Amazon Web Services (AWS)

1.Cloud Computing

What is Cloud?

The term cloud refers to a network or the internet. It is a technology that uses remote servers on the internet to store, manage, and access data online rather than local drives. The data can be anything such as files, images, documents, audio, video, and more.

There are the following operations that we can do using cloud computing:

- Developing new applications and services
- Storage, back up, and recovery of data
- Hosting blogs and websites
- Delivery of software on demand
- Analysis of data
- Streaming videos and audios

Evolution of Cloud Computing

Cloud Computing has evolved from the Distributed system to the current technology. Cloud computing has been used by all types of businesses, of different sizes and fields.

1. Distributed Systems

In the networks, different systems are connected. When they target to send the message from different independent systems which are physically located in various places but are connected through the network. Some examples of distributed systems are Ethernet which is a LAN technology, Telecommunication network, and parallel processing. The Basic functions of the distributed systems are −

- **Resource Sharing** − The Resources like data, hardware, and software can be shared between them.
- **Open-to-all** − The software is designed and can be shared.
- **Fault Detection** − The error or failure in the system is detected and can be corrected.

Apart from the functions, the main disadvantage is that all the plan has to be in the same location and this disadvantage is overcome by the following systems −

- Mainframe Computing
- Cluster Computing
- Grid Computing

2. Mainframe Computing

It was developed in the year 1951 and provides powerful features. Mainframe Computing is still in existence due to its ability to deal with a large amount of data. For a company that needs to access and share a vast amount of data then this computing is preferred. Among the four types of computers, mainframe computer performs very fast and lengthy computations easily.

The type of services handled by them is bulk processing of data and exchanging large-sized hardware. Apart from the performance, mainframe computing is very expensive.

3. Cluster Computing

In Cluster Computing, the computers are connected to make it a single computing. The tasks in Cluster computing are performed concurrently by each computer also known as the nodes which are connected to the network. So the activities performed by any single node are known to all the nodes of the computing which may increase the performance, transparency, and processing speed.

To eliminate the cost, cluster computing has come into existence. We can also resize the cluster computing by removing or adding the nodes.

4. Grid Computing

It was introduced in the year 1990. As the computing structure includes different computers or nodes, in this case, the different nodes are placed in different geographical places but are connected to the same network using the internet.

The other computing methods seen so far, it has homogeneous nodes that are located in the same place. But in this grid computing, the nodes are placed in different organizations. It minimized the problems of cluster computing but the distance between the nodes raised a new problem.

5. Web 2.0

This computing lets the users generate their content and collaborate with other people or share the information using social media, for example, Facebook, Twitter, and Orkut. Web 2.0 is a combination of the second-generation technology World Wide Web (WWW) along with the web services and it is the computing type that is used today.

6. Virtualization

It came into existence 40 years back and it is becoming the current technique used in IT firms. It employs a software layer over the hardware and using this it provides the customer with cloud-based services.

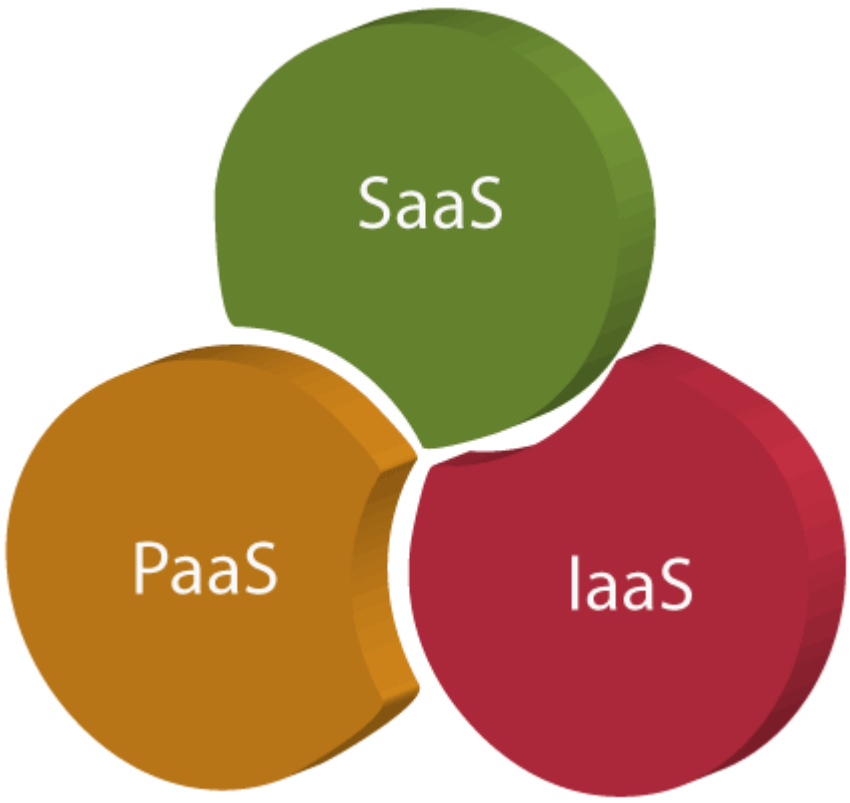
7. Utility Computing

Based on the need of the user, utility computing can be used. It provides the users, company, clients or based on the business need the data storage can be taken for rent and used.

Cloud Service Models (IaaS, PaaS, SaaS)

There are the following three types of cloud service models -

1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Software as a Service (SaaS)



Infrastructure as a Service (IaaS)

IaaS is also known as **Hardware as a Service (HaaS)**. It is a computing infrastructure managed over the internet. The main advantage of using IaaS is that it helps users to avoid the cost and complexity of purchasing and managing the physical servers.

Characteristics of IaaS

There are the following characteristics of IaaS -

- Resources are available as a service
- Services are highly scalable
- Dynamic and flexible
- GUI and API-based access
- Automated administrative tasks

Example: DigitalOcean, Linode, Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine (GCE), Rackspace, and Cisco Metacloud.

To know more about the IaaS, **click here**.

Platform as a Service (PaaS)

PaaS cloud computing platform is created for the programmer to develop, test, run, and manage the applications.

Characteristics of PaaS

There are the following characteristics of PaaS -

- Accessible to various users via the same development application.
- Integrates with web services and databases.
- Builds on virtualization technology, so resources can easily be scaled up or down as per the organization's need.
- Support multiple languages and frameworks.
- Provides an ability to "**Auto-scale**".

Example: AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, Magento Commerce Cloud, and OpenShift.

To know more about PaaS, click here.

Software as a Service (SaaS)

SaaS is also known as "**on-demand software**". It is a software in which the applications are hosted by a cloud service provider. Users can access these applications with the help of internet connection and web browser.

Characteristics of SaaS

There are the following characteristics of SaaS -

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users are not responsible for hardware and software updates. Updates are applied automatically.
- The services are purchased on the pay-as-per-use basis

Example: BigCommerce, Google Apps, Salesforce, Dropbox, ZenDesk, Cisco WebEx, ZenDesk, Slack, and GoToMeeting.

To know more about the SaaS, **click here**.

Difference between IaaS, PaaS, and SaaS

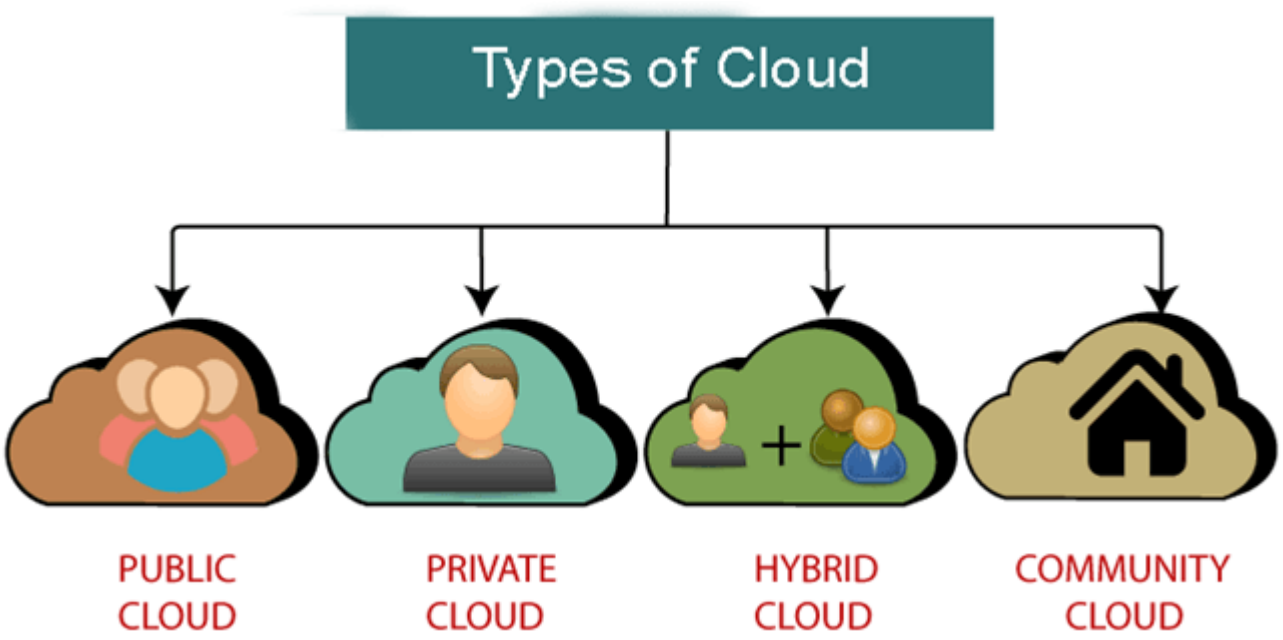
The below table shows the difference between IaaS, PaaS, and SaaS -

IaaS	Paas	SaaS
It provides a virtual data center to store information and create platforms for app development, testing, and deployment.	It provides virtual platforms and tools to create, test, and deploy apps.	It provides web software and apps to complete business tasks.
It provides access to resources such as virtual machines, virtual storage, etc.	It provides runtime environments and deployment tools for applications.	It provides software as a service to the end-users.
It is used by network architects.	It is used by developers.	It is used by end users.

IaaS provides only Infrastructure.	PaaS provides Infrastructure+Platform.	SaaS provides Infrastructure+Platform +Software.

Cloud Deployment Model (Public, Private & Hybrid)

There are the following 4 types of cloud that you can deploy according to the organization's needs-



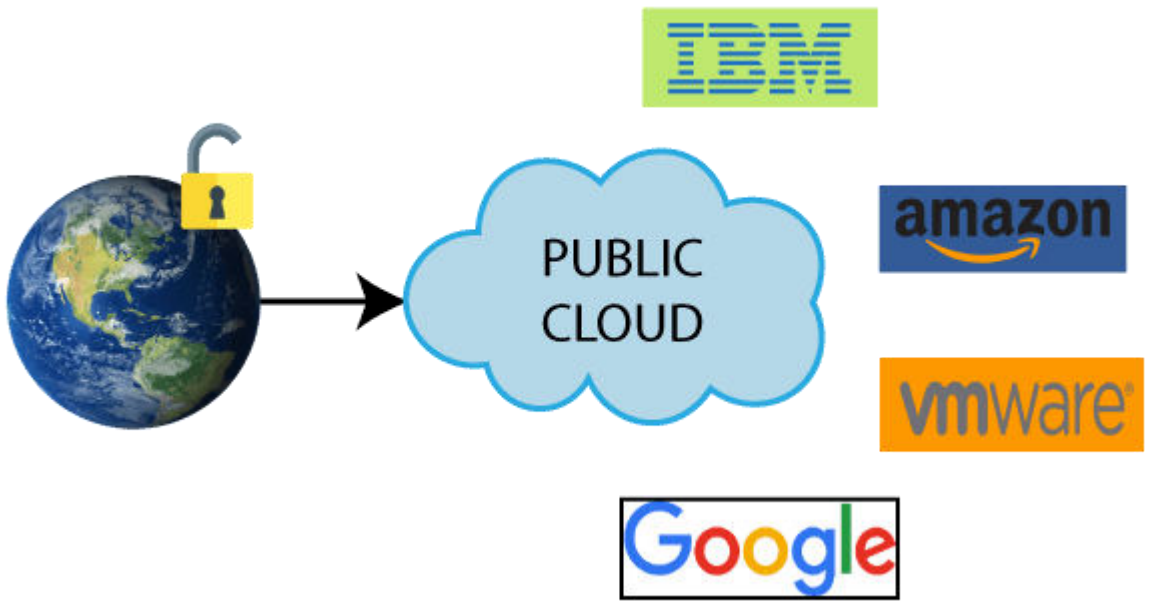
- Public Cloud
- Private Cloud
- Hybrid Cloud
- Community Cloud

Public Cloud

Public cloud is **open to all** to store and access information via the Internet using the pay-per-usage method.

In public cloud, computing resources are managed and operated by the Cloud Service Provider (CSP).

Example: Amazon elastic compute cloud (EC2), IBM SmartCloud Enterprise, Microsoft, Google App Engine, Windows Azure Services Platform.



Advantages of Public Cloud

There are the following advantages of Public Cloud -

- Public cloud is owned at a lower cost than the private and hybrid cloud.

- Public cloud is maintained by the cloud service provider, so do not need to worry about the maintenance.
- Public cloud is easier to integrate. Hence it offers a better flexibility approach to consumers.
- Public cloud is location independent because its services are delivered through the internet.
- Public cloud is highly scalable as per the requirement of computing resources.
- It is accessible by the general public, so there is no limit to the number of users.

Disadvantages of Public Cloud

- Public Cloud is less secure because resources are shared publicly.
- Performance depends upon the high-speed internet network link to the cloud provider.
- The Client has no control of data.

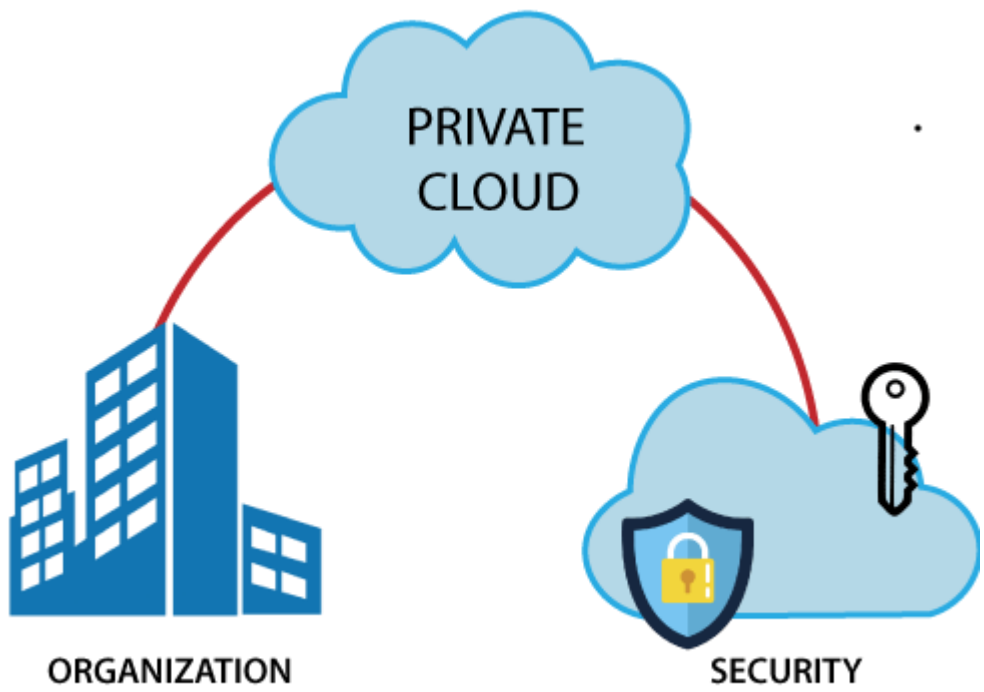
To Read More Click Here

Private Cloud

Private cloud is also known as an **internal cloud** or **corporate cloud**. It is used by organizations to build and manage their own data centers internally or by the third party. It can be deployed using Opensource tools such as Openstack and Eucalyptus.

Based on the location and management, National Institute of Standards and Technology (NIST) divide private cloud into the following two parts-

- On-premise private cloud
- Outsourced private cloud



Advantages of Private Cloud

There are the following advantages of the Private Cloud -

- Private cloud provides a high level of security and privacy to the users.
- Private cloud offers better performance with improved speed and space capacity.
- It allows the IT team to quickly allocate and deliver on-demand IT resources.
- The organization has full control over the cloud because it is managed by the organization itself. So, there is no need for the organization to depends on anybody.
- It is suitable for organizations that require a separate cloud for their personal use and data security is the first priority.

Disadvantages of Private Cloud

- Skilled people are required to manage and operate cloud services.

- Private cloud is accessible within the organization, so the area of operations is limited.
- Private cloud is not suitable for organizations that have a high user base, and organizations that do not have the prebuilt infrastructure, sufficient manpower to maintain and manage the cloud.

To Read More Click Here

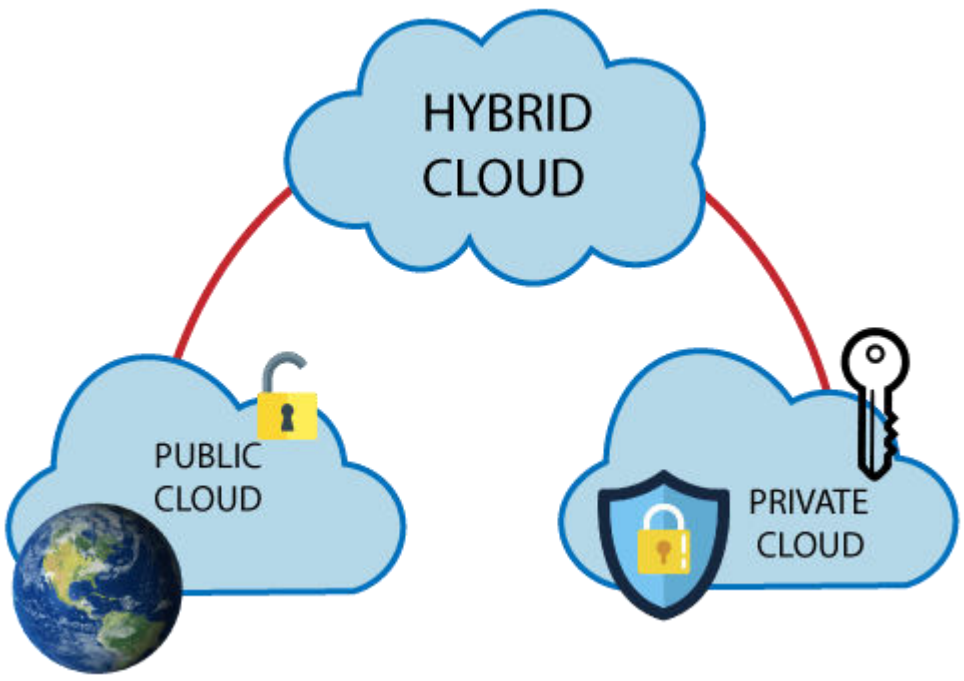
Hybrid Cloud

Hybrid Cloud is a combination of the public cloud and the private cloud. we can say:

Hybrid Cloud = Public Cloud + Private Cloud

Hybrid cloud is partially secure because the services which are running on the public cloud can be accessed by anyone, while the services which are running on a private cloud can be accessed only by the organization's users.

Example: Google Application Suite (Gmail, Google Apps, and Google Drive), Office 365 (MS Office on the Web and One Drive), Amazon Web Services.



Advantages of Hybrid Cloud

There are the following advantages of Hybrid Cloud -

- Hybrid cloud is suitable for organizations that require more security than the public cloud.
- Hybrid cloud helps you to deliver new products and services more quickly.
- Hybrid cloud provides an excellent way to reduce the risk.
- Hybrid cloud offers flexible resources because of the public cloud and secure resources because of the private cloud.

Disadvantages of Hybrid Cloud

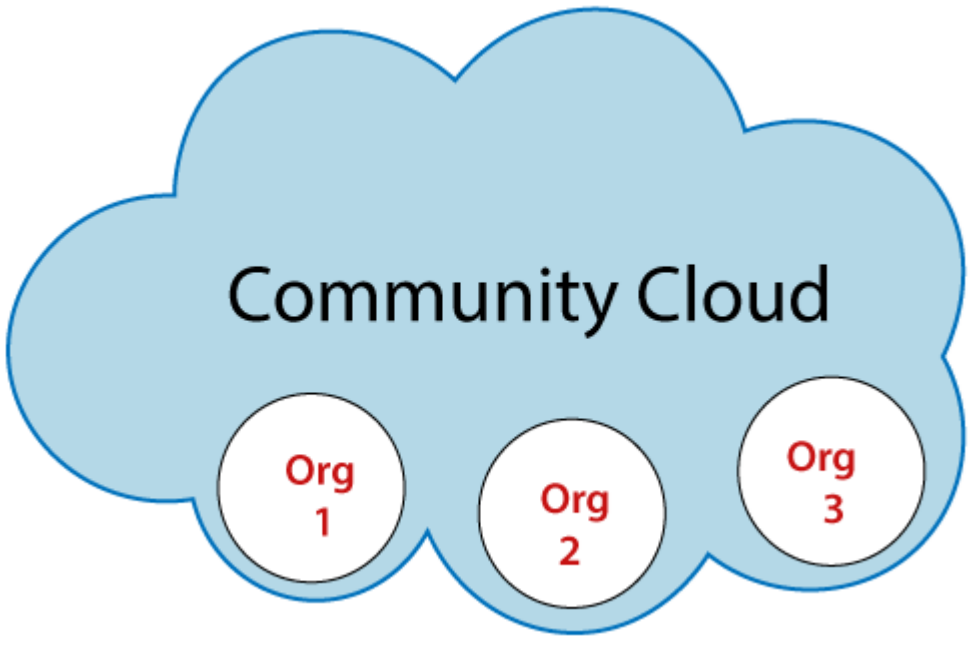
- In Hybrid Cloud, security feature is not as good as the private cloud.
- Managing a hybrid cloud is complex because it is difficult to manage more than one type of deployment model.
- In the hybrid cloud, the reliability of the services depends on cloud service providers.

To Read More Click Here

Community Cloud

Community cloud allows systems and services to be accessible by a group of several organizations to share the information between the organization and a specific community. It is owned, managed, and operated by one or more organizations in the community, a third party, or a combination of them.

Example: Health Care community cloud



Advantages of Community Cloud

There are the following advantages of Community Cloud -

- Community cloud is cost-effective because the whole cloud is being shared by several organizations or communities.
- Community cloud is suitable for organizations that want to have a collaborative cloud with more security features than the public cloud.
- It provides better security than the public cloud.
- It provdes collaborative and distributive environment.
- Community cloud allows us to share cloud resources, infrastructure, and other capabilities among various organizations.

Disadvantages of Community Cloud

- Community cloud is not a good choice for every organization.
- Security features are not as good as the private cloud.
- It is not suitable if there is no collaboration.
- The fixed amount of data storage and bandwidth is shared among all community members.

To Read More Click Here

Difference between public cloud, private cloud, hybrid cloud, and community cloud -

The below table shows the difference between public cloud, private cloud, hybrid cloud, and community cloud.

Parameter	Public Cloud	Private Cloud	Hybrid Cloud	Community Cloud
Host	Service provider	Enterprise (Third party)	Enterprise (Third party)	Community (Third party)
Users	General public	Selected users	Selected users	Community members
Access	Internet	Internet, VPN	Internet, VPN	Internet, VPN
Owner	Service provider	Enterprise	Enterprise	Community

Public Cloud Providers (AWS, MS Azure, GCP)

Amazon Web Services (AWS)

Amazon Web Services (AWS) is a cloud computing platform which was introduced in 2002. It offers a wide range of cloud services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

AWS provides the largest community with millions of active customers as well as thousands of partners globally. Most of the organizations use AWS to expand their business by moving their IT management to the AWS.

Flexibility, security, scalability, and better performance are some important features of AWS.

Microsoft Azure

Microsoft Azure is also called as **Windows Azure**. It is a worldwide cloud platform which is used for building, deploying, and managing services. It supports multiple programming languages such as **Java, Nodejs, C**, and **C#**. The advantage of using Microsoft Azure is that it allows us to a wide variety of services without arranging and purchasing additional hardware components.

Microsoft Azure provides several computing services, including servers, storage, databases, software, networking, and analytics over the Internet.

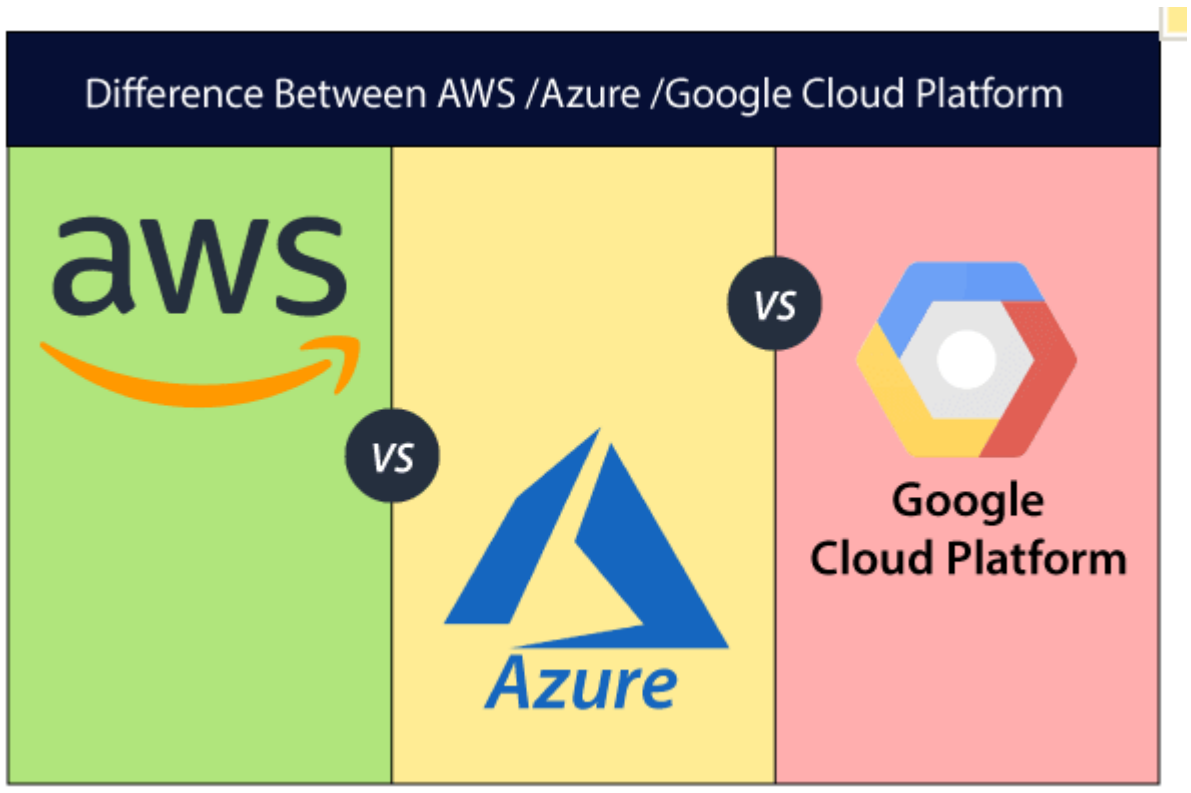
Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is introduced by **Google** in 2011. It allows us to use Google's products such as **Google search engine, Gmail, YouTube**, etc. Most of the companies use this platform to easily build, move, and deploy applications on the cloud. It allows us to access these applications using a high-speed internet connection. The advantage of GCP is that it supports various databases such as **SQL, MYSQL, Oracle, Sam**, and **more**.

Google Cloud Platform (GCP) provides various cloud computing services, including computing, data analytics, data storage, and machine learning.

Difference between AWS, Azure, and Google Cloud Platform (GCP)

Although AWS, Microsoft Azure, and Google cloud platforms offer various high-level features in terms of computing, management, storage, and other services, but there are also some differences between these three.



The below table shows the difference between AWS, Azure, and Google Cloud Platform -

Parameter	AWS	Azure	Google Cloud Platform
App Testing	It uses device farm	It uses DevTest labs	It uses Cloud Test labs.
API Management	Amazon API gateway	Azure API gateway	Cloud endpoints.
Kubernetes Management	EKS	Kubernetes service	Kubernetes engine

Git Repositories	AWS source repositories	Azure source repositories	Cloud source repositories.
Data warehouse	Redshift	SQL warehouse	Big Query
Object Storage	S3	Block Blobs and files	Google cloud storage.
Relational DB	RDS	Relational DBs	Google Cloud SQL
Block Storage	EBS	Page Blobs	Persistent disks
Marketplace	AWS	Azure	G suite
File Storage	EFS	Azure Files	ZFS and Avere
Media Services	Amazon Elastic transcoder	Azure media services	Cloud video intelligence API
Virtual network	VPC	VNet	Subnet
Pricing	Per hour	Per minute	Per minute
Maximum processors in VM	128	128	96
Maximum memory in VM (GiB)	3904	3800	1433
Catching	ElasticCache	RedisCache	CloudCDN
Load Balancing Configuration	Elastic Load Balancing	Load Balancer Application Gateway	Cloud Load Balancing
Global Content Delivery Networks	CloudFront	Content Delivery Network	Cloud Interconnect

2.AWS (Amazon Web Services)

Introduction to AWS

AWS stands for Amazon Web Services which uses distributed IT infrastructure to provide different IT resources on demand.

Our AWS tutorial includes all the topics such as introduction, history of aws, global infrastructure, features of aws, IAM, Storage services, Database services, etc.

What is AWS?

- AWS stands for **Amazon Web Services**.
- The AWS service is provided by the Amazon that uses distributed IT infrastructure to provide different IT resources available on demand. It provides different services such as infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS).
- Amazon launched AWS, a cloud computing platform to allow the different organizations to take advantage of reliable IT infrastructure.

Uses of AWS

- A small manufacturing organization uses their expertise to expand their business by leaving their IT management to the AWS.
- A large enterprise spread across the globe can utilize the AWS to deliver the training to the distributed workforce.
- An architecture consulting company can use AWS to get the high-compute rendering of construction prototype.
- A media company can use the AWS to provide different types of content such as ebox or audio files to the worldwide files.

Pay-As-You-Go

Based on the concept of Pay-As-You-Go, AWS provides the services to the customers.

AWS provides services to customers when required without any prior commitment or upfront investment. Pay-As-You-Go enables the customers to procure services from AWS.

- Computing
- Programming models
- Database storage
- Networking

Advantages of AWS

1) Flexibility

- We can get more time for core business tasks due to the instant availability of new features and services in AWS.
- It provides effortless hosting of legacy applications. AWS does not require learning new technologies and migration of applications to the AWS provides the advanced computing and efficient storage.
- AWS also offers a choice that whether we want to run the applications and services together or not. We can also choose to run a part of the IT infrastructure in AWS and the remaining part in data centres.

2) Cost-effectiveness

AWS requires no upfront investment, long-term commitment, and minimum expense when compared to traditional IT infrastructure that requires a huge investment.

3) Scalability/Elasticity

Through AWS, autoscaling and elastic load balancing techniques are automatically scaled up or down, when demand increases or decreases respectively. AWS techniques are ideal for handling unpredictable or very high loads. Due to this reason, organizations enjoy the benefits of reduced cost and increased user satisfaction.

4) Security

- AWS provides end-to-end security and privacy to customers.
- AWS has a virtual infrastructure that offers optimum availability while managing full privacy and isolation of their operations.
- Customers can expect high-level of physical security because of Amazon's several years of experience in designing, developing and maintaining large-scale IT operation centers.
- AWS ensures the three aspects of security, i.e., Confidentiality, integrity, and availability of user's data.

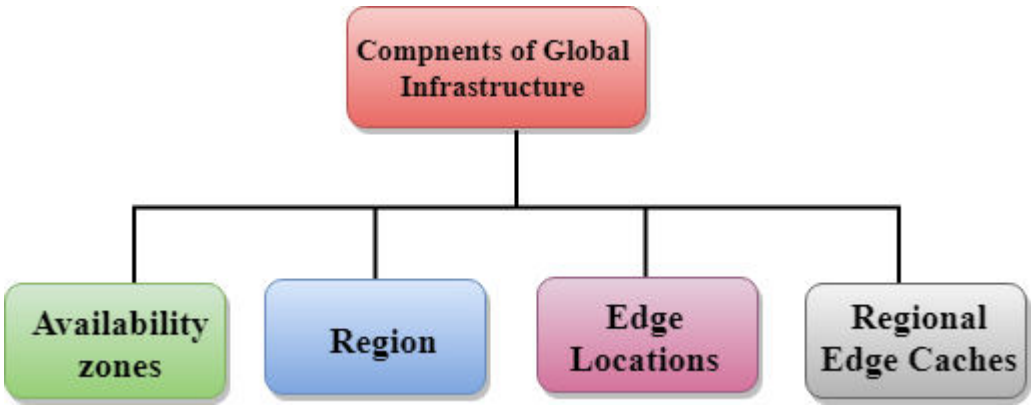
Global Infrastructure

- AWS is a cloud computing platform which is globally available.
- Global infrastructure is a region around the world in which AWS is based. Global infrastructure is a bunch of high-level IT services which is shown below:
- AWS is available in 19 regions, and 57 availability zones in December 2018 and 5 more regions 15 more availability zones for 2019.

The following are the components that make up the AWS infrastructure:

- Availability Zones

- Region
- Edge locations
- Regional Edge Caches

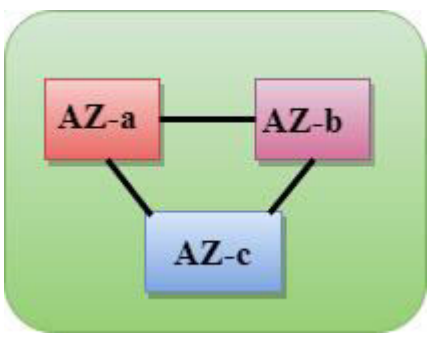


Availability zone as a Data Center

- An availability zone is a facility that can be somewhere in a country or in a city. Inside this facility, i.e., Data Centre, we can have multiple servers, switches, load balancing, firewalls. The things which interact with the cloud sits inside the data centers.
- An availability zone can be a several data centers, but if they are close together, they are counted as 1 availability zone.

Region

- A region is a geographical area. Each region consists of 2 more availability zones.
- A region is a collection of data centers which are completely isolated from other regions.
- A region consists of more than two availability zones connected to each other through links.



- Availability zones are connected through redundant and isolated metro fibers.

Edge Locations

- Edge locations are the endpoints for AWS used for caching content.
- Edge locations consist of CloudFront, Amazon's Content Delivery Network (CDN).
- Edge locations are more than regions. Currently, there are over 150 edge locations.
- Edge location is not a region but a small location that AWS have. It is used for caching the content.
- Edge locations are mainly located in most of the major cities to distribute the content to end users with reduced latency.
- For example, some user accesses your website from Singapore; then this request would be redirected to the edge location closest to Singapore where cached data can be read.

Regional Edge Cache

- AWS announced a new type of edge location in November 2016, known as a Regional Edge Cache.
- Regional Edge cache lies between CloudFront Origin servers and the edge locations.
- A regional edge cache has a large cache than an individual edge location.
- Data is removed from the cache at the edge location while the data is retained at the Regional Edge Caches.
- When the user requests the data, then data is no longer available at the edge location. Therefore, the edge location retrieves the cached data from the Regional edge cache instead of the Origin servers that have high latency.

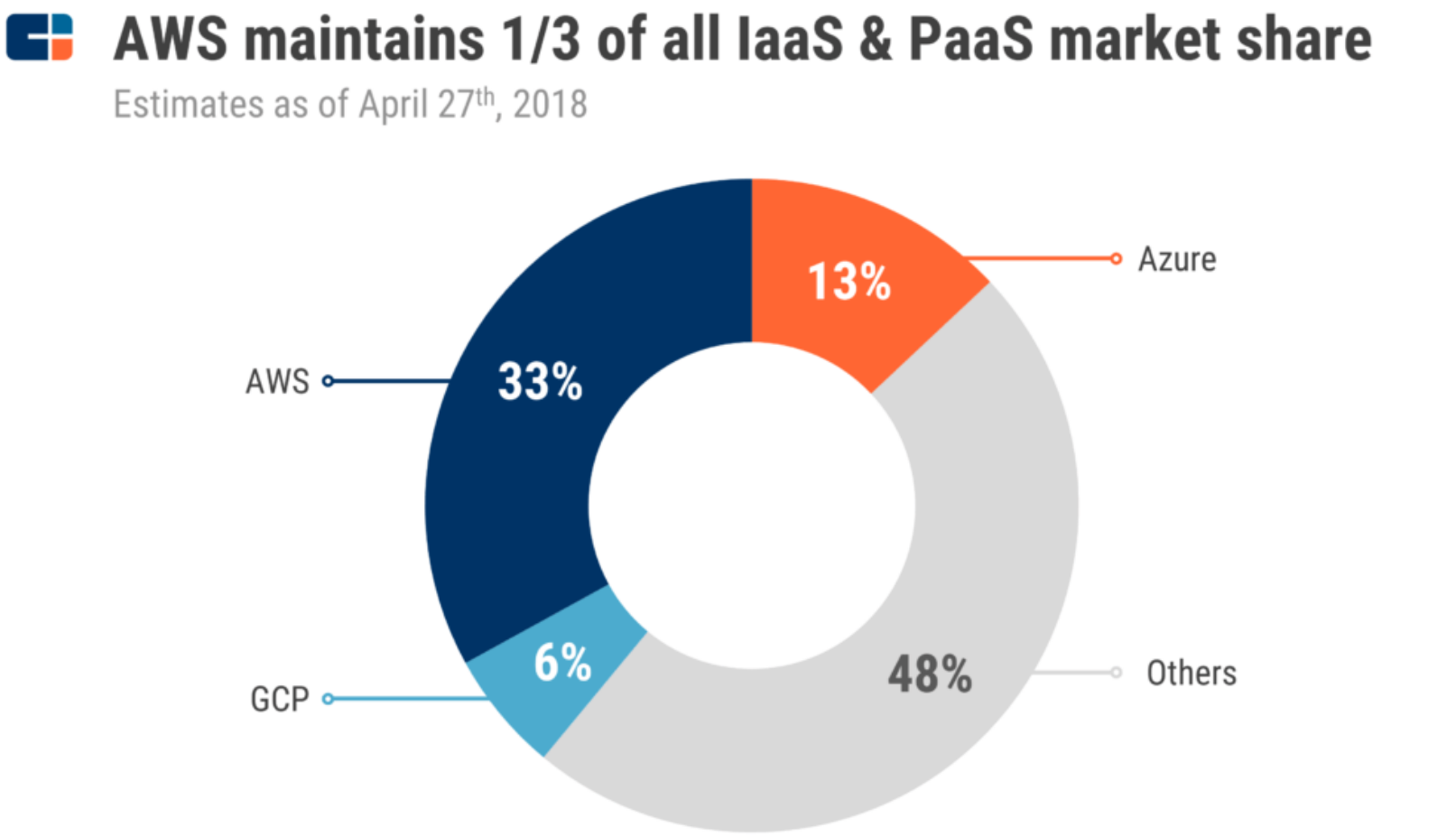
Free tier overview & limitations

he entire cloud platform has come a long way since its' inception and is now a critical part of today's IT landscape.

Cloud computing comes into focus only when you think about what IT always needs: a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. Cloud computing encompasses any subscription-based or pay-per-use service that you access in real-time over the Internet.

Some of the more popular use cases for the cloud:

- **Saas (Software as a Service)**: Delivering a single application through a web browser to thousands of customers using a multitenant architecture.
- **Utility Computing**: This form of cloud computing is getting new life from Amazon, Sun, IBM, and others who now offer storage and virtual servers that IT can access on demand.
- **Web Services in the Cloud**: Closely related to SaaS, Web service providers offer APIs that enable developers to exploit functionality over the Internet, rather than delivering full-blown applications
- **Platform-as-a-Service**: This form of cloud computing delivers development environments as a service. You build your own applications that run on the provider's infrastructure and are delivered to your users via the Internet from the provider's servers.
- **MSP (Managed Service Providers)**: One of the oldest forms of cloud computing, a managed service is basically an application exposed to IT rather than to end-users, such as a virus scanning service for e-mail or an application monitoring service.
- **Service Commerce Platforms**: A hybrid of SaaS and MSP, this cloud computing service offers a service hub that users interact with. They're most common in trading environments.



- Additionally, Apple has spent $30 Million on Amazon Cloud in Q1 2019. While Slack and Pinterest are in the process of spending a total of $250 million and $750 million on Amazon Cloud respectively.
- Amazon Web Services (AWS) continues to be the star of the show, growing 41% in sales to $7.7 billion in Q1,2019. AWS accounted for about 13% of Amazon's total revenue for the quarter.

Part of the allure of AWS is their metered pay-as-you-use billing philosophy. Requiring you only to pay for what you use, as you use it. Taking this benefit one step further, the AWS Free Usage Tier provides the ability to explore and experiment with AWS services free of charge, up to specified limits for each service.

About AWS Free Tier

The

free tier applies to certain participating AWS services up to a specific maximum amount of usage each month. The AWS Free Usage Tier is comprised of three different types of pricing models, a 12-month Free Tier, an Always Free offer, and short term trials.

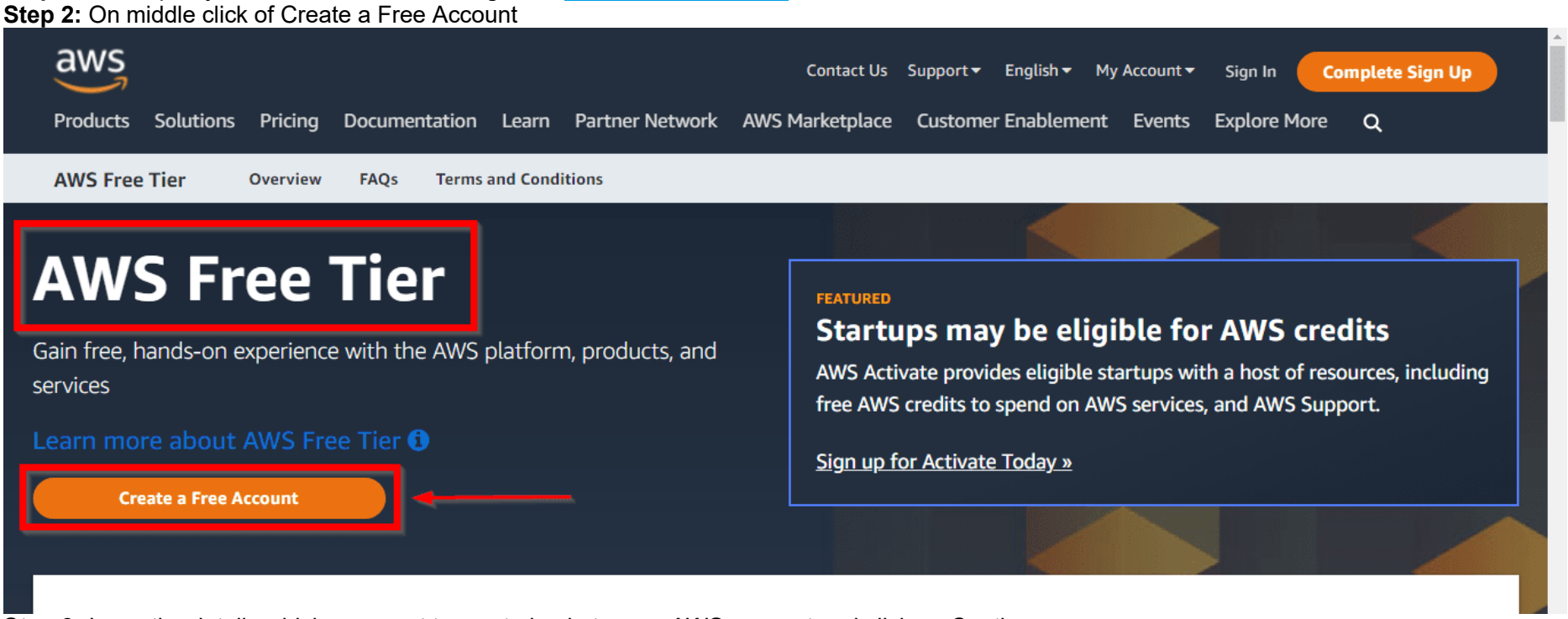
The AWS Free Usage Tier is available to everyone – students, entrepreneurs, small businesses, and Fortune 500 companies. More importantly, the AWS free usage tier is available to new AWS accounts created on or after October 21, 2010.

Services that are available in the AWS Free Usage Tier

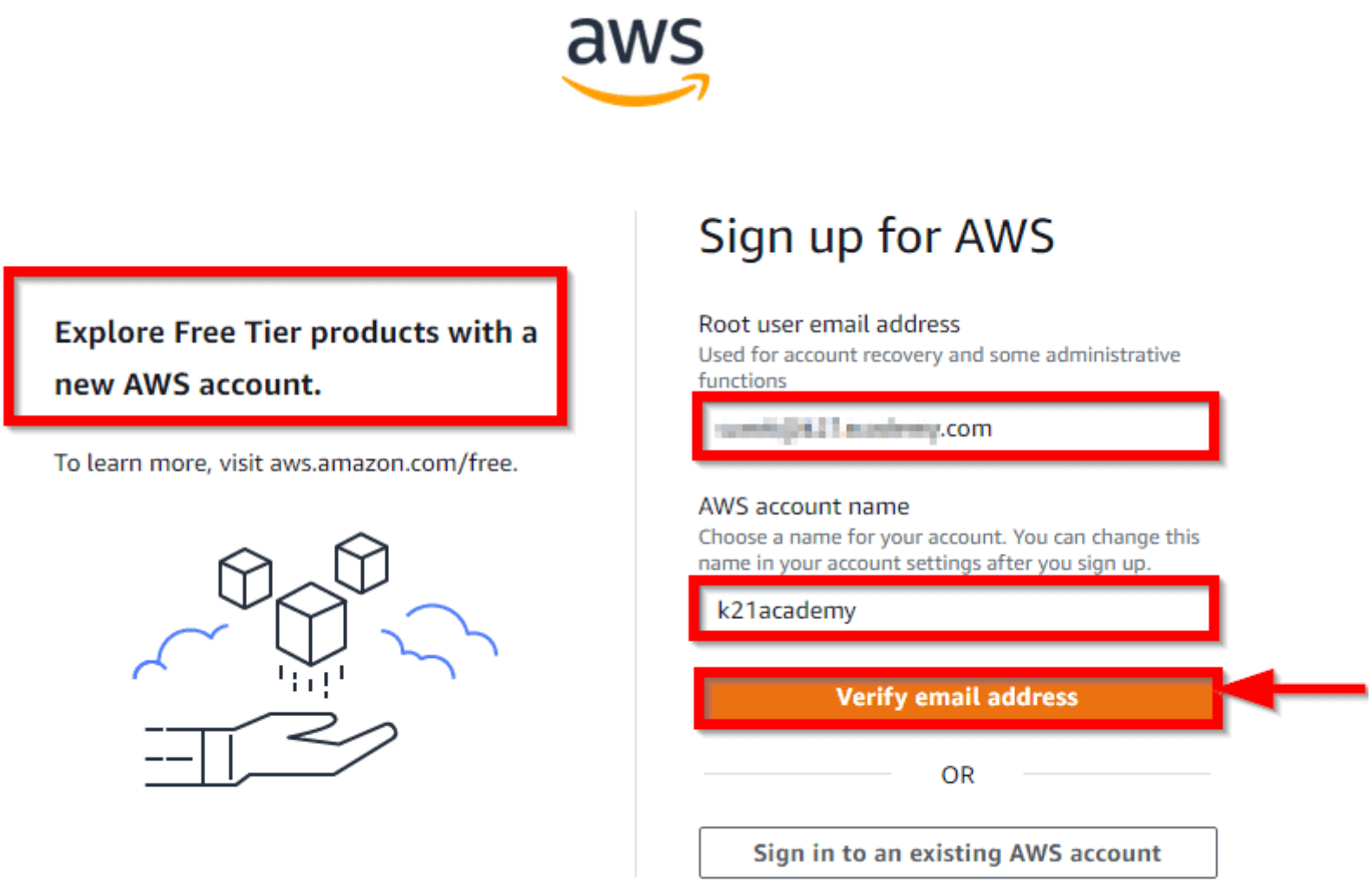
- 750 hours of Amazon EC2 Linux or RHEL or SLES t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support) – enough hours to run continuously each month
- 750 hours of an Elastic Load Balancer plus 15 GB data processing
- 750 hours of Amazon RDS Single-AZ Micro DB Instances, running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

AWS Account creation

Step 1: First Open your web browser and navigate to **AWS Free Tier Page**
Step 2: On middle click of Create a Free Account



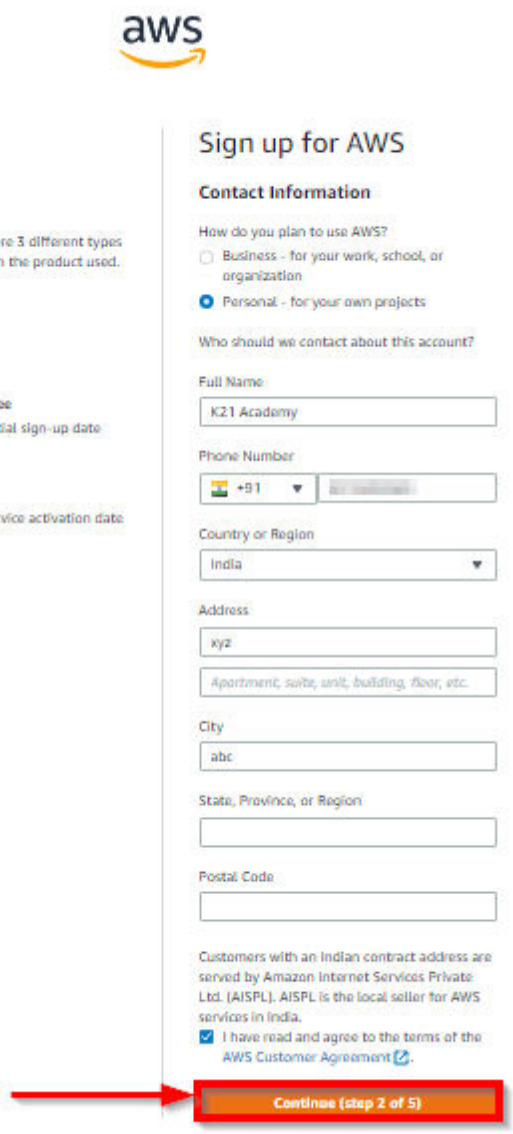
Step 3: Issue the details which you want to use to log in to your AWS account and click on Continue
• **Email address:** Provide the mail id which hasn't been registered yet with Amazon AWS.
• **Password:** Type your password.
• **Confirm password:** Authenticate the password.
• **AWS Account name:** Choose a name for your account. You can change this name in your account settings after you sign up.



Read: [AWS Elastic Beanstalk]

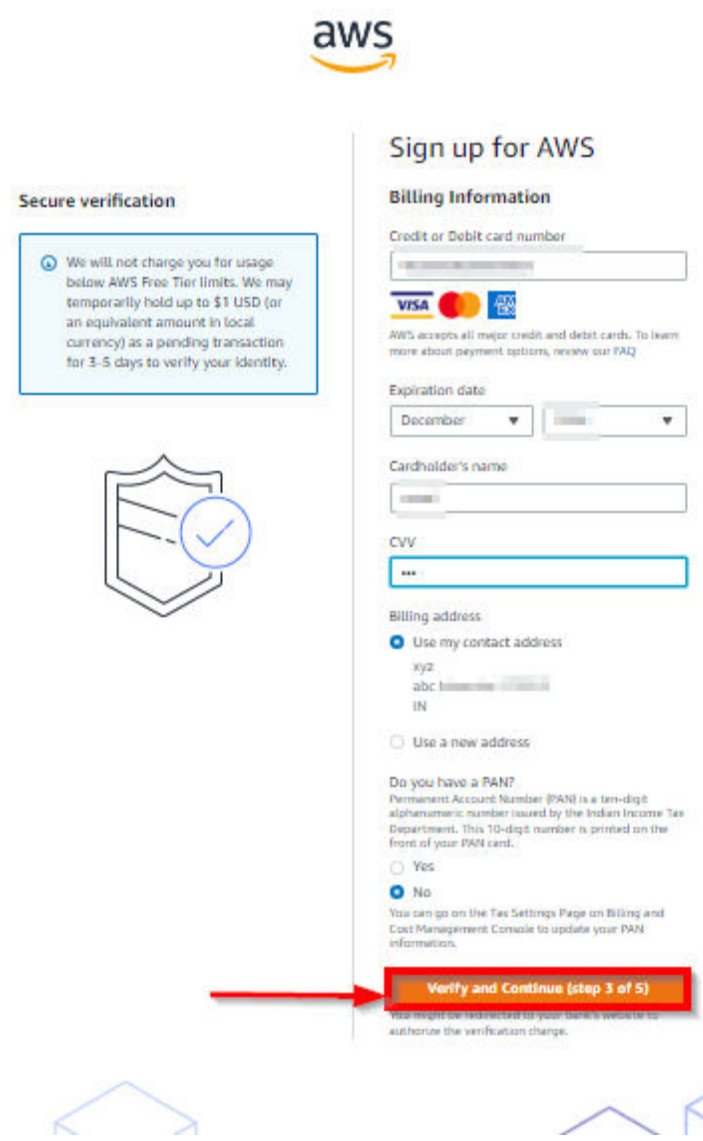
Step 4: Contact Information

Select your AWS type (Profesional/ Personal) Fill in the correct information to validate your account if you're going to create personal use then click on **"Personal Account"** else use **"Company Account"**, Accepts the Terms and condition and then click on Create Account and Continue



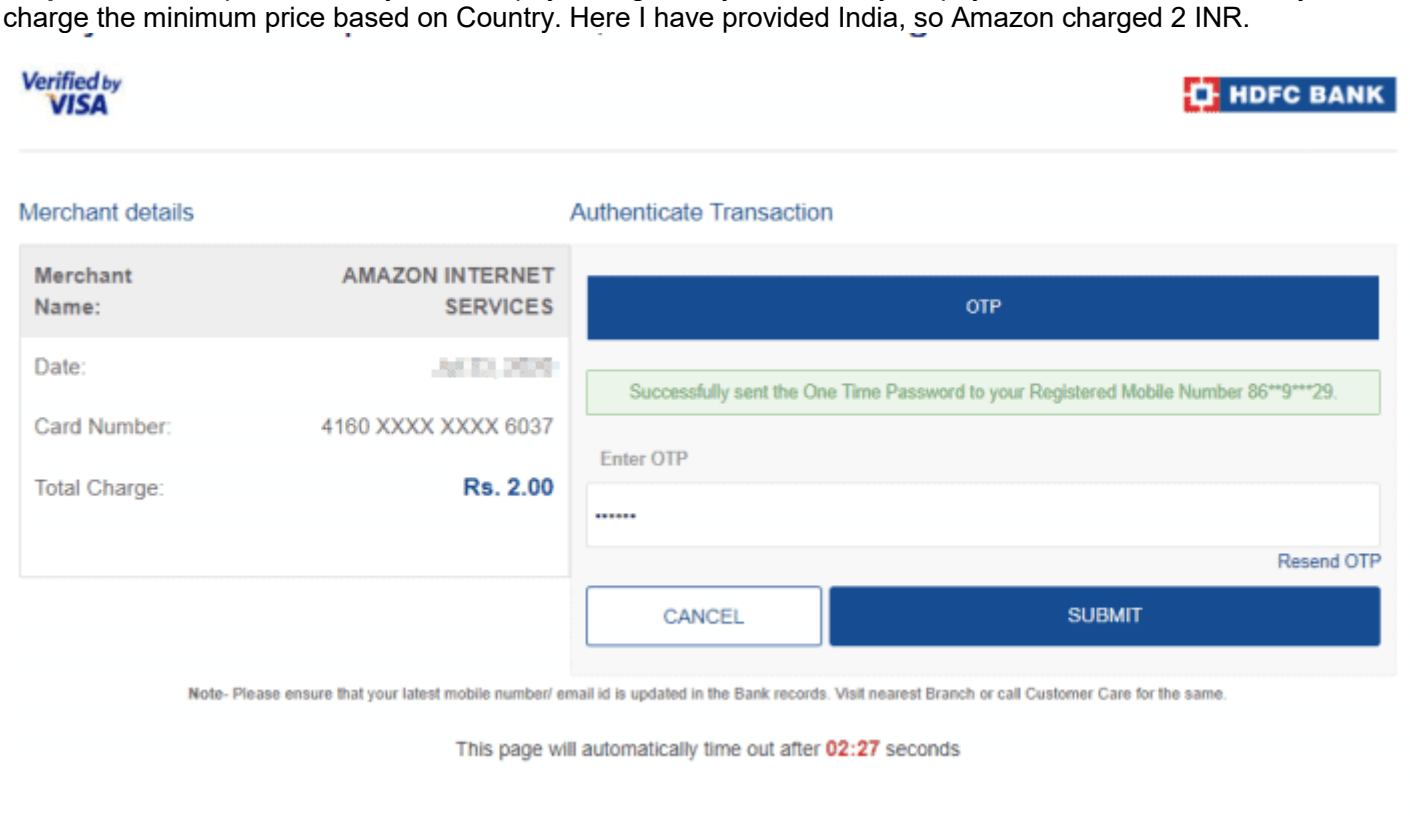
Note: Make sure to provide proper contact details and mobile number to get the Verification code from AWS.

Step 5: Payment and PAN information: In this step, you must fill in your credit card /Debit Card info and billing address and click on Secure Submit.

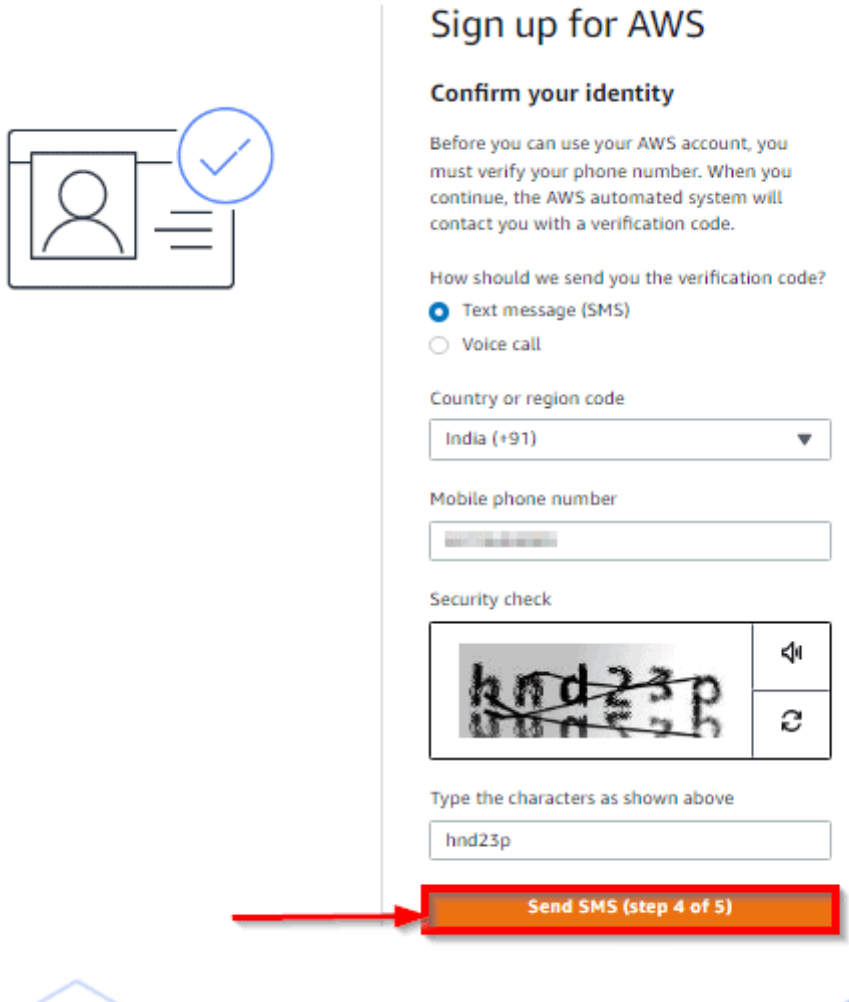


Read AWS CLI Secrets Manager.

Step 6: In this step, it will take you to the payment gateway to validate your payment information and for your credit card verification, Amazon will charge the minimum price based on Country. Here I have provided India, so Amazon charged 2 INR.

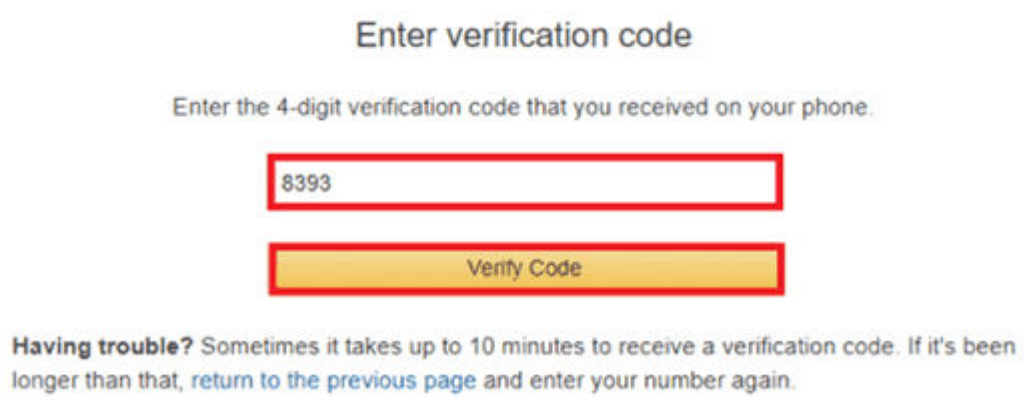


Step 7: Phone verification: Here you will be taken to an identity verification page that will already have your phone number, so you just have to select either “Text message or Voice call” Provide a valid phone number, Solve the captcha, and then click on Send SMS or Call Me Now(depending upon your selection).

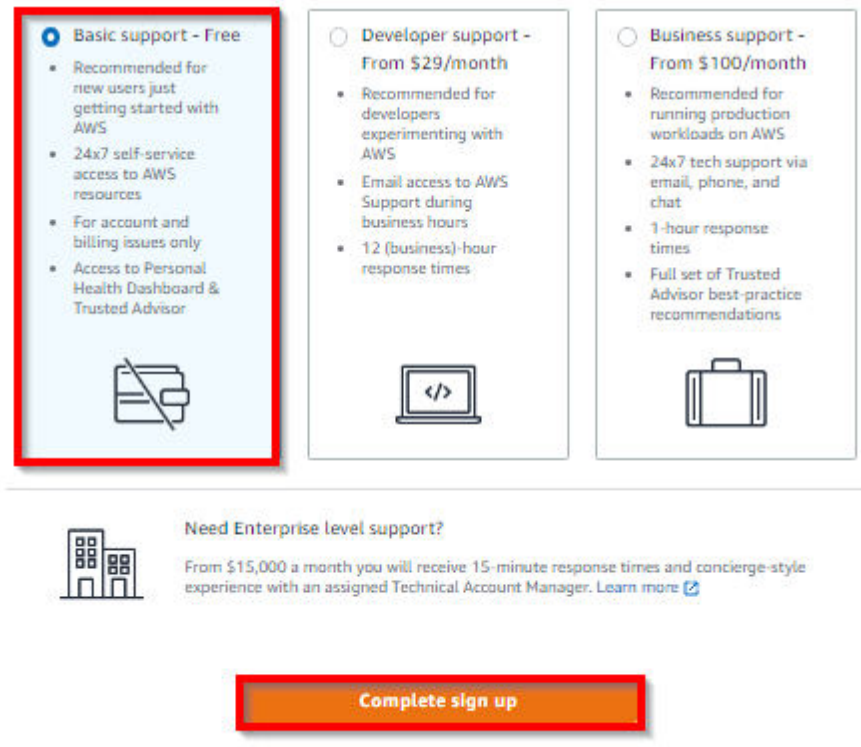


Read: AWS Well-Architected Framework

Step 8: After clicking on Send SMS or Call me Now, you will immediately receive a call or SMS from Amazon, for verification code, Enter your code then click on Verify Code.

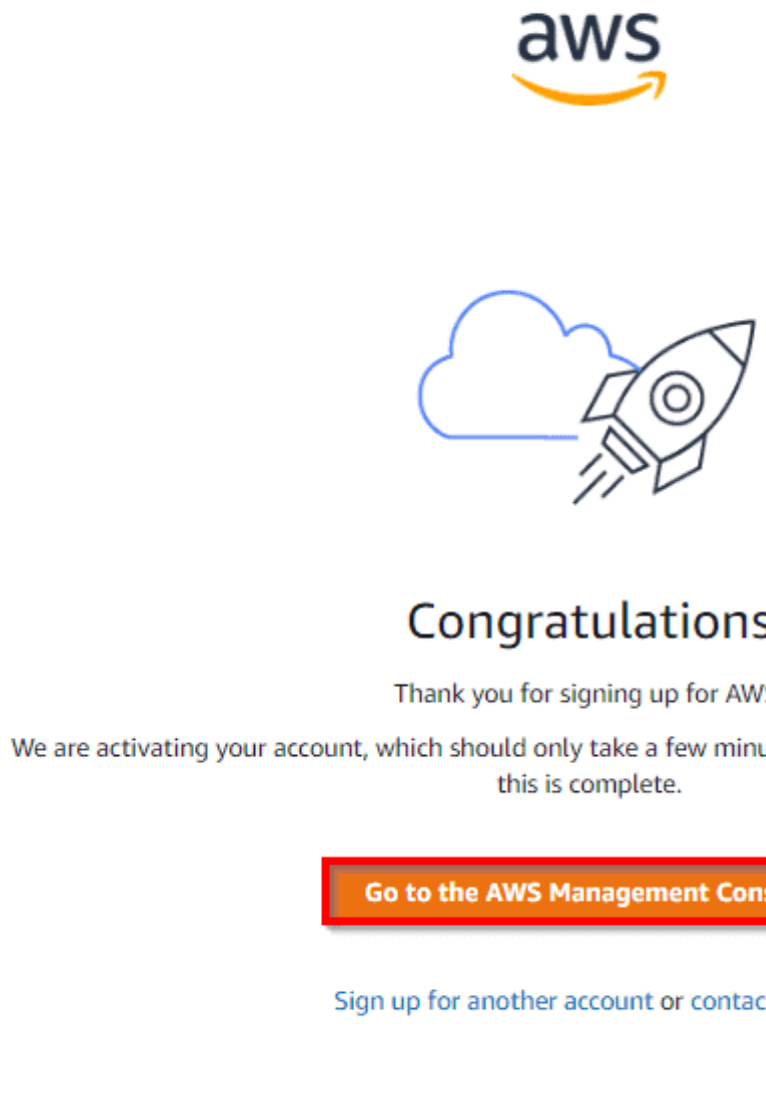


Step 9: Support plan: AWS support offers a selection of plans to meet your business needs.
Select your suitable plan then click continue.



Note: All customers receive free basic support.

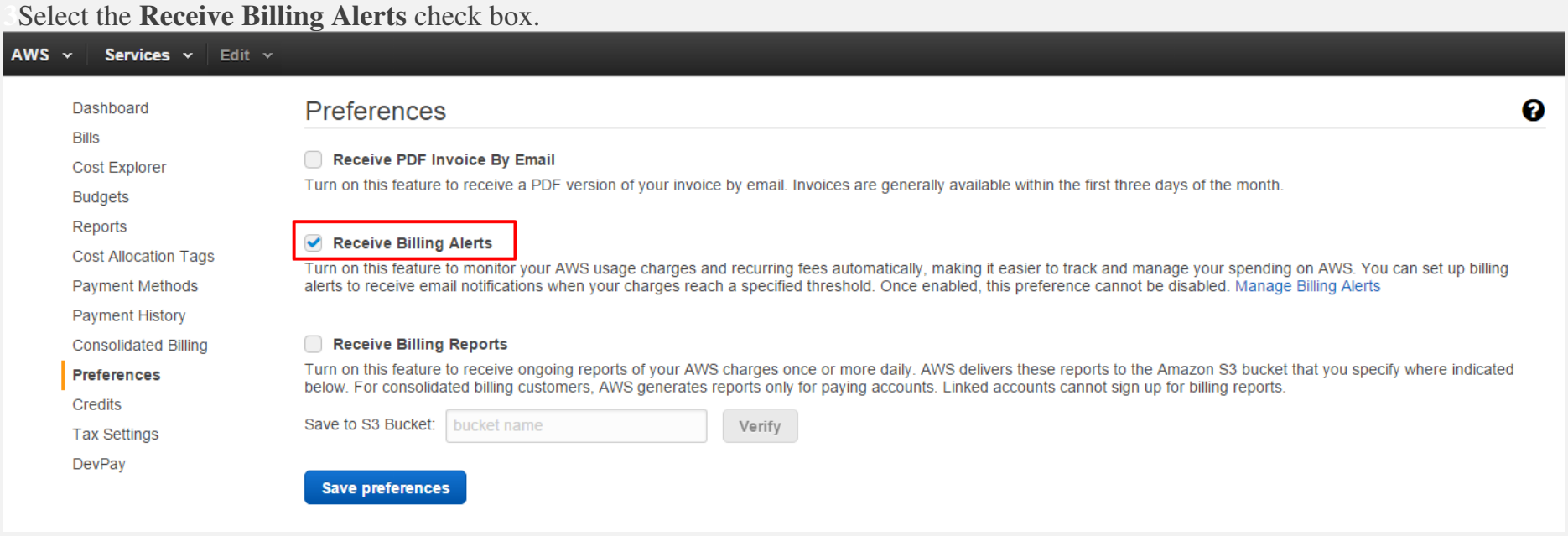
Step 10: Registration Confirmation page.
Once you completed all the above steps and processes. You'll get the confirmation page below. Now your account will be processed for activation. It may take somewhere between 30 minutes to 1 hour for you to receive an email confirmation that your Amazon Cloud Services account has been activated.



Create Billing Alarm for unnecessary billing

Step 1: Enable Billing Alerts for Your Account

1. Sign in to the AWS Management Console and open the Billing and Cost Management Console.
2. On the navigation pane, choose **Preferences.**
3. Select the **Receive Billing Alerts** check box.



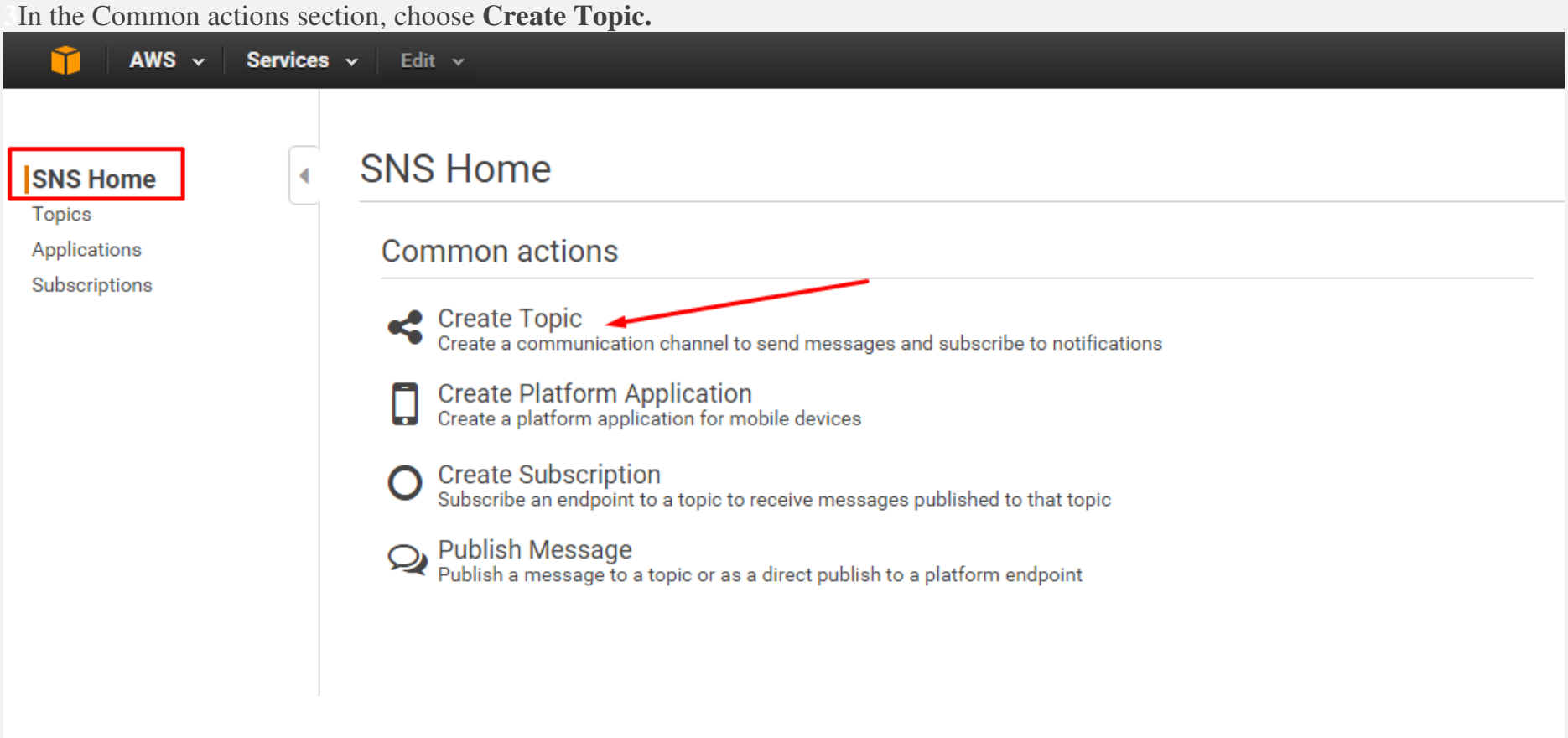
4. Choose **Save Preferences**

Now, you have enabled Billing Alerts for your AWS account. Next step is to create e-mail or SMS notifications that will warn you, if the sum overcomes the desired threshold.

Step 2. Create SNS Topic and Enable Appropriate CloudWatch Alarm

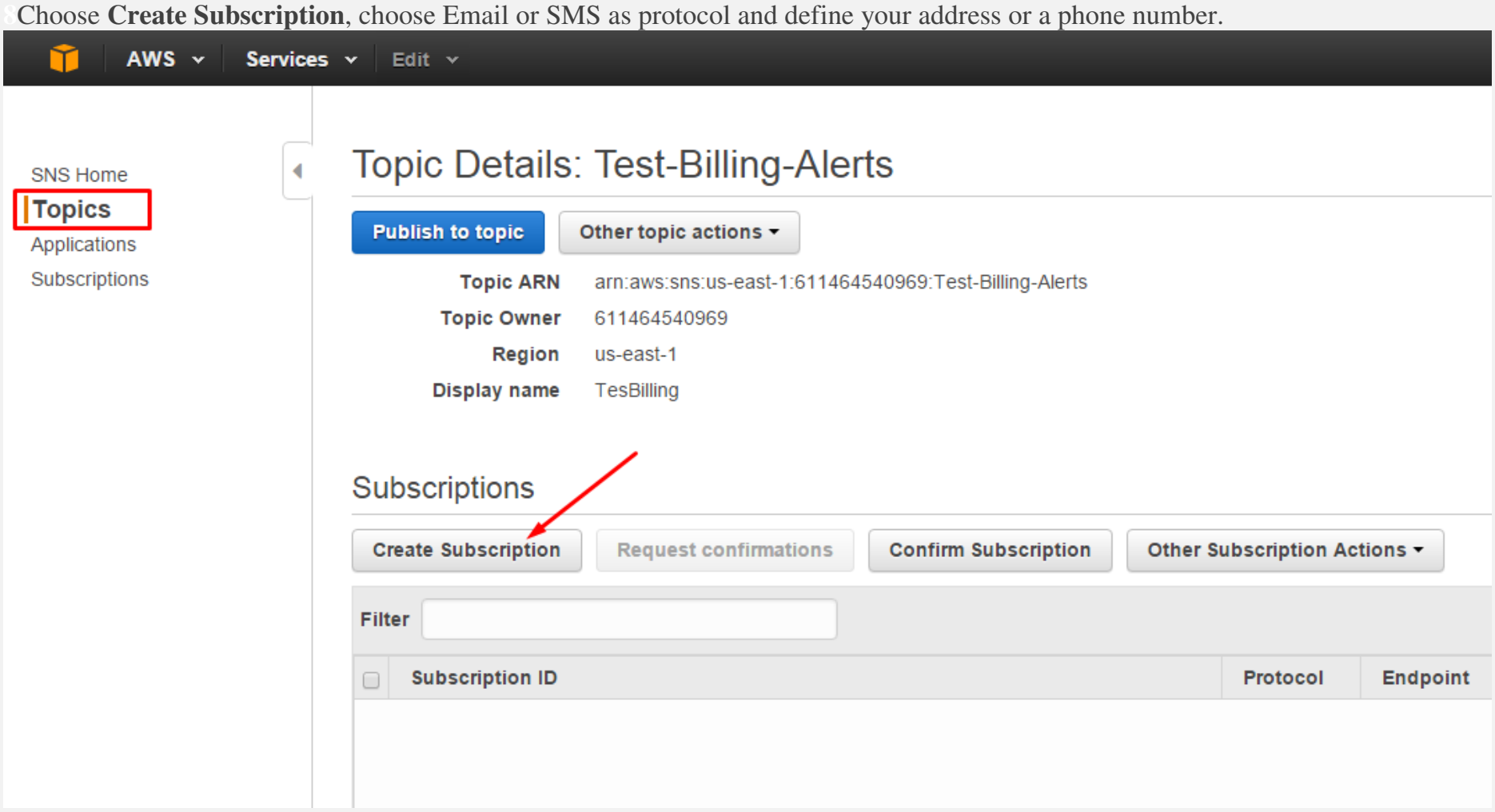
This step is also divided into three consecutive actions. First, you need to create an Amazon SNS notification list:

1. Open the Amazon SNS console.
2. On the navigation pane, choose **SNS Home.**
3. In the Common actions section, choose **Create Topic.**



4. In the dialog box, for **Topic name**, enter the name for your notification list.

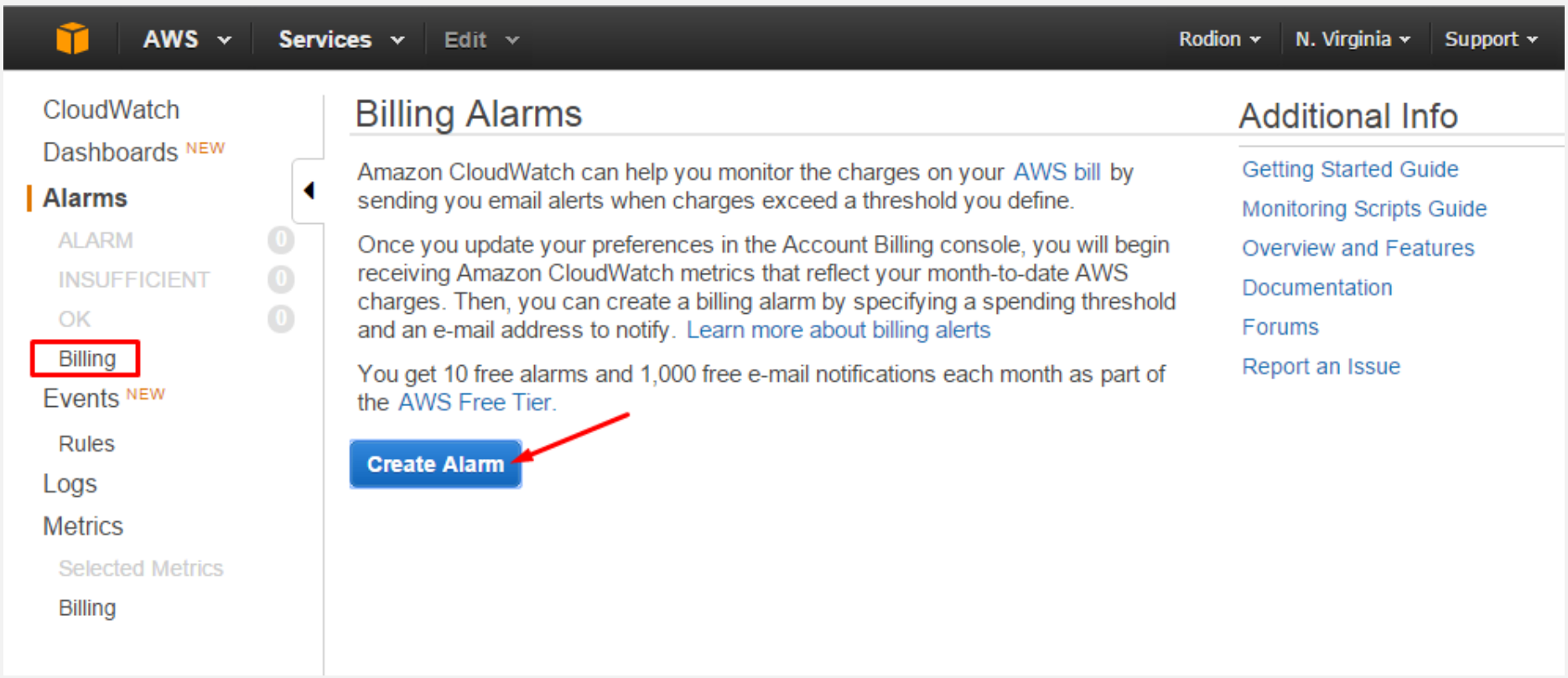
5 (Optional) If you want to use this notification list to send **SMS messages**, for Display name, enter the name you want to appear on your SMS messages.
6 Choose **Create topic.**
7 On the navigation pane, choose **Topics** and choose the topic you've created.
8 Choose **Create Subscription**, choose Email or SMS as protocol and define your address or a phone number.



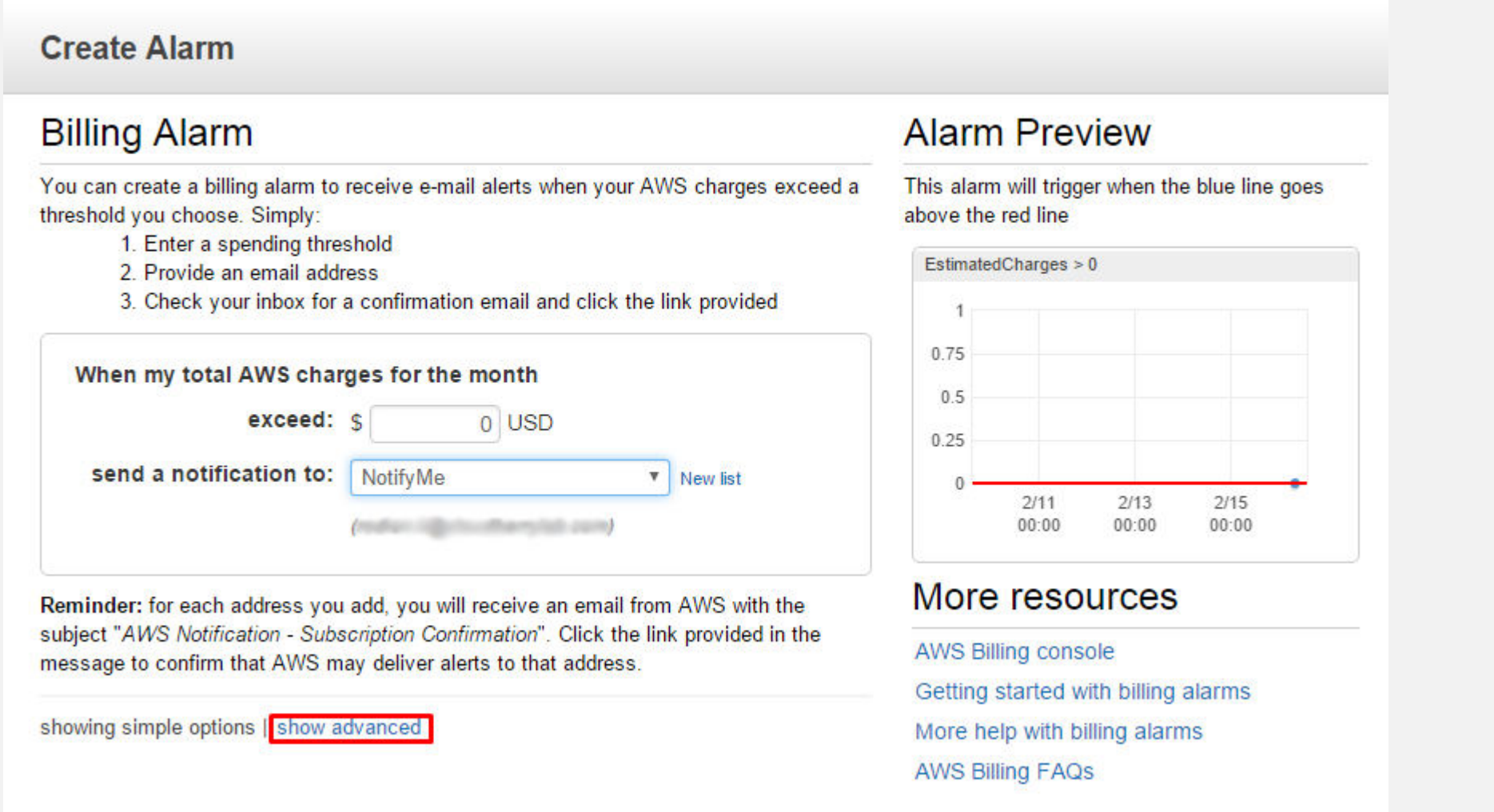
9 Confirm an Email address by following the link in the confirmation letter from AWS.
If you don't confirm the email address, the subscription remains in the pending confirmation status until you do so, and does not send a message.

Enable the CloudWatch Alarm:
1 Open the CloudWatch Console.
2 If necessary, change the region on the navigation bar to **US East (N. Virginia)**. The billing metric data is stored in this region, even for resources in other regions.
3 On the navigation pane, under Metrics, choose **Billing.**

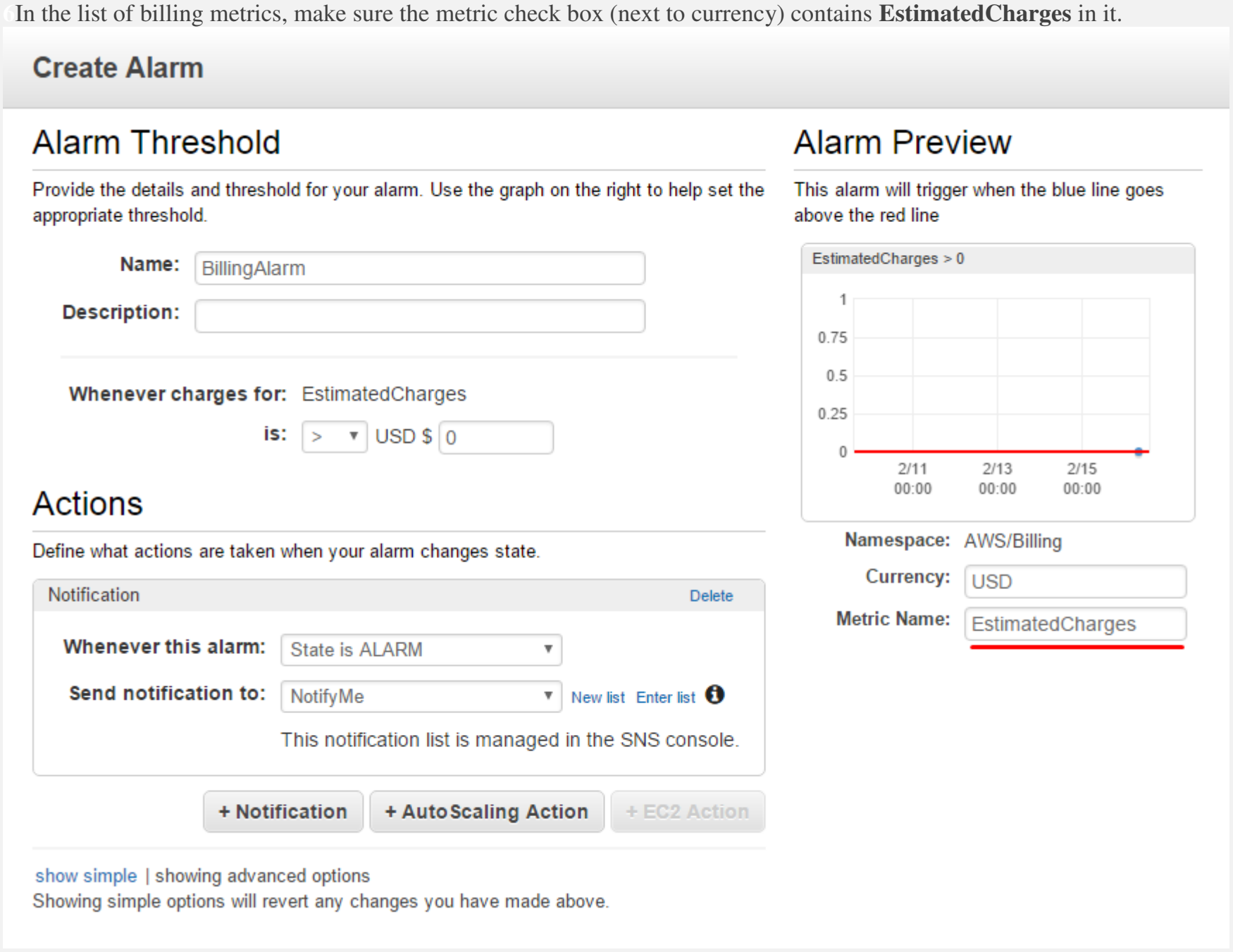
4 Choose **Create Alarm.**



5 In a new window choose **Show advanced** to ensure all metrics are set correctly.



6 In the list of billing metrics, make sure the metric check box (next to currency) contains **EstimatedCharges** in it.



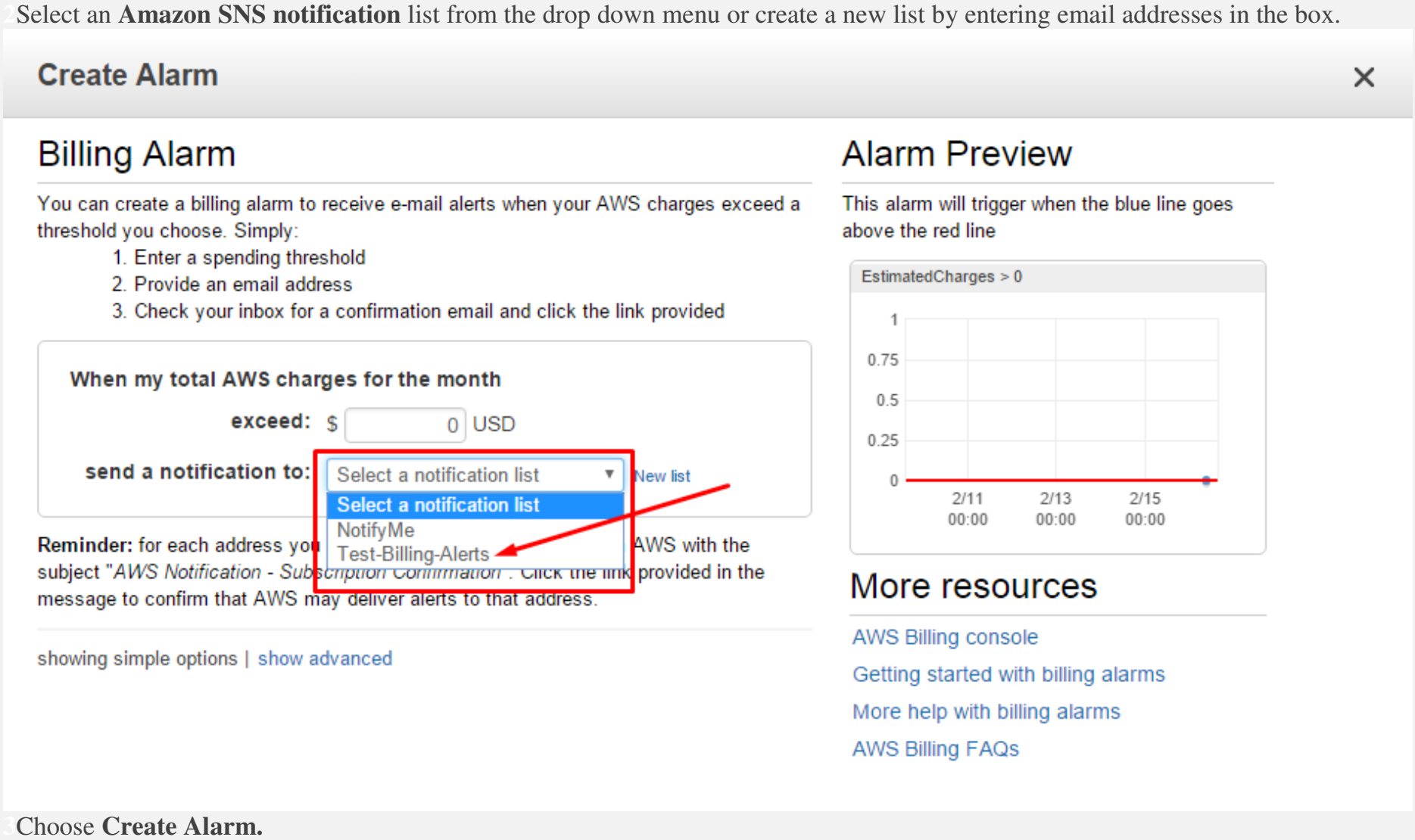
7 Set up the **Name** and (optionally) a description for a new alarm.

Define the Alarm as Follows:

1 Set the desired sum in "when my total AWS charges for the month exceed" check box.

Note: We recommend to set up two alarms: when achieving ½ of the estimated budget and when exceeding the estimated budget.

2 Select an **Amazon SNS notification** list from the drop down menu or create a new list by entering email addresses in the box.

