Removing intermediate container a20e464d6cca
--> bb23e6081fb6
Step 6/7 : EXPOSE 8080
--> Running in 02bf5c7a3e7d
Removing intermediate container 02bf5c7a3e7d
--> 79e0857548bb
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in d2b87fa14a66
Removing intermediate container d2b87fa14a66
--> d8471c89baaf
Successfully built d8471c89baaf
Successfully tagged customer:latest
vocabs:~/environment/microservices/customer (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
customer latest d8471c89baaf About a minute ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
vocabs:~/environment/microservices/customer (dev) $ echo $dbEndpoint
supplierdb.ccylnx0fhcf.us-east-1.rds.amazonaws.com
vocabs:~/environment/microservices/customer (dev) $ docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint" customer
9e837092ac258c1881c7e45c06b116e56ae4bc59160ef3d02be5c96ae4daf69
vocabs:~/environment/microservices/customer (dev) $
```

This screenshot is identical to the previous ones, showing the AWS Lambda CodeWhisperer interface. The left sidebar and Dockerfile content are the same. The right pane shows the Dockerfile code with red annotations and a terminal window at the bottom showing the command `vocabs:~/environment/microservices/customer (dev) \$ docker images` followed by a list of images.

```
FROM node:11_alpine
bash - "ip-10-16-10-71.ec
Step 4/7 : COPY . .
--> 15d37ab065d9
Step 5/7 : RUN npm install
--> Running in a20e464d6cca
--> npm WARN coffee_apigl1.0.0 No repository field.

audited 78 packages in 0.816s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
Removing intermediate container a20e464d6cca
--> bb23e6081fb6
Step 6/7 : EXPOSE 8080
--> Running in 02bf5c7a3e7d
Removing intermediate container 02bf5c7a3e7d
--> 79e0857548bb
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in d2b87fa14a66
Removing intermediate container d2b87fa14a66
--> d8471c89baaf
Successfully built d8471c89baaf
Successfully tagged customer:latest
vocabs:~/environment/microservices/customer (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
customer latest d8471c89baaf About a minute ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
vocabs:~/environment/microservices/customer (dev) $ dbEndpoint=$(cat ~/environment/microservices/customer/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
vocabs:~/environment/microservices/customer (dev) $ echo $dbEndpoint
supplierdb.ccylnx0fhcf.us-east-1.rds.amazonaws.com
vocabs:~/environment/microservices/customer (dev) $ docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint" customer
9e837092ac258c1881c7e45c06b116e56ae4bc59160ef3d02be5c96ae4daf69
vocabs:~/environment/microservices/customer (dev) $ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9e837092ac25 customer "docker-entrypoint.s_..." About a minute ago Up About a minute 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp customer_1
vocabs:~/environment/microservices/customer (dev) $
```

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



Welcome

Use this app to keep track of your coffee suppliers

[List of suppliers](#)



## All suppliers

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

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

```
vclabs:~/environment $ cd ~/environment/microservices
vclabs:~/environment $ cd ~/environment/microservices
vclabs:~/environment/microservices (dev) $ git add .
vclabs:~/environment/microservices (dev) $ git commit -m "Updated code with new Dockerfile and read-only functionality"
[dev cfb8112] Updated code with new Dockerfile and read-only functionality
9 files changed, 22 insertions(+), 266 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
vclabs:~/environment/microservices (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), 1.34 KiB | 1.34 MiB/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
635d342..cbf8112 dev -> dev
vclabs:~/environment/microservices (dev) $ git log
commit cfb811296bd0de24f68a9b698c9af2942c5e290 (HEAD -> dev, origin/dev)
Author: Supreeth <supreeth.hr27@gmail.com>
Date:   Fri Apr 26 21:17:38 2024 +0000

    Updated code with new Dockerfile and read-only functionality

commit 635d342e0a1bc0b7eed7a87a734ea042915948a
Author: EC2 Default User <ec2-user@ip-10-16-10-71.ec2.internal>
Date:   Fri Apr 26 19:25:55 2024 +0000

    two unmodified copies of the application code
vclabs:~/environment/microservices (dev) $
```

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**Developer Tools** CodeCommit

- Source • CodeCommit
  - Getting started
  - Repositories
  - Code
  - Pull requests
  - Commits**
  - Branches
  - Git tags
  - Settings
  - Approval rule templates
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
- Settings

Developer Tools > CodeCommit > Repositories > microservices > Commits

## microservices

Commits Commit visualizer Compare commits

Commits	Info	Reference				
cbf81129	Updated code with new Dockerfile and read-only functionality	5 minutes ago	5 minutes ago	Supreeth	Supreeth	<a href="#">Copy ID</a> <a href="#">Browse</a>
635d342e	two unmodified copies of the application code	1 hour ago	1 hour ago	EC2 Default User	EC2 Default User	<a href="#">Copy ID</a> <a href="#">Browse</a>

https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories?region=us-east-1 © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**Developer Tools** CodeCommit

- Source • CodeCommit
  - Getting started
  - Repositories
  - Code
  - Pull requests
  - Commits**
  - Branches
  - Git tags
  - Settings
  - Approval rule templates
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
- Settings

Developer Tools > CodeCommit > Repositories > microservices > Commits > cbf811296bd0de84f68a98698c9af2d2942c5e290

## Commit cbf811296bd0de84f68a98698c9af2d2942c5e290

Copy commit ID [Copy](#) [Browse](#)

**Details**

Author	Supreeth	Authored date	5 minutes ago	Committer	Supreeth
	supreeth.hr27@gmail.com				supreeth.hr27@gmail.com
Commit date	5 minutes ago	Parent commit	635d342e0a1bc0b7eeed7a734ea042915948a		
Commit message	Updated code with new Dockerfile and read-only functionality				

Page 1 of 1 Go to file Hide comments Hide whitespace changes Unified Split

**customer/Dockerfile** Added

Browse file contents Comment on file

```
1 + FROM node:11-alpine
```

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**Developer Tools** CodeCommit

- Source • CodeCommit
  - Getting started
  - Repositories
  - Code
  - Pull requests
  - Commits**
  - Branches
  - Git tags
  - Settings
  - Approval rule templates
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
- Settings

Developer Tools > CodeCommit > Repositories > microservices > Commits > cbf811296bd0de84f68a98698c9af2d2942c5e290

**customer/Dockerfile** Added

Browse file contents Comment on file

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

**customer/app/controller/supplier.controller.js**

Browse file contents Comment on file

```
1 1 const Supplier = require("../models/supplier.model.js");
2 2 const {body, validationResult} = require("express-validator");
3 3
4 - exports.create = [
5 -   // Validate and sanitize the name field.
6 -   body('name', 'The supplier name is required').trim().isLength({min: 1}).escape(),
7 -   body('address', 'The supplier address is required').trim().isLength({min: 1}).escape(),
8 -   body('city', 'The supplier city is required').trim().isLength({min: 1}).escape(),
9 -   body('state', 'The supplier state is required').trim().isLength({min: 1}).escape(),
10 -  body('phone', 'Phone number should be 10 digit number plus optional country code').trim().isMobilePhone().escape(),
11 -  // Process request after validation and sanitization.
12 -  (req, res, next) => {
13 -    // Extract the validation errors from a request.
```

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

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

The screenshot shows the AWS Cloud9 IDE interface. On the left, the file tree displays a project structure under 'MicroservicesIDE - /home/voclabs/microservices'. The 'employee' directory is selected. Inside 'employee', there are files: Dockerfile, index.js, package-lock.json, package.json, labsuser.pem, and README.md. The 'Dockerfile' is currently selected. On the right, a terminal window titled 'bash - "ip-10-16-10-71.ec2.internal"' shows the following git log output:

```

635d342..cbf8112 dev -> dev
voclabs:~/environment/microservices (dev) $ git log
commit cbf811296bd0de84f68a98698c9af2942c5e290 (HEAD -> dev, origin/dev)
Author: Supreeth <supreeth.hr27@gmail.com>
Date:   Fri Apr 26 21:17:38 2024 +0000

    Updated code with new Dockerfile and read-only functionality

commit 635d342e0a1bc0b7eed7a87a734ea042915948
Author: EC2 Default User <ec2-user@ip-10-16-10-71.ec2.internal>
Date:   Fri Apr 26 19:25:55 2024 +0000

    two unmodified copies of the application code
voclabs:~/environment/microservices (dev) $

```

At the bottom of the terminal, status icons for 'dev', 'CodeWhisperer', and 'AWS profile:default' are visible.

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE - /ho

- microservices
- customer
- employee
  - app
    - config
    - controller
      - JS supplier.controller.j
    - models
    - node\_modules
    - public
    - views
      - JS index.js
      - JS package-lock.json
      - package.json
- JS index.js
- JS package-lock.json
- package.json
- labsuser.pem
- README.md

JS index.js

```
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());

two unmodified copies of the application code
vocabs:~/environment/microservices (dev) $ clear
vocabs:~/environment/microservices (dev) $ cd ~/environment/microservices/employee
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) $ |
```

commit 635d342e0a1bc0b7eed7a87a734ea842915948a
Author: EC2 Default User <ec2-user@ip-10-16-10-71.ec2.internal>
Date: Fri Apr 26 19:25:55 2024 +0000

1:1 JavaScript Spaces: 4

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE - /ho

- microservices
- customer
- employee
  - app
    - config
    - controller
      - JS supplier.controller.j
    - models
    - node\_modules
    - public
    - views
      - JS index.js
      - JS package-lock.json
      - package.json
- JS index.js
- JS package-lock.json
- package.json
- labsuser.pem
- README.md

JS index.js

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

// set port, listen for requests
const app_port = process.env.APP_PORT || 8081;
app.listen(app_port, () => {
  console.log(`Server is running on port ${app_port}.`);
});
```

two unmodified copies of the application code
vocabs:~/environment/microservices (dev) \$ clear
vocabs:~/environment/microservices (dev) \$ cd ~/environment/microservices/employee
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) \$ |

45:46 JavaScript Spaces: 4

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE - /ho

- microservices
- customer
- employee
  - app
    - config
    - controller
    - models
    - node\_modules
    - public
    - views
      - 404.html
      - 500.html
      - footer.html
      - header.html
      - home.html
      - nav.html
      - supplier-add.html
      - supplier-form-fields.h
      - supplier-list-all.html
      - supplier-update.html
- JS index.js
- JS package-lock.json
- package.json
- labsuser.pem
- README.md

JS supplier-add.html

```
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Delete supplier</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&amptimes</span>
        </button>
      </div>
      <div class="modal-body">
        Are you sure you want to delete supplier {{name}}?
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <form action="/admin/supplier-remove/{{id}}" method="POST">
          <button type="submit" class="float-right btn btn-danger">Delete this supplier</button>
        </form>
      </div>
    </div>
  </div>
</div>
```

JS supplier-update.html

```
else res.redirect("/suppliers");
} else res.redirect("/suppliers");
} 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) $ |
```

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) \$ |

38:41 HTML Spaces: 4

The screenshot shows a code editor interface with the following details:

- File Structure:** MicroservicesIDE - /ho/microservices/employee/app.
- Open File:** home.html
- Code Content:**

```

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        <a href="/admin/suppliers">List of suppliers</a></p>
8      </div>
9    </div>
10   {{>footer}}

```
- Logs:**
  - Immediate Javascript (br x)**: Shows a series of API requests and responses related to supplier management.
  - bash - ip-10-16-10-71.ec x**: Shows the same log entries as the Javascript log.
- Bottom Status Bar:** 7.27 HTML Spaces: 4

- For the page headers, I updated employee/views/header.html to have the title "Manage coffee suppliers." In the employee/views/nav.html file, I changed line 3 to "Manage coffee suppliers," updated the link on line 7 to /admin/suppliers labeled "Administrator home," and added a new link after line 8 for navigation back to the customer area (<a class="nav-link" href="/">Customer home</a>).
- These changes ensure that the employee microservice operates with a distinct path structure for administrative tasks, allowing employees to manage coffee suppliers effectively, while also providing navigation to the customer microservice. This setup is designed to work with the planned Application Load Balancer and support a clear separation of responsibilities within the microservices architecture.

The screenshot shows a code editor interface with the following details:

- File Structure:** MicroservicesIDE - /ho/microservices/employee/app.
- Open File:** header.html
- Code Content:**

```

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

```
- Logs:**
  - Immediate Javascript (br x)**: Shows a series of API requests and responses related to supplier management.
  - bash - ip-10-16-10-71.ec x**: Shows the same log entries as the Javascript log.
- Bottom Status Bar:** 7.35 HTML Spaces: 4

The screenshot shows the AWS Lambda Code Editor interface. On the left, a file tree displays files like `nav.html`, `supplier.js`, and `Dockerfile`. The main area has two tabs: `nav.html` and `supplier.js`.

```

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>

```

```

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:30:    <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>
vocabs:~/environment/microservices/employee (dev) $ |

```

- Task 4.5: To create a Docker image for the employee microservice and test its functionality, I first duplicated the Dockerfile from the customer microservice into the employee microservice directory. I then edited the copied Dockerfile to set the EXPOSE line to port 8081 to match the unique port assigned to this microservice.
- I built the Docker image for the employee microservice specifying "employee" as the tag by running the **docker build --tag employee .** command in the terminal from the employee directory. After building the Docker image, I launched a container named "employee\_1" on port 8081, ensuring I passed the database endpoint using the -e parameter to establish connectivity with the RDS database. The command looked like this: **docker run -d --name employee\_1 -p 8081:8081 -e APP\_DB\_HOST="\$dbEndpoint" employee**.

