**Project Report on**

**3 Tier Infrastructure Development using AWS CloudFormation**



Submitted in partial fulfillment for the award of

**Post Graduate Diploma in High Performance Computing System Administration** from **C-DAC ACTS (Pune)**

**Guided by:**

**Mr. Roshan Gami**

Presented by:

## Mr.Prajyot Bhadange PRN: 220940127049 Mr.Amol Deogade PRN: 220940127031 Mr.Arvind Kumar sah PRN: 220940127033

## Mr.Abhishek Kumar PRN: 220940127001

## Mr. Tejas Waghmare PRN: 220940127024

**Centre of Development of Advanced Computing (C-DAC), Pune**



# TABLE OF CONTENTS

1. Abstract

2. Introduction and overview of project

2.1 Introduction of Cloud computing

2.2 3 tire Infrastructure using AWS

3. Introduction to CloudFormation

4. 3 tire Infrastructure Concepts

4.1 Get Started- AWS CloudFormation:

5. CloudWatch in AWS

5.1 SNS

6. Amazon RDS (Relational Database Service)

7. Load balancing

8. Nat Gateway

9. Auto Scaling

10. Results

11. Conclusion

## Abstract

This project endeavors to automate the deployment of a 3-tier web application in AWS utilizing CloudFormation while concurrently monitoring the infrastructure using CloudWatch. The proposed solution includes the design and implementation of a CloudFormation template to define the necessary infrastructure, which comprises three distinct layers - the presentation, application, and database layer - all of which will be deployed in separate Virtual Private Clouds (VPCs) to ensure optimal security and isolation. The presentation layer consists of an Amazon Elastic Load Balancer and Amazon EC2 instances that will run a web server, while the application layer comprises Amazon EC2 instances responsible for hosting application servers that will collaborate with the presentation layer to deliver required functionalities to end-users. On the other hand, the database layer entails an Amazon RDS instance responsible for storing data that the application servers utilize. Following deployment, CloudWatch will be deployed to monitor the infrastructure by setting up CloudWatch alarms to notify the team in the event of predefined threshold exceedances, for example, CPU utilization, memory utilization, network throughput, and disk usage. Ultimately, this approach will allow for proactive infrastructure monitoring and management, ensuring the web application's availability and optimal performance.

## Introduction and overview of project

With cloud computing becoming an increasingly popular choice for deploying web applications, Amazon Web Services (AWS) has emerged as a leading provider of cloud-based solutions. This project aims to harness the power of AWS by automating the deployment of a 3-tier web application using CloudFormation and monitoring the infrastructure with CloudWatch.

Our objective is to deliver a solution that is both reliable and scalable while being easy to deploy, manage, and monitor. To achieve this, we will design a CloudFormation template that outlines the infrastructure needed for hosting the web application, separating it into three distinct layers: presentation, application, and database. Each layer will be deployed in separate Virtual Private Clouds (VPCs) to ensure optimal security and isolation.

The presentation layer will include an Amazon Elastic Load Balancer (ELB) and Amazon EC2 instances running web servers, while the application layer will consist of Amazon EC2 instances running application servers that will interact with the presentation layer to provide the required functionalities to end-users. The database layer will feature an Amazon RDS instance hosting the database engine for storing data that the application servers utilize.

Once the infrastructure is deployed, we will utilize CloudWatch to monitor the resource usage and performance metrics of the infrastructure, setting up alarms to alert the team if predefined thresholds are exceeded, such as CPU utilization, memory usage, network throughput, and disk usage. This will enable us to monitor and manage the infrastructure proactively, ensuring the web application's availability and optimal performance.

Our project aims to deliver a cost-effective, reliable, and scalable solution for deploying and managing web applications in AWS. By automating the deployment process and monitoring the infrastructure using CloudFormation and CloudWatch, respectively, we will empower teams to deploy and manage web applications with ease while focusing on other critical tasks.

### 2.1 Introduction of Cloud computing

Cloud computing is a revolutionary approach to delivering computing services over the internet. It allows businesses and individuals to access computing resources, such as servers, storage, databases, and software applications, on demand and without the need for extensive on-premises infrastructure.

The benefits of cloud computing are many, including the ability to scale up or down quickly and easily, pay only for what you use, and reduce the need for physical hardware and software. Cloud computing also provides enhanced security features, as data is stored and processed on secure remote servers.

Moreover, cloud computing enables users to access their data and applications from anywhere with an internet connection, providing greater flexibility and accessibility. This makes it an ideal solution for businesses and individuals who need to work remotely or require flexible computing resources.

Overall, cloud computing offers many benefits and advantages over traditional computing models, making it an essential tool for modern businesses and individuals seeking agility, flexibility, and cost-effectiveness.

### 2.2 3 tire Infrastructure using AWS

AWS (Amazon Web Services) is a cloud computing platform that enables businesses to build and deploy scalable, flexible, and cost-effective applications and services. AWS offers a range of cloud-based solutions, including compute, storage, database, analytics, networking, security, and application development services, all accessible via the internet.

By using AWS, businesses can take advantage of the flexibility and scalability of cloud computing to innovate and grow their businesses, without being limited by the constraints of traditional IT infrastructure. AWS provides reliable, scalable, and secure cloud-based services, enabling businesses to build, test, and deploy applications quickly and easily.

AWS offers many benefits, including the ability to scale up or down as needed, pay only for what you use, and the ability to easily integrate with other AWS services. Additionally, AWS provides a range of tools and services to help businesses manage their infrastructure and applications, including monitoring, logging, and automation tools.

Overall, AWS is a powerful and reliable cloud platform that can help businesses of all sizes innovate and grow, while reducing costs and increasing efficiency.

The three-tier infrastructure is a well-established architectural design for building scalable and robust web applications or services. This design comprises three distinct layers, each with a unique function and purpose. The topmost layer, the presentation layer, is responsible for delivering the user interface to the users and is composed of web servers and load balancers. The middle layer, the application layer, processes the user requests and executes the business logic and comprises application servers and middleware components. The bottom layer, the data layer, stores and manages the data used by the application and consists of database servers and storage components.

The three-tier infrastructure offers several advantages, including improved scalability, fault tolerance, and security. Each layer can be scaled independently as needed, allowing for flexible resource allocation without impacting other layers. Additionally, the architecture offers enhanced manageability and performance, resulting in a more reliable and available application or service.

## Introduction to CloudFormation:

AWS CloudFormation is a service provided by Amazon Web Services that allows users to manage and deploy AWS resources using infrastructure-as-code. With CloudFormation, users can define their infrastructure in a JSON or YAML template, which can then be used to provision and configure AWS resources in a repeatable and automated way.

CloudFormation can manage a wide range of AWS resources, such as compute instances, storage, databases, load balancers, and security groups, among others. By using CloudFormation templates, users can define their entire infrastructure in a single file and easily deploy it across multiple environments, such as development, staging, and production.

CloudFormation offers several benefits, including easy versioning and management of infrastructure, as well as the ability to quickly create and modify resources in a consistent and repeatable way. CloudFormation templates can also be integrated with other AWS services, such as AWS Identity and Access Management (IAM) and AWS CloudTrail, to provide enhanced security and auditability.

Overall, AWS CloudFormation is a powerful tool for managing AWS resources, allowing users to create, deploy, and manage infrastructure-as-code with ease and consistency

.

## 3 tire Infrastructure Concepts:

A three-tier infrastructure is a common architecture used for web applications that separates the application into three distinct layers: presentation, application logic, and data storage. Each layer is designed to perform specific functions, and the layers work together to deliver a complete web application.

**The presentation layer, also known as the front-end layer**, is responsible for presenting the application's user interface to the user. This layer is typically implemented using HTML, CSS, and JavaScript and is hosted on web servers. The presentation layer communicates with the application logic layer to retrieve and display data.

In a three-tier infrastructure, the presentation layer communicates with the application logic layer to retrieve and display data. This separation of concerns allows for greater flexibility and maintainability of the application, as changes to the front-end can be made without affecting the back-end or middle-tier.

**The application logic layer, also known as the middle tier**, is responsible for processing user requests and business logic. This layer is typically implemented using server-side scripting languages such as PHP, Python, or Java, and it runs on application servers. The application logic layer communicates with the data storage layer to retrieve and store data.

The application logic layer also communicates with other services, such as payment gateways, to fulfill user requests. This layer contains the business logic of the application and is responsible for performing tasks like authentication, authorization, and data validation.

In a three-tier infrastructure, the application logic layer acts as a mediator between the presentation layer and the data storage layer. This separation of concerns allows for greater flexibility and maintainability of the application, as changes to the business logic can be made without affecting the front-end or back-end.

The application logic layer can also be scaled independently of the other layers to meet the demands of the application. For example, additional application servers can be added to handle increased traffic or requests, without affecting the presentation or data storage layers.