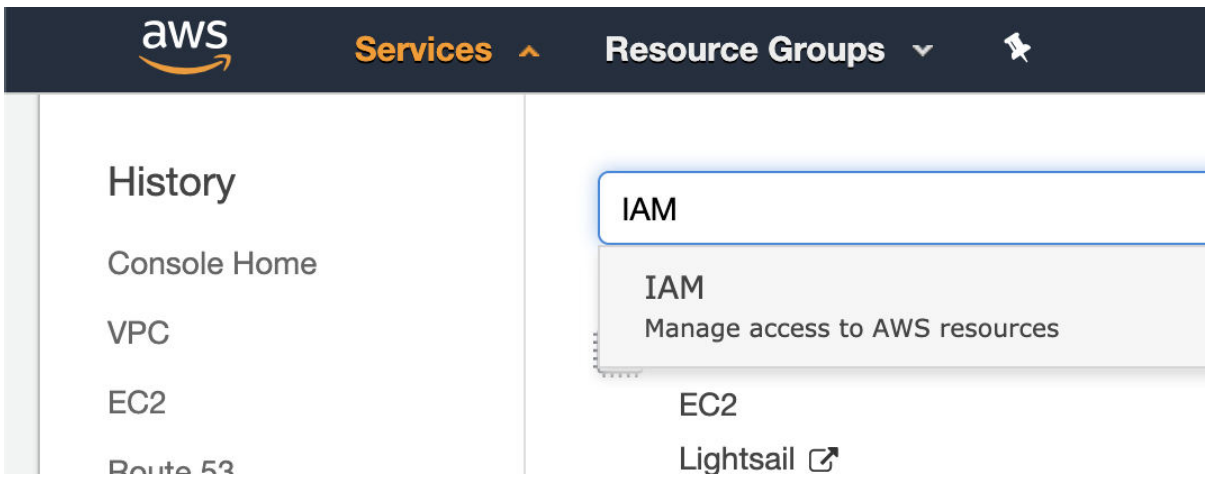


3 Choose **Create Alarm.**

Create IAM User

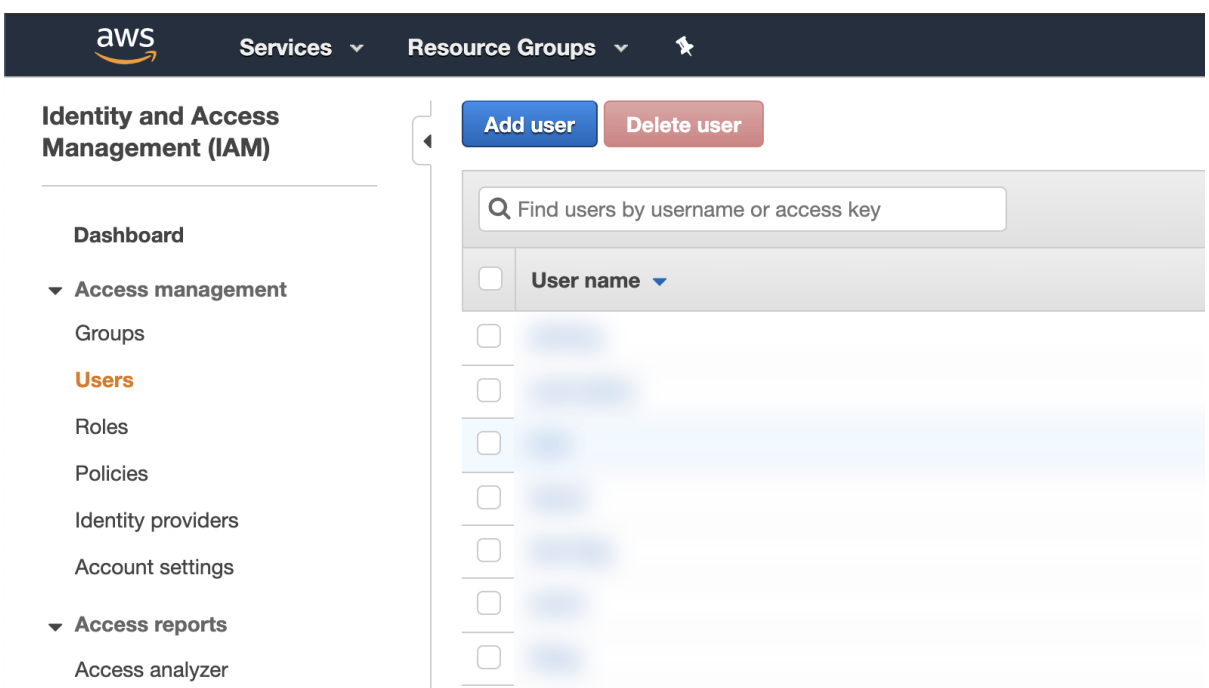
Step 1: AWS Login

Log into your AWS Management Console and select the IAM service.



Step 2: Create a New User

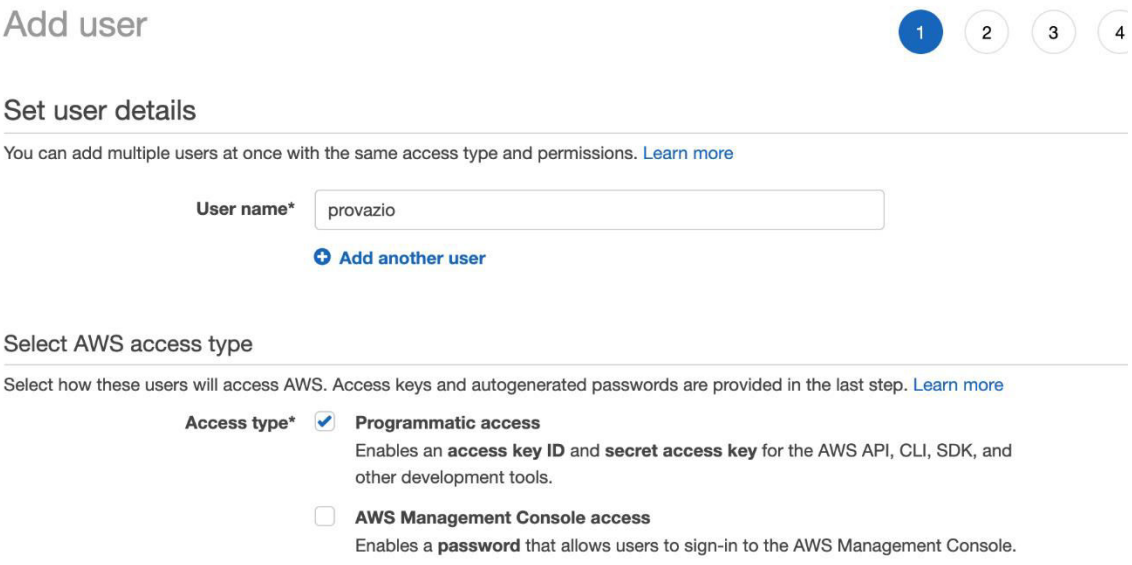
In the side navigation menu, select Access management | Users, and then select Add user.



Step 3: Set the User's Access Permissions and Name

In the Set user details section,

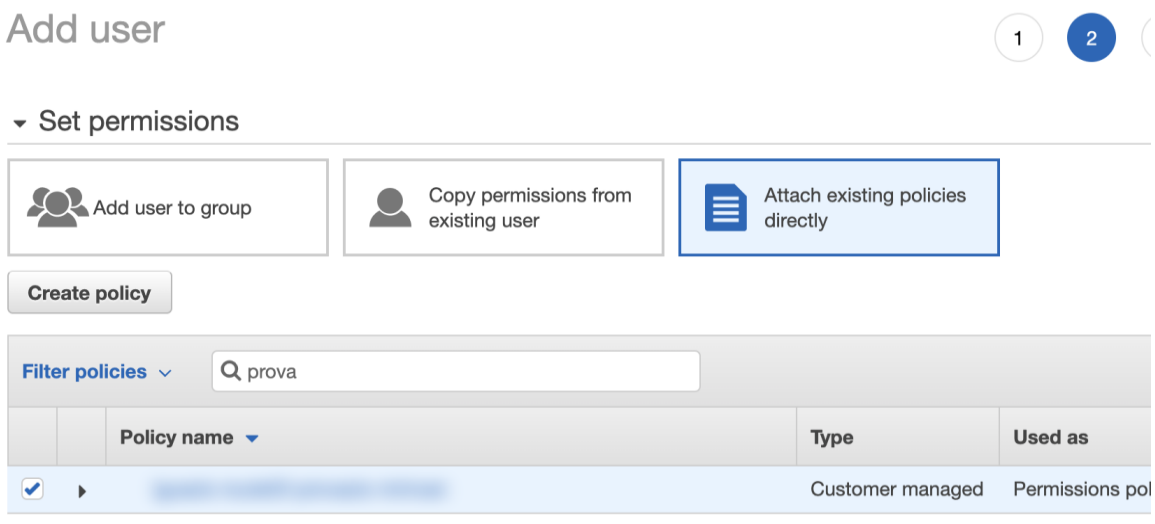
- In the User name field, enter the name of the new user (for example, "Provazio" — recommended).
- In the Access type field, check the Programmatic access option to allow the user only programmatic access.



When you're done, select Next: Permissions.

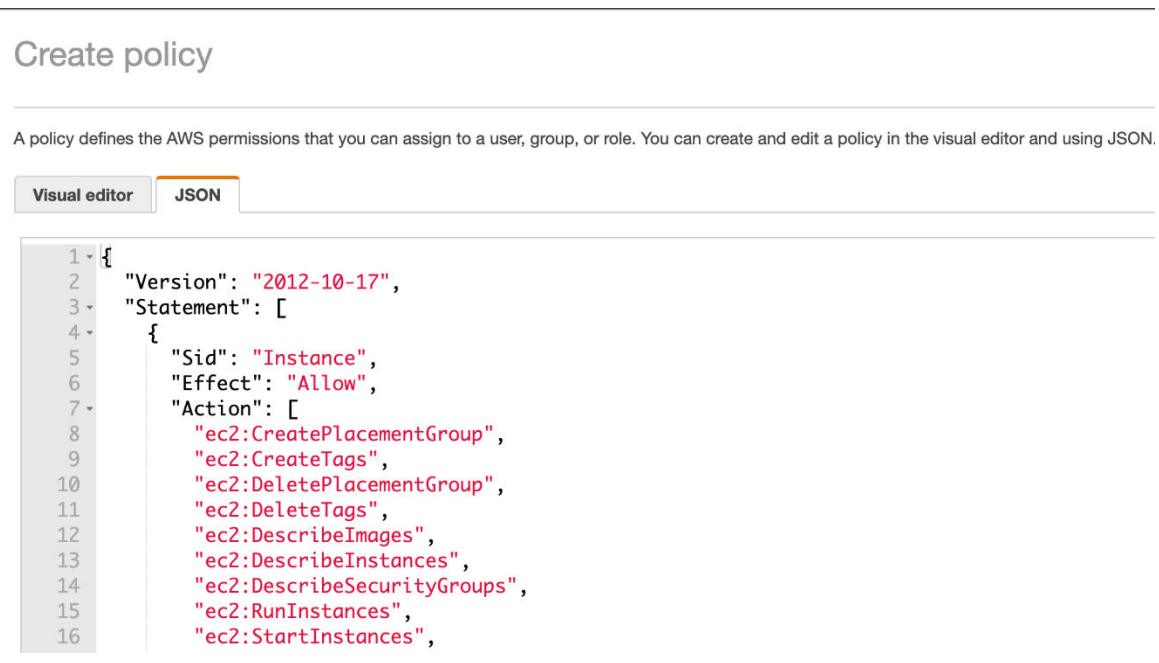
Step 4: Create a Policy

Select Attach existing policies directly, and then select Create policy.



Download the platform IAM policy file provazio-eks.json for an EKS cluster. Edit the file to replace all `$AWS_ACCOUNT_ID` instances with your AWS Account ID.

Paste the contents of your selected policy file in the JSON tab of the AWS Management Console and select Review policy. Give the policy a name (for example, "ManageIguazioSystems" — recommended), optionally add a description, and select Create policy.



Step 5: Create the User

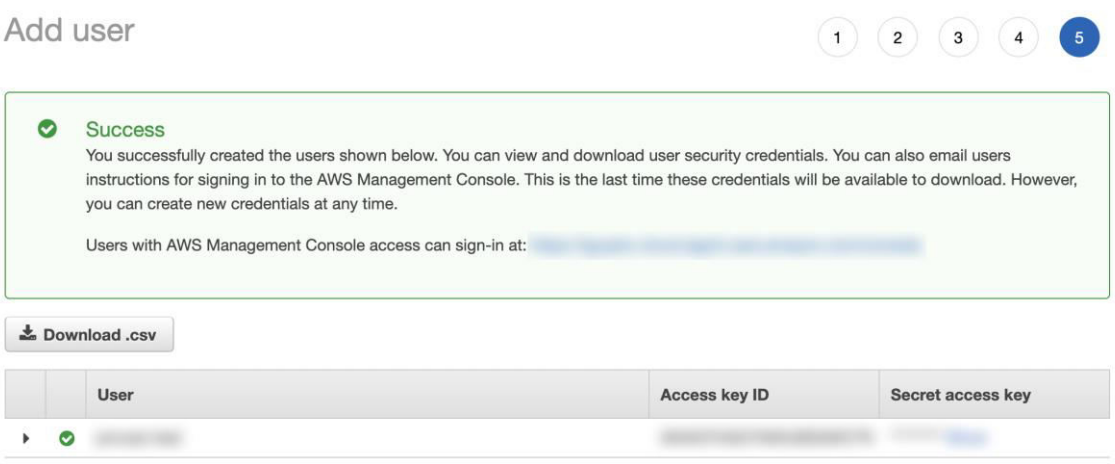
Filter the policies for the name of the policy that you created and select the policy.

Select Next: Tags and optionally assign user tags.

Select Next: Review and review your role definition. When you're ready, select Create user.

Step 6: Save the User Credential

Download and save the credentials of the new user (Access key iD and Secret access key).



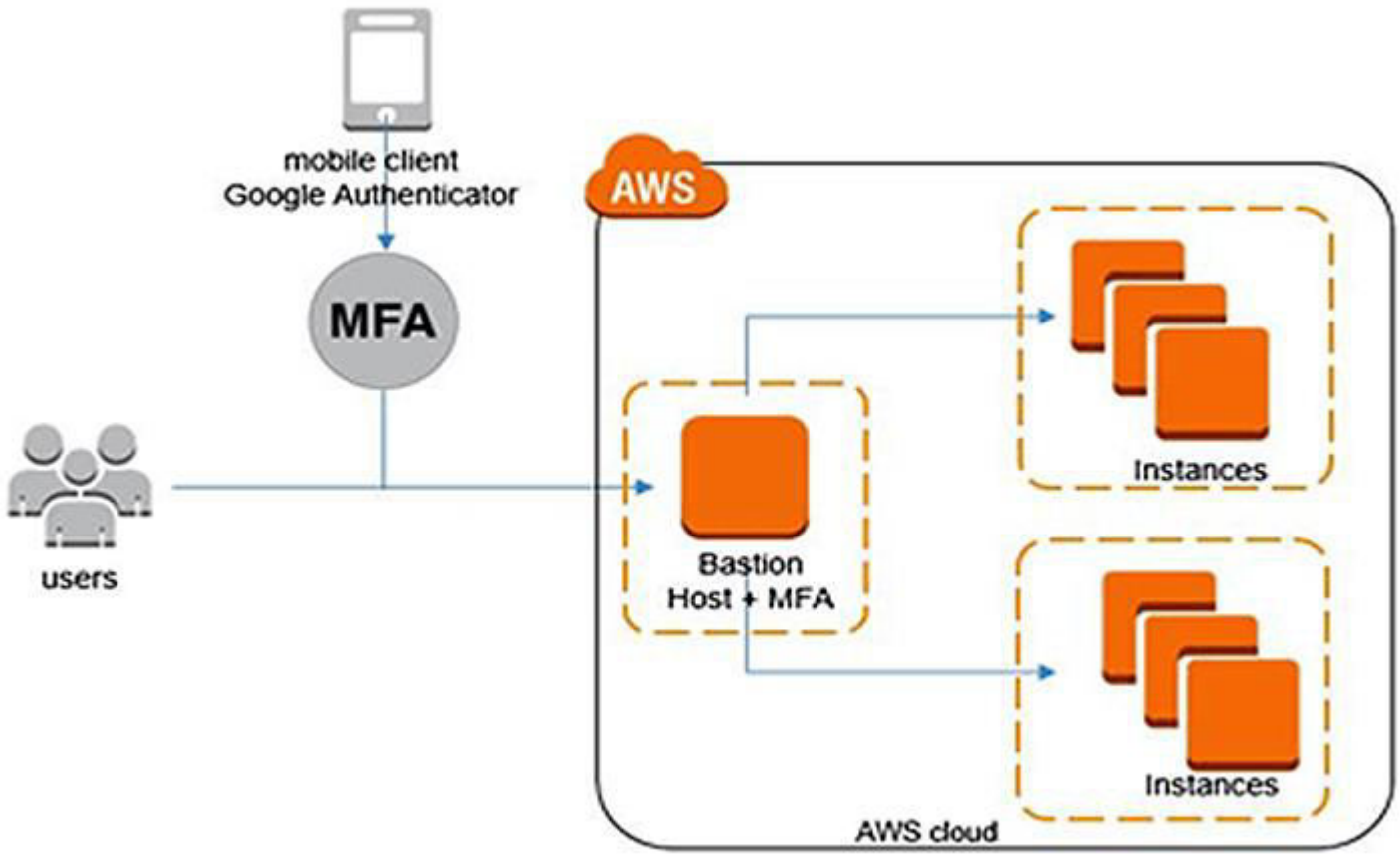
Assign Multi Factor Authentication to user

AWS Multi-Factor Authentication (MFA) is a simple best practice that adds an extra layer of protection on top of your username and password. With MFA enabled, when a user signs in to an AWS Management Console, they will be prompted for their user name and password (the first factor is what they know), as well as for an authentication code from their AWS MFA device (the second factor is what they have). Taken together, these multiple factors provide increased security for your AWS account settings and resources.

Learn With Us: Join our **AWS Solution Architect Training** and understand **AWS** basics in an easy way.

Why AWS MFA is Required

- Users have access to your account and can possibly change configurations and delete resources in your AWS account, so to overcome this it is required
- If you want to protect your root accounts and IAM user.
- Even if the password is stolen or hacked, the account is not compromised.
- When you enable this authentication for the root user, it affects only the root user credentials. IAM users in the account are distinct identities with their own credentials, and each identity has its own MFA configuration.



Check Out : Roles and Responsibilities Of An AWS Certified Solutions Architect.

MFA Device Options In AWS

The following are the MFA device options in AWS:

- **Virtual MFA Device:** Support for multiple tokens on a single device e.g **Google Authenticator** (Phone Only) **Authy** (Multi-Device)
- **Universal 2nd Factor (U2F) Security Key:** Supports multiple root and IAM users using a single security key. e.g **Yubikey** by Yubico (Third Party)
- **Hardware Key Fob MFA Device:** Provided by Gemalto (Third Party)
- **Hardware Key Fob MFA Device AWS GovCloud (US):** Provided by SurePassID (Third Party)

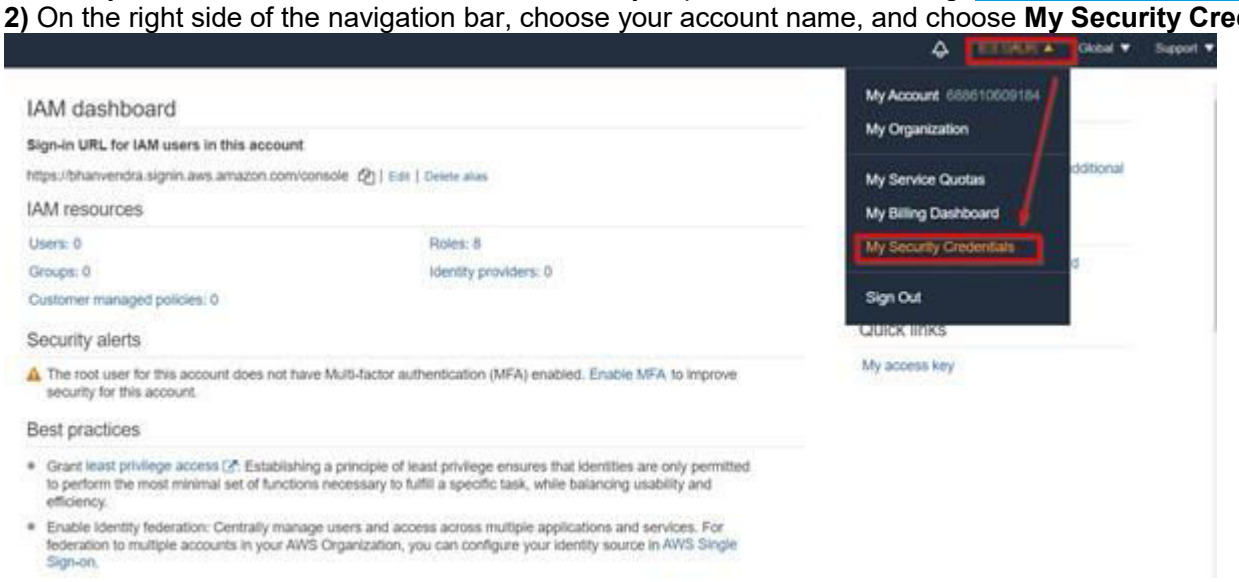
Check also: Free AWS Training

Enabling MFA On Root Account

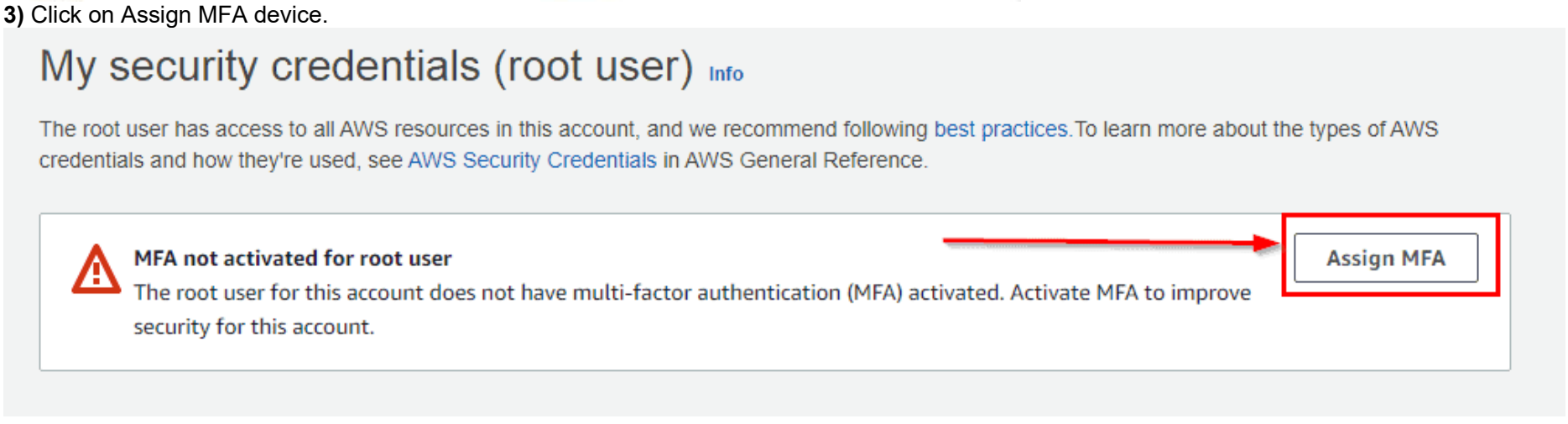
1) Log in to your AWS account by clicking **here**

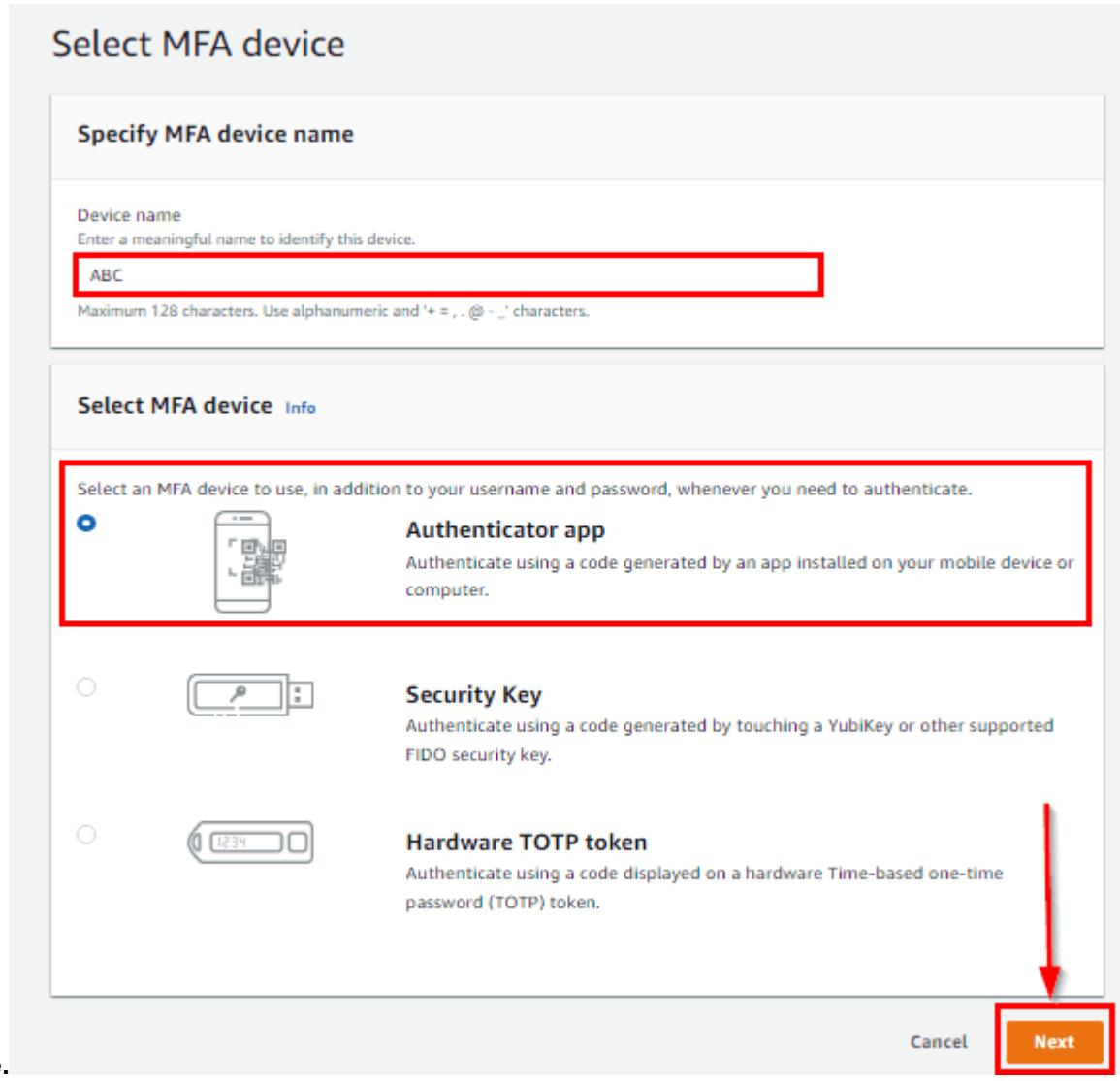
Note: If you have not created the free tier account yet, please check this blog. **How to create a free tier account**

2) On the right side of the navigation bar, choose your account name, and choose **My Security Credentials.**

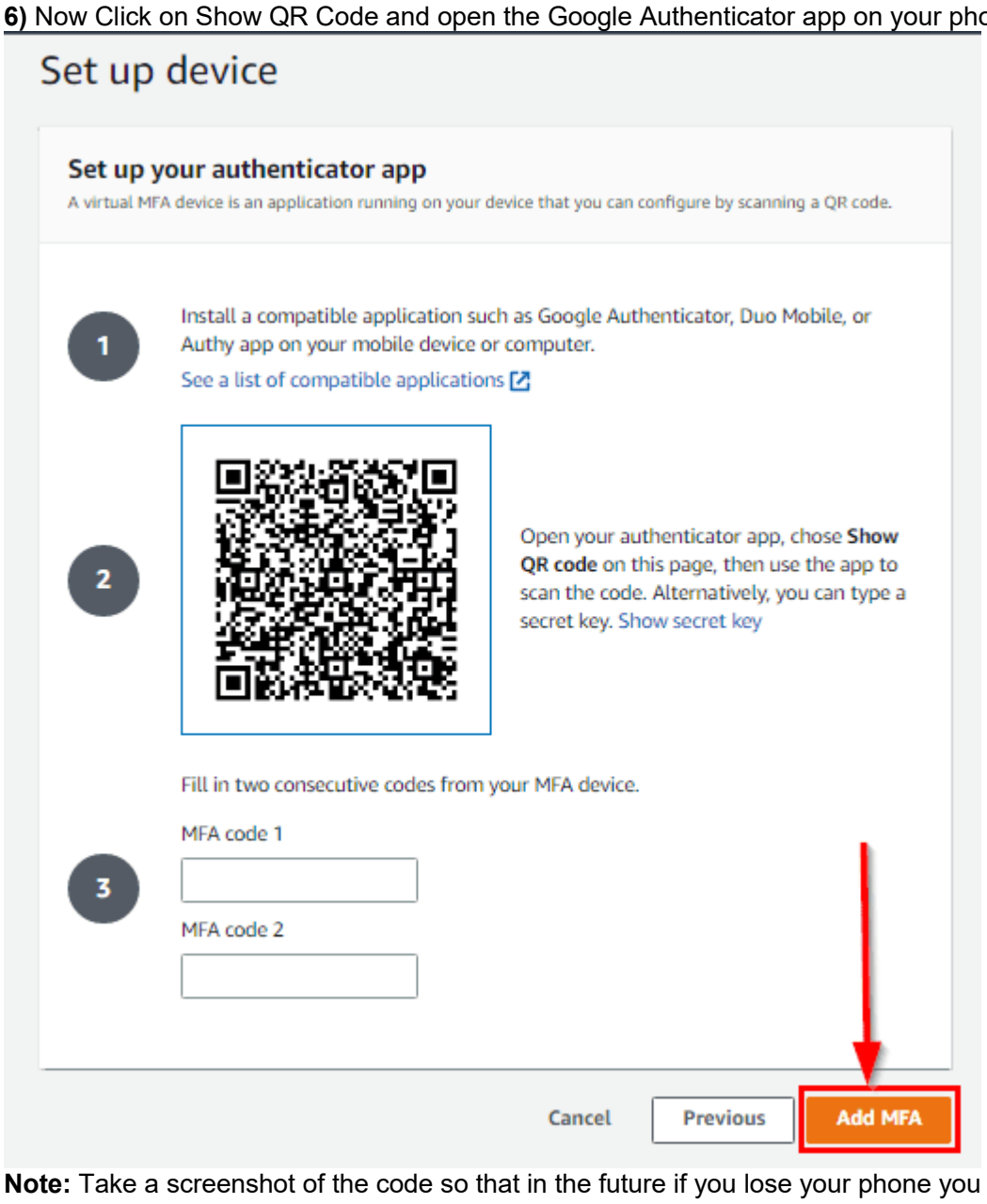


3) Click on Assign MFA device.



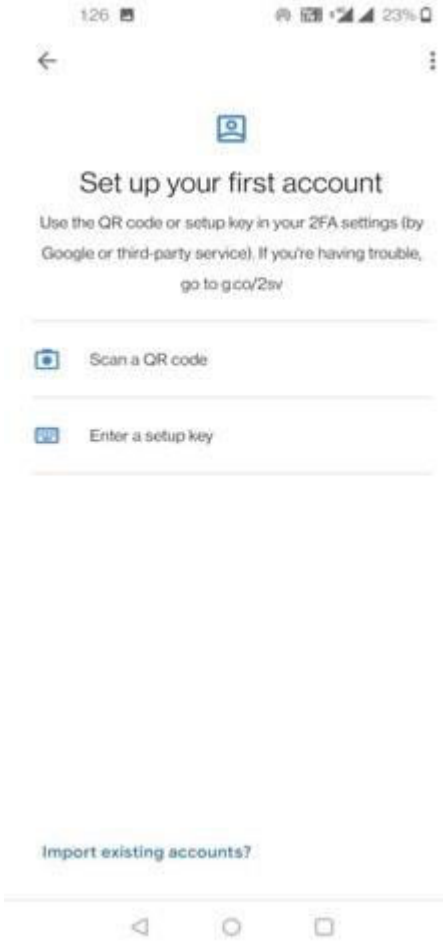


4) Choose **Virtual MFA Device** and click on **Continue.**
5) Now Install Google Authenticator on your phone.
Android: Click here
IOS: Click here
6) Now Click on Show QR Code and open the Google Authenticator app on your phone

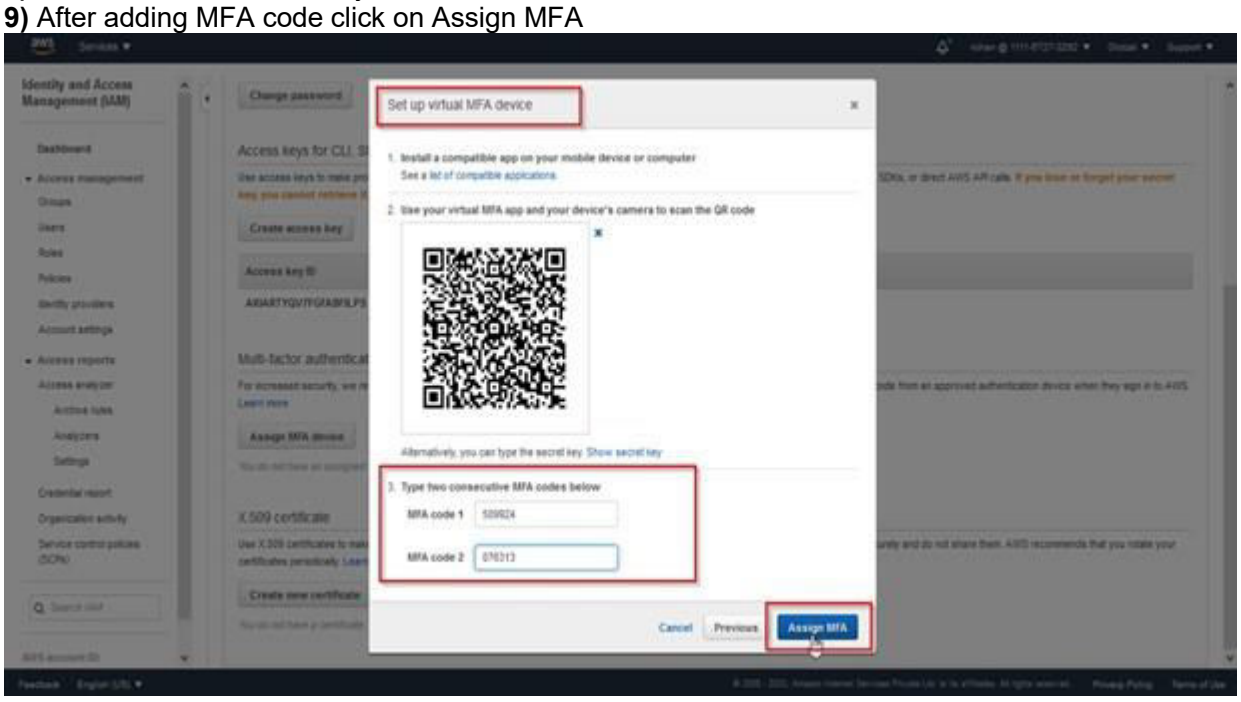


Note: Take a screenshot of the code so that in the future if you lose your phone you can use it to re-enable MFA

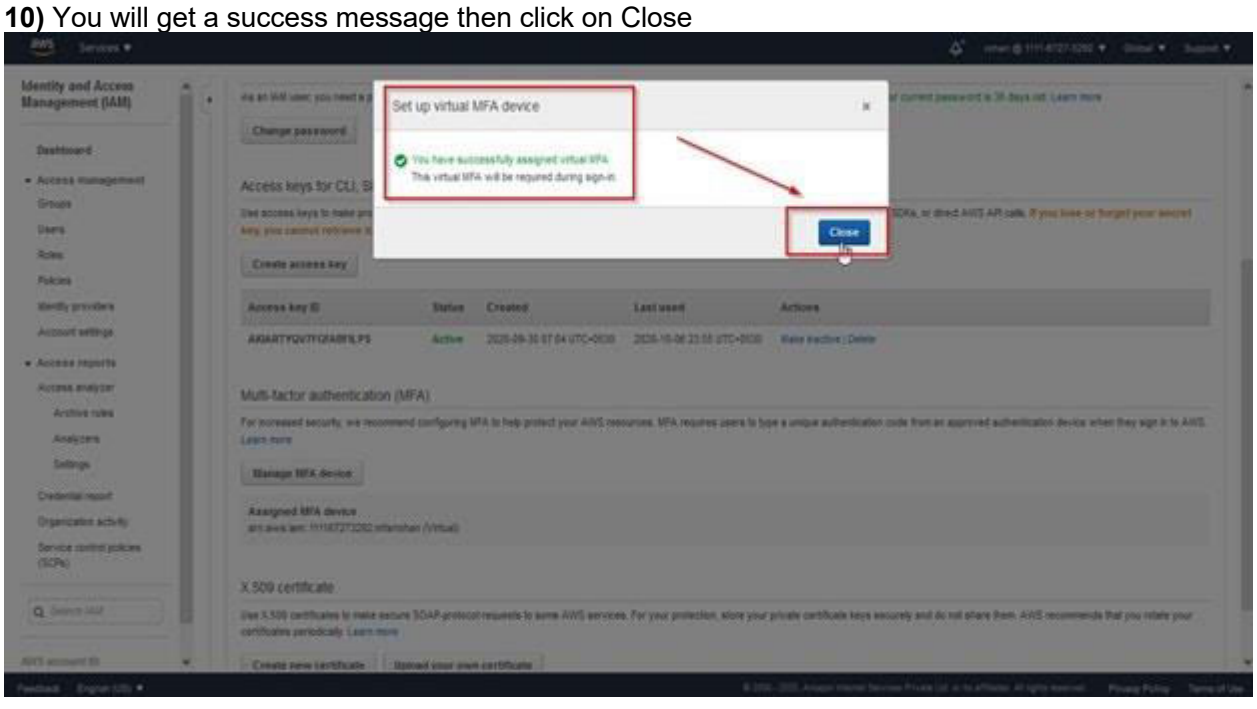
7) Now open the Google Authenticator App Click on Get started and Scan the QR code.



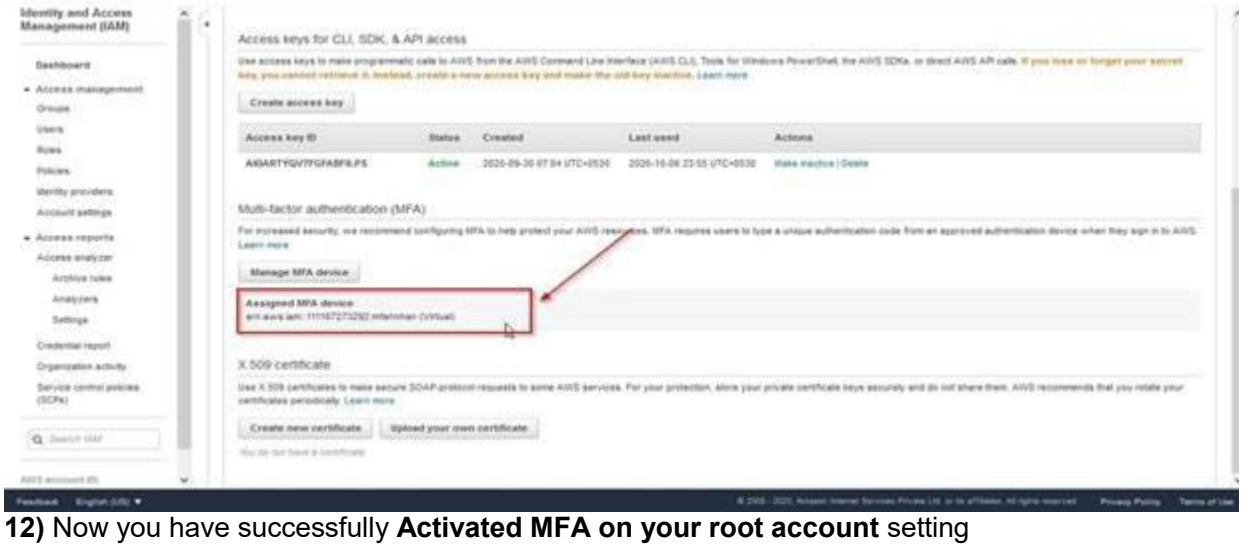
8) Now Enter the code from your Phone into MFA code 1 and MFA code 2.
9) After adding MFA code click on Assign MFA



10) You will get a success message then click on Close



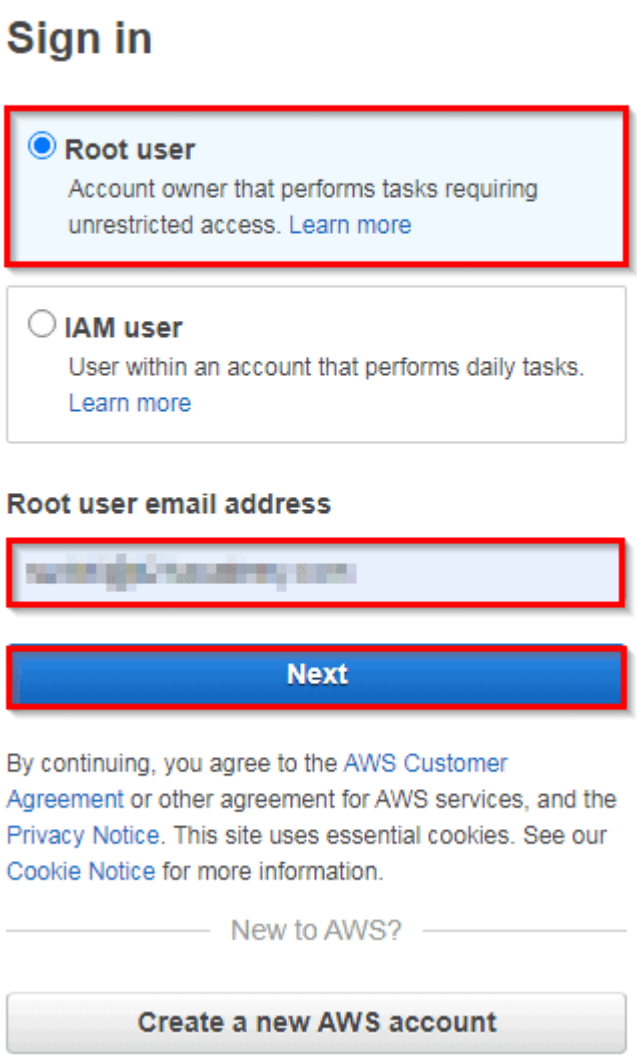
11) Now you will see that the device has been added for MFA



12) Now you have successfully **Activated MFA on your root account** setting

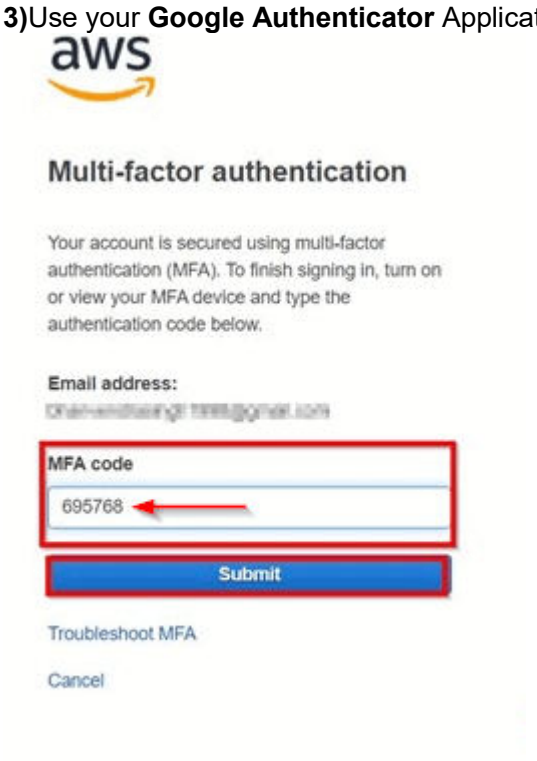
Accessing AWS Console Using MFA

1)Open your AWS console login page and click on **Root User** then enter your email



2) Enter your password corresponding to the Email address

3)Use your **Google Authenticator** Application on mobile and enter MFA code in AWS Console

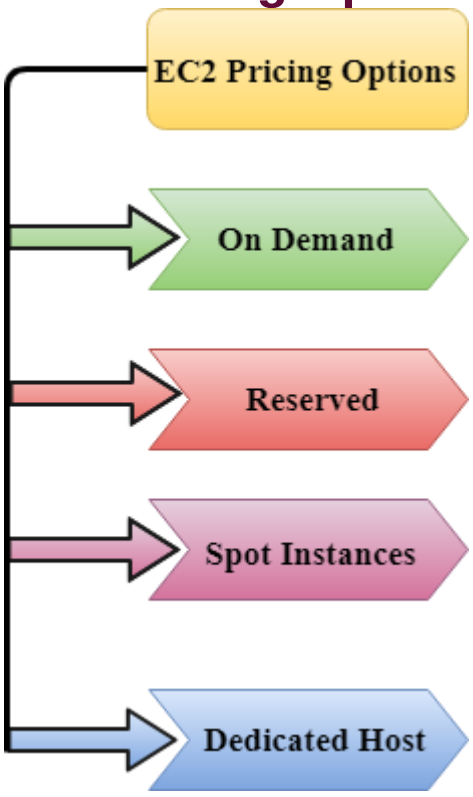


So this was an overview of AWS MFA and how you can enable it.

Elastic Compute Cloud (EC2) Instances

- EC2 stands for Amazon Elastic Compute Cloud.
- Amazon EC2 is a web service that provides resizable compute capacity in the cloud.
- Amazon EC2 reduces the time required to obtain and boot new user instances to minutes rather than in older days, if you need a server then you had to put a purchase order, and cabling is done to get a new server which is a very time-consuming process. Now, Amazon has provided an EC2 which is a virtual machine in the cloud that completely changes the industry.
- You can scale the compute capacity up and down as per the computing requirement changes.
- Amazon EC2 changes the economics of computing by allowing you to pay only for the resources that you actually use. Rather than you previously buy physical servers, you would look for a server that has more CPU capacity, RAM capacity and you buy a server over 5 year term, so you have to plan for 5 years in advance. People spend a lot of capital in such investments. EC2 allows you to pay for the capacity that you actually use.
- Amazon EC2 provides the developers with the tools to build resilient applications that isolate themselves from some common scenarios.

EC2 Pricing Options



On Demand

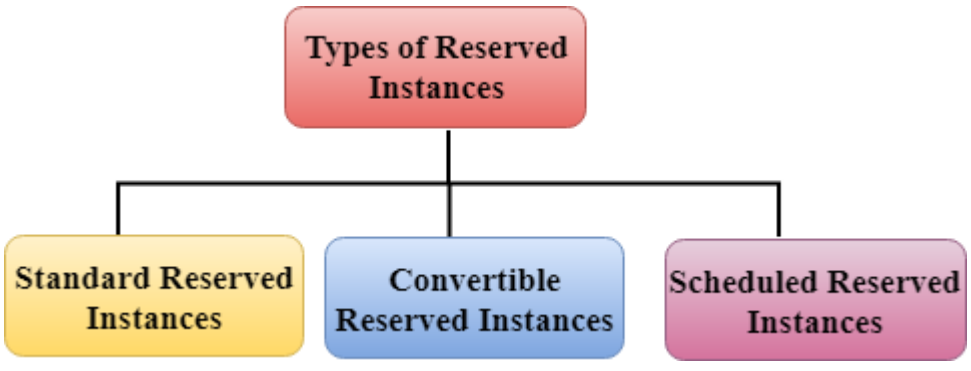
- It allows you to pay a fixed rate by the hour or even by the second with no commitment.
- Linux instance is by the second and windows instance is by the hour.
- On Demand is perfect for the users who want low cost and flexibility of Amazon EC2 without any up-front investment or long-term commitment.
- It is suitable for the applications with short term, spiky or unpredictable workloads that cannot be interrupted.
- It is useful for the applications that have been developed or tested on Amazon EC2 for the first time.
- On Demand instance is recommended when you are not sure which instance type is required for your performance needs.

Reserved

- It is a way of making a reservation with Amazon or we can say that we make a contract with Amazon. The contract can be for 1 or 3 years in length.
- In a Reserved instance, you are making a contract means you are paying some upfront, so it gives you a significant discount on the hourly charge for an instance.
- It is useful for applications with steady state or predictable usage.
- It is used for those applications that require reserved capacity.
- Users can make up-front payments to reduce their total computing costs. For example, if you pay all your upfronts and you do 3 years contract, then only you can get a maximum discount, and if you do not pay all upfronts and do one year contract then you will not be able to get as much discount as you can get If you do 3 year contract and pay all the upfronts.

Types of Reserved Instances:

- Standard Reserved Instances
- Convertible Reserved Instances
- Scheduled Reserved Instances



Standard Reserved Instances

- It provides a discount of up to 75% off on demand. For example, you are paying all up-fronts for 3 year contract.
- It is useful when your Application is at the steady-state.

Convertible Reserved Instances

- It provides a discount of up to 54% off on demand.
- It provides the feature that has the capability to change the attributes of RI as long as the exchange results in the creation of Reserved Instances of equal or greater value.
- Like Standard Reserved Instances, it is also useful for the steady state applications.

Scheduled Reserved Instances

- Scheduled Reserved Instances are available to launch within the specified time window you reserve.
- It allows you to match your capacity reservation to a predictable recurring schedule that only requires a fraction of a day, a week, or a month.

Spot Instances

- It allows you to bid for a price whatever price that you want for instance capacity, and providing better savings if your applications have flexible start and end times.
- Spot Instances are useful for those applications that have flexible start and end times.
- It is useful for those applications that are feasible at very low compute prices.
- It is useful for those users who have an urgent need for large amounts of additional computing capacity.
- EC2 Spot Instances provide less discounts as compared to On Demand prices.
- Spot Instances are used to optimize your costs on the AWS cloud and scale your application's throughput up to 10X.
- EC2 Spot Instances will continue to exist until you terminate these instances.

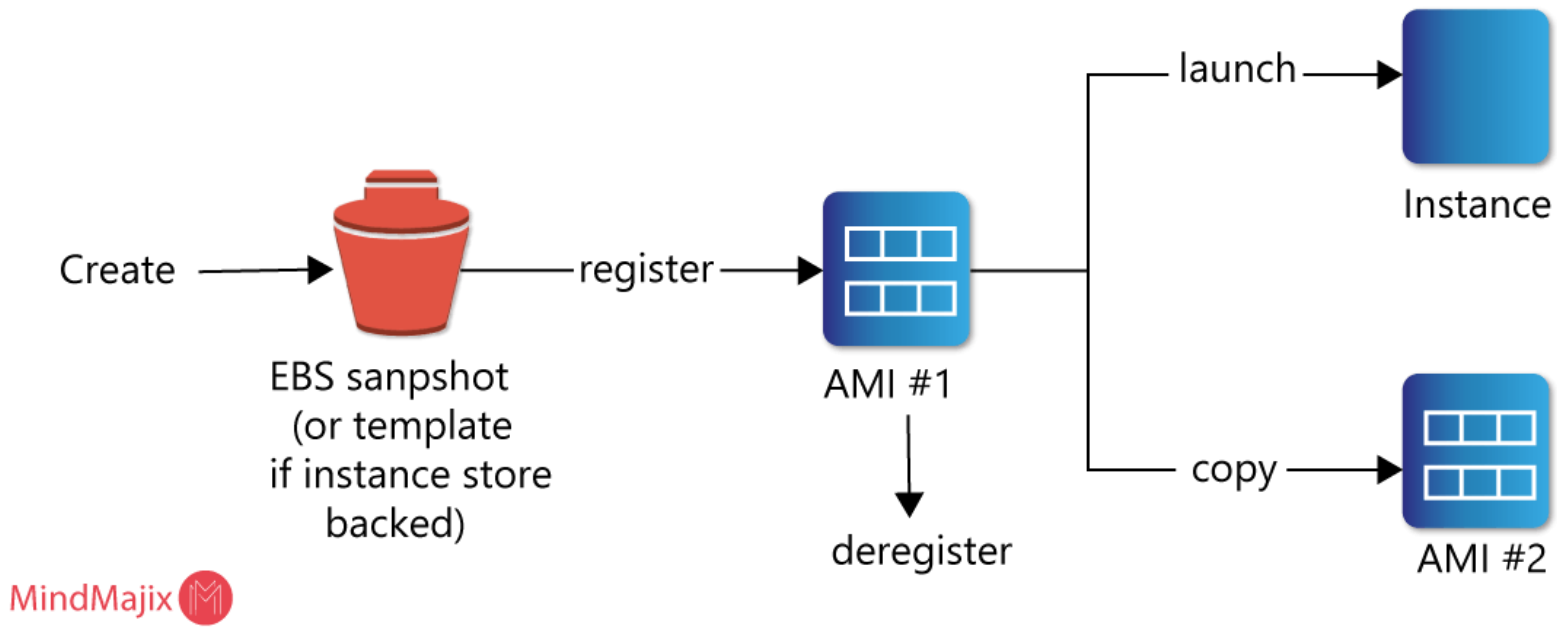
Dedicated Hosts

- A dedicated host is a physical server with EC2 instance capacity which is fully dedicated to your use.
- The physical EC2 server is the dedicated host that can help you to reduce costs by allowing you to use your existing server-bound software licenses. For example, Vmware, Oracle, SQL Server depending on the licenses that you can bring over to AWS and then they can use the Dedicated host.
- Dedicated hosts are used to address compliance requirements and reduces host by allowing to use your existing server-bound server licenses.
- It can be purchased as a Reservation for up to 70% off On-Demand price.

Working with AMIs

In simple words, an AWS AMI is a virtual image that can be used to create an instance of a virtual machine. How does it operate? Here's an illustration that summarizes the AMI lifecycle:

Related Article: What is Amazon AWS



The first step is to create and register the AMI, which is then used to launch or set up AWS instances. Alternatively, you can launch instances from another AMI provided the AMI creator grants you the necessary launch permissions. You can also copy an existing AMI within the same or different AWS region. Finally, you need to deregister an AMI once you no longer require it.

An AMI typically consists of the following three components:

- The template is the root volume for the AWS instances (for example, application server, operating system, or web application).
- Launch permissions that determine which AWS account can use this AMI to set up an instance.
- Block device mapping that specifies the root device volumes that are attached to the AWS instance after launch.

What are the types of AWS AMI?

As an AWS user, you can choose your AMI on the basis of the following parameters:

Operating System

You can choose an AMI on the basis of the supported operating system (or OS) like Windows or Linux.

32-bit or 64-bit Architecture

This parameter is based on the architecture of your selected OS.

Region

This parameter is based on the selected region of the Amazon machine image that comprises regions, availability zones, and local zones. Each region operates in different geographical regions and is independent of each other.

Storage (EBS or Instance store)

This AMI parameter is based on the storage of the root device. Based on storage, AMIs are categorized as either of the following two types namely:

- **EBS-backed instances:** In this case, the root device for an AWS instance – launched using AMI – is an Amazon EBS volume that has been created from Amazon EBS.
- **Instance store-backed instances: In** this case, the root device for an AWS instance – launched using Ami – is an Amazon instance store volume that has been created from an Amazon S3 template.

The table below highlights the main differences between these two storage types:

	EBS-backed AMI	Instance store-backed AMI

Instance boot time	< 1 minute	< 5 minutes
Size limit for root device	16TiB	10GiB
Data persistence	The root volume is deleted on the termination of the instance.	Data persists only during the overall life of the instance.
Cost	Less expensive	More expensive

Launch permissions

An AMI owner can determine their instance availability through the following three launch permissions:

- **Public,** that grants instance launch permission to all AWS account holders.
- **Explicit**, that grants launch permission only to specific AWS accounts.
- **Implicit**, where only the AMI owner has permission to launch an instance.

Related Article: AWS Interview Questions for Experienced Professionals

What are Shared AMIs?

In simple language, a shared AMI is one that an AWS developer creates and shares with other users for use. Shared AMIs enable any new AWS user to easily get started using the AWS platform. On the flip side, Shared AMIs are not guaranteed any security or integrity and must be treated as any other foreign code.

You can locate the following shared AMIs from the Amazon EC2 console by selecting the “AMIs” option from the navigation pane:

- **Private images:** that contain all the shared AMIs that are private and only for your use.
- **Public images:** that contain all the shared AMIs that have been created for public use.

Additionally, you can also create your own AMI and share it for public use or with specific AWS accounts.

Here’s how you can share your AMI with all AWS account holders:

- Open the Amazon EC2 console and choose AMIs from the navigation pane.
- Select your AMI to be shared, then click Actions > Modify Image Permissions.
- Select the “Public” option before saving your changes.

Alternatively, you can share your AMI for public use by modifying the “launchPermission” property of your AMI. To do this, run the “modify-image-attribute” command for the specific AMI as follows:

```
aws ec2 modify-image-attribute
--image-id ami-<ID>
--launch-permission "Add=[{Group=all}]"
```

Here’s how you can share your AMI with specific AWS account holders:

- Open the Amazon EC2 console and choose AMIs from the navigation pane.
- Select your AMI to be shared, then click Actions > Modify Image Permissions.
- Specify the AWS account holder by entering their AWS account number in the respective field and then clicking “Add Permission.”

Alternatively, you can share your AMI for public use by modifying the “launchPermission” property of your AMI. To do this, run the “modify-image-attribute” command for the specific AMI as follows:

```
aws ec2 modify-image-attribute
--image-id ami-<ID>
--launch-permission "Add=[{ UserId=<UserID>}]"
```

What are Paid AMIs?

As the name suggests, a paid AMI is when you purchase an AMI from another developer. As an AMI developer, you can create an AMI and then sell the same on the AWS Marketplace that is an online store for paid AMIs. For paid AMIs, the AWS user is charged for the instance based on the rates set by the AMI owner.

You can find a paid AMI either from the Amazon EC2 console or from the AWS Marketplace. Here’s how to find one from the Amazon EC2 console:

- Open the Amazon EC2 console and choose AMIs from the navigation pane.
- Choose the “Public images” option from the available filter, followed by “Owner” and “AWS Marketplace.”
- Search for available paid AMIs from this owner, or enter the specific product code (if you happen to know it).

To find a paid AMI from the AWS Marketplace:

- Open the AWS Marketplace.
- Specify your operating system.
- Search for your paid AMI by using the available search filters.

Alternatively, you can find a paid AMI by running the “describe-images” command as follows:

```
aws ec2 describe-images
    --owners aws-marketplace
```

On running this command, you are provided with a list of paid AMIs along with its product code. You can filter the list of paid AMIs by searching for the product code.

You need to first sign up or purchase a paid AMI before launching an AWS instance using the AMI. To purchase a paid AMI, you can use the Amazon EC2 launch wizard.

What are the types of Linux AMI Virtualization?

For the Linux OS, AMIs uses either of the following virtualization methods, which are primarily different in their booting processes:

Paravirtual or PV

In this case, the AVM instance boots with the help of a special boot loader, PV-GRUB that starts the boot cycle and then loads the menu.1st file kernel on the image. AVM users deploying this method run their instances on host hardware that do not explicitly support virtualization, hence cannot exploit hardware extensions like GPU processing for better performance.

Hardware virtual machine (HVM)

AMIs that are based on the HVM method has the ability to directly run an OS on the virtual machine without any underlying modification. HVM allows complete access of the virtualized hardware and booting through the execution of the master boot record of your image’s root block device. HVM users can also exploit the hardware extensions on the host system for better performance.

On the whole, the PV method works better for applications with storage and network operations as compared to HVM. However, for the best overall performance, HVM AMIs work better when launching your instances.

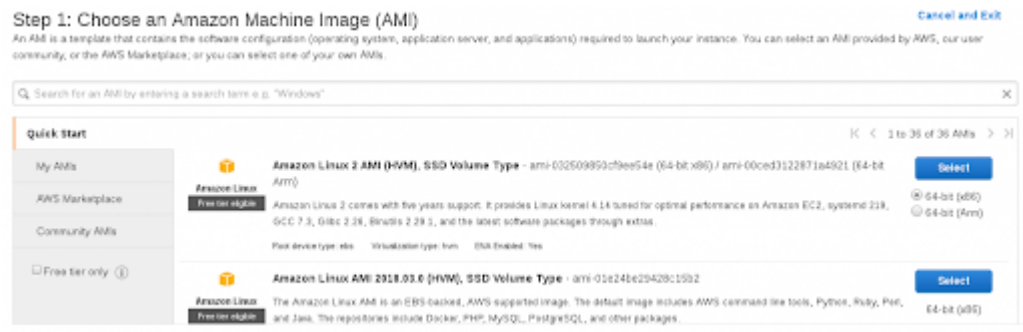
How do you find a Linux AMI?

As discussed before, you can only launch an instance after selecting a Linux AMI. Here are the two best ways to locate and select an AMI from the Amazon EC2 console:

Choosing an AMI during the EC2 launch

1) Sign in to the Amazon EC2 console and then select from the available AWS regions in which you wish to launch your instance.

2) Next, from the dashboard, click the “Launch Instance” button.



3) This will open the section where you can now choose an Amazon Machine Image (AMI). You can choose the AMIs from any of the following four sections:

- **Quick Start:** lists the commonly used AMIs.
- **My AMI:** contains all the AMIs created by you.
- **AWS Marketplace** contains all the paid AMIs created by other AWS users. You can also create and sell your AMIs on this marketplace.
- **Community AMIs:** that contain AMIs that have been created for public use.

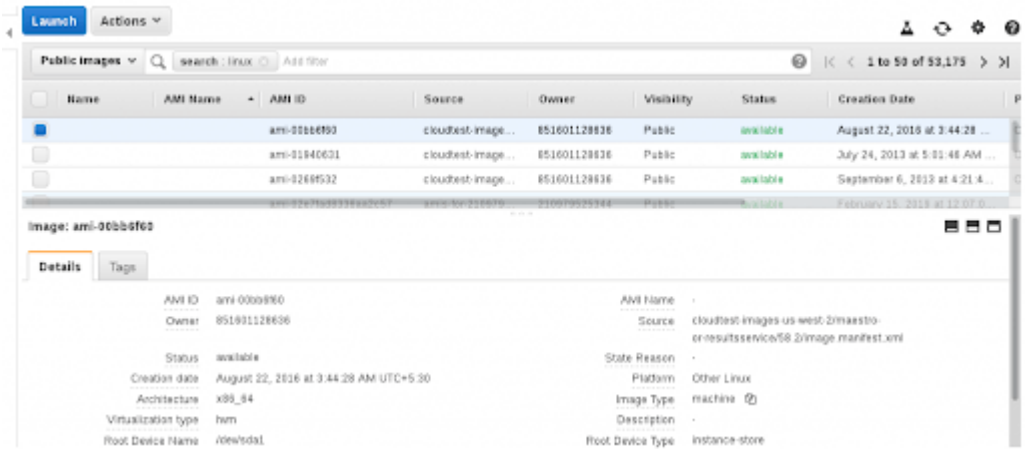
4) Review the existing AMIs and check the Root device and virtualization type of each AMI. Finally, choose the AMI that satisfies your instance requirements.

Choosing an AMI using the Images page

1) Sign in to the Amazon EC2 console and then select from the available AWS regions in which you wish to launch your instance.

2) From the Images menu on the left navigation pane, choose the “AMIs” option.

3) From the Filter section, select the “Public Images” option from the drop-down list to view all the machine images that are safe for public use. Alternatively, you can select “Private Images" to view all the shared AMIs that have been shared with you.



You can also choose to filter the AMIs on the basis of other parameters like AMI name, owner, architecture, and image type.

4) Select the AMI of your choice, then click “Launch” to launch the AWS instance on the next page.

Leave an Inquiry to learn [AWS Course in Bangalore]

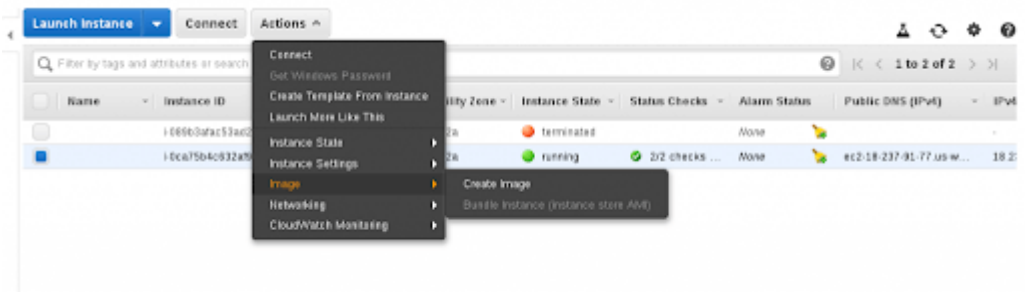
How do you create your own AMI?

You can also create your own AMI from a running AWS instance. You can use this AMI later to launch additional instances in the near future.

Here are the steps you need to follow to create your own AMI:

1) Sign into the Amazon EC2 console.

2) Next, from the dashboard, select the running instance (as illustrated in the figure), then click the Actions > Image > Create Image option.



3) The “Click Image” dialog box is displayed with the following instance-related fields:

- **Instance ID:** that shows the ID of the current EC2 instance.
- **Image name:** where you can specify the name of your AMI.
- **Image description:** an optional field where you can enter the AMI description.
- **Instance volume:** that shows the default settings for the instance volume. If required, you can modify the volume size and type in the respective fields. You can also add more instance volumes if required.

4) Click “Create Image” to start the creation of your AMI.

5) After creating the AMI, you can view your AMI in the “Owned by me” section along with its details.

How to deregister your AMI?

You can deregister your AMI once you have finished using it to launch new instances. This does not impact any of the running instances that you have launched using the same AMI.

Additionally, deregistering an EBS-backed AMI does not impact the snapshots taken for the instance volumes created when creating the AMI. You need to delete those snapshots separately.

For instance, for store-backed AMIs, deregistering does not impact the files uploaded to the Amazon S3 location when creating the AMI. You need to delete uploaded files separately.

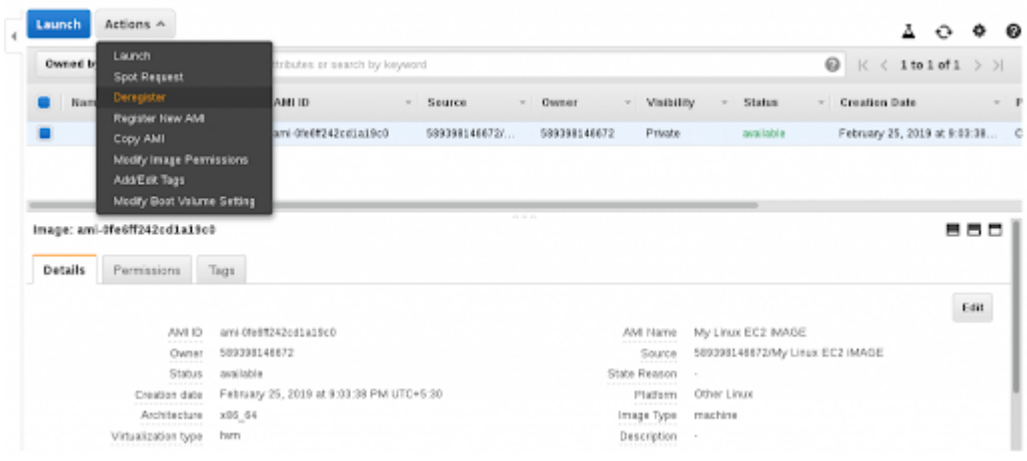
Here’s how you can deregister your AMI:

1) Sign in to the Amazon EC2 console.

2) Next, from the dashboard, click the “AMIs” option in the left navigation pane.

3) Select the “Owned by me” filter option, and then select the AMI that you want to deregister.

4) Click Actions > Deregister



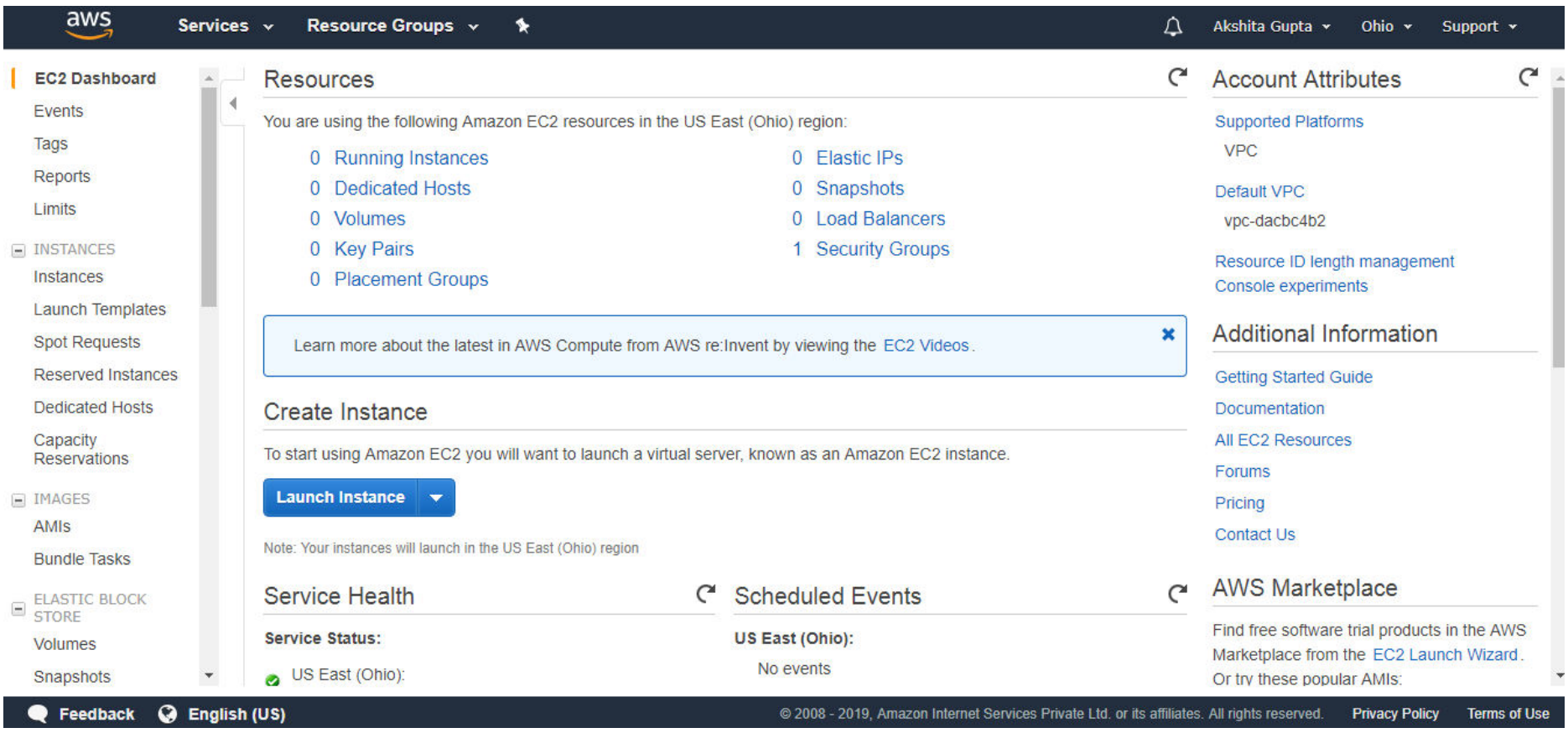
5) Click Continue on the Deregister dialog box to complete the deregistration process.

Alternatively, you can deregister your owned AMI by running the “deregister-image” command as follows:

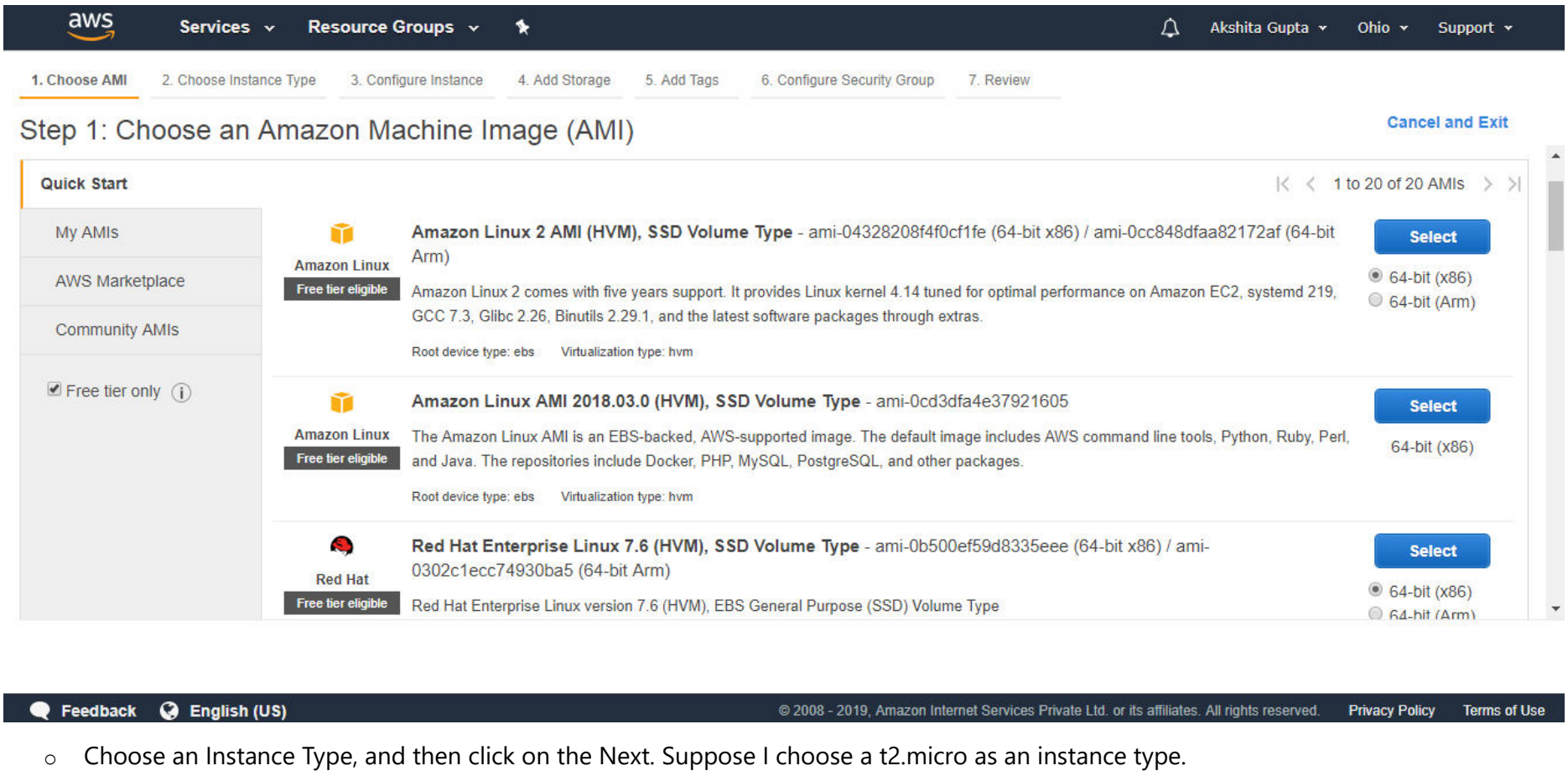
```
aws ec2 deregister-image --image-id ami_id
```

EC2 Instance Creation

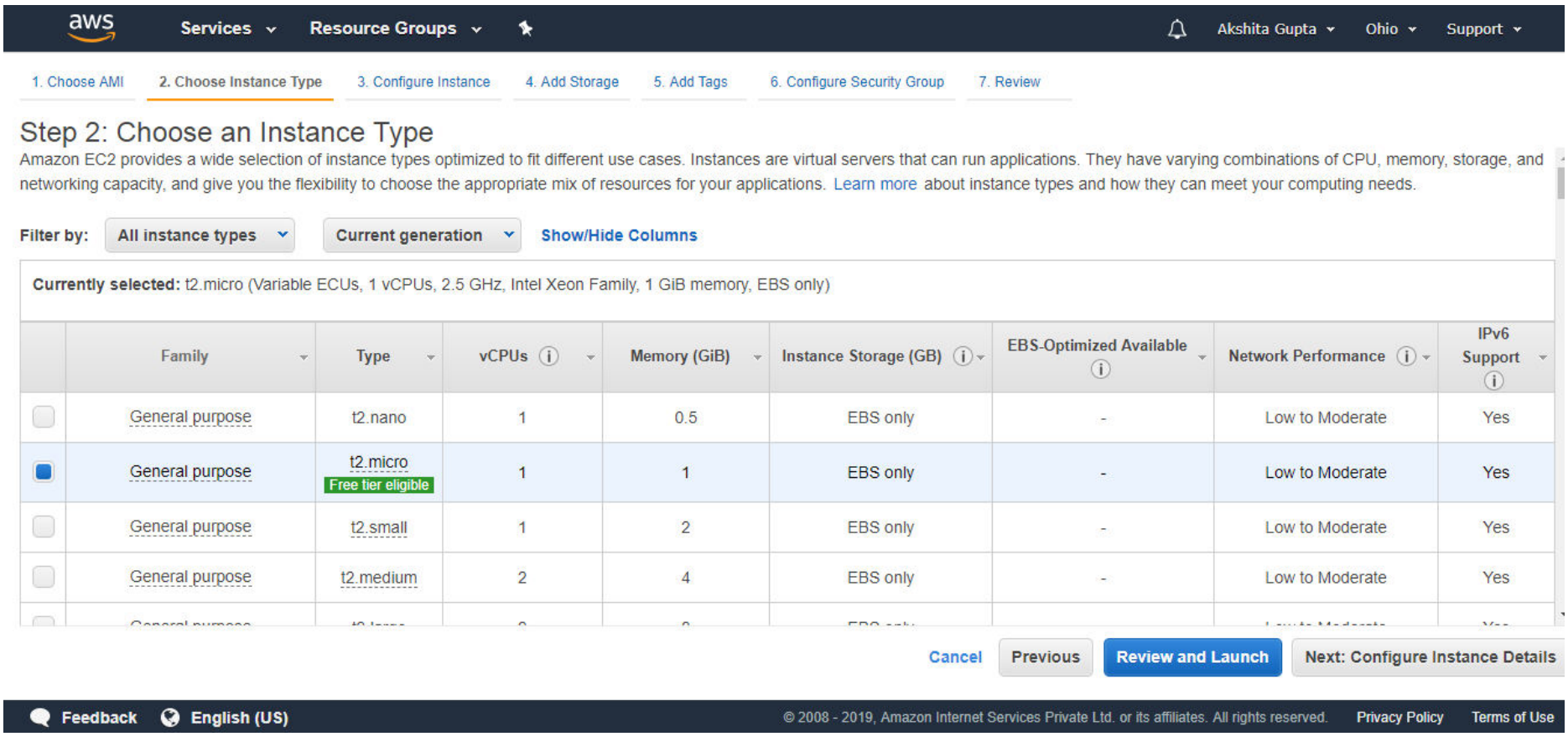
- Sign in to the AWS Management Console.
- Click on the EC2 service.
- Click on the **Launch Instance** button to create a new instance.



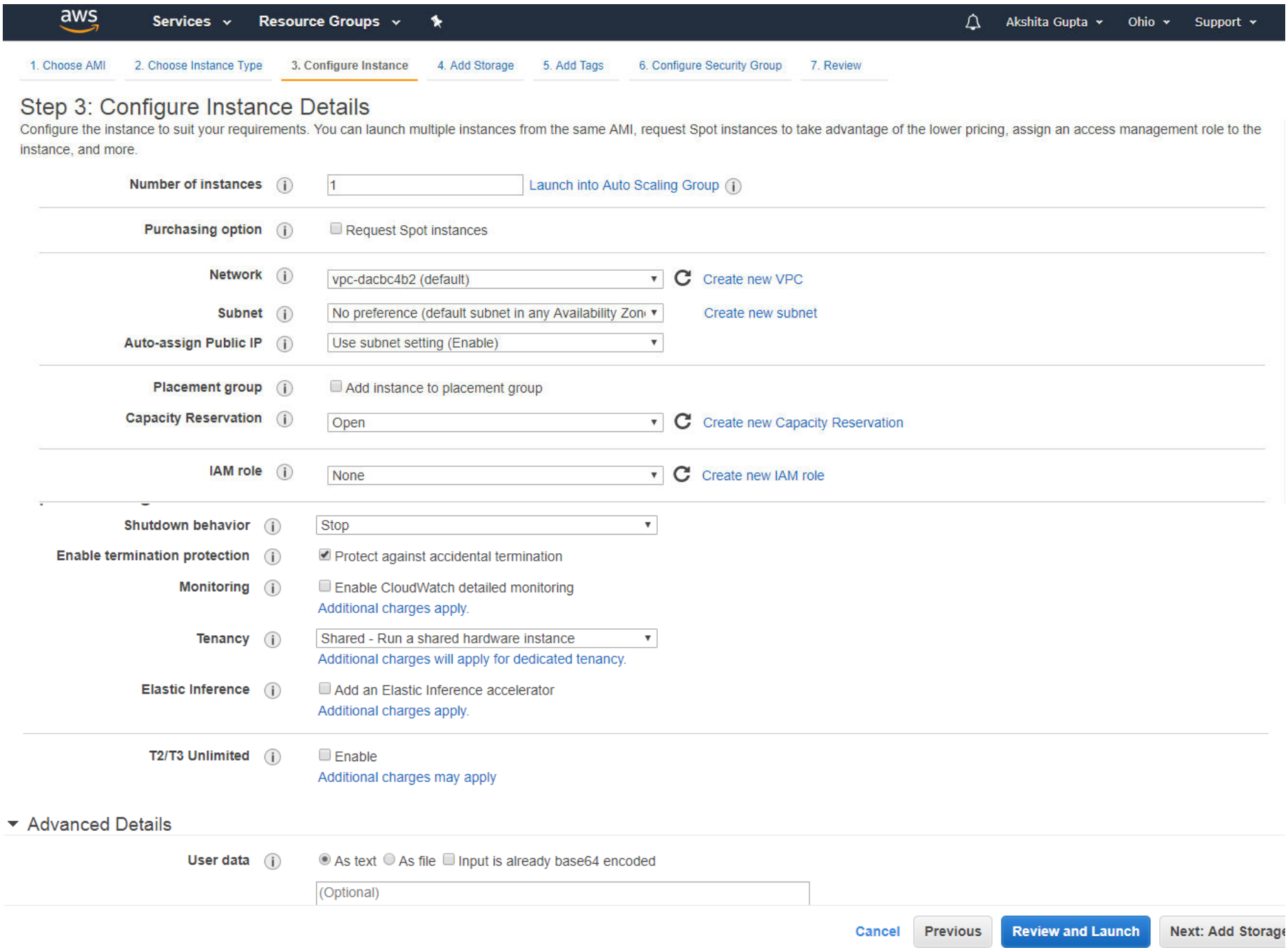
- Now, we have different Amazon Machine Images. These are the snapshots of different virtual machines. We will be using Amazon Linux AMI 2018.03.0 (HVM) as it has built-in tools such as java, python, ruby, perl, and especially AWS command line tools.



- Choose an Instance Type, and then click on the Next. Suppose I choose a t2.micro as an instance type.



- The main setup page of EC2 is shown below where we define setup configuration.



Where,

Number of Instances: It defines how many EC2 instances you want to create. I leave it as 1 as I want to create only one instance.

Purchasing Option: In the purchasing option, you need to set the price, request from, request to, and persistent request. Right now, I leave it as unchecked.

Tenancy: Click on the **Shared-Run a shared hardware instance** from the dropdown menu as we are sharing hardware.

Network: Choose your network, set it as default, i.e., **vpc-dacbc4b2 (default)** where vpc is a virtual private cloud where we can launch the AWS resources such as EC2 instances in a virtual cloud.

Subnet: It is a range of IP addresses in a virtual cloud. In a specified subnet, you can add new AWS resources.

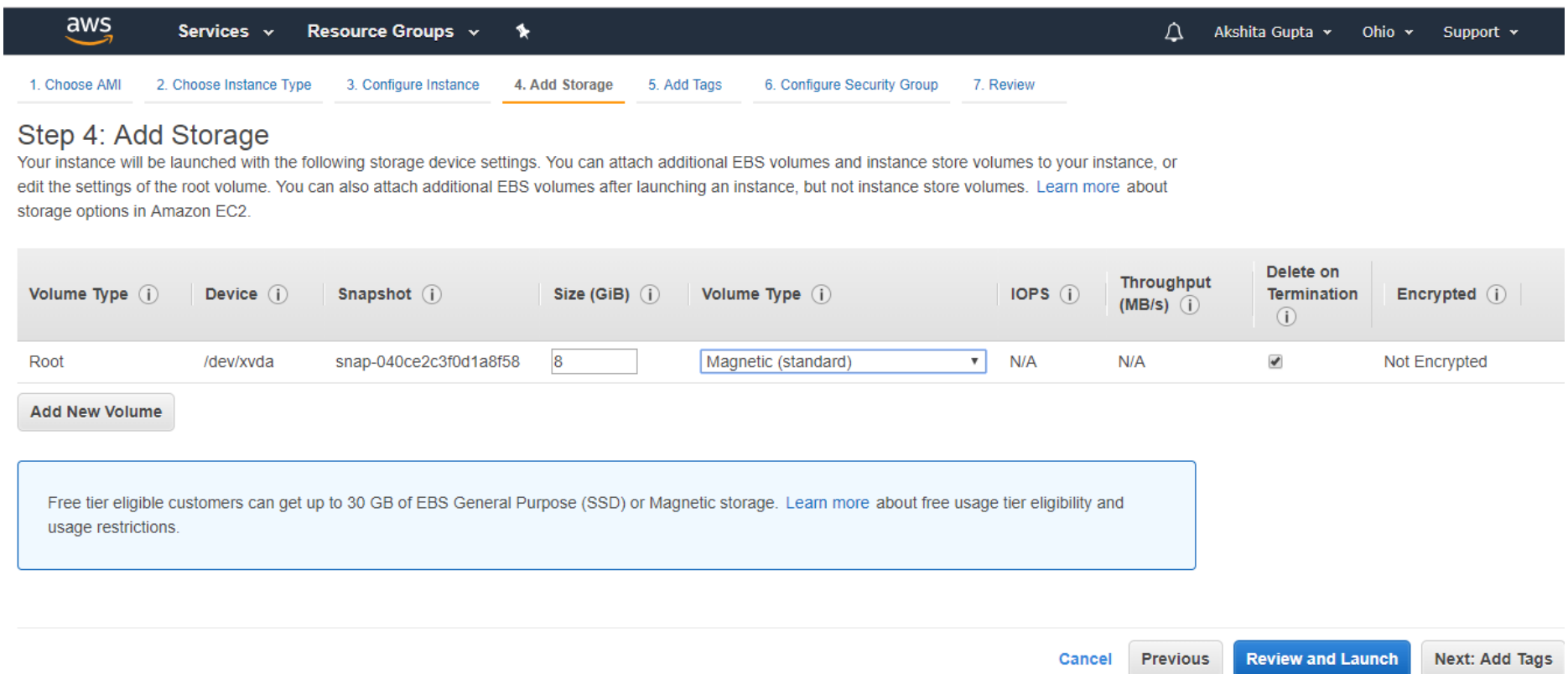
Shutdown behavior: It defines the behavior of the instance type. You can either stop or terminate the instance when you shut down the Linux machine. Now, I leave it as Stop.

Enable Termination Protection: It allows the people to protect against the accidental termination.

Monitoring: We can monitor things such as CPU utilization. Right now, I uncheck the Monitoring.

User data: In Advanced details, you can pass the bootstrap scripts to EC2 instance. You can tell them to download PHP, Apache, install the Apache, etc.

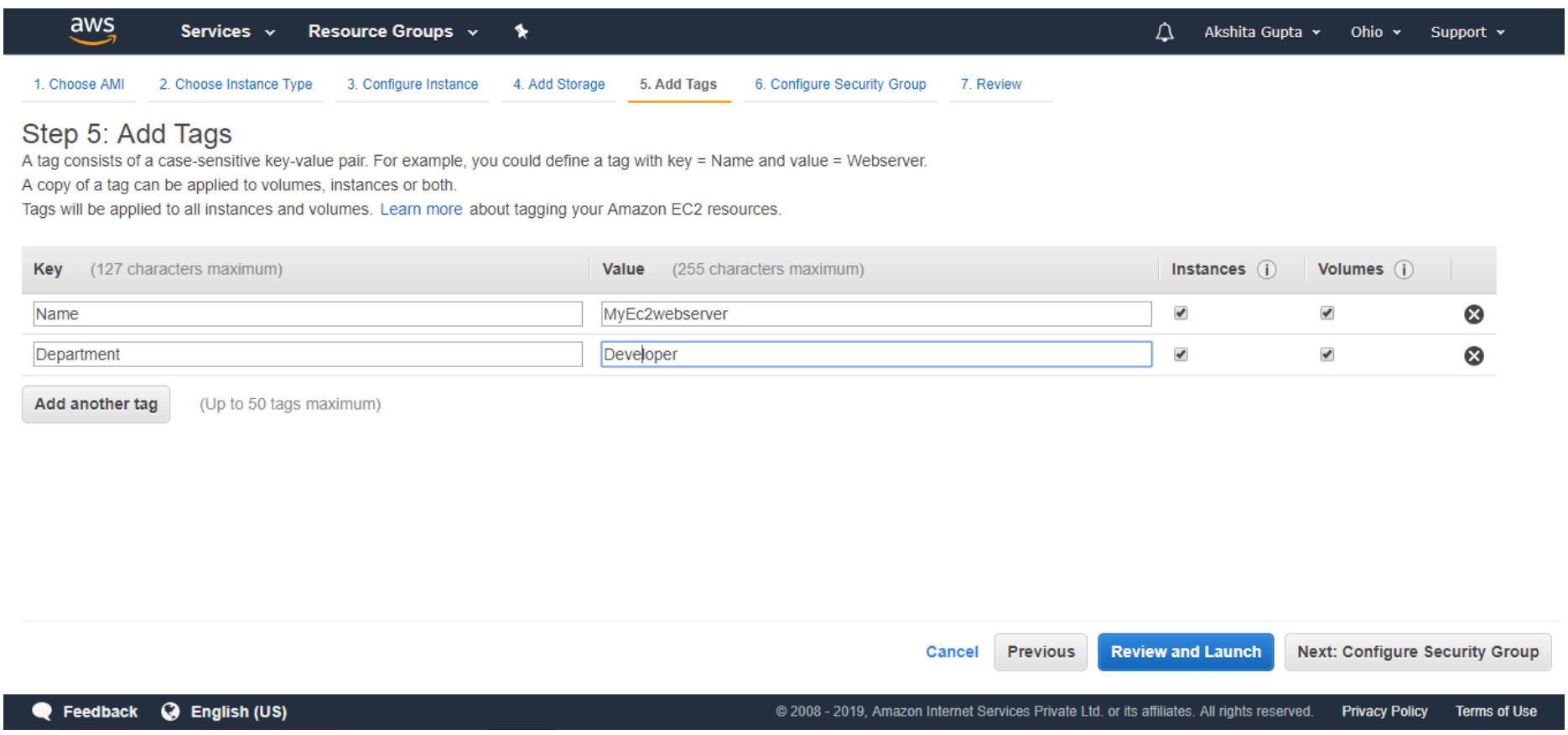
- Now, add the EBS volume and attach it to the EC2 instance. Root is the default EBS volume. Click on the **Next.**



Volume Type: We select the Magnetic (standard) as it is the only disk which is bootable.

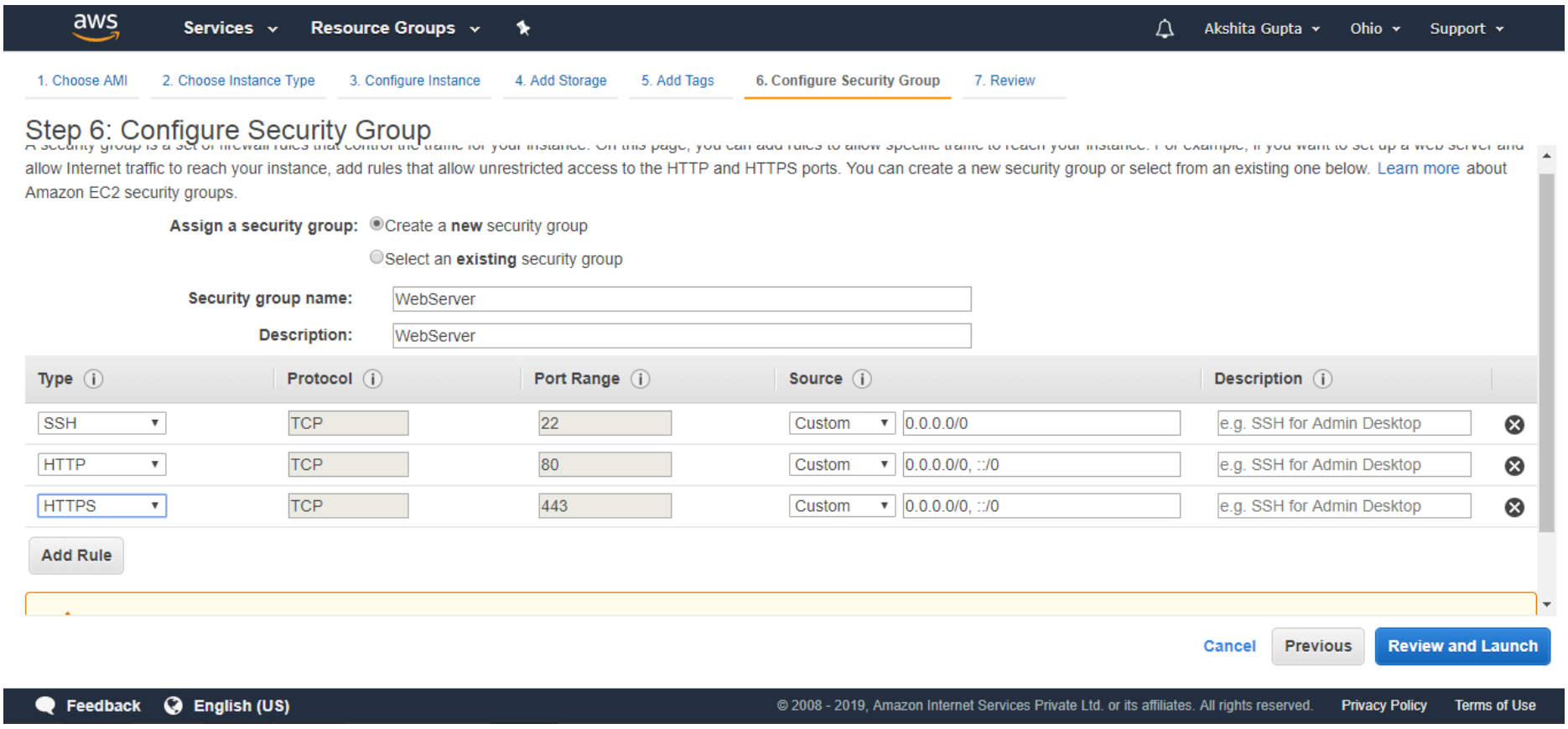
Delete on termination: It is checked means that the termination of an EC2 instance will also delete EBS volume.

- Now, Add the Tags and then click on the Next.