The screenshot shows the AWS Lambda Code Editor interface. On the left, a file tree displays files like `nav.html`, `supplier.js`, and `Dockerfile`. The main area shows the contents of the `Dockerfile`.

```

# Dockerfile for the employee microservice
# Based on Node.js 14.17.0
# https://github.com/nodesource/docker-images/tree/v14.17.0

FROM node:14.17.0-alpine

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8081

```

The screenshot shows a code editor interface with a sidebar containing project files for a 'MicroservicesIDE - /ho' project. The files include 'customer', 'employee', and 'public' directories, along with 'Dockerfile', 'index.js', 'package-lock.json', and 'package.json'. The main pane displays a Dockerfile:

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

Below the Dockerfile, there is a preview area showing the command 'bash -ip 10-16-10-71.ecx' and its output:

```
vocabls:~/environment $ cd ~/environment/microservices/employee
vocabls:~/environment $ cd ~/environment/microservices/employee
vocabls:~/environment/microservices/employee (dev) $ docker build --tag employee .
Sending build context to Docker daemon 9.014MB
Step 1/7 : FROM node:11-alpine
--> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
--> Using cache
--> 0c85f166b717
Step 3/7 : WORKDIR /usr/src/app
--> Using cache
--> 986d5698aa1
Step 4/7 : COPY . .
--> 245a599fbfb0
Step 5/7 : RUN npm install
--> Running in 4b0e85c84df8
npm WARN coffee_spi@1.0.0 No repository field.

audited 78 packages in 0.803s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 4b0e85c84df8
--> 02f024244758
Step 6/7 : EXPOSE 8081
--> Running in 22a871d494b9
Removing intermediate container 22a871d494b9
--> 07c5d4e49274
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in 3617a461ce1b
Removing intermediate container 3617a461ce1b
--> 686acedc55e3
Successfully built 686acedc55e3
Successfully tagged employee:latest
vocabls:~/environment/microservices/employee (dev) $
```

The screenshot shows a code editor interface with a sidebar containing project files for a 'MicroservicesIDE - /ho' project. The files include 'customer', 'employee', and 'public' directories, along with 'Dockerfile', 'index.js', 'package-lock.json', and 'package.json'. The main pane displays a terminal output from a 'bash -ip 10-16-10-71.ecx' session:

```
vocabls:~/environment $ cd ~/environment/microservices/employee
vocabls:~/environment $ cd ~/environment/microservices/employee
vocabls:~/environment/microservices/employee (dev) $ docker build --tag employee .
Sending build context to Docker daemon 9.014MB
Step 1/7 : FROM node:11-alpine
--> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
--> Using cache
--> 0c85f166b717
Step 3/7 : WORKDIR /usr/src/app
--> Using cache
--> 986d5698aa1
Step 4/7 : COPY . .
--> 245a599fbfb0
Step 5/7 : RUN npm install
--> Running in 4b0e85c84df8
npm WARN coffee_spi@1.0.0 No repository field.

audited 78 packages in 0.803s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 4b0e85c84df8
--> 02f024244758
Step 6/7 : EXPOSE 8081
--> Running in 22a871d494b9
Removing intermediate container 22a871d494b9
--> 07c5d4e49274
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in 3617a461ce1b
Removing intermediate container 3617a461ce1b
--> 686acedc55e3
Successfully built 686acedc55e3
Successfully tagged employee:latest
vocabls:~/environment/microservices/employee (dev) $
```

The screenshot shows a code editor interface with a sidebar containing project files for a 'MicroservicesIDE - /ho' project. The files include 'customer', 'employee', and 'public' directories, along with 'Dockerfile', 'index.js', 'package-lock.json', and 'package.json'. The main pane displays a terminal output from a 'bash -ip 10-16-10-71.ecx' session:

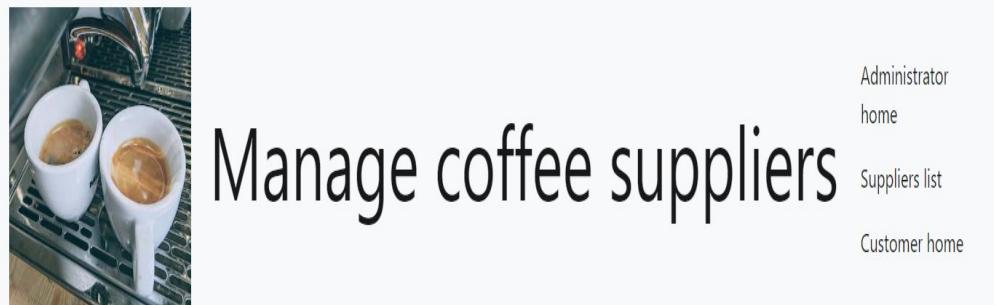
```
vocabls:~/environment $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
employee latest 686acedc55e3 45 seconds ago 82.7MB
customer latest d8471c89baaf About an hour ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
vocabls:~/environment/microservices/employee (dev) $
```

```

bash - *ip-10-16-10-71.ec2.internal* +1
voclabs:~/environment $ dbEndpoint=$(cat ~/environment/microservices/employee/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment $ echo $dbEndpoint
suppliedb.ccylnx0fhcf.us-east-1.rds.amazonaws.com
voclabs:~/environment $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee
docker: Error response from daemon: Conflict. The container name "/employee_1" is already in use by container "74482ca3fa1805ad17a07946dfba415feca72e273e524881712c8efc2faa5f9e". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
voclabs:~/environment $ docker stop employee_1
employee_1
voclabs:~/environment $ docker rm employee_1
employee_1
voclabs:~/environment $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee
1f77e481dabea7a4a47159c0b51f12a3103ebba4164133a904170386e40e705cf
voclabs:~/environment $

```

- To verify that the microservice was running I opened the application in a new browser tab using the public IP address of the AWS Cloud9 instance, followed by :8081/admin/suppliers. I checked that the application loaded successfully and confirmed that it displayed the expected functionality for employees, including buttons to add new suppliers and edit existing ones.
- I tested adding a new supplier and verified that it appeared on the suppliers' page. Then, I tested editing an existing supplier to ensure the updated information was displayed correctly. Additionally, I tested deleting an existing supplier by choosing "Edit," then "Delete this supplier," to ensure the supplier no longer appeared on the suppliers' page after deletion.



## All suppliers

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

[Add a new supplier](#)



# Manage coffee suppliers

Administrator  
home  
Suppliers list  
Customer home

## All suppliers

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

[Add a new supplier](#)



# Manage coffee suppliers

Administrator  
home  
Suppliers list  
Customer home

## All suppliers

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

[Add a new supplier](#)



# Manage coffee suppliers

Administrator  
home  
Suppliers list  
Customer home

## All suppliers

Name	Address	City	State	Email	Phone	
Supreeth Hassan Ravindra	56 Poplar Street, Apt #2	Jersey City	New Jersey	shassanr@stevens.edu	2012047263	<a href="#">edit</a>

[Add a new supplier](#)

The screenshot shows the AWS Lambda Functions IDE interface. On the left, the file structure for the 'employee' service is visible, including files like Dockerfile, index.js, config.json, and package.json. The terminal window on the right displays the following command-line session:

```

bash - "ip-10-16-10-71.ec2.us-east-1.amazonaws.com"
voclab:~/environment $ dbEndpoint=$(cat ~/environment/microservices/employee/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclab:~/environment $ echo $dbEndpoint
supplierdb.ccyln0fhycf.us-east-1.rds.amazonaws.com
voclab:~/environment $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee
docker: Error response from daemon: Conflict. The container name "/employee_1" is already in use by container 74482ca3fa1885ad17a07946dfba415feca72e273e524881712c8efc2faa5f9e. You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
voclab:~/environment $ docker stop employee_1
employee_1
voclab:~/environment $ docker rm employee_1
employee_1
voclab:~/environment $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee
1f77e481dabe2a4a47159c6b51f12a3103ebb4164133a904170386e40e705f
voclab:~/environment $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS                         NAMES
1f77e481dabe        employee            "docker-entrypoint.s"   About a minute ago   Up About a minute   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp   employee_1
9e837092ac25        customer           "docker-entrypoint.s"   About an hour ago    Up About an hour    0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   customer_1
voclab:~/environment $

```

- Task 4.6: To adjust the port number for the employee microservice to prepare for deployment on Amazon ECS I started by editing two files: employee/index.js and employee/Dockerfile, changing the port from 8081 to 8080. This adjustment aligns with the desired port for ECS deployment.
- After updating the files, I rebuilt the Docker image for the employee microservice to incorporate the port change. I first stopped and deleted the existing container using **docker rm -f employee\_1**. Then, ensuring my terminal was in the employee directory I ran **docker build --tag employee .** to rebuild the Docker image with the new port setting.
- This ensured the Docker image for the employee microservice was updated with the correct port, preparing it for deployment on Amazon ECS where it will run on port 8080.

The screenshot shows the AWS Lambda Functions IDE interface. On the left, the file structure for the 'supplier' service is visible, including files like Dockerfile, index.js, config.json, and package.json. The terminal window on the right displays the following command-line session:

```

bash - "ip-10-16-10-71.ec2.us-east-1.amazonaws.com"
voclab:~/environment $ 
voclab:~/environment $ 
voclab:~/environment $ 
voclab:~/environment $ 

```

The screenshot shows the AWS Lambda Functions IDE interface. On the left, the file structure of the 'MicroservicesIDE - /ho' project is displayed, including 'microservices' (with 'customer' and 'employee' subfolders), 'app' (with 'config', 'controller', 'models', 'node\_modules', 'public', 'views' subfolders), and files like 'Dockerfile', 'index.js', 'package-lock.json', 'package.json', 'labuser.pem', and 'README.md'. The right side features a code editor for the 'Dockerfile' which contains the following content:

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

Below the code editor is a terminal window titled 'bash - "ip-10-16-10-71.ec2.internal"' showing the command history:

```
voclabs:~/environment $ clear
voclabs:~/environment $ docker rm -f employee_1
employee_1
voclabs:~/environment $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
Sending build context to Docker daemon 9.014MB
Step 1/7 : FROM node:11-alpine
--> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
--> Using cache
--> 0e85f1b6b717
Step 3/7 : WORKDIR /usr/src/app
--> Using cache
--> 986d5698aa61
Step 4/7 : COPY . .
--> 8a6f2ec0088a
Step 5/7 : RUN npm install
--> Running in c0cef9dfdc2d
--> npm WARN coffee_mpl@1.0.0 No repository field.

audited 78 packages in 0.721s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
Removing intermediate container c0cef9dfdc2d
--> 571cd1a470c7
Step 6/7 : EXPOSE 8080
--> Running in 2640fadecf0d
Removing intermediate container 2640fadecf0d
--> 5257ba2cffb4
Step 7/7 : CMD [ "npm", "run", "start" ]
--> Running in 6947e1794f15
Removing intermediate container 6947e1794f15
--> fb6c47aad857
Successfully built fb6c47aad857
Successfully tagged employee:latest
voclabs:~/environment/microservices/employee (dev) $
```

This screenshot shows the same AWS Lambda Functions IDE interface as the previous one, but the terminal output has been updated to show the result of running the 'docker build' command. The terminal now displays:

```
voclabs:~/environment/microservices/employee (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
employee latest fb6c47aad857 16 seconds ago 82.7MB
<none> <none> 686acedc56e3 23 minutes ago 82.7MB
customer latest d8471c89baaf 2 hours ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment/microservices/employee (dev) $
```

This screenshot shows the AWS Lambda Functions IDE interface again. The terminal output has been updated to show the result of running 'docker images' after the image was successfully tagged. The terminal now displays:

```
voclabs:~/environment/microservices/employee (dev) $ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
employee latest fb6c47aad857 16 seconds ago 82.7MB
<none> <none> 686acedc56e3 23 minutes ago 82.7MB
customer latest d8471c89baaf 2 hours ago 82.7MB
node 11-alpine f18da2f58c3d 4 years ago 75.5MB
voclabs:~/environment/microservices/employee (dev) $
```

```

voclabs:~/environment/microservices/employee (dev) $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ git add .
voclabs:~/environment/microservices/employee (dev) $ git commit -m "Updated employee microservice with new Dockerfile and source code changes"
[dev b10b1d3] Updated employee microservice with new Dockerfile and source code changes
9 files changed, 28 insertions(+), 20 deletions(-)
create mode 100644 employee/Dockerfile
voclabs:~/environment/microservices/employee (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), 1.65 KiB | 845.00 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
  cbf8112..b10b1d3 dev -> dev
voclabs:~/environment/microservices/employee (dev) $

```

## Task 4.7: Checking the code changes into CodeCommit.

Developer Tools > CodeCommit > Repositories > microservices > Commits > b10b1d3a94012056160ce1717aeda6bbdc1fee72

**Commit b10b1d3a94012056160ce1717aeda6bbdc1fee72**

Author: Supreeth (supreeth.hr27@gmail.com)

Authored date: 1 minute ago

Committer: Supreeth (supreeth.hr27@gmail.com)

Commit date: 1 minute ago

Parent commit: cbf811296bd0de84f68a98698c9afcd2942c5e290

Commit message: Updated employee microservice with new Dockerfile and source code changes

employee/Dockerfile

1 + FROM node:11-alpine

The screenshot shows the AWS CodeCommit interface. On the left, there's a sidebar with navigation links like 'Source • CodeCommit', 'Getting started', 'Repositories', 'Code', 'Pull requests', 'Commits' (which is selected), 'Branches', 'Git tags', 'Settings', 'Approval rule templates', 'Artifacts • CodeArtifact', 'Build • CodeBuild', 'Deploy • CodeDeploy', 'Pipeline • CodePipeline', and 'Settings'. The main area displays two code files. The first file is 'employee/Dockerfile' with the following content:

```

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

```

The second file is 'employee/app/controller/supplier.controller.js'. A diff view shows changes between two versions:

```

@@ -22,7 +22,7 @@
 22 22             Supplier.create(supplier, (err, data) => {
 23 23                 if (err)
 24 24                     res.render("500", {message: "Error occurred while creating the Supplier."});
 25 -                   else res.redirect("/suppliers");
 25 +                   else res.redirect("/admin/suppliers");
 26 26             });

```

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

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

The screenshot shows the AWS CodeWhisperer IDE interface. On the left, there's a sidebar with project files: 'MicroservicesIDE - /home/ec2-user/environment', 'microservices', 'customer', 'employee', 'app', 'config', 'JS config.js', 'controller', 'models', 'node\_modules', 'public', 'views', 'Dockerfile', 'JS index.js', 'package-lock.json', 'package.json', 'labuser.pem', and 'README.md'. The main area has a terminal window with the following content:

```

bash - "ip-10-16-10-71.ec2.internal"
voclabs:~/environment $ account_id=$(aws sts get-caller-identity | grep Account|cut -d '"' -f4)
voclabs:~/environment $ echo $account_id
579794416250
voclabs:~/environment $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclabs:~/environment $

```

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

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

**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.

579794416250.dkr.ecr.us-east-1.amazonaws.com/

8 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

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

**Disabled**

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

**Created private repository**  
employee has been successfully created in private registry

**Amazon Elastic Container Registry**

**Private registry**

**Repositories**

**Public registry**

**ECR public gallery**

**Amazon ECS**

**Amazon EKS**

**Getting started**

**Documentation**

**Private repositories**

**Repositories (2)**

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type
customer	579794416250.dkr.ecr.us-east-1.amazonaws.com/customer	April 26, 2024, 20:45:30 (UTC-04)	Disabled	Manual	AES-256
employee	579794416250.dkr.ecr.us-east-1.amazonaws.com/employee	April 26, 2024, 20:46:00 (UTC-04)	Disabled	Manual	AES-256

