AMAZON WEB SERVICES

INDEX

ACKNOWLEDGEMENT

It is a great pleasure to get this opportunity and sincerely thank all those people who have helped me during the successful development of this Project on **"AWS Solutions Architect"** for **Department of Computer Science, SIES College of Arts, Science & Commerce (Autonomous), Sion (W), Mumbai-22.**

Our sincere regards and gratitude to the Head of the Department **Prof. Manoj Singh**, having given us the support throughout the course.

We would like to thank our project guide **Prof. Manoj Singh** and our professors **Prof. Maya Nair**, **Prof. Abuzar Ansari**, **Prof. Soni Yadav** and **Prof. Shivani Deopa** for their constant help and support during the development of this project.

We also express our gratitude towards the non-teaching staff of the Computer Science Department, for providing and maintaining the quality of laboratory required for the development of this project.

Also, we would like to thank our family members and friends for their cooperation and efforts for making this project a success.

By:- **Kris Kamble**

INTRODUCTION

Cloud computing is dominated by three major players: Amazon Web Services (AWS), Microsoft Azure and Google Cloud. Of the two, AWS has a significantly larger market share, with a 33% share of the cloud computing market. AWS is by far the largest cloud computing service provider in the world today.

AWS Solution Architects are the ones that are responsible for managing an organization's cloud computing architecture. They have in-depth knowledge of the architectural principles and services used to develop a technical cloud strategy, review workload architectures, assist with cloud migration efforts, and guides how to address high-risk issues.

Working as an AWS Solutions Architect means that you can come up with advanced cloudbased solutions and migrate the existing ones to the cloud.

EXPECTATION

Expectation from an AWS Solutions Architect:

- Designing and deploying scalable, highly available, resilient, cost effective and fault tolerant systems
- Migration of an existing on-premises application to AWS (Database)
- Ingress and Egress of data and from AWS (VPN and Encryption)
- Selecting the suitable AWS service based on data, compute, database, or security requirements
- Identifying appropriate use of AWS architectural best practices
- Estimating AWS costs and identifying cost control mechanisms

RELATED WORKS

Netflix:

Netflix is the world's leading internet television network, with more than 200 million members in more than 190 countries enjoying 125 million hours of TV shows and movies each day. Netflix uses AWS for nearly all its computing and storage needs, including databases, analytics, recommendation engines, video transcoding, and more—hundreds of functions that in total use more than 100,000 server instances on AWS.

Netflix relies on AWS to help it innovate with speed and consistently deliver best-in-class entertainment. AWS provides Netflix with compute, storage, and infrastructure that allow the company to scale quickly, operate securely, and meet capacity needs anywhere in the world. Moreover, Netflix, a leading content producer, has used AWS to build a studio in the cloud. This virtual studio enables Netflix to engage top artistic talent, no matter the location, and Netflix artists and partners have the freedom to collaborate without technological or geographical barriers.

INDmoney:

As the business started expanding, engineers sought lower costs by containerizing its app in line with its shift to a microservices approach. In the past, some of INDmoney's engineers had prior experience using Amazon Elastic Compute Cloud (Amazon EC2) spot
Instance in conjunction with Amazon Elastic Container Service (Amazon ECS) for stateless applications. They looked to replicate this strategy at INDmoney and reached out to AWS for support.

AWS set up an "Amazon ECS plus Spot Instances" strategy, using a dummy environment for hands-on testing with INDmoney data. In one-and-a-half months—two weeks faster than expected—the start-up had containerized its AWS Auto Scaling entire application stack using groups.

OBJECTIVE

The objective of this project is to find solution for the problems brought by the clients, so their respective website or application can run smoothy without any hindrance. In this project we took some problems that the clients commonly face while deploying their website or application.

- Security Management in AWS
- Object Storage Options
- Amazon EC2
- Load Balancing Auto-scaling and Route 53
- Database services and Analytics
- Networking and Monitoring Services
- Application Services and AWS Lambda
- Configuration Management and Automation
- Migration to AWS

STAKEHOLDERS

1. **Chief Technology Officer:**
   CTO of AWS leads and directly manages a team of Software Development Managers, Technical Program Managers, Data Engineers, Data Scientists, Solutions Architects, and a Development Operations team. CTO partners with multiple cross-functional technology teams at Amazon, and with Fabric's CFO, Head of Operations, and Head of Retail. Responsibilities will also include direct management of process and quality of service improvements, strategic planning, project management for software development, procurement and releases, and management of resources across teams.

2. **Client:**
   A Client can be a person or organization using the services of Amazon Web Services(AWS)

3. **User:**
   User can be termed here as a general Individual or People, Audience around the Globe.

4. **AWS Solutions Architect:**
   AWS Solutions Architect is a cloud computing expert who designs the architecture of an organization's cloud assets and plans implementation of the design. An SA is the keeper of the AWS Well Architected Framework pillars, ensuring customers are following best practices, guidance, and recommendations to develop cloud solutions that are secure, resilient, efficient, and managed with operational excellence.

5. **AWS Solutions Architect Team:**
   AWS Solutions Architect Team role is same as AWS SA, but they work in a Team. They Maintain and Develop the Architecture after the Main SA leaves the Project who was originally assigned for it.

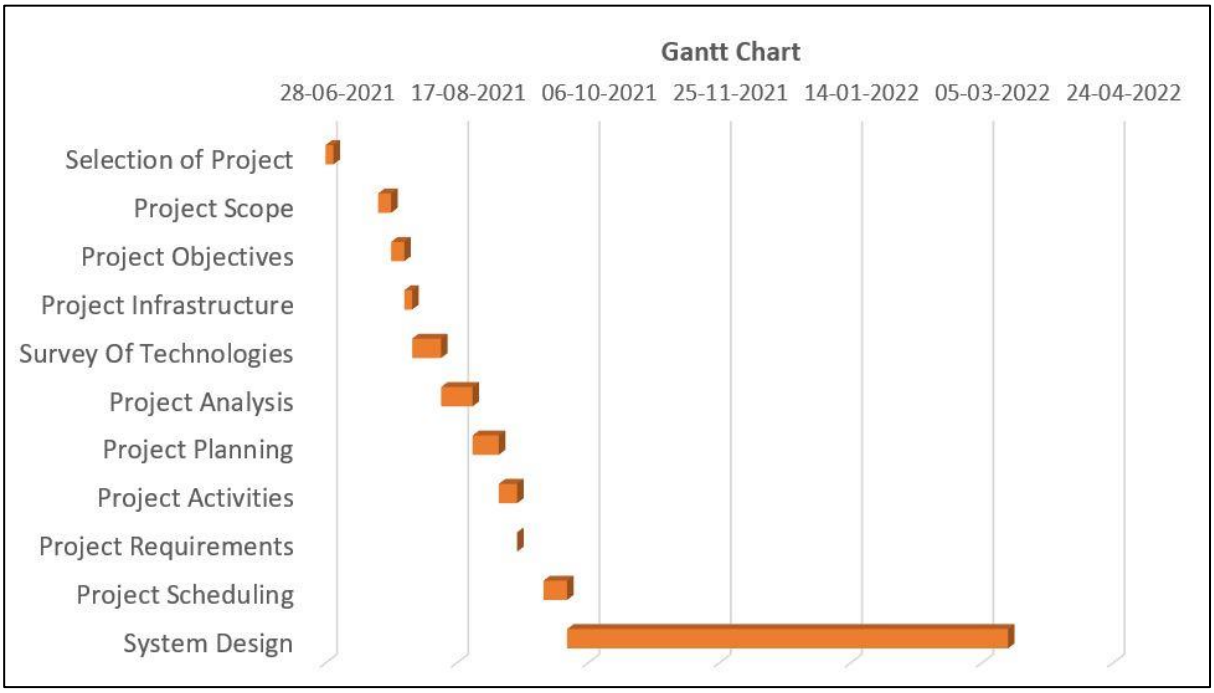
METHODOLOGY

Technologies & Services Used:

Platform	: https://console.aws.amazon.com
Languages	: HTML, CSS, JavaScript, PHP, MySQL, Python, Java
Systems	: Linux, Windows
Tools and Software	: Putty, Pageant, Remote Desktop Application, DBeaver, FileZilla
Technology and Services	: Amazon Relational Database Service(RDS), Amazon Database Migration Service(DMS), Amazon Redshift, Amazon Simple Storage Service(S3), Amazon Virtual Private Cloud(VPC), Amazon CloudFormation, Amazon DynamoDB, Amazon Lambda, Amazon Identity and Access Management(IAM), Amazon Simple Email Service(SES), Amazon OpsWorks Stacks, Chef 11, Amazon Cognito, Amazon Elastic Compute Cloud(EC2), Elastic Load Balancing(ELB), Amazon CloudFront, Amazon CloudWatch, Amazon Auto Scaling, Amazon Elastic Beanstalk, Amazon Simple Queue Service(SQS), Amazon Route 53, Amazon CodePipeline, Amazon CodeDeploy, Amazon CodeCommit, Amazon Certificate Manager.

GANTT CHART

Task	Start Date	End Date	Duration
Selection of Project	28-06-2021	01-07-2021	3
Project Scope	18-07-2021	23-07-2021	5
Project Objectives	23-07-2021	28-07-2021	5
Project Infrastructure	28-07-2021	31-07-2021	3
Survey Of Technologies	31-07-2021	11-08-2021	11
Project Analysis	11-08-2021	23-08-2021	12
Project Planning	23-08-2021	02-09-2021	10
Project Activities	02-09-2021	09-09-2021	7
Project Requirements	09-09-2021	09-09-2021	0
Project Scheduling	19-09-2021	28-09-2021	9
System Design	28-09-2021	15-03-2022	168



SYSTEM ANALYSIS

Use Cases:

Section A

Use Case: 1

Homogenous Database Migration

Use Case: 2

Managing Airline Data using managed Data Warehouse Service

Use Case: 3

Creating, Updating & Deleting the Resources[stacks] using CloudFormation template

Use Case: 4

Sending an Email on Addition of User data in the DynamoDB (NoSQL)

Use Case: 5

Create a Stack and Deploy an Application in the Stack

Use Case: 6

Configuring Strong and Secure Authentication Access Mechanism

Use Case: 7

Scenario Based:

Background:

A start-up company wants to host its Python and React-based application (Backend: Python API and Frontend React) using AWS. But they are not familiar with the AWS cloud platform. They want to ensure that the application is secure, scalable, highly available, and cost efficient. As a solutions architect, you must design a proper solution to meet their below requirements.

Goal:

To architect a solution that is secure, scalable, highly available, and cost-effective using AWS.

Requirements:

- They are concerned about the security of the application, so they have decided to isolate their network from the rest of the customers virtually. Set-up a secure virtual network where the only frontend of application is accessible by users and not the database
- Execute the React application code using AWS Elastic Beanstalk. Ensure that the source code of Web application is automatically picked, pushed to the master branch, and deployed on the servers
- Ensure all the UI images served to the frontend application code are provisioned via a secure storage unit
- There should be enough backups for both the Web and Database server, so if the setup crashes, we can launch a new one from the disaster recovery backups
- They are uncertain about the traffic pattern that how low or high it can be, so they want the Web application to be running on at least two EC2 instances all time, and when there is a high load, they must burst up to four instances in total
- The Web application should be highly available, even if any VM fails to respond to queries, there should be a mechanism to switch the connection to another healthy VM automatically
- Automate the download all the activity logs into a CSV file, create a stream of data, analyse it, and display it via a dashboard
- The Web application should also be cached globally, so users worldwide can access it with low latency

Section B

Use Case: 1

Architecting a Website Using Serverless Technology

Use Case: 2

Managing Tightly Coupled Architecture

Use Case: 3

Using VPC Peering to Communicate between Two Instances

Use Case: 4

Maintaining User Experience with Low Latency using Route 53 Traffic Flow Feature

Use Case: 5

DevOps – Deploy an Application using CodePipeline

Use Case: 6

Solution for Smart Industrial Cooling System

Use Case: 7

Scenario Based:

Background:

You are recently promoted from a Cloud Engineer to a Cloud Architect and assigned a project to prepare a new environment in the cloud, to which your team will later migrate their applications.

Goal:

To architect a solution that is secure, scalable, highly available, and cost-effective using AWS.

Requirements:

- They are concerned about the security of the environment, so they have decided to virtually isolate their network from the rest of the customers and the rest of the environments in the same AWS Cloud Account
- Due to the budget issue, the company cannot afford a dedicated DB engineer, so they are willing to outsource the DB management from a Cloud provider, to store and maintain the customer information received by PHP application. You must pick the right solution from AWS, which should be a Platform as a Service. It should also provide high availability, patching, and back-ups. (Hint: Create DB subnet group)
- And about disaster recovery, you should have enough backups for both the Web and Database server, so if in case the environment crashes, we can launch a new environment from the disaster recovery backups
- Design a dynamic website where the customers can enter their details, which should be stored in a database
- They are uncertain about the traffic pattern that how low or high it can be, so they have a requirement that the environment should be running at least two EC2 servers all time, and when there is a high load, they must burst up to four servers in total
- Now the company cannot afford a dedicated engineer for monitoring, so you must automate the incident management through an event notification. Anytime there is an increase and decrease in the VM's due to high or low traffic, you must receive a notification via email
- The application should be highly available, even if a VM fails to respond to queries, there should be a mechanism to shift the connection to another healthy VM automatically
- Your Dynamic Website should also be cached globally, so users worldwide can access it with less latency. The customer is okay if we get an unfriendly AWS generated URL for accessing the website

EVENT TABLE

General Event Table: Section A and B

Event	Source	Trigger	Activity	Response	Destination
Client wants a solution to the problem	Client	Examine the Problem	Viewing AWS Cloud Provider	Successful	Client, CTO
Client wants Solution to Requirements	Client	Examine the Requirements	Viewing Requirements	Successful	Client, CTO
Client wants cost effective Solution	Client	Examine the Cost	Viewing Cost	Successful	Client, CTO
Client wants to Pay for Services	Client	Check Payment Portal	Viewing Payment Method	Payment Successful	Client, CTO
Client wants to maintain the Security in the Cloud	Client	Check the Security in the Cloud	Viewing Security Protocols	Security Updated	Client
CTO wants to provide the Solution	CTO	Examine the Problem and Solution	Viewing with Solutions Architect	Successful	CTO, Solutions Architect
SA wants to Sign in as AWS Root account	SA	Enter User ID, reCAPTCHA and Password	Validation	1. If credentials are correct: Access granted 2.If details are incorrect: Access denied	SA
SA wants to assess the problem	SA	Examine the Problem	Viewing problems	Successful	SA
SA wants to analyze the requirements	SA	Examine the Requirements	Viewing the requirements	Successful	SA

SA wants to provide Solution	SA	Provide Resources	Viewing Resources	Successful	SA
SA wants to provide Security of the Cloud	SA	Provide Cloud Security	Viewing Security	Security Updated	SA
SA wants to maintain Cloud Solution	SA	Check Cloud Services	Viewing Cloud Services	Successful	SA
SA wants to Connect to Amazon EC2 Instance	SA	Enter Credentials to Putty	Connecting EC2 Instance Putty	Connection Successful	SA
SA Team wants to Sign in	SA Team	Enter IAM Credentials: 1. Access Key ID 2. Secret Access Key 3. Username 4. Password	Validation	Successful	SA Team
SA Team wants to Access Services	SA Team	Check Permissions	Validation	Successful	SA Team
SA Team wants to maintain Cloud Solution	SA Team	Check Cloud Services	Viewing Cloud Services	Successful	SA Team, SA

Event Table: Section A and B

Event	Source	Trigger	Activity	Response	Destination
Client wants to access the Website	Client	Check Website	Viewing Website	Successful	Client
Users wants to access the Website	Users	Check Website	Viewing Website	Successful	Users
Users wants to give Feedback Survey	Users	Check Website	Entering Feedback	Successful	Users, SA, SA Team
SA wants to access Website	SA	Check Website	Viewing Website	Successful	SA
SA wants to use Amazon S3	SA	Open Amazon S3 Service	Accessing Amazon S3	Successful	SA
SA wants to use Amazon DynamoDB	SA	Open Amazon DynamoDB Service	Accessing Amazon DynamoDB	Successful	SA
SA wants to use Amazon DMS	SA	Open Amazon DMS Service	Accessing Amazon DMS	Successful	SA
SA wants to use Amazon Cognito	SA	Open Amazon Cognito Service	Accessing Amazon Cognito	Successful	SA
SA wants to use Amazon Redshift	SA	Open Amazon Redshift Service	Accessing Amazon Redshift	Successful	SA
SA wants to use Amazon IAM	SA	Open IAM Service	Accessing Amazon IAM	Successful	SA
SA wants to use Amazon EC2	SA	Open EC2 Service	Accessing Amazon EC2	Successful	SA

SA wants to use Amazon RDS	SA	Open Amazon RDS Service	Accessing Amazon RDS	Successful	SA
SA wants to use Amazon SES	SA	Open Amazon SES Service	Accessing Amazon SES	Successful	SA
SA wants to use Amazon SQS	SA	Open Amazon SQS Service	Accessing Amazon SQS	Successful	SA
SA wants to use DBeaver	SA	Open DBeaver Client	Accessing DBeaver	Successful	SA
Client wants to Access Resources	Client	Select from AWS Console	Accessing Resources	Successful	Client
SA wants to use Amazon VPC	SA	Open Amazon VPC Service	Accessing Amazon VPC	Successful	SA

SA wants to Peer Amazon VPC's	SA	Open VPC Peering Service	Accessing Amazon VPC Peering	Successful	SA
SA wants to use Remote Desktop Application	SA	Open Remote Desktop Application	Accessing Remote Desktop Application	Successful	SA
Client wants to access Freenom Domain	Client	Check Domain	Viewing Domain	Successful	Client, SA
SA wants to access Freenom Domain	SA	Check Domain	Viewing Domain	Successful	SA, Client
SA wants to use Amazon Route 53	SA	Open Amazon Route 53 Service	Accessing Amazon Route 53	Successful	SA
SA wants to use Amazon Traffic Flow feature	SA	Open Traffic Flow feature	Creating Amazon Traffic Flow	Successful	SA

SA wants to use Elastic Load Balancer	SA	Open Elastic Load Balancer Service	Accessing Elastic Load Balancer	Successful	SA
SA wants to use Elastic Beanstalk	SA	Open Amazon Elastic Beanstalk Service	Accessing Amazon Elastic Beanstalk	Successful	SA
SA wants to use OpsWorks	SA	Open Amazon OpsWorks Service	Accessing OpsWorks	Successful	SA
SA wants to use Amazon Chef 11	SA	Open Amazon Chef 11 Service	Accessing Amazon Chef 11	Successful	SA
SA wants to use Amazon CloudWatch	SA	Open CloudWatch Service	Accessing CloudWatch	Successful	SA
SA wants to use Amazon CloudFormation	SA	Open Amazon CloudFormation Service	Accessing CloudFormation	Successful	SA
SA wants to use AWS CodePipeline	SA	Open AWS CodePipeline Service	Accessing CodePipeline	Successful	SA

SA wants to use AWS CodeCommit	SA	Open AWS CodeCommit Service	Accessing CodeCommit	Successful	SA
SA wants to use AWS CodeDeploy	SA	Open AWS CodeDeploy Service	Accessing CodeDeploy	Successful	SA
SA wants to use AWS Certificate Manager	SA	Open AWS Certificate Manager	Request a Certificate	Successful	SA
SA wants to use AWS Target Group	SA	Open Target Group	Adding Instances in Target Group	Successful	SA
SA wants to use FileZilla	SA	Open FileZilla	Accessing FileZilla	Successful	SA, SA Team

SA wants to use Amazon SNS	SA	Open SNS Service	Accessing Amazon SNS	Successful	SA
SA wants to use Amazon Lambda	SA	Open Lambda Service	Accessing Amazon Lambda	Successful	SA
SA wants to use AWS Auto Scaling	SA	Open AWS Auto Scaling Service	Accessing Auto Scaling	Successful	SA
SA wants to create Web Application	SA	Enter Code	Accessing Putty	Successful	SA, Client
Client wants to Access Web Application	Client	Check Web Application	Accessing Web Application	Successful	Client, SA
SA wants to use Amazon CloudFront	SA	Open Amazon CloudFront Service	Accessing CloudFront	Successful	SA

UML DIAGRAMS

Use Case Diagrams

Use Case diagrams are used during requirements elicitation and analysis as a graphical means of representing the functional requirements of the system. Use cases are developed during requirements elicitation and are further refined and corrected as they are reviewed (by stakeholders) during analysis. Use cases are also very helpful for writing acceptance test cases. The test planner can extract scenarios from the use cases for test cases.

An actor represents whoever or whatever (person, machine, or other) interacts with the system. The actor is not part of the system itself and represents anyone or anything that must interact with the system to:

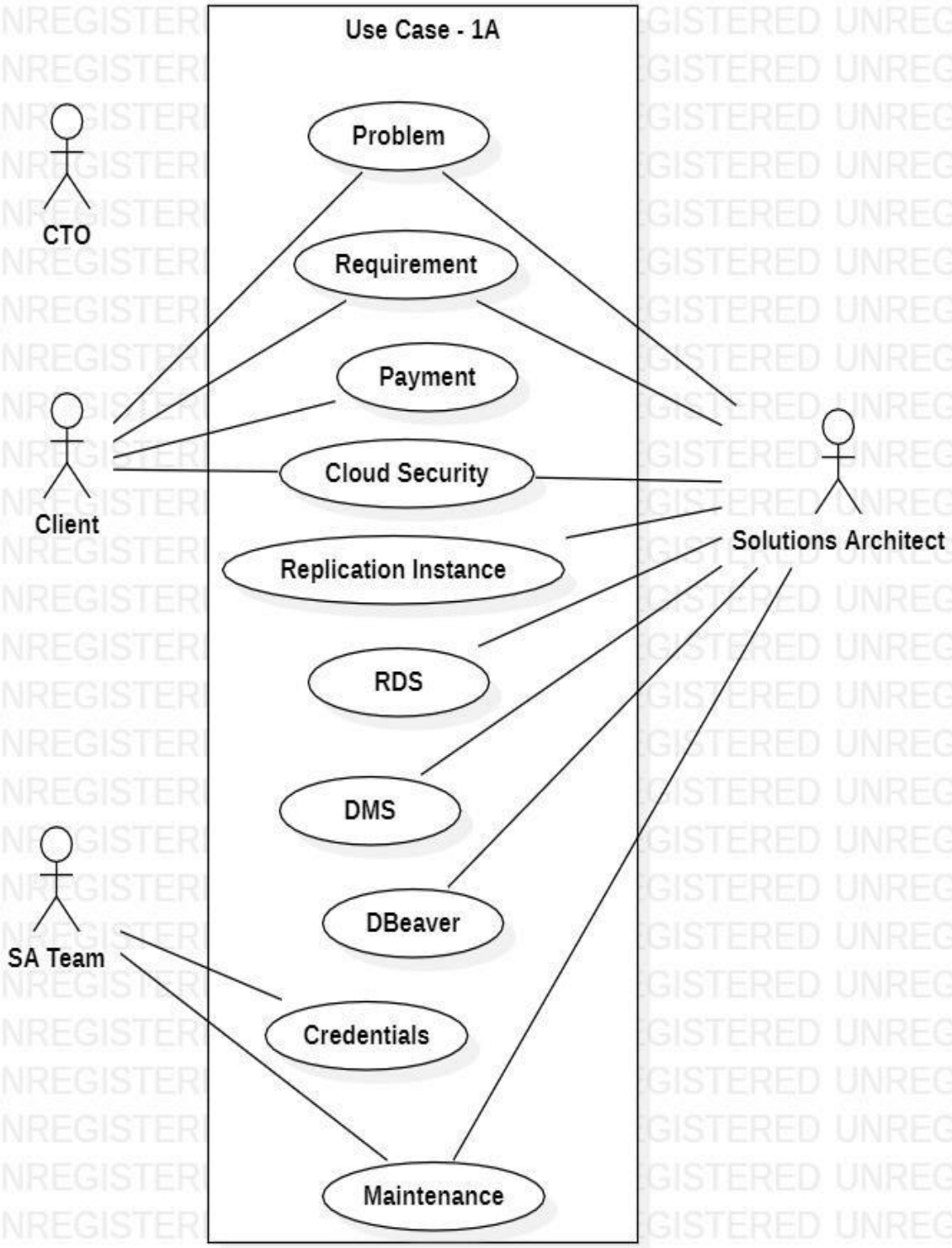
- Input information to the system.
- Receive information from the system; or Both input information to and receive information from the system

Purpose of Use Case Diagram

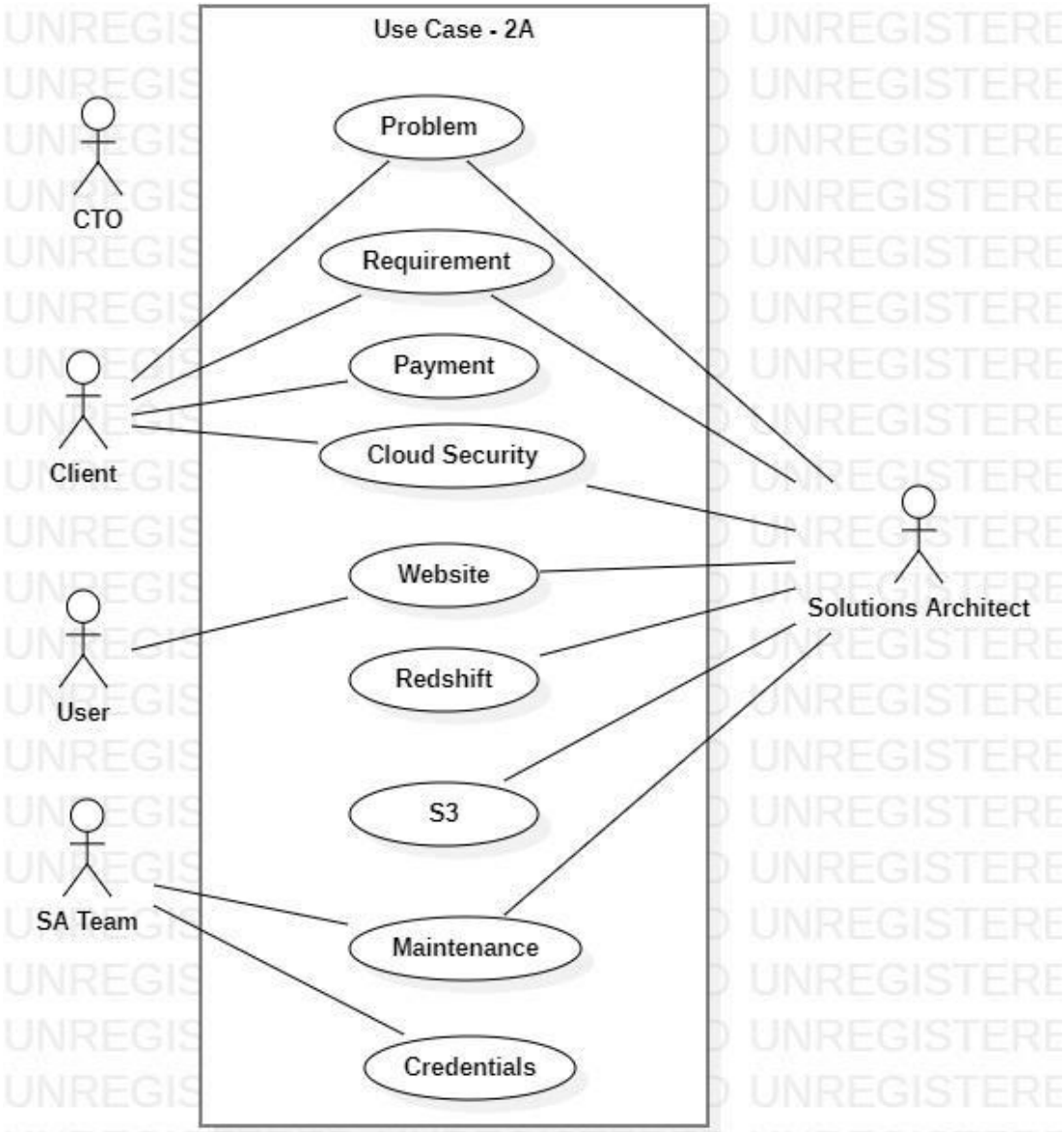
Use Case diagrams are typically developed in the early stage of development and people often apply use case modelling for the following purposes:

- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases
- Developed by analysts together with domain experts

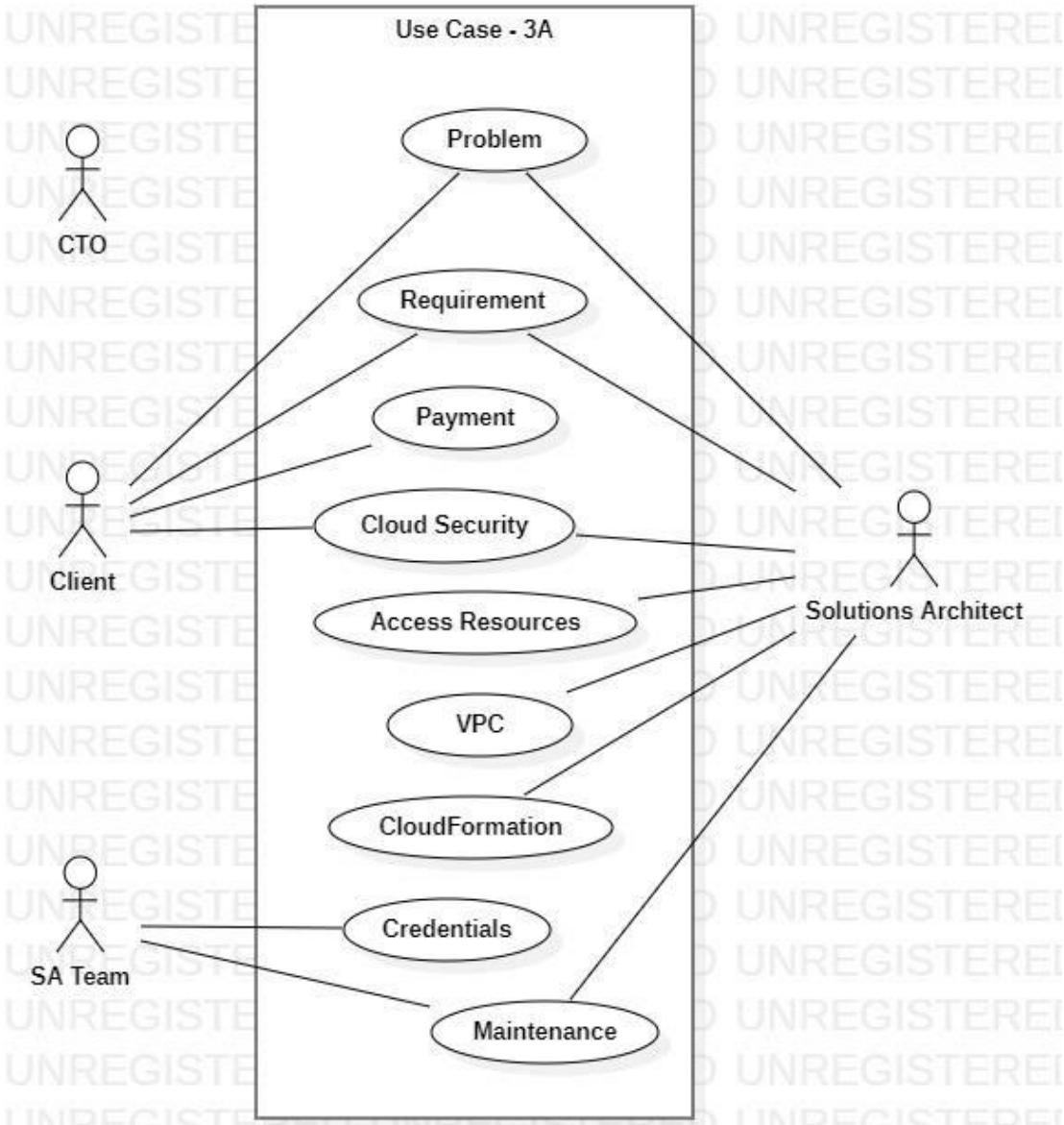
Use Case: 1A


Use Case - 1A
Problem
Requirement
Payment
Cloud Security
Replication Instance
RDS
DMS
DBeaver
Credentials
Maintenance
CTO
Client
SA Team
Solutions Architect

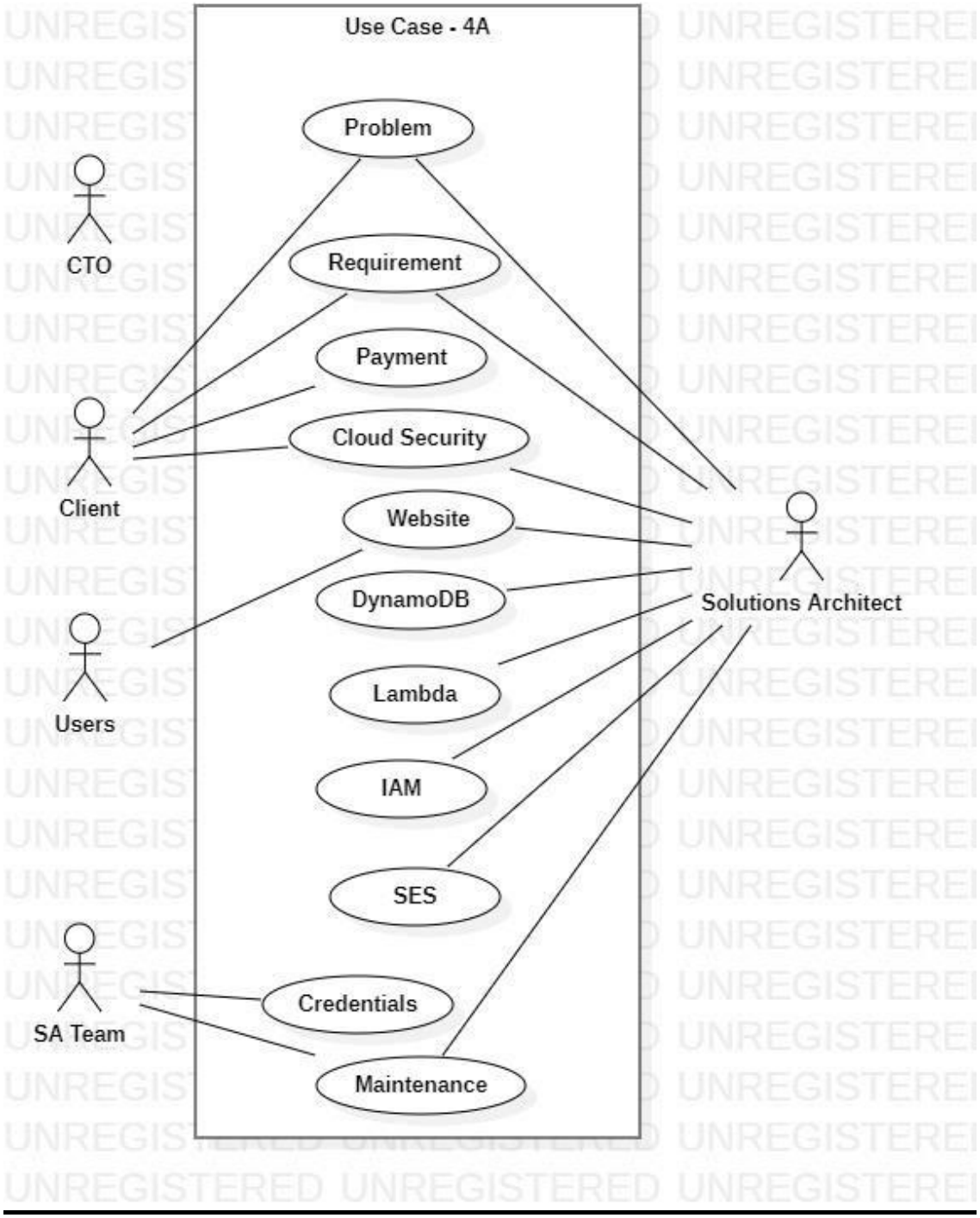
Use Case:2A


Use Case - 2A
CTO
Client
User
SA Team
Problem
Requirement
Payment
Cloud Security
Website
Redshift
S3
Maintenance
Credentials
Solutions Architect