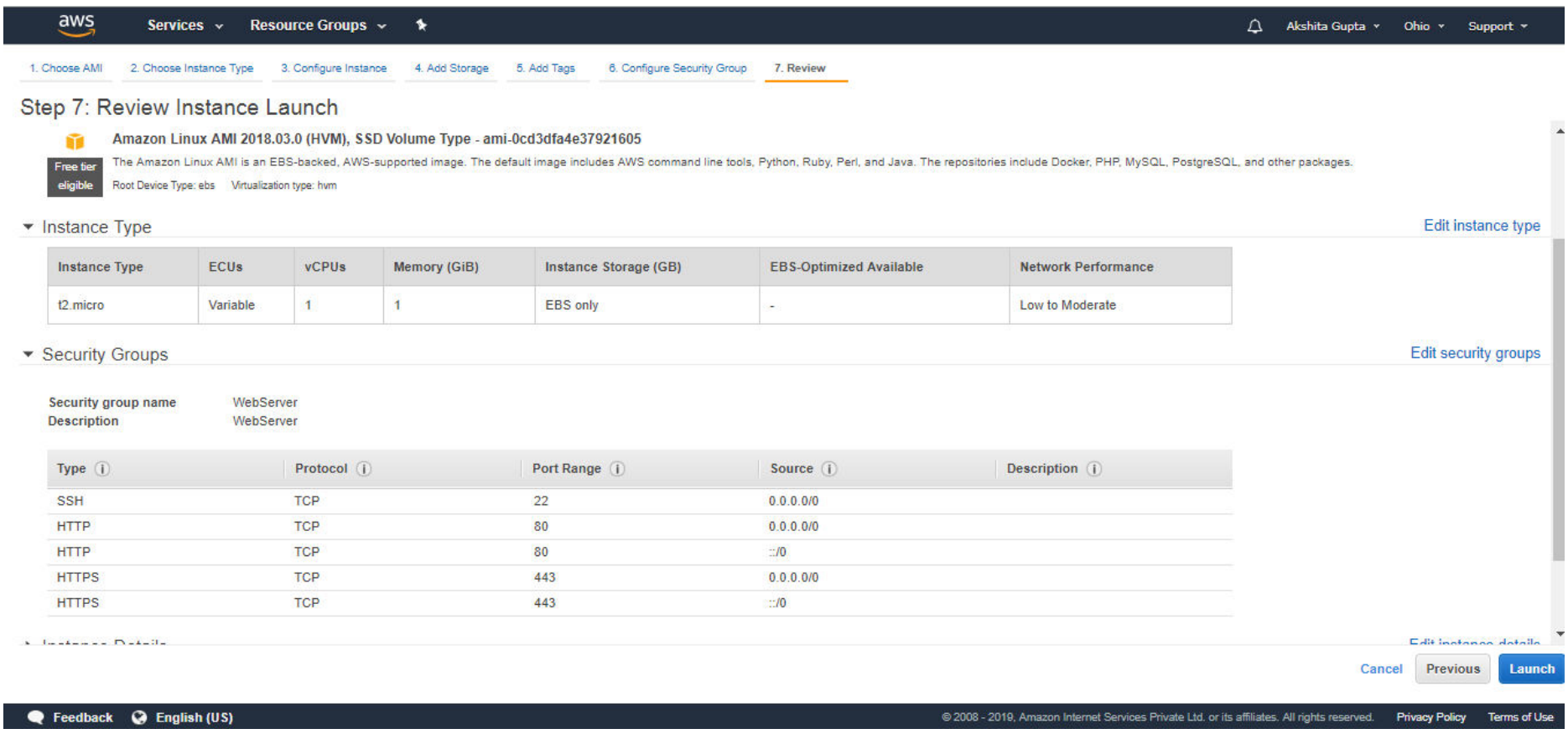


In the above screen, we observe that we add two tags, i.e., the name of the server and department. Create as many tags as you can as it reduces the overall cost.

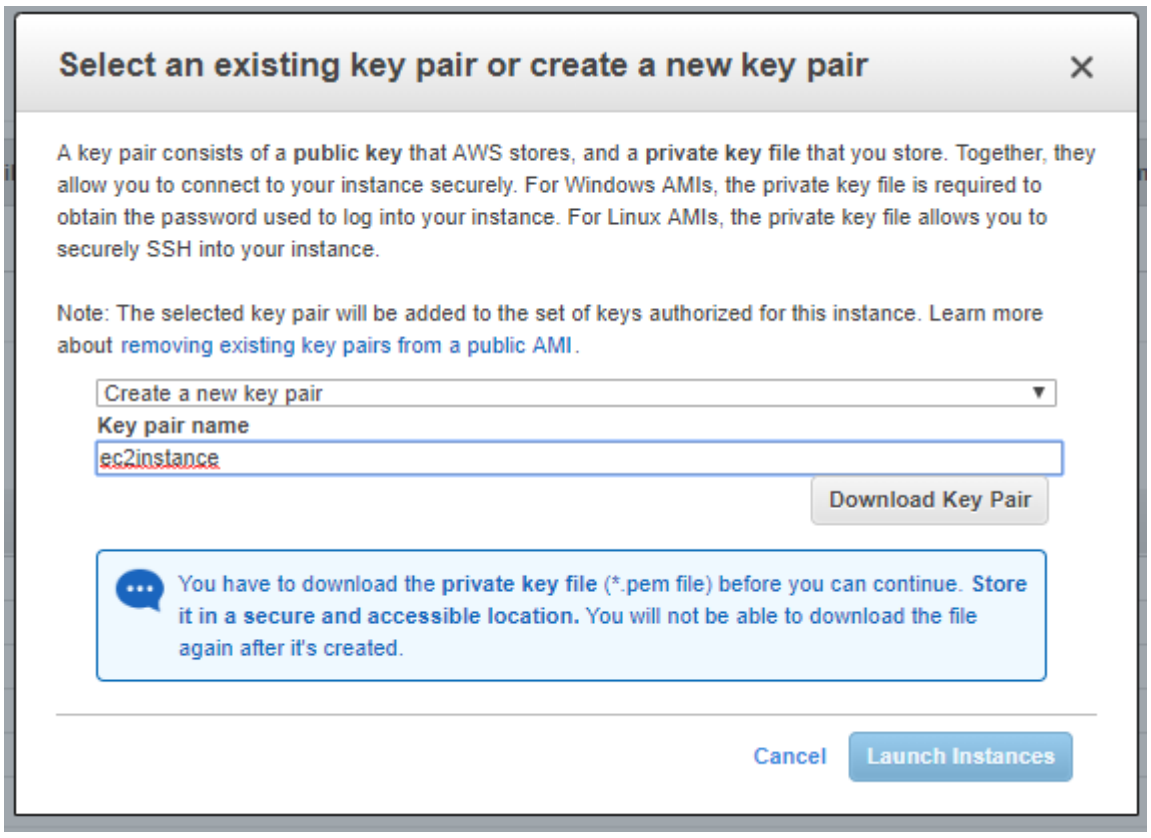
- Configure Security Group. The security group allows some specific traffic to access your instance.



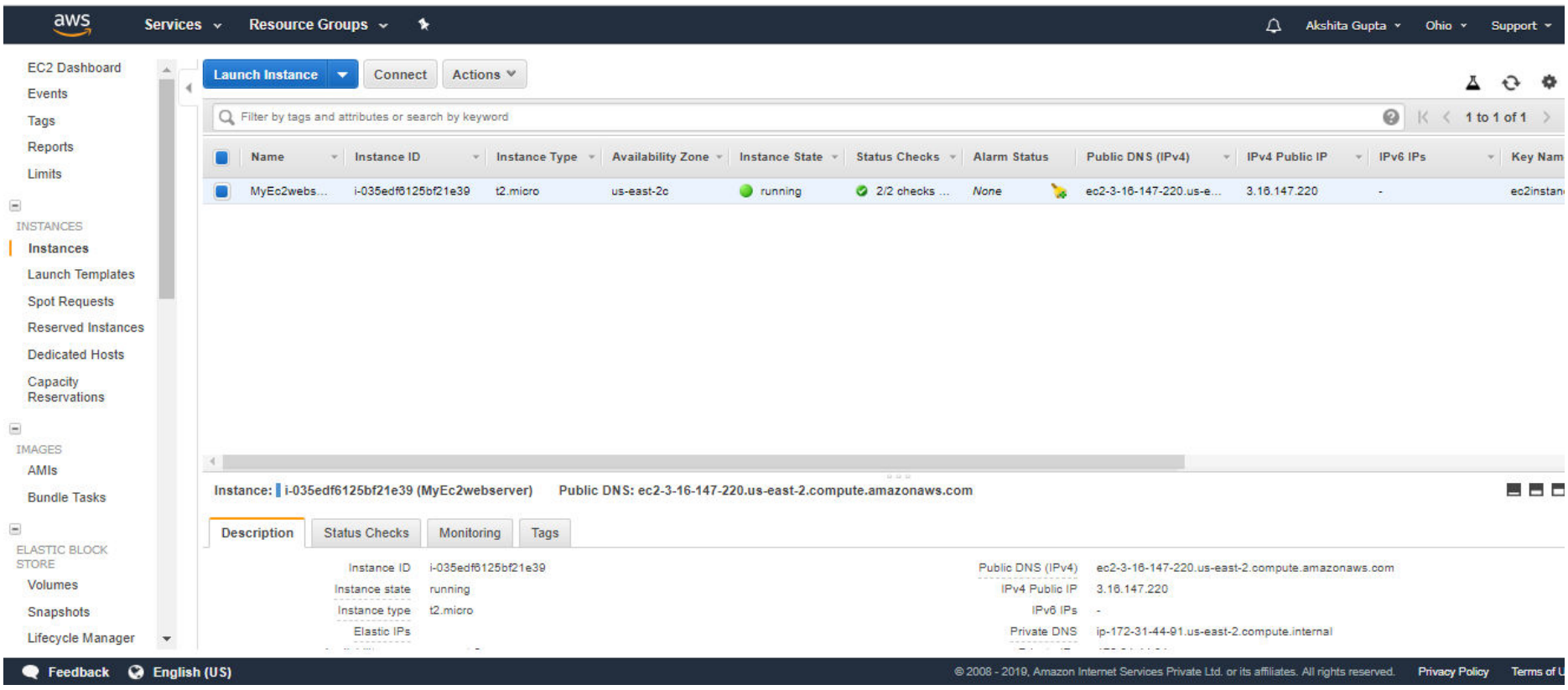
- Review an EC2 instance that you have just configured, and then click on the Launch button.



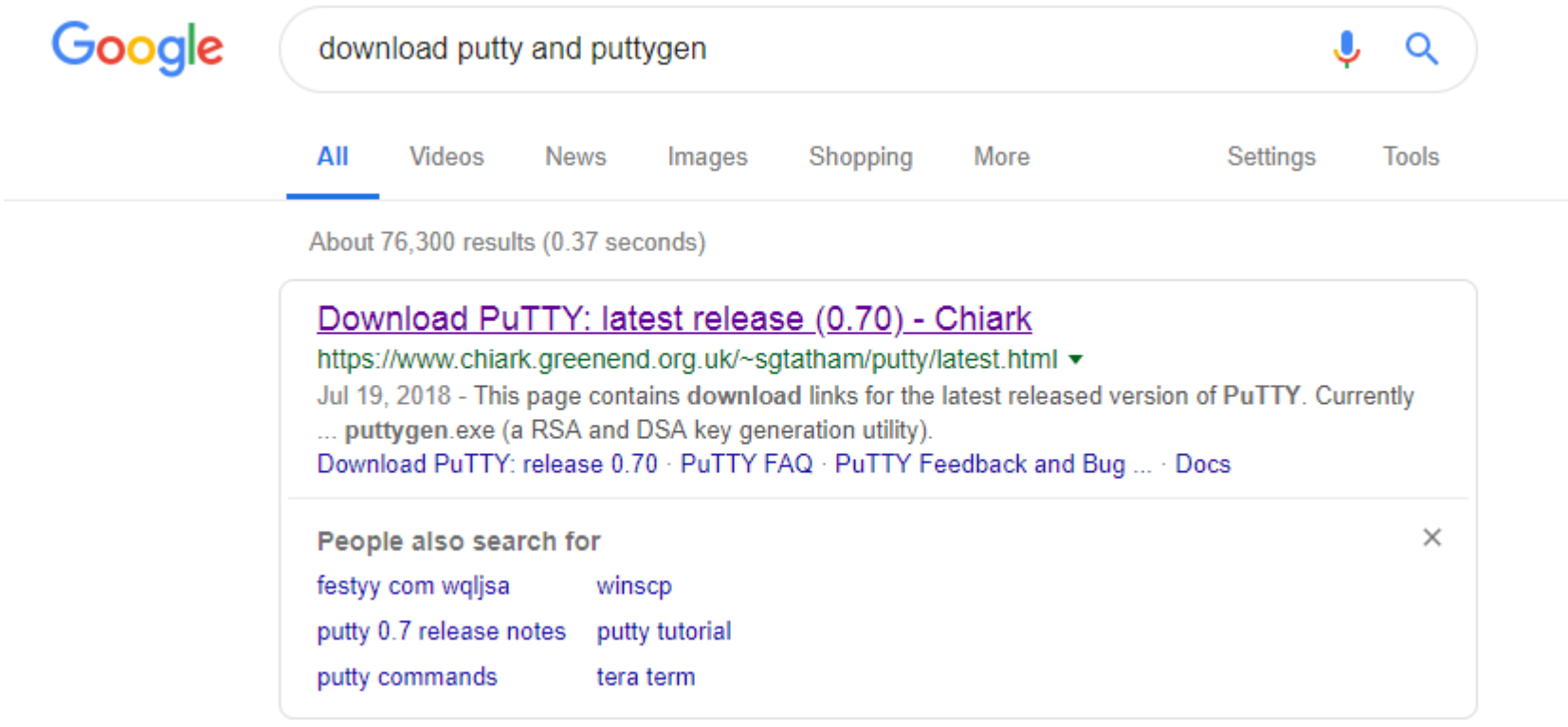
- Create a new key pair and enter the name of the key pair. Download the Key pair.



- Click on the Launch Instances button.



- To use an EC2 instance in Windows, you need to install both **Putty** and **PuttyKeyGen.**
- Download the **Putty** and **PuttyKeyGen.**



- Download the putty.exe and puttygen.exe file.

putty.exe (the SSH and Telnet client itself)

32-bit:	putty.exe	(or by FTP)	(signature)
64-bit:	putty.exe	(or by FTP)	(signature)

pscp.exe (an SCP client, i.e. command-line secure file copy)

32-bit:	pscp.exe	(or by FTP)	(signature)
64-bit:	pscp.exe	(or by FTP)	(signature)

psftp.exe (an SFTP client, i.e. general file transfer sessions much like FTP)

32-bit:	psftp.exe	(or by FTP)	(signature)
64-bit:	psftp.exe	(or by FTP)	(signature)

puttytel.exe (a Telnet-only client)

32-bit:	puttytel.exe	(or by FTP)	(signature)
64-bit:	puttytel.exe	(or by FTP)	(signature)

plink.exe (a command-line interface to the PuTTY back ends)

32-bit:	plink.exe	(or by FTP)	(signature)
64-bit:	plink.exe	(or by FTP)	(signature)

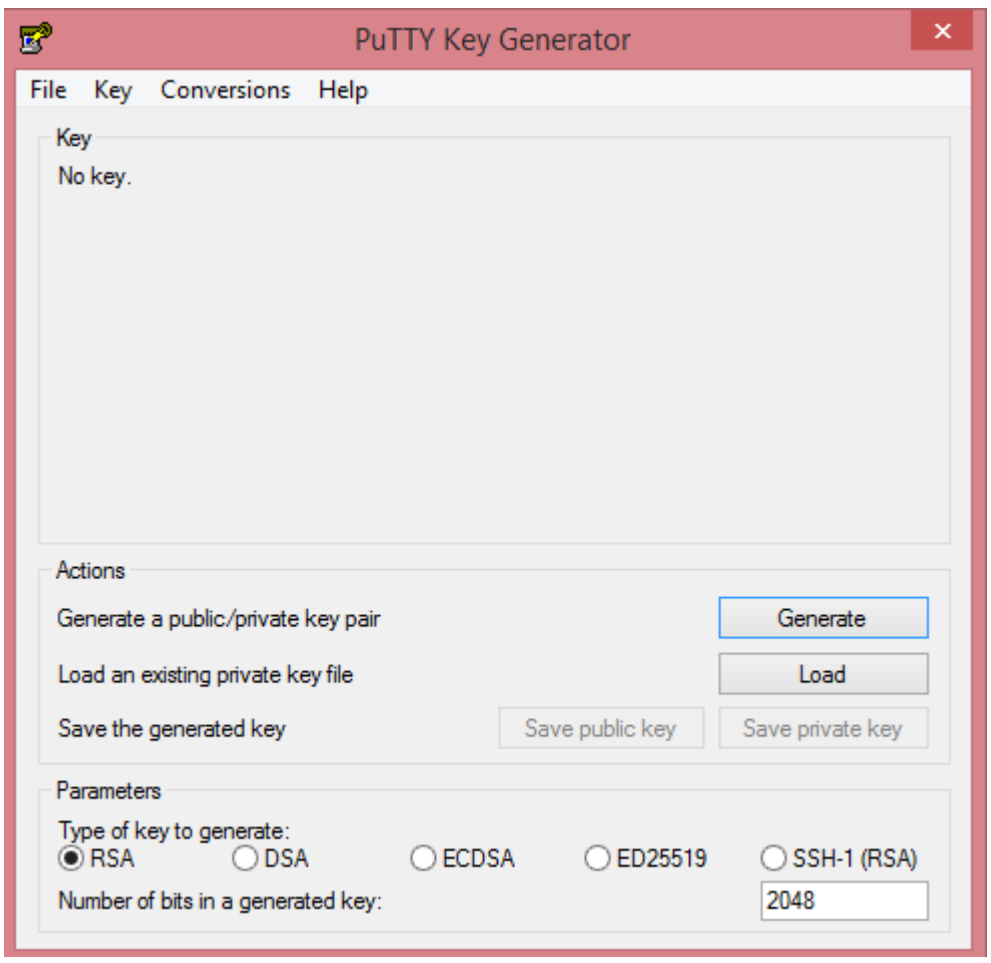
pageant.exe (an SSH authentication agent for PuTTY, PSCP, PSFTP, and Plink)

32-bit:	pageant.exe	(or by FTP)	(signature)
64-bit:	pageant.exe	(or by FTP)	(signature)

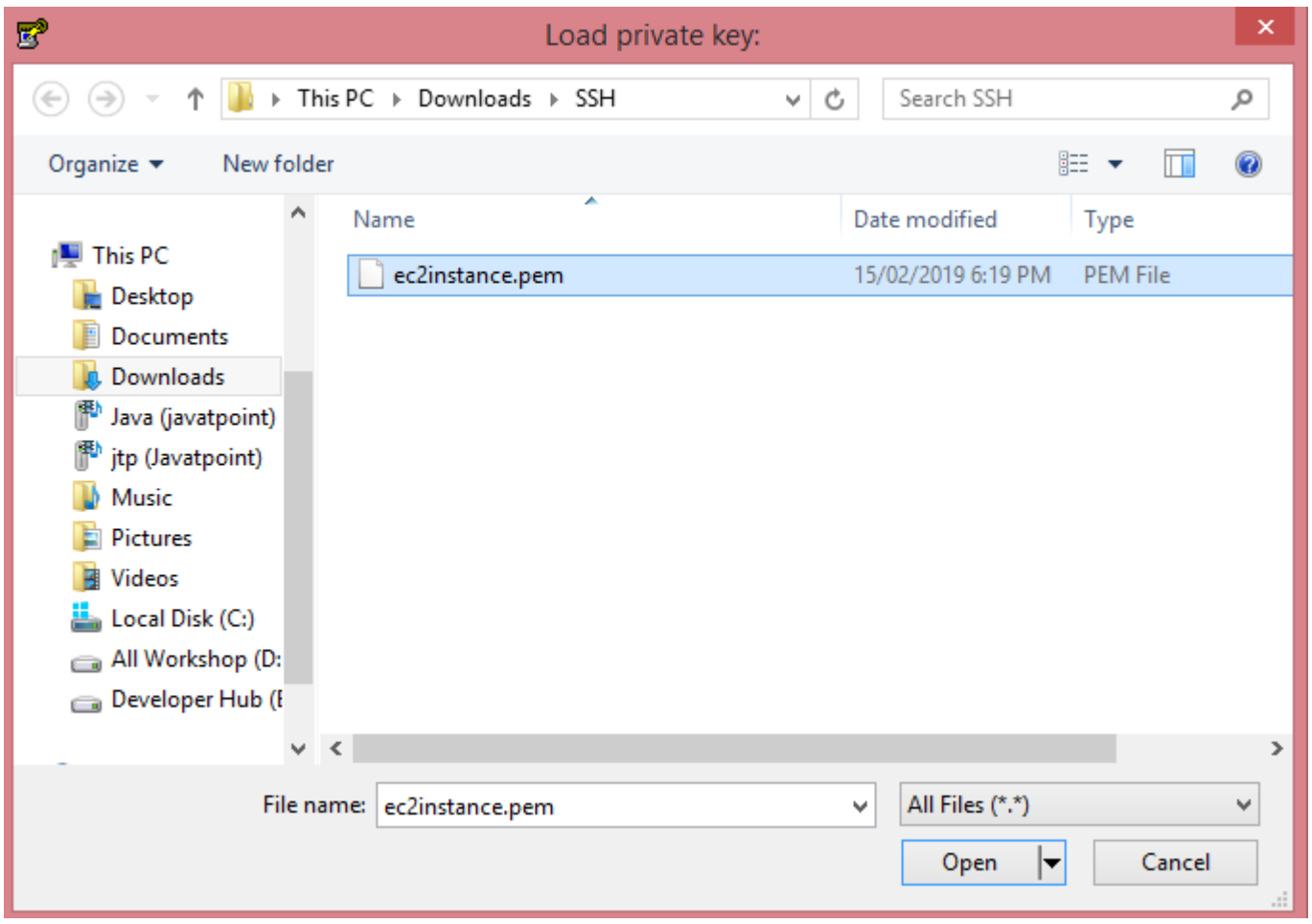
puttygen.exe (a RSA and DSA key generation utility)

32-bit:	puttygen.exe	(or by FTP)	(signature)
64-bit:	puttygen.exe	(or by FTP)	(signature)

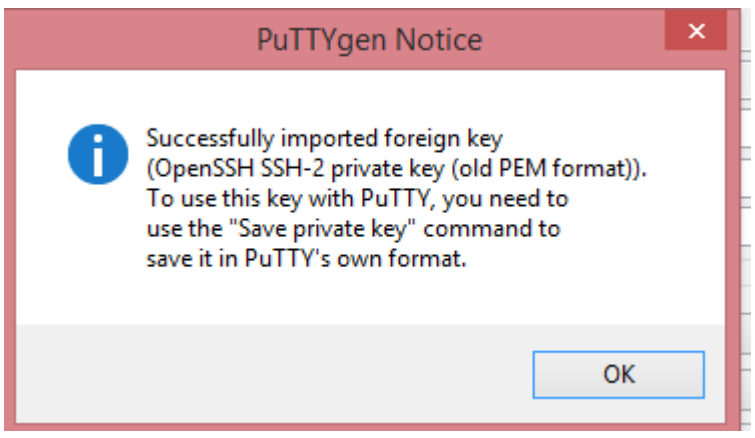
- In order to use the key-pair which we have downloaded previously, we need to convert the pem file to ppk file. Puttygen is used to convert the pem file to ppk file.
- Open the Puttygen software.
- Click on the Load.



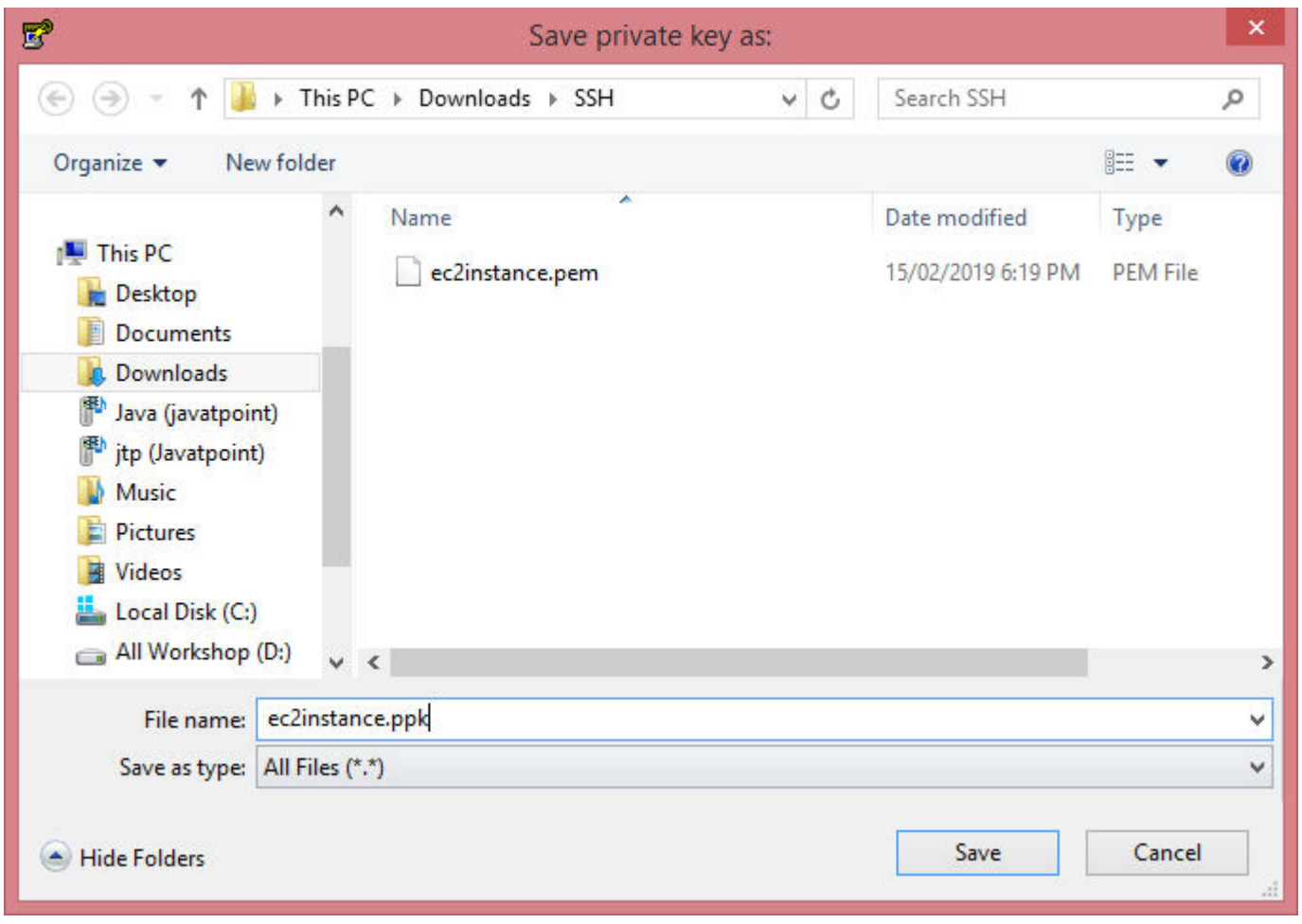
- Open the key-pair file, i.e., ec2instance.pem.



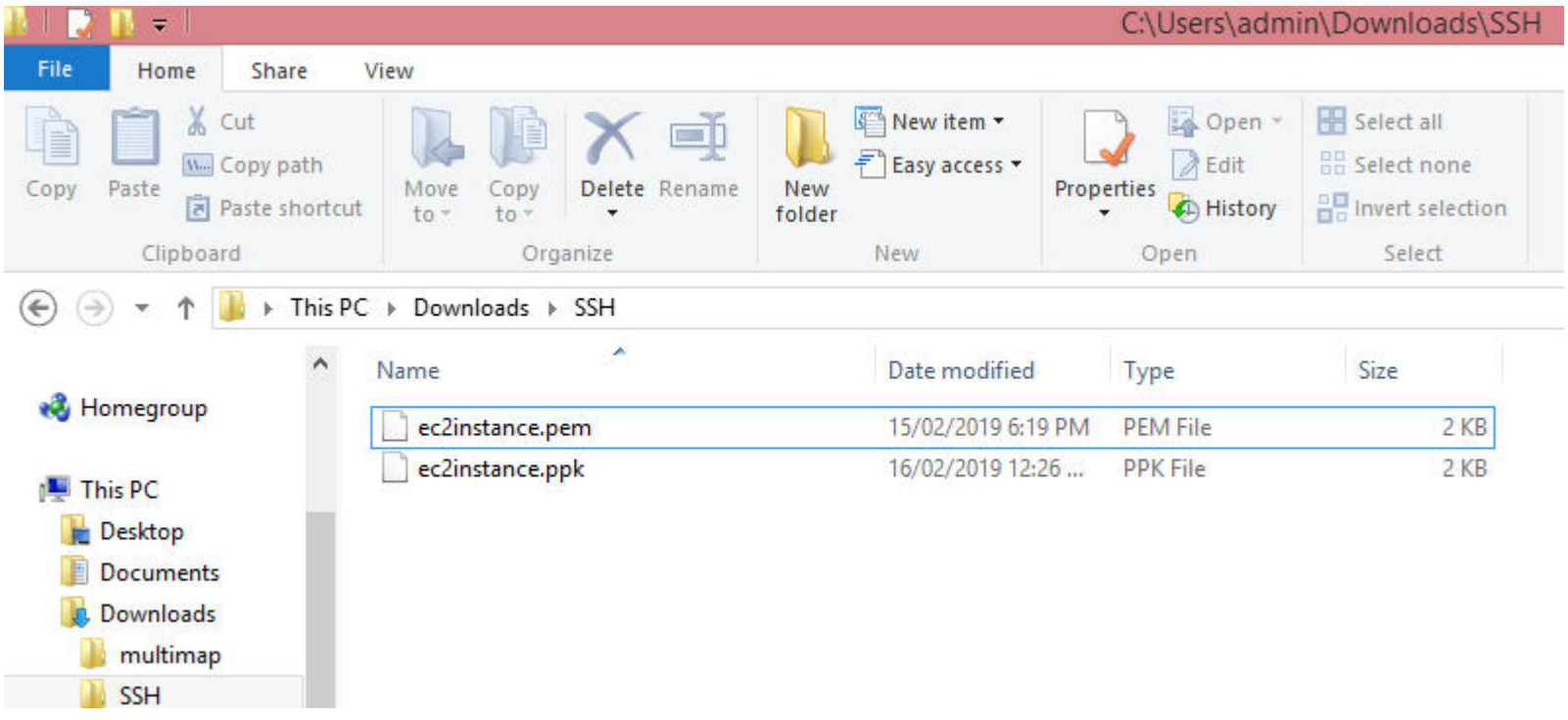
- Click on the OK button.



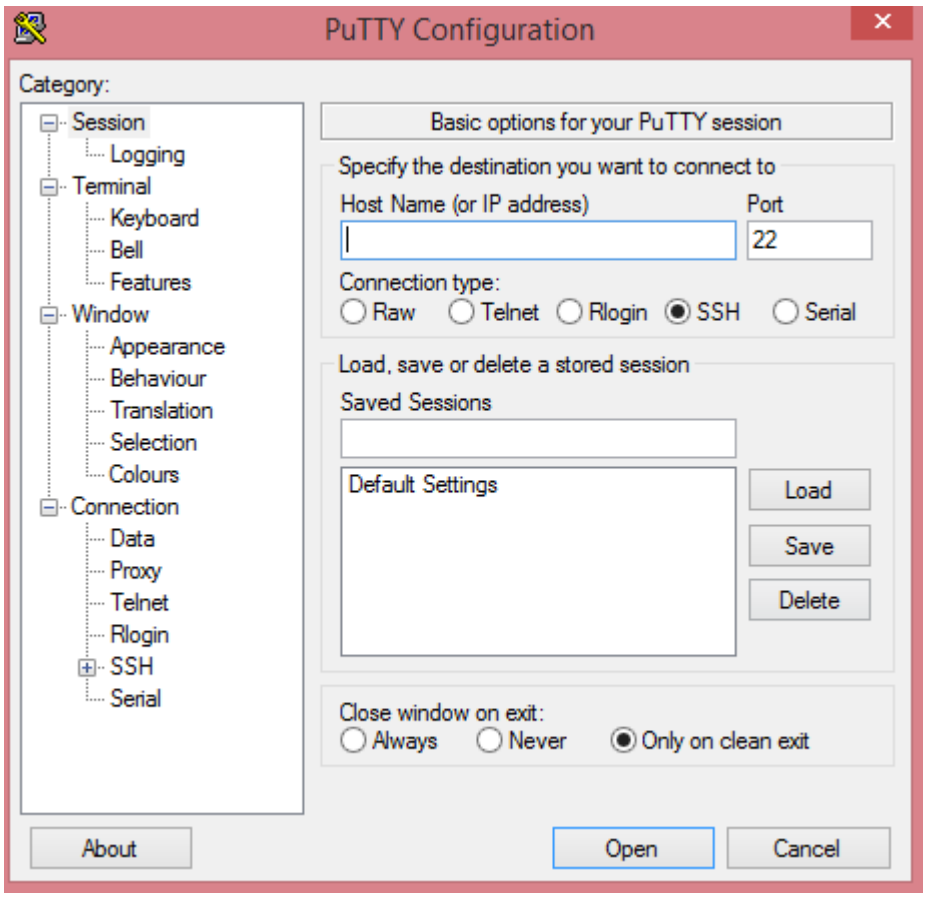
- Click on the Save private key. Change the file extension from pem to ppk.



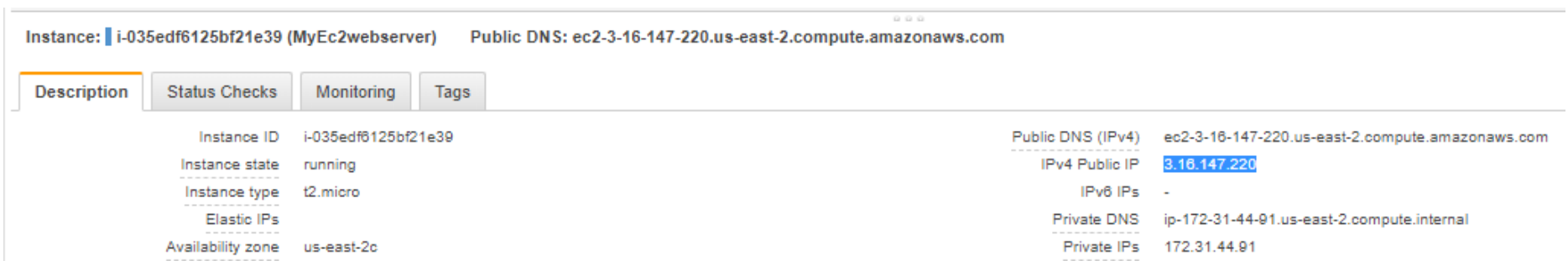
- Click on the Save button.
- Move to the download directory where the ppk file is downloaded.



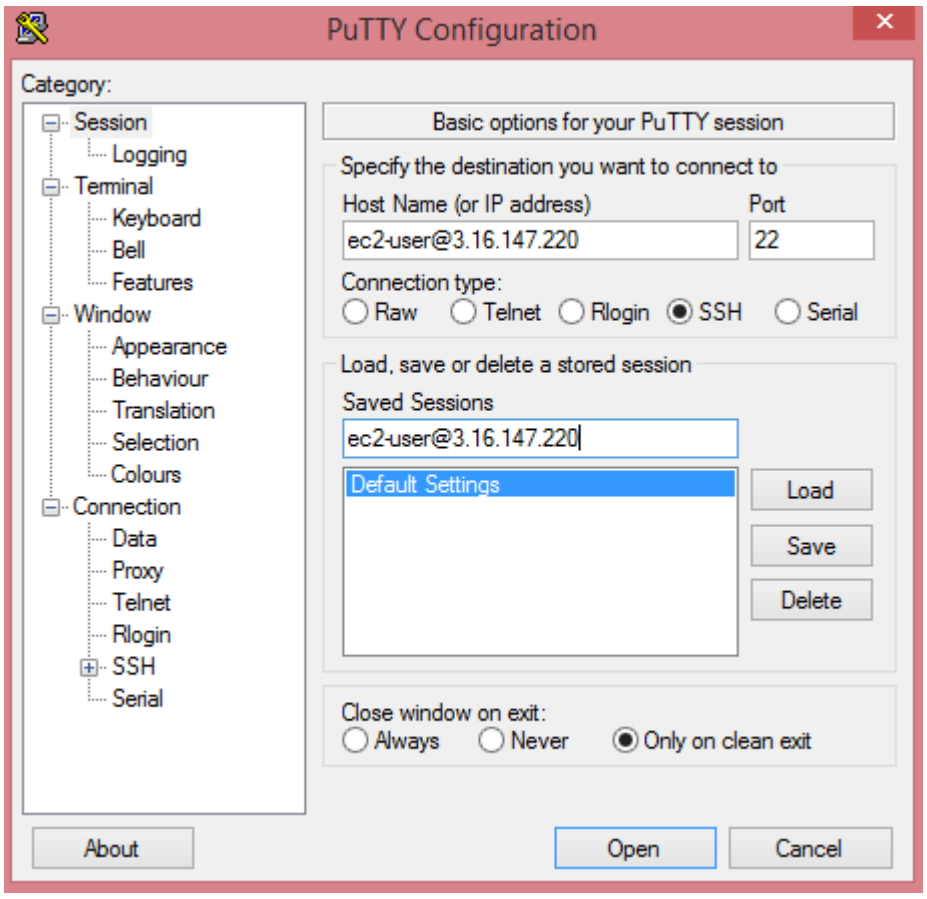
- Open the Putty.



- Move to the EC2 instance that you have created and copy its IP address.

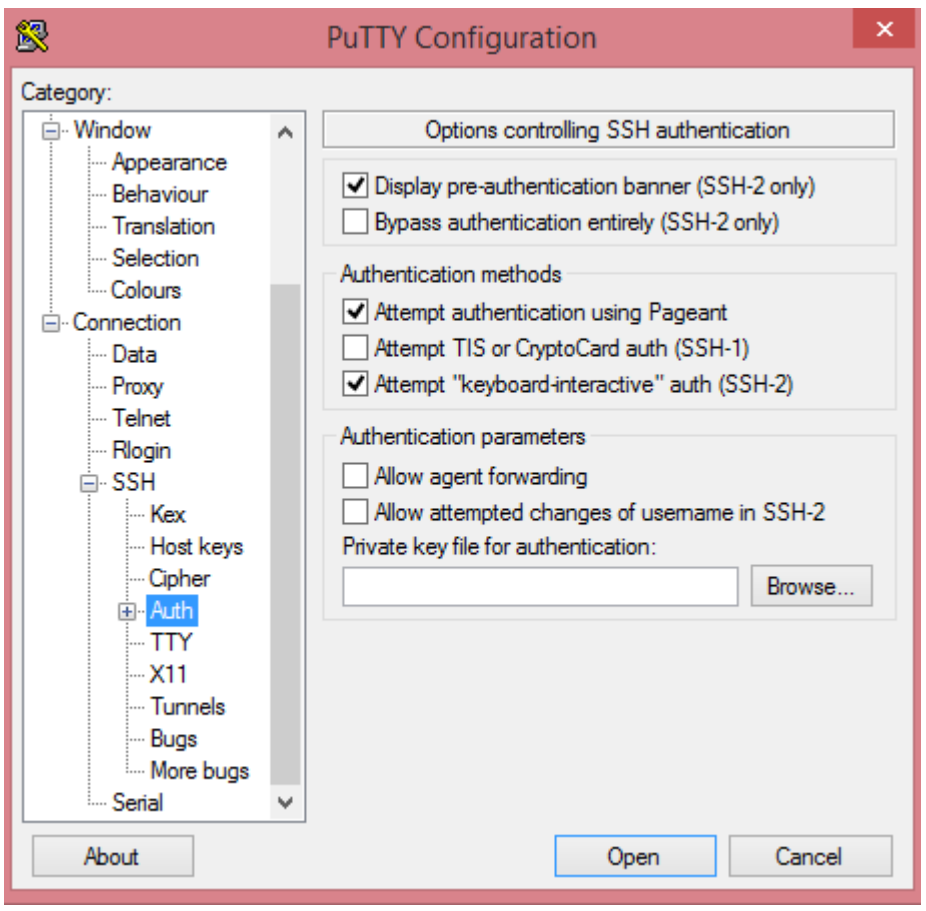


- Now, move to the Putty configuration and type ec2user@, and then paste the IP address that you have copied in a previous step. Copy the Host Name in Saved Sessions.

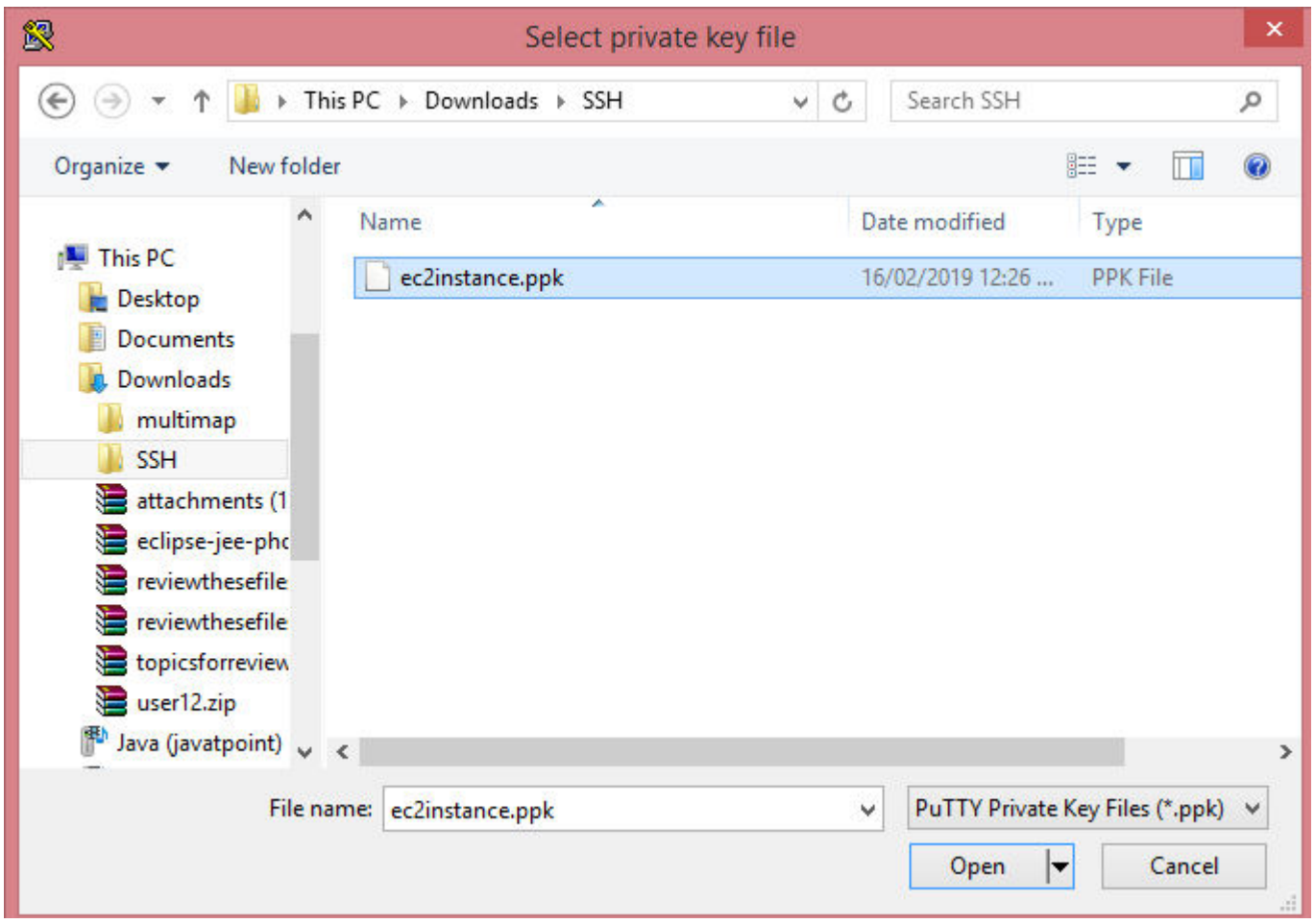


Now, your Host Name is saved in the default settings.

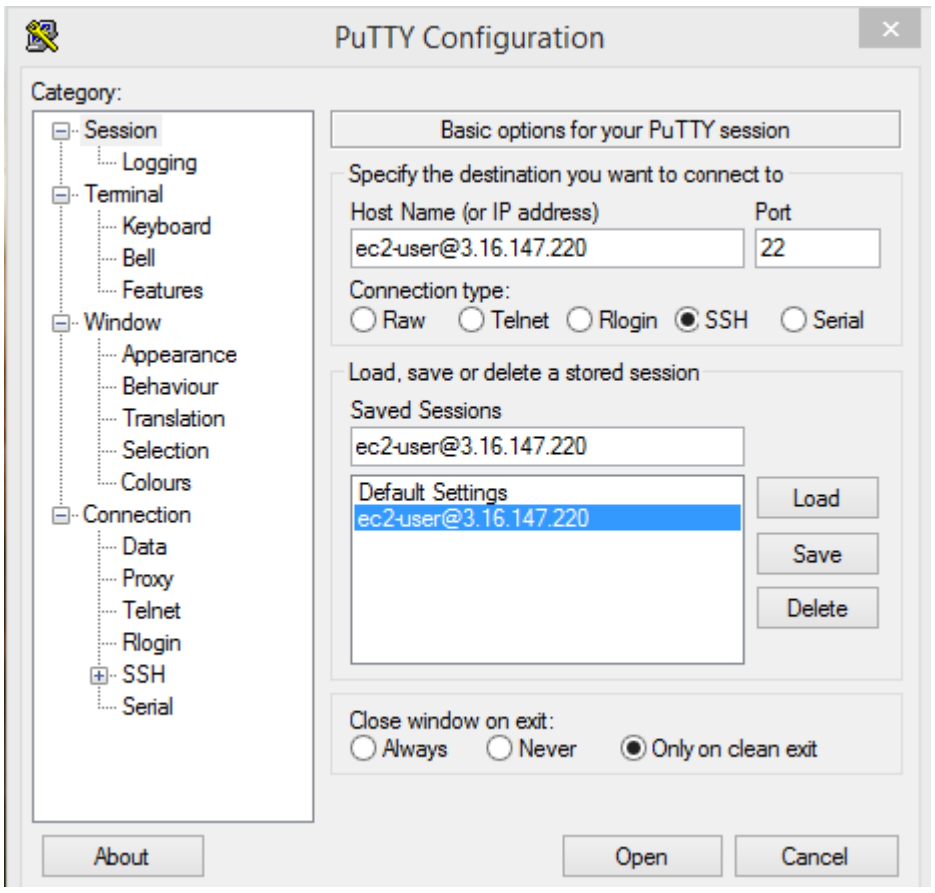
- Click on the SSH category appearing on the left side of the Putty, then click on the Auth.



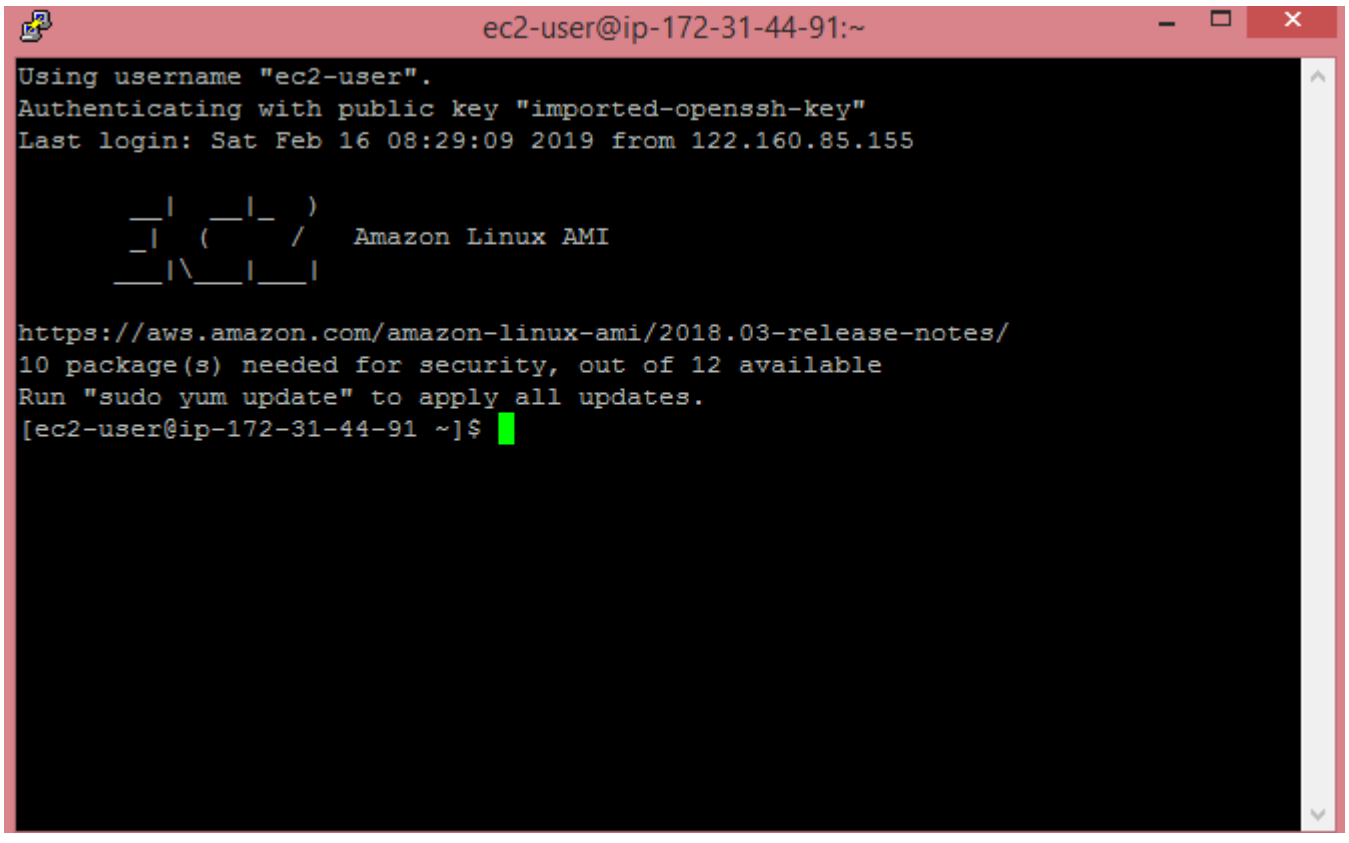
- Click on the Browse to open the ppk file.



- Move to the Session category, click on the Save to save the settings.

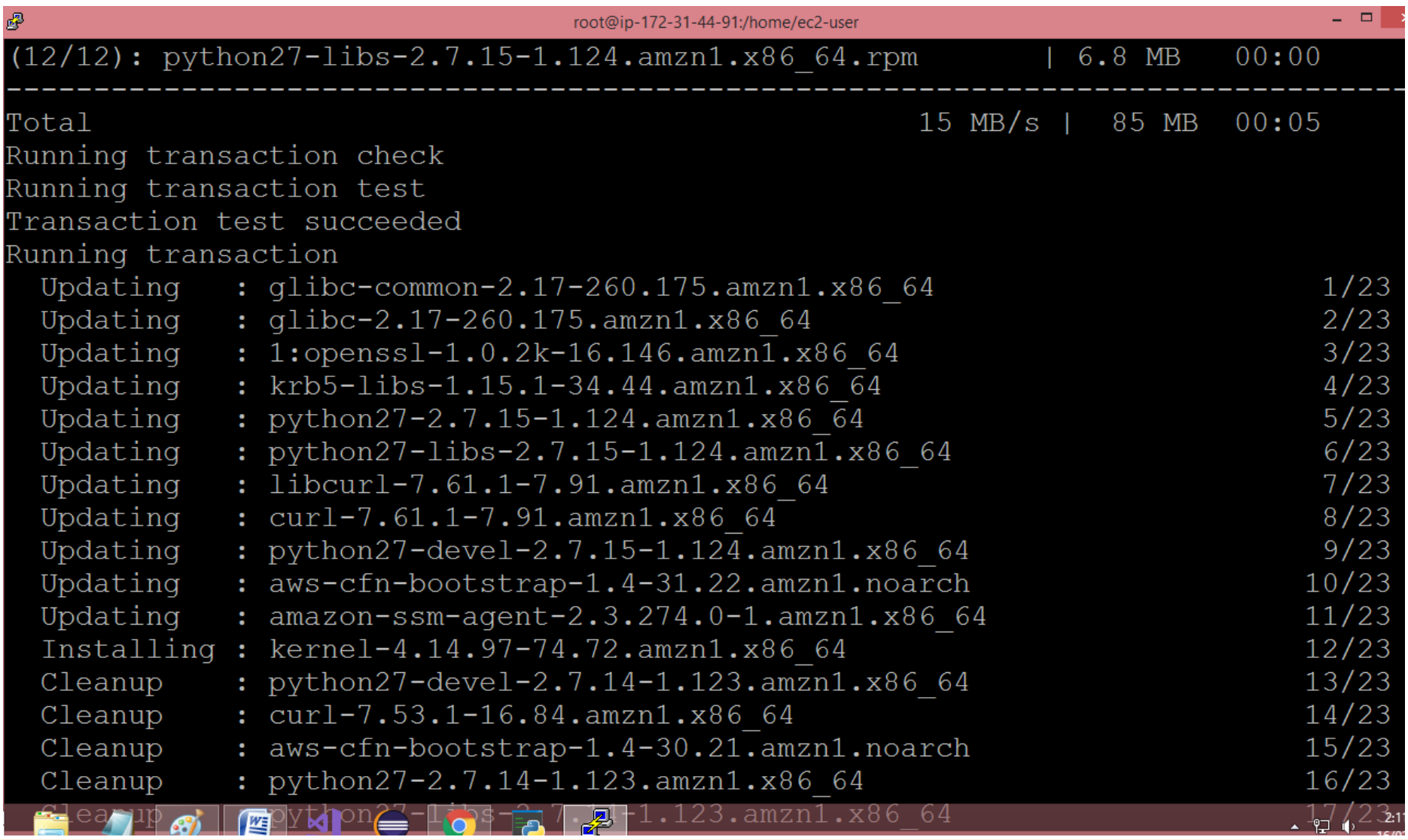


- Click on the Open button to open the Putty window.



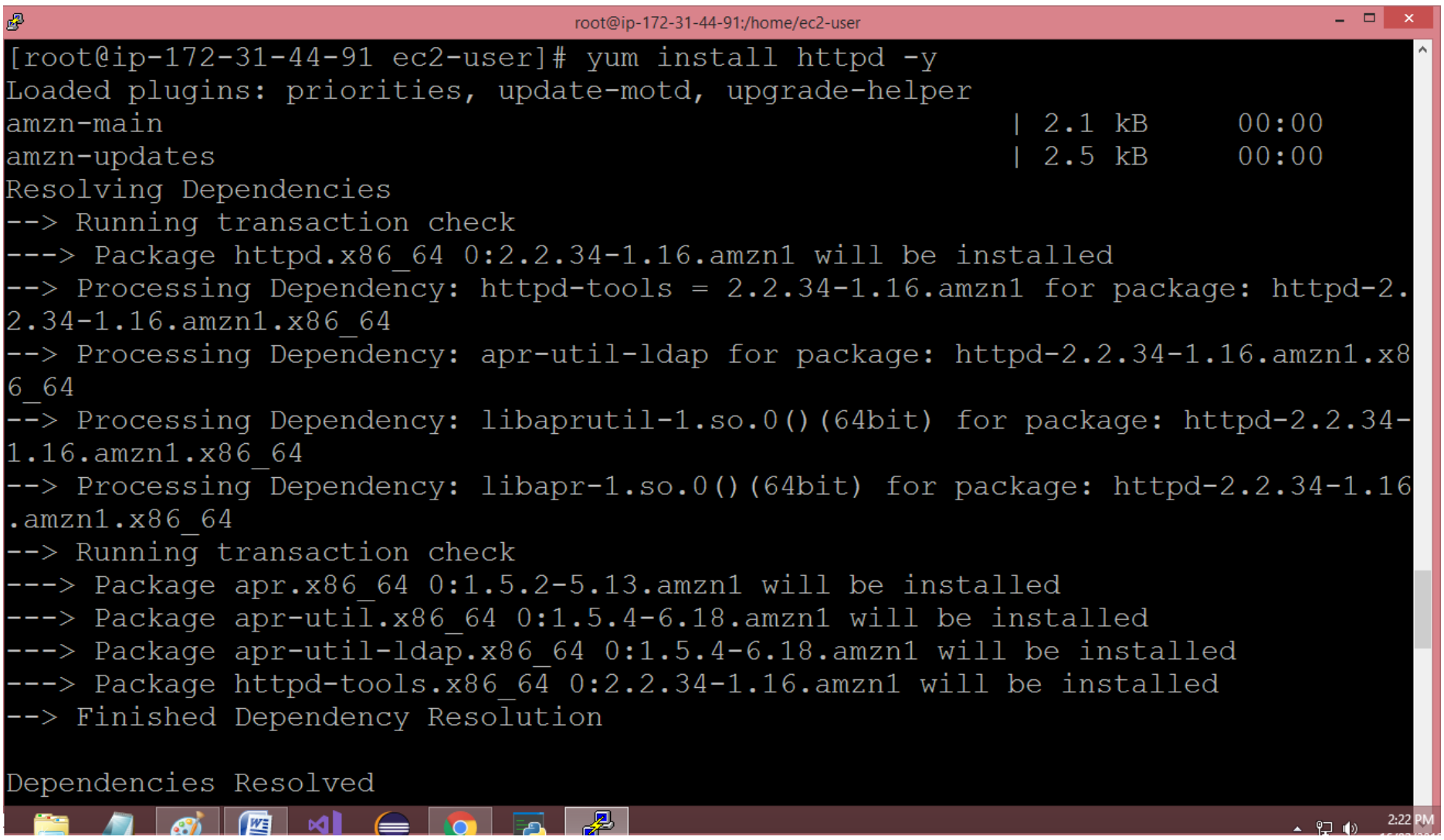
The above screen shows that we are connected to the EC2 instance.

- Run the command **sudo su,** and then run the command **yum update** -y to update the EC2 instance.

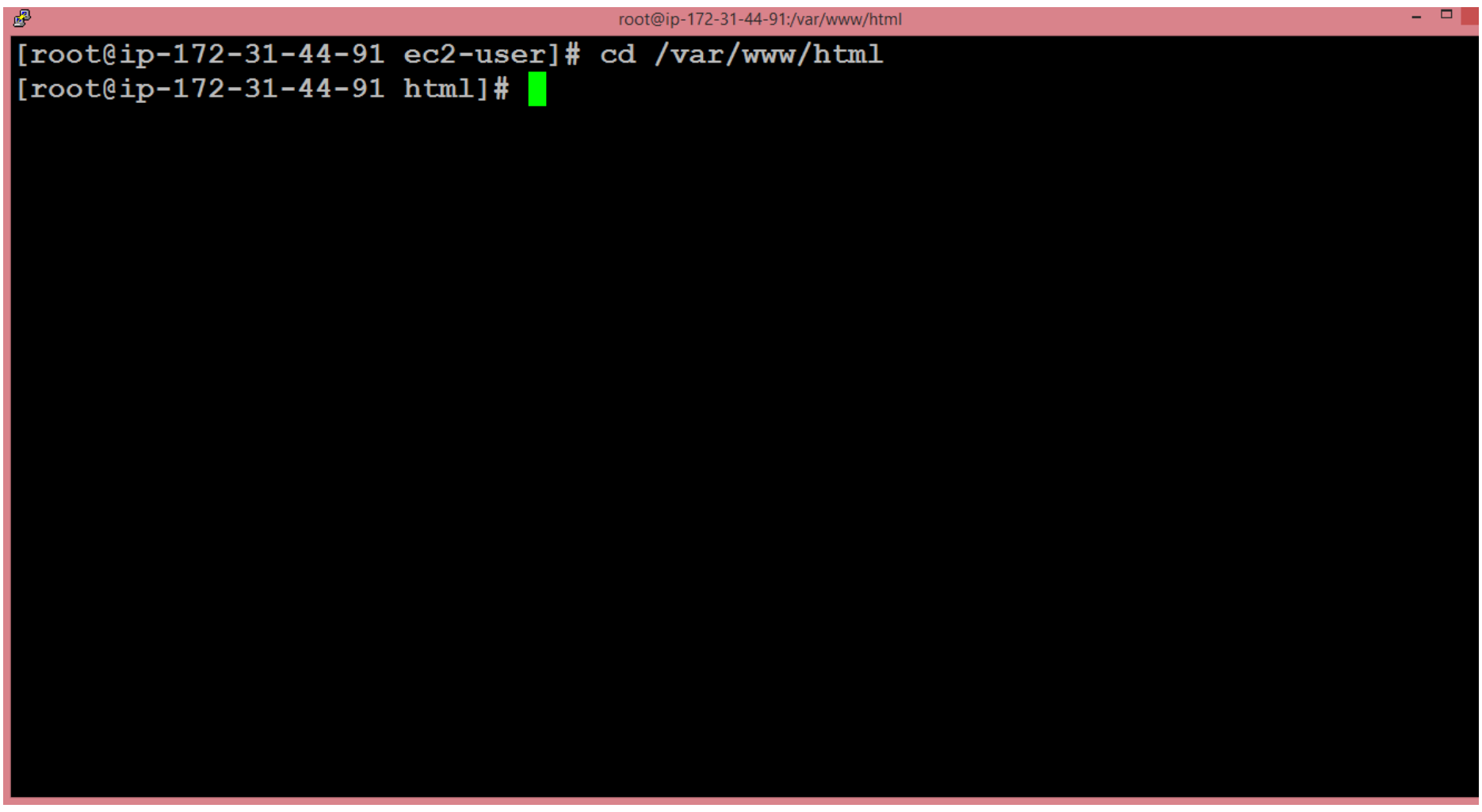


Note: sudo su is a command which is used to provide the privileges to the root level.

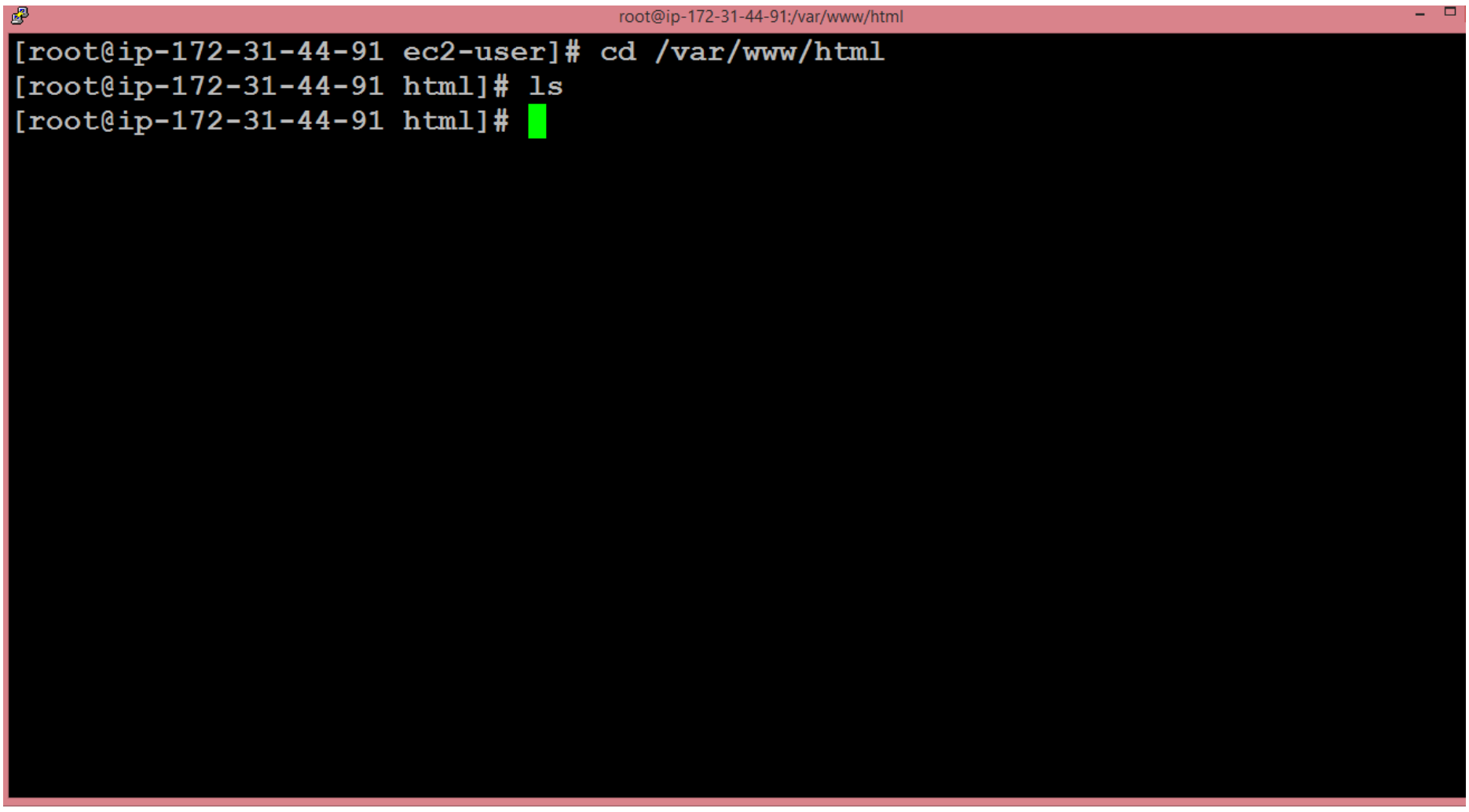
- Now, we install the apache web server to ensure that an EC2 instance becomes a web server by running a command **yum install httpd** -y.



- Run the command **cd /var/www/html.**

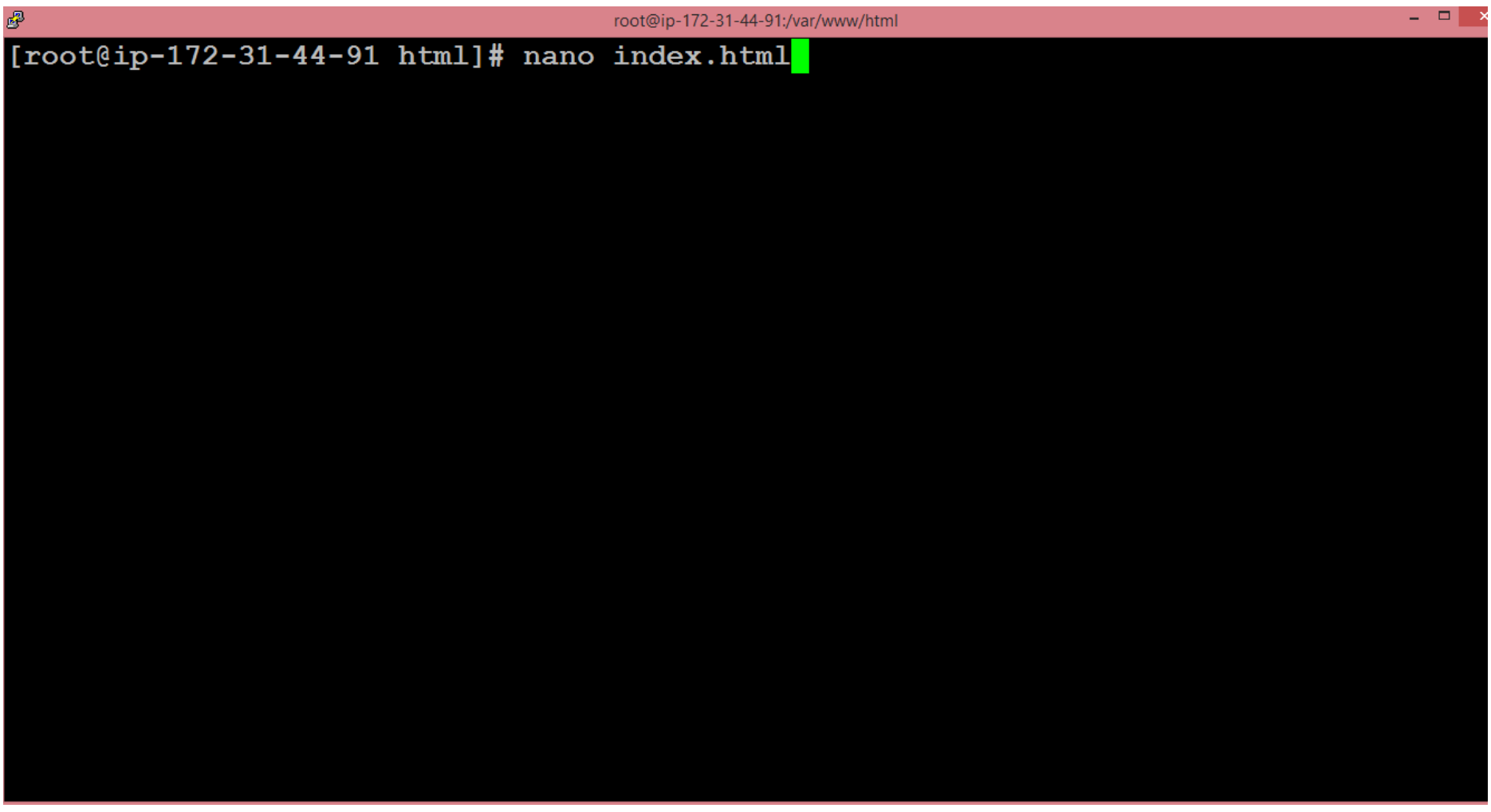


- To list the files available in the html folder, run the command **ls.**

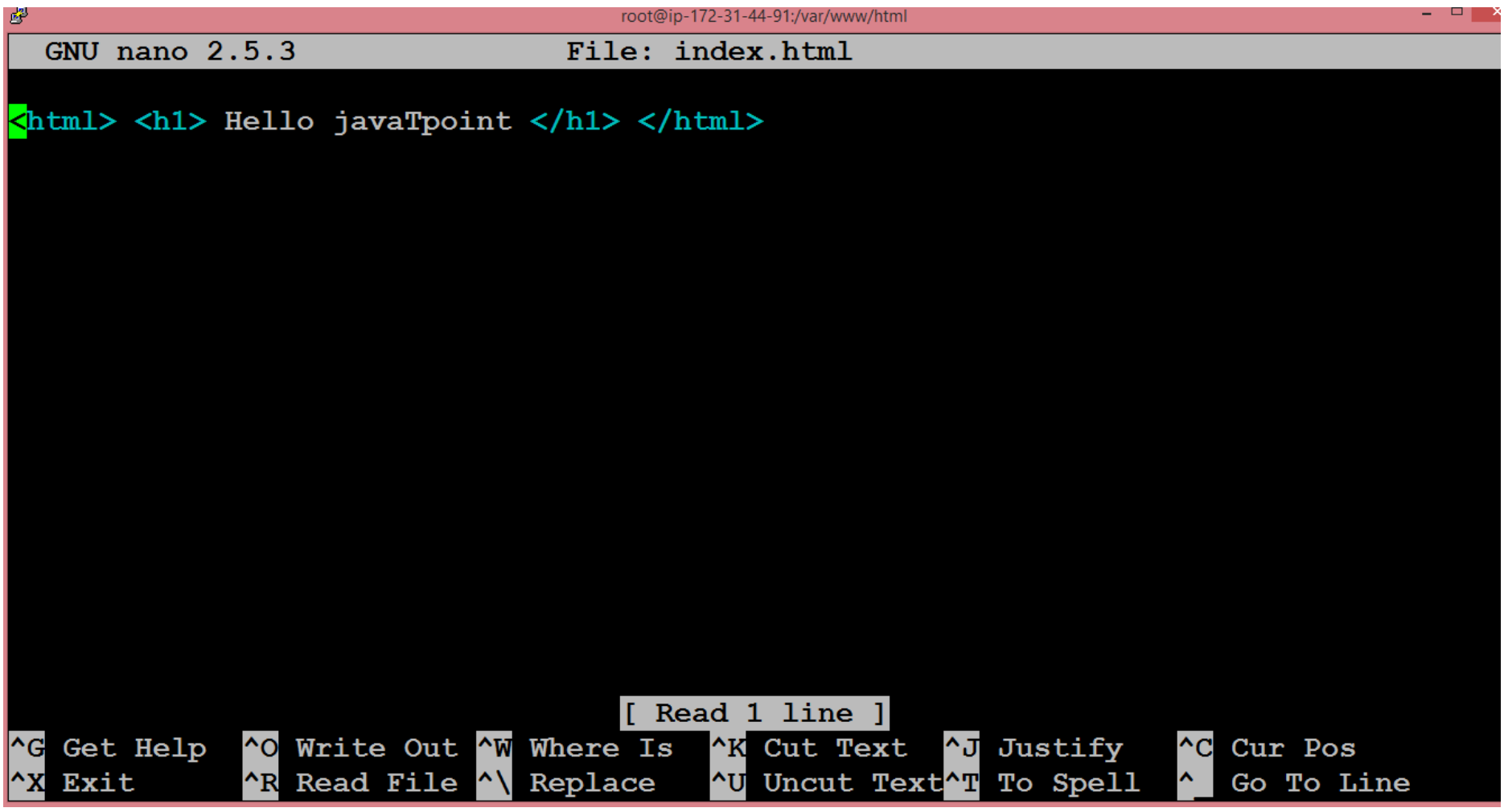


We observe that on running the command **ls,** we do not get any output. It means that it does not contain any file.

- We create a text editor, and the text editor is created by running the command **nano index.html** where index.html is the name of the web page.

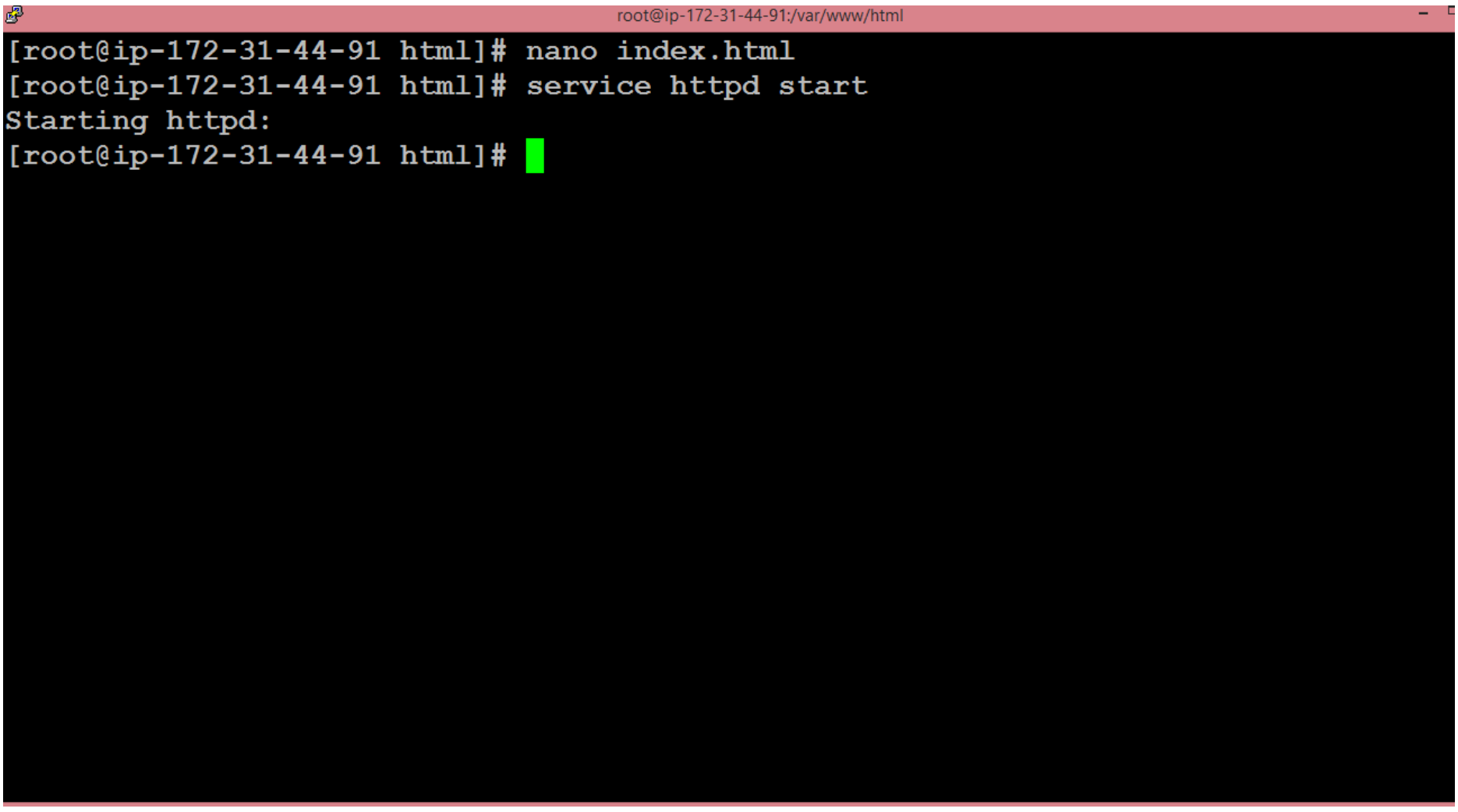


- The text editor is shown below where we write the HTML code.



After writing the HTML code, press ctrl+X to exit, then press 'Y' to save the page and press Enter.

- Start the Apache server by running the command **service httpd start.**



- Go to the web browser and paste the IP address which is used to connect to your EC2 instance. You will see the web page that you have created.

Elastic IPs

An advantage of using Amazon Elastic Compute Cloud (EC2) is the ability to start, stop, create, and terminate instances at any time. However, this flexibility creates a potential challenge with IP addresses. Restarting a stopped instance (or re-creating an instance after another instance is terminated) results in a new IP address. How do you successfully reference a machine when the IP address is constantly changing?

In response to this problem, Amazon offers the ability to allocate an Elastic IP address. An Elastic IP provides you a single IP address that you can associate with different EC2 instances over time. If your EC2 instance has an Elastic IP and that instance is ever stopped or terminated, you can immediately associate a new EC2 instance with the Elastic IP. Your existing applications will not break because the applications see the IP address they were expecting, even though the back-end EC2 instance has changed.

If you're building your own site manually, you should associate an Elastic IP with your enterprise geodatabase instance if you have the geodatabase on its own dedicated instance. Optionally, you can also associate an Elastic IP with your ArcGIS Server instance if you are not already using an Amazon Load Balancer.

If you use ArcGIS Server Cloud Builder on Amazon Web Services to build your site, no Elastic IPs are created. If you want to use SSH or Remote Desktop Connection to connect to instances within your site, you need to create an Elastic IP and assign one to each instance.

If you ever need to stop an instance, you should reassociate it with its Elastic IP after you start the instance again. You can even associate the Elastic IP with a backup instance while the other instance is down. If you don't have an Elastic IP, users' connections to your machines will permanently break if you ever have to stop the instance.

To allocate an Elastic IP and associate it with an Amazon Web Services (AWS) instance, do the following:

1. Open the AWS Management Console, click the **EC2** link, and display the page associated with your region.
2. Click the **Elastic IPs** link in the **EC2 Dashboard**.
3. Click **Allocate New Address** and choose **VPC** or **EC2** from the drop-down list, depending whether you're going to associate this IP with an instance in Amazon EC2-Virtual Private Cloud (VPC) or Amazon EC2-Classic, respectively. Click **Yes, Allocate** to confirm your choice.
4. Right-click the newly created Elastic IP and choose **Associate Address**.
5. Choose your desired EC2 instance from the drop-down list of running instances and click **Associate**.

Placement Groups

When you launch multiple EC2 instances on AWS, the EC2 service makes sure that all of your EC2 instances are spread across different physical machines to minimize the failure of the entire system. But AWS EC2 also provides the customers the ability to put the EC2 instance according to their need. Placement groups are used to determine how the EC2 instances are launched on the underlying hardware. AWS provides the following three types of placement groups strategies which you can use according to your workload.

- **Cluster placement group:** It groups instances into low latency clusters in a single available zone(AZ).
- **Spread placement group:** It spread the instances across underlying hardware.
- **Partition placement group:** It spreads the instances across many different partitions within an AZ.

Cluster Placement Group

In the cluster placement group, all the instances are in the same rack in a single availability zone. Cluster placement groups are designed for high speed performance and low network latency applications as EC2 instances are physically on the same rack and it causes low latency between the EC2 instances in the same cluster placement group. It usually supports up to 10Gbps network. As the EC2 instances in the cluster placement group are in the same physical rack so the problem with cluster placement groups is if the rack fails, all the instances will fail at the same time compromising the high availability of the application.

Spread Placement Group

In the spread placement group, all EC2 instances are located on different hardware racks in a single availability zone. Each rack is isolated from others and has its own power and networks to reduce the failure of all the instances in the spread placement group at a time. You can create up to 7 EC2 instances per availability zone per spread placement group. Unlike Cluster placement groups, EC2 instances in the spread placement group exist on different hardware within the single availability zone minimizing the failure of all the EC2 instances at a time while making sure of the low latency. Spread placement groups are designed for applications that require maximum high availability and where each instance must be isolated from failure from each other.

Looking for Hybrid Cloud That Is Right for You?

By Hewlett Packard Enterprise

Partition Placement Group

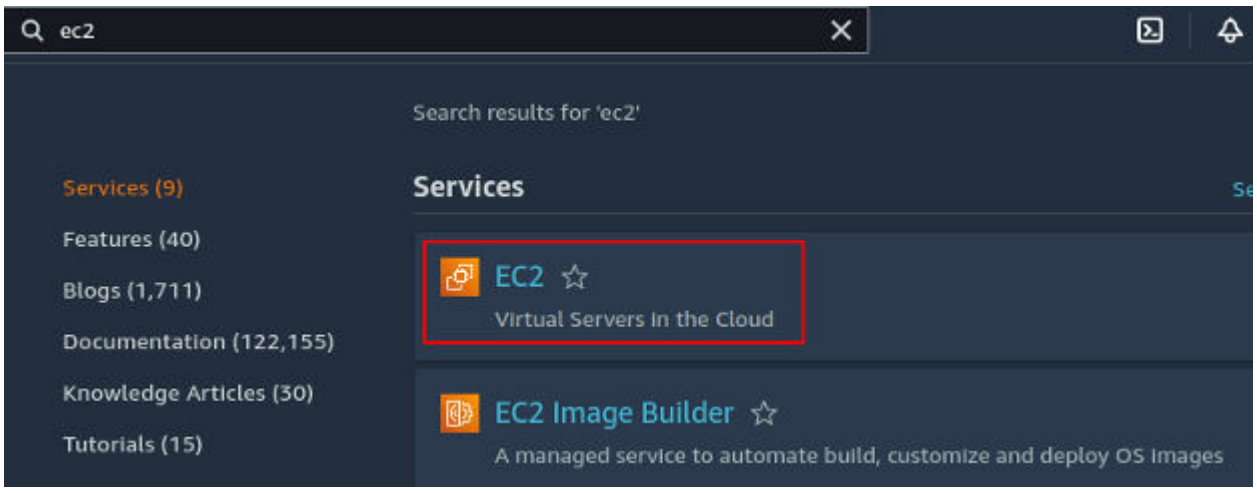
In the partition placement group, instances are launched into different partitions on different hardware racks to make sure of high availability. It can span across multiple AZs in the same region. The instances in a partition do not share racks with the instances in the other partitions. A partition failure can affect many EC2 instances in the same partition but won't affect the EC2 instances on the other partitions. Partition placement groups are designed for applications that require maximum high availability. Partition placement groups are used for big application deployment and are ideal for large distributed and replicated workloads such as kafka, hadoop and cassandra etc.

Creating a placement group

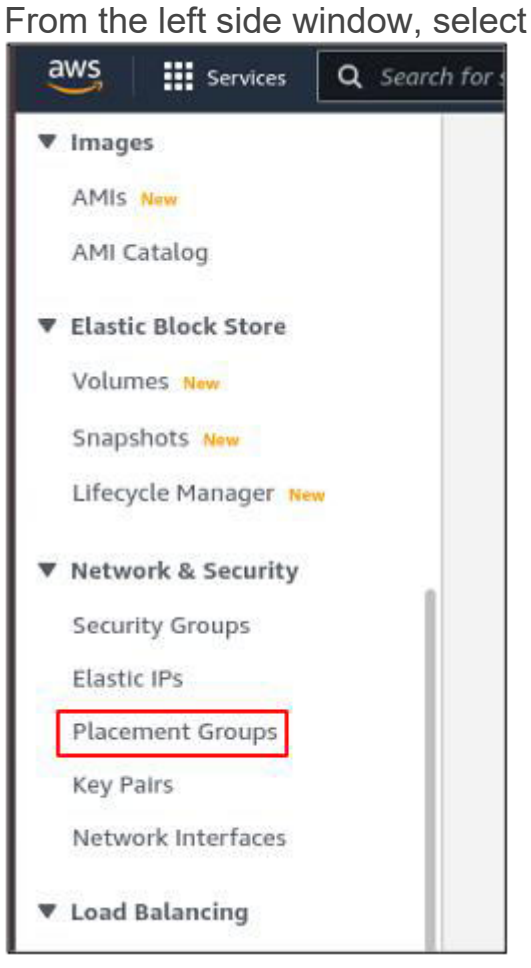
Now in this section, we will see how we can create placement groups on AWS using the AWS management console and AWS command line interface.

Creating Placement Group Using AWS Console Management

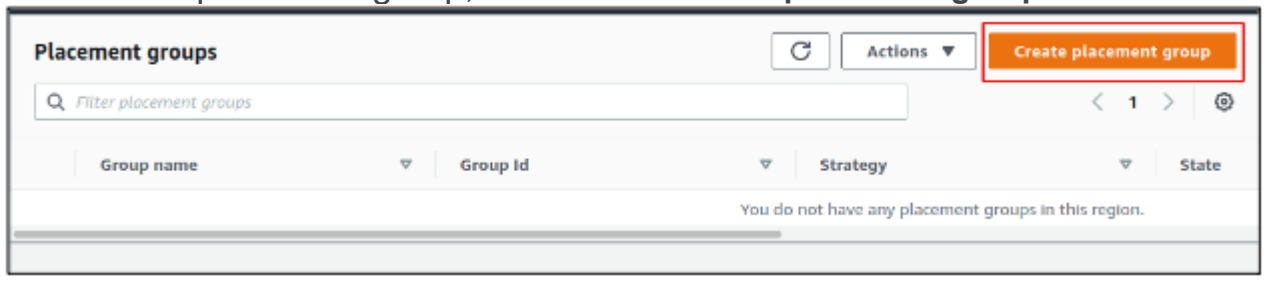
First, log into the AWS management console and search for the EC2 service in the search bar.



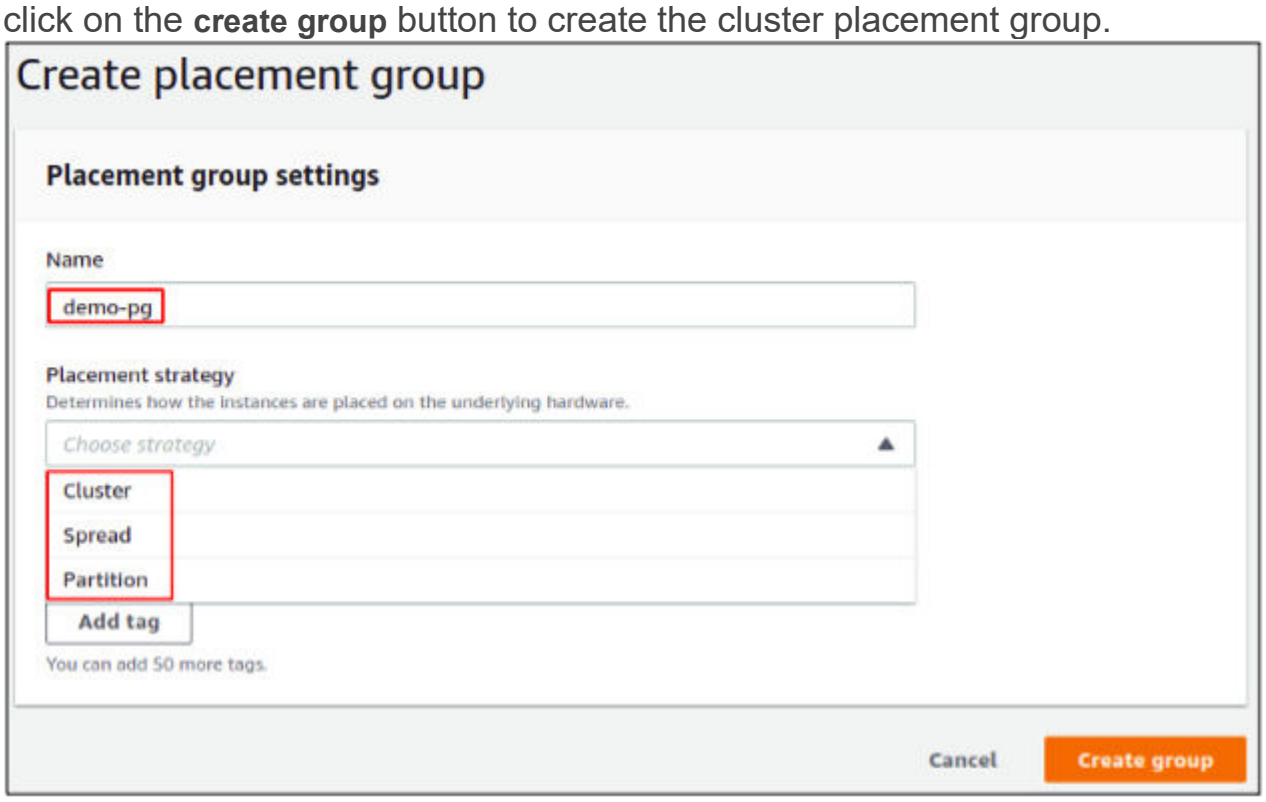
From the left side window, select the **Placement Groups** option.



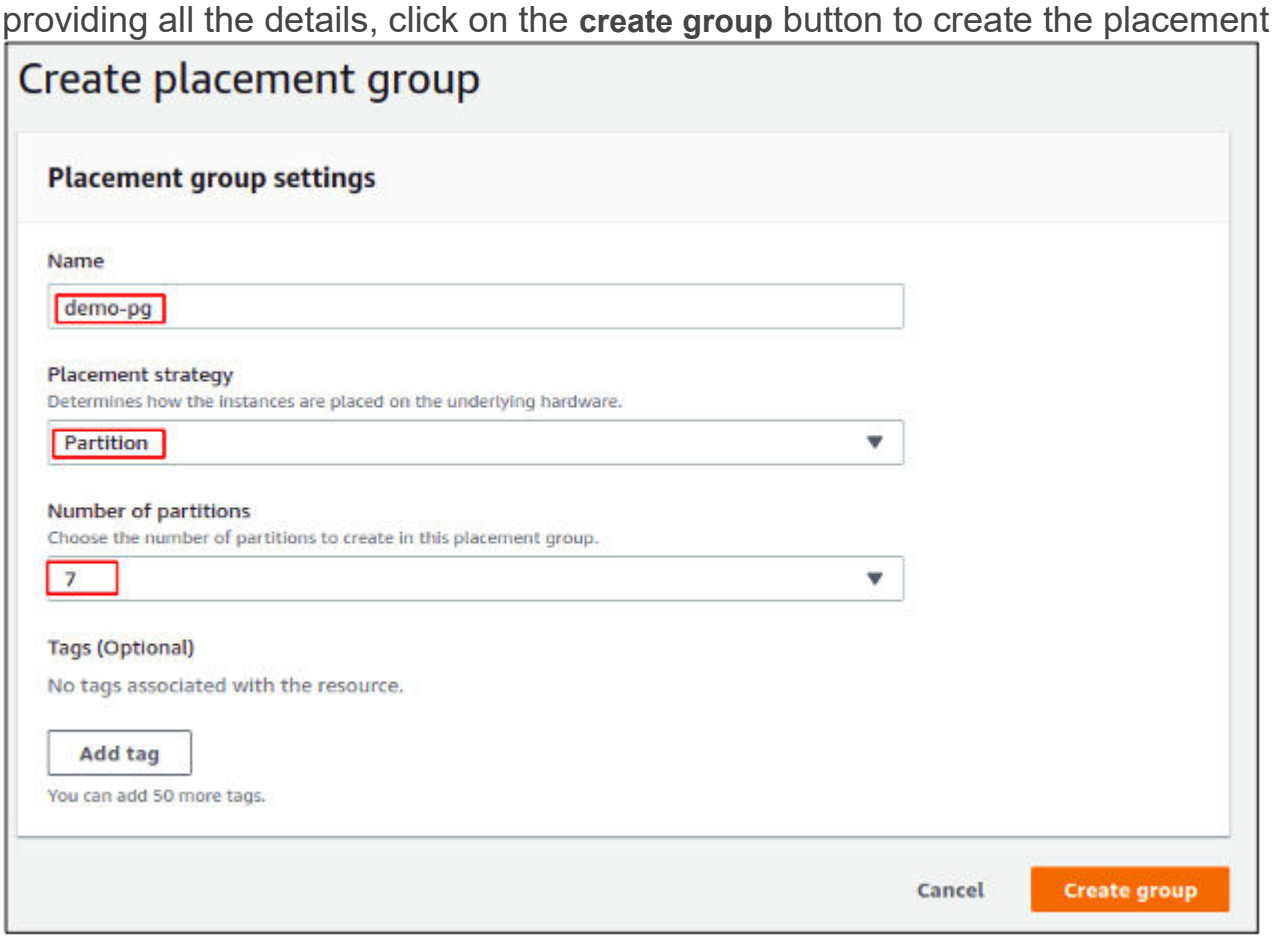
To create a placement group, click on the **Create placement group** button on the top right corner of the page.



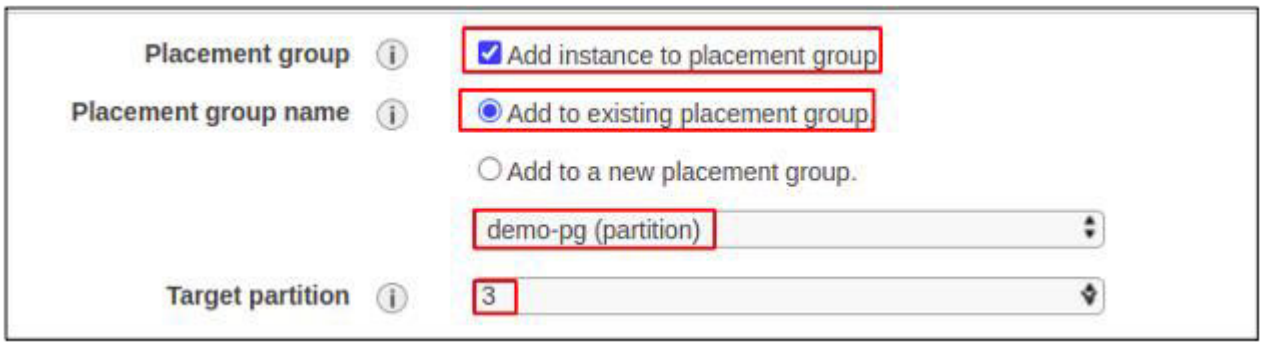
Now it will ask for the name and the placement group strategy. Provide a name to your placement group whatever you want. In placement strategy, you need to choose the type of placement group i.e. cluster, spread and partition. For this demo, we will create the cluster placement group but you can choose any placement strategy you want. Now click on the **create group** button to create the cluster placement group.



In the case of a partition placement group, first give a name whatever you want. After that, in the placement strategy portion select the partition. Then, we need to provide the number of partitions for our placement group. After providing all the details, click on the **create group** button to create the placement group.



After creating the placement group, you can now launch EC2 instances in the placement group. You need to specify the placement group while launching the EC2 instance. While launching the EC2 instance, check the **Placement group** box in the EC2 instance configuration. Then you can select the placement group in which you want to launch the EC2 instance.
In case of the partition placement group, you can select the Target partition in which the EC2 instance will be launched.