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

### Edit JSON

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

Reset Close Save

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

### Amazon Elastic Container Registry

#### Permissions

Amazon ECR > Private registry > Repositories > customer > Permissions

### Permissions

Statement 1

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

Edit policy JSON Edit

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

### Amazon Elastic Container Registry

#### Permissions

Amazon ECR > Private registry > Repositories > employee > Permissions

### Permissions

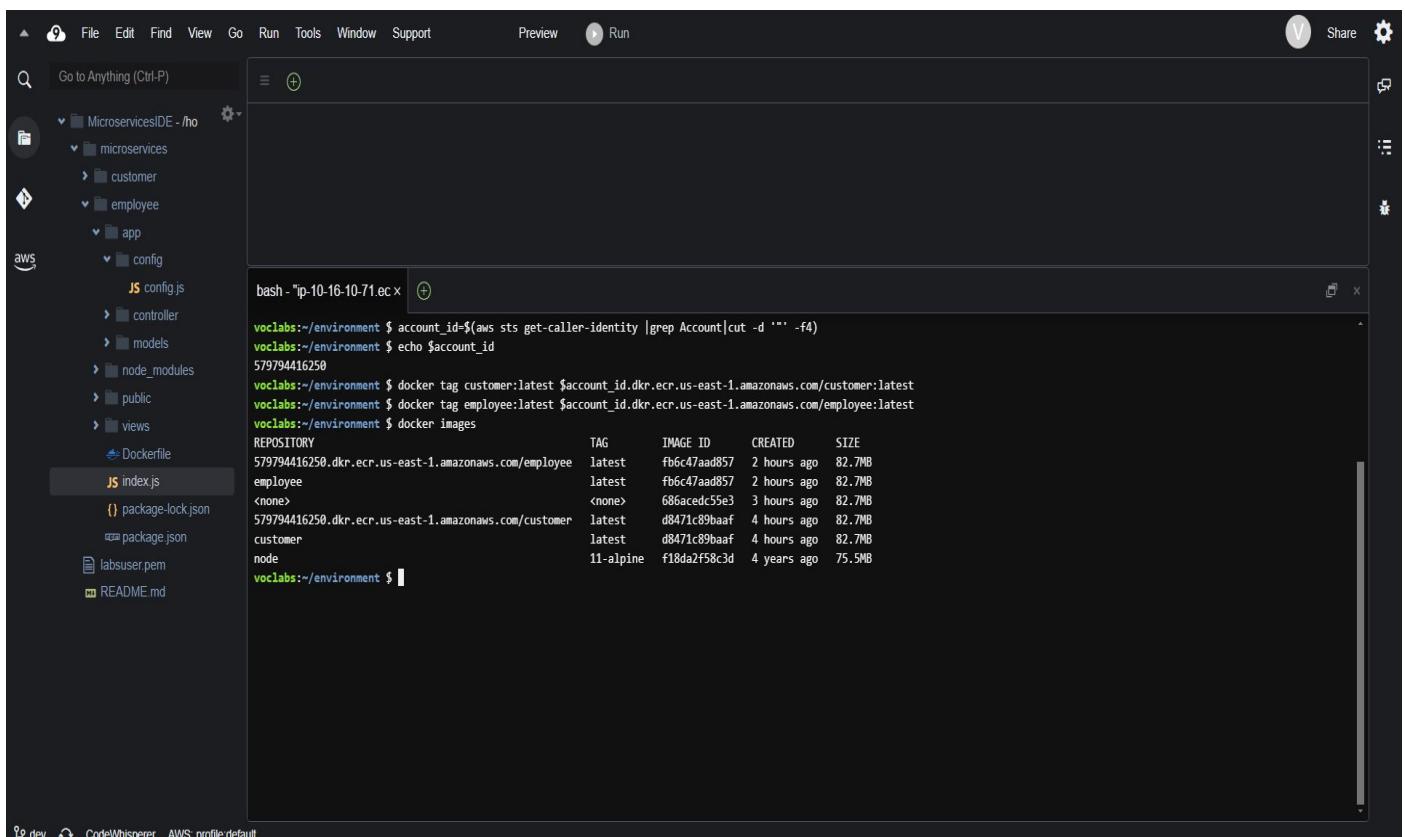
Statement 1

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

Edit policy JSON Edit

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

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



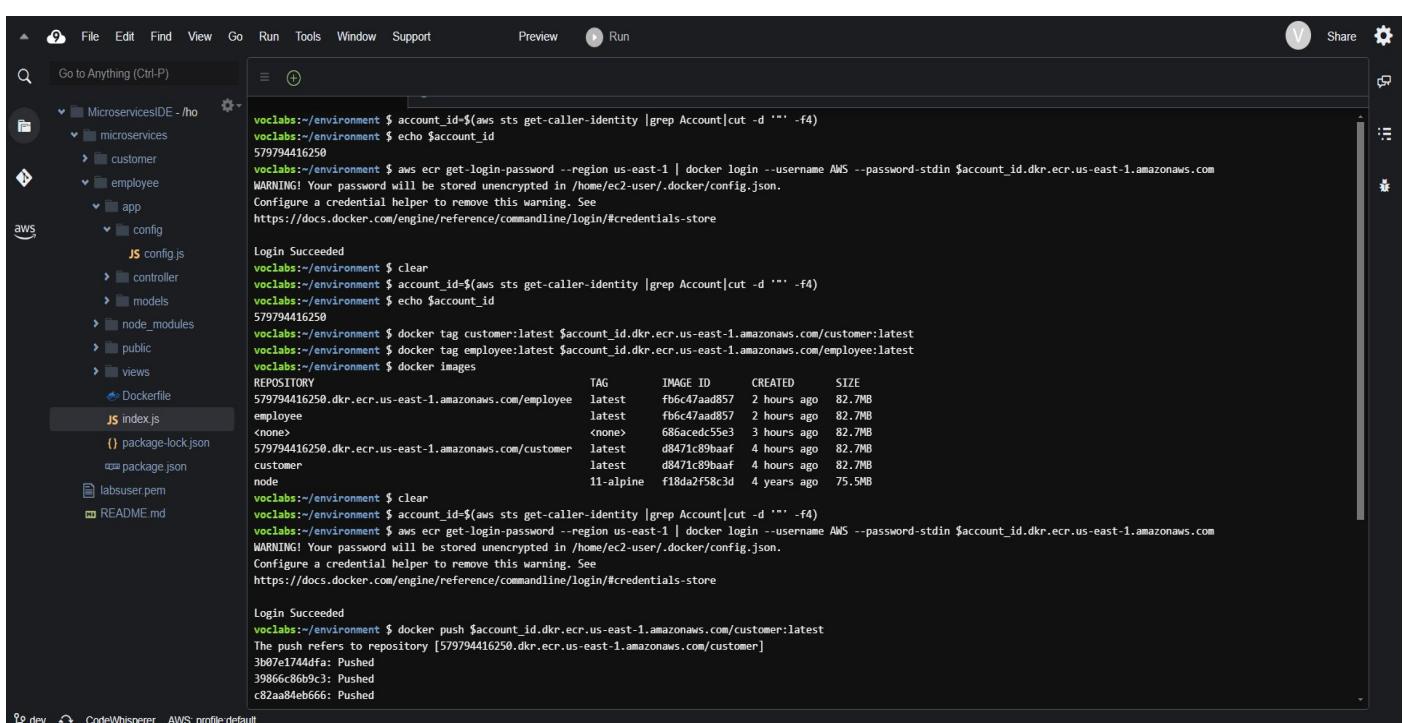
A screenshot of the AWS Lambda function code editor. The left sidebar shows the project structure for 'MicroservicesIDE - /home/voclabs/microservices'. The main editor area contains a terminal window with the following command and output:

```

bash - "ip-10-16-10-71.ec2.internal"
voclabs:~/environment $ account_id=$(aws sts get-caller-identity | grep Account | cut -d '"' -f4)
voclabs:~/environment $ echo $account_id
579794416250
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
579794416250.dkr.ecr.us-east-1.amazonaws.com/employee    latest   fb6c47aad857  2 hours ago  82.7MB
employee                                     latest   fb6c47aad857  2 hours ago  82.7MB
<none>                                         <none>  686ac6edc5e3  3 hours ago  82.7MB
579794416250.dkr.ecr.us-east-1.amazonaws.com/customer    latest   d8471c89baaf  4 hours ago  82.7MB
customer                                      latest   d8471c89baaf  4 hours ago  82.7MB
node                                           11-alpine f18da2f58c3d  4 years ago  75.5MB
voclabs:~/environment $

```

The status bar at the bottom indicates 'dev' and 'CodeWhisperer AWS profile:default'.



A screenshot of the AWS Lambda function code editor, identical to the one above but with additional output from the Docker push command:

```

voclabs:~/environment $ account_id=$(aws sts get-caller-identity | grep Account | cut -d '"' -f4)
voclabs:~/environment $ echo $account_id
579794416250
voclabs:~/environment $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclabs:~/environment $ clear
voclabs:~/environment $ account_id=$(aws sts get-caller-identity | grep Account | cut -d '"' -f4)
voclabs:~/environment $ echo $account_id
579794416250
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
579794416250.dkr.ecr.us-east-1.amazonaws.com/employee    latest   fb6c47aad857  2 hours ago  82.7MB
employee                                     latest   fb6c47aad857  2 hours ago  82.7MB
<none>                                         <none>  686ac6edc5e3  3 hours ago  82.7MB
579794416250.dkr.ecr.us-east-1.amazonaws.com/customer    latest   d8471c89baaf  4 hours ago  82.7MB
customer                                      latest   d8471c89baaf  4 hours ago  82.7MB
node                                           11-alpine f18da2f58c3d  4 years ago  75.5MB
voclabs:~/environment $ clear
voclabs:~/environment $ account_id=$(aws sts get-caller-identity | grep Account | cut -d '"' -f4)
voclabs:~/environment $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclabs:~/environment $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
The push refers to repository [579794416250.dkr.ecr.us-east-1.amazonaws.com/customer]
3bb07e1744dfa: Pushed
39866c86b9c3: Pushed
c82aa84eb666: Pushed

```

The status bar at the bottom indicates 'dev' and 'CodeWhisperer AWS profile:default'.

File Edit Find View Go Run Tools Window Support Preview Run

```

Go to Anything (Ctrl-P)

REPOSITORY TAG IMAGE ID CREATED SIZE
579794416250.dkr.ecr.us-east-1.amazonaws.com/employee latest fb6c47aad857 2 hours ago 82.7MB
employee latest fb6c47aad857 2 hours ago 82.7MB
579794416250.dkr.ecr.us-east-1.amazonaws.com/customer latest d8471c89baaf 4 hours ago 82.7MB
customer latest d8471c89baaf 4 hours ago 82.7MB
node latest f18da2f58c3d 4 years ago 75.5MB

voclab:~/environment $ clear
voclab:~/environment $ account_id=$(aws sts get-caller-identity | grep AccountId | cut -d ":" -f4)
voclab:~/environment $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclab:~/environment $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
The push refers to repository [579794416250.dkr.ecr.us-east-1.amazonaws.com/customer]
3b87e1744dfa: Pushed
39866c8b09c3: Pushed
c82aa84eb666: Pushed
d81d715330b7: Pushed
1dc7f3bb09a4: Pushed
dcaceb729824: Pushed
f1b5933fe4b5: Pushed
latest digest: sha256:a454319a98f1384a180d69999ca6b1421e471b5e266c03ff8a3d7310d07039c size: 1783
voclab:~/environment $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
The push refers to repository [579794416250.dkr.ecr.us-east-1.amazonaws.com/employee]
1023523a4167: Pushed
bdff0228e2dd5: Pushed
c82aa84eb666: Pushed
d81d715330b7: Pushed
1dc7f3bb09a4: Pushed
dcaceb729824: Pushed
f1b5933fe4b5: Pushed
latest digest: sha256:417ce7e7f2cccd746cf8fa5249e8ffac8aa53592dba7c3f03f3a2657e03b08 size: 1783
voclab:~/environment $

```

dev CodeWhisperer AWS profile:default

aws Services Search [Alt+S]

Amazon Elastic Container Registry N. Virginia voclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

Amazon ECR > Private registry > Repositories > customer

### customer

View push commands Edit

**Images (1)**

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	April 26, 2024, 20:56:19 (UTC-04)	27.70	<a href="#">Copy URI</a>	<a href="#">sha256:9a454319a98f1...</a>

https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1 © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Services Search [Alt+S]

Amazon Elastic Container Registry N. Virginia voclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

Amazon ECR > Private registry > Repositories > employee

### employee

View push commands Edit

**Images (1)**

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	April 26, 2024, 20:57:04 (UTC-04)	27.70	<a href="#">Copy URI</a>	<a href="#">sha256:417ce7e7f2cccd7...</a>

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

- Task 5.2: To create an Amazon ECS cluster I started by creating a serverless AWS Fargate cluster named "microservices-serverlesscluster." During the setup I configured the cluster to use LabVPC, PublicSubnet1, and PublicSubnet2, ensuring no other subnets were included in the configuration. I made sure not to select Amazon EC2 instances or ECS Anywhere adhering to the serverless Fargate architecture.
- After choosing to create the cluster I used the "View in CloudFormation" button to monitor the status of the stack that creates the cluster. I waited until the status changed to CREATE\_COMPLETE before proceeding, ensuring the cluster was successfully created.

The screenshot shows the AWS Elastic Container Service (ECS) Clusters page. At the top, a green banner displays the message: "Cluster microservices-serverlesscluster has been created successfully." Below the banner, the page title is "Amazon Elastic Container Service > Clusters". A sub-header "Clusters (1) Info" is followed by a table with one row. The table columns are: Cluster, Services, Tasks, Container instances, CloudWatch monitoring, and Capacity. The row contains: "microservices-serverlesscluster", 0, "No tasks running", 0 EC2, Default, and No d. On the left sidebar, there are links for Clusters, Namespaces, Task definitions, Account settings, and Install AWS Copilot. The bottom of the page includes links for Documentation, Discover products, CloudShell, Feedback, and a copyright notice for 2024, Amazon Web Services, Inc. or its affiliates.

The screenshot shows the AWS CloudFormation Stacks page. The left sidebar includes links for CloudFormation, Stacks, Drifts, StackSets, Exports, Application Composer, Registry, and Spotlight. The main area shows the stack "Infra-ECS-Cluster-microservices-serverlesscluster-c6f1a25a". The "Events" tab is selected, displaying 7 events. The events table has columns: Timestamp, Logical ID, Status, and Detailed status. The events listed are:

Timestamp	Logical ID	Status	Detailed status
2024-04-26 21:08:56 UTC-0400	Infra-ECS-Cluster-microservices-serverlesscluster-c6f1a25a	<span style="color: green;">CREATE_COMPLETE</span>	-
2024-04-26 21:08:54 UTC-0400	ECSCluster	<span style="color: green;">CREATE_COMPLETE</span>	-
2024-04-26 21:08:16 UTC-0400	Infra-ECS-Cluster-microservices-serverlesscluster-c6f1a25a	<span style="color: blue;">CREATE_IN_PROGRESS</span>	<span style="color: green;">CONFIGURATION_COMPLETE</span>
			.....
			.....
			.....

At the bottom of the page, there are links for CloudShell, Feedback, and a copyright notice for 2024, Amazon Web Services, Inc. or its affiliates.

The screenshot shows the 'Create repository' wizard in the AWS CodeCommit console. The 'Repository settings' step is active. The 'Repository name' field contains 'deployment'. The 'Description - optional' field is empty. Below it, there's a note about 1,000 character limit. The 'Tags' section has a 'Add tag' button. Under 'Additional configuration', there's a link to 'AWS KMS key'. At the bottom, there are standard AWS navigation links like CloudShell and Feedback, and a footer with copyright information.

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

The screenshot shows the 'Repositories' list page in the AWS CodeCommit console. The left sidebar shows the 'CodeCommit' service selected. The main area displays a table of repositories. The 'deployment' repository was created just now and has HTTPS, SSH, and GRC clone URLs. The 'microservices' repository was created 2 hours ago and also has HTTPS, SSH, and GRC clone URLs. The table includes columns for Name, Description, Last modified, Clone URL, and AWS KMS Key.

