

PROJECT- Building Microservices and a CI/CD Pipeline with AWS

Submitted by- Nancy Gupta

Part 1 - Project Execution

HOME SCREEN OF PROJECT

The screenshot shows the AWS Academy interface. On the left is a sidebar with icons for Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main area shows the navigation path: ALPMCPB... > Modules > AWS Acad... > Lab Instructions: Building Microservices and a CI/CD Pipeline with AWS. The title of the page is "Lab Instructions: Building Microservices and a CI/CD Pipeline with AWS". The main content area displays a diagram titled "EN-US" showing three entities: "Customers (Café franchise location managers)", "Coffee suppliers application", and "Employees". Arrows indicate interactions between these entities. Below the diagram, there is text explaining the current state of the application and the goal of splitting it into microservices. To the right, a "Cloud Access" panel is open, showing session details: Remaining session time: 11:56:20(717 minutes), Session started at: 2024-04-26T11:45:59-0700, Session to end at: 2024-04-26T23:45:59-0700. It also shows accumulated lab time: 00:03:00 (3 minutes) and no running instances. Buttons for SSH key, Cloud SSO, and download PEM and PPK files are available.

Scenario

The owners of a café corporation with many franchise locations have noticed how popular their gourmet coffee offerings have become.

Customers (the café franchise location managers) cannot seem to get enough of the high-quality coffee beans that are needed to create amazing cappuccinos and lattes in their cafés.

Meanwhile, the *employees* in the café corporate office have been challenged to consistently source the highest-quality coffee beans. Recently, the leaders at the corporate office learned that one of their favorite coffee suppliers wants to sell her company. The café corporate managers jumped at the opportunity to buy the company. The acquired coffee supplier runs a coffee supplier listings application on an AWS account, as shown in the following image.

The coffee suppliers application currently runs as a *monolithic* application. It has reliability and performance issues. That is one of the reasons that you have recently been hired to work in the café corporate office. In this project, you perform tasks that are associated with software development engineer (SDE), app developer, and cloud support engineer roles.

You have been tasked to split the monolithic application into *microservices*, so that you can scale the services independently and allocate more compute resources to the services that experience the highest demand, with the goal of avoiding bottlenecks. A microservices design will also help avoid single points of failure, which could bring down the entire application in a monolithic design. With

services isolated from one another, if one microservice becomes temporarily unavailable, the other microservices might remain available.

You have also been challenged to develop a CI/CD pipeline to automatically deploy updates to the production cluster that runs containers, using a blue/green deployment strategy.

Solution requirements:

The solution must meet the following requirements:

- R1 - Design: The solution must have an architecture diagram.
- R2 - Cost optimized: The solution must include a cost estimate.
- R3 - Microservices-based architecture: Ensure that the solution is functional and deploys a microservice-based architecture.
- R4 - Portability: The solution must be portable so that the application code isn't tied to running on a specific host machine.
- R5 - Scalability/resilience: The solution must provide the ability to increase the amount of compute resources that are dedicated to serving requests as usage patterns change, and the solution must use routing logic that is reliable and scalable.
- R6 - Automated CI/CD: The solution must provide a CI/CD pipeline that can be automatically invoked when code is updated and pushed to a version-controlled code repository.

R1, R2: Phase 1: Planning the design and estimating cost

In this phase, you will plan the design of your architecture. First, you will create an architecture diagram.

You will also estimate the cost of the proposed solution and present the estimate to your educator. An important first step for any solution is to plan the design and estimate the cost. Review the various components in the architecture to adjust the estimated cost. Cost, along with the features and limitations of specific AWS services, is an important factor when you build a solution.

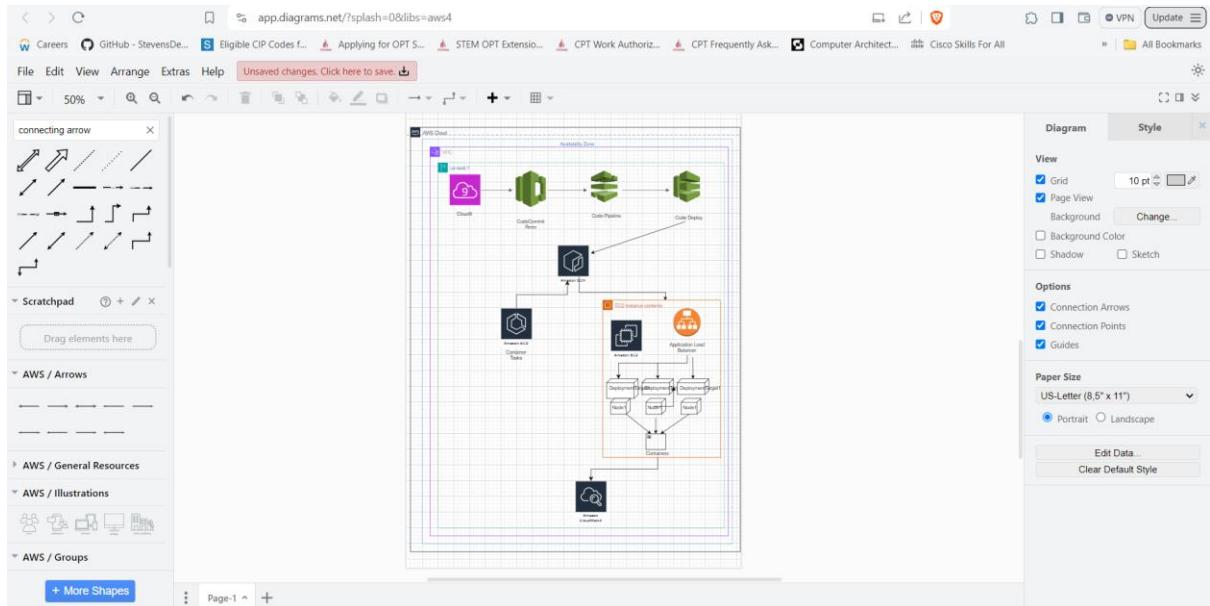
Task 1.1: Create an architecture diagram

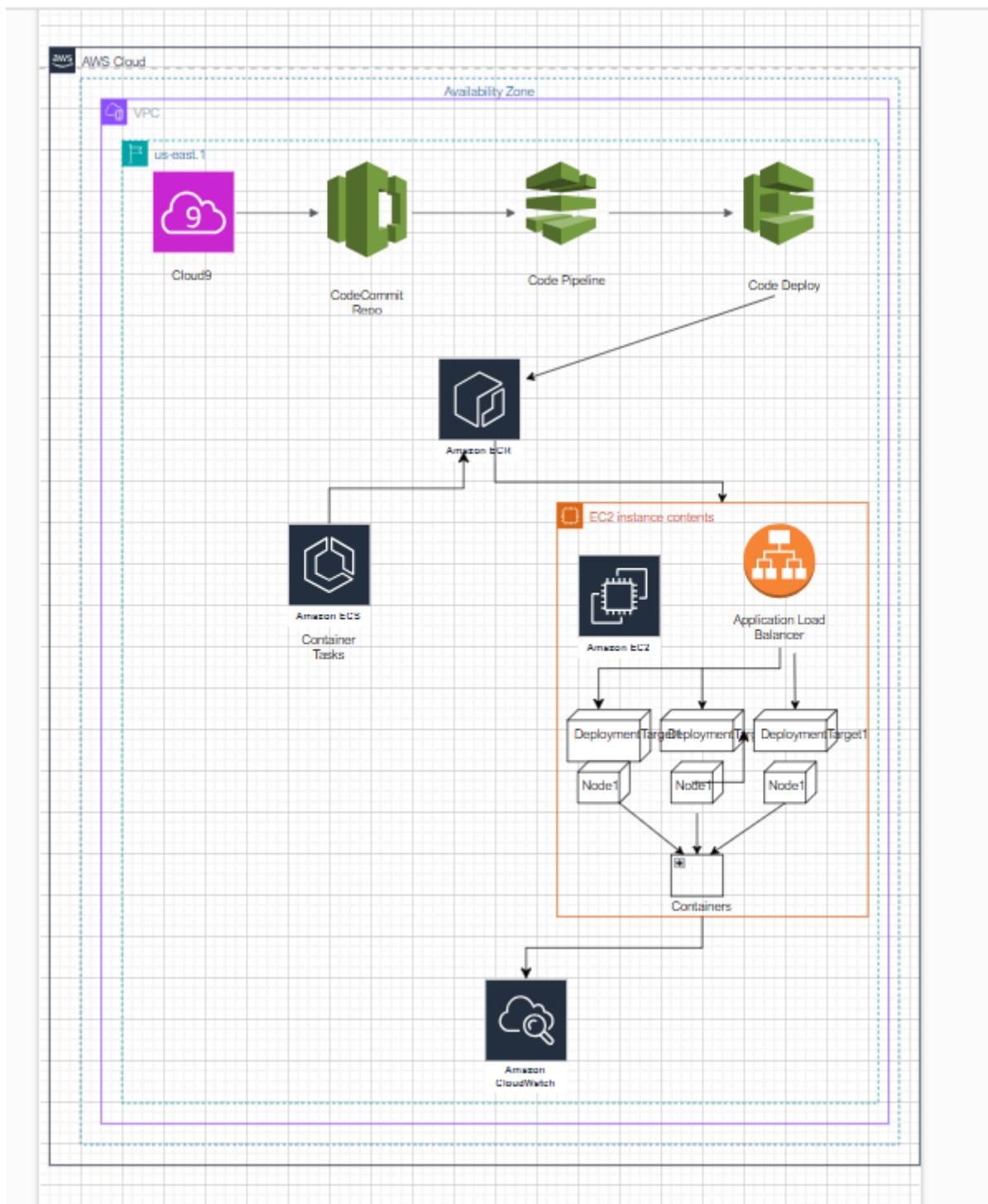
1. Create an architectural diagram to illustrate what you plan to build. Consider how you will accomplish each requirement in the solution. Read through the phases in this document to be aware of which AWS services and features you have been asked to use. Be sure to include the following services or resources in your diagram:

- Amazon Virtual Private Cloud (Amazon VPC)
- Amazon EC2: Instances, Application Load Balancer, target groups
- AWS CodeCommit: Repository
- AWS CodeDeploy

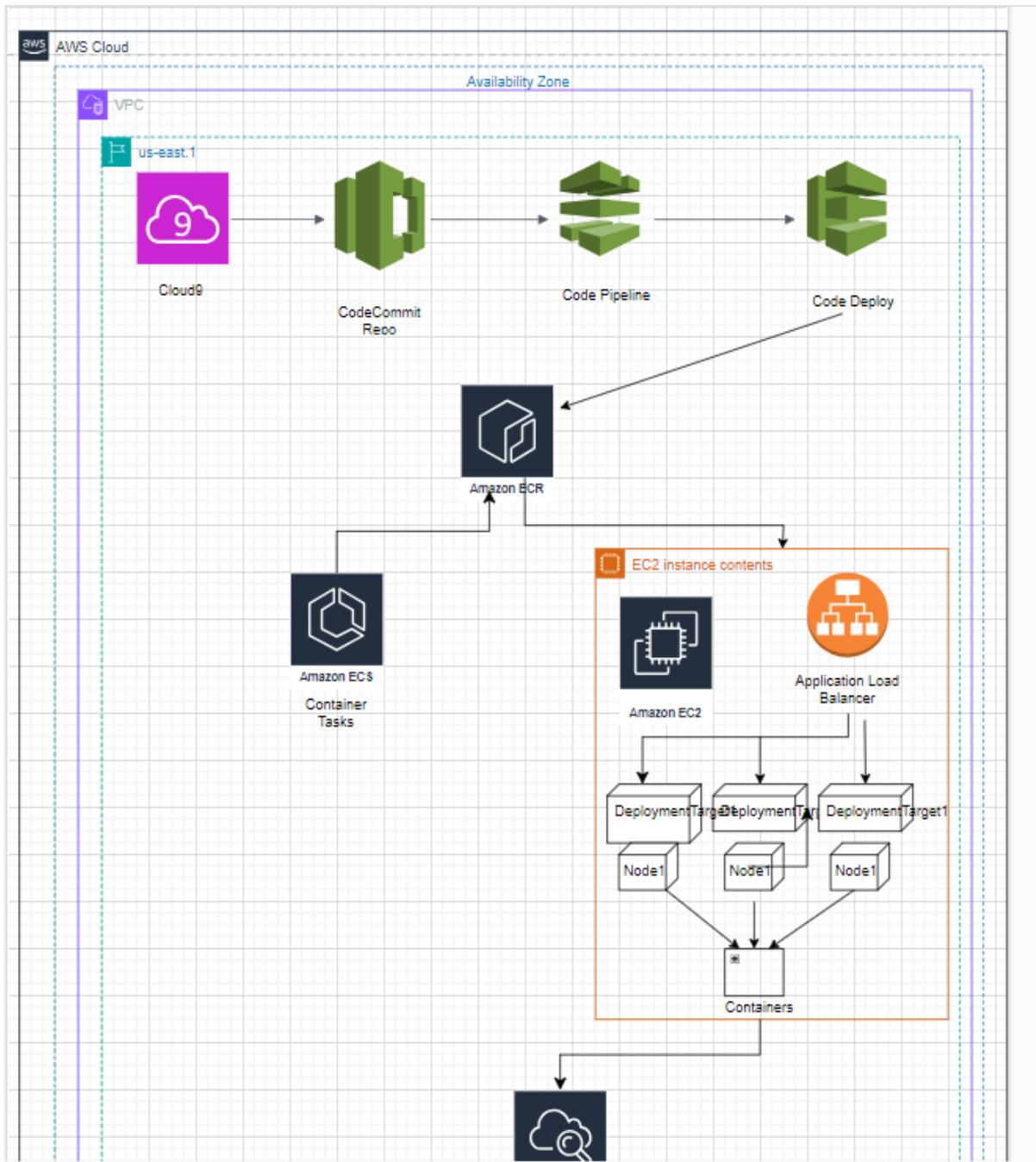
- AWS CodePipeline: Pipeline
- Amazon Elastic Container Service (Amazon ECS): Services, containers, tasks
- Amazon Elastic Container Registry (Amazon ECR): Repository
- AWS Cloud9 environment
- AWS Identity and Access Management (IAM): Roles
- Amazon Relational Database Service (Amazon RDS)
- Amazon CloudWatch: Logs