**The data storage layer, also known as the back-end layer**, is responsible for storing and retrieving data. This layer is typically implemented using a relational database management system (RDBMS), such as MySQL, Oracle, or Microsoft SQL Server, and it runs on database servers. The data storage layer communicates with the application logic layer to provide data for processing.

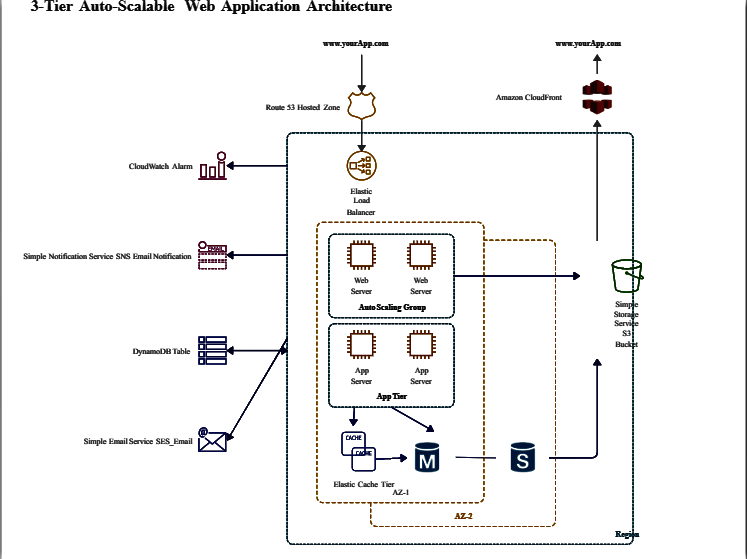
The data storage layer receives requests from the application logic layer to retrieve or store data. It can perform tasks like data validation, data normalization, and data indexing to ensure data consistency and availability. The layer can also be optimized for performance and scalability, with techniques like partitioning, replication, and caching used to handle large amounts of data.

In a three-tier infrastructure, the data storage layer communicates with the application logic layer to provide data to the presentation layer. This separation of concerns allows for greater flexibility and maintainability of the application, as changes to the data storage layer can be made without affecting the front-end or business logic.

The data storage layer can also be scaled independently of the other layers to meet the demands of the application. For example, additional database servers can be added to handle increased data volume or traffic, without affecting the front-end or application logic layers.

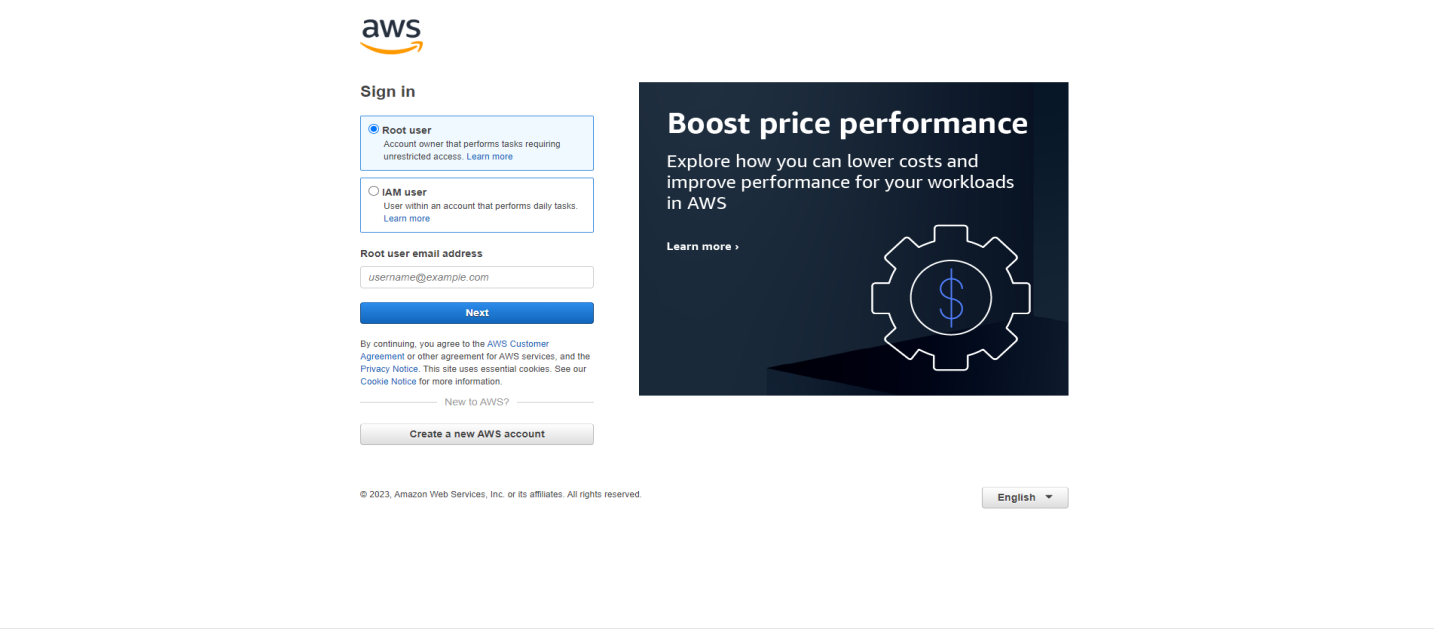
The three-tier infrastructure is designed to provide scalability, flexibility, and maintainability. Each layer can be scaled independently to meet the demand of the application, and changes can be made to one layer without affecting the others. Additionally, the three-tier infrastructure provides a clear separation of concerns, making it easier to maintain and update the application over time.

Overall, the three-tier infrastructure is a common architecture used for web applications that separates the application into three distinct layers, each responsible for specific functions, and working together to deliver a complete web application.