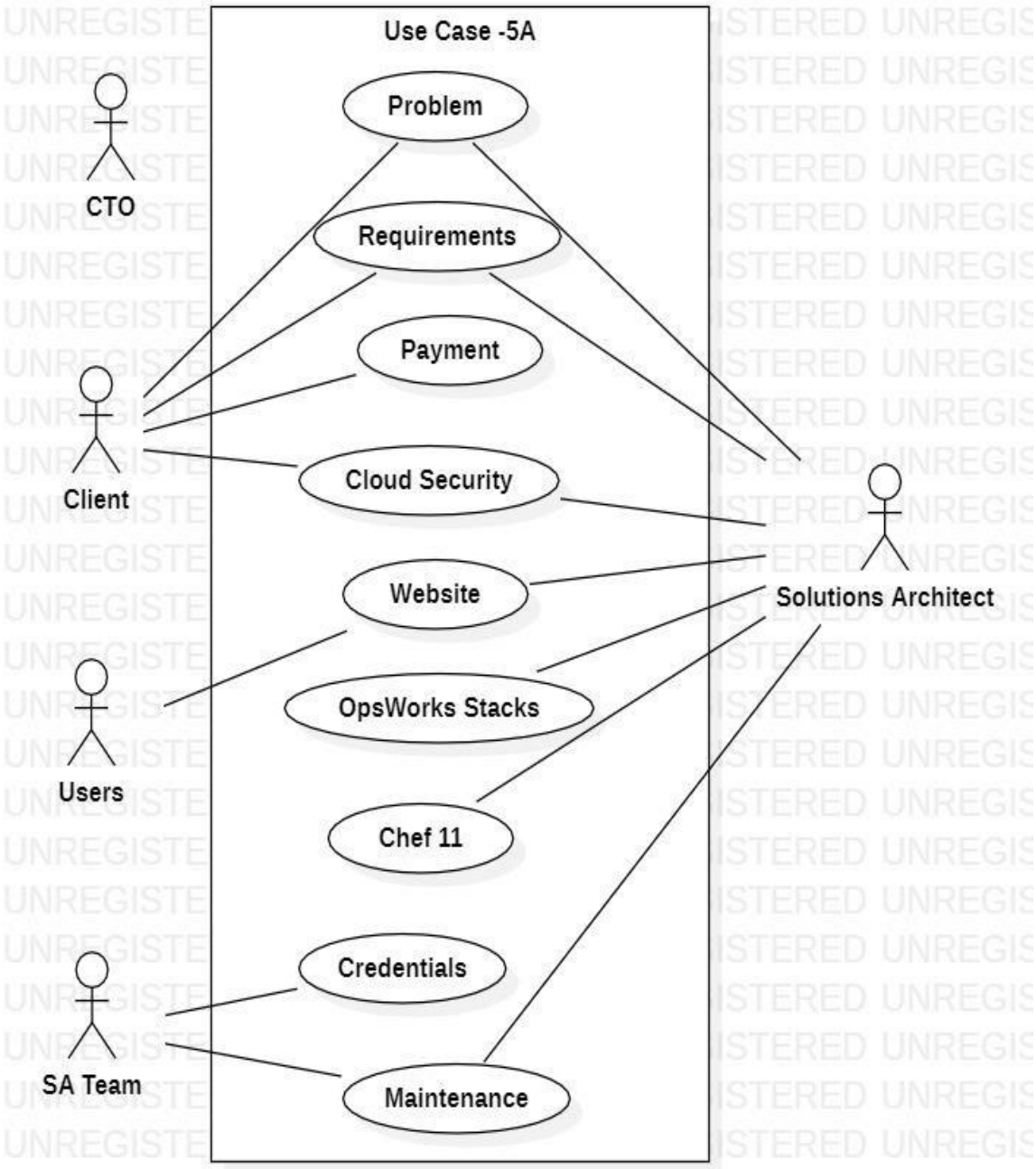
Use Case:3A


Use Case - 3A
Problem
Requirement
Payment
Cloud Security
Access Resources
VPC
CloudFormation
Credentials
Maintenance
CTO
Client
SA Team
Solutions Architect

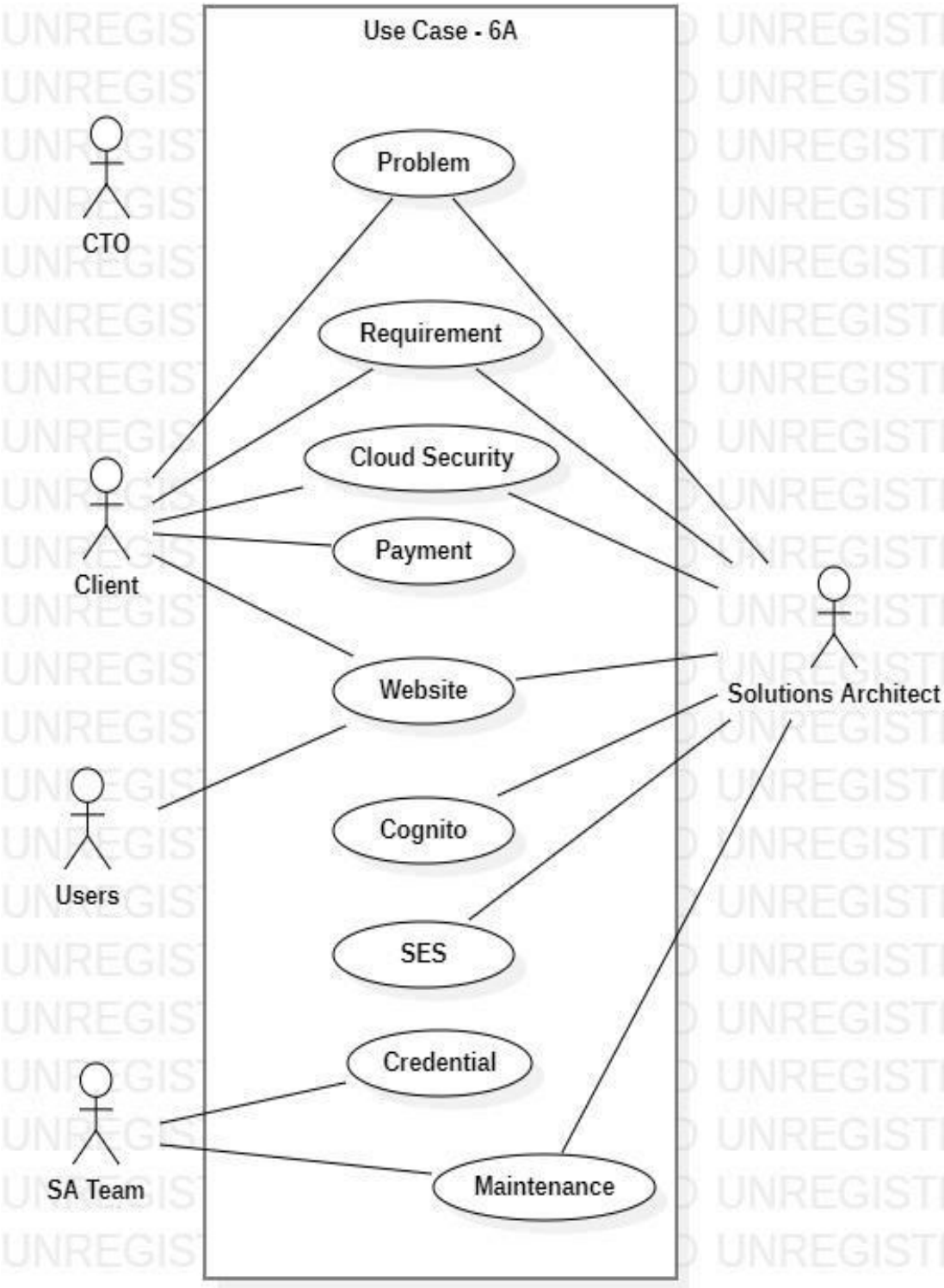
Use Case:4A


Use Case - 4A
Problem
Requirement
Payment
Cloud Security
Website
DynamoDB
Lambda
IAM
SES
Credentials
Maintenance
CTO
Client
Users
SA Team
Solutions Architect

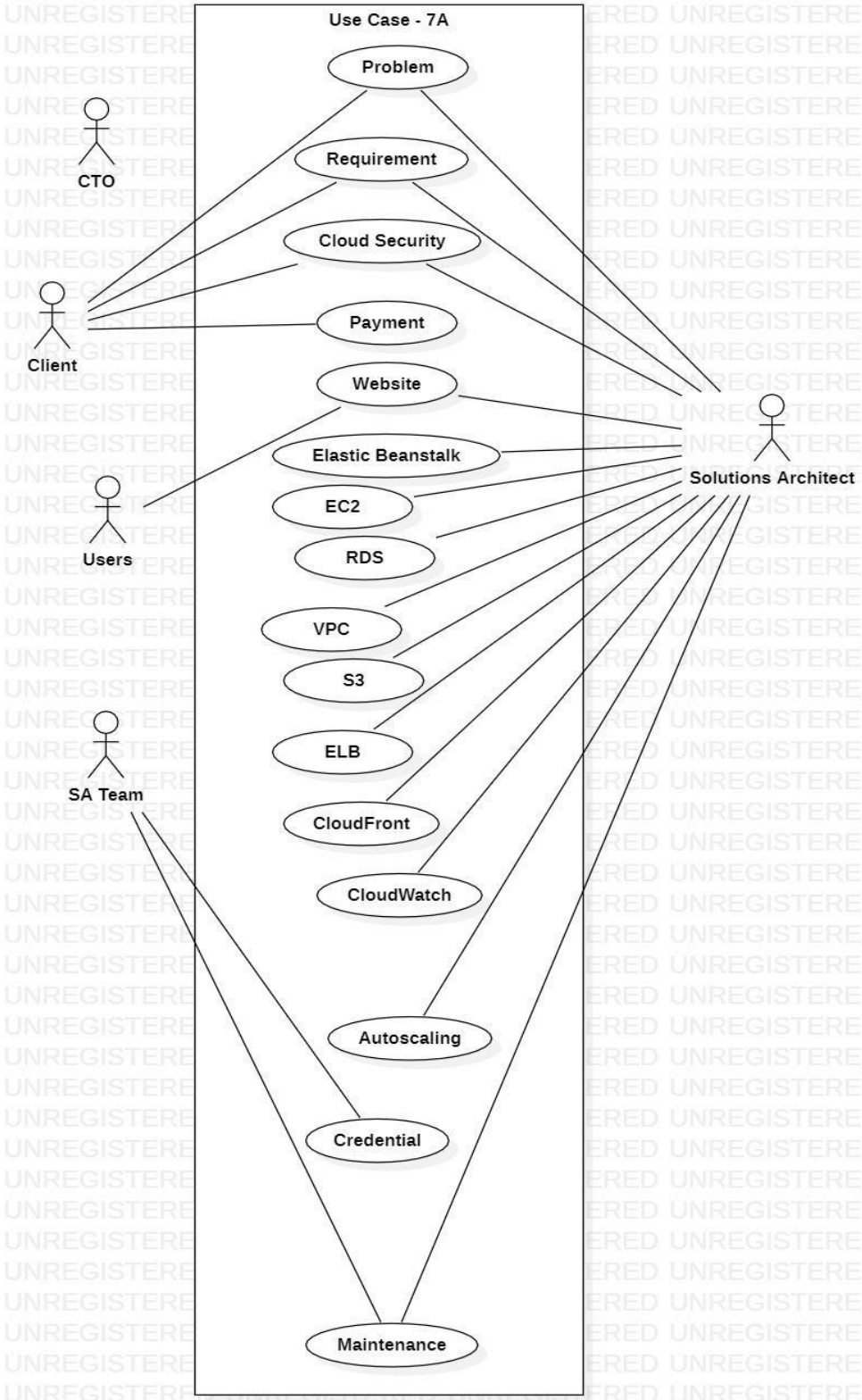
Use Case:5A


Use Case -5A
Problem
Requirements
Payment
Cloud Security
Website
OpsWorks Stacks
Chef 11
Credentials
Maintenance
CTO
Client
Users
SA Team
Solutions Architect

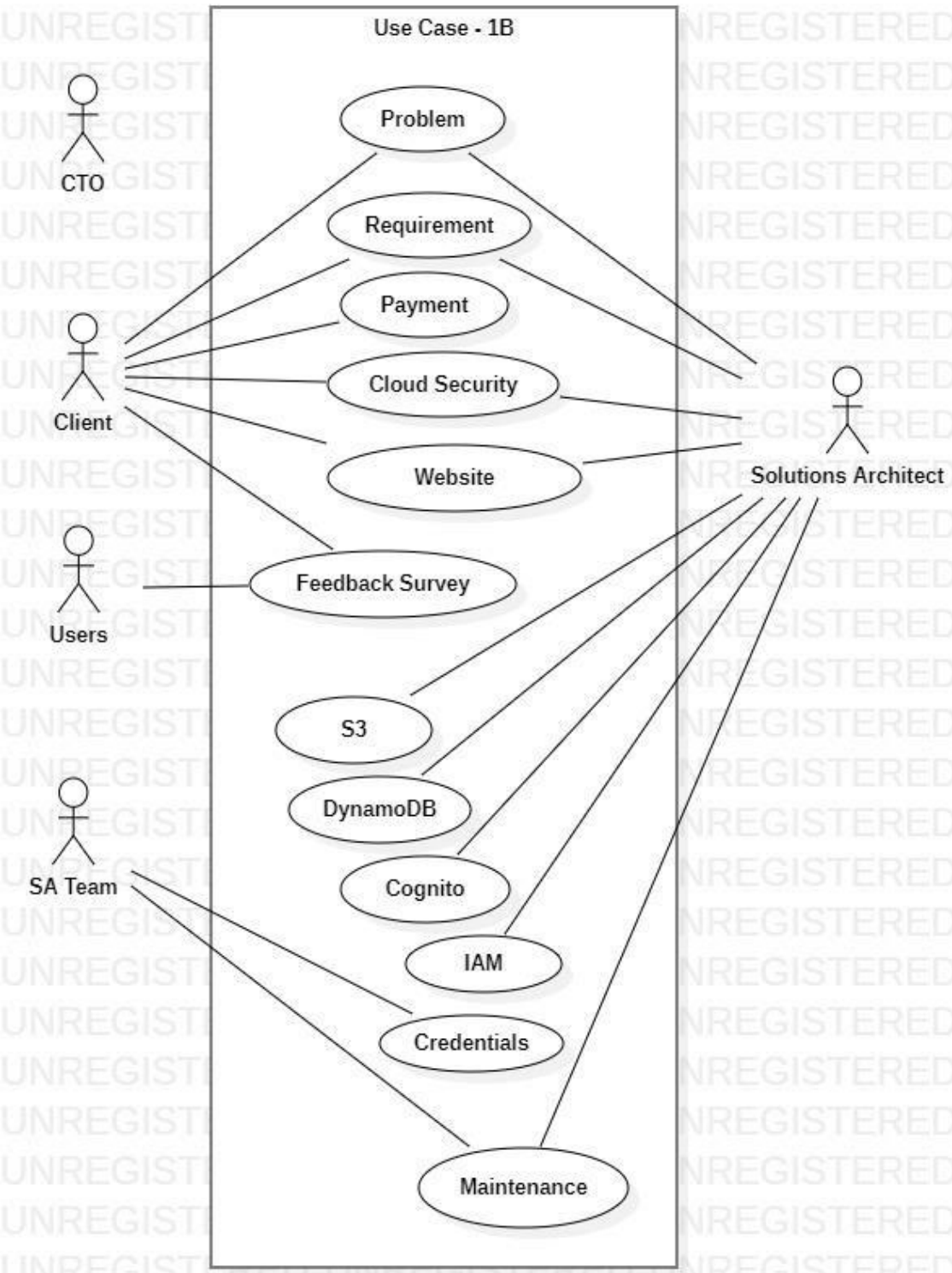
Use Case:6A


Use Case - 6A
CTO
Client
Users
SA Team
Solutions Architect
Problem
Requirement
Cloud Security
Payment
Website
Cognito
SES
Credential
Maintenance

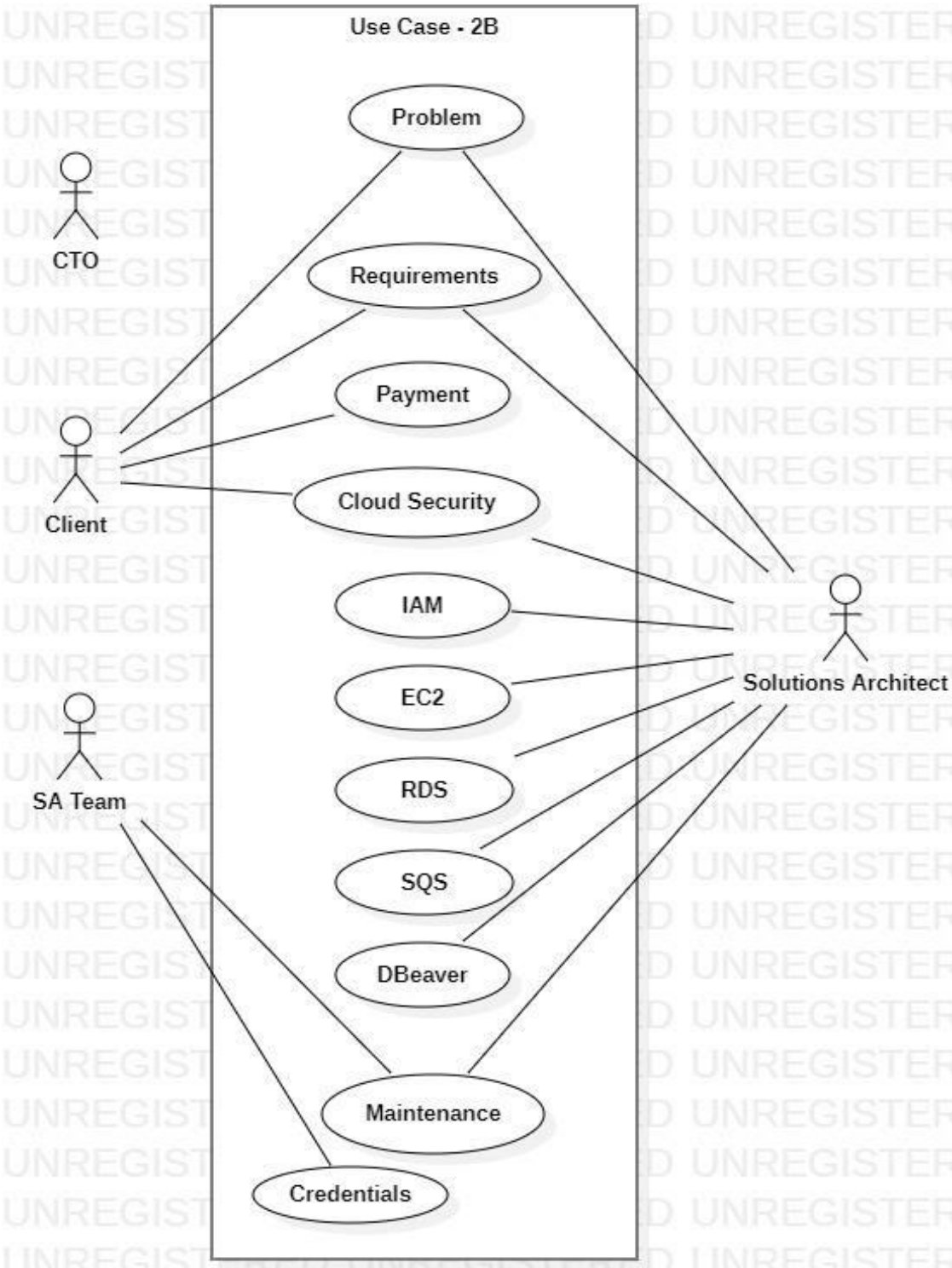
Use Case:7A


Use Case - 7A
Problem
Requirement
Cloud Security
Payment
Website
Elastic Beanstalk
EC2
RDS
VPC
S3
ELB
CloudFront
CloudWatch
Autoscaling
Credential
Maintenance
CTO
Client
Users
SA Team
Solutions Architect

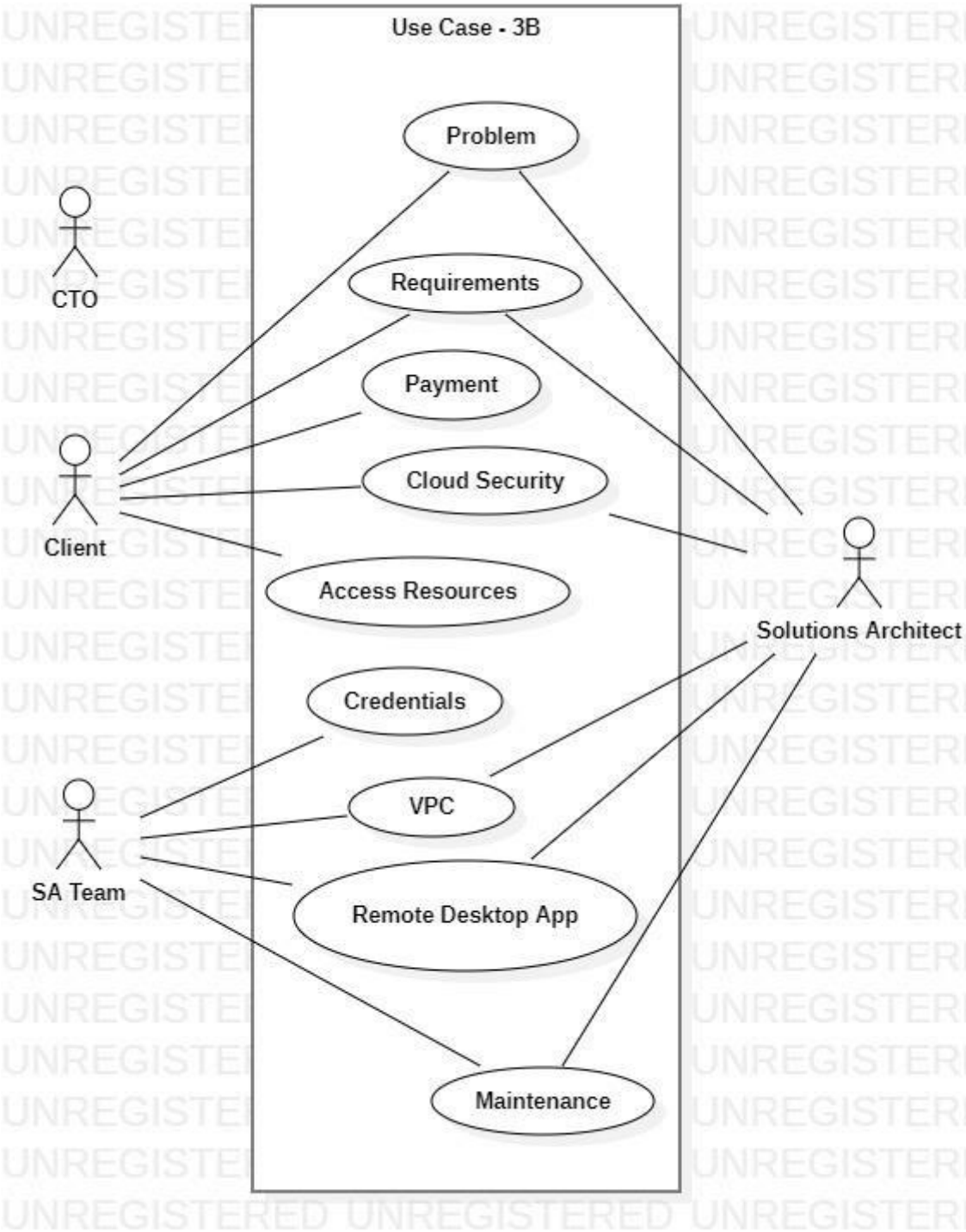
Use Case: 1B


Use Case - 1B
CTO
Client
Users
SA Team
Problem
Requirement
Payment
Cloud Security
Website
Feedback Survey
S3
DynamoDB
Cognito
IAM
Credentials
Maintenance
Solutions Architect

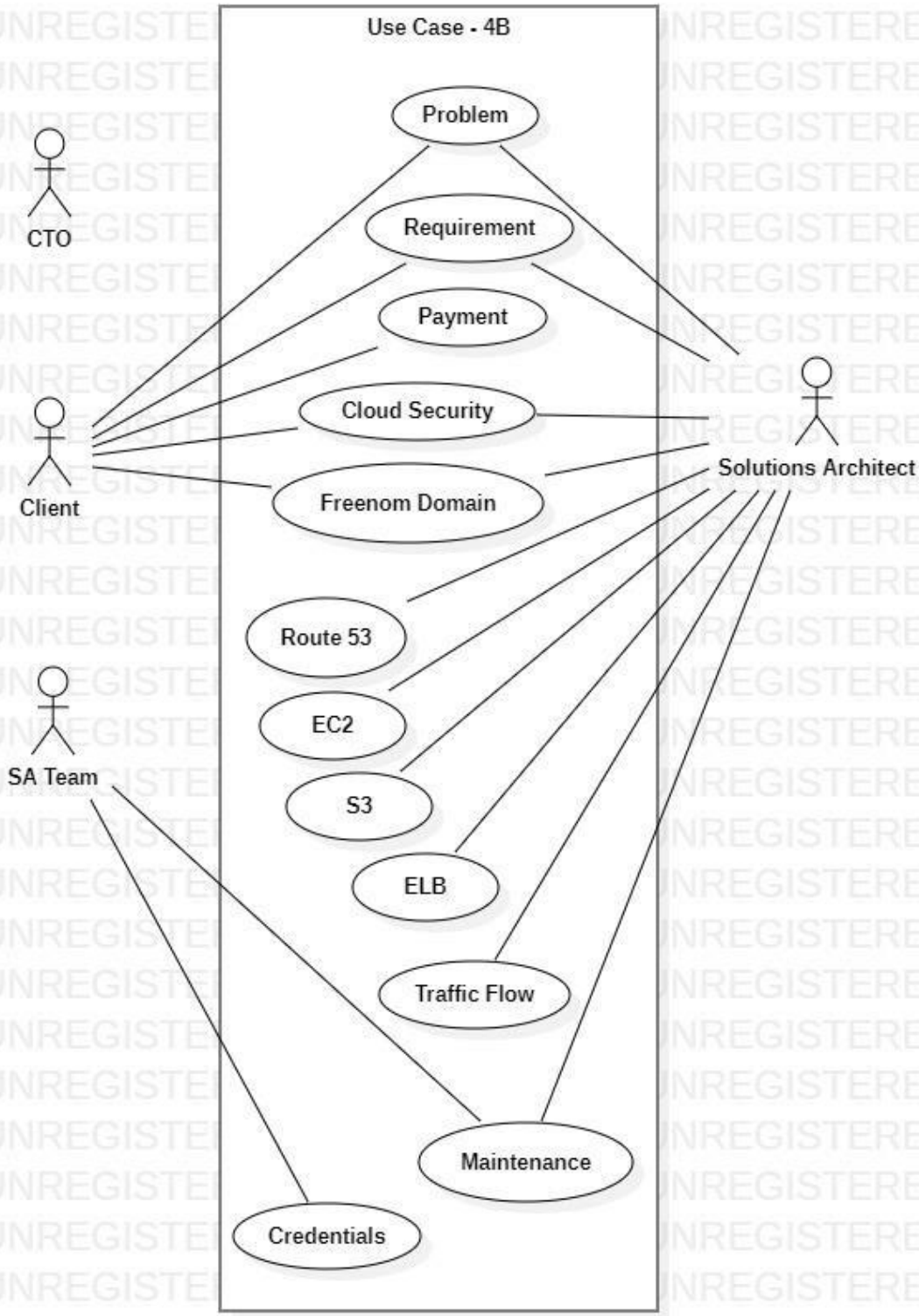
Use Case: 2B


Use Case - 2B
Problem
Requirements
Payment
Cloud Security
IAM
EC2
RDS
SQS
DBeaver
Maintenance
Credentials
CTO
Client
SA Team
Solutions Architect

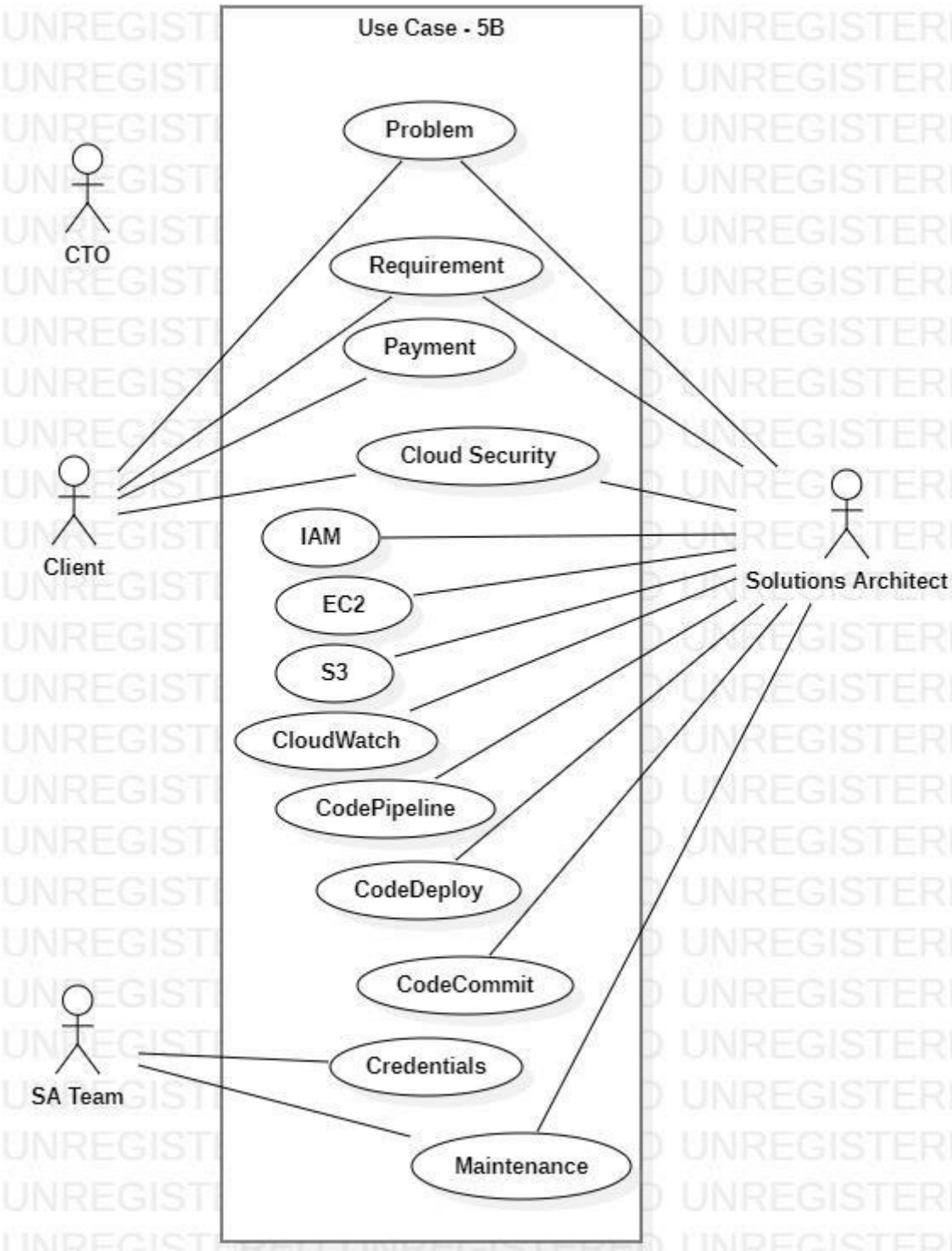
Use Case: 3B


Use Case - 3B
Problem
Requirements
Payment
Cloud Security
Access Resources
Credentials
VPC
Remote Desktop App
Maintenance
CTO
Client
SA Team
Solutions Architect