Creating Placement Group Using AWS Command Line Interface (CLI)

In this section we will see how we can create a placement group on AWS using AWS command line interface. First you need to configure the AWS command line interface credentials. Visit the following article to learn how to configure the AWS command line interface credentials.

https://linuxhint.com/configure-aws-cli-credentials/

Now use the following command in the terminal to create the placement group.

```
$: aws ec2 create-placement-group \
 --group-name  \
 --strategy
```

```
[cloudshell-user@ip-10-0-159-226 ~]$ aws ec2 create-placement-group --group-name demo-cluster --strategy cluster --tag-specifi
cations 'ResourceType=placement-group,Tags={Key=purpose,Value=Learning}'
{
    "PlacementGroup": {
        "GroupName": "demo-cluster",
        "State": "available",
        "Strategy": "cluster",
        "GroupId": "pg-078ff7f7f81f4e541",
        "Tags": [
            {
                "Key": "purpose",
                "Value": "Learning"
            }
        ],
        "GroupArn": "arn:aws:ec2:ap-south-1:820429717320:placement-group/demo-cluster"
    }
}
[cloudshell-user@ip-10-0-159-226 ~]$
```

In case of a partition placement group, you need to also specify the number of partitions.

```
$: aws ec2 create-placement-group \
 --group-name  \
 --strategy  \
 --partition-count
```

```
[cloudshell-user@ip-10-0-159-226 ~]$ aws ec2 create-placement-group --group-name partition-group --strategy partition --partit
ion-count 7
{
    "PlacementGroup": {
        "GroupName": "partition-group",
        "State": "available",
        "Strategy": "partition",
        "PartitionCount": 7,
        "GroupId": "pg-064cd583a34998953",
        "GroupArn": "arn:aws:ec2:ap-south-1:820429717320:placement-group/partition-group"
    }
}
[cloudshell-user@ip-10-0-159-226 ~]$
```

You can view the details of the placement group created by using the following command in the terminal.

```
$: aws ec2 describe-placement-groups \
 --group-names
```

```
[cloudshell-user@ip-10-0-159-226 ~]$ aws ec2 describe-placement-groups --group-names demo-cluster
{
    "PlacementGroups": [
        {
            "GroupName": "demo-cluster",
            "State": "available",
            "Strategy": "cluster",
            "GroupId": "pg-078ff7f7f81f4e541",
            "Tags": [
                {
                    "Key": "purpose",
                    "Value": "Learning"
                }
            ],
            "GroupArn": "arn:aws:ec2:ap-south-1:820429717320:placement-group/demo-cluster"
        }
    ]
}
[cloudshell-user@ip-10-0-159-226 ~]$
```

Amazon EC2 Key Pairs

Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. Public–key cryptography uses a public key to encrypt a piece of data, such as a password, then the recipient uses the private key to decrypt the data. The public and private keys are known as a *key pair*.

To log in to your instance, you must create a key pair, specify the name of the key pair when you launch the instance, and provide the private key when you connect to the instance. On a Linux instance, the public key content is placed in an entry within ~/.ssh/authorized_keys. This is done at boot time and enables you to securely access your instance using the private key instead of a password.

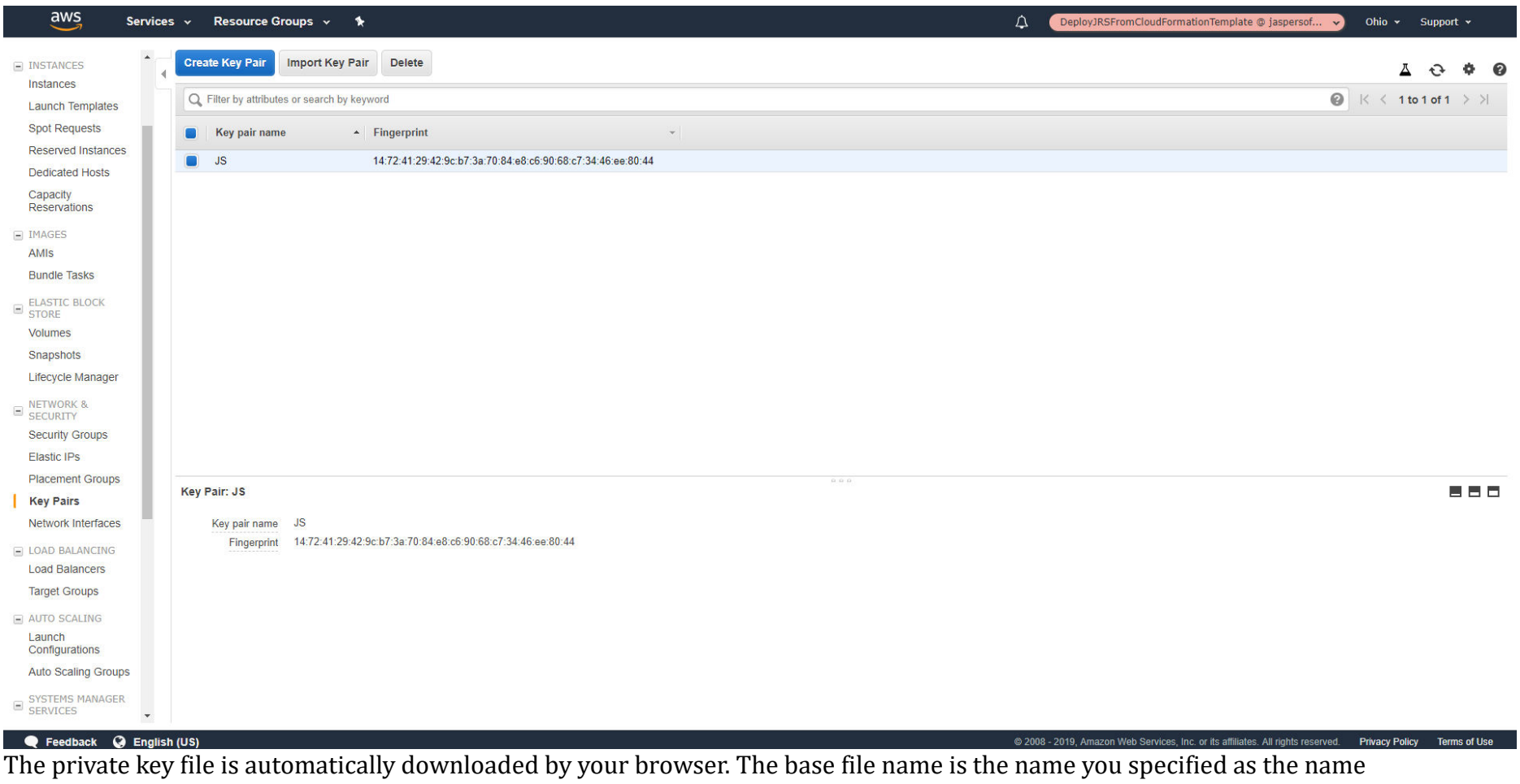
Creating a Key Pair Using Amazon EC2

You can create a key pair using the Amazon EC2 console or the command line. In this article we will review how to use Amazon EC2 console. After you create a key pair, you can specify it when you launch your instance. You can also add the key pair to a running instance to enable another user to connect to the instance. For more information, see **Adding or Replacing a Key Pair for Your Instance.**

To create your key pair using the Amazon EC2 console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **NETWORK & SECURITY**, choose **Key Pairs**.
   Note
   The navigation pane is on the left side of the Amazon EC2 console. If you do not see the pane, it might be minimized; choose the arrow to expand the pane.
3. Choose **Create Key Pair.**

4. For **Key pair name**, enter a name for the new key pair, and then choose **Create**.



5. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is *.pem*. Save the private key file in a safe place.

   Important
   This is the only chance for you to save the private key file. You'll need to provide the name of your key pair when you launch an instance and the corresponding private key each time you connect to the instance.

6. If you will use an SSH client on a Mac or Linux computer to connect to your Linux instance, use the following command to set the permissions of your private key file so that only you can read it.`chmod 400 my-key-pair.pem` If you do not set these permissions, then you cannot connect to your instance using this key pair. For more information, see Error: Unprotected Private Key File.

To access your instance

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (i.e test_key.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:`chmod 400 test_key.pem`

4. Connect to your instance using its Public DNS: i.e. ec2-18-224-39-106.us-east-2.compute.amazonaws.com
   Example:

   ```
   ssh -i "test_key.pem" ubuntu@ec2-18-224-39-106.us-east-2.compute.amazonaws.com
   ```

   Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username..

Elastic Block Store (EBS)

EBS Volume Types

EBS (Elastic Block Storage) volume is a block storage device connected to EBS backed instances. These block storage devices can be accessed on the instance just like a hard drive. Amazon provides 3 different categories of storage drives, and each category includes different types of EBS volumes. Following is the detailed list of EBS volumes provided by Amazon.

Add more details like throughput, iops, etc

- Solid State Drives (SSD)
    - General Purpose SSD
    - Provisioned IOPS SSD
- Hard Disk Drives (HDD)
    - Throughput Optimized HDD
    - Cold HDD
- Previous Generation
    - Magnetic

Solid State Drives (SSD)

Solid State Drives are useful for loads that include frequent read writes to the drive (High IOPS) with lower IOPS size (lower throughput). The dominant performance attribute for SSDs is IOPS. Now we will discuss different categories of EBS volumes under SSD-backed volumes.

General purpose SSD

The General-purpose SSD drives provide a balance between the performance and the case. These EBS volumes are recommended for most of the use cases by Amazon. There are two types of General-purpose SSD volumes that are **gp2** and **gp3**.
The General-purpose SSD volume (**gp3**) provides the consistent 125 MiB/s throughput and 3000 IOPS within the price of provisioned storage. Additional IOPS (up to 16,000) and throughput (1000 MiB/s) can be provisioned with an additional price. The General-purpose SSD volume (**gp2**) provides 3 IOPS per GiB storage provisioned with a minimum of 100 IOPS. Similarly, the volumes provisioned with 5333 GiB or larger (up to 16 TiB) will have a maximum of 16,000 IOPS

Provisioned IOPS SSD

Provisioned IOPS SSD volumes are used for workloads that require high performance, lower latency, and higher throughput. Three types of Provisioned IOPS SSD volumes are **io2 Block Express**, **io2,** and **io1**. The **io2 Block Express** volume type is only supported with **R5b** instances.

The Provisioned IOPS SSD (**io1**) volume can be provisioned with a minimum of 100 and a maximum of 64,000 IOPS. The maximum ratio between provisioned IOPS and storage is 50:1. So you can provision a maximum of 50 IOPS per 1 GiB storage. The Provisioned IOPS SSD (**io2**) volume can be provisioned with a minimum of 100 and 256,000 IOPS, and the maximum ratio between the provisioned IOPS and storage is 1000:1. So you can provision a maximum of 1000 IOPS for 1 GiB storage.

Hard Disk Drives (HDD)

Hard disk drives are useful for large streaming workloads that require higher throughput. The dominant performance attribute for SSDs is throughput. Now we will discuss different categories of EBS volumes under HDD-backed volumes.

Throughput optimized HDD

The throughput optimized HDD volume type is designed for frequently accessed workloads that require higher throughput. The **st1** is the only EBS volume type of throughput optimized HDD.
The Throughput optimized HDD (**st1**) volume provides a baseline throughput of 40 MiB/s per TiB storage provisioned up to 12,800 GiB. From 12,800 GiB to onward, a consistent throughput of 500 MiB/s is provisioned with storage.

Cold HDD

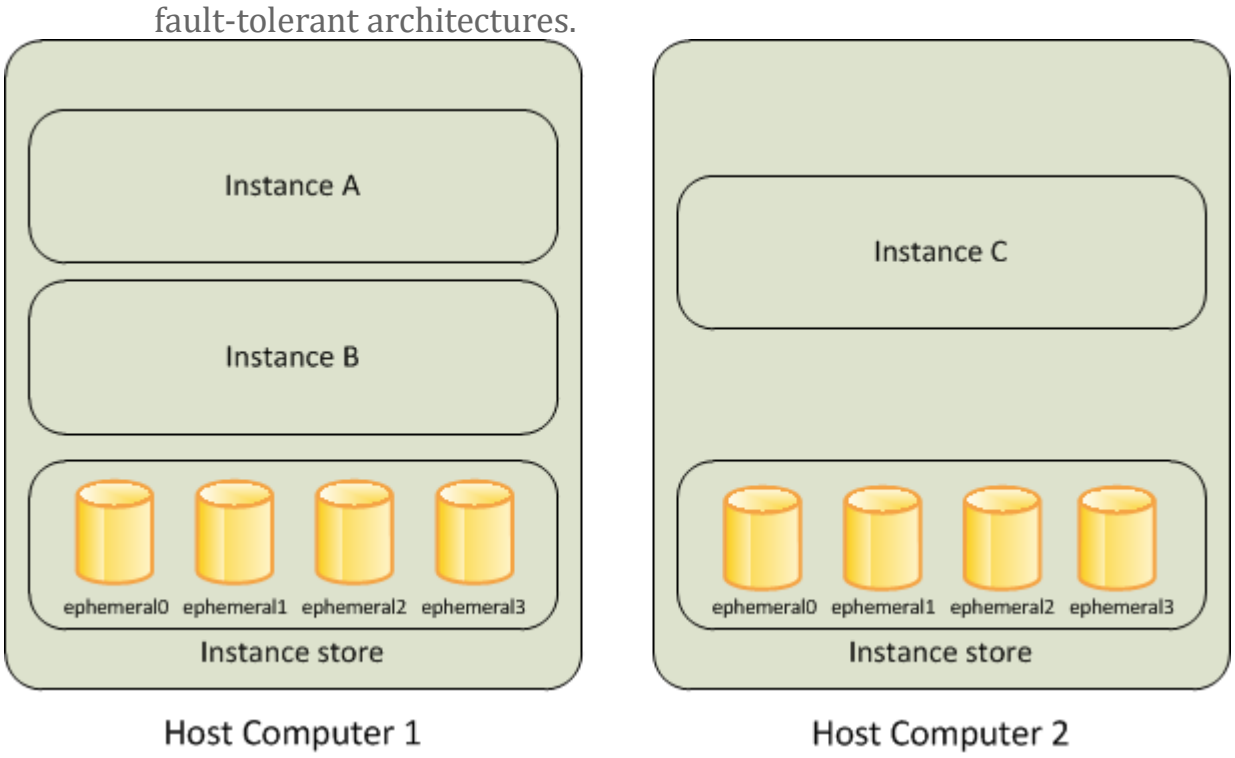
Cold HDD is another type of Hard disk drive provided by AWS for less frequently accessed workloads but higher throughput. Cold HDD is different from Throughput optimized based on IOPS. It has lesser IOPS than Throughput optimized HDD. Like Throughput Optimized HDD, the Cold HDD also provides a baseline throughput of 12 MiB/s per TiB storage provisioned. Following is the performance chart of Cold HDD EBS volume.

Magnetic

Magnetic is the only EBS volume type under the Previous Generation category and is used for workloads where data is infrequently accessed, and performance is not a primary factor.

Instance Store Volumes

- An instance store provides temporary or Ephemeral block-level storage for an Elastic Cloud Compute – EC2 instance.
- is located on the disks that are physically attached to the host computer.
- consists of one or more instance store volumes exposed as block devices.
- The size of an instance store varies by instance type.
- Virtual devices for instance store volumes that are ephemeral[0-23], starting the first one as ephemeral0 and so on.
- While an instance store is dedicated to a particular instance, the disk subsystem is shared among instances on a host computer.
- is ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers.
- delivers very high random I/O performance and is a good option for storage with very low latency requirements, but you don't need the data to persist when the instance terminates or you can take advantage of fault-tolerant architectures.



Optimizing Disk Performance

Choose the Right Volume Types and Sizes

The pricing and the performance capabilities of an EBS volume vary according to volume types. Here are the most commonly used volume types:

- **SSD volumes** — used for high-performance workloads that need high IOPS and low latency. SSD volumes burst up to 3,000 IOPS per volume, they are independent of volume size, and meet the performance needs of most apps.
- **Hard Disk Drives (HDD) volumes** — used for frequently accessed data, throughput-intensive workloads with large I/O sizes and large datasets, such as Kafka, MapReduce, log processing, ETL workloads, and data warehouse.
- **Cold HDD** — used for volumes that do not require frequent access, such as EBS snapshots.

You should choose the type of the volume based on the workload priority. You can use Provisioned IOPS SSD for mission-critical applications such as large database workloads. For low priority workloads you can use the cost efficient General Purpose SSD volumes.

Regardless of priority or workload, provisioned volumes are not a good choice if you are performing many small operations. Provisioned volumes count each operation under 256K as an individual IOPS use, as opposed to general volumes that combine them into chunks of 256K.

Learn from Your EBS Metrics

Monitoring your metrics can assist you in optimizing your EBS environment by giving insights into storage activity and performance. There are three metrics categories you should monitor:

- **Disk I/O** — can tell you if you need to modify your volume size, change their type, or if you need a load balancer.
- **Disk activity** — can inform you if the volume type you are using and the limits you have chosen are correct.
- **Latency** — can inform you if your volumes need higher limits of Input/Output Operations (IOPS).

Monitoring your cloud infrastructure ensures the availability, performance and health of your application. You can pinpoint issues and negative trends before they cause damage. AWS also gathers information on volume status and can send alerts on issues like disabled volumes. Checking volume statuses and responding to alerts can ensure the availability of data and help you avoid payments for unusable storage.

Leverage Burst Credits

Burst credits are used to improve performance during periods of high activity. You automatically get some burst credits when you create a volume. Each new volume has enough credits to support an increase of 3000 IOPS for 30 minutes. After that, your performance can drop back and you will not get new credits, unless you decrease your volume size.

Remember to check your burst credits if you have unresponsive services, or instances that are suddenly crawling. If the lack of credits is what's reducing your productivity, you can attach a mirror volume. Mirror volumes can help distribute IOPS and give you an additional burst credit pool. Another option is to simply increase your volume size to 1TiB or larger.Large size volumes are not limited by burst credits and can reach full performance.

Manage Your Volumes

EBS volumes can quickly get out of control if not properly managed, either because they are unintentionally deleted or not fully utilized. EC2 instances have a built-in `DeleteOnTermination` attribute. This attribute enables you to determine what happens to the attached EBS volumes when an instance is deleted or terminated. You can set it up during the creation of Amazon Machine Images (AMI) or during launch.

Automatically managing the termination of volumes enables you to manage backups of remaining volumes, prevent payment for unused volumes or securely delete persistent data.

Creating and Deleting Volumes

To create an empty EBS volume using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Volumes**.
3. Choose **Create volume**.
4. For **Volume type**, choose the type of volume to create. For more information, see Amazon EBS volume types.
5. For **Size**, enter the size of the volume, in GiB. For more information, see Constraints on the size and configuration of an EBS volume.
6. (`io1`, `io2`, and `gp3` only) For **IOPS**, enter the maximum number of input/output operations per second (IOPS) that the volume should provide.
7. (`gp3` only) For **Throughput**, enter the throughput that the volume should provide, in MiB/s.

8. For **Availability Zone**, choose the Availability Zone in which to create the volume. A volume can be attached only to an instance that is in the same Availability Zone.
9. For **Snapshot ID**, keep the default value (**Don't create volume from a snapshot**).
10. (`io1` and `io2` only) To enable the volume for Amazon EBS Multi-Attach, select **Enable Multi-Attach**. For more information, see Attach a volume to multiple instances with Amazon EBS Multi-Attach.
11. Set the encryption status for the volume.

    If your account is enabled for encryption by default, then encryption is automatically enabled and you can't disable it. You can choose the KMS key to use to encrypt the volume.

    If your account is not enabled for encryption by default, encryption is optional. To encrypt the volume, for **Encryption**, choose **Encrypt this volume** and then select the KMS key to use to encrypt the volume.

    Note

    Encrypted volumes can be attached only to instances that support Amazon EBS encryption. For more information, see Amazon EBS encryption.
12. (Optional) To assign custom tags to the volume, in the **Tags** section, choose **Add tag**, and then enter a tag key and value pair. For more information, see Tag your Amazon EC2 resources.
13. Choose **Create volume**.

    Note

    The volume is ready for use when the **Volume state** is **available**.
14. To use the volume, attach it to an instance. For more information, see Attach an Amazon EBS volume to an instance.

To delete an EBS volume using the console

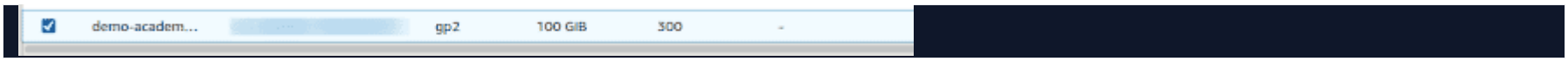
1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Volumes**.
3. Select the volume to delete and choose **Actions**, **Delete volume**.

   Note

   If **Delete volume** is greyed out, the volume is attached to an instance. You must detach the volume from the instance before it can be deleted.
4. In the confirmation dialog box, choose **Delete**.

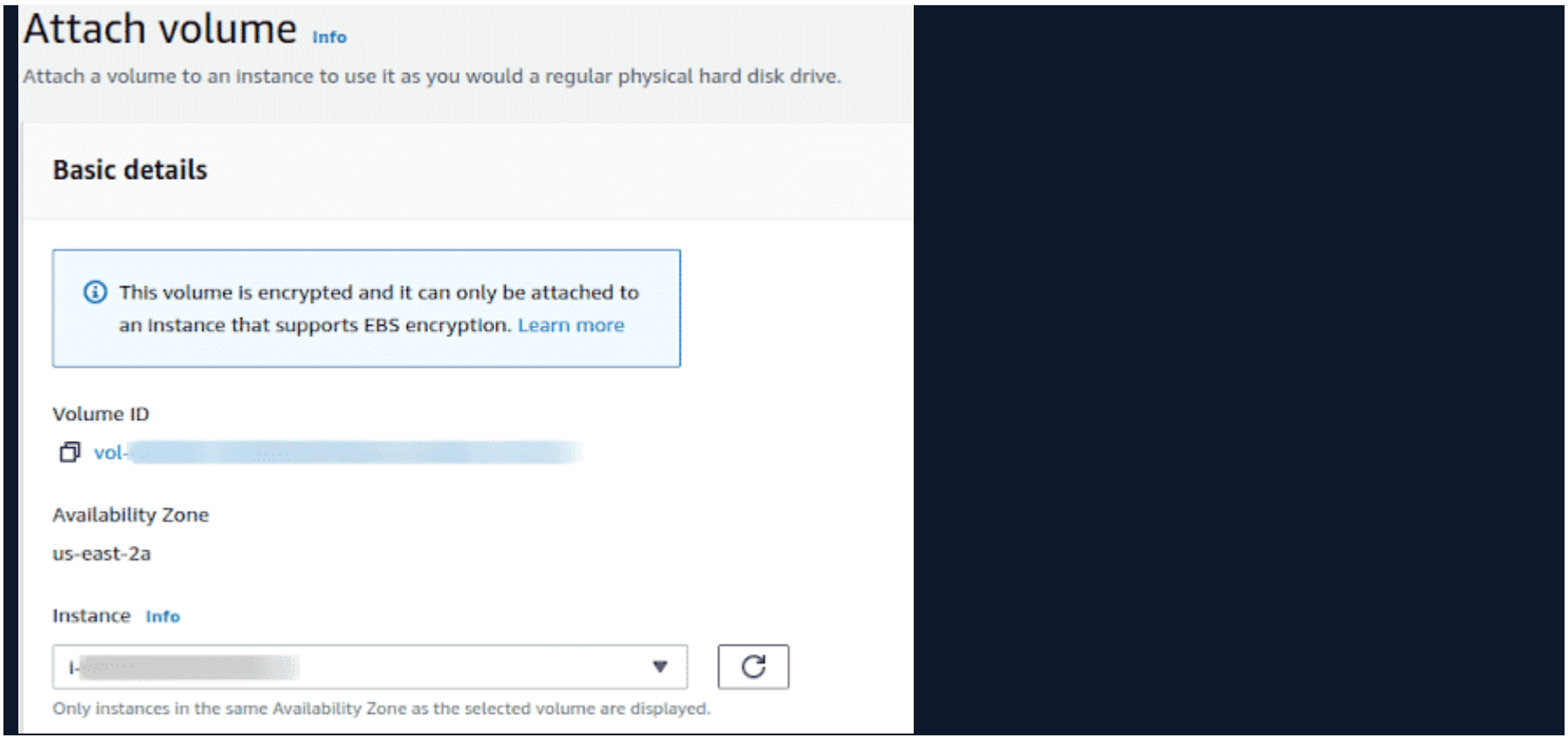
Attach and Detach Volumes

1. Once the EBS volume is created select the EBS volume you want to attach to the EC2 Instance. The EBS volume should be in **Available** state.

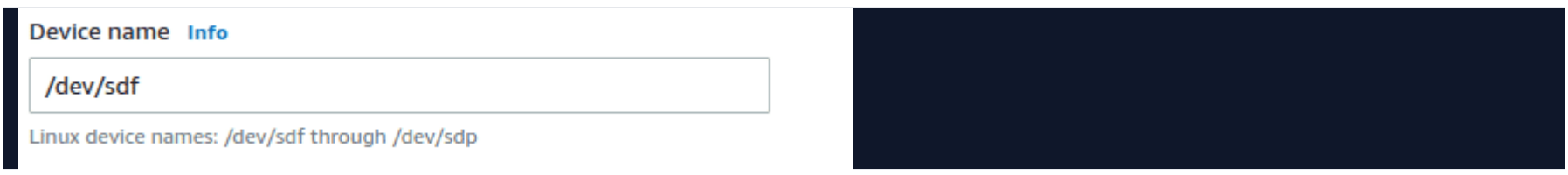


2. Click on the dropdown of **Actions** and select the option **Attach Volume**

3. Once the **Attach Volume** window opens select the EC2 instance you want to attach the volume



4. Select the Device Name which is the same device name used by the EC2 Instance selected in step 3 above

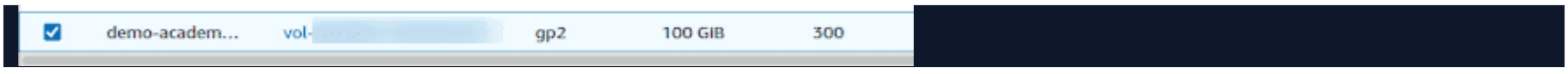


5. Select the **Attach Volume** option and the volume will be successfully attached. Once the volume is attached to an EC2 instance, the Volume Instance will be **In-use**.

Detach an EBS Volume

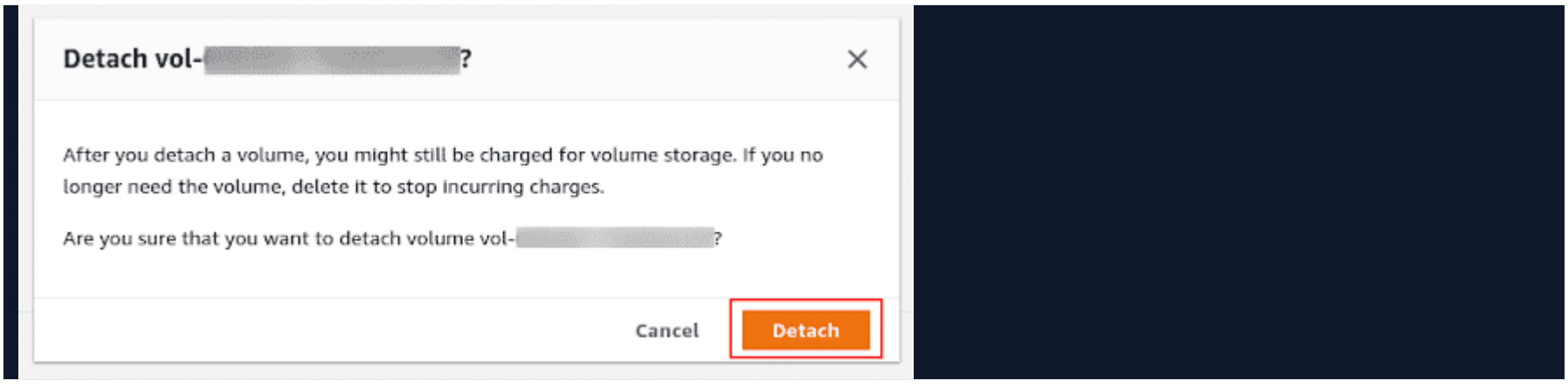
1. In the Amazon EC2 console, choose Volumes in the navigation pane on the right side

2. Once the Volumes page opens, select the volume you want to detach. An attached volume shows **in-use** state



3. Click on the **Actions** drop-down and select the **Detach volume** option.

4. A pop-up will appear for confirmation, select the **Detach** button to detach the volume from the EC2 instance.



5. Once you have successfully detached the volume you can again see that the volume is in **Available** state.

Mount and Un-mounting Volumes:

Step 1 – Create a Volume

Step 2 – Attach a Volume to EC2 Instance

Step 3 – Verify if Volume is attached or not

Verify if Volume is attached or not by running linux command in Ec2-instance $ lsblk

Step 4 – Check if the volume has any data using the following command.

If the above command output shows “/dev/xvdf: data“, it means your volume is empty. $ sudo file -s /dev/xvdf

Step 5: Format the volume to the ext4 filesystem using the following command.

Alternatively, you can also use the xfs format. You have to use either ext4 or xfs.

$ sudo mkfs -t ext4 /dev/xvdf

$ sudo mkfs -t xfs /dev/xvdf

Step 6: Create a directory of your choice to mount our new ext4 volume. I am using the name “newvolume“. You can name it something meaningful to you.$ sudo mkdir /newvolume

Step 7: Mount the volume to “newvolume” directory using the following command.$ sudo mount /dev/xvdf /newvolume/

Step 8: cd into newvolume directory and check the disk space to validate the volume mount.

$ cd /newvolume

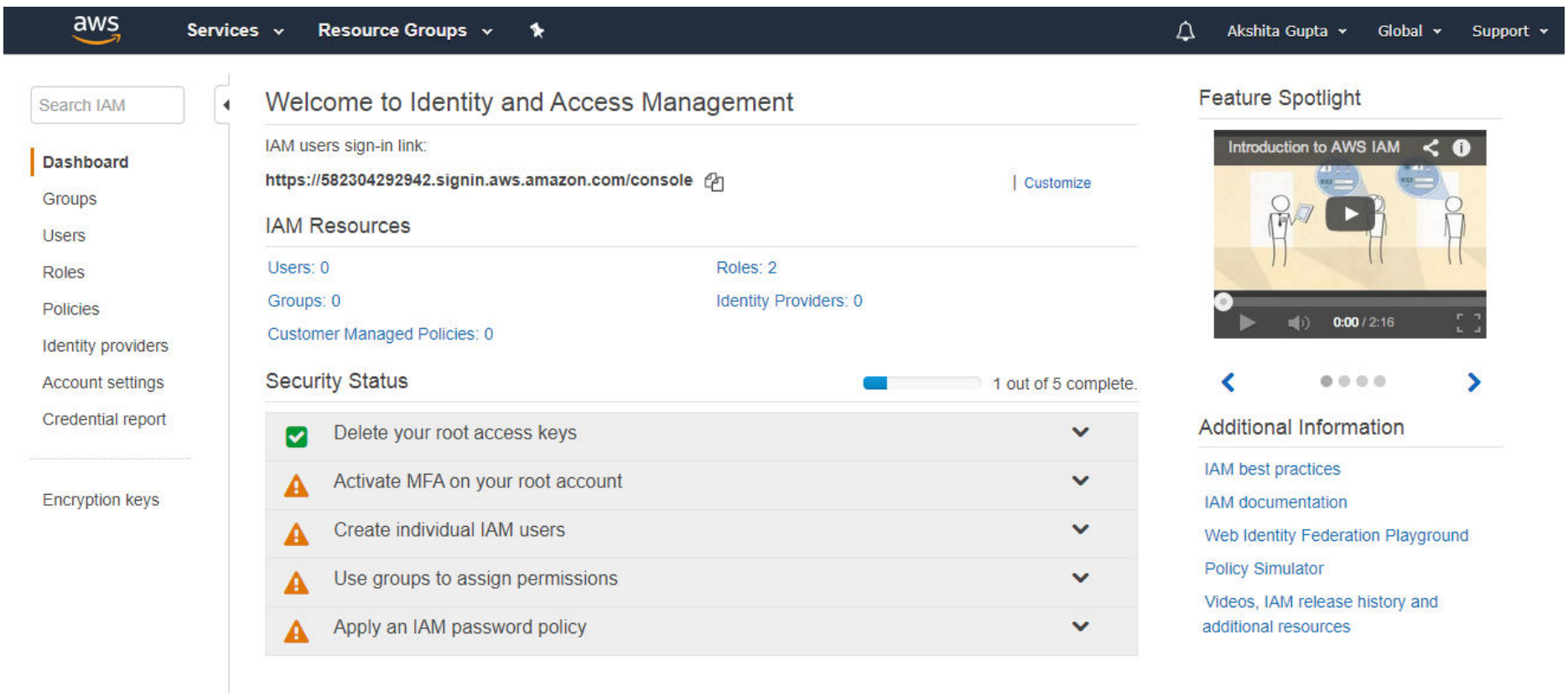
$ df -h .

5. Identity and Access Management (IAM)

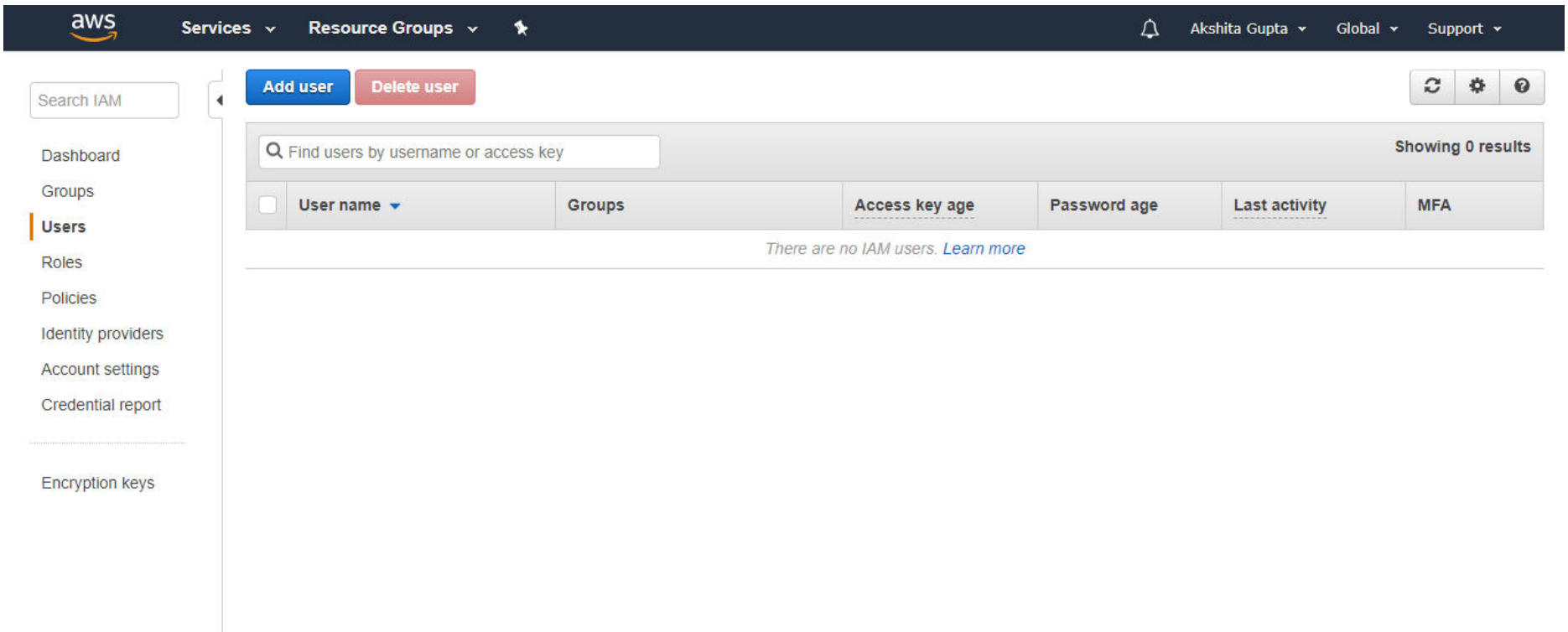
Creation of Users Accounts :

To create a User using AWS Management Console:

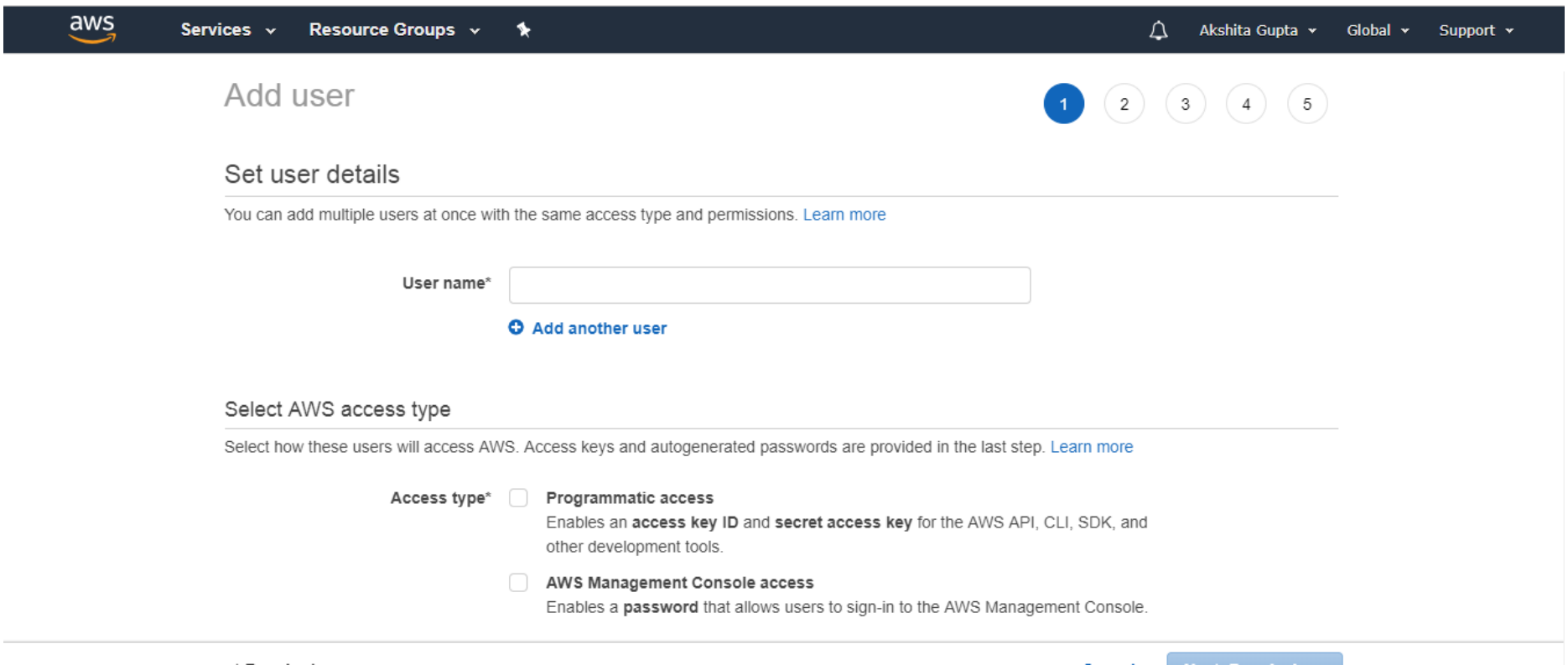
- Sign in to the AWS Management Console.
- Open the IAM Console at https://console.aws.amazon.com/iam/home?region=us-east-2#/home. The screen appears which is shown below:



- On the navigation pane, click on the Users. After clicking on the Users, the screen appears which is shown below:



- Click on the Add User to add new users to your account. After clicking on the Add User, the screen appears which is shown below:



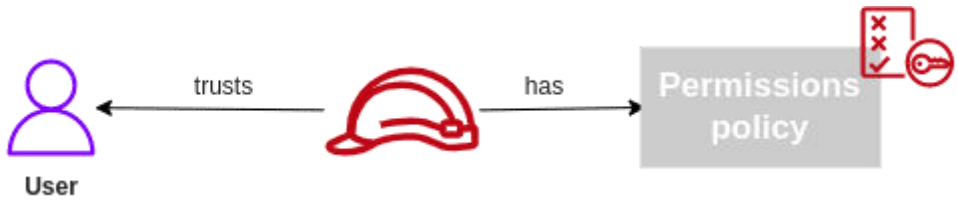
- Enter the User name for the user you want to create. You can create five users at a time.
- Select the AWS access type. Either you want a user to have programmatic access, AWS Management Console access or both.
- You can also give permission to the user to manage his or her security credentials.

Roles in IAM

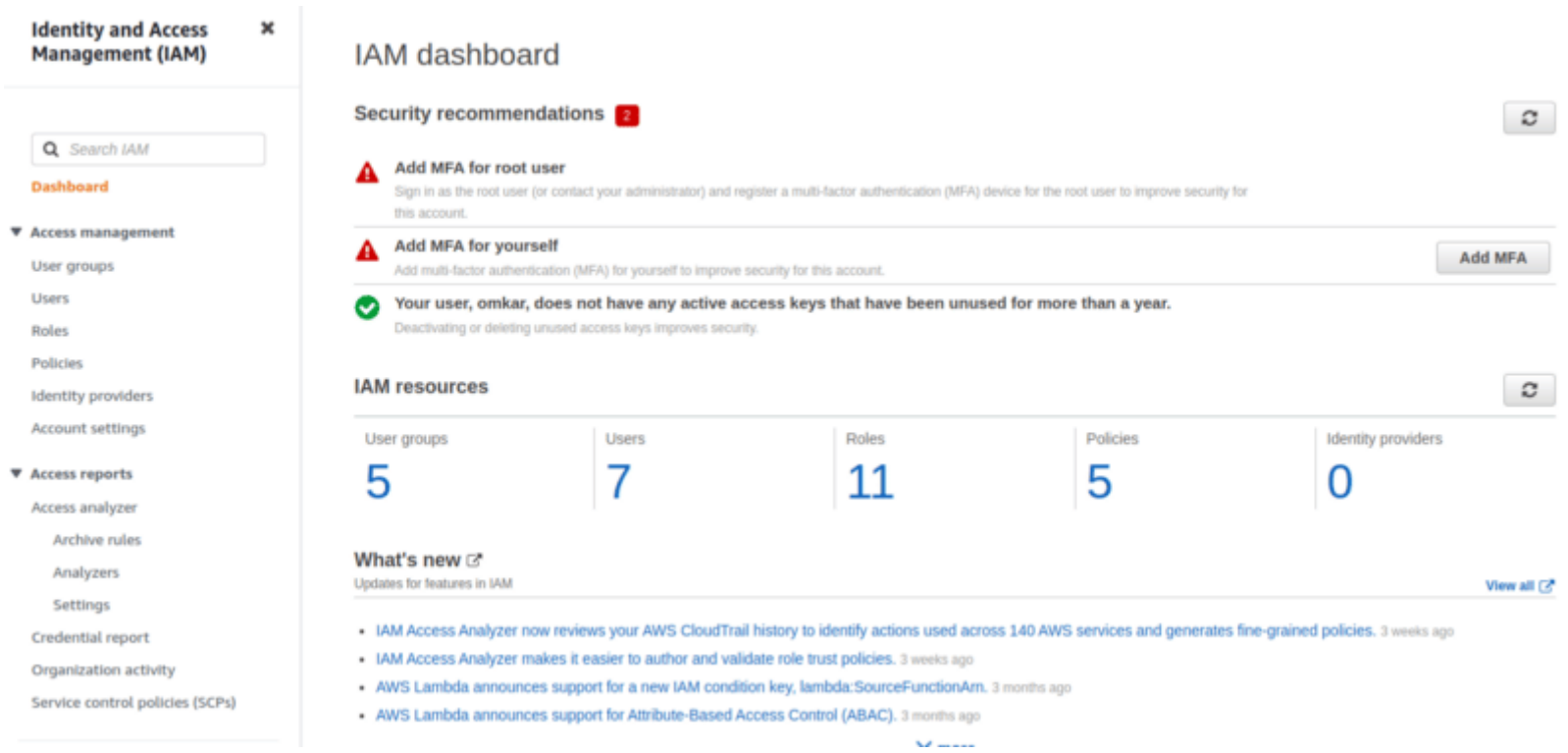
Creating our first role

Pre-requisite: An AWS account with an IAM admin user.

In this tutorial, we will create a custom IAM role with the trusted entity as the AWS user and the permission policy to allow admin access to AWS RDS.

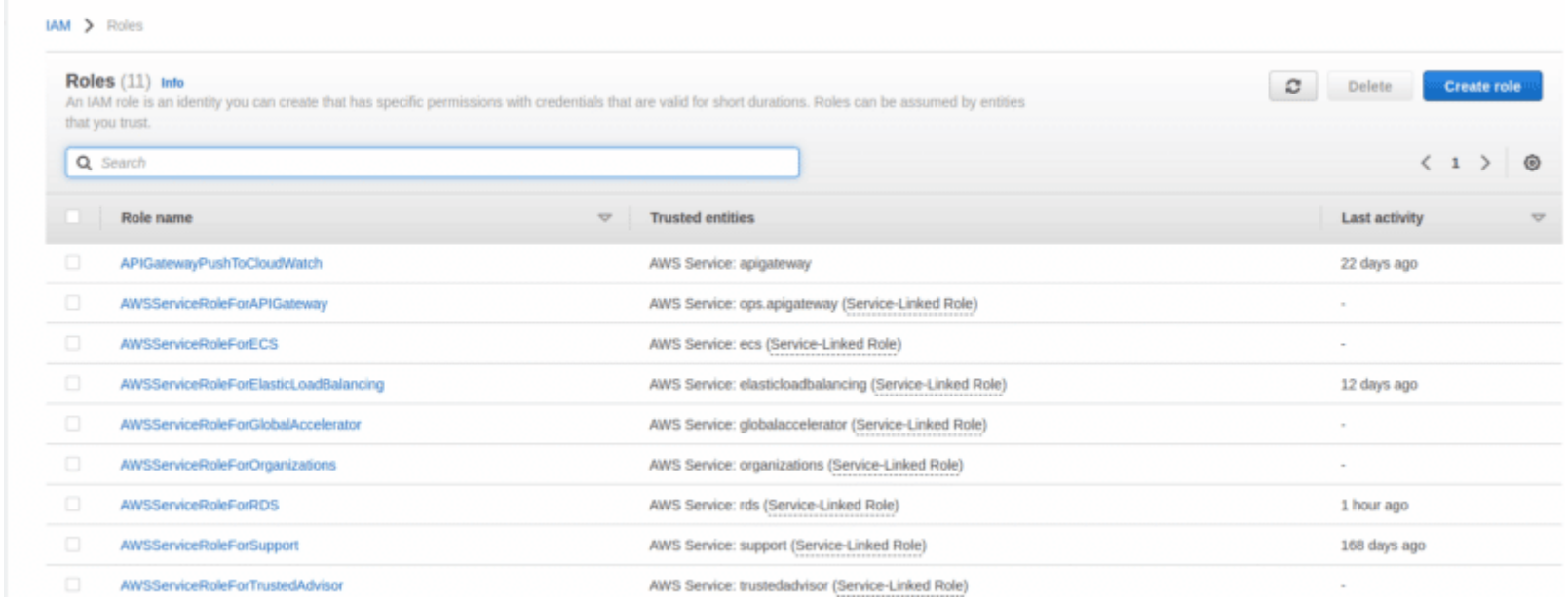


1. Log in to the AWS account and open the IAM service.

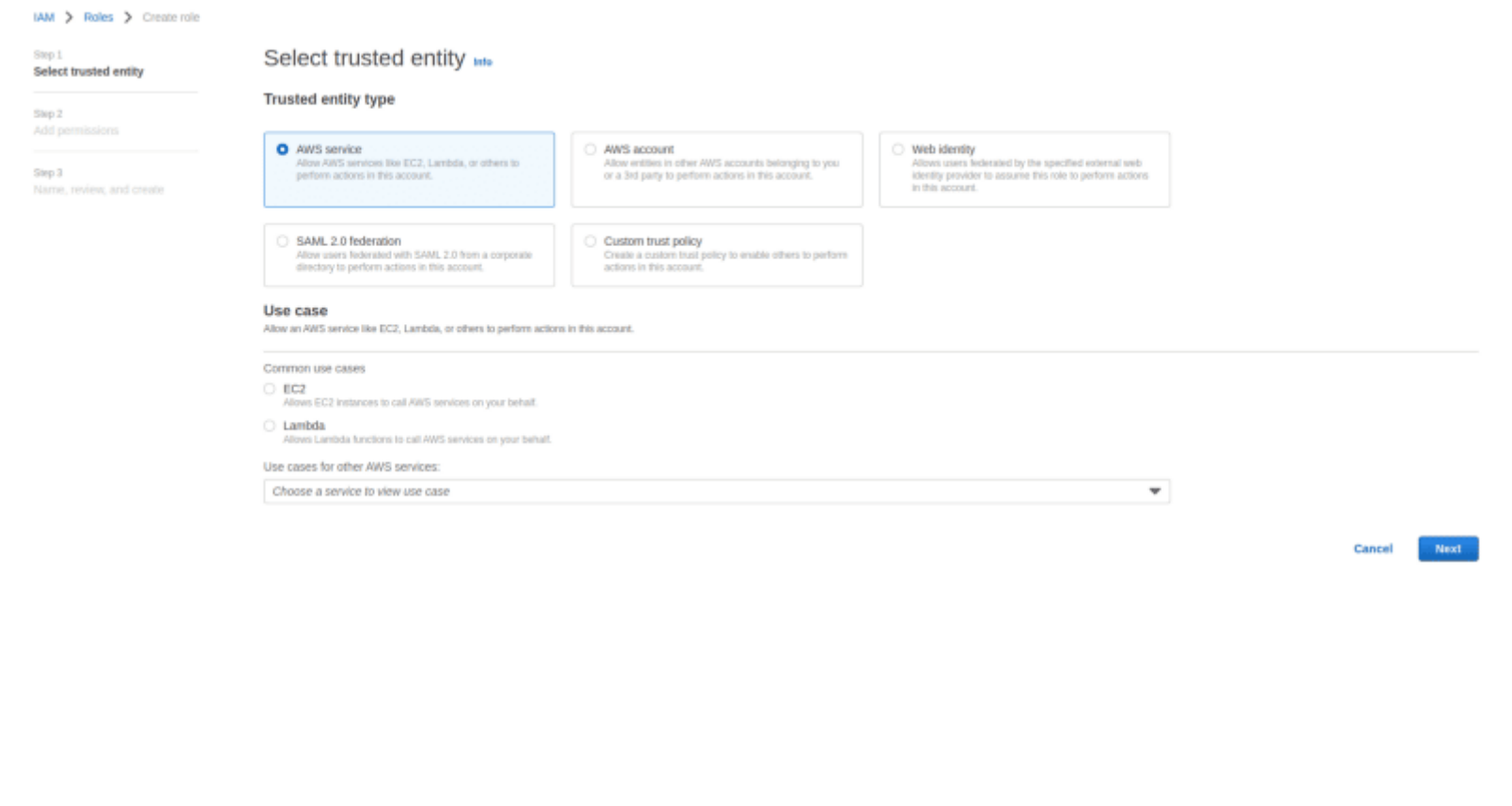


2. Open the roles panel either from the dashboard or from the side nav on the left.

3. You might notice some pre-created roles in your account on the roles panel. Ignore them for the time being; we will go over them in greater detail later in the article.

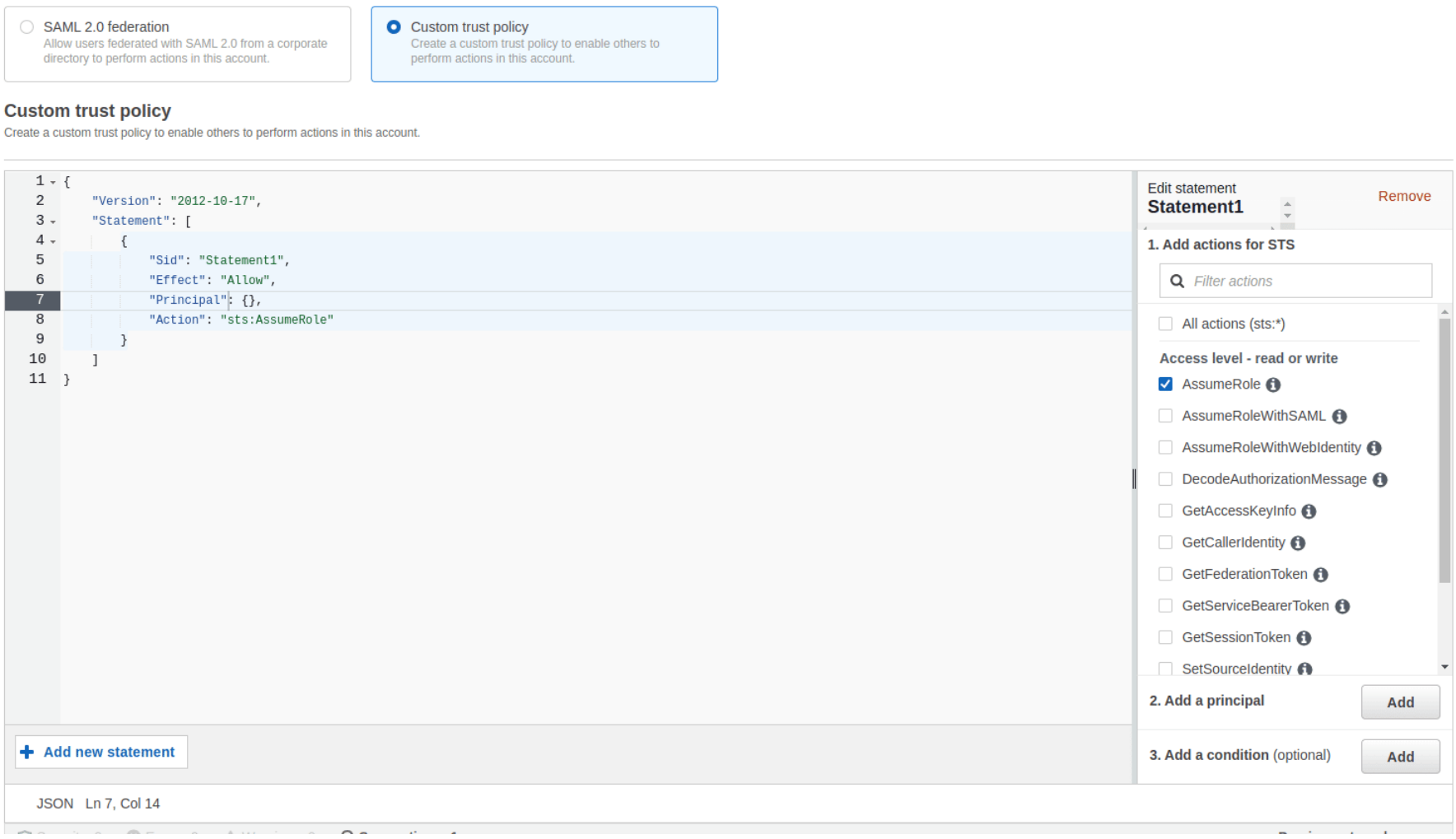


4. To create a new role, click on the “Create role” button.



5. As previously stated, a role has 2 core aspects, ***the trust policy*** and ***the permission policy***. So the first thing that we have to specify is who can assume this role. We will begin by selecting the "Custom trust policy" and then proceed to the other options in the later section of the article.

6. Upon selection of the "Custom trust policy", AWS automatically generates a JSON policy with an empty principal field. The principal field specifies who can assume this role. If we keep it empty then this policy cannot be assumed by any principal.



7. We will add the Amazon Resource Name (ARN) of the AWS IAM user who should be allowed to assume this role. The ARN can be obtained from the user details page in the IAM dashboard.

8. Copy the ARN and paste it as a key-value pair, with the key being "AWS" as shown below.

```
{
"Version": "2012-10-17",
"Statement": [
 {
  "Sid": "Statement1",
  "Effect": "Allow",
  "Principal": {
     "AWS": "arn:aws:iam::xxxxxxxxxxx:user/omkar"
  },
  "Action": "sts:AssumeRole"
 }
]
}
```

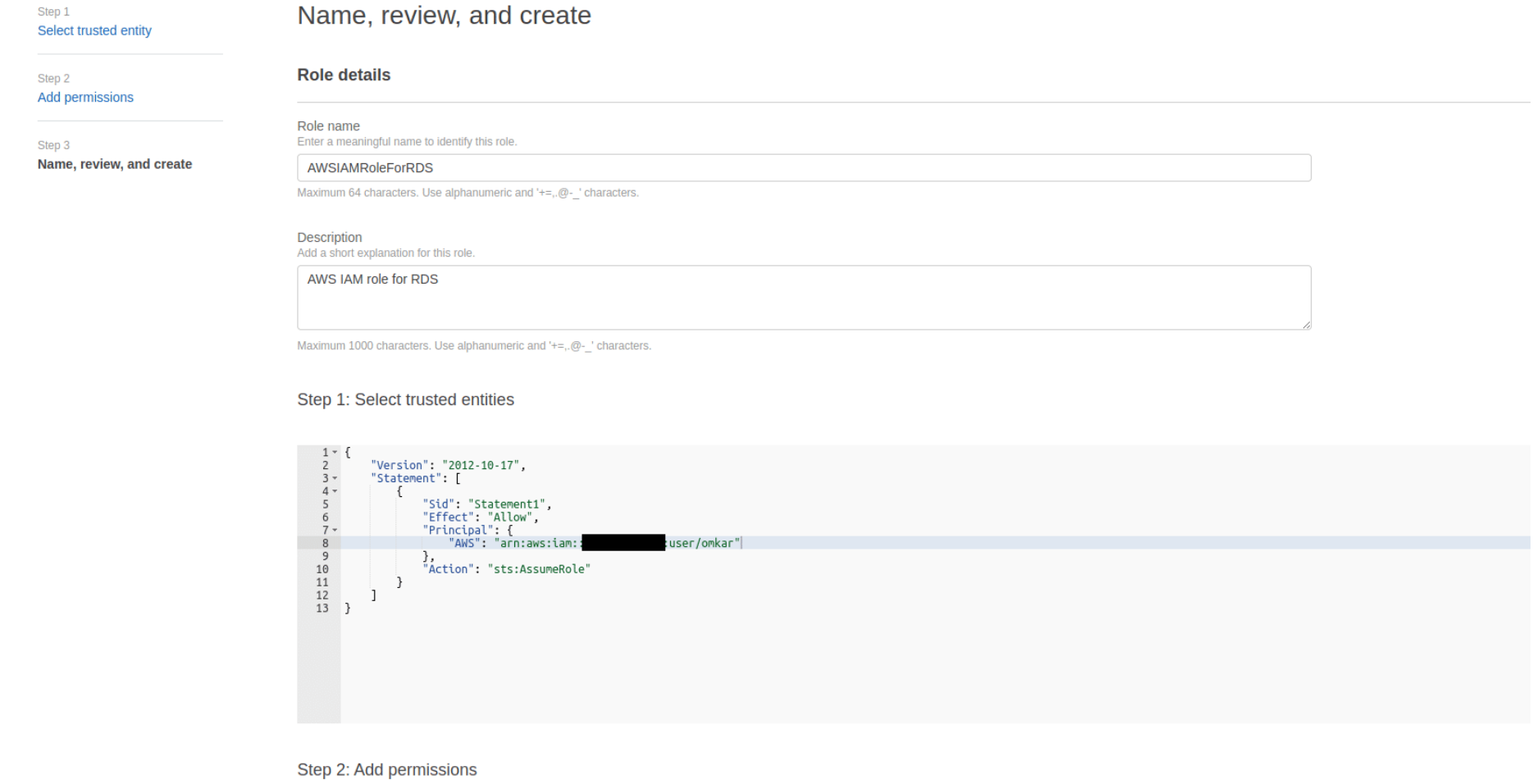
9. Once you have reviewed the trust policy, click on the next button to move on to the next page.

10. The next step, as you might expect, is to choose a *permission policy* for this role. We can either use an existing policy or create a new one. We will choose an existing managed policy by the name *AmazonRDSFullAccess* to grant our role full access to the AWS RDS service.

Remember that AWS denies everything by default. We are only granting the RDS access to this role by attaching this policy.

11. Leave all the other settings unchanged and click on next.

12. We have already taken care of the 2 essential aspects of a role. All that is left is to give the role a name and a description, and we are done.

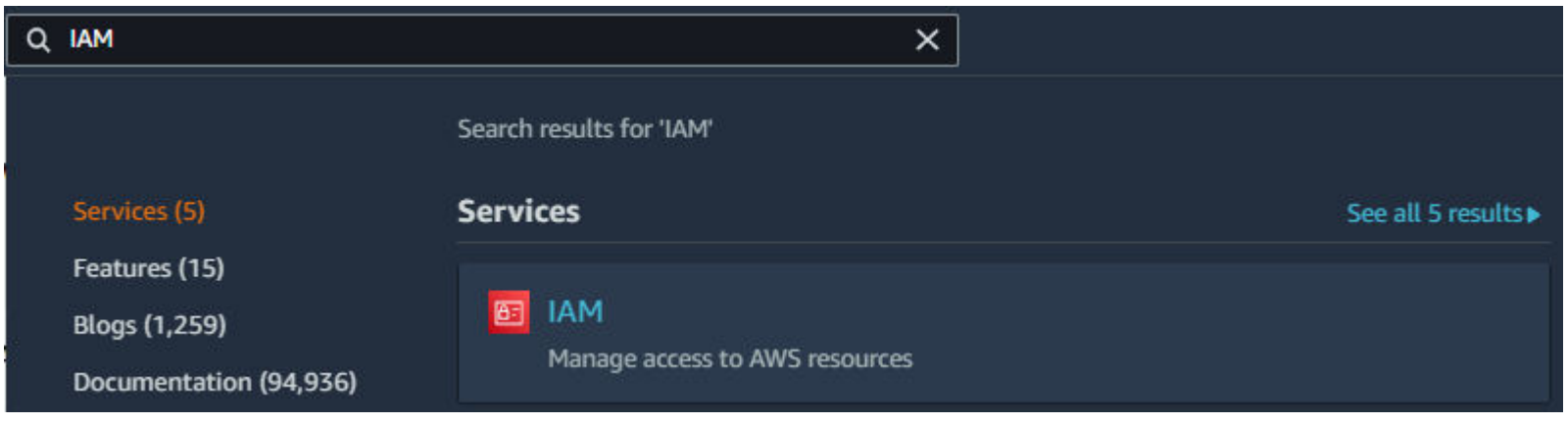


13. Name the role as `AWSIAMRoleForRDS` and provide the description as “AWS IAM role for RDS”

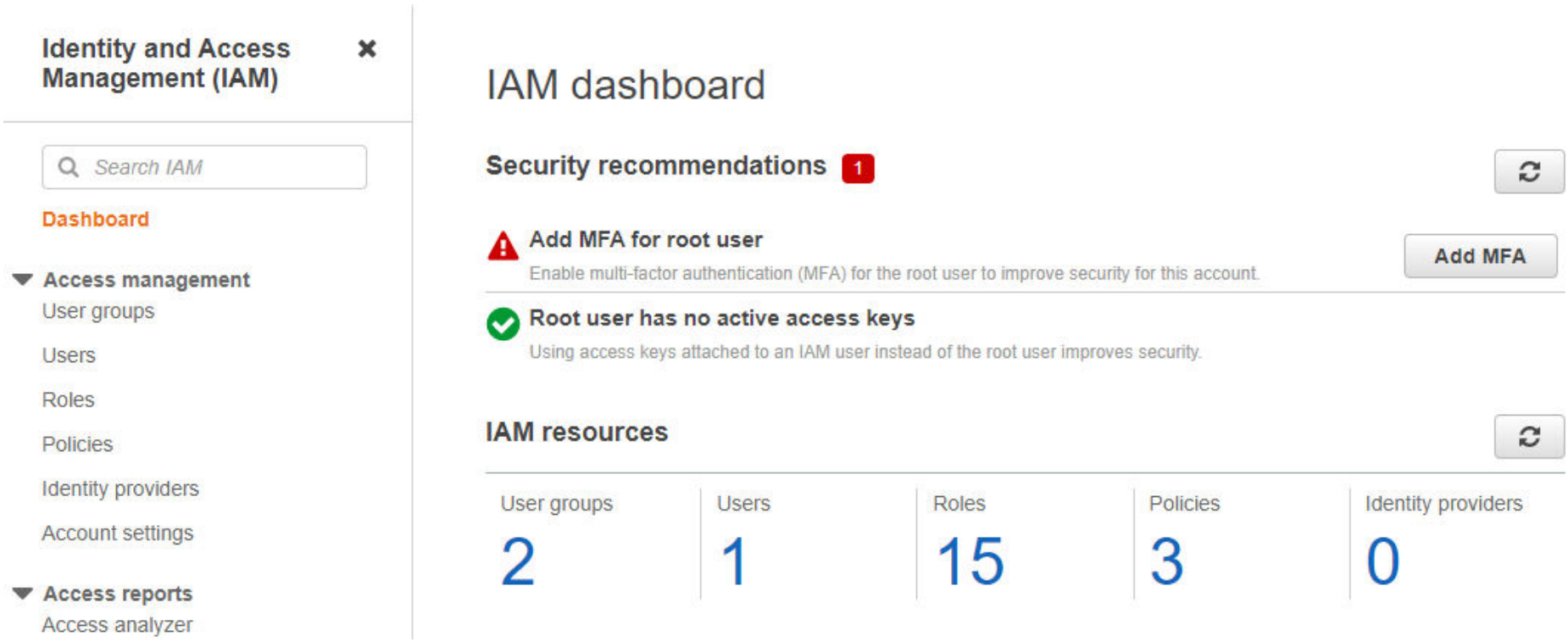
14. Review all the details and click on the **Create role** button.

Groups in IAM

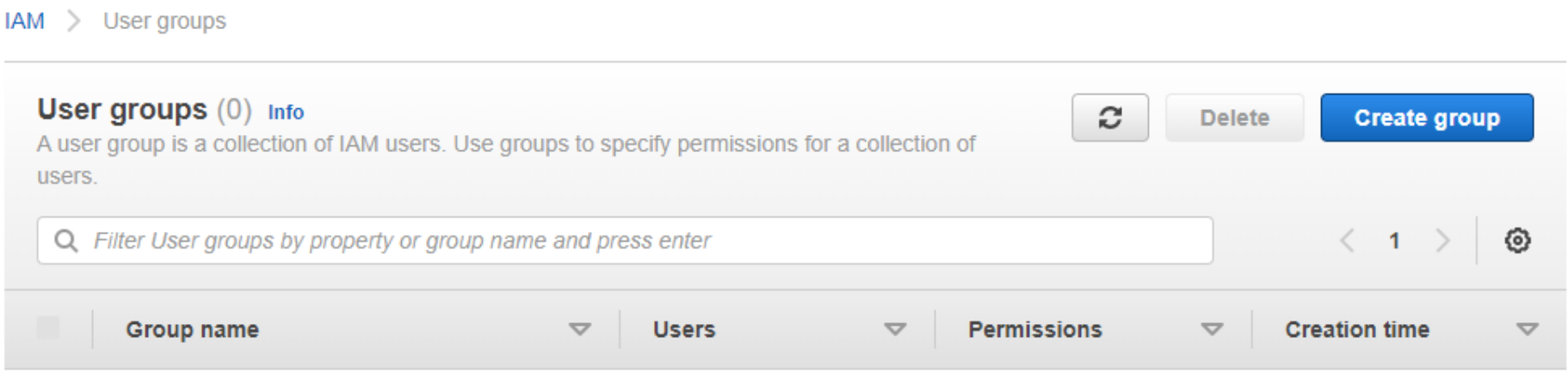
To create a user group just login to your AWS account and in the top search bar type IAM.



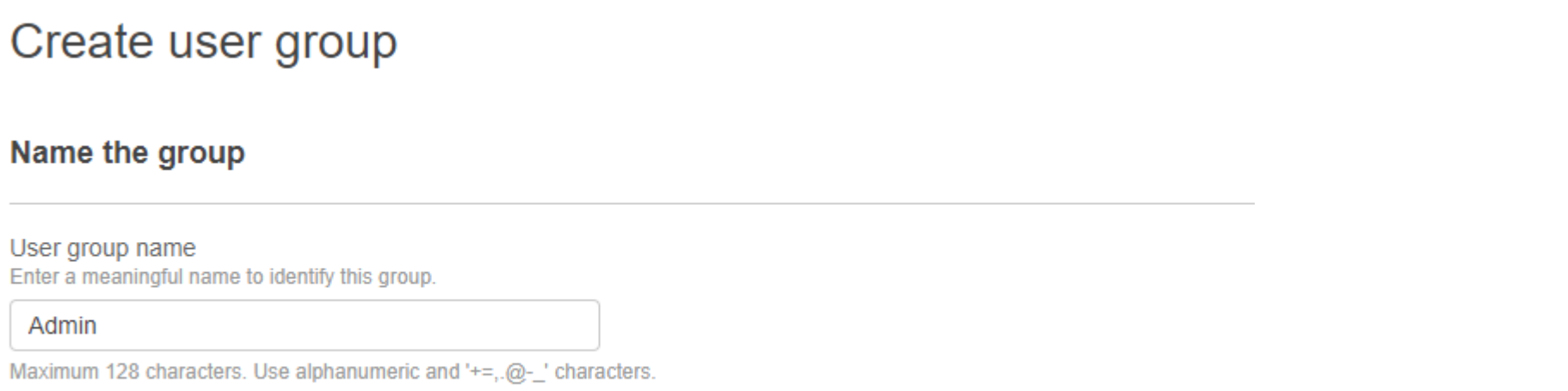
Select the IAM option down the search menu, this will take you to your IAM dashboard.



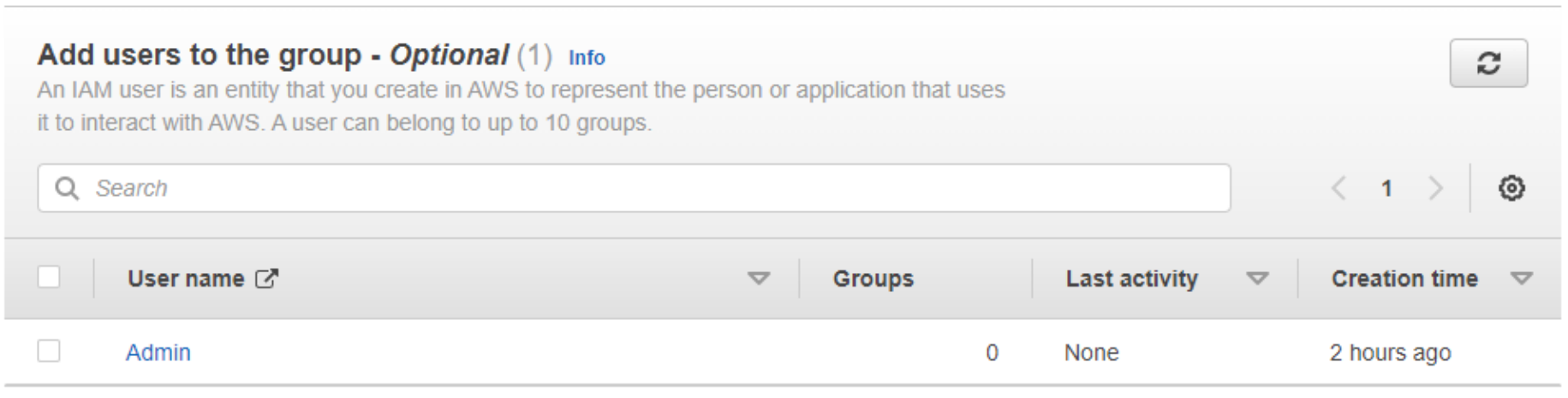
From the left side panel, select the **User Groups** tab. This will take you to your user group management window. Click on **Create Group** and following are the steps to create a user group.



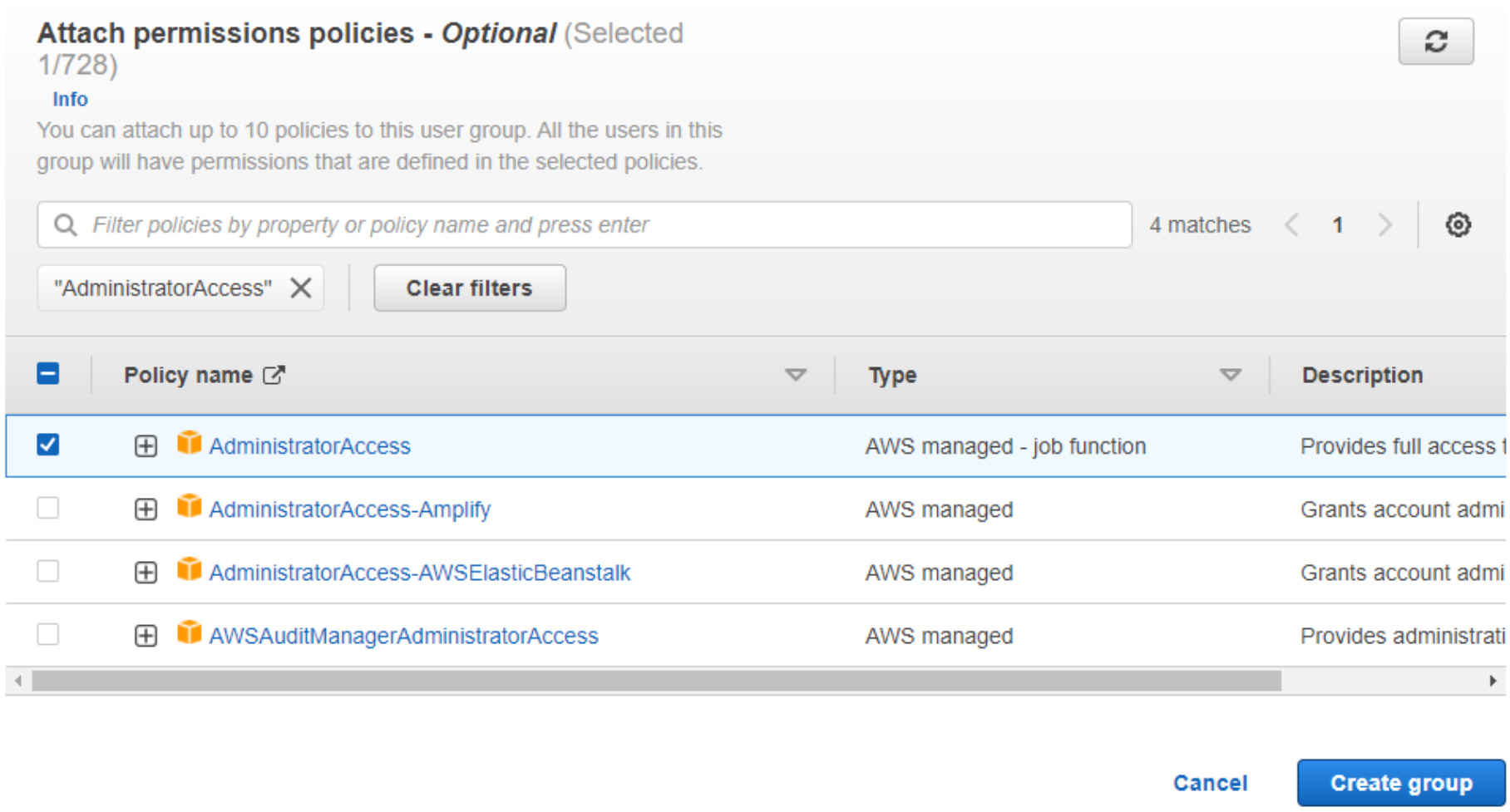
Type the name of the user group.



From the list below, you can select the existing users you want to add to this group. This step is not mandatory as you can also add users in the group later on.



The last and most important step in creating a user group is to attach policies which grant permissions to that group. From the policies list, select those you want to attach to the group and finally just click on create group in the bot<>tom right corner.



Creating Permissions for Users

1. Navigate to IAM using the **Services** menu or the unified search bar.
2. Below **IAM Resources**, select **Users** to view the existing users. You should see **cloud_user** as well as 3 **developer** users.
3. From the left dashboard menu, click **Policies** to create a new policy with developer access.
4. Click **Create Policy**.
5. Select the **Visual editor** tab.
6. Fill in the policy permissions for DynamoDB:
    - **Service**: Select **Choose a service**, then search for and select **DynamoDB**.
    - **Actions**: Below **Manual actions**, select **All DynamoDB actions (dynamodb:*)**.
    - **Resources**: Select **All resources**.
7. Click **Add additional permissions**.
8. Fill in the policy permissions for Lambda:
    - **Service**: Select **Choose a service**, then search for and select **Lambda**.
    - **Actions**: Below **Manual actions**, select **All Lambda actions (lambda:*)**.
    - **Resources**: Select **All resources**.
9. Click **Add additional permissions**.
10. Fill in the policy permissions for S3:
    - **Service**: Select **Choose a service**, then search for and select **S3**.
    - **Actions**: Below **Manual actions**, select **All S3 actions (s3:*)**.
    - **Resources**: Select **All resources**.
11. Click **Add additional permissions**.
12. Fill in the policy permissions for API Gateway:
    - **Service**: Select **Choose a service**, then search for and select **API Gateway**.
    - **Actions**: Below **Manual actions**, select **All API Gateway actions (apigateway:*)**.
    - **Resources**: Select **All resources**.
13. Click **Next: Tags**. You won't need to add any tags for this lab.
14. Click **Next: Review**.
15. In the **Name** field, enter *onlinebookstore-dev-developergroup-fullaccess-iam-policy*.
16. Click **Create policy**.

Deleting Permissions for Users

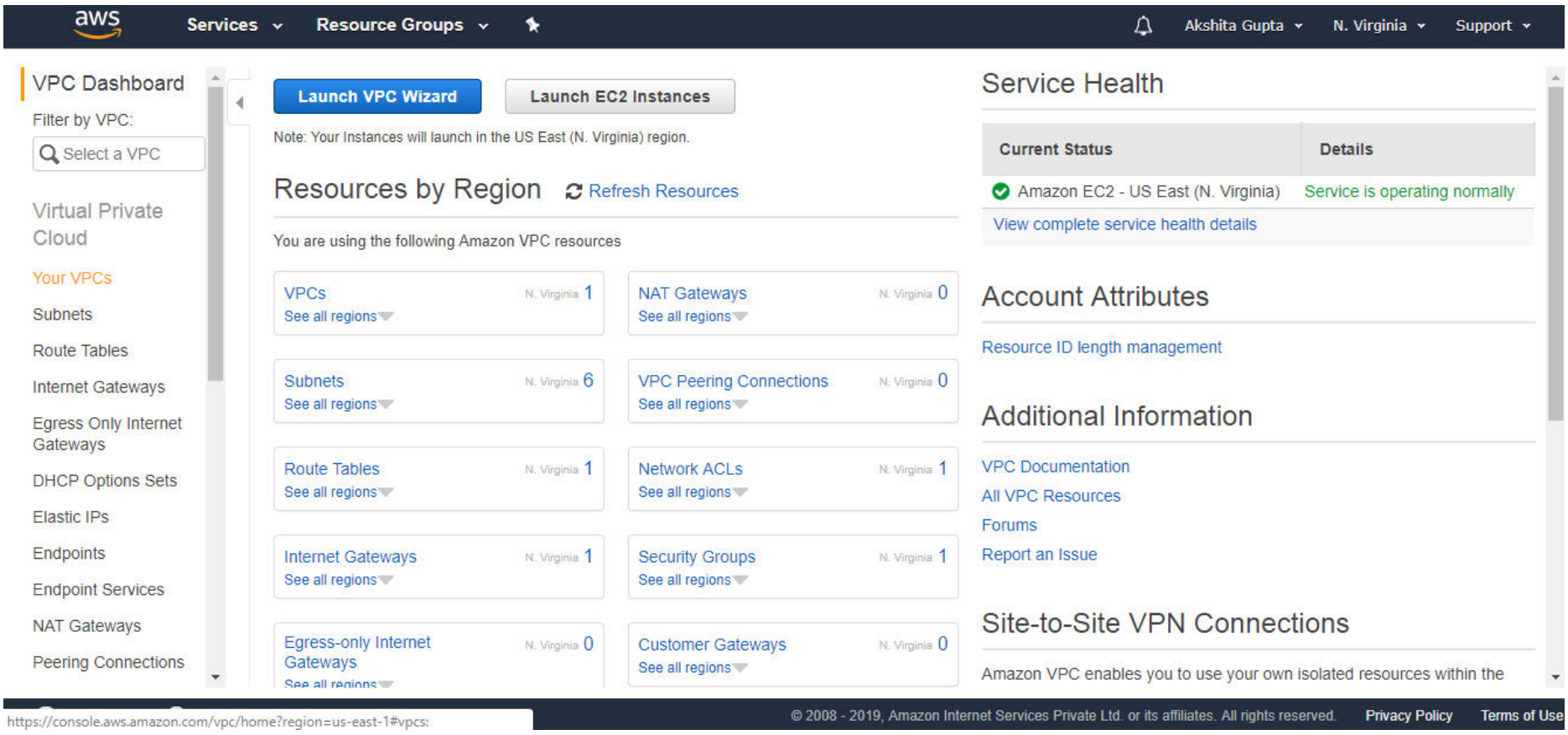
To delete a customer managed policy (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. Select the radio button next to the customer managed policy to delete. You can use the search box to filter the list of policies.
4. Choose **Actions**, and then choose **Delete**.
5. Follow the instructions to confirm that you want to delete the policy, and then choose **Delete**.

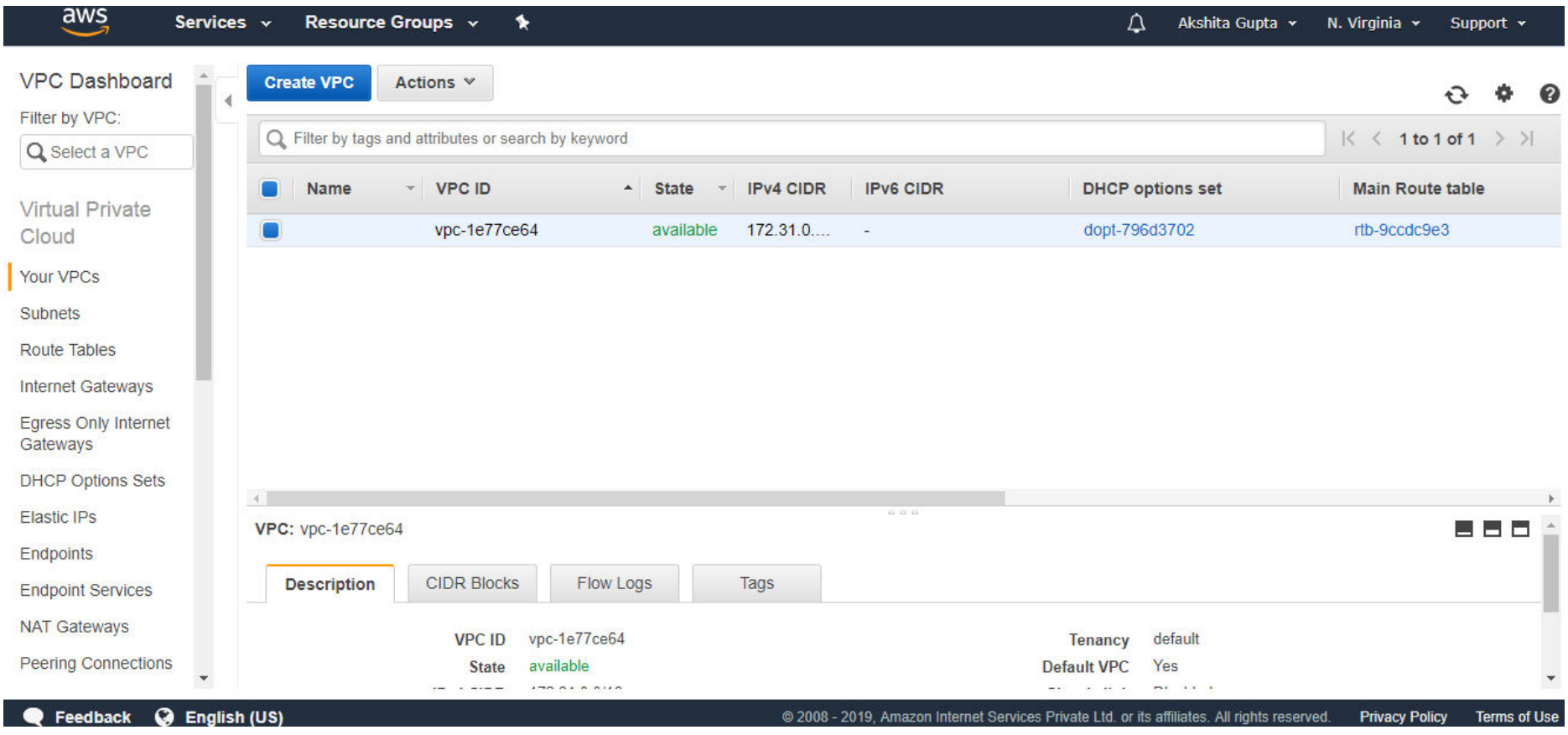
6. Virtual Private Cloud (VPC)

Creating a Custom VPC

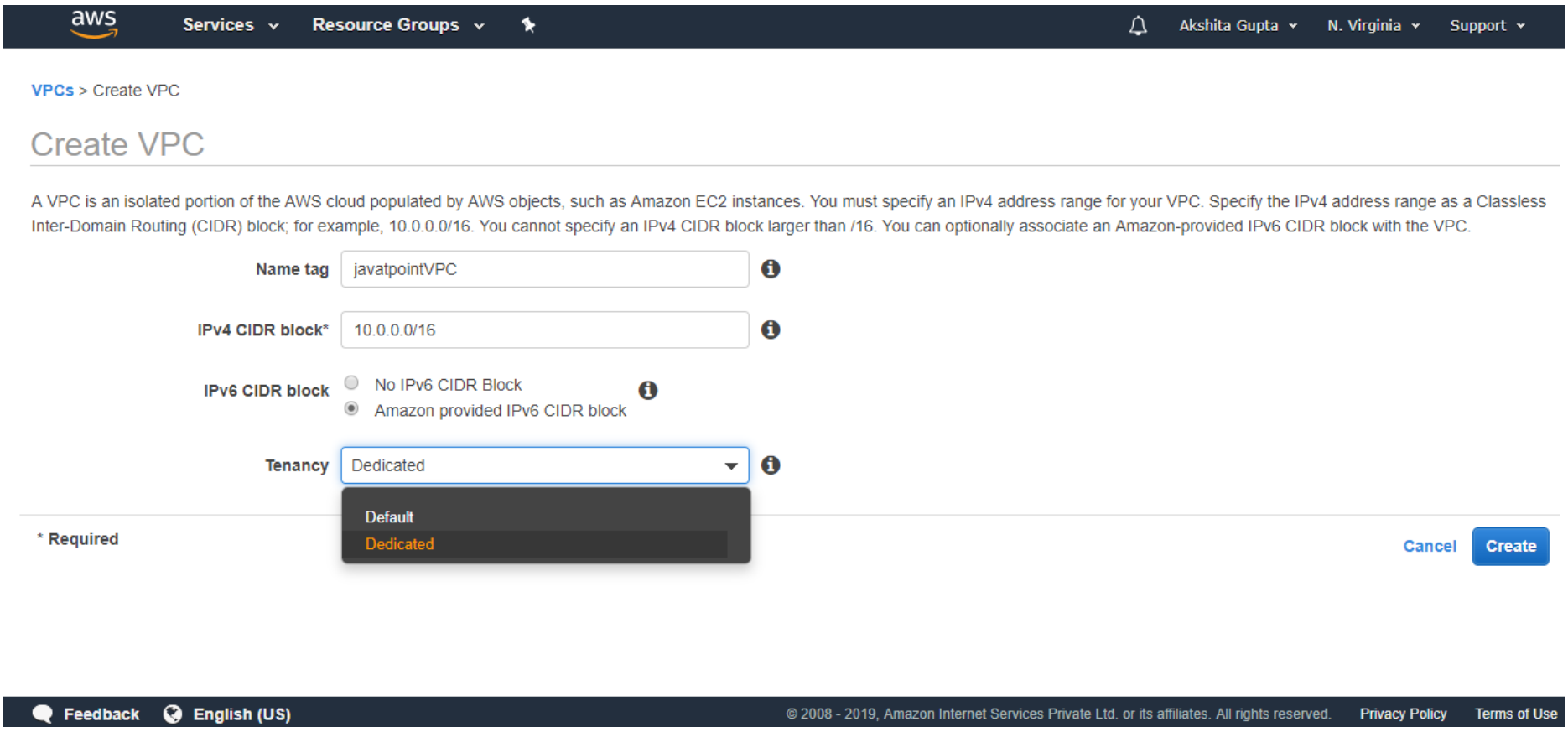
- Sign in to the AWS Management Console.
- Click on the VPC service under **Networking and Content Delivery**.
- Click on the "**Your VPCs**" appearing on the left side of the console.



- Click on the **Create VPC** to create your own custom VPC.



- Fill the details to create a custom VPC.



Where,

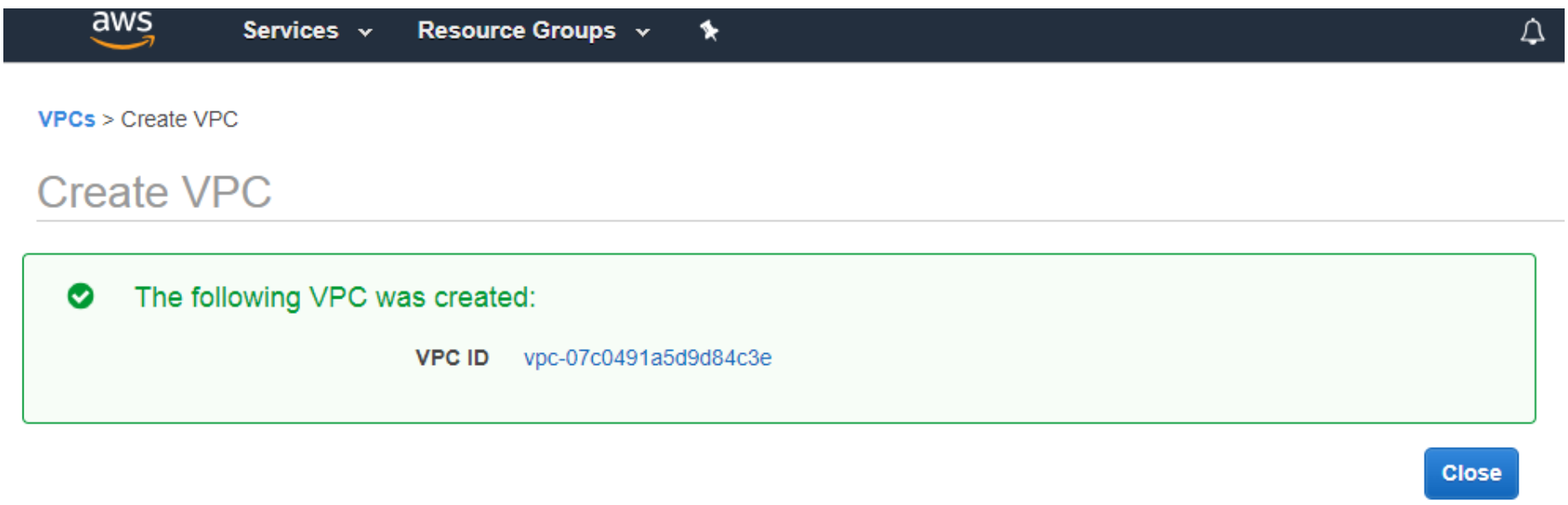
Name tag: It is the name of the VPC that you give to your VPC. Suppose I have given the name "**javatpointVPC**".

IPv4 CIDR block: I make this address block as big as possible. I provide the address block as 10.0.0.0/16.

IPv6 CIDR block: You can also provide IPv6 CIDR block. So, I provide Amazon provided **IPv6 CIDR block**.

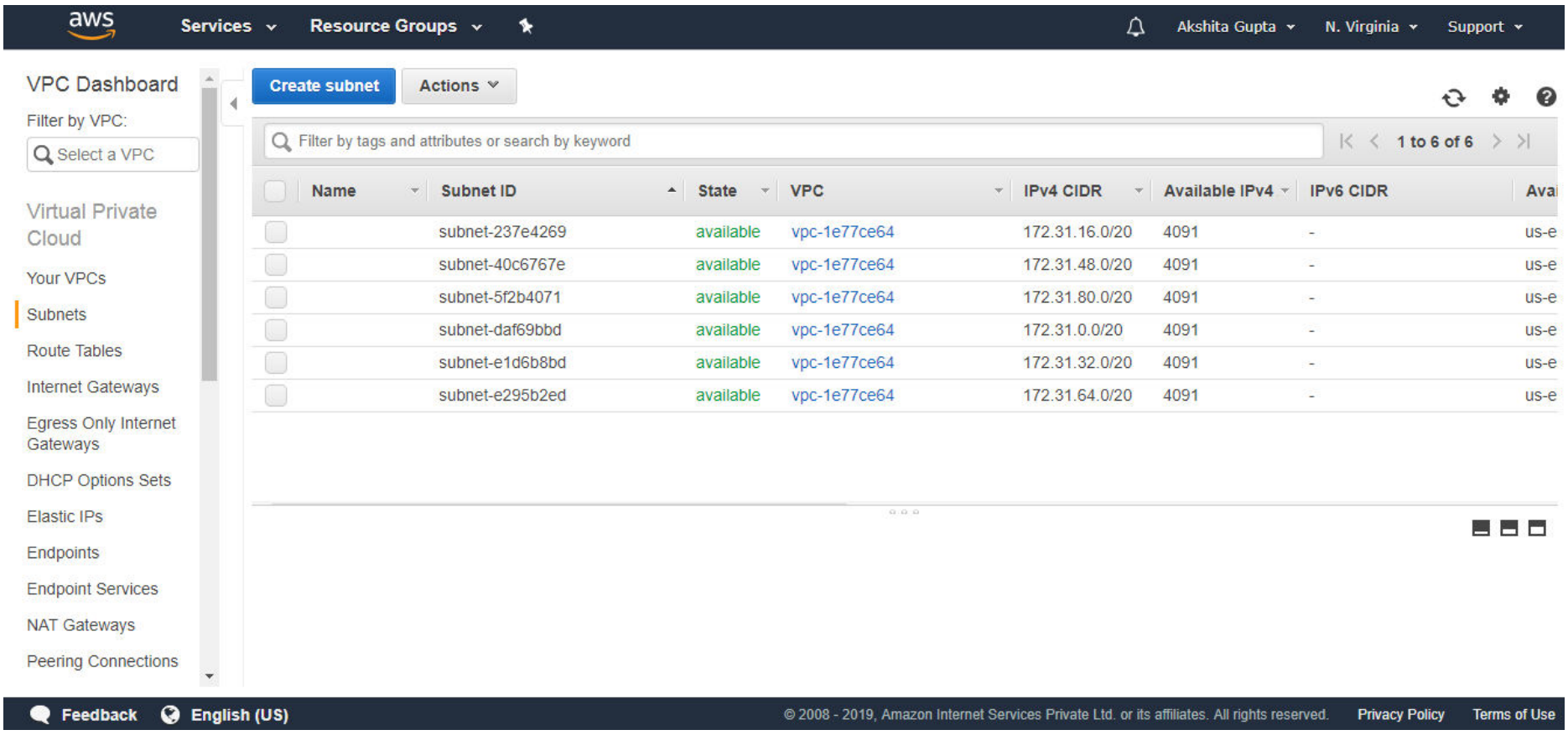
Tenancy: We make it as Default.

- The below figure shows that VPC has been created.



