Seamless Data Replication:

Integrating GitHub Data with DynamoDB via S3 Bucket Across Regions

TABLE OF CONTENTS

TITLE: Seamless Data Replication: Integrating GitHub Data with DynamoDB via S3 Bucket Across Regions

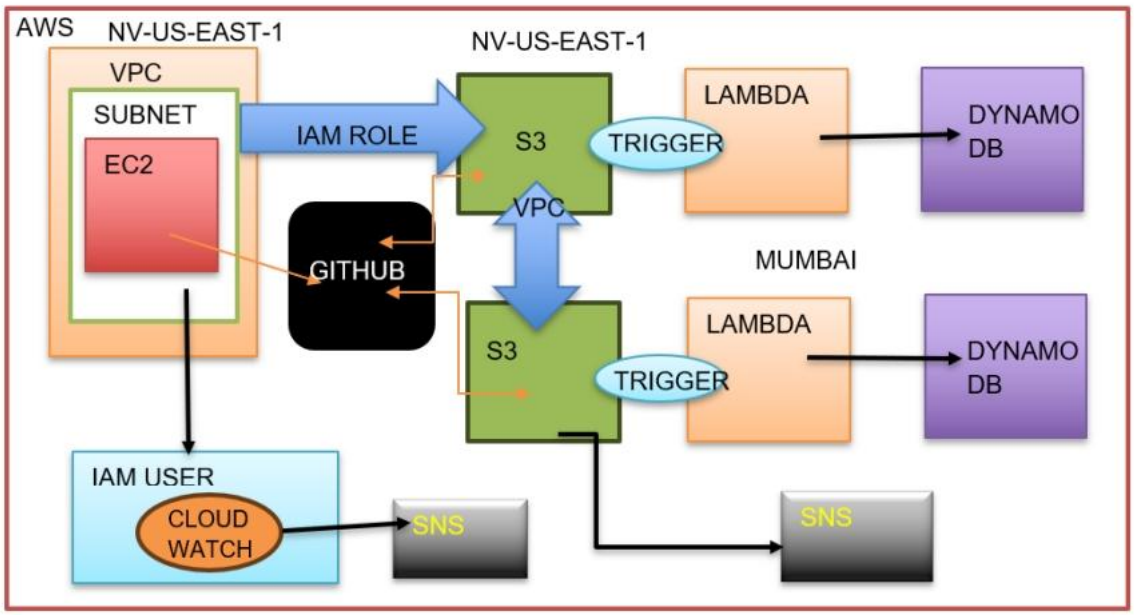
ABSTRACT:

The above architecture is to store the data in the NO SQL format using DYNAMODB. Once the file is uploaded in the S3 bucket of nv-us-east-1 it triggers LAMBDA function and store the data in DYNAMODB as well as S3 bucket in Mumbai-ap-south-1 there also one lambda function triggers and store it in DYNAMODB. By uploading the data, information is send to the user mail through SNS via Email .The uploaded data should be store in GITHUB repository by using EC2 instance in the VPC of nv-us-east-1 with IAM role giving permissions of s3fullacess,dynamodbfullaccess.Through the instance we create a file and try to load it into the S3 bucket. Once the file is uploaded, we usually use this existing data in S3 bucket to load into the database instead of creating and inserting the data into the databases.The data in the repository is also pull into the S3 bucket.The entire process is watched by the CLOUDWATCH which is in another IAM user,it send the email to the subscriber when it reaches the constraints.

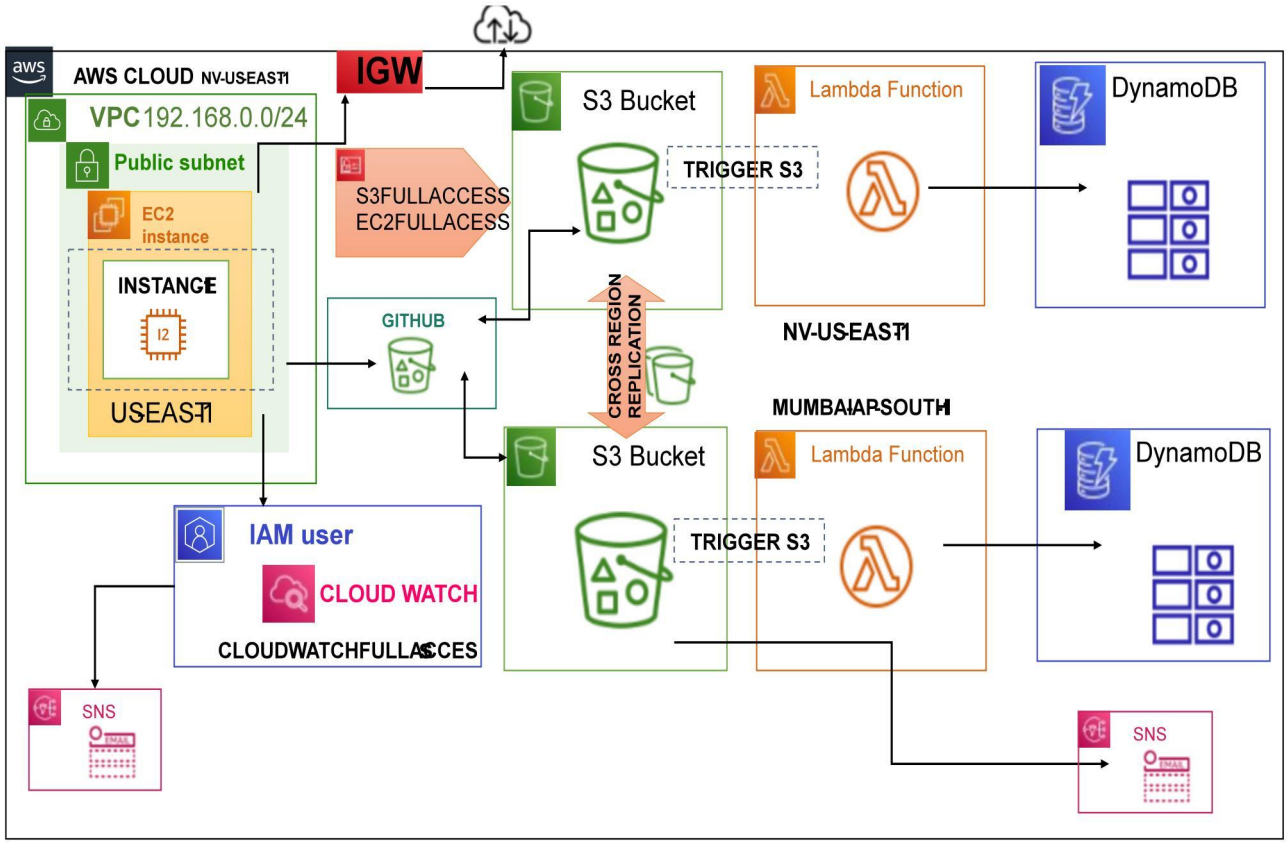
SERVICES USED:

EC2 (Elastic Compute Cloud)
VPC (Virtual Private Cloud)
S3 (Simple Storage Service)
DYNAMODB (No SQL Data Base)
IAM (Identity and Access Management) : USER AND ROLE
AWS Lambda
CLOUD WATCH
SNS (Simple Notification Service)

ROUGH ARCHITECTURE:



FINAL ARCHITECTURE:



CLOUDCOMPUTING

Cloud computing is on-demand access, via the internet, to computing resources—applications, servers (physical servers and virtual servers), data storage, development tools, networking capabilities, and more—hosted at a remote data center managed by a cloud services provider (or CSP). The CSP makes these resources available for a monthly subscription fee or bills them according to usage.

Compared to traditional on-premises IT, and depending on the cloud services you select, cloud computing helps do the following:

- **Lower IT costs:** Cloud lets you offload some or most of the costs and effort of purchasing, installing, configuring, and managing your own on-premises infrastructure.

- **Improve agility and time-to-value:** With cloud, your organization can start using enterprise applications in minutes, instead of waiting weeks or months for IT to respond to a request, purchase and configure supporting hardware, and install software. Cloud also lets you empower certain users— specifically developers and data scientists.

- **Scale more easily and cost-effectively:** Cloud provides elasticity—instead of purchasing excess capacity that sits unused during slow periods, you can scale capacity up and down in response to spikes and dips in traffic. You can also take advantage of your cloud provider's global network to spread your applications closer to users around the world.

The term 'cloud computing' also refers to the technology that makes cloud work. This includes some form of virtualized IT infrastructure—servers, operating system software, networking, and other infrastructure that's abstracted, using special software, so that it can be pooled and divided irrespective of physical hardware boundaries. For example, a single hardware server can be divided into multiple virtual servers.

Cloud Computing Services:

- IaaS (Infrastructure-as-a-Service)
- PaaS (Platform-as-a-Service) SaaS (Software-as-a-service) are the three most common models of cloud services, and it's not uncommon for an organization to use all three.

IaaS (Infrastructure-as-a-Service)

IaaS provides on-demand access to fundamental computing resources—physical and virtual servers, networking, and storage—over the internet on a pay-as-you-go basis. IaaS enables end users to scale and shrink resources on an as-needed basis, reducing the need for high, up-front capital expenditures or unnecessary on-premises or 'owned' infrastructure and for overbuying resources to accommodate periodic spikes in usage.

In contrast to SaaS and PaaS (and even newer PaaS computing models such as containers and serverless), IaaS provides the users with the lowest-level control of computing resources in the cloud.

IaaS was the most popular cloud computing model when it emerged in the early 2010s. While it remains the cloud model for many types of workloads, use of SaaS and PaaS is growing at a much faster rate.

PaaS (Platform-as-a-service)

PaaS provides software developers with on-demand platform—hardware, complete software stack, infrastructure, and even development tools—for running, developing, and managing applications without the cost, complexity, and inflexibility of maintaining that platform on-premises.

With PaaS, the cloud provider hosts everything—servers, networks, storage, operating system software, middleware, databases—at their data center. Developers simply pick from a menu to 'spin up' servers and environments they need to run, build, test, deploy, maintain, update, and scale applications.

Today, PaaS is often built around containers, a virtualized compute model one step removed from virtual servers. containers virtualize the operating system, enabling developers to package the application with only the operating system services it needs to run on any platform, without modification and without need for middleware.

SaaS (Software-as-a-Service)

SaaS—also known as cloud-based software or cloud applications—is application software that's hosted in the cloud, and that user's access via a web browser, a dedicated desktop client, or an API that integrates with a desktop or mobile operating system. In most cases, SaaS users pay a monthly or annual subscription fee; some may offer 'pay-as-you-go' pricing based on your actual usage.

In addition to the cost savings, time-to-value, and scalability benefits of cloud, SaaS offers the following:

- **Automatic upgrades:** With SaaS, users take advantage of new features as soon as the provider adds them, without having to orchestrate an on-premises upgrade.

- **Protection from data loss:** Because SaaS stores application data in the cloud with the application, users don't lose data if their device crashes or breaks.

SaaS is the primary delivery model for most commercial software today—there are hundreds of thousands of SaaS solutions available, from the most focused industry and departmental applications to powerful enterprise software database and AI (artificial intelligence) software.

Cloud Service Providers:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- Oracle
- IBM cloud Salesforce

AMAZON WEB SERVICES:

Amazon Web Services, Inc. (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Oftentimes, clients will use this in combination with autoscaling (a process that allows a client to use more computing in times of high application usage, and then scale down to reduce costs when there is less traffic). These cloud computing web services provide various services related to networking, computing, storage, middleware, IoT and other processing capacity, as well as software tools via AWS server farms. This frees clients from managing, scaling, and patching hardware, and operating systems.

One of the foundational services is Amazon Elastic Compute Cloud (EC2), which allows users to have at their disposal a virtual cluster of computers, with extremely high availability, which can be interacted with over the internet via REST APIs, a CLI or the AWS console. AWS's virtual computers emulate most of the attributes of a real computer, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing; local/RAM memory; hard disk /SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).

AWS services are delivered to customers via a network of AWS server farms located throughout the world. Fees are based on a combination of usage (known as a "Pay-as-you-go" model), hardware, operating system, software, or networking features chosen by the subscriber required availability, redundancy, security, and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either.

Amazon provides select portions of security for subscribers (e.g., physical security of the data centers) while other aspects of security are the responsibility of the subscriber (e.g., account management, vulnerability scanning, patching). AWS operates for many global geographical regions including seven in North America.

Amazon markets AWS to subscribers as a way of obtaining large-scale computing capacity more quickly and cheaply than building an actual physical server farm. All services are billed based on usage, but each service measures usage in varying ways. As of 2021 Q4, AWS has 33% market share for cloud infrastructure while the next two competitors Microsoft Azure and Google Cloud have 21%, and 10% respectively, according to Synergy Group.

Why AWS?

- **Easy to use:**

  AWS is designed to allow application providers, ISVs, and vendors to host your applications

quickly and securely – whether an existing application or a new SaaS-based application.
You can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform**.**

- **Flexible:**
  AWS enables you to select the operating system, programming language, web application platform, database, and other services you need. With AWS, you receive a virtual environment that lets you load the software and services your application requires. This eases the migration process for existing applications while preserving options for building new solutions.
- **Cost-effective:**
  You pay only for the compute power, storage, and other resources you use, with no long-term contracts or up-front commitments. For more information on comparing the costs of other hosting alternatives with AWS, see the AWS Economics Center.
- **Reliable:**
  With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure, the virtual backbone of Amazon.com's multi-billion-dollar online business that has been honed for over a decade**.**
- **Scalable and High performance:**
  Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.
- **Secure:**
  Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

List of AWS Services:

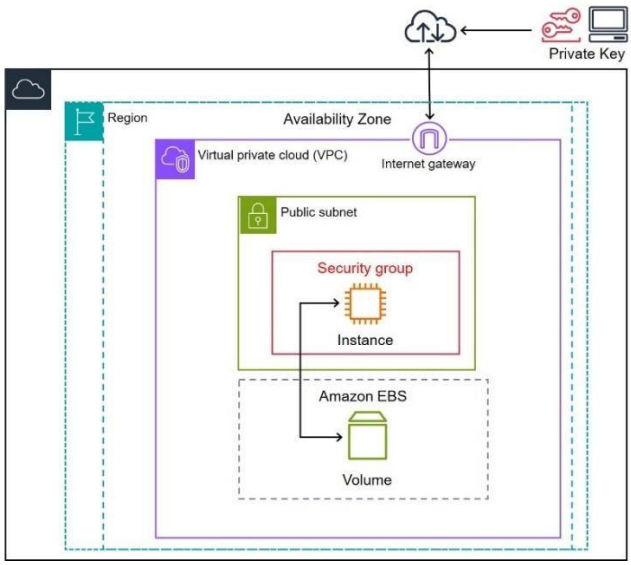
Amazon, the preeminent cloud vendor, broke new ground by establishing the first cloud computing service, Amazon EC2, in 2008. AWS offers more solutions and features than any other provider and has free tiers with access to the AWS Console, where users can centrally control their ministrations.
Designed around ease-of-use for various skill sets, AWS is tailored for those unaccustomed to software development utilities. Web applications can be deployed in minutes with AWS facilities, without provisioning servers or writing additional code.

- Amazon EC2 (Elastic Compute Cloud)
- Amazon RDS (Relational Database Services)
- Amazon S3 (Simple Storage Service)
- Amazon Lambda
- Amazon Cognito
- Amazon Glacier
- Amazon SNS (Simple Notification Service)
- Amazon VPC (Virtual Private Cloud)

- Amazon Lightsail
- Amazon CloudWatch
- Amazon Cloud9
- Amazon Elastic Beanstalk
- Amazon CodeCommit
- Amazon IAM (Identity and Access Management)
- Amazon Inspector
- Amazon Kinesis
- Amazon Dynamo DB
- Amazon Codecatalyst
- Amazon Kinesis
- AWS Athena
- AWS Amplify
- AWS Quicksight
- AWS Cloudformation

Amazon EC2:

Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. In November 2010, Amazon switched its own retail website platform to EC2 and AWS.



Amazon announced a limited public beta test of EC2 on August 25, 2006, offering access on a first-come, first-served basis. Amazon added two new instance types (Large and Extra-Large) on October 16, 2007. On May 29, 2008, two more types were added, High-CPU Medium and High-CPU Extra Large. There were twelve types of instances available.

Amazon added three new features on March 27, 2008, static IP addresses, availability zones, and user selectable kernels. On August 20, 2008, Amazon added Elastic Block Store (EBS) This provides persistent storage, a feature that had been lacking since the service was introduced.

Instance types:

Initially, EC2 used Xen virtualization exclusively. However, on November 6, 2017, Amazon announced the new C5 family of instances that were based on a custom architecture around the KVM hypervisor, called Nitro. Each virtual machine, called an "instance", functions as a virtual private server. Amazon sizes instances based on "Elastic Compute Units". The performance of otherwise identical virtual machines may vary. On November 28, 2017, AWS announced a bare-metal instance type offering marking a remarkable departure from exclusively offering virtualized instance types. As of January 2019, the following instance types were offered:

- General Purpose: A1, T3, T2, M5, M5a, M4, T3a
- Compute Optimized: C5, C5n, C4
- Memory Optimized: R5, R5a, R4, X1e, X1, High Memory, z1d
- Accelerated Computing: P3, P2, G3, F1
- Storage Optimized: H1, I3, D2

As of April 2018, the following payment methods by instance were offered:

- On-demand: pay by the hour without commitment.
- Reserved: rent instances with one-time payment receiving discounts on the hourly charge.
- Spot: bid-based service runs the jobs only if the spot price is below the bid specified by bidder. The spot price is claimed to be supply-demand based, however a 2011 study concluded that the price was generally not set to clear the market but was dominated by an undisclosed reserve price.

Amazon RDS:

Amazon Relational Database Service (or **Amazon RDS**) is a distributed relational database service by Amazon Web Services (AWS). It is a web service running "in the cloud" designed to simplify the setup, operation, and scaling of a relational database for use in applications. Administration processes like patching the database software, backing up databases and enabling point-in-time recovery are managed automatically. Scaling storage and compute resources can be performed by a single API call to the AWS control plane ondemand. AWS does not offer an SSH connection to the underlying virtual machine as part of the managed service.

Multiple Availability Zone (AZ) Deployment

In May 2010 Amazon announced Multi-Availability Zone deployment support. Amazon RDS MultiAvailability Zone (AZ) allows users to automatically provision and maintain a synchronous physical or logical "standby" replica, depending on database engine, in a different Availability Zone (independent infrastructure in a physically separate location). Multi-AZ database instance can be developed at creation time or modified to run as a multi-AZ deployment later. Multi-AZ deployments aim to provide enhanced

availability and data durability for MySQL, MariaDB, Oracle, PostgreSQL and SQL Server instances and are targeted for production environments. In the event of planned database maintenance or unplanned service disruption, Amazon RDS automatically fails over to the up-to-date standby, allowing database operations to resume without administrative intervention.

Multi-AZ RDS instances are optional and have a cost associated with them. When creating a RDS instance, the user is asked if they would like to use a multi-AZ RDS instance. In Multi-AZ RDS deployments backups are done in the standby instance so I/O activity is not suspended any time, but users may experience elevated latencies for a few minutes during backups.

Read replicas.

Read replicas allow different use cases such as scale in for read-heavy database workloads. There are up to five replicas available for MySQL, MariaDB, and PostgreSQL. Instances use the native, asynchronous replication functionality of their respective database engines. They have no backups configured by default and are accessible and can be used for read scaling. MySQL and MariaDB read replicas and can be made writeable again since October 2012; PostgreSQL read replicas do not support it. Replicas are done at database instance level and do not support replication at database or table level.

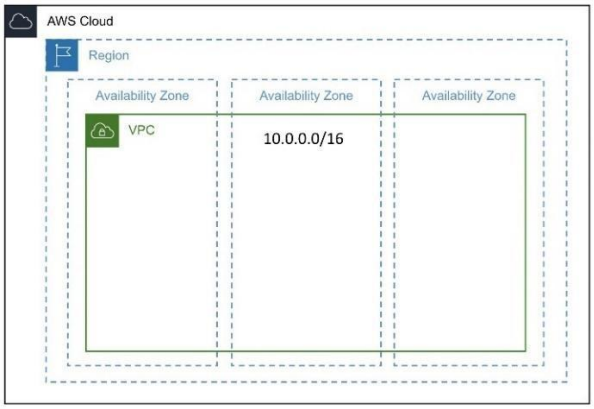
Performance metrics and monitoring

Performance metrics for Amazon RDS are available from the AWS Management Console or the Amazon CloudWatch API. In December 2015, Amazon announced an optional enhanced monitoring feature that provides an expanded set of metrics for the MySQL, MariaDB, and Aurora database engines.

Amazon VPC:

Amazon Virtual Private Cloud (VPC) is a commercial cloud computing service that provides a virtual private cloud, by provisioning a logically isolated section of Amazon Web Services (AWS) Cloud. Enterprise customers are able to access the Amazon Elastic Compute Cloud (EC2) over an IPsec based virtual private network. Unlike traditional EC2 instances which are allocated internal and external IP numbers by Amazon, the customer can assign IP numbers of their choosing from one or more subnets.

Use the following procedure to create a VPC with the subnets, gateways, and routing configuration that you need.