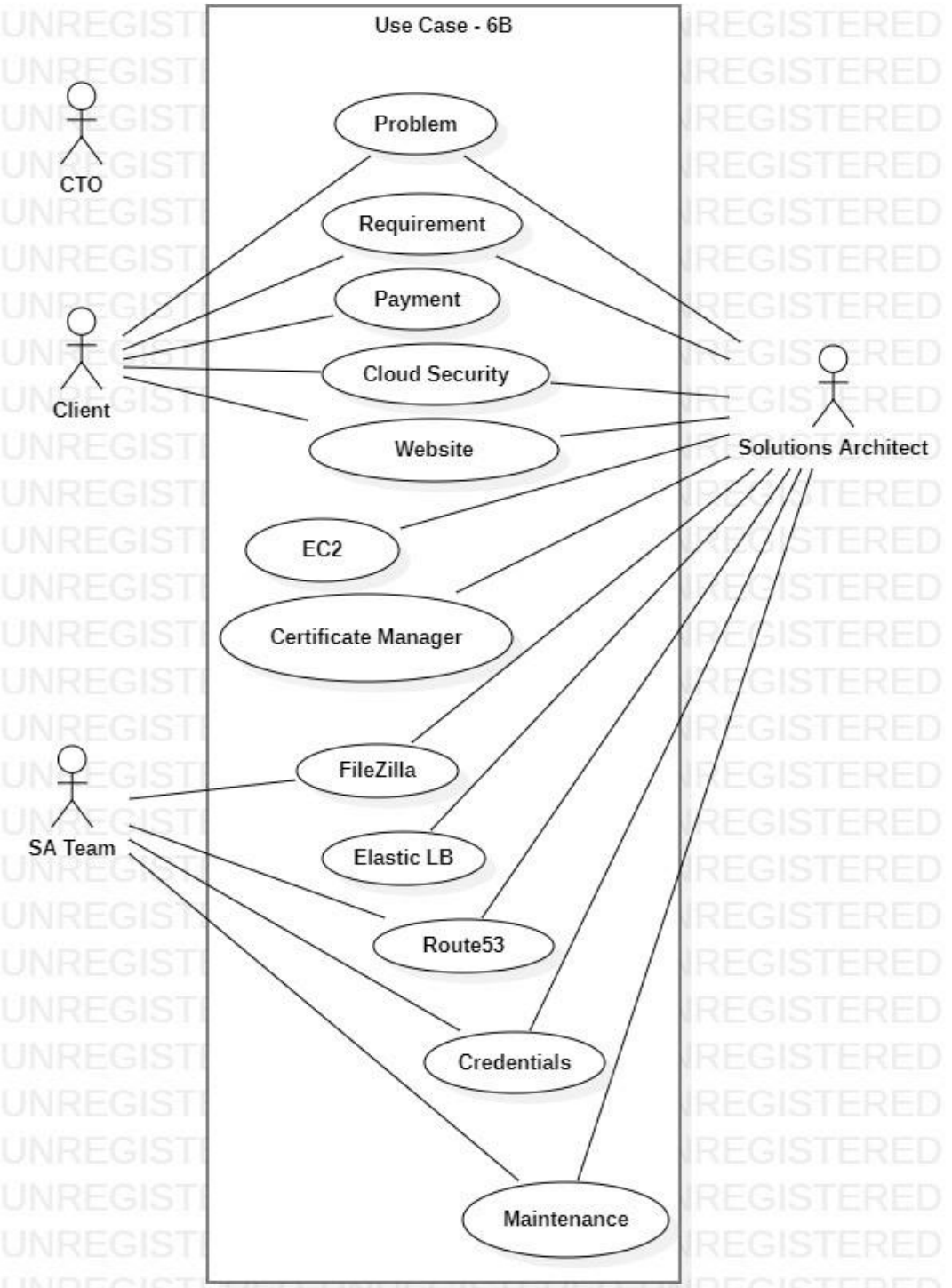
Use Case: 4B


Use Case - 4B
Problem
Requirement
Payment
Cloud Security
Freenom Domain
Route 53
EC2
S3
ELB
Traffic Flow
Maintenance
Credentials
CTO
Client
SA Team
Solutions Architect

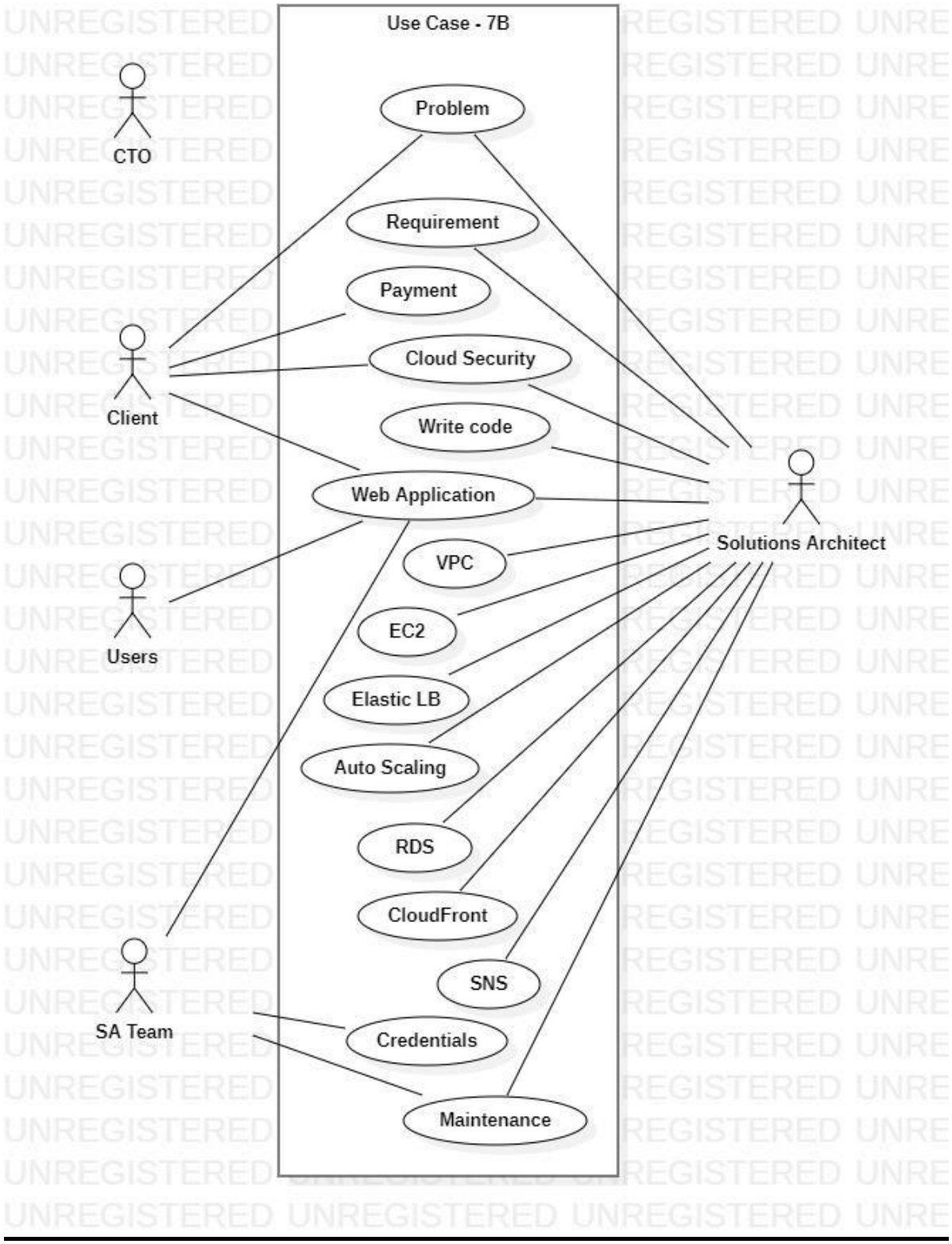
Use Case: 5B


Use Case - 5B
Problem
Requirement
Payment
Cloud Security
IAM
EC2
S3
CloudWatch
CodePipeline
CodeDeploy
CodeCommit
Credentials
Maintenance
CTO
Client
SA Team
Solutions Architect

Use Case: 6B


Use Case - 6B
CTO
Client
SA Team
Solutions Architect
Problem
Requirement
Payment
Cloud Security
Website
EC2
Certificate Manager
FileZilla
Elastic LB
Route53
Credentials
Maintenance

Use Case: 7B

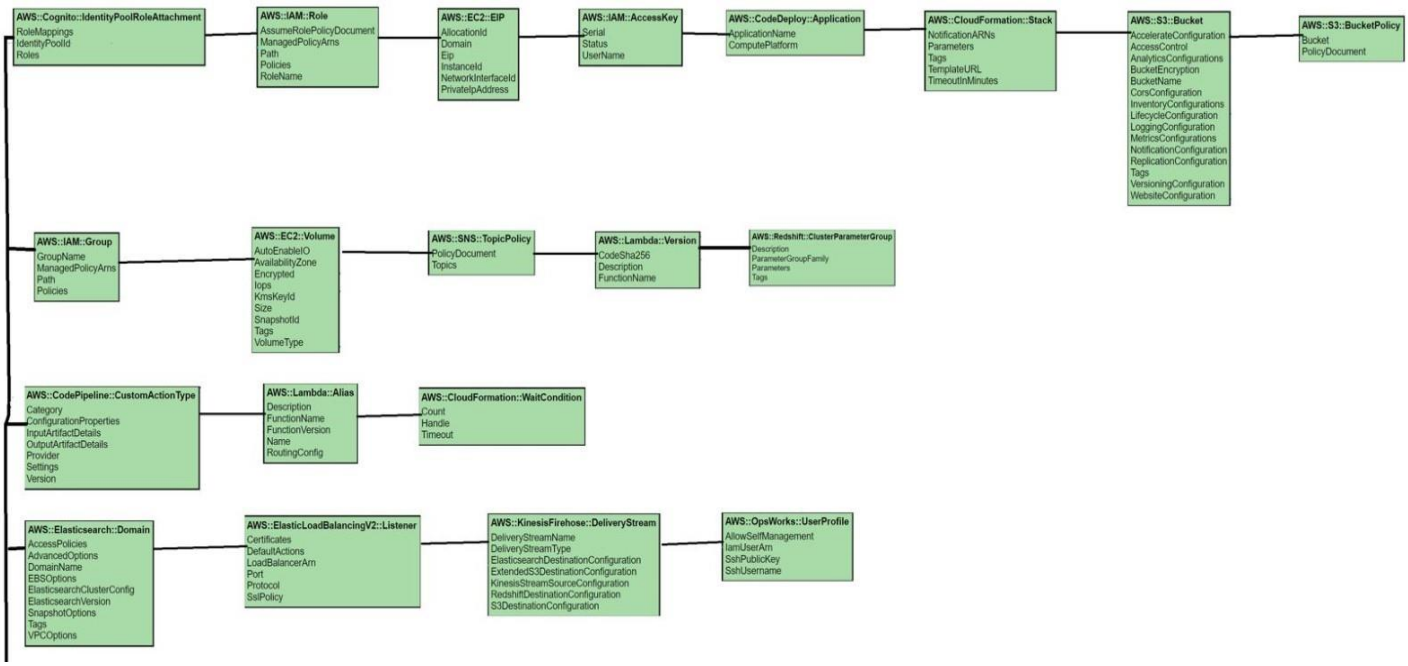

Use Case - 7B
CTO
Client
Users
SA Team
Solutions Architect
Problem
Requirement
Payment
Cloud Security
Write code
Web Application
VPC
EC2
Elastic LB
Auto Scaling
RDS
CloudFront
SNS
Credentials
Maintenance

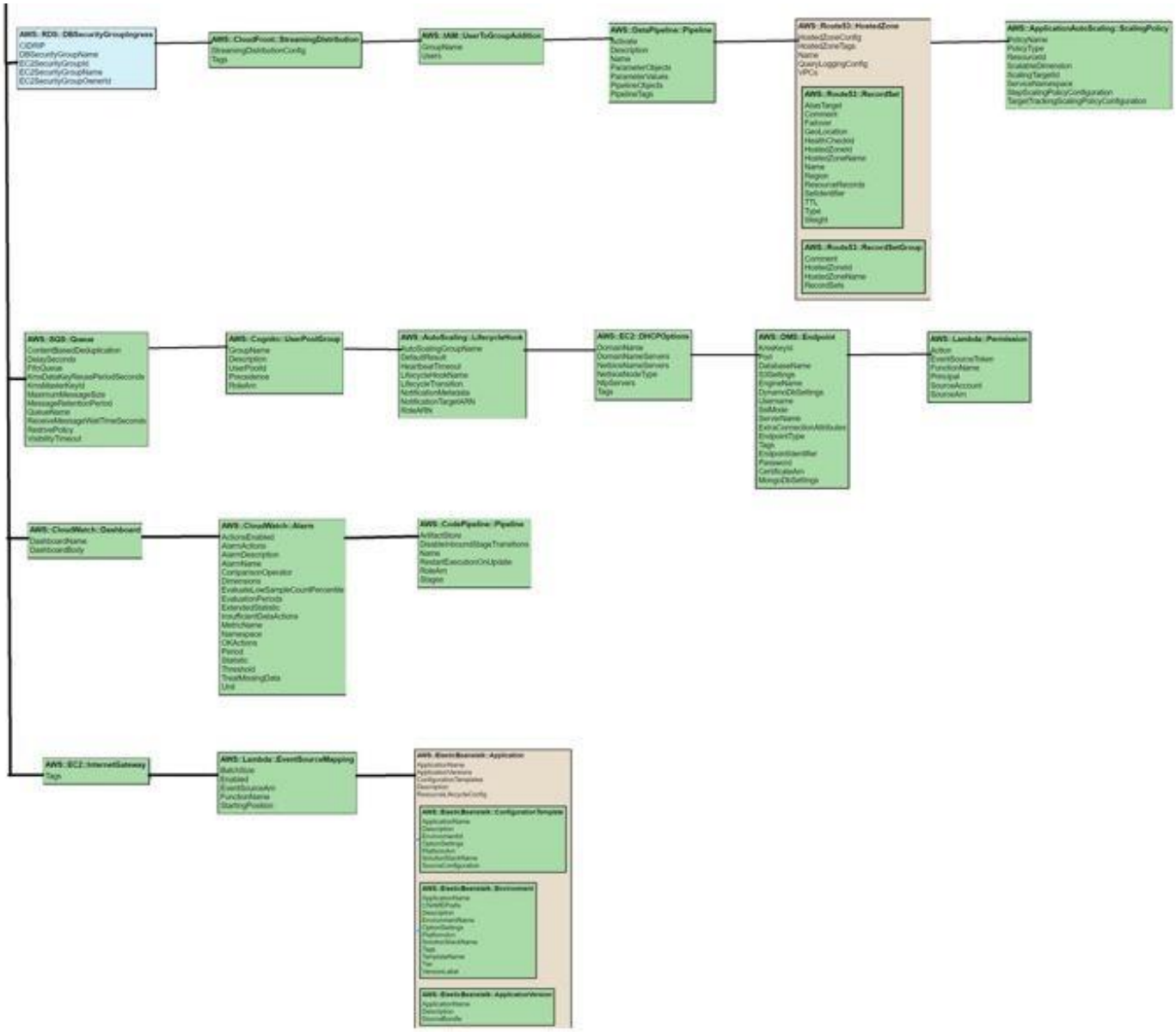
Entity Relationship Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

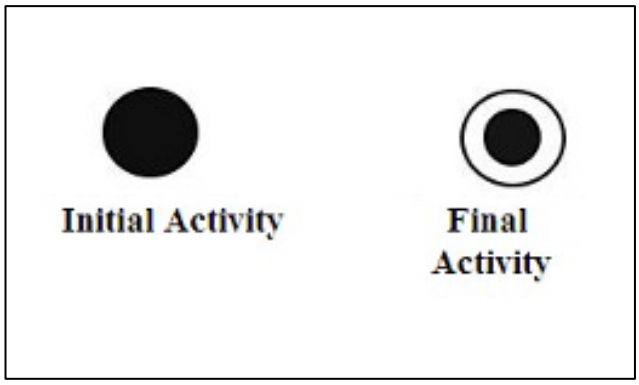


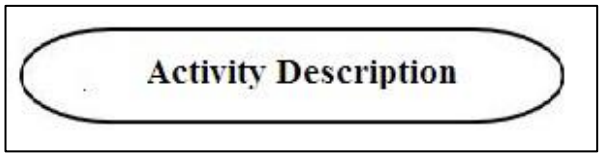


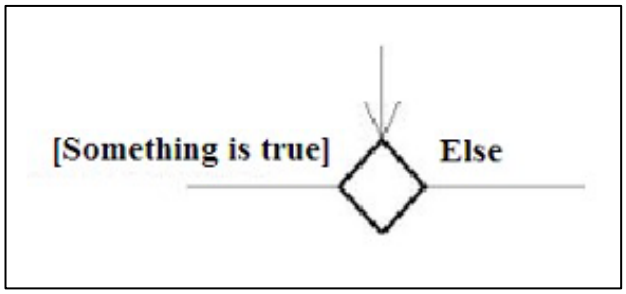
Activity Diagram

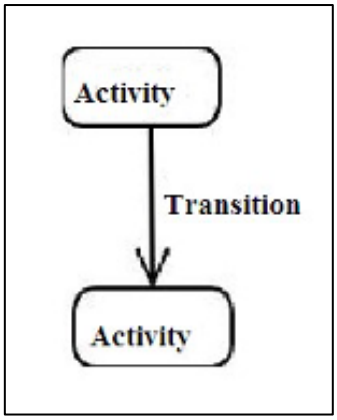
Activity diagrams are useful for analysing a use case by describing what actions needs to be taken place a complicated sequential algorithm modelling applications with parallel processes.

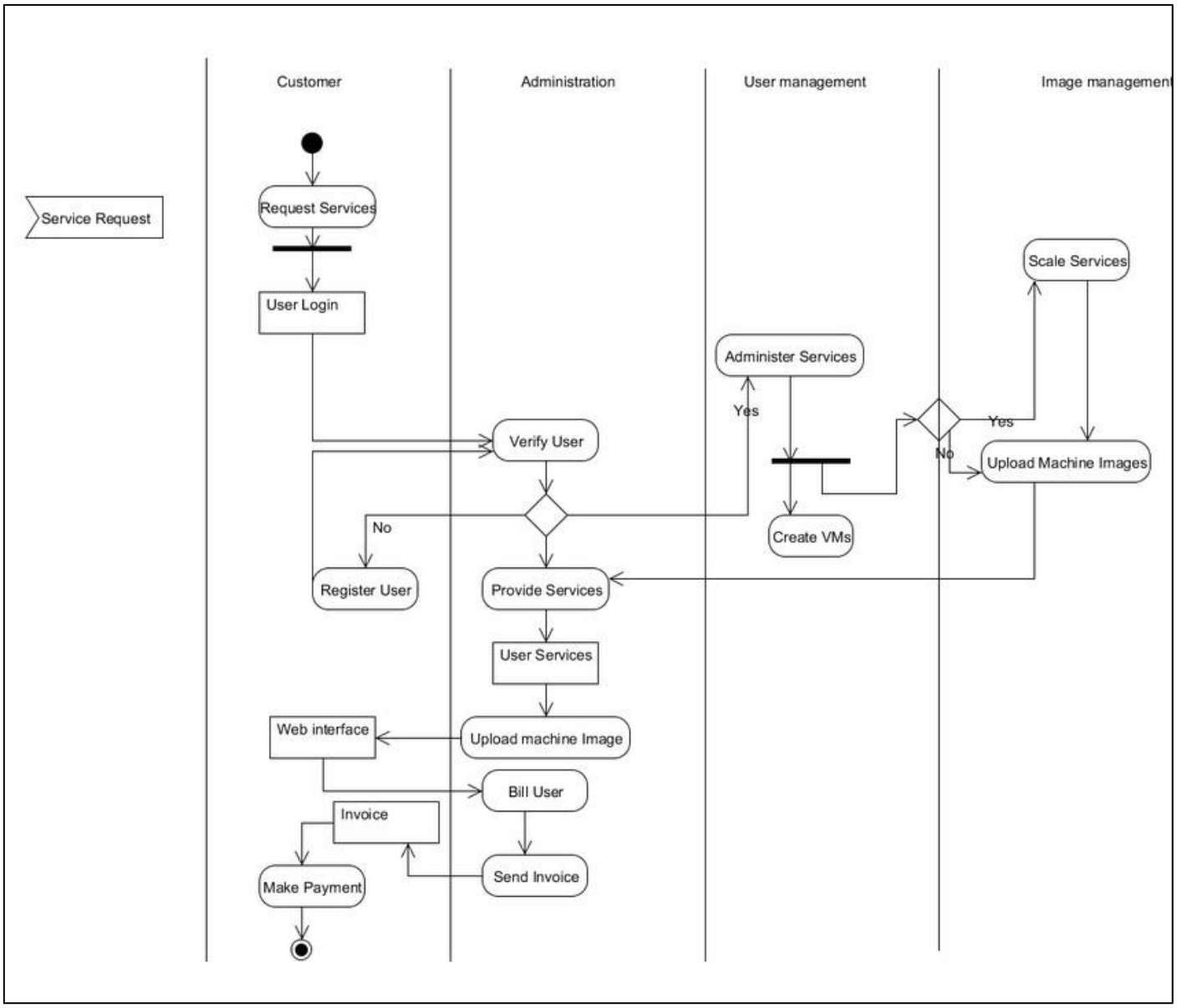
Elements of activity diagram:





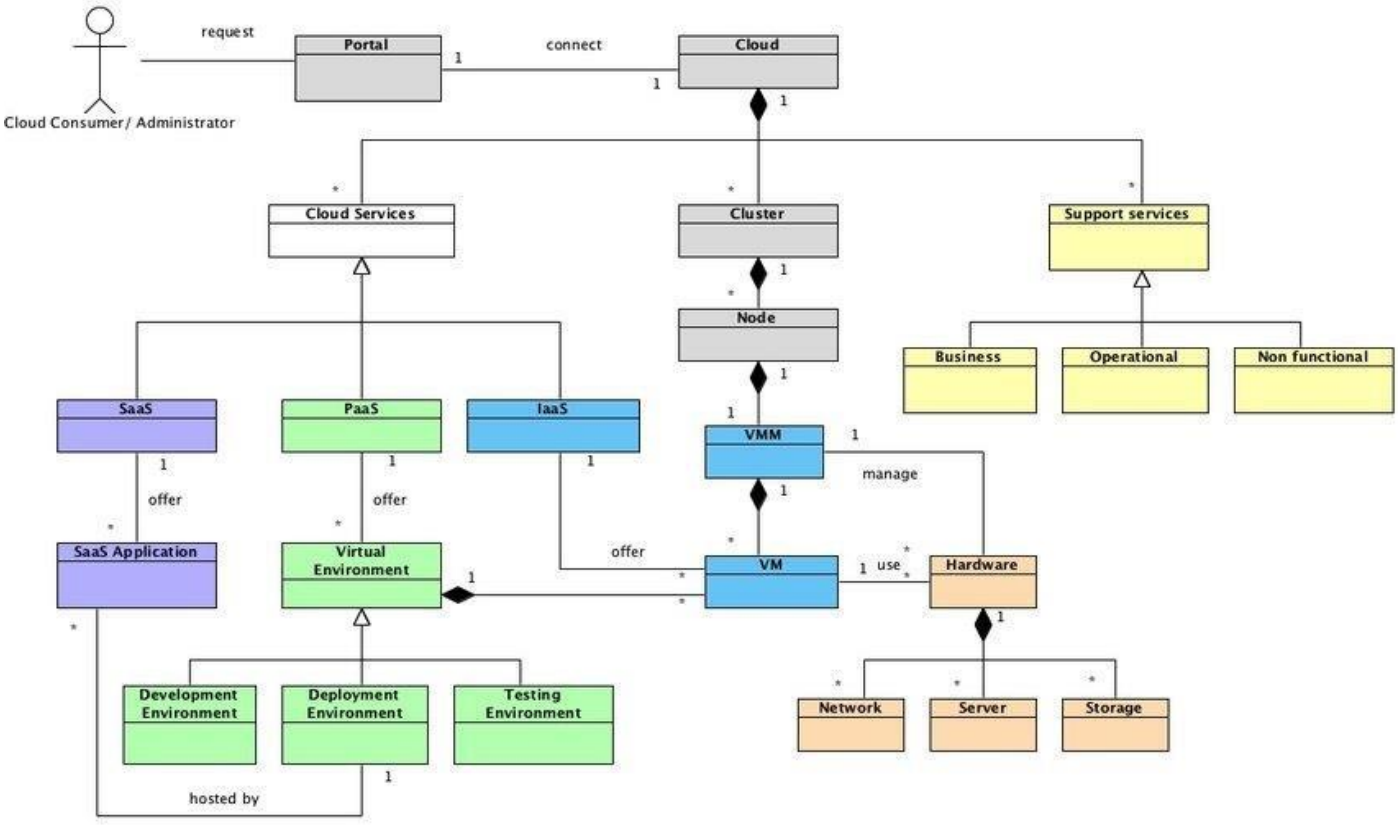





Customer
Administration
User management
Image management
Service Request
Request Services
User Login
Verify User
No
Register User
Provide Services
User Services
Upload machine Image
Web interface
Bill User
Invoice
Send Invoice
Make Payment
Administer Services
Yes
Create VMs
Yes
No
Scale Services
Upload Machine Images

Class Diagram

Class diagrams are used in both the analysis and the design phases. During the analysis phase, a very high-level conceptual design is created. At this time, a class diagram might be created with only the class names shown or possibly some pseudo code-like phrases may be added to describe the responsibilities of the class. The class diagram created during the analysis phase is used to describe the classes and relationships in the problem domain, but it does not suggest how the system is implemented. By the end of the design phase, class diagrams that describe how the system to be implemented should be developed. The class diagram created after the design phase has detailed implementation information, including the class names, the methods and attributes of the classes, and the relationships among classes.



Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modelling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

Purpose of Deployment Diagrams

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.
- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes

Deployment Diagram at a Glance

Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and for managing executable systems through forward and reverse engineering.

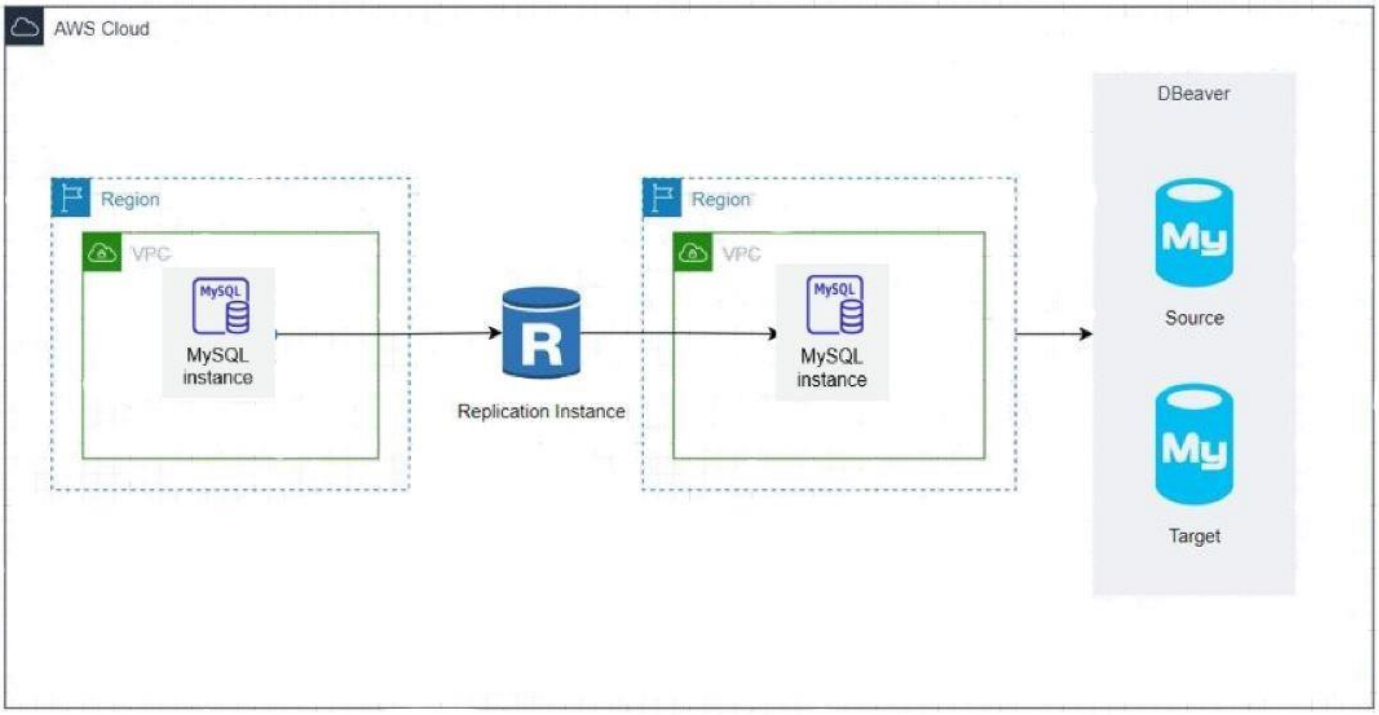
Node: A node is a physical resource that executes code components. A node is either hardware or a software element. It is shown as a three-dimensional box shape.

Node-Instance: A node instance can be shown on a diagram. An instance can be distinguished from a node by the fact that its name is underlined and has a colon before its base node type.

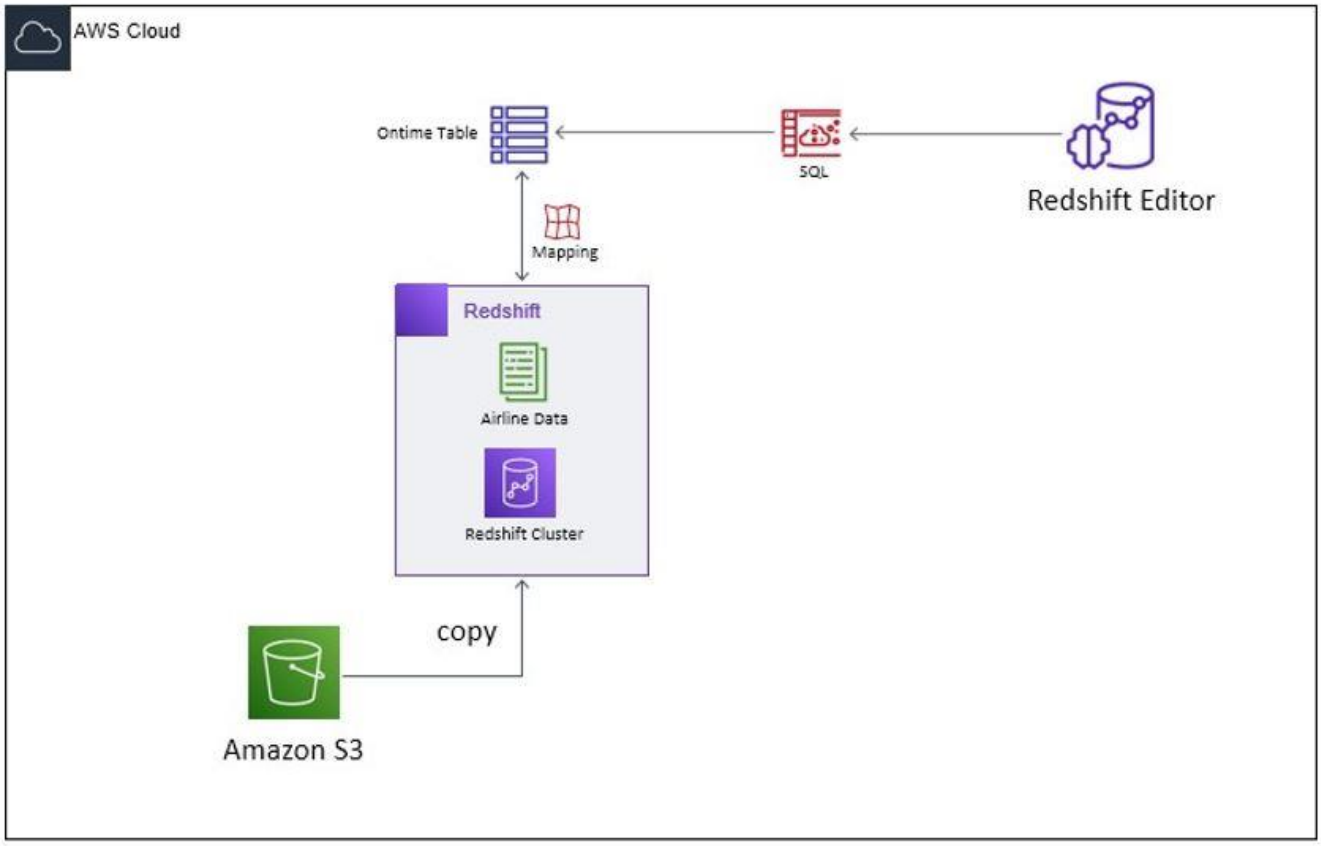
Node Stereotype: Stereotypes are provided for nodes, namely <<cd-rom>>, <<computer>> etc