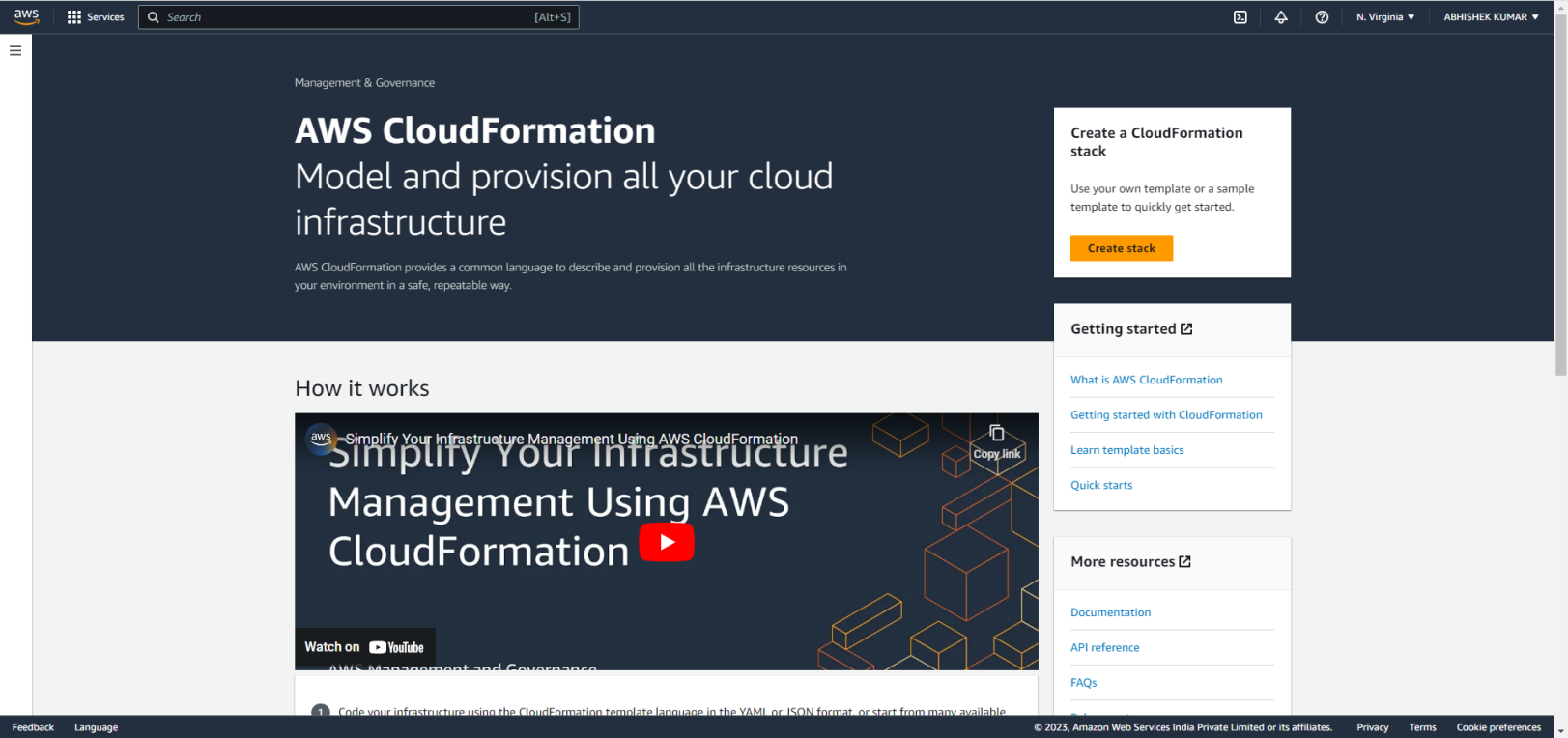


## Get Started- AWS CloudFormation:

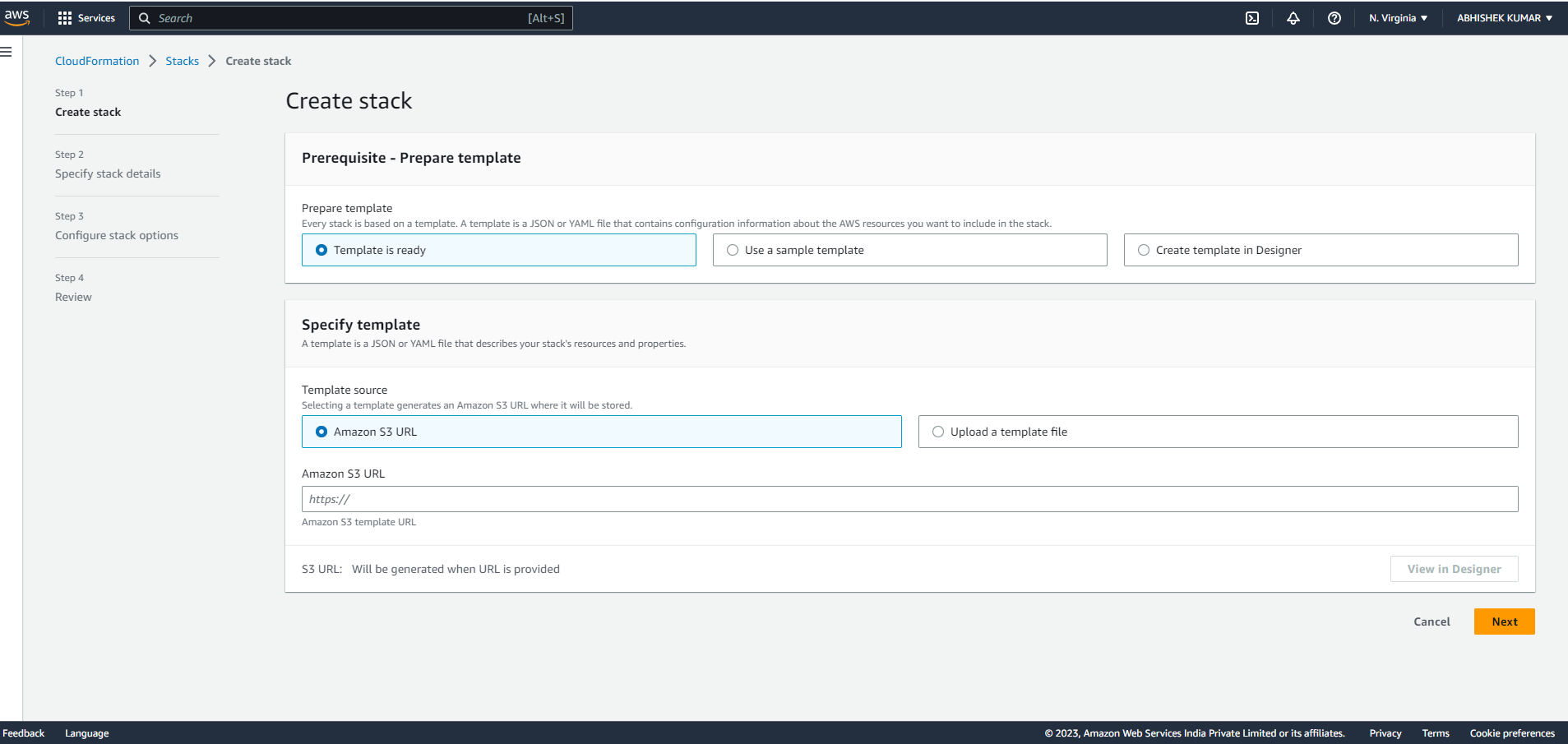
1. Create an AWS account: If you don't already have one, sign up for an AWS account at [https://aws.amazon.com/](https://aws.amazon.com/" \t "_new).



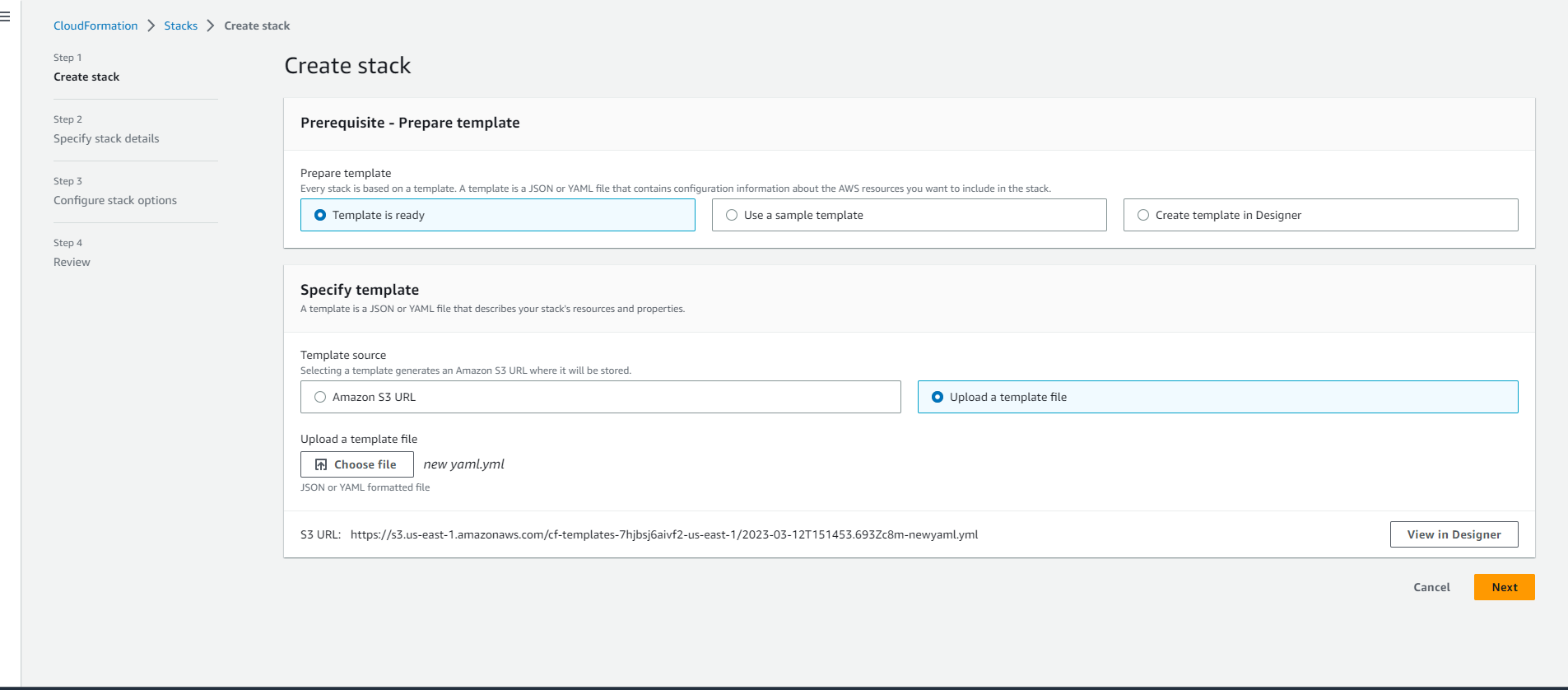
1. Access the CloudFormation console: Once you have an AWS account, log in to the AWS Management Console and navigate to the CloudFormation service.