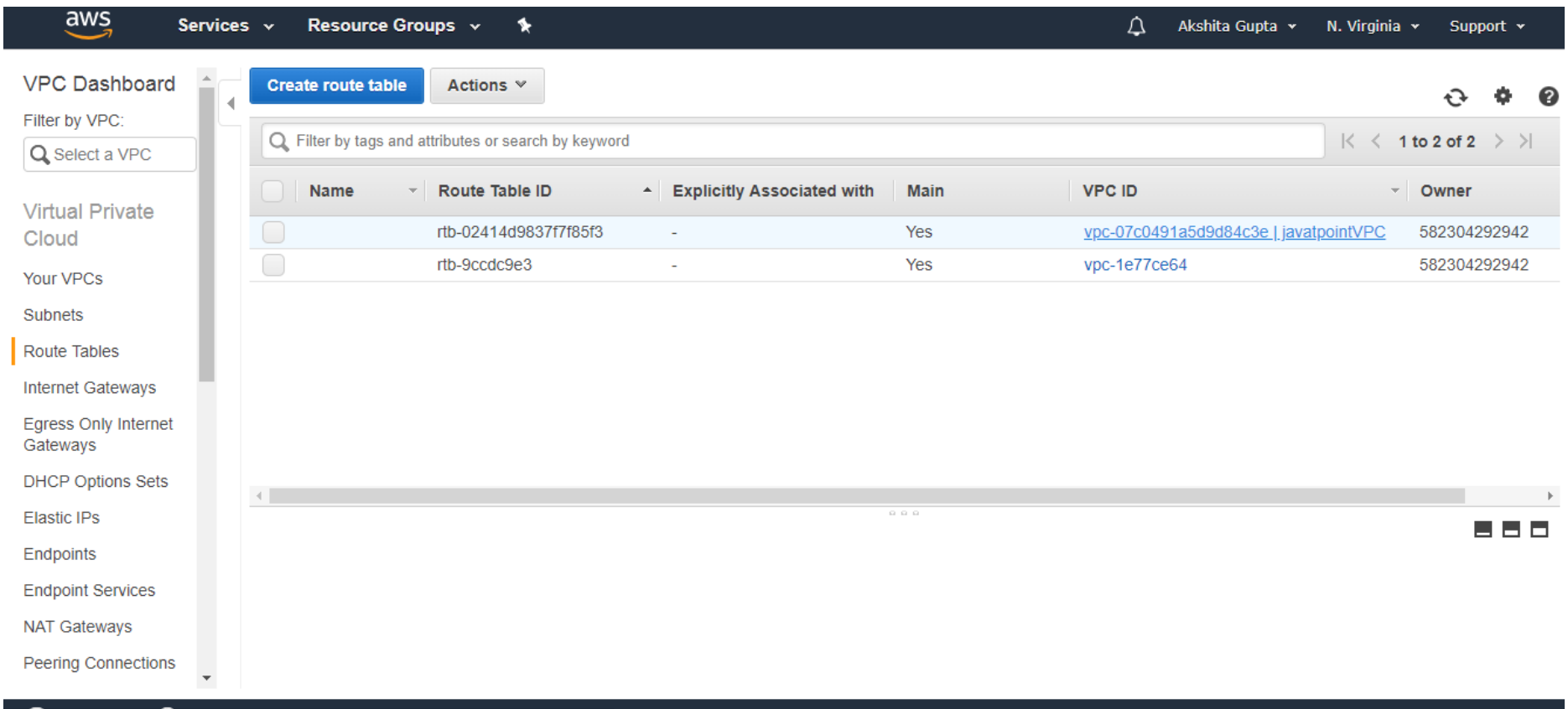
Now we will see what has been created after creating the VPC.

- First, we will look at the subnet.



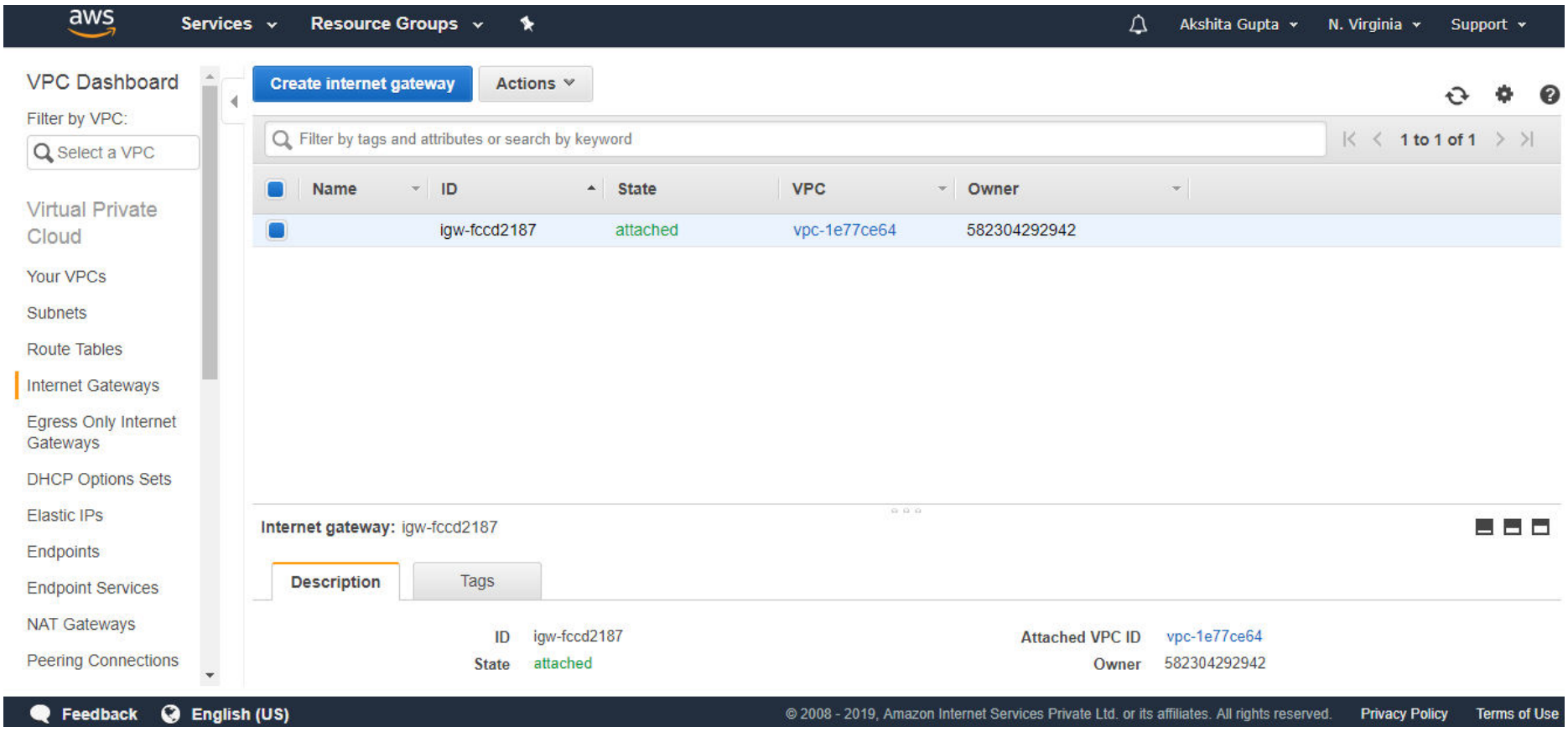
We observe from the figure that all the subnets are of default VPC.

- Click on the Route tables.



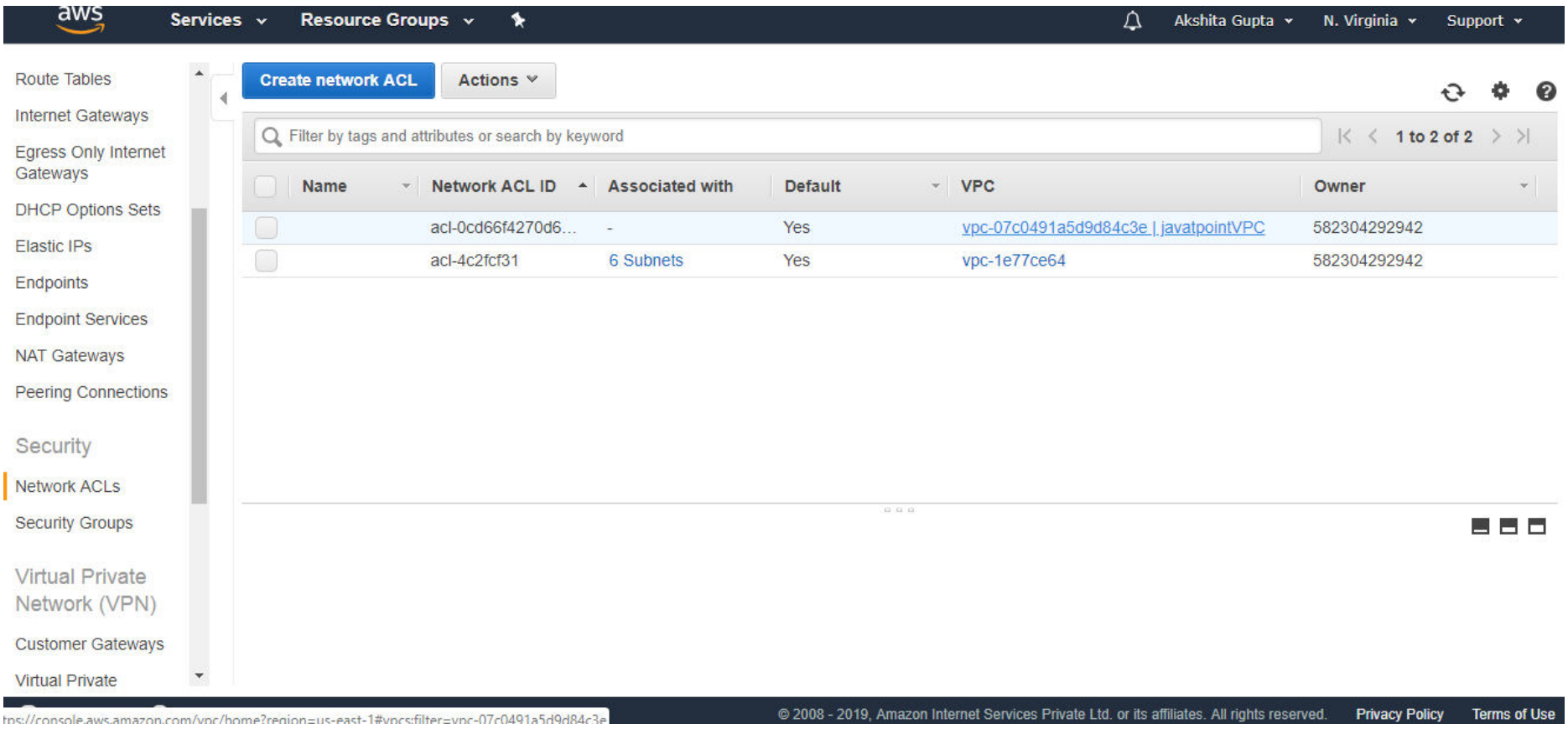
We can observe from the above figure that the route table of "**javatpointVPC**" has been created.

- Now, click on the **internet gateway** to check whether it has been created or not.



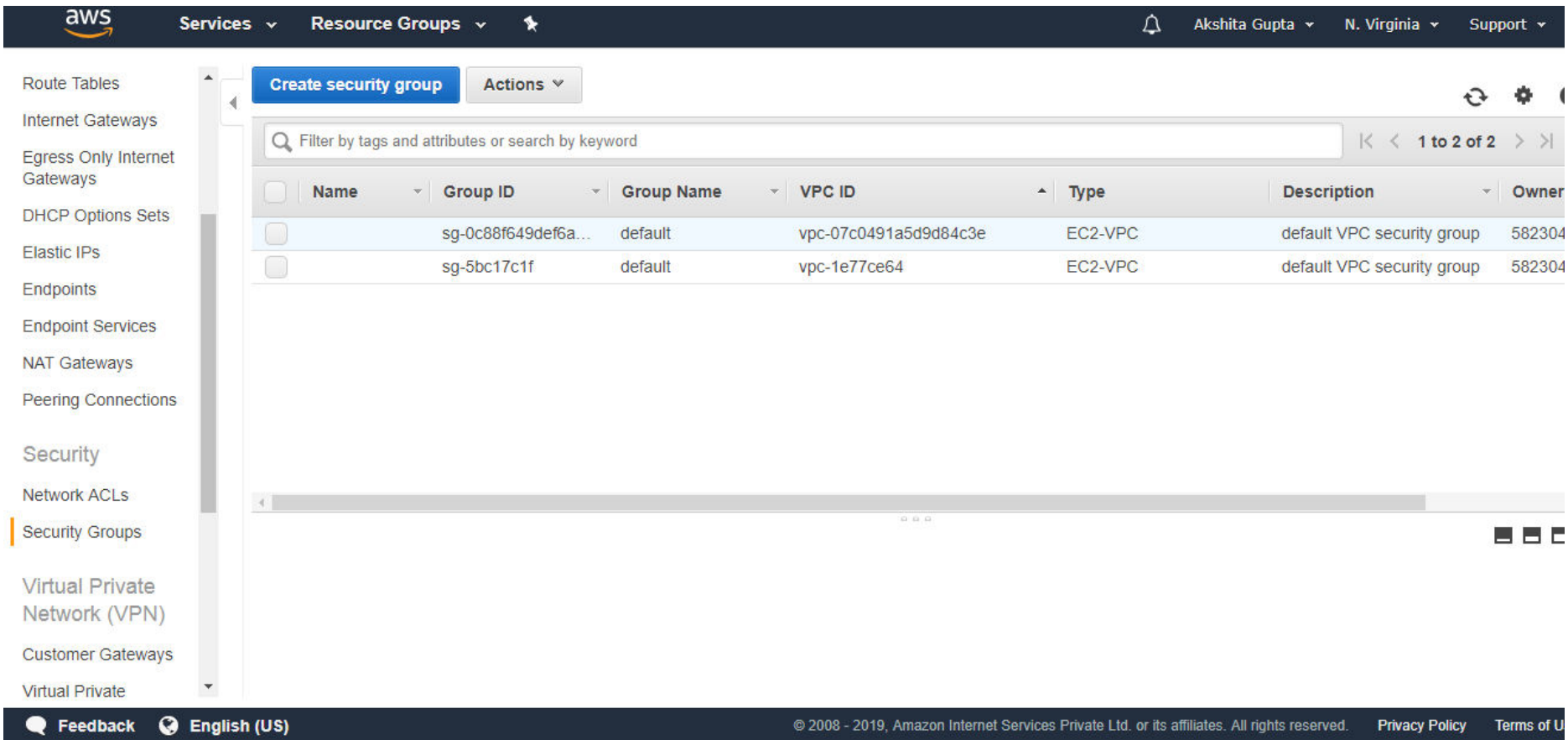
The above figure shows the internet gateway of default VPC. The internet gateway of **javatpointVPC** has not been created.

- Click on the **Network ACL**.



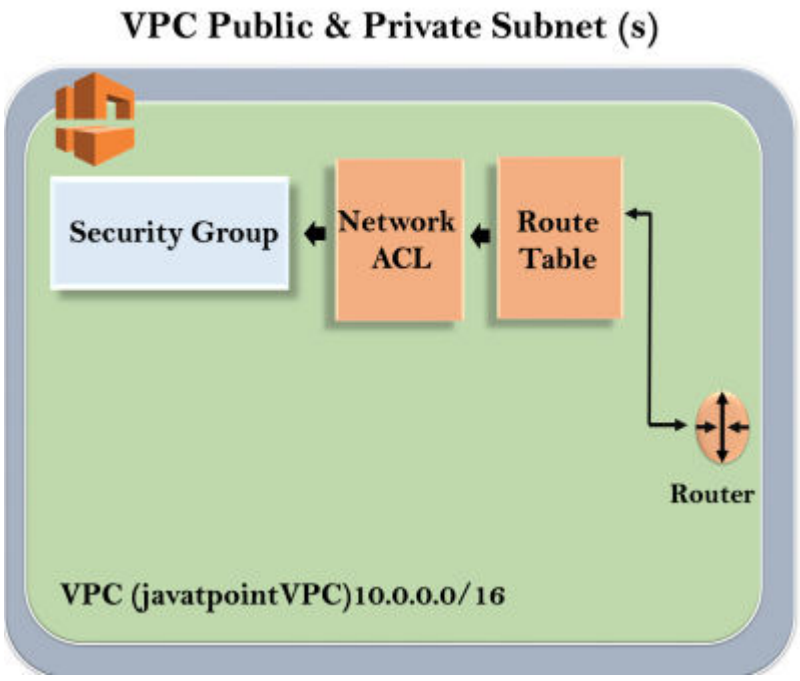
The above figure shows the Network ACL of a VPC that we created, i.e., javatpointVPC.

- Click on the Security Groups.



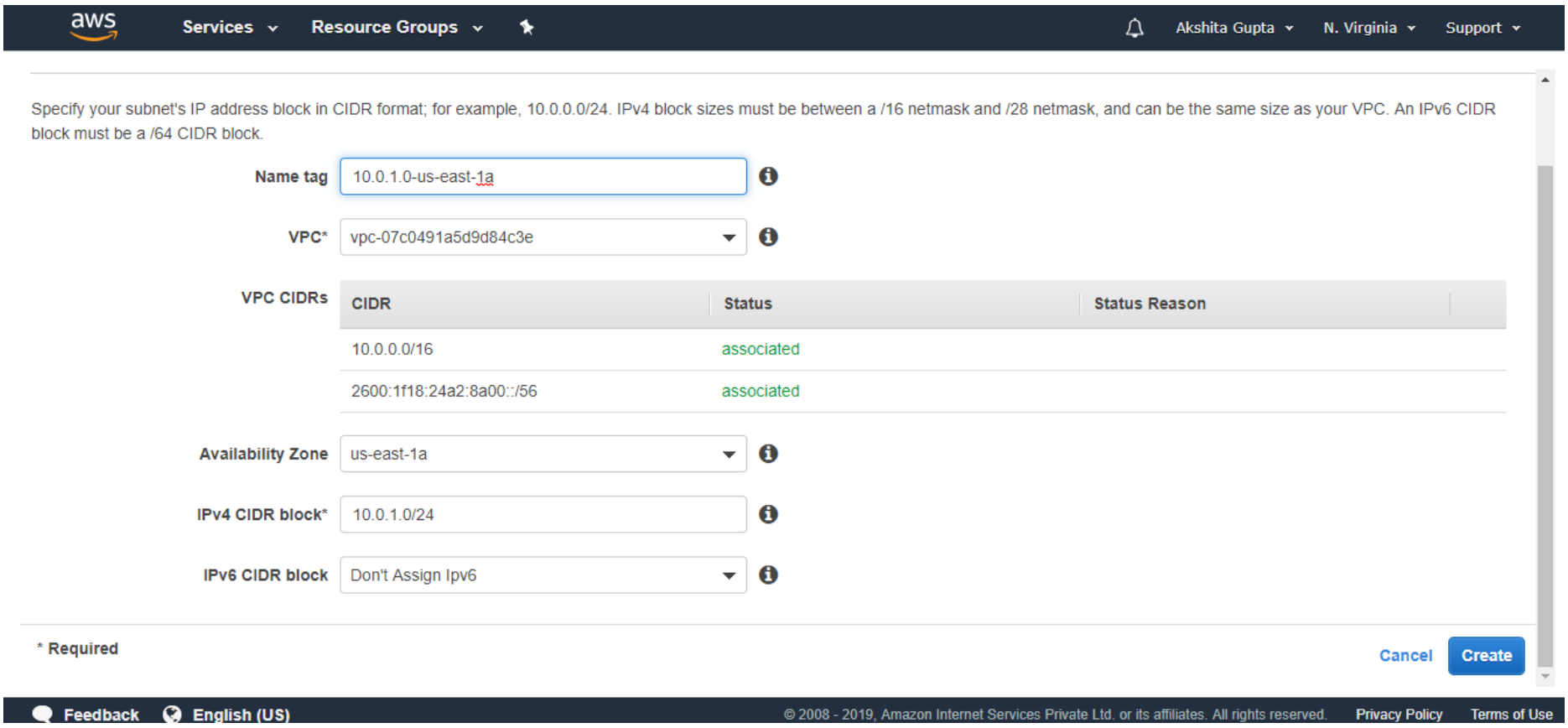
The above figure shows that the security group of VPC, i.e., javatpointVPC has been created.

Till now, we observe that VPC creates three services, i.e., Route tables, Network ACL and Security Groups. It is shown in the below figure:

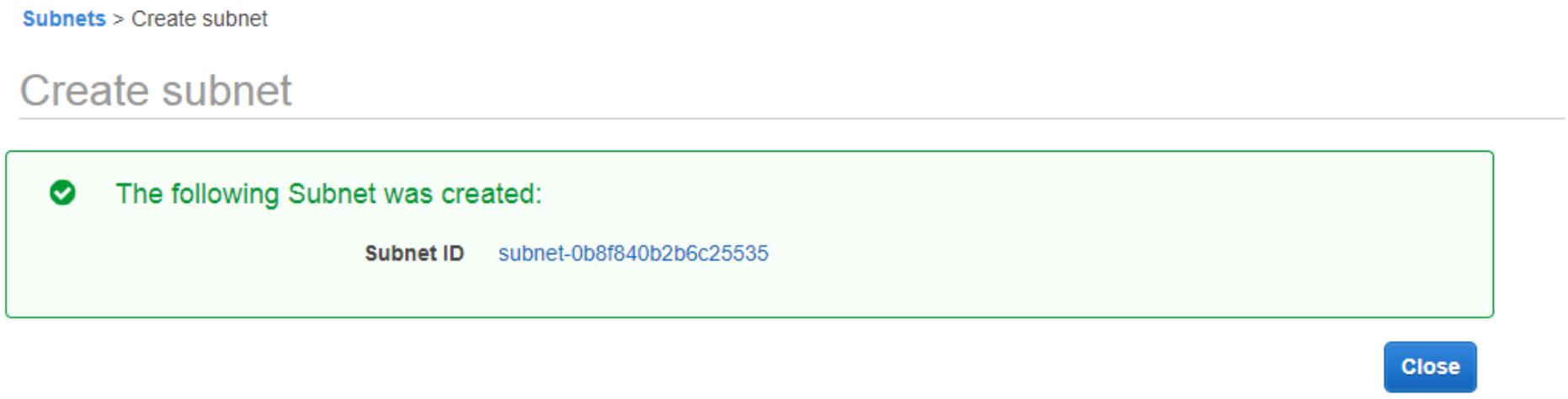


In order to use VPC, we need to create some subnets.

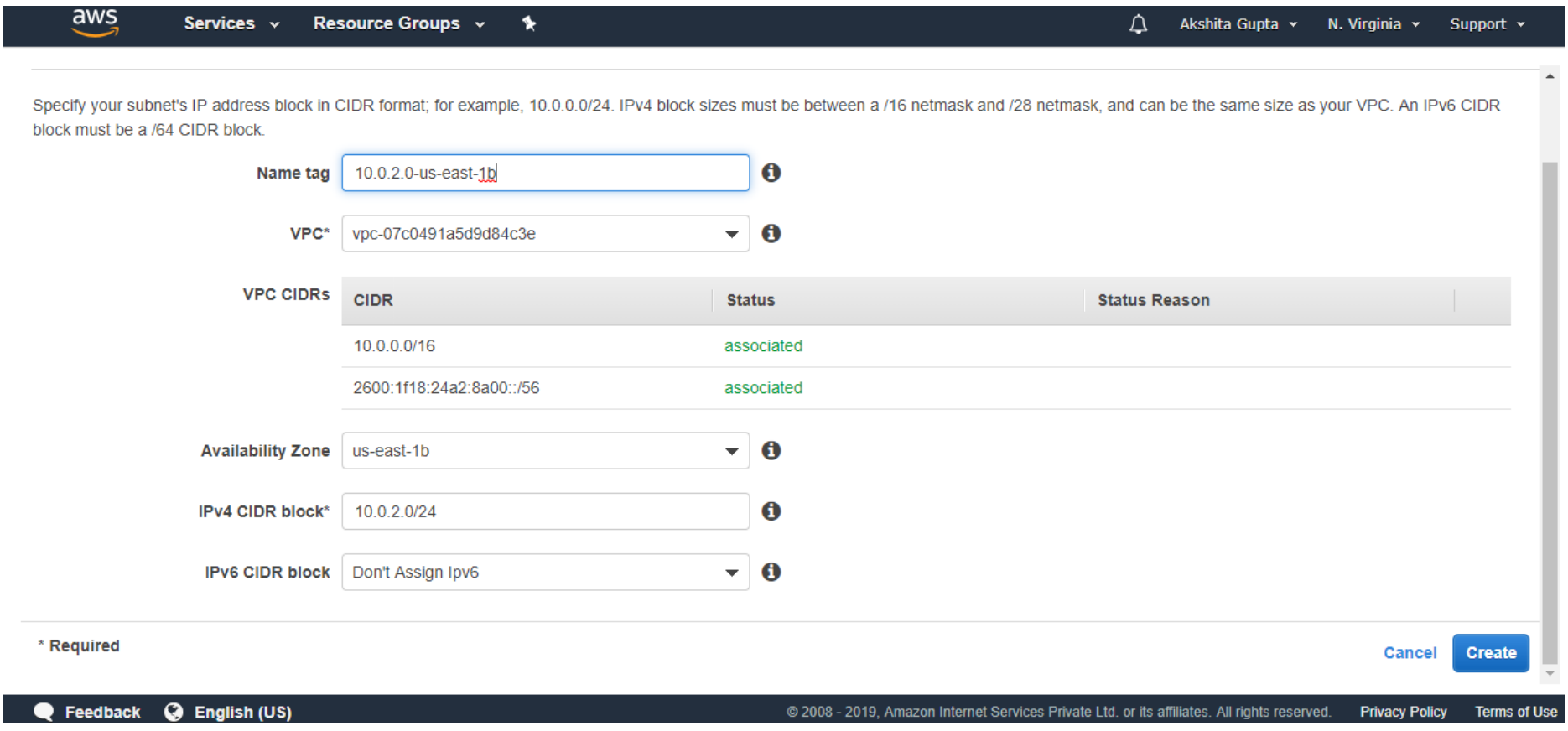
- Enter the details to create a subnet.



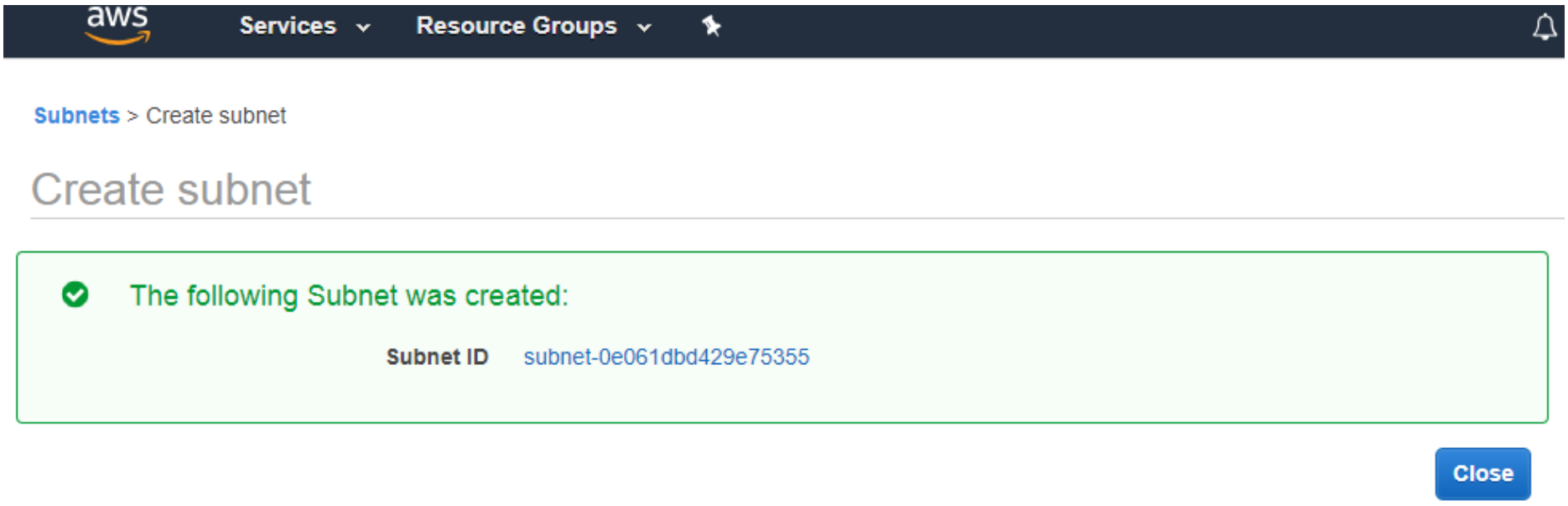
- The below screen shows that subnet has been created.



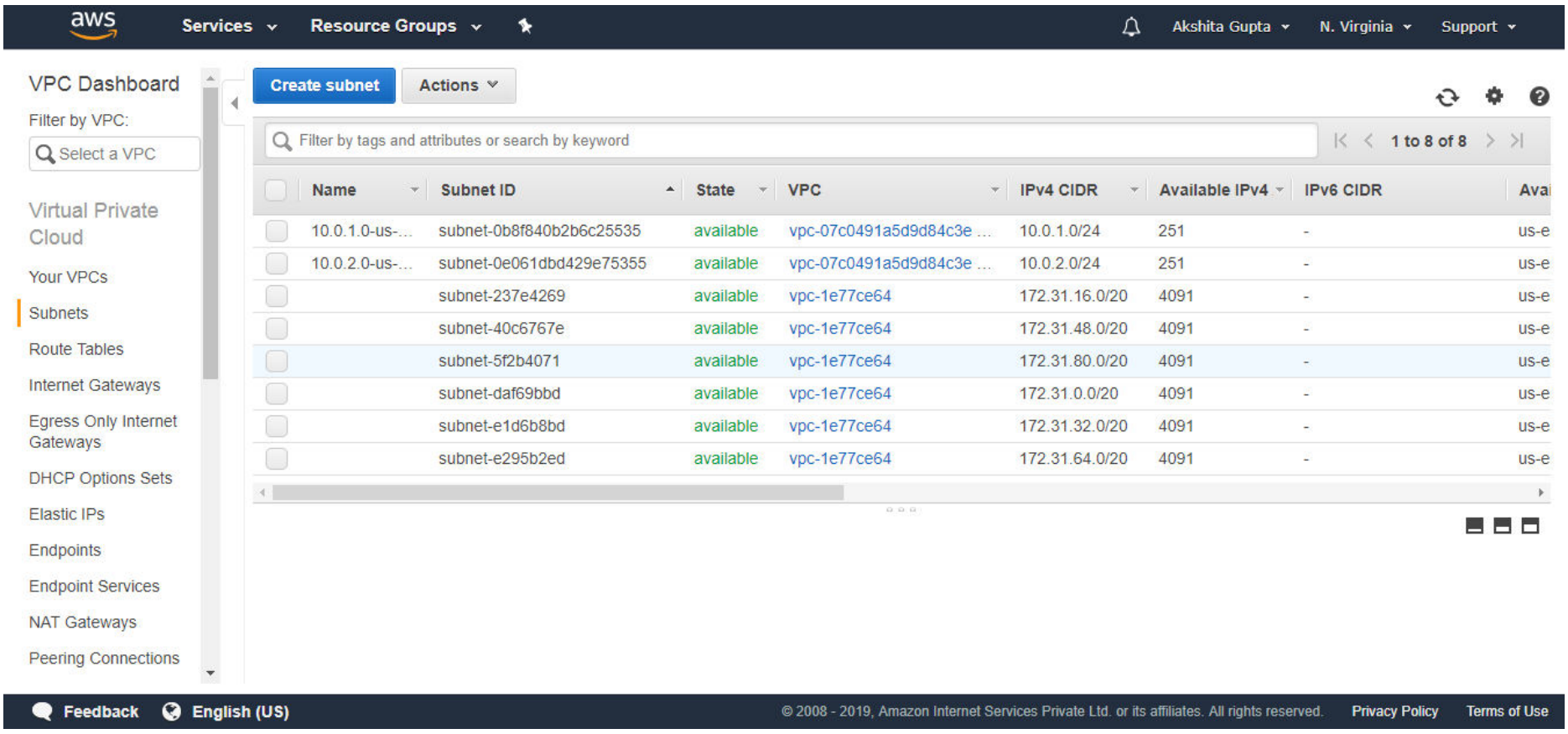
- Now we create one more subnet.



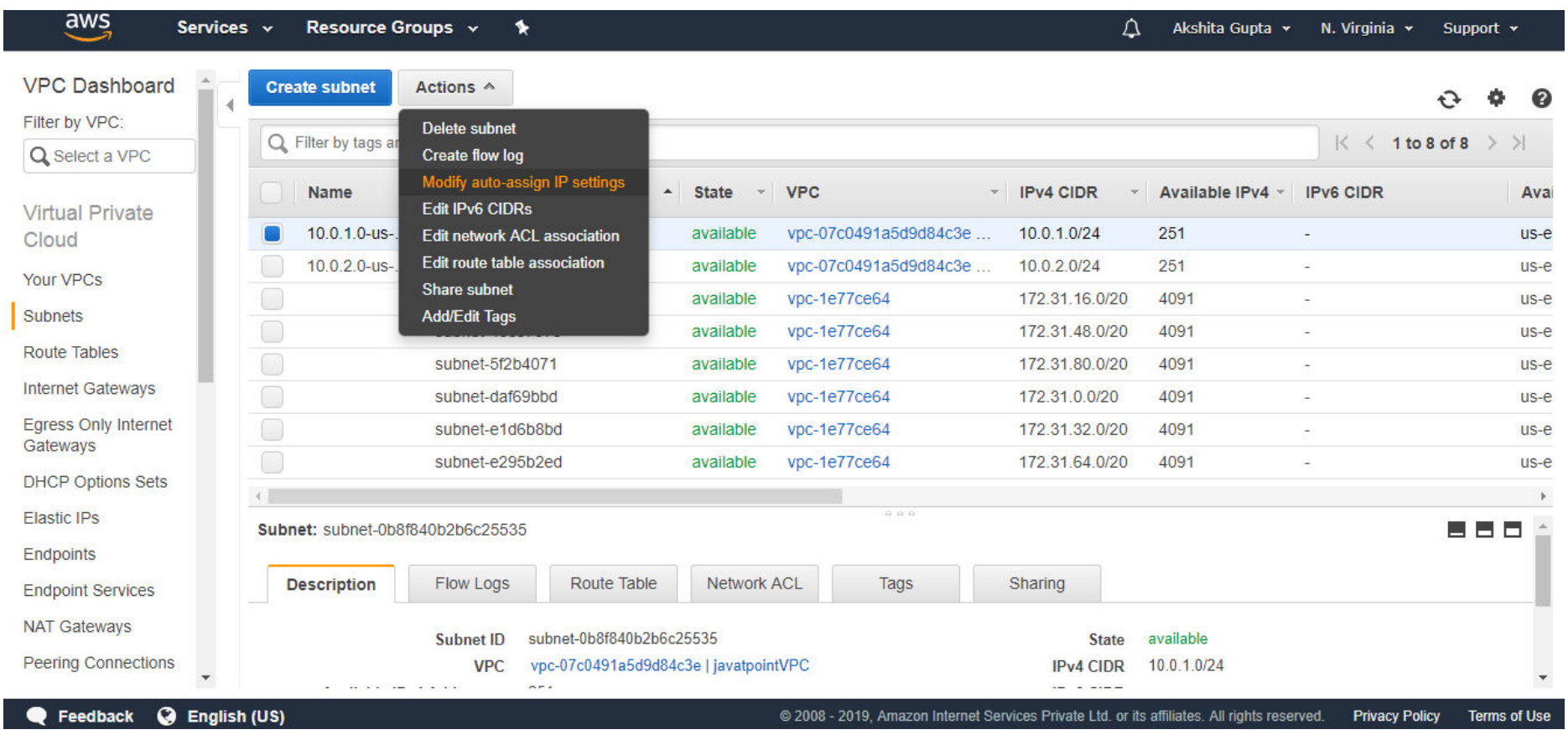
- The below screen shows that subnet has been created.



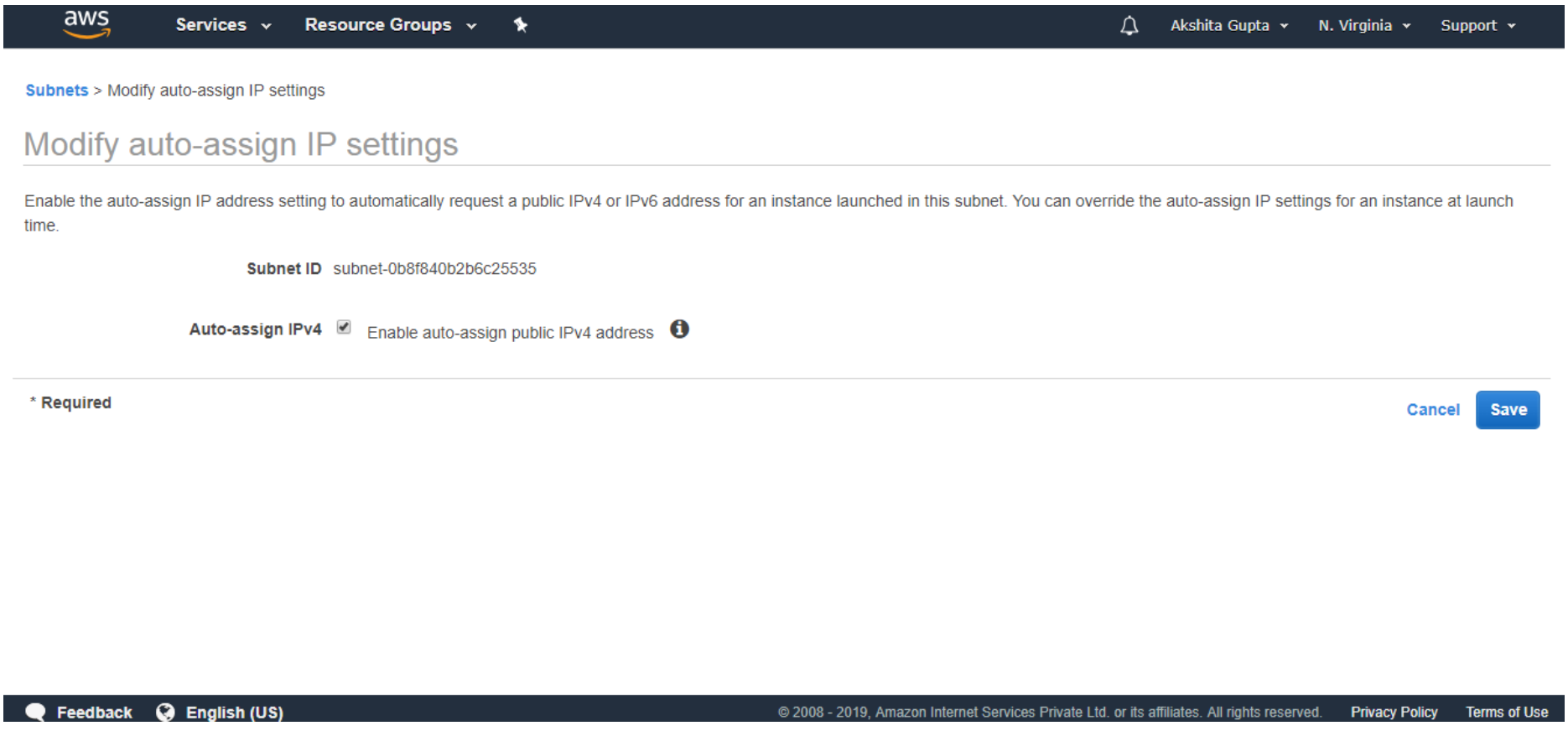
- The below screen shows the lists of all the subnets. The top two subnets are created by us, and others are default subnets.



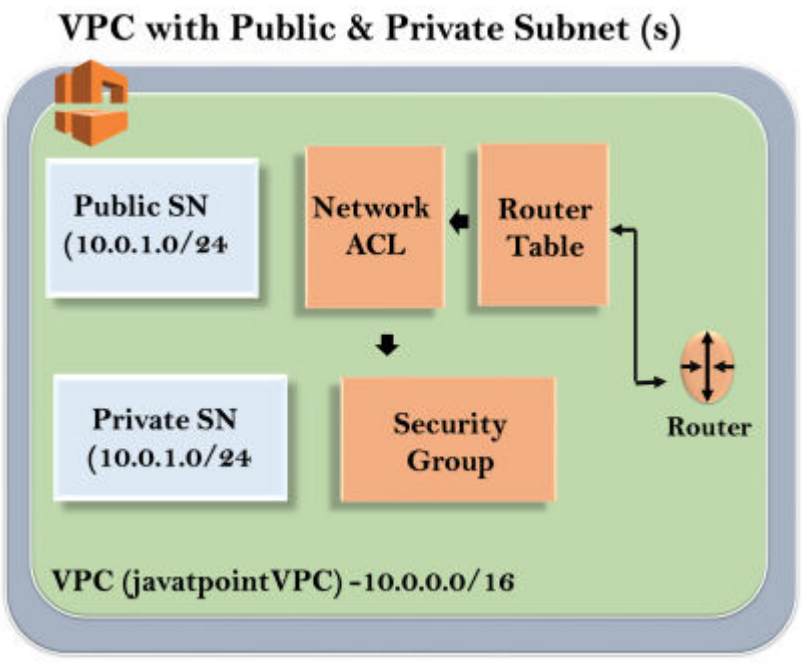
- We already know that in VPC, we have one public subnet and one private subnet. Till now, both are private. So, we make the first subnet as public.
- Now we make a 10.0.1.0-us-east-1a as a public subnet. To make a subnet public, click on the **Actions** drop down menu and then click on the **Modify auto assign IP settings**.



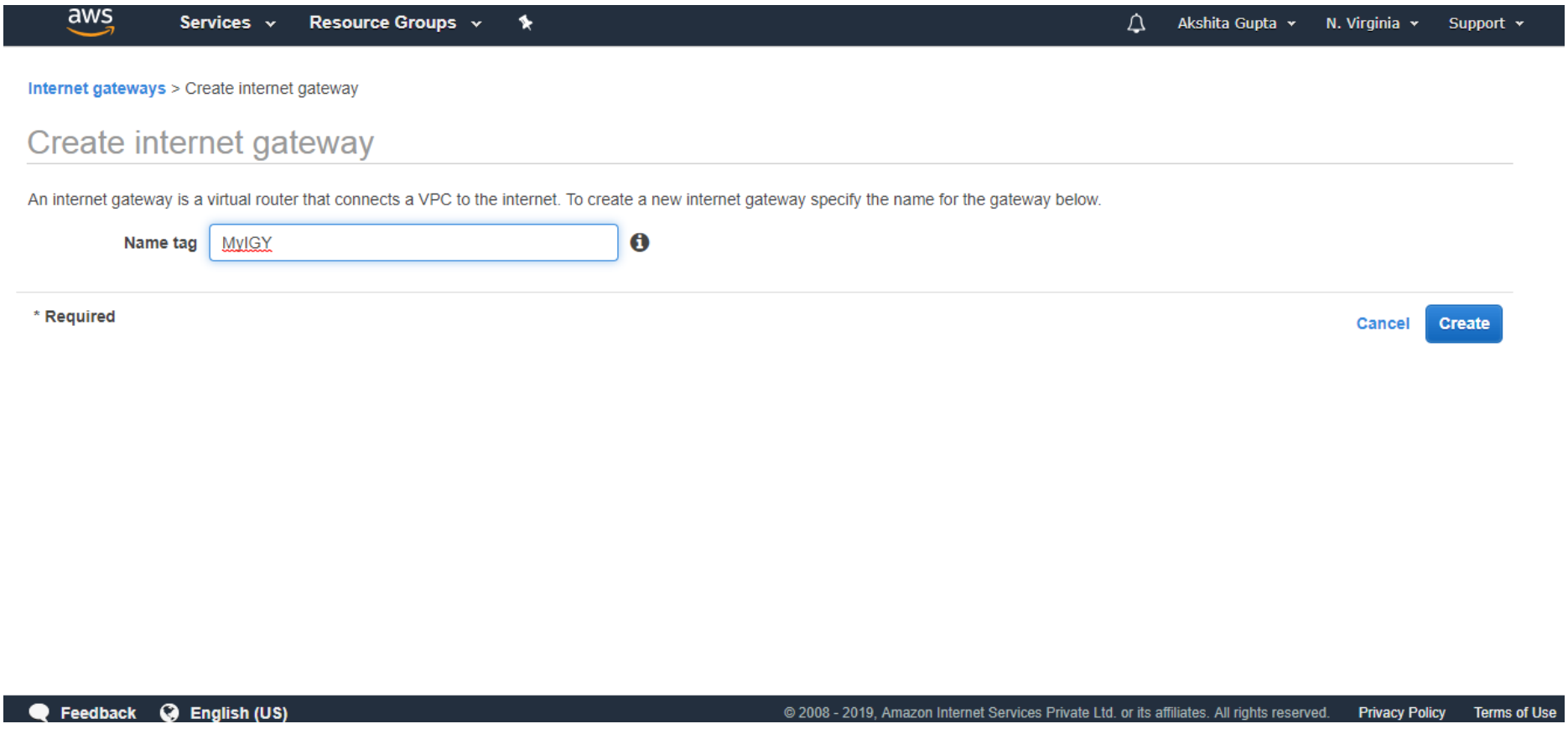
- Check the **Auto Assign IPv4 box**, and then save the settings.



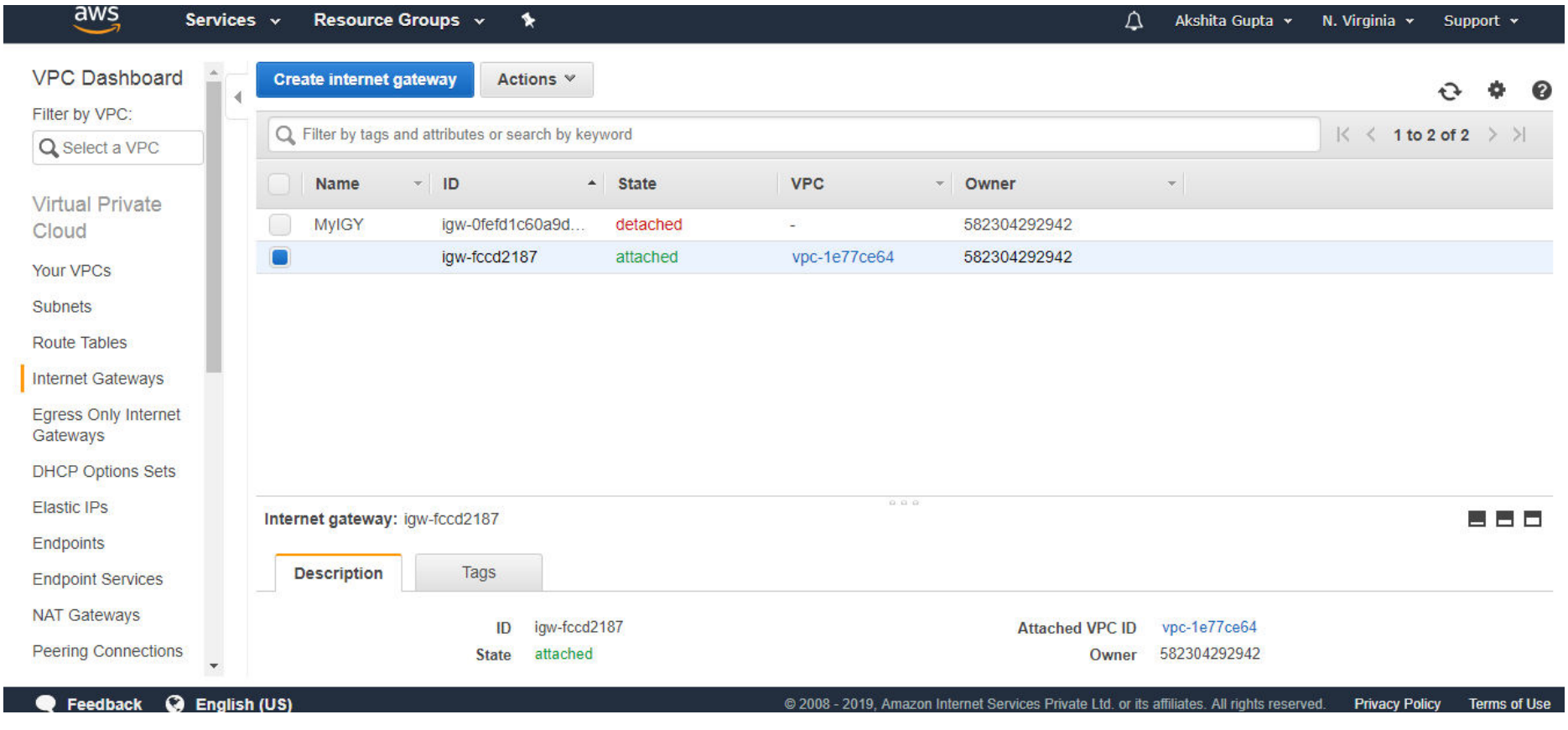
Till now, our VPC looks:



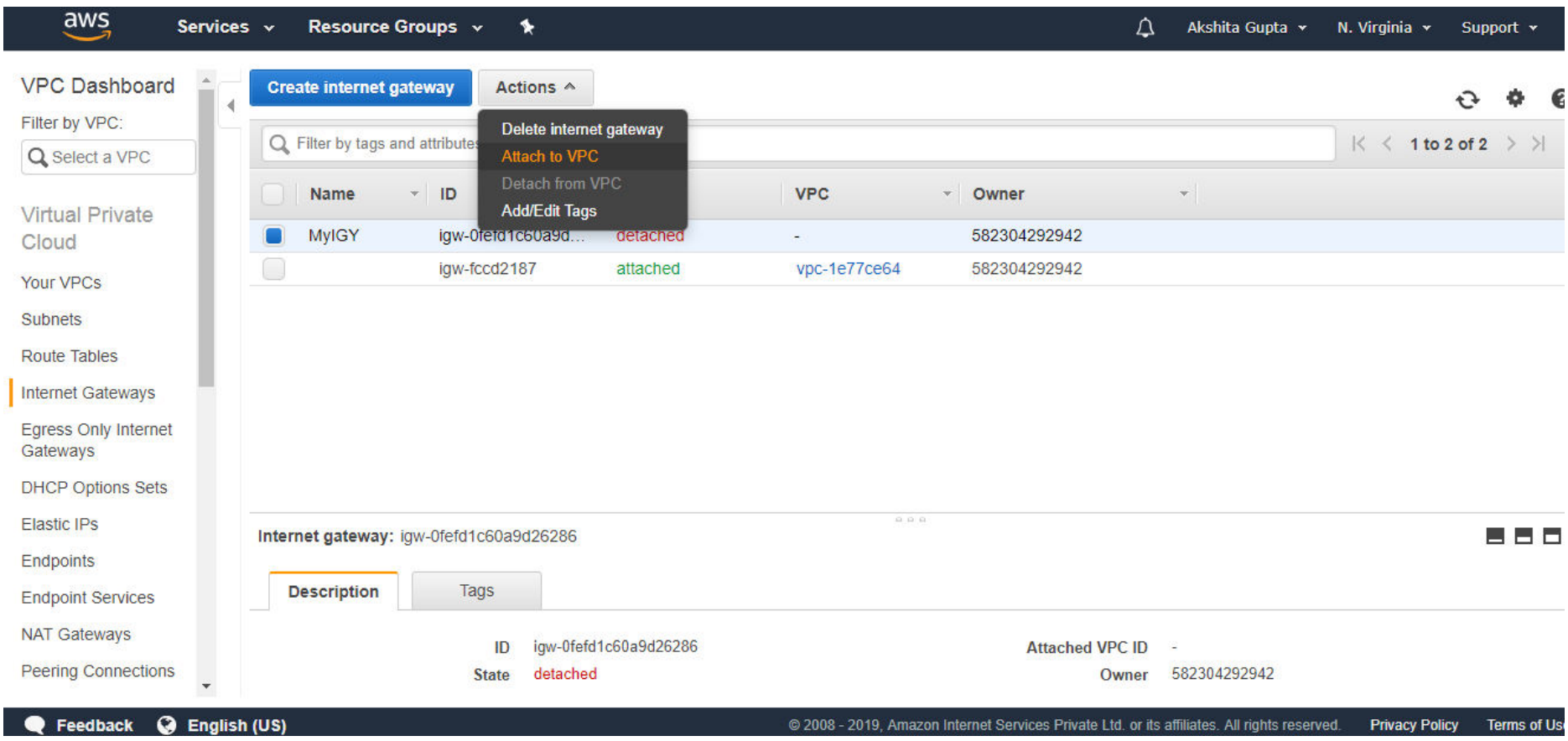
- Now we need a way to get into the VPC, so we need to create an Internet gateway. Click on the Internet Gateway and then click on the Create Internet Gateway.



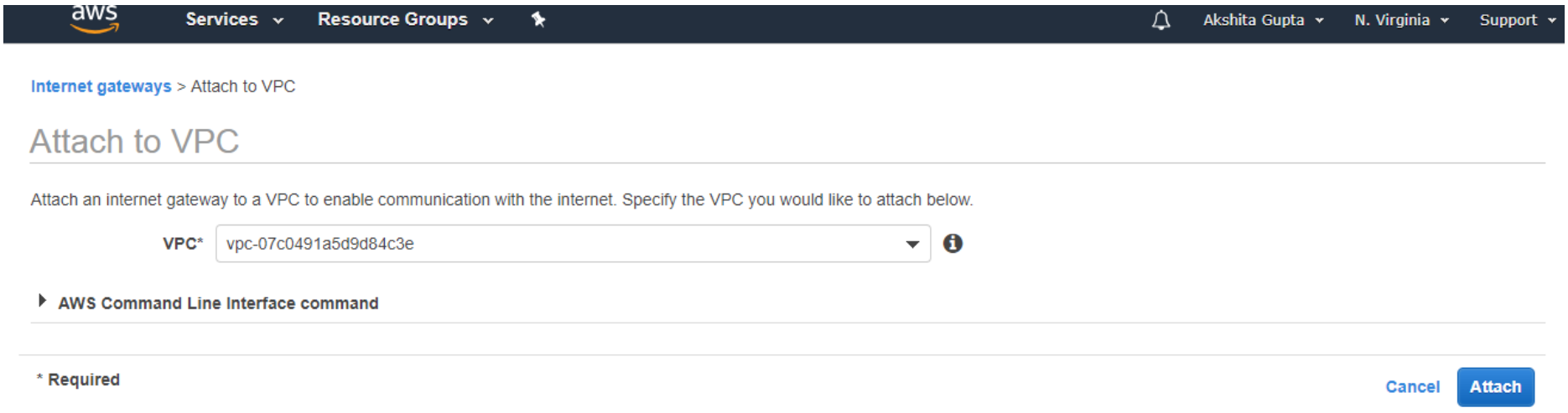
- The below screen shows that internet gateway has been detached from the VPC.



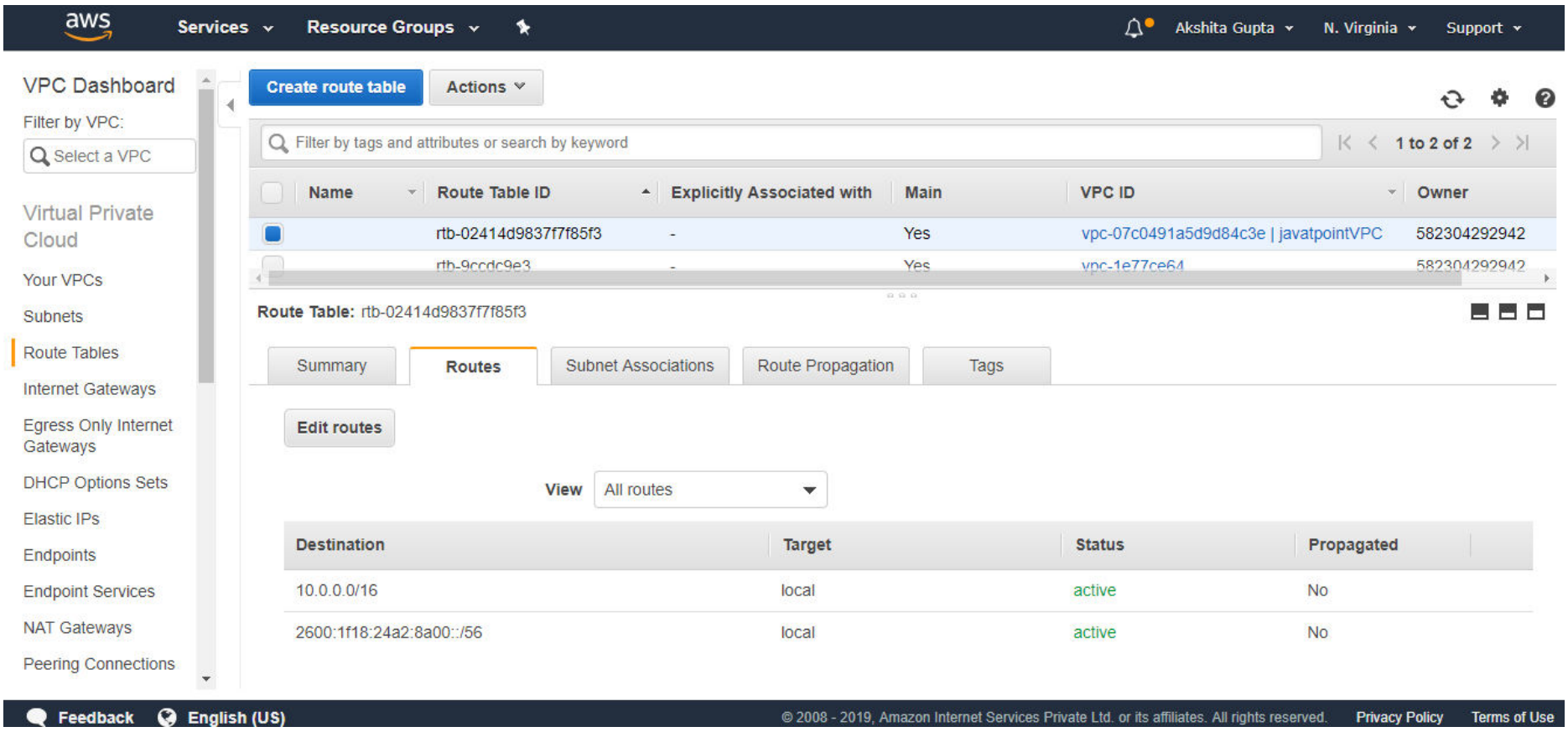
- To attach the internet gateway to VPC, Click on the Actions drop-down menu and then click on the **Attach to VPC**.



- Select the VPC to which you want to attach your internet gateway.

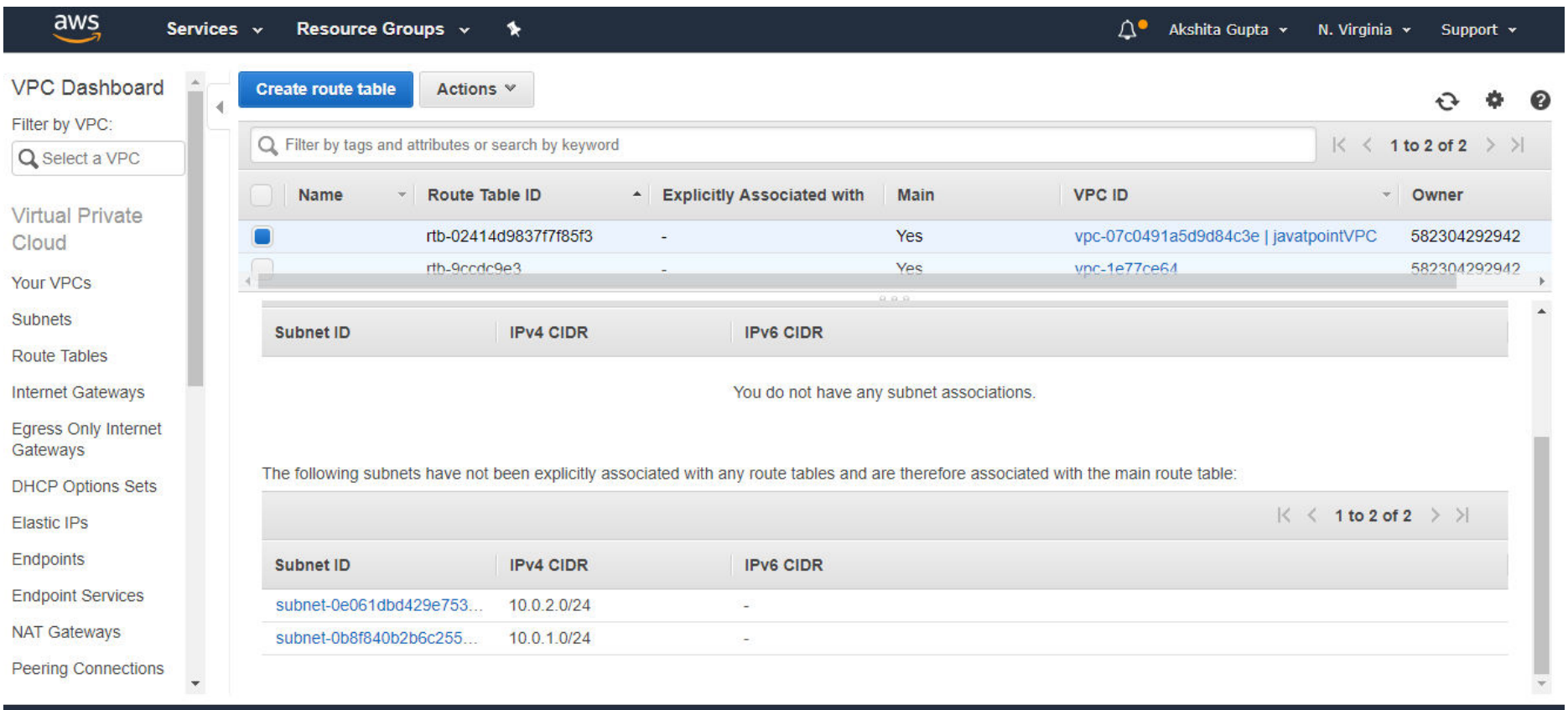


- Click on the Route Table.



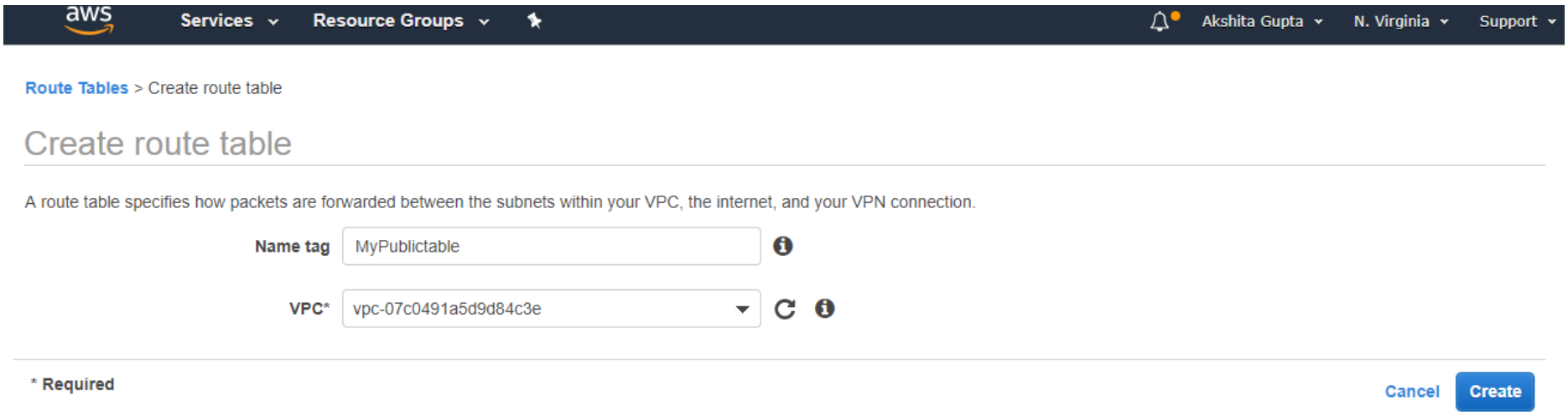
Under Routes, we can observe that the subnets can communicate with each other under these specified routes.

- Click on the subnet associations.

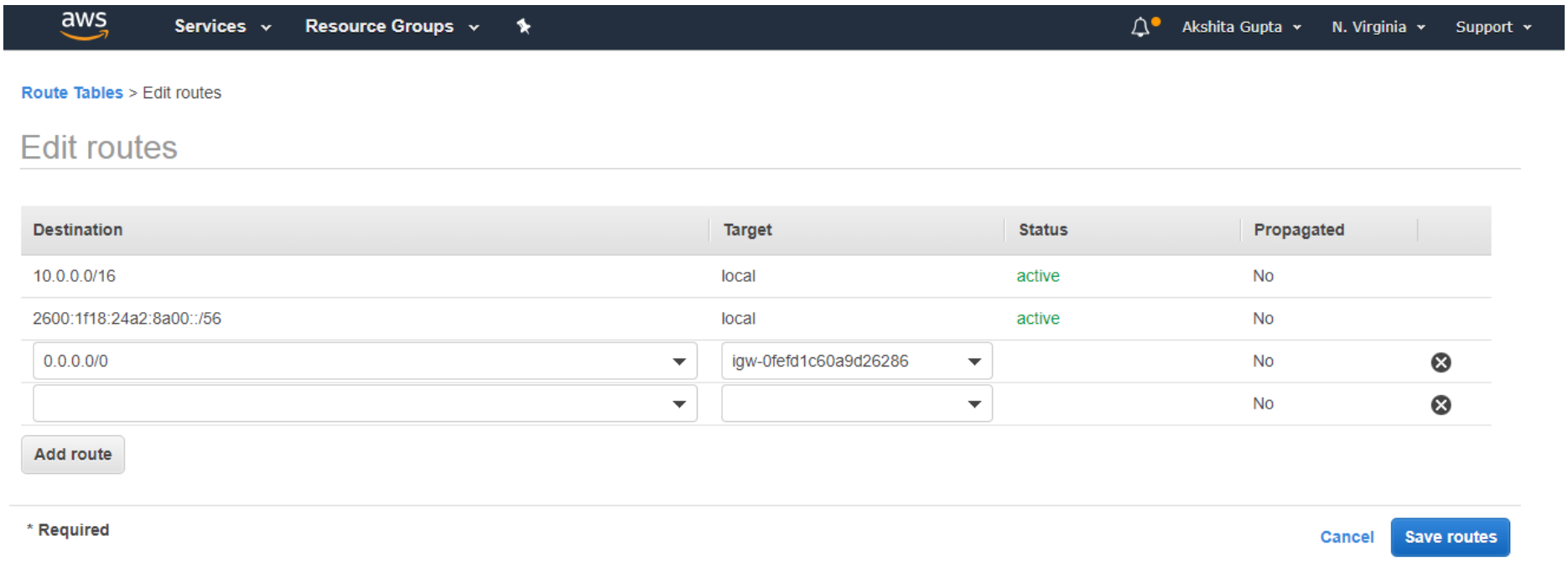


From the above screen, we observe that the subnets we create are automatically associated with the main route table which would be a security concern. To overcome this problem, we create another route table which would be public, and the main table would be private.

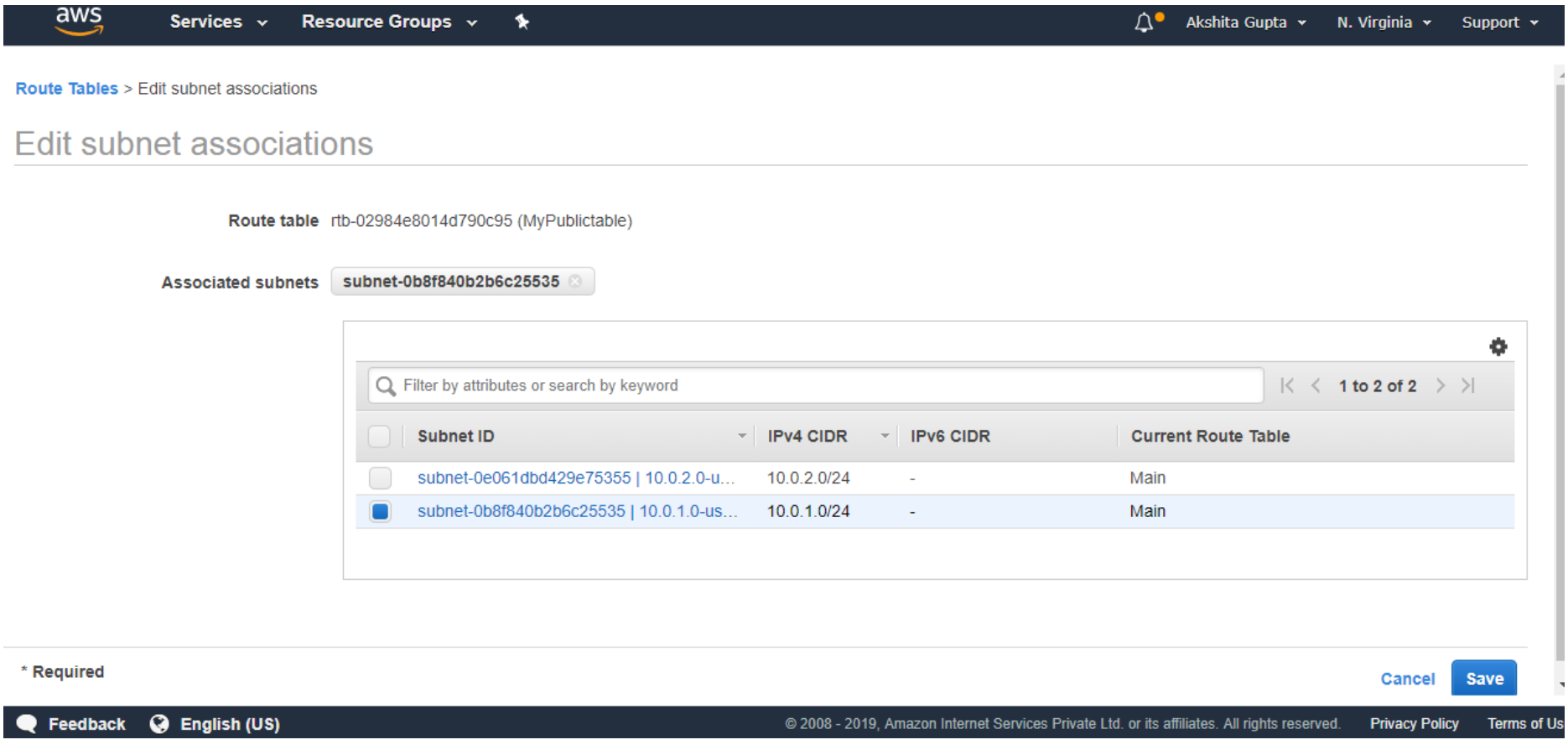
- Click on the **route table** and then fill the following details.



- Edit the routes in public route table.

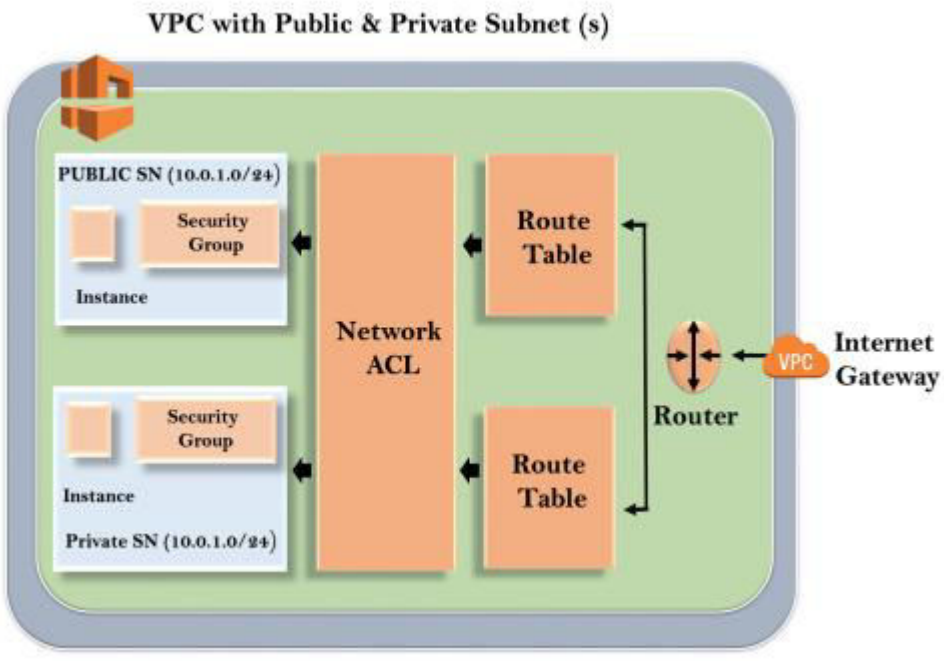


- Click on the **subnet associations** of a public route table and then click on the **Edit subnet associations**. In Edit subnet associations, check 10.0.1.0-us-east-1a subnet box and this includes the subnet in a public route table. An Unchecked subnet is associated with the main route table.



- Now we have the last step left, and the last step is to create two EC2 instances. One EC2 instance is created in private subnet and another EC2 instance is created in public subnet.

Finally, our VPC looks, as shown below:



Important points to remember:

- When you create a VPC, a default route table, Network Access Control List and default security group are automatically created.
- It won't create any subnets, nor it will create a default internet gateway.
- Us-east-1a in your AWS account can be completely different availability zone to us-east-1a in different AWS account. AZ's are randomized.
- Amazon always reserves 5 IP addresses within your subnet.
- You can keep only one internet gateway per VPC.
- Security Groups cannot span VPCs.

VPC Peering

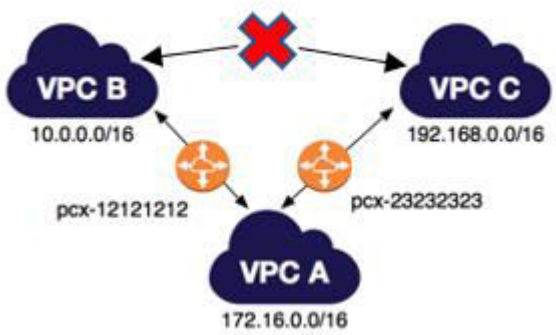
VPC Peering: IPV4 or IPV6 traffic routes between VPCs created to establish communication between one or more multiple VPCs.

AWS definition: “A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. Instances in either VPC can communicate with each other as if they are within the same network. “

- Through VPC Peering, traffic stays within the AWS network and not go over the internet.
- Non-overlapping CIDRs – The 2 VPCs you are trying to peer, must have a mutually exclusive set of IP ranges.
- Transitive VPC Peering – not allowed i.e

To understand what is transitive peering, Please refer to the following image.

(*If VPC A & B have peered and VPC A & C have peered, VPC B & C cannot share contents **until** there is an exclusive peering done between VPC B & C)*



As mentioned earlier VPC peering can be done to another region or another account as well.

Will VPC Cost me?

No. VPC itself won't cost you, however, the resources deployed inside the VPC and data transfers are done will cost you.

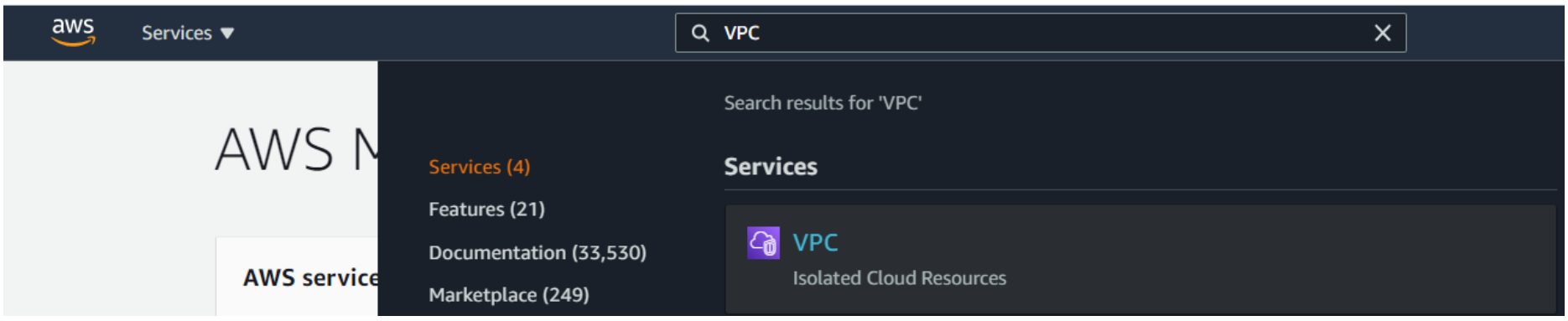
How to establish peering between 2 VPCs in AWS

In this section, we are going to see, how to establish a VPC Peering between 2 VPC in AWS.

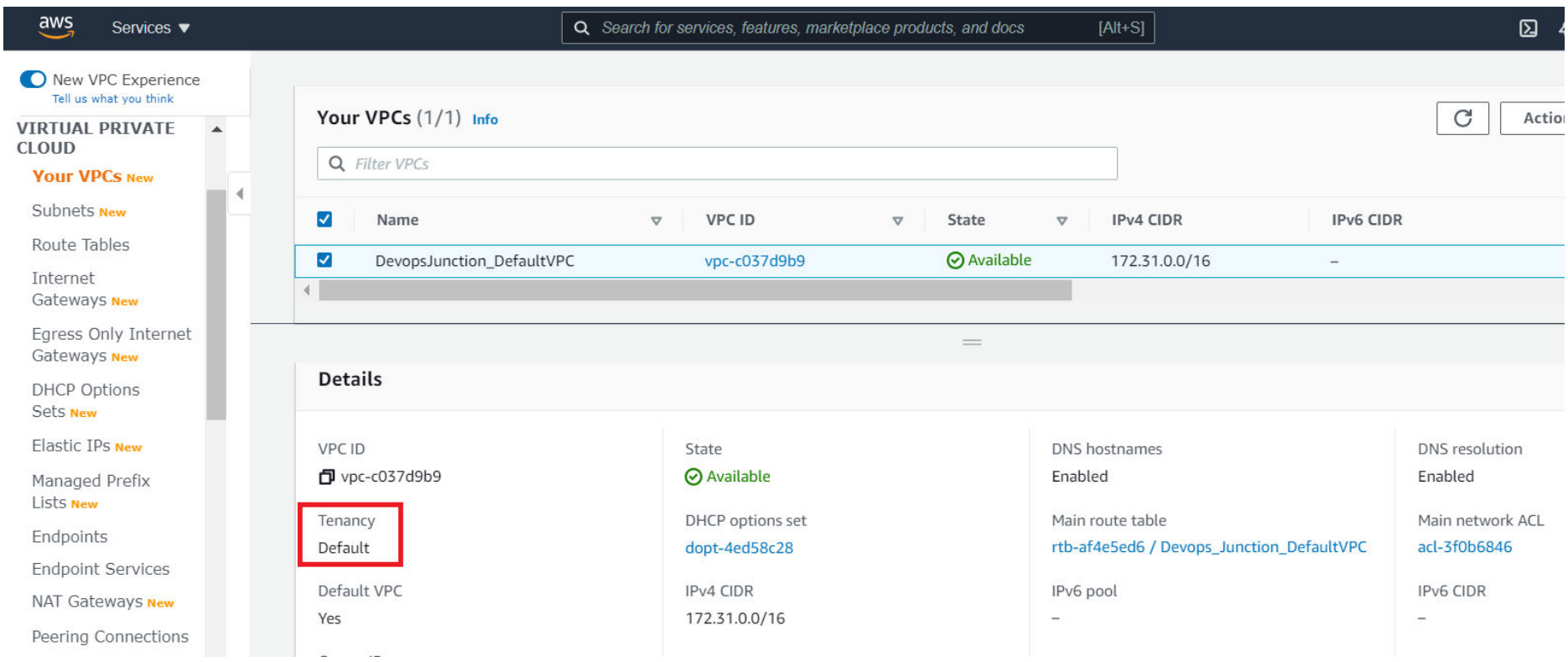
Since every AWS region comes with a default VPC, let's connect a default and custom VPC through peering for this article.

Find your Default VPC

To find your default VPC, in your AWS Management Console type VPC in the search bar:



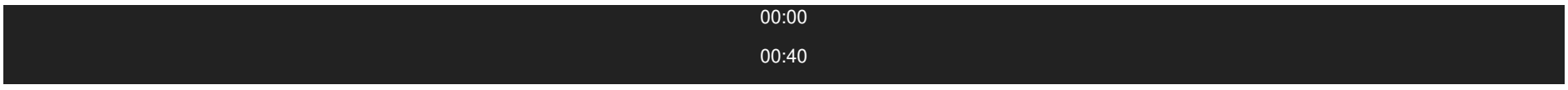
On the left navigation panel click on 'Your VPC' and if Tenancy is 'Default' then it is your default VPC.



Create a custom VPC

Name your VPC and use the IPV4 CIDR range of your choice, I have used 10.0.0.0/16 for this example.

Video Player



At the end of the above video, you should be able to see 2 VPC:

Note: Following components would be auto-created when you are creating a new VPC

- Route Table
- Security Group
- NACL - Network Access Control List

Creating a Subnet in VPC

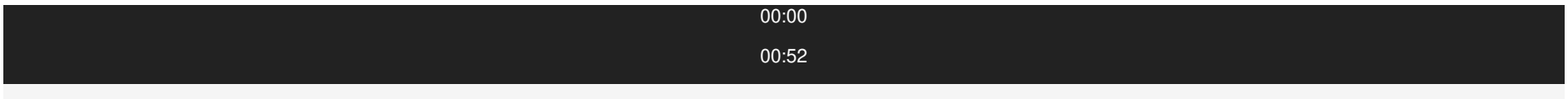
VPC (Virtual Private Cloud) is your private cloud inside AWS as we mentioned earlier. typically we have to divide our big network into multiple subnets so we can place different resources.

Depends on our requirement we can create multiple subnets.

For example, A Subnet allowed to connect to a public network for webserver hosting, A Subnet for a database with no public access and monitored IN/OUT connections (private subnets) etc.

Now we are going to create a subnet in our newly create VPC, Let's navigate to Subnets and click on the ‘Create Subnet’ and follow the video given below

Video Player



Note: Subnets for Default VPC are auto created but not for custom VPC, so go ahead and create one

Creating EC2 instances

Let’s create an EC2 instances under each VPC (default and custom).

Under Compute, select EC2 -> Launch instances -> (in the Configure Instance Details form be sure to select the correct VPC under ‘Network’)

For the instance we are creating under the Custom VPC add the following User_Data, A Startup script to install the apache HTTPD service

```
#!/bin/bash
yum update -y
yum install -y httpd.x86_64
systemctl start httpd.service
systemctl enable httpd.service
echo "Hello world from $(hostname -f)" > /var/www/html/index.html
```

Follow the attached video guide for completing the EC2 creation for both VPCs.

Video Player

00:00

01:14

Creating Internet Gateway for the Custom VPC created

As we have mentioned earlier, we are going to use the EC2 instance we have created under the custom VPC for hosting an Apache HTTP server.

In order to allow traffic, we must create a Gateway.

For the EC2 under Customer VPC, create an Internet Gateway and make sure to add the following entries in its Route Table

Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID	Owner
Devops_Junction_DefaultVPC_RT	rtb-af4e5ed6	-	-	Yes	vpc-c037d9b9 | DevopsJu...	339639447745
DevOpsJunction_CustomVPC_RT	rtb-047fd9d573711bf4a	subnet-01f664fecd0062f3e	-	Yes	vpc-0987dfc548599d6b0 |...	339639447745

Route Table: rtb-047fd9d573711bf4a

Summary | **Routes** | Subnet Associations | Edge Associations | Route Propagation | Tags

Edit routes

View All routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	active	No
0.0.0.0/0	igw-077b72b40bf0c0328	active	No

And associate the Custom VPC Public Subnet to it

Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID	Owner
Devops_Junction_DefaultVPC_RT	rtb-af4e5ed6	-	-	Yes	vpc-c037d9b9 | DevopsJu...	339639447745
DevOpsJunction_CustomVPC_RT	rtb-047fd9d573711bf4a	subnet-01f664fecd0062f3e	-	Yes	vpc-0987dfc548599d6b0 |...	339639447745

Route Table: rtb-047fd9d573711bf4a

Summary | Routes | **Subnet Associations** | Edge Associations | Route Propagation | Tags

Edit subnet associations

1 to 1 of 1

Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet-01f664fecd0062f3e | DevopsJunction_Subnet	10.0.0.0/20	-

Security Group of both EC2 instances are set to allow SSH Connection from Global (anywhere) for testing purpose.

Description | **Inbound** | Outbound | Tags

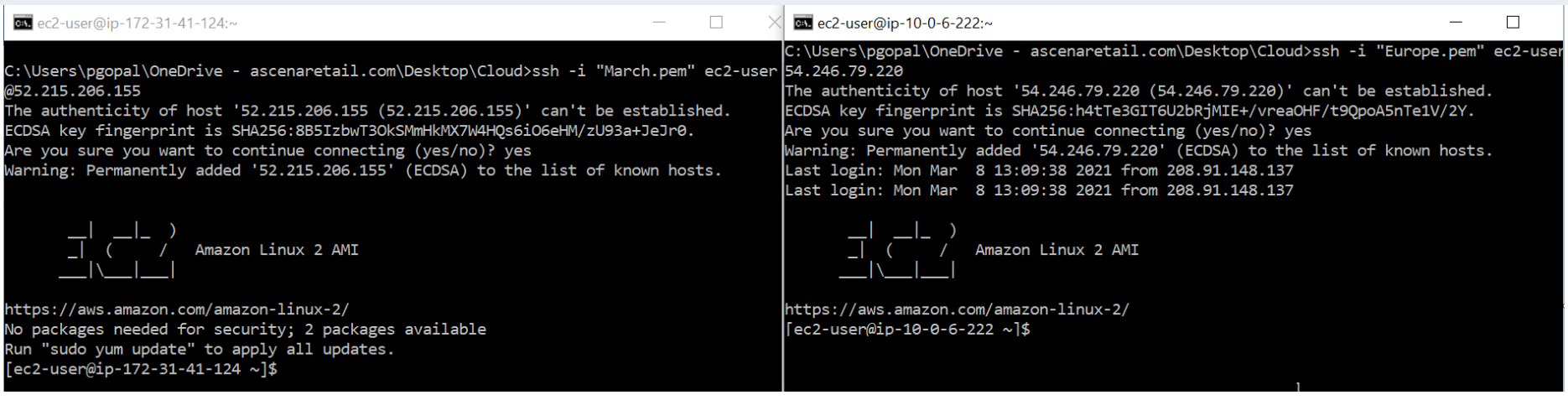
Edit

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	

Login to EC2 instances

Now login to both the EC2 instances using their Public IP using the command:

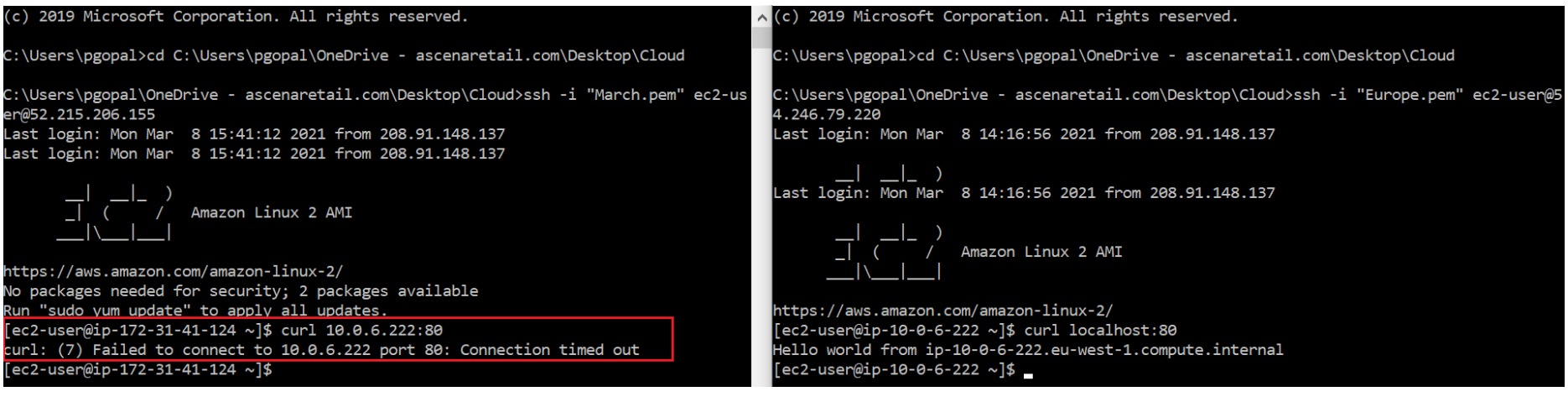
```
ssh -I "<ssh_key.pem>" ec2-user@<public_ip>
```



The instance on left is under Default_VPC(Instance A 172) and the one on the right is under Custom_VPC(Instance B 10.0*)*

To prove there is no connectivity between `Instance A` and `Instance B` ,

let's curl the URL which returns the metadata from `Instance B` – *(since we had added user-data and setup an apache httpd web server)*



Look at the preceding snapshot, See how `Instance A` times out to access `Instance B`.

Let's create VPC Peering to enable communication

To establish the connection, lets create VPC peering from both VPCs. On the left navigation panel under VPC -> Peering Connections:

-
  - VPC (Requester) = DevopsJunction_CustomVPC
  - VPC (Accepter) = DevopsJunction_DefaultVPC

Create Peering Connection

Peering connection name tag: DevopsJunction_VPC_Peering

Select a local VPC to peer with

VPC (Requester)*: vpc-0987dfc548599d6b0

CIDRs

CIDR	Status	Status Reason
10.0.0.0/16	associated	

Select another VPC to peer with

Account: My account / Another account

Region: This region (eu-west-1) / Another Region

VPC (Accepter)*: vpc-c037d9b9

CIDRs

CIDR	Status	Status Reason
172.31.0.0/16	associated	

Tags

Key (128 characters maximum)	Value (256 characters maximum)
Name	DevopsJunction_VPC_Peering

Add Tag 49 remaining (Up to 50 tags maximum)

Create a Peering Connection.

Create Peering Connection

Success

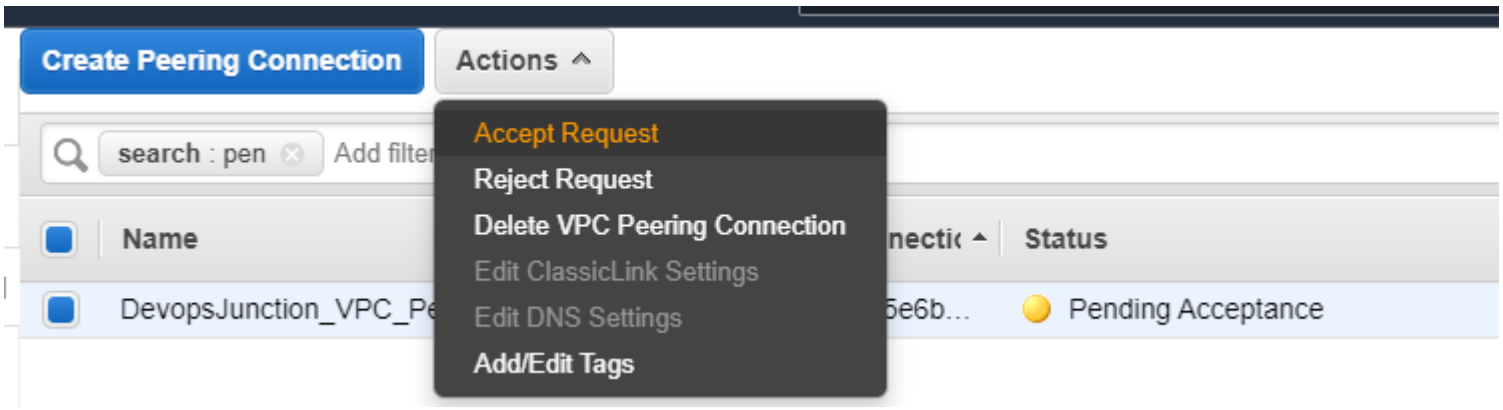
A VPC peering connection (pcx-02596be5e6b99d930) has been requested.

Requester VPC owner	339639447745 (This account)	Accepter VPC owner	339639447745 (This account)
Requester VPC ID	vpc-0987dfc548599d6b0	Accepter VPC ID	vpc-c037d9b9
Requester VPC Region	eu-west-1	Accepter VPC Region	eu-west-1
Requester VPC CIDRs	10.0.0.0/16	Accepter VPC CIDRs	-

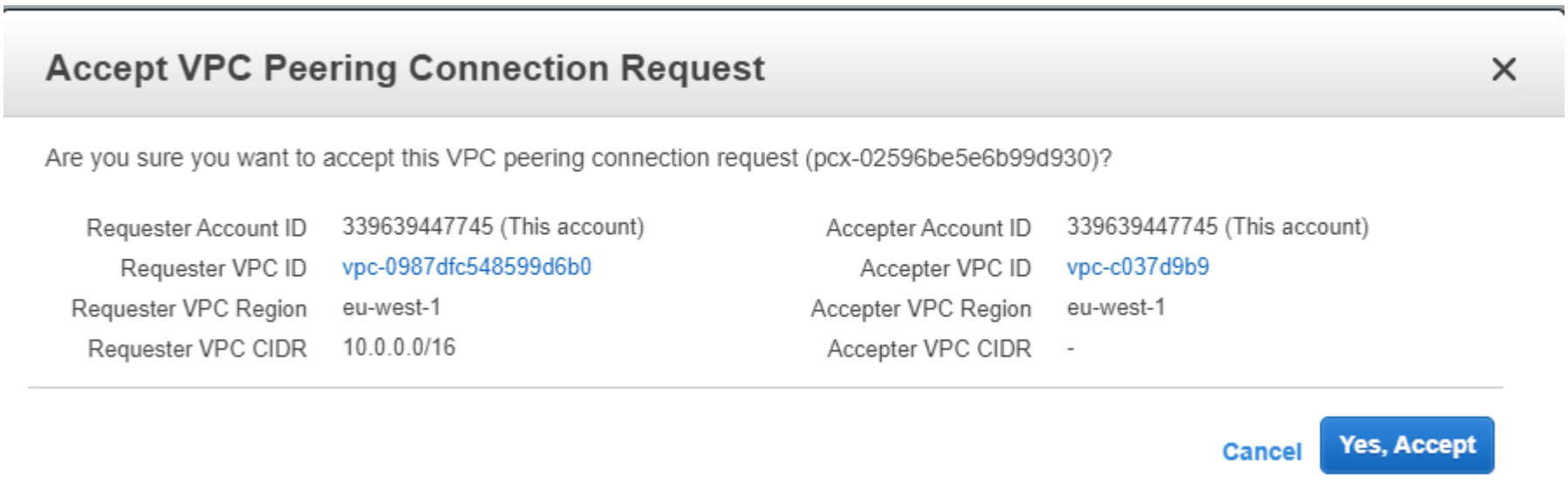
OK

Now you would see the status `Pending Acceptance` which means, Requestor has sent a request to the peer now target VPC needs to accept the request.