Section: A

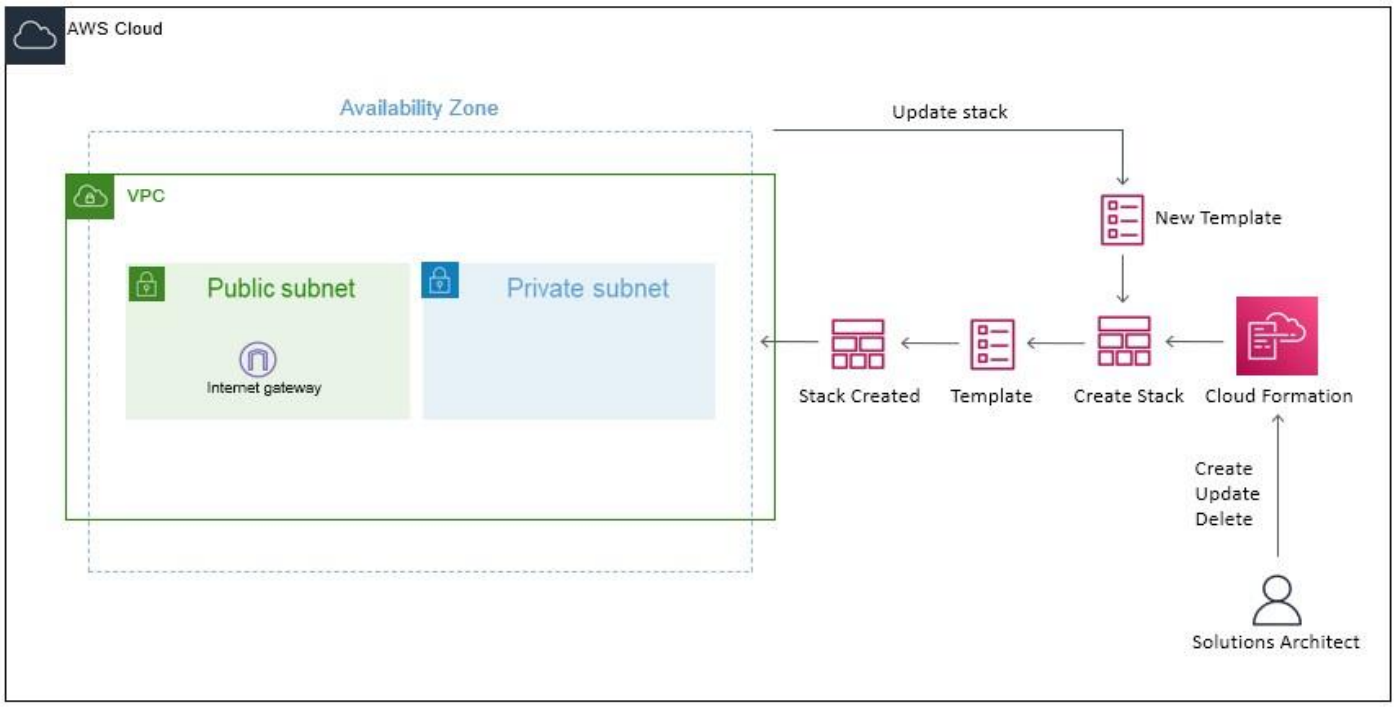
Use Case: 1



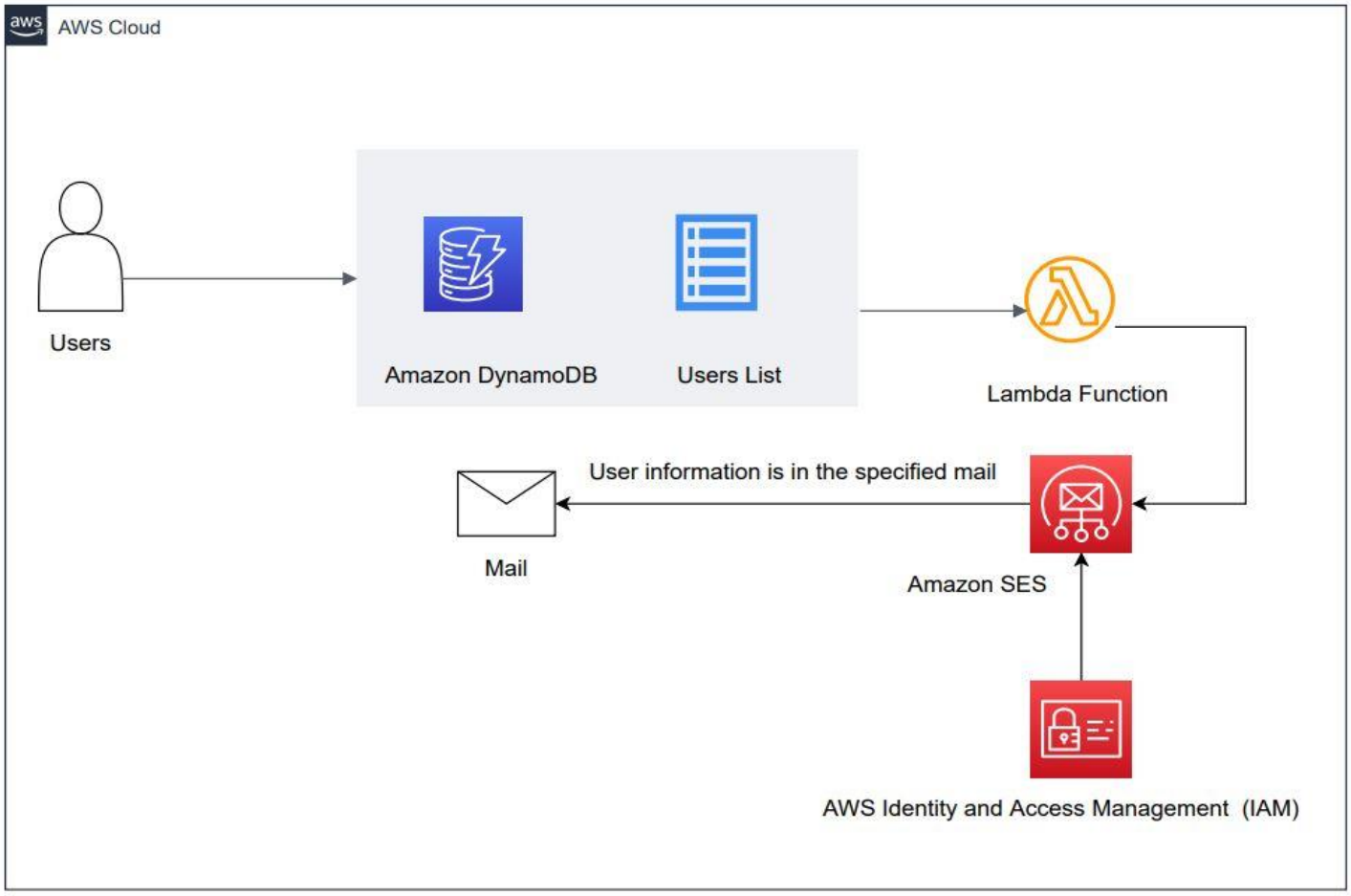
Use Case: 2


AWS Cloud
Ontime Table
SQL
Redshift Editor
Mapping
Redshift
Airline Data
Redshift Cluster
copy
Amazon S3

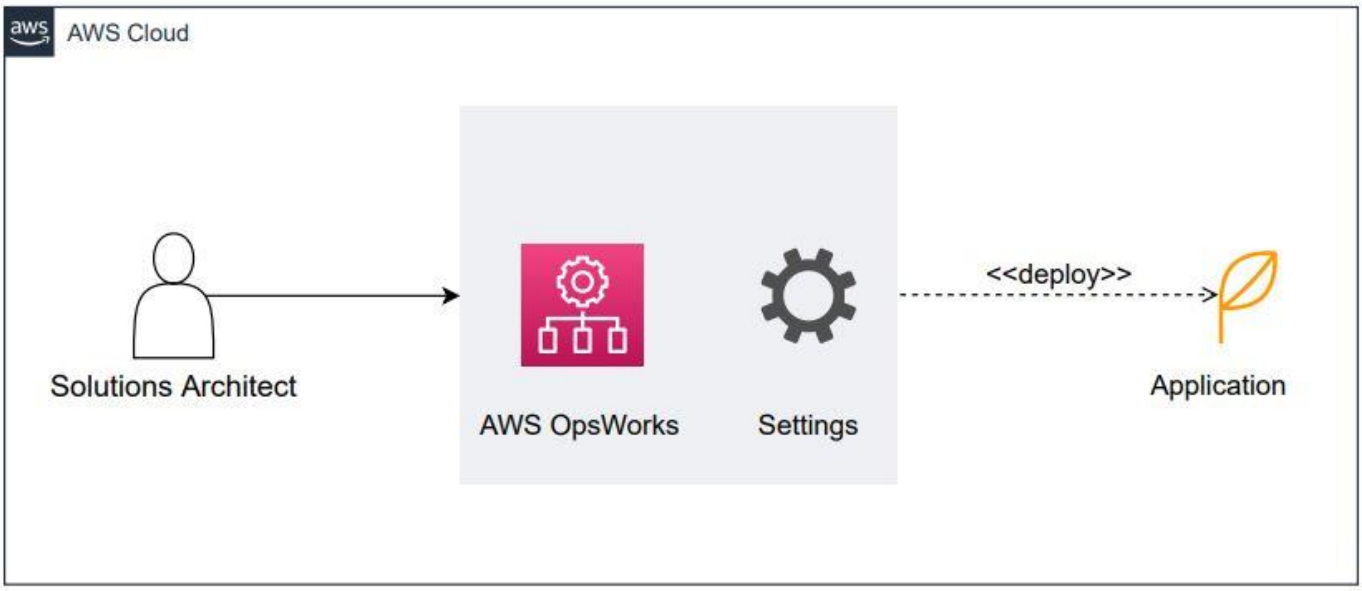
Use Case: 3


AWS Cloud
Availability Zone
VPC
Public subnet
Internet gateway
Private subnet
Update stack
New Template
Stack Created
Template
Create Stack
Cloud Formation
Create
Update
Delete
Solutions Architect

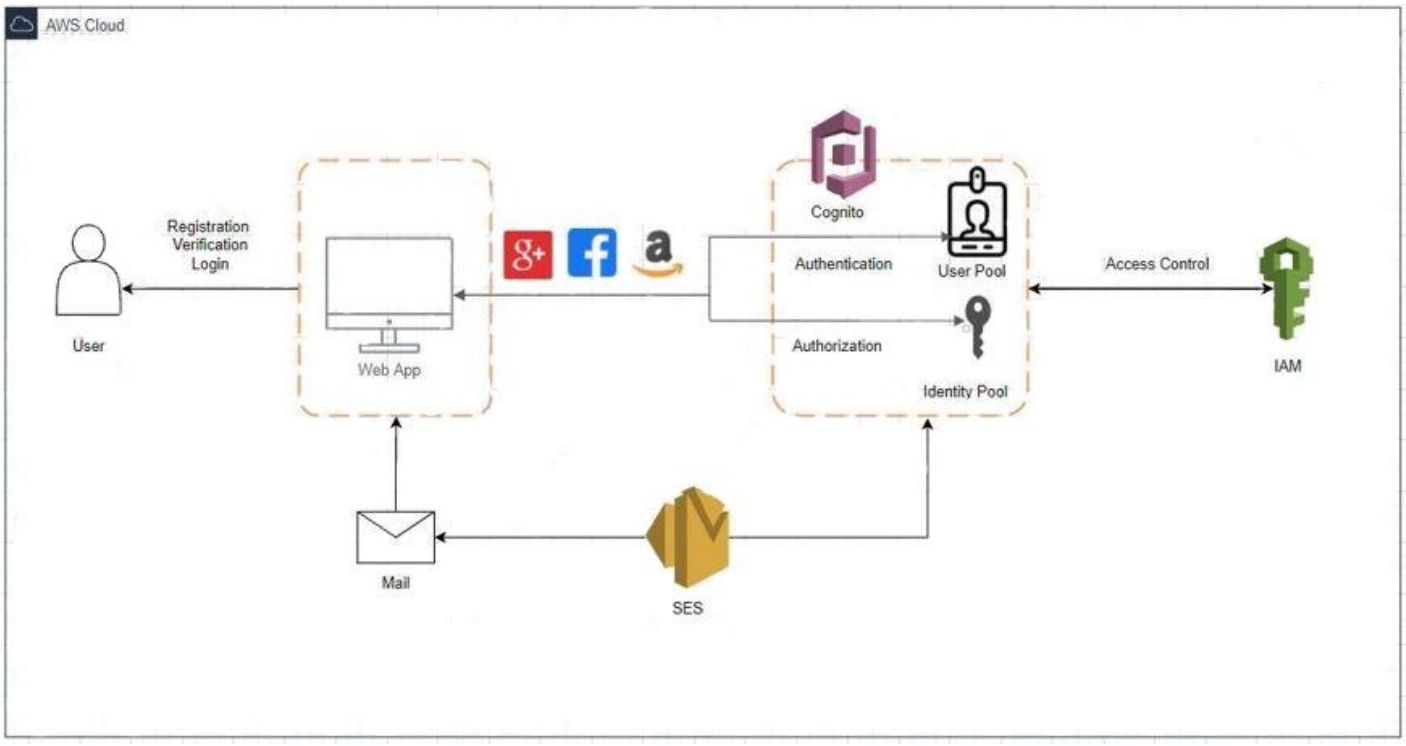
Use Case: 4


AWS Cloud
Users
Amazon DynamoDB
Users List
Lambda Function
User information is in the specified mail
Mail
Amazon SES
AWS Identity and Access Management (IAM)

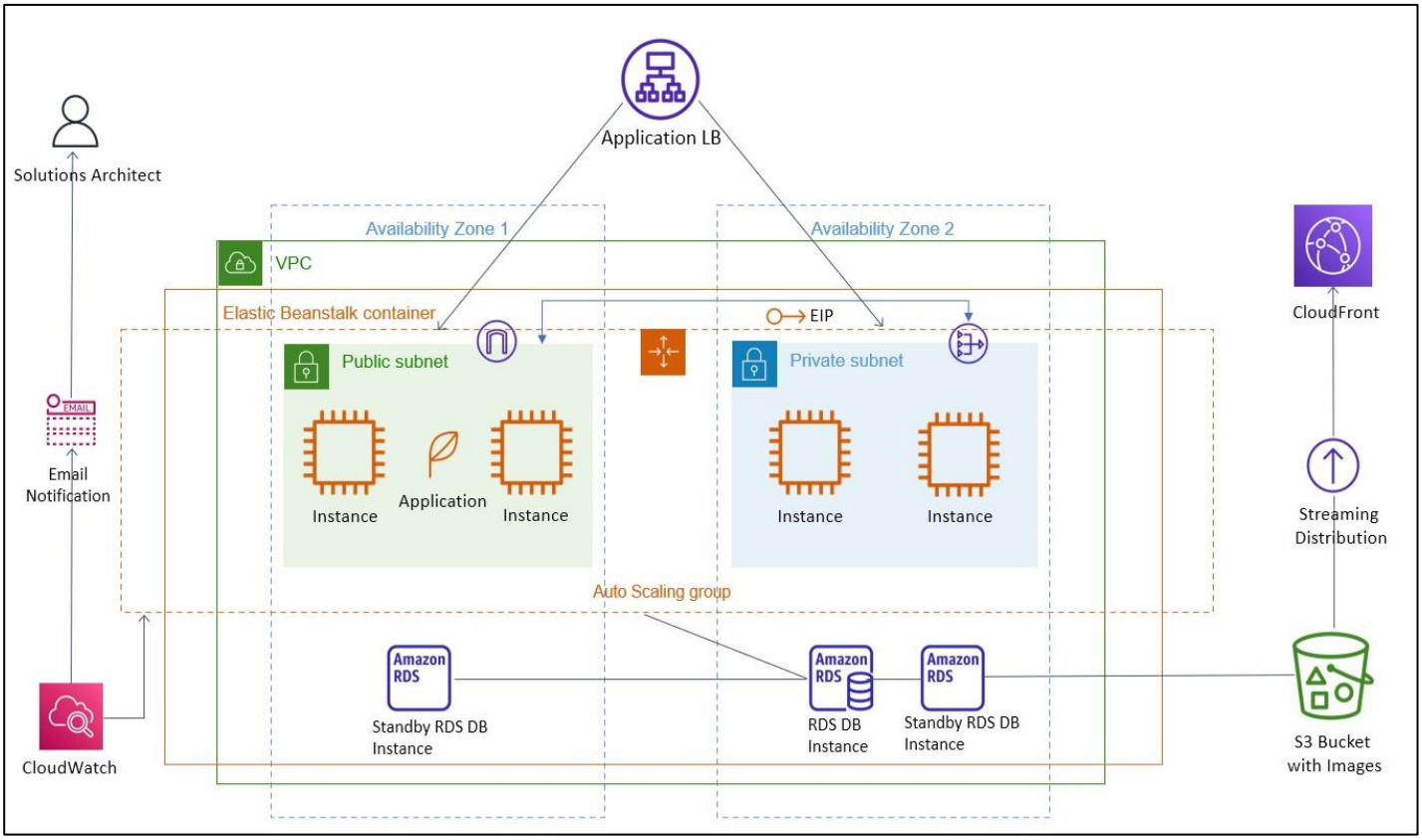
Use Case: 5



Use Case: 6

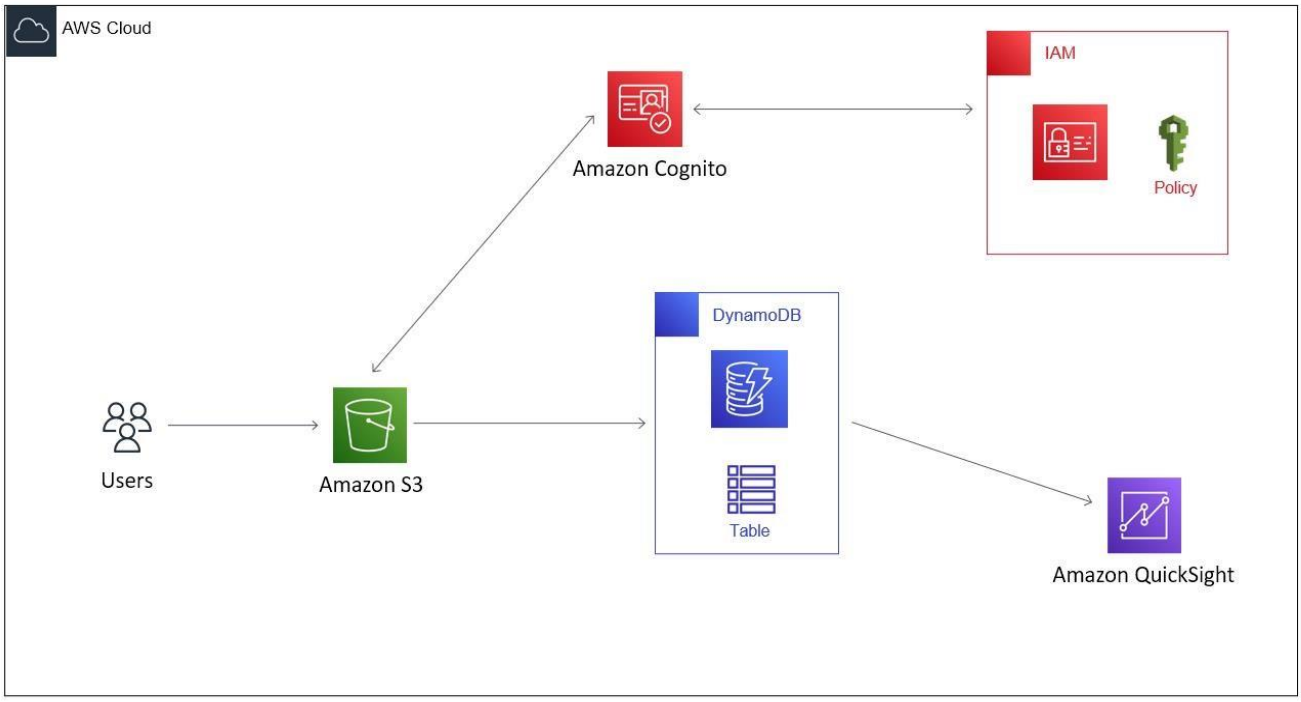


Use Case: 7


Application LB
Solutions Architect
Availability Zone 1
Availability Zone 2
VPC
Elastic Beanstalk container
EIP
Public subnet
Private subnet
Instance
Application
Instance
Instance
Instance
CloudFront
Email
Notification
Streaming
Distribution
Auto Scaling group
Amazon RDS
Standby RDS DB Instance
Amazon RDS
RDS DB Instance
Amazon RDS
Standby RDS DB Instance
CloudWatch
S3 Bucket with Images

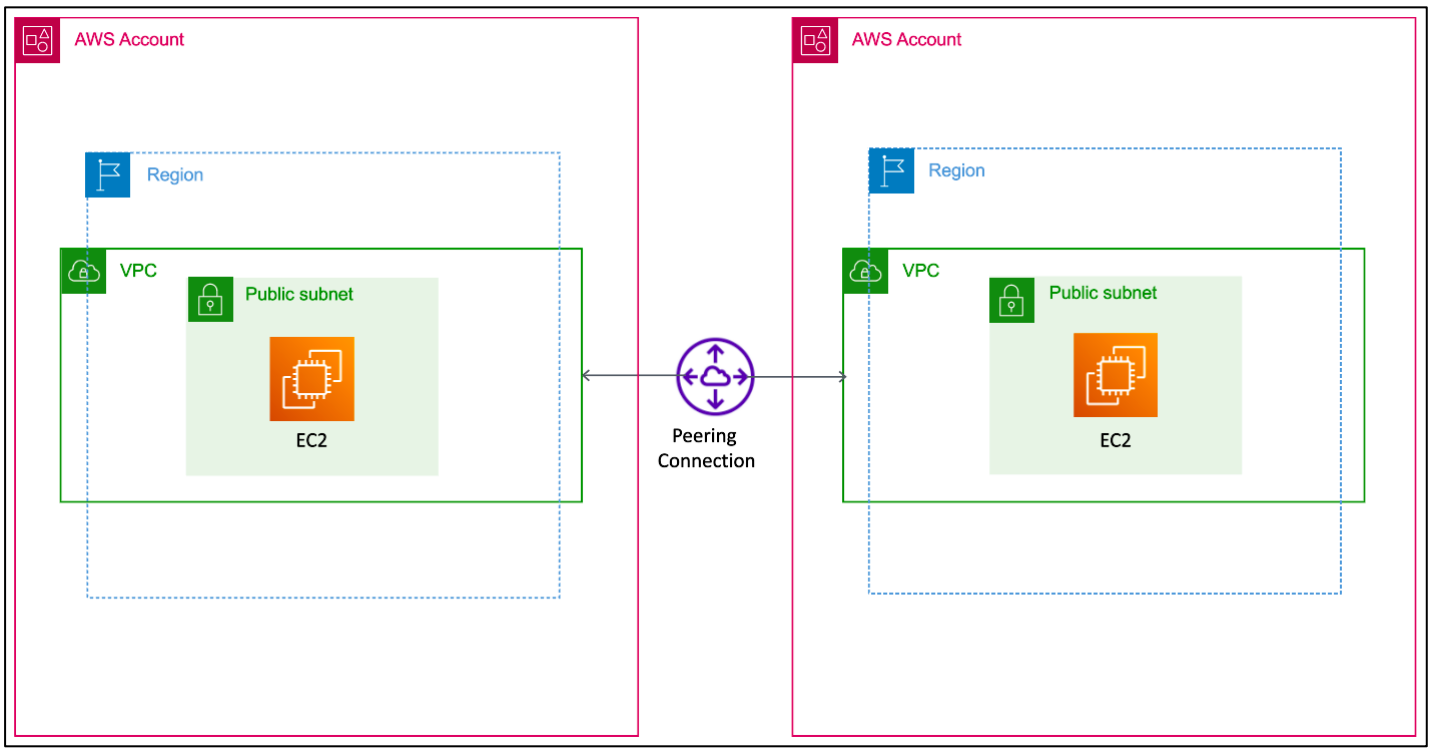
Section B:

Use Case: 1

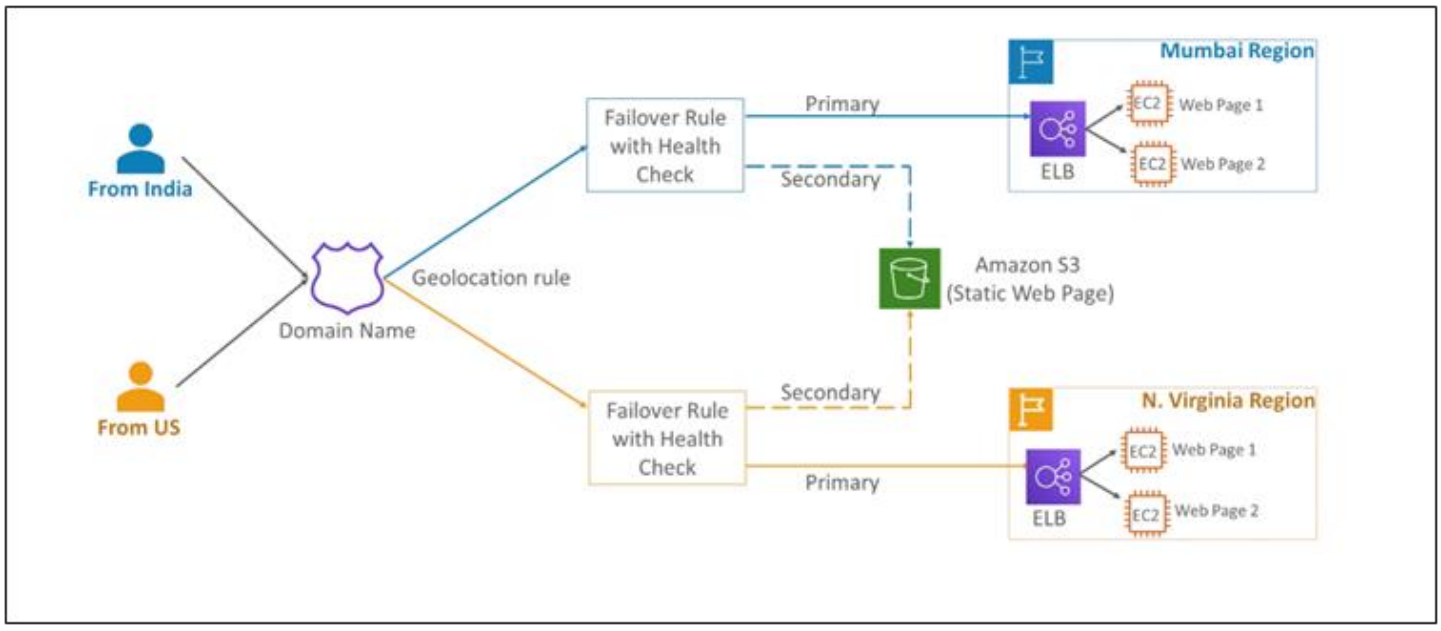


Use Case: 2

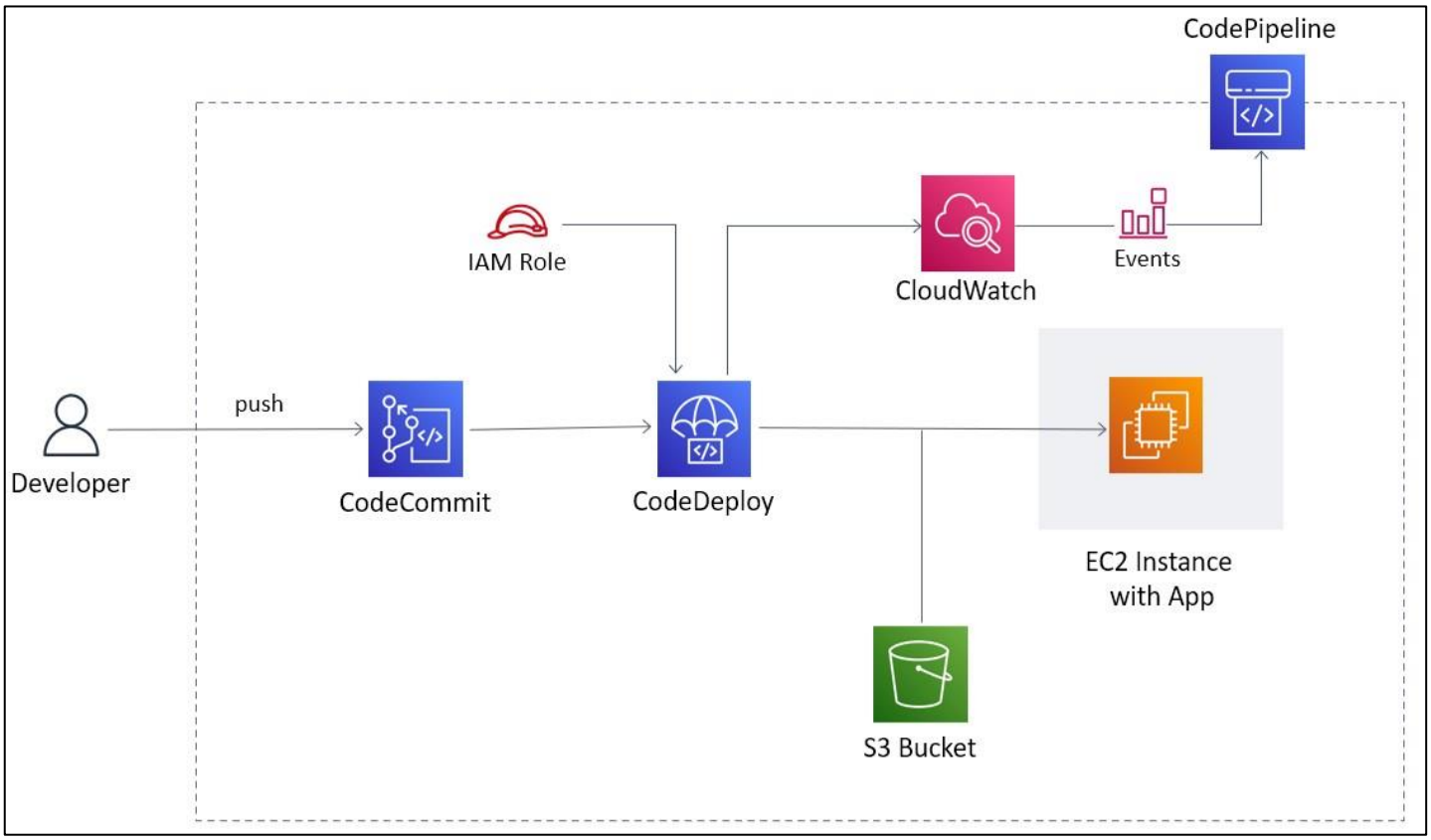
Use Case: 3



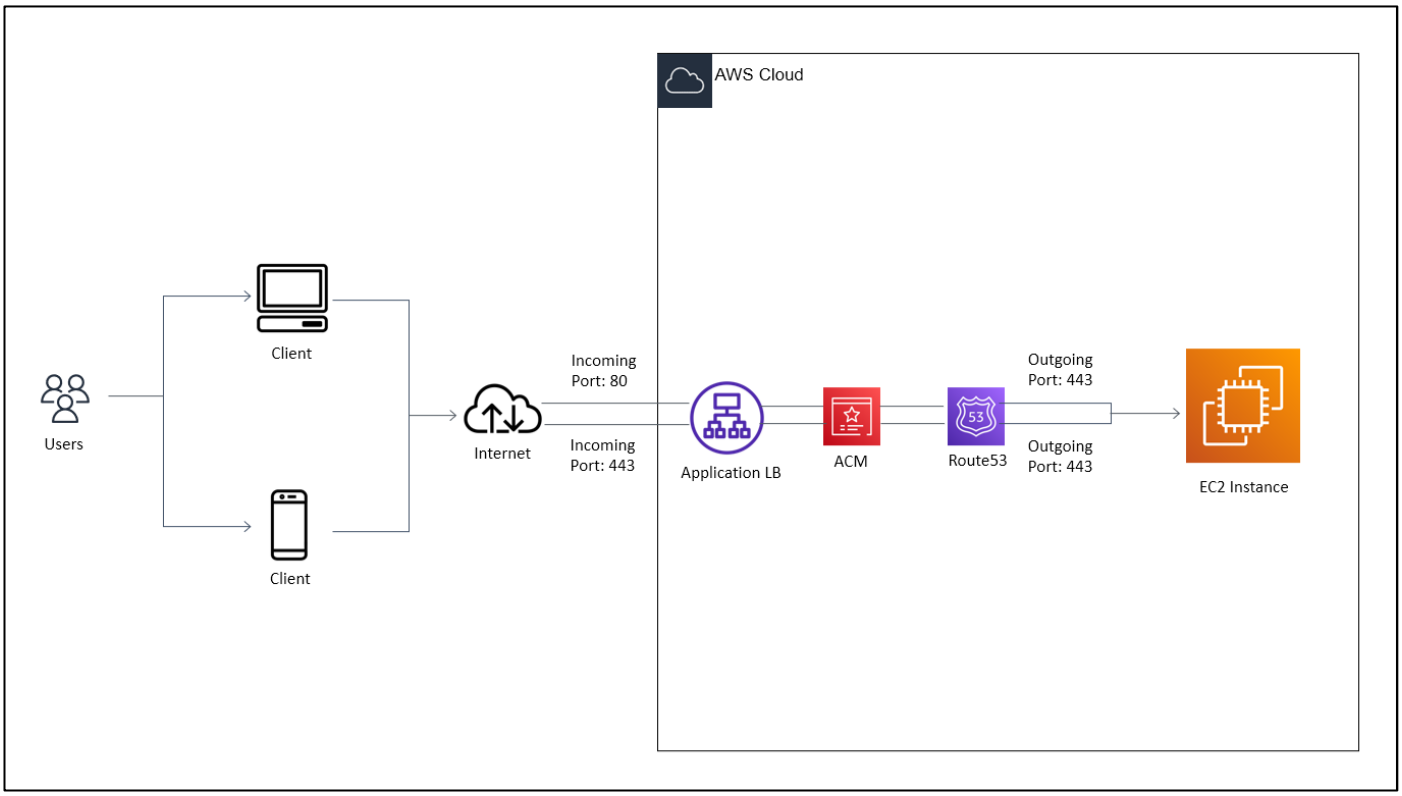
Use Case: 4



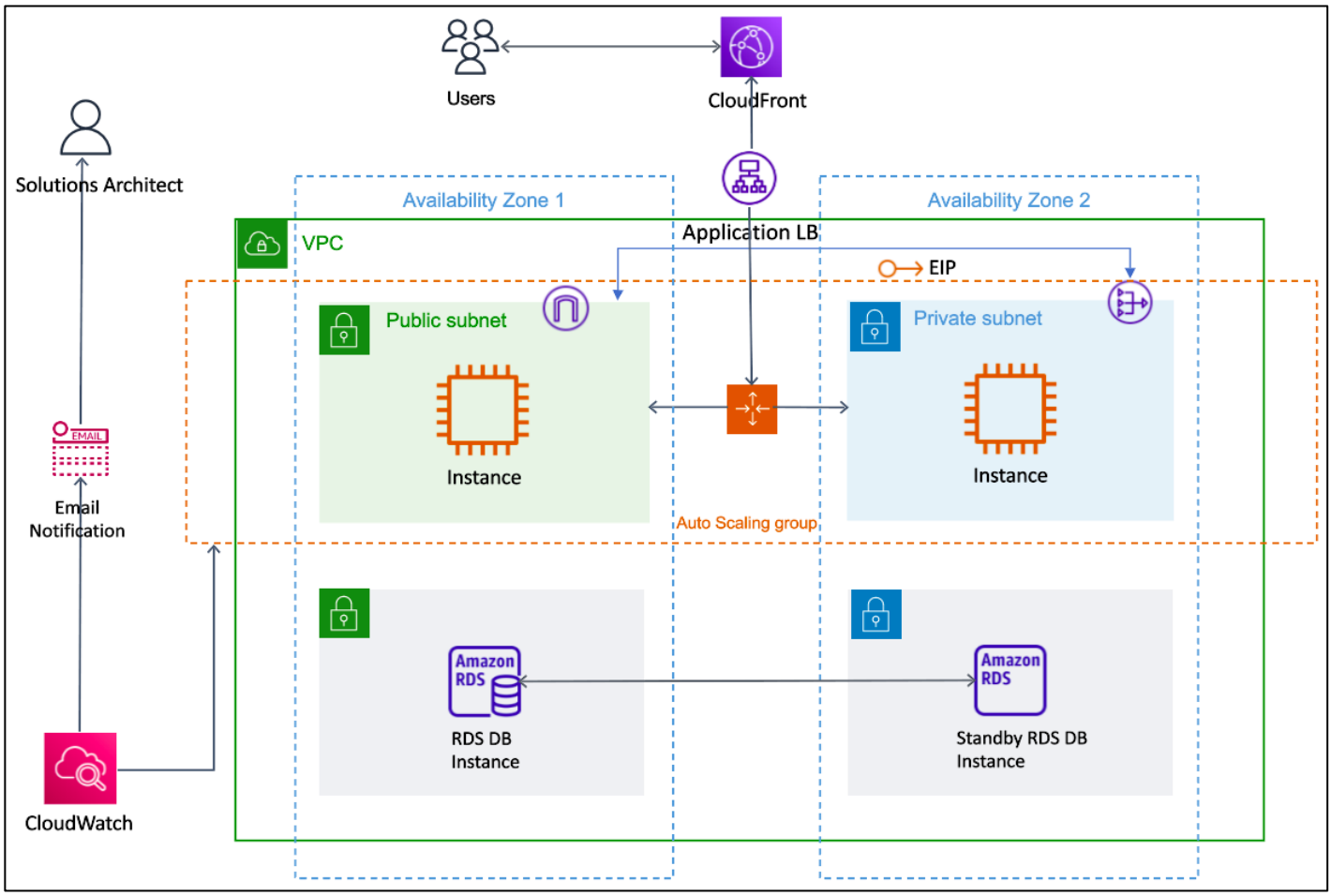
Use Case: 5


CodePipeline
IAM Role
CloudWatch
Events
push
Developer
CodeCommit
CodeDeploy
EC2 Instance
with App
S3 Bucket

Use Case: 6

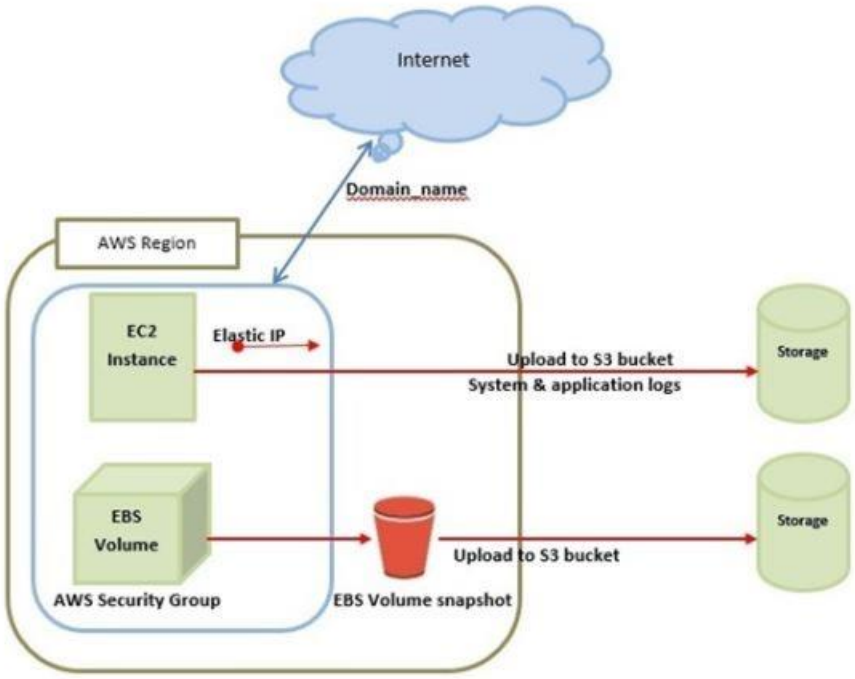

AWS Cloud
Client
Users
Client
Internet
Incoming
Port: 80
Incoming
Port: 443
Application LB
ACM
Route53
Outgoing
Port: 443
Outgoing
Port: 443
EC2 Instance

Use Case: 7


Users
CloudFront
Solutions Architect
Availability Zone 1
Availability Zone 2
VPC
Application LB
EIP
Public subnet
Private subnet
Instance
Instance
Email
Email Notification
Auto Scaling group
Amazon RDS
RDS DB Instance
Amazon RDS
Standby RDS DB Instance
CloudWatch

SYSTEM DESIGN AND ARCHITECTURE

The below diagram is the basic structure of **AWS Architecture** where it provides cloud computing services accordingly.