1. Create a new stack: In the CloudFormation console, click the "Create stack" button to create a new stack.

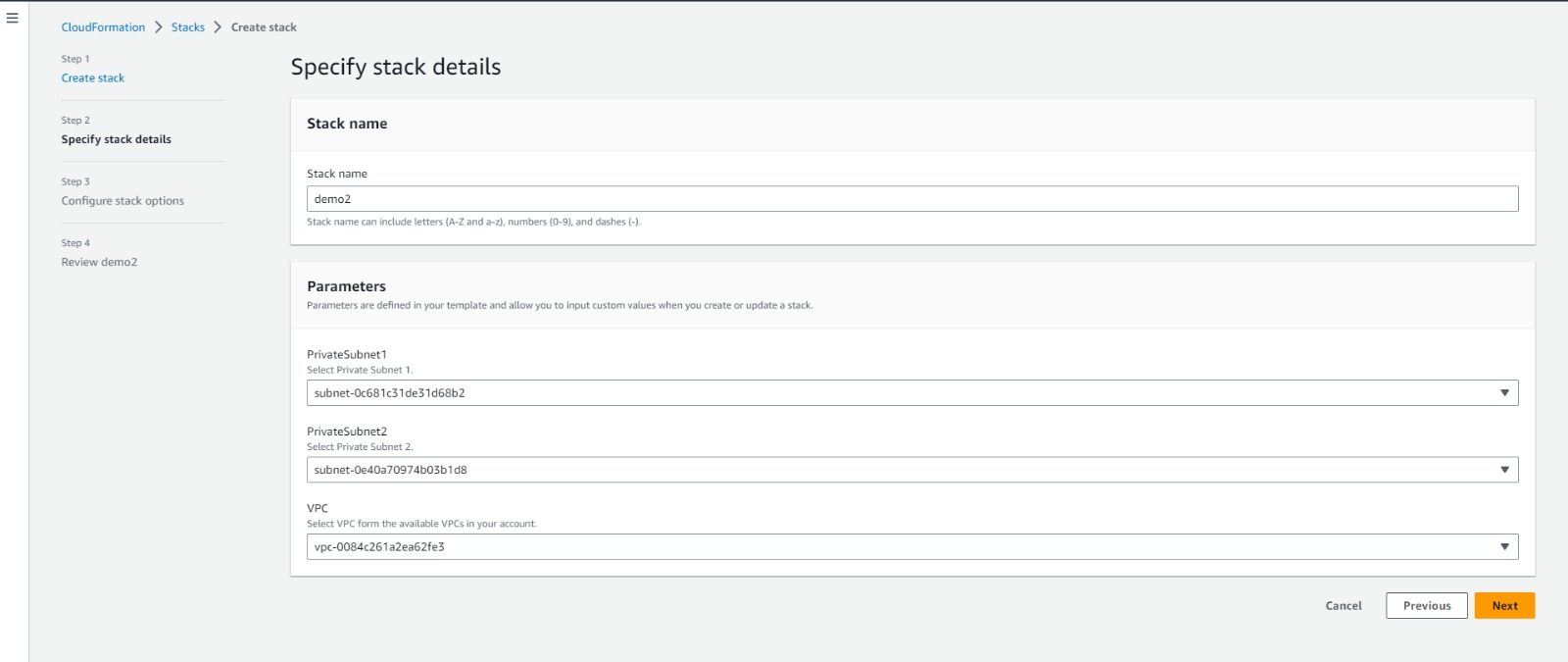


1. Define a template: In the stack creation wizard, you can define your template using either JSON or YAML format. The template defines the AWS resources that you want to create, such as EC2 instances, load balancers, and databases.

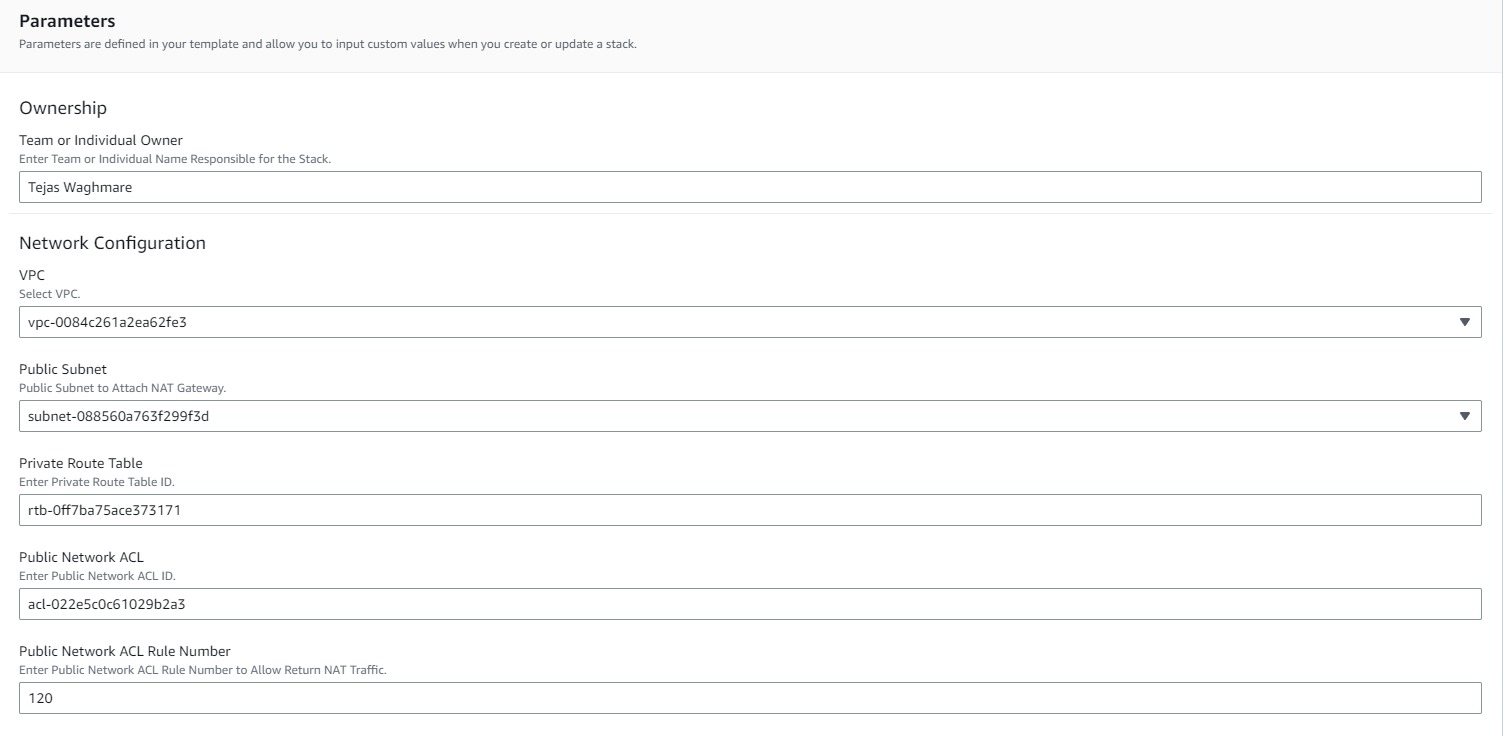


1. Launch the stack: Once you have defined your template, you can launch the stack to create the resources and configure them according to the parameters specified in the template.