Name	Description	Last modified	Clone URL	AWS KMS Key
deployment	-	Just now	HTTPS, SSH, HTTPS (GRC)	arn:aws:kms:us-east-1:579794416250:key/b10f5c05-0927-4d56-b8ac-458033de8de6
microservices	-	2 hours ago	HTTPS, SSH, HTTPS (GRC)	arn:aws:kms:us-east-1:579794416250:key/b10f5c05-0927-4d56-b8ac-458033de8de6

A screenshot of the AWS Lambda CodeWhisperer interface. The left sidebar shows a project structure with a 'deployment' folder selected. The main area is a terminal window titled 'bash - "ip-10-16-10-71.ecx"'. The terminal output shows a git init process for a new repository, with hints about branch naming and configuration. It also shows the creation of a 'dev' branch and a message indicating no commits yet.

```

bash - "ip-10-16-10-71.ecx"
1dc7f3bb094: Pushed
dcae679824: Pushed
f1b5933fe4b5: Pushed
latest: digest: sha256:417ce7e7f2cccd746cf8fa5249e8ffac8aa53592dba87c3f03f3a2657e03b08 size: 1783
voclabs:~/environment $ clear
voclabs:~/environment $ clear
voclabs:~/environment/deployment $ cd ~/environment/deployment
voclabs:~/environment/deployment $ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint: git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint: git branch -m <name>
Initialized empty Git repository in /home/ec2-user/environment/deployment/.git/
voclabs:~/environment/deployment (master) $ git branch -m dev
voclabs:~/environment/deployment (dev) $ git status
On branch dev

No commits yet

nothing to commit (create/copy files and use "git add" to track)
voclabs:~/environment/deployment (dev) $

```

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

A screenshot of the AWS Lambda CodeWhisperer interface. The left sidebar shows a project structure with a 'taskdef-customer.json' file selected in the 'deployment' folder. The main area is a code editor showing the JSON content of the task definition file. The file defines a container definition for a 'customer' service, mapping port 8080 to 8080, and using AWSLogs as the log configuration provider. Line 37 contains the placeholder '579794416250' and line 9 contains the placeholder 'supplierdb.ccylinx0fhcy.us-east-1.rds.amazonaws.com'.

```

{
  "containerDefinitions": [
    {
      "name": "customer",
      "image": "customer",
      "environment": [
        {
          "name": "APP_DB_HOST",
          "value": "supplierdb.ccylinx0fhcy.us-east-1.rds.amazonaws.com"
        }
      ],
      "essential": true,
      "portMappings": [
        {
          "hostPort": 8080,
          "protocol": "tcp",
          "containerPort": 8080
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "true",
          "awslogs-group": "awslogs-capstone",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "awslogs-capstone"
        }
      }
    }
  ]
}

```

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

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

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

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

```

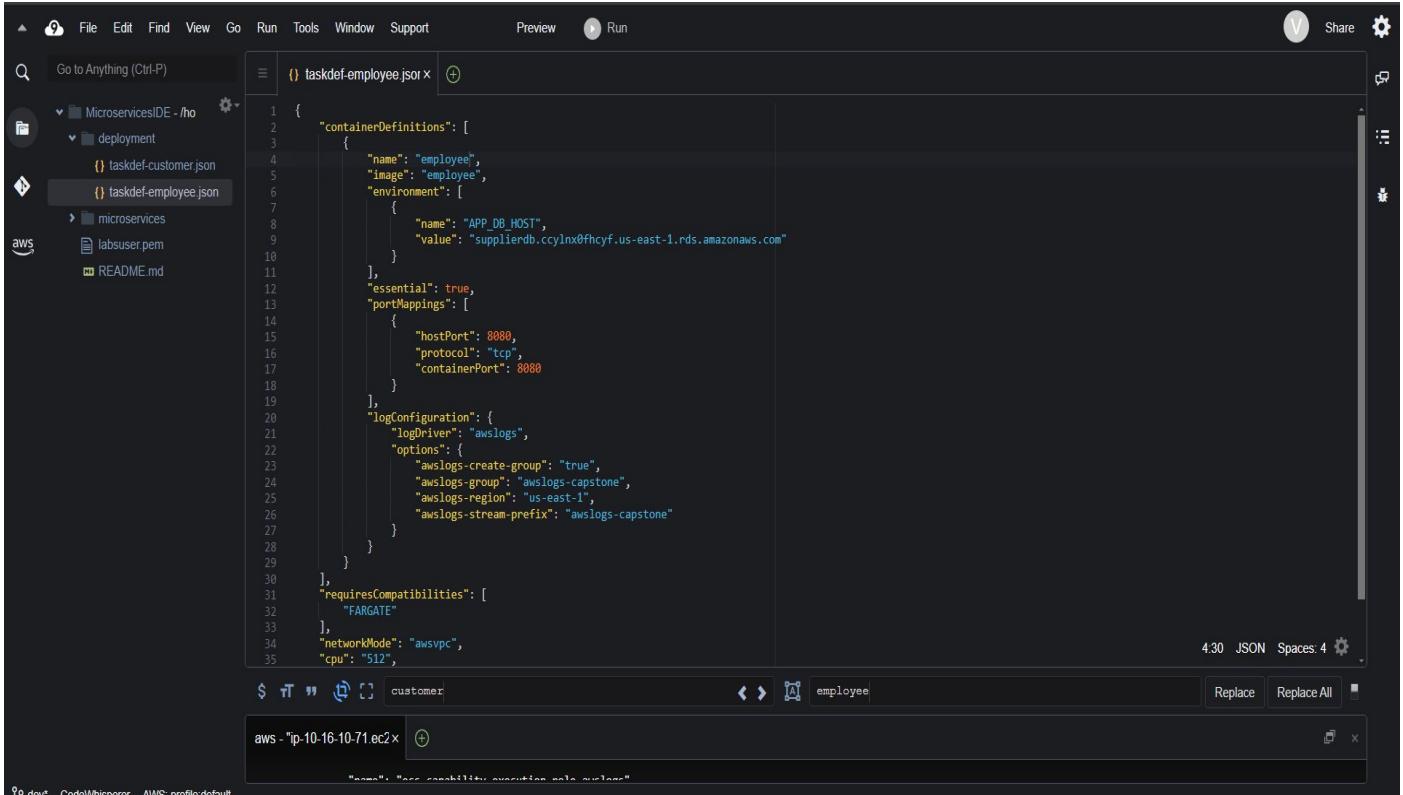
1  {
2    "containerDefinitions": [
3      {
4        "name": "customer",
5
6          "name": "ecs.capability.execution-role-awslogs"
7        },
8        {
9          "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
10         },
11        {
12          "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
13        },
14        {
15          "name": "ecs.capability.task-eni"
16        },
17        {
18          "name": "com.amazonaws.ecs.capability.docker-remote-api.1.29"
19        }
20      ],
21      "placementConstraints": [],
22      "compatibilities": [
23        "EC2",
24        "FARGATE"
25      ],
26      "requiresCompatibilities": [
27        "FARGATE"
28      ],
29      "cpu": "512",
30      "memory": "1024",
31      "registeredAt": "2024-04-27T01:52:06.558000+00:00",
32      "registeredBy": "arn:aws:sts::579794416250:assumed-role/voclabs/user3233567=Hassan_Ravindra,_Supreeth"
33    }
34  }
35  (END)

```

Task definition	Status of last revision
<a href="#">customer-microservice</a>	ACTIVE

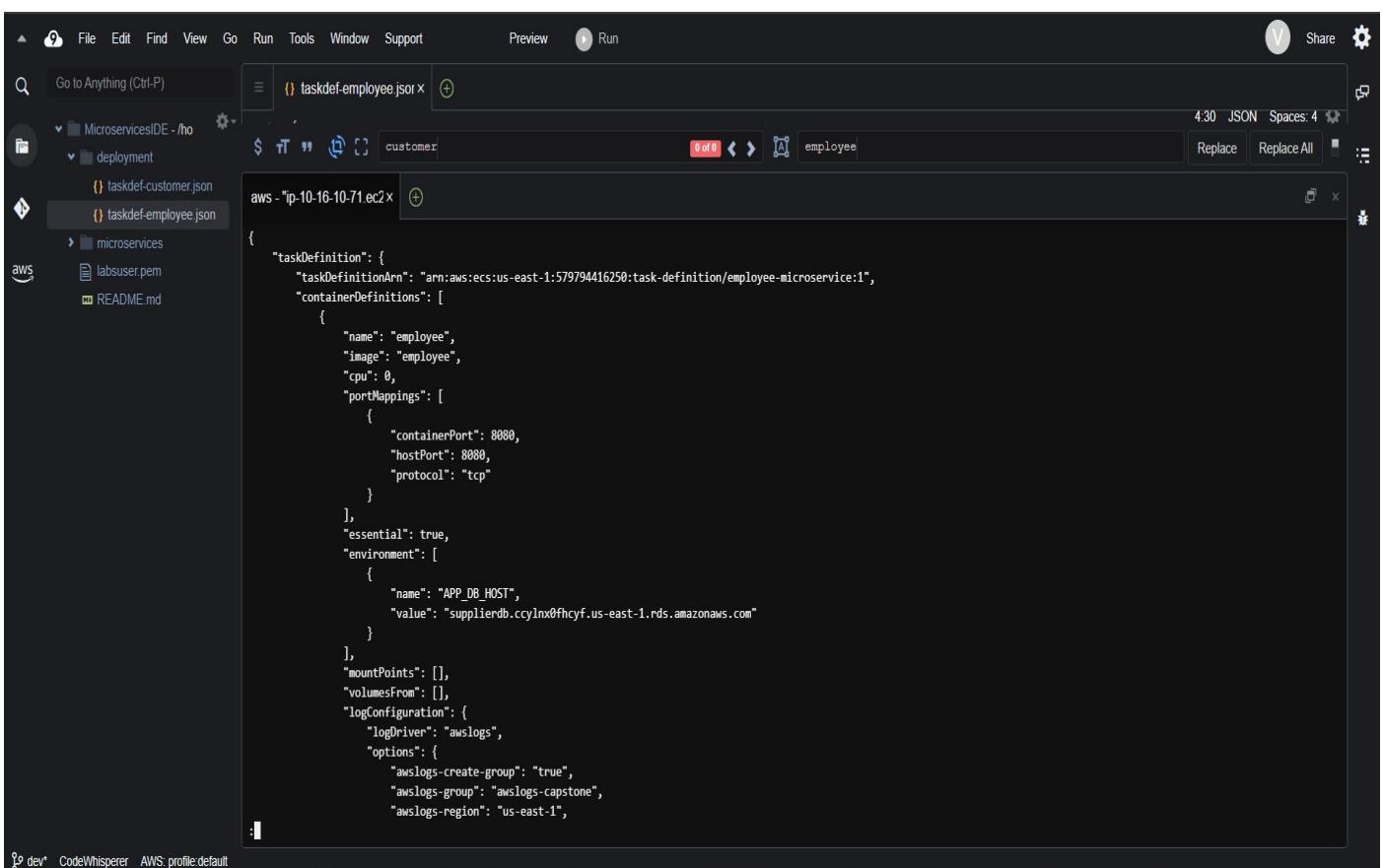
- I created a `taskdef-employee.json` specification file in the deployment directory, using the same JSON code from `taskdef-customer.json`. After pasting the code, I changed three occurrences of "customer" to "employee" to tailor the task definition to the employee microservice. I registered

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



The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes File, Edit, Find, View, Go, Run, Tools, Window, Support, Preview, and Run buttons. The left sidebar shows a project structure under 'MicroservicesIDE - /ho': deployment, taskdef-customer.json, taskdef-employee.json, microservices, labsuser.pem, and README.md. The main editor window displays the JSON content of taskdef-employee.json. A search bar at the bottom has 'aws - "ip-10-16-10-71.ec2x"' entered. The status bar at the bottom right shows '4:30 JSON Spaces: 4'.

```
1 {
  "containerDefinitions": [
    {
      "name": "employee",
      "image": "employee",
      "environment": [
        {
          "name": "APP_DB_HOST",
          "value": "supplierdb.ccylnxfhcyf.us-east-1.rds.amazonaws.com"
        }
      ],
      "essential": true,
      "portMappings": [
        {
          "hostPort": 8080,
          "protocol": "tcp",
          "containerPort": 8080
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "true",
          "awslogs-group": "awslogs-capstone",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "awslogs-capstone"
        }
      },
      "requiresCompatibilities": [
        "FARGATE"
      ],
      "networkMode": "awsvpc",
      "cpu": "512",
      "memory": "128"
    }
  ],
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "networkMode": "awsvpc",
  "cpu": "512",
  "memory": "128"
}
```



The screenshot shows the AWS Cloud9 IDE interface with a preview tab open for the taskdef-employee.json file. The preview tab displays the JSON content with some fields redacted. The status bar at the bottom right shows '4:30 JSON Spaces: 4'.

```
$ aws - "ip-10-16-10-71.ec2x" customer
```

```
{
  "taskDefinition": {
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:570794416250:task-definition/employee-microservice:1",
    "containerDefinitions": [
      {
        "name": "employee",
        "image": "employee",
        "cpu": 0,
        "portMappings": [
          {
            "containerPort": 8080,
            "hostPort": 8080,
            "protocol": "tcp"
          }
        ],
        "essential": true,
        "environment": [
          {
            "name": "APP_DB_HOST",
            "value": "supplierdb.ccylnxfhcyf.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"
          }
        }
      }
    ]
}
```

The screenshot shows the AWS Elastic Container Service (ECS) Task Definitions page. On the left, there's a sidebar with links for Clusters, Namespaces, Task definitions (which is selected), Account settings, Install AWS Copilot, Amazon ECR, and AWS Batch. The main content area shows a table of task definitions with two entries:

Task definition	Status of last revision
<a href="#">customer-microservice</a>	ACTIVE
<a href="#">employee-microservice</a>	ACTIVE

At the bottom of the page, there are links for CloudShell and Feedback, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

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

The screenshot shows the AWS Lambda CodeWhisperer IDE interface. On the left, there's a sidebar with AWS services like MicroservicesIDE, deployment, and AWS Lambda. The main area has a code editor with the file `appspec-customer.yaml` open. The code is as follows:

```

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

```

Below the code editor is a terminal window with the following command history:

```

aws - "ip-10-16-10-71.ec2.x" customer
{
  "taskDefinition": {
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:579794416250:task-definition/employee-microservice:1",
    "containerDefinitions": [
      {
        "name": "employee",
        :

```

The terminal shows the command `aws - "ip-10-16-10-71.ec2.x" customer` followed by a JSON object representing the task definition. The JSON includes the task definition ARN and container definitions for the "employee" service.