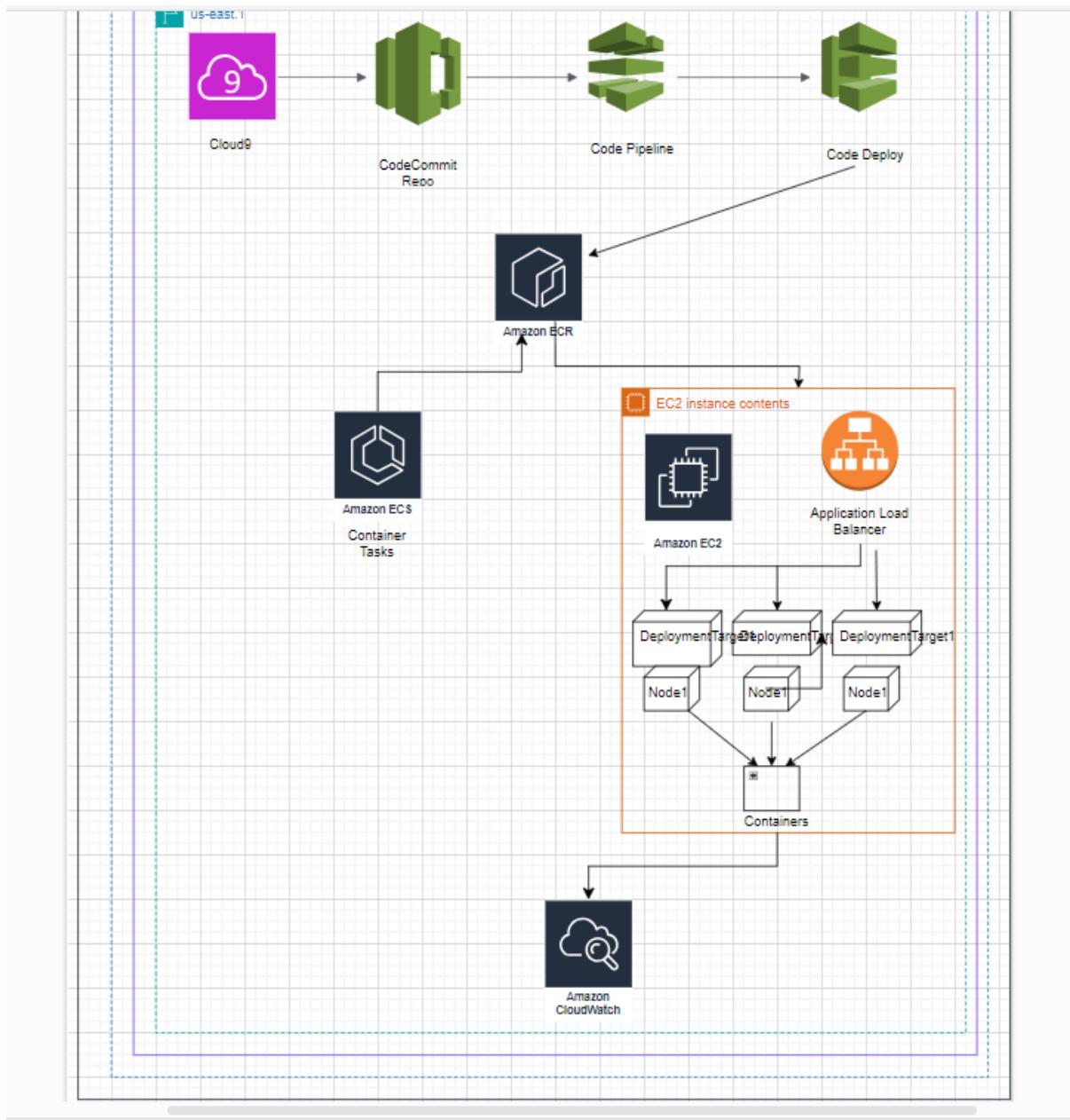
I used **draw.io** to design the AWS architecture diagram.





This is more zoomed version of the above architecture diagram split in two parts:





Flow Explanation:

The entire architecture should be deployed within an Amazon Virtual Private Cloud (Amazon VPC), and the VPC should be associated with one or more Availability Zones.

1. Developers use the AWS Cloud9 IDE to write, build, and test the application code.
2. The application code is committed to the AWS CodeCommit repository.
3. AWS CodePipeline orchestrates the CI/CD pipeline, triggering the build, test, and deployment stages.
4. Docker container images for the microservices are built and pushed to the Amazon Elastic Container Registry (ECR).
5. AWS CodeDeploy deploys the new container images to the Amazon ECS cluster, using a blue/green deployment strategy.

6. The application running on the ECS cluster within the Amazon VPC sends logs and metrics to AWS CloudWatch Logs for centralized monitoring and logging.
7. The application communicates with the Amazon RDS database for data storage and retrieval.
8. The Application Load Balancer (ALB) distributes incoming traffic across the microservices running as tasks within the ECS cluster on EC2 instances.

AWS IAM is a service that allows you to manage access to AWS resources and services. In this architecture, IAM plays a crucial role in securing and controlling access to the various components and services.

Here's how IAM can be integrated into the architecture:

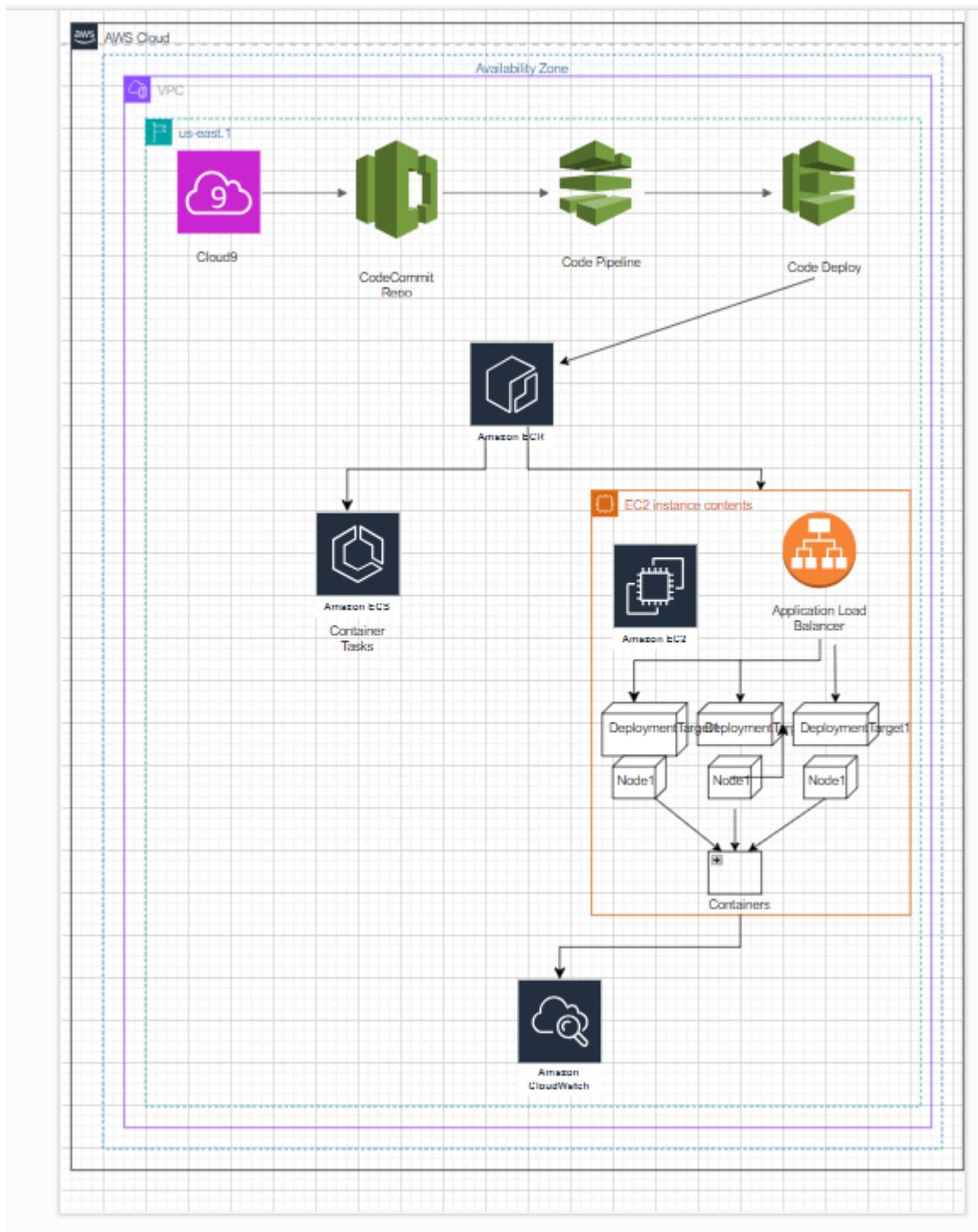
[1] IAM Roles: IAM Roles are used to grant specific permissions and access to AWS services and resources. Different IAM Roles can be created for different components of the architecture.

- CodeCommit Repository: An IAM Role can be created to allow the CodeCommit repository to be accessed and managed by authorized users or services.
- CodePipeline: An IAM Role can be created for the CodePipeline to have the necessary permissions to access the CodeCommit repository, Amazon ECR, ECS, and other services involved in the CI/CD pipeline.
- CodeDeploy: An IAM Role can be created to allow CodeDeploy to deploy applications to the ECS cluster.
- Amazon ECS: An IAM Role can be created for the ECS service to interact with other AWS services like ECR, ELB, CloudWatch Logs, etc.
- Amazon EC2: IAM Roles can be assigned to the EC2 instances in the ECS cluster to grant them the necessary permissions to interact with other services like ECR, ELB, and CloudWatch Logs.

[2] IAM Policies: IAM Policies define the permissions that an IAM Role or User has. Different policies can be created and attached to the respective IAM Roles to grant them the necessary permissions.

[3] IAM Users: IAM Users can be created for developers or administrators who need to access and manage the AWS resources and services. These users can be assigned appropriate IAM Roles and policies based on their responsibilities.

This flow demonstrates how the different AWS services are interconnected and work together to enable a continuous integration, continuous deployment (CI/CD) pipeline, containerized microservices deployment, load balancing, and centralized logging and monitoring.



Task 1.2: Develop a cost estimate

Develop a cost estimate that shows the cost to run the solution that you created an architecture diagram for. Assume that the solution will run in the us-east-1 Region for 12 months. Use the for this estimate.

AWS Pricing Calculator

Estimate the cost for your architecture solution.

Configure a cost estimate that fits your unique business or personal needs with AWS products and services.

Create an estimate
Start your estimate with no commitment, and explore AWS services and pricing for your architecture needs.

Create estimate

How it works

More resources

- User guide
- Pricing assumptions and variations
- Need help with estimation? Connect with AWS certified expert on AWS IQ

Hi, I can connect you with an AWS representative or answer questions you have on AWS.

Add service

AWS services (149)

Search by location type
See the services that are available in your region, wave length zone, and local zone.

Search all services
Choose a service or workload to configure an estimate.

Choose a location type Info
Region: US East (Ohio)

Find Service:

AWS Application Migration Service
AWS Application Migration Service minimizes time-intensive, error-prone manual processes by automatically converting your source servers from

Amazon API Gateway
Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs

Amazon AppFlow
Amazon AppFlow offers significant cost-savings advantage compared to building connectors in-house or using other application integration

Upfront cost: 0.00 USD Monthly cost: 0.00 USD Total 12 months cost 0.00 USD (Includes upfront cost)

View summary

Create estimate: Configure Amazon EC2

Description

Choose a location type Info
Region: US East (Ohio)

Choose a Region

EC2 specifications Info

Tenancy
Choose the tenancy type to run your Amazon EC2 instances on.
 Shared Instances

Operating system
Choose the operating system to run your Amazon EC2 instances on.
 Linux

Workloads
Choose the graph that best represents your monthly workload.

Constant usage Daily spike traffic Weekly spike traffic Monthly spike traffic

Total Upfront cost: 0.00 USD Total Monthly cost: 5.07 USD

Show Details ▾ Cancel Save and view summary Save and add service

aws pricing calculator

Feedback Language: English ▾ Contact Sales ⓘ Create an AWS Account

Choose the graph that best represents your monthly workload

Constant usage Daily spike traffic Weekly spike traffic Monthly spike traffic

Number of instances
Please specify the total number of instances that you need each month.

1

EC2 Instances (5)
Based on your inputs, this is the lowest-cost EC2 instance: **c5.9xlarge**
Chosen instance: **t4g.nano** | Family: **t4g** | 2vCPU | 0.5 GiB Memory

Search instance type

Instance family [Info](#) vCPUs Memory (GiB) Network performance
 8 60 GiB Any Network Performance

Show only current generation instances.