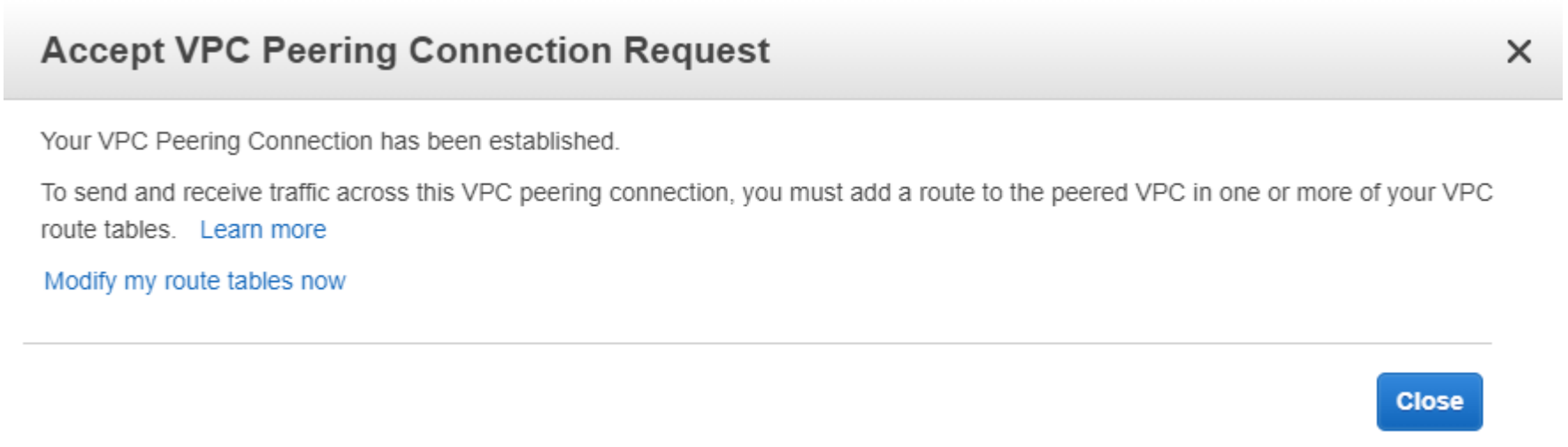
So drop down Actions -> Accept Request



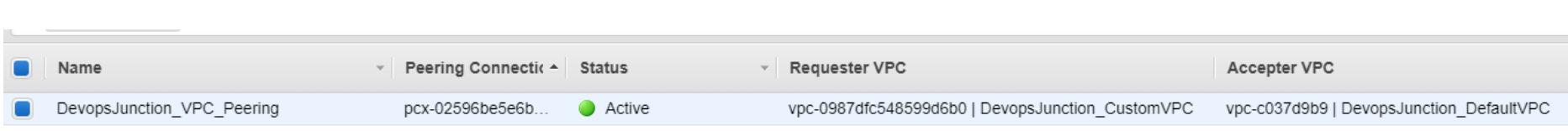
Yes Accept



Now we need to make entries in Route Tables



The status of VPC Peering turns `Active`

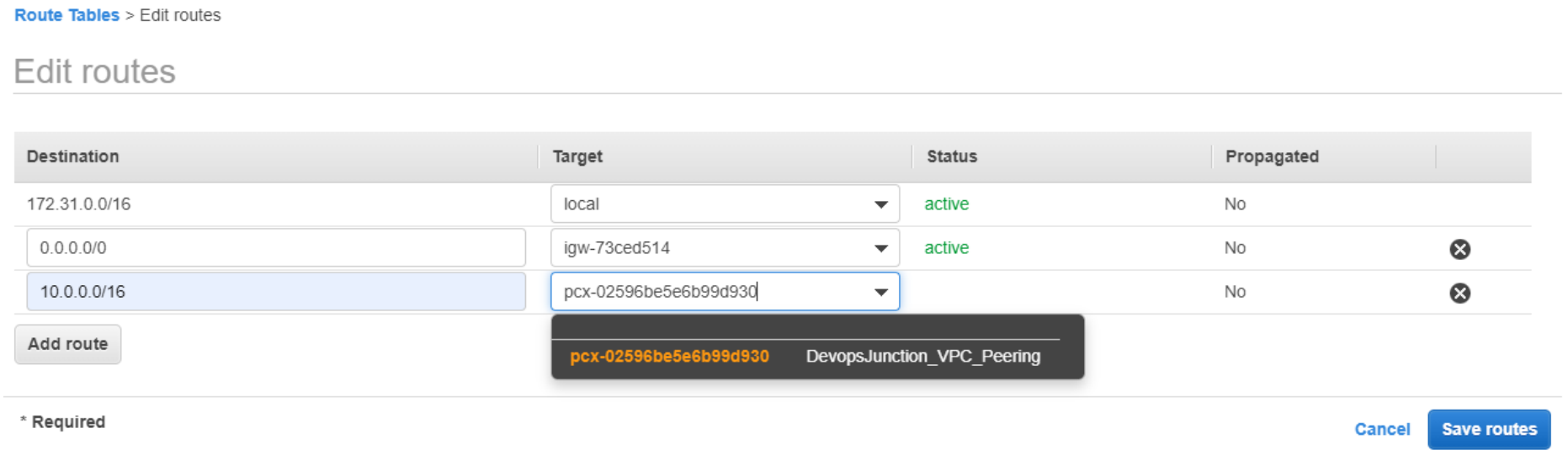


To edit Route Tables, copy the IPv4 CIDR ranges of both Default and Custom VPC, in my case it is

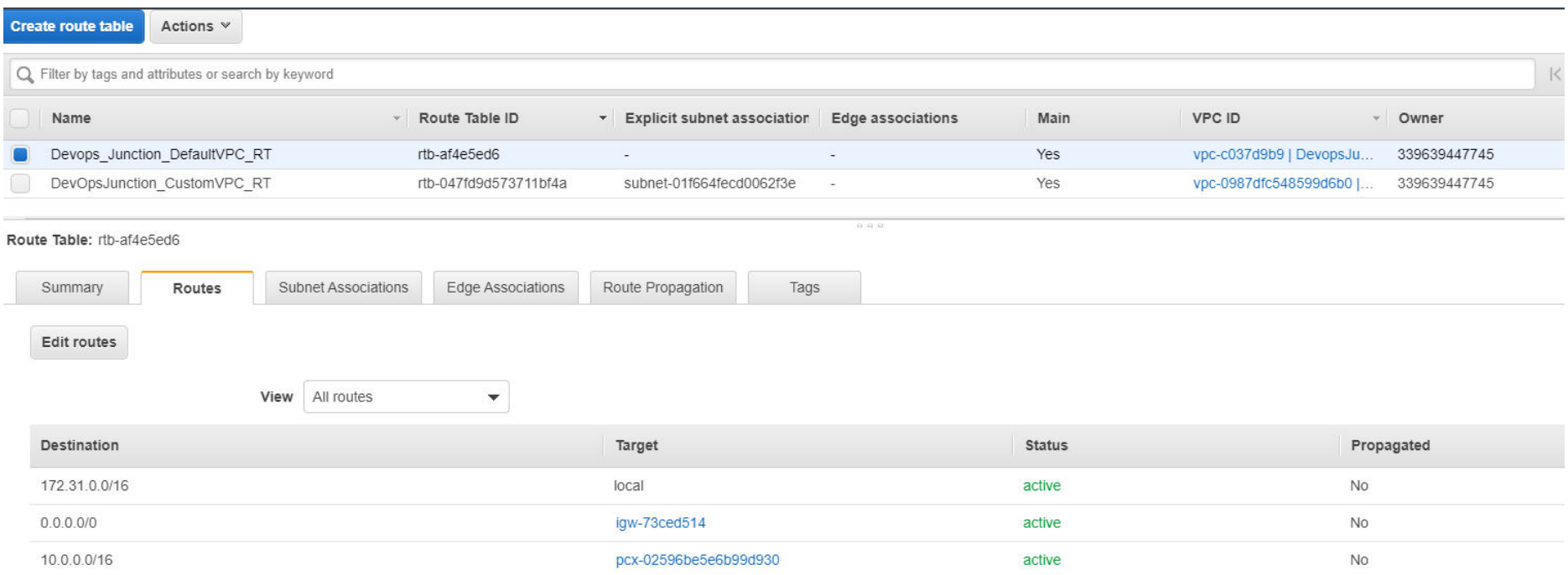
DevopsJunction_CustomVPC = 10.0.0.0/16

DevopsJunction_DefaultVPC = 172.31.0.0/16

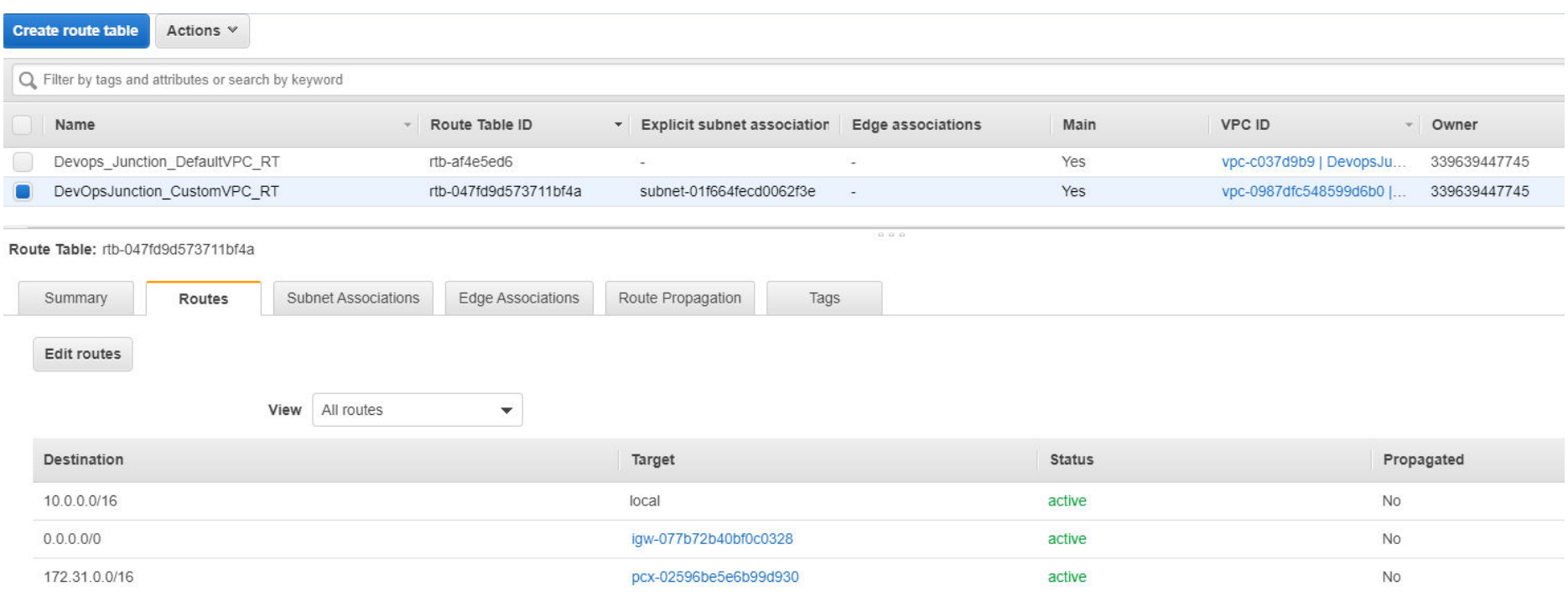
Now navigate to Route Tables, in `Default VPC RT(Route Table) -> Edit routes`



DefaultVPC Route Table would look as below



Do the same `CustomVPC` Route Table and it would look as below



Now try to check the connectivity

Select ec2-user@ip-172-31-41-124:~

```
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\pgopal>cd C:\Users\pgopal\OneDrive - ascenaretail.com\Desktop\Cloud

C:\Users\pgopal\OneDrive - ascenaretail.com\Desktop\Cloud>ssh -i "March.pem" ec2-us
er@52.215.206.155
Last login: Mon Mar  8 15:41:12 2021 from 208.91.148.137
Last login: Mon Mar  8 15:41:12 2021 from 208.91.148.137

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 2 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-41-124 ~]$ curl 10.0.6.222:80
curl: (7) Failed to connect to 10.0.6.222 port 80: Connection timed out
[ec2-user@ip-172-31-41-124 ~]$ curl 10.0.6.222:80
curl: (7) Failed to connect to 10.0.6.222 port 80: Connection timed out
[ec2-user@ip-172-31-41-124 ~]$ curl 10.0.6.222:80
curl: (7) Failed to connect to 10.0.6.222 port 80: Connection timed out
[ec2-user@ip-172-31-41-124 ~]$ hostname -I
172.31.41.124
[ec2-user@ip-172-31-41-124 ~]$
```

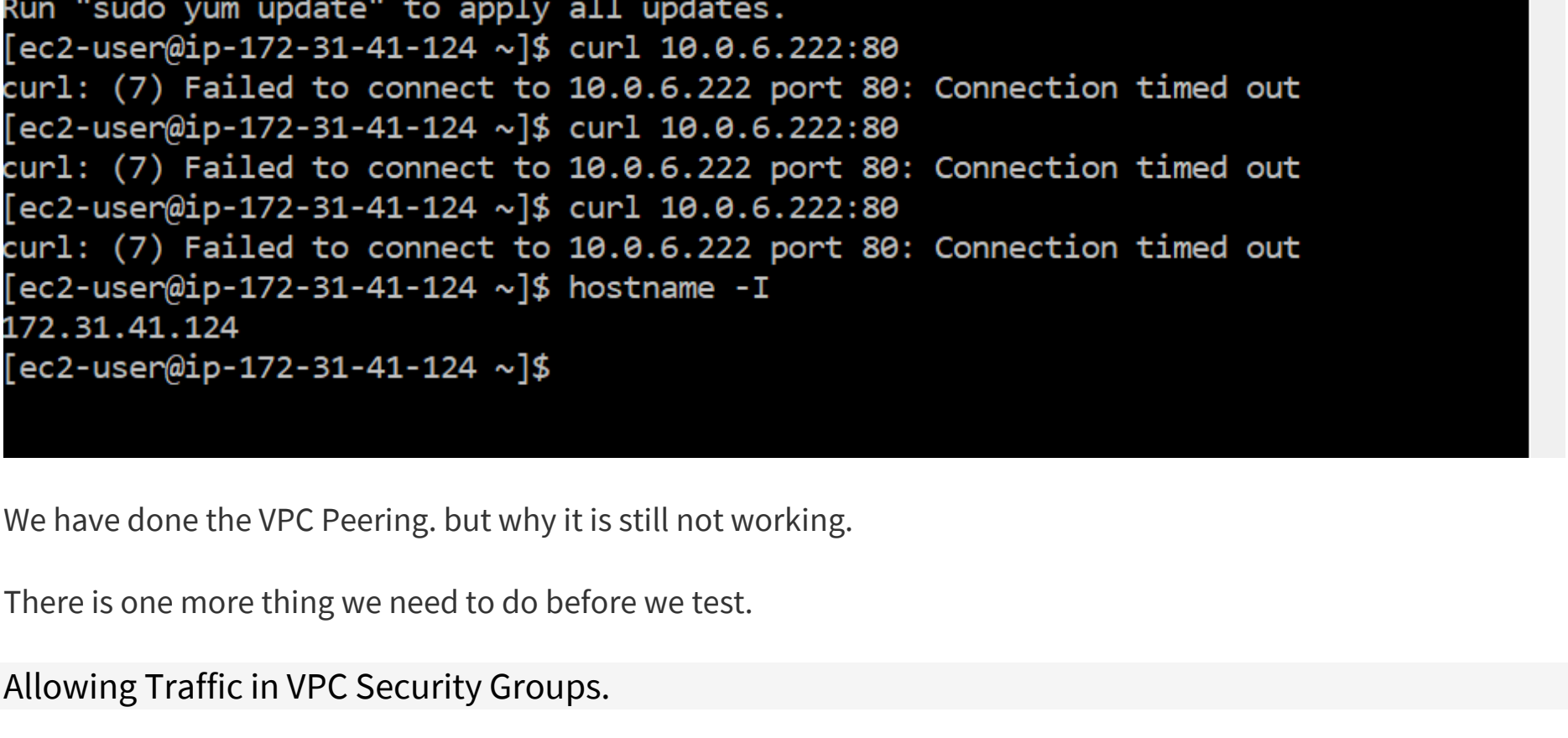
We have done the VPC Peering. but why it is still not working.

There is one more thing we need to do before we test.

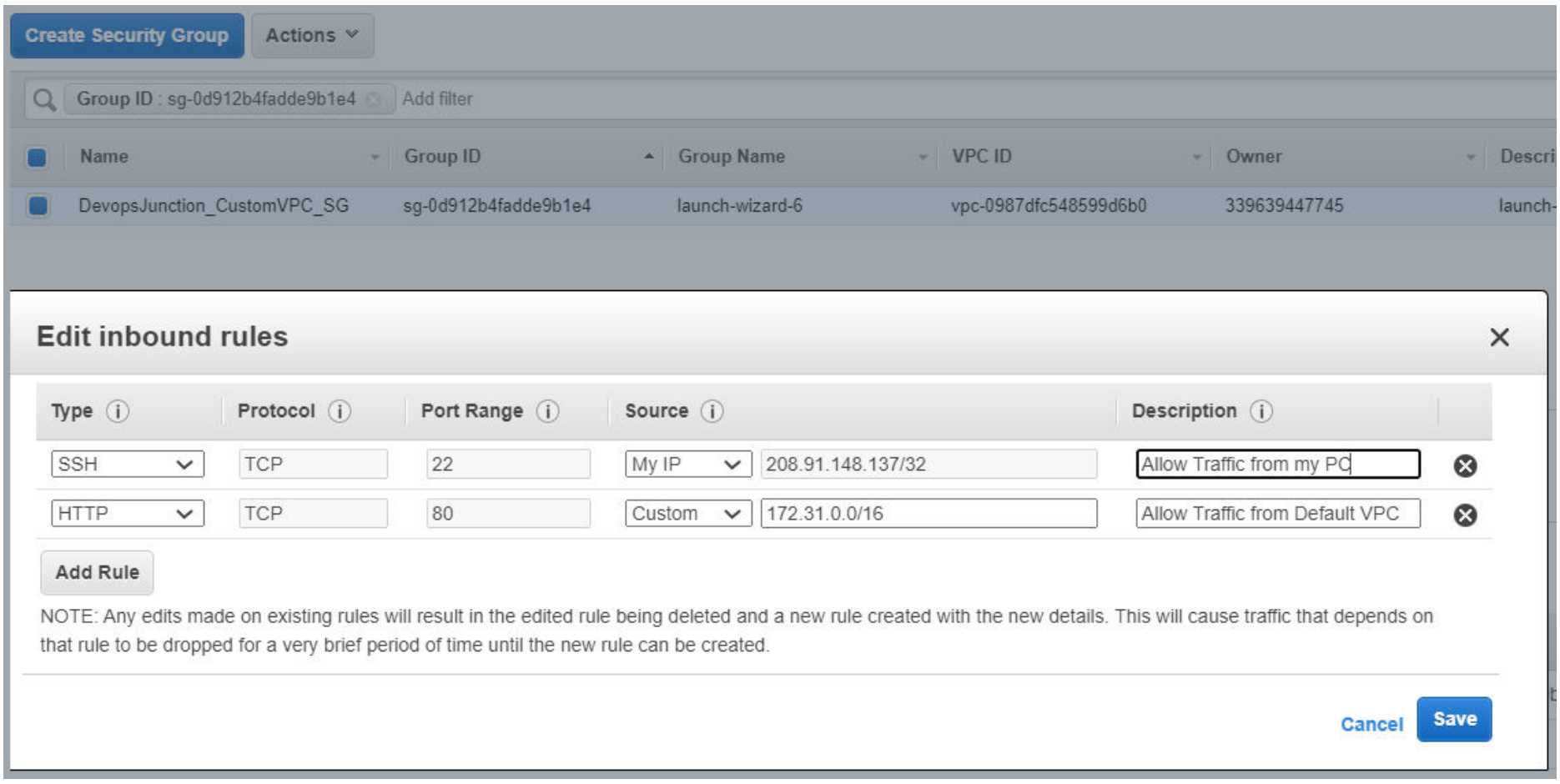
Allowing Traffic in VPC Security Groups.

Edit Security Group of Default and Custom VPC to allow traffic from each other

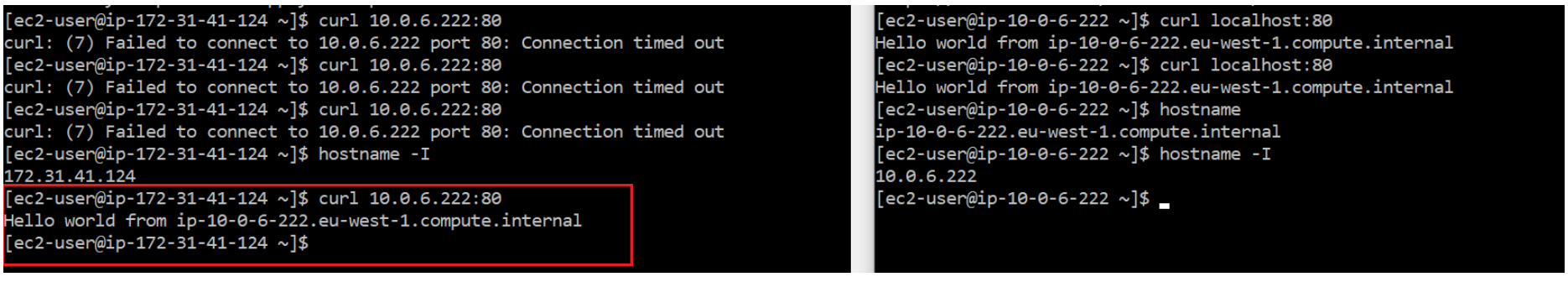
Default VPC Security Group looks like (to secure it better, route SSH from your PC only)



Custom VPC Security Group would look like



Now it works like a charm using the PRIVATE IP of the destination VPC !!!



NAT Gateway

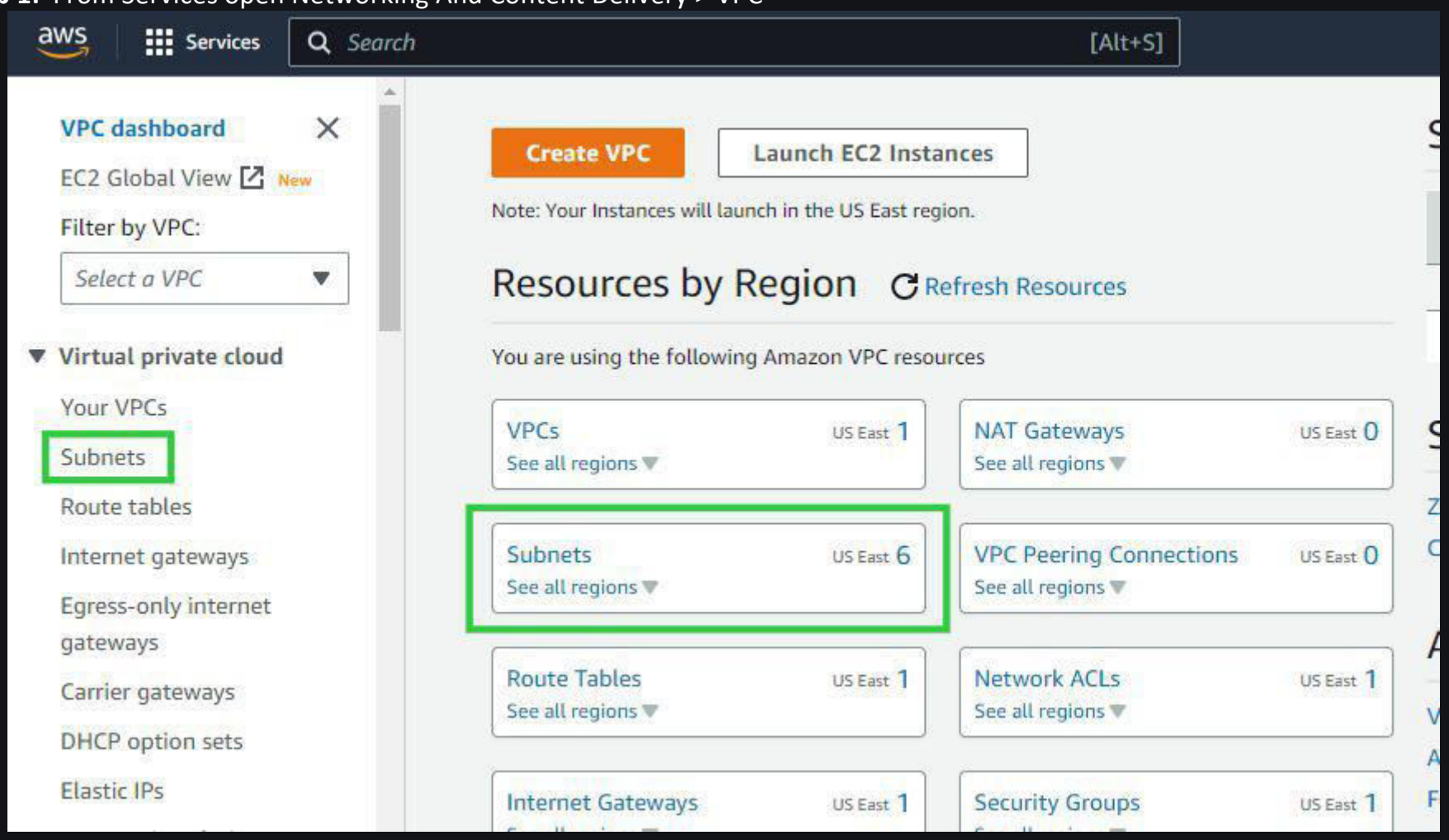
- **NAT Gateway:** NAT gateway is a service that allows private subnets in VPC to connect to outside networks and services preventing outside access.
- **VPC:** Virtual Private Cloud is an isolated cloud environment hosted within the public cloud.
- **Inbound Traffic:** Traffic generated outside networks towards internal services.
- **Outbound Traffic:** Traffic generated in inside networks accessing outside services.

Steps To Setup NAT Gateway for a Private Subnet in Amazon VPC

Scenario 1: Steps to setup public NAT Gateway in AWS

The public NAT Gateway is the one that has public IP associated with it. A Public NAT gateway must be allocated in a public subnet. Internet access is available with public NAT Gateway. Public NAT Gateway can be connected to a private subnet which allows the private subnet to access the internet restricting outside networks from accessing resources in a private network.

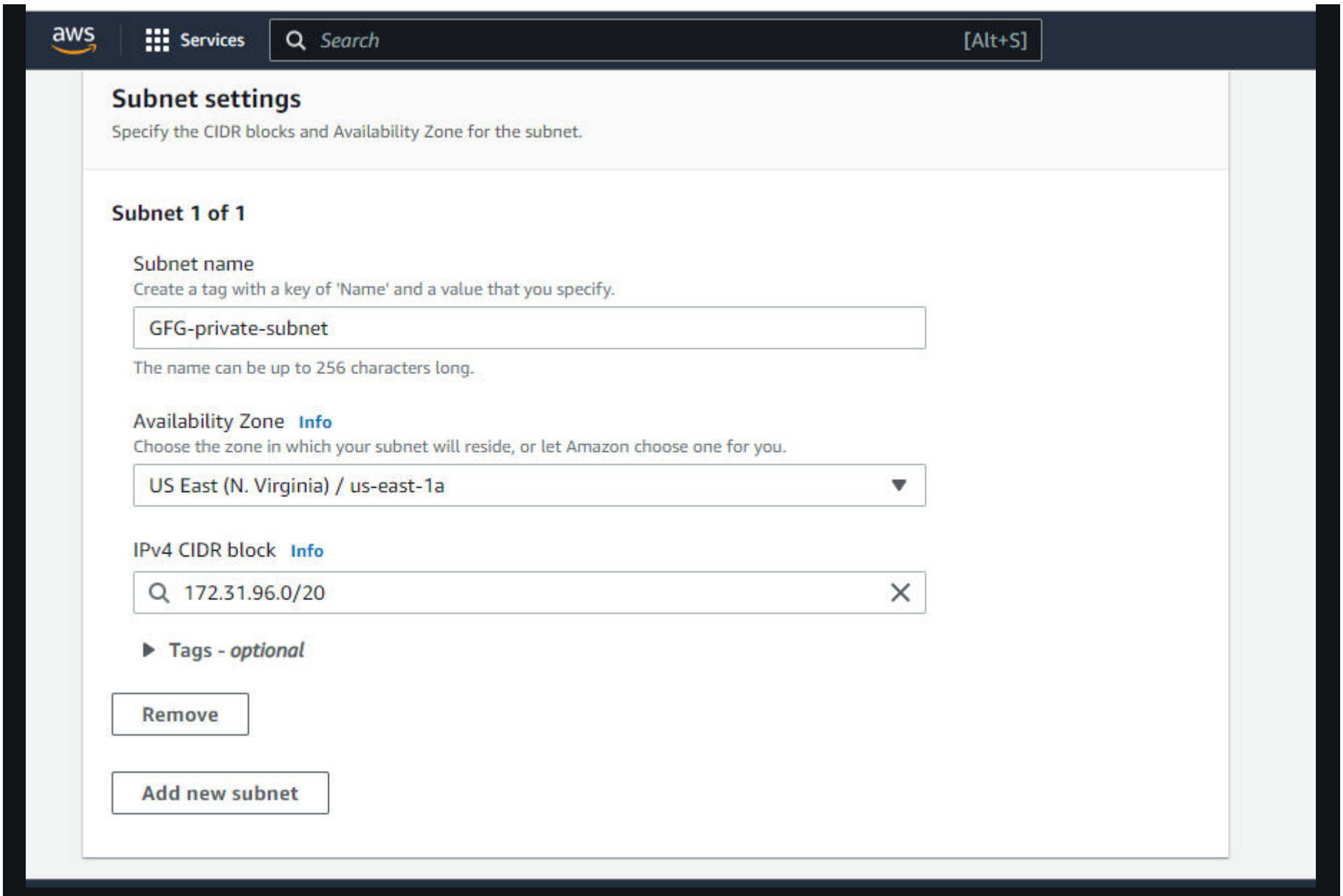
Step 1: From Services open Networking And Content Delivery > VPC



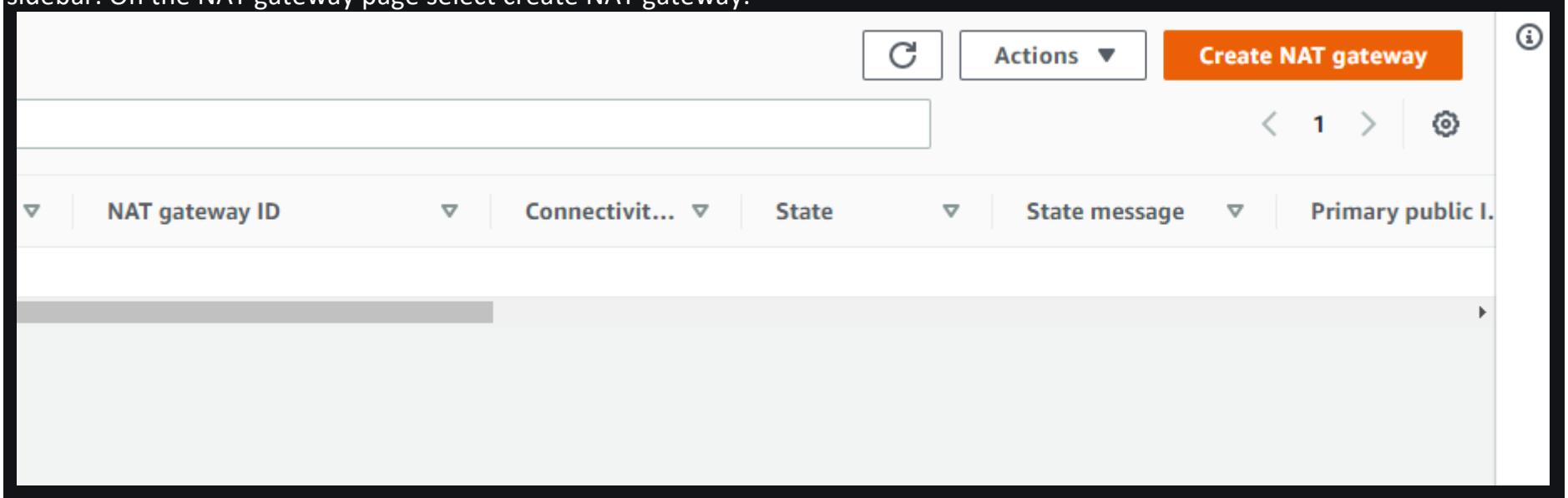
Step 2: On the VPC page open Subnets. In subnets click on create subnet to create a public and private subnet. fill in the following details on creating a subnet page. The difference between the private and public subnet is public subnet will have at least one route to the internet. route for the internet will default for all subnets. For the private subnet, we will create another routing table without an internet route.

After specifying the below options click on create a subnet (You can also specify tags if you want). Follow the same steps for the Public subnet.

Subnet Name	Specify any subnet name of your choice
Availability Zone	Select any available availability zone of your choice
IPv4 CIDR block	Specify your required IPv4 CIDR range for the subnet



Step 3: After creating subnets let's create our public NAT gateway. For creating a NAT gateway first select NAT gateways from the sidebar. On the NAT gateway page select create NAT gateway.



Step 4: On creating the NAT gateway page . We have to select Public as the connectivity type. Public NAT should have an elastic IP for internet access hence we have to allocate existing elastic IP or allocate new elastic IP. Specify details as below then click create (Again Tags are optional).

Name	Specify the Name of your choice Or leave blank
Subnet	select your public subnet from the options
Connectivity Type	Public
Elastic IP allocation ID	Click on Allocate Elastic IP

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.

GFG-public-NAT

The name can be up to 256 characters long.

Subnet
Select a subnet in which to create the NAT gateway.

subnet-06604c851afcf69de (GFG-public subnet)

Connectivity type
Select a connectivity type for the NAT gateway.

- (•) Public
- ( ) Private

Elastic IP allocation ID Info
Assign an Elastic IP address to the NAT gateway.

eipalloc-063eb0d9644a568e2 | Allocate Elastic IP

Step 5: After successful creation, the NAT gateway state will be pending. Once the state changes to available we can associate it with the private subnet.

Step 6: Now let's create a routing table that will route traffic from our private subnet to the internet through public NAT gateway. Select route tables from the sidebar. On the route tables page select create route table and specify the below options. Then click on create (Tags are Optional).

VPC > Route tables > Create route table

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, an connection.

Route table settings

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.

GFG-private-NAT-route

VPC
The VPC to use for this route table.

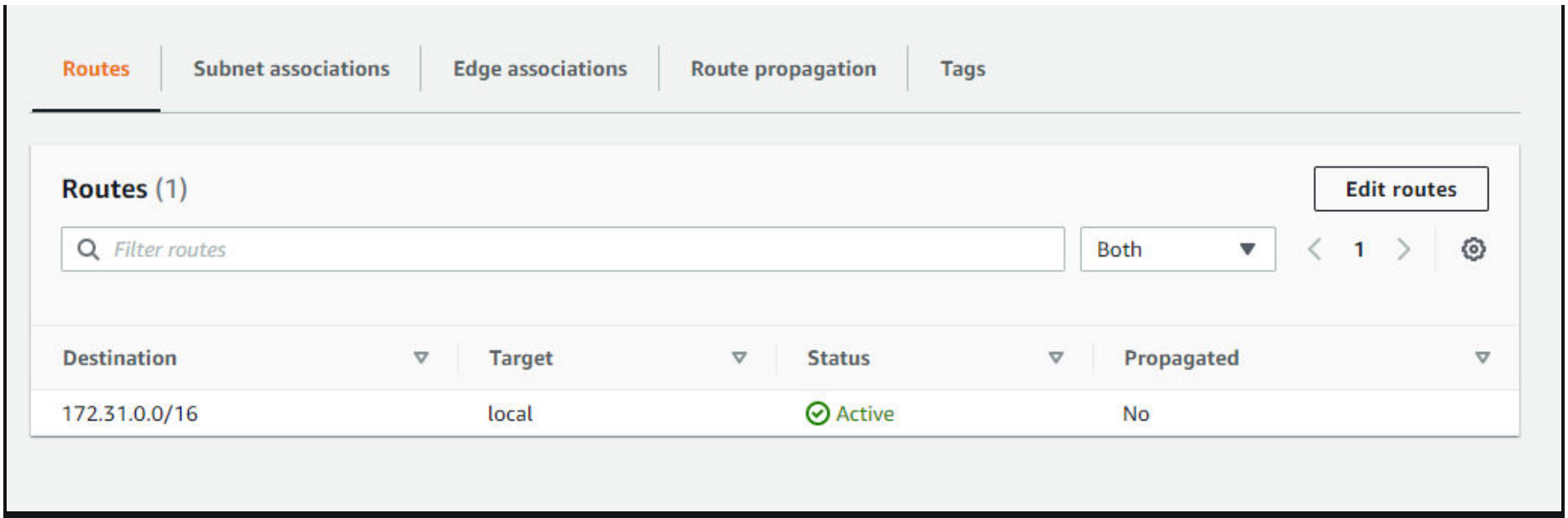
vpc-074cf1add87ff649f

Tags

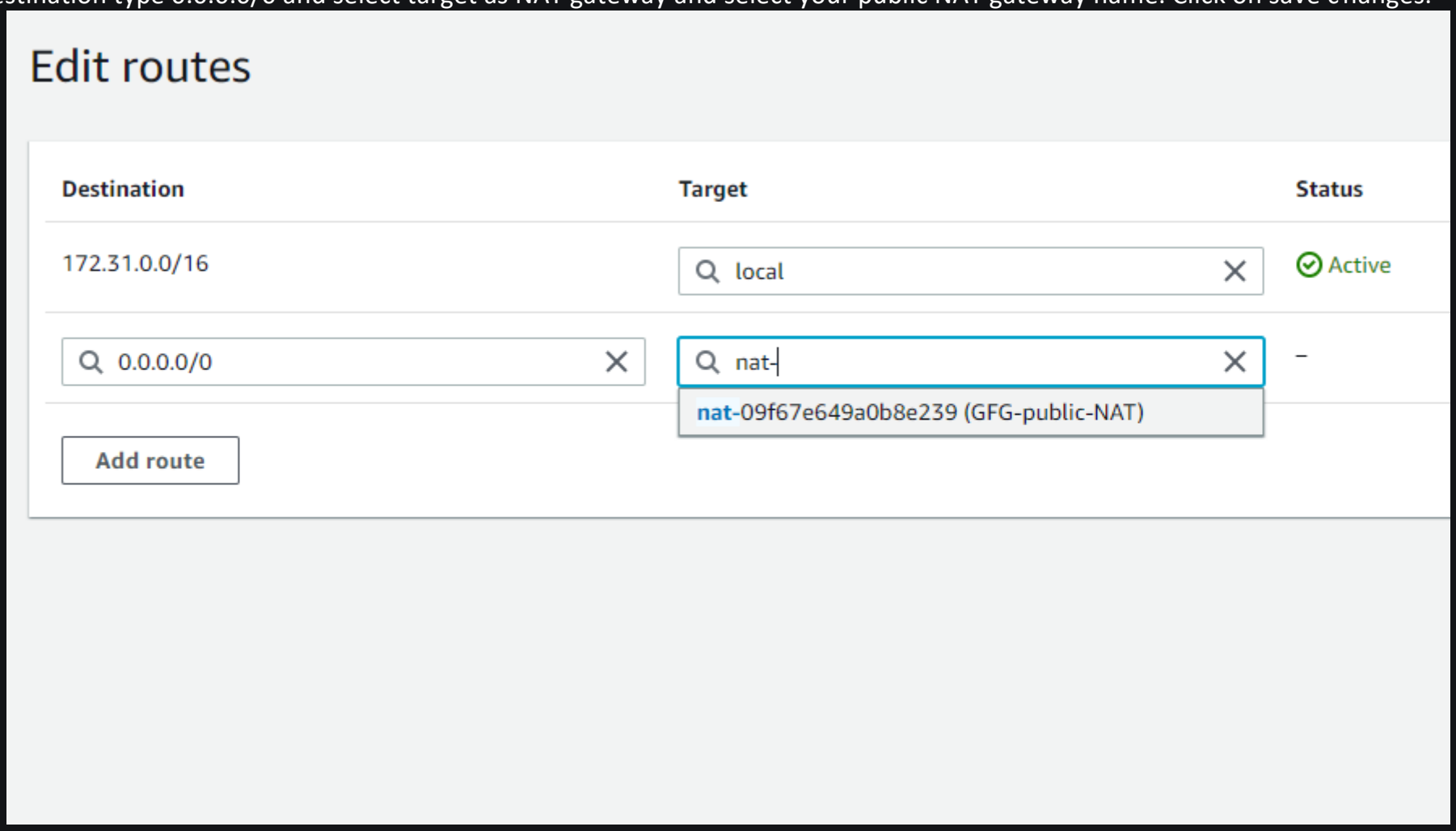
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags t your resources or track your AWS costs.

Key | Value - *optional*

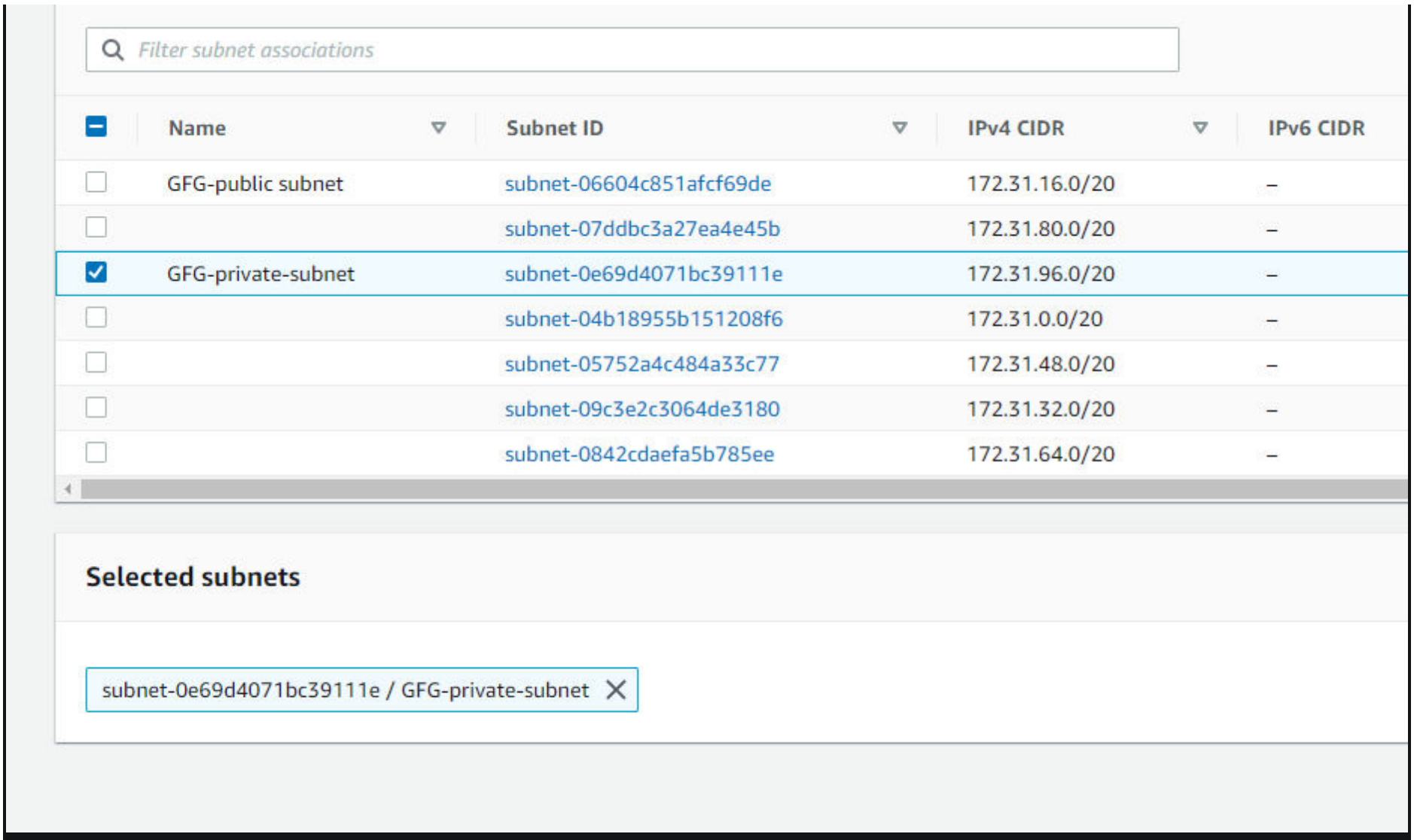
Step 7: Once the table is ready under table details under routes select edit routes.



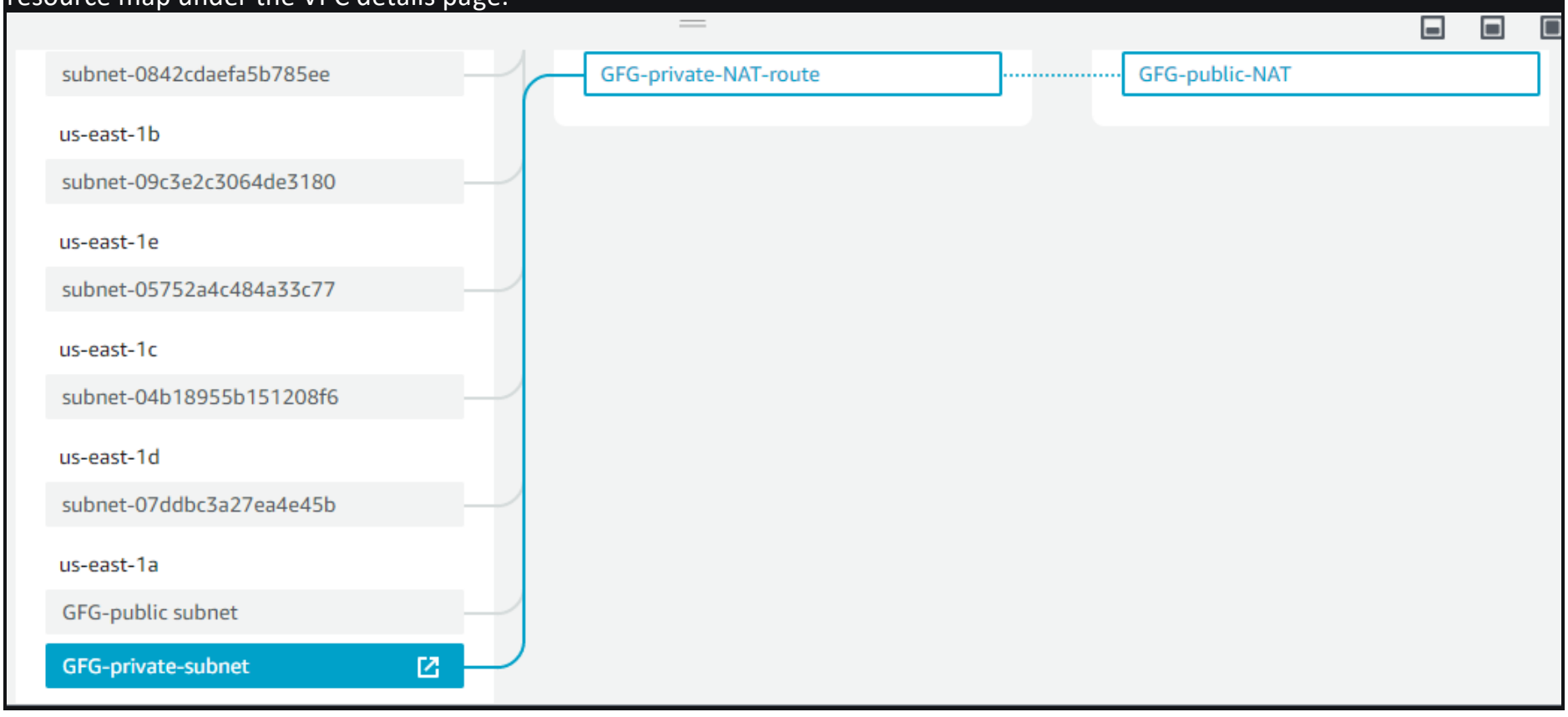
Step 8: Specify all traffic other than local forward to the public NAT gateway. i.e forward 0.0.0.0/0 to the NAT gateway. under destination type 0.0.0.0/0 and select target as NAT gateway and select your public NAT gateway name. Click on save changes.



Step 9: After saving changes select route tables and click on actions then edit subnet associations. Now select your private subnet from the list excluding all other subnets. Then click save associations.



Step 10: Once you save the associations the routing table will forward all traffic other than local to the public NAT gateway deployed in a public subnet. Through public NAT gateway resources in private subnets can access the internet. You can view the resource map under the VPC details page.



7.Elastic Load Balancer (ELB)

What is Elastic Load Balancing?

Elastic Load Balancing (ELB) is a load-balancing service for Amazon Web Services (AWS) deployments. ELB automatically distributes incoming application traffic and scales resources to meet traffic demands.

ELB helps an IT team adjust capacity according to incoming application and network traffic. Users enable ELB within a single availability zone or across multiple availability zones to maintain consistent application performance.

Historically, load balancing divides the amount of work that a computer has to do among multiple computers so that users, in general, get served faster. ELB offers enhanced features including:

- Detection of unhealthy Elastic Compute Cloud (EC2) instances.
- Spreading instances across healthy channels only.
- Flexible cipher support.
- Centralized management of Secure Sockets Layer (SSL) certificates.
- Optional public key authentication.
- Support for both IPv4 and IPv6.

High availability

The most well-known service that relies on ELB is Amazon's EC2, as ELB performs a health check to ensure an instance is still running before sending traffic to it. When an instance fails or is unhealthy, ELB routes traffic to the remaining healthy EC2 instances. If all EC2 instances in a particular availability zone are unhealthy, ELB can route traffic to other availability zones until the original instances restore to a healthy state.

A developer can integrate Amazon Route 53 and domain name system (DNS) failover to further boost application resiliency. Route 53 can route traffic to another healthy ELB and fail over across AWS regions.

Automatic scaling

A developer can use AWS' Auto Scaling feature to guarantee he or she has enough EC2 instances running behind an ELB. The developer sets Auto Scaling conditions, and when a condition is met, a new EC2 instance can spin up to meet the desired minimum. A developer can also set a condition to spin up new EC2 instances to reduce latency.

Security

ELB supports applications within an Amazon Virtual Private Cloud for stronger network security. An IT team can specify whether it wants an internet-facing or internal load balancer. The latter option enables a developer to route traffic through an ELB using private IP addresses. A developer could also route traffic between different tiers of an application by using multiple internet-facing and internal load balancers; this approach allows an IT team to use a security group along with private IP addresses while exposing only the web-facing tier and its public IP addresses.

In addition to certificate management, ELB allows SSL/Transport Layer Security (TLS) decryption.

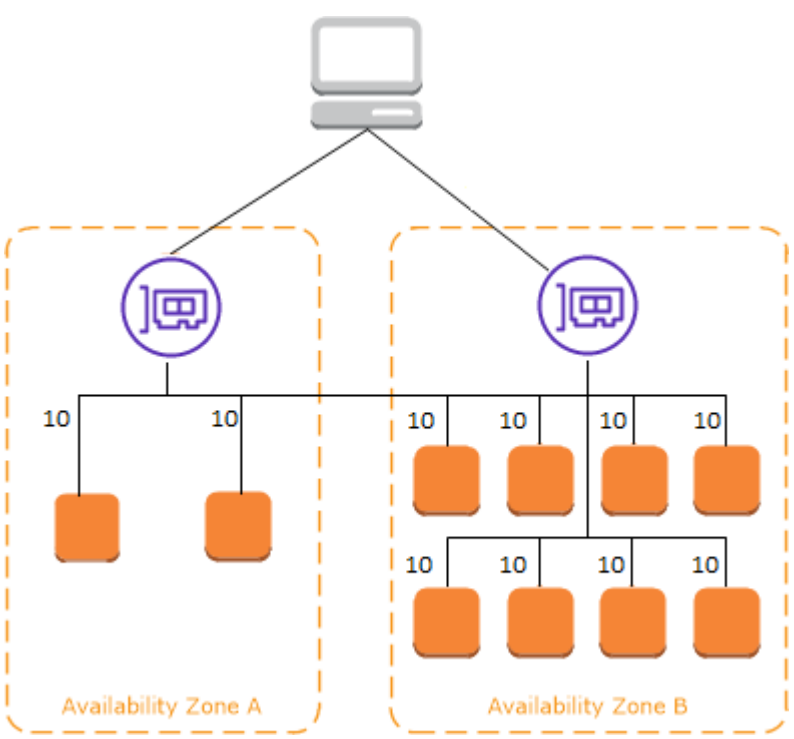
How Elastic Load Balancing Works

Cross-zone load balancing

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is disabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone.

The following diagrams demonstrate the effect of cross-zone load balancing with round robin as the default routing algorithm. There are two enabled Availability Zones, with two targets in Availability Zone A and eight targets in Availability Zone B. Clients send requests, and Amazon Route 53 responds to each request with the IP address of one of the load balancer nodes. Based on the round robin routing algorithm, traffic is distributed such that each load balancer node receives 50% of the traffic from the clients. Each load balancer node distributes its share of the traffic across the registered targets in its scope.

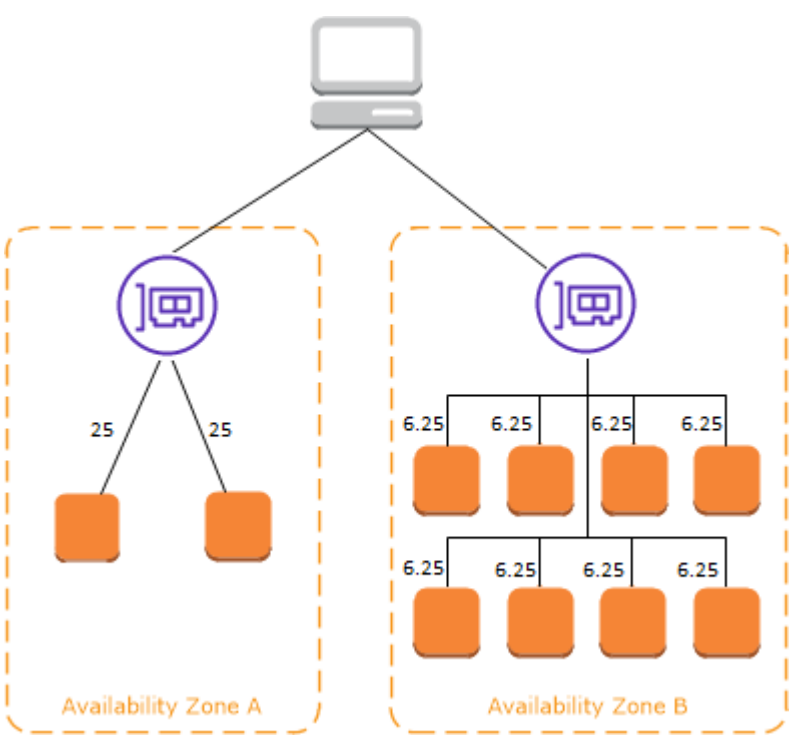
If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



If cross-zone load balancing is disabled:

- Each of the two targets in Availability Zone A receives 25% of the traffic.
- Each of the eight targets in Availability Zone B receives 6.25% of the traffic.

This is because each load balancer node can route its 50% of the client traffic only to targets in its Availability Zone.



With Application Load Balancers, cross-zone load balancing is always enabled at the load balancer level. At the target group level, cross-zone load balancing can be disabled. For more information, see Turn off cross-zone load balancing in the *User Guide for Application Load Balancers*.

With Network Load Balancers and Gateway Load Balancers, cross-zone load balancing is disabled by default. After you create the load balancer, you can enable or disable cross-zone load balancing at any time.

When you create a Classic Load Balancer, the default for cross-zone load balancing depends on how you create the load balancer. With the API or CLI, cross-zone load balancing is disabled by default. With the AWS Management Console, the option to enable cross-zone load balancing is selected by default. After you create a Classic Load Balancer, you can enable or disable cross-zone load balancing at any time. For more information, see Enable cross-zone load balancing in the *User Guide for Classic Load Balancers*.

Zonal shift

Zonal shift is a capability in Amazon Route 53 Application Recovery Controller (Route 53 ARC). With zonal shift, you can shift a load balancer resource away from an impaired Availability Zone with a single action. This way, you can continue operating from other healthy Availability Zones in an AWS Region.

When you start a zonal shift, your load balancer stops sending traffic for the resource to the affected Availability Zone. Route 53 ARC creates the zonal shift immediately. However, it can take a short time, typically up to a few minutes, to complete existing, in-progress connections in the affected Availability Zone. For more information, see How a zonal shift works: health checks and zonal IP addresses in the *Amazon Route 53 Application Recovery Controller Developer Guide*.

Zonal shifts are only supported on Application Load Balancers and Network Load Balancers with cross-zone load balancing turned off. If you turn on cross-zone load balancing, you can't start a zonal shift. For more information, see Resources supported for zonal shifts in the *Amazon Route 53 Application Recovery Controller Developer Guide*.

Before you use a zonal shift, review the following:

- Cross-zone load balancing isn't supported with zonal shifts. You must turn off cross-zone load balancing to use this capability.
- Zonal shift isn't supported when you use an Application Load Balancer as an accelerator endpoint in AWS Global Accelerator.
- You can start a zonal shift for a specific load balancer only for a single Availability Zone. You can't start a zonal shift for multiple Availability Zones.
- AWS proactively removes zonal load balancer IP addresses from DNS when multiple infrastructure issues impact services. Always check current Availability Zone capacity before you start a zonal shift. If your load balancers have cross-zone load balancing turned off and you use a zonal shift to remove a zonal load balancer IP address, the Availability Zone affected by the zonal shift also loses target capacity.
- When an Application Load Balancer is a target of a Network Load Balancer, always start the zonal shift from the Network Load Balancer. If you start a zonal shift from the Application Load Balancer, the Network Load Balancer doesn't recognize the shift and continues to send traffic to the Application Load Balancer.

For more guidance and information, see Best practices with Route 53 ARC zonal shifts in the *Amazon Route 53 Application Recovery Controller Developer Guide*.

Request routing

Before a client sends a request to your load balancer, it resolves the load balancer's domain name using a Domain Name System (DNS) server. The DNS entry is controlled by Amazon, because your load balancers are in the `amazonaws.com` domain. The Amazon DNS servers return one or more IP addresses to the client. These are the IP addresses of the load balancer nodes for your load balancer. With Network Load Balancers, Elastic Load Balancing creates a network interface for each Availability Zone that you enable, and uses it to get a static IP address. You can optionally associate one Elastic IP address with each network interface when you create the Network Load Balancer.

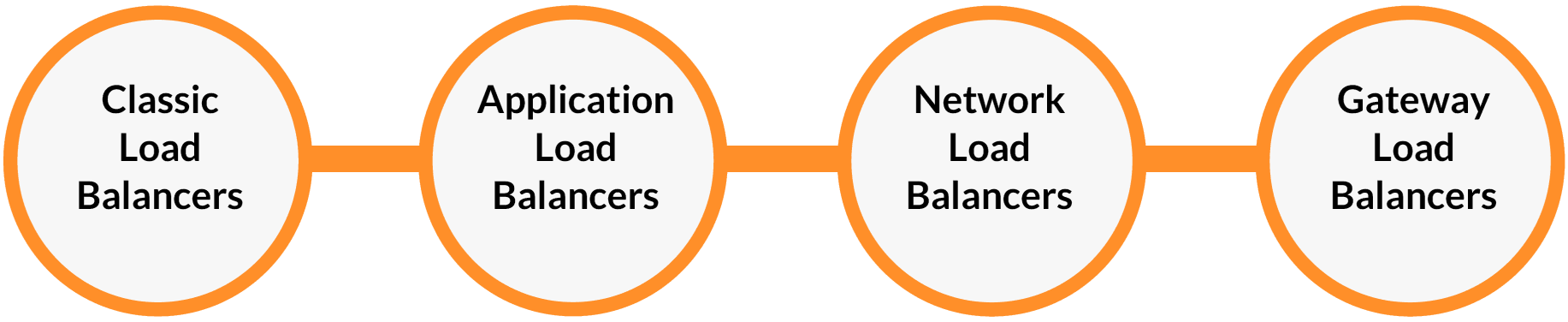
As traffic to your application changes over time, Elastic Load Balancing scales your load balancer and updates the DNS entry. The DNS entry also specifies the time-to-live (TTL) of 60 seconds. This helps ensure that the IP addresses can be remapped quickly in response to changing traffic.

The client determines which IP address to use to send requests to the load balancer. The load balancer node that receives the request selects a healthy registered target and sends the request to the target using its private IP address.

Types of ELB

4 Types of Load Balancers in AWS

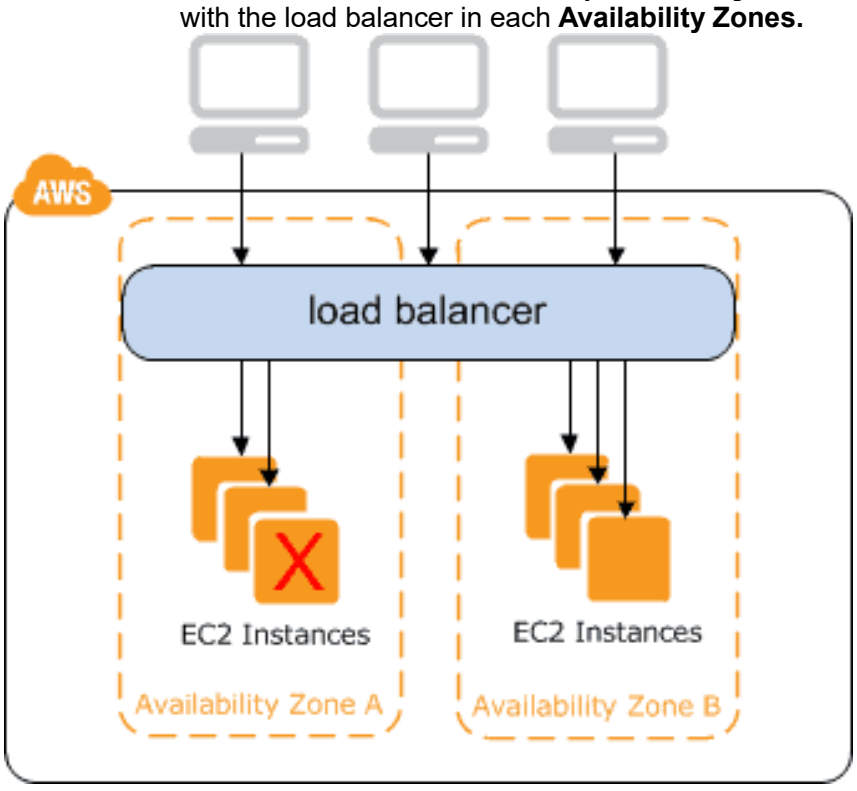
- **Classic Load Balancer**
- **Application Load Balancer**
- **Network Load Balancer**
- **Gateway Load Balancer**



Also Check Our blog post on **AWS Security**.

Classic Load Balancer

- The Load Balancer which balances the traffic across multiple instances in multiple availability zones is called a **Classic Load Balancer**.
- It supports both EC2 Classic EC2-VPC and Increases the availability of your application by sending traffic to healthy Instances.
- Supports HTTP, HTTPS, TCP, and SSL listeners and supports sticky sessions using application-generated cookies.
- To make sure that the instances you have registered can handle the demand Keep roughly the same number of instances registered with the load balancer in each **Availability Zones.**



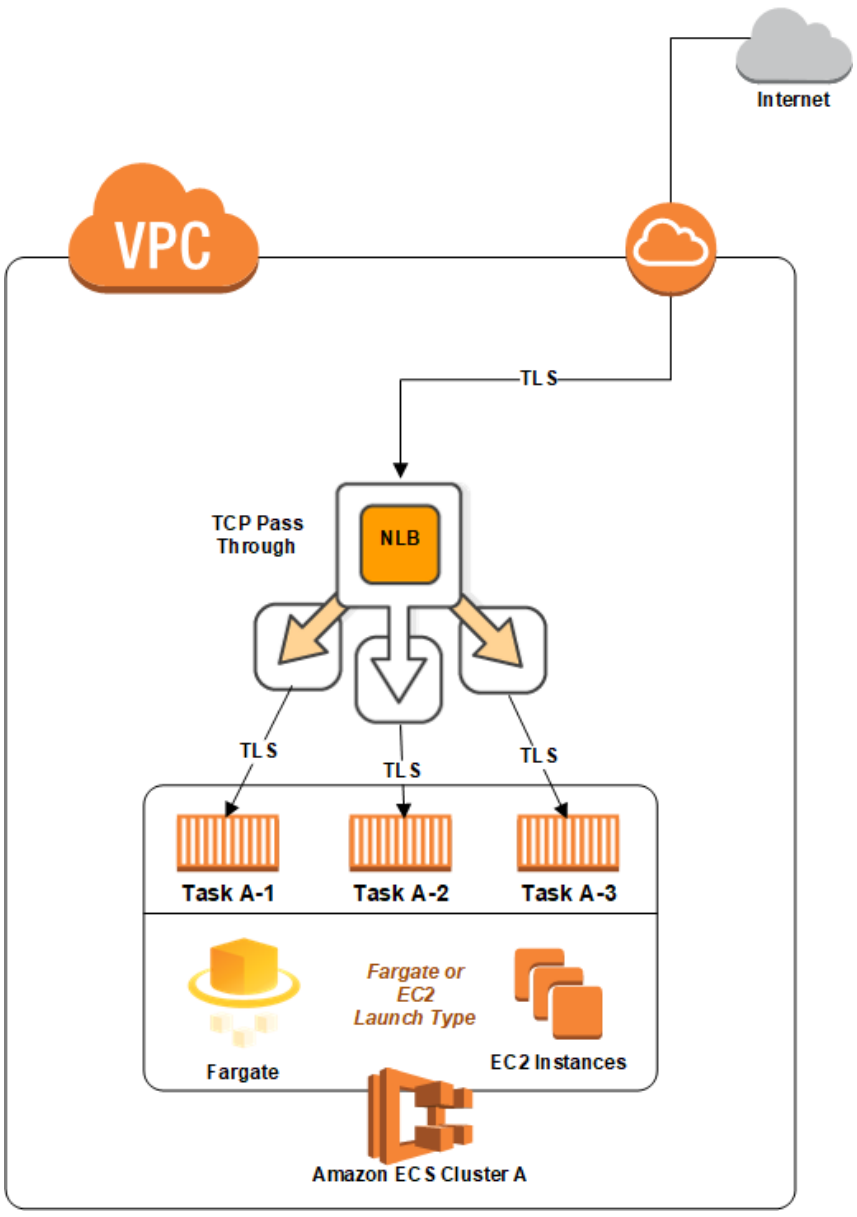
Limitations:

Regional Limit	
LB per region	20
Load Balancer Components Limit	
Listeners	50
Security groups	5
Registered instances	1000
Subnets per Availability Zone	1

Also read AWS Identity And Access Management.

Network Load balancer

- Network Load Balancer handles sudden and violates traffic across the EC2 Instances in order to avoid any latency.
- Connection baseload Balancing and it supports TCP protocol.
- Ability to handle volatile workloads and scale to millions of requests per second.
- Support for static IP addresses for the load balancer. or assign one Elastic IP address per subnet enabled for the load balancer.
- Cross-zone load balancing is disabled by default
- The source IP addresses of the clients are maintained and made available to your apps when you designate targets using an instance ID. The source IP addresses are the private IP addresses of the load balancer nodes if targets are specified by IP address.
- Network Load Balancers support connections from clients over inter-region VPC peering. AWS managed VPN and third-party VPN solutions.



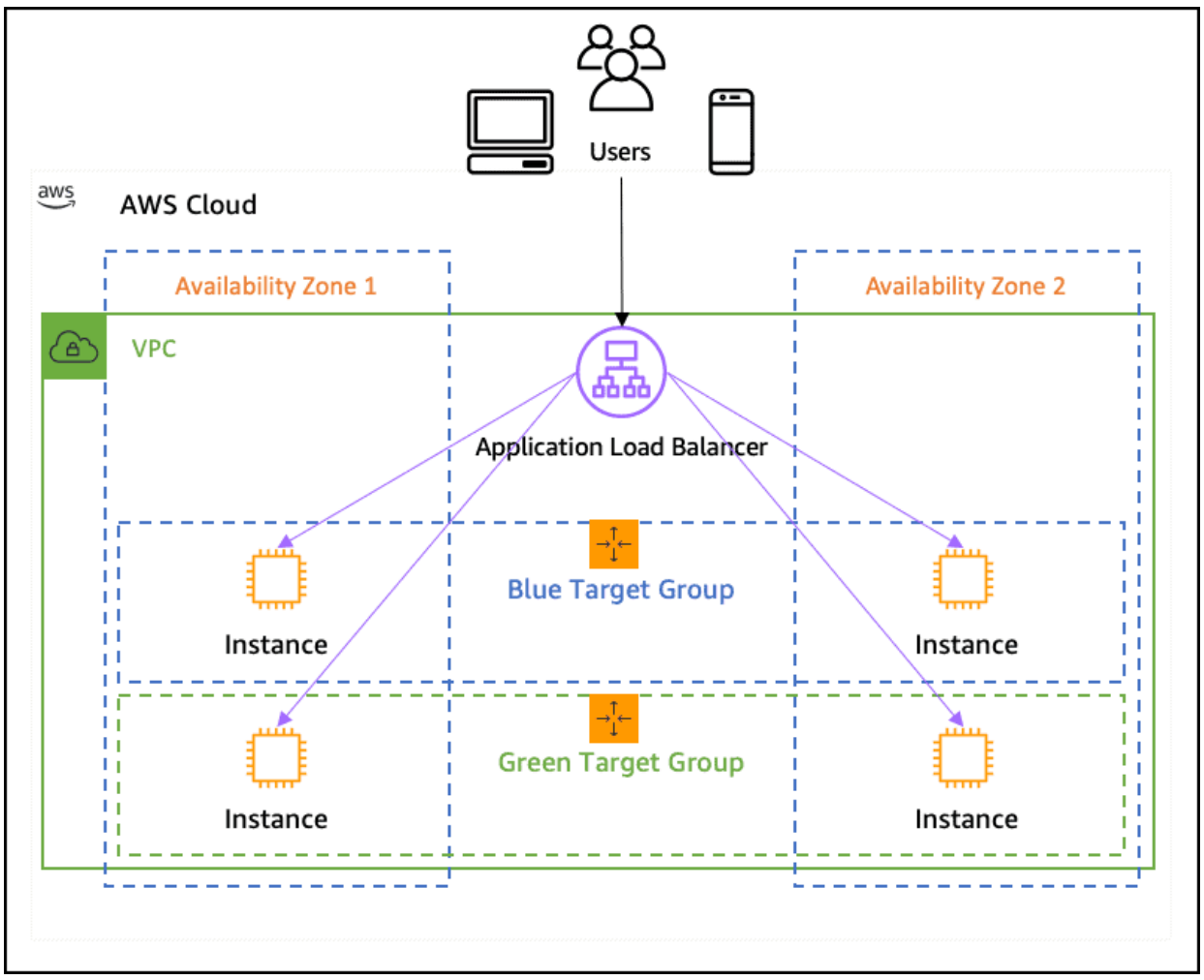
Also Read AWS Cloud Certification.
Limitations:

Regional Limit per Region	
Number of Network LB	20
Target groups	3000
Components Limit per LB	
Listeners	50
Targets per Availability Zone With Cross-zone load balancing disabled	200
Targets With Cross-zone load balancing enabled	200
Subnets per Availability Zone	1

Read More: AWS Serverless Application Model.

Application Load balancer

- The Load Balancer that distributes the traffic to appropriate target groups on the basis of content is called Application Load Balancer.
- New feature-rich, layer 7 loads balancing platform.
- Supports web sockets, HTTP, HTTPS, microservices, and container-based applications, including deep integration with EC2 container service.
- Support for path-based and host-based routing. Also, provide routing requests to multiple applications on a single EC2 instance.
- Cross-Zone load balancing is always enabled and you can also specify Lambda functions are targeted to serve HTTP(S) requests.
- Supports load balancer-generated cookies only for sticky sessions.



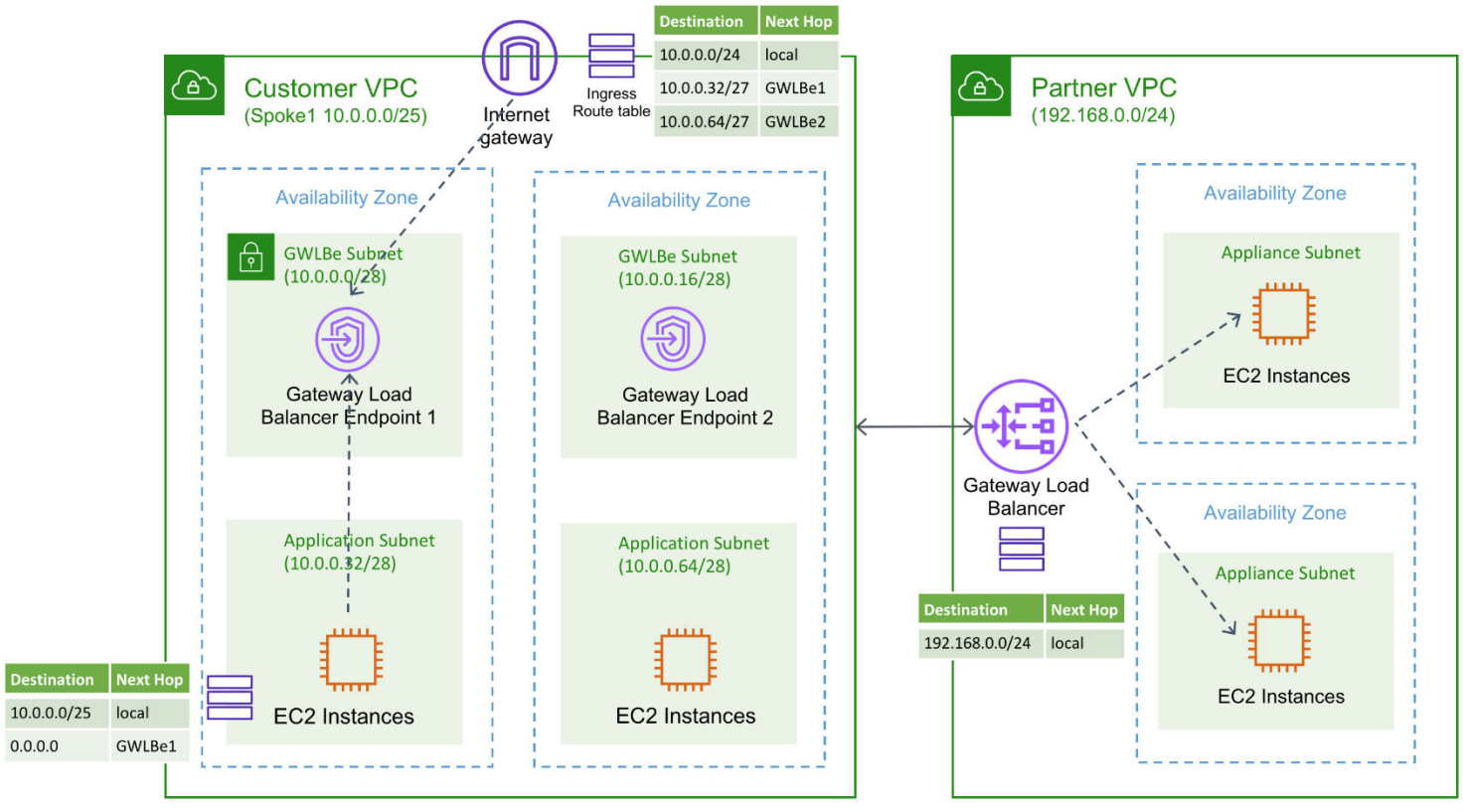
Also Read: How to Configure MFA in AWS.
Limitations:

Regional Limit	
LB Per Region	20
Target groups/Region	3000
Load Balancer Components Limit	
Listeners/load balancer	50
Targets /load balancer	1000
Subnets /Availability Zone per load balancer	1
Security groups / load balancer	5
Rules(not counting default rules)	100
Certificates (not counting default certificates)	25
Number of times a target can be registered	100

Also Check Our previous blog post on **Blue-Green Deployment**.

Gateway Load Balancer

- It makes it simple to scale, install, and manage yaour third-party virtual appliances.
- Provide you with one gateway for distributing traffic across multiple virtual appliances, while scaling them up, or down, based on demand.
- It improves availability and removes potential points of failure from your network.
- Users can find, test, and buy virtual appliances from third-party vendors directly in AWS Marketplace.
- This integrated experience streamlines the deployment process, so users can see value from your virtual appliances more quickly—whether you want to work with the same vendors you do today, or try something new.



Limitations:

Regional Limit	
LB Per Region	20
Load Balancer Components Limit	
Gateway Load Balancers per VPC	**10**
Target groups with GENEVE protocol	**100**
Targets per Availability Zone per target group with GENEVE protocol	**300**

Target Groups

A network load balancer distributes the load across cloud resources that are combined into a *target group*.

A *target* is defined by two parameters: the subnet ID and the internal IP address of the resource. Targets within a single group must be located in the same cloud network. All targets must be connected to the same subnet within a single availability zone. The maximum number of resources in a target group is 254.

The targets must receive traffic on the same port as the one specified in the listener configuration.

Attached target group

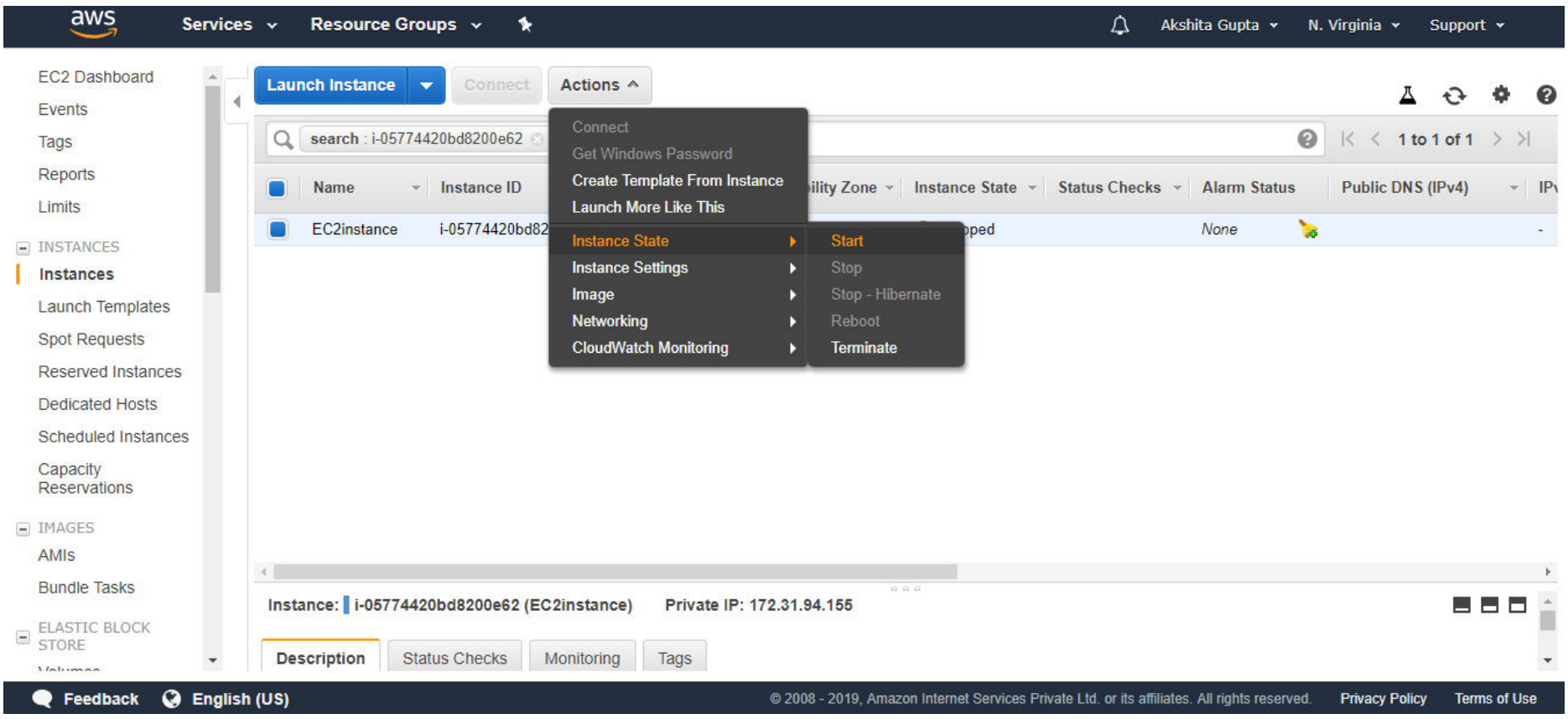
An attached target group is a group of targets that is attached to a network load balancer. You can attach a target group to multiple load balancers. However, you cannot attach a target group to ports with the same number on different load balancers. For example, if a group is attached to a load balancer on port 8080, you should attach it to another load balancer on port 8081.

Once a target group is attached, the load balancer will start performing target health checks and distributing the load across the targets in the group.

If a target group is used in at least one load balancer, you cannot delete it. First, you need to delete it from all load balancers.

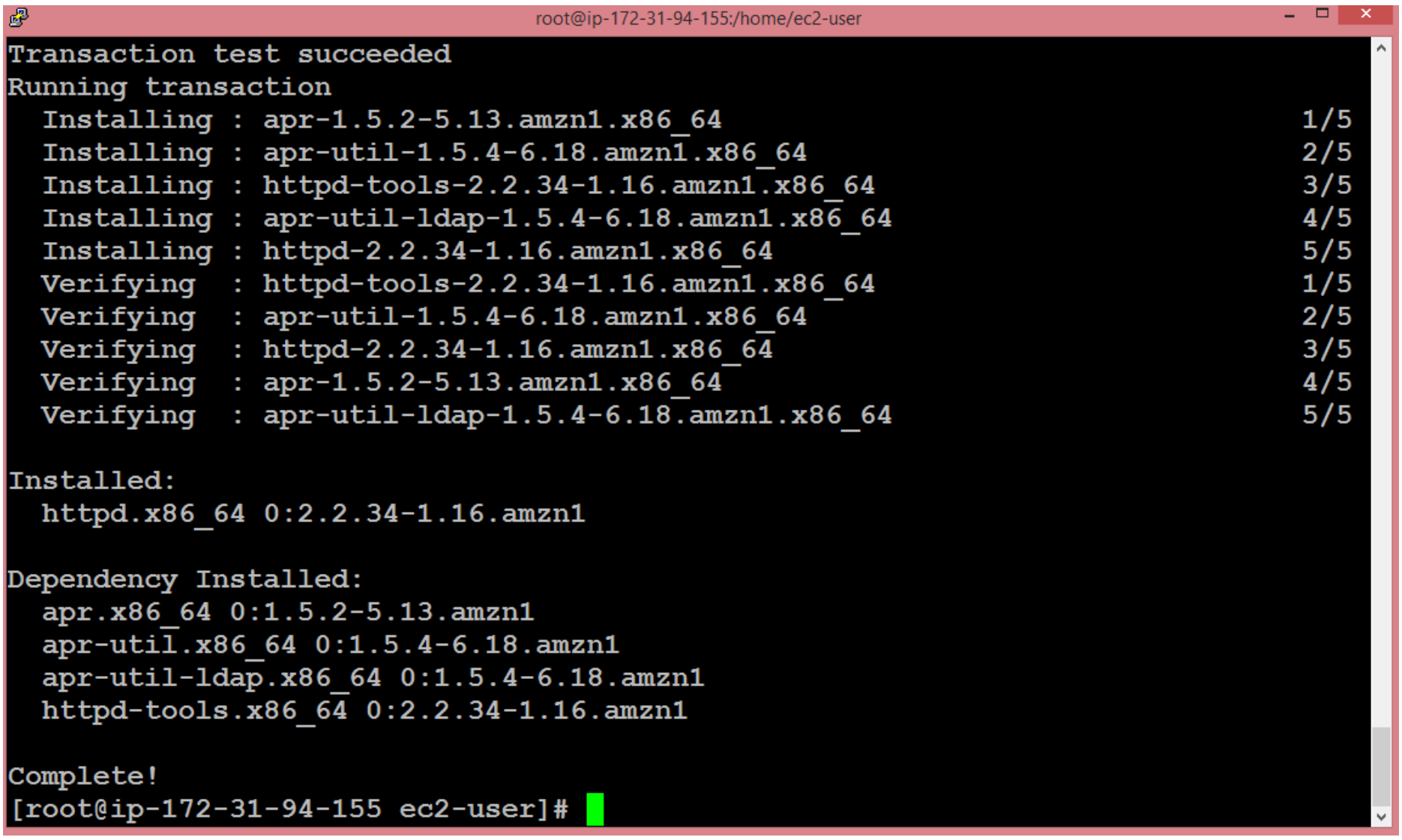
Creating Load Balancer

- Sign in to the AWS Management Console.
- Create an EC2 instance.
- An EC2 instance is in a stopped state. Start the instance by clicking on the **Actions** dropdown menu and then click on the **start**.



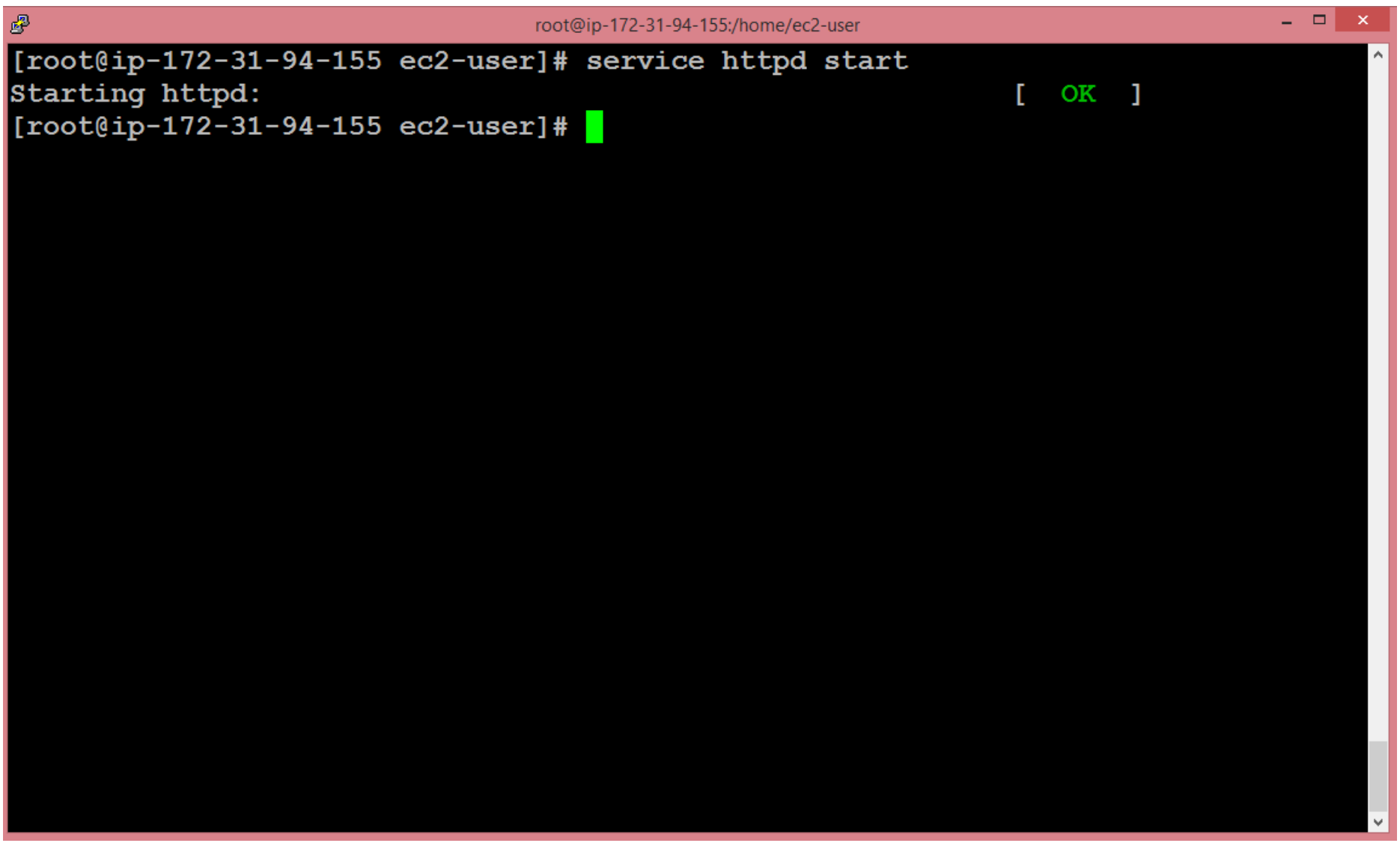
Now, my instance is running, and its IP address is 18.191.224.149.

- Open the putty.
- Run the command **sudo su** to provide the privileges to the root device.
- Run the command **yum update ?y** to update the EC2 instance.
- Install the Apache server by using the command **yum install httpd ?y**.

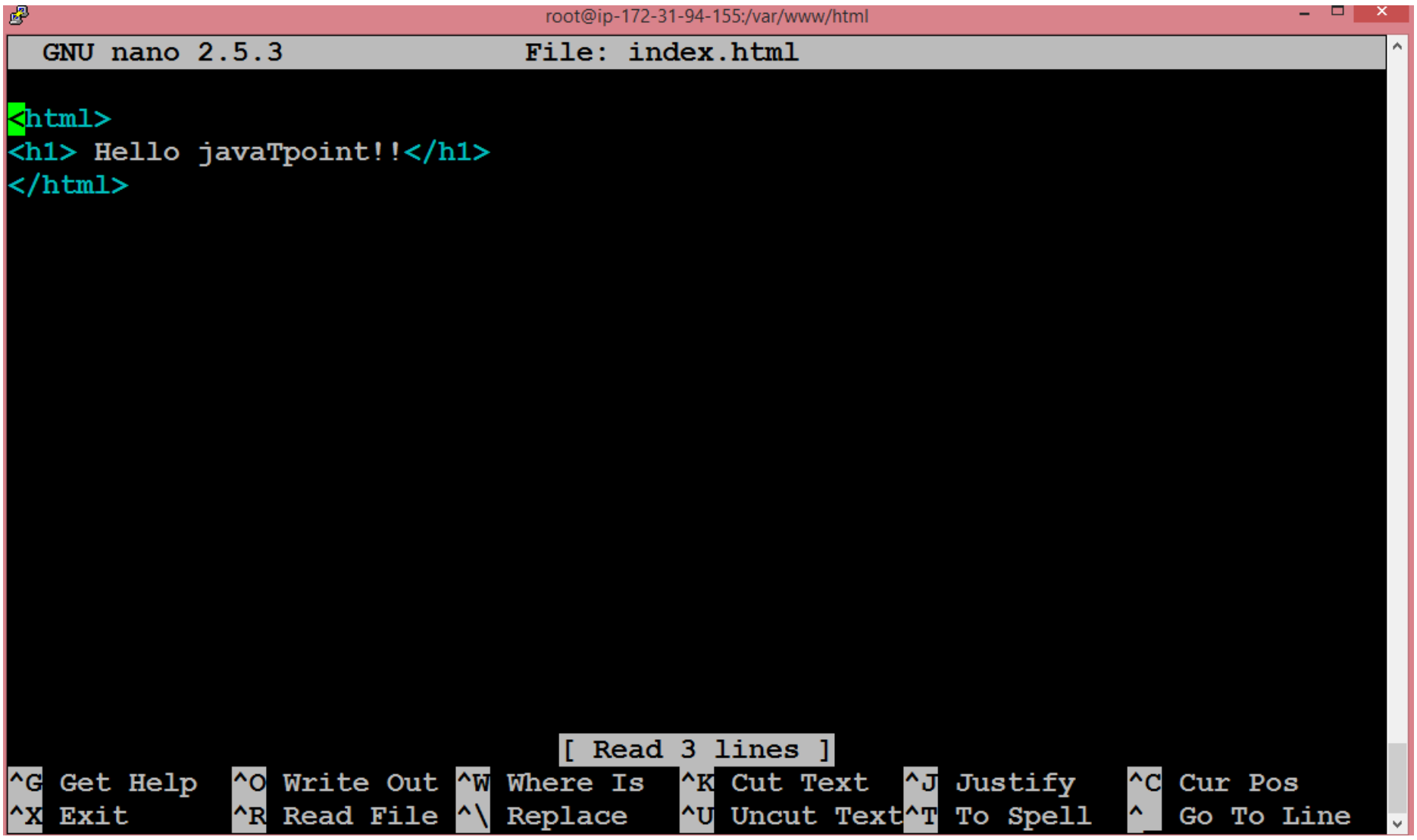


The above screen shows that the server has been installed successfully.

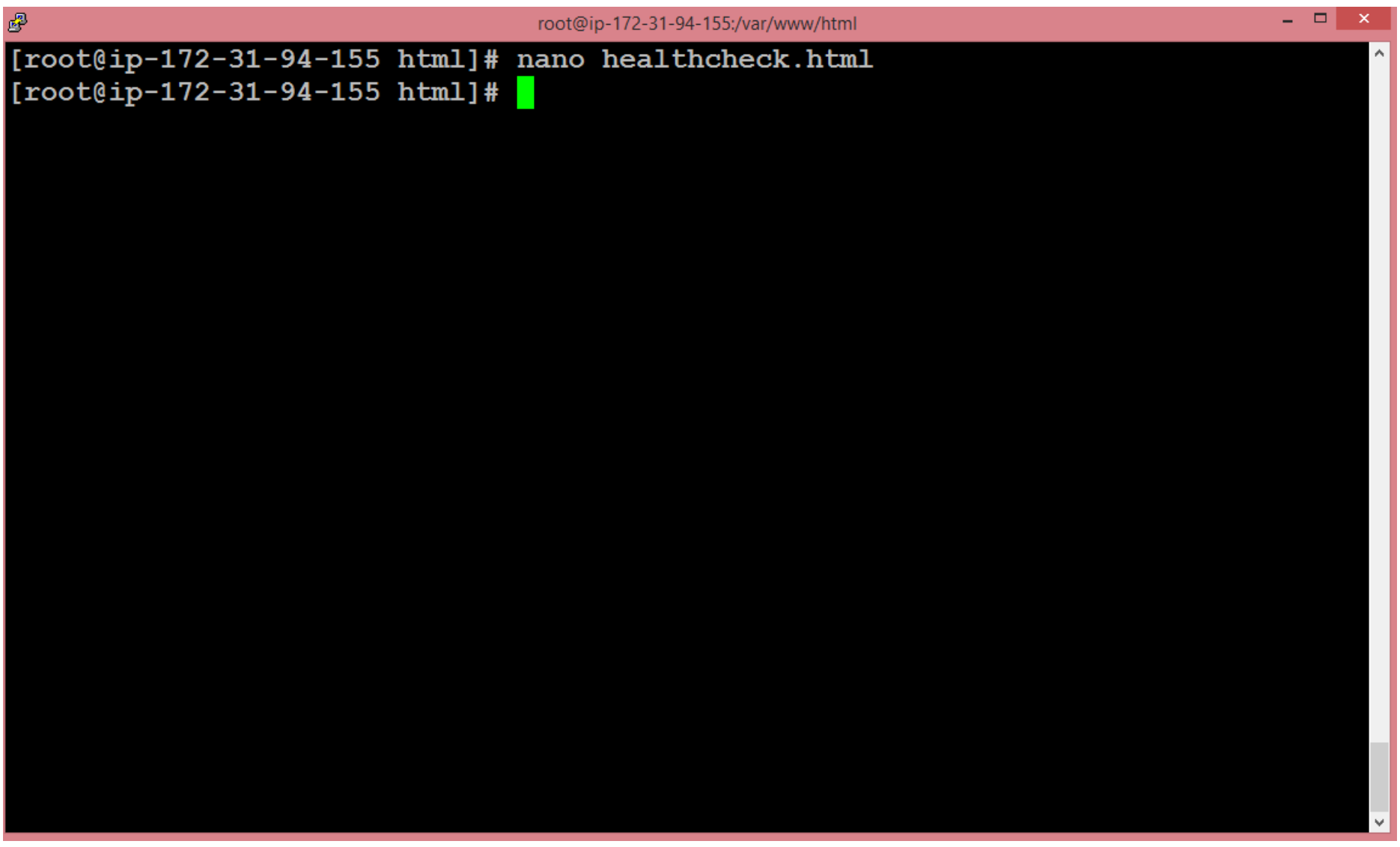
- Start the server by using the command **service httpd start**.