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

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

The screenshot shows the AWS Lambda IDE interface. On the left, the file tree displays several files: MicroservicesIDE /home, deployment, appspec-employee.yaml, taskdef-customer.json, taskdef-employee.json, microservices, labsuser.pem, and README.md. The main editor window shows the content of appspec-employee.yaml:

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

Below the editor, a terminal window shows the AWS Lambda function code:

```
aws ->ip-10-16-10-71.ec2x
```

```
{
  "taskDefinition": {
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:579794416250:task-definition/employee-microservice:1",
    "containerDefinitions": [
      {
        "name": "employee",
        :
```

The bottom status bar indicates the session is in dev mode, using CodeWhisperer, and the AWS profile is default. The time is 9:30 and there are 4 spaces.

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

The screenshot shows the AWS Lambda CodeWhisperer IDE interface. On the left, the file tree displays 'MicroservicesIDE - /ho' with subfolders 'deployment' containing 'appspec-worker.yaml', 'appspec-worker.yml', 'taskdef-worker.json', and 'taskdef-worker.json'. Below these are 'microservices', 'labuser.pem', and 'README.md'. In the center, two tabs are open: 'taskdef-worker.json' and 'taskdef-worker.json'. The code in 'taskdef-worker.json' is as follows:

```

1  {
2      "containerDefinitions": [
3          {
4              "name": "customer",
5              "image": "<IMAGE1_NAME>",
6              "environment": [
7                  {
8                      "name": "APP_DB_HOST",
9                      "value": "supplierdb.ccylinx0fhcyf.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 }

```

Below the code editor, there are tabs for 'customer' and 'employee'. At the bottom, there are 'Replace' and 'Replace All' buttons. The status bar at the bottom right shows '5:37 JSON Spaces: 4'.

The screenshot shows the AWS Lambda CodeWhisperer IDE interface with a terminal window open. The terminal output shows the following sequence of commands and their results:

```

aws - "ip-10-16-10-71.ec2x" + aws

{
  "taskDefinition": {
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:579794416250:task-definition/employee-microservice:1",
    "containerDefinitions": [
      {
        "name": "employee",
        ...
      }
    ]
  }
}

```

Below the terminal, the file tree is identical to the first screenshot. The status bar at the bottom right shows '5:37 JSON Spaces: 4'.

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

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

EC2 > Target groups > Create target group

Step 1 Specify group details

Step 2 Register targets

## Specify group details

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

### Basic configuration

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

#### Choose a target type

- Instances
  - Supports load balancing to instances within a specific VPC.
  - Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.
- IP addresses
  - Supports load balancing to VPC and on-premises resources.
  - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
  - Offers flexibility with microservice based architectures, simplifying inter-application communication.
  - Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
- Lambda function
  - Facilitates routing to a single Lambda function.
  - Accessible to Application Load Balancers only.

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

CloudShell Feedback

customer-tg-one

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

#### Protocol : Port

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

HTTP 8080 1-65535

#### IP address type

Only targets with the indicated IP address type can be registered to this target group.

IPv4  
 IPv6

#### VPC

Select the VPC that hosts the load balancer. Only VPCs that support the IP address type selected above are available in this list. On the [Register targets](#) page, you can register IP addresses from this VPC, or from private IP addresses located outside of this load balancer's VPC (such as a peered VPC, EC2-Classic, or on-premises targets that are reachable over Direct Connect or VPN).

LabVPC  
 vpc-0930b2fdf65345dee  
 IPv4 VPC CIDR: 10.16.0.0/16

#### Protocol version

HTTP1  
 Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.  
 HTTP2  
 Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC,

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

CloudShell Feedback

EC2 > Target groups > Create target group

Step 1 Specify group details

Step 2 Register targets

## Register targets

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

### IP addresses

#### Step 1: Choose a network

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

#### Network

LabVPC  
 vpc-0930b2fdf65345dee  
 IPv4 VPC CIDR: 10.16.0.0/16

#### Step 2: Specify IPs and define ports

You can manually enter IP addresses from the selected network.

Enter an IPv4 address from a VPC subnet.

10.16.0. Remove

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

**customer-tg-one**

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

**Target groups (2) Info**

Name	ARN	Port	Protocol	Target type	Load balancer
customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-two/8ae0d658b1c92f7	8080	HTTP	IP	None associated
customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-one/8ae0d658b1c92f7	8080	HTTP	IP	None associated

**0 target groups selected**

Select a target group above.

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

The screenshot shows the AWS EC2 Target Groups page. The left sidebar includes sections for EC2 Dashboard, EC2 Global View, Events, Console-to-Code, Instances, Images, and Elastic Block Store. The main content area displays a table titled "Target groups (4) Info" with columns: Name, ARN, Port, Protocol, Target type, and Load balancer. The table lists four target groups: "employee-tg-two", "employee-tg-one", "customer-tg-two", and "customer-tg-one", all associated with "None". A modal window titled "0 target groups selected" is open, instructing the user to "Select a target group above."

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

The screenshot shows the AWS Security Groups page. The left sidebar includes sections for Services, CloudShell, and Feedback. The main content area shows the details for a security group named "sg-03bc4d5baf12c5aa5 - microservices-sg". The "Details" section includes fields for Security group name, Owner, Security group ID, Inbound rules count, Description, and VPC ID. Below this, tabs for "Inbound rules", "Outbound rules", and "Tags" are visible. The "Inbound rules (2)" table lists two rules: one for port 8080 using Custom TCP and another for port 80 using HTTP.

**Successfully created load balancer: microservicesLB**

It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

EC2 > Load balancers > microservicesLB

## microservicesLB

**Details**

Load balancer type	Status	VPC	IP address type
Application	Provisioning	vpc-0930b2fdf65345dee	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z355XDOTRQ7X7K	subnet-00a87838db6561c0 us-east-1b (use1-az4)	April 27, 2024, 15:52 (UTC-04:00)
		subnet-079b846921b332a35 us-east-1a (use1-az2)	
Load balancer ARN		DNS name	
arn:aws:elasticloadbalancing:us-east-1:579794416250:loadbalancer/app/microservicesLB/5beeb23e6d4c3b12	3e6d4c3b12	Info	microservicesLB-103312331.us-east-1.elb.amazonaws.com (A Record)

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

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

Step 5 Review and create

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

1 - 50000

**Listener rules (2) Info**

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

**Rule limits**

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

Cancel Previous Next

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

HTTP:8080 microservicesLB

Listener ARN

arn:aws:elasticloadbalancing:us-east-1:579794416250:listener/app/microservicesLB/5beeb23e6d4c3b12/15216122989b5776

**Rules Tags**

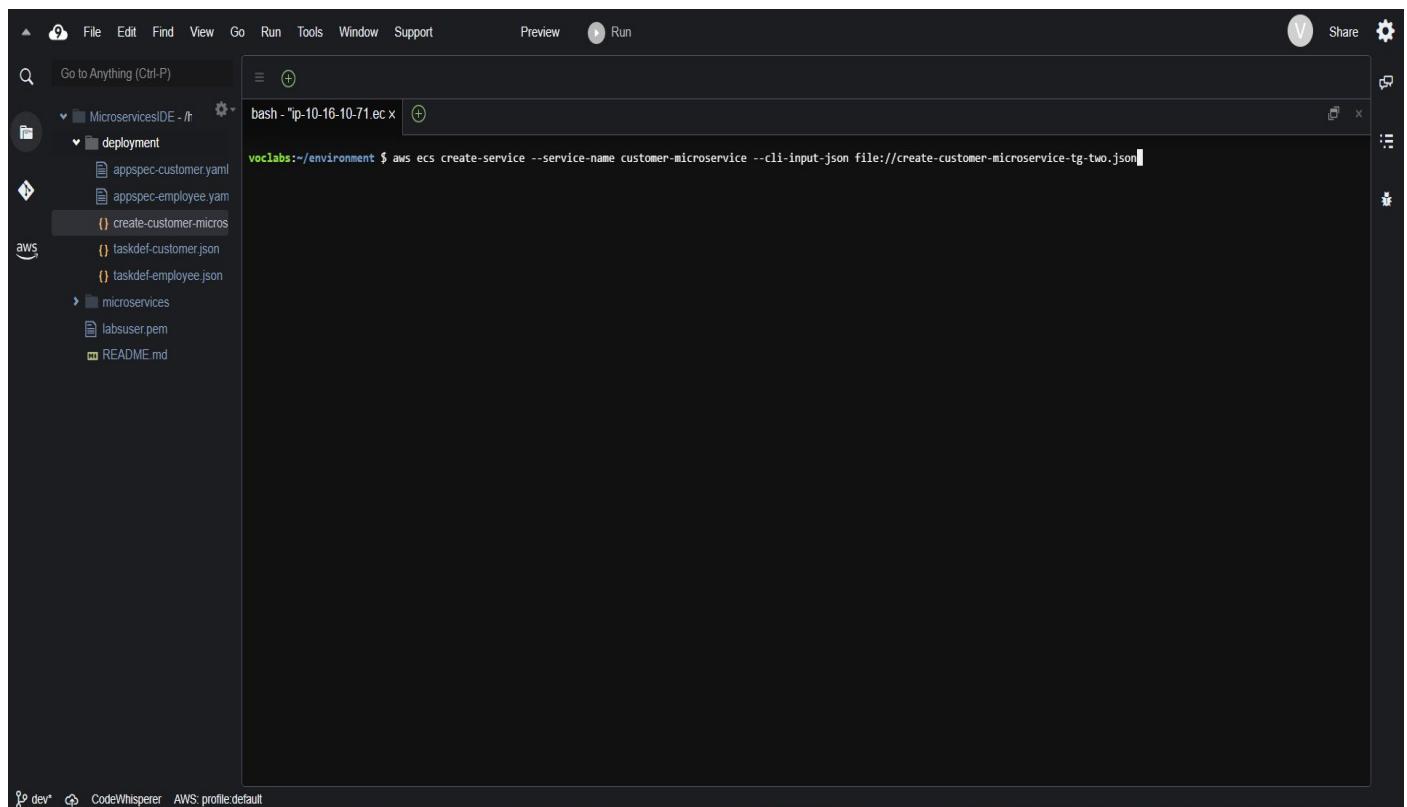
**Listener rules (2) Info**

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 <ul style="list-style-type: none"> <li>employee-tg-one: 1 (100%)</li> <li>Group-level stickiness: Off</li> </ul>	ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> <li>customer-tg-one: 1 (100%)</li> <li>Group-level stickiness: Off</li> </ul>	ARN	0 tags

## PHASE-7: Creating two Amazon ECS services

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



The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays a file tree for a project named 'MicroservicesIDE'. Inside the 'deployment' folder, there are several files: 'appspec-customer.yaml', 'appspec-employee.yaml', 'create-customer-microservice-tg-two.json' (which is currently selected), 'taskdef-customer.json', 'taskdef-employee.json', and 'README.md'. The main workspace is a terminal window titled 'bash - [ip-10-16-10-71.ec2.internal]'. The terminal output shows the command: 'voclabs:~/environment \$ aws ecs create-service --service-name customer-microservice --cli-input-json file://create-customer-microservice-tg-two.json'. The terminal has a dark theme with light-colored text. The status bar at the bottom indicates 'dev' and 'AWS profile default'.

```

1 {
  "taskDefinition": "customer-microservice:1",
  "cluster": "Microservices-serverlesscluster",
  "loadBalancers": [
    {
      "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-two/f2574315f65a0a42",
      "containerName": "customer",
      "containerPort": 8080
    }
  ],
  "desiredCount": 1,
  "launchType": "FARGATE",
  "schedulingStrategy": "REPLICA",
  "deploymentController": {
    "type": "CODE_DEPLOY"
  },
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "subnet-079b846921b332a35",
        "subnet-000a87838db6561c0"
      ],
      "securityGroups": [
        "sg-03bc4d5ba12c5aa5"
      ],
      "assignPublicIp": "ENABLED"
    }
  }
}

```

bash - "ip-10-16-10-71.ecx" 24.38 JSON Spaces: 4

```

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

```

dev CodeWhisperer AWS.profile.default

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

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

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

```

1 {
  "taskDefinition": "employee-microservice:1",
  "cluster": "Microservices-serverlesscluster",
  "loadBalancers": [
    {
      "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/employee-tg-two/851286b3cd13eaa",
      "containerName": "employee",
      "containerPort": 8080
    }
  ],
  "desiredCount": 1,
  "launchType": "FARGATE",
  "schedulingStrategy": "REPLICA",
  "deploymentController": {
    "type": "CODE_DEPLOY"
  },
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "subnet-079b846921b332a35",
        "subnet-000a87838db6561c0"
      ],
      "securityGroups": [
        "sg-03bc4d5ba12c5aa5"
      ],
      "assignPublicIp": "ENABLED"
    }
  }
}

```

dev CodeWhisperer AWS.profile.default

```

        "unit": "PERCENT"
    },
    "stabilityStatus": "STABILIZING",
    "stabilityStatusAt": "2024-04-27T20:39:29.127000+00:00",
    "tags": []
}
],
"roleArn": "arn:aws:iam::579794416250:role/aws-service-role/ecs.amazonaws.com/AMServiceRoleForECS",
"events": [],
"createdAt": "2024-04-27T20:39:29.127000+00:00",
"placementConstraints": [],
"placementStrategy": [],
"networkConfiguration": {
    "awsvpcConfiguration": {
        "subnets": [
            "subnet-000a87838db6561c0",
            "subnet-079b46921b332a35"
        ],
        "securityGroups": [
            "sg-03bc4d5baf12c5aa5"
        ],
        "assignPublicIp": "ENABLED"
    }
},
"healthCheckGracePeriodSeconds": 0,
"schedulingStrategy": "REPLICA",
"deploymentController": {
    "type": "CODE_DEPLOY"
},
"createdBy": "arn:aws:iam::579794416250:role/voclabs",
"enableECSManagedTags": false,
"propagateTags": "NONE",
"enableExecuteCommand": false
}
}
}
(EID)

```

**Amazon Elastic Container Service**

**Clusters**

Namespaces

Task definitions

Account settings

Install AWS Copilot

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

CloudShell Feedback

File Edit Find View Go Run Tools Window Support Preview Run Share

Search [Alt+S]

N. Virginia ▾ vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:579794416250:cluster/microservices-serverlesscluster	Active	Default	-
<b>Services</b>		<b>Tasks</b>	
Draining	Active	Pending	Running
-	2	1	-

Services Tasks Infrastructure Metrics Scheduled tasks Tags

**Services (2) Info**

C Manage tags Update Delete service Create

Filter launch type Filter service type

Service name	ARN	Status	Service...	Deployments and tasks	Last dep...
employee-microservice	arn:aws:ec...	Active	REPLICA	0/1 Tasks running	-
customer-microservice	arn:aws:ec...	Active	REPLICA	0/1 Tasks running	-

Filter services by value Any launch type Any service type

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

## PHASE-8: Configuring CodeDeploy and CodePipeline

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

The screenshot shows the AWS CodeDeploy console home page. The left sidebar navigation includes 'Developer Tools' and 'CodeDeploy' sections, with 'Getting started' currently selected. The main content area features the heading 'AWS CodeDeploy' and the sub-headline 'Automate code deployments to maintain application uptime'. A call-to-action button 'Create application' is visible. Below this, a description of AWS CodeDeploy is provided, mentioning it's a fully managed deployment service for various compute platforms. A 'Pricing (US)' section indicates it's free for CodeDeploy on AWS Lambda. The bottom of the page includes a 'How it works' section, copyright information, and links for Privacy, Terms, and Cookie preferences.