Total Upfront cost: 0.00 USD Total Monthly cost: 1.53 USD

Show Details ⓘ Cancel Save and view summary Save and add service

aws pricing calculator

Feedback Language: English ▾ Contact Sales ⓘ Create an AWS Account

Select the container and options to find your best price

Compute Savings Plans
One plan that automatically applies to all usage on EC2, Fargate, and Lambda. Up to 66% discount. [Learn more](#)

Reservation term
 1 year 3 year

Payment Options
 No upfront Partial upfront All upfront

EC2 Instance Savings Plans
Get deeper discount when you only need one instance family and region. Up to 72% discount. [Learn more](#)

Reservation term
 1 year 3 year

Payment Options
 No upfront Partial upfront All upfront

On-Demand
Maximize flexibility. [Learn more](#)

Expected utilization
Enter the expected usage of Amazon EC2 instances

Usage
100

Spot Instances
Minimize cost by leveraging EC2's spare capacity. Recommended for fault tolerant and interruption tolerant applications. [Learn more](#)

The historical average discount for t4g.nano is 57%

Assume percentage discount for my estimate
-1

Actual spot instance pricing varies
With spot instances, you pay the spot price that's in effect for the time period your instance is running

Total Upfront cost: 0.00 USD Total Monthly cost: 1.90 USD

Show Details ⓘ Cancel Save and view summary Save and add service

aws pricing calculator

Feedback Language: English ▾ Contact Sales ⓘ Create an AWS Account

► Amazon Elastic Block Store (EBS) - *optional* [Info](#)

► Detailed monitoring - *optional* [Info](#)

► Data transfer - *optional*

► Additional costs - *optional*

Acknowledgement
AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. [Learn more](#) ⓘ

Total Upfront cost: 0.00 USD Total Monthly cost: 1.90 USD

Show Details ⓘ Cancel Save and view summary Save and add service

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Successfully added Amazon EC2 estimate.

AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon Virtual Private Cloud (VPC) [Info](#)

Step 1 Add service

Description
Enter a description for your estimate

Step 2 Configure service

Choose a location type [Info](#) Choose a Region

Region US East (Ohio)

Select VPC service(s) that you want to estimate

VPN Connection Network Address Translation (NAT) Gateway Public IPv4 Address
 Transit Gateway AWS PrivateLink Data Transfer
 Gateway Load Balancer IPAM Network Access Analyzer

Total Upfront cost: 0.00 USD Total Monthly cost: 0.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Site-to-Site VPN settings [Info](#)

Number of Site-to-Site VPN Connections
4

Average duration for each connection Unit
24 hours per day

Show calculations

Client VPN settings [Info](#)

Number of subnet associations
Number of subnets associated to an AWS Client VPN endpoint
Enter the amount

Number of active Client VPN connections (or users)

Total Upfront cost: 0.00 USD Total Monthly cost: 146.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Client VPN settings [Info](#)

Number of subnet associations
Number of subnets associated to an AWS Client VPN endpoint
2

Number of active Client VPN connections (or users)
Connection limitations apply based on the number of subnet associations
Value Unit
25 per day

Average duration for each connection Unit
10 hours per day

Working days per month
22

Show calculations

Total Upfront cost: 0.00 USD Total Monthly cost: 567.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

AWS Pricing Calculator > My Estimate > Add service

Add service Info

Step 1
Add service

Step 2
Configure service

AWS services (11)

Search by location type See the services that are available in your region, wave length zone, and local zone.

Search all services Choose a service or workload to configure an estimate.

Choose a location type Info

Region: US East (Ohio)

Find Service: Q code

Amazon CodeGuru Reviewer

Amazon CodeGuru Reviewer is a service that uses program analysis and machine learning to detect potential defects that are difficult for developers to find and recommends fixes in your Java code.

AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.

AWS CodeDeploy

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises

Upfront cost: 0.00 USD Monthly cost: 568.90 USD Total 12 months cost 6,826.80 USD (Includes upfront cost)

View summary

aws pricing calculator

Successfully added Amazon Virtual Private Cloud (VPC) estimate.

AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS CodeDeploy Info

Step 1
Add service

Step 2
Configure service

Create estimate: Configure AWS CodeDeploy Info

Description: AWSCodeDeploy

Choose a location type Info

Region: US East (Ohio)

CodeDeploy On-Premises Info

For CodeDeploy you only pay for executing deployments to on-premises instances.

Number of on-premise instances: 10

Number of deployments: 1 Unit: per month

Total Upfront cost: 0.00 USD Total Monthly cost: 0.80 USD

Show Details ▾ Cancel Save and view summary Save and add service

aws pricing calculator

Configure service

Region: US East (Ohio)

CodeDeploy On-Premises Info

For CodeDeploy you only pay for executing deployments to on-premises instances.

Number of on-premise instances: 10

Number of deployments: 4 Unit: per month

▶ Show calculations

Acknowledgement

Total Upfront cost: 0.00 USD Total Monthly cost: 0.80 USD

Show Details ▾ Cancel Save and view summary Save and add service

AWS pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Step 1 Add service Step 2 Configure service

Add service Info

AWS services (1)

Search by location type See the services that are available in your region, wave length zone, and local zone.

Search all services Choose a service or workload to configure an estimate.

Choose a location type Info

Region US East (Ohio)

Find Service

AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates.

Upfront cost: 0.00 USD Monthly cost: 569.70 USD Total 12 months cost 6,836.40 USD (Includes upfront cost)

[View summary](#)

AWS pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Successfully added AWS CodeDeploy estimate.

AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS CodePipeline Info

Step 1 Add service Step 2 Configure service

Create estimate: Configure AWS CodePipeline Info

Description

Choose a location type Info

Region US East (Ohio)

Choose a Region

CodePipeline Pricing Info

Number of active pipelines used per account per month
An active pipeline is a pipeline that has existed for more than 30 days and has at least one code change that runs through it during the month. An active pipeline is not prorated for partial months.

Total Upfront cost: 0.00 USD Total Monthly cost: 1.00 USD

Show Details ▾ Cancel Save and view summary Save and add service

AWS pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Step 1 Add service Step 2 Configure service

Add service Info

AWS services (1)

Search by location type See the services that are available in your region, wave length zone, and local zone.

Search all services Choose a service or workload to configure an estimate.

Choose a location type Info

Region US East (Ohio)

Find Service

Amazon Elastic Container Registry

Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR eliminates the need to operate your own container repositories

Upfront cost: 0.00 USD Monthly cost: 570.70 USD Total 12 months cost 6,848.40 USD (Includes upfront cost)

[View summary](#)

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Successfully added AWS CodePipelineestimate.

AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon Elastic Container Registry [Info](#)

Step 1 Add service

Description

Step 2 Configure service

Choose a location type [Info](#) Choose a Region

Region

Storage settings [Info](#)

To estimate your monthly cost, enter the amount of data you will store in your repositories. The calculations below exclude Free Tier discounts.

Amount of data stored Unit

Total Upfront cost: 0.00 USD Total Monthly cost: 5.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Data Transfer [Info](#)

Inbound Data Transfer
Enter the data you expect to transfer into US East (Ohio)

Data transfer from Enter Amount Data amount
Add inbound data transfer

Outbound Data Transfer
Enter the data you expect to transfer out of US East (Ohio)

Data transfer to Enter Amount Data amount
Add outbound data transfer

Show calculations

Total Upfront cost: 0.00 USD Total Monthly cost: 5.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Step 1 Add service [Info](#) Bulk import

Step 2 Configure service

AWS services (1)

Search by location type See the services that are available in your region, wave length zone, and local zone.

Search all services Choose a service or workload to configure an estimate.

Choose a location type [Info](#) Choose a Region

Region

Find Service

Amazon CloudWatch

Amazon CloudWatch is a monitoring and management service that provides data and actionable insights for AWS, hybrid, and on-premises applications and infrastructure resources.

Upfront cost: 0.00 USD Monthly cost: 575.70 USD Total 12 months cost 6,908.40 USD (Includes upfront cost)

View summary

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Successfully added Amazon Elastic Container Registry estimate.

AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon CloudWatch

Create estimate: Configure Amazon CloudWatch [Info](#)

Step 1 [Add service](#)

Description

Step 2 [Configure service](#)

Choose a location type [Info](#) Choose a Region

Region US East (Ohio)

Metrics [Info](#)

The calculations below exclude free tier discounts.

Number of Metrics (includes detailed and custom metrics)

Total Upfront cost: 0.00 USD Total Monthly cost: 0.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

AWS Pricing Calculator > My Estimate > Add service

Add service [Info](#)

Step 1 [Add service](#)

Step 2 [Configure service](#)

Bulk import Cancel

Search by location type See the services that are available in your region, wave length zone, and local zone.

Choose a location type [Info](#) Choose a Region

Region US East (Ohio)

Find Service X

Amazon RDS Custom for Oracle Amazon RDS Custom for SQL Server Amazon RDS for Db2

Amazon Relational Database Service (Amazon RDS) Custom is a managed database service for legacy, custom, and packaged applications that require access to the underlying OS and DB environment. Amazon RDS Custom automates setup, operation, and scaling.

Amazon Relational Database Service (Amazon RDS) Custom is a managed database service for legacy, custom, and packaged applications that require access to the underlying OS and DB environment. Amazon RDS Custom automates setup, operation, and scaling.

Amazon RDS makes it easy to set up, operate, and scale db2 deployments in the cloud. With Amazon RDS, you can deploy scalable db2 deployments in minutes with cost-efficient and resizable hardware capacity.

Upfront cost: 0.00 USD Monthly cost: 576.90 USD Total 12 months cost: 6,922.80 USD (Includes upfront cost)

View summary

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Successfully added Amazon CloudWatch estimate.

AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon RDS for Db2

Create estimate: Configure Amazon RDS for Db2 [Info](#)

Step 1 [Add service](#)

Description

Step 2 [Configure service](#)

Choose a location type [Info](#) Choose a Region

Region US East (Ohio)

▼ Db2 instance specifications [Info](#)

Nodes Enter the number of DB instances that you need.

Q db.m6i.12xlarge X

Total Upfront cost: 0.00 USD Total Monthly cost: 6,241.72 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Selected Instance:
db.m6i.12xlarge
vCPU: 48
Memory: 192 GiB

Utilization (On-Demand only)
With utilization, you still have to stop the instance to get the cost benefit. Utilization only affects OnDemand pricing for instances and not the storage, backups, etc.

Value	Unit
100	%Utilized/Month

Deployment option
Multi-AZ

Pricing Model
OnDemand

Database edition
Advanced

Total Upfront cost: 0.00 USD
Total Monthly cost: 6,241.72 USD

Show Details ▾ Cancel Save and view summary Save and add service

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

Storage [Info](#)

Storage volume
General Purpose SSD (gp3)

Note
For GP3 Storage volumes, all IOPS and throughput use below the baseline is included at no additional charge. For volumes below 400 GiB of allocated storage, the baseline provisioned IOPS is 3,000 and baseline throughput is 125 MiBps. Volumes of 400 GiB and above, baseline provisioned IOPS is 12,000 and baseline throughput is 500 MiBps. There is an additional charge for provisioned IOPS and throughput above baseline.

Storage amount	Unit
20	GB

▶ Show calculations

▶ Backup Storage [Info](#)

Total Upfront cost: 0.00 USD
Total Monthly cost: 6,241.72 USD

Show Details ▾ Cancel Save and view summary Save and add service

aws pricing calculator

Feedback Language: English ▾ Contact Sales ▾ Create an AWS Account

AWS Pricing Calculator > My Estimate > Add service

Step 1 Add service

Step 2 Configure service

Add service [Info](#)

AWS services (1)

Search by location type
See the services that are available in your region, wave length zone, and local zone.

Search all services
Choose a service or workload to configure an estimate.

Choose a location type [Info](#)

Region	Choose a Region
Region	US East (Ohio)

Find Service

AWS IAM Access Analyzer

IAM Access Analyzer guides you towards least privilege by providing tools to set, verify, and refine permissions. IAM Access Analyzer provides access

Upfront cost: 0.00 USD Monthly cost: 6,818.62 USD Total 12 months cost 81,823.44 USD (includes upfront cost)

View summary

AWS pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS IAM Access Analyzer

Create estimate: Configure AWS IAM Access Analyzer [Info](#)

Step 1

Add service

Description: AWS_IAM

Step 2

Configure service

Choose a location type [Info](#) Choose a Region

Region: US East (Ohio)

▼ Unused Access Findings [Info](#)

The charges for unused access findings are based on the number of IAM roles and users analyzed per analyzer per month. As IAM roles and users are global, you don't need to create analyzers across multiple regions.

Number of accounts to monitor: 2

Average roles per account: 1

Total Upfront cost: 0.00 USD Total Monthly cost: 0.80 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

AWS pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

1

▶ Show calculations

▼ Custom Policy Checks [Info](#)

Number of requests to CheckNoNewAccess API Unit: per month

Enter a number

Number of requests to CheckAccessNotGranted API Unit: per month

Enter a number

▶ Show calculations

Total Upfront cost: 0.00 USD Total Monthly cost: 0.80 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

AWS pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

AWS Pricing Calculator > My Estimate > Add service

Add service [Info](#)

Step 1

Add service

Step 2

Configure service

Bulk import

Cancel

AWS services (1)

Search by location type
See the services that are available in your region, wave length zone, and local zone.

Search all services
Choose a service or workload to configure an estimate.

Find Service

AWS CodeCommit

AWS CodeCommit is a secure, highly scalable, fully managed source control service that hosts private Git repositories.

[Product page](#) [Configure](#)

Upfront cost: 0.00 USD Monthly cost: 6,819.42 USD Total 12 months cost 81,833.04 USD (includes upfront cost) **View summary**

AWS pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS CodeCommit

Create estimate: Configure AWS CodeCommit Info

Step 1

Add service

Description
Amazon_CodeCommit

Step 2

Configure service

Service Setting Info

Any Active user Receives:

- 1,000 repositories per account; up to 25,000 upon requests. [Learn More](#)
- 50 GB-month of storage
- 10,000 Git requests/month

Number of active users
1

Total Upfront cost: 0.00 USD Total Monthly cost: 0.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

AWS pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

AWS Pricing Calculator > My Estimate > Add service

Add service Info

Step 1

Add service

Step 2

Configure service

Bulk import Cancel

AWS services (1)

Search by location type See the services that are available in your region, wave length zone, and local zone.