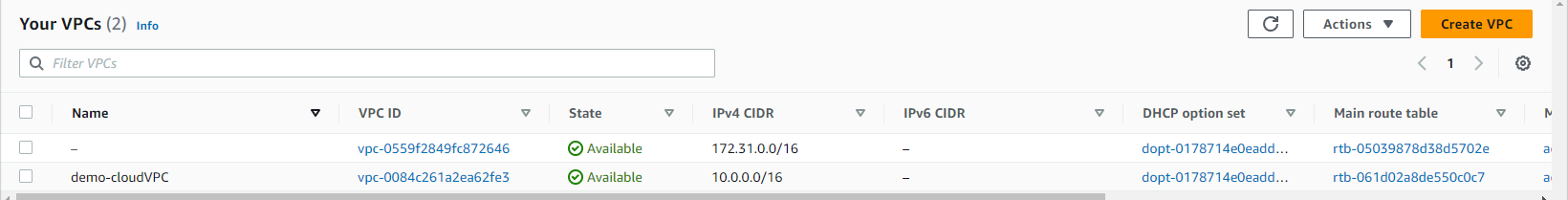
Stack Name



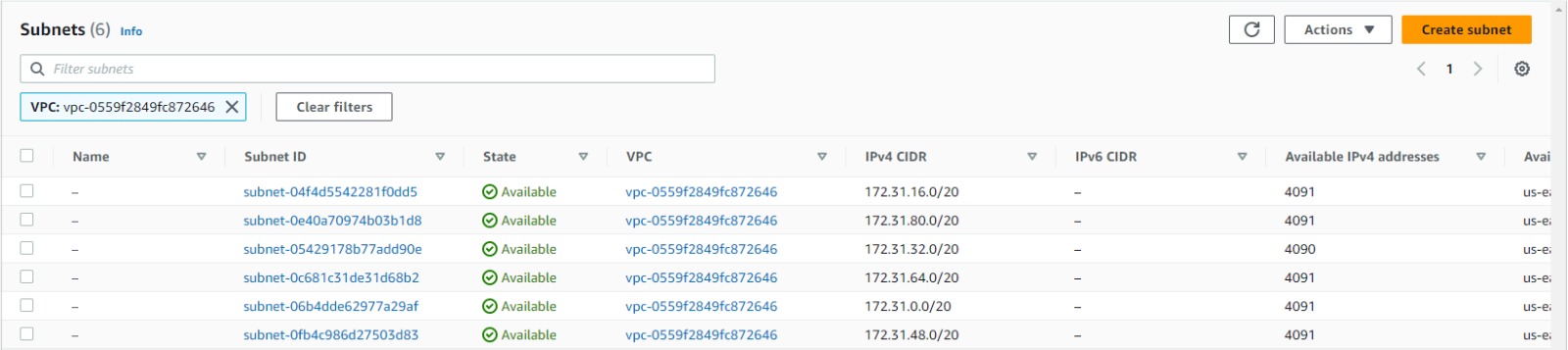
Parameters



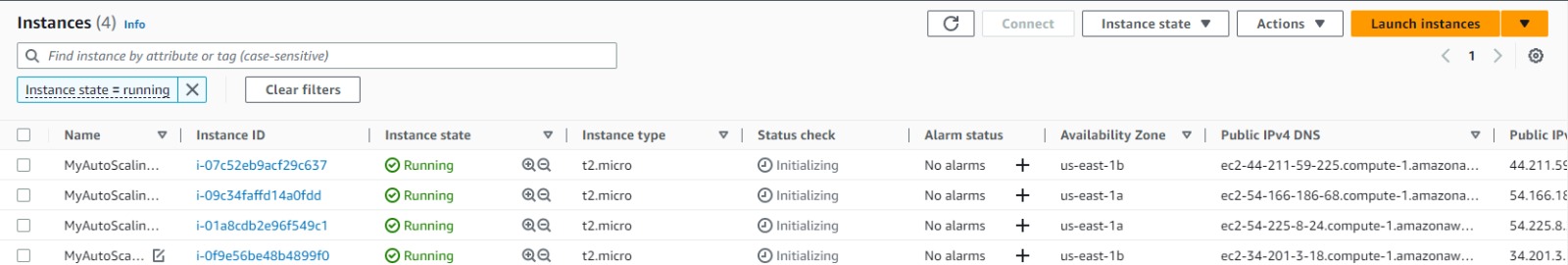
VPC



Subnets



Instances



Monitor and manage your stack: After launching the stack, you can monitor and manage it using the CloudFormation console or the AWS CLI or SDKs. This includes updating the stack, deleting resources, and rolling back changes if necessary.

Overall, getting started with AWS CloudFormation involves creating an account, accessing the console, defining a template, launching a stack, and then monitoring and managing your resources as needed.

## 5.CloudWatch in AWS:

Amazon Web Services (AWS) CloudWatch is a monitoring and logging service that provides real-time visibility into your AWS resources, applications, and services. CloudWatch enables you to monitor and analyze metrics and logs to identify and troubleshoot issues, as well as create alarms to notify you when specific metrics reach a certain threshold.

Features

1. **Metrics and Logging**: CloudWatch collects and displays metrics from your AWS resources and applications, such as CPU utilization, network traffic, and database connections. You can use these metrics to track the performance and health of your systems, as well as identify any bottlenecks or areas for optimization. In addition to metrics, CloudWatch also provides a centralized repository for logs from your applications and AWS resources.
2. **Alarms**: CloudWatch alarms enable you to create customized alerts based on specific metrics or log events. When a metric or log event reaches a certain threshold, CloudWatch can send an email, SMS, or other notification to alert you of the issue. This enables you to take action before problems occur, such as scaling resources, restarting instances, or investigating issues.
3. **Dashboards**: CloudWatch also provides dashboards that enable you to customize and visualize your metrics, logs, and alarms in a single place. You can create multiple dashboards for different teams or use cases, and customize them with different widgets to display specific metrics or logs.
4. **Enhanced Monitoring and Troubleshooting**: CloudWatch provides several features for enhanced monitoring and troubleshooting, including CloudWatch Logs Insights to run advanced queries on your log data, CloudWatch Metrics Filters to extract specific data from your logs and publish them as custom metrics, and support for Amazon CloudWatch Agent to collect and publish additional system-level metrics, logs, and events.
5. **Integration**: CloudWatch integrates with other AWS services, such as EC2, RDS, and Lambda, as well as third-party tools, making it a flexible and extensible monitoring solution. CloudWatch also provides integration with AWS CloudTrail, which provides a record of all API calls made within your AWS account.
6. **Automated Actions**: CloudWatch provides support for automated actions through Amazon CloudWatch Events, allowing you to create rules that automatically trigger actions based on specific events. These actions can be performed by AWS services or by custom applications that you configure as targets.
7. **Cross-Account and Cross-Region Monitoring**: CloudWatch provides support for monitoring resources and events across multiple AWS accounts and regions from a single console.

Overall, AWS CloudWatch is an essential tool for monitoring and managing your AWS resources and applications, providing real-time visibility and alerts that help you maintain the health and performance of your systems. By leveraging CloudWatch, you can quickly identify and resolve issues, optimize resource usage, and ensure the availability and reliability of your systems.

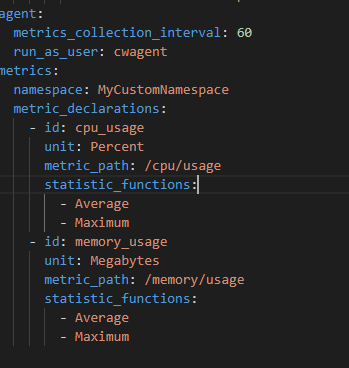
CloudWatch Agent is a software agent provided by Amazon Web Services (AWS) that can be installed on EC2 instances or on-premises servers to collect system-level metrics, logs, and events. The agent runs on the host operating system and sends the collected data to CloudWatch, a monitoring service provided by AWS.

The CloudWatch Agent supports multiple operating systems including Windows, Linux, and MacOS. It can be configured to collect a wide range of metrics such as CPU usage, disk usage, network traffic, memory usage, and custom metrics. It can also collect log files from the operating system, applications, and custom log files.

By collecting and sending metrics and logs to CloudWatch, the agent enables users to gain insights into the performance and health of their applications and infrastructure. This can help with troubleshooting, identifying and resolving issues, and optimizing resource utilization. The agent can also be used to trigger alarms and notifications based on certain metric thresholds or log patterns.

here are some additional details about the CloudWatch Agent:

* Installation: The CloudWatch Agent can be installed manually on an instance or server or it can be installed using AWS Systems Manager, AWS CloudFormation, or AWS OpsWorks.
* Configuration: The agent is configured using JSON files which specify the metrics, logs, and events to be collected, and how often to collect them. Configuration files can be stored locally on the instance or server or they can be stored in an S3 bucket.
* Custom Metrics: In addition to the default metrics collected by the agent, users can also configure custom metrics to be collected. Custom metrics can be defined based on application-specific metrics, log data, or other sources.
* Log Processing: The agent can process logs before sending them to CloudWatch. This can include filtering, aggregating, or transforming log data.
* Integration with AWS Services: The agent can be integrated with other AWS services such as AWS Lambda, Amazon SNS, and Amazon Kinesis. This enables users to trigger automated actions or send notifications based on certain metrics or log events.
* Pricing: The CloudWatch Agent is free to use, but there may be charges associated with storing and analyzing the data collected by the agent in CloudWatch. Users should consult the AWS pricing documentation for details on pricing.



## SNS:

Amazon Simple Notification Service (SNS) is a messaging service provided by Amazon Web Services (AWS) that allows users to send and receive messages between distributed software components and microservices, as well as between other AWS services.

SNS enables users to publish messages to topics, which are logical access points or channels for messages. Subscribers can then receive the published messages by subscribing to the topic. SNS supports a variety of message protocols including HTTP, HTTPS, email, SMS, mobile push notifications, and other custom protocols.

SNS offers a number of features and benefits including:

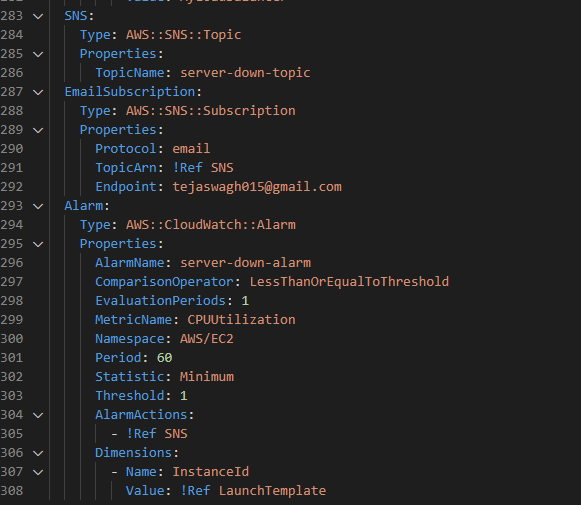
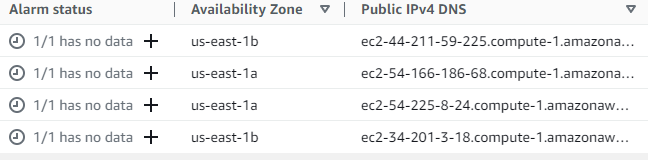
* Scalability: SNS is designed to handle high throughput of messages and can scale to meet the demands of large-scale applications and services.
* Reliability: SNS ensures reliable message delivery by automatically retrying failed deliveries and providing delivery status notifications.
* Flexibility: SNS supports a variety of message protocols and allows users to choose the most appropriate protocol for their use case.
* Customization: SNS provides the ability to customize message formatting and filtering based on message attributes.
* Integration: SNS integrates with a wide variety of other AWS services such as Amazon S3, AWS Lambda, Amazon CloudWatch, and Amazon SQS.
* Security: SNS provides a number of security features such as message encryption, access control, and monitoring and logging of message activity.

