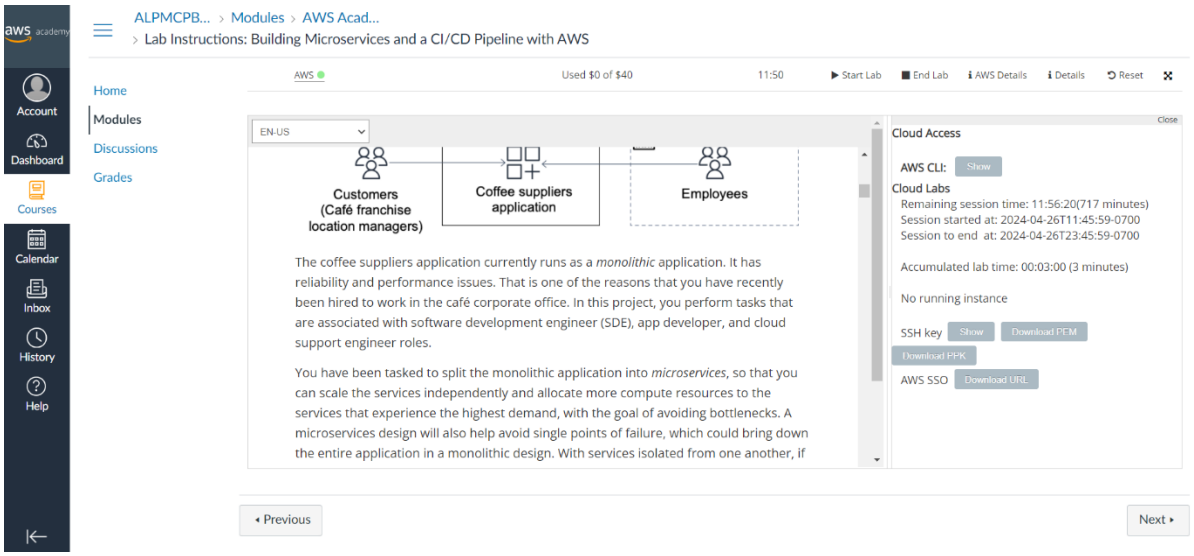
PROJECT- Building Microservices and a CI/CD Pipeline with AWS

Submitted by- Nancy Gupta

Part 1 - Project Execution

HOME SCREEN OF PROJECT



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.

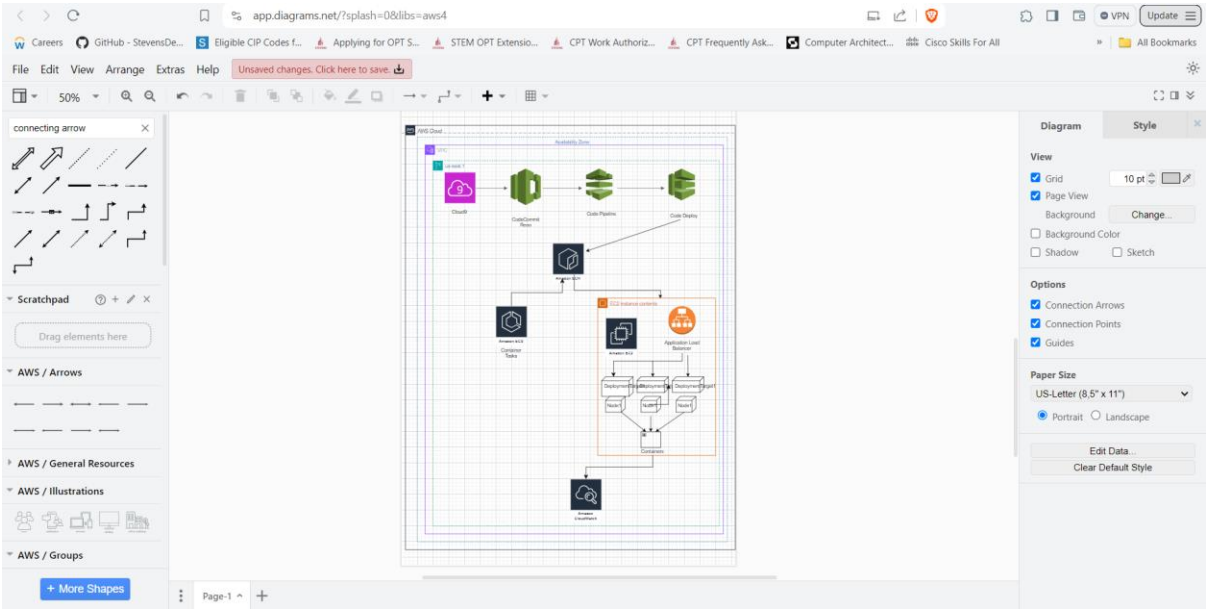
You will also estimate the cost of the proposed solution and present the estimate to your educator. An important first step for any solution is to plan the design and estimate the cost. Review the various components in the architecture to adjust the estimated cost. Cost, along with the features and limitations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

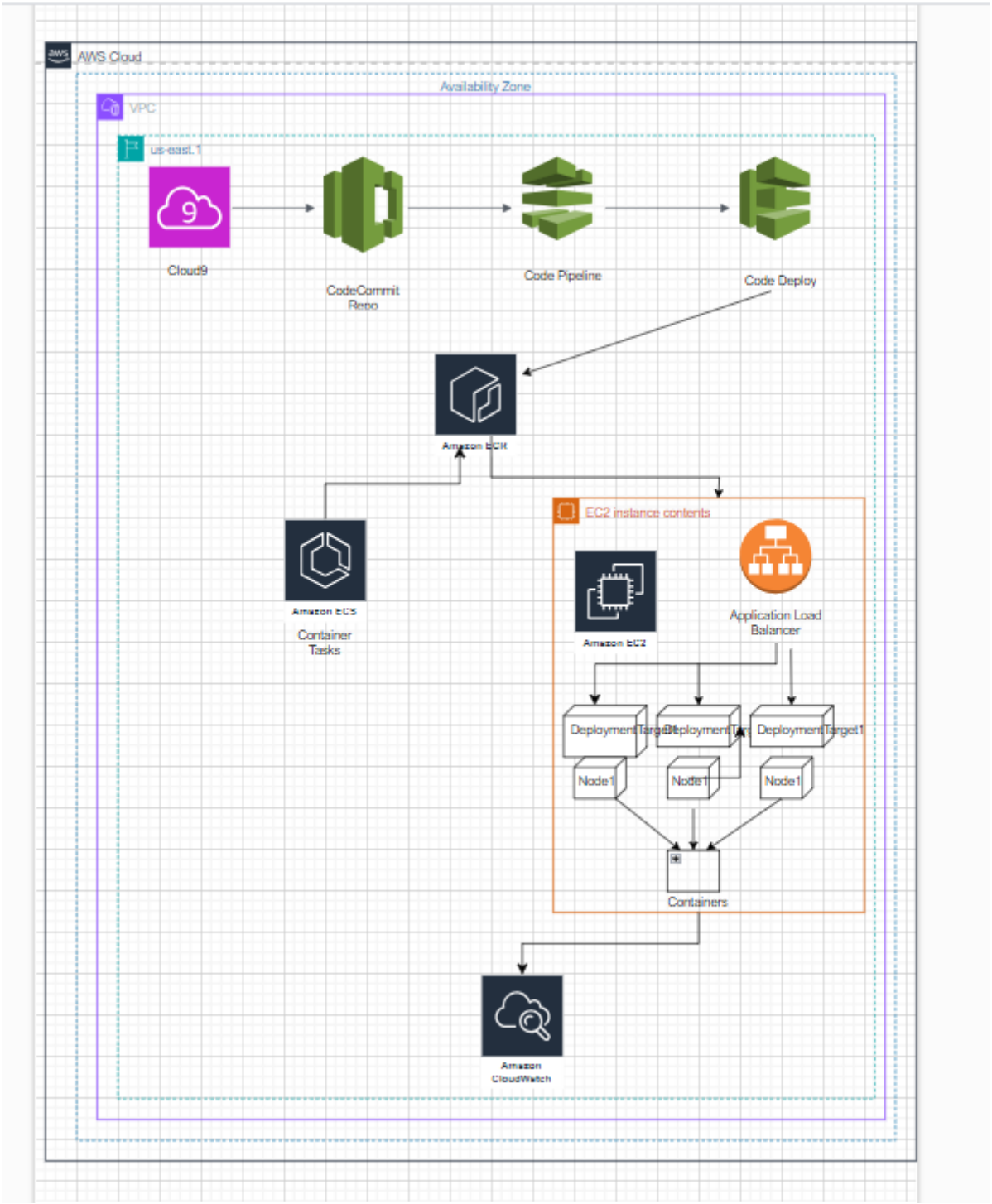
Task 1.1: Create an architecture diagram

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

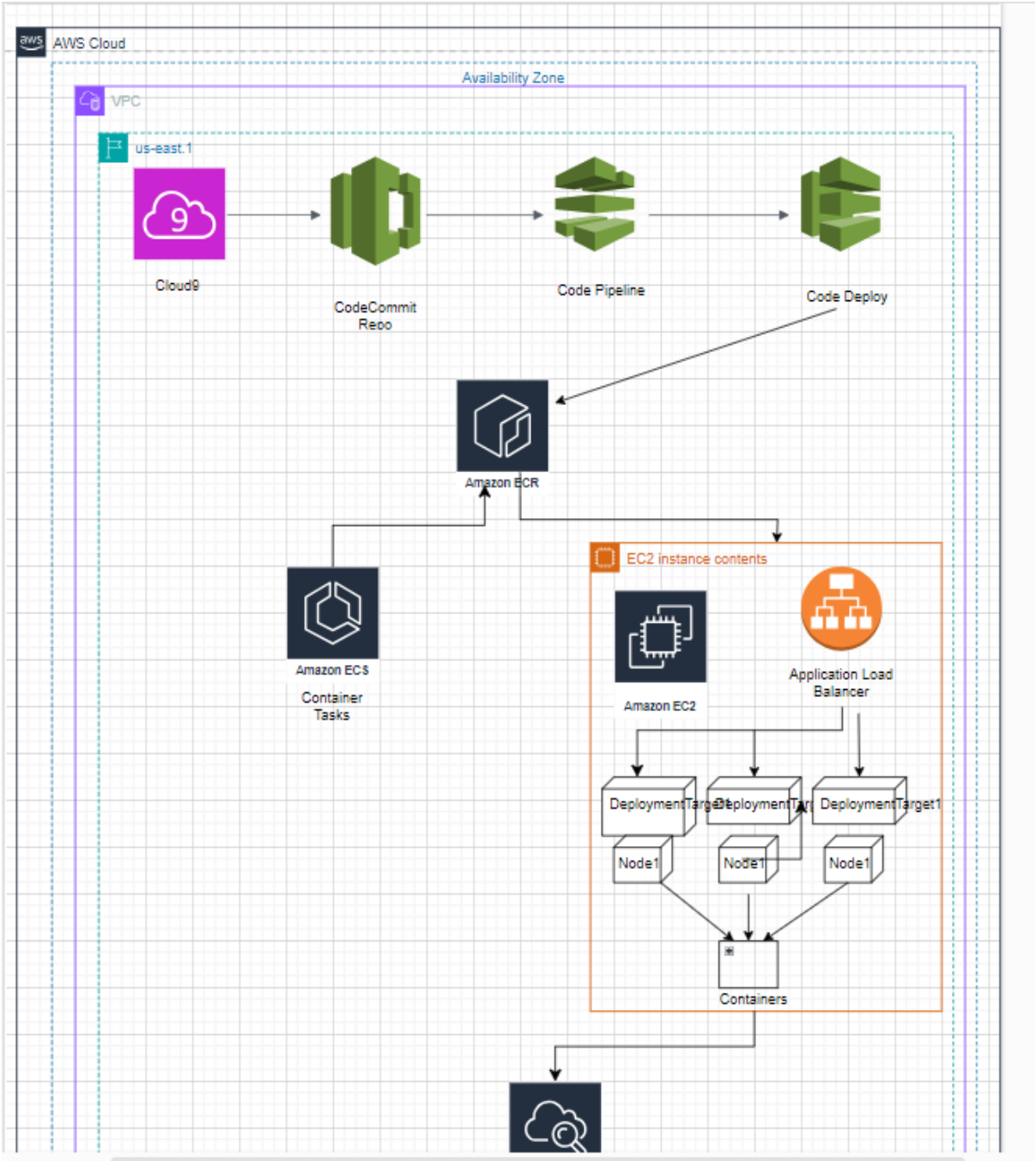
- AWS CodePipeline: Pipeline
- Amazon Elastic Container Service (Amazon ECS): Services, containers, tasks
- Amazon Elastic Container Registry (Amazon ECR): Repository
- AWS Cloud9 environment
- AWS Identity and Access Management (IAM): Roles
- Amazon Relational Database Service (Amazon RDS)
- Amazon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

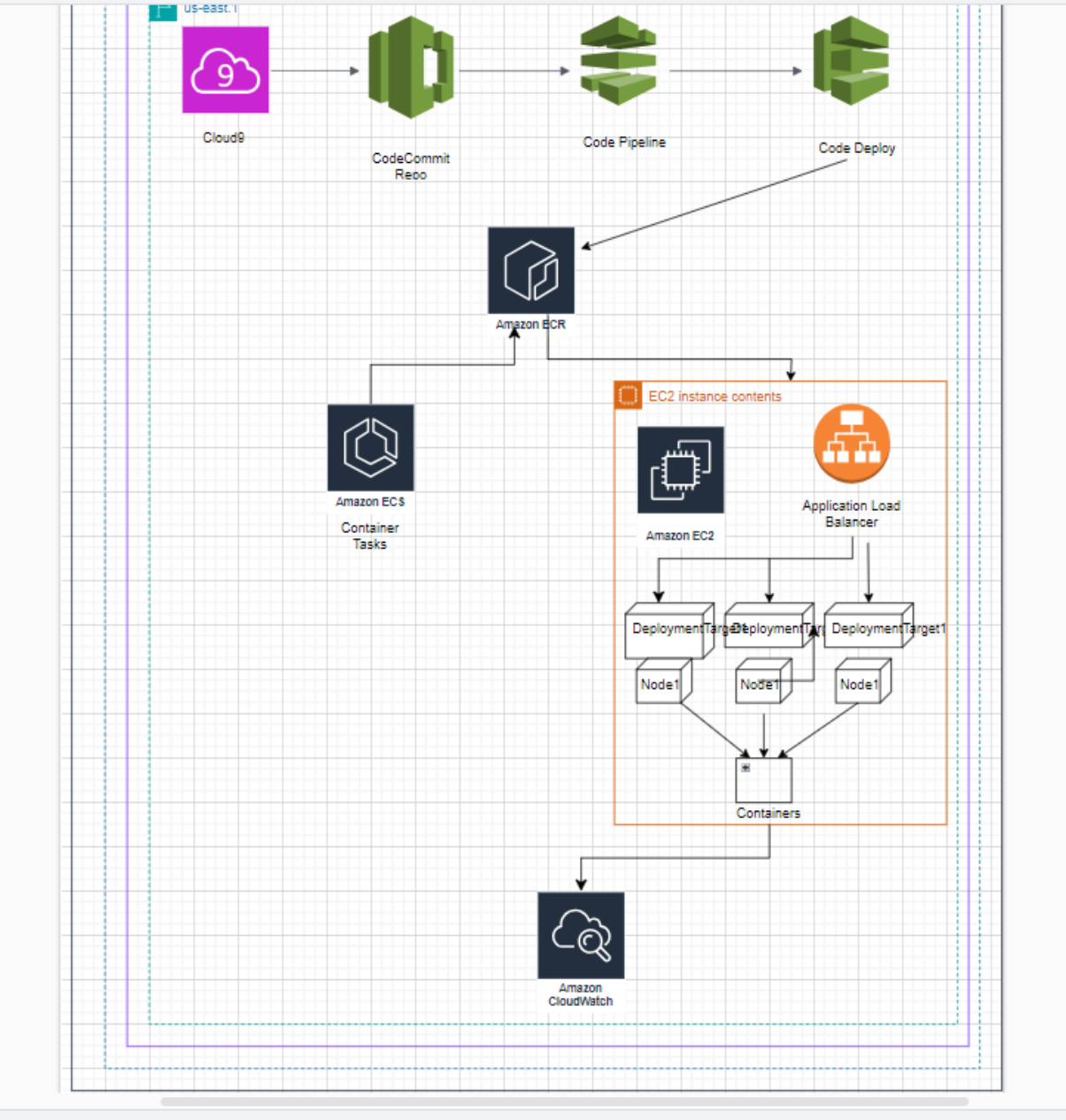
I used **draw.io** to design the AWS architecture diagram.





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


AWS Cloud
Availability Zone
VPC
us-east.1
Cloud9
CodeCommit Repo
Code Pipeline
Code Deploy
Amazon ECR
Amazon ECS
Container Tasks
EC2 instance contents
Amazon EC2
Application Load Balancer
DeploymentTarget1
DeploymentTarget1
DeploymentTarget1
Node1
Node1
Node1
Containers



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:

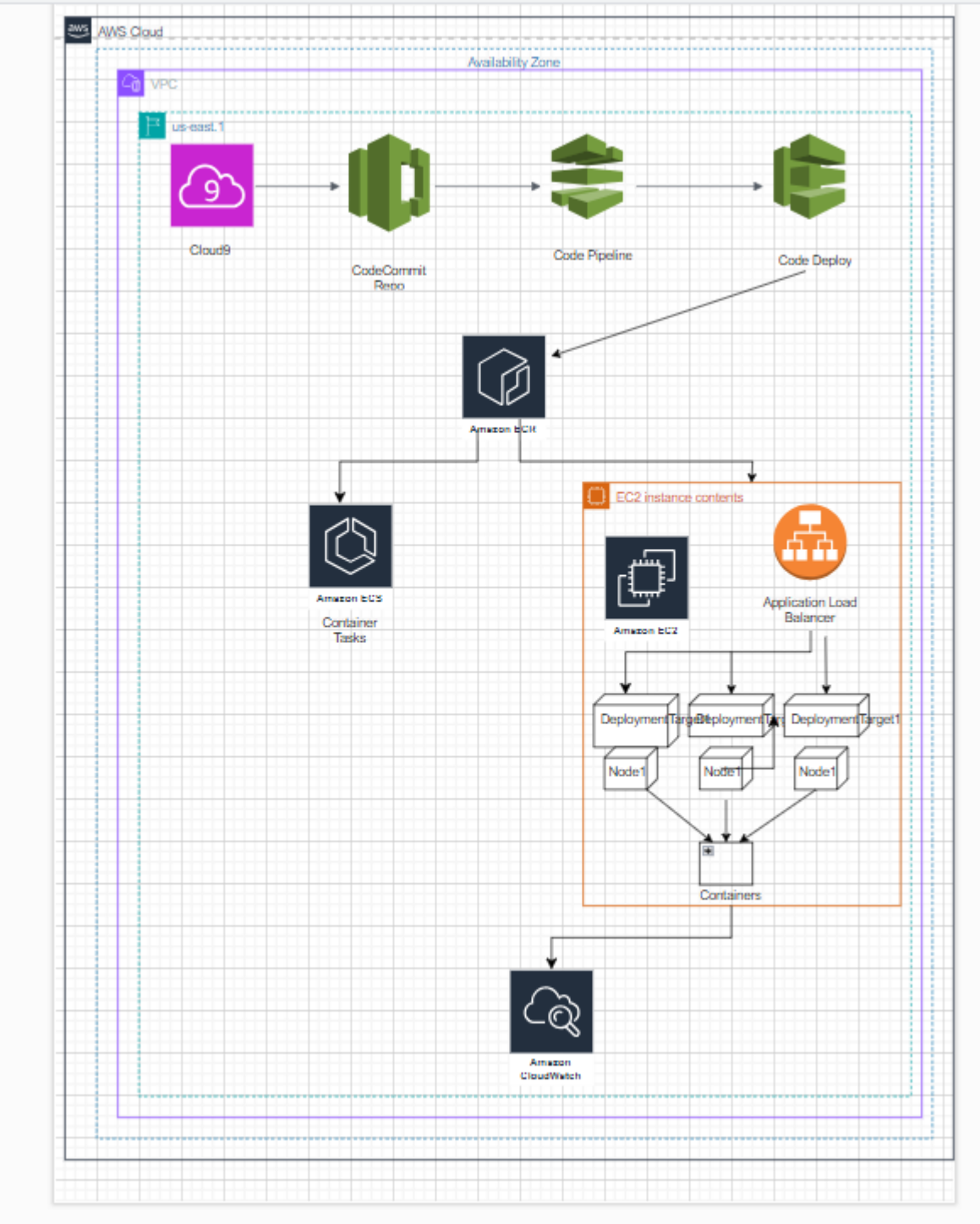
[1] IAM Roles: IAM Roles are used to grant specific permissions and access to AWS services and resources. Different IAM Roles can be created for different components of the architecture.

- 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.

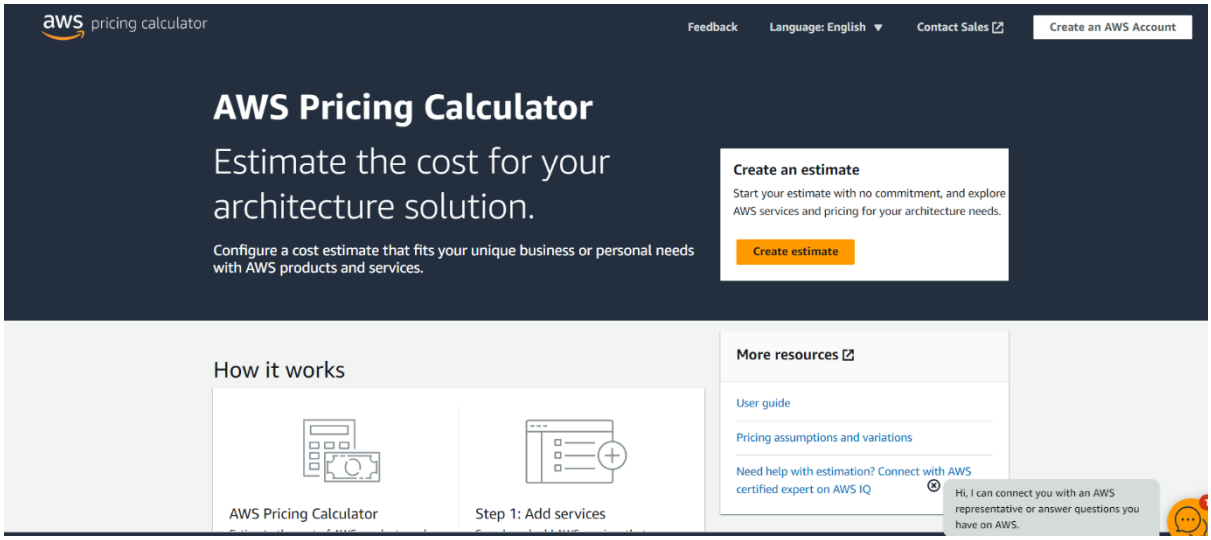
This flow demonstrates how the different AWS services are interconnected and work together to enable a continuous integration, continuous deployment (CI/CD) pipeline, containerized microservices deployment, load balancing, and centralized logging and monitoring.

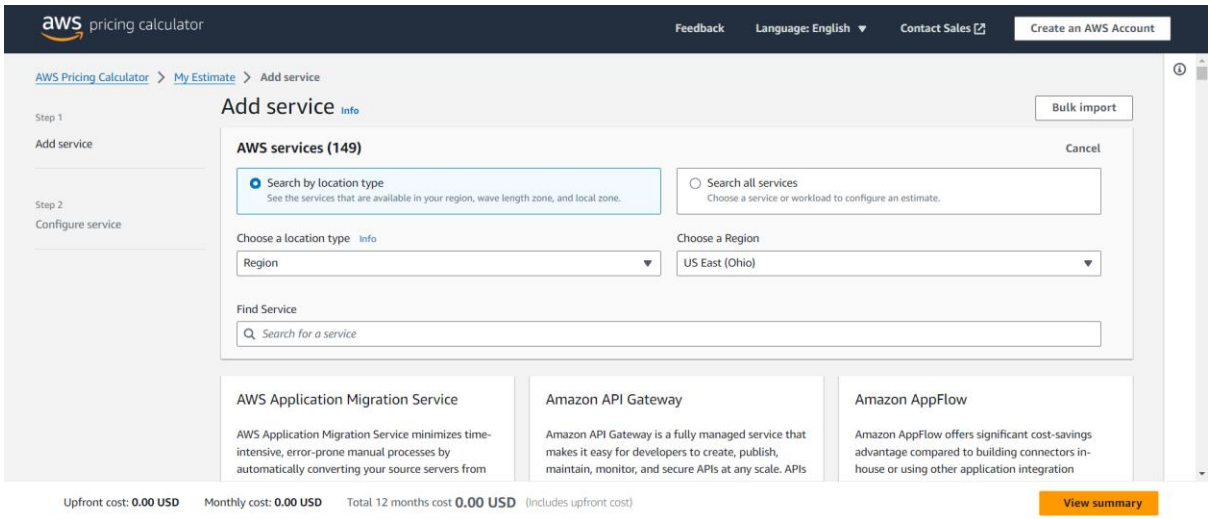


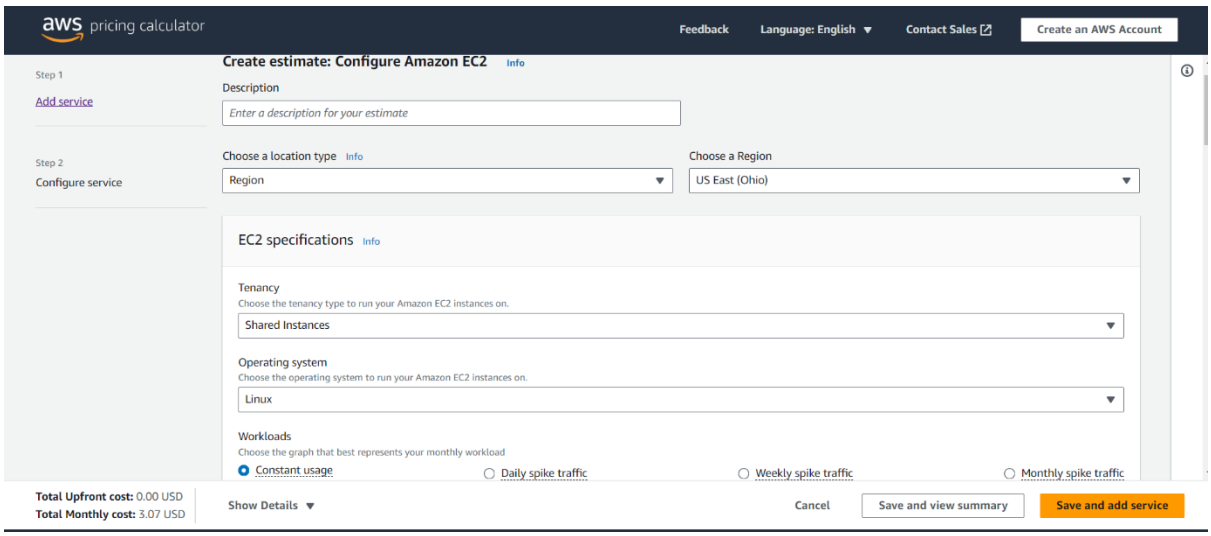
Task 1.2: Develop a cost estimate

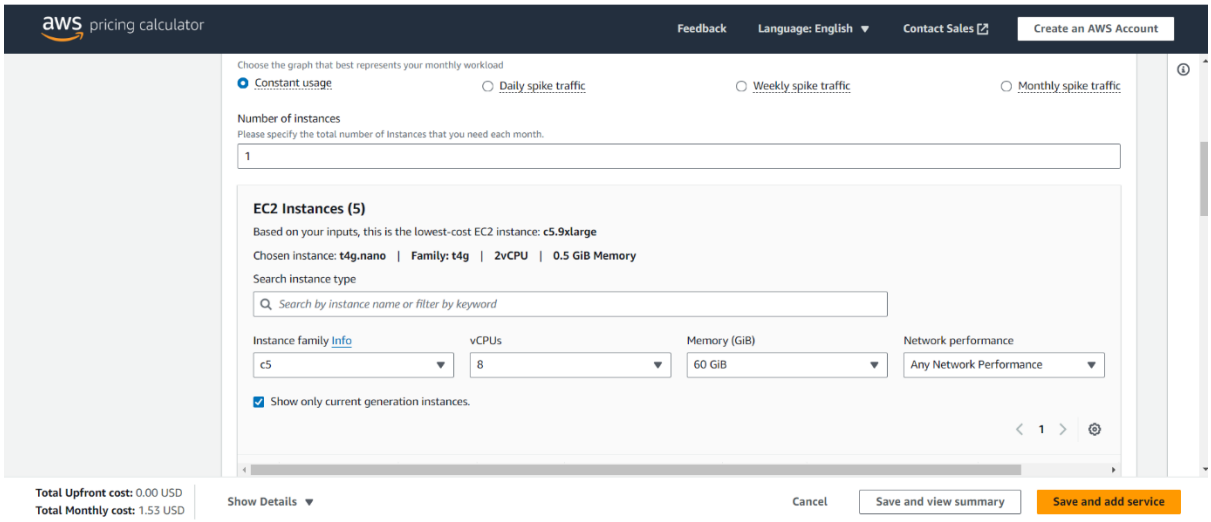
Develop a cost estimate that shows the cost to run the solution that you created an architecture diagram for. Assume that the solution will run in the us-east-1 Region for 12 months. Use the

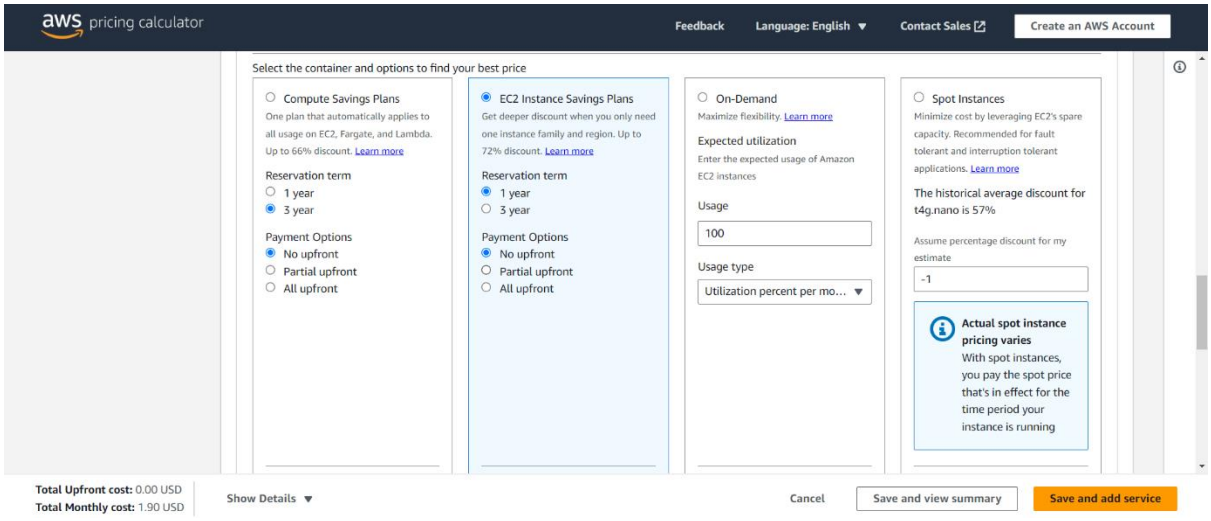
for this estimate.