Amazon Web Services launched Amazon Virtual Private Cloud on 26 August 2009, which allows the Amazon Elastic Compute Cloud service to be connected to legacy infrastructure over an IPsec VPN.In AWS, the basic VPC is free to use, with users being charged by usage for additional features.EC2 and RDS instances running in a VPC can also be purchased using Reserved Instances, however will have a limitation

on resources being guaranteed.[citation needed]

IBM Cloud launched IBM Cloud VPC on 4 June 2019, provides an ability to manage virtual machine-based compute, storage, and networking resources. Pricing for IBM Cloud Virtual Private Cloud is applied separately for internet data transfer, virtual server instances, and block storage used within IBM Cloud VPC.

Google Cloud Platform resources can be provisioned, connected, and isolated in a virtual private cloud (VPC) across all GCP regions. With GCP, VPCs are global resources and subnets within that VPC are regional resources. This allows users to connect zones and regions without the use of additional networking complexity as all data travels, encrypted in transit and at rest, on Google's own global, private network. Identity management policies and security rules allow for private access to Google's storage, big data, and analytics managed services. VPCs on Google Cloud Platform leverage the security of Google's data centers.

Amazon S3:

Amazon S3 manages data with an object storage architecture which aims to provide scalability, high availability, and low latency with high durability. The basic storage units of Amazon S3 are objects which are organized into buckets. Each object is identified by a unique, user-assigned key. Buckets can be managed using the console provided by Amazon S3, programmatically with the AWS SDK, or the REST application programming interface.

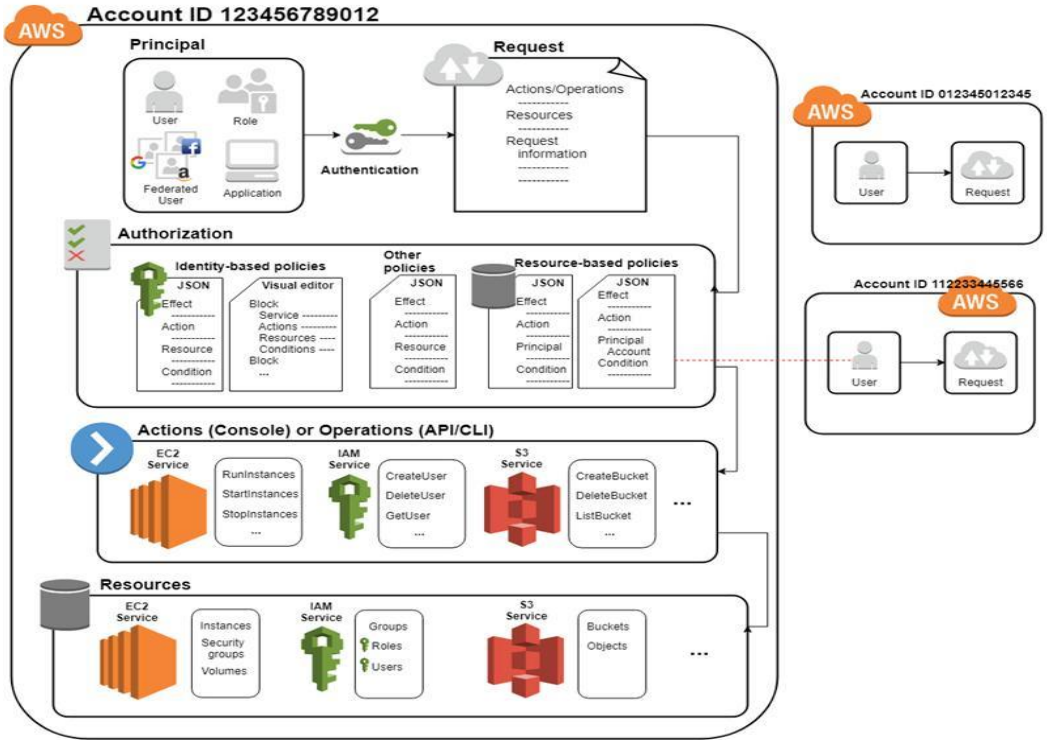
Objects can be up to five terabytes in size. Requests are authorized using an access control list associated with each object bucket and support versioning which is disabled by default. Since buckets are typically the size of an entire file system mount in other systems, this access control scheme is very coarse-grained. In other words, unique access controls cannot be associated with individual files. [citation needed] Amazon S3 can be used to replace static web-hosting infrastructure with HTTP client-accessible objects, index document support and error document support.

The Amazon AWS authentication mechanism allows the creation of authenticated URLs, valid for a specified amount of time. Every item in a bucket can also be served as a BitTorrent feed. The Amazon S3 store can act as a seed host for a torrent and any BitTorrent client can retrieve the file. This can drastically reduce the bandwidth cost for the download of popular objects. A bucket can be configured to save HTTP log information to a sibling bucket; this can be used in data mining operations.

There are various User Mode File System (FUSE)–based file systems for Unix-like operating systems (for example, Linux) that can be used to mount an S3 bucket as a file system. The semantics of the Amazon S3 file system is not that of a POSIX file system, so the file system may not behave entirely as expected.

Amazon IAM:

IAM provides the infrastructure necessary to control authentication and authorization for your AWS account. The IAM infrastructure is illustrated by the following diagram.



First, a human user or an application uses their sign-in credentials to authenticate with AWS. Authentication is provided by matching the sign-in credentials to a principal (an IAM user, federated user, IAM role, or application) trusted by the AWS account.

Next, a request is made to grant the principal access to resources. Access is granted in response to an authorization request. For example, when you first sign into the console and are on the console home page, you are not accessing a specific service. When you select a service, the request for authorization is sent to that service and it looks to see if your identity is on the list of authorized users, what policies are being enforced to control the level of access granted, and any other policies that might be in effect. Authorization requests can be made by principals within your AWS account or from another AWS account that you trust.

Once authorized, the principal can take action or perform operations on resources in your AWS account. For example, the principal could launch a new Amazon Elastic Compute Cloud instance, modify IAM group membership, or delete Amazon Simple Storage Service buckets.

The previous illustration we used specific terminology to describe how to obtain access to resources. These IAM terms are commonly used when working with AWS:

- IAM Resources
    - The user, group, role, policy, and identity provider objects that are stored in IAM. As with other AWS services, you can add, edit, and remove resources from IAM.
- IAM Identities ○ The IAM resource objects that are used to identify and group. You can attach a policy to an IAM identity. These include users, groups, and roles.
- IAM Entities ○ The IAM resource objects that AWS uses for authentication. These include IAM users and roles.
- Principals ○ A person or application that uses the AWS account root user, an IAM user, or an IAM role to sign in and make requests to AWS. Principals include federated users and assumed roles.
- Human users
    - Also known as human identities; the people, administrators, developers, operators, and consumers of your applications.
- Workload
    - A collection of resources and code that delivers business value, such as an application or backend process. Can include applications, operational tools, and components.

AWS Lambda:

AWS Lambda is a compute service that lets you run code without provisioning or managing servers.

Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, all you need to do is supply your code in one of the language runtimes that Lambda supports.

You organize your code into Lambda functions. The Lambda service runs your function only when needed and scales automatically. You only pay for the compute time that you consume—there is no charge when your code is not running.

When using Lambda, you are responsible only for your code. Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources to run your code. Because Lambda manages these resources, you cannot log in to compute instances or customize the operating system on provided runtimes.

Lambda performs operational and administrative activities on your behalf, including managing capacity, monitoring, and logging your Lambda functions.

If you do need to manage your compute resources, AWS has other compute services to consider, such as:

- AWS App Runner builds and deploys containerized web applications automatically, load balances traffic with encryption, scales to meet your traffic needs, and allows for the configuration of how services are accessed and communicate with other AWS applications in a private Amazon VPC.
- AWS Fargate with Amazon ECS runs containers without having to provision, configure, or scale clusters of virtual machines.
- Amazon EC2 lets you customize operating system, network and security settings, and the entire software stack. You are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.

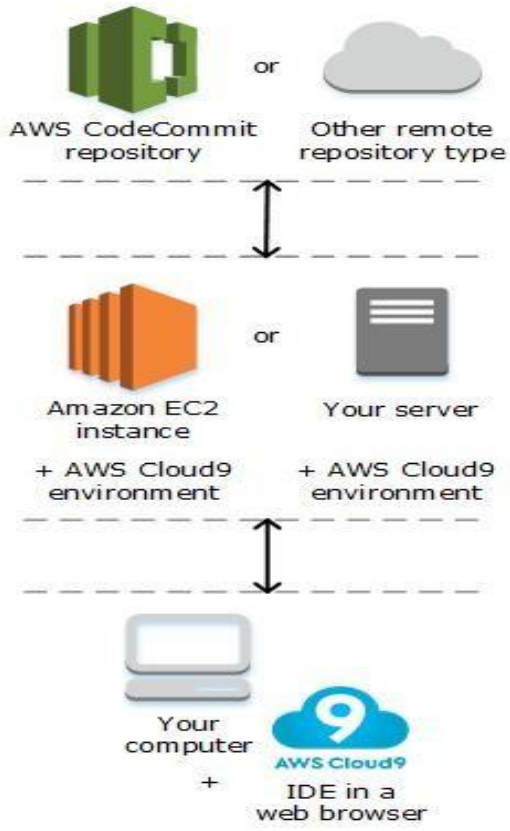
You can use environment variables to adjust your function's behavior without updating code. An environment variable is a pair of strings that is stored in a function's version-specific configuration. The Lambda runtime makes environment variables available to your code and sets additional environment variables that contain information about the function and invocation request.

AWS Cloud9:

AWS Cloud9 is an integrated development environment, or *IDE*.

The AWS Cloud9 IDE offers a rich code-editing experience with support for several programming languages and runtime debuggers, and a built-in terminal. It contains a collection of tools that you use to code, build, run, test, and debug software, and helps you release software to the cloud.

You access the AWS Cloud9 IDE through a web browser. You can configure the IDE to your preferences. You can switch color themes, bind shortcut keys, enable programming language-specific syntax coloring and code formatting, and more.



Environments and computing resources

Behind the scenes, there are a couple of ways you can connect your environments to computing resources:

- You can instruct AWS Cloud9 to create an Amazon EC2 instance, and then connect the environment to that newly created EC2 instance. This type of setup is called an EC2 environment.
- You can instruct AWS Cloud9 to connect an environment to an existing cloud compute instance or to your own server. This type of setup is called an SSH environment.

EC2 environments and SSH environments have some similarities and some differences. If you're new to AWS Cloud9, we recommend that you use an EC2 environment because AWS Cloud9 takes care of much of the configuration for you. As you learn more about AWS Cloud9, and want to understand these similarities and differences better, see EC2 environments compared with SSH environments in AWS Cloud9.

AWS Elastic BeanStalk:

Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them.

With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

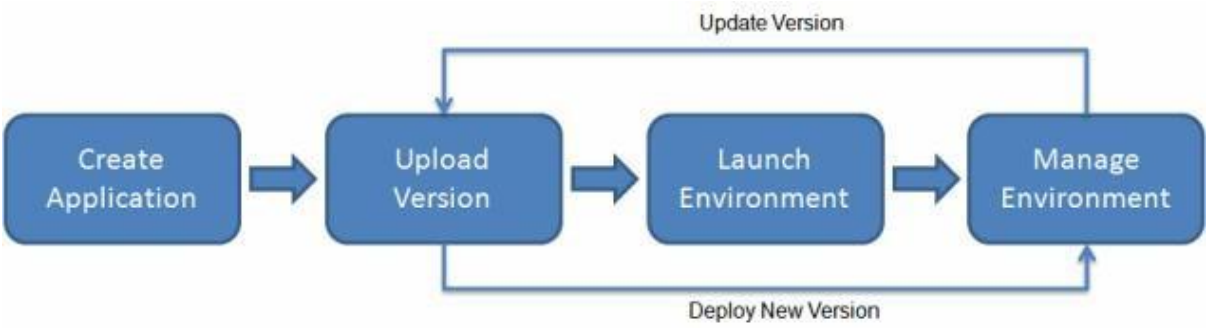
Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.

You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or **eb**, a high-level CLI designed specifically for Elastic Beanstalk.

To learn more about how to deploy a sample web application using Elastic Beanstalk, see Getting Started with AWS: Deploying a Web App.

You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the Elastic Beanstalk web interface (console).

To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions. The following diagram illustrates the workflow of Elastic Beanstalk.



AWS CodeCommit:

CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.

With CodeCommit, you can:

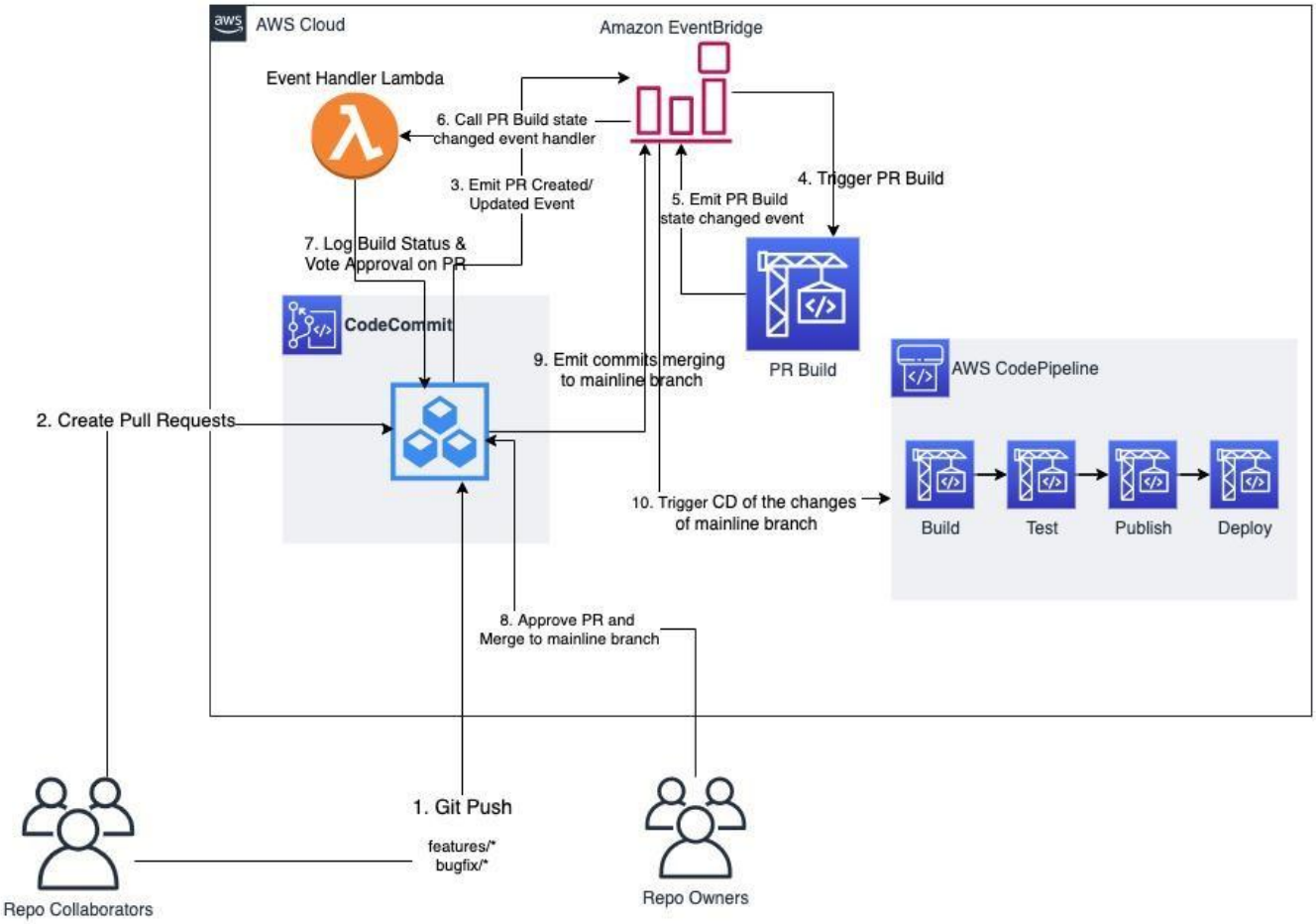
- **Benefit from a fully managed service hosted by AWS**. CodeCommit provides high service availability and durability and eliminates the administrative overhead of managing your own hardware and software. There is no hardware to provision and scale and no server software to install, configure, and update.
- **Store your code securely**. CodeCommit repositories are encrypted at rest as well as in transit.
- **Work collaboratively on code.** CodeCommit repositories support pull requests, where users can review and comment on each other's code changes before merging them to branches; notifications that automatically send emails to users about pull requests and comments; and more.
- **Easily scale your version control projects**. CodeCommit repositories can scale up to meet your

development needs. The service can handle repositories with large numbers of files or branches, large file sizes, and lengthy revision histories.

- **Store anything, anytime**. CodeCommit has no limit on the size of your repositories or on the file types you can store.
- **Integrate with other AWS and third-party services**. CodeCommit keeps your repositories close to your other production resources in the AWS Cloud, which helps increase the speed and frequency of your development lifecycle. It is integrated with IAM and can be used with other AWS services and in parallel with other repositories. For more information, see Product and service integrations with AWS CodeCommit.

Easily migrate files from other remote repositories. You can migrate to CodeCommit from any Git-based repository.

Use the Git tools you already know. CodeCommit supports Git commands as well as its own AWS CLI commands and APIs.



Amazon CloudWatch:

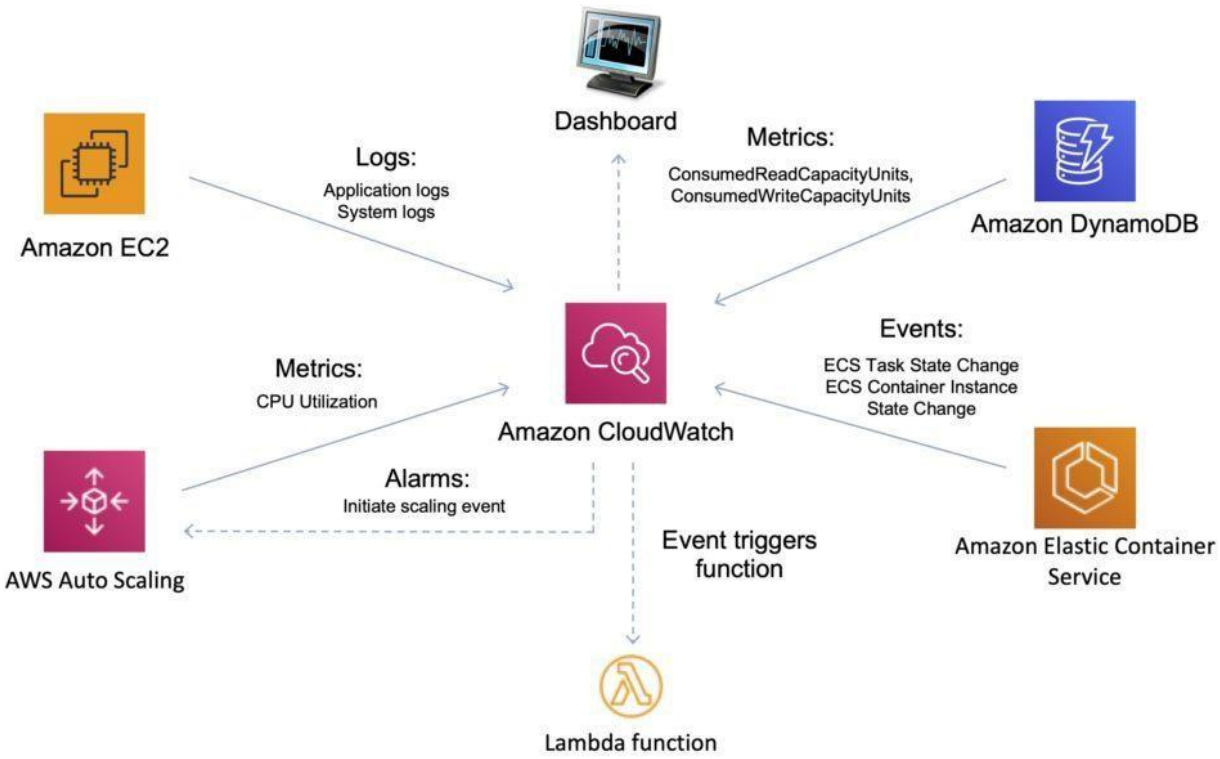
Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the

resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use that data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop underused instances to save money.

With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.



Amazon EBS (Elastic Block Store):

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. You can create a file system on top of these volumes or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

We recommend Amazon EBS for data that must be quickly accessible and requires long-term persistence. EBS volumes are particularly well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage. Amazon EBS is well suited to both database-style applications that rely on random reads and writes, and to throughput-intensive applications that perform long, continuous reads and writes.

With Amazon EBS, you pay only for what you use. For more information about Amazon EBS pricing, see the Projecting Costs Section of the Amazon Elastic Block Store page.

Features of Amazon EBS

- You create an EBS volume in a specific Availability Zone, and then attach it to an instance in that same Availability Zone. To make a volume available outside of the Availability Zone, you can create a snapshot and restore that snapshot to a new volume anywhere in that Region. You can copy snapshots to other Regions and then restore them to new volumes there, making it easier to leverage multiple AWS Regions for geographical expansion, data center migration, and disaster recovery.