Overall, SNS is a powerful messaging service that can be used to build flexible and scalable applications and services in AWS.

Here are some additional details about Amazon SNS:

* Topics: A topic in SNS is a logical access point or channel for messages. Users can publish messages to a topic and subscribers can receive the messages by subscribing to the topic. Topics can be created, updated, or deleted using the AWS Management Console, AWS CLI, or SDKs.
* Subscribers: A subscriber in SNS is an endpoint or service that receives messages published to a topic. SNS supports a wide range of subscribers including email addresses, mobile devices, HTTP/S endpoints, AWS Lambda functions, and more.
* Message Attributes: SNS provides the ability to customize message formatting and filtering based on message attributes. Users can add custom attributes to messages which can be used to filter messages based on specific criteria.
* Delivery Policies: SNS provides delivery policies which can be used to control the rate at which messages are delivered to subscribers. Delivery policies can be used to set limits on message delivery, retry behavior, and more.
* Access Control: SNS provides access control features to allow users to manage access to their topics and subscribers. Users can define policies that control who can publish messages to a topic or subscribe to a topic.
* Monitoring and Logging: SNS provides monitoring and logging features that enable users to monitor the activity of their topics and subscribers. Users can view metrics, configure alarms, and set up CloudWatch Logs to track message delivery activity.
* Pricing: SNS is a pay-as-you-go service and users are charged based on the number of messages sent and received, as well as any other additional features they use. Users should consult the AWS pricing documentation for details on pricing.

Overall, Amazon SNS is a versatile messaging service that enables users to build scalable and flexible applications and services in AWS.



## Amazon RDS (Relational Database Service):

Amazon RDS (Relational Database Service) is a managed database service provided by AWS that allows you to easily set up, operate, and scale a relational database in the cloud. In a 3-tier architecture, RDS can be used to store and manage the application data layer. Here are some of the benefits of using RDS in a 3-tier architecture:

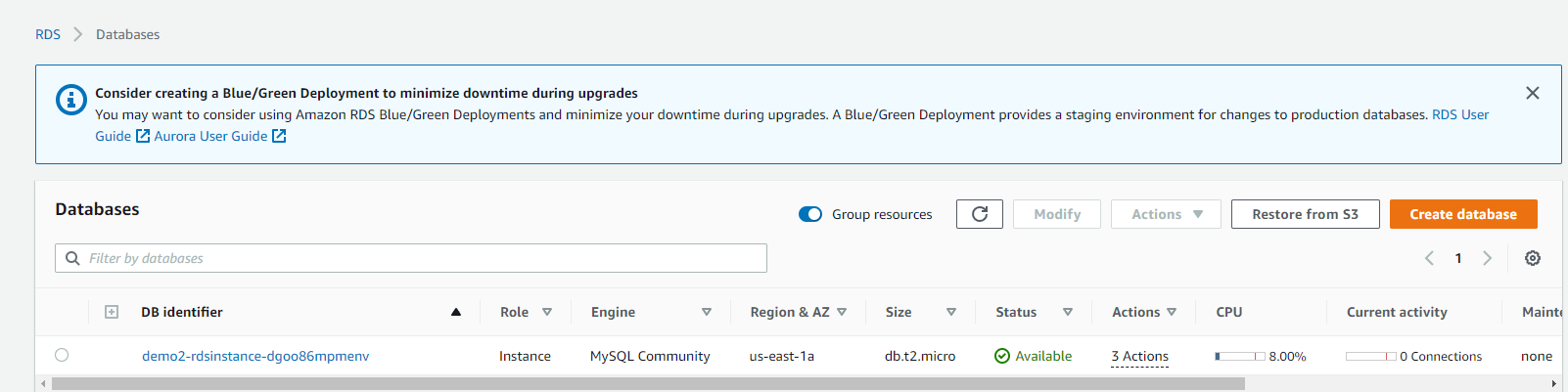
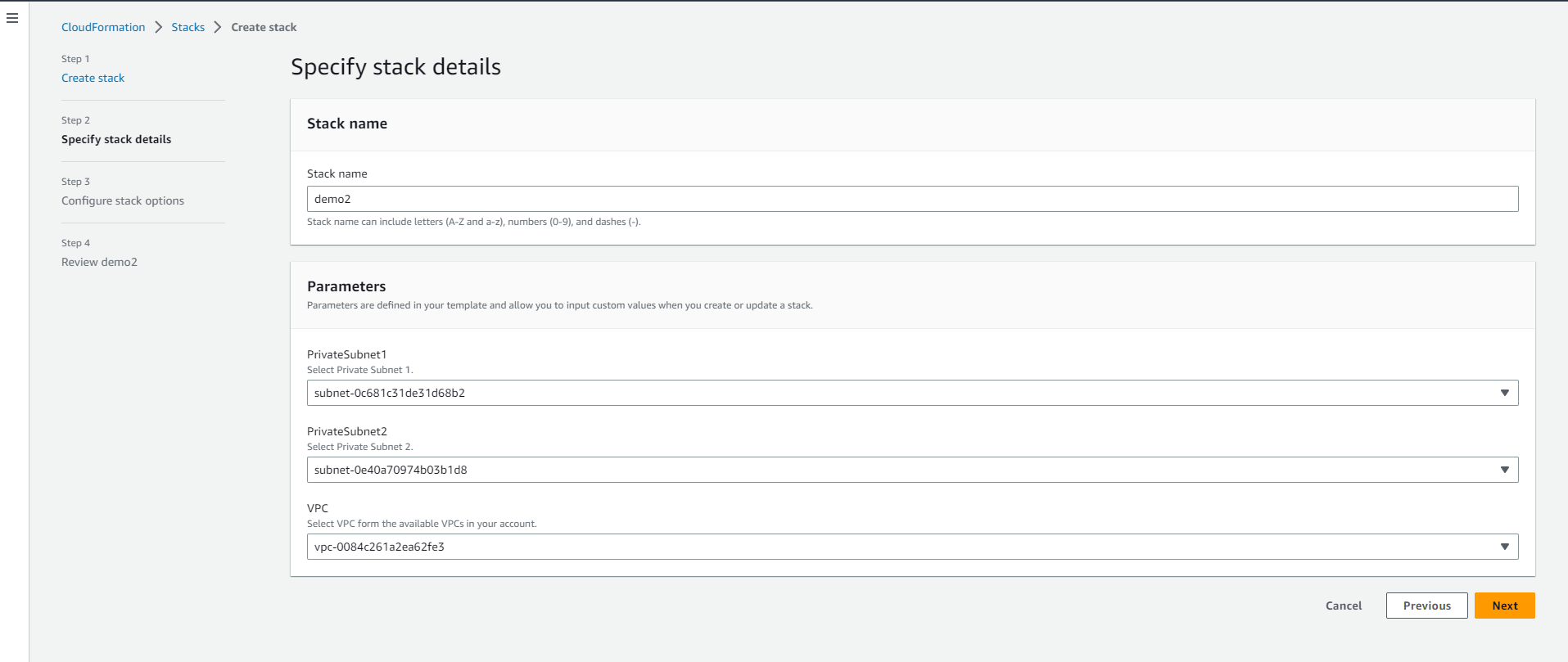
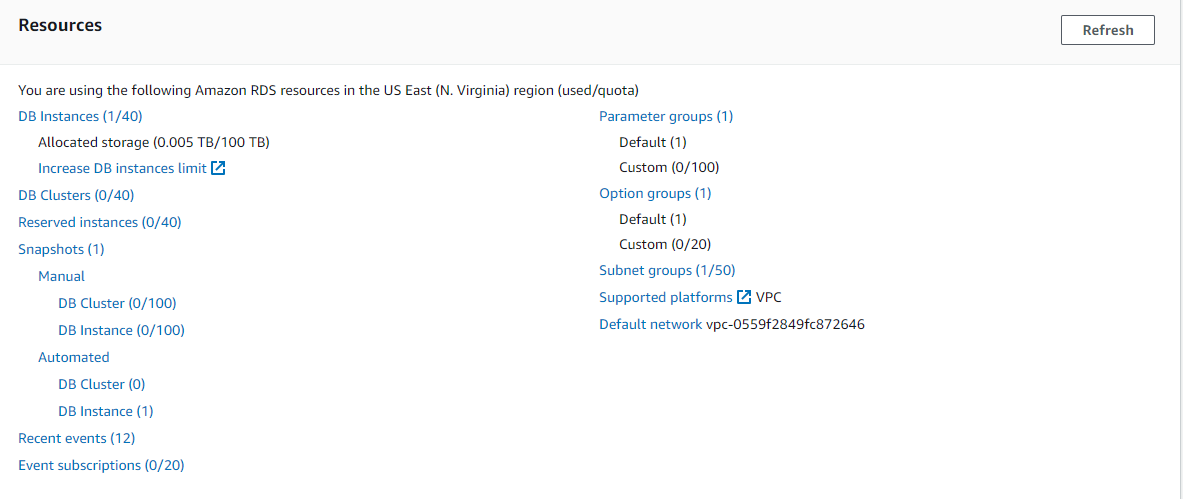
1. Scalability: RDS allows you to easily scale your database resources up or down depending on the needs of your application. This can help to ensure that your application remains responsive even as the load on your database increases.
2. High availability: RDS provides automated backups, replication, and failover to help ensure that your database remains available even in the event of a failure. This can help to improve the availability and reliability of your application.
3. Security: RDS provides several security features, including encryption at rest and in transit, security groups, and database auditing. This can help to ensure that your application data remains secure.
4. Cost-effective: RDS allows you to pay only for the resources that you use, making it a cost-effective option for managing your application data layer.