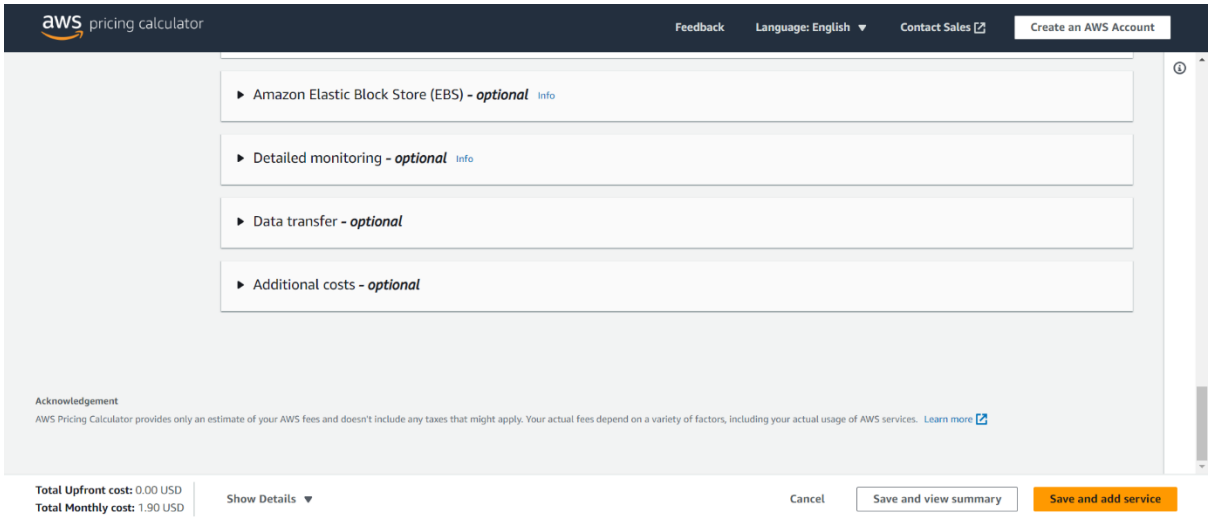

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator
Estimate the cost for your architecture solution.
Configure a cost estimate that fits your unique business or personal needs with AWS products and services.
Create an estimate
Start your estimate with no commitment, and explore AWS services and pricing for your architecture needs.
Create estimate
How it works
AWS Pricing Calculator
Step 1: Add services
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.

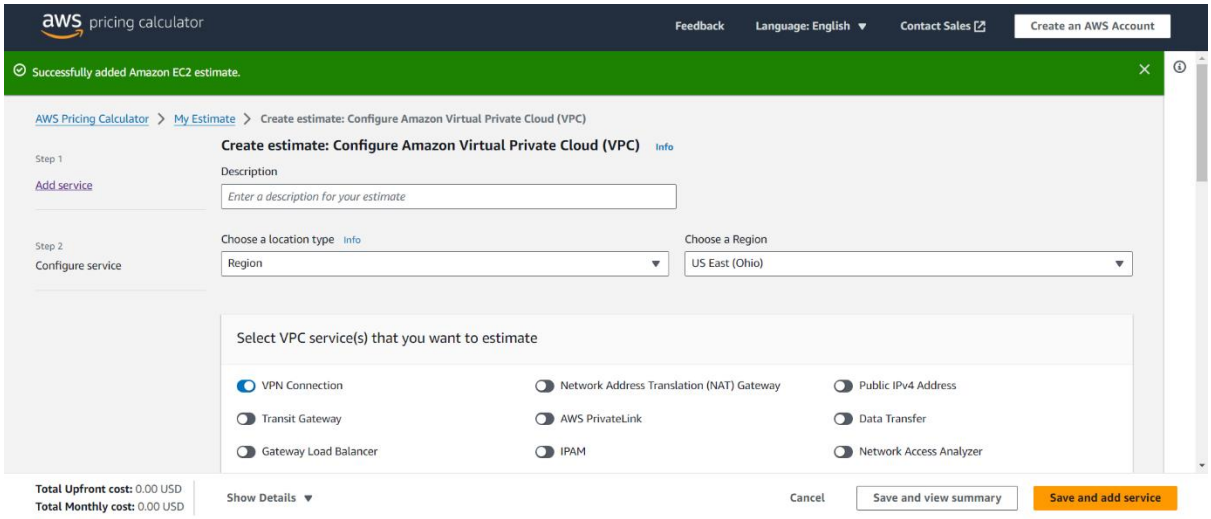

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator > My Estimate > Add service
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (149)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Find Service
Search for a 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

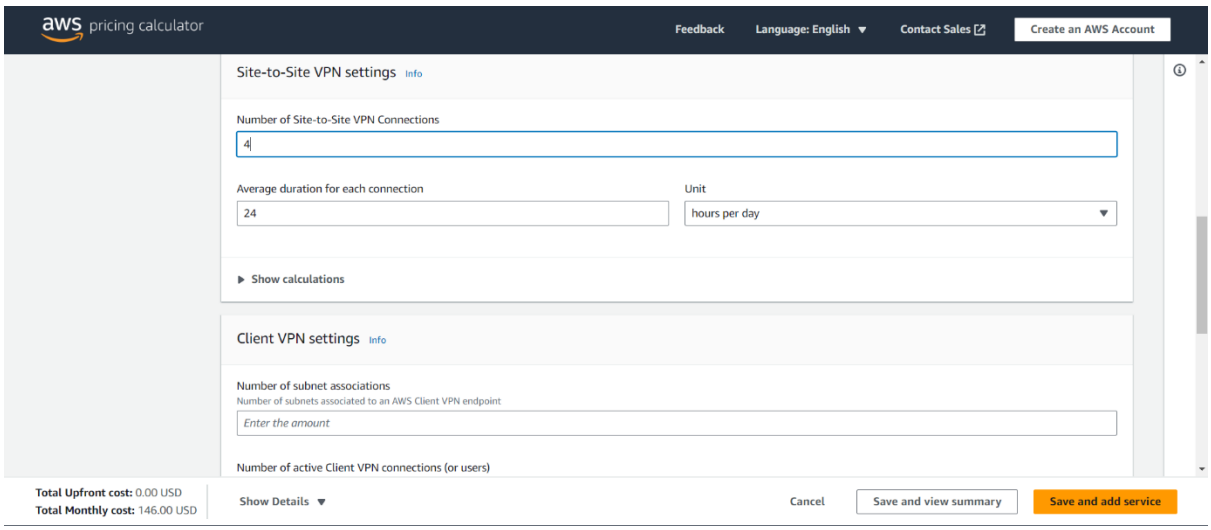

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Step 1
Add service
Step 2
Configure service
Create estimate: Configure Amazon EC2 Info
Description
Enter a description for your estimate
Choose a location type Info
Region
Choose a Region
US East (Ohio)
EC2 specifications 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: 3.07 USD
Show Details
Cancel
Save and view summary
Save and add service

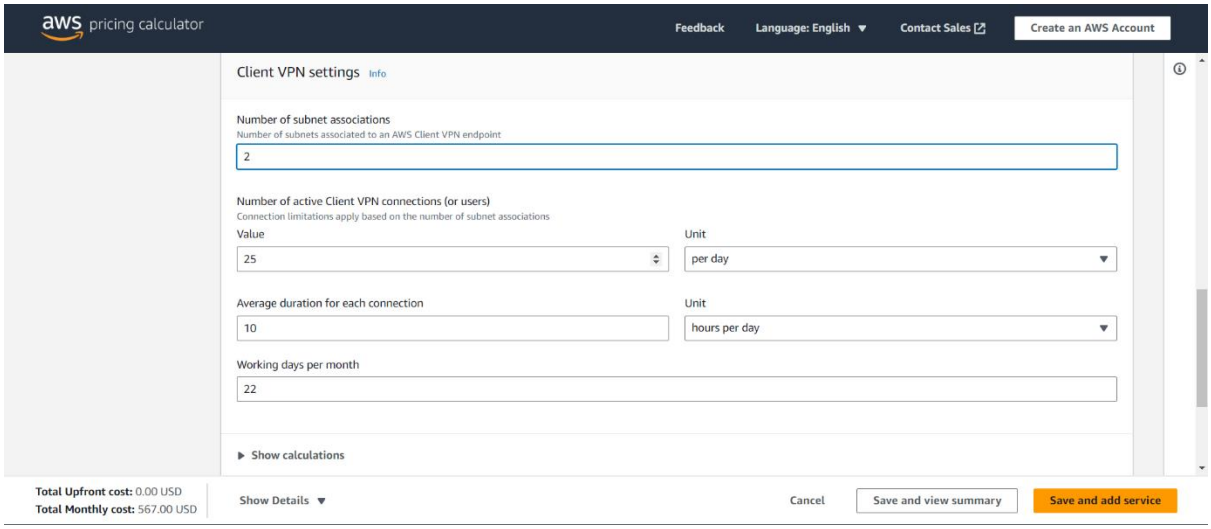

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Choose the graph that best represents your monthly workload
Constant usage
Daily spike traffic
Weekly spike traffic
Monthly spike traffic
Number of instances
Please specify the total number of Instances that you need 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
Search by instance name or filter by keyword
Instance family Info
c5
vCPUs
8
Memory (GiB)
60 GiB
Network performance
Any Network Performance
Show only current generation instances.
1
Total Upfront cost: 0.00 USD
Total Monthly cost: 1.53 USD
Show Details
Cancel
Save and view summary
Save and add service

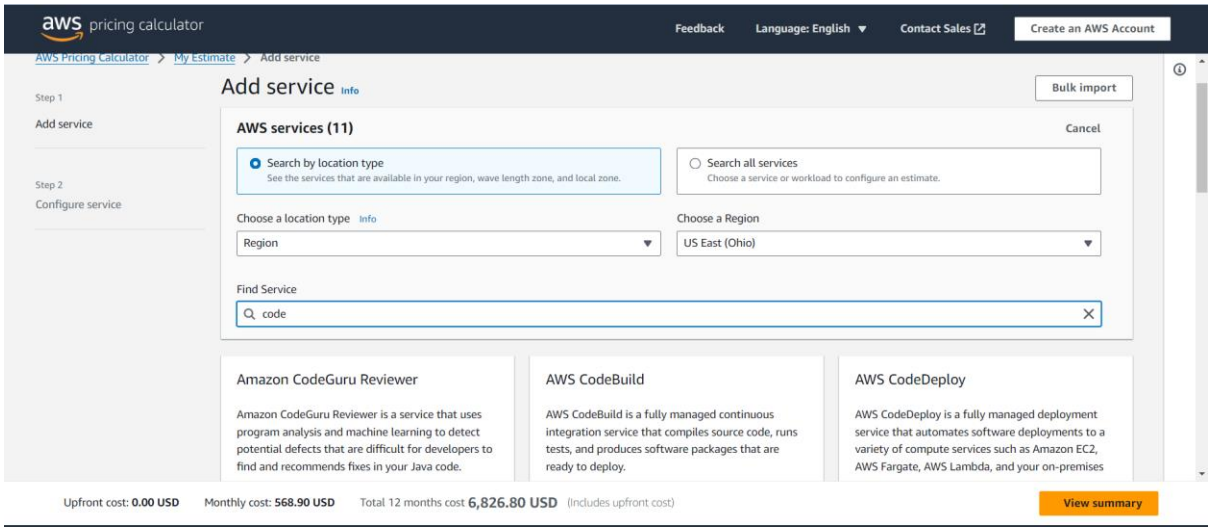

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Select the container and options to find your best price
Compute Savings Plans
One plan that automatically applies to all usage on EC2, Fargate, and Lambda. Up to 66% discount. Learn more
Reservation term
1 year
3 year
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
Usage type
Utilization percent per mo...
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

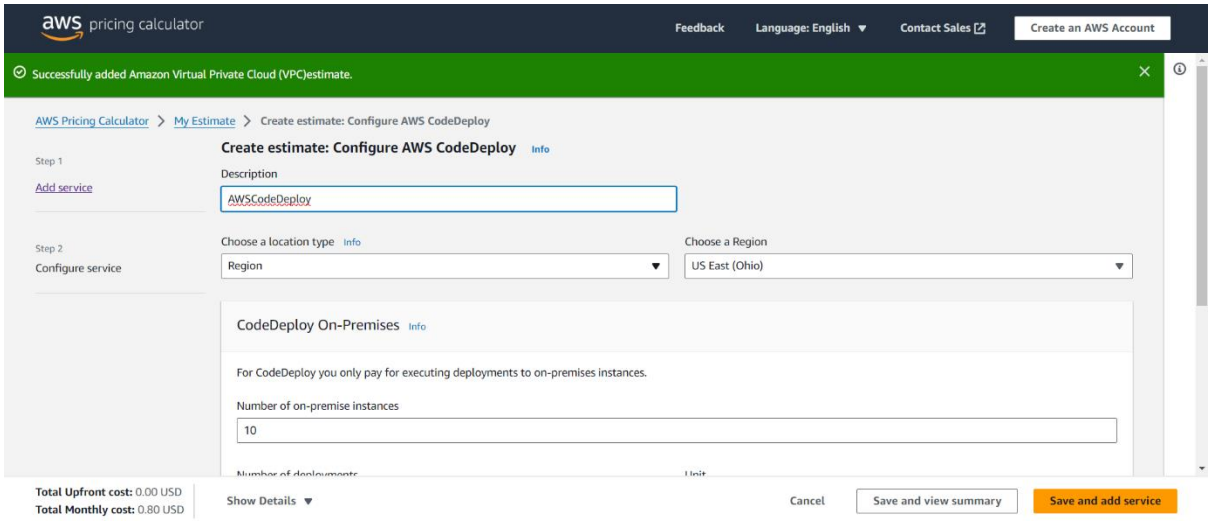

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Amazon Elastic Block Store (EBS) - optional Info
Detailed monitoring - optional Info
Data transfer - optional
Additional costs - optional
Acknowledgement
AWS Pricing Calculator provides only an estimate of your AWS 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

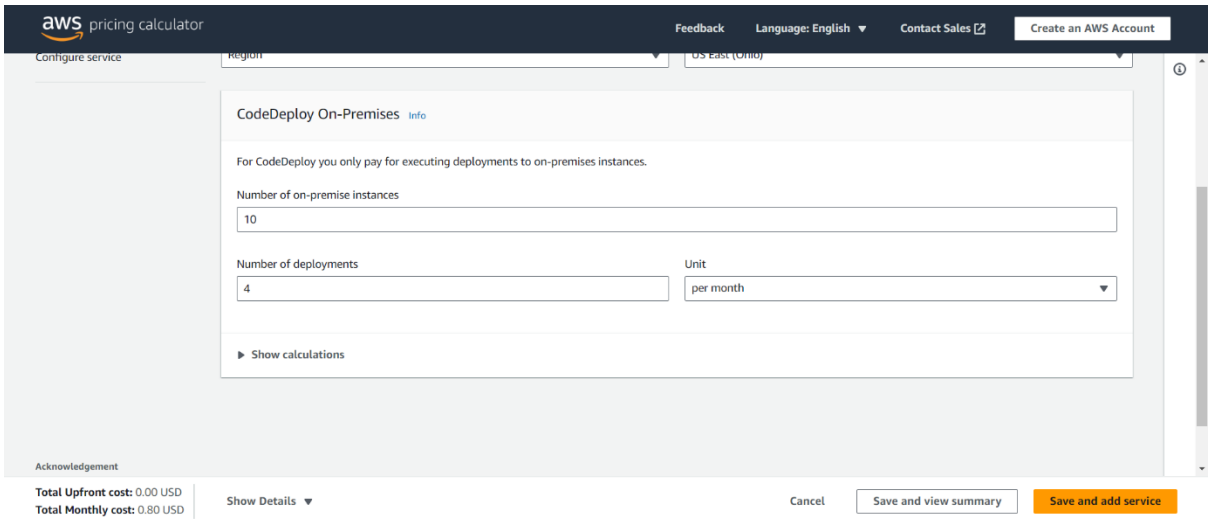

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Successfully added Amazon EC2 estimate.
AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon Virtual Private Cloud (VPC)
Step 1
Add service
Step 2
Configure service
Create estimate: Configure Amazon Virtual Private Cloud (VPC) Info
Description
Enter a description for your estimate
Choose a location type Info
Region
Choose a 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

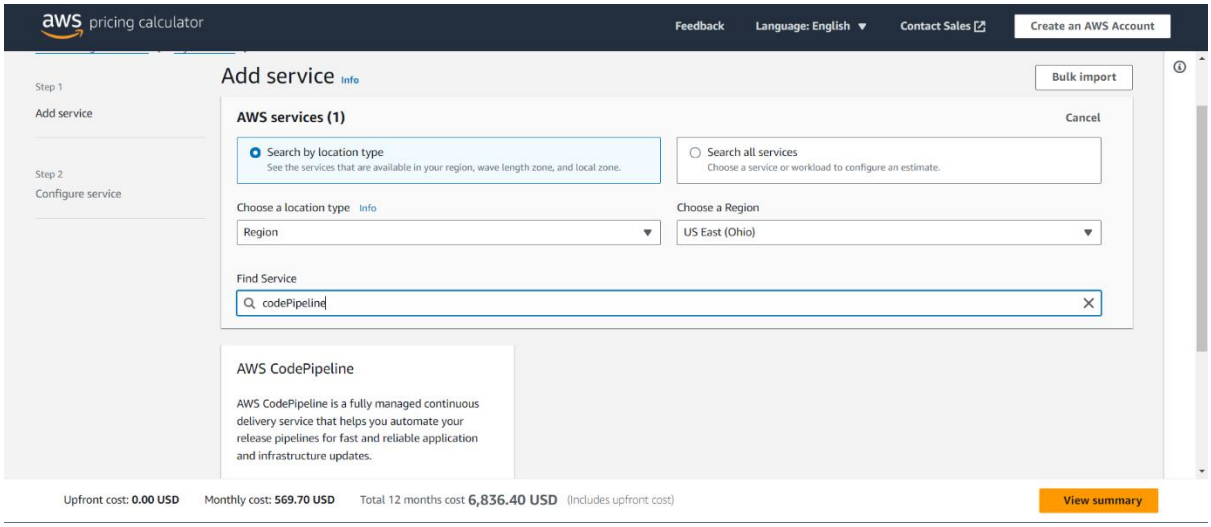

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Site-to-Site VPN settings Info
Number of Site-to-Site VPN Connections
4
Average duration for each connection
24
Unit
hours per day
Show calculations
Client VPN settings Info
Number of subnet associations
Number of subnets 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

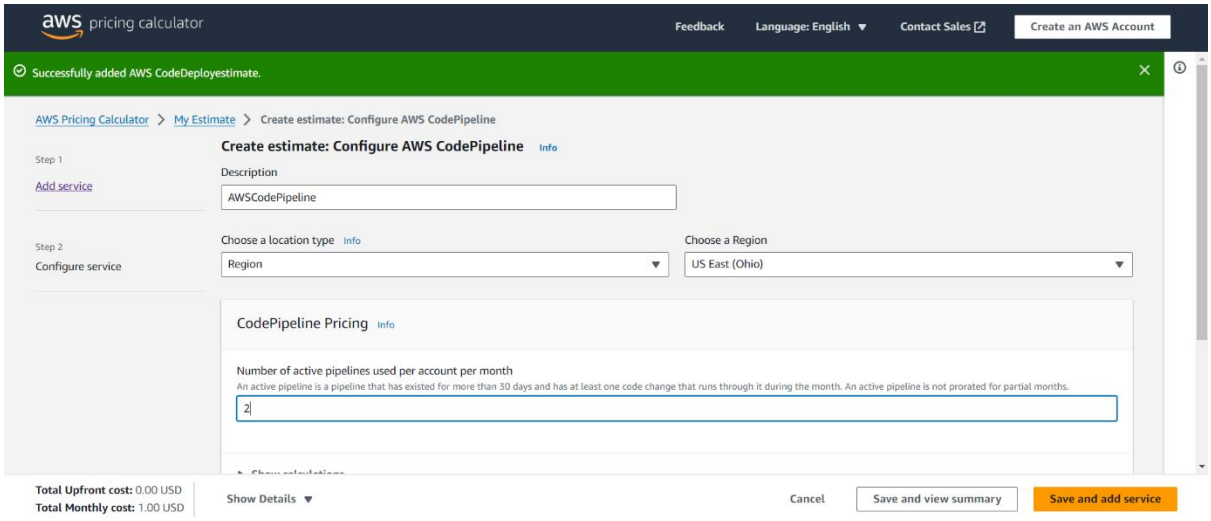

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Client VPN settings Info
Number of subnet associations
Number of subnets associated to an AWS Client VPN endpoint
2
Number of active Client VPN connections (or users)
Connection limitations apply based on the number of subnet associations
Value
25
Unit
per day
Average duration for each connection
10
Unit
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

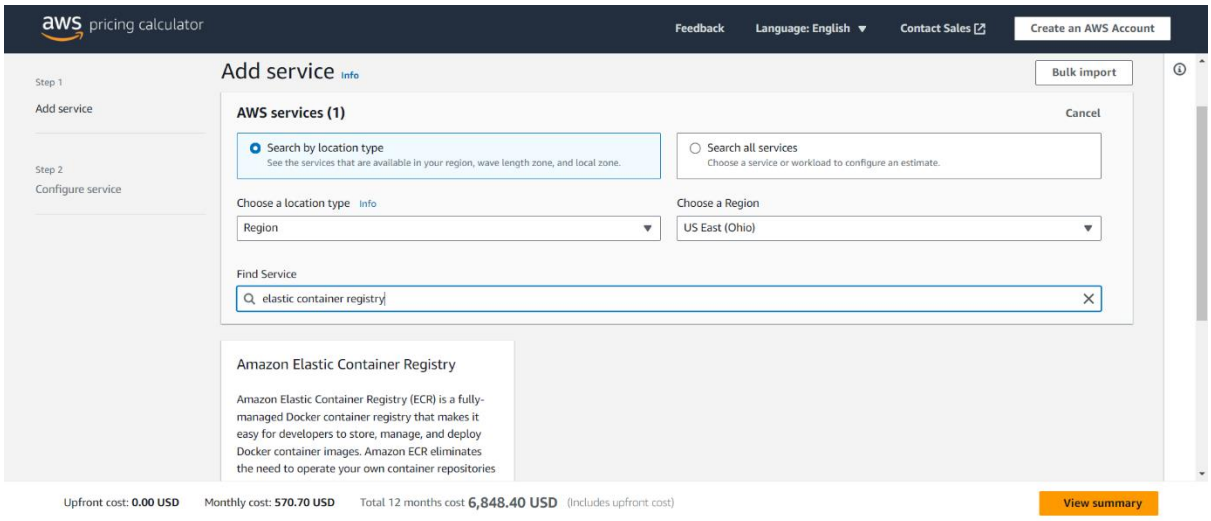

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator > My Estimate > Add service
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (11)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Find Service
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

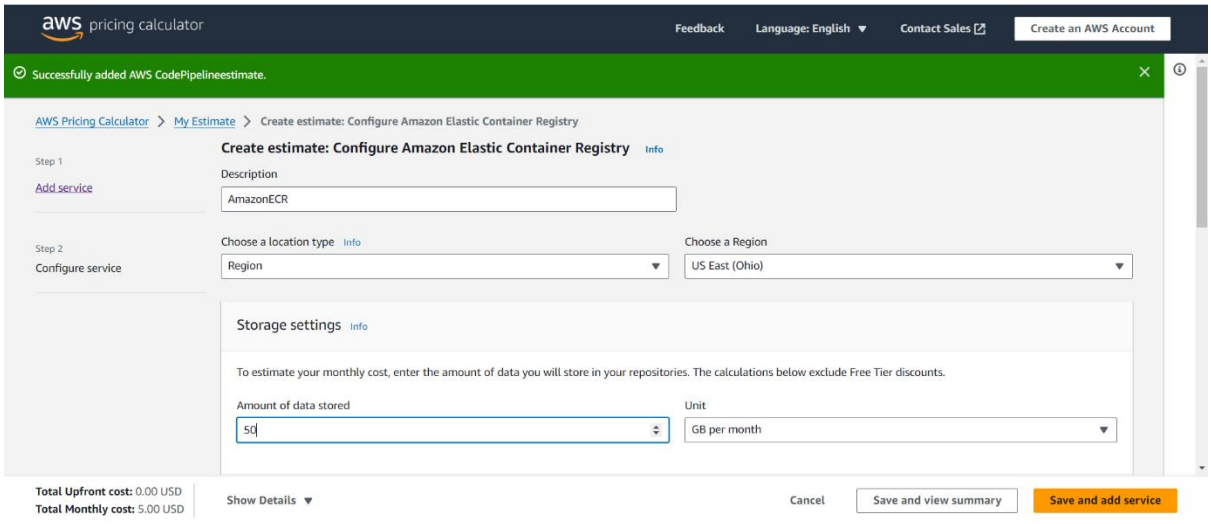

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Successfully added Amazon Virtual Private Cloud (VPC)estimate.
AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS CodeDeploy
Step 1
Add service
Step 2
Configure service
Create estimate: Configure AWS CodeDeploy Info
Description
AWSCodeDeploy
Choose a location type Info
Region
Choose a 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
Total Upfront cost: 0.00 USD
Total Monthly cost: 0.80 USD
Show Details
Cancel
Save and view summary
Save and add service

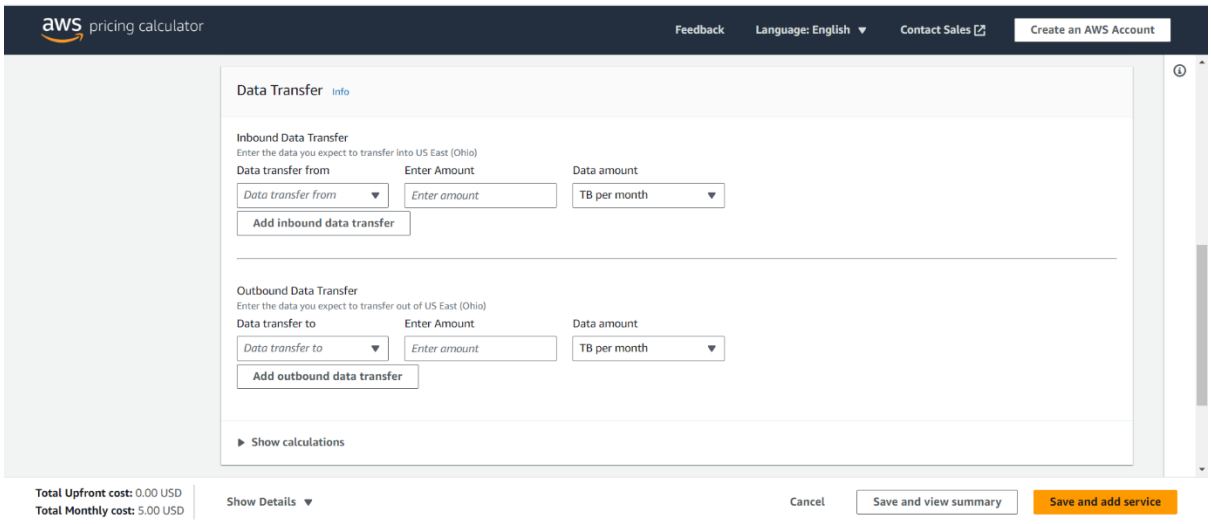

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Configure service
CodeDeploy On-Premises Info
For CodeDeploy you only pay for executing deployments to on-premises instances.
Number of on-premise instances
10
Number of deployments
4
Unit
per month
Show calculations
Acknowledgement
Total Upfront cost: 0.00 USD
Total Monthly cost: 0.80 USD
Show Details
Cancel
Save and view summary
Save and add service

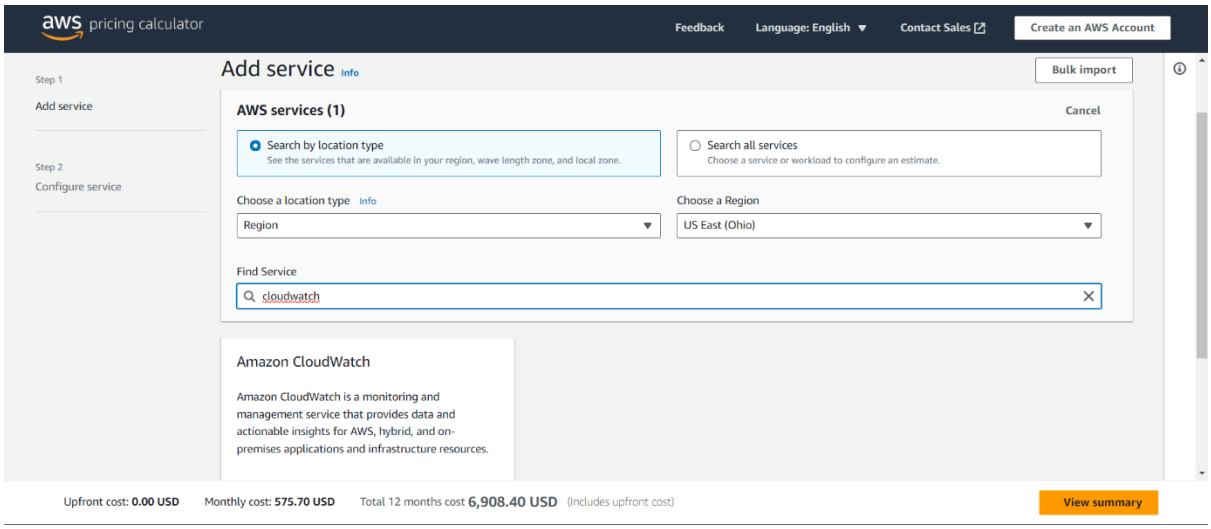

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (1)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Find Service
codePipeline
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