The screenshot shows the 'microservices' application details page. A green banner at the top states 'Application created' and instructs the user to create a deployment group. The main content area shows the 'Application details' section with the application name 'microservices' and compute platform 'Amazon ECS'. The 'Deployment groups' tab is selected, showing a table of deployment groups. A 'Create deployment group' button is located at the top right of the deployment groups table. The bottom of the page includes a 'CloudShell' link and standard AWS footer links for Privacy, Terms, and Cookie preferences.

The screenshot shows the AWS CodeDeploy console. On the left, a sidebar menu includes options like 'Source', 'Artifacts', 'Build', 'Deploy', 'Pipeline', 'Settings', and 'Pipelines'. The 'Deploy' section is expanded, showing 'CodeDeploy' under 'Deploy'. The main content area shows a 'Success' message: 'Deployment group created'. The path in the breadcrumb navigation is 'Developer Tools > CodeDeploy > Applications > microservices > microservices-customer'. The page title is 'microservices-customer'. It features two sections: 'Deployment group details' and 'Environment configuration'. In 'Deployment group details', fields include Deployment group name (microservices-customer), Application name (microservices), Compute platform (Amazon ECS), Deployment type (Blue/green), Service role ARN (arn:aws:iam::579794416250:role/DeployRole), and Rollback enabled (False). In 'Environment configuration', fields include ECS cluster name (microservices-serverlesscluster) and ECS service name (customer-microservice).

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

The screenshot shows the AWS CodePipeline console. The sidebar menu includes 'Source', 'Artifacts', 'Build', 'Deploy', 'Pipeline', 'Settings', and 'Pipelines'. The 'Pipeline' section is expanded, showing 'CodePipeline' under 'Pipeline'. The main content area shows a 'Pipelines' list with a search bar and a 'Create pipeline' button. A message at the bottom states 'No results' and 'There are no results to display.'

S | Services | Search | [Alt+S] | N. Virginia | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

## Add deploy stage Info

Step 4 of 5

**You cannot skip this stage**

Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.

### Deploy

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

Amazon ECS (Blue/Green)

Region  
US East (N. Virginia)

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

AWS CodeDeploy deployment group  
Choose the deployment group for this stage. This deployment group must be associated with the application you selected above in AWS CodeDeploy.

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

S | Services | Search | [Alt+S] | N. Virginia | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

## Review Info

Step 5 of 5

### Step 1: Choose pipeline settings

Pipeline settings

Pipeline name  
update-customer-microservice

Pipeline type  
V2

Execution mode  
QUEUED

Artifact location  
A new Amazon S3 bucket will be created as the default artifact store for your pipeline

Service role name  
arn:aws:iam::579794416250:role/PipelineRole

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

S | Services | Search | [Alt+S] | N. Virginia | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

## Step 2: Add source stage

Source action provider

Source action provider  
AWS CodeCommit

RepositoryName  
deployment

Default branch  
dev

PollForSourceChanges  
false

OutputArtifactFormat  
CODE\_ZIP

## Step 3: Add build stage

Build action provider

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

aws | Services | Q Search [Alt+S] | N. Virginia ▾ | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

### Deploy action provider

Deploy action provider  
Amazon ECS (Blue/Green)

ApplicationName  
microservices

DeploymentGroupName  
microservices-customer

TaskDefinitionTemplateArtifact  
SourceArtifact

TaskDefinitionTemplatePath  
taskdef-customer.json

AppSpecTemplateArtifact  
SourceArtifact

AppSpecTemplatePath  
appspec-customer.yaml

Configure automatic rollback on stage failure  
Disabled

Cancel Previous Create pipeline

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

aws | Services | Q Search [Alt+S] | N. Virginia ▾ | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**Success**  
Congratulations! The pipeline update-customer-microservice has been created.

Create a notification rule for this pipeline

Developer Tools > CodePipeline > Pipelines > update-customer-microservice

**update-customer-microservice**

Pipeline type: V2 Execution mode: QUEUED

**Source** In progress  
Pipeline execution ID: 17c9a08a-c7a6-4fd8-9d26-a91fe296ce6

Source  
AWS CodeCommit  
In progress - Just now  
View details

Disable transition

**Deploy** Didn't Run

Start rollback

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

aws | Services | Q Search [Alt+S] | N. Virginia ▾ | vclabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

### Edit action

Action name  
Choose a name for your action  
Image

Action provider  
Amazon ECR

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

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

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

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

No more than 100 characters

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

X
Create application

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

X

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 Remove

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

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

Success  
The most recent change will re-run through the pipeline. It might take a few moments for the status of the run to show in the pipeline view.

Developer Tools > CodePipeline > Pipelines

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
update-customer-microservice (Type: V2   Execution mode: QUEUED)	Failed	Source - 46a74065: Set image field to IMAGE1_NAME in task definition files	13 minutes ago	<span>View details</span>

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

## CodePipeline

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

Developer Tools > CodePipeline > Pipelines > update-customer-microservice

update-customer-microservice Pipeline type: V2 Execution mode: QUEUED

**Source** Succeeded Pipeline execution ID: 93256599-6635-48f1-8782-5f676ee19746

Source	Image
AWS CodeCommit	Amazon ECR
<b>Succeeded - Just now</b> 46a74065	<b>Succeeded - Just now</b> sha256:9a454319a98f
<a href="#">View details</a>	

46a74065 Source: Set image field to IMAGE1\_NAME in task definition files  
Image: sha256:9a454319a98f

[Disable transition](#)

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

<https://us-east-1.console.aws.amazon.com/console/home/?region=us-east-1>

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

## CodePipeline

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

Developer Tools > CodePipeline > Pipelines > update-customer-microservice

update-customer-microservice Pipeline type: V2 Execution mode: QUEUED

**Source** Succeeded - Just now 46a74065 **Image** Succeeded - Just now sha256:9a454319a98f [View details](#)

46a74065 Source: Set image field to IMAGE1\_NAME in task definition files  
Image: sha256:9a454319a98f

[Disable transition](#)

**Deploy** In progress Pipeline execution ID: 93256599-6635-48f1-8782-5f676ee19746

Deploy
Amazon ECS (Blue/Green)
<b>In progress - Just now</b>
<a href="#">View details</a>

46a74065 Source: Set image field to IMAGE1\_NAME in task definition files  
Image: sha256:9a454319a98f

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

[CloudShell](#) [Feedback](#)

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

## CodeDeploy

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
  - Getting started
  - Deployments
  - Deployment**
  - Applications
  - Deployment configurations
  - On-premises instances
- Pipeline • CodePipeline
- Settings
- Go to resource
- Feedback

Developer Tools > CodeDeploy > Deployments > Task set activity

### Task set activity

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/3348309877829856275	Original	PRIMARY	100%	1	0	0
ecs-svc/5360225217073829580	Replacement	ACTIVE	0	1	1	0

Deployment lifecycle events

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	<b>Succeeded</b>	Apr 27, 2024 5:31 PM (UTC-4:00)	Apr 27, 2024 5:31 PM (UTC-4:00)
Install	-	<b>In progress</b>	Apr 27, 2024 5:31 PM (UTC-4:00)	-
AfterInstall	-	<b>Pending</b>	-	-
AllowTestTraffic	-	<b>Pending</b>	-	-
AfterAllowTestTraffic	-	<b>Pending</b>	-	-
BeforeAllowTraffic	-	<b>Pending</b>	-	-
AllowTraffic	-	<b>Pending</b>	-	-
AfterAllowTraffic	-	<b>Pending</b>	-	-

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

[CloudShell](#) [Feedback](#)

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**CodePipeline**

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

Succeeded - 2 days ago 46a74065 View details Succeeded - 2 days ago sha256:9a454319a98f View details

46a74065 Source: Set image field to IMAGE1\_NAME in task definition files Image: sha256:9a454319a98f

Disable transition

**Deploy** Succeeded Pipeline execution ID: 93256599-6635-48f1-8782-5f676ee19746 Start rollback

Deploy Amazon ECS (Blue/Green) Succeeded - 2 days ago View details

46a74065 Source: Set image field to IMAGE1\_NAME in task definition files Image: sha256:9a454319a98f

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**CodeDeploy**

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
  - Getting started
  - Deployments
  - Deployment**
  - Applications
  - Deployment configurations
  - On-premises instances
- Pipeline • CodePipeline
- Settings
- Go to resource
- Feedback

**Task set activity**

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/5360225217073829580	Replacement	PRIMARY	100%	1	1	0
ecs-svc/3348309877829856275	Original	ACTIVE	0	1	0	0

**Deployment lifecycle events**

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 27, 2024 5:31 PM (UTC-4:00)	Apr 27, 2024 5:31 PM (UTC-4:00)
Install	2 minutes 2 seconds	Succeeded	Apr 27, 2024 5:31 PM (UTC-4:00)	Apr 27, 2024 5:33 PM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 27, 2024 5:33 PM (UTC-4:00)	Apr 27, 2024 5:33 PM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 27, 2024 5:33 PM (UTC-4:00)	Apr 27, 2024 5:33 PM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 27, 2024 5:33 PM (UTC-4:00)	Apr 27, 2024 5:33 PM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 27, 2024 5:33 PM (UTC-4:00)	Apr 27, 2024 5:33 PM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 27, 2024 5:33 PM (UTC-4:00)	Apr 27, 2024 5:33 PM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 27, 2024 5:33 PM (UTC-4:00)	Apr 27, 2024 5:33 PM (UTC-4:00)

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

Lab Instructions: Building Microservices | update-customer-microservice | Load balancer details | EC2 | Coffee suppliers | Create Temp Directory AWS CLI | +

Not secure microserviceseslb-10312331.us-east-1.elb.amazonaws.com

Youtube Gmail LinkedIn Indeed MyStevens ChatGPT Gemini Bank of America LeetCode - The World's Largest Coding Platform Careers Course Modules: A... Homeworkify.st

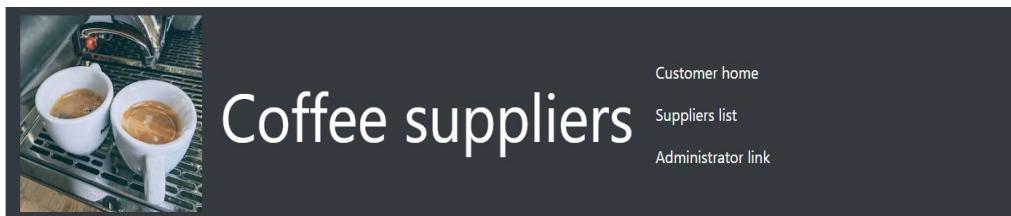
**Coffee suppliers**

Welcome

Customer home  
Suppliers list  
Administrator link

Use this app to keep track of your coffee suppliers

List of suppliers



## All suppliers

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

Tell us what you think X

**Amazon Elastic Container Service**

- Clusters**
- Namespaces
- Task definitions
- Account settings

Install AWS Copilot ▾

Amazon ECR ▾

Repositories

AWS Batch ▾

Documentation ▾

Discover products ▾

CloudShell Feedback

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

ARN arn:aws:ecs:us-east-1:579794416250:cluster/microservices-serverlesscluster Status Active CloudWatch monitoring Default Registered container instances -

Services Draining Active Tasks Pending - Running 1

Services (2) Info Manage tags Update Delete service Create

Filter launch type Any launch type Any service type

Service name	ARN	Status	Service...	Deployments and tasks	Last dep...
employee-microservice	arn:aws:ec...	Active	REPLICA	0/1 Tasks running	-
customer-microservice	arn:aws:ec...	Active	REPLICA	1/1 Tasks running	-

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

Tell us what you think X

**Amazon Elastic Container Service**

- Clusters**
- Namespaces
- Task definitions
- Account settings

Install AWS Copilot ▾

Amazon ECR ▾

Repositories

AWS Batch ▾

Documentation ▾

Discover products ▾

https://us-east-1.console.aws.amazon.com/ecs/v2/clusters/microservices-serverlesscluster/tasks...

Cluster overview

ARN arn:aws:ecs:us-east-1:579794416250:cluster/microservices-serverlesscluster Status Active CloudWatch monitoring Default Registered container instances -

Services Draining Active Tasks Pending - Running 1

Tasks (1) Manage tags Stop Run new task

Filter desired status Running Filter launch type Any launch type

Task	Last status	Desired st...	Tas...	Health sta...	Started at	Container instan...	Launch type
a2e29...	Running	Running	custo...	Unknown	5 minutes ago	-	FARGATE

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

Tell us what you think X

## Amazon Elastic Container Service

**Clusters**

- Namespaces
- Task definitions
- Account settings

Install AWS Copilot ▾

Amazon ECR ▾

- Repositories

AWS Batch ▾

Documentation ▾

Discover products ▾

CloudShell Feedback

ARN arm:aws:ecs:us-east-1:579794416250:task/microservices-serverlesscluster/a2e2981aac2b446f8f888e83be8dfe43 Last status Running Desired status Running Started/Created at 2024-04-27T21:32:17.773Z 2024-04-27T21:31:44.110Z

### Task overview

### Container details for customer

Details Log configuration Network bindings Docker labels and hosts Environment variables and files Volume configuration

#### Details

Image URI	579794416250.dkr.ecr.us-east-1.amazonaws.co	Essential Yes	Command
-----------	---	---------------	---------

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

AMIs AMI Catalog

Elastic Block Store Volumes Snapshots Lifecycle Manager

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

Load Balancing Load Balancers Target Groups Trust Stores New

Auto Scaling Auto Scaling Groups

CloudShell Feedback

EC2 Target groups

### Target groups (4) Info

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balancer
customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-one/579794416250	8080	HTTP	IP	microservicesLB
customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-two/579794416250	8080	HTTP	IP	None associated
employee-tg-one	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/employee-tg-one/579794416250	8080	HTTP	IP	microservicesLB
employee-tg-two	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/employee-tg-two/579794416250	8080	HTTP	IP	microservicesLB

0 target groups selected

Select a target group above.

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

AMIs AMI Catalog

Elastic Block Store Volumes Snapshots Lifecycle Manager

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

Load Balancing Load Balancers Target Groups Trust Stores New

Auto Scaling Auto Scaling Groups

CloudShell Feedback

EC2 Load balancers

Introducing resource map for Network Load Balancers Resource map is a visual representation of the relationships between load balancer resources and provides the ability to view, explore, and troubleshoot the architecture of your load balancer. Resource map can be viewed on the load balancers detail page. Share feedback to help us improve your experience. Give feedback X

### Load balancers (1/1)

Load balancer: microservicesLB

Forward to target group • customer-tg-one: 1 (100%) 2 rules ARN Not applicable Not applicable

Forward to target group • customer-tg-one: 1 (100%) 2 rules ARN Not applicable Not applicable

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

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

Screenshot of the AWS CodePipeline console showing a newly created pipeline named "update-employee-microservice".