- Amazon EBS provides the following volume types: General Purpose SSD, Provisioned IOPS SSD, Throughput Optimized HDD, and Cold HDD. For more information, see EBS volume types.
  The following is a summary of performance and use cases for each volume type.
  - General Purpose SSD volumes (gp2 and gp3) balance price and performance for a wide variety of transactional workloads. These volumes are ideal for use cases such as boot volumes, medium-size single instance databases, and development and test environments.
  - Provisioned IOPS SSD volumes (io1 and io2) are designed to meet the needs of I/O-intensive workloads that are sensitive to storage performance and consistency. They provide a consistent IOPS rate that you specify when you create the volume. This enables you to predictably scale to tens of thousands of IOPS per instance. Additionally, io2 volumes provide the highest levels of volume durability.
  - Throughput Optimized HDD volumes (st1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential workloads such as Amazon EMR, ETL, data warehouses, and log processing.
  - Cold HDD volumes (sc1) provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. These volumes are ideal for large, sequential, cold-data workloads. If you require infrequent access to your data and are looking to save costs, these volumes provide inexpensive block storage.

- You can create your EBS volumes as encrypted volumes, in order to meet a wide range of data-atrest encryption requirements for regulated/audited data and applications. When you create an encrypted EBS volume and attach it to a supported instance type, data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. Encryption occurs on the servers that host EC2 instances, providing encryption of data-in-transit from EC2 instances to EBS storage.
  For more information, see Amazon EBS encryption.

- Performance metrics, such as bandwidth, throughput, latency, and average queue length, are available through the AWS Management Console. These metrics, provided by Amazon CloudWatch, allow you to monitor the performance of your volumes to make sure that you are providing enough performance for your applications without paying for resources you don't need.

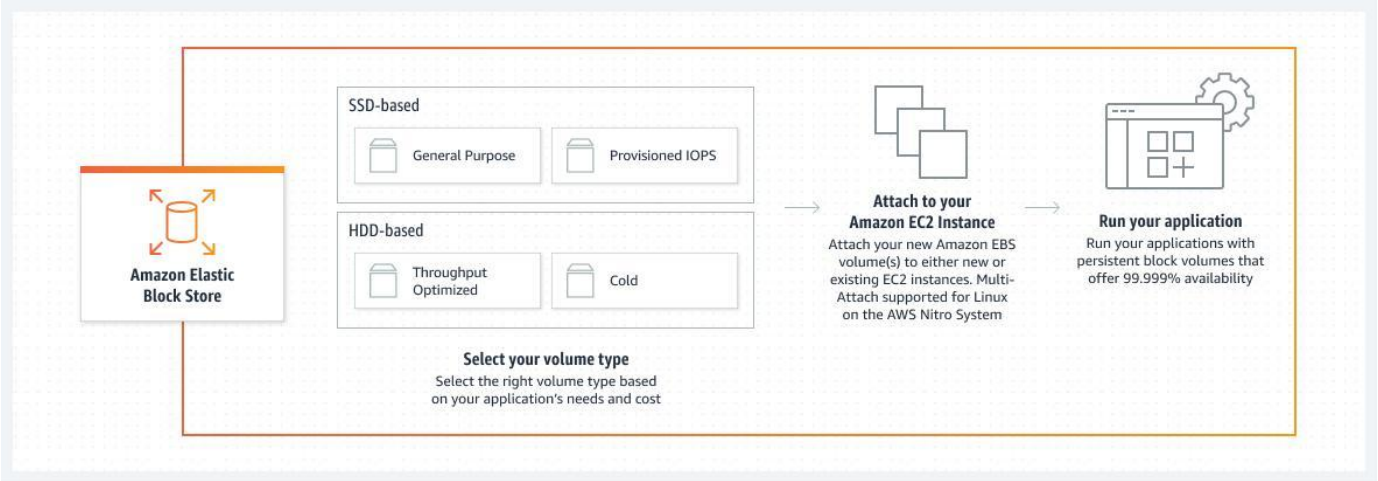


Fig. High-Performance Block Storage

Amazon Aurora:

Amazon Aurora (Aurora) is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. You already know how MySQL and PostgreSQL combine the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. The code, tools, and applications you use today with your existing MySQL and PostgreSQL databases can be used with Aurora. With some workloads, Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications.

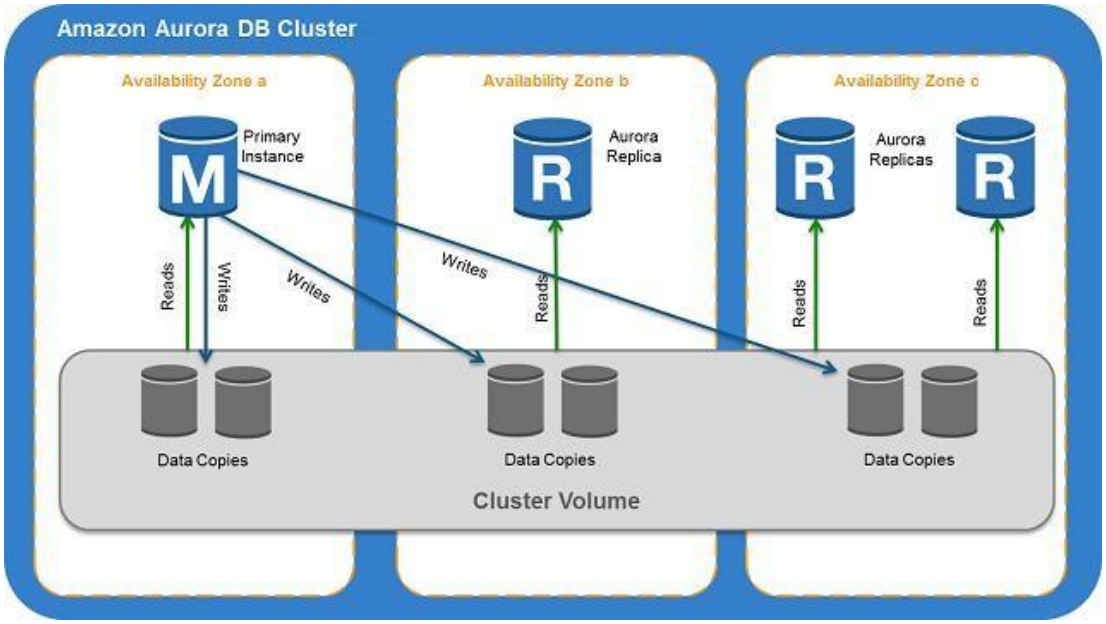
Aurora includes a high-performance storage subsystem. Its MySQL- and PostgreSQL-compatible database engines are customized to take advantage of that fast distributed storage. The underlying storage grows automatically as needed. An Aurora cluster volume can grow to a maximum size of 128 tebibytes (TiB). Aurora also automates and standardizes database clustering and replication, which are typically among the most challenging aspects of database configuration and administration.

Aurora is part of the managed database service Amazon Relational Database Service (Amazon RDS). Amazon RDS is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. If you are not already familiar with Amazon RDS, see the Amazon Relational Database Service User Guide.

The following points illustrate how Amazon Aurora relates to the standard MySQL and PostgreSQL engines available in Amazon RDS:

- You choose Aurora MySQL or Aurora PostgreSQL as the DB engine option when setting up new database servers through Amazon RDS.
- Aurora takes advantage of the familiar Amazon Relational Database Service (Amazon RDS) features for management and administration. Aurora uses the Amazon RDS AWS Management Console interface, AWS CLI commands, and API operations to handle routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.
- Aurora management operations typically involve entire clusters of database servers that are synchronized through replication, instead of individual database instances. The automatic clustering, replication, and storage allocation make it simple and cost-effective to set up, operate, and scale your largest MySQL and PostgreSQL deployments.

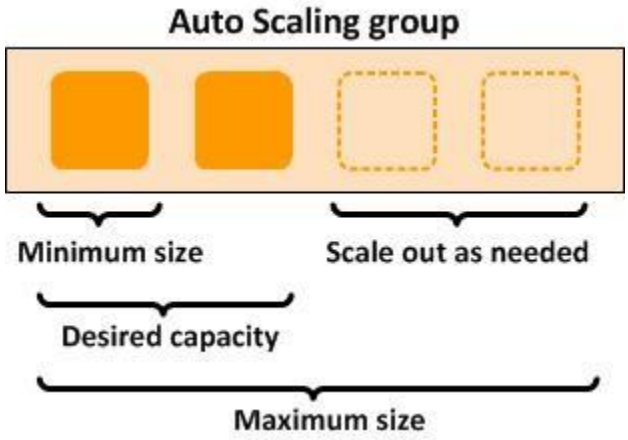
- You can bring data from Amazon RDS for MySQL and Amazon RDS for PostgreSQL into Aurora by creating and restoring snapshots, or by setting up one-way replication. You can use push-button migration tools to convert your existing RDS for MySQL and RDS for PostgreSQL applications to Aurora.



AWS Autoscaling:

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of one instance, a desired capacity of two instances, and a maximum size of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.

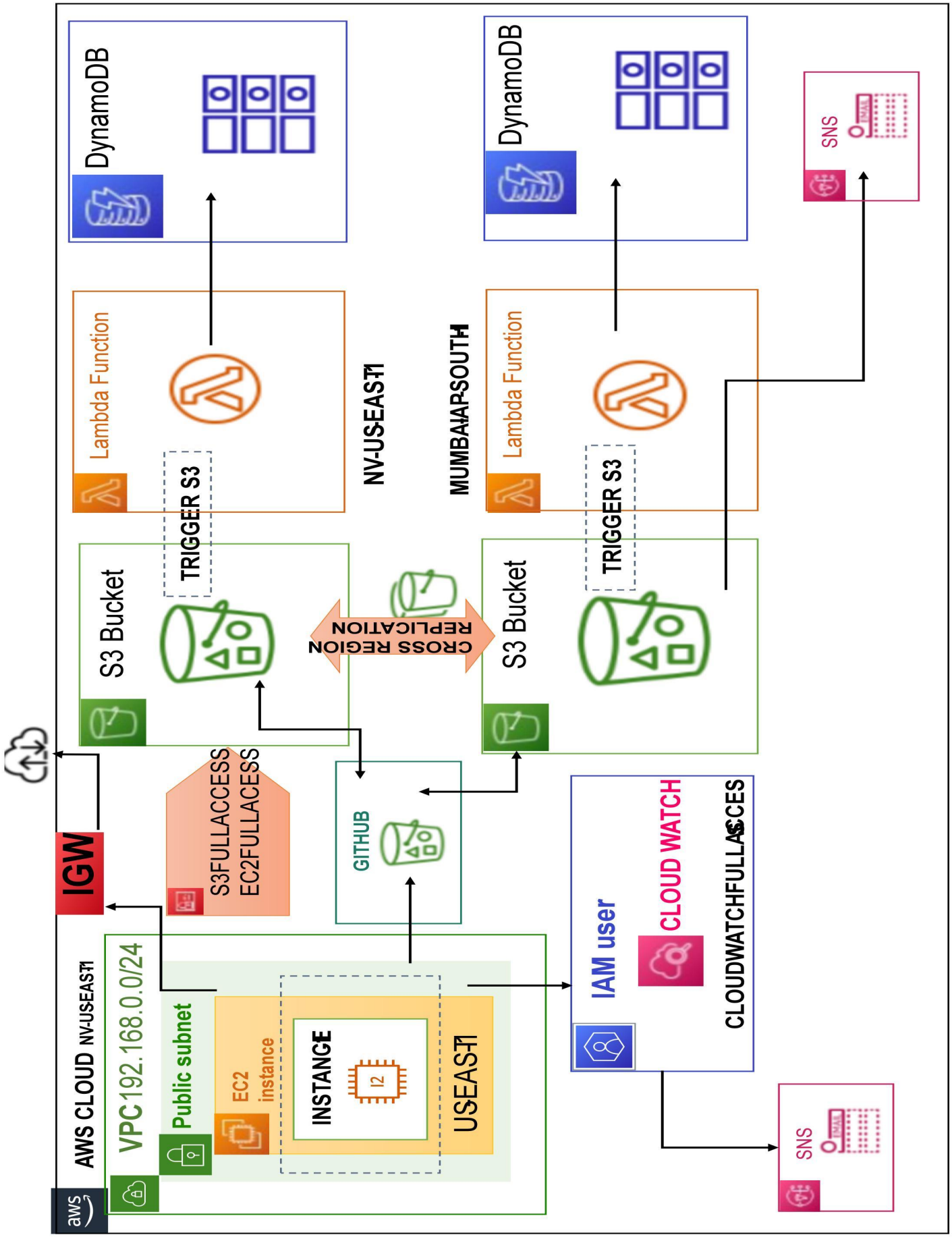


Auto scaling benefits

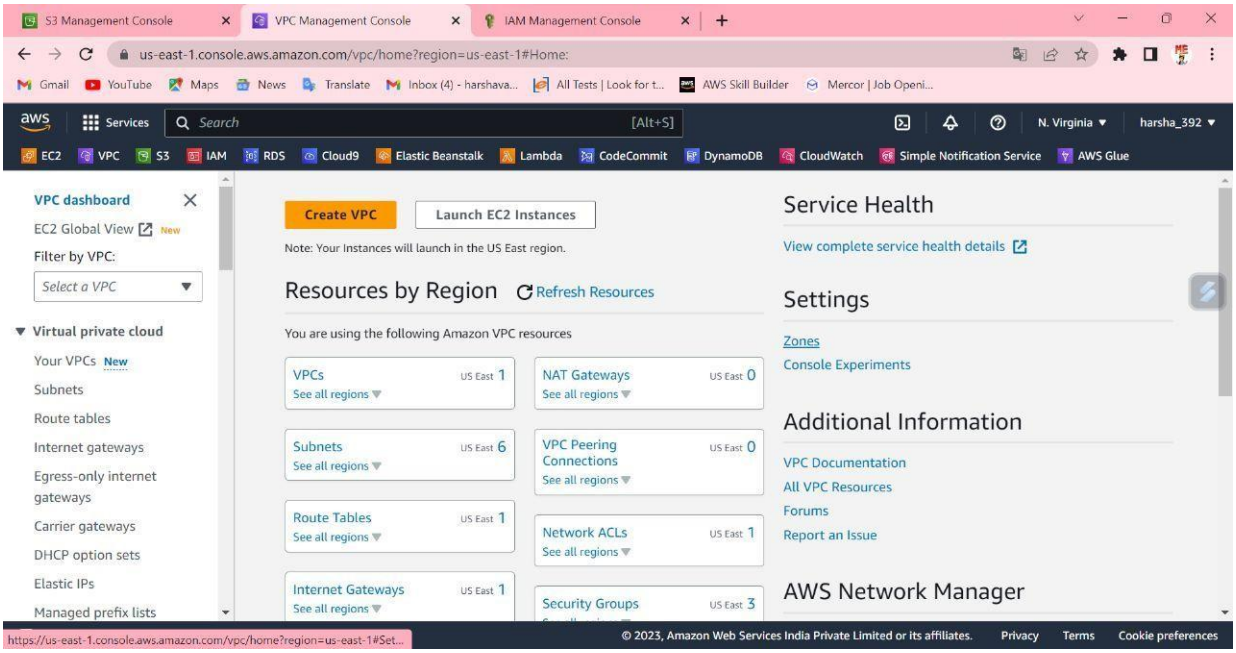
Adding Amazon EC2 Auto Scaling to your application architecture is one way to maximize the benefits of the AWS Cloud. When you use Amazon EC2 Auto Scaling, your applications gain the following benefits:

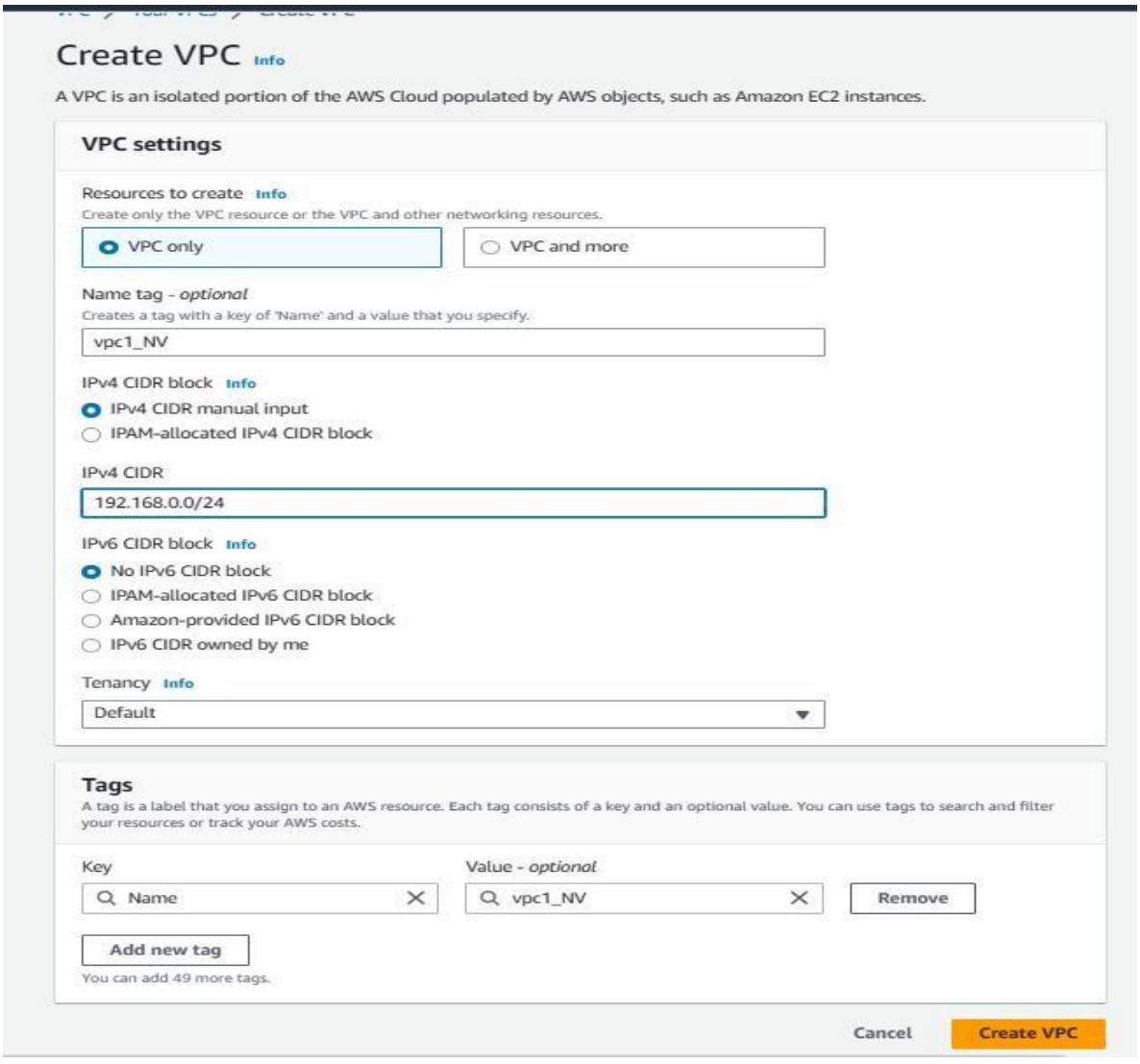
- Better fault tolerance. Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Amazon EC2 Auto Scaling can launch instances in another one to compensate.
- Better availability. Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.
- Better cost management. Amazon EC2 Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are needed and terminating them when they aren't.