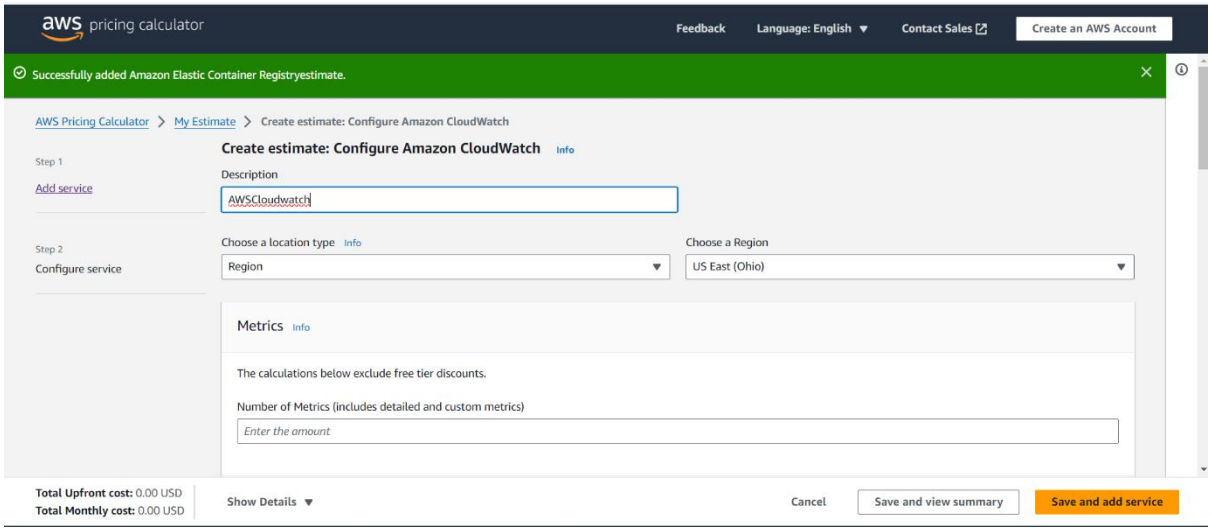

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Successfully added AWS CodeDeployestimate.
AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS CodePipeline
Step 1
Add service
Step 2
Configure service
Create estimate: Configure AWS CodePipeline Info
Description
AWSCodePipeline
Choose a location type Info
Region
Choose a Region
US East (Ohio)
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.
2
Total Upfront cost: 0.00 USD
Total Monthly cost: 1.00 USD
Show Details
Cancel
Save and view summary
Save and add service

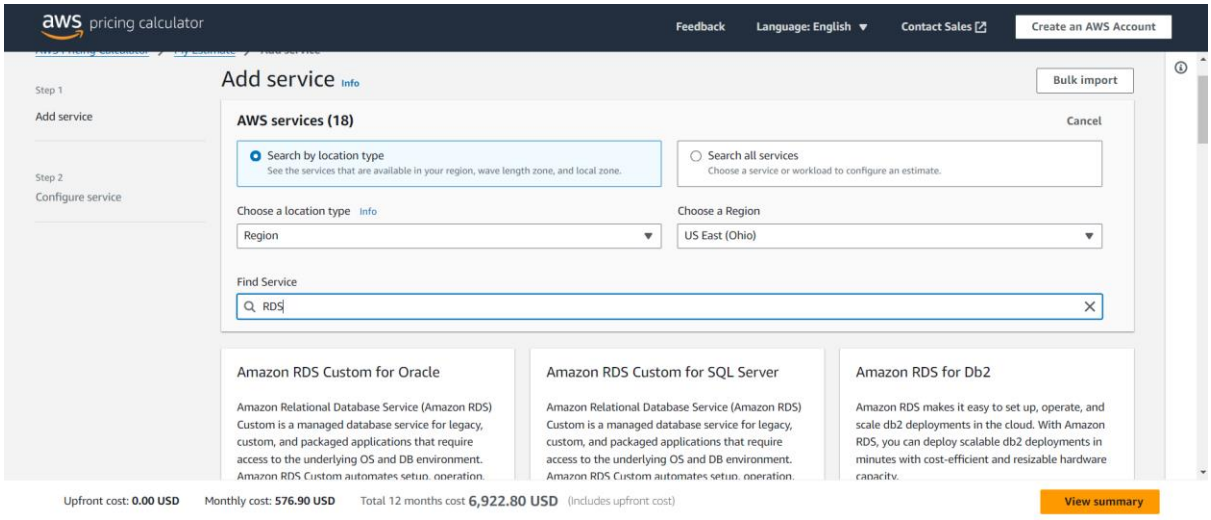

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (1)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Find Service
elastic container registry
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

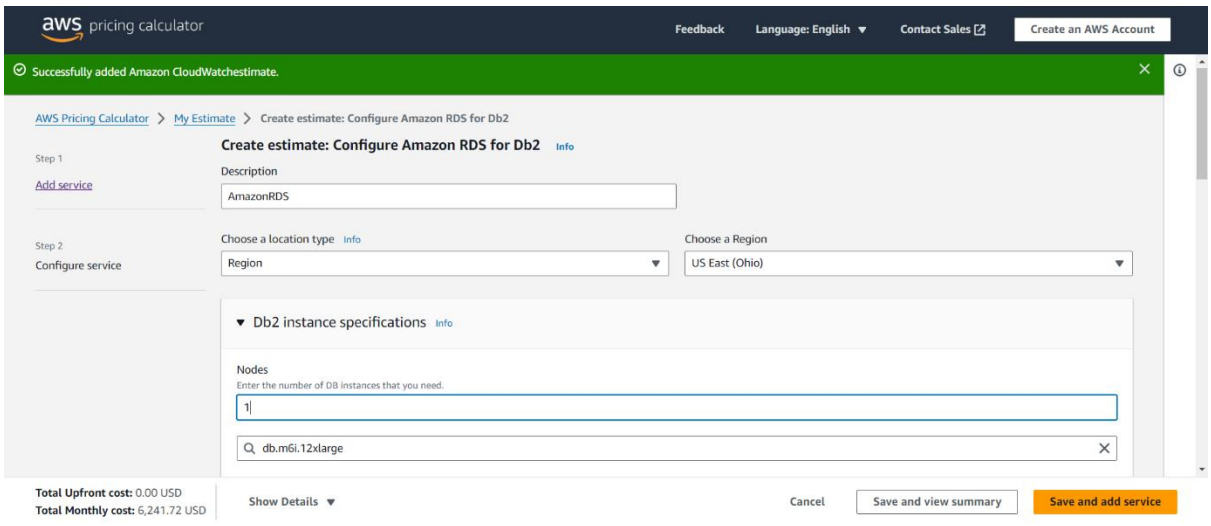

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Successfully added AWS CodePipelineestimate.
AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon Elastic Container Registry
Step 1
Add service
Step 2
Configure service
Create estimate: Configure Amazon Elastic Container Registry Info
Description
AmazonECR
Choose a location type Info
Region
Choose a Region
US East (Ohio)
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
50
Unit
GB per month
Total Upfront cost: 0.00 USD
Total Monthly cost: 5.00 USD
Show Details
Cancel
Save and view summary
Save and add service

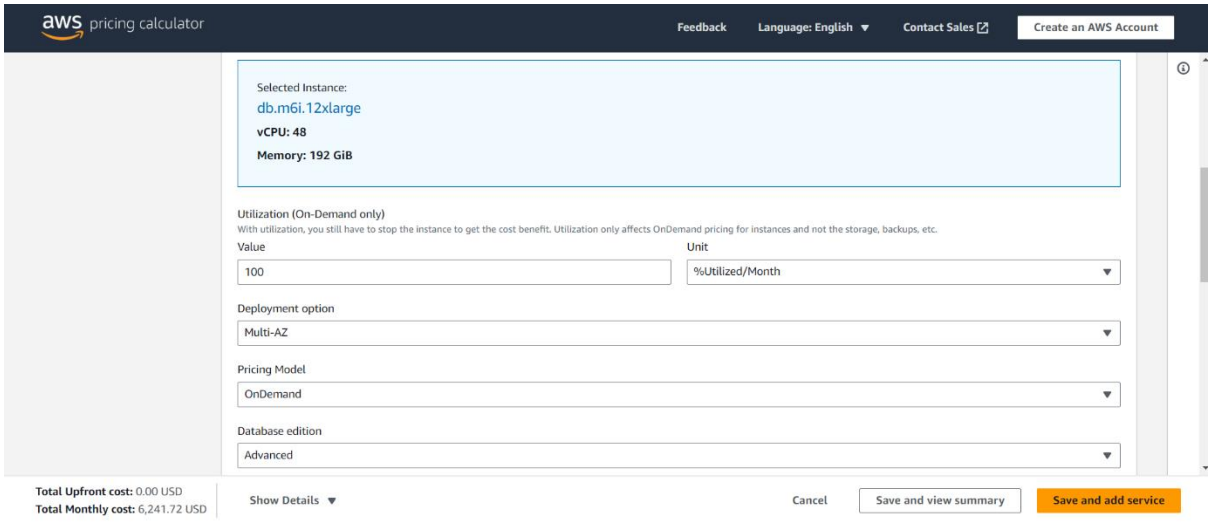

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Data Transfer Info
Inbound Data Transfer
Enter the data you expect to transfer into US East (Ohio)
Data transfer from
Enter Amount
Data amount
Data transfer from
Enter amount
TB per month
Add inbound data transfer
Outbound Data Transfer
Enter the data you expect to transfer out of US East (Ohio)
Data transfer to
Enter Amount
Data amount
Data transfer to
Enter amount
TB per month
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

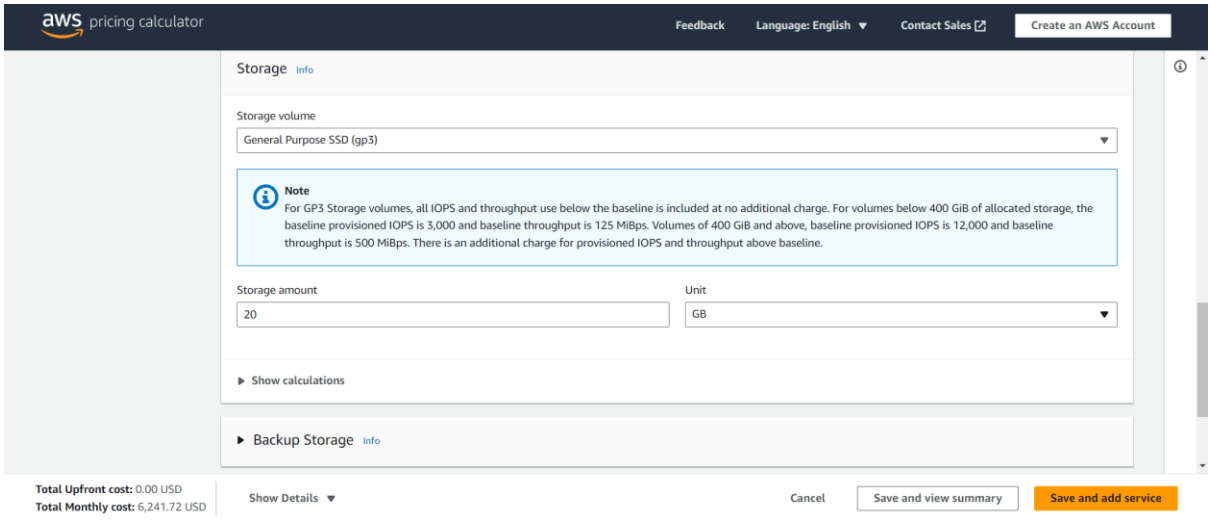

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (1)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Find Service
cloudwatch
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

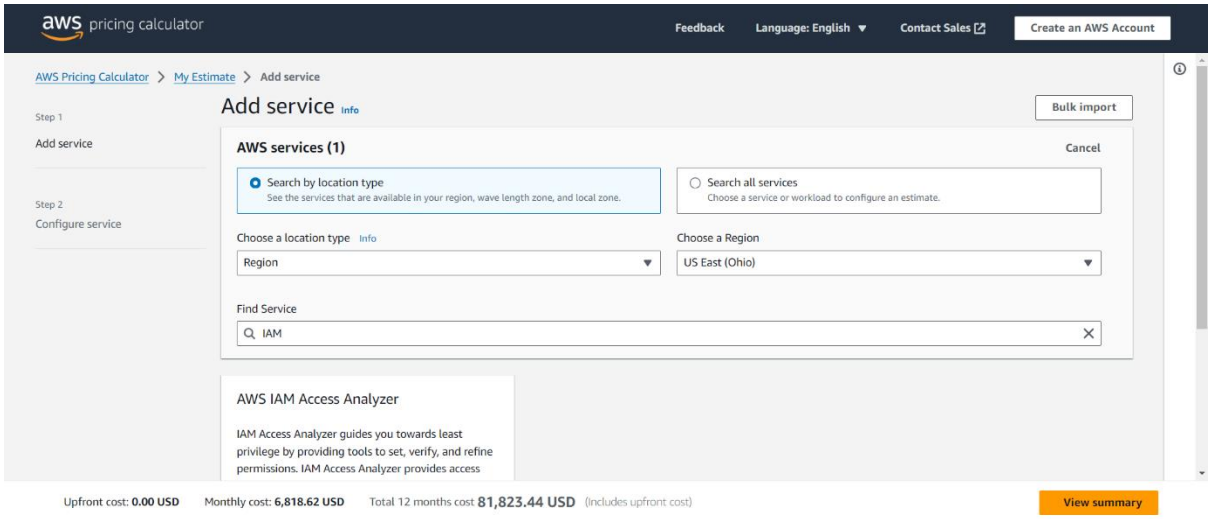

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Successfully added Amazon Elastic Container Registryestimate.
AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon CloudWatch
Step 1
Add service
Step 2
Configure service
Create estimate: Configure Amazon CloudWatch Info
Description
AWSCloudwatch
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Metrics Info
The calculations below exclude free tier discounts.
Number of Metrics (includes detailed and custom metrics)
Enter the amount
Total Upfront cost: 0.00 USD
Total Monthly cost: 0.00 USD
Show Details
Cancel
Save and view summary
Save and add service

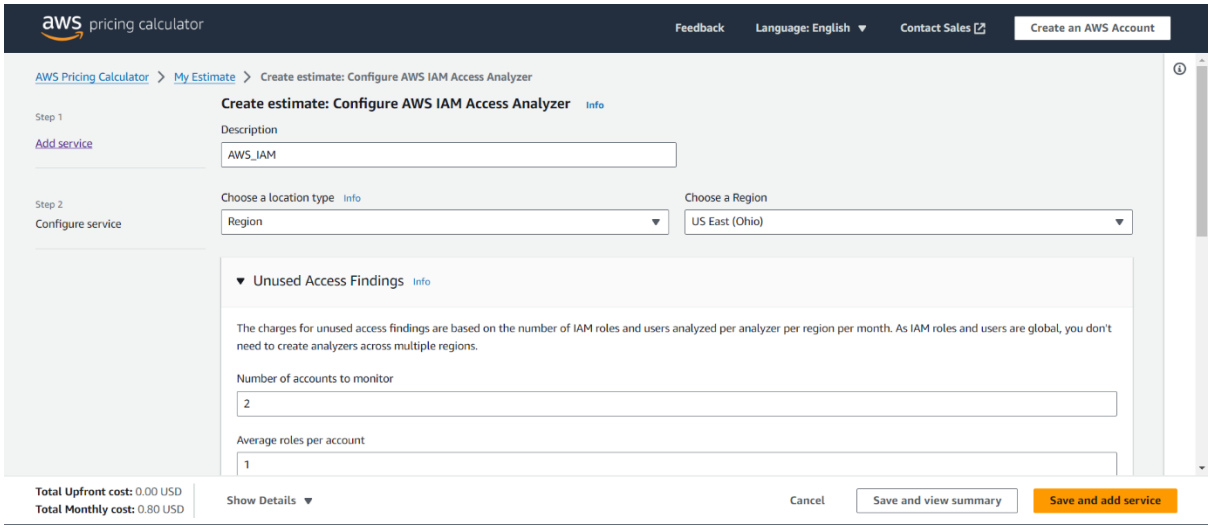

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (18)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Find Service
RDS
Amazon RDS Custom for Oracle
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 for SQL Server
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 for Db2
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

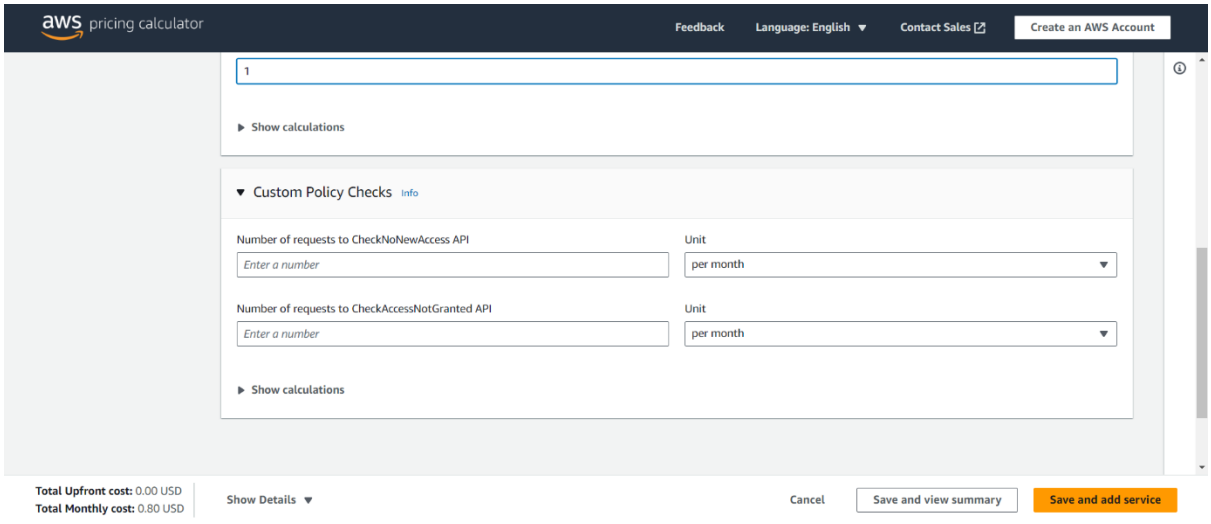

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Successfully added Amazon CloudWatchestimate.
AWS Pricing Calculator > My Estimate > Create estimate: Configure Amazon RDS for Db2
Step 1
Add service
Step 2
Configure service
Create estimate: Configure Amazon RDS for Db2 Info
Description
AmazonRDS
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Db2 instance specifications Info
Nodes
Enter the number of DB instances that you need.
1
db.m6i.12xlarge
Total Upfront cost: 0.00 USD
Total Monthly cost: 6,241.72 USD
Show Details
Cancel
Save and view summary
Save and add service

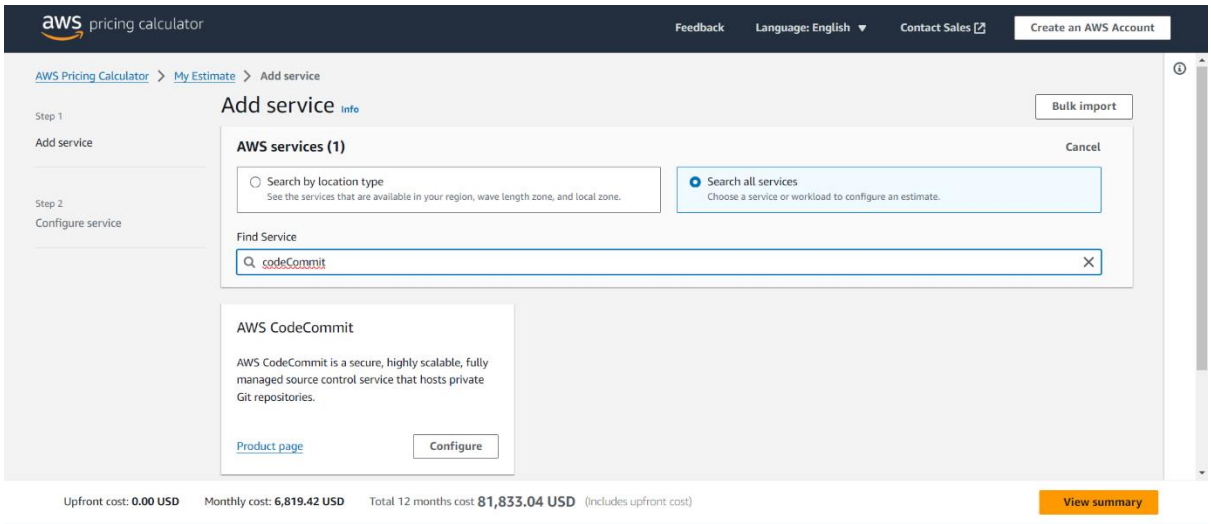

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Selected Instance:
db.m6i.12xlarge
vCPU: 48
Memory: 192 GiB
Utilization (On-Demand only)
With utilization, you still have to stop the instance to get the cost benefit. Utilization only affects OnDemand pricing for instances and not the storage, backups, etc.
Value
100
Unit
%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

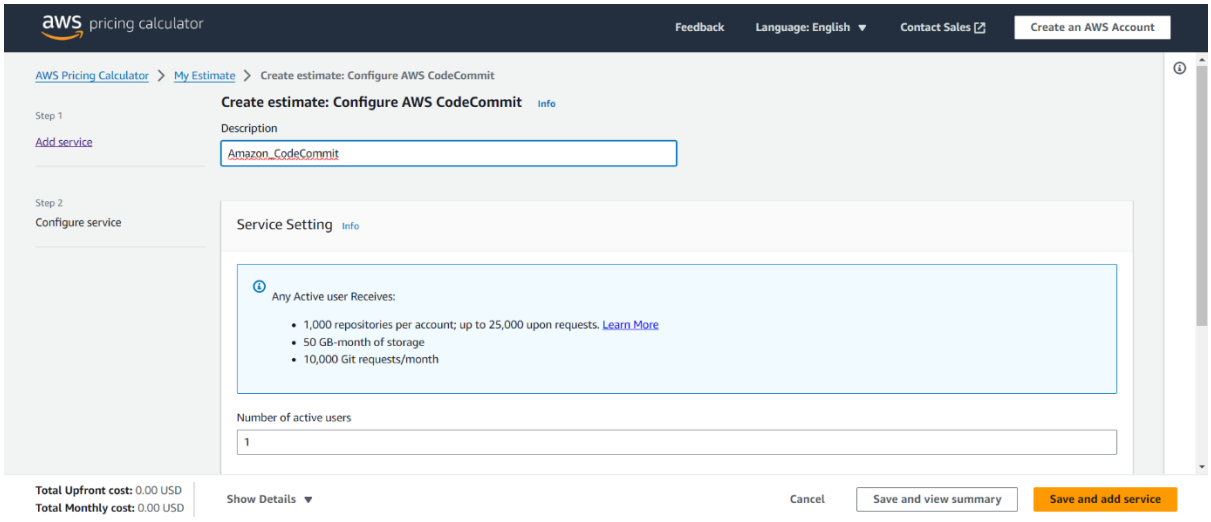

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
Storage Info
Storage volume
General Purpose SSD (gp3)
Note
For GP3 Storage volumes, all IOPS and throughput use below the baseline is included at no additional charge. For volumes below 400 GiB of allocated storage, 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
20
Unit
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

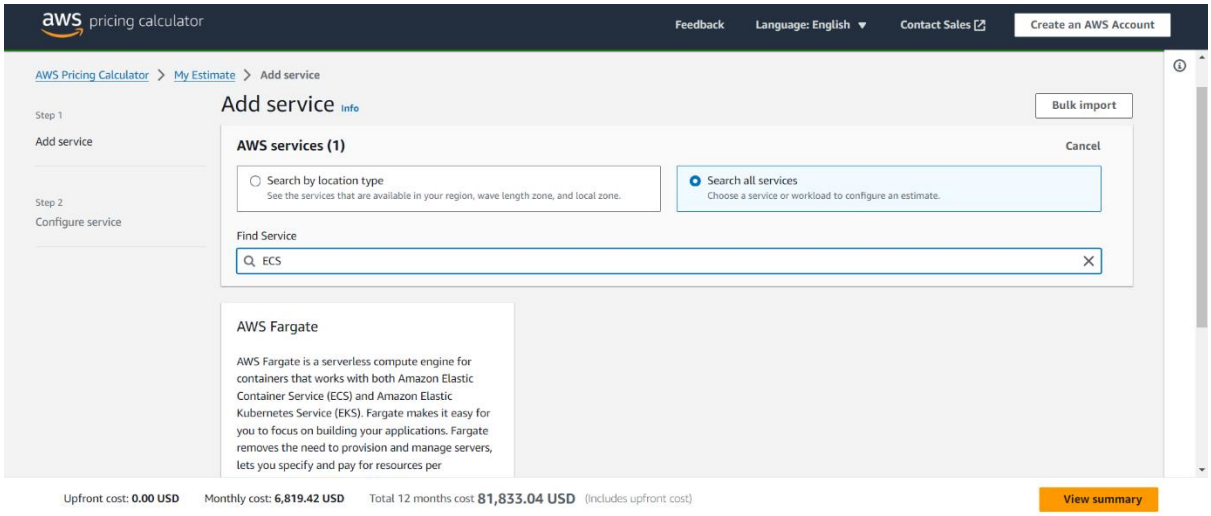

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator > My Estimate > Add service
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (1)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Choose a location type Info
Region
Choose a Region
US East (Ohio)
Find Service
IAM
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

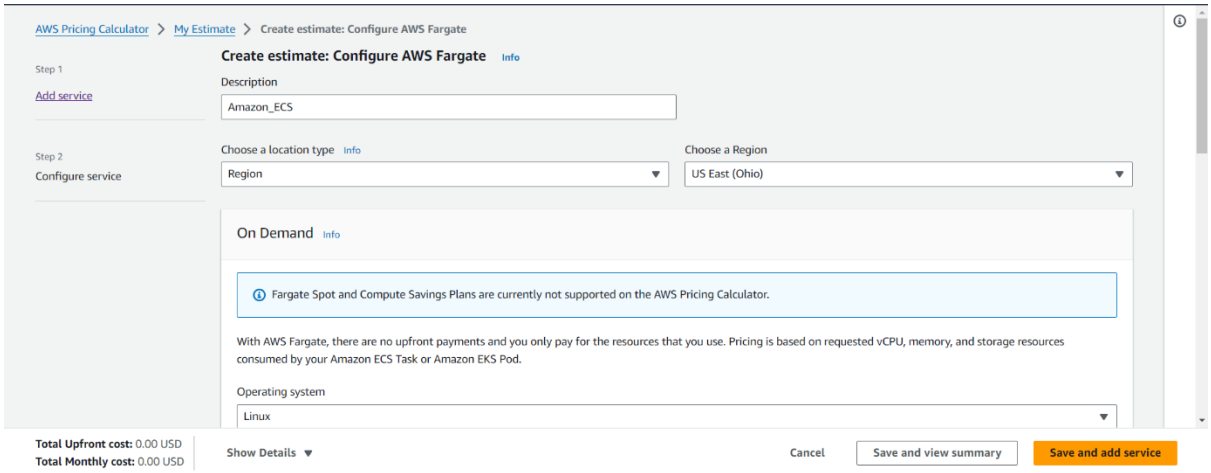

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS IAM Access Analyzer
Step 1
Add service
Step 2
Configure service
Create estimate: Configure AWS IAM Access Analyzer Info
Description
AWS_IAM
Choose a location type Info
Region
Choose a 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 region 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

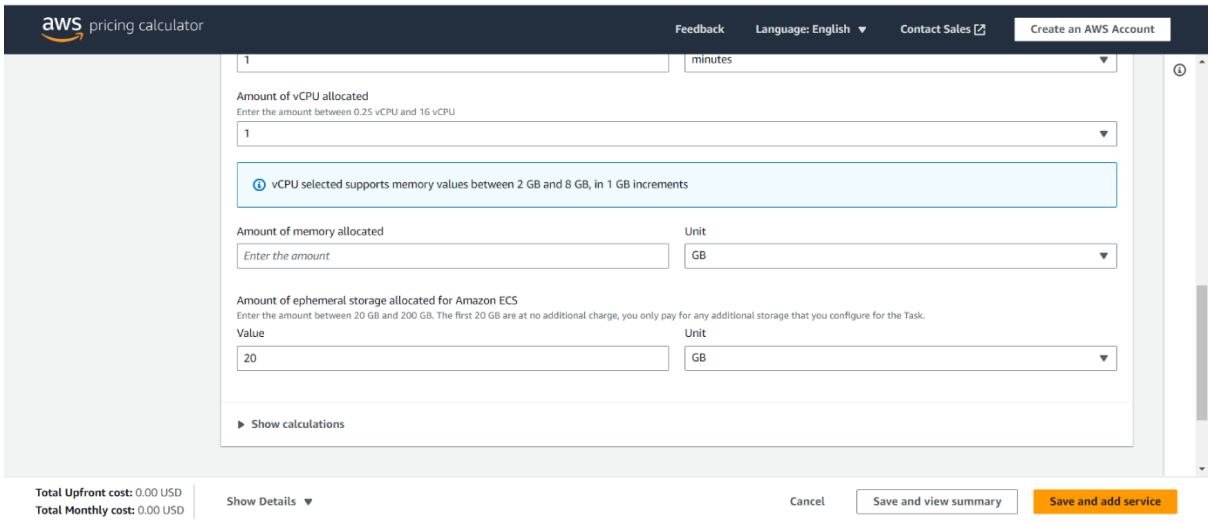

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
1
Show calculations
Custom Policy Checks Info
Number of requests to CheckNoNewAccess API
Enter a number
Unit
per month
Number of requests to CheckAccessNotGranted API
Enter a number
Unit
per month
Show calculations
Total Upfront cost: 0.00 USD
Total Monthly cost: 0.80 USD
Show Details
Cancel
Save and view summary
Save and add service

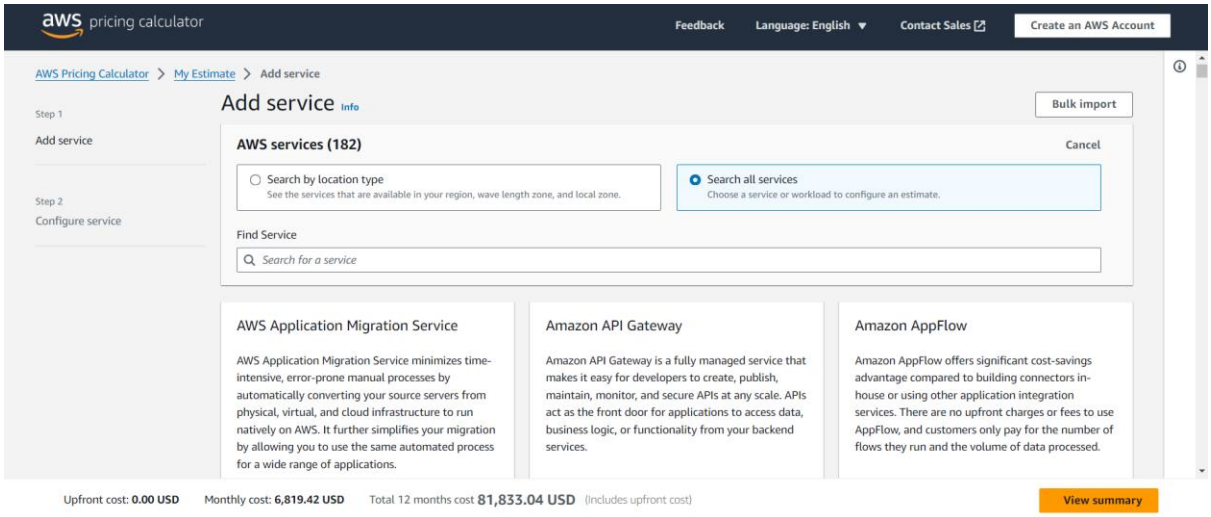

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator > My Estimate > Add service
Step 1
Add service
Step 2
Configure service
Add service Info
Bulk import
AWS services (1)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Find Service
codeCommit
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