In a 3-tier architecture, RDS can be used as the data tier, with the web tier and application tier running on separate EC2 instances. The application tier can access the database using the RDS endpoint, which can be configured for high availability and scalability using features such as Multi-AZ and read replicas. RDS can also be integrated with other AWS services such as AWS Identity and Access Management (IAM), AWS CloudTrail, and Amazon CloudWatch for additional security, monitoring, and management capabilities.

In addition to the benefits mentioned earlier, using RDS in a 3-tier architecture can also provide the following advantages:

1. Easy deployment and management: RDS allows you to easily launch and manage database instances using the AWS Management Console, CLI, or API. This can help to reduce the time and effort required to set up and manage your database infrastructure.
2. Compatibility with popular database engines: RDS supports several popular database engines such as MySQL, PostgreSQL, Oracle, and SQL Server. This allows you to choose the database engine that best suits your application requirements.
3. Flexible storage options: RDS provides several storage options such as General Purpose SSD, Provisioned IOPS, and Magnetic storage. This allows you to choose the storage option that best suits your performance and cost requirements.
4. Integration with other AWS services: RDS can be easily integrated with other AWS services such as AWS Lambda, Amazon API Gateway, and Amazon S3 to build scalable and serverless applications.

Overall, using RDS in a 3-tier architecture can help to improve the performance, scalability, availability, and security of your application data layer. With its ease of use, compatibility with popular database engines, flexible storage options, and integration with other AWS services, RDS is a popular choice for managing application data in the cloud.



**7.Load balancing**

Load balancing is a technique used to distribute incoming network traffic across multiple servers or resources to improve the performance, availability, and scalability of applications. In AWS, there are two types of load balancers available:

1. Application Load Balancer (ALB): This is a layer 7 load balancer that is used to distribute incoming traffic based on URL patterns, HTTP headers, or query strings. It supports routing traffic to multiple services or containers within an Amazon Elastic Container Service (ECS) cluster.
2. Network Load Balancer (NLB): This is a layer 4 load balancer that is used to distribute incoming traffic based on IP protocol data. It supports routing traffic to multiple EC2 instances, containers, or IP addresses.

Here are some of the benefits of using load balancing in AWS:

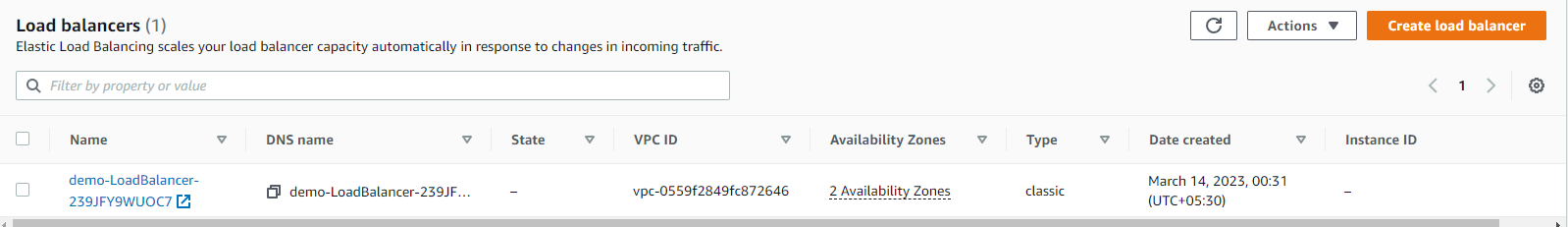
1. Improved performance: Load balancing helps to distribute traffic evenly across multiple servers or resources, which can help to reduce the response time of applications and improve the user experience.
2. Increased availability: Load balancing can help to improve the availability of applications by redirecting traffic to healthy instances or resources in the event of a failure.
3. Scalability: Load balancing can help to improve the scalability of applications by adding or removing instances or resources based on the traffic demand.
4. Security: Load balancing can help to improve the security of applications by providing features such as SSL termination, cross-zone load balancing, and connection draining.

In a 3-tier architecture, load balancing can be used to distribute incoming traffic across multiple EC2 instances running in the application tier. This can help to improve the availability, scalability, and performance of the application by reducing the load on individual instances and by providing automatic failover in the event of a failure. Load balancing can also be integrated with other AWS services such as Auto Scaling, CloudWatch, and Route 53 to provide additional management, monitoring, and DNS capabilities.

here are some additional important points about load balancing in AWS:

1. Elasticity: Load balancing allows you to easily add or remove instances or resources to meet the changing traffic demands of your application. This can help to improve the elasticity of your infrastructure and reduce the risk of overprovisioning or underprovisioning resources.
2. Health checks: Load balancing provides health checks that continuously monitor the status of instances or resources and automatically redirect traffic to healthy instances or resources. This can help to improve the availability and reliability of your application.
3. SSL termination: Load balancing supports SSL termination, which allows you to offload the processing of SSL certificates from the application instances or resources. This can help to reduce the workload on the instances or resources and improve the performance of your application.
4. Cross-zone load balancing: Load balancing supports cross-zone load balancing, which allows you to distribute traffic evenly across multiple availability zones. This can help to improve the availability and resiliency of your application by reducing the impact of a single availability zone failure.
5. Connection draining: Load balancing provides connection draining, which allows you to gracefully terminate existing connections to instances or resources that are being removed from the load balancer. This can help to avoid disruption to users and ensure a smooth transition to the new instances or resources.

Overall, load balancing is an important component of a 3-tier architecture in AWS. By distributing traffic across multiple instances or resources, load balancing can help to improve the availability, scalability, and performance of your application. With its elasticity, health checks, SSL termination, cross-zone load balancing, and connection draining features, load balancing provides a powerful tool for managing your application traffic in the cloud.



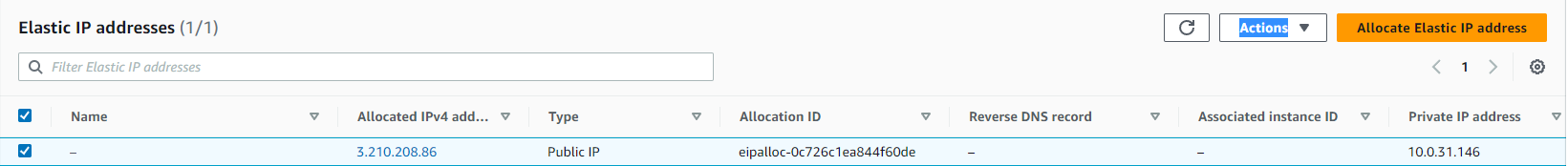
**8.Nat Gateway**

A NAT (Network Address Translation) Gateway is a highly available managed service provided by Amazon Web Services (AWS) that allows resources within a private subnet in a virtual private cloud (VPC) to access the internet or other AWS services while remaining isolated from the public internet.

A NAT Gateway functions as an intermediary between the resources in the private subnet and the internet or other AWS services. It translates the private IP addresses of the resources in the private subnet to the public IP address assigned to the NAT Gateway, and vice versa. This allows the resources in the private subnet to communicate with the internet or other AWS services without exposing their private IP addresses.

Some of the benefits of using a NAT Gateway in AWS include:

* Improved security: By using a NAT Gateway, resources within a private subnet can access the internet or other AWS services without exposing their private IP addresses.
* Reduced operational overhead: NAT Gateway is a managed service provided by AWS, which means that AWS handles the maintenance and scaling of the NAT Gateway infrastructure.
* High availability: NAT Gateway is designed to be highly available and provides automatic fail-over capabilities across multiple availability zones.
* Scalability: NAT Gateway can handle up to 45 Gbps of network traffic, and multiple NAT Gateways can be used to handle larger traffic volumes.
* Cost-effective: AWS charges for NAT Gateway based on the amount of data processed by the service, which can be a cost-effective solution compared to using a dedicated hardware device.