IMPLEMENTATION OF MY ARCHITECTURE

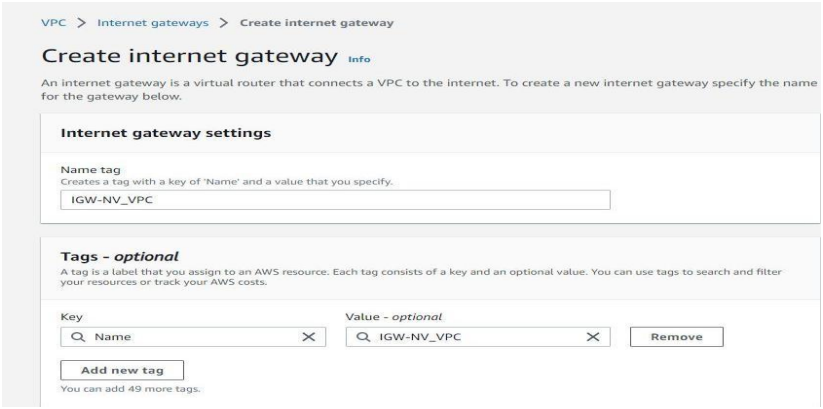


CREATE A VPC:

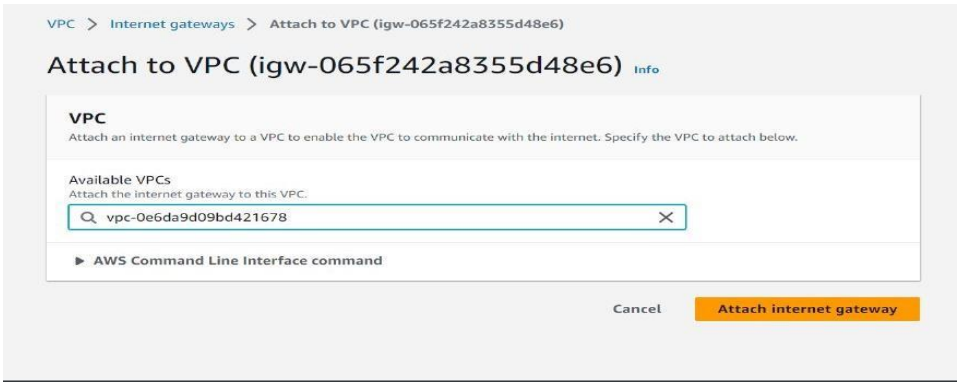

S3 Management Console
VPC Management Console
IAM Management Console
us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#Home:
Services
Search
[Alt+S]
N. Virginia
harsha_392
EC2 VPC S3 IAM RDS Cloud9 Elastic Beanstalk Lambda CodeCommit DynamoDB CloudWatch Simple Notification Service AWS Glue
VPC dashboard
EC2 Global View
Filter by VPC:
Select a VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Create VPC
Launch EC2 Instances
Note: Your Instances will launch in the US East region.
Resources by Region
Refresh Resources
You are using the following Amazon VPC resources
VPCs US East 1
NAT Gateways US East 0
Subnets US East 6
VPC Peering Connections US East 0
Route Tables US East 1
Network ACLs US East 1
Internet Gateways US East 1
Security Groups US East 3
Service Health
View complete service health details
Settings
Zones
Console Experiments
Additional Information
VPC Documentation
All VPC Resources
Forums
Report an Issue
AWS Network Manager
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences


Create VPC Info
A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.
VPC settings
Resources to create Info
Create only the VPC resource or the VPC and other networking resources.
VPC only
VPC and more
Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.
vpc1_NV
IPv4 CIDR block Info
IPv4 CIDR manual input
IPAM-allocated IPv4 CIDR block
IPv4 CIDR
192.168.0.0/24
IPv6 CIDR block Info
No IPv6 CIDR block
IPAM-allocated IPv6 CIDR block
Amazon-provided IPv6 CIDR block
IPv6 CIDR owned by me
Tenancy Info
Default
Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
Key
Name
Value - optional
vpc1_NV
Remove
Add new tag
You can add 49 more tags.
Cancel
Create VPC

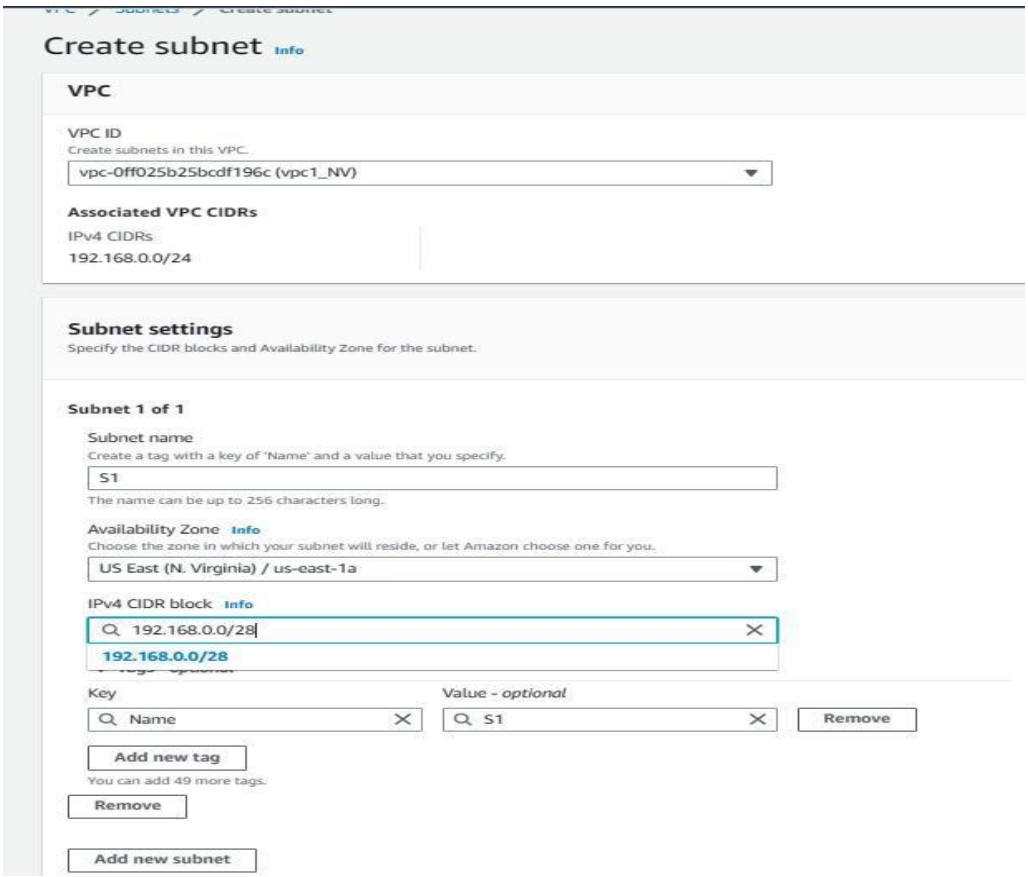
CREATE INTERNET GATEWAY:



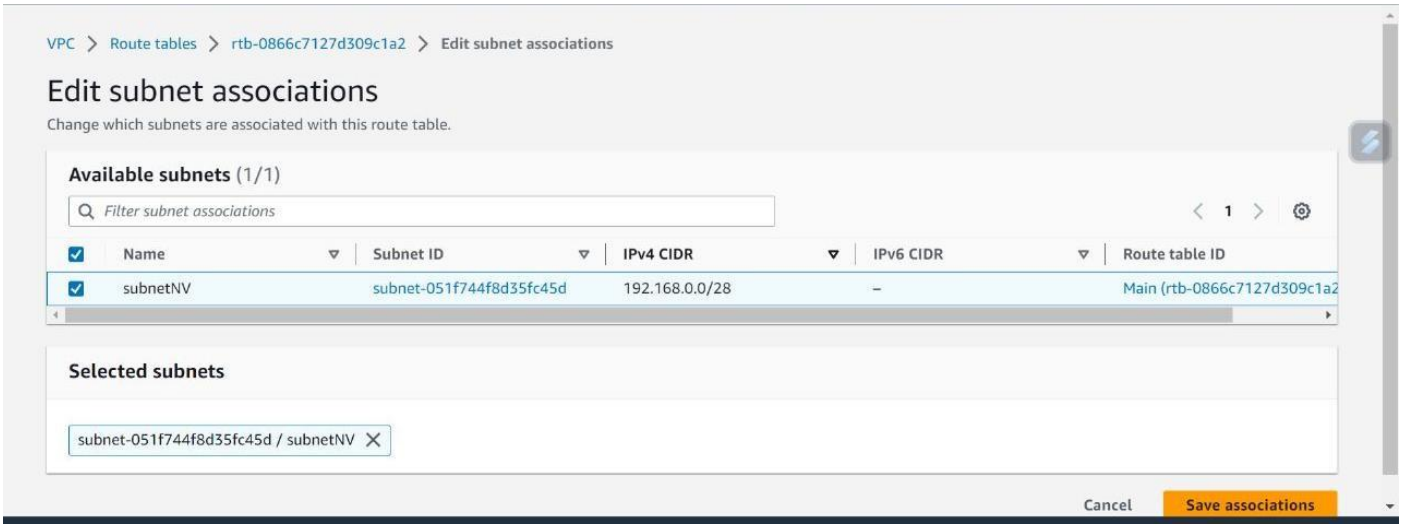
ATTACH INTERNET GATEWAY TO VPC:



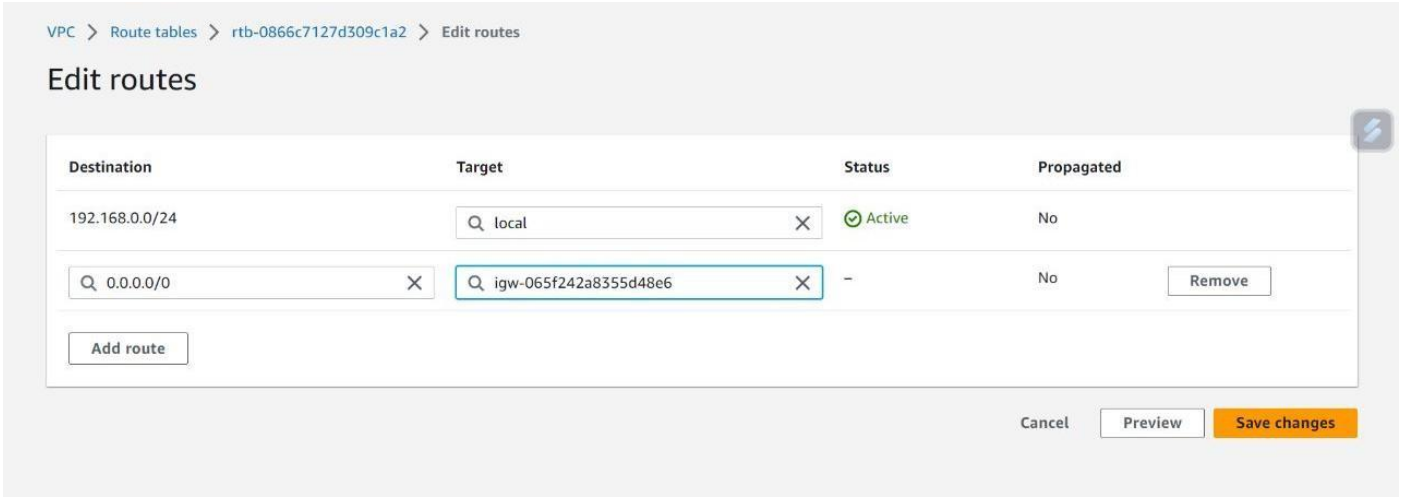
CREATE SUBNET:



EDIT SUBNET ASSOCIATIONS:

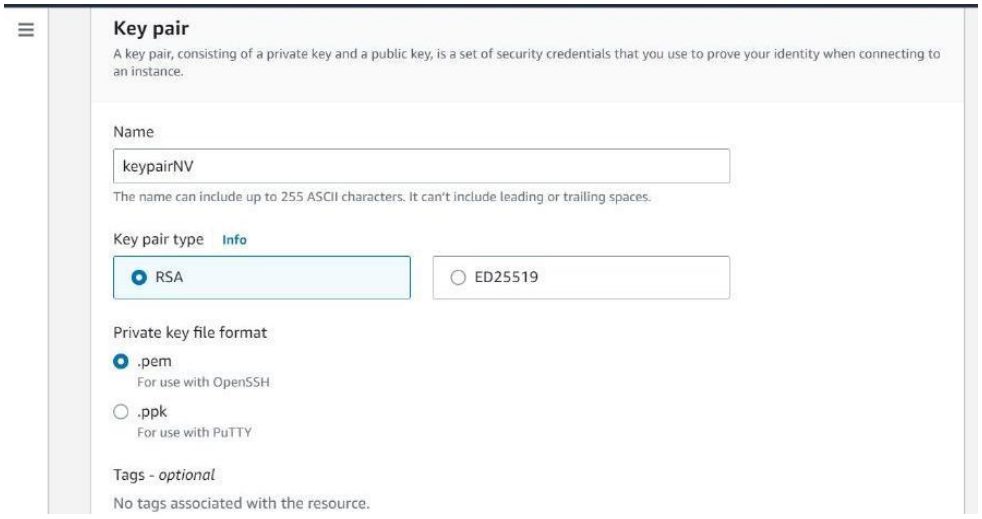


EDIT ROUTES:

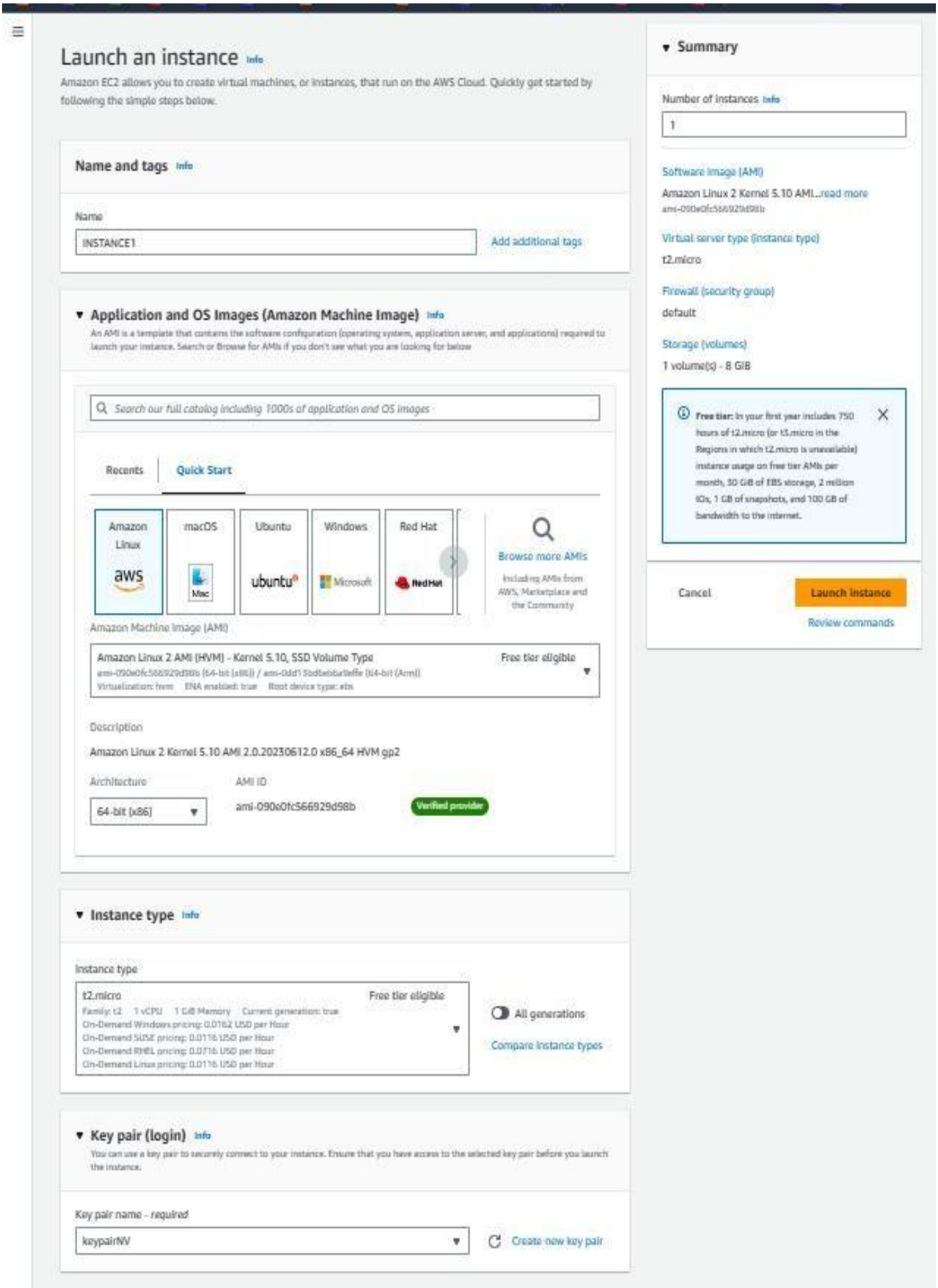


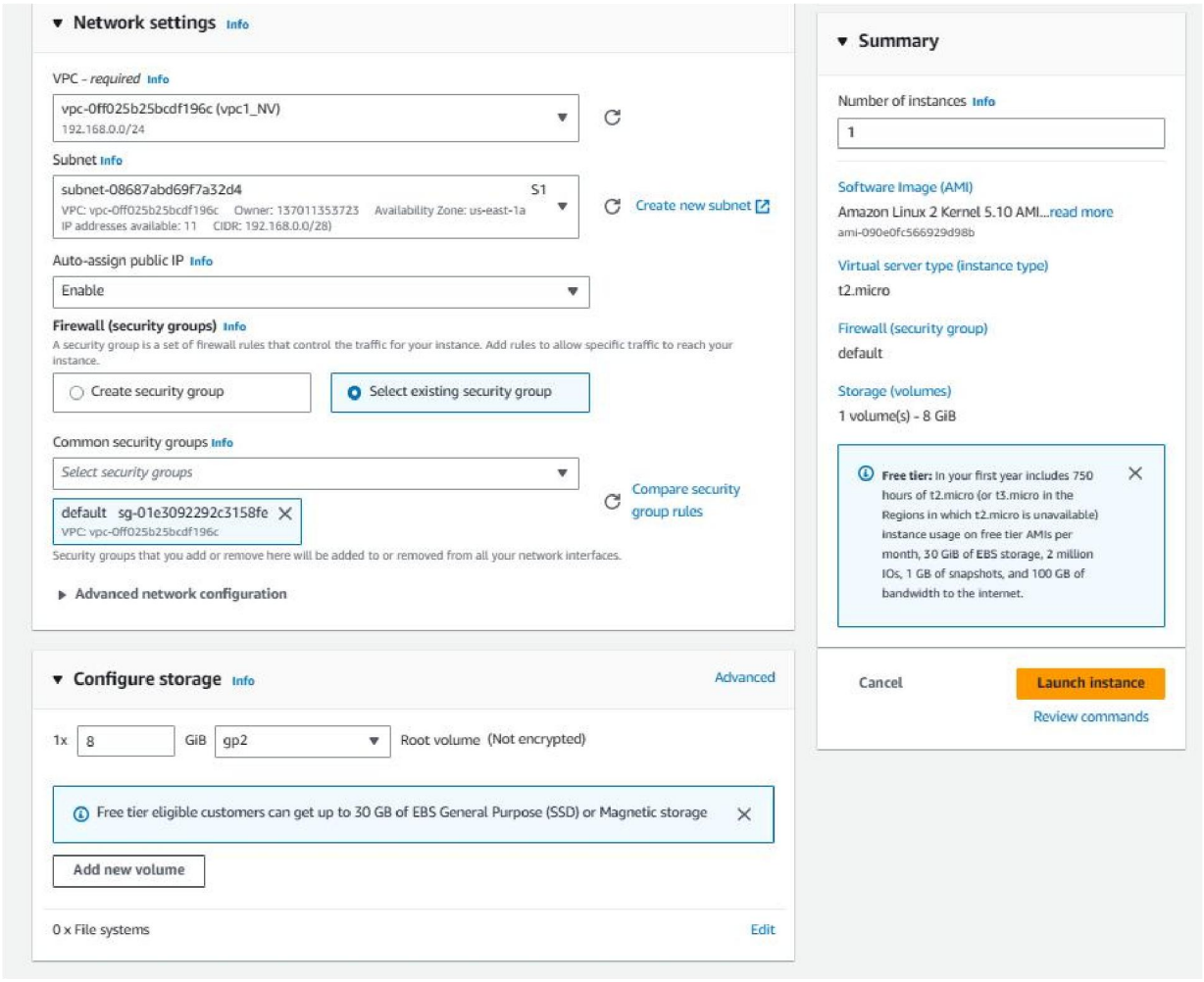
EC2 INSTANCE:

GENERATE KEYPAIR FOR INSTANCE:



LAUNCH INSTANCE:





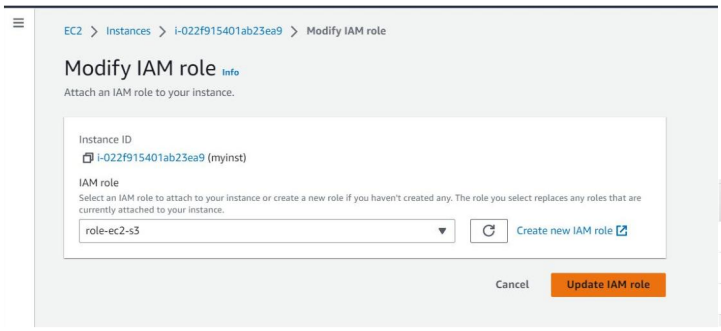
MODIFY IAM ROLE FOR THAT INSTANCE:

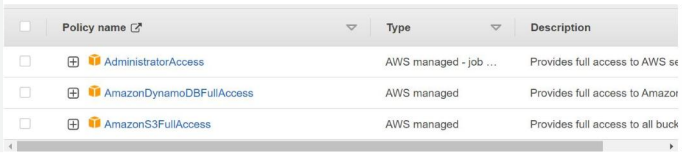
To create an IAM (Identity and Access Management) role in AWS, follow these steps:

1. Sign in to the AWS Management Console and open the IAM service.
2. In the IAM console, navigate to "Roles" in the left-hand menu and click on "Create role".
3. Choose the trusted entity type:
- If you want to create a role that can be assumed by an AWS service (e.g., EC2, Lambda), select the service from the list of services.
- If you want to create a role that can be assumed by another AWS account, select "Another AWS account" and provide the account ID.
4. Configure the role permissions:
- If you selected an AWS service, select the use case or service-specific policies that define the permissions for the role. You can also create a custom policy if needed.
- If you selected "Another AWS account", you can either select an existing policy that allows access or create a custom policy. You'll also need to specify the external account ID that can assume this role.
5. Add tags (optional):
- You can add tags to categorize and organize your IAM resources. Tags are key-value pairs that you can assign to the role.
6. Review the role configuration:
- Verify the details you provided for the role, including the trusted entity, permissions, and tags.
7. Provide a name for the role and optionally provide a description.
8. Click on "Create role" to create the IAM role.