Search all services Choose a service or workload to configure an estimate.

Find Service

AWS Fargate

AWS Fargate is a serverless compute engine for containers that works with both Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). Fargate makes it easy for you to focus on building your applications. Fargate removes the need to provision and manage servers, lets you specify and pay for resources per

Upfront cost: 0.00 USD Monthly cost: 6,819.42 USD Total 12 months cost: \$1,833.04 USD (Includes upfront cost)

View summary

AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS Fargate

Create estimate: Configure AWS Fargate Info

Step 1

Add service

Description
Amazon_ECS

Step 2

Configure service

Choose a location type Info Choose a Region

Region US East (Ohio)

On Demand Info

Fargate Spot and Compute Savings Plans are currently not supported on the AWS Pricing Calculator.

With AWS Fargate, there are no upfront payments and you only pay for the resources that you use. Pricing is based on requested vCPU, memory, and storage resources consumed by your Amazon ECS Task or Amazon EKS Pod.

Operating system
Linux

Total Upfront cost: 0.00 USD Total Monthly cost: 0.00 USD

Show Details ▾ Cancel Save and view summary **Save and add service**

aws pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

1	minutes
Amount of vCPU allocated Enter the amount between 0.25 vCPU and 16 vCPU	
1	
vCPU selected supports memory values between 2 GB and 8 GB, in 1 GB increments	
Amount of memory allocated Unit Enter the amount GB	
Amount of ephemeral storage allocated for Amazon ECS Enter the amount between 20 GB and 200 GB. The first 20 GB are at no additional charge, you only pay for any additional storage that you configure for the Task.	
Value 20	Unit GB
Show calculations	

Total Upfront cost: 0.00 USD Total Monthly cost: 0.00 USD Show Details ▾ Cancel Save and view summary Save and add service

aws pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

AWS Pricing Calculator > My Estimate > Add service

Step 1 Add service Step 2 Configure service

Add service [info](#) Bulk import Cancel

Search by location type See the services that are available in your region, wave length zone, and local zone.

Search all services Choose a service or workload to configure an estimate.

Find Service

AWS services (182)

AWS Application Migration Service AWS Application Migration Service minimizes time-intensive, error-prone manual processes by automatically converting your source servers from physical, virtual, and cloud infrastructure to run natively on AWS. It further simplifies your migration by allowing you to use the same automated process for a wide range of applications.	Amazon API Gateway Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the front door for applications to access data, business logic, or functionality from your backend services.	Amazon AppFlow Amazon AppFlow offers significant cost-savings advantage compared to building connectors in-house or using other application integration services. There are no upfront charges or fees to use AppFlow, and customers only pay for the number of flows they run and the volume of data processed.
--	--	--

Upfront cost: 0.00 USD Monthly cost: 6,819.42 USD Total 12 months cost **81,833.04 USD** (Includes upfront cost) [View summary](#)

aws pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

AWS Pricing Calculator > My Estimate My Estimate [Edit](#) Export ▾ Share

Estimate summary Info	Getting Started with AWS
Upfront cost 0.00 USD	Monthly cost 6,819.42 USD
Total 12 months cost 81,833.04 USD Includes upfront cost	

My Estimate [Add service](#)

<input type="checkbox"/> Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
Amazon EC2	-	0.00 USD	1.90 USD	-	US East (Ohio)	Troubleshoot
Amazon Virtual Priv...	-	0.00 USD	567.00 USD	-	US East (Ohio)	Hi, I can connect you with an AWS representative or answer questions you have on AWS.

AWS pricing calculator

Feedback Language: English ▾ Contact Sales Create an AWS Account

AWS Pricing Calculator > My Estimate

My Estimate [Edit](#)

Estimate summary [Info](#)

Upfront cost 0.00 USD	Monthly cost 6,819.42 USD	Total 12 months cost 81,833.04 USD Includes upfront cost
--------------------------	------------------------------	---

Getting Started with AWS

Export ▾ Share CSV PDF JSON

Get started for free Contact Sales

My Estimate

<input type="checkbox"/>	Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
<input type="checkbox"/>	Amazon EC2	-	0.00 USD	1.90 USD	-	US East (Ohio)	Tenancy (Shared Inst...)
<input type="checkbox"/>	Amazon Virtual Priva...	-	0.00 USD	567.00 USD	-	US East (Ohio)	Working days per mo...

[Find resources](#)

Duplicate Delete Move to Create group Add support **Add service**

< 1 > ⓘ

(...)

Contact your AWS representative: [Contact Sales](#)

Export date: 4/26/2024

Language: English

Estimate URL: <https://calculator.aws/#/estimate?id=9de1d50773d18c50560cd1f0f72cf264d5cf6701>

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	6,819.42 USD	81,833.04 USD
Includes upfront cost		

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon EC2	No group applied	US East (Ohio)	0.00 USD	1.90 USD

Status: -**Description:**

Config summary: Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1), Advance EC2 instance (t4g.nano), Pricing strategy (EC2 Instance Savings Plans 1yr No Upfront), Enable monitoring (disabled), DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month), DT Intra-Region: (0 TB per month)

Amazon Virtual Private Cloud (VPC)	No group applied	US East (Ohio)	0.00 USD	567.00 USD
------------------------------------	------------------	----------------	----------	------------

Status: -**Description:**

Config summary: Working days per month (22), Number of Site-to-Site VPN Connections (4), Number of subnet associations (2)

AWS CodeDeploy	No group applied	US East (Ohio)	0.00 USD	0.80 USD
----------------	------------------	----------------	----------	----------

Status: -**Description:** AWSCodeDeploy

Config summary: Number of on-premise instances (10), Number of deployments (4 per month)

4/26/24, 8:24 PM	My Estimate - AWS Pricing Calculator			
AWS CodePipeline	No group applied	US East (Ohio)	0.00 USD	1.00 USD
Status: -				
Description: AWSCodePipeline				
Config summary: Number of active pipelines used per account per month (2)				
Amazon Elastic Container Registry	No group applied	US East (Ohio)	0.00 USD	5.00 USD
Status: -				
Description: AmazonECR				
Config summary: Amount of data stored (50 GB per month)				
Amazon CloudWatch	No group applied	US East (Ohio)	0.00 USD	1.20 USD
Status: -				
Description: AWSCloudwatch				
Config summary: Number of Metrics (includes detailed and custom metrics) (4)				
Amazon RDS for Db2	No group applied	US East (Ohio)	0.00 USD	6,241.72 USD
Status: -				
Description: AmazonRDS				
Config summary: Storage volume (General Purpose SSD (gp3)), Storage amount (20 GB), Nodes (1), Instance Type (db.m6i.12xlarge), Utilization (On-Demand only) (100 %Utilized/Month), Deployment option (Multi-AZ), Pricing Model (OnDemand), Database edition (Advanced)				
AWS IAM Access Analyzer	No group applied	US East (Ohio)	0.00 USD	0.80 USD
Status: -				
Description: AWS_IAM				
Config summary: Number of accounts to monitor (2), Average roles per account (1), Average users per account (1)				
AWS CodeCommit	No group applied		0.00 USD	0.00 USD
Status: -				
Description: Amazon_CodeCommit				
Config summary: Number of active users (1)				
AWS Fargate	No group applied	US East (Ohio)	0.00 USD	0.00 USD
Status: -				
Description: Amazon_ECS				
Config summary: Operating system (Linux), CPU Architecture (x86), Average duration (1 minutes), Number of tasks or pods (1 per day), Amount of ephemeral storage allocated for Amazon ECS (20 GB)				

Phase 2: Analyzing the infrastructure of the monolithic application

In this task, you will analyze the current application infrastructure and then test the web application.

Task 2.1: Verify that the monolithic application is available

1. Verify that the monolithic web application is accessible from the internet.

The screenshot shows two side-by-side views of the AWS console. On the left is the 'Console Home' page, which includes sections for 'Recently visited' services (EC2, Simple Queue Service, EC2 Image Builder, CloudFormation), 'Applications' (empty), and 'Welcome to AWS'. On the right is the 'EC2 Dashboard', which displays resource statistics (1 instance running, 0 auto scaling groups, 0 dedicated hosts, 0 elastic IPs, 1 instance, 1 key pair, 0 load balancers, 1 placement group, 0 security groups, 0 snapshots, 1 volume) and provides options to 'Launch instance' or 'Migrate a server'.

- Navigate to the Amazon EC2 console.

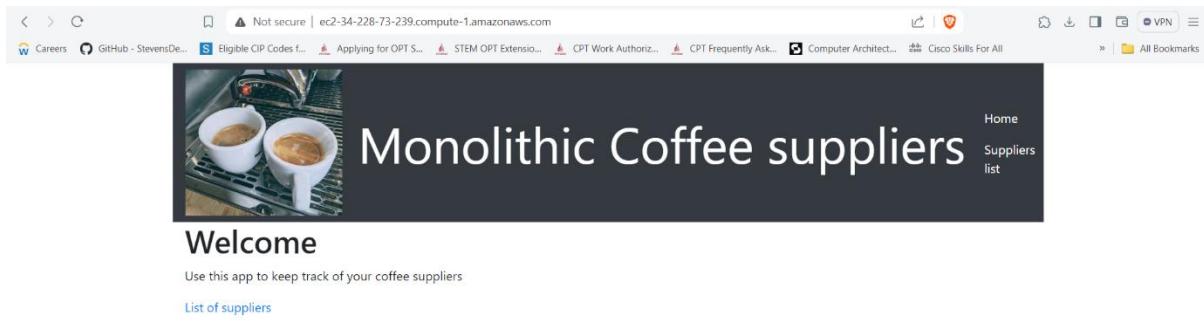
The screenshot shows the AWS EC2 Instances page. A single instance, "MonolithicApp...", is listed as "Running". The Public IPv4 address is 34.228.73.239. The instance type is t2.micro.

- Copy the Public IPv4 address of the **MonolithicAppServer** instance, and load it in a new browser tab.

The screenshot shows the AWS EC2 Instances page with the same instance details. A tooltip message "Public IPv4 DNS copied" appears near the Public IPv4 DNS link, which is now highlighted in blue.

The coffee suppliers website displays.

Note: The page is available at <http://> instead of <https://>. Your browser might indicate that the site isn't secure because it doesn't have a valid SSL/TLS certificate. You can ignore the warning in this development environment.



Task 2.2: Test the monolithic web application

In this task, you add data to the web application, test the functionality, and observe the different URL paths that are used to display the different pages. These URL paths are important to understand for when you divide this application into microservices later.

1. Choose List of suppliers.

Notice that the URL path includes /suppliers.



2. Add a new supplier.

- On the page where you add a new supplier, notice that the URL path includes /supplier-add.

The screenshot shows a web browser with the URL <http://ec2-34-228-73-239.compute-1.amazonaws.com/suppliers>. The page title is "Monolithic Coffee suppliers". A header bar includes links for "Home" and "Suppliers list". Below the header is a banner image of two cups of coffee on a espresso machine. The main content area is titled "All suppliers" and contains a table with columns: Name, Address, City, State, Email, and Phone. A green button labeled "Add a new supplier" is visible.

- Fill in all of the fields to create a supplier entry.

The screenshot shows a web browser with the URL <http://ec2-34-228-73-239.compute-1.amazonaws.com/supplier-add>. The form fields are as follows:

Name	<input type="text" value="Nancy_Supplier1"/>
Name of this supplier	
Address	<input type="text" value="1 Castle Point Terrace"/>
Address for this supplier	
City	<input type="text" value="Hoboken"/>
City for this supplier	
State	<input type="text" value="New Jersey"/>
State for this supplier	
Email	<input type="text" value="ngupta19@stevens.edu"/>
Email for this supplier	
Phone	<input type="text" value="5512098571"/>
Phone number for this supplier	
<input type="button" value="Submit"/>	

The screenshot shows the same web browser and URL as the previous screenshot. The table now includes a new row for the supplier "Nancy_Supplier1". The "Phone" column for this entry is highlighted in blue, indicating it has been edited. The "edit" button is also visible in the same row.

3. Edit an entry.

- On the page where you edit a supplier entry, notice that the URL path now includes `supplier-update/1`.

Not secure | ec2-34-228-73-239.compute-1.amazonaws.com/supplier-update/1

Careers GitHub - StevensDe... Eligible CIP Codes f... Applying for OPT S... STEM OPT Extension... CPT Work Authoriz... CPT Frequently Ask... Computer Architect... Cisco Skills For All

Home Suppliers list

All fields are required

Name
Nancy_Supplier1
Name of this supplier

Address
1 castle point terrace
Address for this supplier

City
Hoboken
City for this supplier

State
New Jersey
State for this supplier

- **Modify the record in some way and save the change.**

Not secure | ec2-34-228-73-239.compute-1.amazonaws.com/supplier-update/1

Careers GitHub - StevensDe... Eligible CIP Codes f... Applying for OPT S... STEM OPT Extension... CPT Work Authoriz... CPT Frequently Ask... Computer Architect... Cisco Skills For All

Home Suppliers list

All fields are required

Name
Nancy_Supplier1
Name of this supplier

Address
1 castle point terrace
Address for this supplier

City
Hoboken
City for this supplier

State
New Jersey
State for this supplier

Notice that the change was saved in the record.