**Success**  
Congratulations! The pipeline update-employee-microservice has been created.

Developer Tools > CodePipeline > Pipelines > update-employee-microservice

**update-employee-microservice**

Pipeline type: V2 Execution mode: QUEUED

**Source** In progress

Source: AWS CodeCommit  
Status: Didn't Run  
No executions yet

**Deploy** Didn't Run

Start rollback

Disable transition

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

Screenshot of the AWS CodePipeline console showing the pipeline "update-employee-microservice" saved successfully.

**Success**  
Pipeline was saved successfully.

Developer Tools > CodePipeline > Pipelines > update-employee-microservice

**update-employee-microservice**

Pipeline type: V2 Execution mode: QUEUED

**Source** Succeeded

Pipeline execution ID: [aeab34f5-8a16-4b2a-99e6-9991e83fd1b1](#)

Source: AWS CodeCommit  
Image: Amazon ECR  
Status: Succeeded - 1 minute ago  
Execution ID: [46a74065](#)  
View details

46a74065 Source: Set image field to IMAGE1\_NAME in task definition files

Disable transition

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

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

Screenshot of the AWS CodePipeline console showing the Pipelines list. A success message at the top indicates a recent change will re-run through the pipeline.

**Pipelines Info**

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
update-employee-microservice	Failed	Source - 46a74065: Set image field to IMAGE1_NAME in task definition files	2 minutes ago	<a href="#">View details</a>
update-customer-microservice	Succeeded	Source - 46a74065: Set image field to IMAGE1_NAME in task definition files	17 minutes ago	<a href="#">View details</a>

Screenshot of the AWS CodePipeline console showing the details for the 'update-employee-microservice' pipeline.

**update-employee-microservice**

Pipeline type: V2 Execution mode: QUEUED

**Source** Succeeded

Pipeline execution ID: [61e475d6-9915-4c91-bad8-1bae9dc13fc5](#)

Source	Image
AWS CodeCommit	Amazon ECR
<a href="#">Succeeded - 1 minute ago</a> 46a74065	<a href="#">Succeeded - 1 minute ago</a> sha256:417ce7e7f2cc

[View details](#)

[46a74065](#) Source: Set image field to IMAGE1\_NAME in task definition files  
Image: sha256:417ce7e7f2cc

[Disable transition](#)

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**CodeDeploy**

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

Q Go to resource F Feedback

### Task set activity

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/8170170770624423825	Original	PRIMARY	100%	1	0	0
ecs-svc/8647511775004021825	Replacement	ACTIVE	0	1	1	0

### Deployment lifecycle events

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 27, 2024 6:19 PM (UTC-4:00)	Apr 27, 2024 6:19 PM (UTC-4:00)
Install	-	In progress	Apr 27, 2024 6:19 PM (UTC-4:00)	-
AfterInstall	-	Pending	-	-
AllowTestTraffic	-	Pending	-	-
AfterAllowTestTraffic	-	Pending	-	-
BeforeAllowTraffic	-	Pending	-	-
AllowTraffic	-	Pending	-	-
AfterAllowTraffic	-	Pending	-	-

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**CodePipeline**

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

Q Go to resource F Feedback

**Succeeded - 55 minutes ago**  
46a74065 View details

**Succeeded - 55 minutes ago**  
sha256:8b4a39eac54a View details

Image: sha256:8b4a39eac54a  
[46a74065](#) Source: Set image field to IMAGE1\_NAME in task definition files

**Disable transition**

**Deploy** Succeeded  
Pipeline execution ID: [f6eedd8ff-5b9f-430c-9f5d-19b98b3b4335](#) Start rollback

**Deploy**  
Amazon ECS (Blue/Green) View details

**Succeeded - 48 minutes ago**  
46a74065 View details

Image: sha256:8b4a39eac54a  
[46a74065](#) Source: Set image field to IMAGE1\_NAME in task definition files

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

AWS Services Search [Alt+S] N. Virginia v vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ▾

**CodeDeploy**

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

Q Go to resource F Feedback

### Task set activity

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/8170170770624423825	Original	ACTIVE	0	1	0	0
ecs-svc/8647511775004021825	Replacement	PRIMARY	100%	1	1	0

### Deployment lifecycle events

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 27, 2024 6:19 PM (UTC-4:00)	Apr 27, 2024 6:19 PM (UTC-4:00)
Install	2 minutes 2 seconds	Succeeded	Apr 27, 2024 6:19 PM (UTC-4:00)	Apr 27, 2024 6:21 PM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 27, 2024 6:21 PM (UTC-4:00)	Apr 27, 2024 6:21 PM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 27, 2024 6:21 PM (UTC-4:00)	Apr 27, 2024 6:21 PM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 27, 2024 6:21 PM (UTC-4:00)	Apr 27, 2024 6:21 PM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 27, 2024 6:21 PM (UTC-4:00)	Apr 27, 2024 6:21 PM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 27, 2024 6:21 PM (UTC-4:00)	Apr 27, 2024 6:21 PM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 27, 2024 6:21 PM (UTC-4:00)	Apr 27, 2024 6:21 PM (UTC-4:00)

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



# Manage coffee suppliers

Administrator  
home  
  
Suppliers list  
  
Customer home

## All suppliers

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

[Add a new supplier](#)



# Coffee suppliers

Customer home  
Suppliers list  
Administrator link

## All suppliers

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

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

**EC2 > Target groups**

### Target groups (4) Info

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balancer	VPC
<a href="#">customer-tg-one</a>	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-one/5beeb23e6d4c3b12	8080	HTTP	IP	microservicesLB	vpc-00000000
<a href="#">customer-tg-two</a>	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-two/5beeb23e6d4c3b12	8080	HTTP	IP	None associated	vpc-00000000
<a href="#">employee-tg-one</a>	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/employee-tg-one/5beeb23e6d4c3b12	8080	HTTP	IP	microservicesLB	vpc-00000000
<a href="#">employee-tg-two</a>	arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/employee-tg-two/5beeb23e6d4c3b12	8080	HTTP	IP	None associated	vpc-00000000

**0 target groups selected**

Select a target group above.

**CloudShell Feedback**

**N. Virginia** vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

**Listeners and rules**

Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:579794416250:loadbalancer/app/microservicesLB/5beeb23e6d4c3b12

DNS name: microservicesLB-103312331.us-east-1.elb.amazonaws.com (A Record)

**Listeners and rules (2) Info**

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

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

**CloudShell Feedback**

**N. Virginia** vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

**Developer Tools**

**CodePipeline**

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
  - Getting started
  - Pipelines**
- Settings
- Go to resource
- Feedback

**Pipelines Info**

Create pipeline

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
update-employee-microservice (Type: V2   Execution mode: QUEUED)	Succeeded	Source – 46a74065: Set image field to IMAGE1_NAME in task definition files Image – sha256:4:	1 hour ago	<span style="color: green;">✓</span> <span style="color: red;">✗</span> <span style="color: red;">✗</span> <span style="color: red;">✗</span> View details
update-customer-microservice (Type: V2   Execution mode: QUEUED)	Succeeded	Source – 46a74065: Set image field to IMAGE1_NAME in task definition files Image – sha256:9:	1 hour ago	<span style="color: green;">✓</span> <span style="color: red;">✗</span> <span style="color: red;">✗</span> <span style="color: red;">✗</span> View details

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

- Task 9.1: To limit access to the employee microservice to a specific IP address I completed the following steps:
  - In the Amazon EC2 console, I verified that all four target groups were still associated with the Application Load Balancer. If necessary, I reassigned any missing target groups.
  - I used an online resource (like whatismyip.com) to find my public IPv4 address.
  - I added a condition to the rule with "IF Path is /admin/\*" to limit access to a specific IP address. This condition allowed routing to employee target groups only if the source IP matched my public IPv4 address with a /32 subnet mask.
  - Similarly, I added the same source IP condition to the corresponding rule ensuring that the employee microservice could only be accessed from my specific IP address.
  - By implementing these changes, I restricted access to the employee microservice to users connecting from a designated IP address providing an additional layer of security and preventing unauthorized users from modifying or deleting supplier entries.

The screenshot shows the AWS EC2 Target Groups page. The left sidebar lists services like AMIs, Elastic Block Store, Network & Security, Load Balancing, Auto Scaling, and Target Groups (which is currently selected). The main content area displays a table of target groups:

Name	ARN	Port	Protocol	Target type	Load balancer
customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/customer-tg-one/5555555555555555	8080	HTTP	IP	microservicesLB
customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/customer-tg-two/5555555555555555	8080	HTTP	IP	None associated
employee-tg-one	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/employee-tg-one/5555555555555555	8080	HTTP	IP	microservicesLB
employee-tg-two	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/employee-tg-two/5555555555555555	8080	HTTP	IP	None associated

Below the table, a message says "4 target groups selected". Underneath that, there are four buttons labeled "customer-tg-one X", "customer-tg-two X", "employee-tg-one X", and "employee-tg-two X". The "Monitoring" tab is currently selected at the bottom of the page.

WhatIsMyIP.com  Pricing API Sign Up Login Help

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

## What Is My IP?

**My Public IPv4:** [100.8.206.166](#) ⓘ  
**My Public IPv6:** Not Detected  
**My IP Location:** Newark, NJ US ⓘ  
**My ISP:** Verizon Business ⓘ

What is an IP address? ⓘ

aws Services Search [Alt+S] N. Virginia vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ⓘ

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

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

Images AMIs AMI Catalog

Elastic Block Store Volumes

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

HTTP:80 microservicesLB Listener ARN arn:aws:elasticloadbalancing:us-east-1:579794416250:listener/app/microservicesLB/5beeb23e6d4c3b12/eea4f00c190bbc4

Rules Tags

**Listener rules (2) Info** Rule limits Actions Add rule

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

Filter rules	ARN
<input type="checkbox"/>	Name tag Priority Conditions (If) Actions (Then)
<input type="checkbox"/>	- 1 • Path Pattern is /admin/*, AND • Source IP is 100.8.206.166/32 Forward to target group • employee-tg-one: 1 (100%) ARN • Group-level stickiness: Off
<input type="checkbox"/>	Default Last (default) If no other rule applies Forward to target group • customer-tg-one: 100 (100%) ARN • Group-level stickiness: Off

aws Services Search [Alt+S] N. Virginia vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250 ⓘ

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

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

Images AMIs AMI Catalog

Elastic Block Store Volumes

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

HTTP:8080 microservicesLB Listener ARN arn:aws:elasticloadbalancing:us-east-1:579794416250:listener/app/microservicesLB/5beeb23e6d4c3b12/15216122989b5776

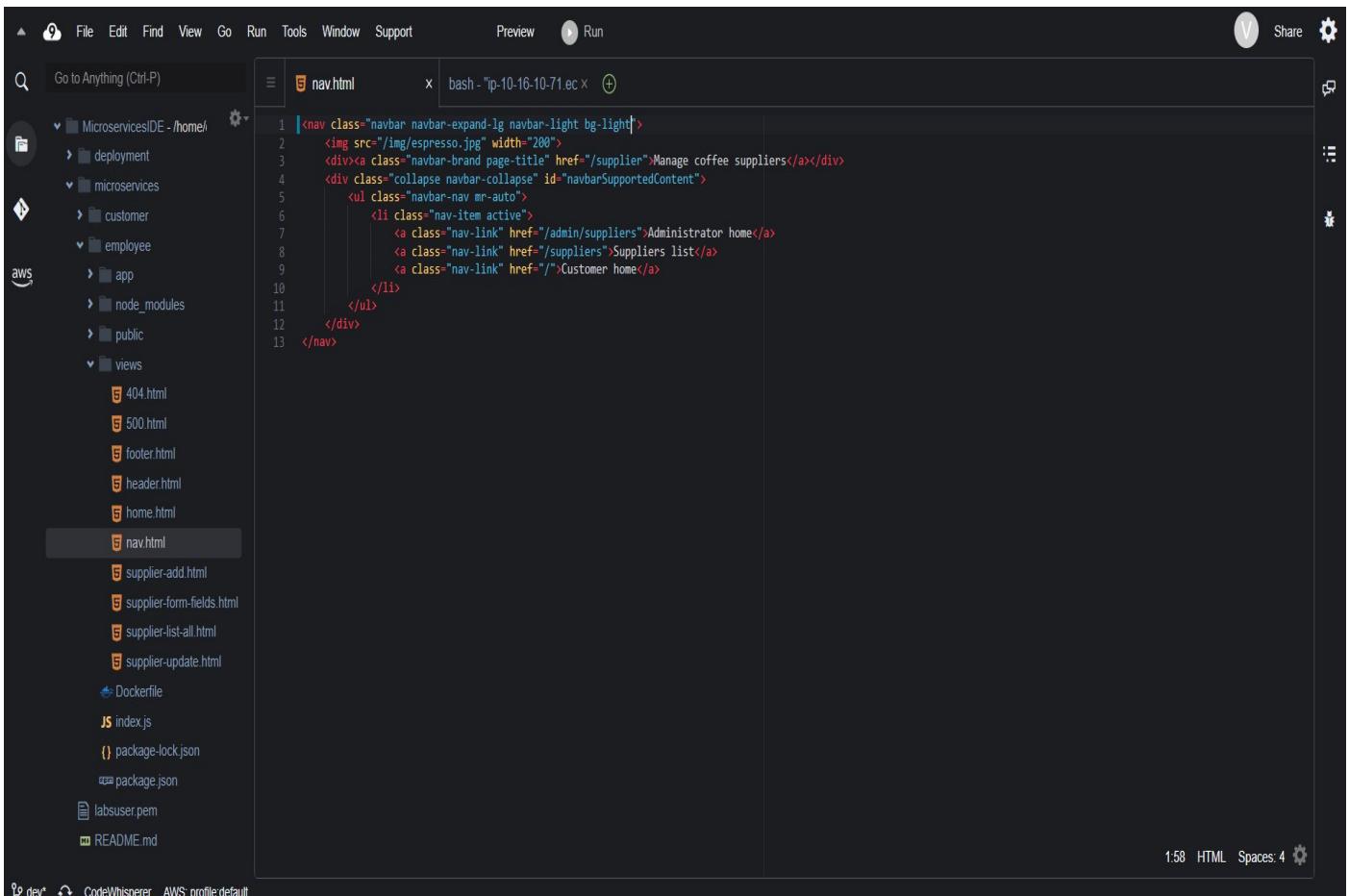
Rules Tags

**Listener rules (2) Info** Rule limits Actions Add rule

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

Filter rules	ARN
<input type="checkbox"/>	Name tag Priority Conditions (If) Actions (Then)
<input type="checkbox"/>	- 1 • Path Pattern is /admin/*, AND • Source IP is 100.8.206.166/32 Forward to target group • employee-tg-one: 1 (100%) ARN • Group-level stickiness: Off
<input type="checkbox"/>	Default Last (default) If no other rule applies Forward to target group • customer-tg-one: 100 (100%) ARN • Group-level stickiness: Off