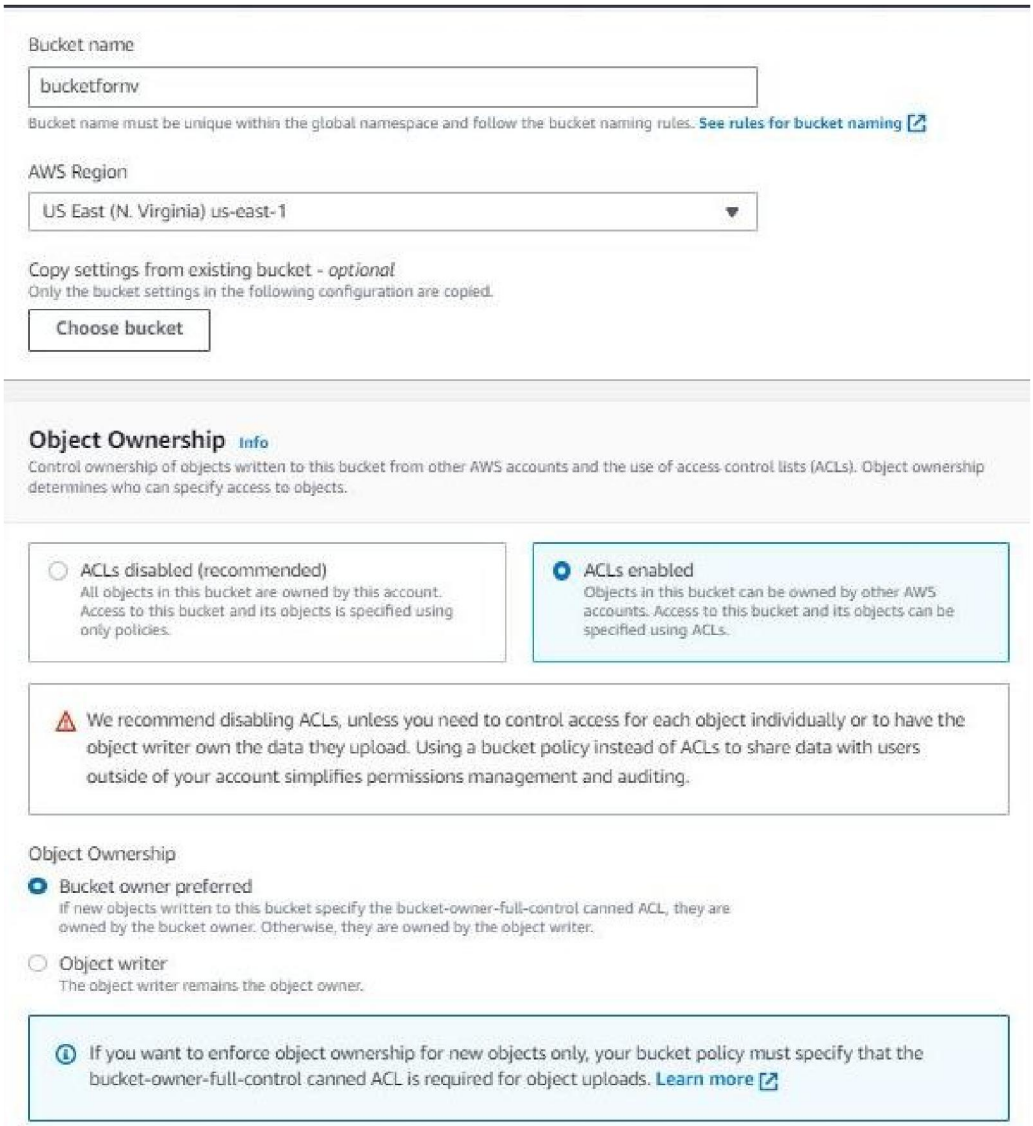
Once the IAM role is created, you can assign it to AWS resources like EC2 instances, Lambda functions, or use it for cross-account access control.
Note: It's important to carefully define the permissions associated with an IAM role to ensure the principle of least privilege. Grant only the necessary permissions required for the role's intended purpose.

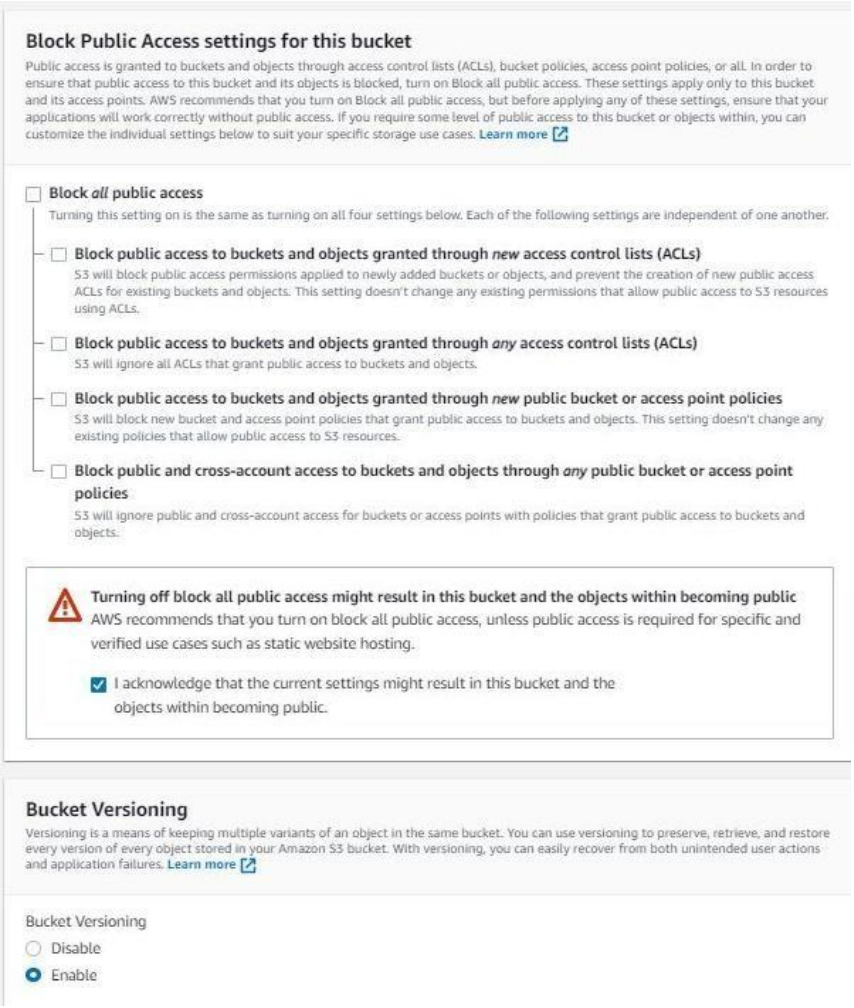
Go to actions→security→modify IAM role→add role with some permissions to access the data from s3,dynamodb





CREATE 2 BUCKETS IN NV-US-EAST-1 AND MUMBAI:





CREATE A CROSS REGION REPLICATION FOR S3 FOR 2 REGIONS NORTHERN VIRGINIA AND MUMBAI:

Click on the bucket in NV-US-EAST-1→ go to management→create replication rule→

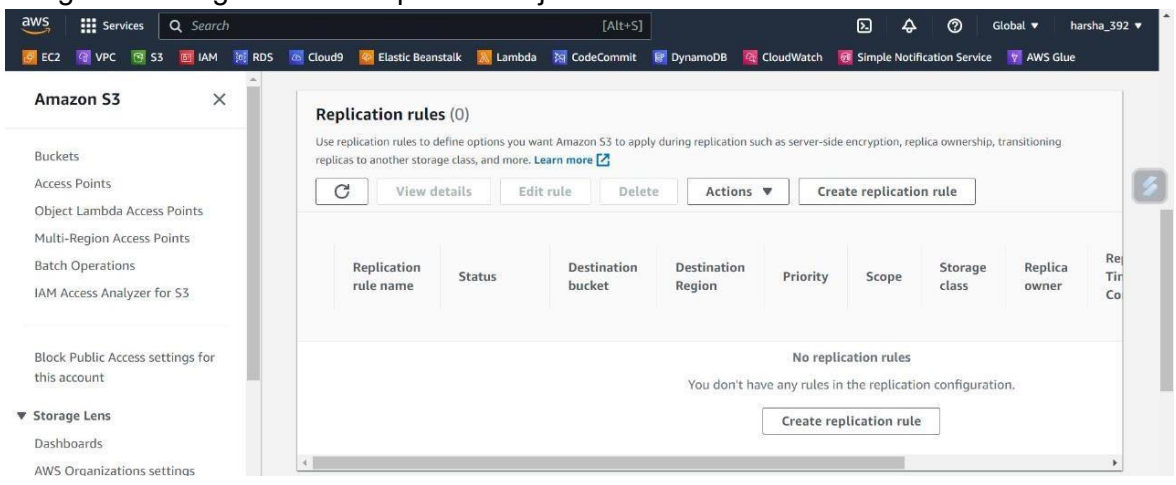
Replicate rule name

Apply to all objects in the bucket

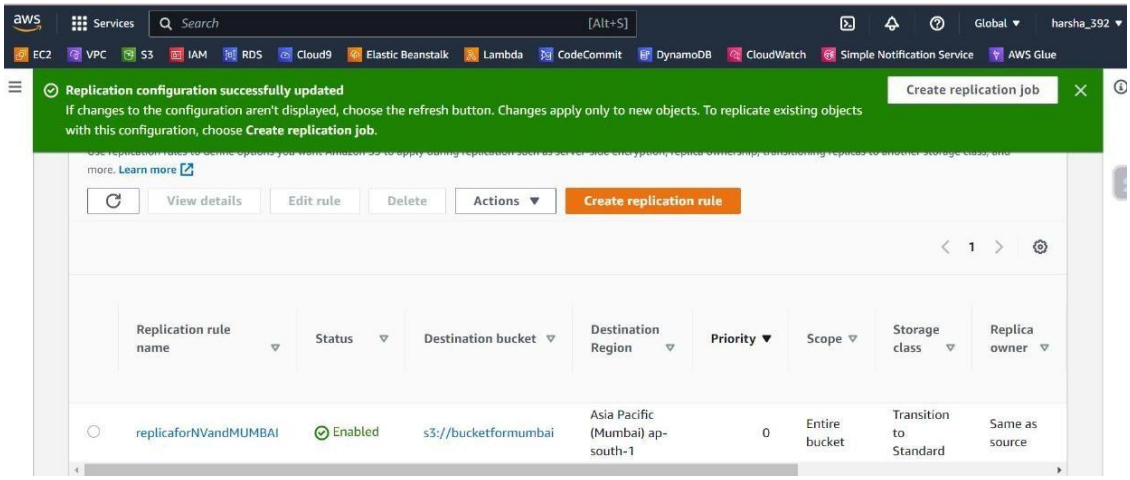
Choose destination bucket : bucket in Mumbai

Create a replication role: aws services→ec2→add permissions

Change the storage class for replicated objects→standard 1A

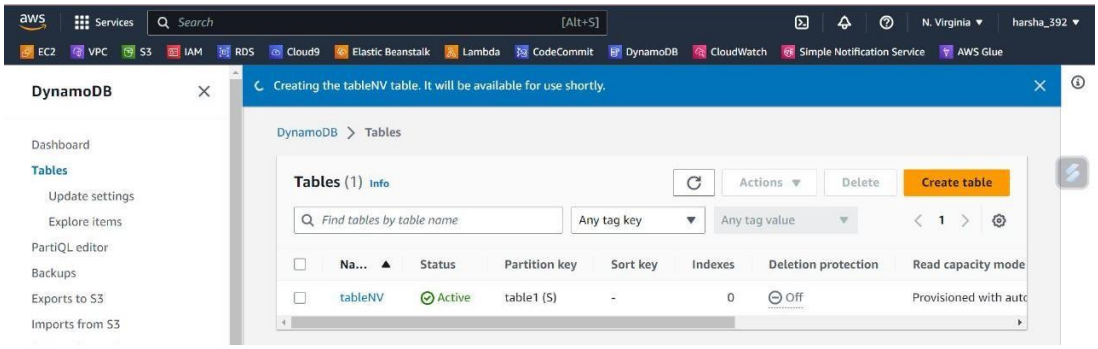


AdministratorAccess	AWS managed - job function	Provides full access to AWS services an…
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon Dynamo…
AmazonEC2FullAccess	AWS managed	Provides full access to Amazon EC2 via …
AmazonS3FullAccess	AWS managed	Provides full access to all buckets via th…
AWSLambda_FullAccess	AWS managed	Grants full access to AWS Lambda servi…

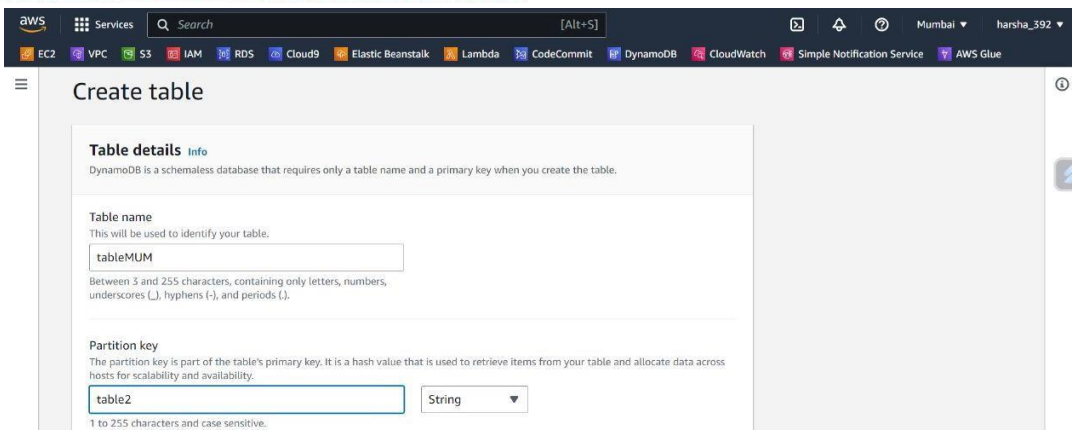


CREATE DYNAMO DB TABLES TO STORE DATA OF ANY FORMAT:

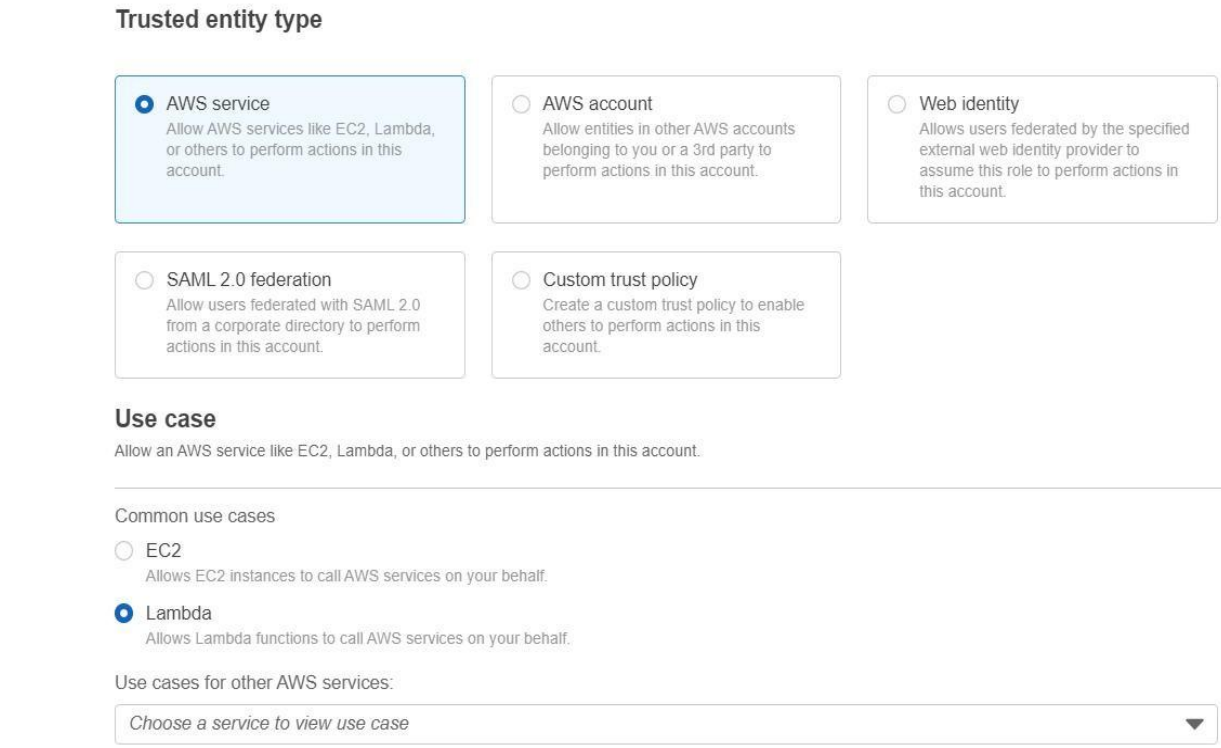
Go to dynamodb service
Click on create table
Give the name of the table
Give partial key
Default settings
Click create table

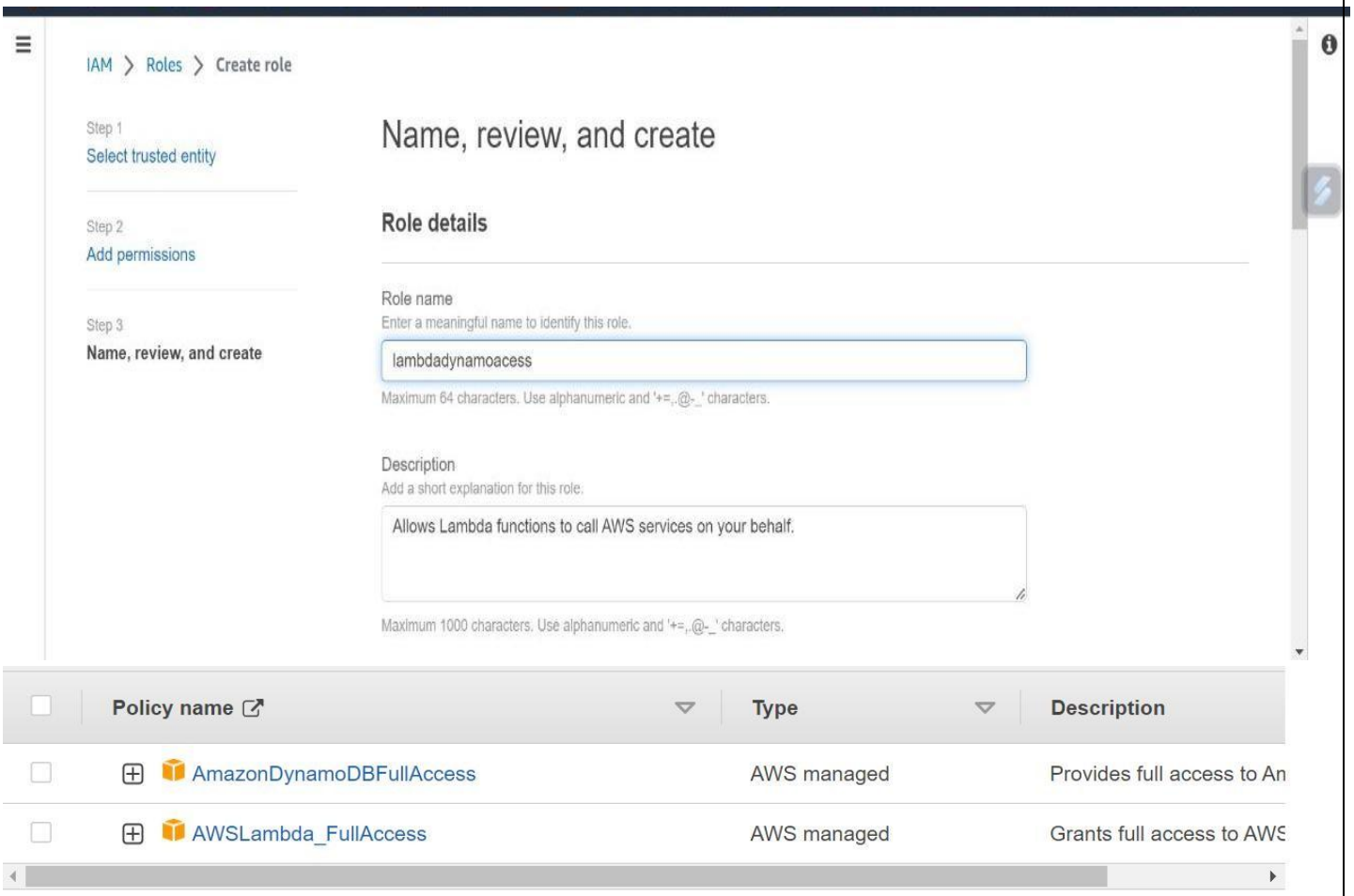


CREATE ANOTHER TABLE IN MUMBAI REGION

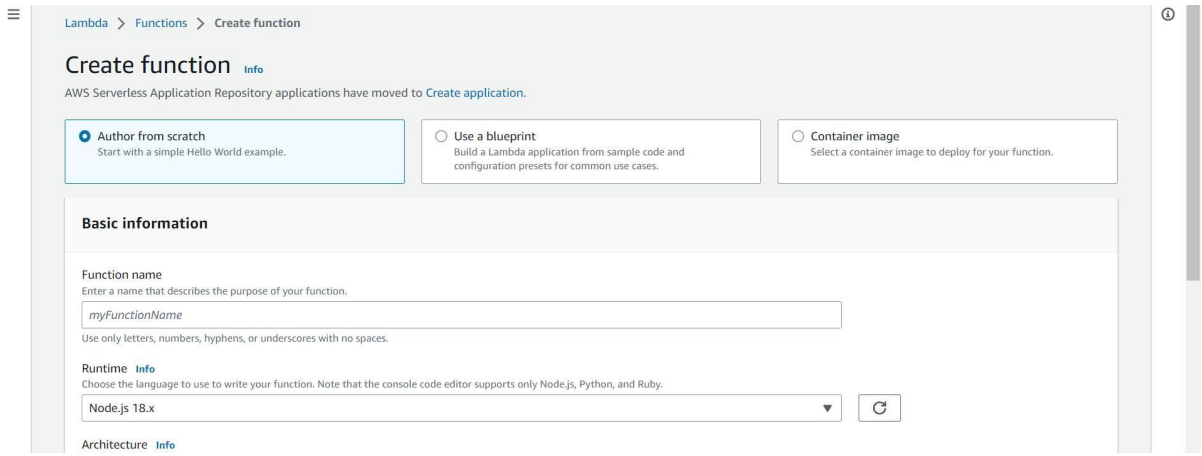


IAM ROLE TO CREATE A LAMBDA FUNCTION:

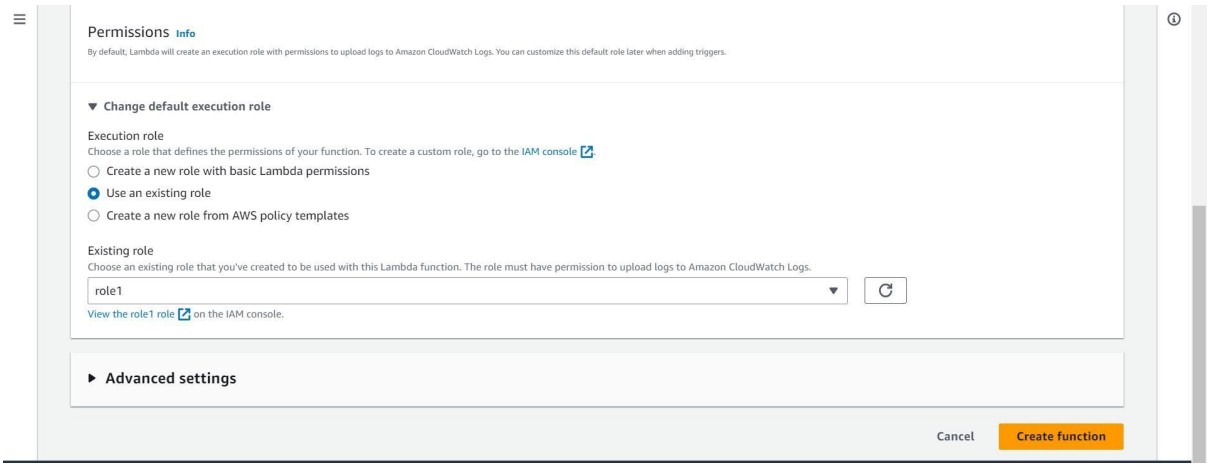




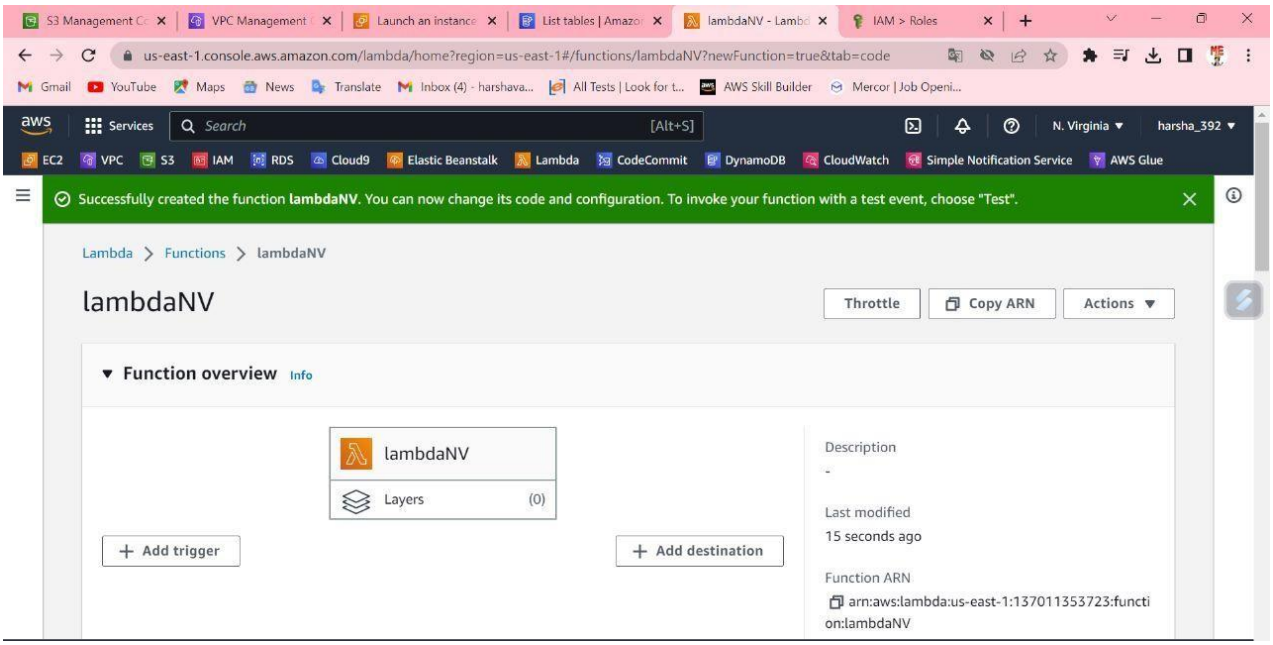
CREATE 2 LAMBDA FUNCTIONS FOR 2 REGIONS(NV-US-EAST-1 AND MUMBAI):



Now go to lambda and create the function by giving the function name and the runtime Language like python.

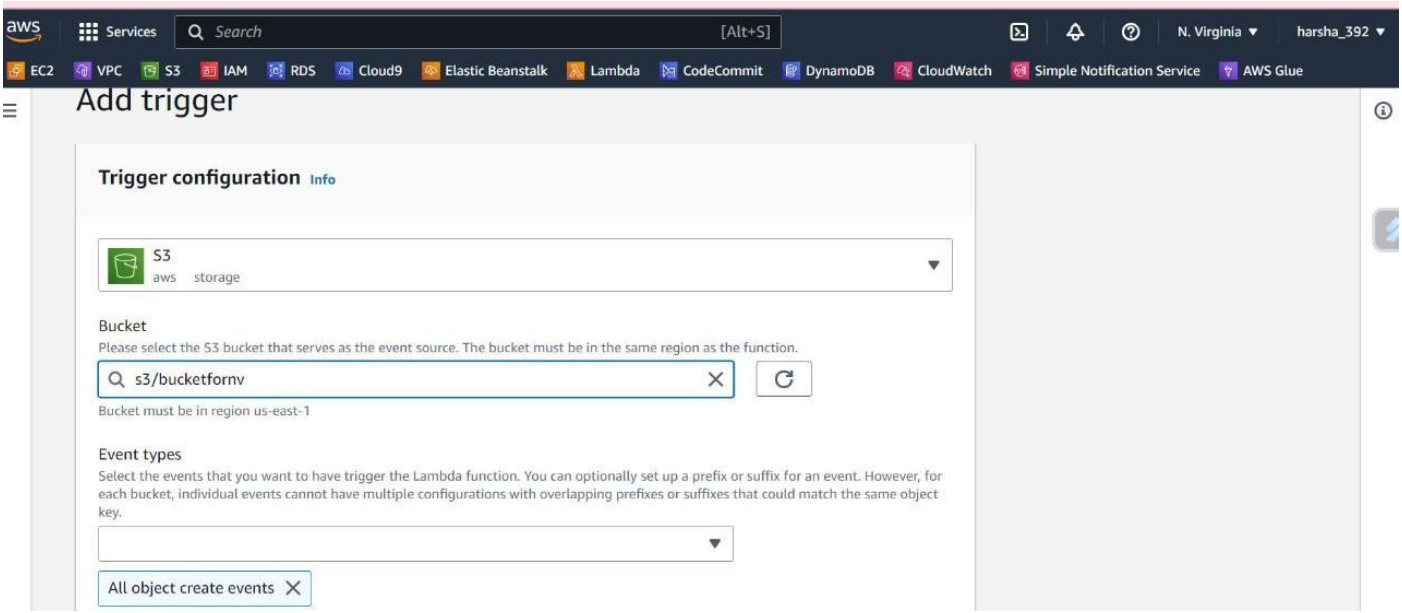


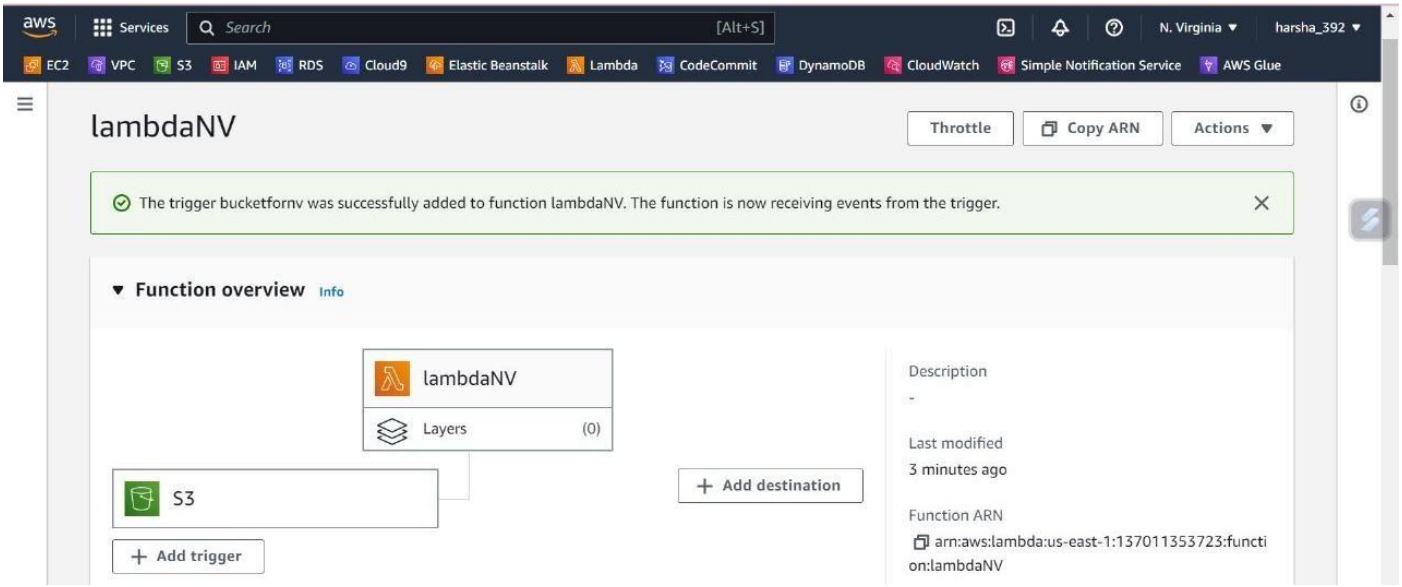
Use the created role (LAMBDADYNAMOACCESS) while creating the function and click on create function.

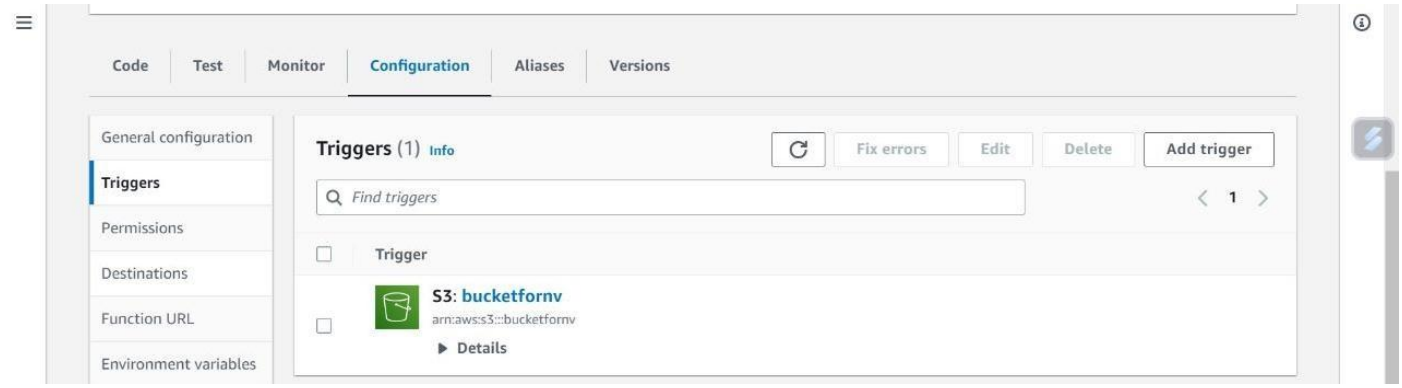


Lambda function is created successfully.

ADD TRIGGER FOR THE LAMBDA FUNCTION





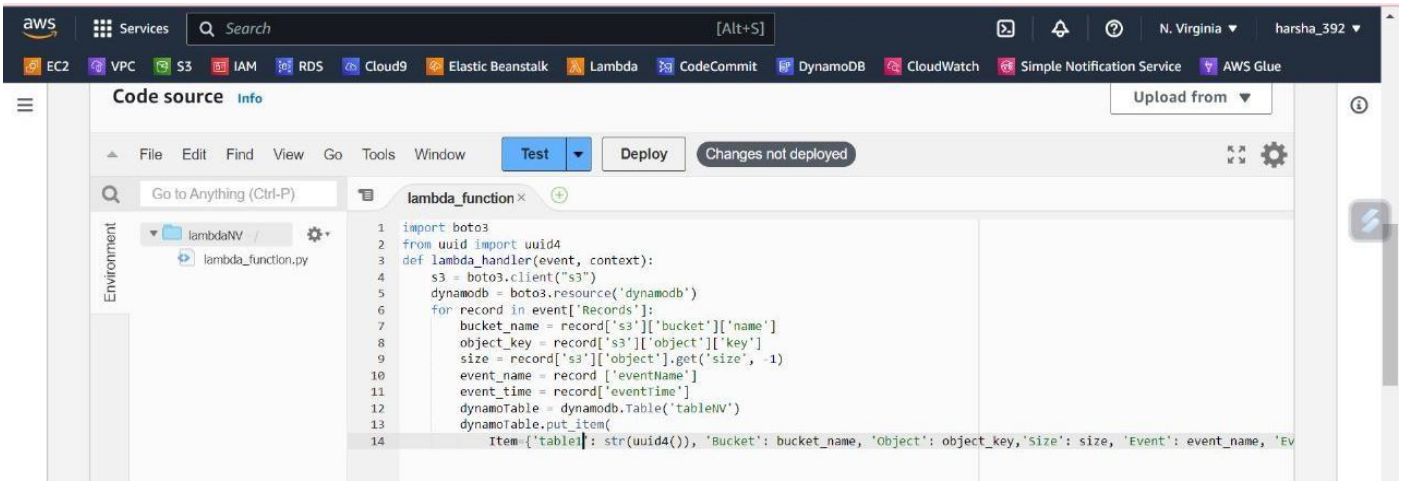


GO TO THE CODE SECTION AND WRITE THE CODE TO PUT DATA FROM S3 TO DYNAMODB:

```
import boto3
from uuid import uuid4
```

```
def lambda_handler(event, context):
s3 = boto3.client("s3")
dynamodb = boto3.resource('dynamodb')
for record in event['Records']:
bucket_name = record['s3']['bucket']['name']
object_key = record['s3']['object']['key']
size = record['s3']['object'].get('size', -1)
event_name = record ['eventName']
event_time = record['eventTime']
dynamoTable = dynamodb.Table('TableName')
dynamoTable.put_item(
Item={'PartitionKeyName': str(uuid4()), 'Bucket': bucket_name, 'Object': object_key,'Size': size, 'Event': event_name,
'EventTime': event_time})
```

Change the table name and partitionkey name in the above code.



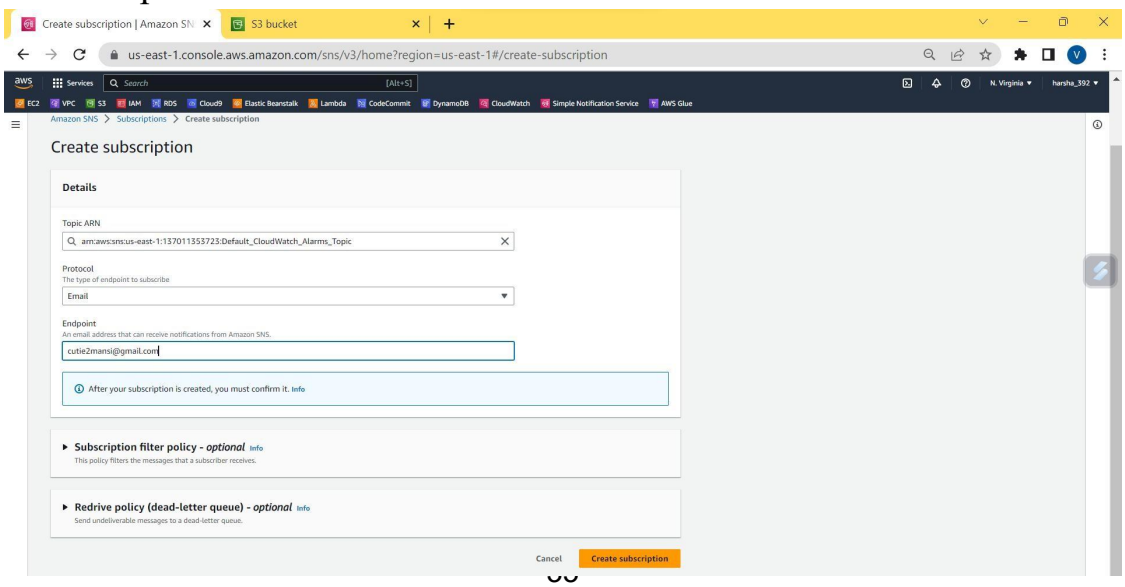
Now click on the DEPLOY.