Nancy_Supplier2

Name of this supplier

Address

5 castle point terrace

Address for this supplier

City

Jersey City

City for this supplier

State

New Jersey

State for this supplier

Email

ngupta20@stevens.edu

Email for this supplier

Phone

987456321

Phone number for this supplier

Submit

Delete this supplier

Monolithic Coffee suppliers

All suppliers

Name	Address	City	State	Email	Phone
Nancy_Supplier2	5 castle point terrace	Jersey City	New Jersey	ngupta20@stevens.edu	987456321

Add a new supplier

Task 2.3: Analyze how the monolithic application runs

1. Use EC2 Instance Connect to connect to the *MonolithicAppServer* instance.

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running

MonolithicApp... i-0a1861b7fa6d1ac29 Running t2.micro 2/2 checks passed us-east-1a ec2-34-228-73-239.compute-1.amazonaws.com

i-0a1861b7fa6d1ac29 (MonolithicAppServer)

Details Status and alarms New Monitoring Security Networking Storage Tags

Instance summary

Instance ID: i-0a1861b7fa6d1ac29 (MonolithicAppServer)

Public IPv4 address: 34.228.73.239 | open address

Private IPv4 addresses

Public IPv4 DNS copied

IP address: ip-10-16-10-141.ec2.internal

Instance state: Running

Private IP DNS name (IPv4 only): ip-10-16-10-141.ec2.internal

Answer private resource DNS name

Instance type: -

Elastic IP addresses

The screenshot shows the AWS EC2 Connect interface. At the top, the URL is `us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInstance:instanceId=i-0a1861b7fa6d1ac29`. The page title is "Connect to instance". Below the title, it says "Connect to your instance i-0a1861b7fa6d1ac29 (MonolithicAppServer) using any of these options". There are four tabs: "EC2 Instance Connect" (selected), "Session Manager", "SSH client", and "EC2 serial console". Under "EC2 Instance Connect", there are two options: "Connect using EC2 Instance Connect" (selected) and "Connect using EC2 Instance Connect Endpoint". The "Connect using EC2 Instance Connect" option is described as "Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address." The "Connect using 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, there is a "Public IP address" field containing "34.228.73.239" and a "Username" field containing "ubuntu". A note at the bottom 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."

This screenshot is identical to the one above, but the "Connect" button at the bottom right of the form is now highlighted in orange, indicating it is the next step to initiate the connection.

```

System information as of Sat Apr 27 03:09:29 UTC 2024
System load: 0.39      Processes:          106
Usage of /: 31.9% of 7.69GB  Users logged in: 0
Memory usage: 23%        IPv4 address for eth0: 10.16.10.141
Swap usage: 0%
* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

299 updates can be installed immediately.
211 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: Sat Apr 27 01:30:47 2024 from 18.206.107.27
to run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-16-10-141:~$
```

i-0a1861b7fa6d1ac29 (MonolithicAppServer)
PublicIPs: 52.207.230.237 PrivateIPs: 10.16.10.141

2. Analyze how the application is running.

- In the terminal session, run the following command:

```
sudo lsof -i :80
```

What did you notice in the command output? What port and protocol is the *node* daemon using?

In the command output, it shows that the process with the PID (Process ID) 447, running as the user root, is listening on port 80. The protocol being used is TCP. This indicates that the node daemon is currently listening for incoming connections on port 80 using the TCP protocol.

```
aws | Services | Q Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674

System load: 0.39 Processes: 106
Usage of /: 31.9% of 7.69GB Users logged in: 0
Memory usage: 23% IPv4 address for eth0: 10.16.10.141
Swap usage: 0%

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

299 updates can be installed immediately,
211 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: Sat Apr 27 01:30:47 2024 from 18.206.107.27
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-16-10-141:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 447 root 1bu IPv6 21100 0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-141:~$ i-0a1861bfaf6d1ac29 (MonolithicAppServer)
PublicIPs: 52.207.230.237 PrivateIPs: 10.16.10.141
```

- Next, run the following command:

```
ps -ef | head -1; ps -ef | grep node
```

What did you notice in the command output? Which user on this EC2 instance is running a *node* process? Does the node process ID (PID) match any of the PIDs from the output of the command that you ran before the last one?

In the command output, it shows that there are two node processes running on the EC2 instance:

1. The first node process is being run by the user "root" with the PID 447. It appears to be the same node process that was identified in the previous command output, where it was shown to be listening on port 80.
 2. The second node process is also being run by the user "root" with the PID 428. This process appears to be started with the command "sudo node index.js".

The node process with PID 447 matches the PID of the process that was identified as listening on port 80 in the previous command output.

```

aws Services Search [Alt+S] N. Virginia voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾
compliance features.

https://ubuntu.com/aws/pro

299 updates can be installed immediately.
211 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: Sat Apr 27 01:30:47 2024 from 18.206.107.27
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-16-10-141:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 447 root 10u IPv6 21100 0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-141:~$ ^C
ubuntu@ip-10-16-10-141:~$ ps -ef | head -1; ps -ef | grep node
UID PID PPID C STIME TTY TIME CMD
root 428 428 0 03:09 ? 00:00:00 sudo node index.js
root 447 428 0 03:08 ? 00:00:00 node index.js
ubuntu 1200 1163 0 03:13 pts/0 00:00:00 grep --color=auto node
ubuntu@ip-10-16-10-141:~$ [REDACTED]

i-0a1861b7fa6d1ac29 (MonolithicAppServer)
Public IPs: 52.207.230.237 Private IPs: 10.16.10.141

```



```

aws Services Search [Alt+S] N. Virginia voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾
To see these additional updates run: apt list --upgradable

*** system restart required ***

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

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

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

ubuntu@ip-10-16-10-117:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 15086 root 1bu IPv6 50598 0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-117:~$ ^C
ubuntu@ip-10-16-10-117:~$ ps -ef | head -1; ps -ef | grep node
UID PID PPID C STIME TTY TIME CMD
root 15086 1 0 02:55 ? 00:00:00 node index.js
ubuntu 15612 15590 0 03:02 pts/0 00:00:00 grep --color=auto node
ubuntu@ip-10-16-10-117:~/resources/codebase_partner$ ls
app index.js node_modules package-lock.json package.json public views
ubuntu@ip-10-16-10-117:~/resources/codebase_partner$ [REDACTED]

i-0cb3245c7a777300d2 (MonolithicAppServer)
Public IPs: 3.88.29.143 Private IPs: 10.16.10.117

```

3. To analyze the application structure, run the following commands:

`cd ~/resources/codebase_partner`

`ls`

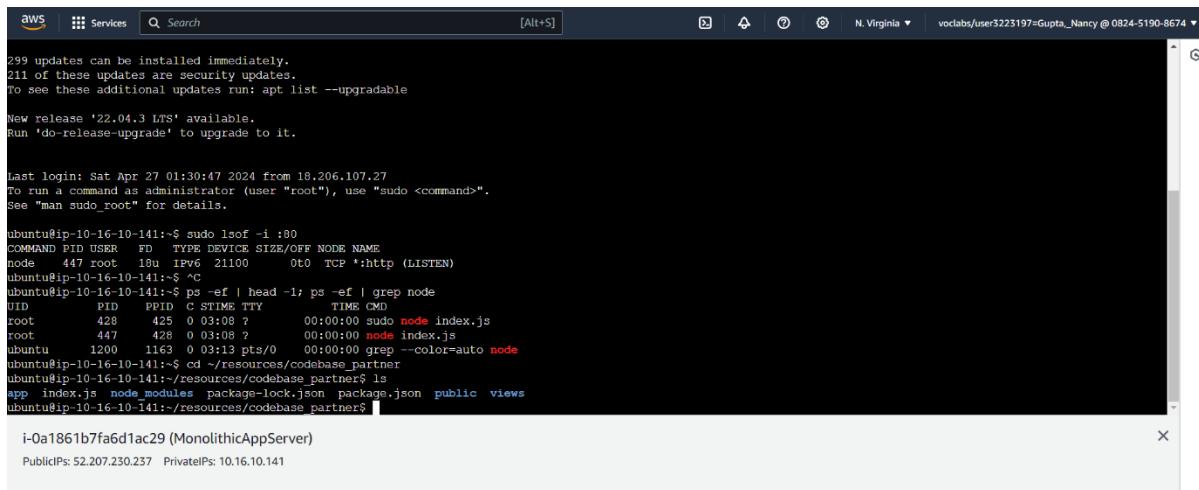
This is where the `index.js` file exists. It contains the base application logic, which you will look at in detail in a moment.

Questions for thought: Based on what you have observed, what can you determine about how and where this node application is running? How do you think it was installed? What prerequisite libraries, if any, were required to make it run? Where does the application store data?

Based on the commands I ran and the directory structure I observed:

1. Location of Node Application: The node application is located in the directory `~/resources/codebase_partner`.
2. Application Structure: Within the `codebase_partner` directory, there are several files and directories:
 - `app`: This directory likely contains additional application-specific logic or modules.
 - `index.js`: This file contains the base application logic.

- node_modules: This directory stores all the external dependencies or libraries required by the application.
 - package-lock.json and package.json: These files are related to Node.js package management. package.json typically contains metadata about the project and its dependencies, while package-lock.json contains information about the exact versions of dependencies installed.
 - public: This directory might contain static assets (e.g., images, stylesheets) served by the application.
 - views: This directory likely contains view templates used by the application, if it's a web application.
3. How it was Installed: The application was likely installed using Node.js package management tools such as npm (Node Package Manager) or yarn. These tools would have been used to install the dependencies listed in package.json from the npm registry or other sources.
 4. Prerequisite Libraries: The application requires external libraries or modules, as indicated by the presence of the node_modules directory. These dependencies are listed in the package.json file.
 5. Data Storage: Based on the provided directory structure, it's not immediately clear where the application stores its data. However, it's possible that the app directory might contain modules responsible for interacting with a database or other data storage systems. Further examination of the application's code would be needed to determine exactly where and how data is stored.



The screenshot shows a terminal window within the AWS CloudWatch interface. The terminal output includes:

```

aws | Services | Search [Alt+S] | N. Virginia | voclabs/user3223197=Gupta_Nancy @ 0824-5190-8674
299 updates can be installed immediately.
211 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: Sat Apr 27 01:30:47 2024 from 18.206.107.27
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-16-10-141:~$ sudo lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 447 root 19u IPv6 21100 0t0 TCP *:http (LISTEN)
ubuntu@ip-10-16-10-141:~$ ^C
ubuntu@ip-10-16-10-141:~$ ps -ef | head -1; ps -ef | grep node
UID PID PPID C STIME TTY TIME CMD
root 428 425 0 03:08 ? 00:00:00 sudo node index.js
root 447 428 0 03:08 ? 00:00:00 node index.js
ubuntu 1200 1163 0 03:13 pts/0 00:00:00 grep --color=auto node
ubuntu@ip-10-16-10-141:~$ cd ~/resources/codebase_partner
ubuntu@ip-10-16-10-141:~/resources/codebase_partner$ ls
app index.js node_modules package-lock.json package.json public views
ubuntu@ip-10-16-10-141:~/resources/codebase_partner$ 

i-0a1861b7fa6d1ac29 (MonolithicAppServer)
Public IPs: 52.207.230.237 Private IPs: 10.16.10.141

```

4. Connect a MySQL client to the RDS database that the node application stores data in.

aws Services RDS

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

Search results for 'RDS'

Services (12) See all 12 results ▾

RDS ☆ Managed Relational Database Service

Database Migration Service ☆ Managed Database Migration Service

Kinesis ☆ Work with Real-Time Streaming Data

Amazon OpenSearch Service ☆ Run open-source OpenSearch or Elasticsearch using Managed Clusters or Serverless ...

Features (38) See all 38 results ▾

Reserved instances

RDS feature

Explore Zones

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

Default VPC [Details](#) vpc-000e564836083ddda8

Encryption protection and security

Serial Console

Credit specification

Experiments

Explore AWS

Up to 40% Better Price Performance

Instances deliver the best price performance for durable general purpose workloads in the cloud on EC2. [Learn more](#)

Things You Can Do Today to Reduce AWS Costs

Learn how to effectively manage your AWS costs without compromising on performance or capacity.

aws Services Search

Amazon RDS

Dashboard

Databases

Query Editor

Performance insights

Snapshots

Exports in Amazon S3

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Zero-ETL integrations [New](#)

Events

Event subscriptions

CloudShell Feedback

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.

Resources Refresh

You are using the following Amazon RDS resources in the US East (N. Virginia) region (used/quota)

DB Instances (1/40) Parameter groups (1)

- Allocated storage (0.02 TB/100 TB)
- Increase DB instances limit [Details](#)

DB Clusters (0/40) Option groups (1)

- Reserved instances (0/40)
- Snapshots (0)
- Manual
- DB Cluster (0/100)
- DB Instance (0/100)
- Automated
- DB Cluster (0)
- DB Instance (0)

Recent events (5) Subnet groups (1/50) Supported platforms [VPC](#)

Event subscriptions (0/20) Default network vpc-000e564836083ddda8

Recommended services [More](#)

No recommendations yet Recommended services will display based on your AWS console usage.

Recommended for you

Time-Series Tables in PostgreSQL Step-by-step guide to design high-performance time series data tables on Amazon RDS for PostgreSQL. [Learn more](#)

Test Your DR Strategy in Minutes Amazon Aurora Global Database now supports

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

aws Services Search

Amazon RDS

RDS > Databases

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

Databases (1) Group resources [C](#) Modify Actions ▾ Restore from S3 Create database

Filter by databases

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

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

The screenshot shows the AWS RDS console with the database 'supplierdb' selected. The left sidebar has 'Databases' selected. The main area shows the 'Summary' tab for 'supplierdb'. Key details include:

DB identifier	Status	Role	Engine	Recommendations
supplierdb	Available	Instance	MySQL Community	5 Informational
CPU	Class	Current activity	Region & AZ	
2.54%	db.t3.micro	0 Connections	us-east-1b	

The 'Connectivity & security' tab is active, showing the endpoint information:

Endpoint	Availability Zone	VPC security groups
supplierdb.cjoadg7hycv.us-east-1.rds.amazonaws.com	us-east-1b	DBSecurityGroup (sg-012f4abb21b3b3d8)

Copy endpoint button: `Copy endpoint`

- Find and copy the endpoint of the RDS database that is running in the lab environment.