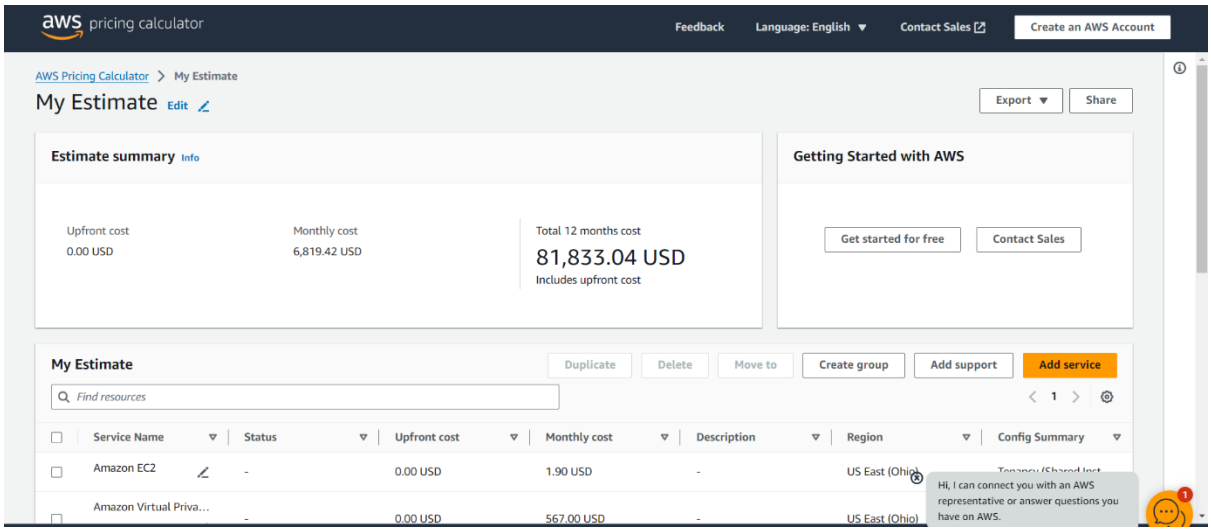

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS CodeCommit
Step 1
Add service
Create estimate: Configure AWS CodeCommit Info
Description
Amazon_CodeCommit
Step 2
Configure service
Service Setting 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

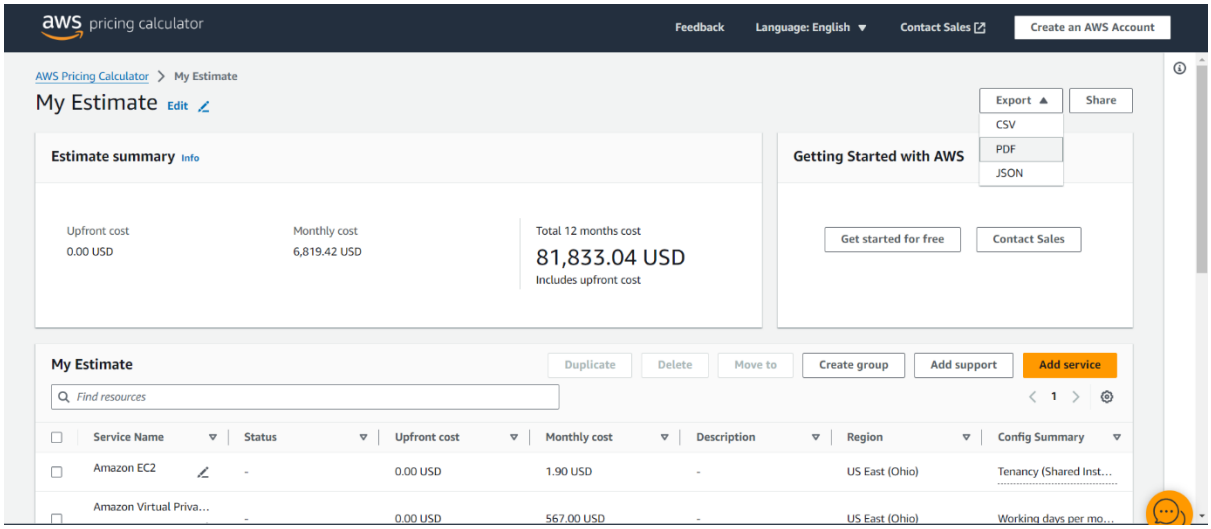

aws pricing calculator
Feedback
Language: English
Contact Sales
Create an AWS Account
AWS Pricing Calculator > My Estimate > Add service
Step 1
Add service
Add service Info
Bulk import
Step 2
Configure service
AWS services (1)
Cancel
Search by location type
See the services that are available in your region, wave length zone, and local zone.
Search all services
Choose a service or workload to configure an estimate.
Find Service
ECS
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 81,833.04 USD (Includes upfront cost)
View summary


AWS Pricing Calculator > My Estimate > Create estimate: Configure AWS Fargate
Step 1
Add service
Create estimate: Configure AWS Fargate Info
Description
Amazon_ECS
Step 2
Configure service
Choose a location type Info
Region
Choose a 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
Enter the amount
Unit
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
AWS services (182)
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
Search for a service
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
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
Get started for free
Contact Sales
My Estimate
Duplicate
Delete
Move to
Create group
Add support
Add service
Find resources
Service Name
Status
Upfront cost
Monthly cost
Description
Region
Config Summary
Amazon EC2
0.00 USD
1.90 USD
US East (Ohio)
Amazon Virtual Priva...
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
Export
Share
CSV
PDF
JSON
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
Get started for free
Contact Sales
My Estimate
Duplicate
Delete
Move to
Create group
Add support
Add service
Find resources
1
Service Name
Status
Upfront cost
Monthly cost
Description
Region
Config Summary
Amazon EC2
-
0.00 USD
1.90 USD
-
US East (Ohio)
Tenancy (Shared Inst...
Amazon Virtual Priva...
-
0.00 USD
567.00 USD
-
US East (Ohio)
Working days per mo...

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)				

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

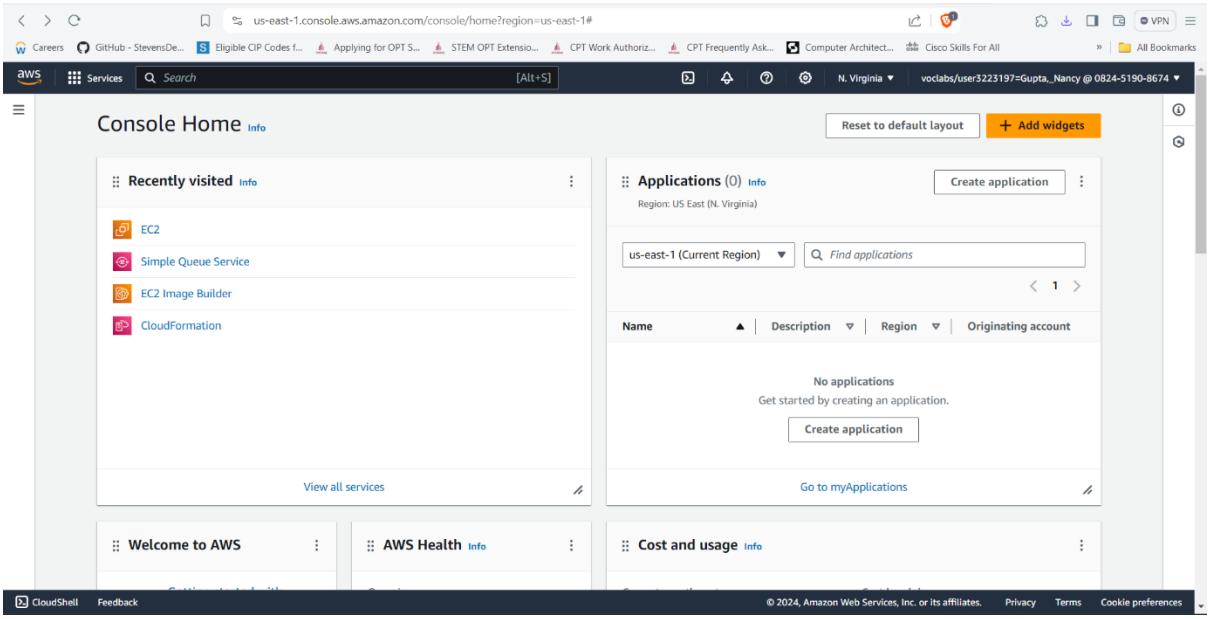
Status: -
Description: Amazon_ECS
Config summary: Operating system (Linux), CPU Architecture (x86), Average duration (1 minutes), Number of tasks or pods (1 per day), Amount of ephemeral storage allocated for Amazon ECS (20 GB)

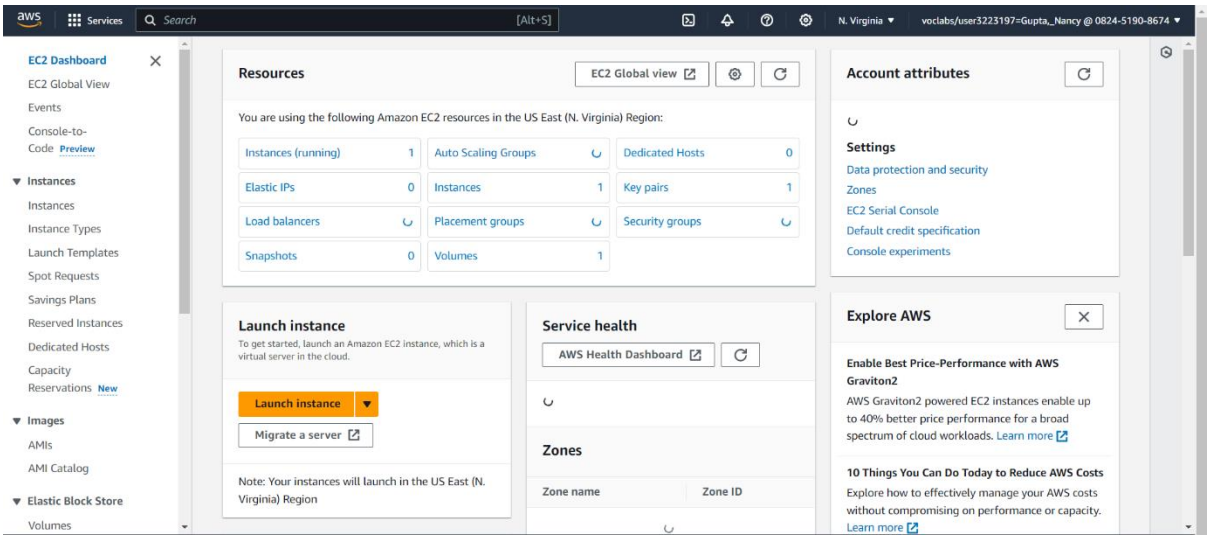
Phase 2: Analyzing the infrastructure of the monolithic application

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

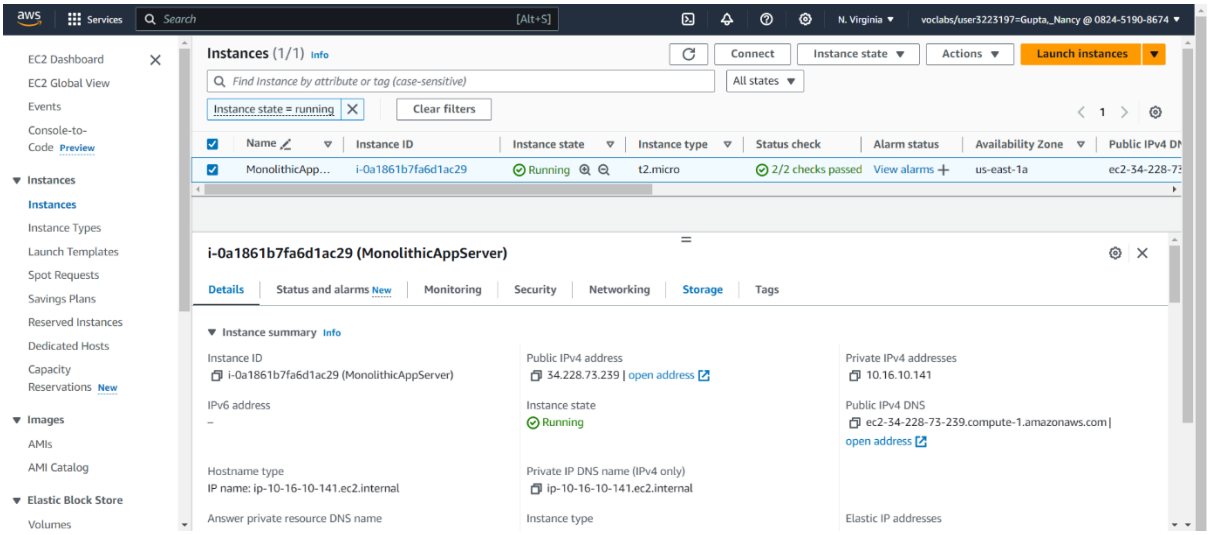
Task 2.1: Verify that the monolithic application is available

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

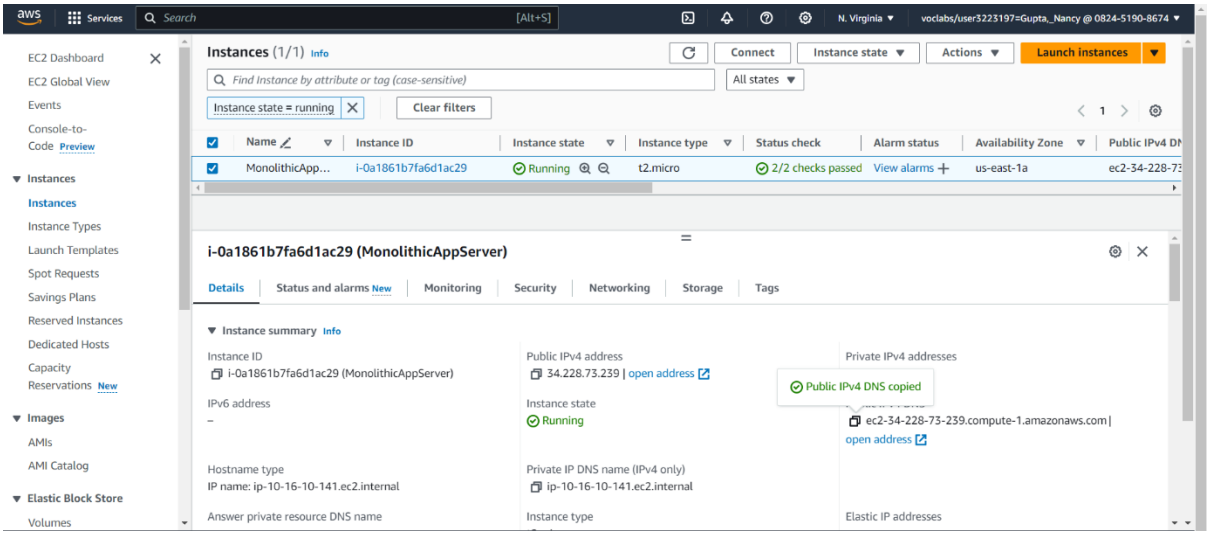




- **Navigate to the Amazon EC2 console.**

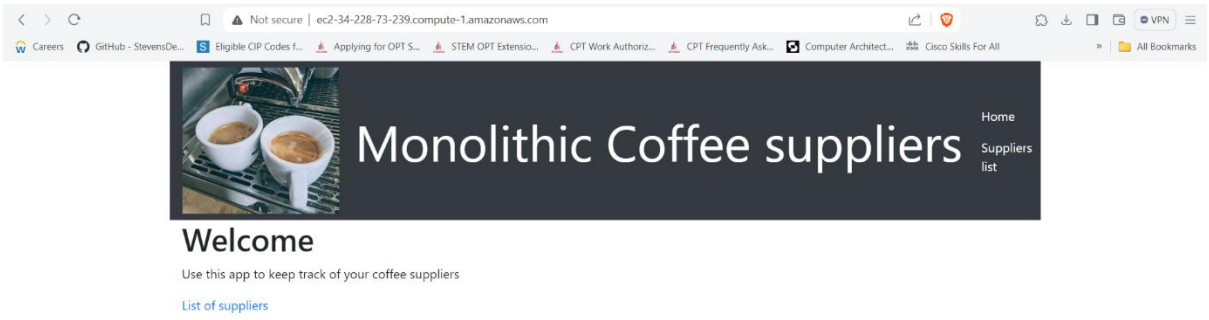


- **Copy the Public IPv4 address of the *MonolithicAppServer* instance, and load it in a new browser tab.**



The coffee suppliers website displays.

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.

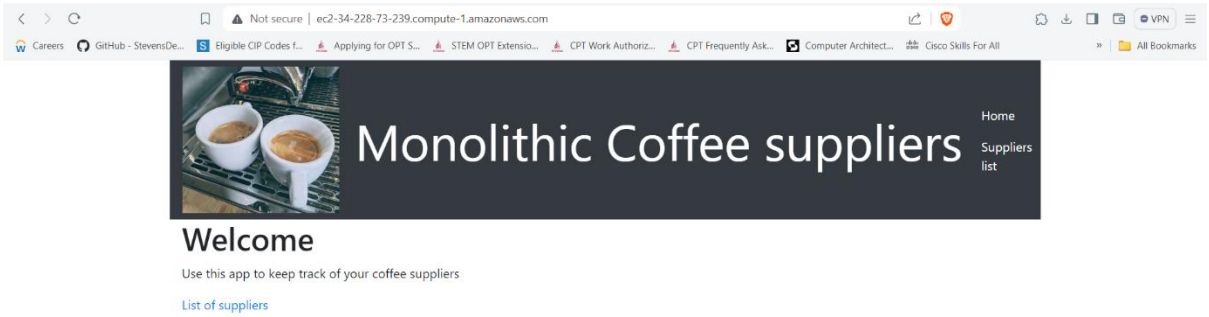


Task 2.2: Test the monolithic web application

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.

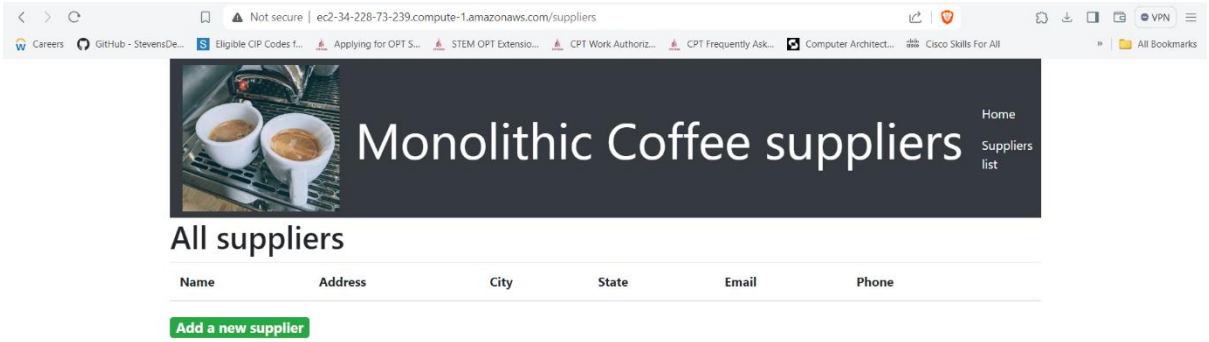
1. **Choose List of suppliers.**

   Notice that the URL path includes /suppliers.

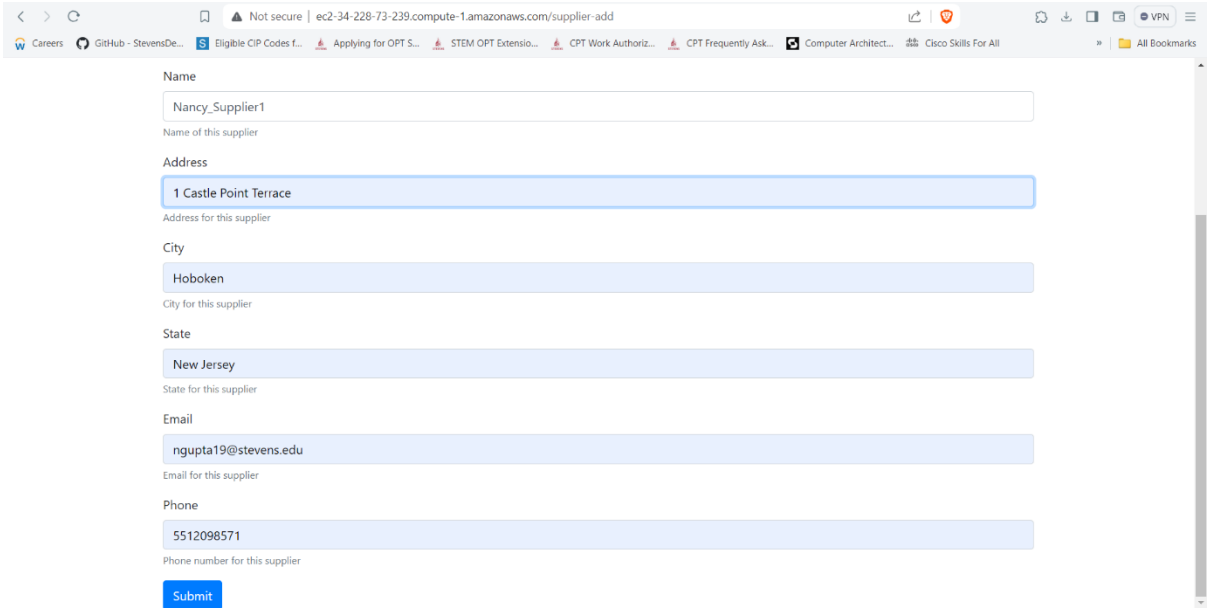


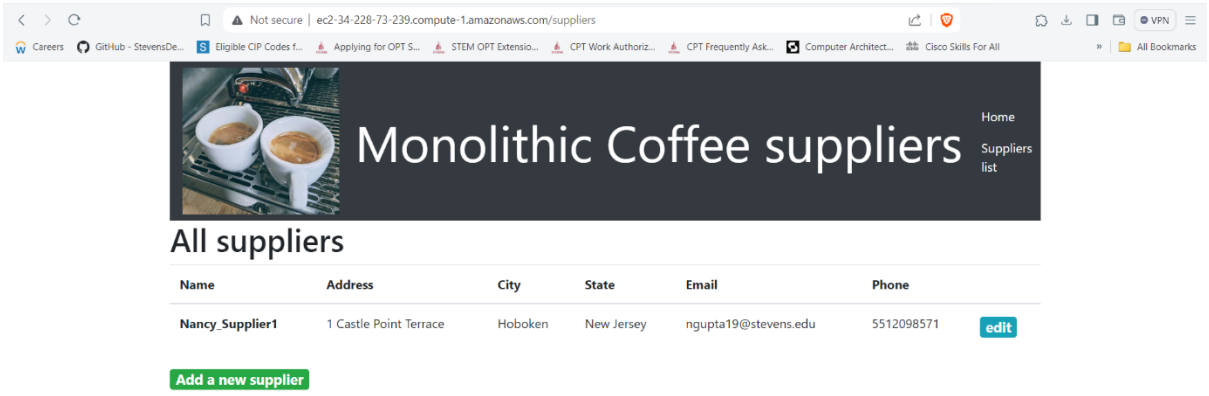
2. **Add a new supplier.**

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



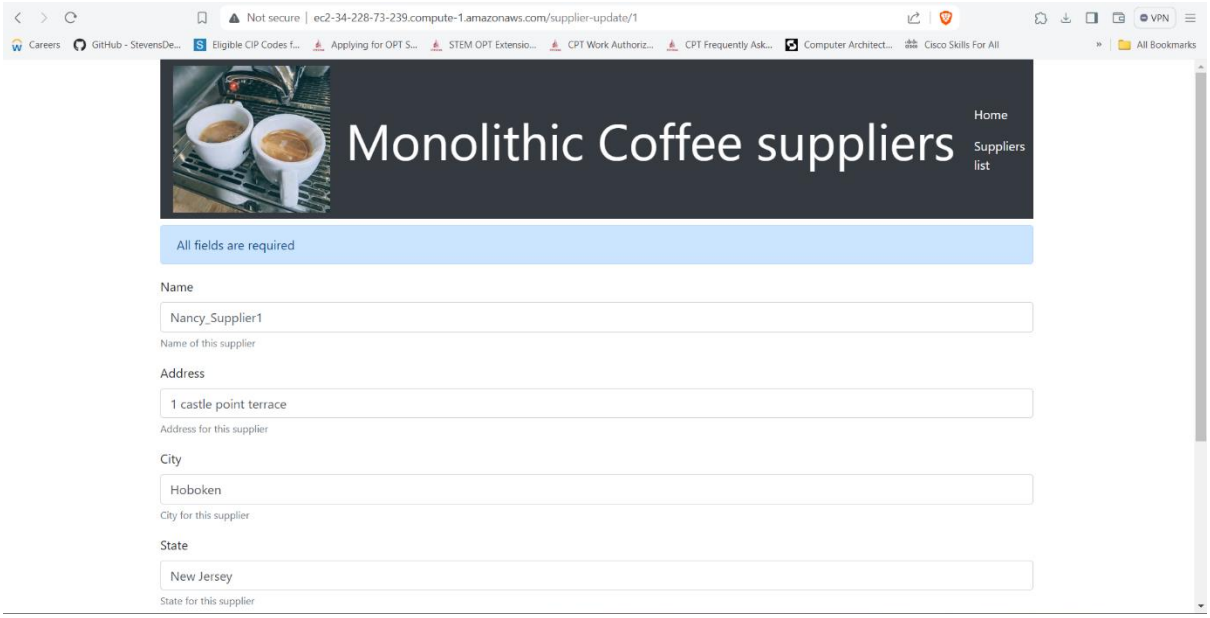
- **Fill in all of the fields to create a supplier entry.**



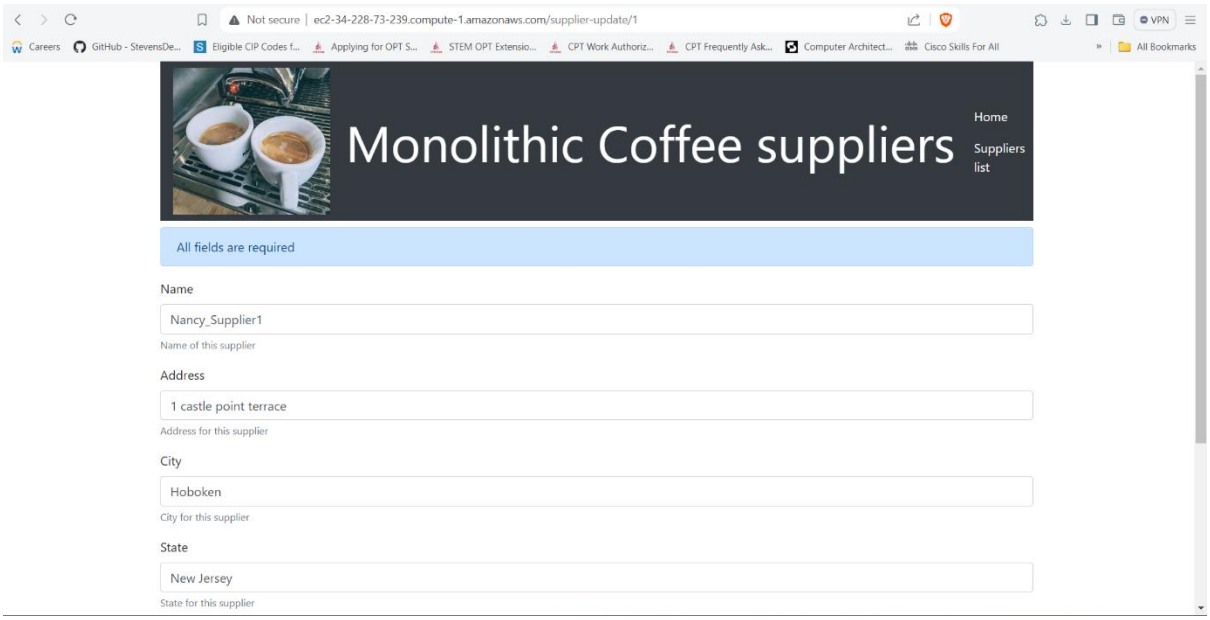


3. **Edit an entry.**

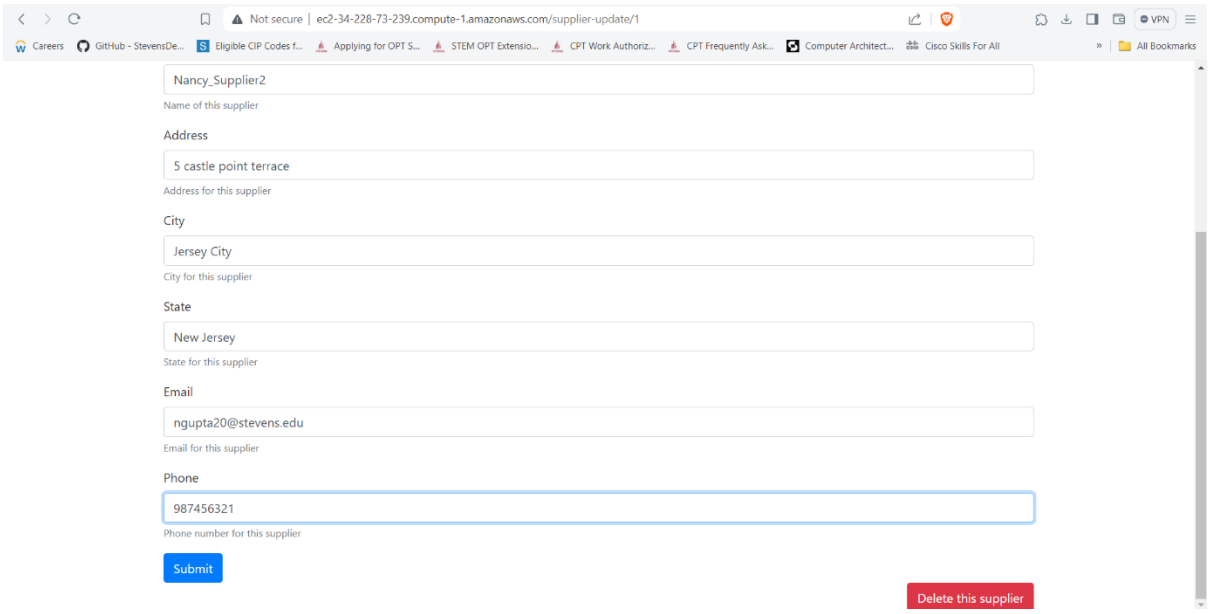
   - **On the page where you edit a supplier entry, notice that the URL path now includes supplier-update/1.**

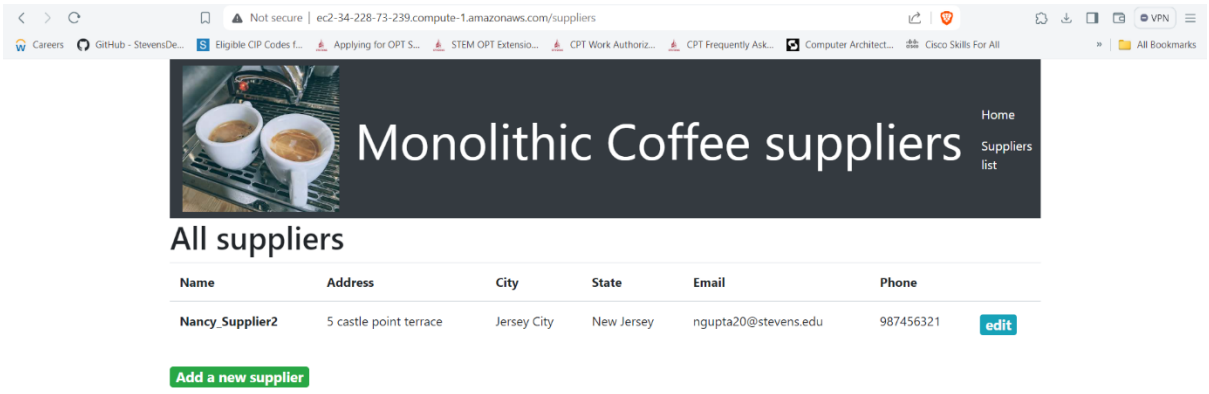


- **Modify the record in some way and save the change.**



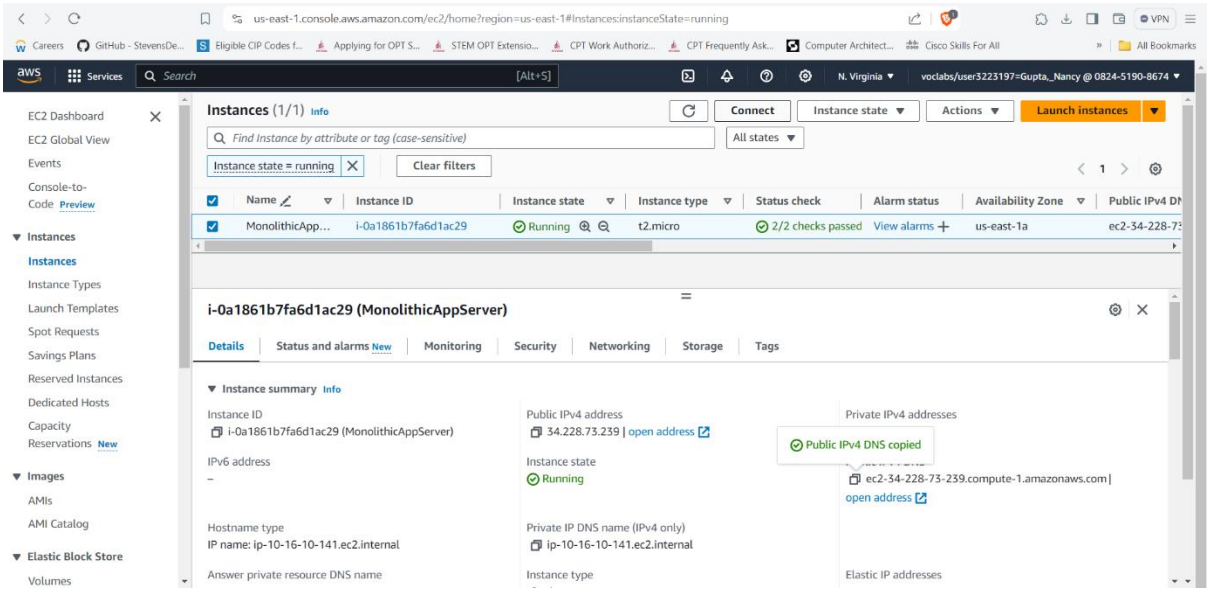
Notice that the change was saved in the record.