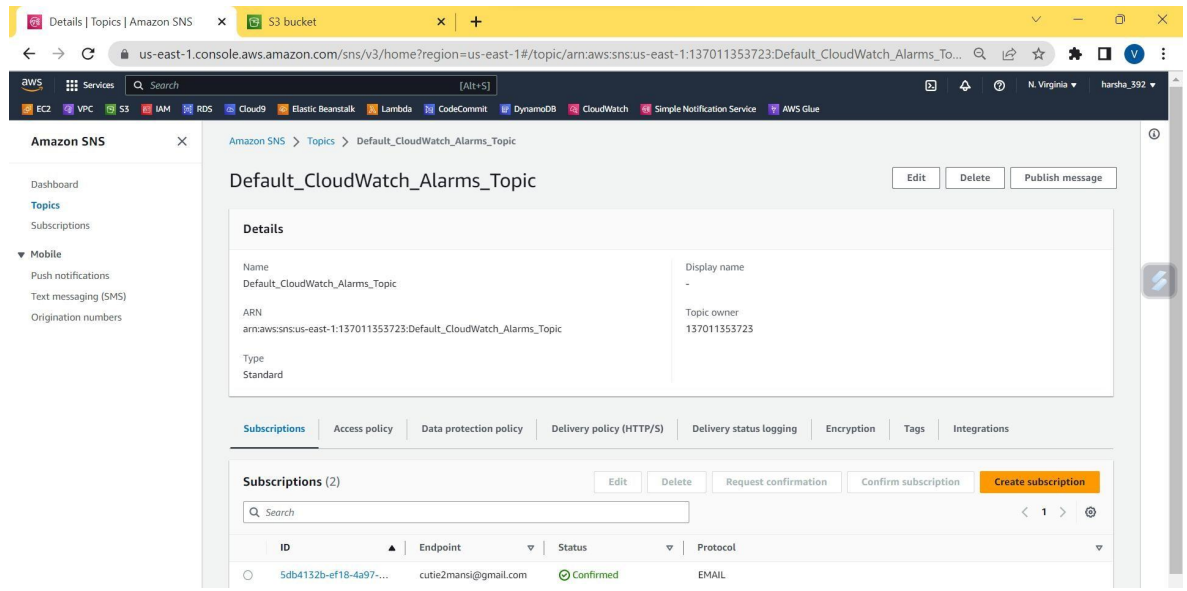
ADD SIMPLE NOTIFICATION SERVICE FOR S3 WHEN IT UPLOADS DATA INTO DYNAMO

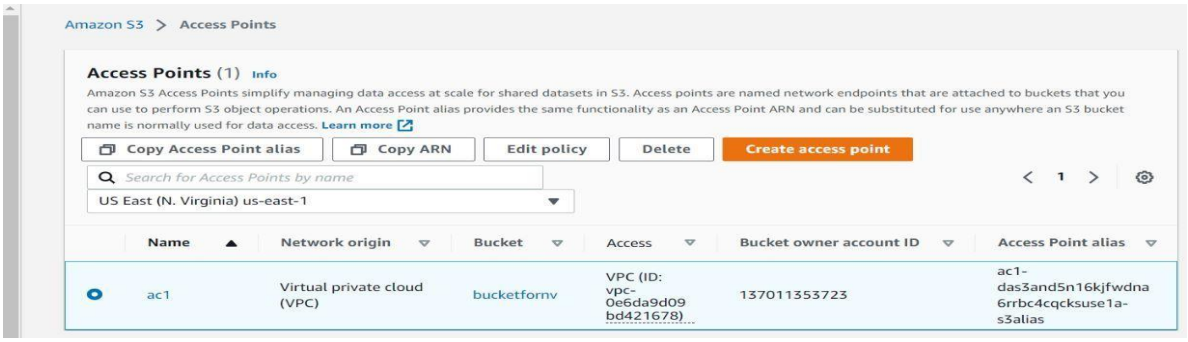
Go to SNS service

Click on topics: create topic→standard→topic name→create

Add subscription



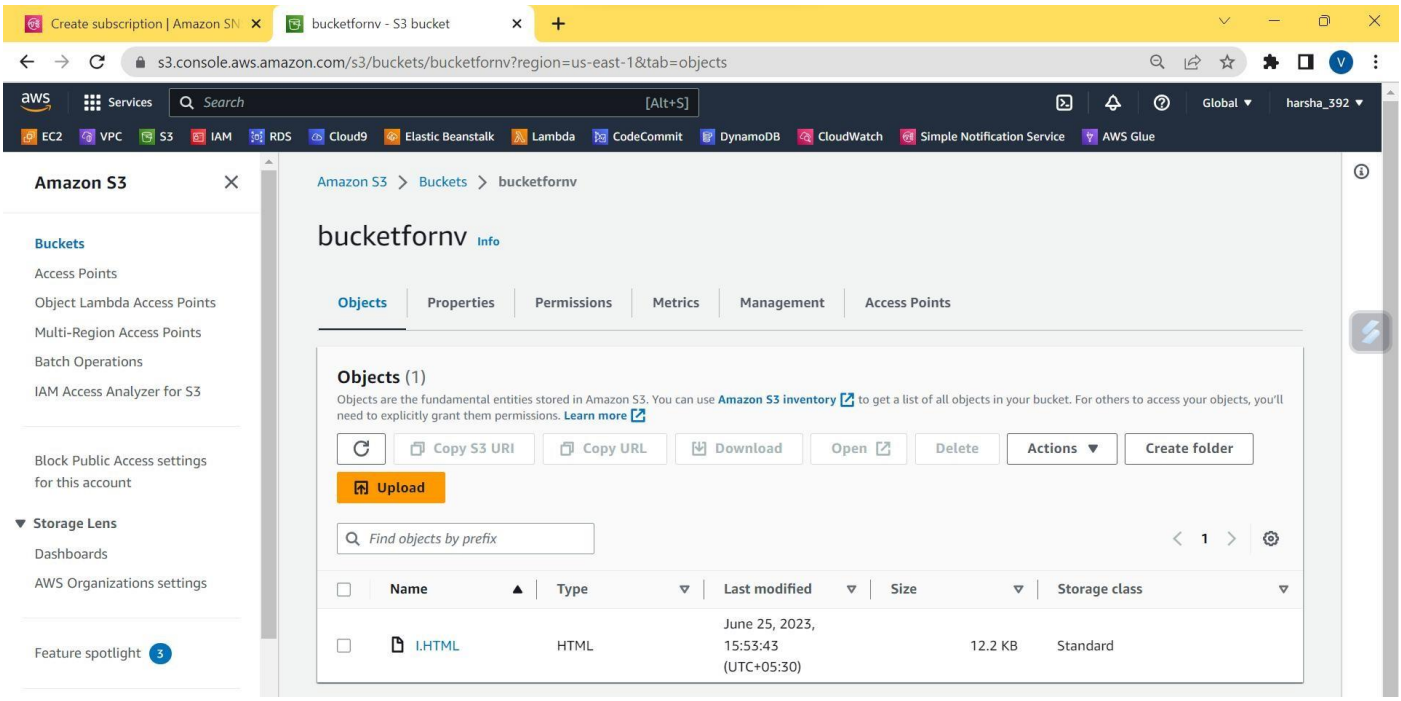




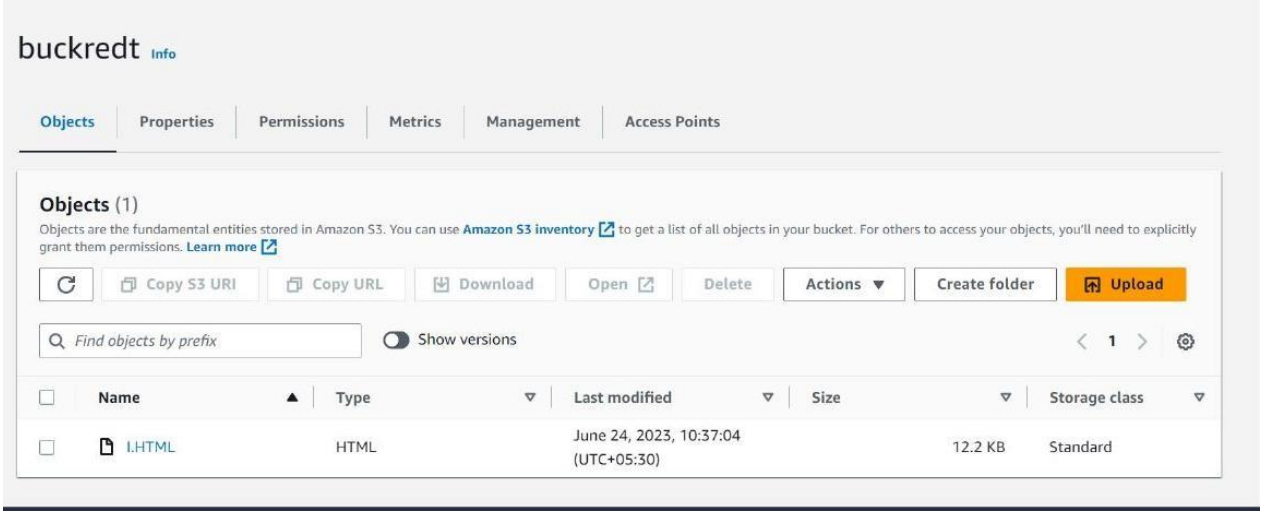
Go to access policy →edit →add the code

```
{
"Version": "2012-10-17",
"Id": "example-ID",
"Statement": [
{
"Sid": "default_cloudwatch_alarm_topic",
"Effect": "Allow",
"Principal": {
"Service": "s3.amazonaws.com"
},
"Action": [
"SNS:Publish"
],
"Resource": "arn:aws:sns:ap-south-1:137011353723: default_cloudwatch_alarm_topic ",
"Condition": {
"ArnLike": {
"aws:SourceArn": "arn:aws:s3:::bucketfornv"
},
"StringEquals": {
"aws:SourceAccount": "bucket-owner-account-id"
}}}
]}
```

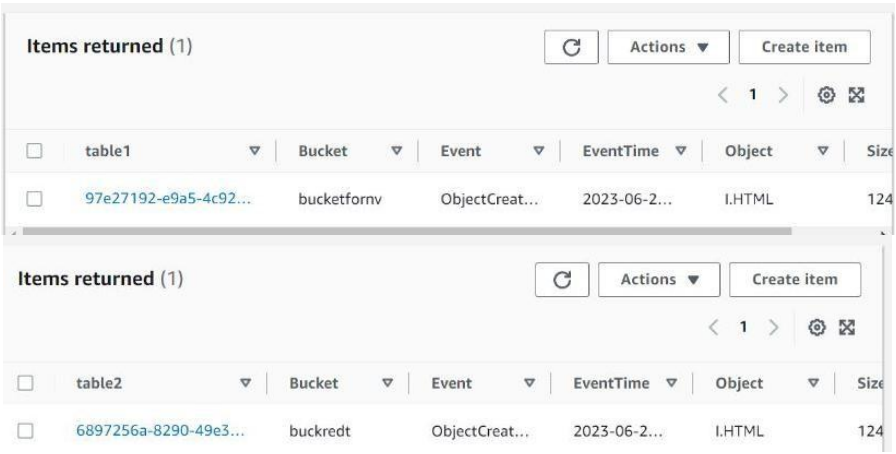
UPLOAD THE DATA IN S3 BUCKET OF NV-US-EAST-1:



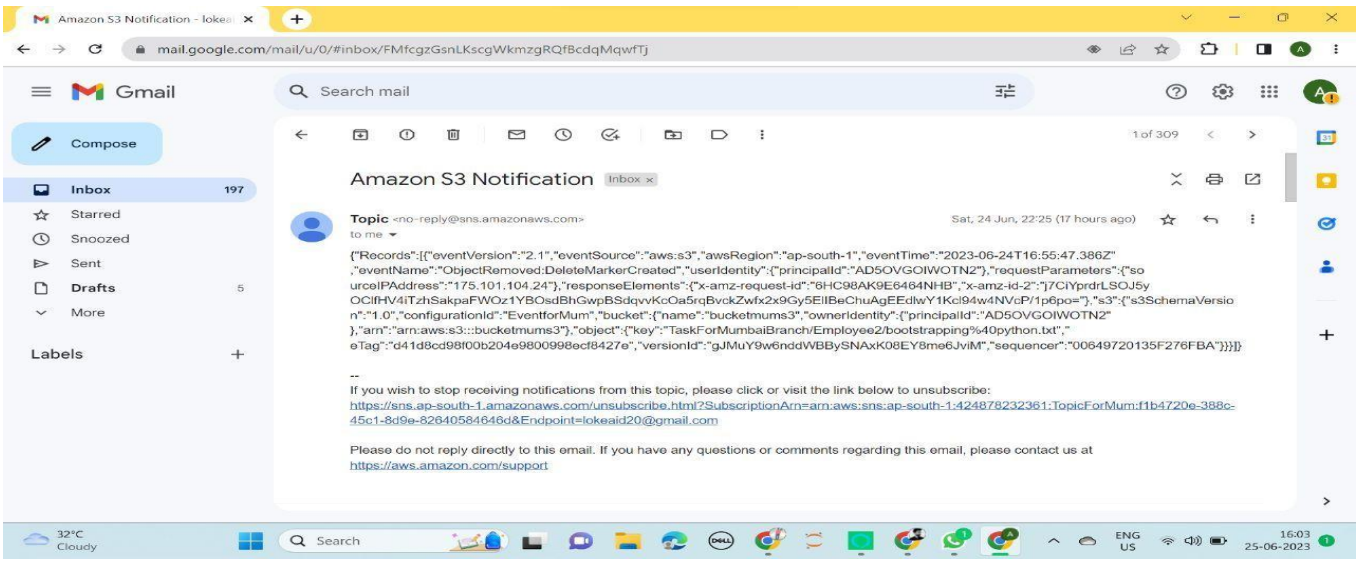
It replicates the data in mumbai bucket also



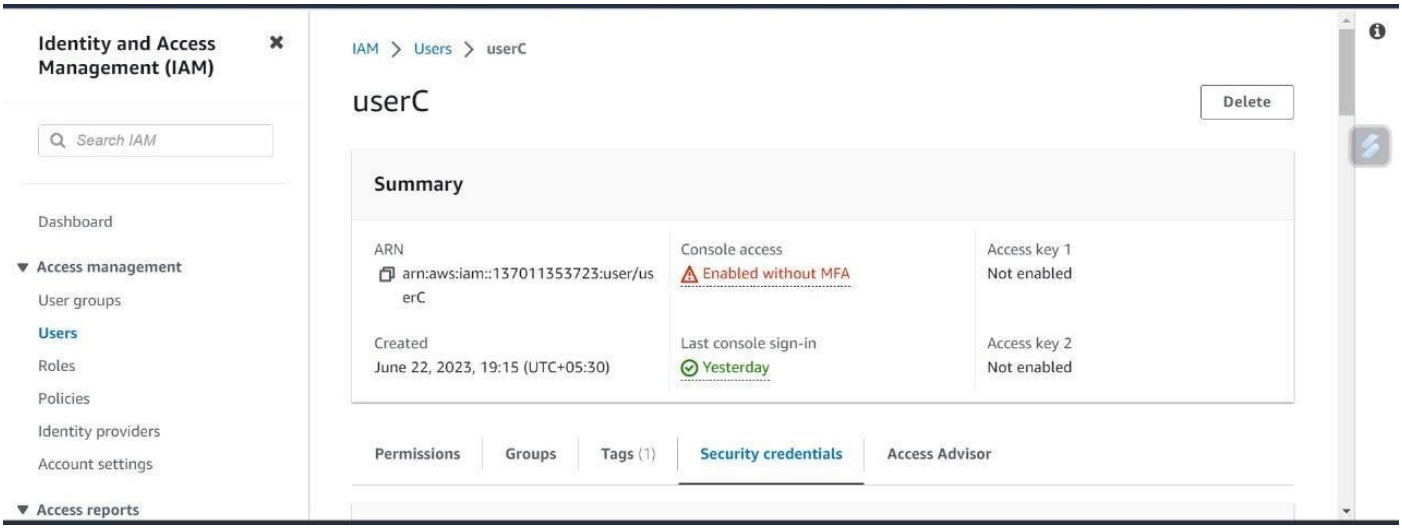
Now check the dynamodb tables:



By uploading the data we get an email through the SNS:



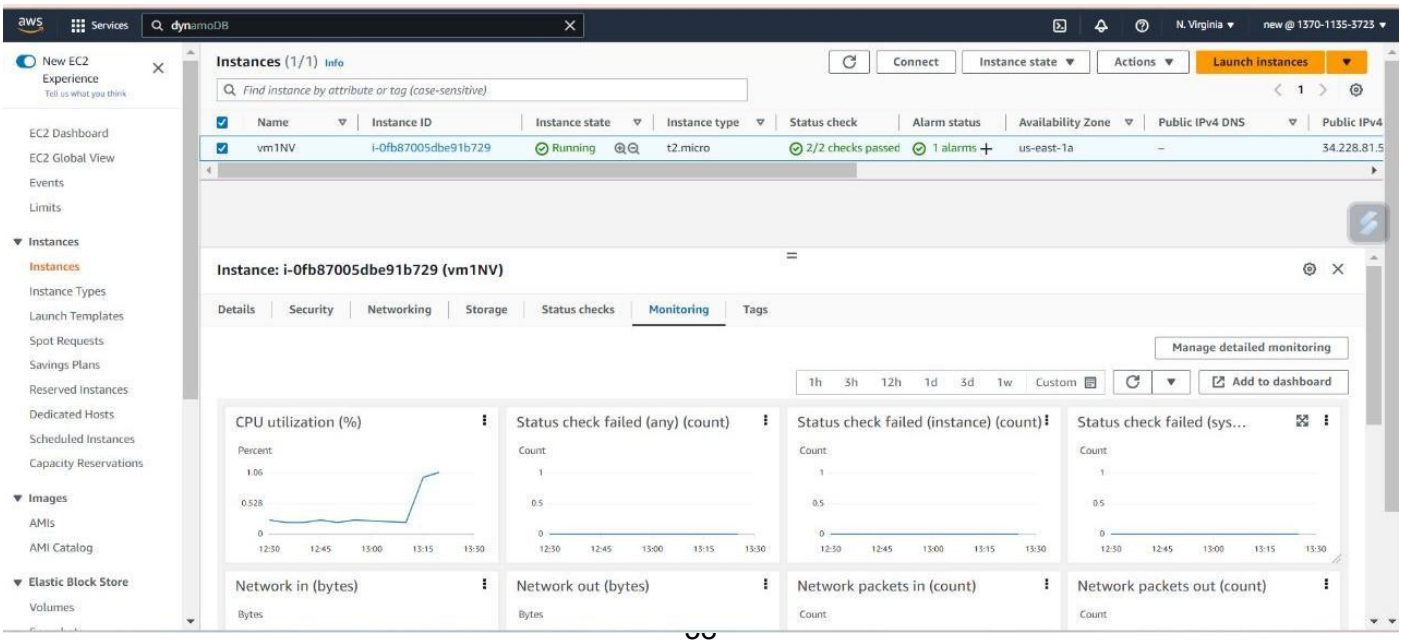
NOW CREATE AN IAM USER

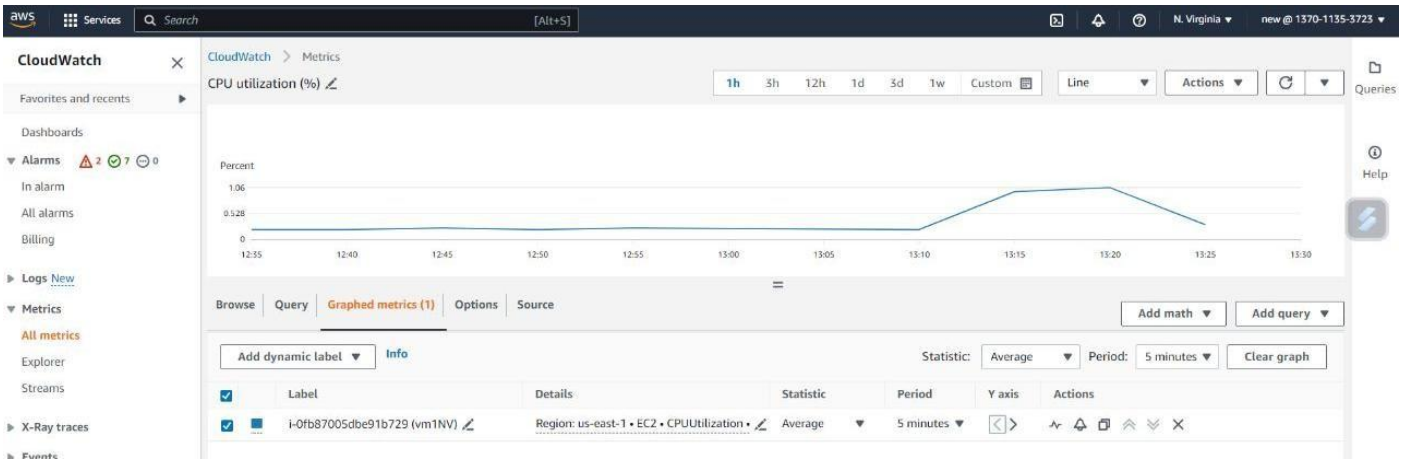


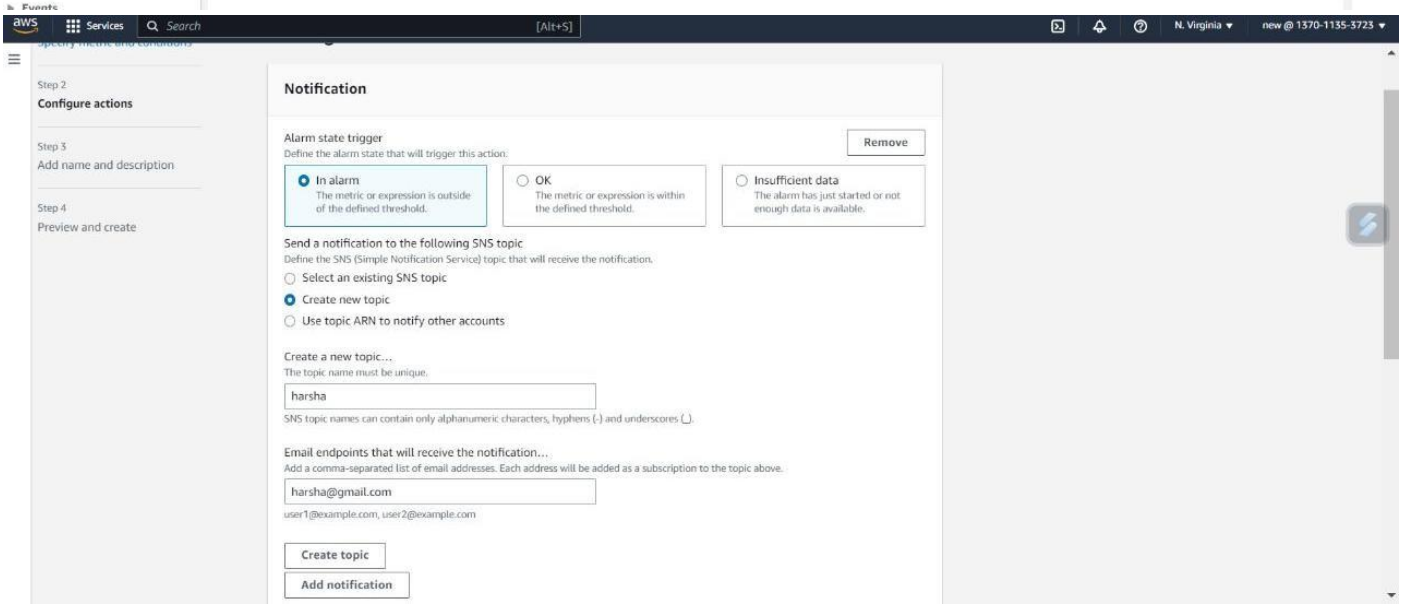
Give the permissions for that user for s3fullaccess and ec2fullaccess to track the cpu utilization and data utilization of s3 bucket