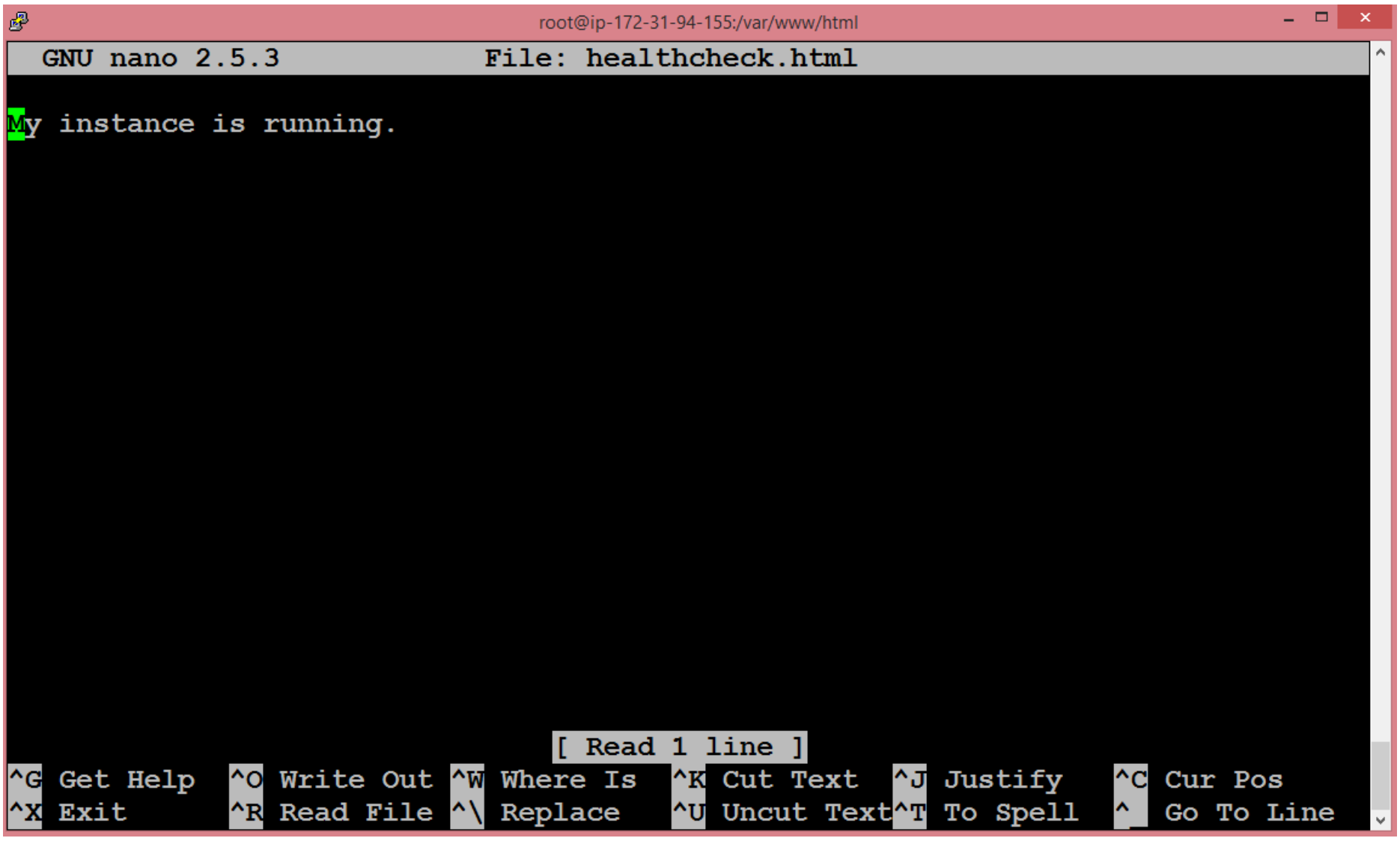
- Move to the html directory by running the command **cd /var/www/html**.
- Run the command **nano index.html** to create the editor.



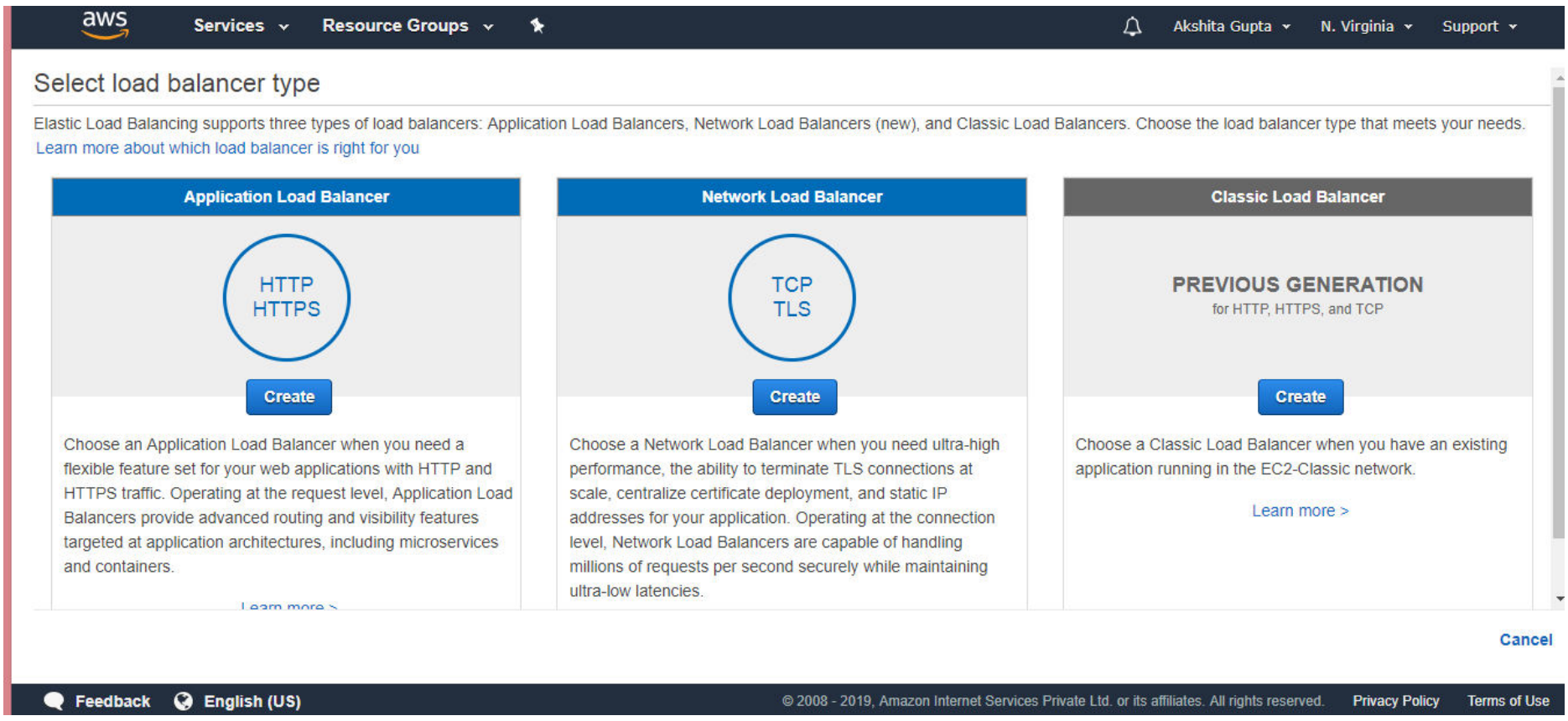
- Now, we are going to create **nano healthcheck.html** file.



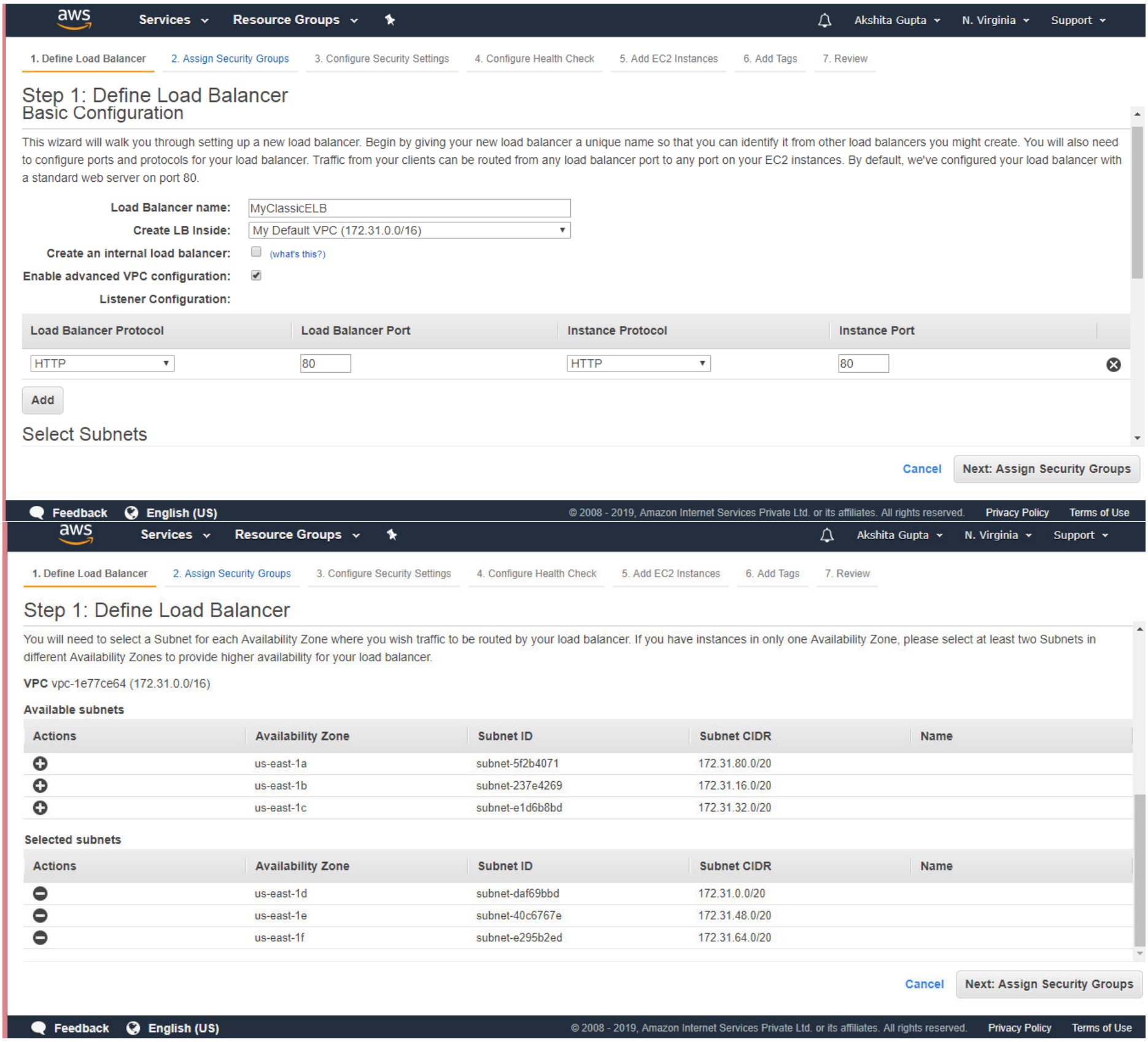
- I write "**My instance is running**" to healthcheck file.



- Move to the EC2 service, click on the Load Balancer appearing on the left side of the console.
- Click on the Create Load Balancer. On clicking, three types of Load Balancers are shown:



- We create a **Classic Load Balancer**.
- On clicking on the **create** button, the screen appears shown below:



Load Balancer name: It is the name of the Load balancer that the user provides. Suppose I have given a ClassicELB as a load balancer name.

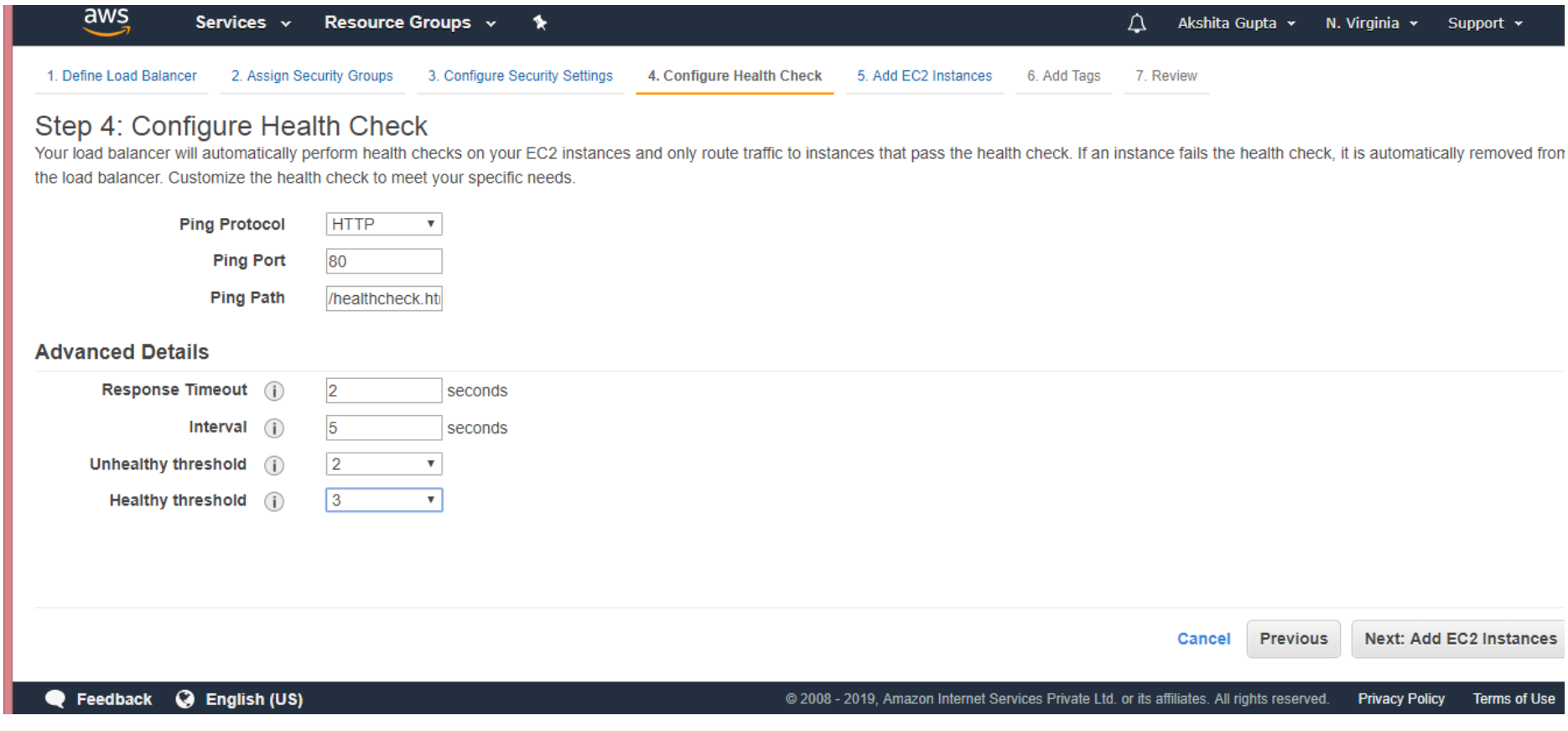
Create LB inside: I kept it as a default VPC.

Create an internal load balancer: As we want to serve external web traffic, so we need an external load balancer, not an internal load balancer. Uncheck this field.

Enable advanced VPC configuration: Check this field to add at least one subnet.

Linear Configuration: It describes from which protocol and port, it is listening, and to which port it is passing.

- Click on the **Next** button.
- Configure Health check.



Ping Protocol: It defines the type of protocol.

Ping port: It defines the port number.

Ping Path: It defines the path of the web page that we created, i.e., **healthcheck.html**.

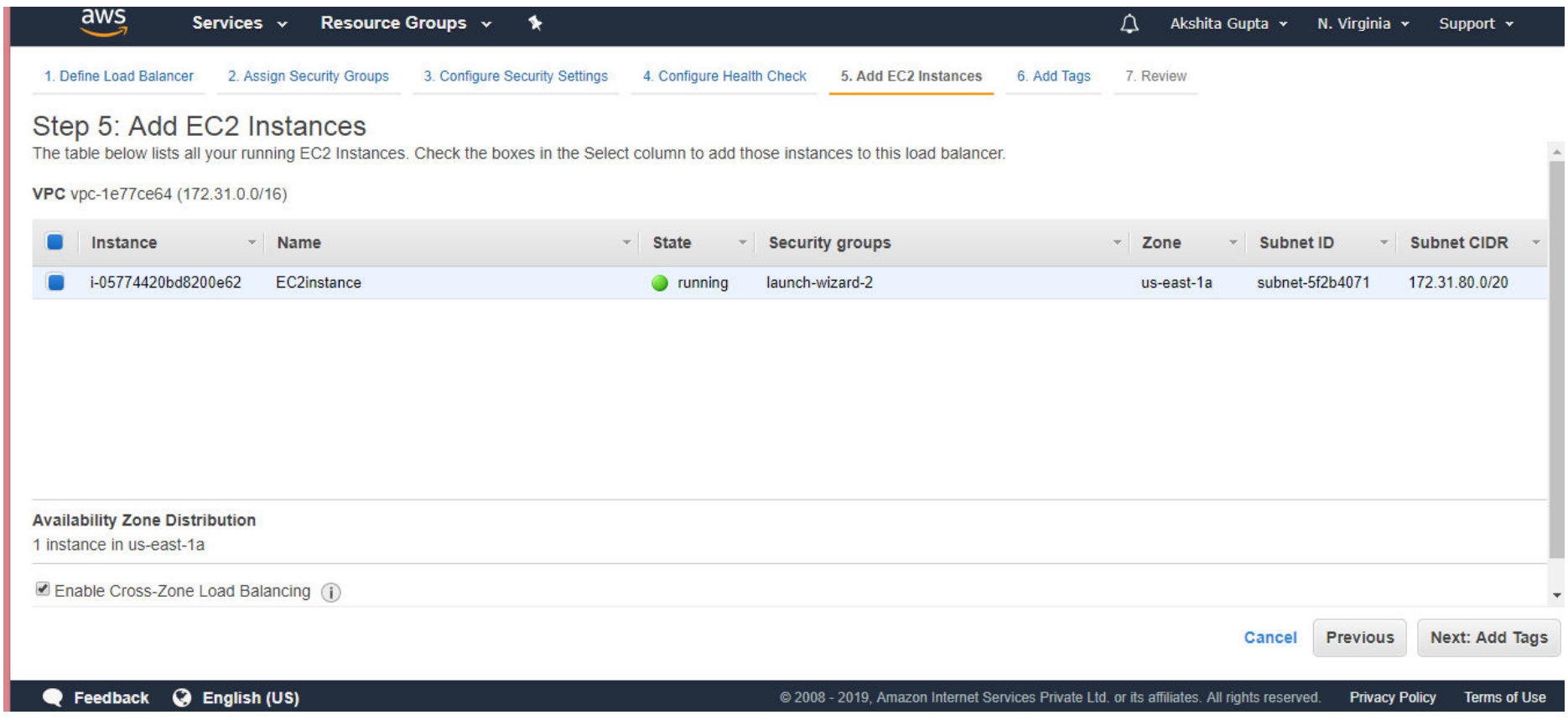
Response Timeout: It defines how long it will take and waits for the response.

Interval: It is the amount of time between health checks.

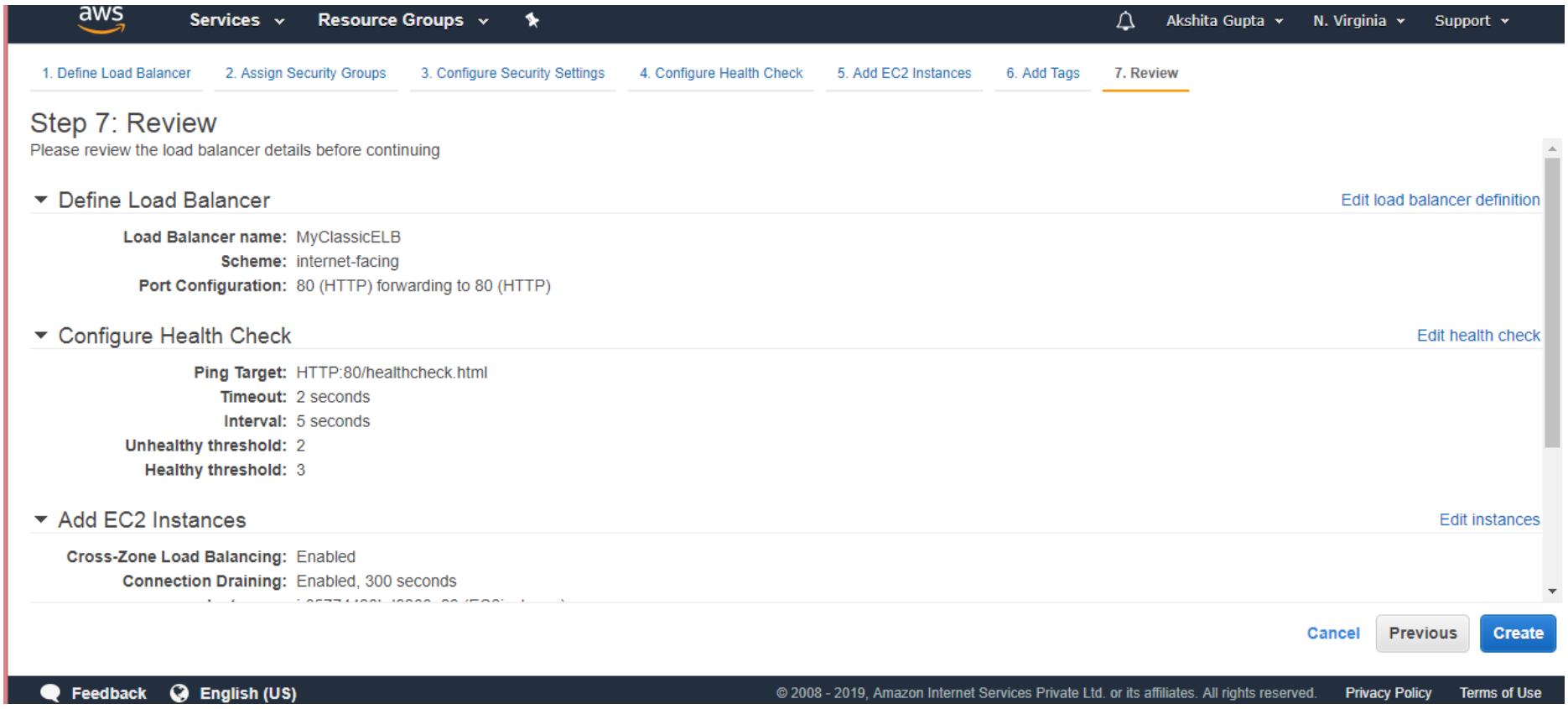
Unhealthy threshold: It defines the number of consecutive health check failures before declaring an EC2 instance unhealthy.

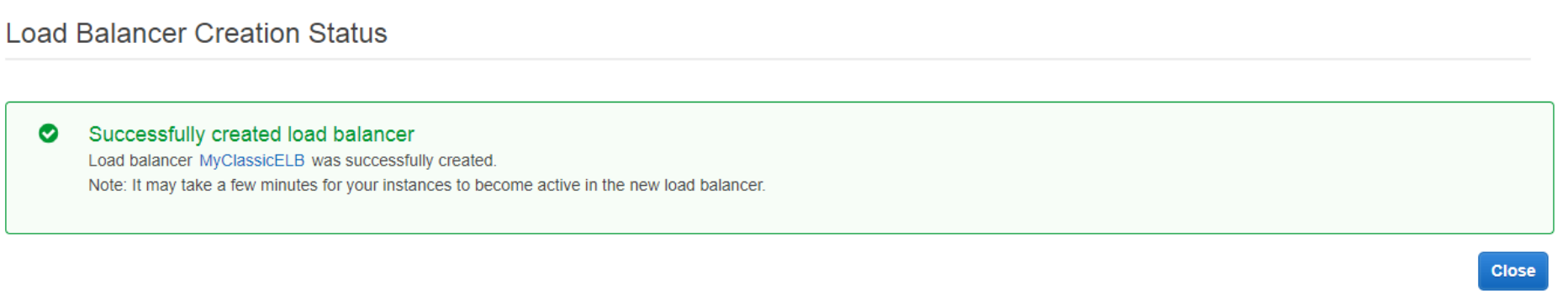
Healthy threshold: It defines the number of consecutive health check successes before declaring an EC2 instance healthy.

- Click on the **Next**.
- Add your EC2 instance to the Load Balancer. Check the EC2 instance box.



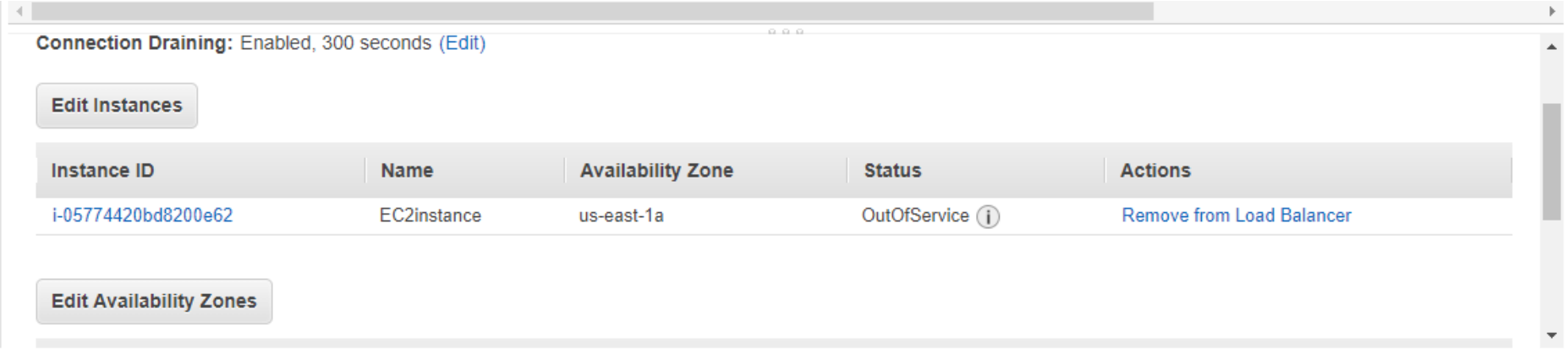
- Click on the **Next**.
- Review the load balancer, and then click on the **Create** button.



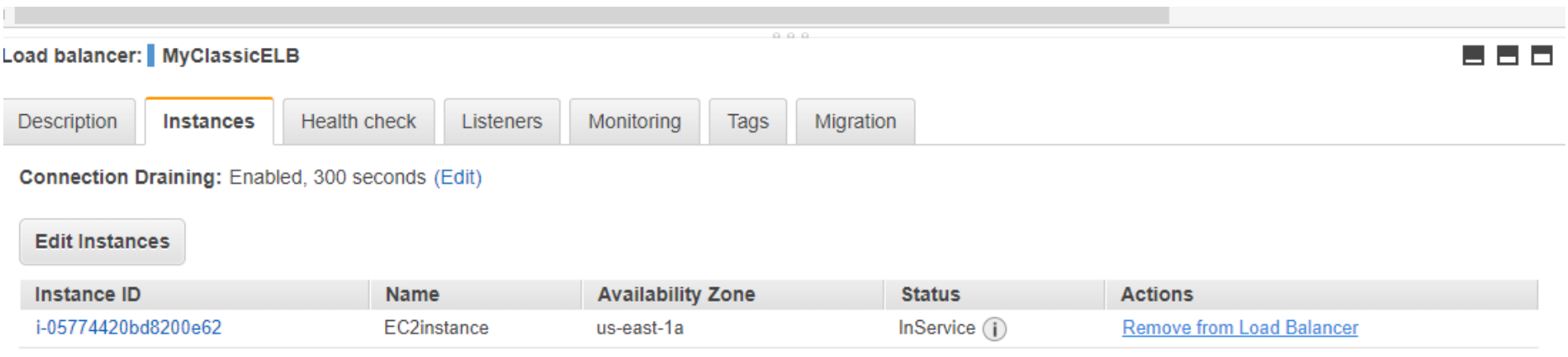


The above screen shows that Class load balancer has been successfully created.

- When we check the status of an instance, the status appears as **OutOfService**.



- After 1 or 2 minutes, the status of an instance appears as **InService**.



- Copy the DNS name of a Load balancer and paste it to the clipboard of a web browser. You will see the output which is shown below:

Therefore, we can say that DNS name is converted into a public IP address which is directing you to the **index.html**. Amazon provides DNS name to the Classic Load Balancer rather than a public IP address as the public IP address can be changed.

8. Auto Scaling

What is Auto Scaling?

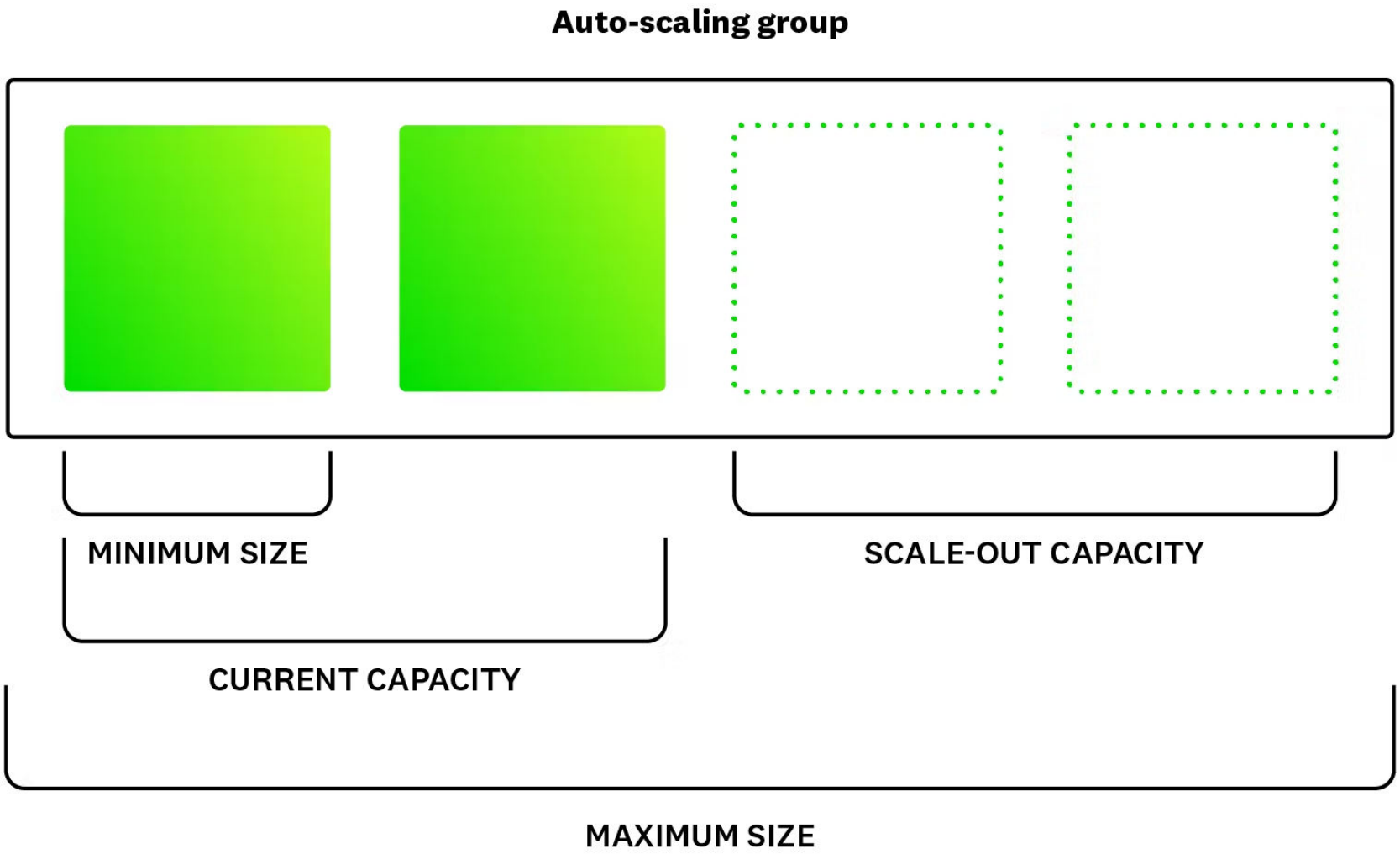
Auto-scaling is a scaling technique you can apply to workloads hosted in a cloud environment. One of the major benefits of cloud-based hosting is that you can readily scale capacity to whatever extent is needed to support the demand for your service. Auto-scaling takes that advantage a step further. With auto-scaling, as the demand for a given workload changes over time, the amount of resources allocated to support that workload adapts *automatically* to meet your performance requirements.

Before auto-scaling was an option, scaling workloads was often challenging. Allocating resources manually to support a workload is inherently error-prone because it is difficult to precisely predict changes in demand or know how many resources are needed to handle those changes. This ambiguity can lead to costly over-provisioning on the one hand, or potential service disruptions due to under-provisioning on the other. Auto-scaling helps solve these problems by automatically increasing or decreasing the amount of resources assigned to your workload in direct proportion to the amount that demand also increases or decreases.

In this article, we'll cover how auto-scaling works, why it is important, and when organizations are likely to use it. We'll also look at the most common offerings for auto-scaling, along with potential challenges that accompany its implementation. Finally, we'll discuss some important considerations for implementing an auto-scaling architecture effectively.

With auto-scaling, you typically configure resources to scale automatically in response to an event or metric threshold chosen by your organization. Engineers identify these events and metric thresholds as ones that most closely correlate with degraded performance.

For example, a developer could create a threshold of 70 percent memory usage for greater than four minutes. The developer could then additionally configure a response that would launch two additional instances every time this threshold is met or crossed. The developer could also define a minimum and maximum limit for scaling. What is the minimum acceptable number of nodes that should ever be used to run this workload, and conversely, what is the maximum?



Besides relying on triggering events or metrics, you can also configure auto-scaling to occur according to a predetermined schedule. For companies and services that have cyclical (or otherwise predictable) load demands, this method lets you preemptively scale infrastructure in anticipation of higher demand and then scale back as needed.

Auto Scaling Components

The three components of Amazon EC2 Auto scaling define the three aspects i.e. what, where, and when.

Here are the details:

- **Configuration templates**

This component defines **WHAT** will be launched by the auto scaler. As a configuration template for such EC2 instances, your organization can utilize a launch template and perhaps a launch configuration. For your instances, you can specify details like the AMI ID, instance type, key pair, private networks, as well as block device mapping.

- **Groups**

This component defines WHERE will the autoscaling take place. Basically, defining which VPC and subnets to use, the load balancer, as well as the minimum and the maximum number of EC2 instances. Your EC2 instances are grouped to enable scaling and administration by treating them as logical units. You can select the minimum, maximum, as well as desired number of EC2 instances when creating an AWS EC2 auto scaling group. Exploring the Auto Scaling groups will help you find several additional features in detail.

- **Scaling options**

This component defines **WHEN** autoscaling takes place. You can scale your Auto Scaling groups in several ways with AWS EC2 Auto Scaling. For example, users can set up a group to scale depending on a timetable or when certain circumstances are met for dynamic scaling.

Advantages of Auto Scaling

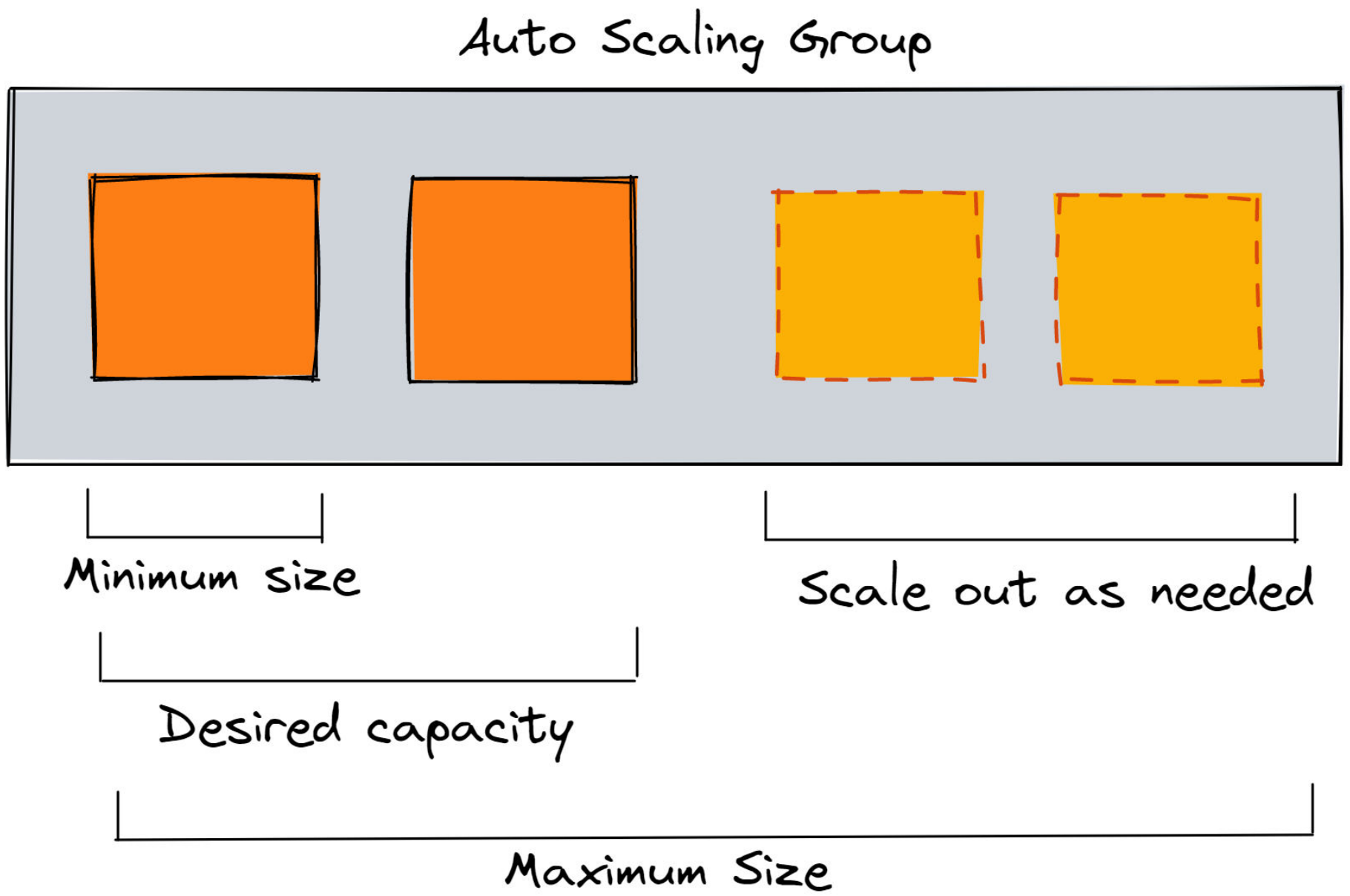
- Amazon Auto Scaling constantly provides surveillance to the application to confirm that they are operating at the level in which you have ordered. It can change automatically as the demand skyrockets Auto Scaling automatically adjusts the capacity of the resources so as to maintain the high quality of the service. It also helps when the workload is periodic, irregular, unpredictable, and changes continuously.
- AWS Autoscaling automatically creates all of the scaling policies and sets targets for the person which is based on the preference. AWS also monitors the application and adjusts the capacity of the resource group as per the demand. It also enhances us with the fact that how groups of different resources respond to changes in demand.
- Autoscaling will help to manage all the resource provisioning for all the EC2 auto-scaling groups and database tables in the application. One can easily observe the average utilization of all of the scalable resources without navigating into other consoles.
- It helps you to monitor your utilization and cost efficiencies while using the services of AWS. This helps to pay only for what you have utilized and what you need. AWS manages the capacity used and notifies the user according to it. AWS Autoscaling is free and removes the quantity, not in use and thus, helps to avoid overspending.

Auto Scaling Groups (ASG)

An Amazon **Auto-Scaling Group (ASG)** is a logical group of Amazon EC2 instances with identical features. ASG is a cloud services technology that allows for the appropriate allocation of computational resources. Every Amazon EC2 instance in the group must adhere to auto-scaling policies. The number of occurrences in ASG is used to calculate the size of ASG. It's also known as auto-scaling or automated scaling. As a user's needs change, the number of active servers will also change automatically depending on the demand on the group servers.

The ASG is widely used to monitor applications and automatically adjust capacity to maintain a steady and predictable performance at the lowest possible cost. It also makes it easy to set up application scaling.

Auto-Scaling in Cloud Computing



Auto-scaling is a feature of **cloud computing** that helps organizations scale the capacity of cloud services or virtual machines up or down based on traffic levels. **Amazon Web Services (AWS)**, **Microsoft Azure**, and **Google Cloud Platform (GCP)** are the cloud computing components that provide the auto-scaling feature.

By automatically expanding and lowering fresh instances as demand rises and falls, auto-scaling reduces cost and enables consistent functionality. As a result, amidst changing and often unforeseen requests for services, auto-scaling ensures stability. Auto-scaling also avoids the need to react explicitly to heavy traffic in real-time which would necessitate more tools and instances. Furthermore, auto-scaling allows for the installation, tracking, and deactivation of each unit.

Launch Templates

AWS launch configurations and templates could appear to be pretty similar at first glance. Both allow you to define the EC2 Instance specifics. Let’s have a look at their differences and see which one we should prefer.

What is Launch Configuration?

AWS Launch configuration is an instance configuration template that an Auto Scaling group uses to launch EC2 instances since 2010. Launch configuration requires information like Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.

AWS strongly recommends that you **do not use launch configurations**. They do not provide full functionality for Amazon EC2 Auto Scaling or Amazon EC2. Amazon requests customers to migrate Launch configuration templates to Launch templates.

What is Launch Templates?

Launch Templates is not a new capability. It’s available since 2017. Launch Templates enable a new way of templating your launch requests. Launch Templates reduce the number of steps required to create an instance by capturing all launch parameters within one resource. This makes the process easy to reproduce.

Also, with support for Auto Scaling, Spot Fleet, Spot and On-Demand instances, Launch Templates make it easier to implement standards and best practices, helping you to manage costs better, improve your security posture, and minimize the risk of deployment errors. This capability is available at no additional cost

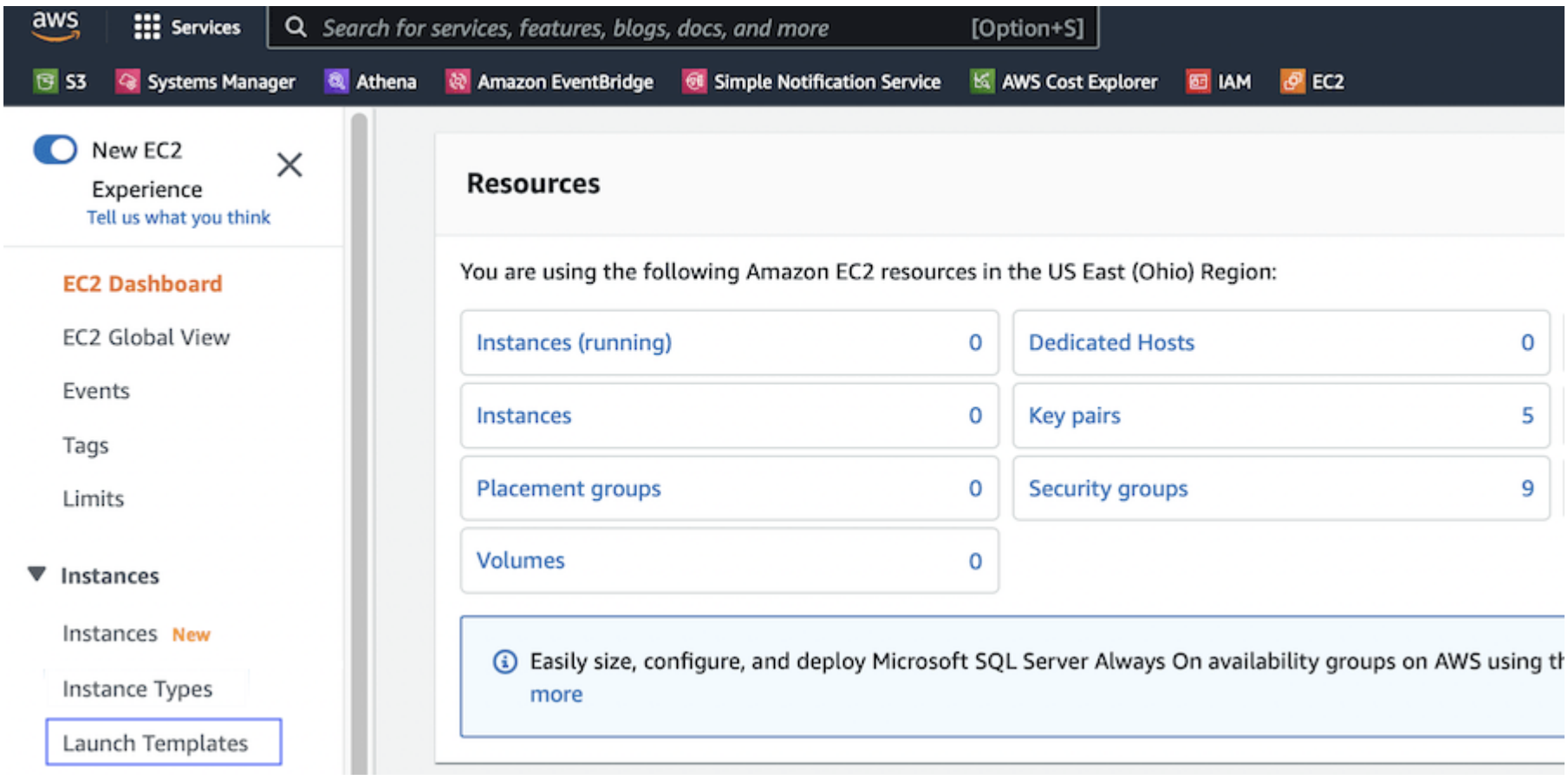
Advantages of Launch Templates

AWS has released many new features such as Mixed Instance Policies with Auto Scaling groups, Targeted Capacity Reservations, and unlimited mode for burstable performance instances that only work with launch templates.

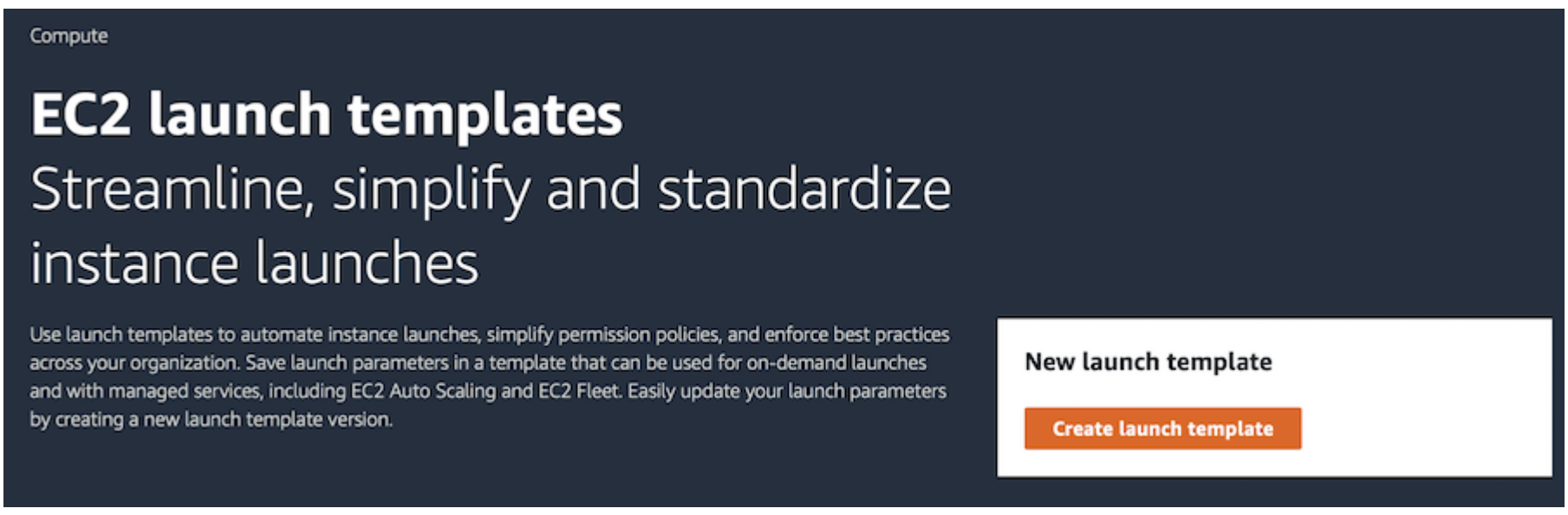
Creating an EC2 Launch Template

Let’s create a new launch template for Amazon Linux.

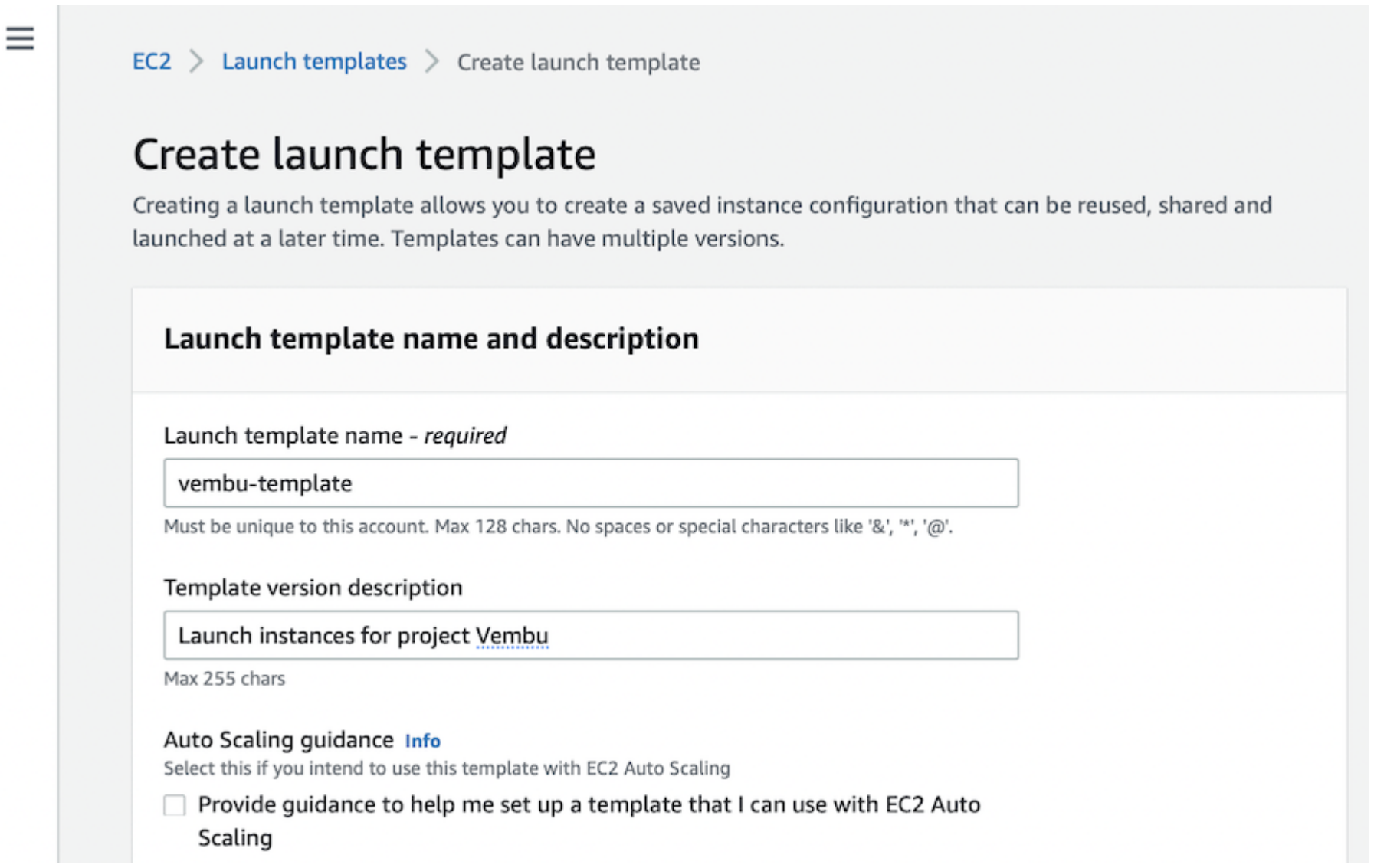
1. Login to AWS console and navigate to EC2



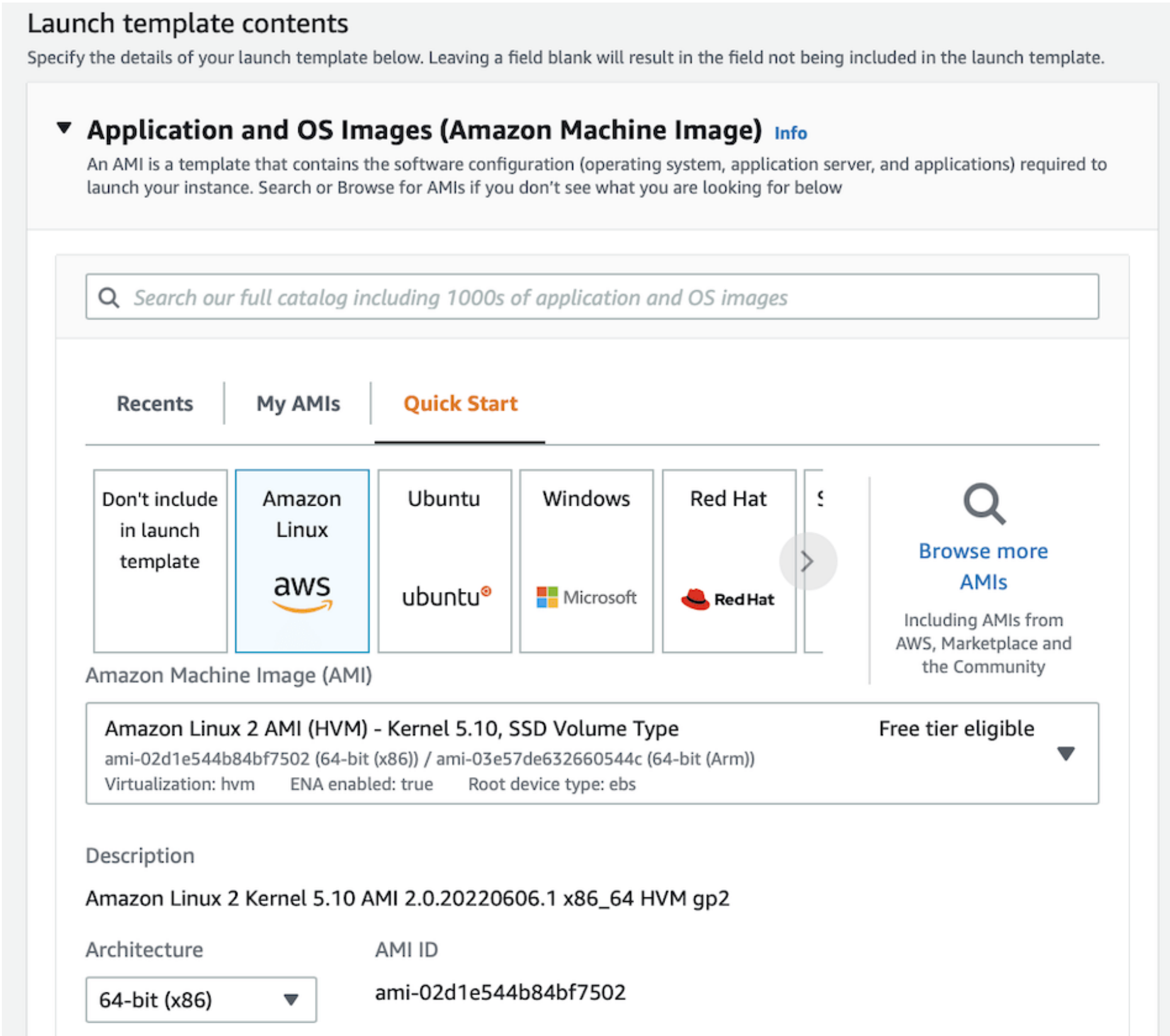
2. AWS Launch template section. Click on Create Launch Template



3. Enter the new launch template name

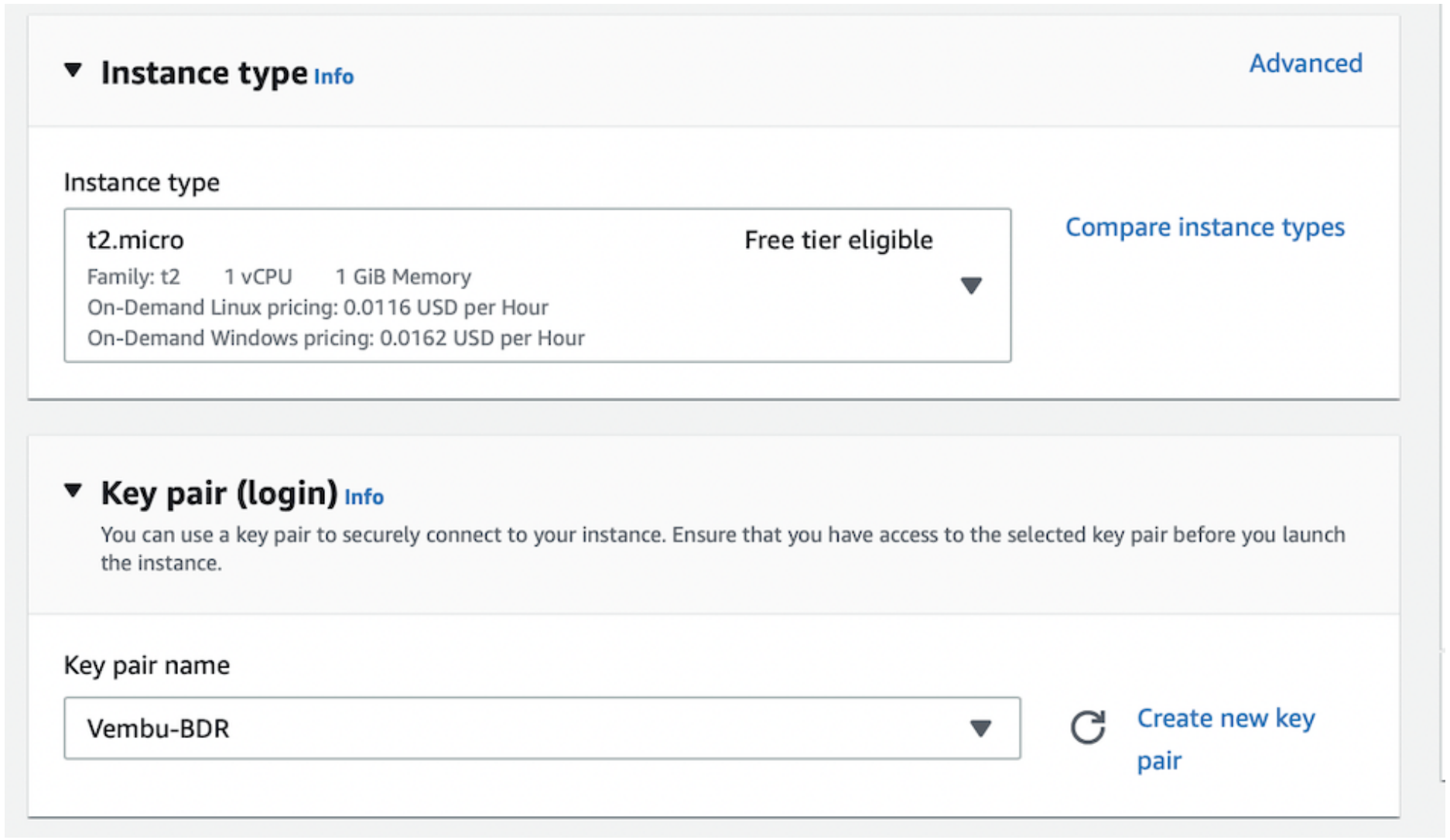


4. Select the OS image and architecture

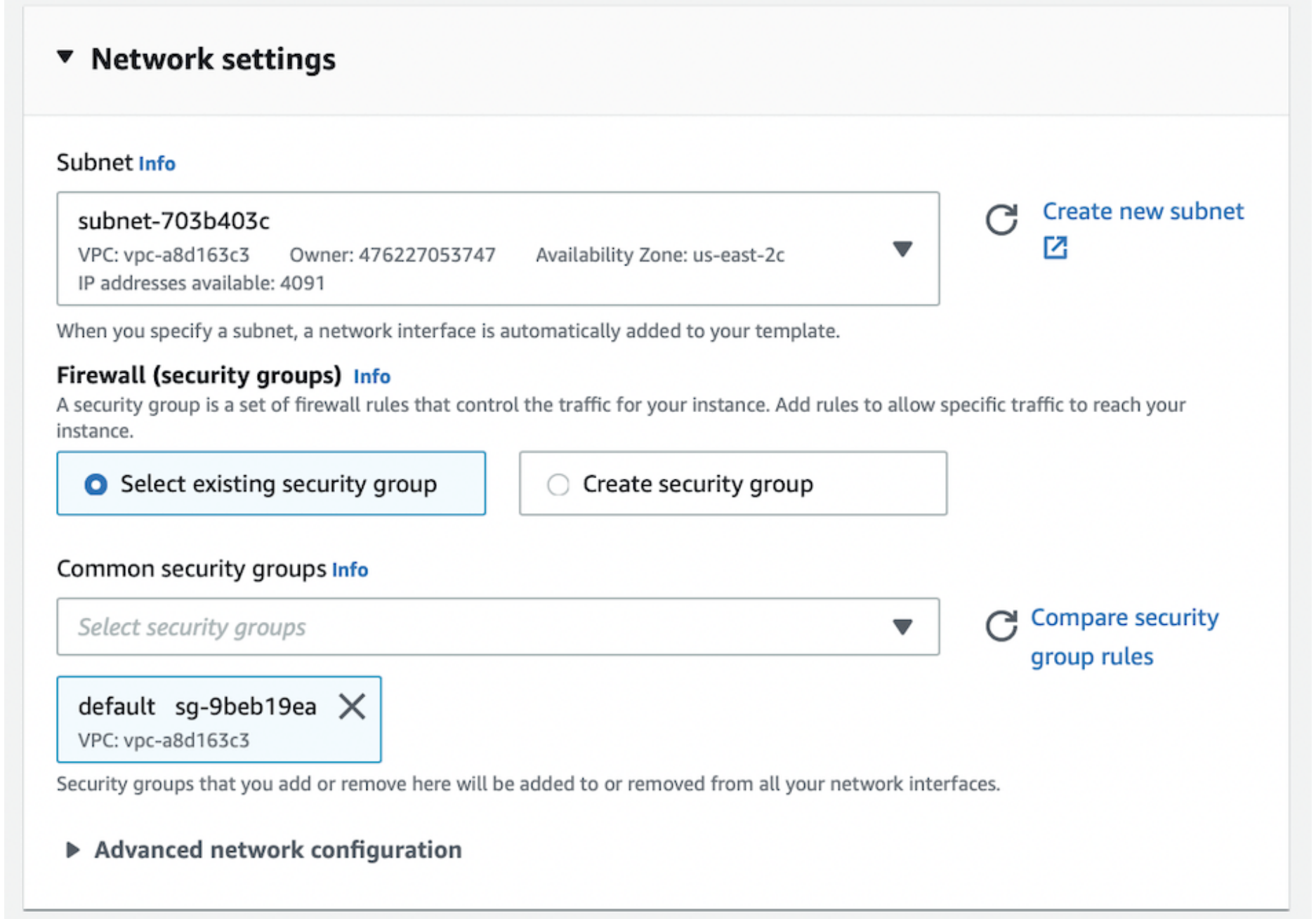


Are you looking to protect your EC2 instances on Cloud with flexible scheduling? please check out [BDRSuite]

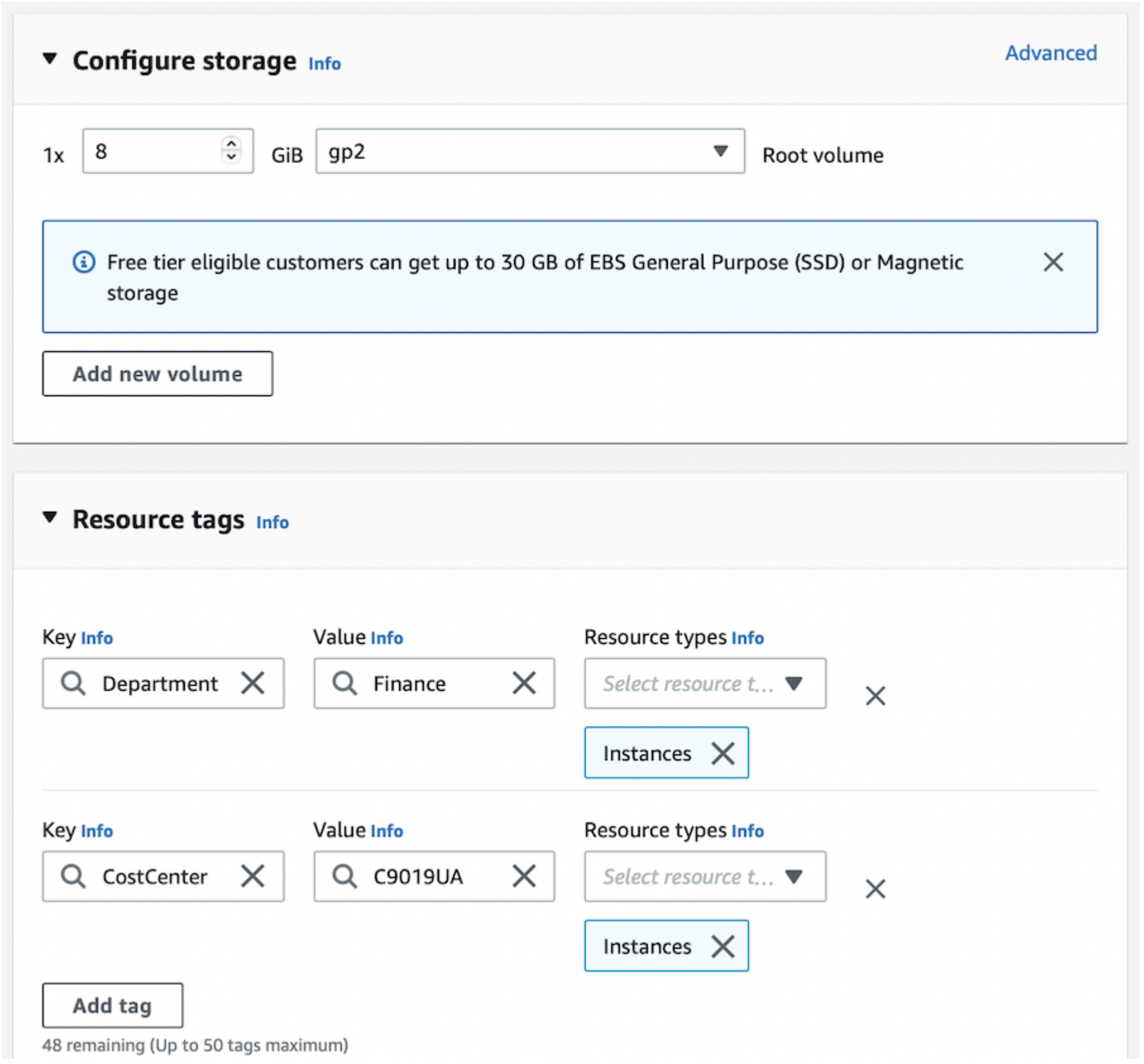
5. Select the instance type and key pair



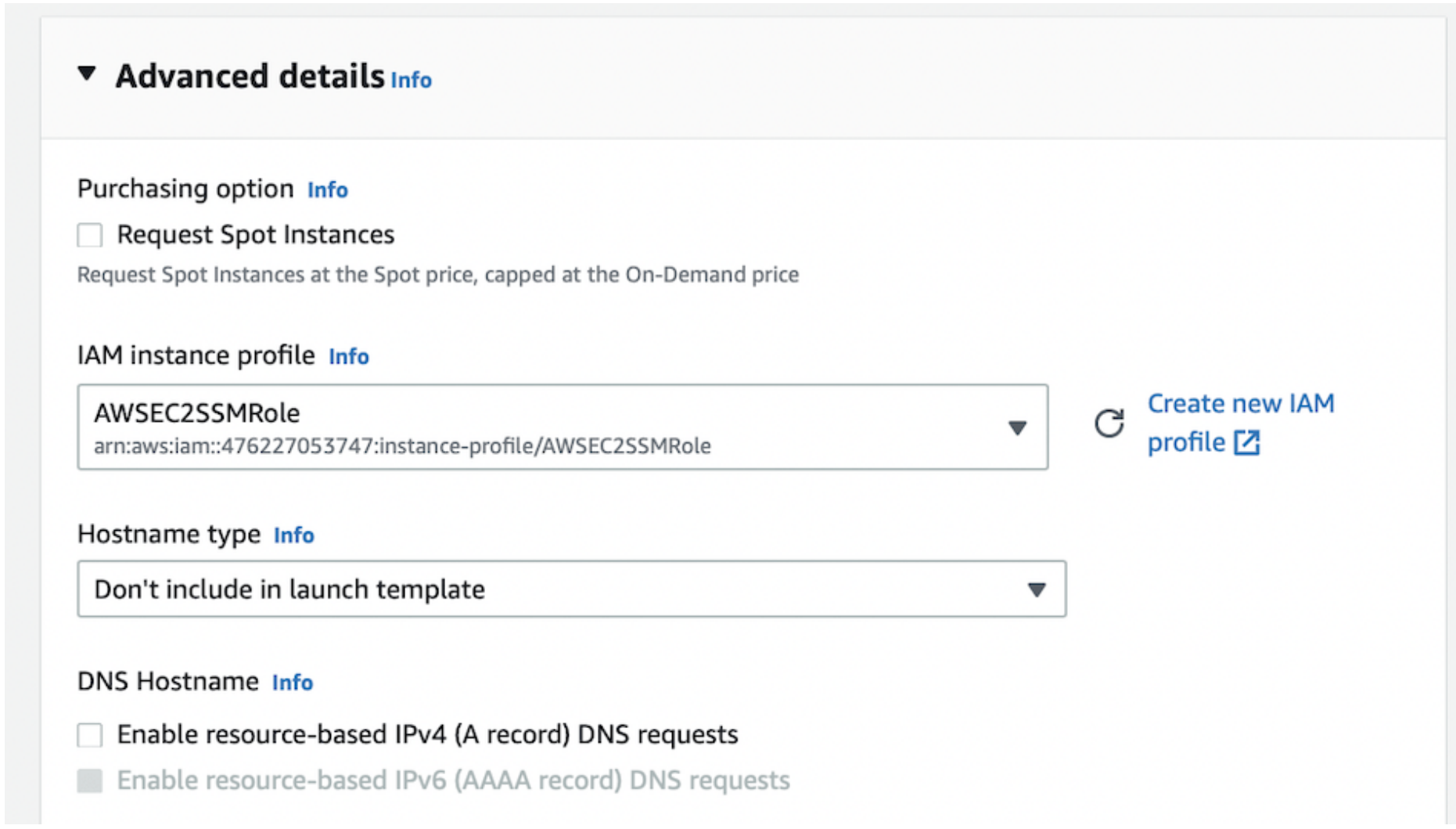
6. Select the subnet and security group



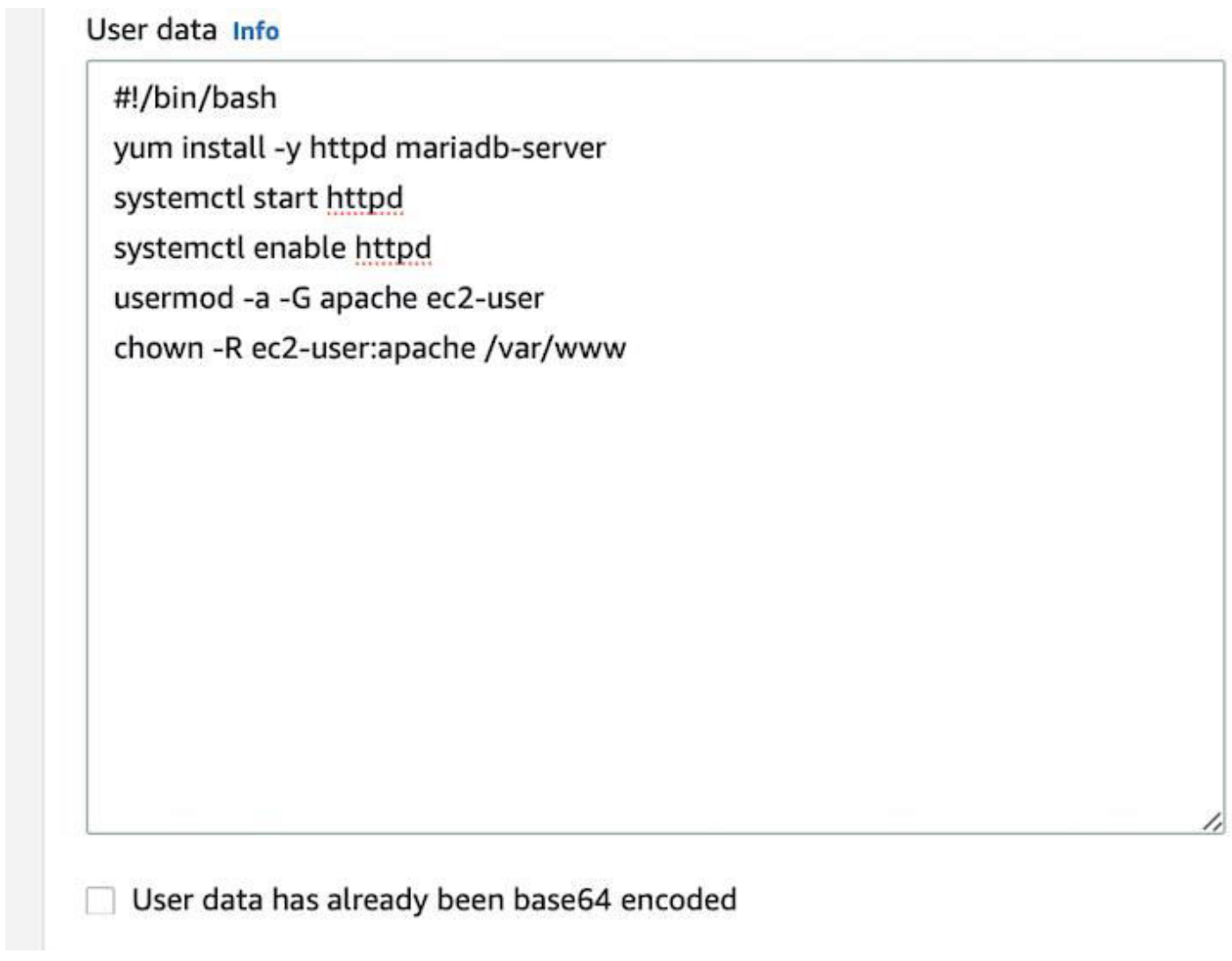
7. Select the root volume size and add the standard tags for your organization



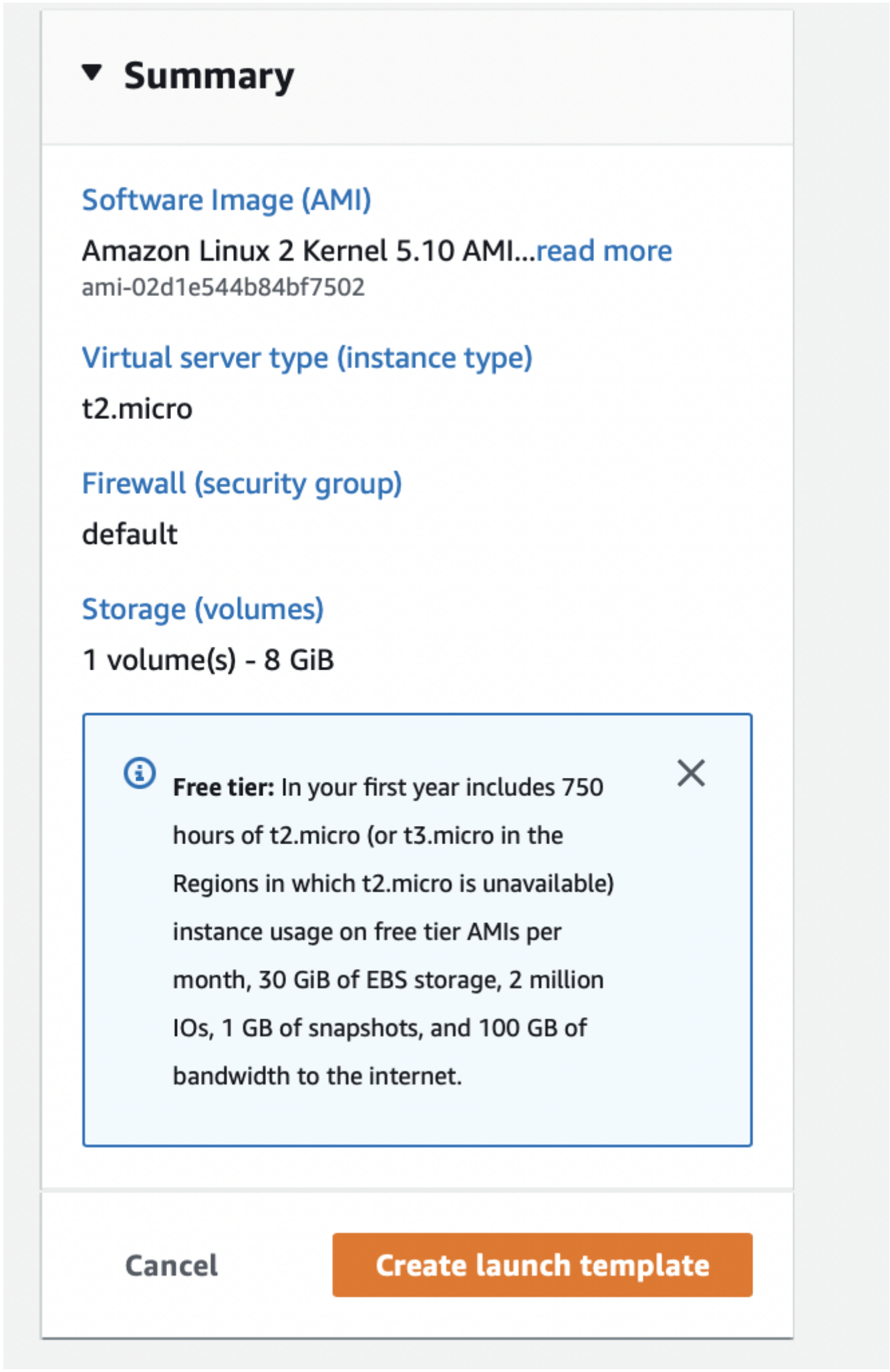
8. In Advanced settings, you could select the IAM role for the instance. Advanced details sections have other EC2 features which were missing in Launch configuration



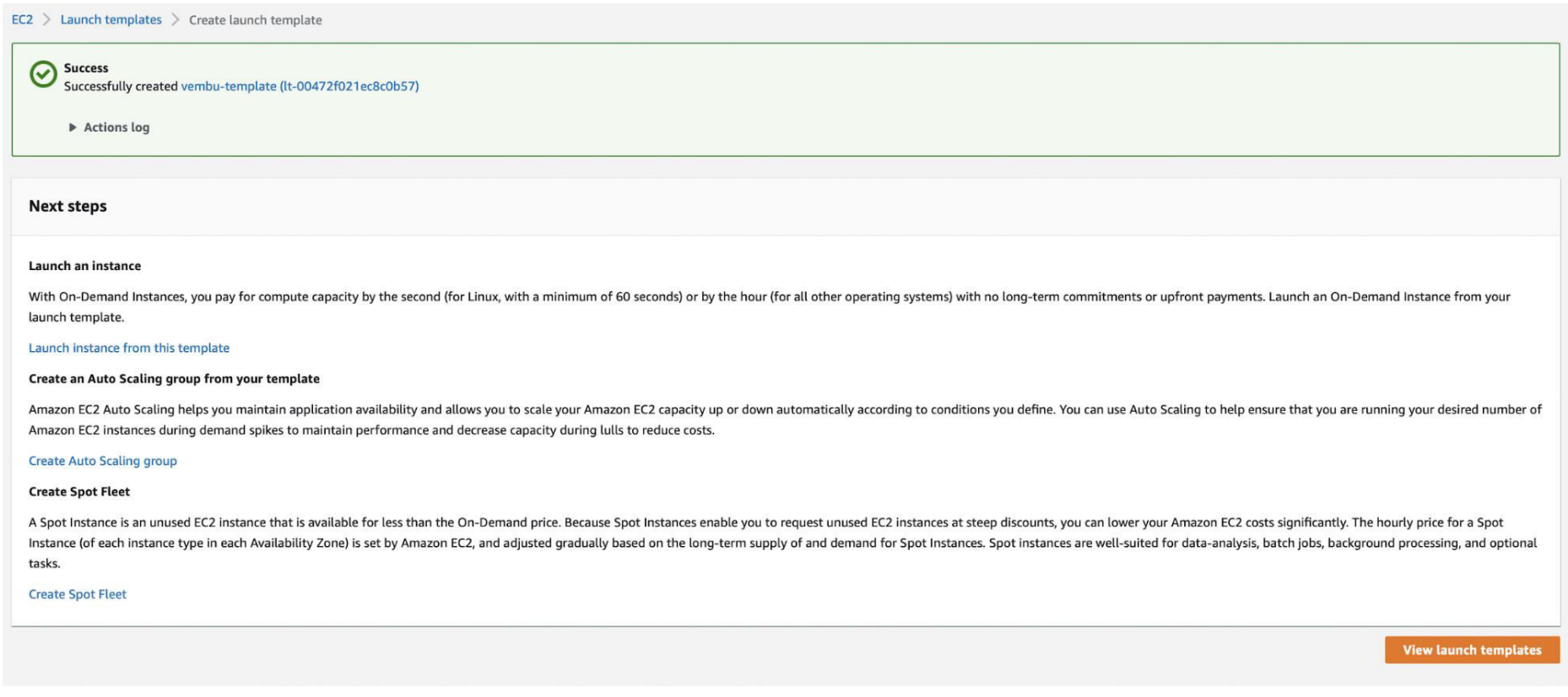
9. You can also add the “user data” to execute once the instance is getting started



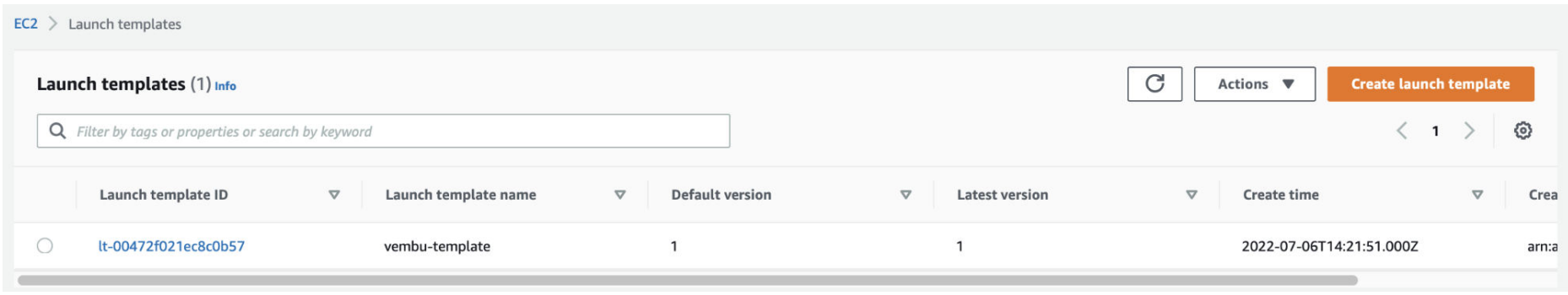
10. Create the launch template



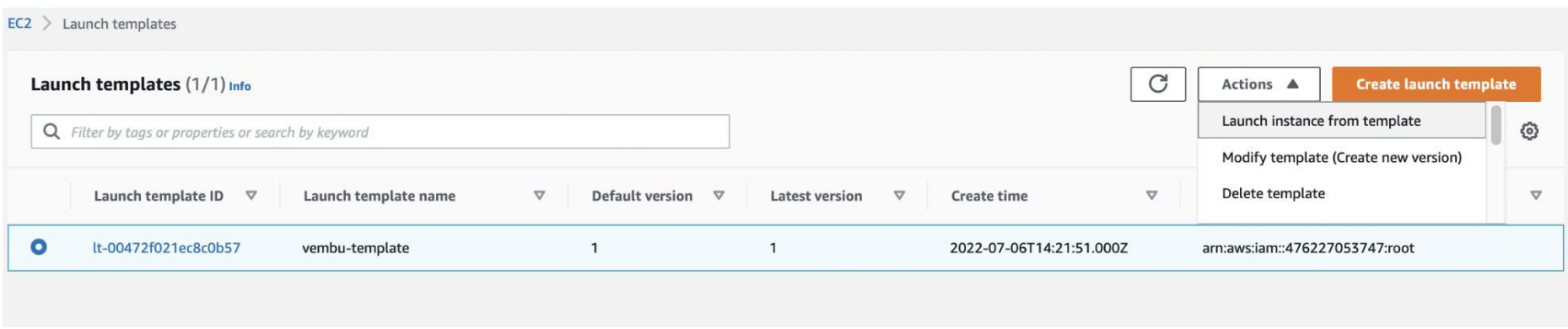
11. New launch template has been successfully created



12. Here we can see the newly created launch template



13. Let’s launch a new instance using the newly created launch template. Select the template and from actions – Select “Launch Instance from template”



14. For new instance launch, You just need to provide the number of instances required to launch

Launch instance from template

Launching from a template allows you to launch from an instance configuration that you would have saved in the past. These saved configurations can be reused and shared with other users to standardize launches across an organisation.

Choose a launch template

Source template

vembu-template
ID: lt-00472f021ec8c0b57

Source template version

1 (Default)
Launch instances for project Vembu

Instance details

Your instance details are listed below. Any fields that are not specified as part of the configuration below will use the template or default values for those fields. Ensure that you have permissions to override these parameters or your instance launch will fail.

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

AMI from catalog | Recents | My AMIs | Quick Start

Amazon Machine Image (AMI)

amzn2-ami-kernel-5.10-hvm-2.0.20220606.1-x86_64-gp2
ami-02d1e544b84bf7502

Free tier eligible

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Published	Architecture	Virtualization	Root device type	ENA Enabled
2022-06-14T19:54:18.000Z	x86_64	hvm	ebs	Yes

Summary

Number of instances Info: 1

Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI...read more
ami-02d1e544b84bf7502

Virtual server type (instance type)
t2.micro

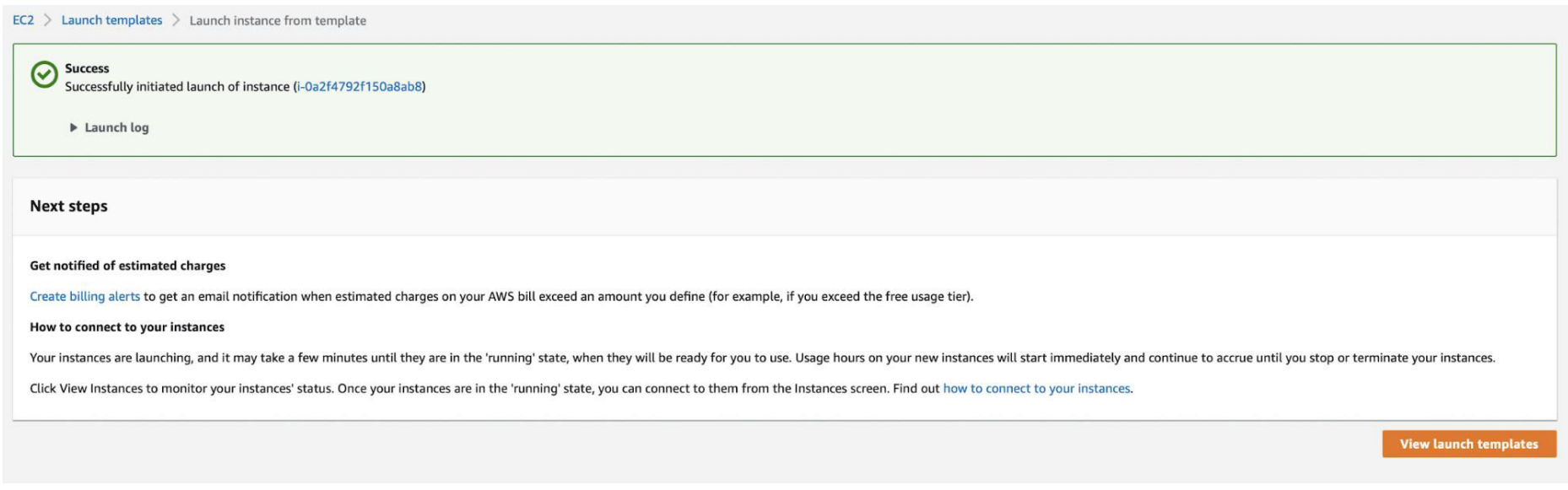
Firewall (security group)
default

Storage (volumes)
1 volume(s) - 8 GiB

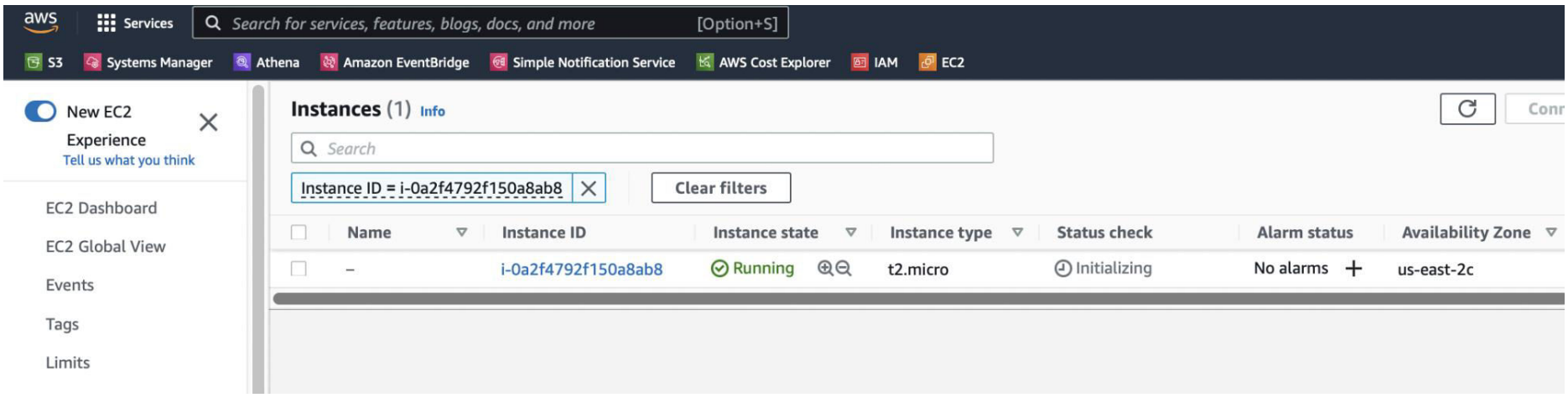
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel | Launch instance

15. Here we can see that new instance has been successfully launched



16. Here we can see that instance is starting



Create Auto Scaling Group

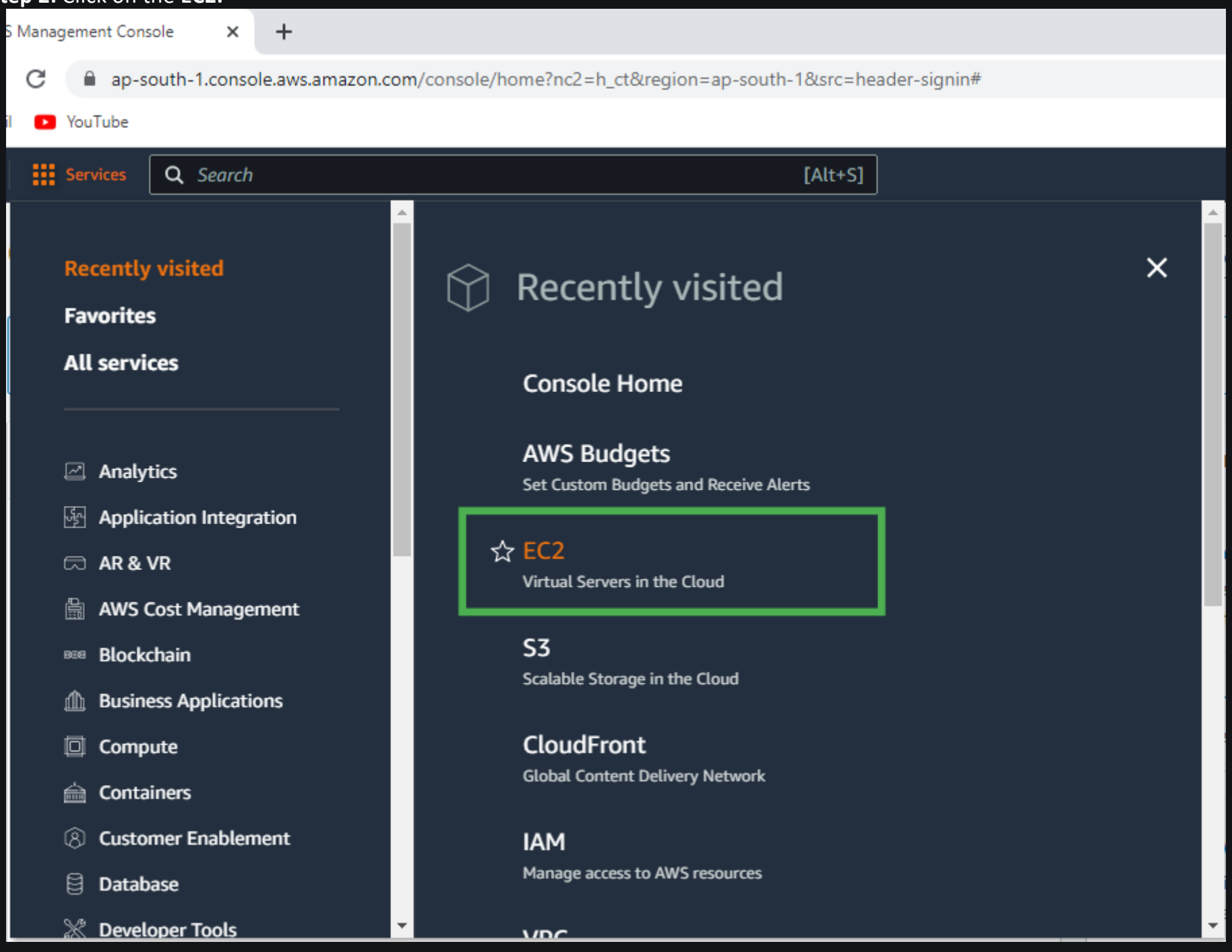
Auto Scaling is an Amazon Web Services it allows instance scalable when the traffic increases or CPU load increases. Auto-scaling is a service monitoring all instances that are configured into the Auto Scaling group and it ensures that loads are balanced in all instances.