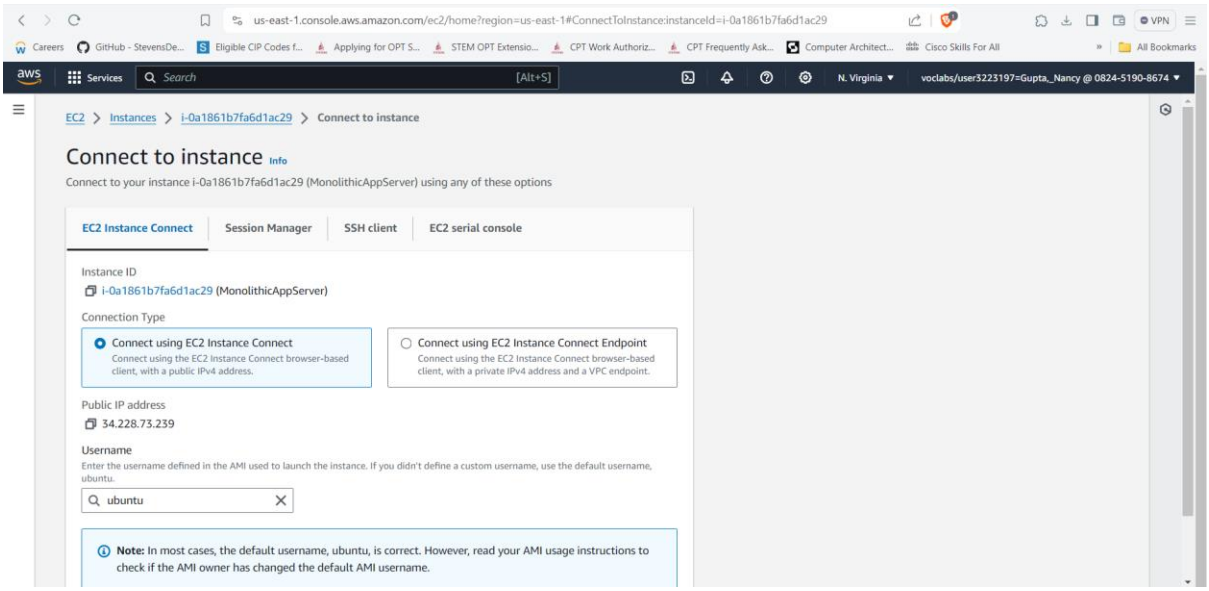


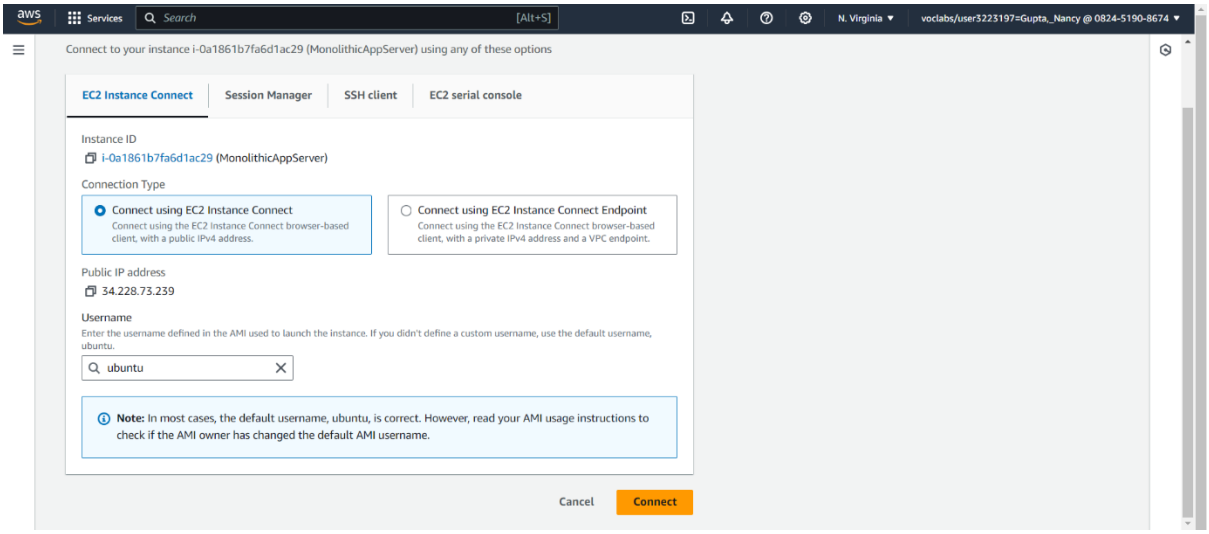


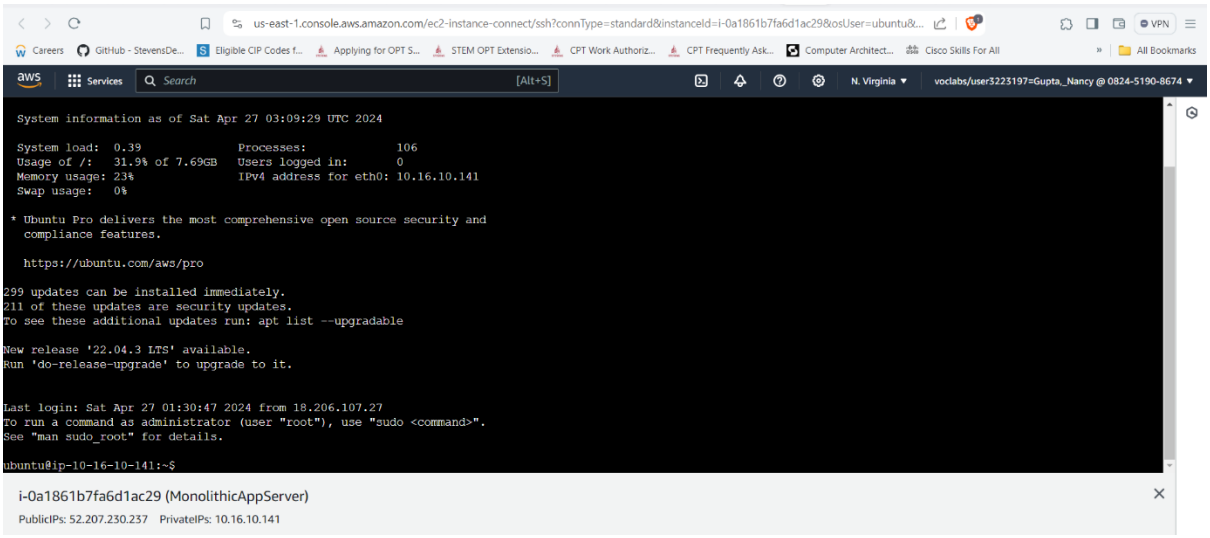
Task 2.3: Analyze how the monolithic application runs

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









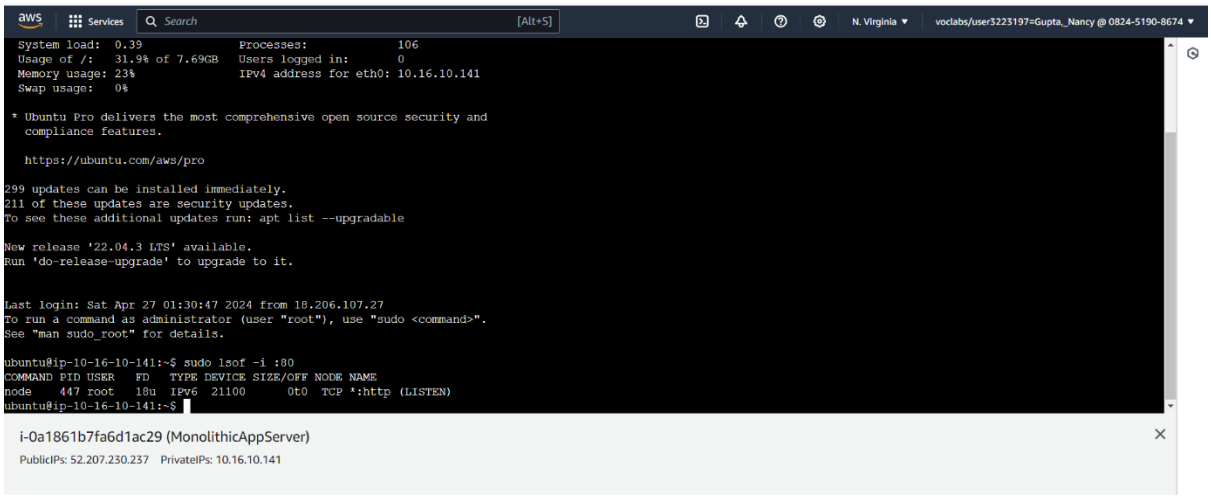
2. **Analyze how the application is running.**

- **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.



- **Next, run the following command:**

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

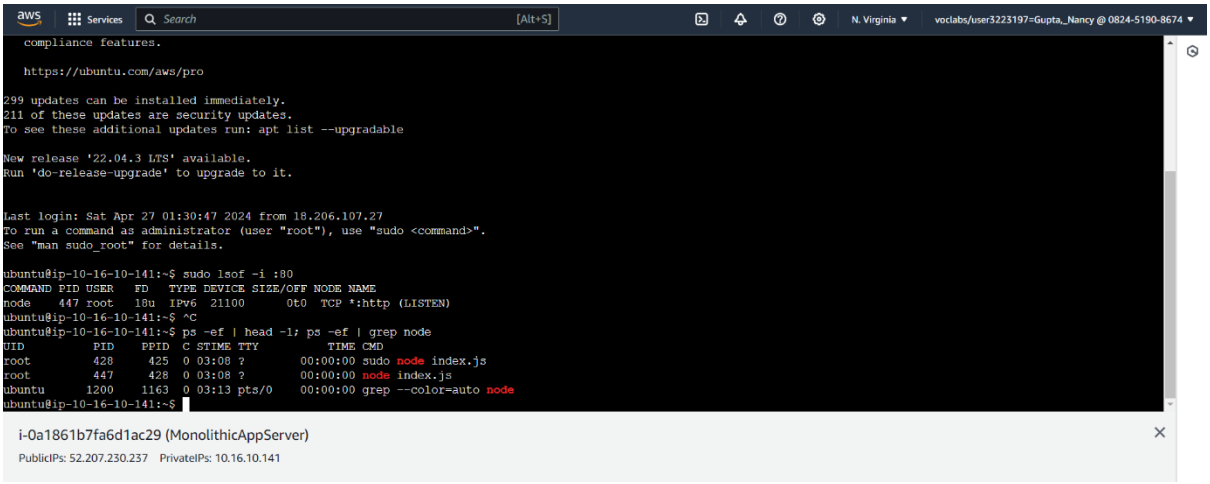
What did you notice in the command output? Which user on this EC2 instance is running a *node* process? Does the node process ID (PID) match any of the PIDs from the output of the command that you ran before the last one?

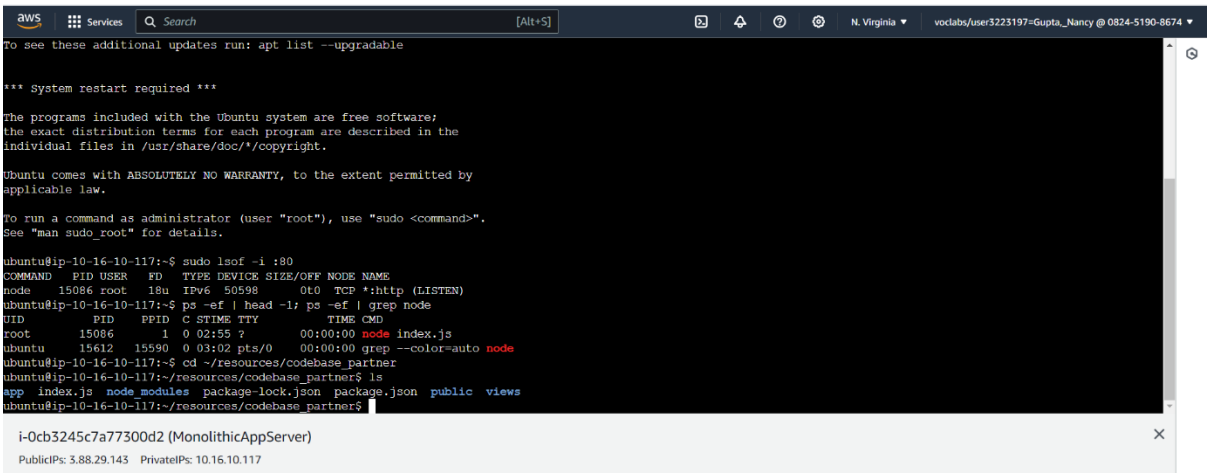
In the command output, it shows that there are two node processes running on the EC2 instance:

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.





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.

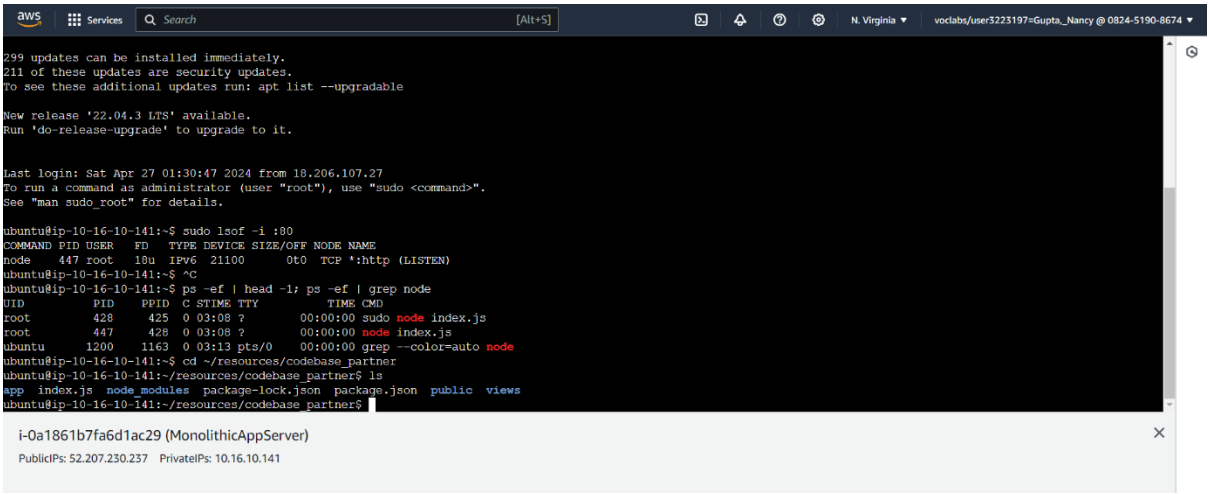
   Questions for thought: Based on what you have observed, what can you determine about how and where this node application is running? How do you think it was installed? What prerequisite libraries, if any, were required to make it run? Where does the application store data?

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

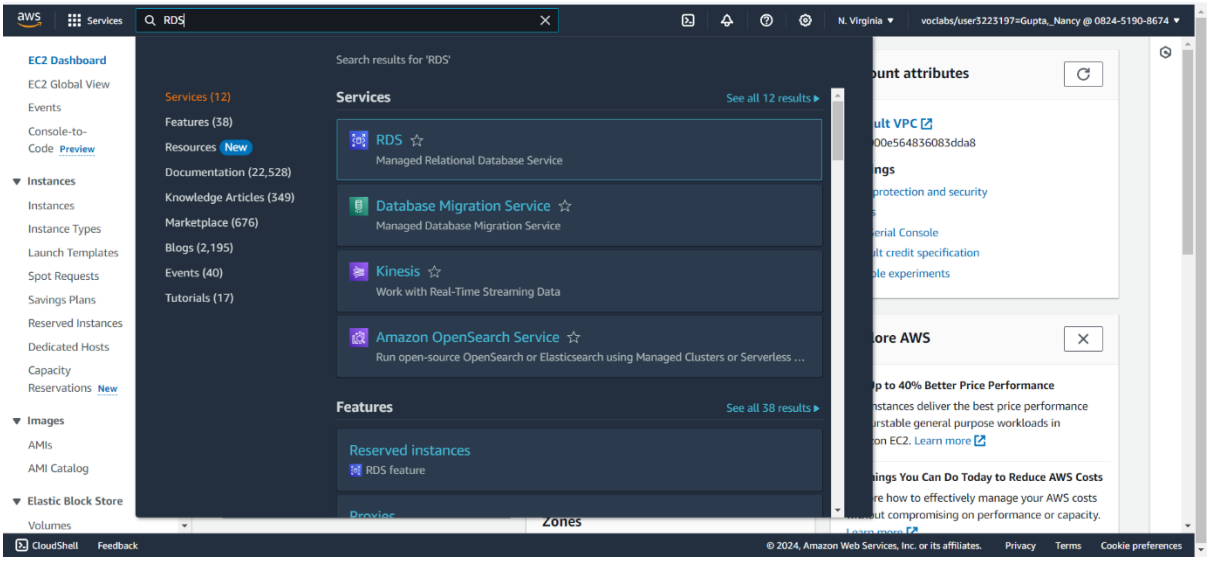
1. Location of Node Application: The node application is located in the directory ~/resources/codebase_partner.
2. Application Structure: Within the codebase_partner directory, there are several files and directories:
   - app: This directory likely contains additional application-specific logic or modules.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

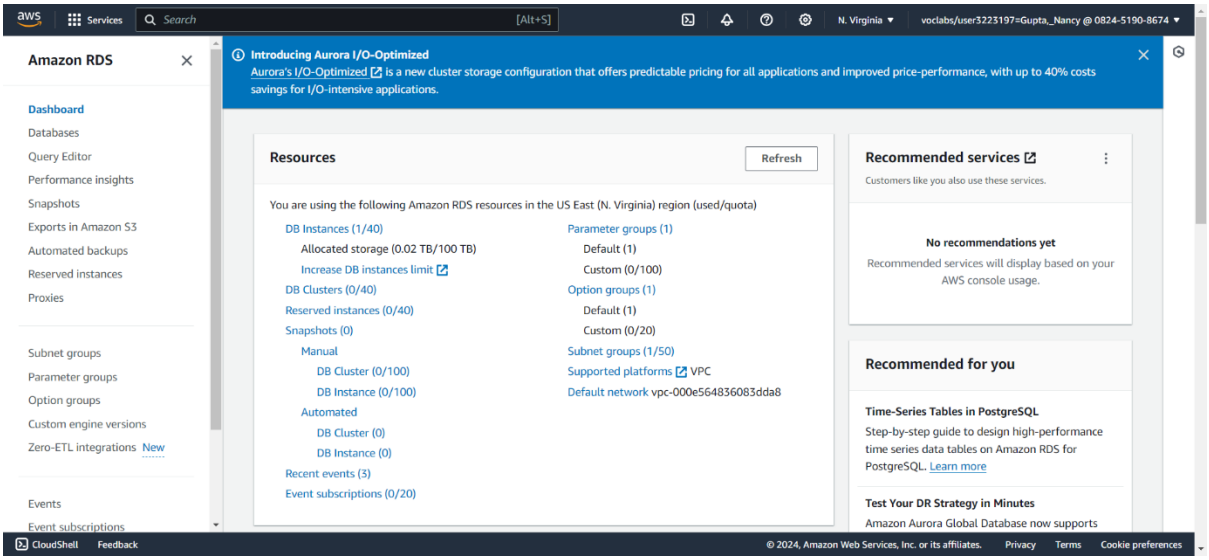
- node_modules: This directory stores all the external dependencies or libraries required by the application.
- package-lock.json and package.json: These files are related to Node.js package management. package.json typically contains metadata about the project and its dependencies, while package-lock.json 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.

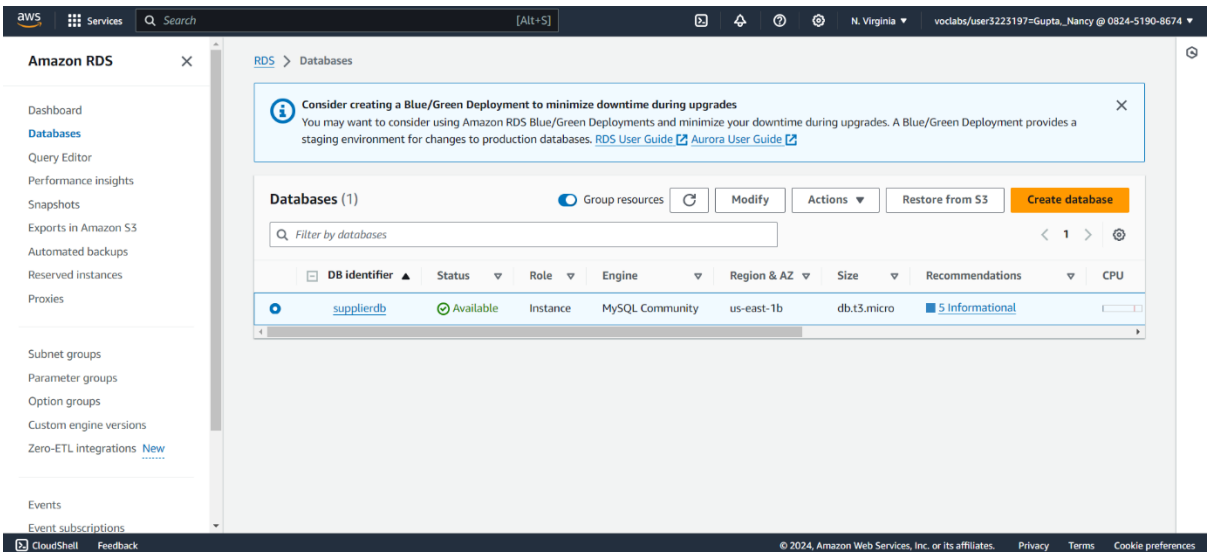
3. How it was Installed: The application was likely installed using Node.js package management tools such as npm (Node Package Manager) or yarn. These tools would have been used to install the dependencies listed in package.json from the npm registry or other sources.
4. Prerequisite Libraries: The application 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.

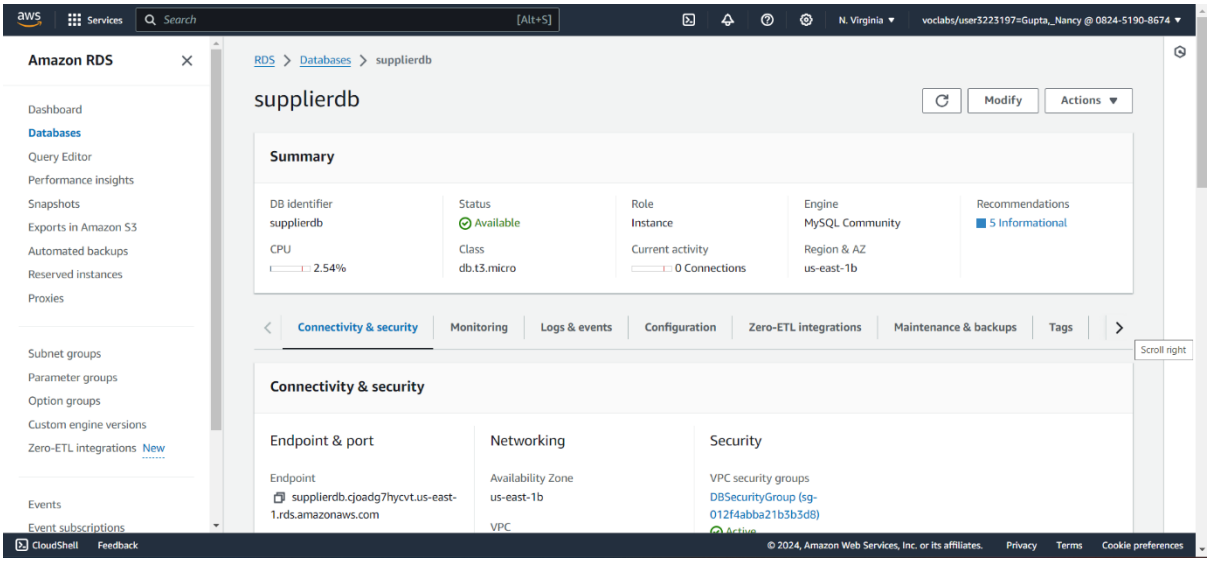


4. **Connect a MySQL client to the RDS database that the node application stores data in.**

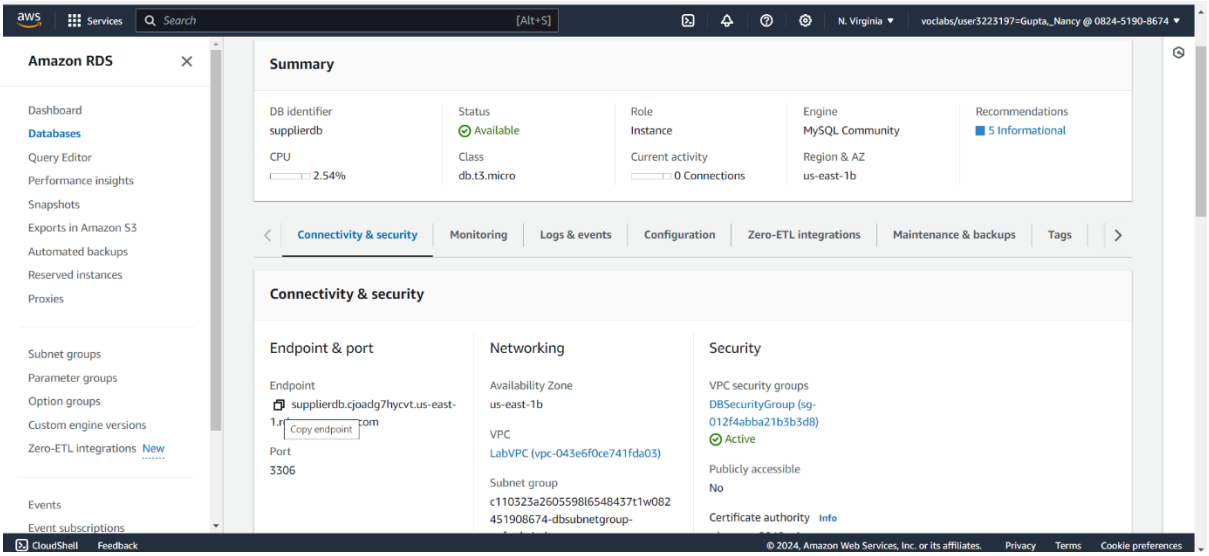

aws
Services
RDS
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Search results for 'RDS'
Services (12)
Features (38)
Resources New
Documentation (22,528)
Knowledge Articles (349)
Marketplace (676)
Blogs (2,195)
Events (40)
Tutorials (17)
Services
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
See all 38 results
Reserved instances
RDS feature
Zones
unt attributes
ult VPC
00e564836083dda8
ngs
protection and security
erial Console
lt credit specification
le experiments
ore AWS
p to 40% Better Price Performance
nstances deliver the best price performance
rstable general purpose workloads in
on EC2. Learn more
ings You Can Do Today to Reduce AWS Costs
re how to effectively manage your AWS costs
ut compromising on performance or capacity.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
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
Introducing Aurora I/O-Optimized
Aurora's I/O-Optimized is a new cluster storage configuration that offers predictable pricing for all applications and improved price-performance, with up to 40% costs savings for I/O-intensive applications.
Resources
Refresh
You are using the following Amazon RDS resources in the US East (N. Virginia) region (used/quota)
DB Instances (1/40)
Allocated storage (0.02 TB/100 TB)
Increase DB instances limit
DB Clusters (0/40)
Reserved instances (0/40)
Snapshots (0)
Manual
DB Cluster (0/100)
DB Instance (0/100)
Automated
DB Cluster (0)
DB Instance (0)
Recent events (3)
Event subscriptions (0/20)
Parameter groups (1)
Default (1)
Custom (0/100)
Option groups (1)
Default (1)
Custom (0/20)
Subnet groups (1/50)
Supported platforms VPC
Default network vpc-000e564836083dda8
Recommended services
Customers like you also use these services.
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
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