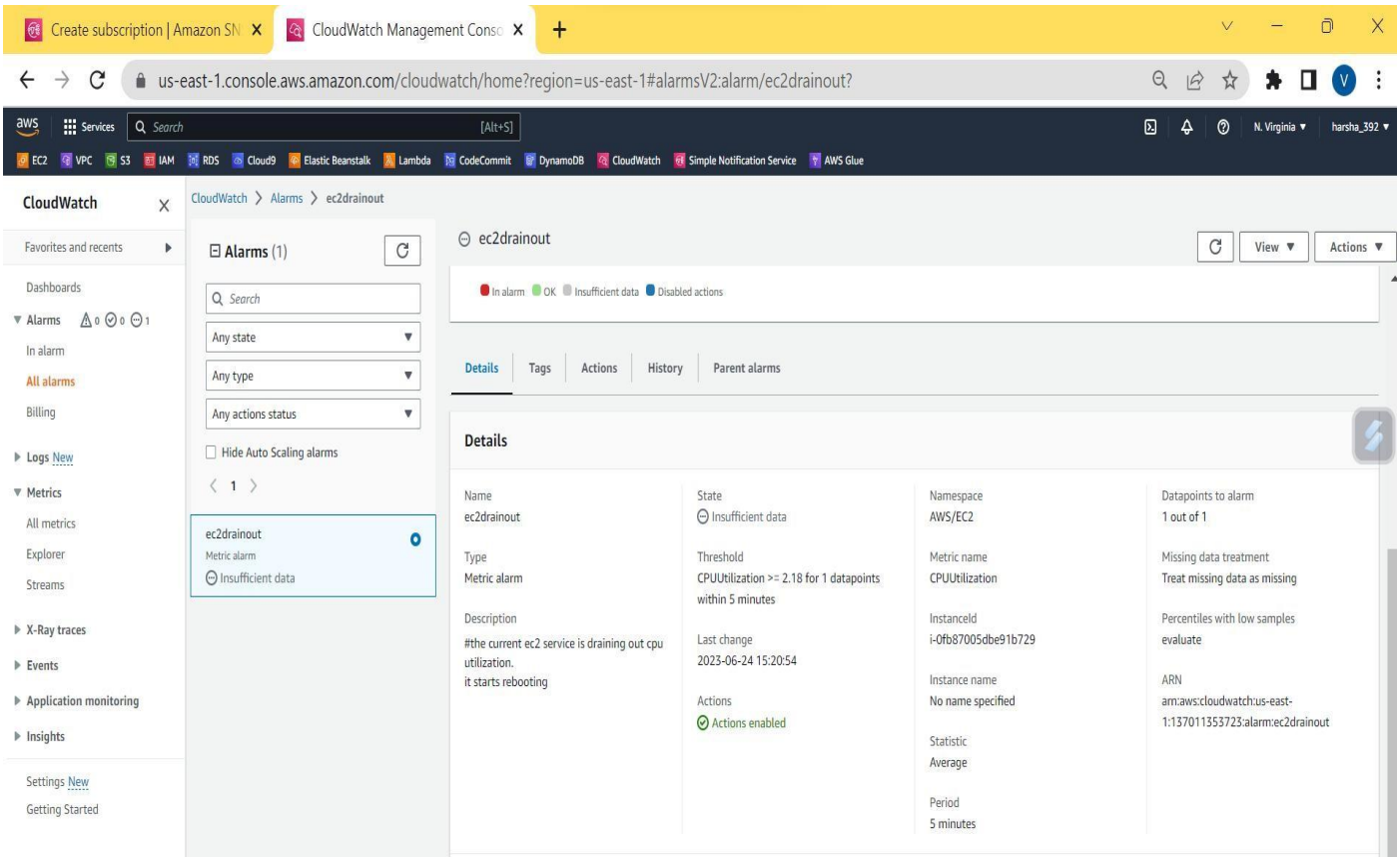
Open cloudwatch in iam user

Create alarms and subscription to send email when utilization reaches our limits

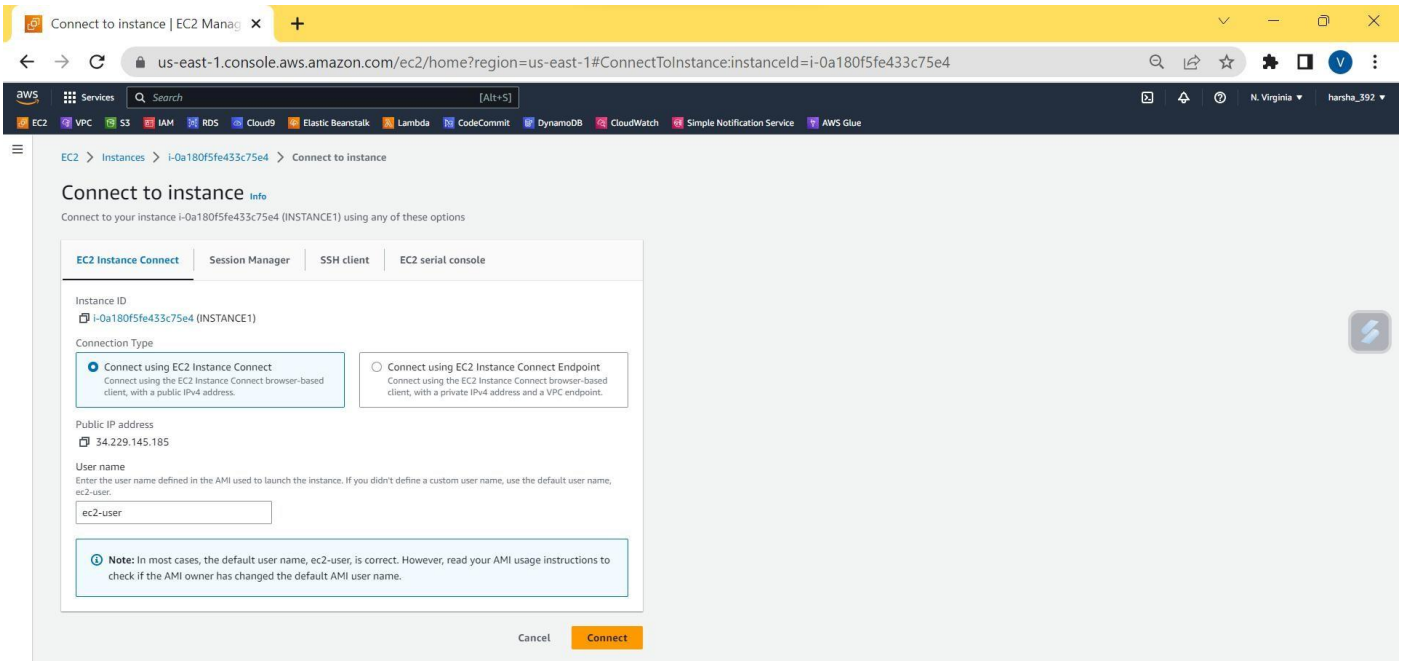


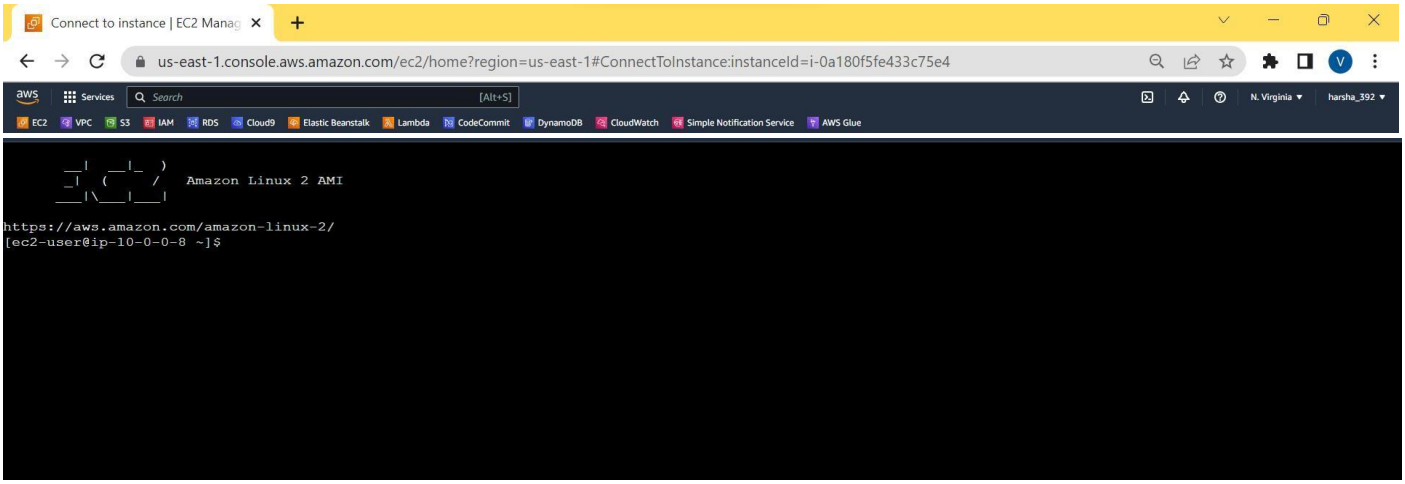

CloudWatch
CPU utilization (%)
Graphed metrics (1)
Add dynamic label
Clear graph


Notification
Alarm state trigger
Remove
In alarm
OK
Insufficient data
Send a notification to the following SNS topic
Create new topic
harsha
harsha@gmail.com
Create topic
Add notification


Create subscription | Amazon SN
CloudWatch Management Conso
us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#alarmsV2:alarm/ec2drainout?
CloudWatch
Alarms (1)
ec2drainout
Details
Tags
Actions
History
Parent alarms
Name
ec2drainout
Type
Metric alarm
Description
#the current ec2 service is draining out cpu utilization.
it starts rebooting
State
Insufficient data
Threshold
CPUUtilization >= 2.18 for 1 datapoints within 5 minutes
Last change
2023-06-24 15:20:54
Actions
Actions enabled
Namespace
AWS/EC2
Metric name
CPUUtilization
InstanceId
i-0fb87005dbe91b729
Instance name
No name specified
Statistic
Average
Period
5 minutes
Datapoints to alarm
1 out of 1
Missing data treatment
Treat missing data as missing
Percentiles with low samples
evaluate
ARN
arn:aws:cloudwatch:us-east-1:137011353723:alarm:ec2drainout

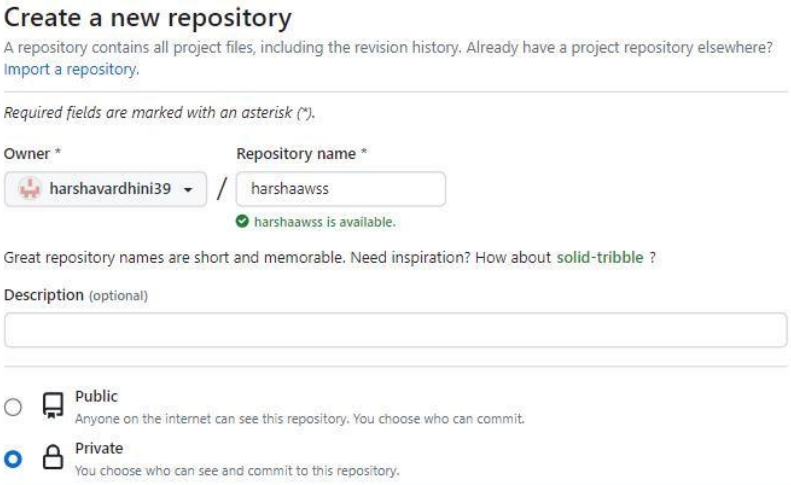
CONNECT THE INSTANCE IN NORTHERN VIRGINIA:





The instance created is running successfully

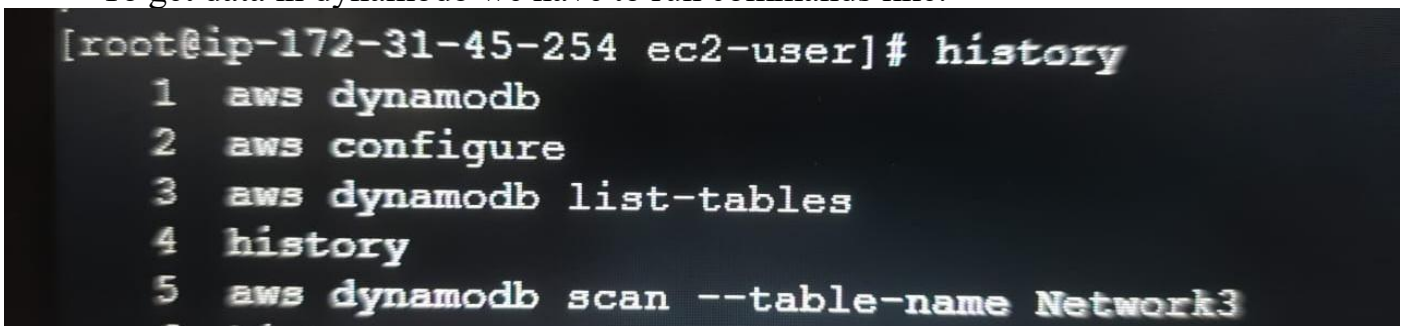
CREATE A GIT REPOSITORY WITH PRIVATE ACCESS:



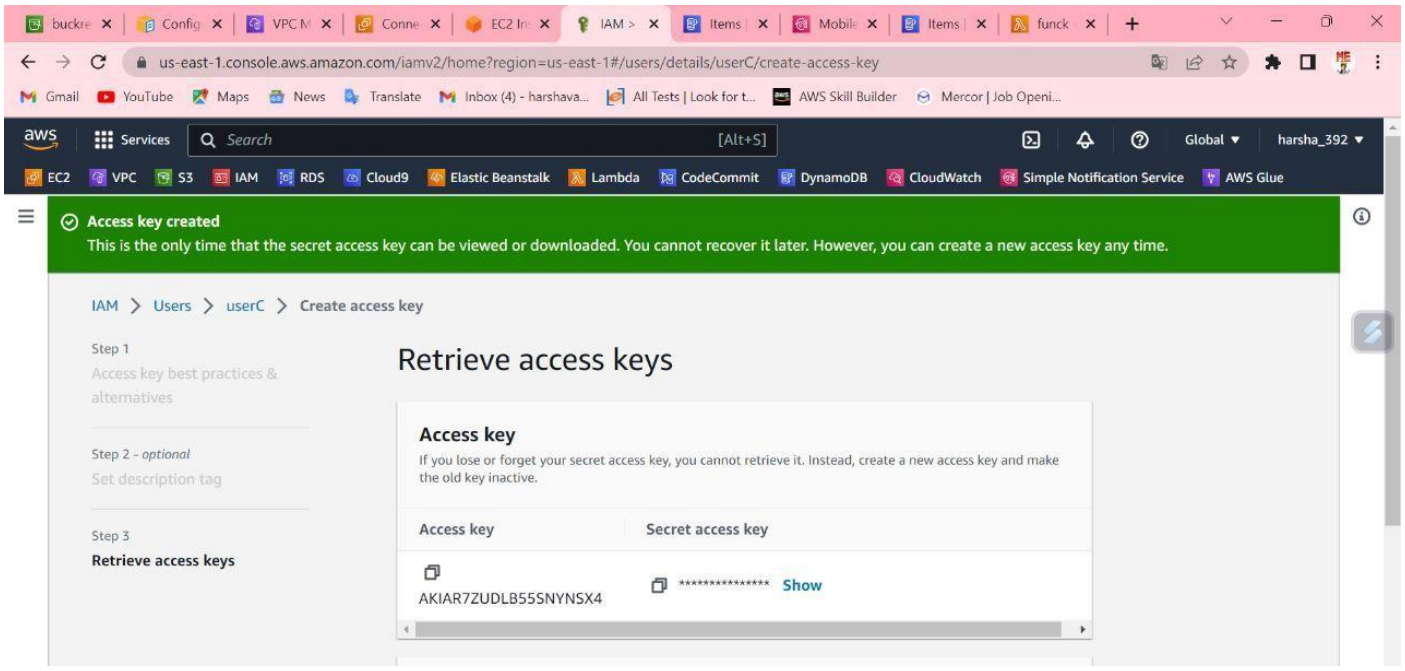
Now run the codes in ec2 connect to access the git and s3 via ec2:

```
git init
git status
[root@ip-192-168-0-6 ec2-user]# git clone https://github.com/harshavardhini39/harshaaws Cloning into 'harshaaws'...
Username for 'https://github.com': harshavardhini39
Password for 'https://harshavardhini39@github.com'
warning: You appear to have cloned an empty repository.
```

- To get data in dynamodb we have to run commands like:

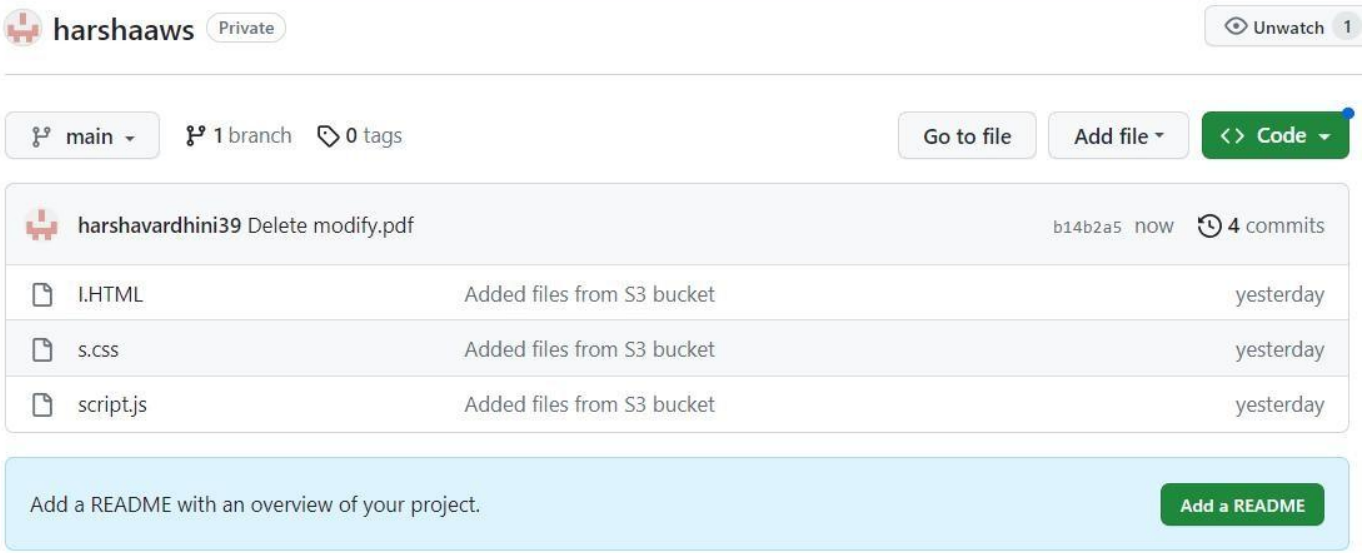


- NOW GO TO NEW IAM USER→CREATE ACCESS KEY→SELECT COMMAND LINE INTERFACE(CLI)→DESCRIPTION→RETRIVE ACCESS KEY



```
[root@ip-192-168-0-6 ec2-user]# aws configure
AWS Access Key ID: AKIAR7ZUDLB55SNYNSX4
AWS Secret Access Key : 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
Default region name :us-east-1 Default output format [None]:
[root@ip-192-168-0-6 harshaaws]# aws s3 ls
2023-06-24 02:08:02 bucketfornv
```

```
2023-06-24 04:50:41 bucketformumbai
[root@ip-192-168-0-6 ec2-user]# aws s3 sync s3://bucketformumbai
f1 download: s3://buckredt/script.js to f1/script.js download:
s3://buckredt/s.css to f1/s.css download: s3://buckredt/I.HTML to
f1/I.HTML
[root@ip-192-168-0-6 ec2-user]# cd harshaaws
[root@ip-192-168-0-6 harshaaws]# mv /home/ec2-user/f1/* .
[root@ip-192-168-0-6 harshaaws]# git add .
[root@ip-192-168-0-6 harshaaws]# git commit -m "Added files from S3 bucket"
[main 7230e04] Added files from S3 bucket
Committer: root <root@ip-192-168-0-6.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are
accurate. You can suppress this message by setting them explicitly.
Run the following command and follow the instructions in your
editor to edit your configuration file: git config --global --edit
After doing this, you may fix the identity used for this commit
with: git commit --amend --reset-author 2 files changed, 881
insertions(+) create mode 100644 s.css create mode 100644
script.js
[root@ip-192-168-0-6 harshaaws]# git remote add origin
https://github.com/harshavardhini39/harshaaws.git error: remote origin already exists.
[root@ip-192-168-0-6 harshaaws]# git push origin main
Username for 'https://github.com': harshavardhini39
Password for 'https://harshavardhini39@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 4.43 KiB | 4.43 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/harshavardhini39/harshaaws
b7d7d48..7230e04 main -> main
[root@ip-192-168-0-6 harshaaws]# mkdir temp_directory
[root@ip-192-168-0-6 harshaaws]# cd temp_directory
[root@ip-192-168-0-6 temp_directory]# git clone https://github.com/harshavardhini39/harshaaws
Cloning into 'harshaaws'...
Username for 'https://github.com': harshavardhini39 Password
for 'https://harshavardhini39@github.com': remote: Enumerating
objects: 10, done. remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done. remote: Total 10
(delta 1), reused 6 (delta 0), pack-reused 0 Receiving objects:
100% (10/10), 224.85 KiB | 20.44 MiB/s, done.
Resolving deltas: 100% (1/1), done.
[root@ip-192-168-0-6 temp_directory]# aws s3 cp . s3://bucketfornv/ --recursive
upload: harshaaws/s.css to s3://bucketfornv/harshaaws/s.css
upload: harshaaws/script.js to s3://bucketfornv/harshaaws/script.js
upload: harshaaws/I.HTML to
s3://bucketfornv/harshaaws/I.HTML
```


harshaaws
Private
Unwatch 1
main
1 branch
0 tags
Go to file
Add file
Code
harshavardhini39 Delete modify.pdf
b14b2a5 now
4 commits
I.HTML
Added files from S3 bucket
yesterday
s.css
Added files from S3 bucket
yesterday
script.js
Added files from S3 bucket
yesterday
Add a README with an overview of your project.
Add a README