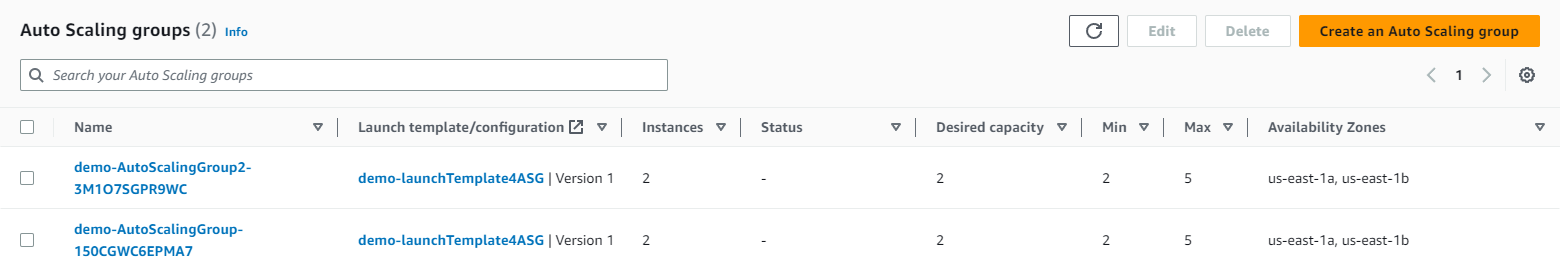
## 9.Auto Scaling:

Auto Scaling is an AWS service that automatically adjusts the capacity of your EC2 instances according to the demand of your application. It enables you to dynamically add or remove instances to match the current load on your application, helping to maintain optimal performance while minimizing costs.

Using Auto Scaling has a number of key benefits, including increased availability, improved performance, cost savings, and flexibility. By automatically adjusting the number of instances based on the current load, Auto Scaling ensures that your application always has enough capacity to handle the workload, while also avoiding overprovisioning or underprovisioning resources.

In a 3-tier architecture, Auto Scaling can be used to automatically adjust the number of instances running in the application tier based on the demand of your application. For example, you can define a scaling policy that increases the number of instances when the CPU utilization exceeds a certain threshold, and decreases the number of instances when the CPU utilization falls below that threshold. This helps to ensure that your application always has the required capacity, while minimizing costs.

Auto Scaling can be integrated with other AWS services such as Elastic Load Balancing, CloudWatch, and AWS Identity and Access Management (IAM) to provide additional management, monitoring, and security capabilities.

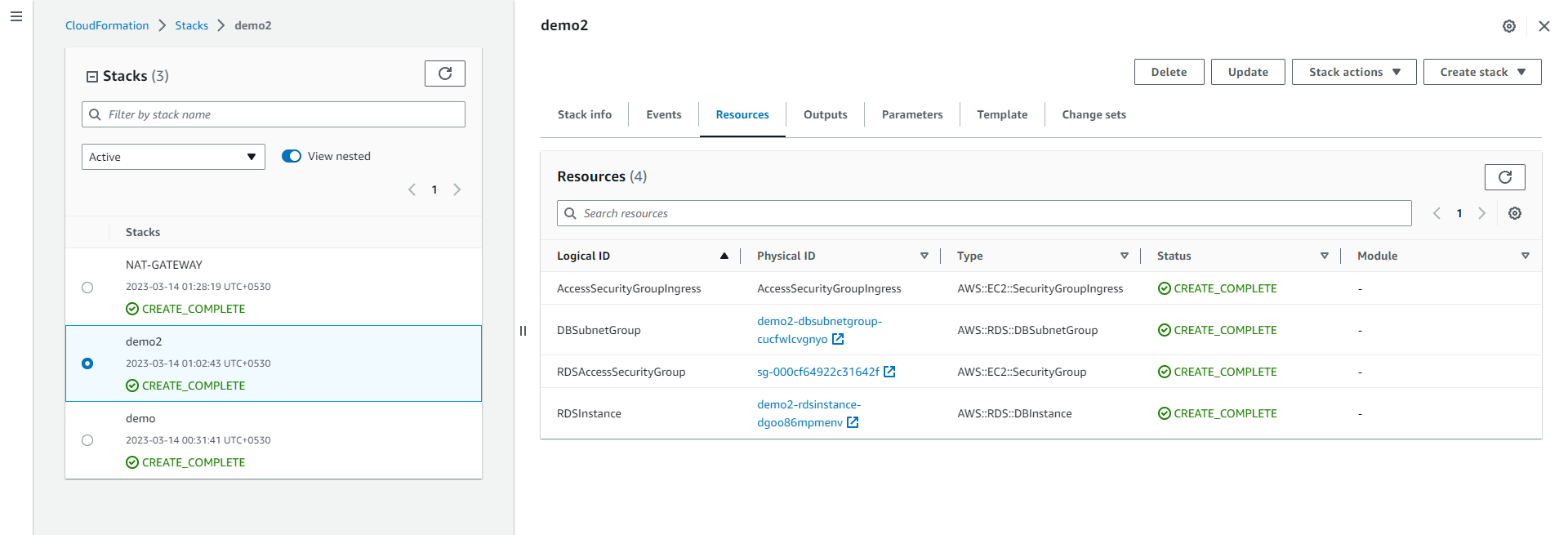


**10.Results :**

Results for the project of automated deployment of a 3-tier web application in AWS using CloudFormation and monitoring using CloudWatch could include:

1. Reduced deployment time: Automated deployment with CloudFormation can significantly reduce the time required to deploy the 3-tier web application. This can be measured by comparing the time taken to deploy the application manually versus the time taken using automated deployment.
2. Improved scalability: By using CloudFormation and Auto Scaling, the 3-tier web application can be made more scalable, allowing it to handle an increasing number of users and traffic. This can be measured by tracking the number of instances added or removed by Auto Scaling in response to changes in demand.
3. Improved performance: CloudWatch can be used to monitor the performance of the 3-tier web application, including resource usage and performance metrics such as response time and latency. By setting up alarms for these metrics, the performance of the application can be improved and issues can be quickly identified and resolved.
4. Cost savings: Automated deployment using CloudFormation and Auto Scaling can help to minimize costs by optimizing resource usage and avoiding overprovisioning. This can be measured by comparing the cost of running the 3-tier web application using manual deployment versus automated deployment.
5. Improved reliability: By automating the deployment process and monitoring resource usage and performance metrics using CloudWatch, the 3-tier web application can be made more reliable and resilient to failures. This can be measured by tracking the number of incidents and downtime, and comparing these metrics with those of manual deployment.

Overall, the results and benchmarks of the project would depend on factors such as the size and complexity of the 3-tier web application, the level of automation implemented using CloudFormation and Auto Scaling, and the performance and reliability of the application. By continuously monitoring and optimizing the deployment and monitoring

 process, the results and benchmarks can be improved over time.

## 11.Conclusion

The project '3 tier infrastructure development using CloudFormation' aimed to develop a robust and scalable infrastructure for hosting web applications using Amazon Web Services (AWS). The three-tier architecture included a presentation layer, application layer, and database layer.

The project was implemented using AWS CloudFormation, which allowed for easy creation and management of AWS resources in a repeatable and automated way. The CloudFormation templates were used to define the infrastructure, including networking, EC2 instances, load balancers, RDS database, and security groups.

The project was successful in achieving its goals, as it provided a highly available and scalable infrastructure with automatic scaling capabilities. The CloudFormation templates allowed for easy deployment of the infrastructure across multiple environments, such as development, staging, and production.

Overall, the project demonstrated the benefits of using CloudFormation for infrastructure development, such as increased efficiency, consistency, and scalability. It also highlighted the importance of following best practices for security, monitoring, and cost optimization when designing and deploying infrastructure in the cloud.

## 

## 12.Bibliography

* https://www.amazon.in/Cloud-Computing-Black-Kailash-Jayaswal/dp/9351194183
* Bottom of Form
* [https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html](https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html" \t "_blank)
* [https://levelup.gitconnected.com/create-a-3-tier-architecture-via-aws-cloudformation-b25e450e8acd](https://levelup.gitconnected.com/create-a-3-tier-architecture-via-aws-cloudformation-b25e450e8acd" \t "_blank)
* [https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html" \t "_blank)
* [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-elasticloadbalancingv2-loadbalancer.html](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-elasticloadbalancingv2-loadbalancer.html" \t "_blank)
* [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-sns-topic.html](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-sns-topic.html" \t "_blank)
* [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-subnet.html](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-subnet.html" \t "_blank)
* [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/quickref-rds.html](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/quickref-rds.html" \t "_blank)