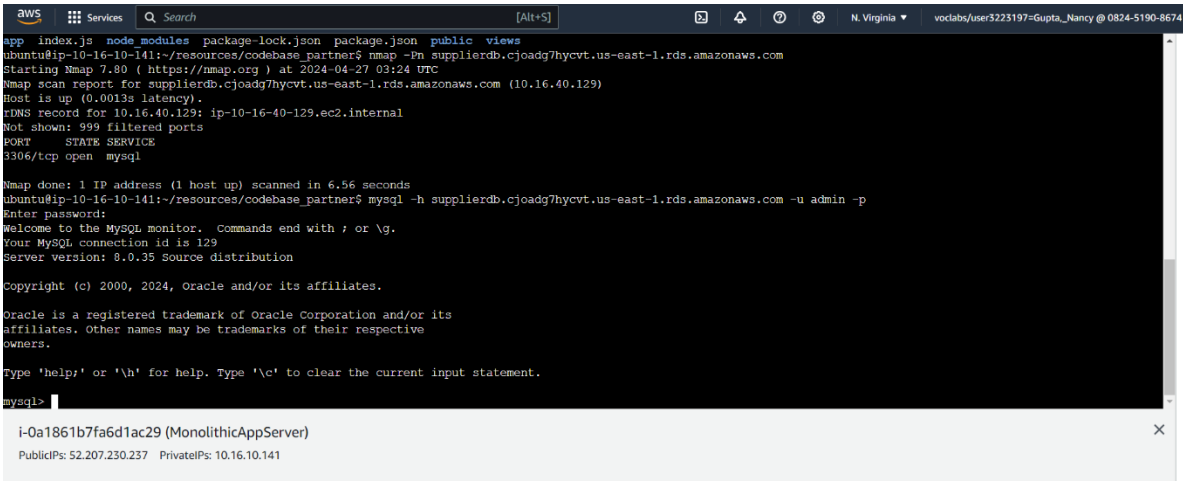

aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Amazon RDS
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Events
Event subscriptions
RDS > Databases
Consider creating a Blue/Green Deployment to minimize downtime during upgrades
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. RDS User Guide Aurora User Guide
Databases (1)
Group resources
Modify
Actions
Restore from S3
Create database
Filter by databases
1
DB identifier
Status
Role
Engine
Region & AZ
Size
Recommendations
CPU
supplierdb
Available
Instance
MySQL Community
us-east-1b
db.t3.micro
5 Informational
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



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



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

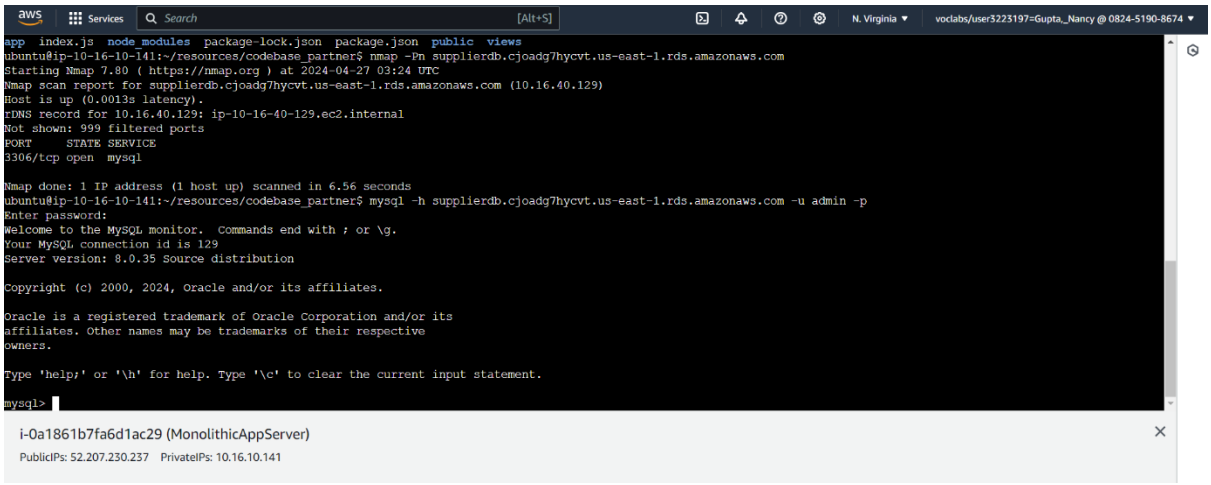


- **To connect to the database, use the MySQL client that is already installed on the *MonolithicAppServer* instance. Use the following values:**
    - **Username: admin**
    - **Password: lab-password**

  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



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*.**

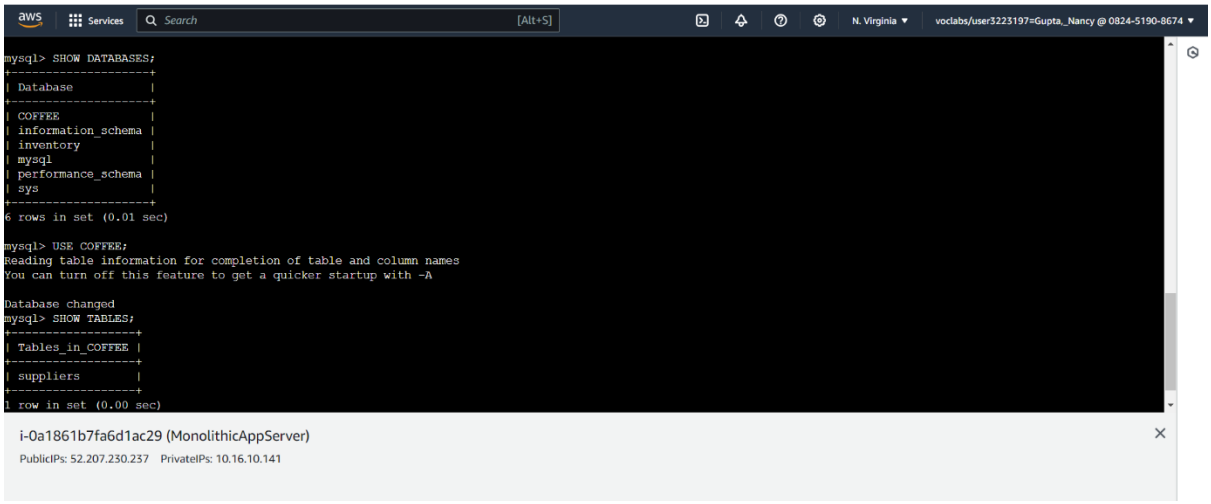
This table contains the supplier entry or entries that you added earlier when you tested the web application.

Use the following SQL commands:

SHOW DATABASES;

USE COFFEE;

SHOW TABLES;

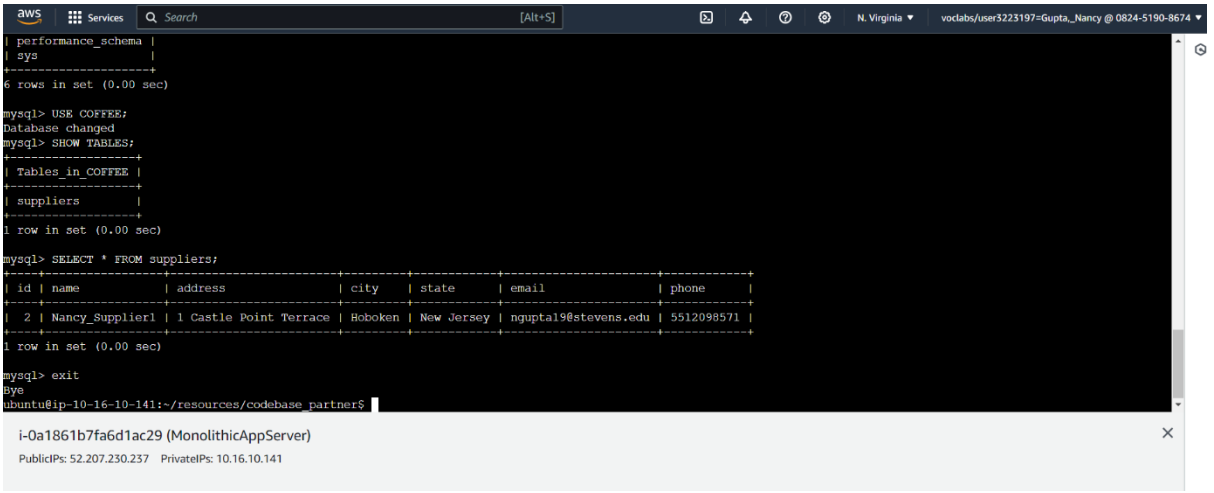


If the COFFEE database and suppliers table exist, you can view the data in the suppliers table. Use the following SQL command:

SELECT * FROM suppliers;

- **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.



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.

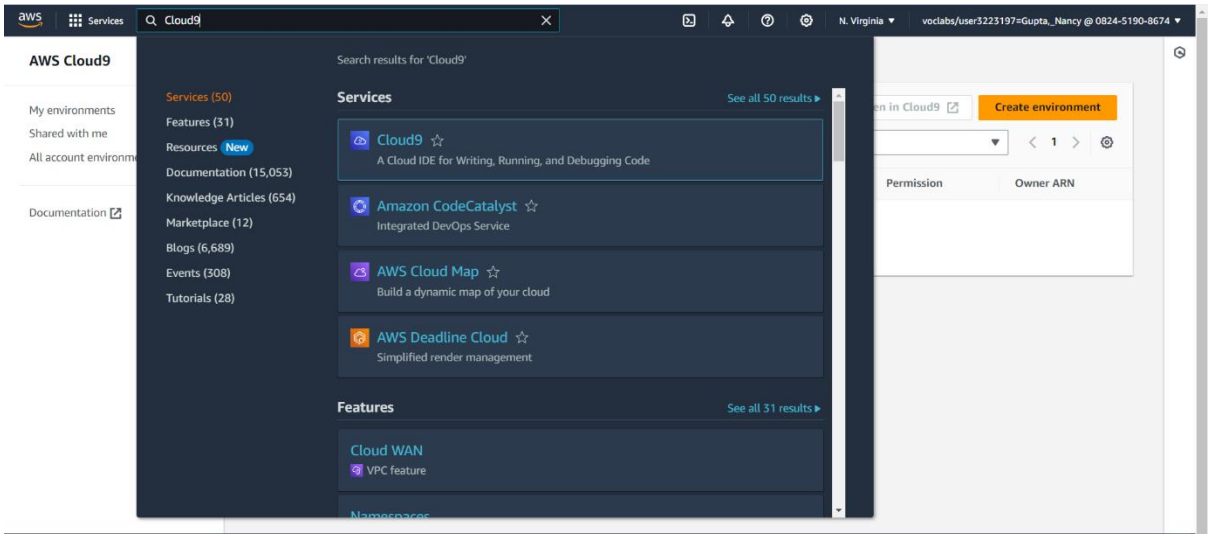
Task 3.1: Create an AWS Cloud9 IDE as your work environment

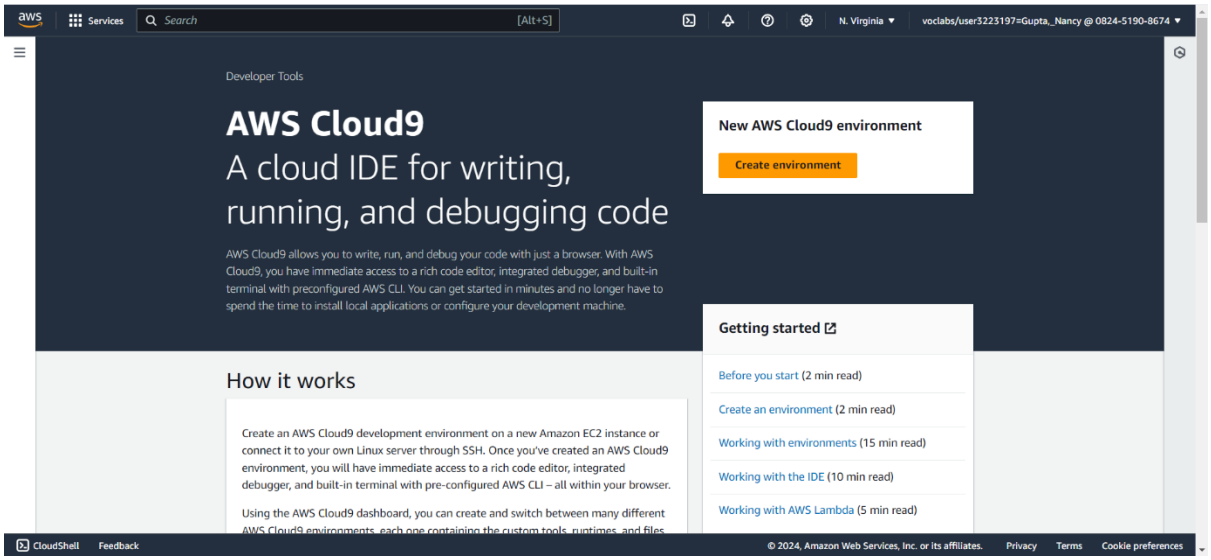
In this task, you will create your development environment. If you aren't familiar with AWS Cloud9, the following references might be helpful:

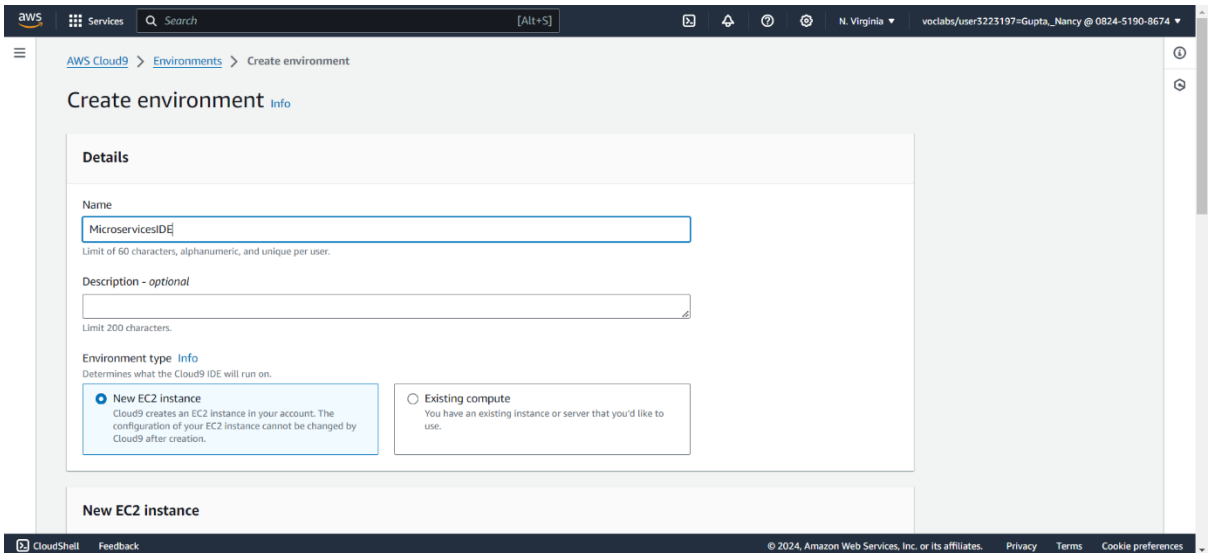
- **Creating an Environment in AWS Cloud9**
- **Tour of the AWS Cloud9 IDE**

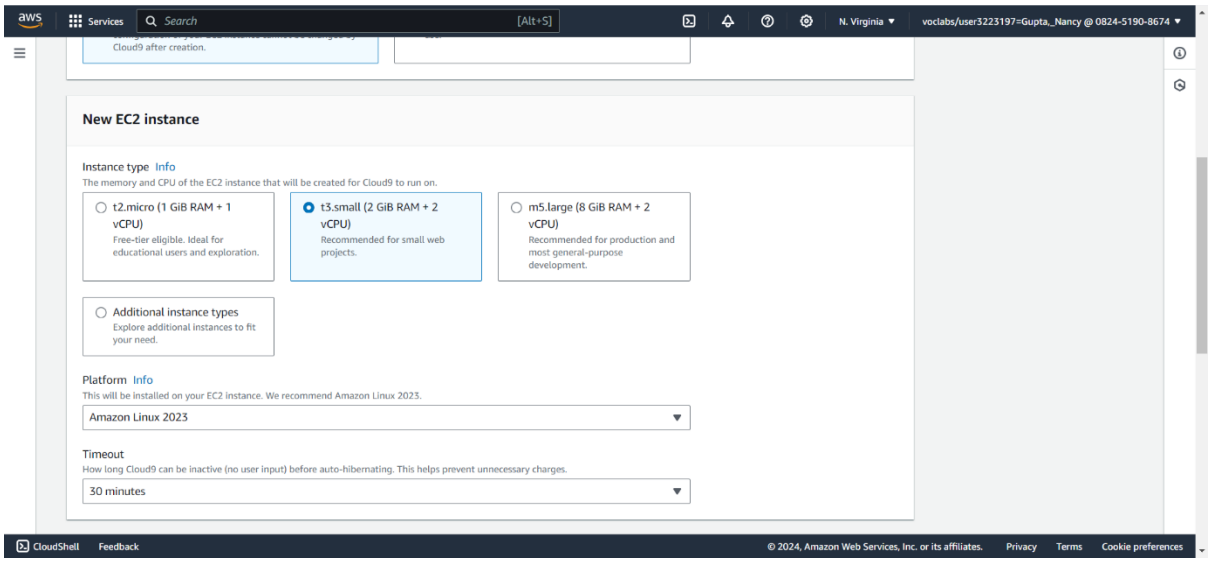
1. **Create an AWS Cloud9 instance that is named MicroservicesIDE and then open the IDE.**

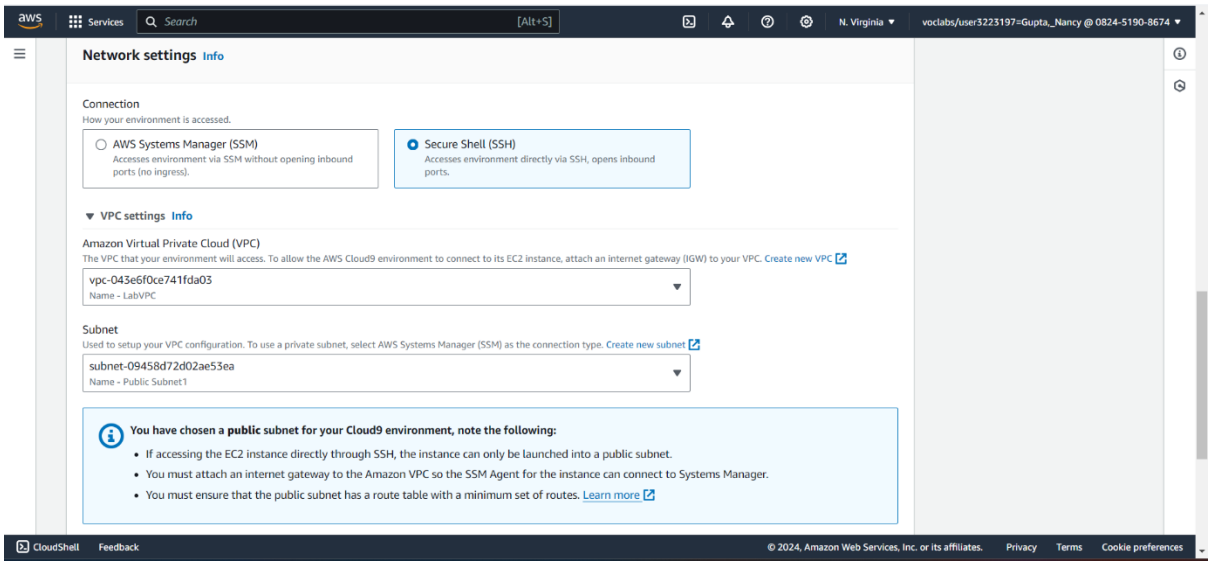
It should run as a new EC2 instance of size *t3.small* and run Amazon Linux 2. The instance should support SSH connections and run in the *LabVPC* in *Public Subnet1*.

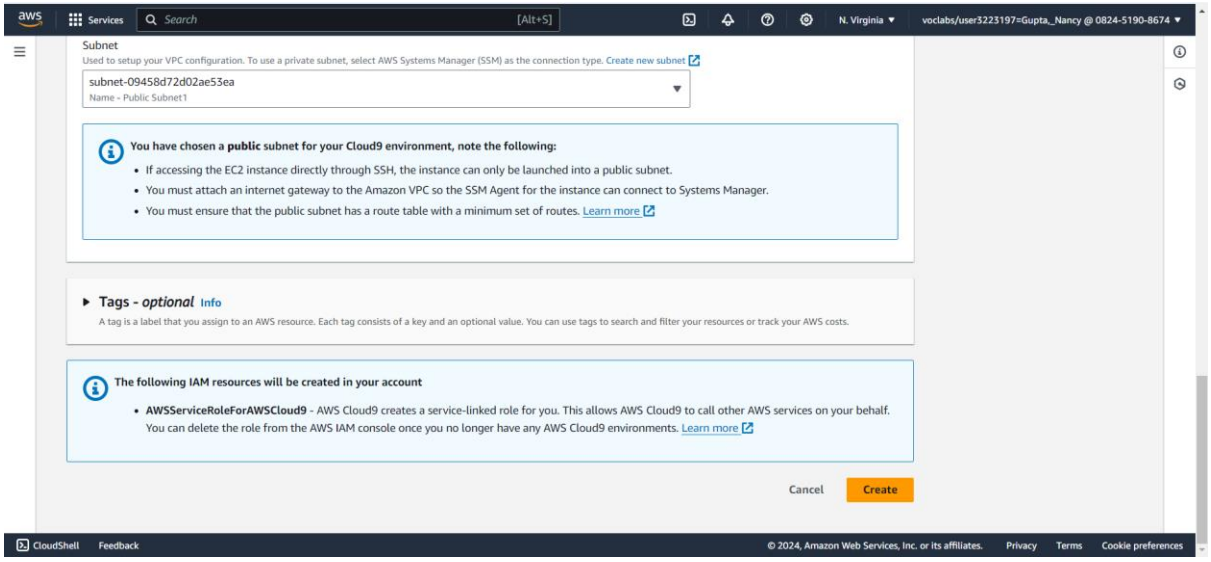


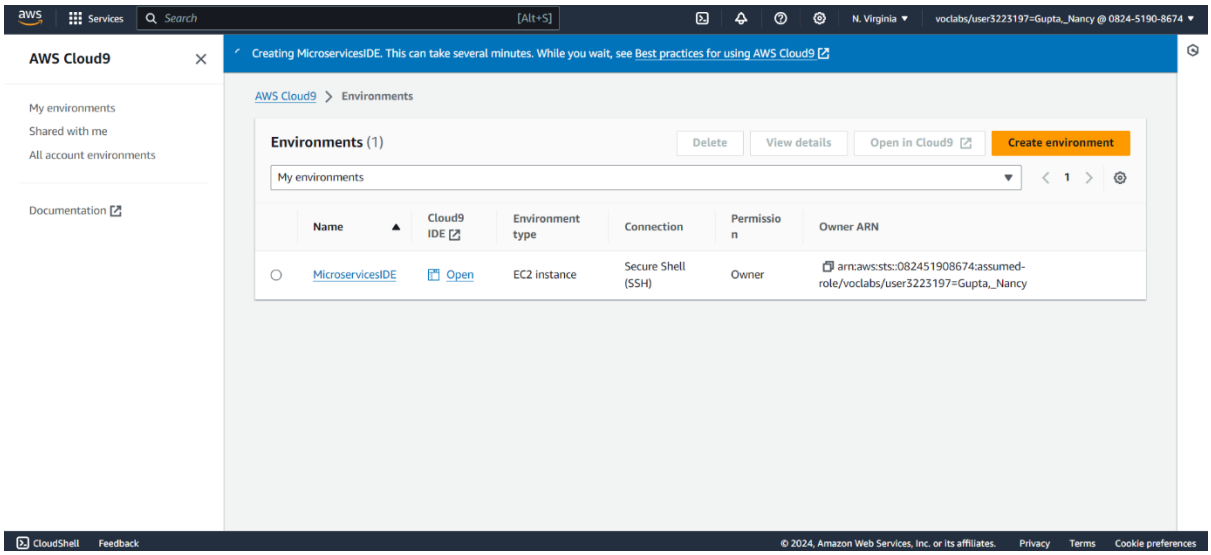

Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Developer Tools
AWS Cloud9
A cloud IDE for writing, running, and debugging code
AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.
New AWS Cloud9 environment
Create environment
Getting started
Before you start (2 min read)
Create an environment (2 min read)
Working with environments (15 min read)
Working with the IDE (10 min read)
Working with AWS Lambda (5 min read)
How it works
Create an AWS Cloud9 development environment on a new Amazon EC2 instance or connect it to your own Linux server through SSH. Once you've created an AWS Cloud9 environment, you will have immediate access to a rich code editor, integrated debugger, and built-in terminal with pre-configured AWS CLI – all within your browser.
Using the AWS Cloud9 dashboard, you can create and switch between many different
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

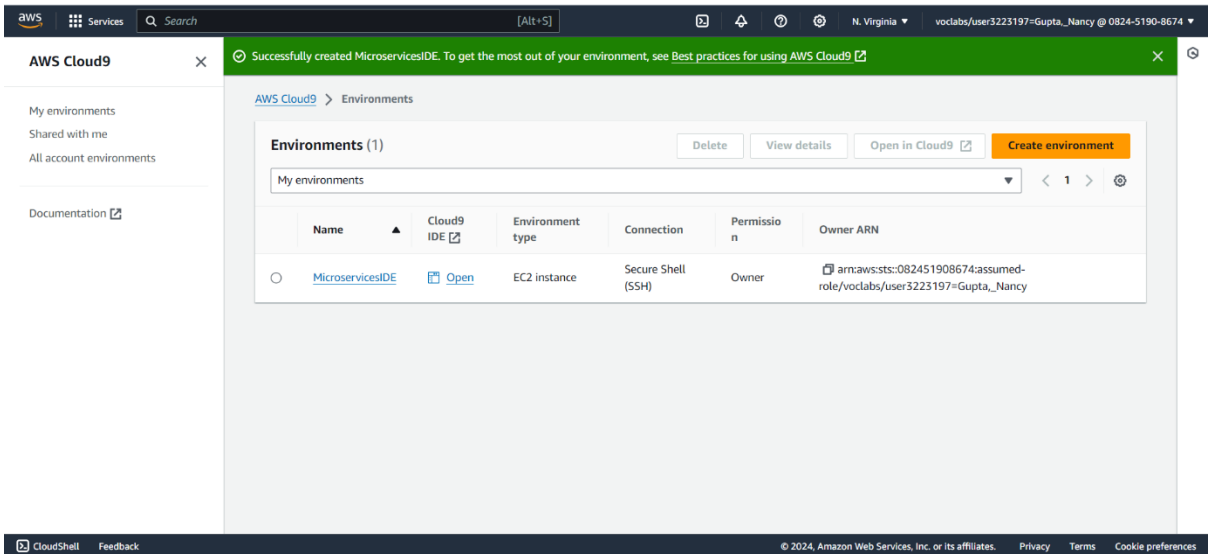

Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
AWS Cloud9 > Environments > Create environment
Create environment Info
Details
Name
MicroservicesIDE
Limit of 60 characters, alphanumeric, and unique per user.
Description - optional
Limit 200 characters.
Environment type Info
Determines what the Cloud9 IDE will run on.
New EC2 instance
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.
Existing compute
You have an existing instance or server that you'd like to use.
New EC2 instance
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

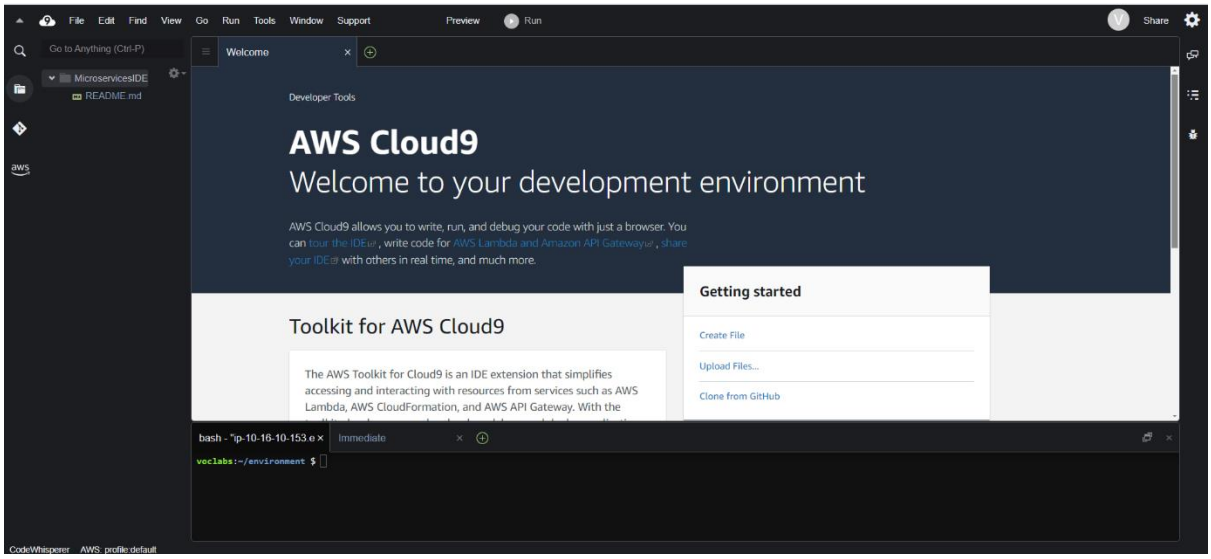

Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Cloud9 after creation.
New EC2 instance
Instance type Info
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.
t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.
t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.
m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.
Additional instance types
Explore additional instances to fit your need.
Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.
Amazon Linux 2023
Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
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 Subnet1
You have chosen a public subnet for your Cloud9 environment, note the following:
If accessing the EC2 instance directly through SSH, the instance can only be launched into a public subnet.
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


Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
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 Subnet1
You have chosen a public subnet for your Cloud9 environment, note the following:
If accessing the EC2 instance directly through SSH, the instance can only be launched into a public subnet.
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


Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
AWS Cloud9
My environments
Shared with me
All account environments
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
My environments
Name
Cloud9 IDE
Environment type
Connection
Permission
Owner ARN
MicroservicesIDE
Open
EC2 instance
Secure Shell (SSH)
Owner
arn:aws:sts::082451908674:assumed-role/voclabs/user3223197=Gupta,_Nancy
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

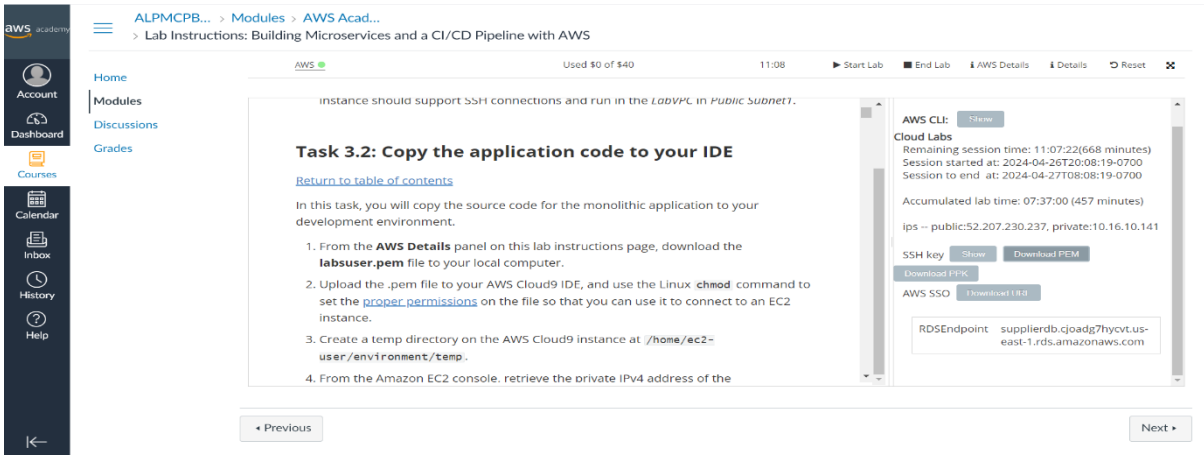


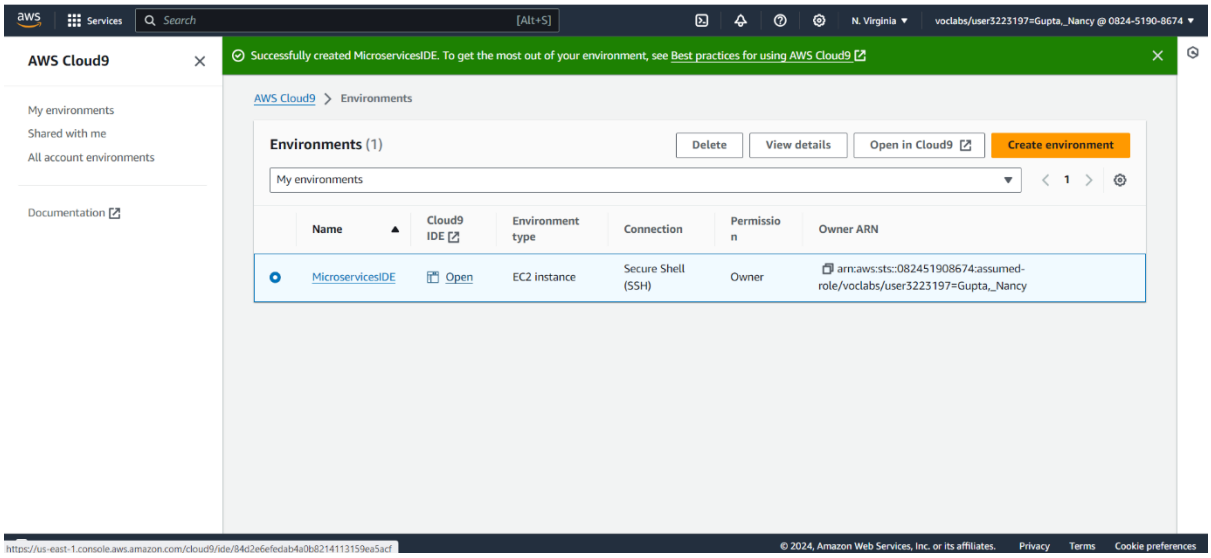


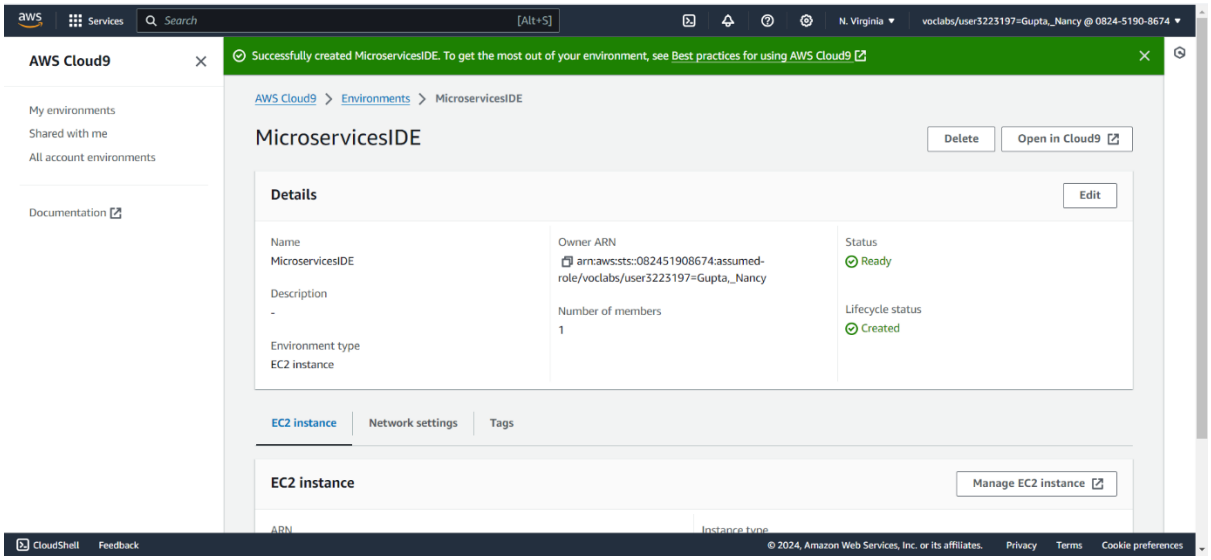
Task 3.2: Copy the application code to your IDE

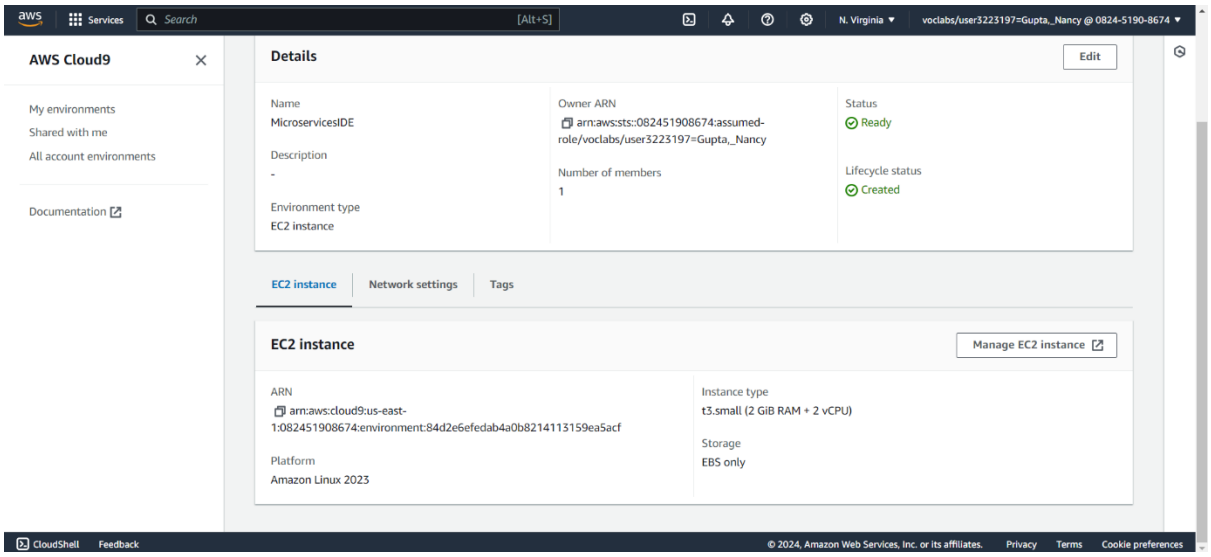
In this task, you will copy the source code for the monolithic application to your development environment.

1. **From the AWS Details panel on this lab instructions page, download the labsuser.pem file to your local computer.**

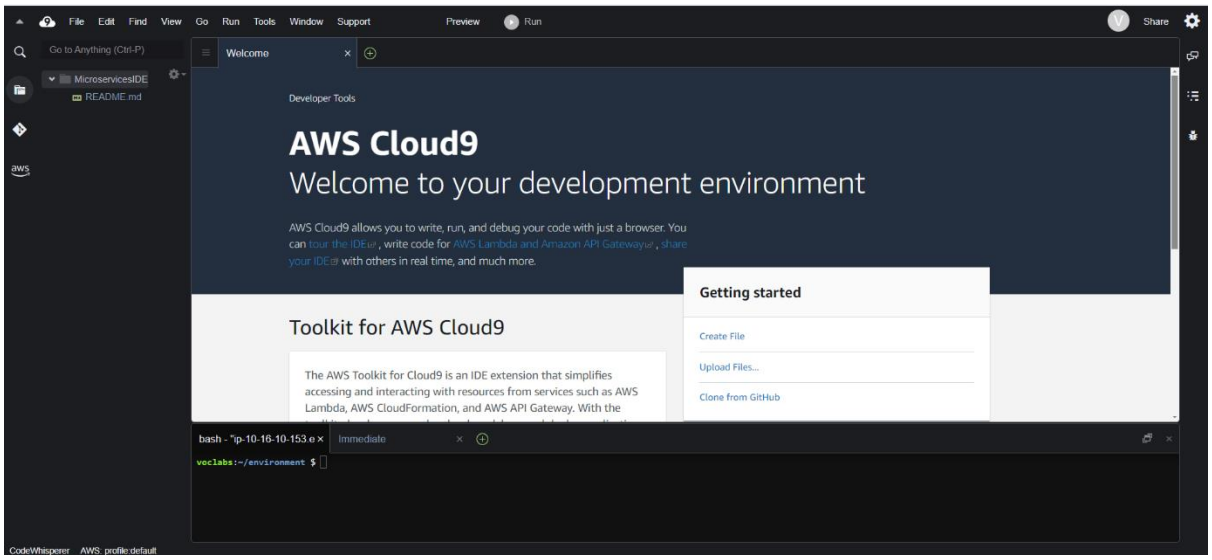


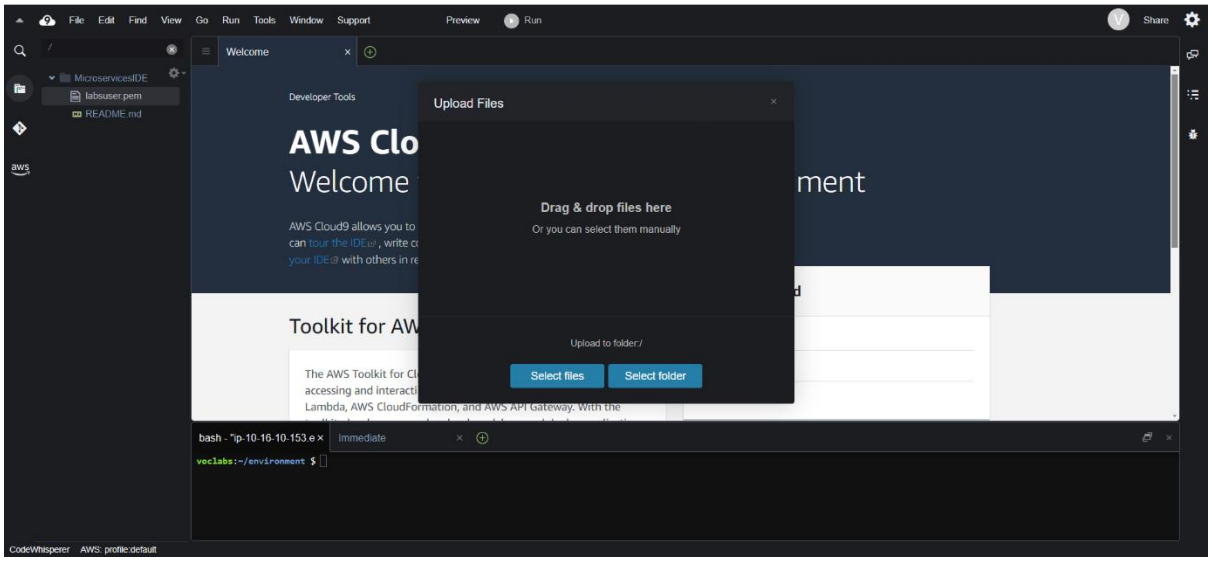


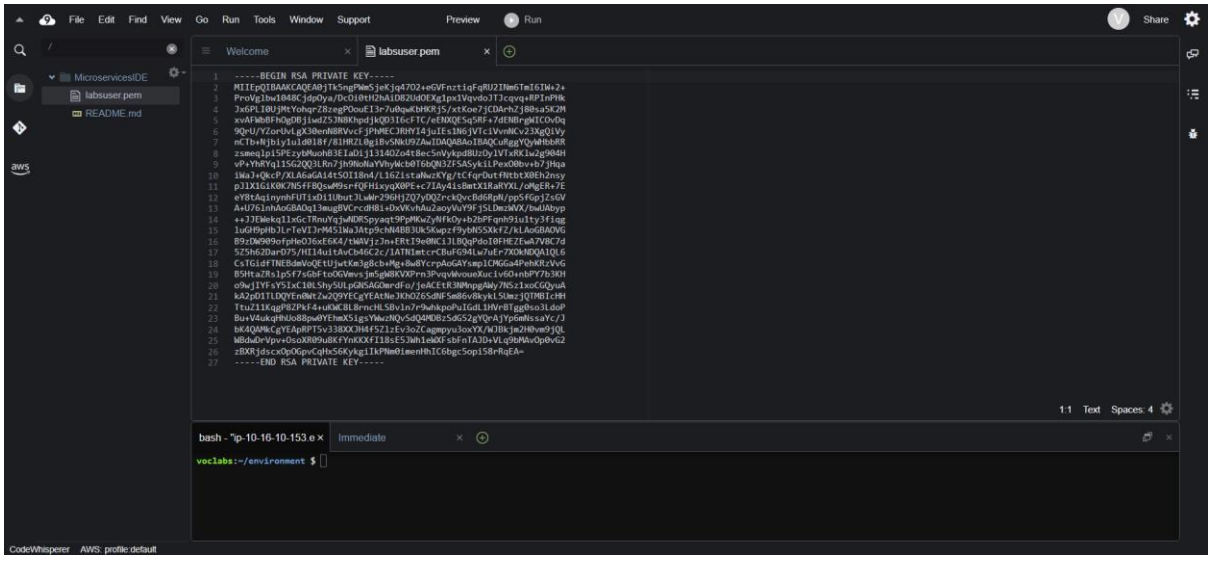


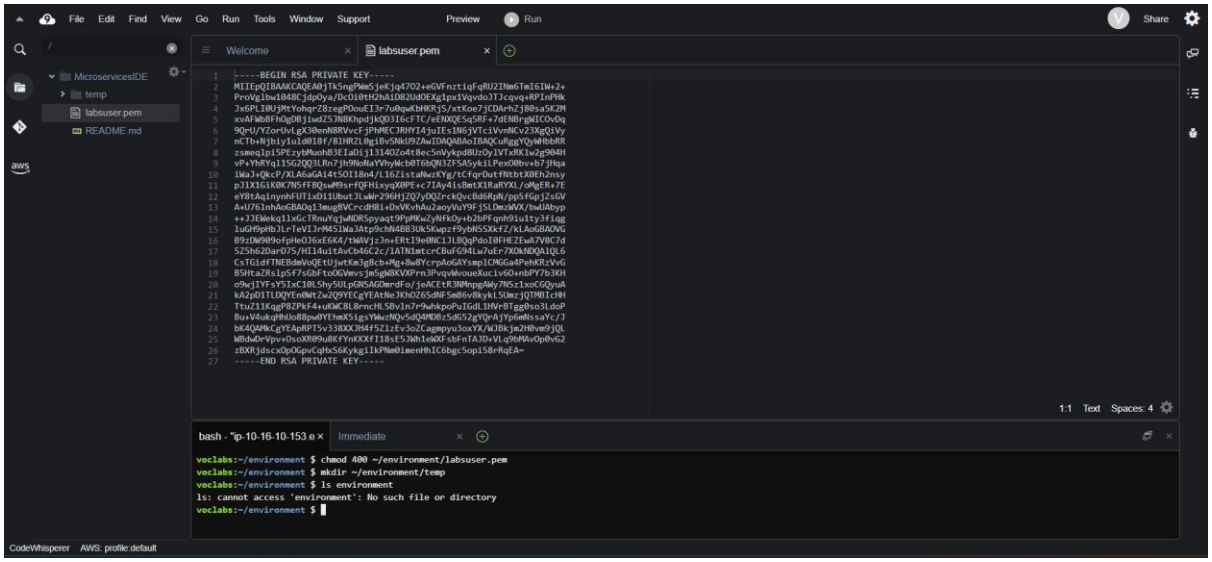


2. **Upload the .pem file to your AWS Cloud9 IDE, and use the Linux chmod command to set the proper permissions on the file so that you can use it to connect to an EC2 instance.**

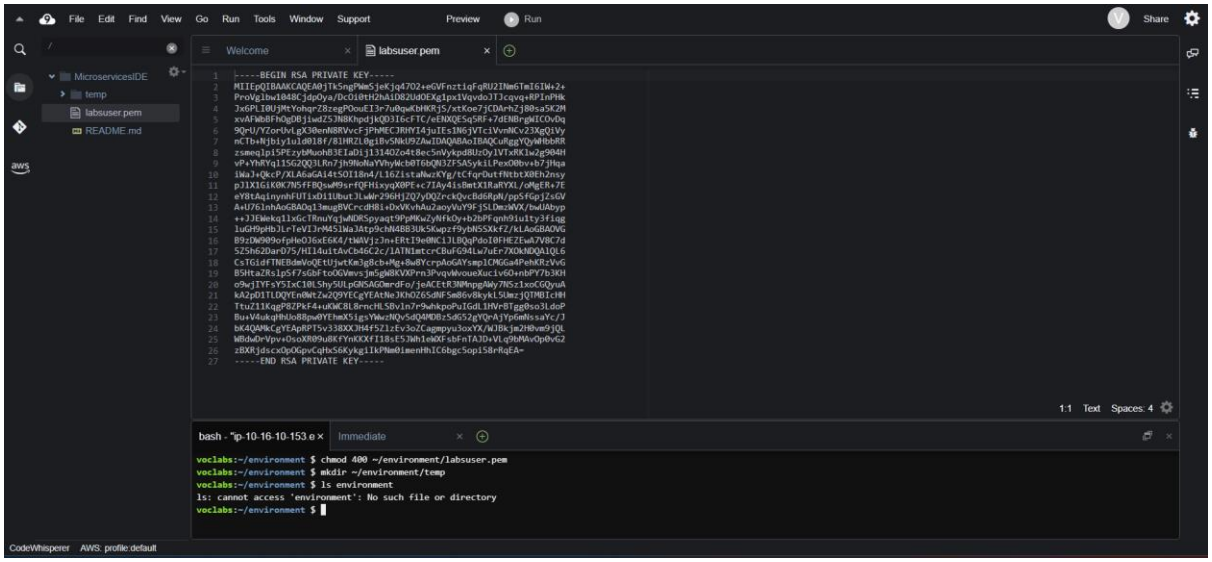

File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
MicroservicesIDE
README.md
Welcome
Developer Tools
AWS Cloud9
Welcome to your development environment
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.
Getting started
Create File
Upload Files...
Clone from GitHub
Toolkit for AWS Cloud9
The AWS Toolkit for Cloud9 is an IDE extension that simplifies accessing and interacting with resources from services such as AWS Lambda, AWS CloudFormation, and AWS API Gateway. With the
bash - "ip-10-16-10-153.e
Immediate
voclabs:~/environment $
CodeWhisperer AWS: profile:default


File Edit Find View Go Run Tools Window Support Preview Run Share
MicroservicesIDE
labsuser.pem
README.md
Welcome
Developer Tools
Upload Files
Drag & drop files here
Or you can select them manually
Upload to folder: /
Select files
Select folder
Toolkit for AW
bash - "ip-10-16-10-153.e
Immediate
voclabs:~/environment $
CodeWhisperer AWS: profile:default

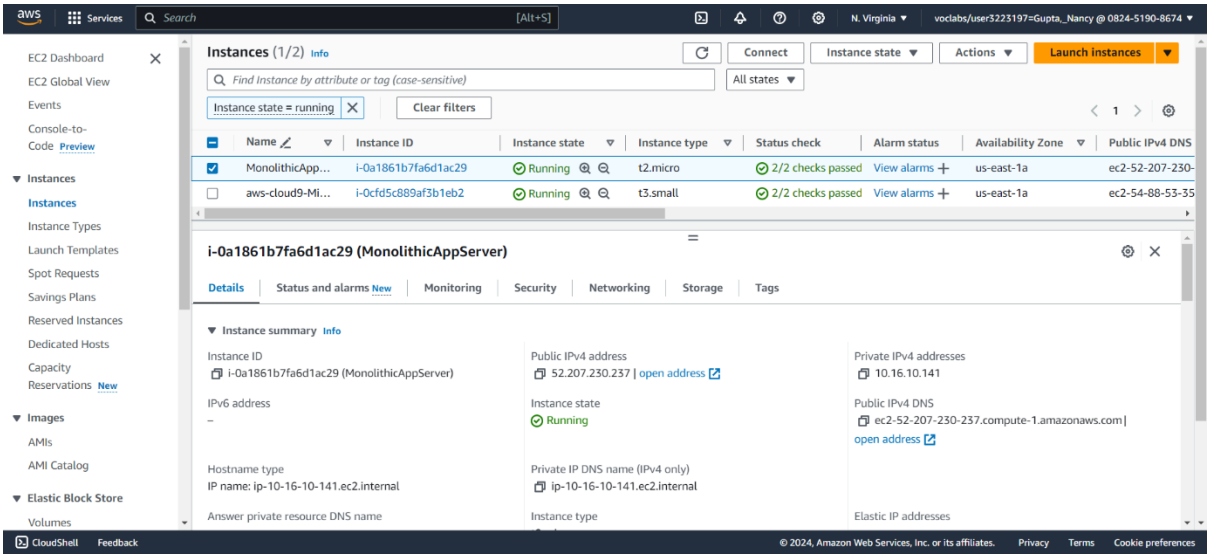

File Edit Find View Go Run Tools Window Support Preview Run Share
MicroservicesIDE
labsuser.pem
README.md
Welcome
labsuser.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA0jTk5ngPWm5jeKjq47O2+eGVFnztiqFqRU2INm6TmI6IW+2+
ProVglbw1048CjdpOya/DcOi0tH2hAiD82UdOEXg1px1VqvdoJTJcqvq+RPInPHk
Jx6PLI0UjMtYohqrZ8zegPOouEI3r7u0qwKbHKRjS/xtKoe7jCDArhZj80sa5K2M
xvAFWbBFhQgOEjiwdZ5JN8KhpdjkQD3I6cFTC/eENXQE5q5RF+7dENBrgWICOvDq
9QrU/YZorUvLgX30enN8RVvcFjPhMECJ0bYIAjuIFs1N6jUTciVvnNCv23XgQ1Vy
nCTb+NjbiylulO18f/81HRZL0g1BvSNkO9ZAwIDAQABAoIBAQCuRggYQyWHbbRR
zsmeqlpi5PEzybMuoHB3EIaDij1314OZo4t8ec5nVykpd8UzOy1VTxRK1w2g904H
vP+YhRYq115G2QQ3LRn7jh9NoNaYVhyWcb0T6bQN3ZFSASykiLPexO0bv+b7jHqa
iWaJ+QkcP/XLA6aGAi4tSOI18n4/L16ZistaNwzKYg/tCfqrOutfNtbtX0Eh2nsy
p1lX1GiK0K7N5fFBQsaM9sr+fQFHixyqX0PE+c7IAy4isBmtX1RaRYXL/oMgERx7E
eY8tAqInynhFUTixDi1UbutJLwWr296HjZQ7yDQZrckQvcBdGRpN/pp5fGpjZsGV
A+U761nhAoGBAOqi3mugBVCrcdH8i+DxVKvhAu2aoyVuY9FjSLDmzWVX/bwUAbyp
++JIEWekq1lxGcTRnuYqjwNDRSpyaqt9PpMKwZyNfkOy+b2bPFqnh9iu1ty3figg
luGF9phbJLrTeVIJrM45lWa3Atp9chN4BB3Uk5WupzF9ybN5SkFZ/kLAoGBAOVG
B9zDW989ofpHeO36xE6X4/tWAVijJn+ERtI9e08KC3LBQpFdoI8FHEZEwA7VRC7d
5Z5h62DarD75/HIl4uitAvCb46C2c/1ATN1mtcrCBuFG94Lw7uEr7XOkNDQAlQL6
CsTGidFTNEBdmVoQEtUjwtKm3g8cb+Mg+8w8YcrpAoGAYsmp1CMGGa4PehKRzVvG
85HtaZRs1p5f7sGbFtoOGVmvsjm5gN8KVXPrn3PvqvWvoueXuciv6OnhPY7b3KH
oRwjIYF+Y5ICl0L5hy5ULgGR5AGDmrdFo/jeACEtR3NHqgNWy7N5z1xuCGQyuA
kA2pD1TLDQYEn0WtZw2Q9YECgYEAtNeJKhOZ6SdNFSm86v80kykL5UmzjQTMBIcHH
TtuZ11KqgP8ZPkF4+uKWC8L8rncHL5Bv1n7r9whkpoPuIGdL1HVrBTgg0so3LdoP
Bu+V4ukq9hUo88pw0YEhmX5igsYWwzNQv5dQ4MDBzSd6S2gYQrAjYp6mNssaYc/J
bK4QANkCgYEApRPT5v338OOJH4f5Z1zEv3oZCagmpyu3oxYX/WJBkjm2H0vm9jQL
WBdwDrVpvxOsoXR09u8KfYoKXXfI18sE53Wh1eWXF+bbFnTAJOvVLgRbMAvOp0vG2
zBXRjdscxOpOGpvCqHxSGKykgiIkPNm0imenHhIC6bgc5op158rRqEA=
-----END RSA PRIVATE KEY-----
1:1 Text Spaces: 4
bash - "ip-10-16-10-153.e
Immediate
voclabs:~/environment $
CodeWhisperer AWS: profile:default