The screenshot shows the AWS RDS console with the database 'supplierdb' selected. The left sidebar has 'Databases' selected. The main area shows the 'Summary' tab for 'supplierdb'. The 'Connectivity & security' tab is active, showing the endpoint information. A tooltip 'Copy endpoint' is shown over the endpoint URL.

Endpoint	Availability Zone	VPC security groups
supplierdb.cjoadg7hycv.us-east-1.rds.amazonaws.com	us-east-1b	DBSecurityGroup (sg-012f4abb21b3b3d8)

Copy endpoint button: `Copy endpoint`

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

```

aws Services Search [Alt+S] N. Virginia voclabs/user3223197=Gupta_Nancy @ 0824-5190-8674
app index.js node_modules package-lock.json package.json public views
ubuntu@ip-10-16-10-141:~/resources/codebase_partner$ nmap -Pn supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-27 03:24 UTC
Nmap scan report for supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com (10.16.40.129)
Host is up (0.0013s latency).
rDNS record for 10.16.40.129: ip-10-16-40-129.ec2.internal
Not shown: 999 filtered ports
PORT      STATE SERVICE
3306/tcp   open  mysql

Nmap done: 1 IP address (1 host up) scanned in 6.56 seconds
ubuntu@ip-10-16-10-141:~/resources/codebase_partner$ mysql -h supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com -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> 
```

i-0a1861b7fa6d1ac29 (MonolithicAppServer)
PublicIPs: 52.207.230.237 PrivateIPs: 10.16.10.141

- To connect to the database, use the MySQL client that is already installed on the **MonolithicAppServer** instance. Use the following values:

- Username: admin
- Password: lab-password

Use the mysql command-line client to connect to the RDS database using the following syntax:

```
mysql -h supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com -u admin -p
```

Then enter the password as written above

```

aws Services Search [Alt+S] N. Virginia voclabs/user3223197=Gupta_Nancy @ 0824-5190-8674
app index.js node_modules package-lock.json package.json public views
ubuntu@ip-10-16-10-141:~/resources/codebase_partner$ nmap -Pn supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-27 03:24 UTC
Nmap scan report for supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com (10.16.40.129)
Host is up (0.0013s latency).
rDNS record for 10.16.40.129: ip-10-16-40-129.ec2.internal
Not shown: 999 filtered ports
PORT      STATE SERVICE
3306/tcp   open  mysql

Nmap done: 1 IP address (1 host up) scanned in 6.56 seconds
ubuntu@ip-10-16-10-141:~/resources/codebase_partner$ mysql -h supplierdb.cjoadg7hycvt.us-east-1.rds.amazonaws.com -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> 
```

i-0a1861b7fa6d1ac29 (MonolithicAppServer)
PublicIPs: 52.207.230.237 PrivateIPs: 10.16.10.141

5. Observe the data in the database.

- From the mysql> prompt, run SQL commands as appropriate to see that a database named **COFFEE** contains a table named **suppliers**.

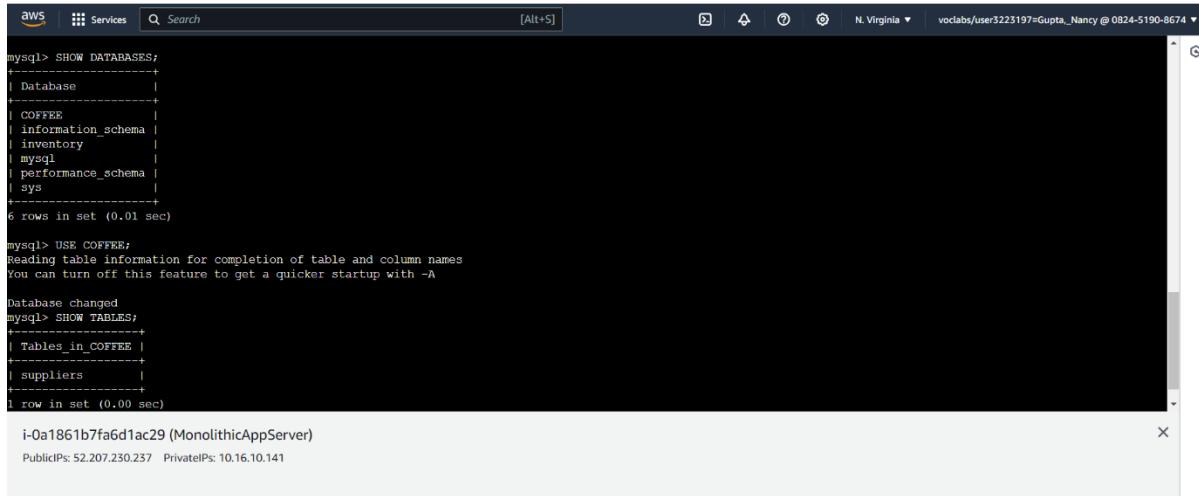
This table contains the supplier entry or entries that you added earlier when you tested the web application.

Use the following SQL commands:

```
SHOW DATABASES;
```

```
USE COFFEE;
```

```
SHOW TABLES;
```



```
aws | Services | Search [Alt+S] | N. Virginia | vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| COFFEE   |
| information_schema |
| inventory |
| mysql    |
| performance_schema |
| sys      |
+-----+
6 rows in set (0.01 sec)

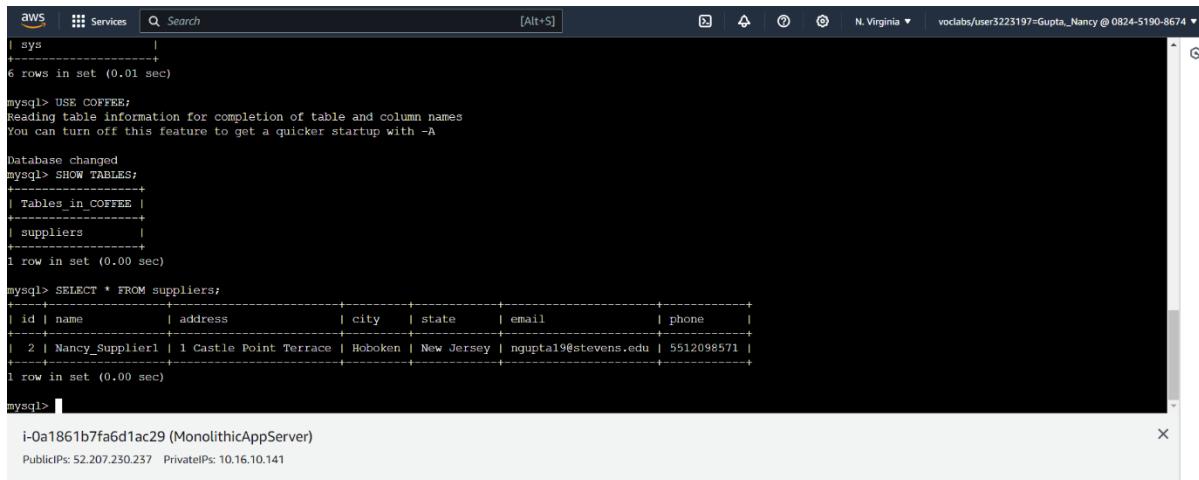
mysql> USE COFFEE;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_COFFEE |
+-----+
| suppliers        |
+-----+
1 row in set (0.00 sec)

i-0a1861b7fa6d1ac29 (MonolithicAppServer)
PublicIPs: 52.207.230.237 PrivateIPs: 10.16.10.141
```

If the COFFEE database and suppliers table exist, you can view the data in the suppliers table.
Use the following SQL command:

```
SELECT * FROM suppliers;
```



```
aws | Services | Search [Alt+S] | N. Virginia | vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| sys      |
+-----+
6 rows in set (0.01 sec)

mysql> USE COFFEE;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_COFFEE |
+-----+
| suppliers        |
+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM suppliers;
+----+-----+-----+-----+-----+-----+
| id | name          | address       | city        | state       | email        | phone        |
+----+-----+-----+-----+-----+-----+
| 2  | Nancy_Supplier1 | 1 Castle Point Terrace | Hoboken | New Jersey | ngupta19@stevens.edu | 5512098571 |
+----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> |
i-0a1861b7fa6d1ac29 (MonolithicAppServer)
PublicIPs: 52.207.230.237 PrivateIPs: 10.16.10.141
```

- **Exit the MySQL client and then close the EC2 Instance Connect tab. Also close the coffee suppliers web application tab.**

To exit the MySQL client, simply type exit or \q and press Enter.

```

aws | Services | Search [Alt+S] | N. Virginia | vclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 |
+-----+
| performance_schema |
| sys |
+-----+
6 rows in set (0.00 sec)

mysql> USE COFFEE;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_COFFEE |
+-----+
| suppliers |
+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM suppliers;
+----+-----+-----+-----+-----+-----+
| id | name | address | city | state | email | phone |
+----+-----+-----+-----+-----+-----+
| 2 | Nancy_Supplier1 | 1 Castle Point Terrace | Hoboken | New Jersey | ngupta9@stevens.edu | 5512098571 |
+----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> exit
Bye
ubuntu@ip-10-16-10-141:~/resources/codebase_partner$ 

```

i-0a1861b7fa6d1ac29 (MonolithicAppServer)
Public IPs: 52.207.230.237 Private IPs: 10.16.10.141

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

In this phase, you will create a development environment by using AWS Cloud9. You will also check your application code into AWS CodeCommit, which is a Git-compatible repository.

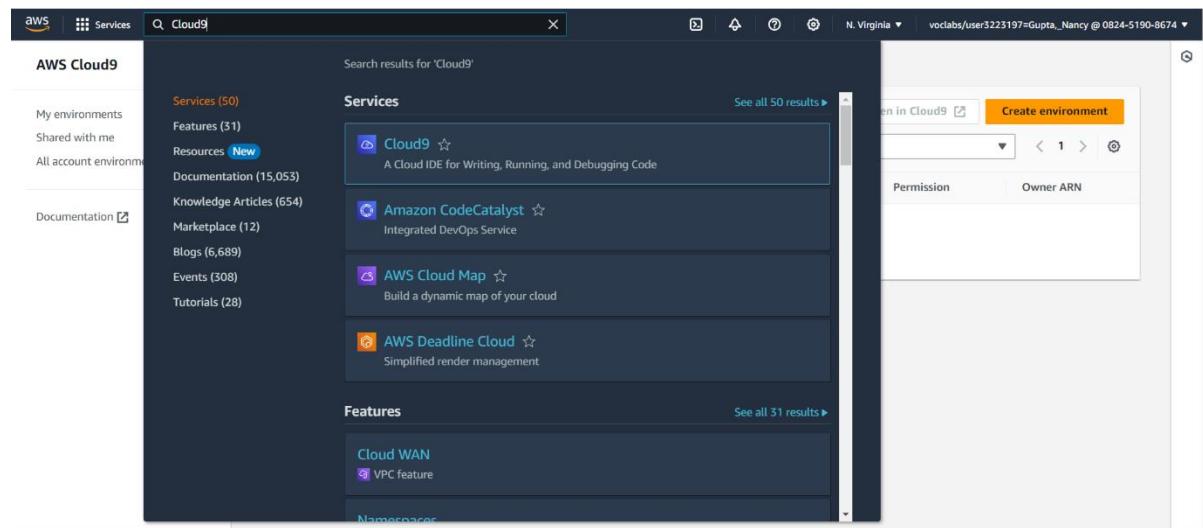
Task 3.1: Create an AWS Cloud9 IDE as your work environment

In this task, you will create your development environment. If you aren't familiar with AWS Cloud9, the following references might be helpful:

- [Creating an Environment in AWS Cloud9](#)
- [Tour of the AWS Cloud9 IDE](#)

1. Create an AWS Cloud9 instance that is named MicroservicesIDE and then open the IDE.

It should run as a new EC2 instance of size *t3.small* and run Amazon Linux 2. The instance should support SSH connections and run in the *LabVPC* in *Public Subnet1*.



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 AWS Cloud9 environments, each one containing the custom tools, runtimes, and files.

New AWS Cloud9 environment

Create environment

Getting started

Before you start (2 min read)

Create an environment (2 min read)

Working with environments (15 min read)

Working with the IDE (10 min read)

Working with AWS Lambda (5 min read)

AWS Cloud9 > Environments > Create environment

Create environment

Details

Name

MicroservicesIDE

Description - optional

Environment type

New EC2 instance

Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.

Existing compute

You have an existing instance or server that you'd like to use.

New EC2 instance

New EC2 instance

Instance type

t2.micro (1 GiB RAM + 1 vCPU)

Free-tier eligible. Ideal for educational users and exploration.

t3.small (2 GiB RAM + 2 vCPU)

Recommended for small web projects.

m5.large (8 GiB RAM + 2 vCPU)

Recommended for production and most general-purpose development.

Additional instance types

Explore additional instances to fit your need.

Platform

Amazon Linux 2023

Timeout

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

30 minutes

Network settings [Info](#)

Connection
How your environment is accessed.

AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).

Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.