It is considered as the basic structure of AWS architecture or AWS EC2. Simply, EC2 is also called Elastic Compute cloud which will allow the clients or else the users of using various configurations in their own project or method as per their requirement. There are also different amazing options such as pricing options, individual server mapping, configuration server, etc. S3 which is present in the AWS architecture is called Simple Storage Services. By using this S3, users can easily retrieve or else store data through various data types using Application Programming Interface calls. There will be no computing element for the services as well.

Key Components

1. Load Balancing:

2. Elastic Load Balancing:

3. Amazon CloudFront:

4. Elastic Load Balancer:

5. Security Management:

6. Elastic Cache:

7. Amazon RDS:

What is the Importance of AWS Architecture?

AWS architecture diagrams are mostly used to enhance the solution with the help of powerful drawing tools, plenty of pre-designed icons of Amazon, and the various simple icons that are used for the creation of the AWS diagrams of the Architecture.

AWS Architecture also makes sure to provide incredible services based on the web technologies, uploading and unloading of virtual servers, the selection service and the service of transferring messages, etc. Moreover, the resources of AWS can be available worldwide and can also be able to deploy solutions exactly where the customers are required of them.

Here are the main benefits of AWS Architecture and its uses:

- It has a wide range of benefits from the massive economies of the scale
- It also helps to stop guessing capacity and can easily achieve higher economic rates which can easily translate from the lower prices to the upper prices.
- It can easily enhance the agility and the speed that can reduce the time to complete a task.

Top 5 Pillars of AWS Well-Architected Framework

1. Security

Security is the basic thing that matters a lot in AWS Technology. It is entirely an infra design that can easily serve complete data protection, infrastructure protection, privilege Management of all AWS accounts and identifying the security breach with certain detective controls reliably. Basically, it follows certain design principles that are:

- One can apply security at every level
- Implementation of Principle of Least Privilege
- Enable Traceability
- Secured System Applications, data, and OS Level
- Automate Security Best practices

2. Reliability

AWS is a good architecture that has come up with well-planned foundations and monitoring in place with various mechanism rates to handle demand rates as per

requirements. The system can easily detect the failure and must come out with an optimized solution. The design principles are in the given way like

- Test Recovery Procedures
- Usage of Horizontal Scalability in an increment of system availability
- Recovery from failure in an automatic way
- Add or else Removing resources
- Manage Changes in the automation

3. Performance Efficiency

Performance Efficiency is kept the focus on the efficient use of computing resources to meet the given requirements in a reliable manner. It is also to maintain efficiency as demand changes and technology evolves. The design principles go in the given way:

- Democratize advanced Technologies
- Globally Deploying of the given system at a minimal cost of lower latency
- To keep aside of operational burden, use a serverless architecture
- Various comparative testing and configurations for better performance

4. Cost Optimization

It is one of the main pillars of AWS Architecture that is completely optimizing costs, unused, elimination or else sub-optimal resources. It is most probably considered with the matching supply with demand and being aware of expenditure and optimizes over costs. The following design principles are delivered in the cost optimization are:

- Adopting of consumption model
- High benefits values from economies of scale
- Stop investing in Data Center Operations
- Analyzing and Attribute Expenditure
- Usage of Well Managed services for reducing some cost of ownership

5. Operational Excellence

Generally, this Operational Excellence of the product is checked for reliability, agility, and performance. The most optimized way is to standardize and manage workflows in an efficient manner. It mostly suggests various principles like:

- Performing Operations with code
- Making of some regular incremental changes
- Test for certain responses to unexpected events
- Learning new from the events and failures of certain operations
- Operations Procedures are always kept current

AWS Serverless architecture helps to build and run applications without a second thought of servers. These Serverless applications do not require any of the provision, managing and scaling of servers in a reliable manner. The applications can easily be built to the backend service and everything is required to run the applications and scale up it.

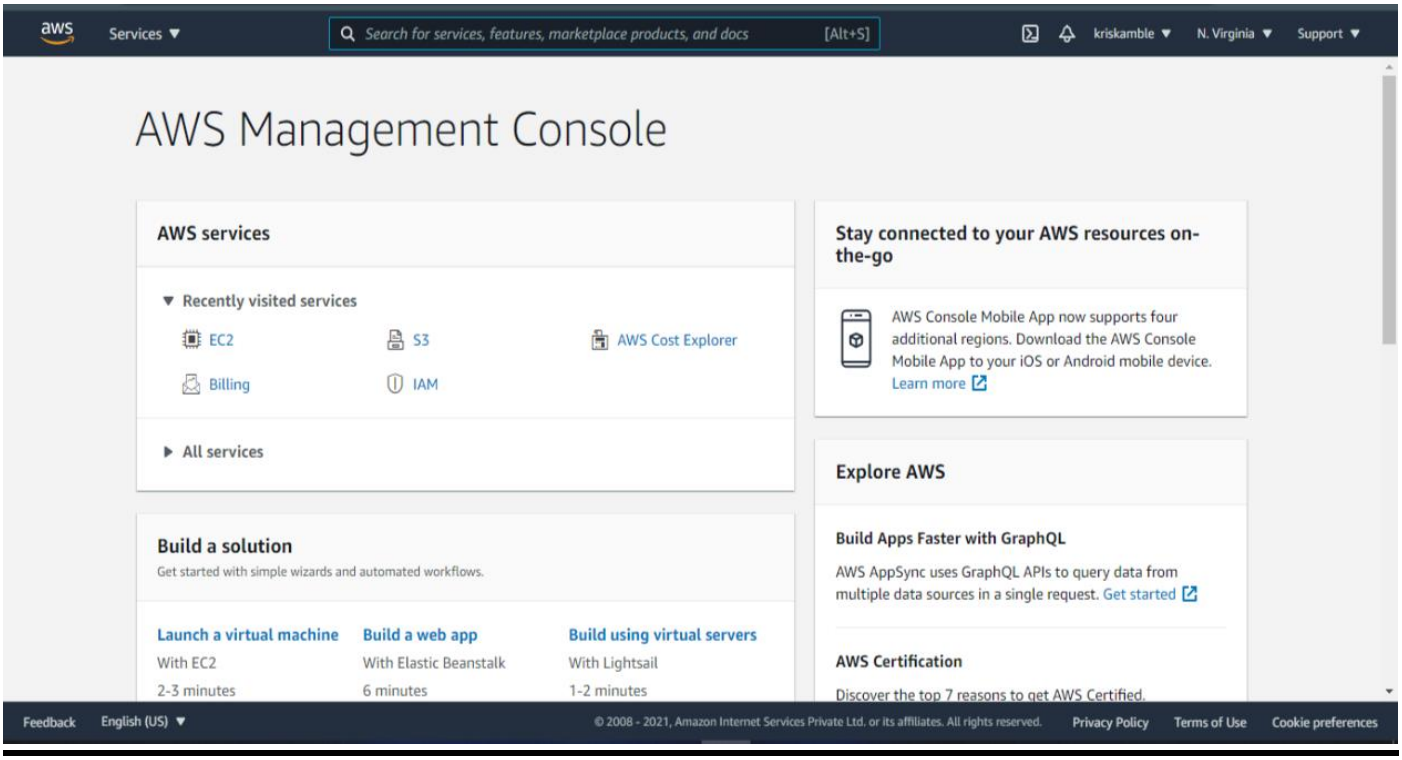
USER INTERFACE

AWS Management Console

Everything you need to access and manage the AWS cloud — in one web interface.

The AWS Management Console brings the unmatched breadth and depth of AWS right to your computer or mobile phone with a secure, easy-to-access, web-based portal. Discover new services, manage your entire account, build new applications, and learn how to do even more with AWS.

The console makes it easy to find new AWS services, configure services, view service usage, and so much more. From updating user groups to building applications to troubleshooting issues, with the Console, you can take action quickly.



CODING

SECTION – A

Use Case – 2

```
create table ontime ( Year integer,
Month integer,
DayofMonth integer,
DepTime integer,
CRSDepTime integer,
Origin varchar(120),
Dest varchar(120)
);
copy ontime from 's3://akul-airline/data.txt' iam_role
'arn:aws:iam::304000509264:role/Role4RedShift-S3RO' delimiter ',' region 'us-west-2’
select * from ontime;
```

Use Case - 4:

```
var aws = require('aws-sdk');
var ses = new aws.SES({region: 'us-east-1'});
exports.handler = function(event, context) {
console.log("Incoming: ", event);
// var output = querystring.parse(event);
var eParams = {
Destination: {
ToAddresses: ["abc@gmail.com"]//give the email ID which is verified by SES
}, Message: {
Body: {Text: {
Data: "Hurray a new user has been created !!!!"
}
},
Subject: { Data: "New User"} }, Source: "xyz@gmail.com" //give the email ID which is
verified by SES
 };
console.log('===SENDING EMAIL===');
var email = ses.sendEmail(eParams, function(err, data){
if(err) console.log(err);
 else {
console.log("===EMAIL SENT===");
console.log(data);
console.log("EMAIL CODE END");
console.log('EMAIL: ', email);
```

```
 context.succeed(event);
 }
 });
};
```

Use Case - 7:

```
#become root
sudo su
#get the list of softwares
apt-get update
#install python and pip
apt-get install python2.7 python-pip -y
#install python aws sdk
pip install boto3
exit
mkdir .aws
echo -e "[default]\nregion=us-east-2" > .aws/config


if __name__ == '__main__':
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('users')
    response = table.put_item(
        Item={
            'userid': "123"
            'name': "Akul Thayyil",
            'city': "Hyderabad",
            'country': "India"
        }
    )
    print("Put movie succeeded:")

Use Case 7:
var port = process.env.PORT || 3000,
    http = require('http'),
    fs = require('fs'),
    html = fs.readFileSync('index.html');

var log = function(entry) {
    fs.appendFileSync('/tmp/sample-app.log', new Date().toISOString() + ' - ' + entry +
'\n');
};
```

```
var server = http.createServer(function (req, res) {
    if (req.method === 'POST') {
        var body = '';

        req.on('data', function(chunk) {
            body += chunk;
        });

        req.on('end', function() {
            if (req.url === '/') {
                log('Received message: ' + body);
            } else if (req.url = '/scheduled') {
                log('Received task ' + req.headers['x-aws-sqsd-taskname'] + ' scheduled at
' + req.headers['x-aws-sqsd-scheduled-at']);
            }

            res.writeHead(200, 'OK', {'Content-Type': 'text/plain'});
            res.end();
        });
    } else {
        res.writeHead(200);
        res.write(html);
        res.end();
    }
});

// Listen on port 3000, IP defaults to 127.0.0.1
server.listen(port);

// Put a friendly message on the terminal
console.log('Server running at http://127.0.0.1:' + port + '/');
```

SECTION – B

Use Case – 1

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Survey for abc.com</title>
</head>

<body>
  <div id="submit-survey">
    <b>Your email :</b><input autofocus size="23" type="email" id="email"
      placeholder="Enter your email here" /><br /><br />
    <b>Your City :</b><input autofocus size="23" type="text" id="city" placeholder="Enter
your City here" /><br /><br />
    <b>Your feedback :</b><input autofocus size="100" type="text" id="feedback"
      placeholder="Enter your feedback here" /><br /><br />
    <p id="result"><b>Enter the survey above then click Submit</b></p>
    <button class="btn default" onClick="insertSurveyData()">Submit</button>
  </div>
  <script src="https://sdk.amazonaws.com/js/aws-sdk-2.410.0.min.js"></script>
  <script type="text/javascript">

    // Initialize the Amazon Cognito credentials provider
    AWS.config.region = 'us-east-1';
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({ IdentityPoolId: 'us-
east-1:0a5b211d-d266-4ea9-8208-90f2c74ea7cc' });

    // Function invoked by button click
    function insertSurveyData() {
      // Create the DynamoDB service object
      var ddb = new AWS.DynamoDB({ apiVersion: '2012-08-10' });

      var params = {
        TableName: 'survey',
        Item: {
          'email': { S: document.getElementById("email").value },
          'city': { S: document.getElementById("city").value },
          'feedback': { S: document.getElementById("feedback").value }
        }
      };

      // Call DynamoDB to add the item to the table
```

```
      ddb.putItem(params, function (err, data) {
        if (err) {
          document.getElementById('result').innerHTML = error;
        } else {
          document.getElementById('result').innerHTML = "Thanks for helping with the
survey";
        }
      });
    }
  </script>
</body>

</html>
```

Use Case – 2

```
import time
import boto3
import mysql.connector

queue_url = 'https://sqs.us-east-1.amazonaws.com/304000509264/CustomerQueue'

#Specify the database details
host = 'customerdb.cmeeo0ikklen.us-east-1.rds.amazonaws.com'
user = 'praveen'
password = 'praveen123'
database='customer_db'

#Create a SQS Client
sqs = boto3.client('sqs')

#Connect to the RDS MySQL Instance
mydb = mysql.connector.connect(host=host, user=user, password=password, database=database)
mycursor = mydb.cursor()

# Receive message from SQS queue
response = sqs.receive_message(QueueUrl=queue_url)
message = response['Messages'][0]

# Delete received message from queue
receipt_handle = message['ReceiptHandle']

sqs.delete_message(
    QueueUrl=queue_url,
    ReceiptHandle=receipt_handle
)
```

```
print('Received and deleted message: %s' % message["Body"])

#Get the customer name and address from the message
customerDetails = message["Body"]
customerDetailsList = customerDetails.split(',')
name = customerDetailsList[0]
address = customerDetailsList[1]

#Write the record to the database
val = (name, address)
sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

mycursor.execute(sql, val)
mydb.commit()
print("Record inserted in the DB")
```

Use Case – 7

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
<?php
/* Connect to MySQL and select the database. */
$connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);
if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
mysqli_connect_error();
$database = mysqli_select_db($connection, DB_DATABASE);
/* Ensure that the Employees table exists. */
VerifyTable($connection, DB_DATABASE);
/* If input fields are populated, add a row to the Employees table. */
$employee_name = htmlentities($_POST['Name']);
$employee_address = htmlentities($_POST['Address']);
if (strlen($employee_name) || strlen($employee_address)) {
AddEmployee($connection, $employee_name, $employee_address);
}
?>
<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
<table border="0">
<tr>
<td>Name</td>
<td>Address</td>
```

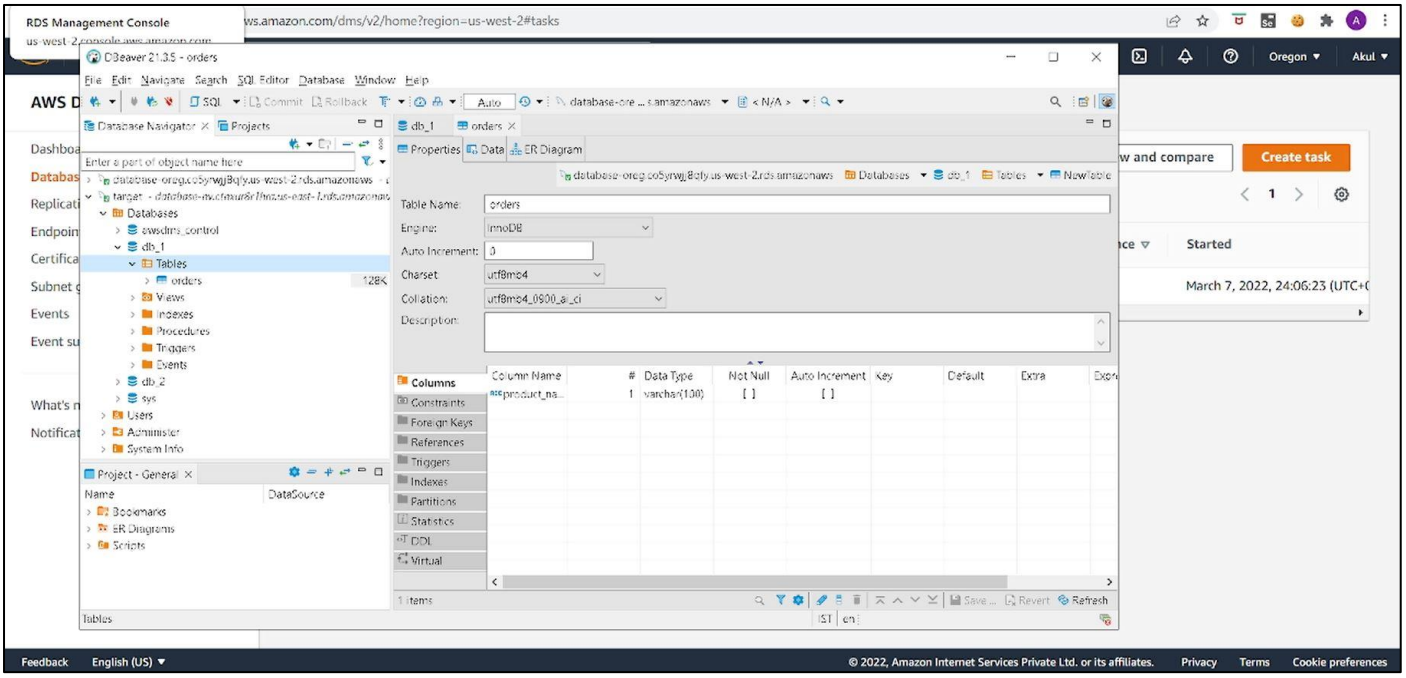
```
</tr>
<tr>
<td>
<input type="text" name="Name" maxlength="45" size="30" />
</td>
<td>
<input type="text" name="Address" maxlength="90" size="60" />
</td>
<td>
<input type="submit" value="Add Data" />
</td>
</tr>
</table>
</form>
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
<tr>
<td>ID</td>
<td>Name</td>
<td>Address</td>
</tr>
<?php
$result = mysqli_query($connection, "SELECT * FROM Employees");
while($query_data = mysqli_fetch_row($result)) {
echo "<tr>";
echo "<td>",$query_data[0], "</td>",
"<td>",$query_data[1], "</td>",
"<td>",$query_data[2], "</td>";
echo "</tr>";
}
?>
</table>
<!-- Clean up. -->
<?php
mysqli_free_result($result);
mysqli_close($connection);
?>
</body>
</html>
<?php
/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
$n = mysqli_real_escape_string($connection, $name);
$a = mysqli_real_escape_string($connection, $address);
$query = "INSERT INTO `Employees` (`Name`, `Address`) VALUES ('$n', '$a');";
if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</p>");
}
```

```
/* Check whether the table exists and, if not, create it. */
function VerifyTable($connection, $dbName) {
if(!TableExists("Employees", $connection, $dbName))
{
$query = "CREATE TABLE `Employees` (
`ID` int(11) NOT NULL AUTO_INCREMENT,
`Name` varchar(45) DEFAULT NULL,
`Address` varchar(90) DEFAULT NULL,
PRIMARY KEY (`ID`),
UNIQUE KEY `ID_UNIQUE` (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1";
if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</p>");
}}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
$t = mysqli_real_escape_string($connection, $tableName);
$d = mysqli_real_escape_string($connection, $dbName);
$checktable = mysqli_query($connection,
"SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
AND TABLE_SCHEMA = '$d'");
if(mysqli_num_rows($checktable) > 0) return true;
return false;
}
?>
```

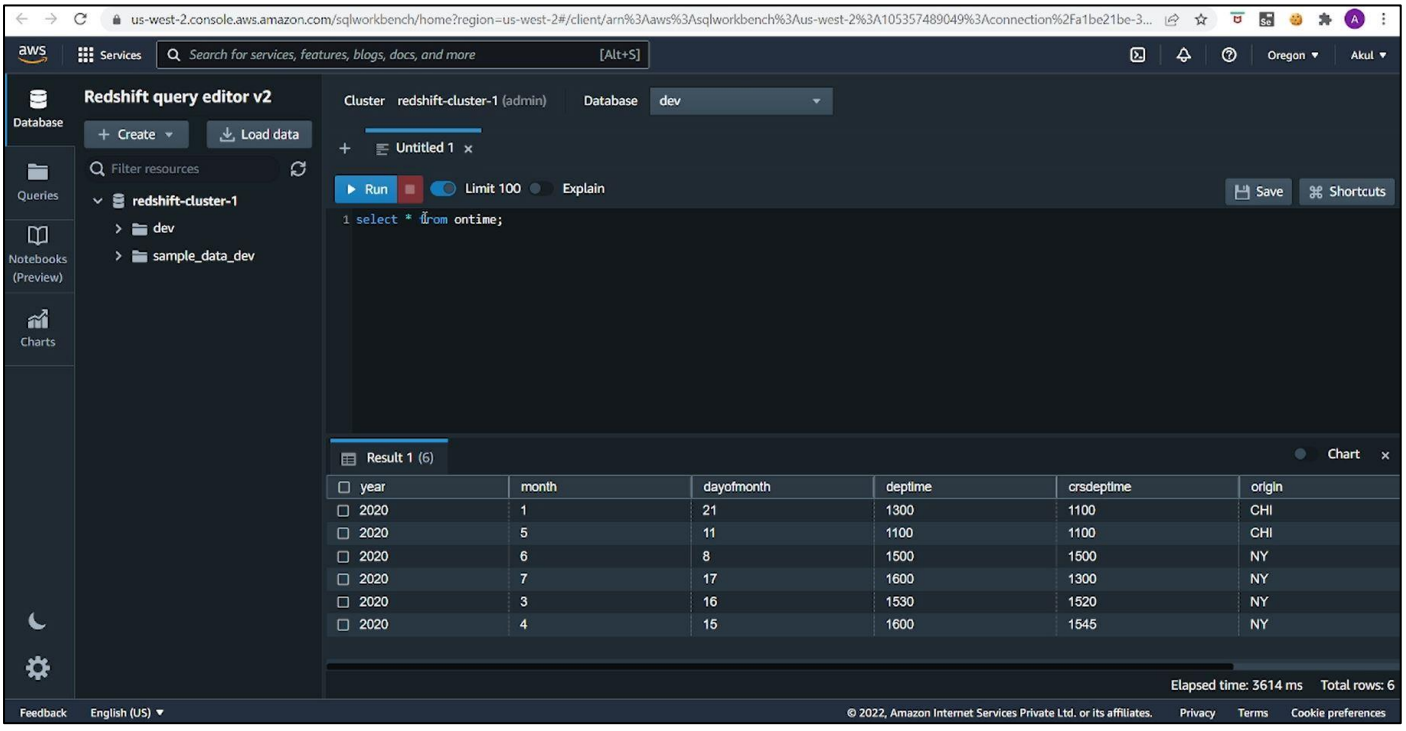
OUTPUTS

SECTION - A

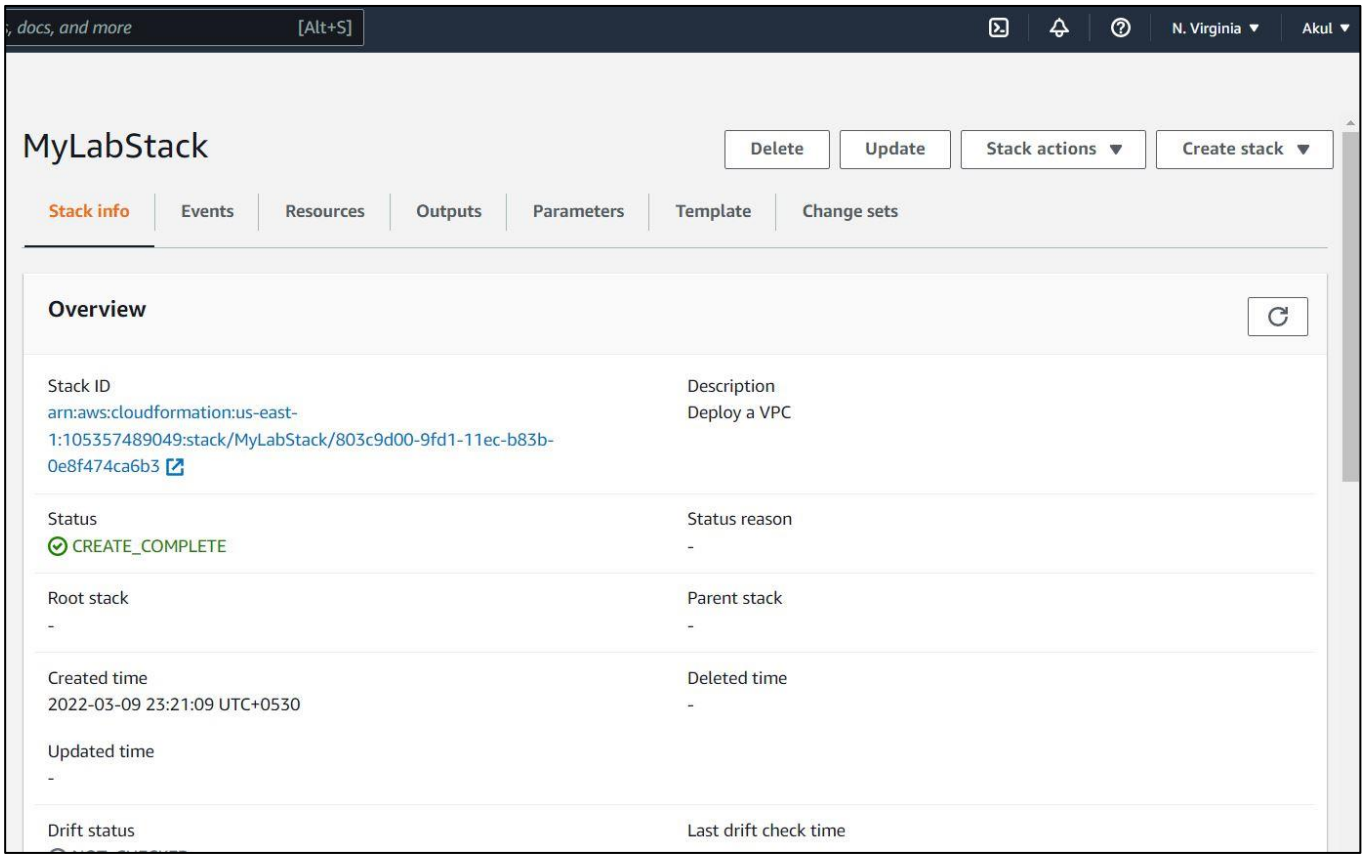
Use Case - 1:

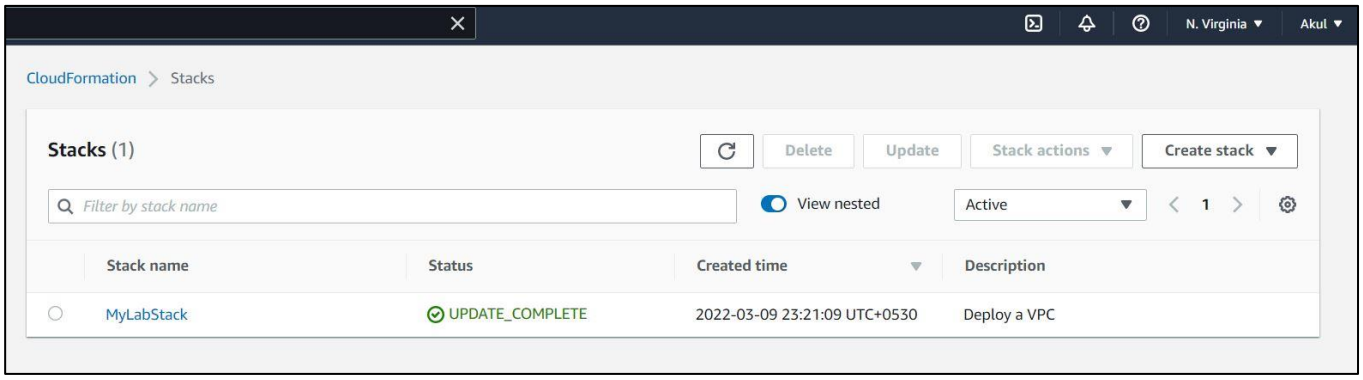


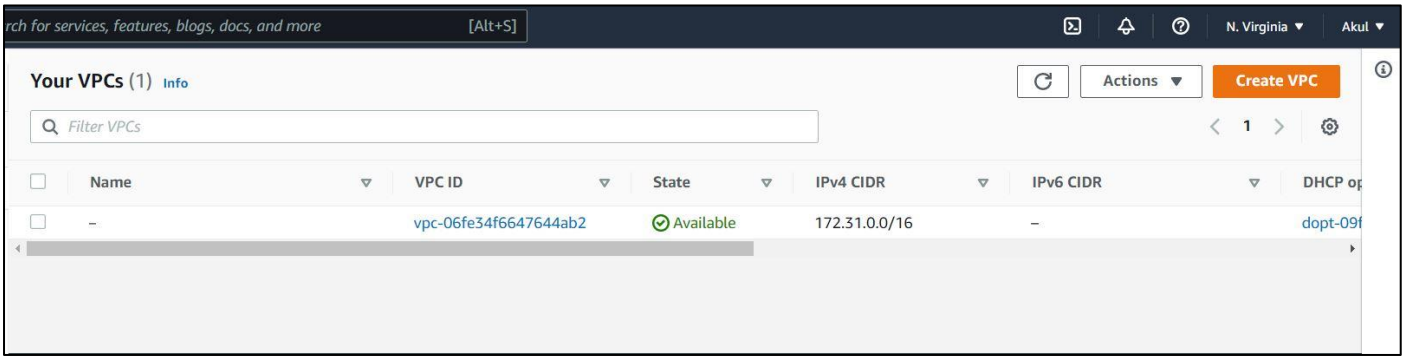
Use Case - 2:


Redshift query editor v2
Cluster redshift-cluster-1 (admin)
Database dev
Untitled 1
Run
Limit 100
Explain
Save
Shortcuts
1 select * from ontime;
Result 1 (6)
Chart
year month dayofmonth deptime crsdeptime origin
2020 1 21 1300 1100 CHI
2020 5 11 1100 1100 CHI
2020 6 8 1500 1500 NY
2020 7 17 1600 1300 NY
2020 3 16 1530 1520 NY
2020 4 15 1600 1545 NY
Elapsed time: 3614 ms
Total rows: 6