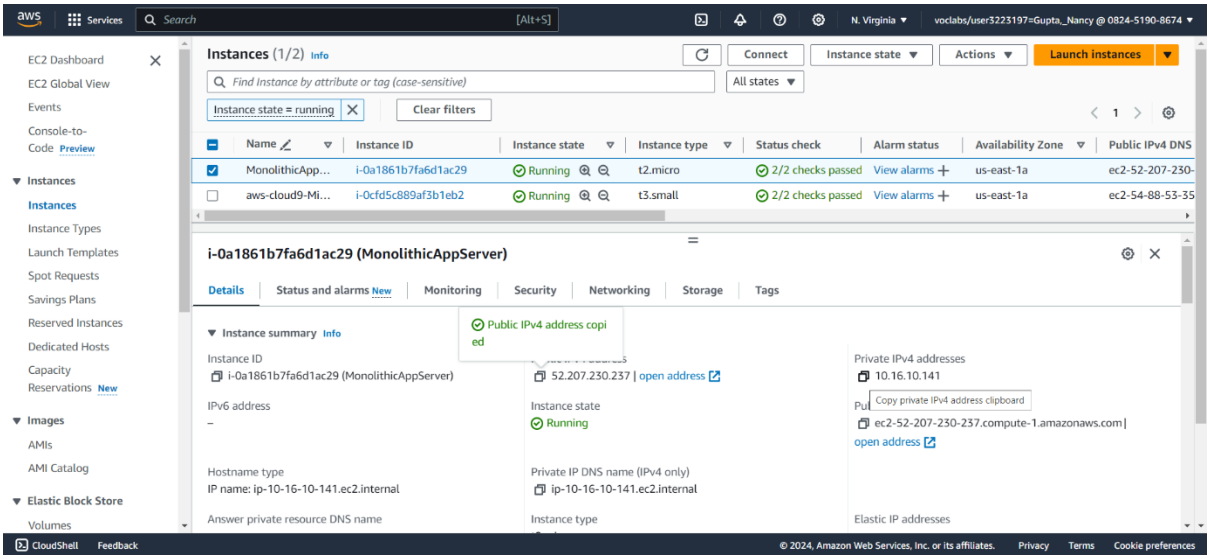


3. Create a temp directory on the AWS Cloud9 instance at /home/ec2-user/environment/temp.



4. From the Amazon EC2 console, retrieve the private IPv4 address of the *MonolithicAppServer* instance.



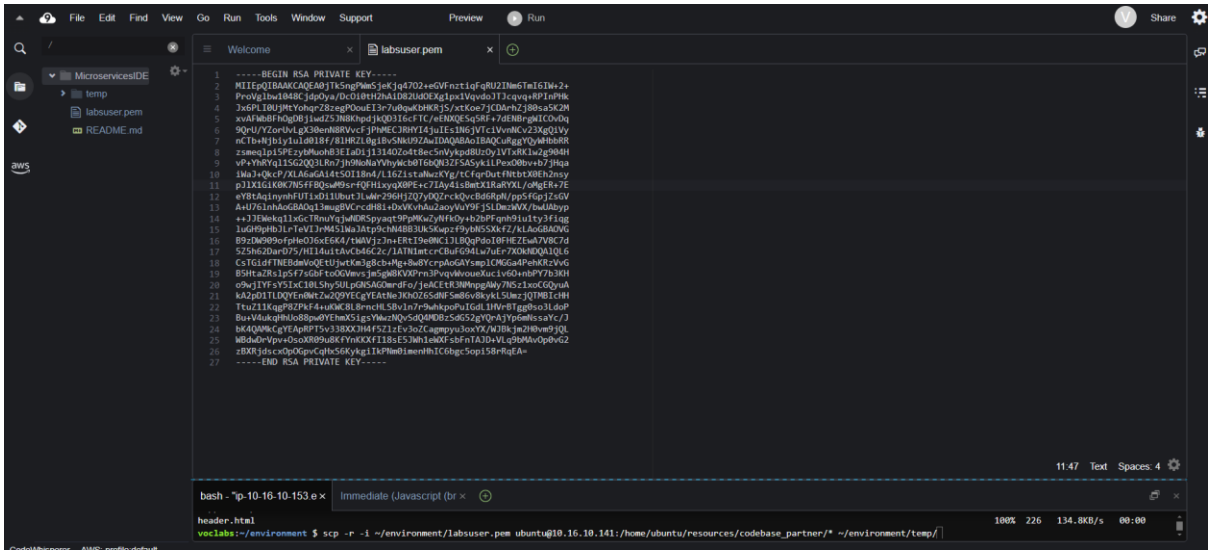


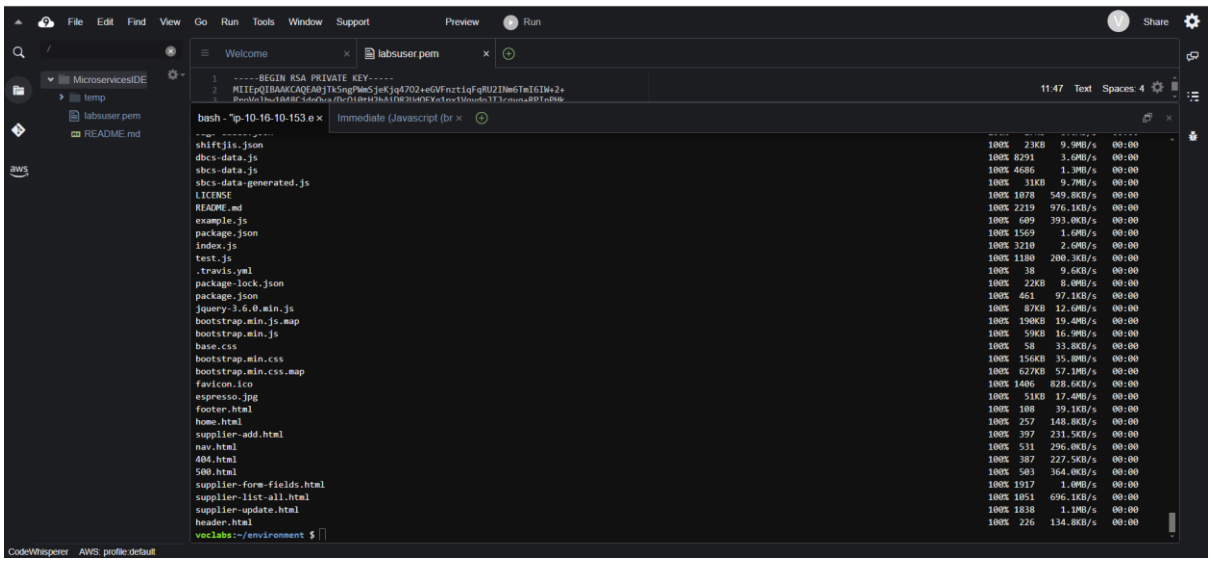
5. **Use the Linux scp command in the Bash terminal on the AWS Cloud9 instance to copy the source code for the node application from the *MonolithicAppServer* instance to the temp directory that you created on the AWS Cloud9 instance.**

The following snippet provides an example scp command:

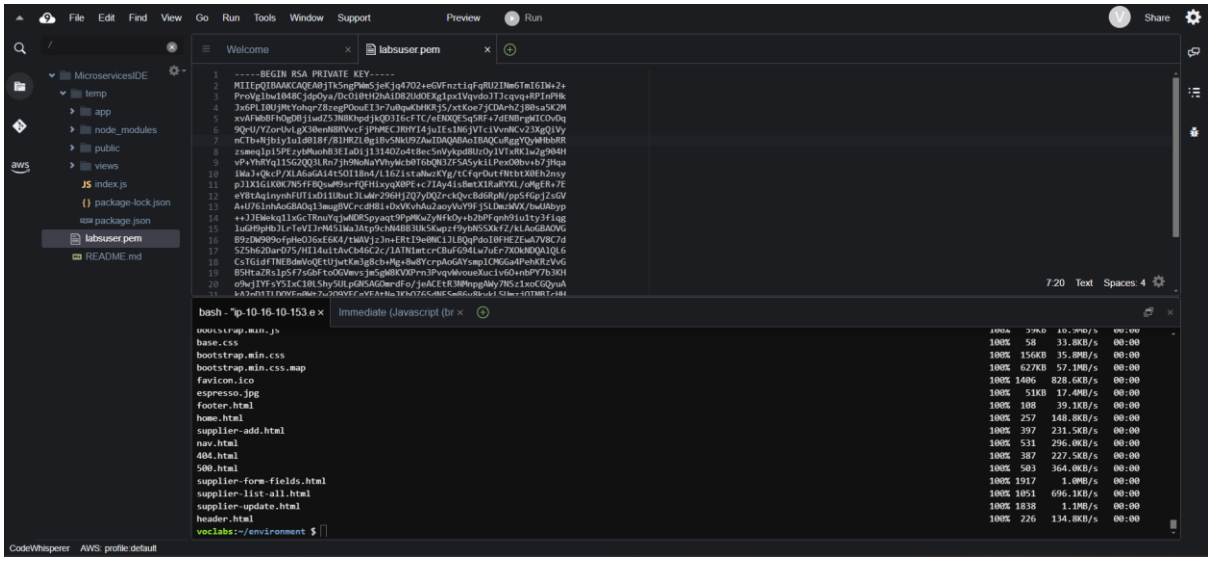
Scp -r -i ~/environment/labsuser.pem ubuntu@$appServerPrivIp:/home/ubuntu/resources/codebase_partner/* ~/environment/temp/

Here I have replaced appServerPrivIp with Private IPV4 of the MonolithicAppServer





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.**



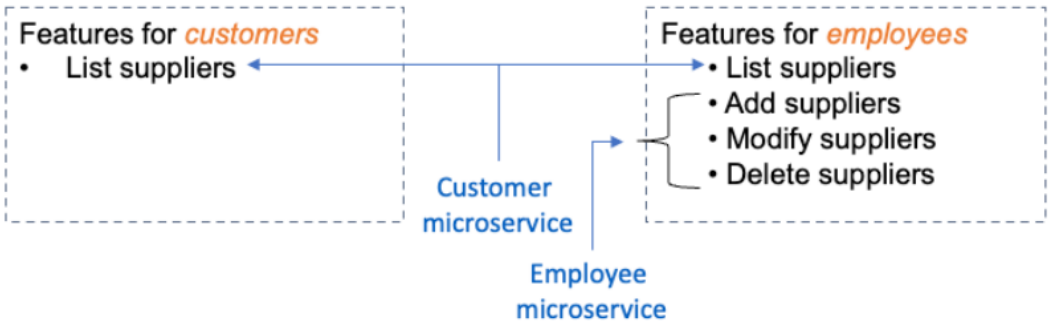
Task 3.3: Create working directories with starter code for the two microservices

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*.

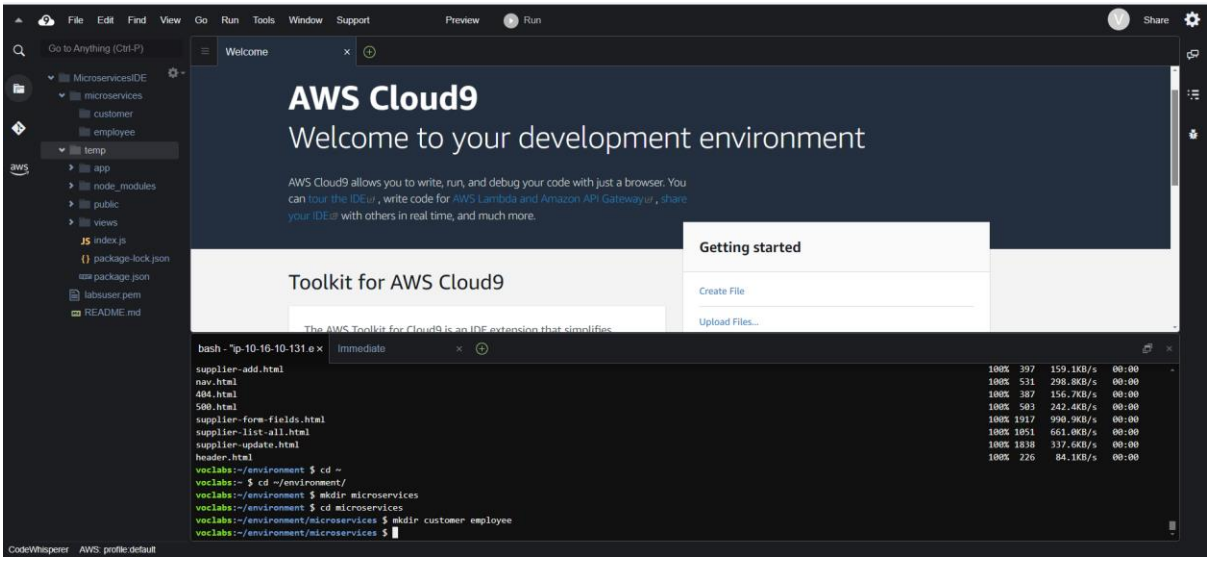
The following table explains the functionality that is needed for each microservice.

Primary User	Microservice Functionality	Access Level
Customer	**The *customer* 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 *customers* of the application.**	**Read-only**
Employee	**The *employee* 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**

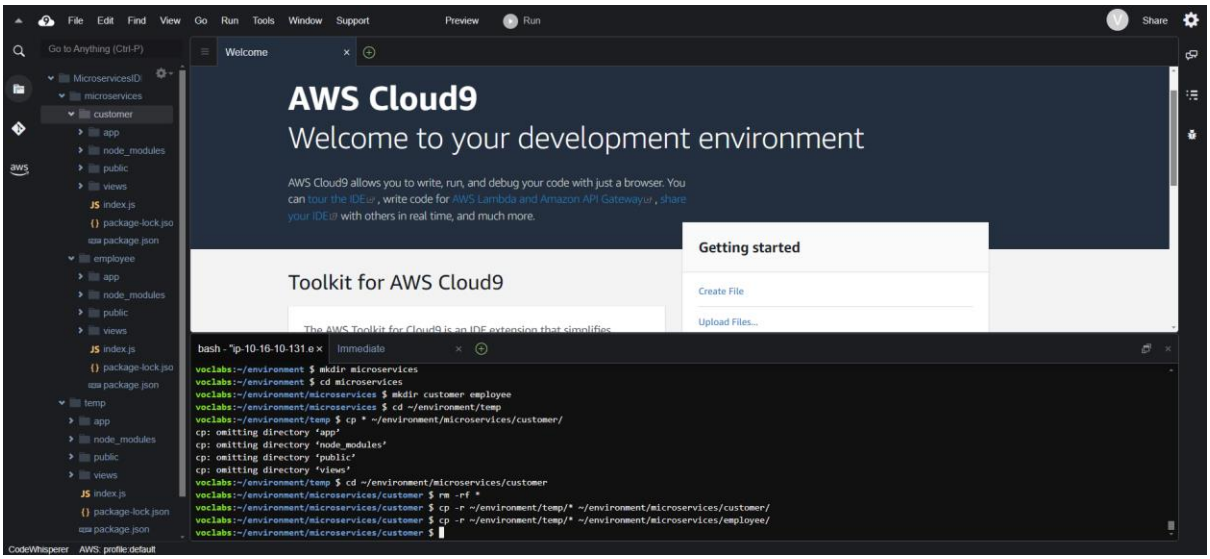


The employee microservice will eventually be made available only to employees. You will accomplish this by first encapsulating them as a separate microservice (in phases 3 and 4 of the project), and then later in phase 9 of the project, you will limit who can access the employee microservice.

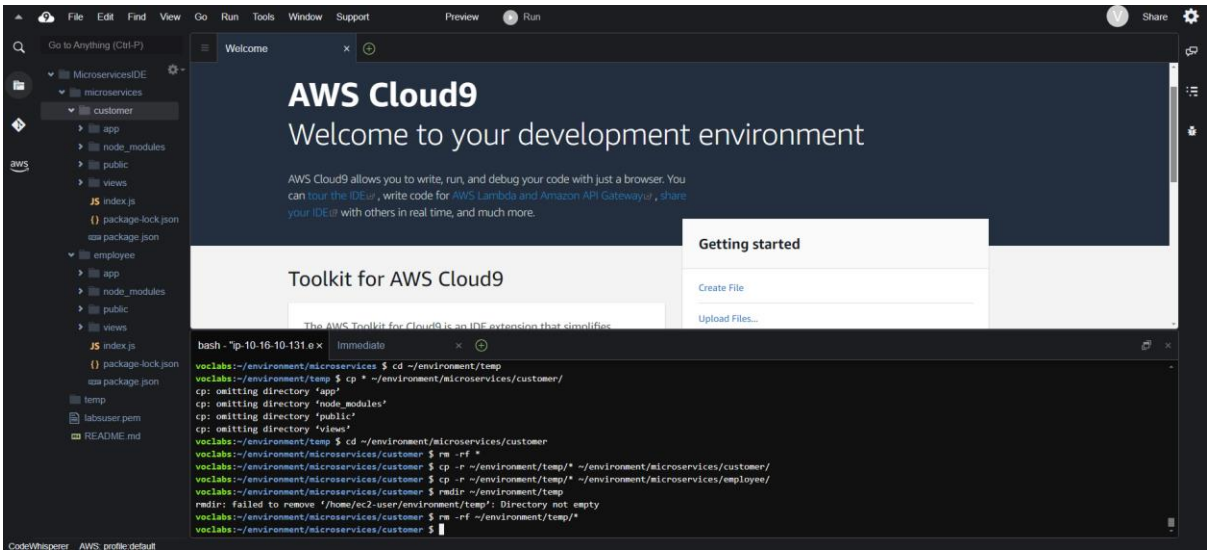
1. **In the microservices directory, create two new directories that are named customer and employee.**

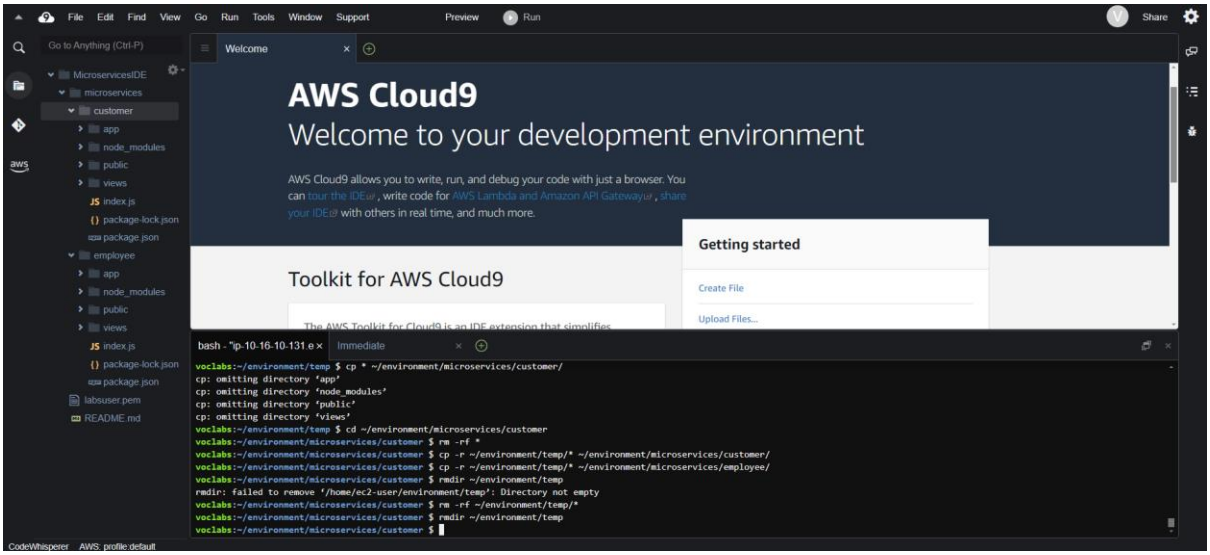


2. **Place a copy of the source code for the monolithic application in each new directory, and remove the files from the temp directory.**



3. **Delete the empty temp directory.**



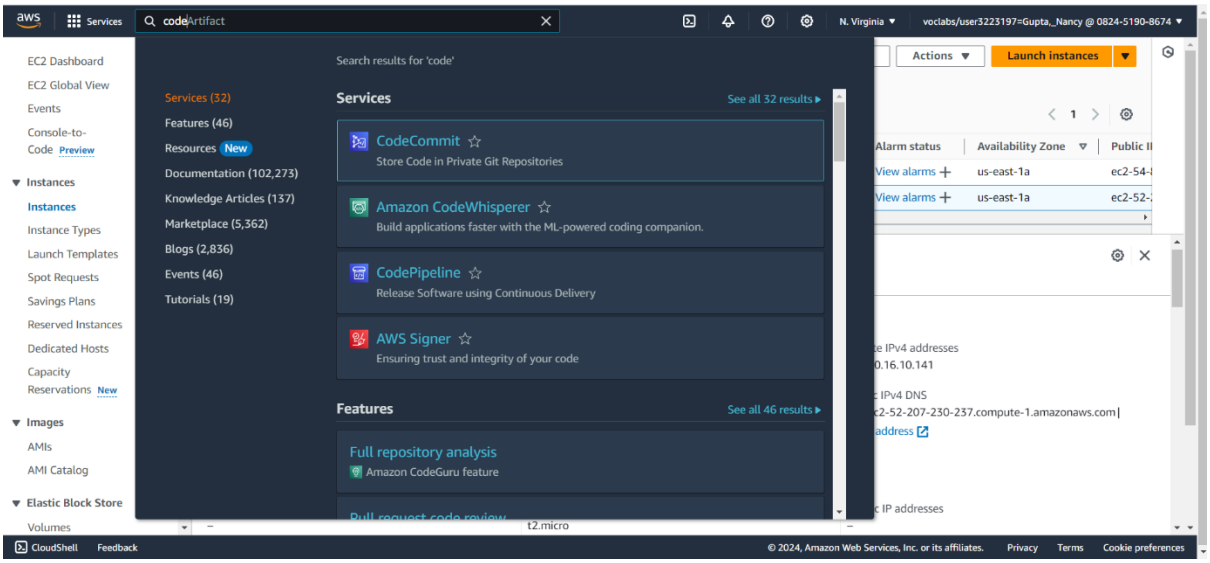


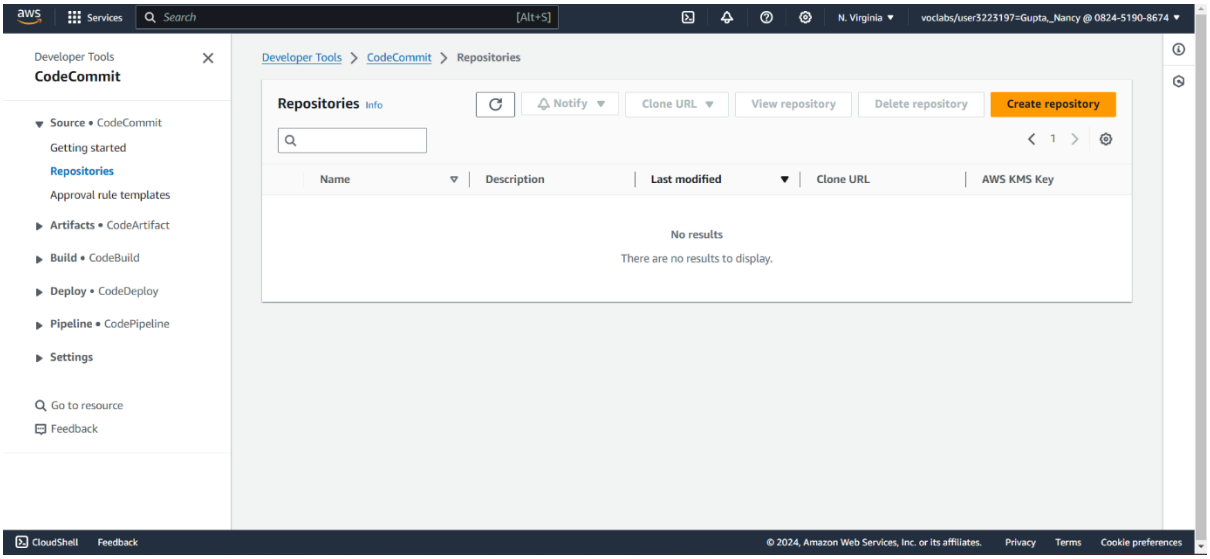
Task 3.4: Create a Git repository for the microservices code and push the code to CodeCommit

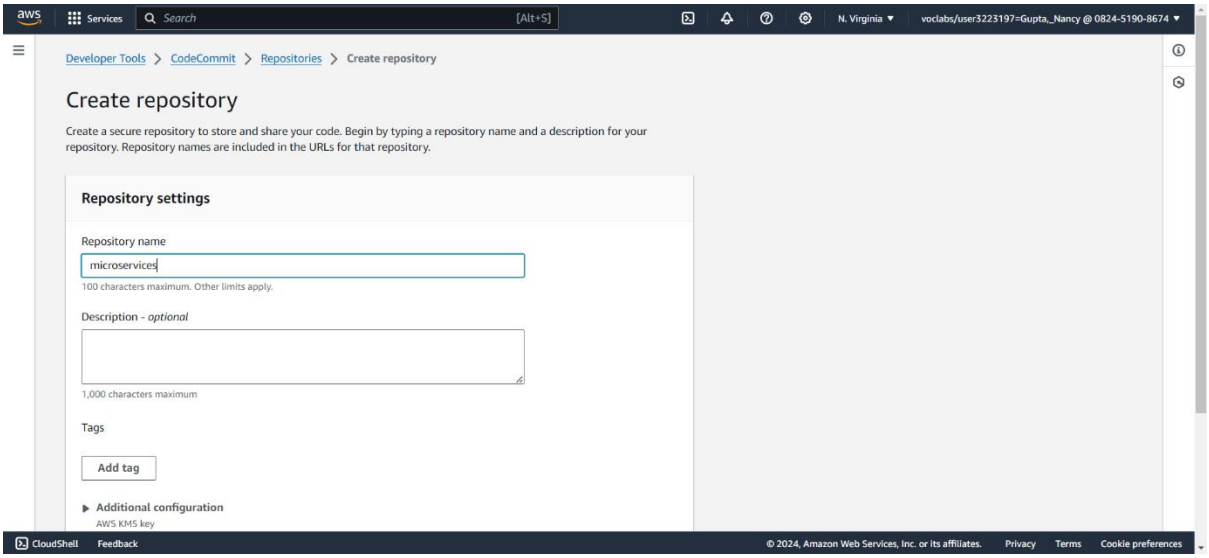
You now have directories named *customer* and *employee*, and each will contain a microservice. The two microservices will replicate the logic in your monolithic application. However, you will also be able to evolve the application functionality and to time the deployment of feature enhancements for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

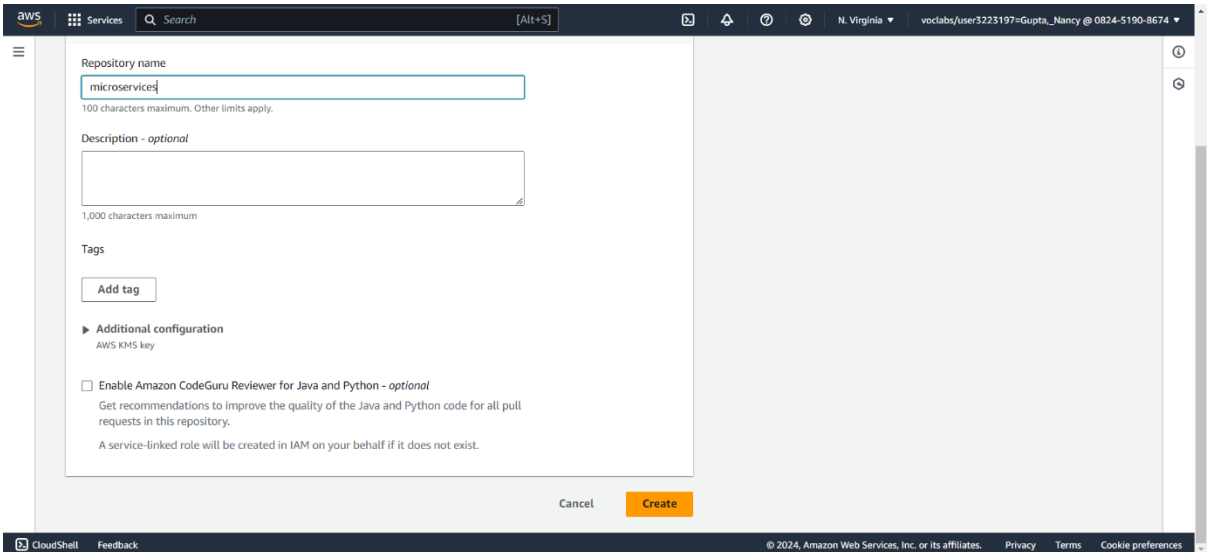
You will benefit from checking this source code into a Git repository. In this project, you will use CodeCommit as your Git repository (repo).

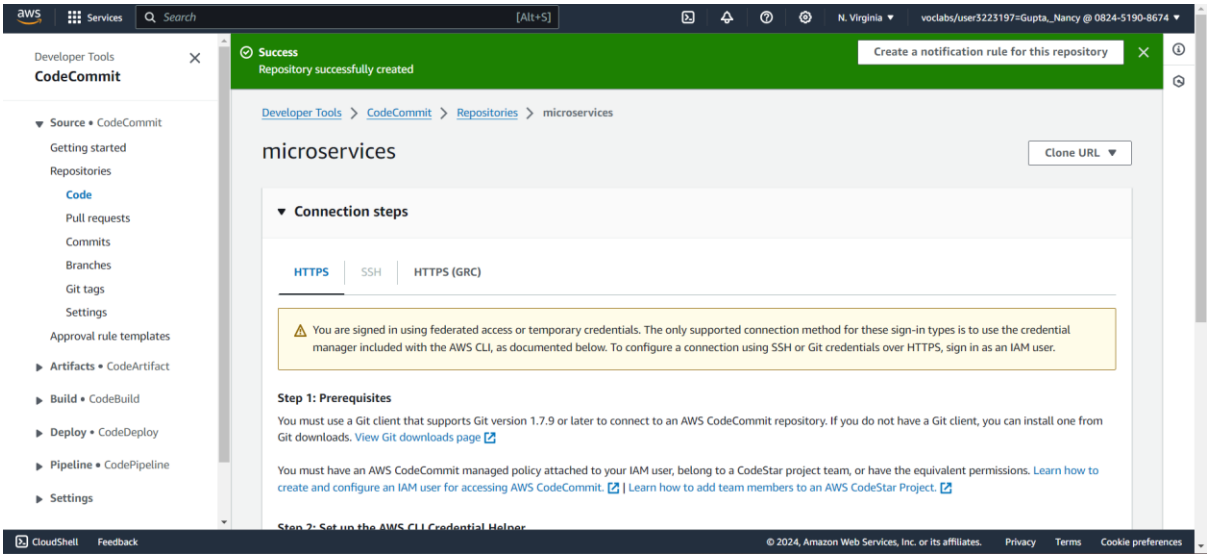
1. **Create a CodeCommit repository that is named microservices.**

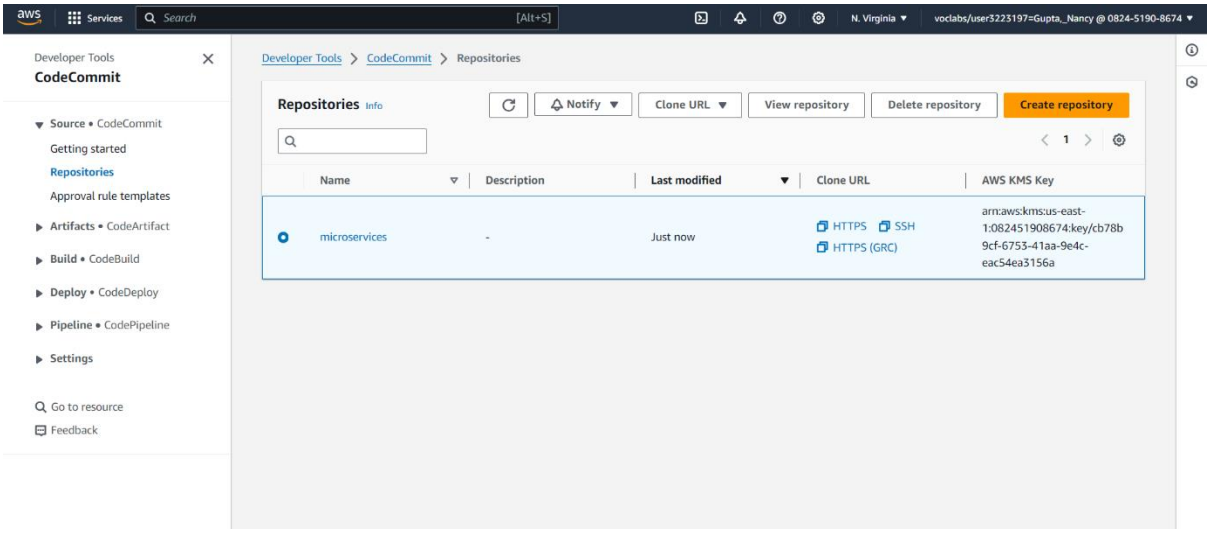



Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Developer Tools > CodeCommit > Repositories
Repositories Info
Notify
Clone URL
View repository
Delete repository
Create repository
Name
Description
Last modified
Clone URL
AWS KMS Key
No results
There are no results to display.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Developer Tools > CodeCommit > Repositories > Create repository
Create repository
Create a secure repository to store and share your code. Begin by typing a repository name and a description for your repository. Repository names are included in the URLs for that repository.
Repository settings
Repository name
microservices
100 characters maximum. Other limits apply.
Description - optional
1,000 characters maximum
Tags
Add tag
Additional configuration
AWS KMS key
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
N. Virginia
voclabs/user3223197=Gupta,_Nancy @ 0824-5190-8674
Repository name
microservices
100 characters maximum. Other limits apply.
Description - optional
1,000 characters maximum
Tags
Add tag
Additional configuration
AWS KMS key
Enable Amazon CodeGuru Reviewer for Java and Python - optional
Get recommendations to improve the quality of the Java and Python code for all pull requests in this repository.
A service-linked role will be created in IAM on your behalf if it does not exist.
Cancel
Create
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences





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