VPC settings [Info](#)

Amazon Virtual Private Cloud (VPC)
The VPC that your environment will access. To allow the AWS Cloud9 environment to connect to its EC2 instance, attach an internet gateway (IGW) to your VPC. [Create new VPC](#)

vpc-043e6f0ce741fda03
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-09458d72d02ae53ea
Name - Public Subnet

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.
- You must attach an internet gateway to the Amazon VPC so the SSM Agent for the instance can connect to Systems Manager.
- You must ensure that the public subnet has a route table with a minimum set of routes. [Learn more](#)

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

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-09458d72d02ae53ea
Name - Public Subnet

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.
- You must attach an internet gateway to the Amazon VPC so the SSM Agent for the instance can connect to Systems Manager.
- You must ensure that the public subnet has a route table with a minimum set of routes. [Learn more](#)

Tags - optional [Info](#)
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

The following IAM resources will be created in your account

- AWSServiceRoleForAWSCloud9** - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf.
You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. [Learn more](#)

[Cancel](#) [Create](#)

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

AWS Cloud9 [Documentation](#)

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

AWS Cloud9 > Environments

Environments (1) [Delete](#) [View details](#) [Open in Cloud9](#) [Create environment](#)

Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
MicroservicesIDE	Open	EC2 instance	Secure Shell (SSH)	Owner	arn:aws:sts::082451908674:assumed-role/vocabs/user3223197=Gupta,_Nancy

The screenshot shows the AWS Cloud9 interface. On the left, a sidebar lists 'My environments', 'Shared with me', and 'All account environments'. A central panel displays 'Environments (1)' with a table showing one environment named 'MicroservicesIDE'. The main area shows the AWS Cloud9 IDE interface with tabs for 'Welcome' and 'Developer Tools'. The 'Welcome' tab displays the 'AWS Cloud9' logo and the message 'Welcome to your development environment'. Below this, a 'Toolkit for AWS Cloud9' section provides instructions for using the toolkit. A terminal window at the bottom shows a bash session with the command 'veclabs:~/environment \$'. The top right corner of the main window shows the user's session details: 'N. Virginia' and 'veclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674'.

Task 3.2: Copy the application code to your IDE

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.

The screenshot shows the AWS Lab Instructions page for 'Building Microservices and a CI/CD Pipeline with AWS'. The left sidebar includes links for Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main content area shows the 'AWS Details' panel with the following information:

- AWS Details**: Shows session usage: 'Used \$0 of \$40', '11:08', and buttons for 'Start Lab', 'End Lab', 'AWS Details', 'Details', and 'Reset'.
- AWS Labs**: Shows session details: 'Remaining session time: 11:07:22(668 minutes)', 'Session started at: 2024-04-26T20:08:19-0700', 'Session to end: 2024-04-27T08:08:19-0700', 'Accumulated lab time: 07:37:00 (457 minutes)', and 'ips -- public:52.207.230.237, private:10.16.10.141'.
- AWS CLI**: Buttons for 'Show' and 'Download PEM'.
- AWS SSO**: Buttons for 'Download PPK' and 'Download LIFD'.
- RDS Endpoint**: Shows 'supplierdb.cjoadg7hycv.us-east-1.rds.amazonaws.com'.

The central content area contains the task instructions:

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.

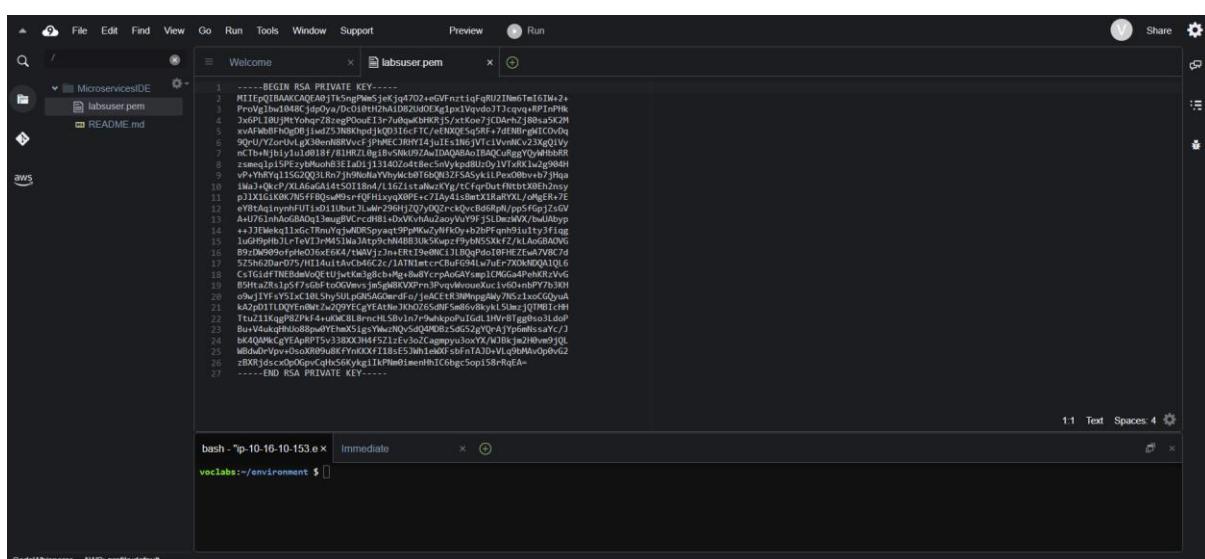
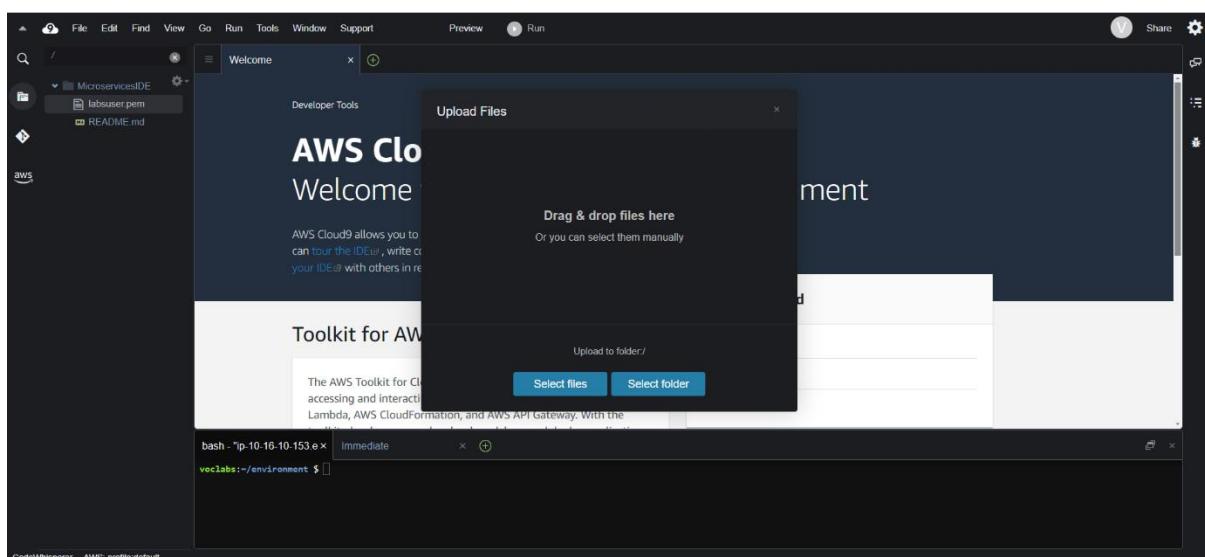
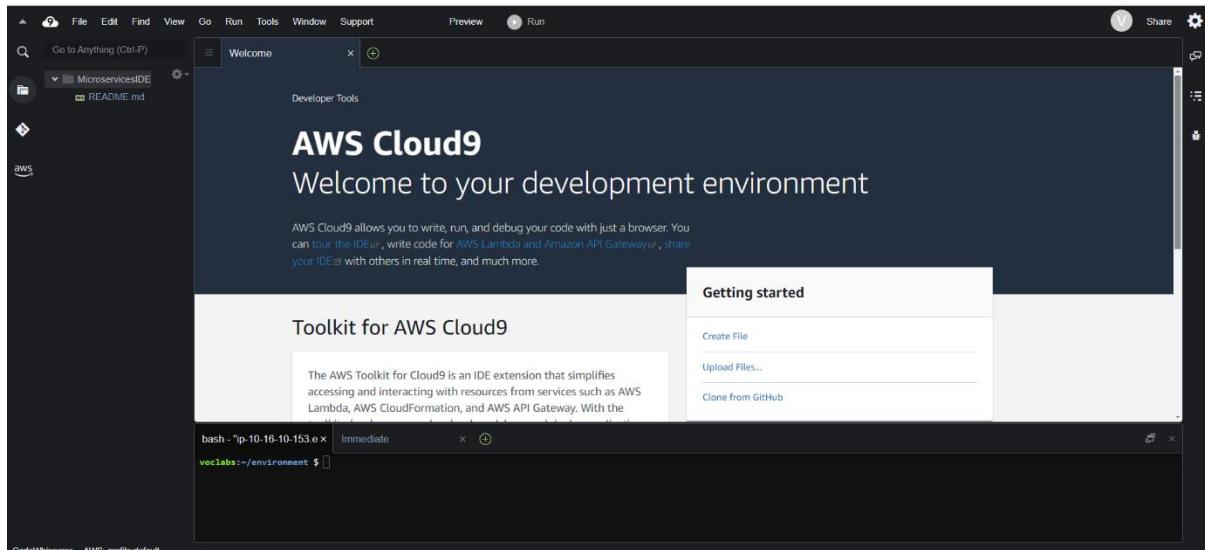
- From the **AWS Details** panel on this lab instructions page, download the `labsuser.pem` file to your local computer.
- Upload the `.pem` file to your AWS Cloud9 IDE, and use the Linux `chmod` command to set the [proper permissions](#) on the file so that you can use it to connect to an EC2 instance.
- Create a temp directory on the AWS Cloud9 instance at `/home/ec2-user/environment/temp`.
- From the Amazon EC2 console, retrieve the private IPv4 address of the

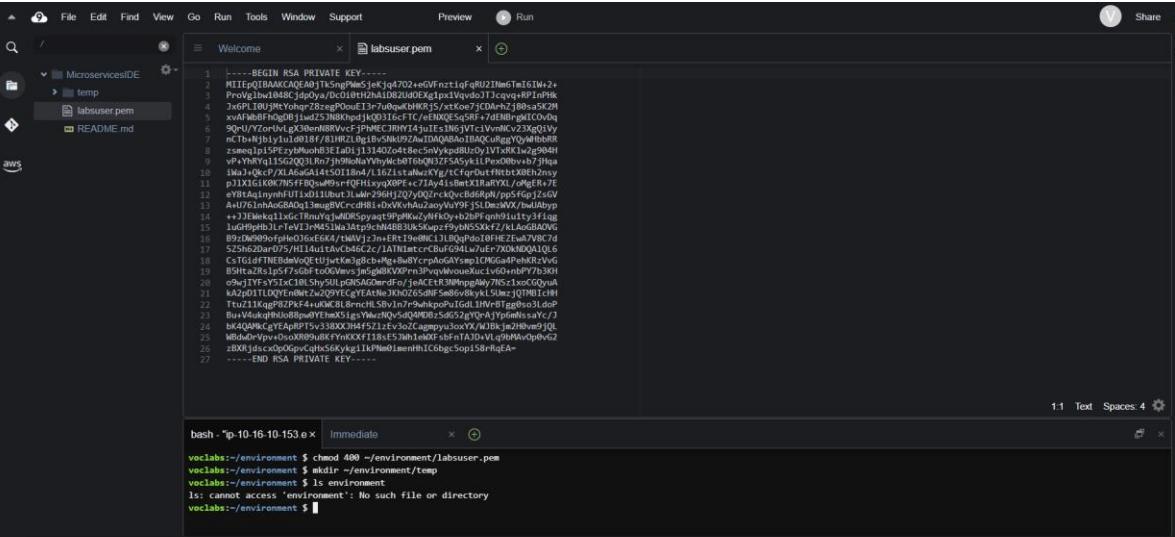
The screenshot shows the AWS Cloud9 interface. On the left, a sidebar lists 'My environments', 'Shared with me', and 'All account environments'. Below that is a 'Documentation' link. The main content area is titled 'Environments (1)' and shows a table with one row. The row for 'MicroservicesIDE' includes a blue circular icon, the name 'MicroservicesIDE', a 'Cloud9 IDE' button, 'EC2 instance' under 'Environment type', 'Secure Shell (SSH)' under 'Connection', 'Owner' under 'Permission', and the ARN 'arn:aws:sts::082451908674:assumed-role/voclabs/user3223197=Gupta,_Nancy' under 'Owner ARN'. Buttons for 'Delete', 'View details', 'Open in Cloud9', and 'Create environment' are at the top right.

This screenshot shows the 'MicroservicesIDE' details page. The sidebar on the left is identical to the first screenshot. The main content area has a 'Details' section with tabs for 'Edit', 'EC2 instance', 'Network settings', and 'Tags'. The 'EC2 instance' tab is selected, showing an ARN 'arn:aws:cloud9:us-east-1:082451908674:environment:84d2e6fefedab4a0b8214113159ea5acf' and an 'Instance type' 't3.small (2 GiB RAM + 2 vCPU)'. Other details include 'Owner ARN' (same as above), 'Status' 'Ready', 'Lifecycle status' 'Created', and 'Number of members' 1. A 'Manage EC2 instance' button is at the bottom right.

This screenshot shows the same 'MicroservicesIDE' details page as the previous one, but with a different ARN: 'arn:aws:cloud9:us-east-1:1082451908674:environment:84d2e6fefedab4a0b8214113159ea5acf'. The rest of the information is identical, including the EC2 instance details and the 'Manage EC2 instance' button.

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.





The screenshot shows a terminal window with the following session:

```
bash -c "ip=10-16-10-153 x environment" Immediate
vocabs:=/environment $ chmod 400 ~/environment/labuser.pem
vocabs:=/environment $ mkdir -p ~/environment/temp
vocabs:=/environment $ environment
ls: cannot access 'environment': No such file or directory
vocabs:=/environment $
```

At the top of the terminal, there is a navigation bar with tabs for 'File', 'Edit', 'Find', 'View', 'Go', 'Run', 'Tools', 'Window', 'Support', 'Preview', and 'Run'. The 'Run' tab is currently selected. On the left side, there is a sidebar with icons for search, file system, and AWS services like Lambda and CloudWatch.

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.