Use Case - 3:

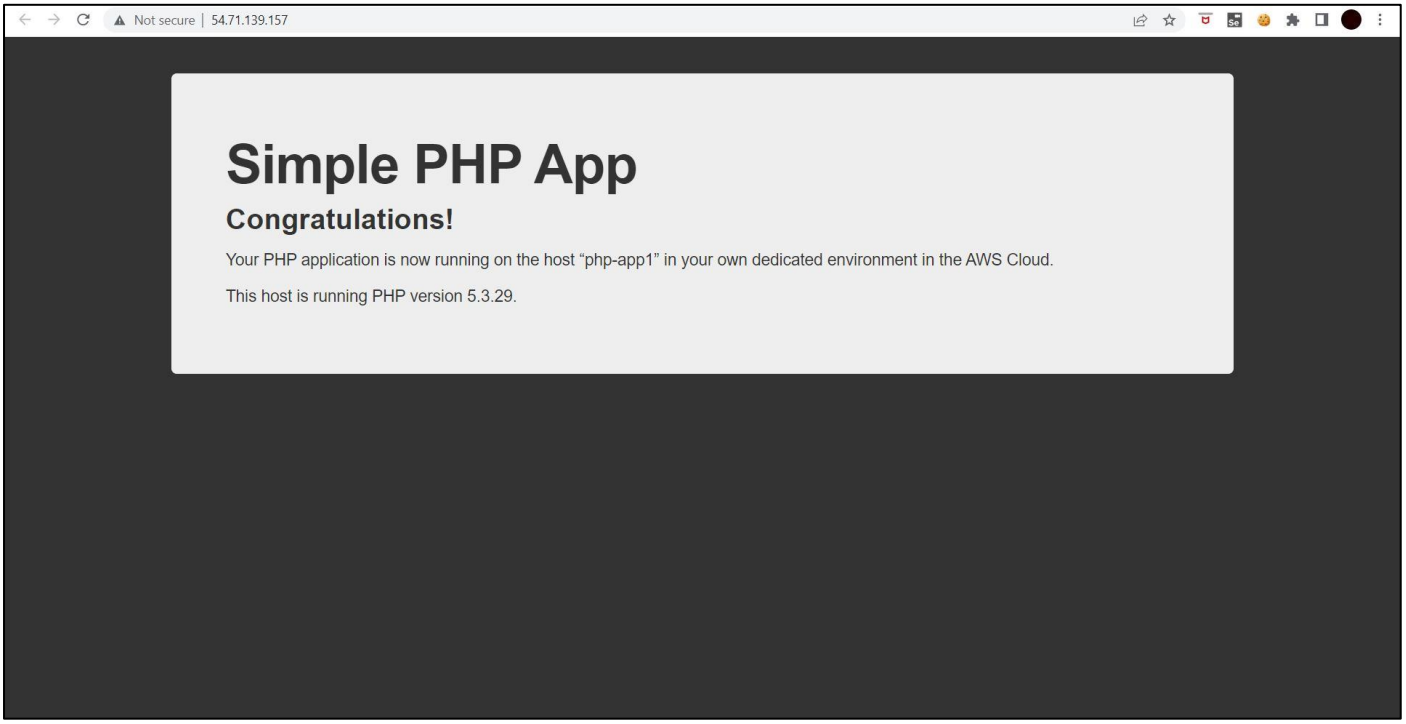

docs, and more [Alt+S]
N. Virginia
Akul
MyLabStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Overview
Stack ID
arn:aws:cloudformation:us-east-1:105357489049:stack/MyLabStack/803c9d00-9fd1-11ec-b83b-0e8f474ca6b3
Description
Deploy a VPC
Status
CREATE_COMPLETE
Status reason
-
Root stack
-
Parent stack
-
Created time
2022-03-09 23:21:09 UTC+0530
Deleted time
-
Updated time
-
Drift status
Last drift check time


N. Virginia
Akul
CloudFormation
Stacks
Stacks (1)
Delete
Update
Stack actions
Create stack
Filter by stack name
View nested
Active
1
Stack name
Status
Created time
Description
MyLabStack
UPDATE_COMPLETE
2022-03-09 23:21:09 UTC+0530
Deploy a VPC

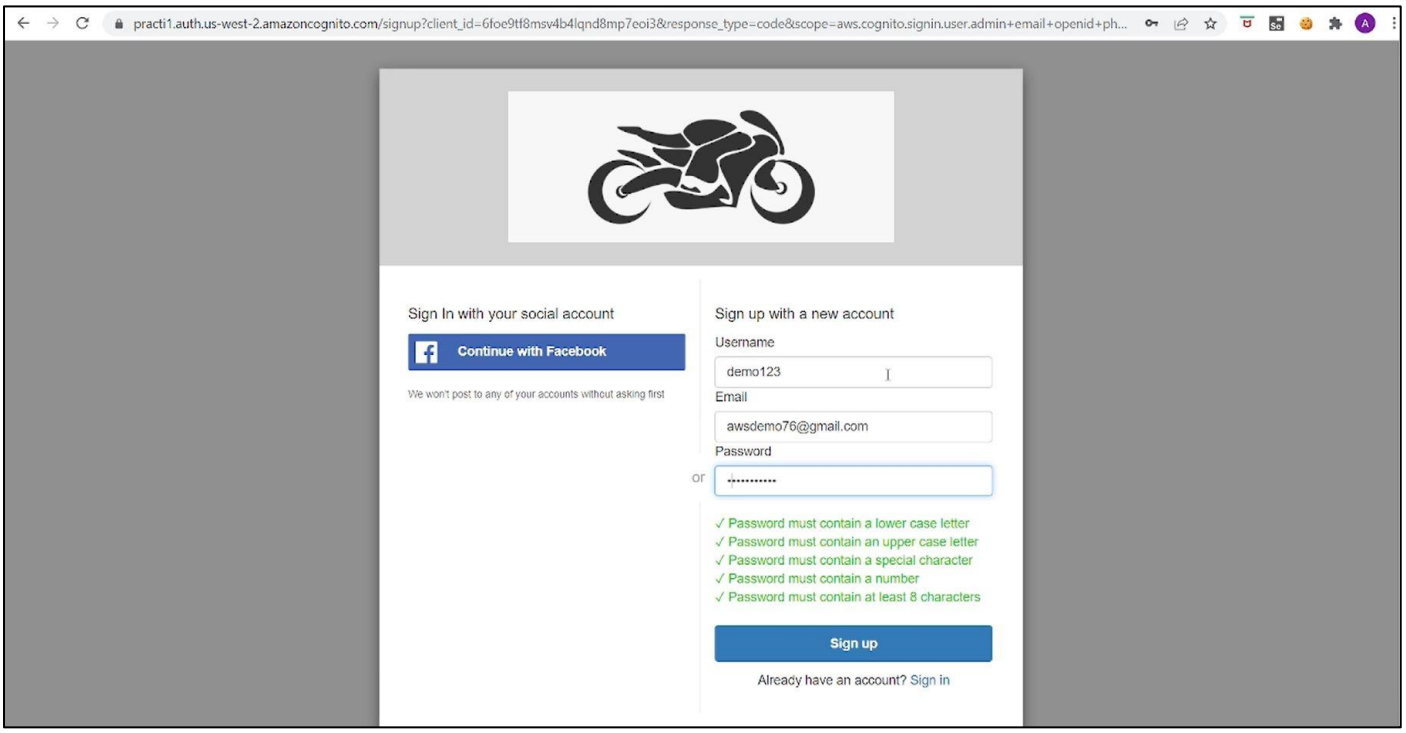


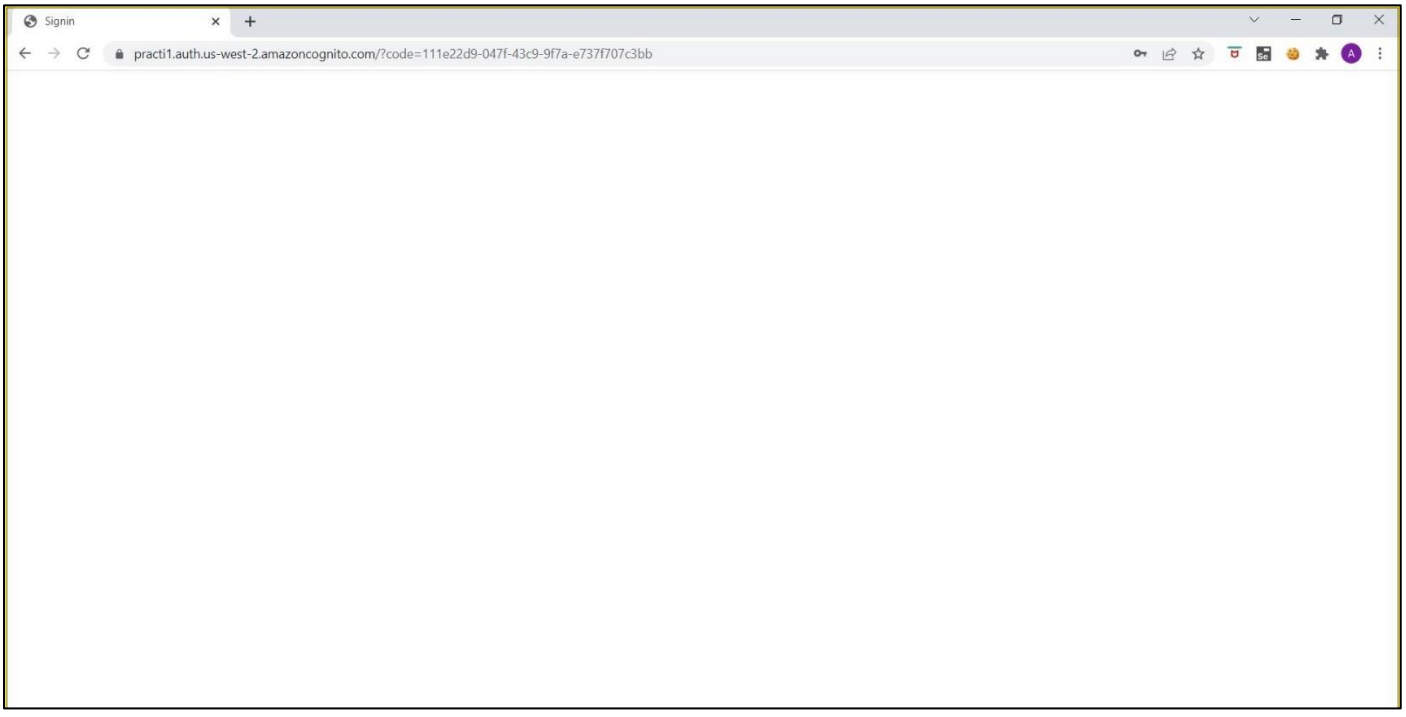
Use Case-4:

Use Case - 5:

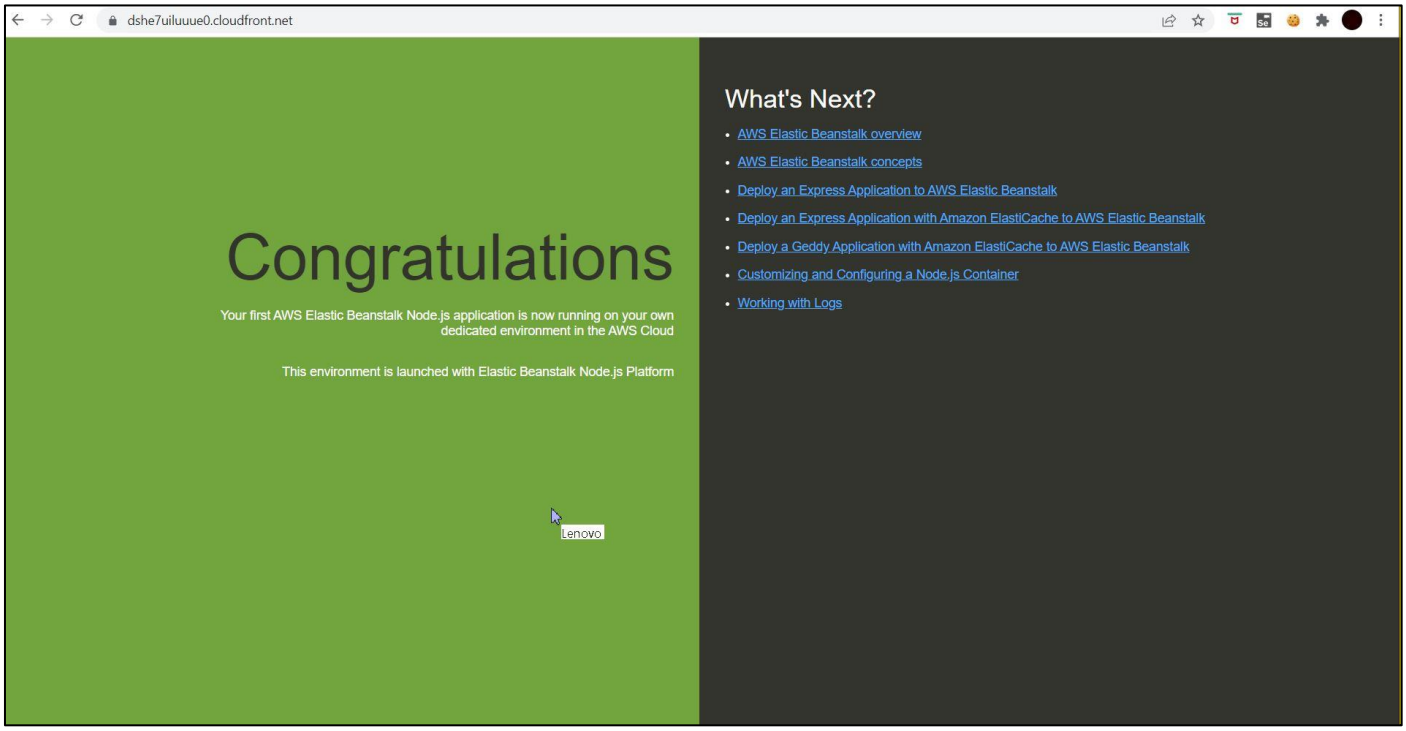


Use Case - 6:


practi1.auth.us-west-2.amazoncognito.com/signup?client_id=6foe9tf8msv4b4lqnd8mp7eoi3&response_type=code&scope=aws.cognito.signin.user.admin+email+openid+ph...
Sign In with your social account
Continue with Facebook
We won't post to any of your accounts without asking first
or
Sign up with a new account
Username
demo123
Email
awsdemo76@gmail.com
Password
✓ Password must contain a lower case letter
✓ Password must contain an upper case letter
✓ Password must contain a special character
✓ Password must contain a number
✓ Password must contain at least 8 characters
Sign up
Already have an account? Sign in

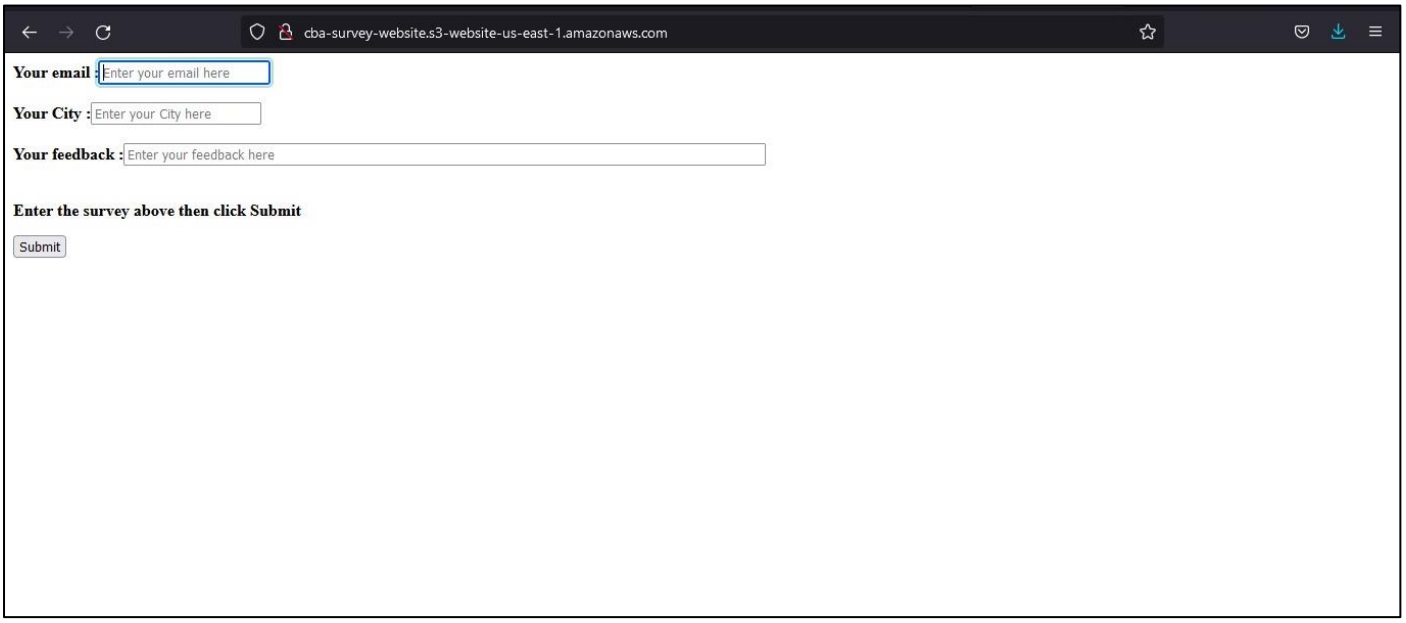

Signin
practi1.auth.us-west-2.amazoncognito.com/?code=111e22d9-047f-43c9-9f7a-e737f707c3bb

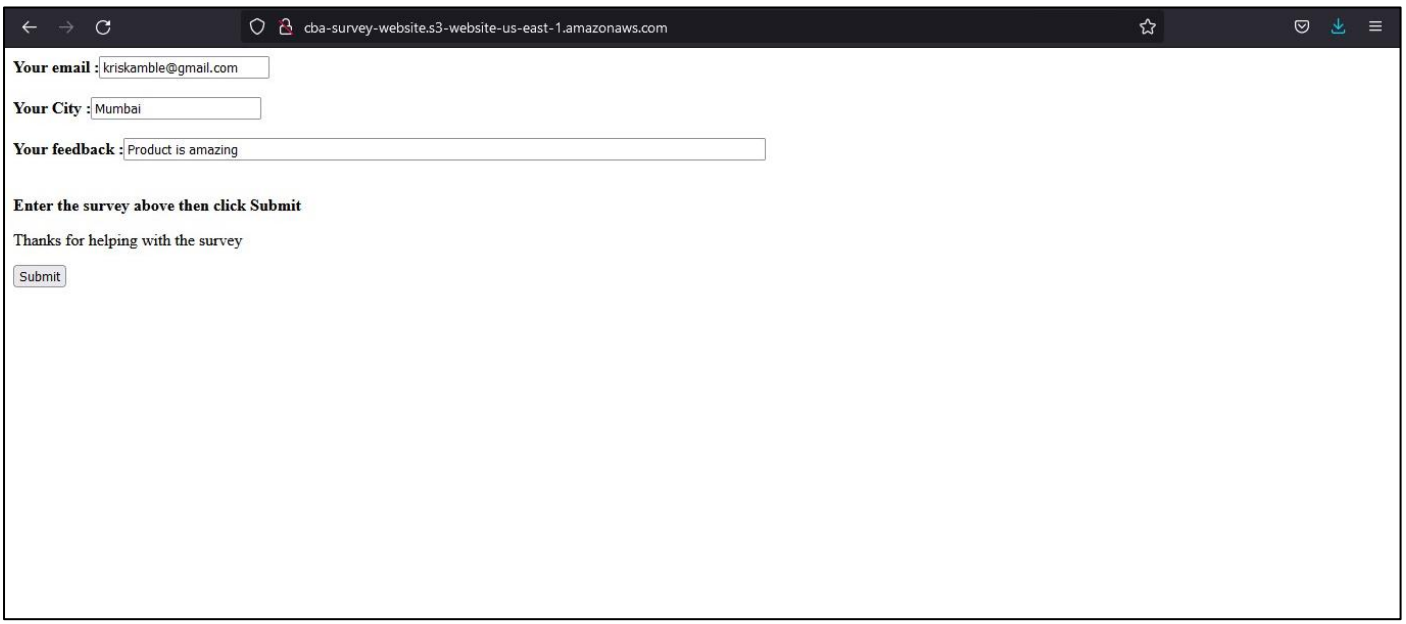
Use Case - 7:

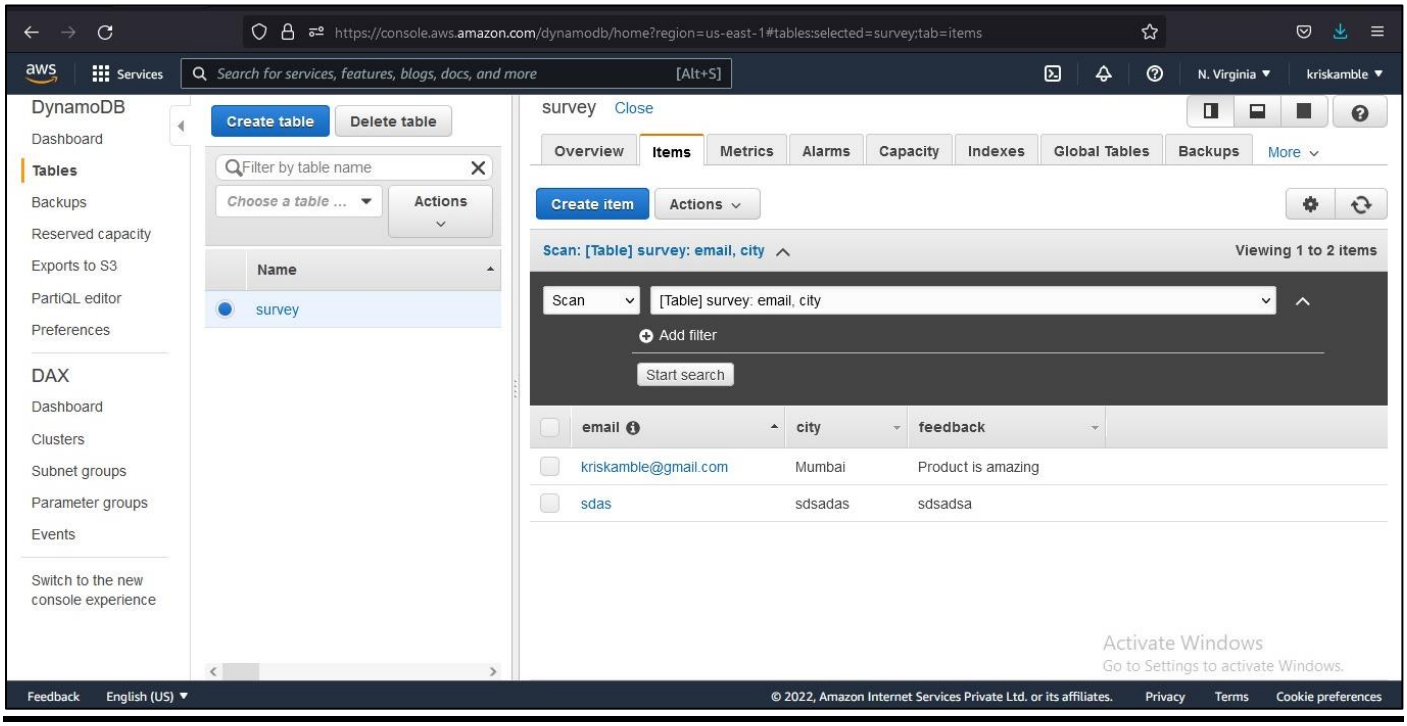

dshe7uiluuue0.cloudfront.net
Congratulations
Your first AWS Elastic Beanstalk Node.js application is now running on your own dedicated environment in the AWS Cloud
This environment is launched with Elastic Beanstalk Node.js Platform
Lenovo
What's Next?
AWS Elastic Beanstalk overview
AWS Elastic Beanstalk concepts
Deploy an Express Application to AWS Elastic Beanstalk
Deploy an Express Application with Amazon ElastiCache to AWS Elastic Beanstalk
Deploy a Geddy Application with Amazon ElastiCache to AWS Elastic Beanstalk
Customizing and Configuring a Node.js Container
Working with Logs

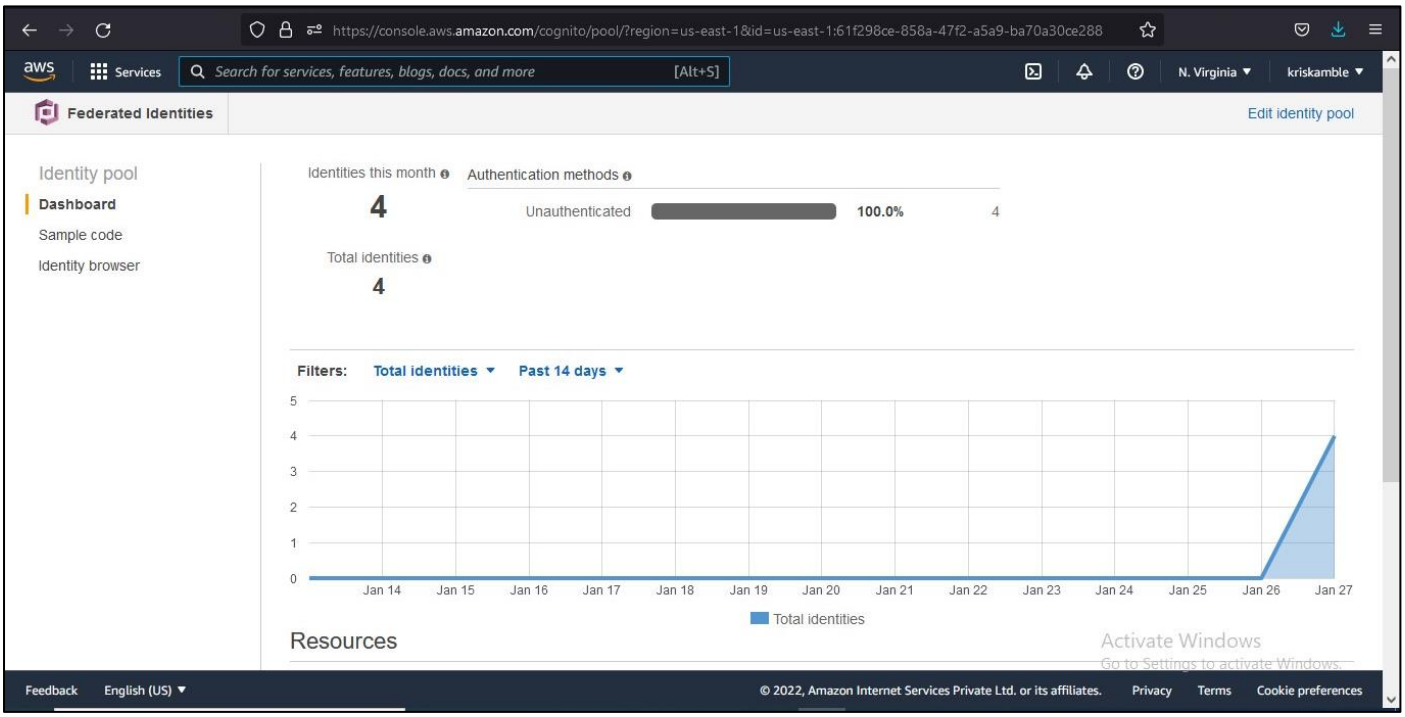
SECTION - B

Use Case - 1:

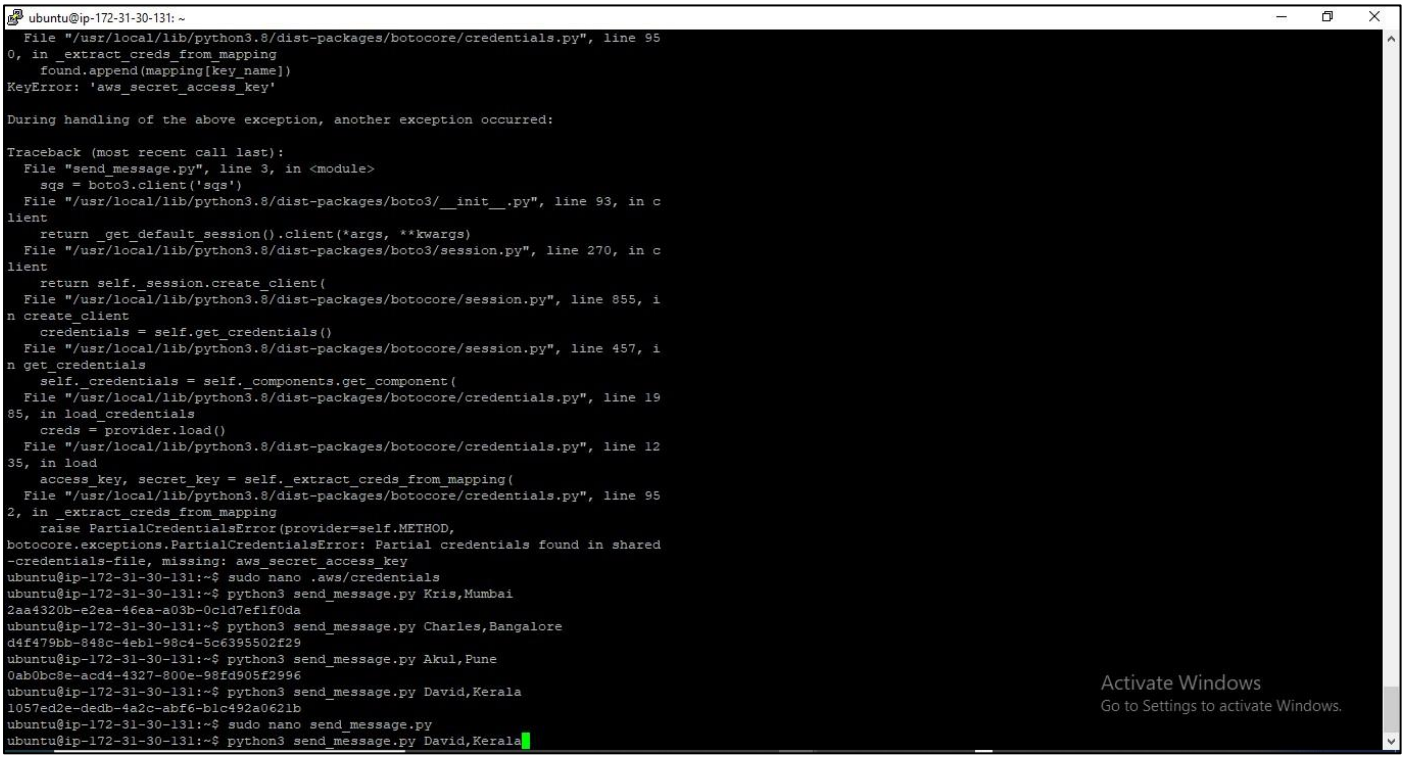


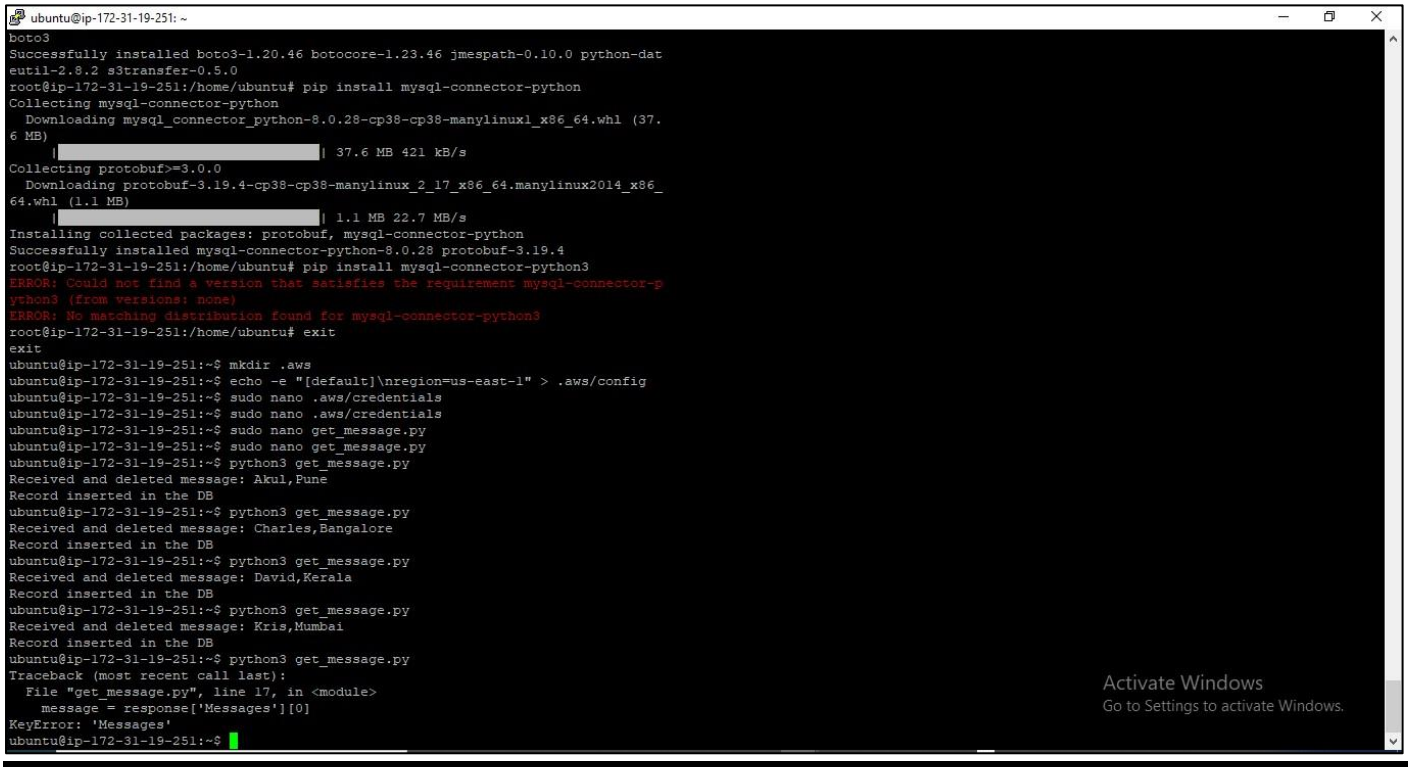


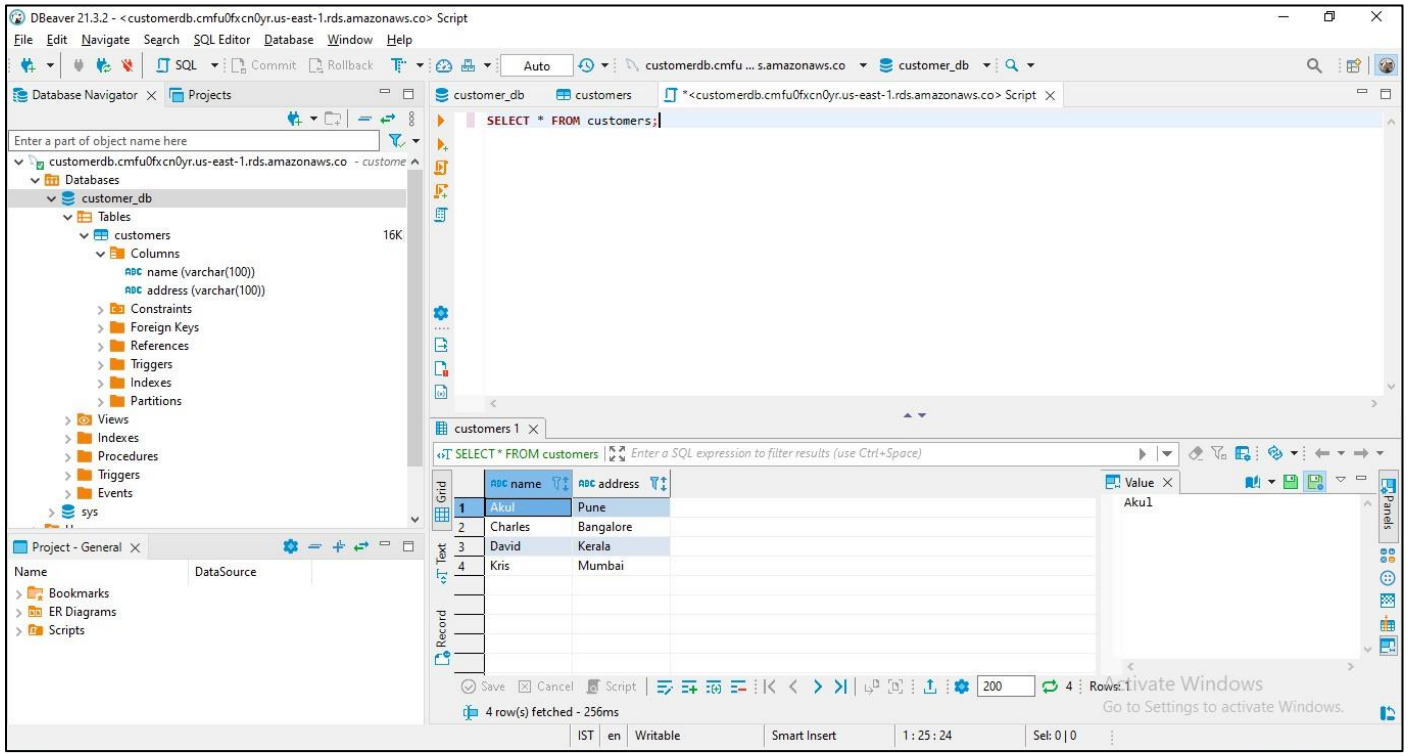

https://console.aws.amazon.com/dynamodb/home?region=us-east-1#tables:selected=survey;tab=items
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
kriskamble
DynamoDB
Dashboard
Tables
Backups
Reserved capacity
Exports to S3
PartiQL editor
Preferences
DAX
Dashboard
Clusters
Subnet groups
Parameter groups
Events
Switch to the new console experience
Create table
Delete table
Filter by table name
Choose a table ...
Actions
Name
survey
survey Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Global Tables
Backups
More
Create item
Actions
Scan: [Table] survey: email, city
Viewing 1 to 2 items
Scan
[Table] survey: email, city
Add filter
Start search
email
city
feedback
kriskamble@gmail.com
Mumbai
Product is amazing
sdas
sdsadas
sdsadsa
Activate Windows
Go to Settings to activate Windows.
Feedback
English (US)
© 2022, Amazon Internet Services Private Ltd. or its affiliates.
Privacy
Terms
Cookie preferences