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



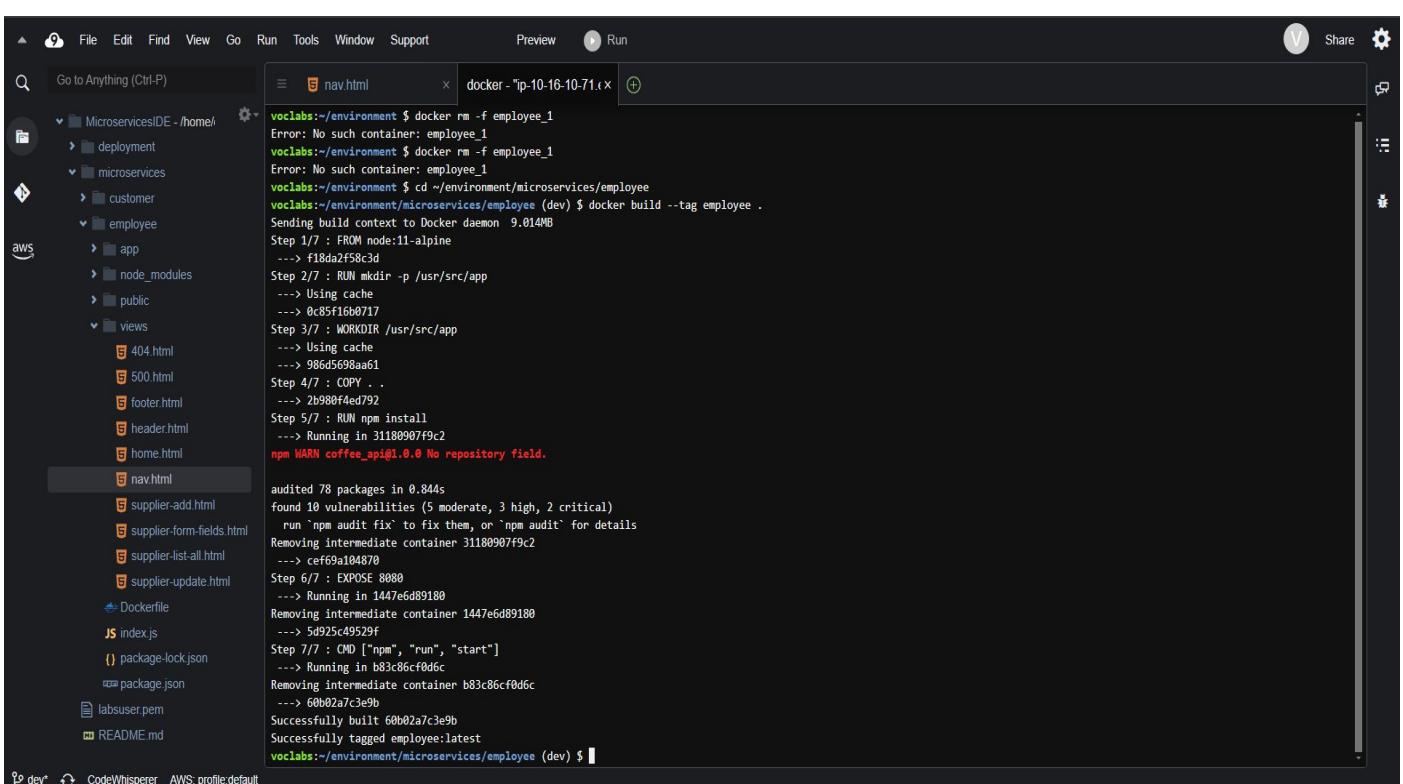
File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE - /home/labsuser

- deployment
- microservices
  - customer
  - employee
    - app
    - node\_modules
    - public
  - views
    - 404.html
    - 500.html
    - footer.html
    - header.html
    - home.html
    - nav.html**
    - supplier-add.html
    - supplier-form-fields.html
    - supplier-list-all.html
    - supplier-update.html
- Dockerfile
- index.js
- package-lock.json
- package.json
- labsuser.pem
- README.md

158 HTML Spaces: 4



File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE - /home/labsuser

- deployment
- microservices
  - customer
  - employee
    - app
    - node\_modules
    - public
  - views
    - 404.html
    - 500.html
    - footer.html
    - header.html
    - home.html
    - nav.html**
    - supplier-add.html
    - supplier-form-fields.html
    - supplier-list-all.html
    - supplier-update.html
- Dockerfile
- index.js
- package-lock.json
- package.json
- labsuser.pem
- README.md

```

voclabs:~/environment $ docker rm -f employee_1
Error: No such container: employee_1
voclabs:~/environment $ docker rm -f employee_1
Error: No such container: employee_1
voclabs:~/environment $ cd ~/environment/microservices/employee
voclabs:~/environment/microservices/employee (dev) $ docker build --tag employee .
Sending build context to Docker daemon 9.014MB
Step 1/7 : FROM node:11-alpine
--> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
--> Using cache
--> 0c85f16b0717
Step 3/7 : WORKDIR /usr/src/app
--> Using cache
--> 986d509aa61
Step 4/7 : COPY .
--> 2b980f4ed792
Step 5/7 : RUN npm install
--> Running in 31180907f9c2
npm WARN coffee_api@1.0.0 No repository field.

audited 78 packages in 0.844s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 31180907f9c2
--> cef69a104870
Step 6/7 : EXPOSE 8080
--> Running in 1447e6d89180
Removing intermediate container 1447e6d89180
--> 5d925c49529f
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in b83c86cf0d6c
Removing intermediate container b83c86cf0d6c
--> 60b02a7c3e9b
Successfully built 60b02a7c3e9b
Successfully tagged employee:latest
voclabs:~/environment/microservices/employee (dev) $

```

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

MicroservicesIDE - /home/labsuser

- deployment
- microservices
  - customer
  - employee
    - app
    - node\_modules
    - public
  - views
    - 404.html
    - 500.html
    - footer.html
    - header.html
    - home.html
    - nav.html**
    - supplier-add.html
    - supplier-form-fields.html
    - supplier-list-all.html
    - supplier-update.html
- Dockerfile
- index.js
- package-lock.json
- package.json
- labsuser.pem
- README.md

```

Step 1/7 : FROM node:11-alpine
--> f18da2f58c3d
Step 2/7 : RUN mkdir -p /usr/src/app
--> Using cache
--> 0e85f16b0717
Step 3/7 : WORKDIR /usr/src/app
--> Using cache
--> 986d5d98aa61
Step 4/7 : COPY . .
--> 2b980f4ed792
Step 5/7 : RUN npm install
--> Running in 31180907f9c2
npm WARN coffee_api@1.0.0 No repository field.

audited 78 packages in 0.844s
found 10 vulnerabilities (5 moderate, 3 high, 2 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 31180907f9c2
--> ceef69a1b4870
Step 6/7 : EXPOSE 8088
--> Running in 1447e6d89180
Removing intermediate container 1447e6d89180
--> 5d925c49529f
Step 7/7 : CMD ["npm", "run", "start"]
--> Running in b83c86cf0d6c
Removing intermediate container b83c86cf0d6c
--> 60b02a73e9b
Successfully built 60b02a73e9b
Successfully tagged employee:latest
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.cyclnx0fhcy.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
579794416250
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 [579794416250.dkr.ecr.us-east-1.amazonaws.com/employee]
dead8521269c: Pushed
55d2c5088432: Pushed
c82aa84eb666: Layer already exists
d81d7151330b7: Layer already exists
1dc7f3bb09a4: Layer already exists
dcaceb729824: Layer already exists
f1b5933fe4b5: Layer already exists
latest: digest: sha256:8d4a39eac54a5abd4ccc0974a5198ea3d71a5ef8259616f2d300a294c2876c01 size: 1783
voclabs:~/environment/microservices/employee (dev) $ 

```

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

AWS Services Search [Alt+S] N. Virginia vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

**CodePipeline**

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

Go to resource Feedback CloudShell

update-employee-microservice Pipeline type: V2 Execution mode: QUEUED

**Source** Succeeded Pipeline execution ID: 1b87ede6-239a-41fb-a690-dbf624657377

Source	Image
<a href="#">AWS CodeCommit</a>	<a href="#">Amazon ECR</a>
Succeeded - 2 minutes ago <a href="#">46a74065</a>	Succeeded - 2 minutes ago sha256:8b4a39eac54a
<a href="#">View details</a>	<a href="#">View details</a>

Image: sha256:8b4a39eac54a  
[46a74065](#) Source: Set image field to IMAGE1\_NAME in task definition files

Disable transition

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

AWS Services Search [Alt+S] N. Virginia vocabs/user3233567=Hassan\_Ravindra,\_Supreeth @ 5797-9441-6250

**CodeDeploy**

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

Go to resource Feedback CloudShell

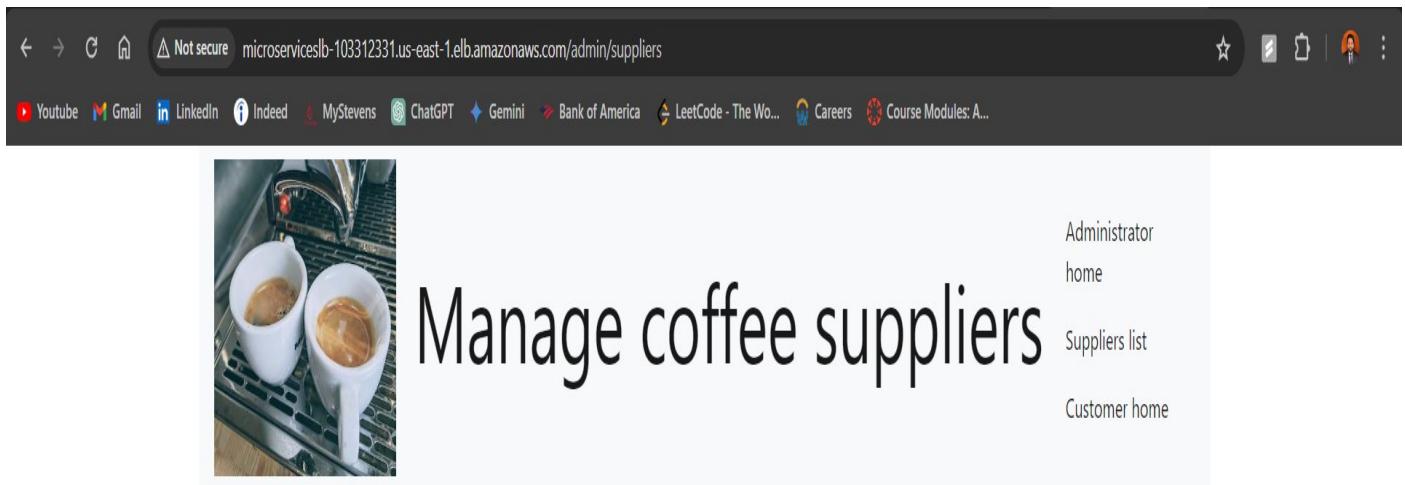
Deployment history

Deployment Id	Status	Deploy...	Compute ...	Application	Deploym...	Revision L...	Initiating ...	Start
d-DVBRRFCT5	In progress	Blue/green	Amazon ECS	microservi...	microservi...	179077bf...	User action	Apr 2
d-ORSBZ1CT5	Succeeded	Blue/green	Amazon ECS	microservi...	microservi...	f42cbcc0...	User action	Apr 2
d-F9BJGLAT5	Succeeded	Blue/green	Amazon ECS	microservi...	microservi...	ddb0fc79...	User action	Apr 2

View details Actions ▾ Copy deployment Retry deployment

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

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



## All suppliers

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

[Add a new supplier](#)

```
bash: alb-endpoint: No such file or directory
vocabs:~/environment/microservices/employee (dev) $ curl http://microservicesLB-10331231.us-east-1.elb.amazonaws.com/admin/suppliers
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <link rel="stylesheet" href="/css/base.css">
    <title>Coffee suppliers</title>
</head>
<body>

<div class="container">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        
        <div><a class="navbar-brand page-title" href="/supplier">Coffee suppliers</a></div>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="/">Customer home</a>
                    <a class="nav-link" href="/suppliers">Suppliers list</a>
                    <a class="nav-link" href="/admin/suppliers">Administrator link</a>
                </li>
            </ul>
        </div>
    </nav>
    <div class="jumbotron">
        <h1>404</h1>
        <p>Sorry, we don't seem to have that page in stock</p>
        <hr class="my-4">
        <p>
            <a class="btn btn-primary btn-lg" href="/" role="button">Return to home page</a>
        </p>
    </div>
</div>
<script src="/js/jquery-3.6.0.min.js"></script>
<script src="/js/bootstrap.min.js"></script>
</body>
```

7:53

5G 60%



l.elb.amazonaws.com:8080



This site can't provide a secure connection

**microserviceslb-103312331.us-east-1.elb.amazonaws.com** sent an invalid response.

ERR\_SSL\_PROTOCOL\_ERROR

Reload

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

```

{
  "service": {
    "serviceArn": "arn:aws:ecs:us-east-1:579794416250:service/microservices-serverlesscluster/customer-microservice",
    "serviceName": "customer-microservice",
    "clusterArn": "arn:aws:ecs:us-east-1:579794416250:cluster/microservices-serverlesscluster",
    "loadBalancers": [
      {
        "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:579794416250:targetgroup/customer-tg-one/8aea0d658b1c92f7",
        "containerName": "customer",
        "containerPort": 8080
      }
    ],
    "serviceRegistries": [],
    "status": "ACTIVE",
    "desiredCount": 3,
    "runningCount": 1,
    "pendingCount": 0,
    "launchType": "FARGATE",
    "platformVersion": "1.4.0",
    "platformFamily": "Linux",
    "deploymentDefinition": "arn:aws:ecs:us-east-1:579794416250:task-definition/customer-microservice:3",
    "deploymentConfiguration": {
      "maximumPercent": 200,
      "minimumHealthyPercent": 100
    },
    "taskSets": [
      {
        "id": "ecs-svc/536022517073829580",
        "taskSetArn": "arn:aws:ecs:us-east-1:579794416250:task-set/microservices-serverlesscluster/customer-microservice/ecs-svc/536022517073829580",
        "serviceArn": "arn:aws:ecs:us-east-1:579794416250:service/microservices-serverlesscluster/customer-microservice",
        "clusterArn": "arn:aws:ecs:us-east-1:579794416250:cluster/microservices-serverlesscluster",
        "startedBy": "CodeDeploy",
        "externalId": "d-F9B1GLAT5",
        "status": "PRIMARY",
        "taskDefinition": "arn:aws:ecs:us-east-1:579794416250:task-definition/customer-microservice:3"
      }
    ]
  }
}

```

Service name	ARN	Status	Deployments and tasks	Last dep
<a href="#">employee-microservice</a>	arn:aws:ec...	Active	REPLICA	<div style="width: 100%;">1/1 Tasks running</div>
<a href="#">customer-microservice</a>	arn:aws:ec...	Active	REPLICA	<div style="width: 33%; background-color: #0070C0; color: white;">1/3 Tasks running</div>

## **PART-2: ANALYSIS**

### **PHASE-2: Analyzing the infrastructure of the monolithic application**

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

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

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

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

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

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

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

## **Phase 6: Creating target groups and an Application Load Balancer**

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

## **Phase 7: Creating two Amazon ECS services**

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

## Phase 8: Configuring CodeDeploy and CodePipeline

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

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

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