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
MonolithicApp...	i-0a1861b7fa6d1ac29	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-52-207-230-
aws-cloud9-Mi...	i-0cf5c889af3b1eb2	Running	t3.small	2/2 checks passed	View alarms +	us-east-1a	ec2-54-88-53-35

i-0a1861b7fa6d1ac29 (MonolithicAppServer)

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

Instance summary [Info](#)

Instance ID	i-0a1861b7fa6d1ac29 (MonolithicAppServer)	Public IPv4 address	52.207.230.237 open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-10-16-10-141.ec2.internal	Private IP DNS name (IPv4 only)	ip-10-16-10-141.ec2.internal
Answer private resource DNS name		Instance type	

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
MonolithicApp...	i-0a1861b7fa6d1ac29	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-52-207-230-
aws-cloud9-Mi...	i-0cf5c889af3b1eb2	Running	t3.small	2/2 checks passed	View alarms +	us-east-1a	ec2-54-88-53-35

i-0a1861b7fa6d1ac29 (MonolithicAppServer)

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

Instance summary [Info](#)

Instance ID	i-0a1861b7fa6d1ac29 (MonolithicAppServer)	Public IPv4 address	52.207.230.237 open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-10-16-10-141.ec2.internal	Private IP DNS name (IPv4 only)	ip-10-16-10-141.ec2.internal
Answer private resource DNS name		Instance type	

- Use the Linux [scp](#) command in the Bash terminal on the AWS Cloud9 instance to copy the source code for the node application from the *MonolithicAppServer* instance to the temp directory that you created on the AWS Cloud9 instance.

The following snippet provides an example scp command:

```
Scp -r -i ~environment/labsuser.pem
ubuntu@$appServerPrivIp:/home/ubuntu/resources/codebase_partner/* ~environment/temp/
```

Here I have replaced appServerPrivIp with Private IPV4 of the MonolithicAppServer

The screenshot shows the AWS Cloud9 IDE interface. In the top navigation bar, the tabs are: File, Edit, Find, View, Go, Run, Tools, Window, Support, Preview, Run. Below the tabs, there's a search bar and a file browser window titled "Welcome" showing the directory structure: MicroservicesIDE, temp, labsuser.pem, README.md. The main workspace shows a terminal window titled "bash - ip-10-16-10-153.x" with the command "vocabs:/environment \$ scp -r -i ~/environment/labsuser.pem ubuntu@10.16.10.141:/home/ubuntu/resources/codebase_partner/* ~/environment/temp/". The terminal output shows the files being copied from the partner codebase to the temporary directory.

This screenshot shows the AWS Cloud9 IDE interface with a different file structure in the file browser. The "temp" directory now contains several files: shifitjs.json, desc.js, sbs-data.js, sbs-data-generated.js, LICENSE, README.md, example.js, package.json, index.js, test.js, .travis.yml, package-lock.json, package.json, jquery-3.6.0.min.js, bootstrap_min.js.map, bootstrap_min.js, base.css, bootstrap_min.css, bootstrap_min.css.map, favicon.ico, espresso.jpg, footer.html, home.html, supplier-add.html, nav.html, 404.html, 500.html, supplier-form-fields.html, supplier-list-all.html, supplier-update.html, header.html, and vocabs:/environment \$. The terminal window shows the same scp command as the previous screenshot, indicating the files have been copied.

6. In the file browser of the IDE, verify that the source files for the application have been copied to the temp directory on the AWS Cloud9 instance.

This screenshot shows the AWS Cloud9 IDE interface with the same file structure as the previous one. The "temp" directory now contains many more files, including shifitjs.json, desc.js, sbs-data.js, sbs-data-generated.js, LICENSE, README.md, example.js, package.json, index.js, test.js, .travis.yml, package-lock.json, package.json, jquery-3.6.0.min.js, bootstrap_min.js.map, bootstrap_min.js, base.css, bootstrap_min.css, bootstrap_min.css.map, favicon.ico, espresso.jpg, footer.html, home.html, supplier-add.html, nav.html, 404.html, 500.html, supplier-form-fields.html, supplier-list-all.html, supplier-update.html, header.html, and vocabs:/environment \$. The terminal window shows the same scp command, indicating the files have been copied.

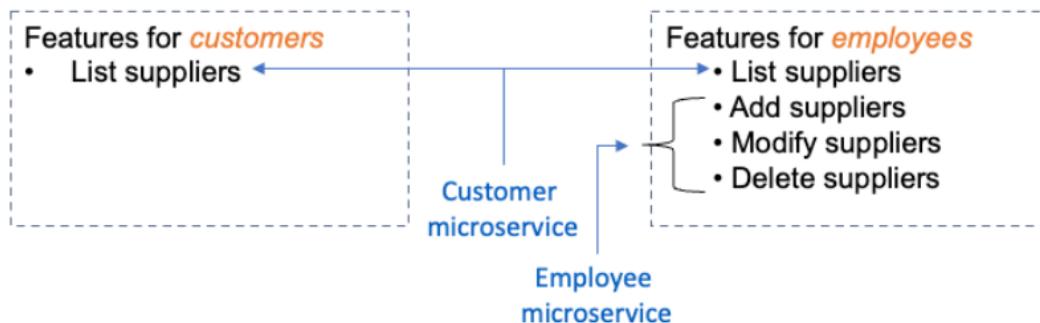
Task 3.3: Create working directories with starter code for the two microservices

In this task, you will create areas in your development environment to support separating the application logic into two different microservices.

Based on the solution requirements of this project, it makes sense to split the monolithic application into two microservices. You will name the microservices *customer* and *employee*.

The following table explains the functionality that is needed for each microservice.

Primary User	Microservice Functionality	Access Level
Customer	The <i>customer</i> microservice will provide the functionality that customers (the café franchise location managers who want to buy coffee beans) need. The customers need a read-only view of the contact information for the suppliers to be able to buy coffee beans from them. You can think of the café franchise location managers as <i>customers</i> of the application.	Read-only
Employee	The <i>employee</i> microservice will provide the functionality that employees (the café corporate office employees) need. Employees need to add, modify, and delete suppliers who are listed in the application. Employees are responsible for keeping the listings accurate and up to date.	Read-write



The employee microservice will eventually be made available only to employees. You will accomplish this by first encapsulating them as a separate microservice (in phases 3 and 4 of the project), and then later in phase 9 of the project, you will limit who can access the employee microservice.

1. In the microservices directory, create two new directories that are named *customer* and *employee*.

```

bash - *ip-16-10-131.e.x* Immediate
100% 397 159.1KB/s 00:00
100% 531 298.8KB/s 00:00
100% 387 156.7KB/s 00:00
100% 503 242.4KB/s 00:00
100% 1917 996.9KB/s 00:00
100% 1851 661.0KB/s 00:00
100% 1838 337.6KB/s 00:00
100% 226 84.1KB/s 00:00

supplier-add.html
nav.html
404.html
500.html
supplier-form-fields.html
supplier-list-all.html
supplier-update.html
header.html
veclabs:-/environment $ cd ~
veclabs:-/environment $ cd ~/environment/
veclabs:-/environment $ mkdir microservices
veclabs:-/environment $ cd microservices
veclabs:-/environment/microservices $ mkdir customer employee
veclabs:-/environment/microservices $ 

```

- Place a copy of the source code for the monolithic application in each new directory, and remove the files from the temp directory.

```

bash - *ip-16-10-131.e.x* Immediate
100% 397 159.1KB/s 00:00
100% 531 298.8KB/s 00:00
100% 387 156.7KB/s 00:00
100% 503 242.4KB/s 00:00
100% 1917 996.9KB/s 00:00
100% 1851 661.0KB/s 00:00
100% 1838 337.6KB/s 00:00
100% 226 84.1KB/s 00:00

veclabs:-/environment $ mkdir microservices
veclabs:-/environment $ cd microservices
veclabs:-/environment/microservices $ mkdir customer employee
veclabs:-/environment/microservices $ cd ~environment/temp
veclabs:-/environment/temp $ cp * ~/environment/microservices/customer/
cp: omitting directory 'app'
cp: omitting directory 'node_modules'
cp: omitting directory 'public'
cp: omitting directory 'views'
veclabs:-/environment/temp $ cp * ~/environment/microservices/customer/
veclabs:-/environment/microservices/customer $ rm -rf *
veclabs:-/environment/microservices/customer $ cp -r ~/environment/temp/* ~/environment/microservices/customer/
veclabs:-/environment/microservices/customer $ cp -r ~/environment/temp/* ~/environment/microservices/employee/
veclabs:-/environment/microservices/customer $ 

```

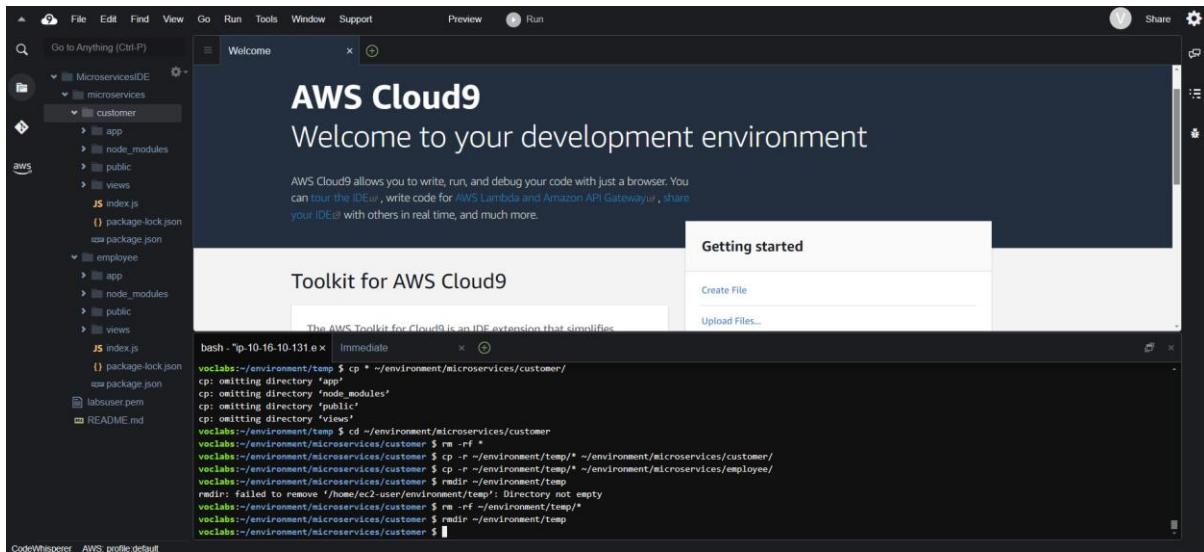
- Delete the empty temp directory.

```

bash - *ip-16-10-131.e.x* Immediate
100% 397 159.1KB/s 00:00
100% 531 298.8KB/s 00:00
100% 387 156.7KB/s 00:00
100% 503 242.4KB/s 00:00
100% 1917 996.9KB/s 00:00
100% 1851 661.0KB/s 00:00
100% 1838 337.6KB/s 00:00
100% 226 84.1KB/s 00:00

veclabs:-/environment/microservices $ cd ~environment/temp
veclabs:-/environment/temp $ cp * ~/environment/microservices/customer/
cp: omitting directory 'app'
cp: omitting directory 'node_modules'
cp: omitting directory 'public'
cp: omitting directory 'views'
veclabs:-/environment/temp $ cd ~/environment/microservices/customer
veclabs:-/environment/microservices/customer $ rm -rf *
veclabs:-/environment/microservices/customer $ cp -r ~/environment/temp/* ~/environment/microservices/customer/
veclabs:-/environment/microservices/customer $ rmdir ~/environment/temp
rmdir: failed to remove '/home/ec2-user/environment/temp': Directory not empty
veclabs:-/environment/microservices/customer $ rm -rf ~/environment/temp/
veclabs:-/environment/microservices/customer $ 

```



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

You now have directories named *customer* and *employee*, and each will contain a microservice. The two microservices will replicate the logic in your monolithic application. However, you will also be able to evolve the application functionality and to time the deployment of feature enhancements for these microservices separately.

You will benefit from checking this source code into a Git repository. In this project, you will use CodeCommit as your Git repository (repo).

1. Create a CodeCommit repository that is named **microservices**.

Screenshot of the AWS CodeCommit Repositories page. The left sidebar shows navigation for CodeCommit, including Source, Artifacts, Build, Deploy, Pipeline, and Settings. The main content area displays a table with columns for Name, Description, Last modified, Clone URL, and AWS KMS Key. A search bar and various actions like Notify, Clone URL, View repository, and Delete repository are at the top. A message indicates "No results" and "There are no results to display."

Screenshot of the "Create repository" wizard. Step 1: Repository settings. It shows a "Repository name" field containing "microservices", a "Description - optional" field, and a "Tags" section with an "Add tag" button. A note says "Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository."

Screenshot of the "Create repository" wizard. Step 2: Additional configuration. It includes an "AWS KMS key" section, a checkbox for "Enable Amazon CodeGuru Reviewer for Java and Python - optional" (unchecked), and a note about creating a service-linked role. At the bottom are "Cancel" and "Create" buttons.

The screenshot shows two views of the AWS CodeCommit console. The top view displays a success message: "Repository successfully created" for a repository named "microservices". It includes connection steps for HTTPS, SSH, and HTTPS (GRC), with a note about using federated access or temporary credentials. The bottom view shows the list of repositories, where "microservices" is listed with its details: Name (microservices), Description (-), Last modified (Just now), Clone URL (HTTPS, SSH, HTTPS (GRC)), and AWS KMS Key (arm:aws:kms:us-east-1:082451908674:key/cb78b9cf-6753-41a6-9e4c-eac54ea3156a).

2. To check the unmodified application code into the microservices CodeCommit repository, run the following commands:

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

Tip: For information about Git commands, see the [Git documentation](#).

Analysis: By running these commands, you first *initialized* the microservices directory to be a Git repository. Then, you created a *branch* in the repository named *dev*. You *added* all files from the microservices directory to the Git repository and *committed* them. Then, you defined