https://console.aws.amazon.com/cognito/pool/?region=us-east-1&id=us-east-1:61f298ce-858a-47f2-a5a9-ba70a30ce288
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
kriskamble
Federated Identities
Edit identity pool
Identity pool
Dashboard
Sample code
Identity browser
Identities this month
4
Total identities
4
Authentication methods
Unauthenticated
100.0%
4
Filters:
Total identities
Past 14 days
5
4
3
2
1
0
Jan 14
Jan 15
Jan 16
Jan 17
Jan 18
Jan 19
Jan 20
Jan 21
Jan 22
Jan 23
Jan 24
Jan 25
Jan 26
Jan 27
Total identities
Resources
Activate Windows
Go to Settings to activate Windows.
Feedback
English (US)
© 2022, Amazon Internet Services Private Ltd. or its affiliates.
Privacy
Terms
Cookie preferences

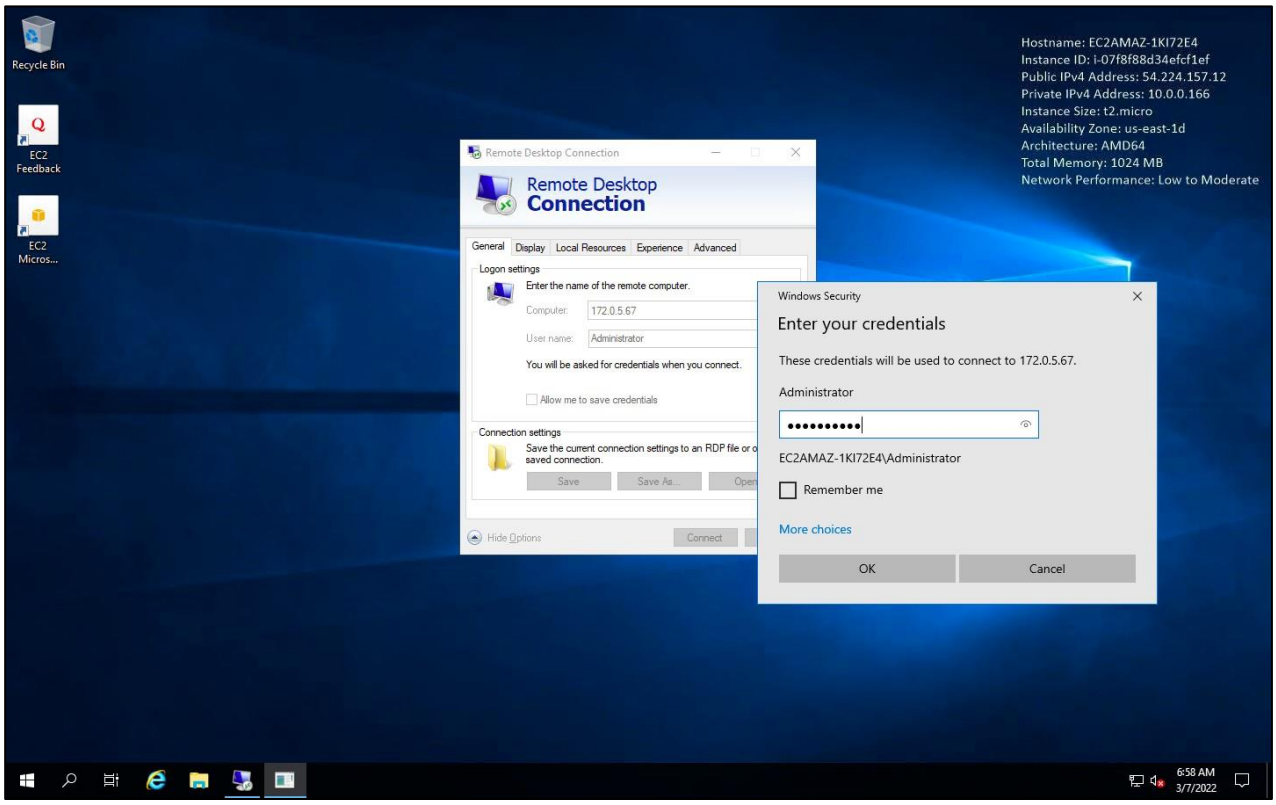
Use Case - 2:

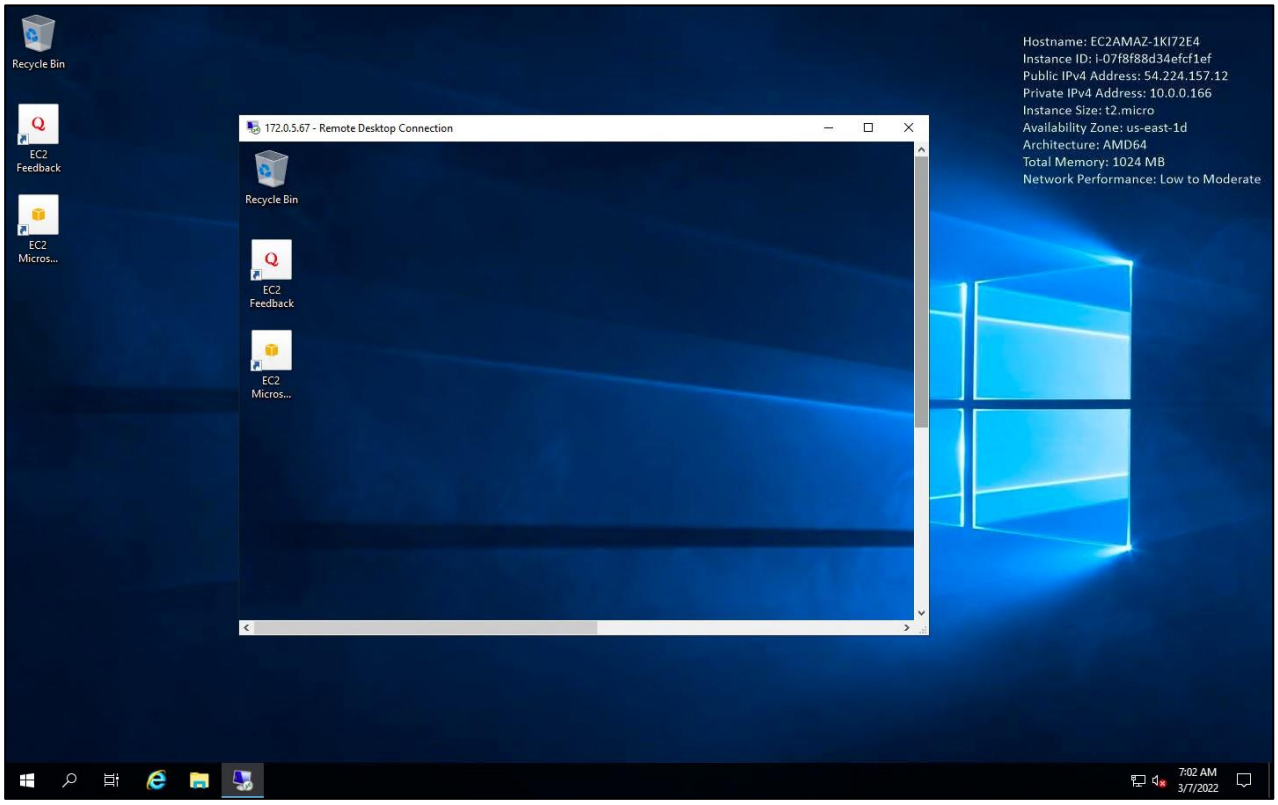

ubuntu@ip-172-31-30-131: ~
File "/usr/local/lib/python3.8/dist-packages/botocore/credentials.py", line 950, in _extract_creds_from_mapping
found.append(mapping[key_name])
KeyError: 'aws_secret_access_key'
During handling of the above exception, another exception occurred:
Traceback (most recent call last):
File "send_message.py", line 3, in <module>
sqs = boto3.client('sqs')
File "/usr/local/lib/python3.8/dist-packages/boto3/__init__.py", line 93, in client
return _get_default_session().client(*args, **kwargs)
File "/usr/local/lib/python3.8/dist-packages/boto3/session.py", line 270, in client
return self._session.create_client(
File "/usr/local/lib/python3.8/dist-packages/botocore/session.py", line 855, in create_client
credentials = self.get_credentials()
File "/usr/local/lib/python3.8/dist-packages/botocore/session.py", line 457, in get_credentials
self._credentials = self._components.get_component(
File "/usr/local/lib/python3.8/dist-packages/botocore/credentials.py", line 1985, in load_credentials
creds = provider.load()
File "/usr/local/lib/python3.8/dist-packages/botocore/credentials.py", line 1235, in load
access_key, secret_key = self._extract_creds_from_mapping(
File "/usr/local/lib/python3.8/dist-packages/botocore/credentials.py", line 952, in _extract_creds_from_mapping
raise PartialCredentialsError(provider=self.METHOD,
botocore.exceptions.PartialCredentialsError: Partial credentials found in shared-credentials-file, missing: aws_secret_access_key
ubuntu@ip-172-31-30-131:~$ sudo nano .aws/credentials
ubuntu@ip-172-31-30-131:~$ python3 send_message.py Kris,Mumbai
2aa4320b-e2ea-46ea-a03b-0c1d7ef1f0da
ubuntu@ip-172-31-30-131:~$ python3 send_message.py Charles,Bangalore
d4f479bb-848c-4eb1-98c4-5c6395502f29
ubuntu@ip-172-31-30-131:~$ python3 send_message.py Akul,Pune
0ab0bc8e-acd4-4327-800e-98fd905f2996
ubuntu@ip-172-31-30-131:~$ python3 send_message.py David,Kerala
1057ed2e-dedb-4a2c-abf6-b1c492a0621b
ubuntu@ip-172-31-30-131:~$ sudo nano send_message.py
ubuntu@ip-172-31-30-131:~$ python3 send_message.py David,Kerala
Activate Windows
Go to Settings to activate Windows.


ubuntu@ip-172-31-19-251: ~
boto3
Successfully installed boto3-1.20.46 botocore-1.23.46 jmespath-0.10.0 python-dateutil-2.8.2 s3transfer-0.5.0
root@ip-172-31-19-251:/home/ubuntu# pip install mysql-connector-python
Collecting mysql-connector-python
Downloading mysql_connector_python-8.0.28-cp38-cp38-manylinux1_x86_64.whl (37.6 MB)
| 37.6 MB 421 kB/s
Collecting protobuf>=3.0.0
Downloading protobuf-3.19.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.1 MB)
| 1.1 MB 22.7 MB/s
Installing collected packages: protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.28 protobuf-3.19.4
root@ip-172-31-19-251:/home/ubuntu# pip install mysql-connector-python3
ERROR: Could not find a version that satisfies the requirement mysql-connector-python3 (from versions: none)
ERROR: No matching distribution found for mysql-connector-python3
root@ip-172-31-19-251:/home/ubuntu# exit
exit
ubuntu@ip-172-31-19-251:~$ mkdir .aws
ubuntu@ip-172-31-19-251:~$ echo -e "[default]\nregion=us-east-1" > .aws/config
ubuntu@ip-172-31-19-251:~$ sudo nano .aws/credentials
ubuntu@ip-172-31-19-251:~$ sudo nano .aws/credentials
ubuntu@ip-172-31-19-251:~$ sudo nano get_message.py
ubuntu@ip-172-31-19-251:~$ sudo nano get_message.py
ubuntu@ip-172-31-19-251:~$ python3 get_message.py
Received and deleted message: Akul,Pune
Record inserted in the DB
ubuntu@ip-172-31-19-251:~$ python3 get_message.py
Received and deleted message: Charles,Bangalore
Record inserted in the DB
ubuntu@ip-172-31-19-251:~$ python3 get_message.py
Received and deleted message: David,Kerala
Record inserted in the DB
ubuntu@ip-172-31-19-251:~$ python3 get_message.py
Received and deleted message: Kris,Mumbai
Record inserted in the DB
ubuntu@ip-172-31-19-251:~$ python3 get_message.py
Traceback (most recent call last):
File "get_message.py", line 17, in <module>
message = response['Messages'][0]
KeyError: 'Messages'
ubuntu@ip-172-31-19-251:~$
Activate Windows
Go to Settings to activate Windows.

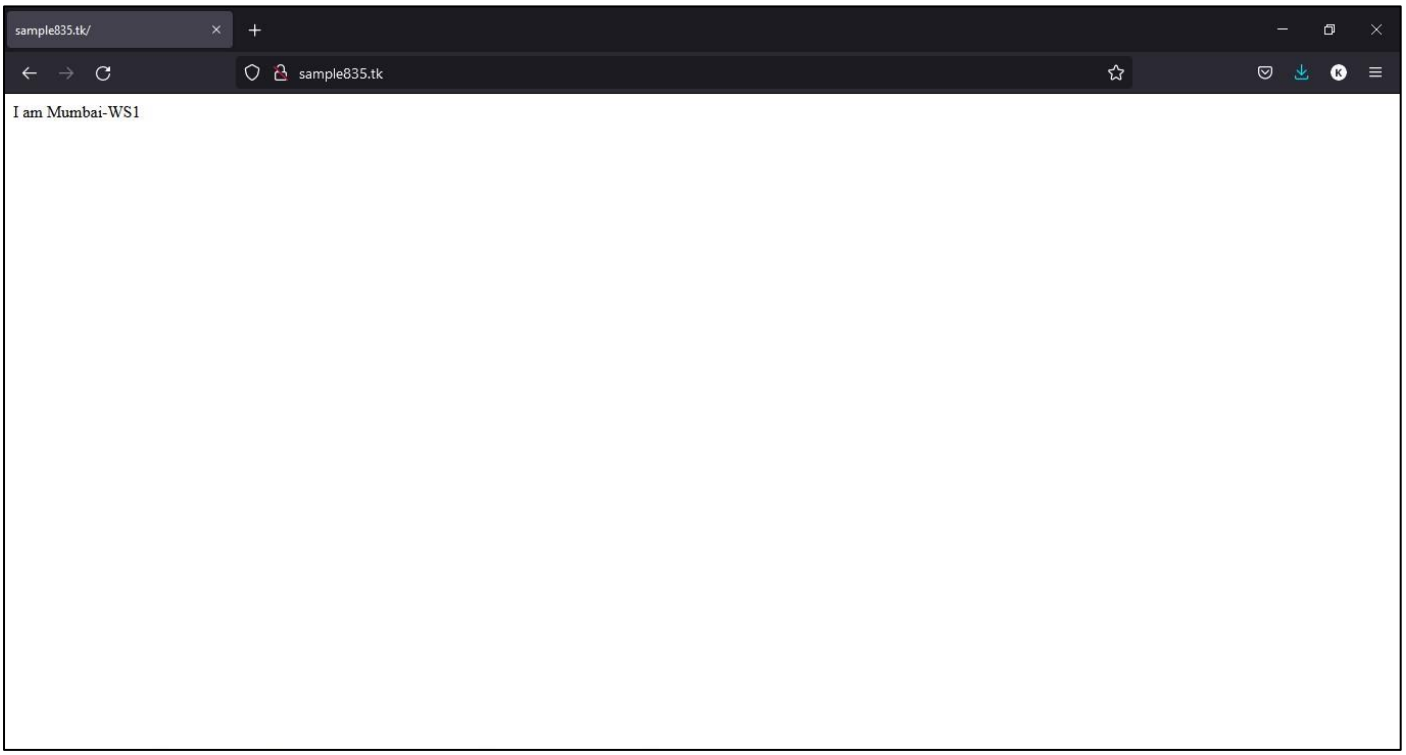

DBeaver 21.3.2 - <customerdb.cmfu0fxcn0yr.us-east-1.rds.amazonaws.co> Script
File Edit Navigate Search SQL Editor Database Window Help
Database Navigator
Projects
customerdb.cmfu0fxcn0yr.us-east-1.rds.amazonaws.co
Databases
customer_db
Tables
customers
Columns
name (varchar(100))
address (varchar(100))
Constraints
Foreign Keys
References
Triggers
Indexes
Partitions
Views
Indexes
Procedures
Triggers
Events
sys
Project - General
Bookmarks
ER Diagrams
Scripts
SELECT * FROM customers;
customers 1
SELECT * FROM customers
name
address
1 Akul Pune
2 Charles Bangalore
3 David Kerala
4 Kris Mumbai
Value
Akul
4 row(s) fetched - 256ms
Activate Windows
Go to Settings to activate Windows.
IST en Writable Smart Insert 1 : 25 : 24 Sel: 0 | 0

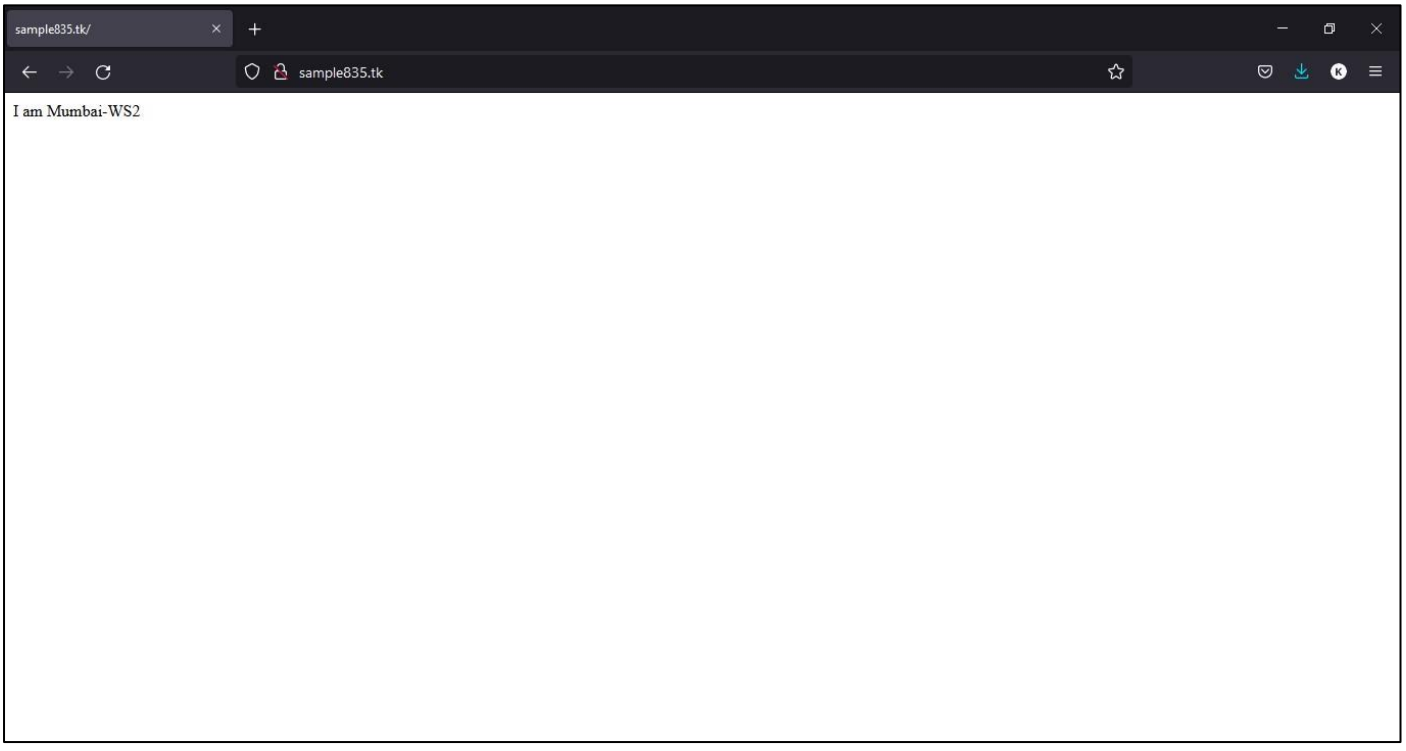
Use Case - 3:

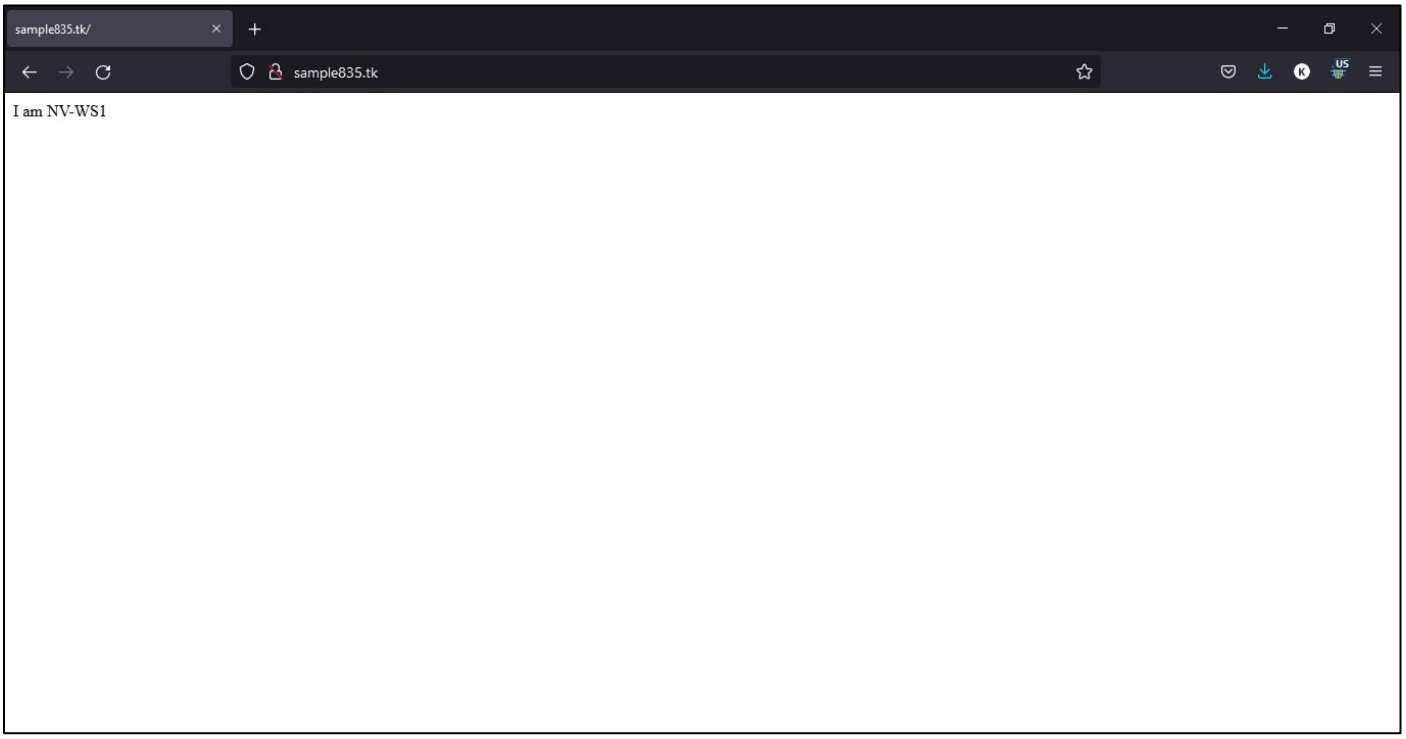

Recycle Bin
EC2 Feedback
EC2 Micros...
Hostname: EC2AMAZ-1KI72E4
Instance ID: i-07f8f88d34efcf1ef
Public IPv4 Address: 54.224.157.12
Private IPv4 Address: 10.0.0.166
Instance Size: t2.micro
Availability Zone: us-east-1d
Architecture: AMD64
Total Memory: 1024 MB
Network Performance: Low to Moderate
Remote Desktop Connection
Remote Desktop Connection
General
Display
Local Resources
Experience
Advanced
Logon settings
Enter the name of the remote computer.
Computer: 172.0.5.67
User name: Administrator
You will be asked for credentials when you connect.
Allow me to save credentials
Connection settings
Save the current connection settings to an RDP file or o
saved connection.
Save
Save As...
Open
Hide Options
Connect
Windows Security
Enter your credentials
These credentials will be used to connect to 172.0.5.67.
Administrator
EC2AMAZ-1KI72E4\Administrator
Remember me
More choices
OK
Cancel
6:58 AM
3/7/2022

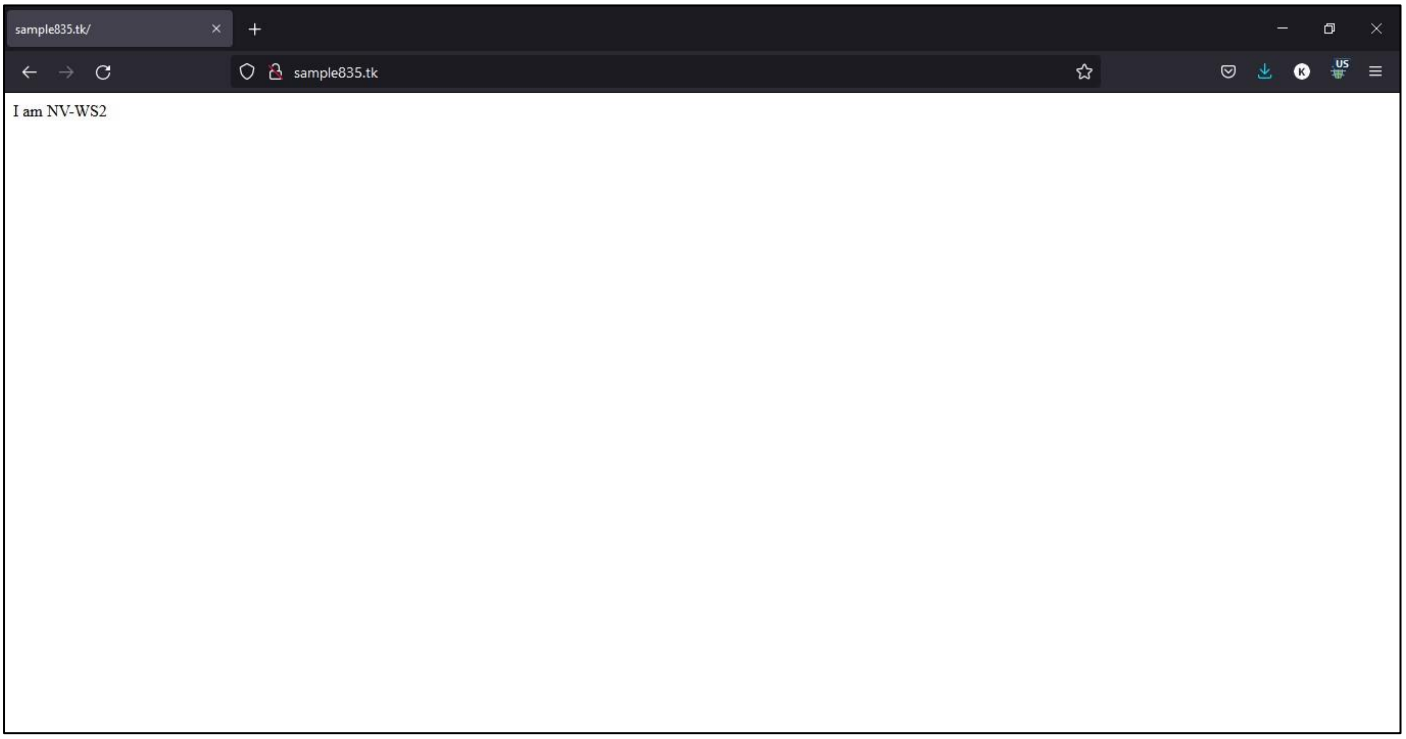

Recycle Bin
EC2 Feedback
EC2 Micros...
Hostname: EC2AMAZ-1KI72E4
Instance ID: i-07f8f88d34efcf1ef
Public IPv4 Address: 54.224.157.12
Private IPv4 Address: 10.0.0.166
Instance Size: t2.micro
Availability Zone: us-east-1d
Architecture: AMD64
Total Memory: 1024 MB
Network Performance: Low to Moderate
172.0.5.67 - Remote Desktop Connection
Recycle Bin
EC2 Feedback
EC2 Micros...
7:02 AM
3/7/2022

Use Case - 4:

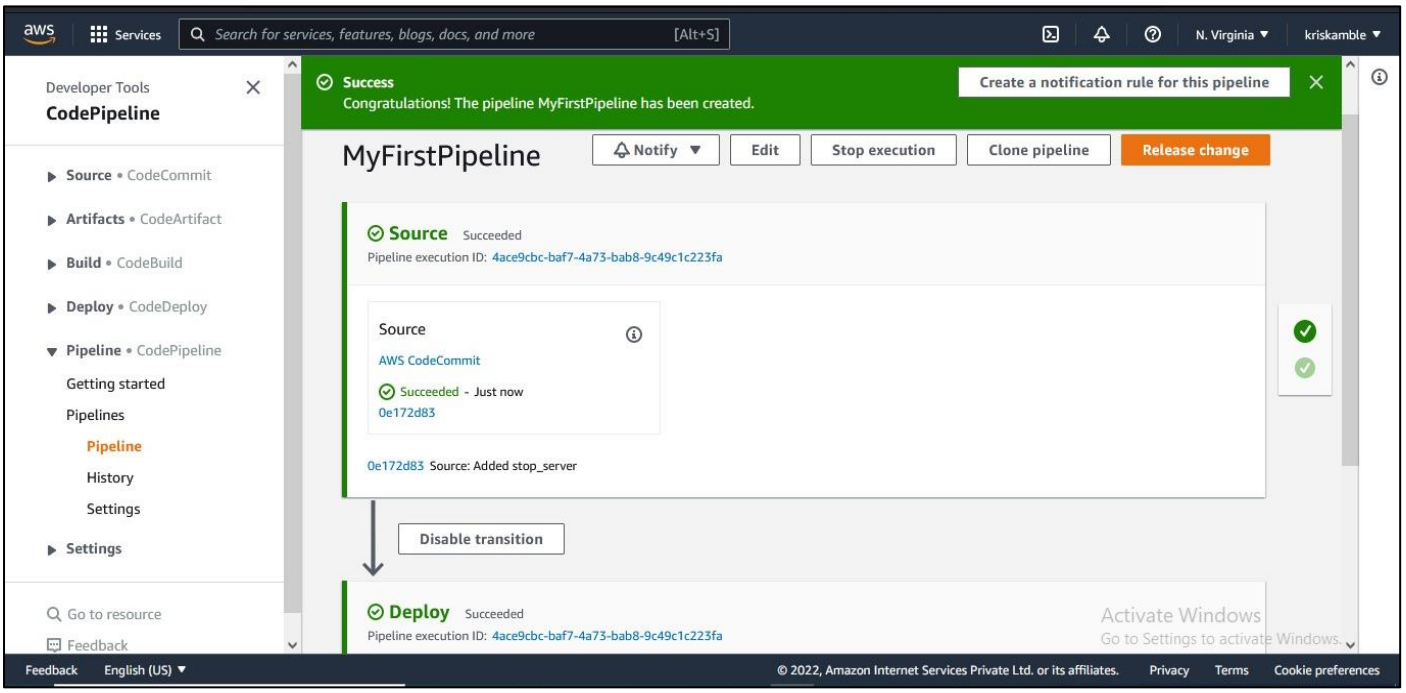

sample835.tk/
sample835.tk
I am Mumbai-WS1

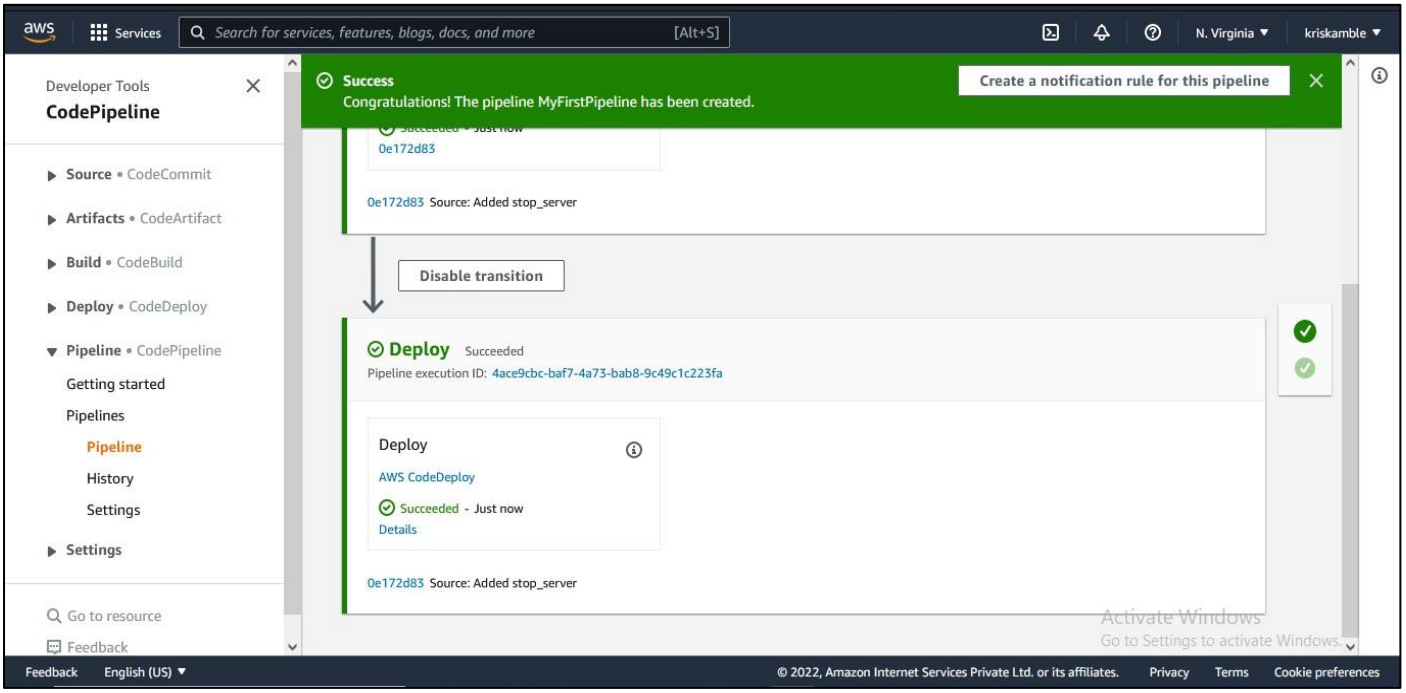

sample835.tk/
sample835.tk
I am Mumbai-WS2

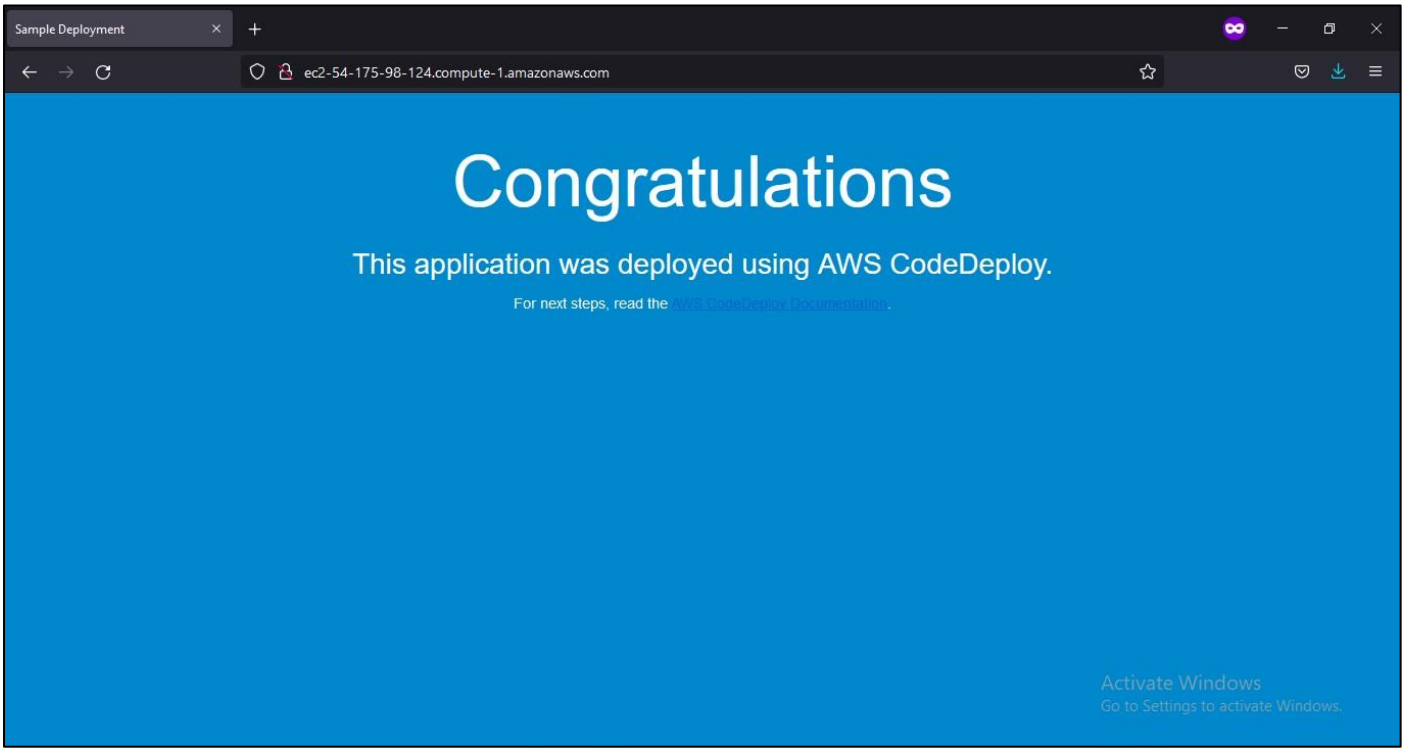

sample835.tk/
sample835.tk
I am NV-WS1


sample835.tk/
sample835.tk
I am NV-WS2

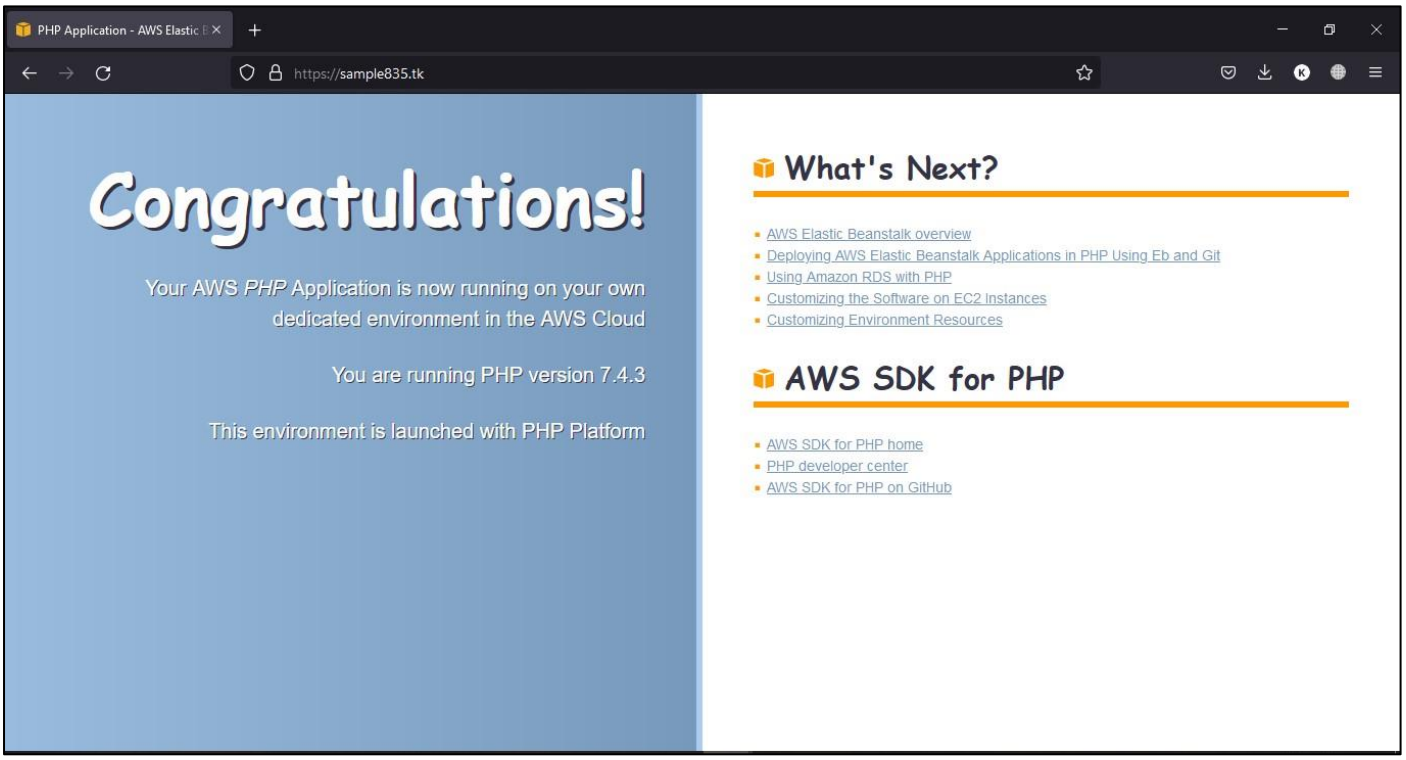
Use Case - 5:

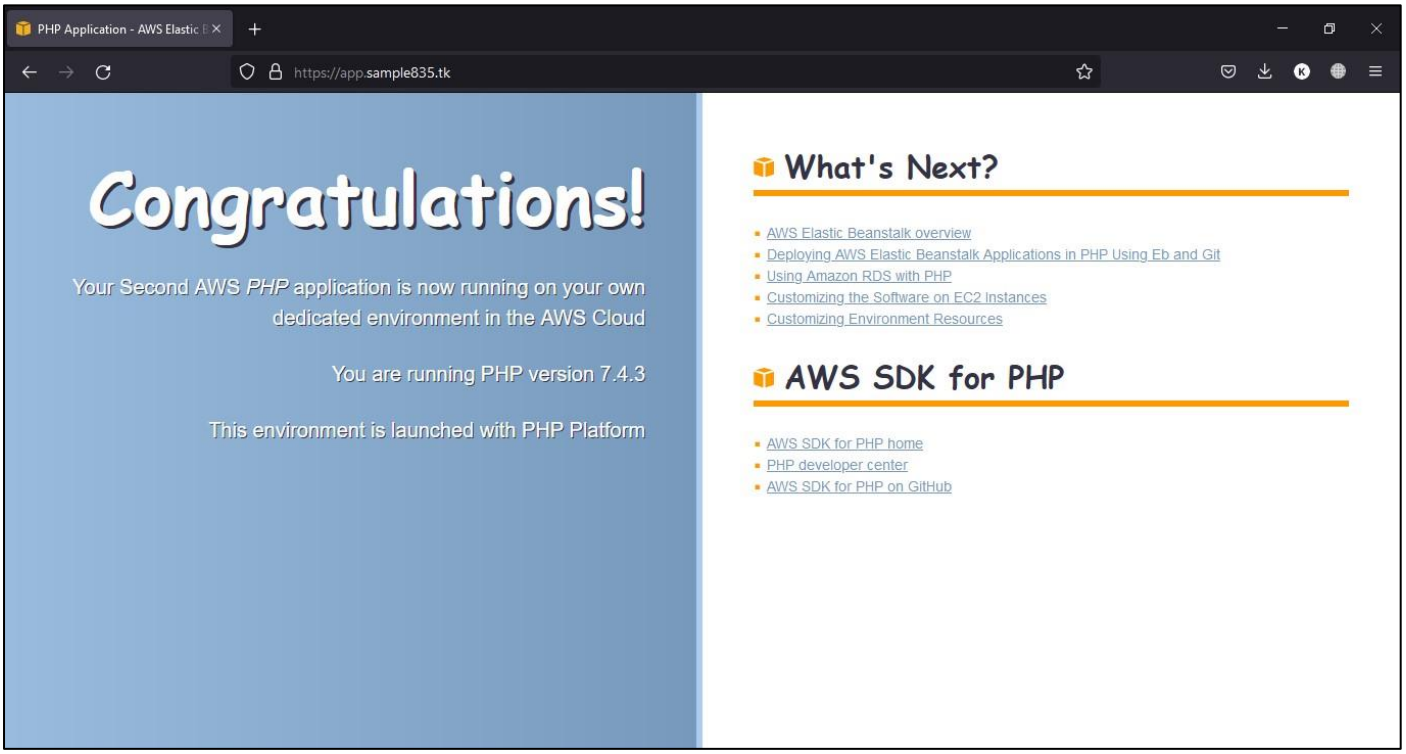

Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
kriskamble
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Success
Congratulations! The pipeline MyFirstPipeline has been created.
Create a notification rule for this pipeline
MyFirstPipeline
Notify
Edit
Stop execution
Clone pipeline
Release change
Source Succeeded
Pipeline execution ID: 4ace9cbc-baf7-4a73-bab8-9c49c1c223fa
Source
AWS CodeCommit
Succeeded - Just now
0e172d83
0e172d83 Source: Added stop_server
Disable transition
Deploy Succeeded
Pipeline execution ID: 4ace9cbc-baf7-4a73-bab8-9c49c1c223fa
Activate Windows
Go to Settings to activate Windows.
Feedback
English (US)
© 2022, Amazon Internet Services Private Ltd. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
kriskamble
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Success
Congratulations! The pipeline MyFirstPipeline has been created.
Create a notification rule for this pipeline
0e172d83
0e172d83 Source: Added stop_server
Disable transition
Deploy Succeeded
Pipeline execution ID: 4ace9cbc-baf7-4a73-bab8-9c49c1c223fa
Deploy
AWS CodeDeploy
Succeeded - Just now
Details
0e172d83 Source: Added stop_server
Activate Windows
Go to Settings to activate Windows.
Feedback
English (US)
© 2022, Amazon Internet Services Private Ltd. or its affiliates.
Privacy
Terms
Cookie preferences

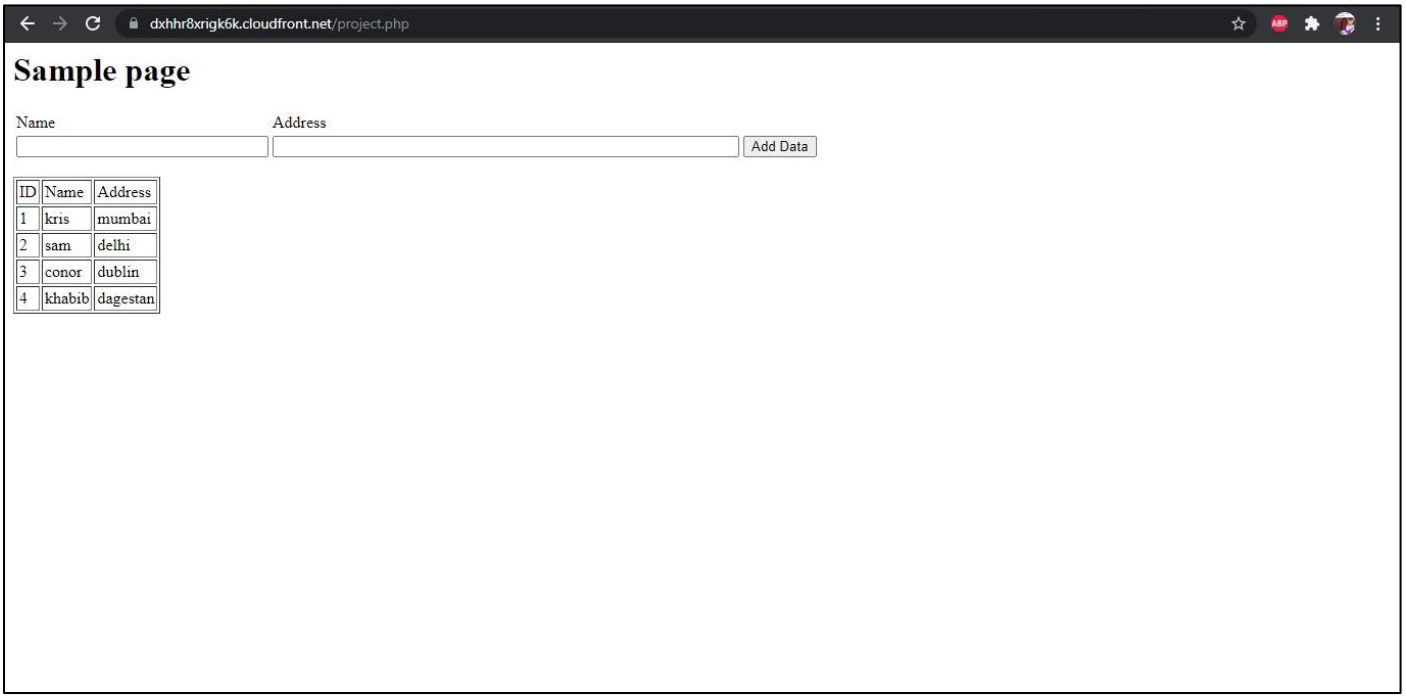

Sample Deployment
ec2-54-175-98-124.compute-1.amazonaws.com
Congratulations
This application was deployed using AWS CodeDeploy.
For next steps, read the AWS CodeDeploy Documentation.
Activate Windows
Go to Settings to activate Windows.

Use Case - 6:


PHP Application - AWS Elastic
https://sample835.tk
Congratulations!
Your AWS PHP Application is now running on your own dedicated environment in the AWS Cloud
You are running PHP version 7.4.3
This environment is launched with PHP Platform
What's Next?
AWS Elastic Beanstalk overview
Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git
Using Amazon RDS with PHP
Customizing the Software on EC2 Instances
Customizing Environment Resources
AWS SDK for PHP
AWS SDK for PHP home
PHP developer center
AWS SDK for PHP on GitHub


PHP Application - AWS Elastic
https://app.sample835.tk
Congratulations!
Your Second AWS PHP application is now running on your own dedicated environment in the AWS Cloud
You are running PHP version 7.4.3
This environment is launched with PHP Platform
What's Next?
AWS Elastic Beanstalk overview
Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git
Using Amazon RDS with PHP
Customizing the Software on EC2 Instances
Customizing Environment Resources
AWS SDK for PHP
AWS SDK for PHP home
PHP developer center
AWS SDK for PHP on GitHub

Use Case - 7:


dxhhr8xrigk6k.cloudfront.net/project.php
Sample page
Name
Address
Add Data
ID Name Address
1 kris mumbai
2 sam delhi
3 conor dublin
4 khabib dagestan

AWS PRICING

Free Tier:

Amazon services' prices vary and depend on the service that you are using. These prices can be calculated by different methods, but the main one is for the time of usage, as Amazon (AWS) says so, you pay for what you use. Hours are the main factor on this method, followed by the minutes and seconds of use of certain services.

Services that are available in the AWS Free Usage Tier

- 750 hours of Amazon EC2 Linux or RHEL or SLES t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support) – enough hours to run continuously each month
- 750 hours of an Elastic Load Balancer plus 15 GB data processing
- 750 hours of Amazon RDS Single-AZ Micro DB Instances, running MySQL, MariaDB, PostgreSQL, Oracle BYOL or SQL Server Express Edition – enough hours to run a DB Instance continuously each month. You also get 20 GB of database storage and 20 GB of backup storage
- 750 hours of Amazon ElastiCache Micro Cache Node usage – enough hours to run continuously each month.
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with EBS Magnetic) and 1 GB of snapshot storage
- 5 GB of Amazon S3 standard storage, 20,000 Get Requests, and 2,000 Put Requests
- 25 GB of Storage, 25 Units of Read Capacity and 25 Units of Write Capacity, enough to handle up to 200M requests per month with Amazon DynamoDB
- 25 Amazon SimpleDB Machine Hours and 1 GB of Storage
- 1,000 Amazon SWF workflow executions can be initiated for free. A total of 10,000 activity tasks, signals, timers and markers, and 30,000 workflow-days can also be used for free
- 100,000 Requests of Amazon Simple Queue Service
- 100,000 Requests, 100,000 HTTP notifications and 1,000 email notifications for Amazon Simple Notification Service
- 10 Amazon CloudWatch metrics, 10 alarms, and 1,000,000 API requests

- 50 GB Data Transfer Out, 2,000,000 HTTP and HTTPS Requests for Amazon CloudFront
- 15 GB of bandwidth out aggregated across all AWS services

Estimated Cost Per Use case:

Section A:

Use case - 1:

RDS MySQL cost (monthly) – 90 % CPU Utilization

31.536 USD

Storage pricing (monthly)

2.096 USD

DMS (per hour)

0.54 USD

Total Use Case cost (monthly):

34.17 USD

Use case - 2:

Redshift and S3:

Redshift instance cost (monthly)

229.95 USD

Data transfer cost (monthly)

0.86 USD

S3 Cost (monthly):

0.25 USD

Total Use Case cost (monthly):

230.81 USD

Use case - 3:

CloudFormation:

Monthly total of operations

- Total number of operations per extension: 5 per day * (730 hours in a month / 24 hours in a day) = 152.08 per month

Pricing calculations

- 100 extensions x 152.08 operations = 15,208 total operations
- 15,208 operations - 1000 free operations per month = 14,208 billable operations
- Max (14208 billable operations, 0 min billable seconds) = 14,208 operations (billable)
- 14,208 operations x 0.0009 USD = 12.79 USD for third-party extension operations
- **Total Use Case cost (monthly): 12.79 USD**

Use case - 4:

DynamoDB provisioned capacity estimate:

Monthly write cost (monthly)

26.64 USD

Monthly read cost (monthly)

2.12 USD

Total monthly cost:

28.76 USD

Upfront write cost (upfront)

171.00 USD

Upfront read cost (upfront)

34.20 USD

Total upfront cost:

205.20 USD

Amazon DynamoDB estimate:

Total monthly cost:

28.76 USD

Total upfront cost:

205.20 USD

Lambda:

Unit conversions

- Number of requests: 100000 per day * (730 hours in a month / 24 hours in a day) = 3041666.67 per month
- Amount of ephemeral storage allocated: 512 MB x 0.0009765625 GB in a MB = 0.5 GB

Pricing calculations

- 3,041,666.67 requests x 0.0000002 USD = 0.61 USD (monthly request charges)
- 0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function
- **Lambda costs - Without Free Tier (monthly): 0.61 USD**

SES:

Unit conversions

- Email messages sent from EC2: 100 per day * (730 hours in a month / 24 hours in a day) = 3041.67 messages per month

Pricing calculations

- 3,041.67 messages per month x 0.0001 USD = 0.3042 USD (Messages sent from EC2 cost)
- **SES usage cost (monthly): 0.30 USD**

Total Use Case Cost (monthly): 206.11 USD

Use case - 5:

OpsWorks for Chef Automate Pricing:

Node hour pricing:

Number of Nodes per month	Number of Node Hours	Pricing
First 150 nodes / month	Equal to 112,500 node hours	$0.0155 per node hour
Next 500 nodes / month	Equal to 375,000 node hours	$0.014 per node hour

Total Use Case cost (monthly): 0.914 USD

Use case - 6:

Amazon Cognito:

Advanced security feature cost (monthly): 5 USD

For 1 Monthly Active Users(MAU): 0.05 USD

For 100 MAU:

Cognito Monthly Active Users cost (monthly): 5.00 USD

No cost for Identity Pools

Total Use case Cost (monthly): 5.00 USD

Use case - 7:

There is no additional charge for AWS Elastic Beanstalk. You pay for AWS resources (e.g. EC2 instances or S3 buckets) you create to store and run your application. You only pay for what you use, as you use it; there are no minimum fees and no upfront commitments

EC2:

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.42 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.62 USD

Total monthly cost:

9.04 USD x 2

= 18.08 USD

RDS:

RDS MySQL cost (monthly) – 90% CPU Utilization

31.536 USD

Storage pricing (monthly) – 8GB

2.096 USD

Total monthly cost:

33.63 USD x 2

= 67.26 USD

CloudWatch:

Tiered price for: 10 metrics

10 metrics x 0.3000000000 USD = 3.00 USD

Total tier cost = 3.00 USD (Metrics cost (includes custom metrics))

CloudWatch Metrics cost (monthly): 3.00 USD

Application Load Balancer:

1 load balancers x 0.0239 USD per hour x 730 hours in a month = 17.45 USD

Application Load Balancer fixed hourly charges (monthly): 17.45 USD

SES:

Unit conversions

- Email messages sent from EC2: 100 per day * (730 hours in a month / 24 hours in a day) = 3041.67 messages per month

Pricing calculations

- 3,041.67 messages per month x 0.0001 USD = 0.3042 USD (Messages sent from EC2 cost)
- **SES usage cost (monthly): 0.30 USD**

CloudFront:

For 1GB/month data transfer out to internet:

CloudFront price United States (monthly)

0.09 USD

CloudFront price Canada (monthly)

0.09 USD

CloudFront price Asia Pacific (monthly)

0.12 USD

CloudFront price Australia (monthly)

0.11 USD

CloudFront price Europe (monthly)

0.09 USD

CloudFront price India (monthly)

0.11 USD

CloudFront price Japan (monthly)

0.11 USD

CloudFront price Middle East (monthly)

0.11 USD

CloudFront price South Africa (monthly)

0.11 USD

CloudFront price South America (monthly)

0.11 USD

Total monthly cost:

1.05 USD

NAT Gateway:

No. of NAT Gateways: 1

Data processed per NAT Gateway:

1 GB/month

Total for NAT Gateway: 40.94 USD

Elastic IP:

0 USD

S3:

S3 Cost (monthly):

0.25 USD

Total Use Case cost:

148.33 USD

Section - B:

Use case - 1:

Amazon DynamoDB estimate:

Total monthly cost:

28.76 USD

Total upfront cost:

205.20 USD

Amazon Cognito:

Advanced security feature cost (monthly): 5 USD

For 1 Monthly Active Users(MAU): 0.05 USD

For 100 MAU:

Cognito Monthly Active Users cost (monthly): 5.00 USD

No cost for Identity Pools

Total Cost (monthly): 5.00 USD

S3:

S3 Cost (monthly):

0.25 USD

Total Use Case cost (monthly):

210.45 USD

Use case - 2:

EC2:

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.42 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.62 USD

Total monthly cost:

9.04 USD

RDS:

RDS MySQL cost (monthly) – 90% CPU Utilization

31.536 USD

Storage pricing (monthly) – 8GB

2.096 USD

Total monthly cost:

33.63 USD

SQS:

Standard Request Queues:

10 Million per month

Total monthly cost:

3.60 USD

Total Use Case cost (monthly):

46.27 USD

Use case - 3:

EC2:

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.42 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.62 USD

Total monthly cost:

9.04 USD

VPC:

For Data Transfer rate:

VPC peering (different accounts and regions):

Instances: t2.micro

On-Demand Hourly rate:

0.0116 USD

NAT Gateway:

No. of NAT Gateways: 1

Data processed per NAT Gateway:

1 GB/month

Total for NAT Gateway: 40.94 USD

Total Use Case cost (monthly):

58.448 USD

Use case - 4:

EC2 (Mumbai):

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.42 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.62 USD

Total monthly cost:

For 2 EC2 Instances

9.04 USD x 2

= 18.08 USD

EC2 (N. Virginia):

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.00 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.26 USD

Total monthly cost:

For 2 EC2 Instances

8.26 USD x 2

= 16.52 USD

Application Load Balancer (Mumbai):

1 load balancers x 0.0239 USD per hour x 730 hours in a month = 17.45 USD

Application Load Balancer fixed hourly charges (monthly): 17.45 USD

Application Load Balancer (N. Virginia):

1 load balancers x 0.0225 USD per hour x 730 hours in a month = 16.43 USD

Application Load Balancer fixed hourly charges (monthly): 16.43 USD

Route 53:

Total monthly cost:

0.51 USD

Total Use Case cost (monthly):

68.99 USD

Use case - 5:

EC2:

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.42 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.62 USD

Total monthly cost:

9.04 USD

CloudWatch:

Tiered price for: 10 metrics

10 metrics x 0.3000000000 USD = 3.00 USD

Total tier cost = 3.00 USD (Metrics cost (includes custom metrics))

CloudWatch Metrics cost (monthly): 3.00 USD

CodeDeploy:

1 instance(s) x 20 deployment(s) x 0.02 USD = 0.40 USD for CodeDeploy

Total CodeDeploy cost (monthly): 0.40 USD

CodeCommit:

First 5 active users: 0.00 USD

Receives:

- 1,000 repositories per account; up to 25,000 upon request
- 50 GB-month of storage
- 10,000 Git requests/month

CodePipeline:

- 1 active pipelines - 1 free active pipeline each month = 0 total active pipelines per month
- **CodePipeline cost (monthly): 0.00 USD**

S3:

S3 Cost (monthly):

0.25 USD

Total Use Case cost (monthly):

12.69 USD

Use case - 6:

EC2:

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.42 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.62 USD

Total monthly cost:

9.04 USD

Application Load Balancer:

1 load balancers x 0.0239 USD per hour x 730 hours in a month = 17.45 USD

Application Load Balancer fixed hourly charges (monthly): 17.45 USD

Route 53:

Total monthly cost:

0.51 USD

Amazon Certificate Manager:

Public SSL/TLS certificates provisioned through Amazon Certificate Manager are free. You pay only for the Amazon Web Services resources you create to run your application.

Total ACM cost (monthly):

0 USD

Total Use Case cost (monthly):

27.00 USD

Use case - 7:

EC2:

Amazon Elastic Block Storage (EBS) pricing (monthly)

3.42 USD

Amazon EC2 Instance Savings Plans instances (monthly)

5.62 USD

Total monthly cost:

9.04 USD

RDS:

RDS MySQL cost (monthly) – 90% CPU Utilization

31.536 USD

Storage pricing (monthly) – 8GB

2.096 USD

Total monthly cost:

33.63 USD

CloudWatch:

Tiered price for: 10 metrics

10 metrics x 0.3000000000 USD = 3.00 USD

Total tier cost = 3.00 USD (Metrics cost (includes custom metrics))

CloudWatch Metrics cost (monthly): 3.00 USD

Application Load Balancer:

1 load balancers x 0.0239 USD per hour x 730 hours in a month = 17.45 USD

Application Load Balancer fixed hourly charges (monthly): 17.45 USD

SES:

Unit conversions

- Email messages sent from EC2: 100 per day * (730 hours in a month / 24 hours in a day) = 3041.67 messages per month

Pricing calculations

- 3,041.67 messages per month x 0.0001 USD = 0.3042 USD (Messages sent from EC2 cost)
- **SES usage cost (monthly): 0.30 USD**

CloudFront:

For 1GB/month data transfer out to internet:

CloudFront price United States (monthly)

0.09 USD

CloudFront price Canada (monthly)

0.09 USD

CloudFront price Asia Pacific (monthly)

0.12 USD

CloudFront price Australia (monthly)

0.11 USD

CloudFront price Europe (monthly)

0.09 USD

CloudFront price India (monthly)

0.11 USD

CloudFront price Japan (monthly)

0.11 USD

CloudFront price Middle East (monthly)

0.11 USD

CloudFront price South Africa (monthly)

0.11 USD

CloudFront price South America (monthly)

0.11 USD

Total monthly cost:

1.05 USD

NAT Gateway:

No. of NAT Gateways: 1

Data processed per NAT Gateway:

1 GB/month

Total for NAT Gateway: 40.94 USD

Elastic IP:

0 USD

Total Use Case cost (monthly):

105.41 USD

Note:

The above costs per use case is estimated. Detailed Pricing Link of AWS is mentioned in Bibliography

BIBLIOGRAPHY

- https://aws.amazon.com/architecture/icons/
- https://docs.aws.amazon.com/
- https://console.aws.amazon.com/
- https://aws.amazon.com/pricing/
- https://www.sourcefuse.com/blog/aws-free-tier-limits/
- https://docs.rightscale.com/cm/designers_guide/cm-cloud-computing-systemarchitecture-diagrams.html
- https://www.researchgate.net/figure/Activity-Diagram-of-VE-enabled-CloudEnterprise-Architecture-for-SMMEs_fig3_304287508
- https://www.researchgate.net/figure/Class-Diagram-for-a-Cloud-ComputingEnvironment_fig1_303165430
- https://aws.amazon.com/solutions/case-studies/indmoney/
- https://aws.amazon.com/solutions/case-studies/netflix/
- https://app.diagrams.net/?splash=0&libs=aws4
- https://mindmajix.com/aws-architecture
- https://www.freenom.com/en/index.html?lang=en
- https://aws.amazon.com/pricing/
- https://calculator.aws/#/