Depending on the load scaling group increase the instance according to the configuration. When we create the Auto Scaling Group we configured the Desired capacity, Minimum capacity, or Maximum capacity and also configured CPU utilization. If CPU utilization increases by 60% in all instances then one more instance create and if CPU utilization decreases by 30% in all instances then terminate one Instance. These are setting totally up to us what is our requirement. If any Instance fails due to any reason then the Scaling group maintains the Desired capacity and starts another instance.

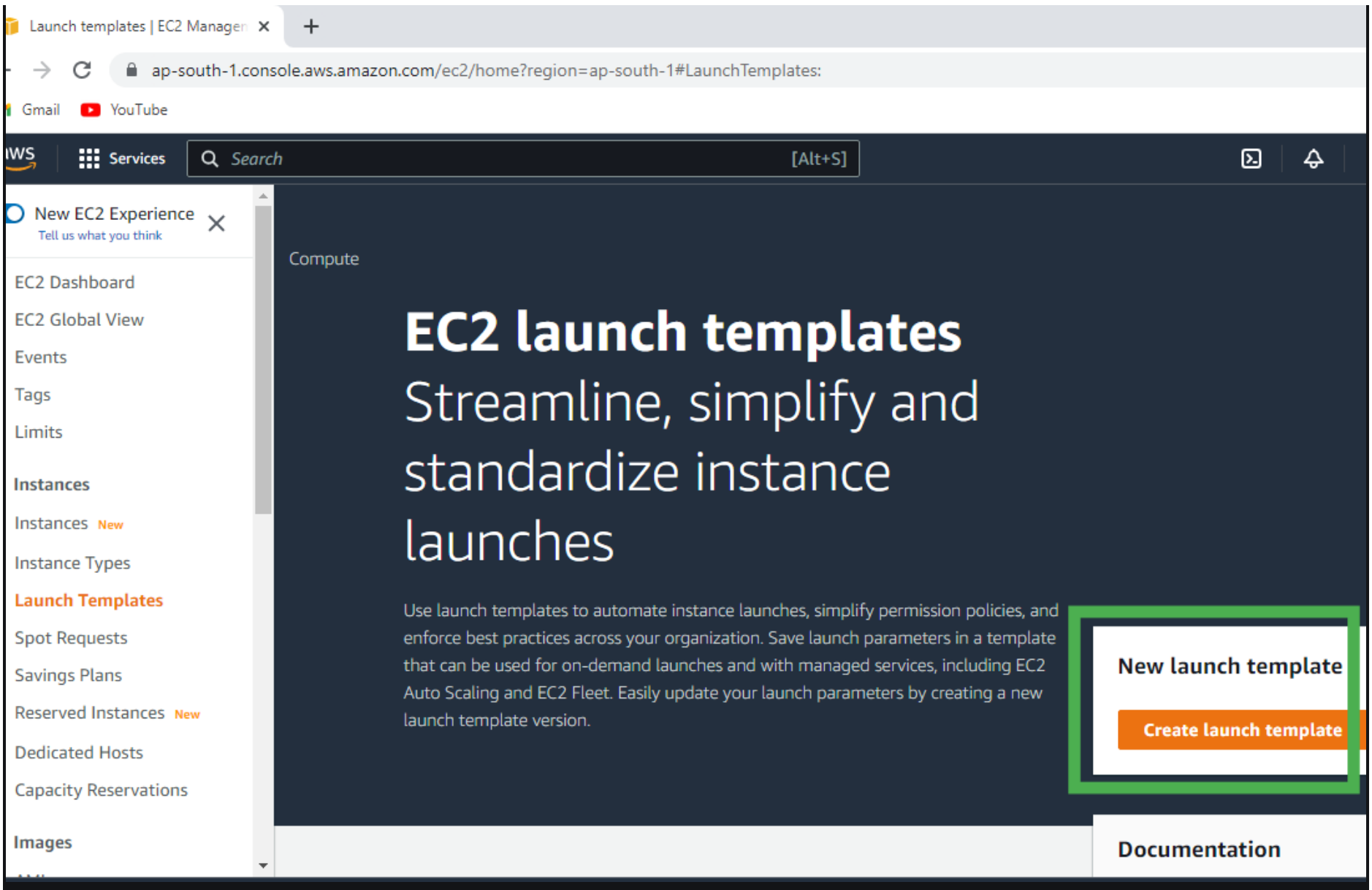
The Auto Scaling group follows Horizontal Scaling. This service is very important for us nowadays because we do not need to create new instances manually and do not require monitoring manually.

Steps to Setup the Auto Scaling Group in EC2

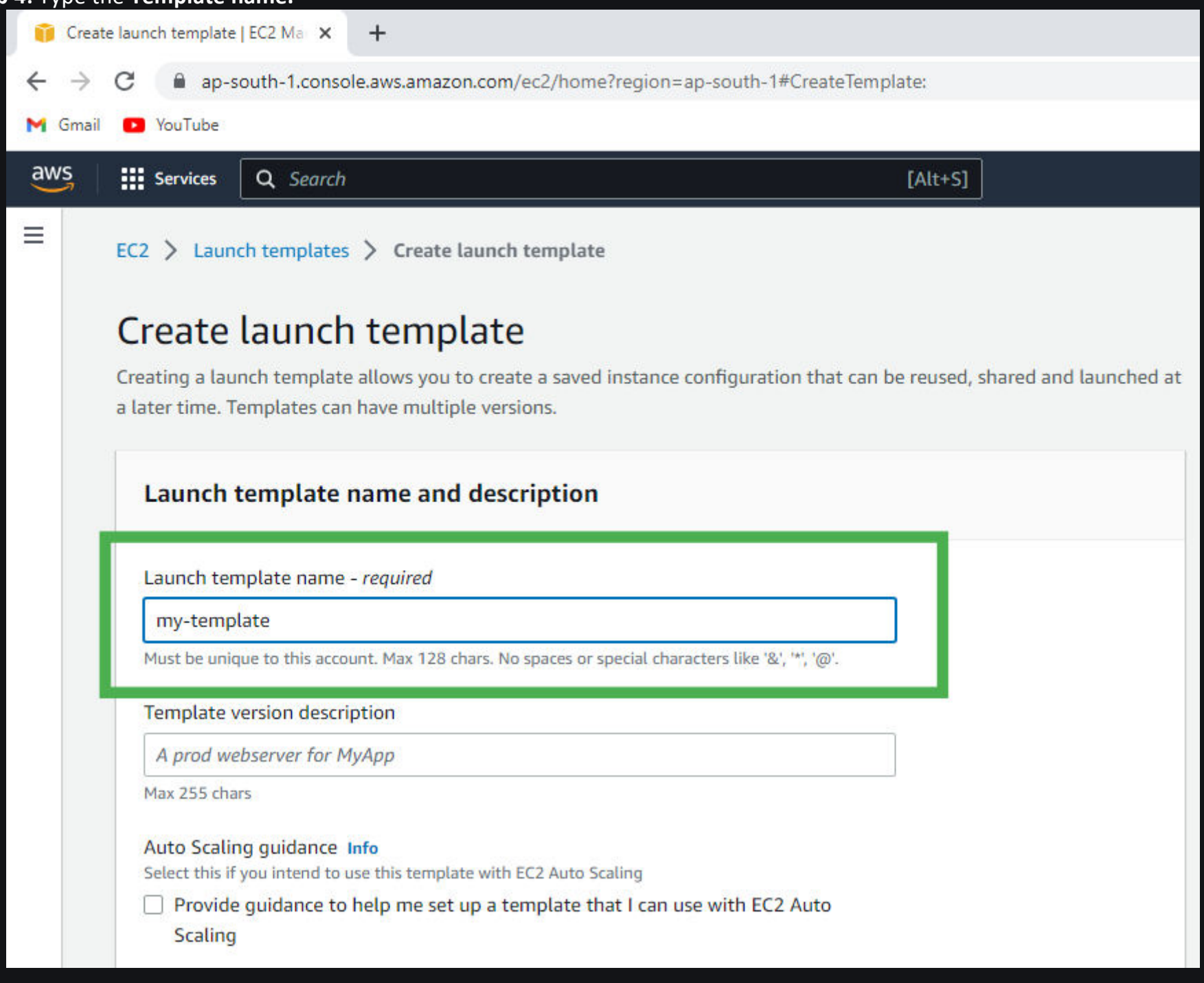
Step 1: Click on the **All Services.**
Step 2: Click on the **EC2.**



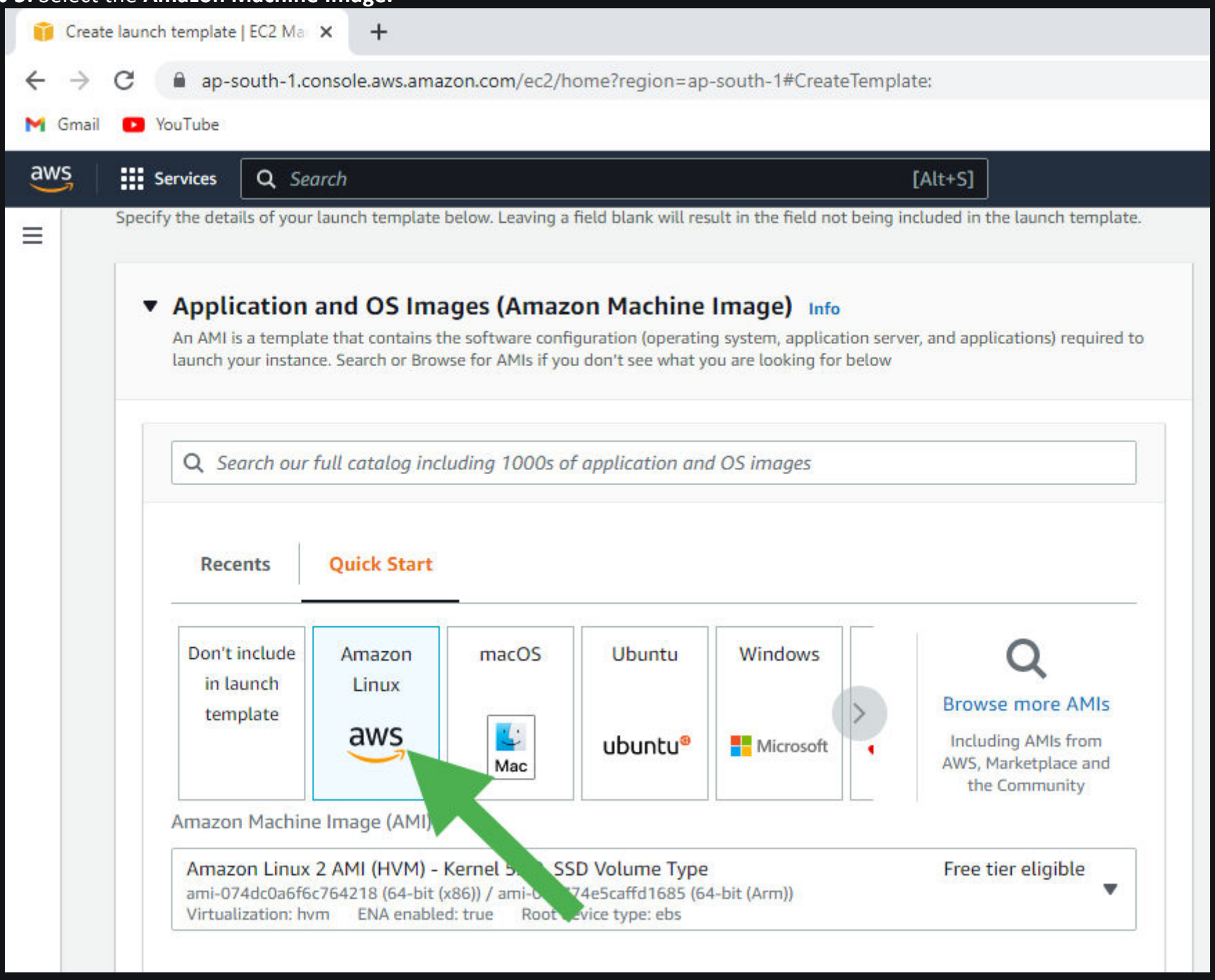
Step 3: Scroll Down and click on the **Launch Templates** and click on the **Create launch template**



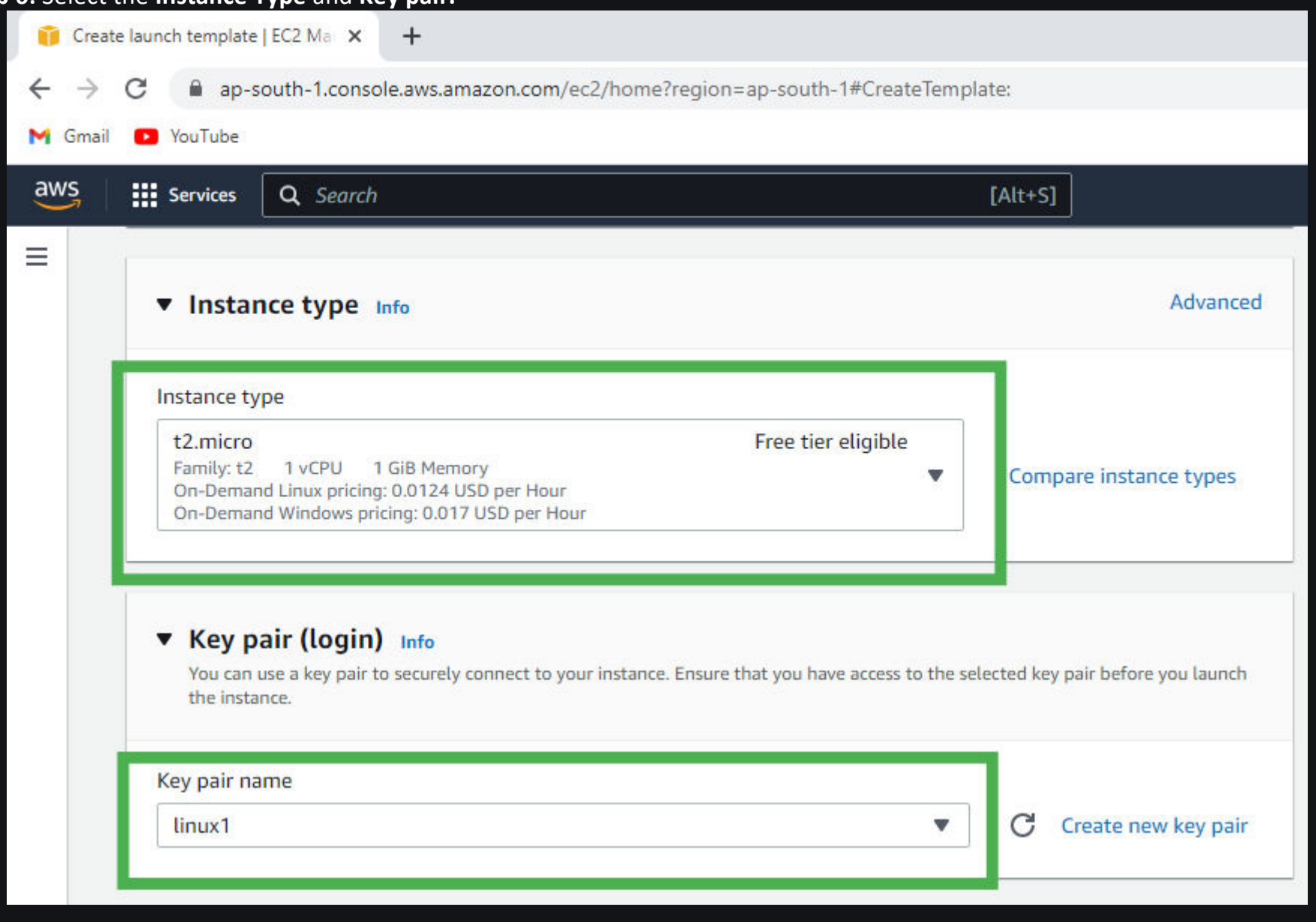
Step 4: Type the **Template name.**



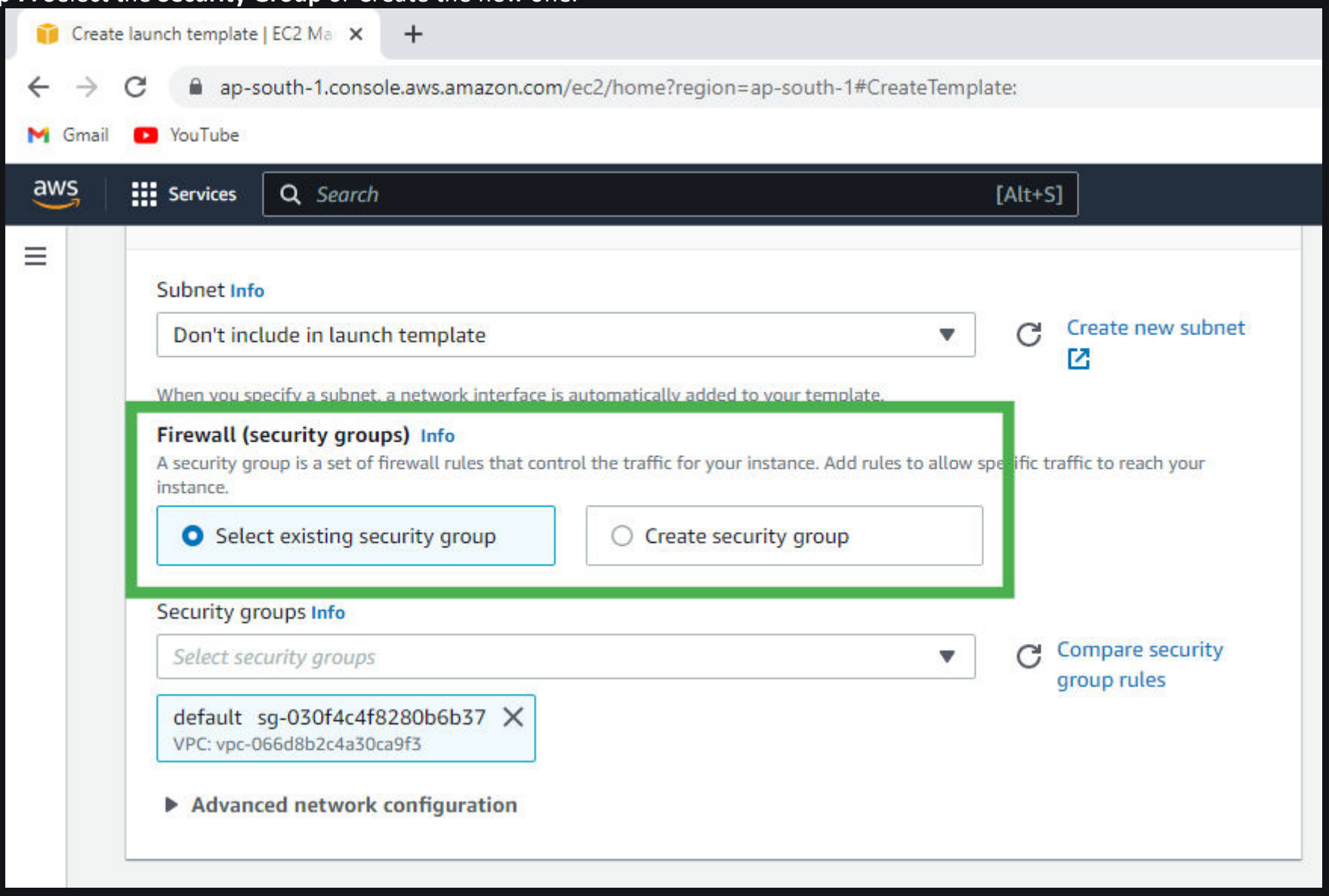
Step 5: Select the **Amazon Machine Image.**



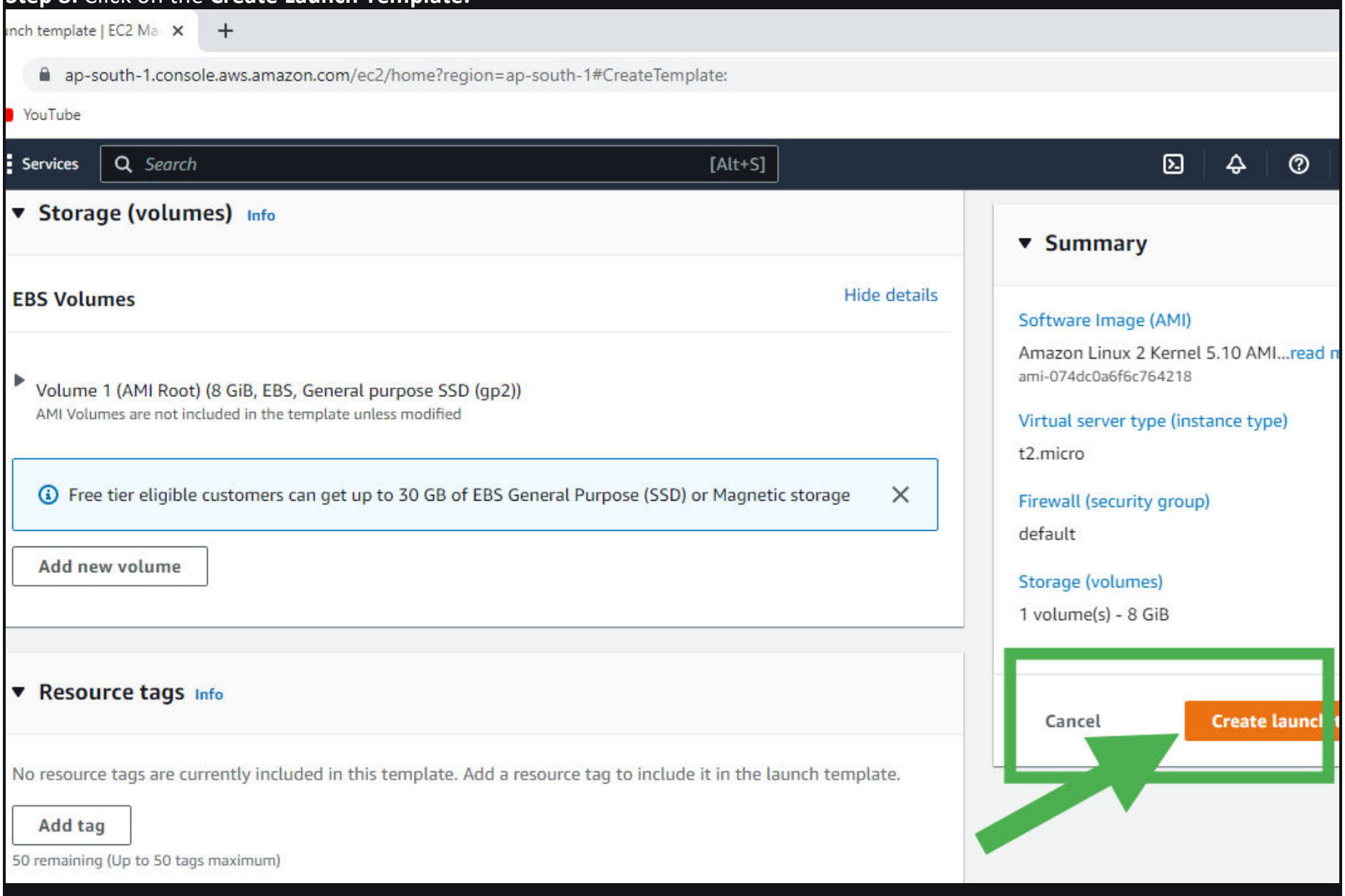
Step 6: Select the **Instance Type** and **Key pair.**



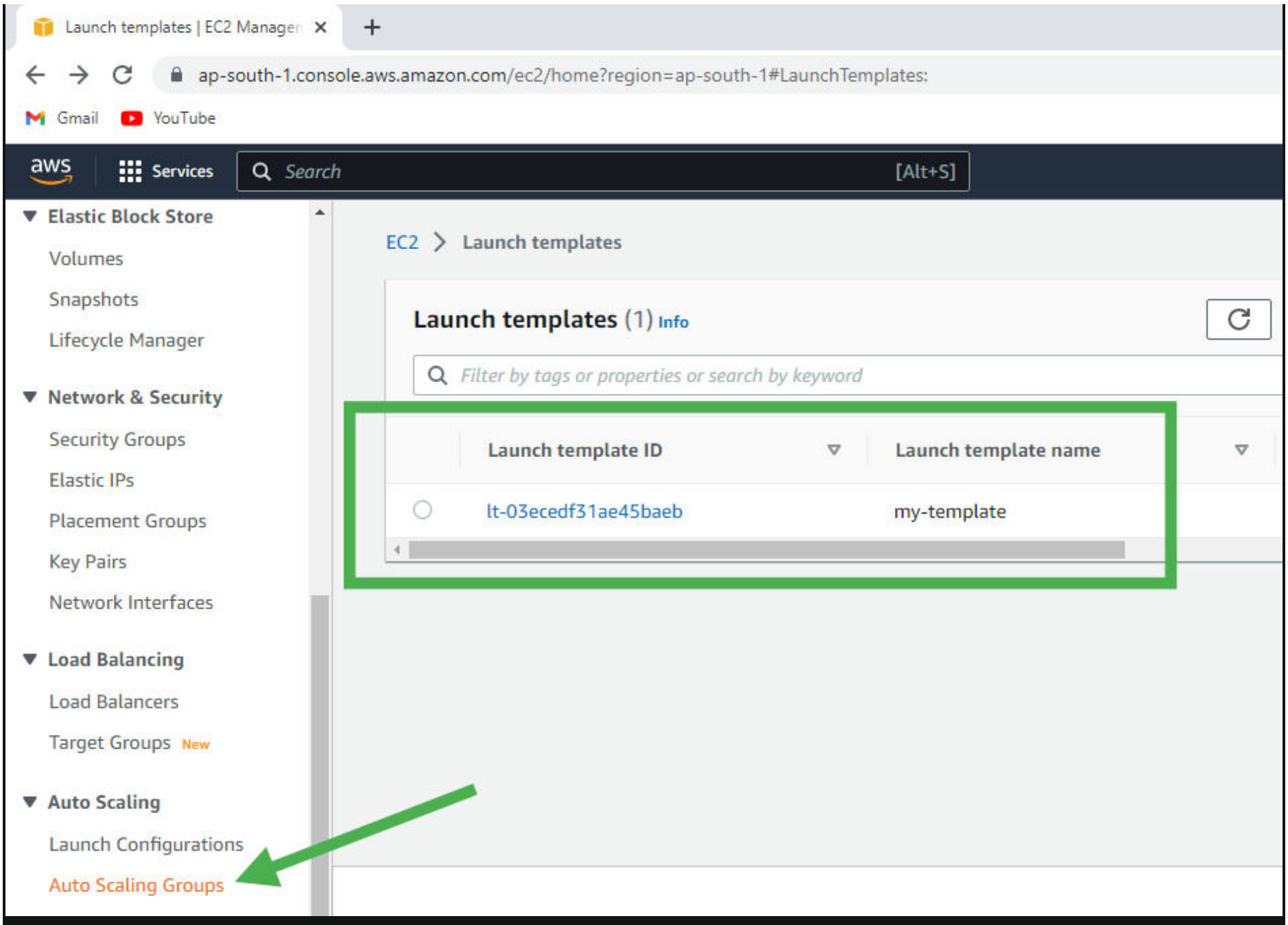
Step 7: Select the **Security Group** or Create the new one.



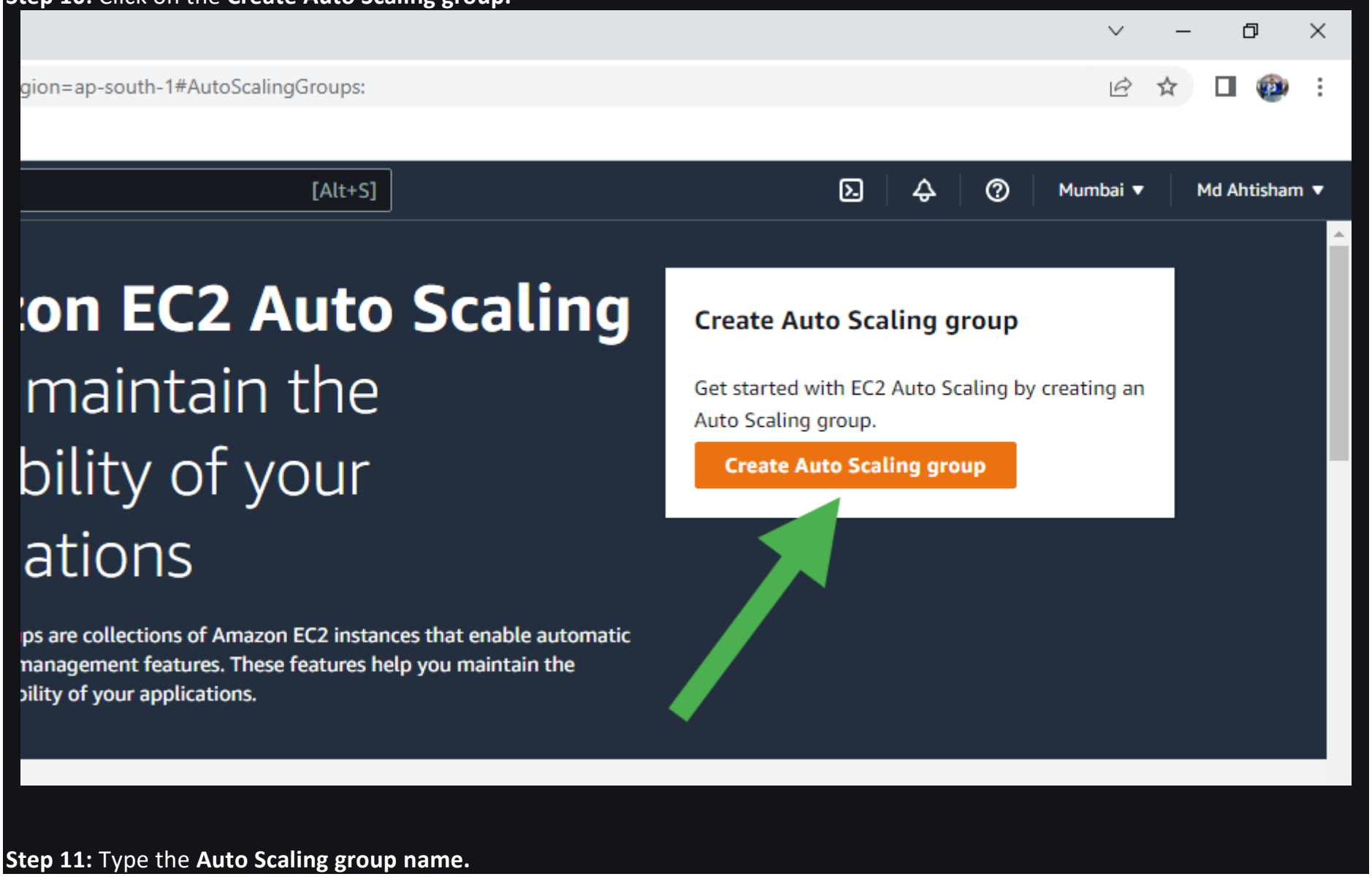
Step 8: Click on the **Create Launch Template.**



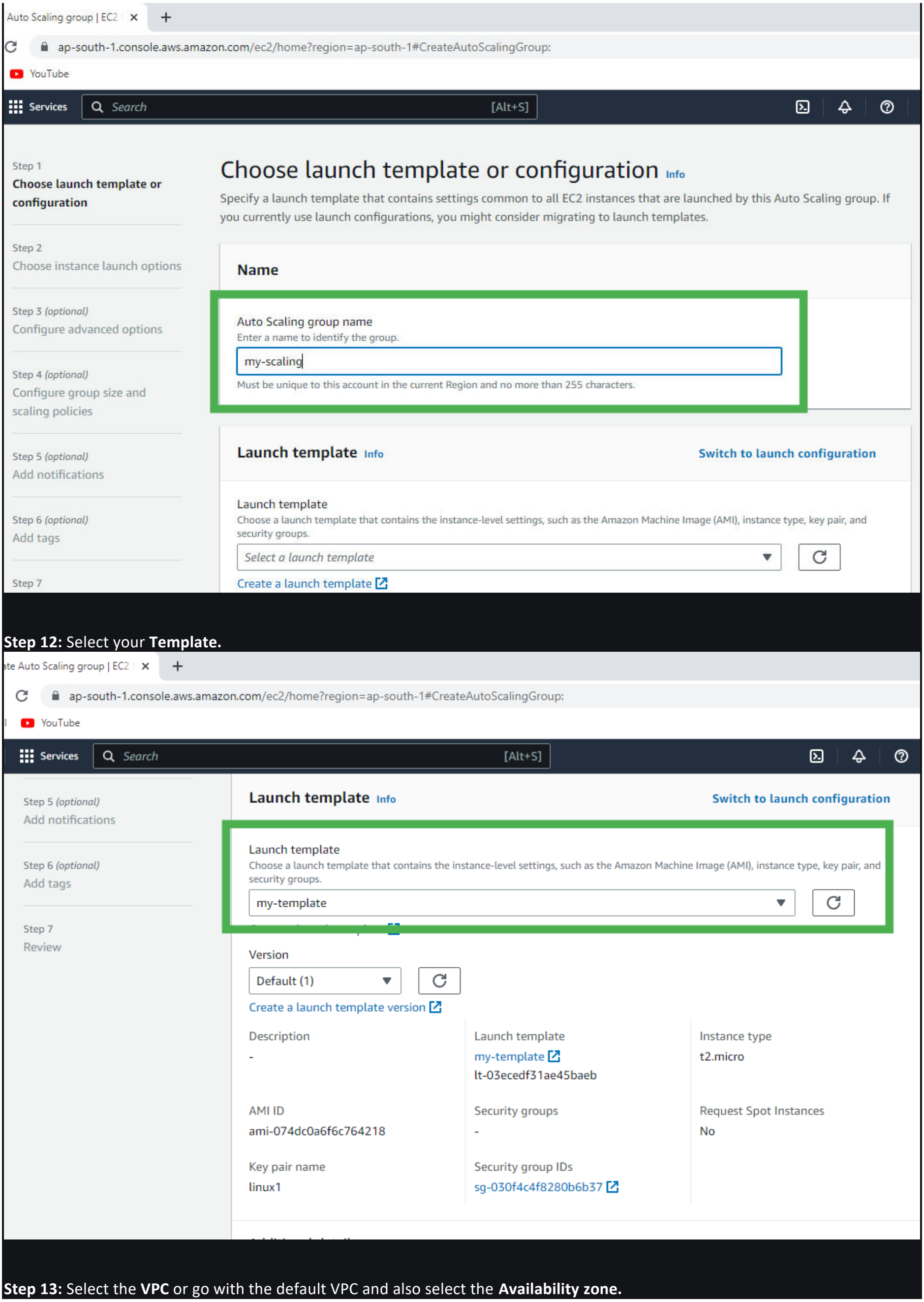
Step 9: Now you can see the template is **created.** Now, scroll down and click on the **Auto Scaling Groups.**



Step 10: Click on the **Create Auto Scaling group.**

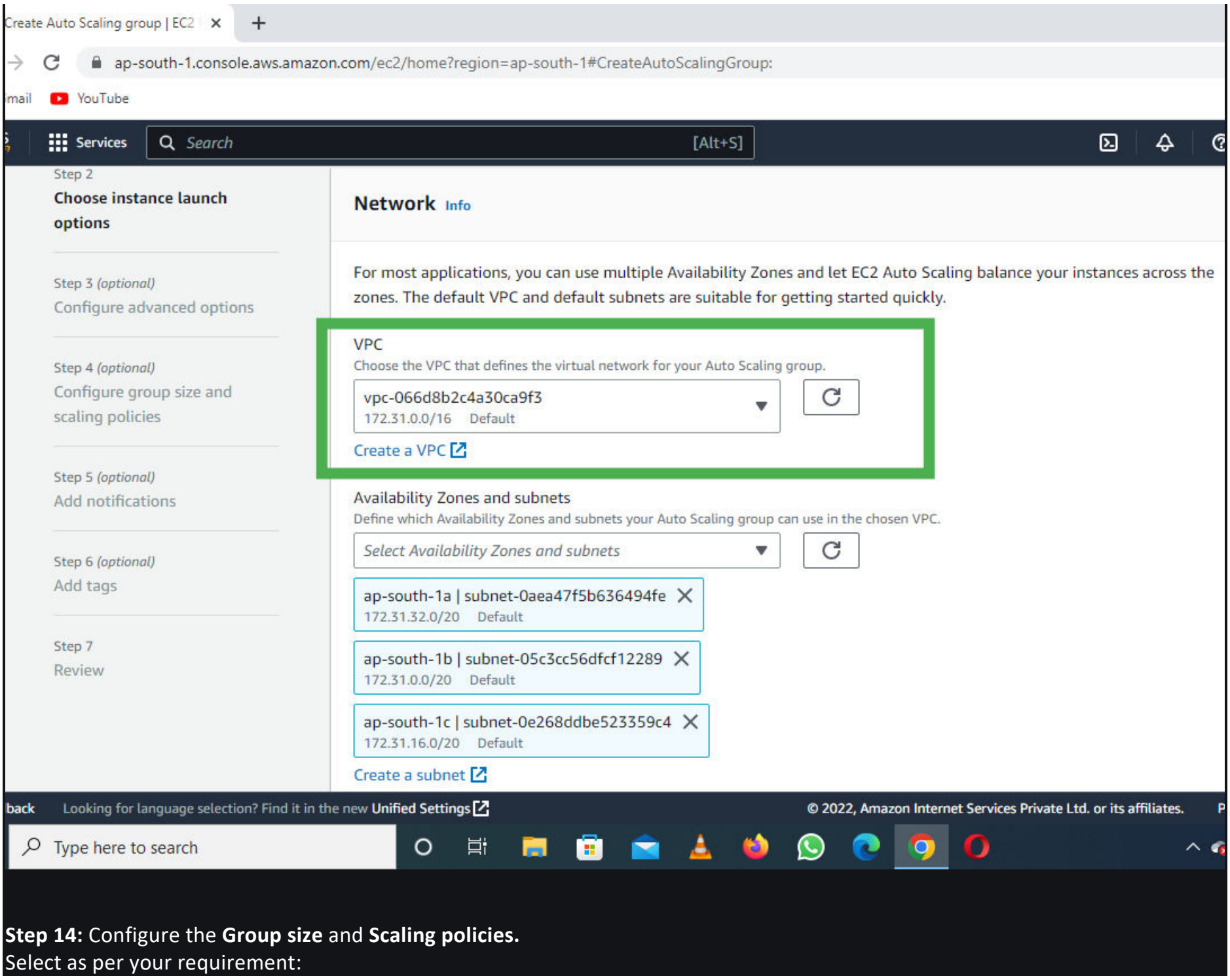


Step 11: Type the **Auto Scaling group name.**



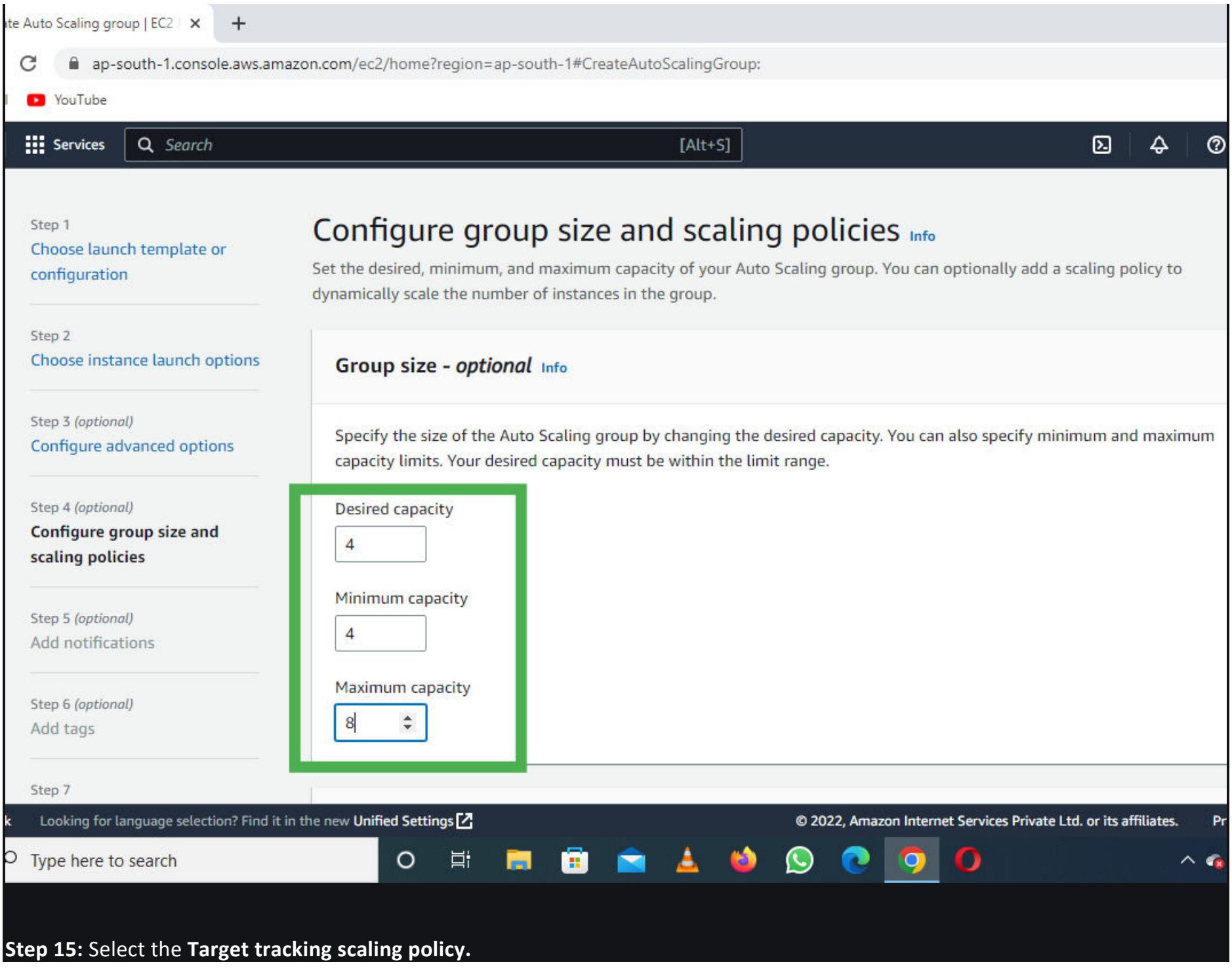
Step 12: Select your **Template.**

Step 13: Select the **VPC** or go with the default VPC and also select the **Availability zone.**

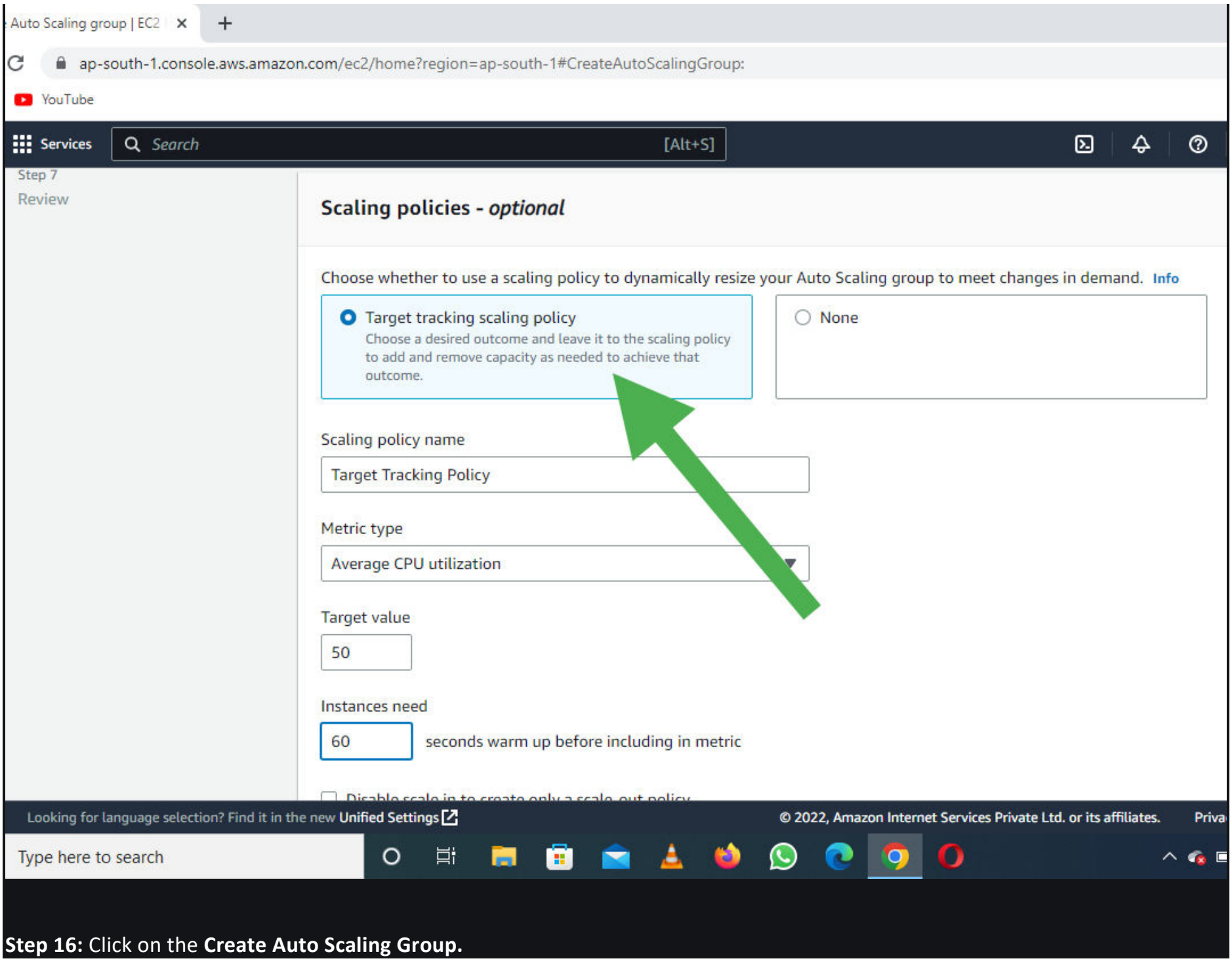


Step 14: Configure the **Group size** and **Scaling policies.**
Select as per your requirement:

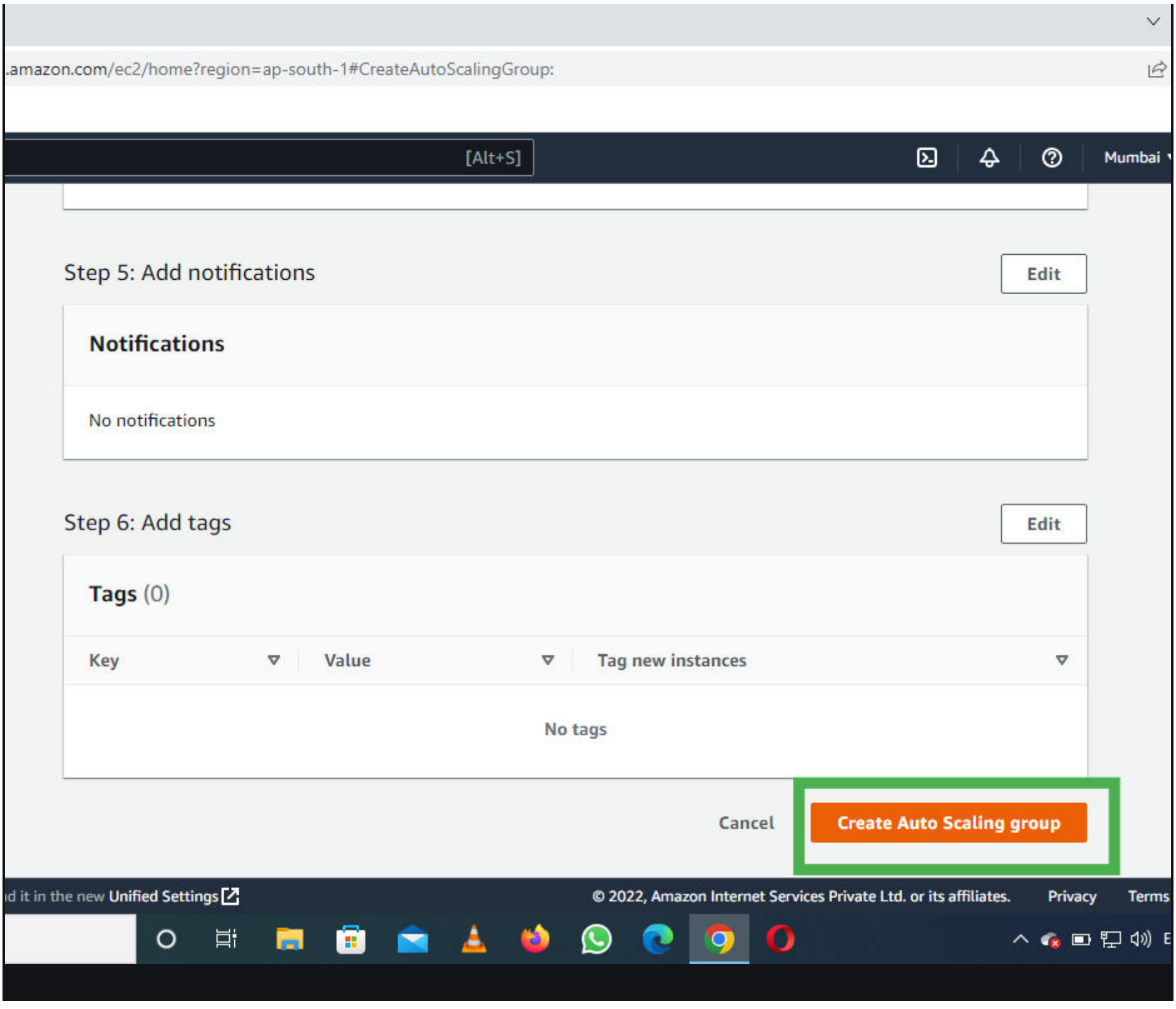
- Desired: 4
- Minimum: 4
- Maximum: 8



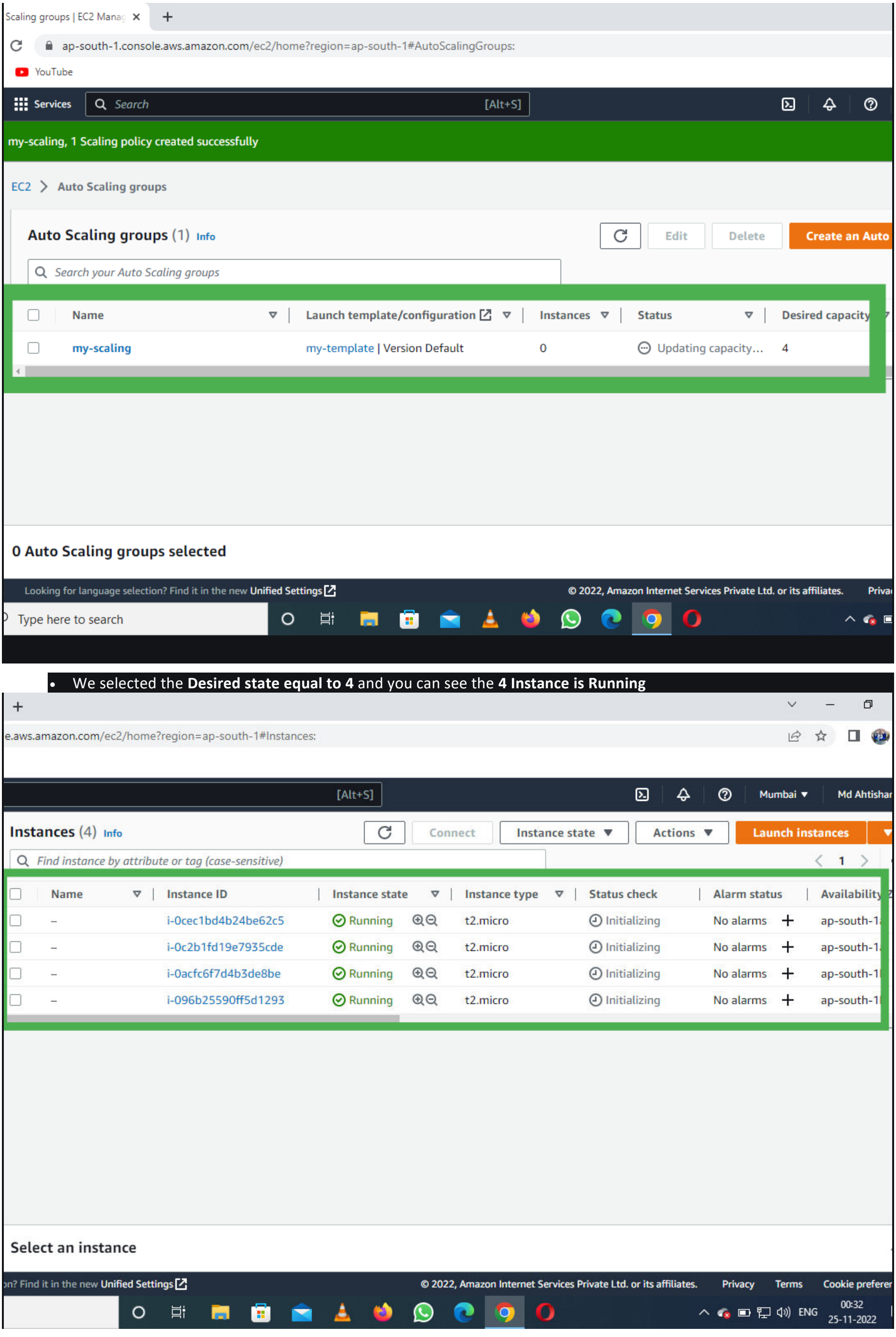
Step 15: Select the **Target tracking scaling policy.**



Step 16: Click on the **Create Auto Scaling Group.**



- Now you can see the **Auto Scaling is creating** and it is also creating the desired state of the EC2 Instance



- We selected the **Desired state equal to 4** and you can see the **4 Instance is Running**

9.Simple Storage Services (S3)

Classes & Life Cycle

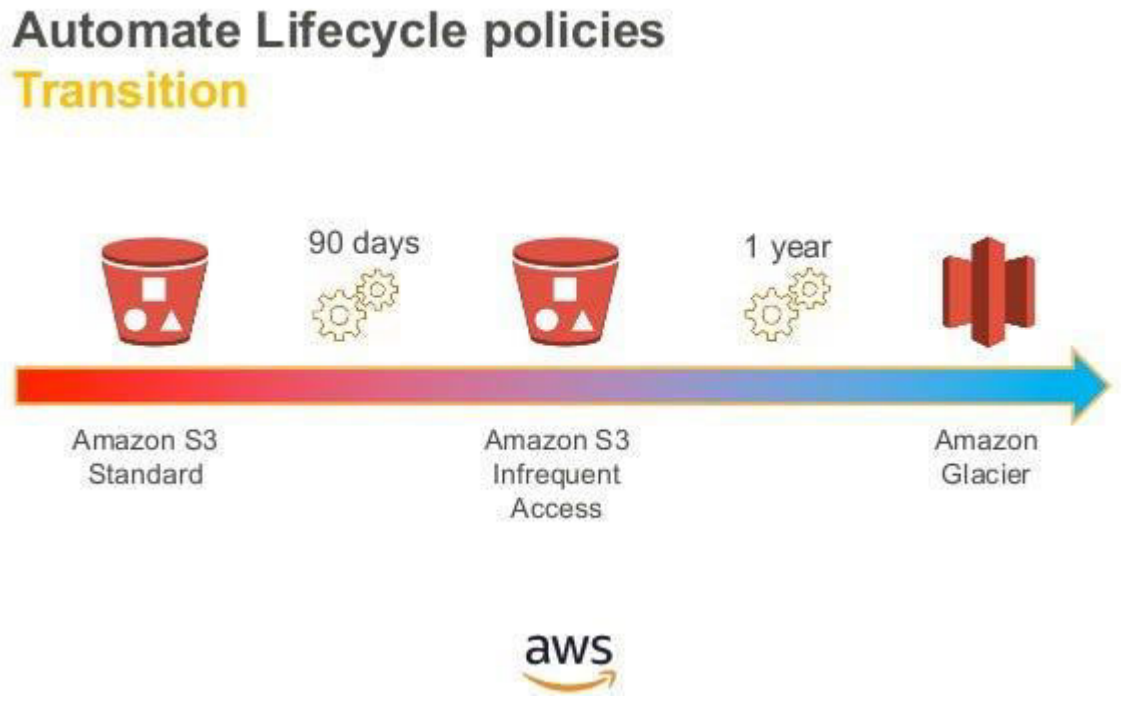
Lifecycle Management is used so that objects are stored cost-effectively throughout their lifecycle.
A **lifecycle configuration** is a set of rules that define the actions applied by S3 to a group of objects.

The lifecycle defines two types of actions:

- **Transition actions:** When you define the transition to another storage class. For example, you choose to transit the objects to the Standard IA storage class 30 days after you have created them or archive the objects to the Glacier storage class 60 days after you have created them.
- **Expiration actions:** You need to define when objects expire, the Amazon S3 deletes the expired object on your behalf.

Suppose a business generates a lot of data in the form of test files, images, audio, or videos and the data is relevant for 30 days only. After that, you might want to transition from standard to standard IA as storage cost is lower. After 60 days, you might want to transit to Glacier storage class for the longtime archival. Perhaps you want to expire the object after 60 days completely, so Amazon has a service known as Lifecycle Management, and this service exist within the S3 bucket.

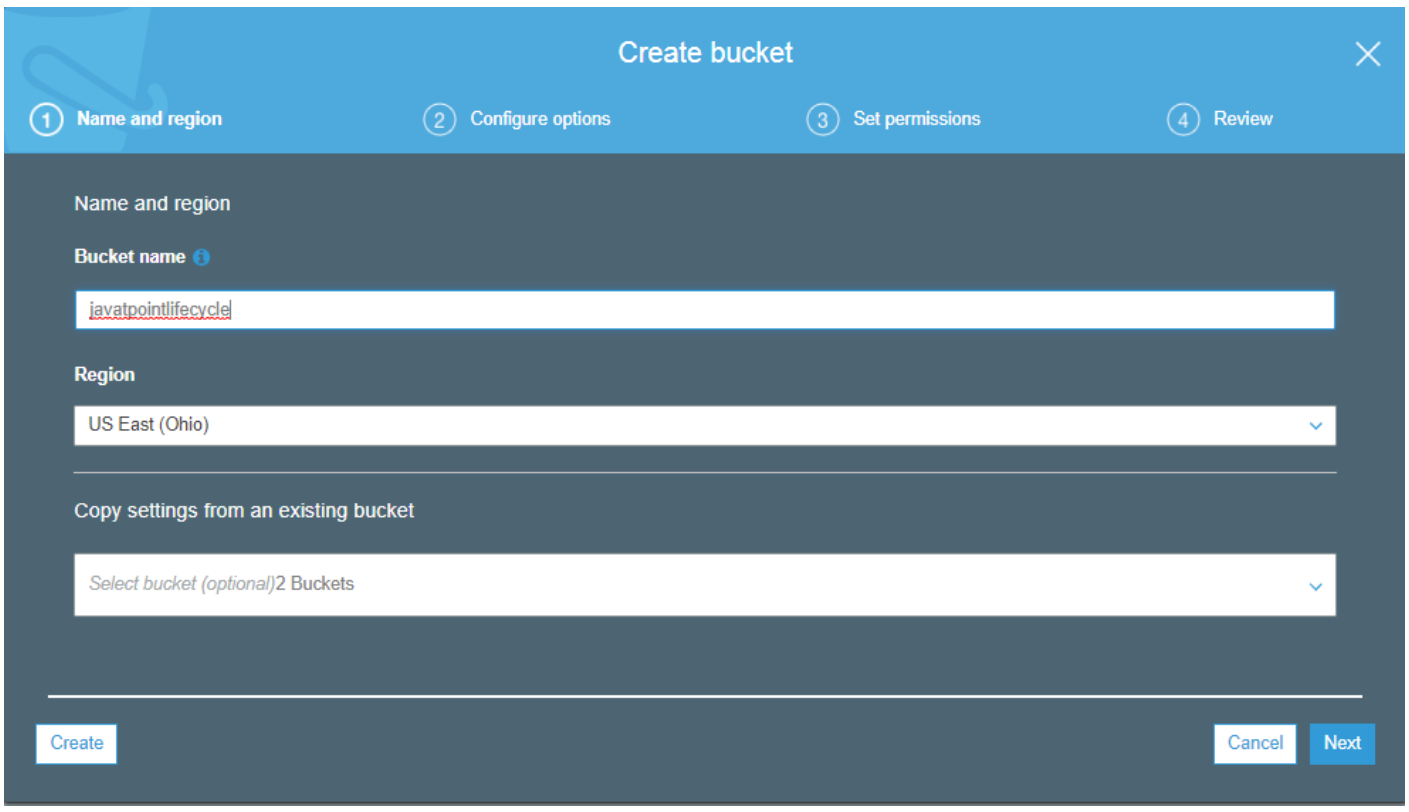
Lifecycle policies:



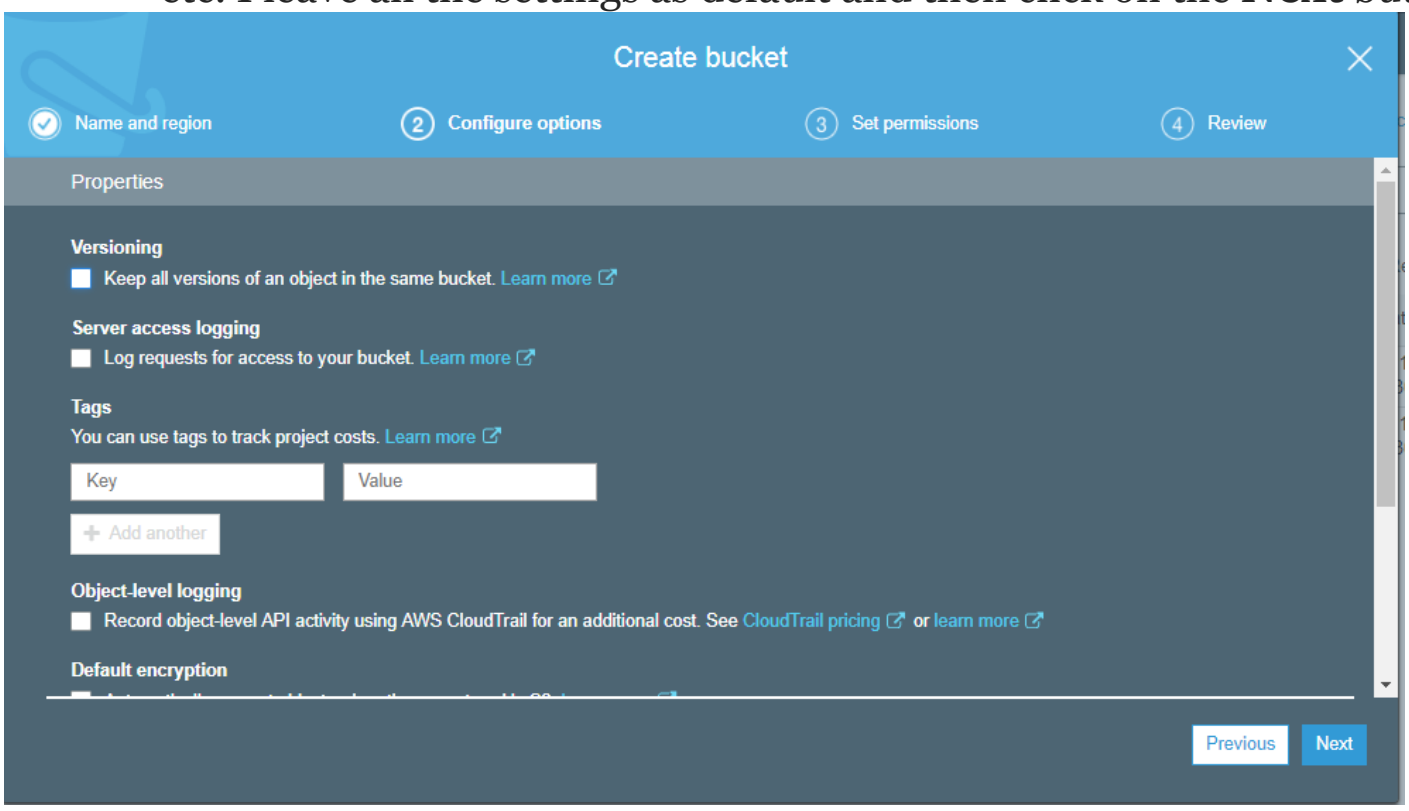
- **Use Lifecycle rules to manage your object:** You can manage the Lifecycle of an object by using a Lifecycle rule that defines how Amazon S3 manages objects during their lifetime.
- **Automate transition to tiered storage:** Lifecycle allows you to transition objects to the Standard IA storage class automatically and then to the Glacier storage class.
- **Expire your objects:** Using the Lifecycle rule, you can automatically expire your objects.

Creation of Lifecycle rule

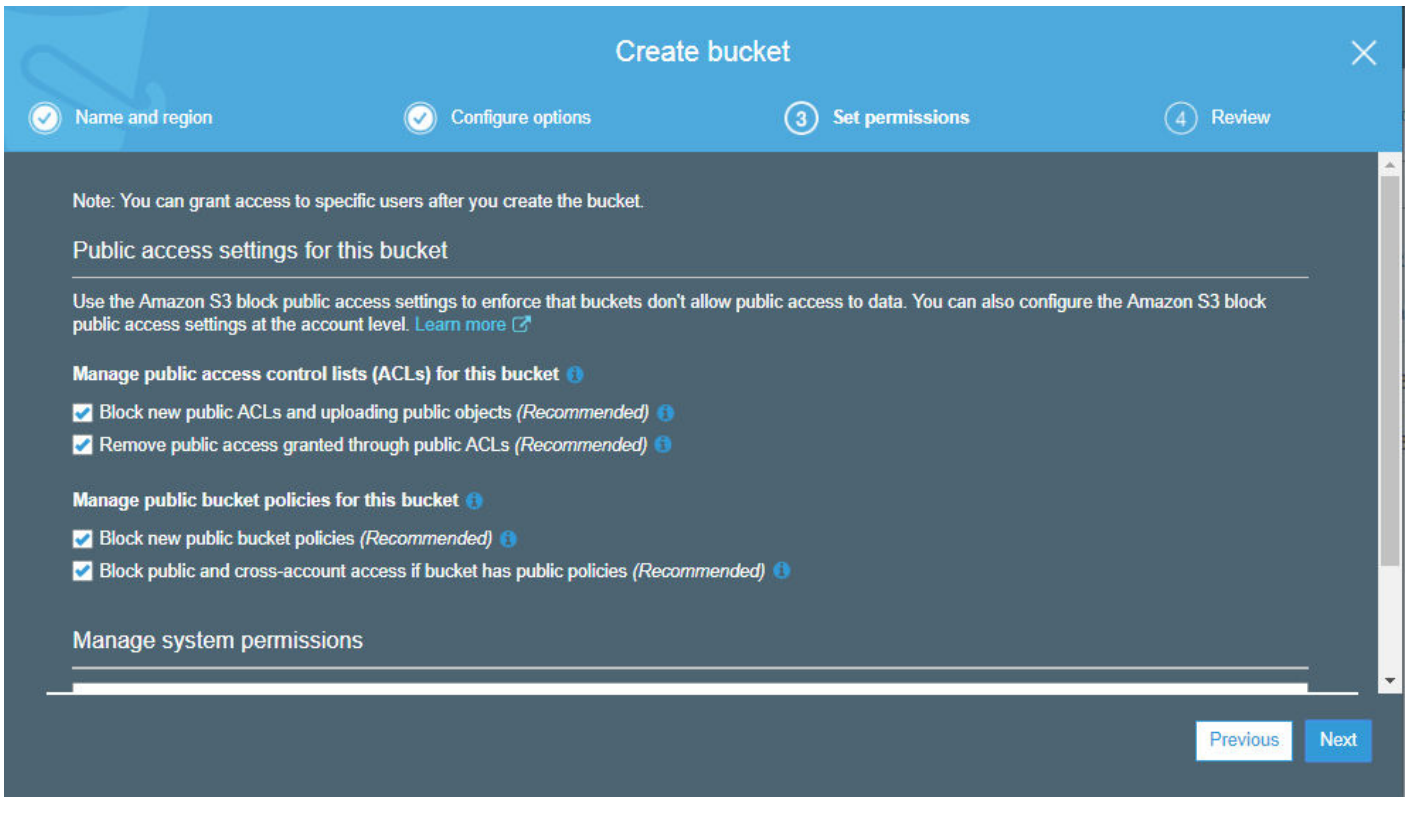
- Sign in to the AWS Management console.
- Click on the **S3** service
- Create a new bucket in S3.
- Enter the bucket name and then click on the **Next** button.



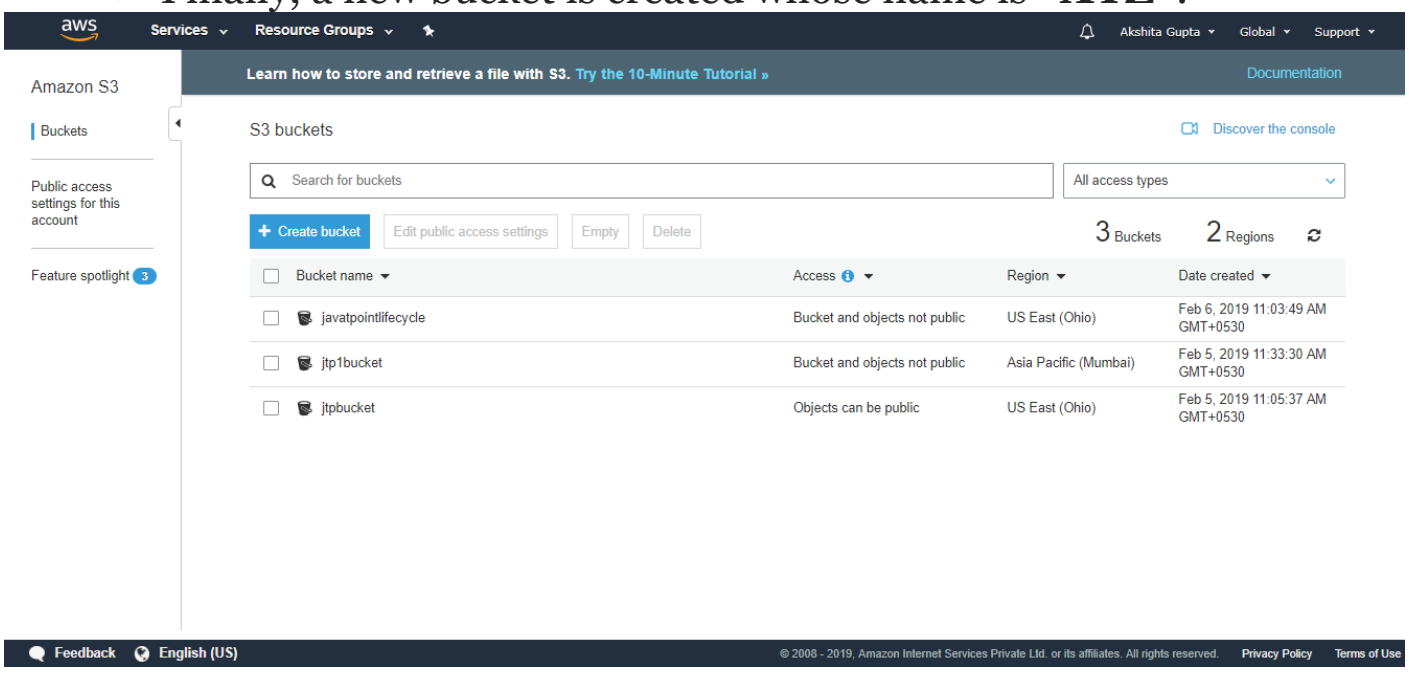
- Now, you can configure the options, i.e., you can set the versioning, server access logging, etc. I leave all the settings as default and then click on the **Next** button.



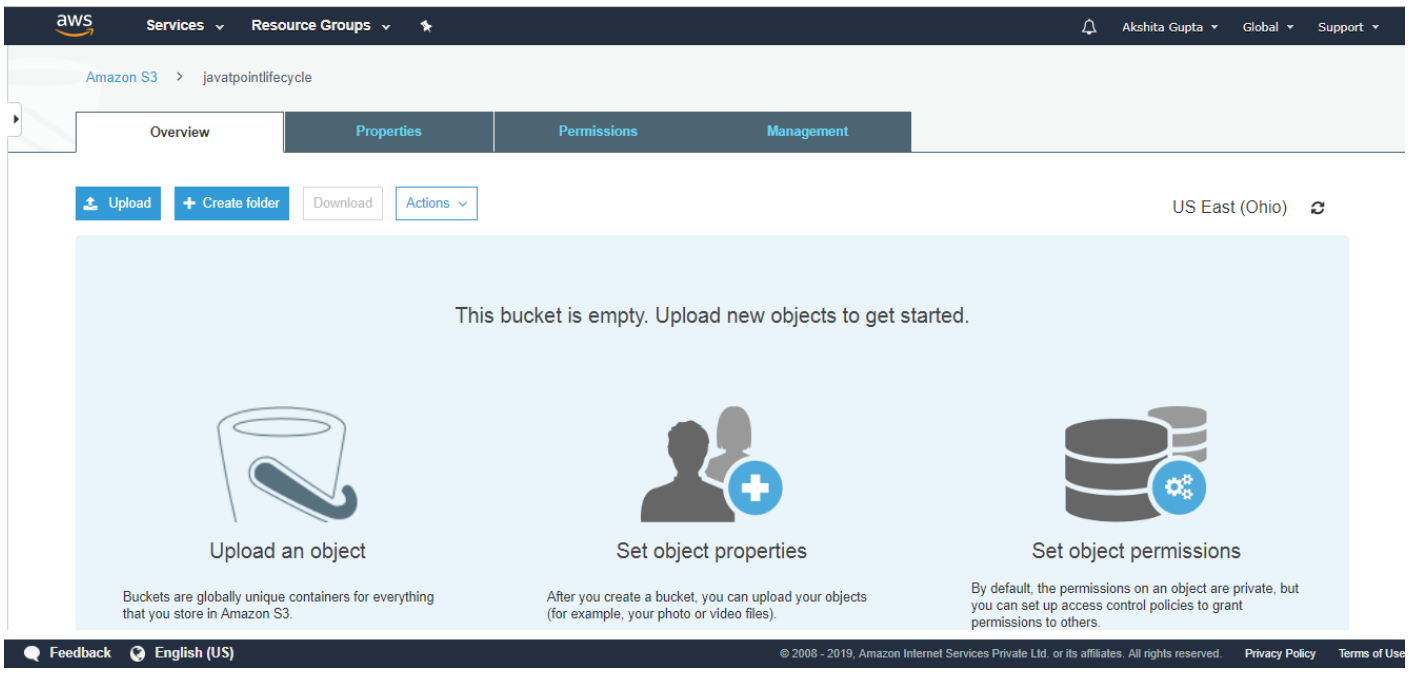
- Set the permissions. I leave all the permissions as default and then click on the **Next** button.



- Click on the **Create bucket** button.
- Finally, a new bucket is created whose name is "**XYZ**".

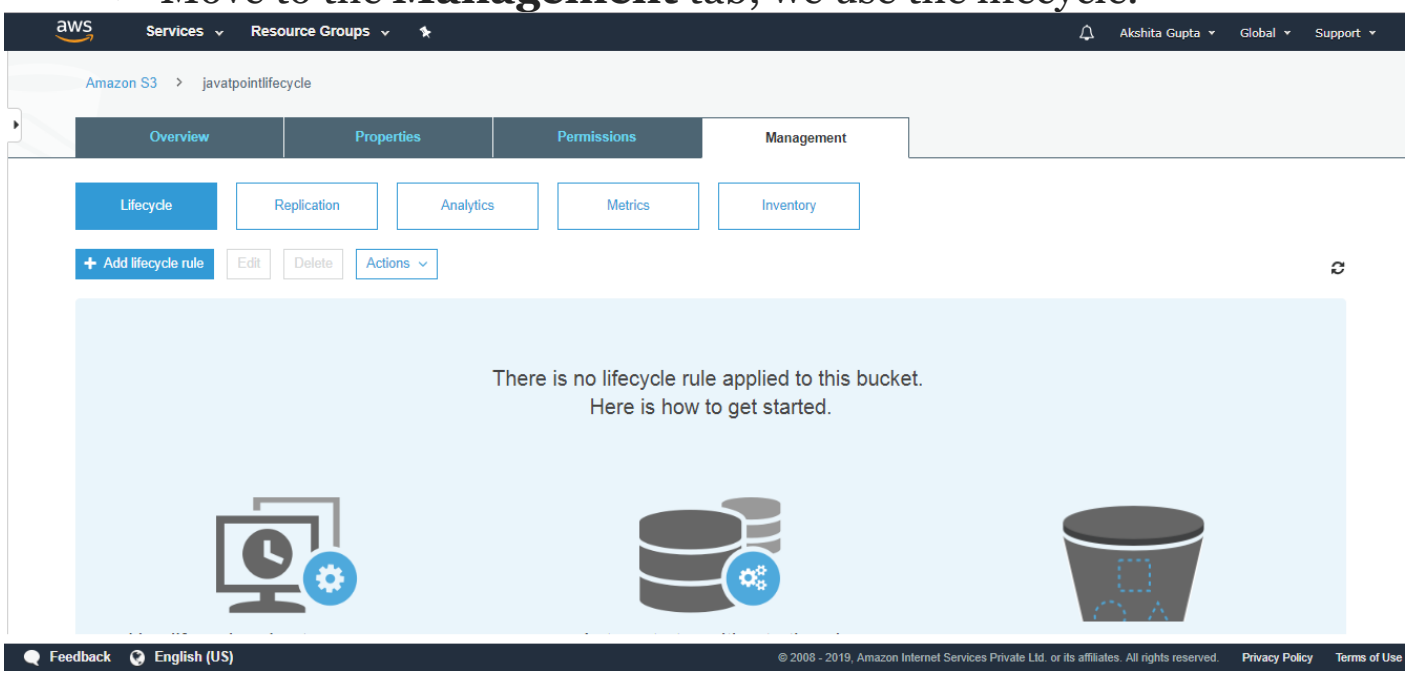


- Click on the XYZ bucket.

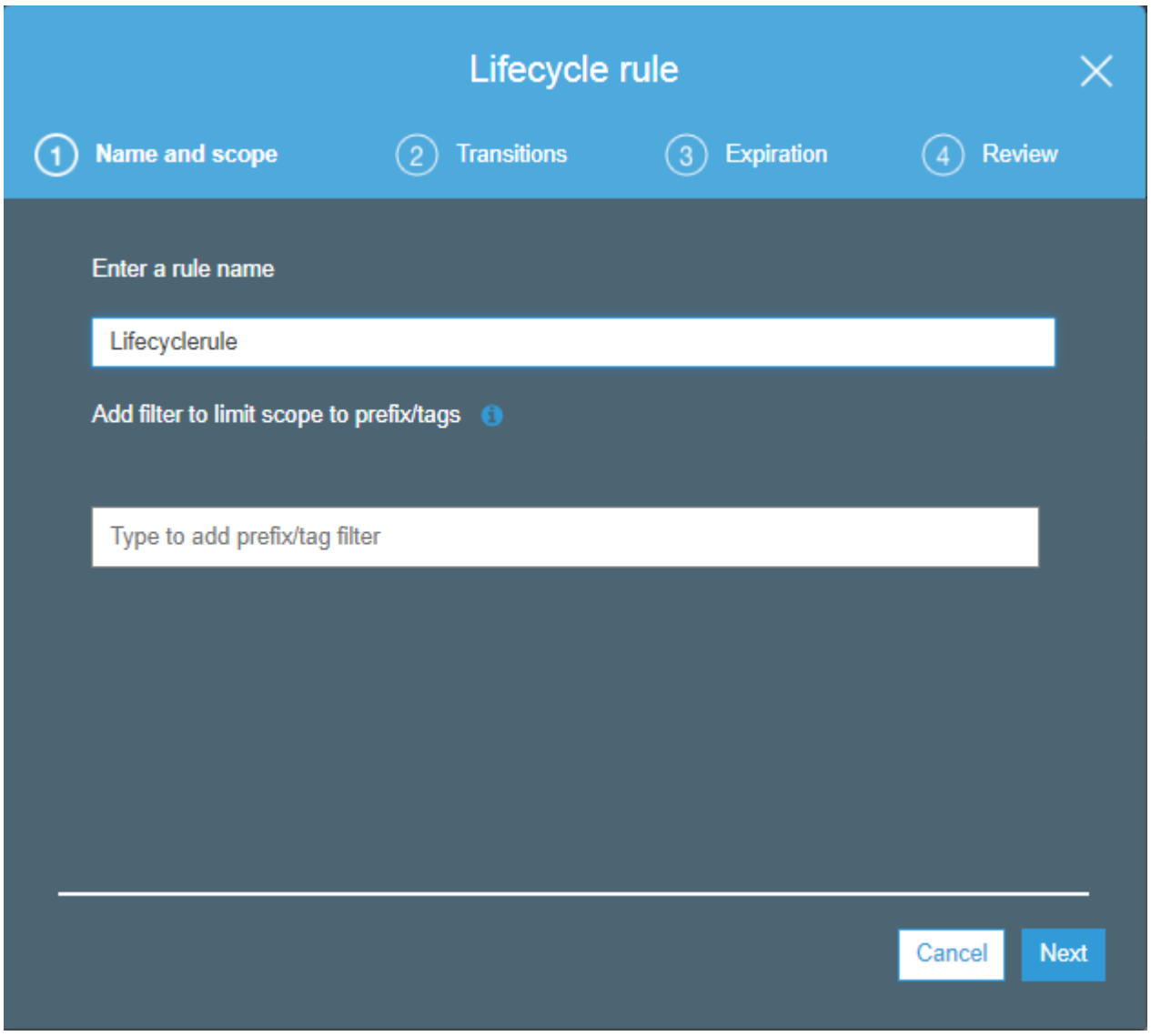


From the above screen, we observe that the bucket is empty. Before uploading the objects in a bucket, we first create the policy.

- Move to the **Management** tab; we use the lifecycle.

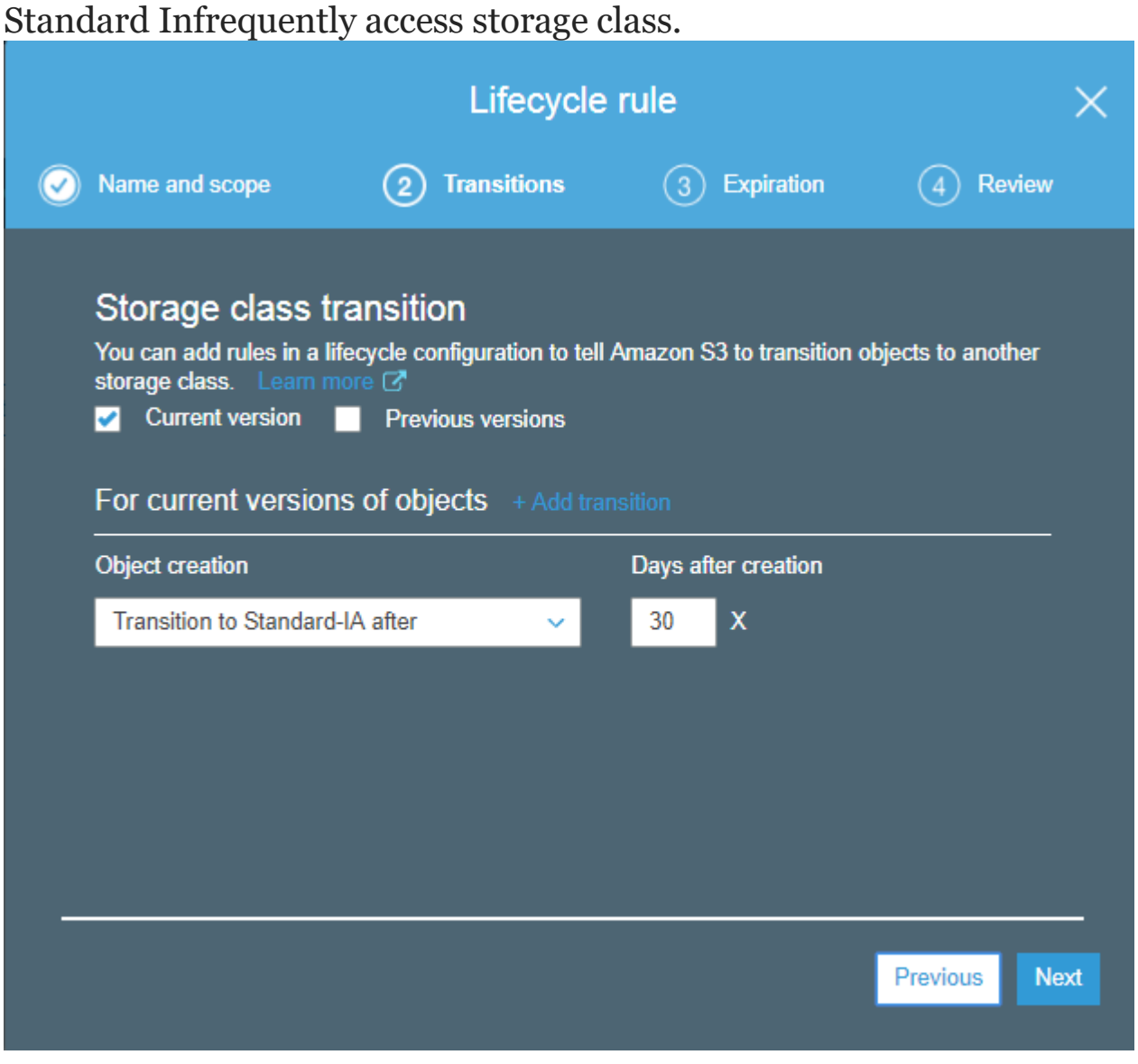


- Add the Lifecycle rule and then enter the rule name. Click on the **Next**.

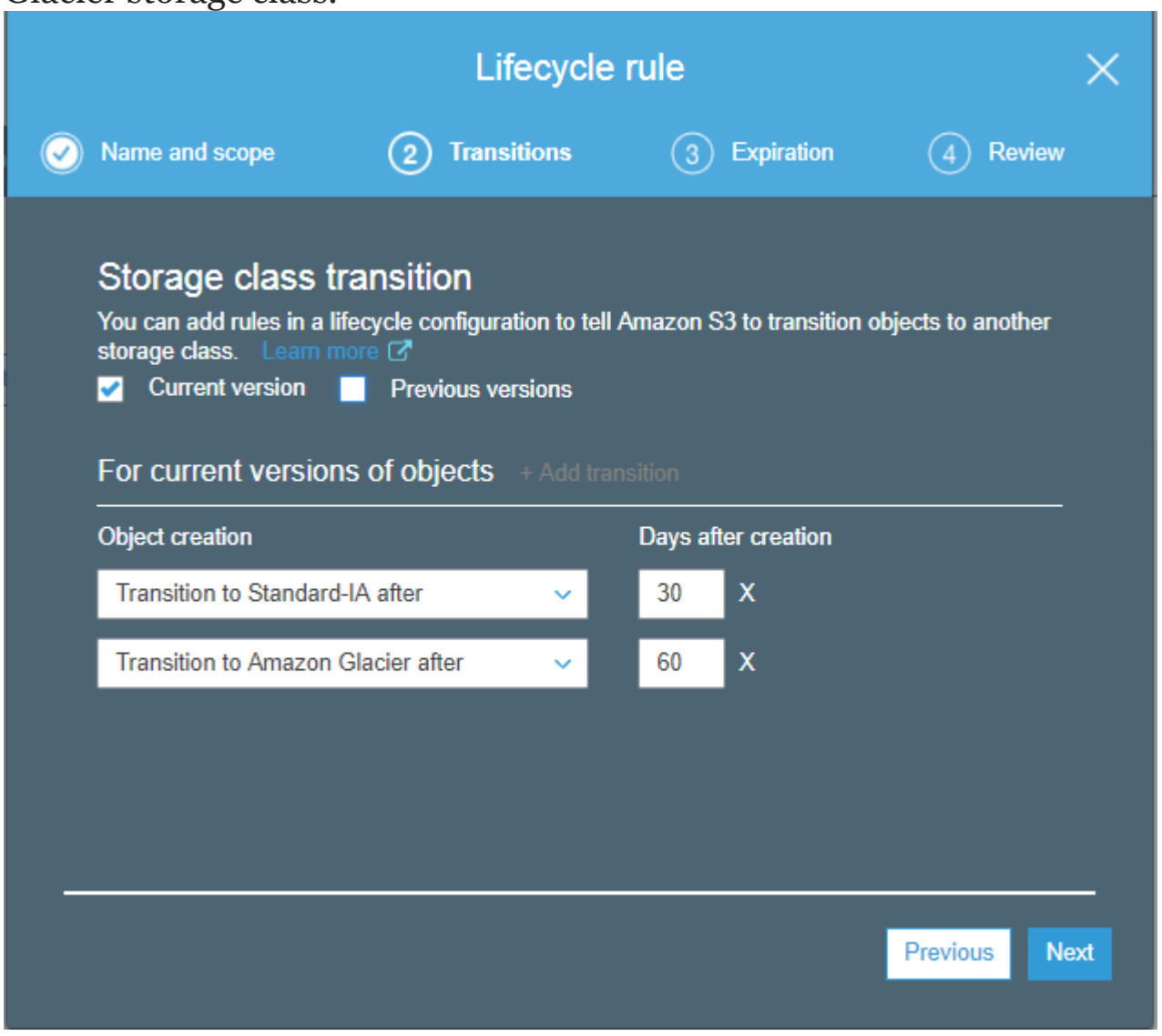


- You can create the storage class transition in both the current version and the previous version. Initially, I create the transition in the current version. Check the **current version** and then click on the **Add transition**.

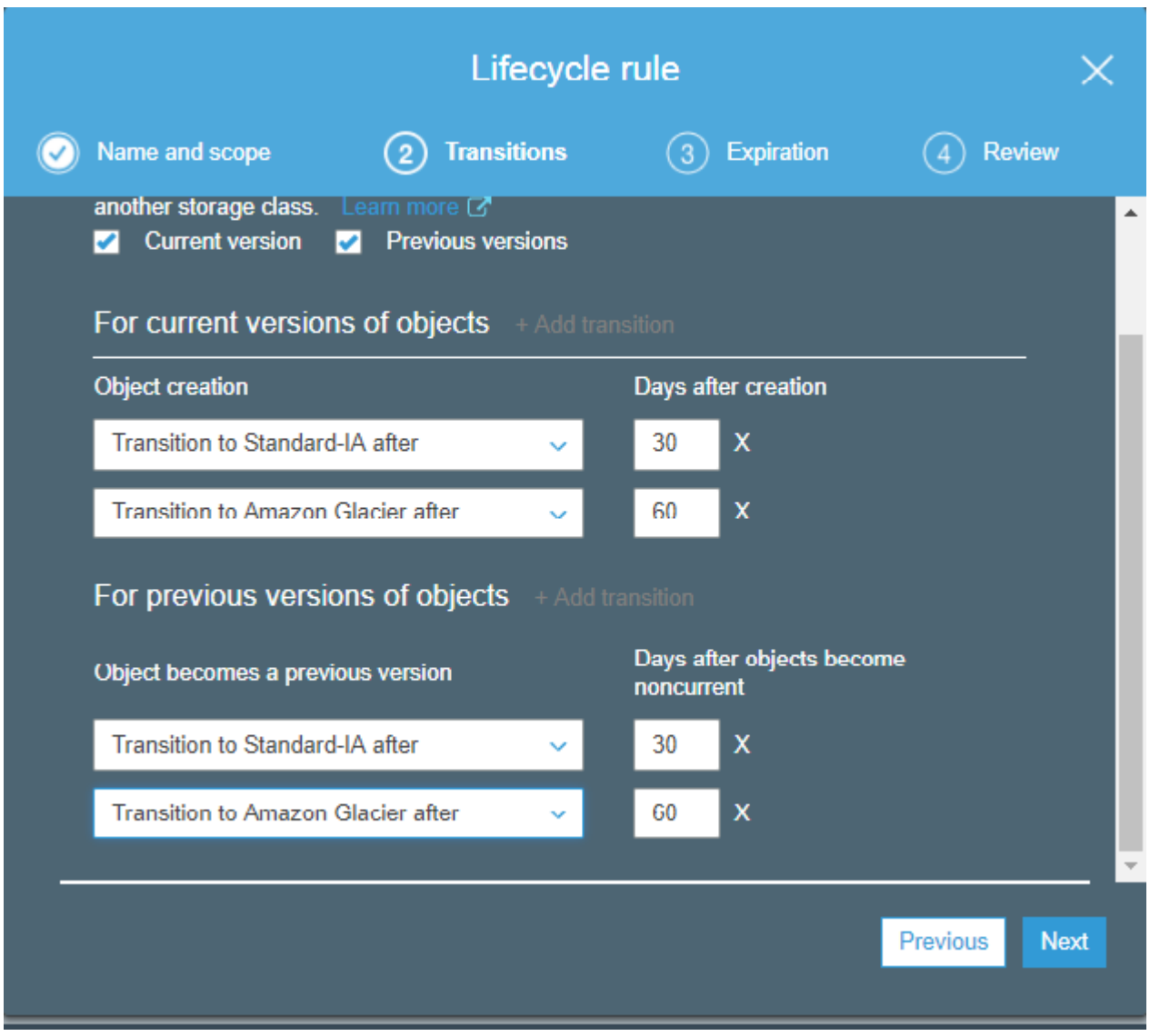
First transition: 30 days after the creation of an object, the object's storage class is converted to the Standard Infrequently access storage class.



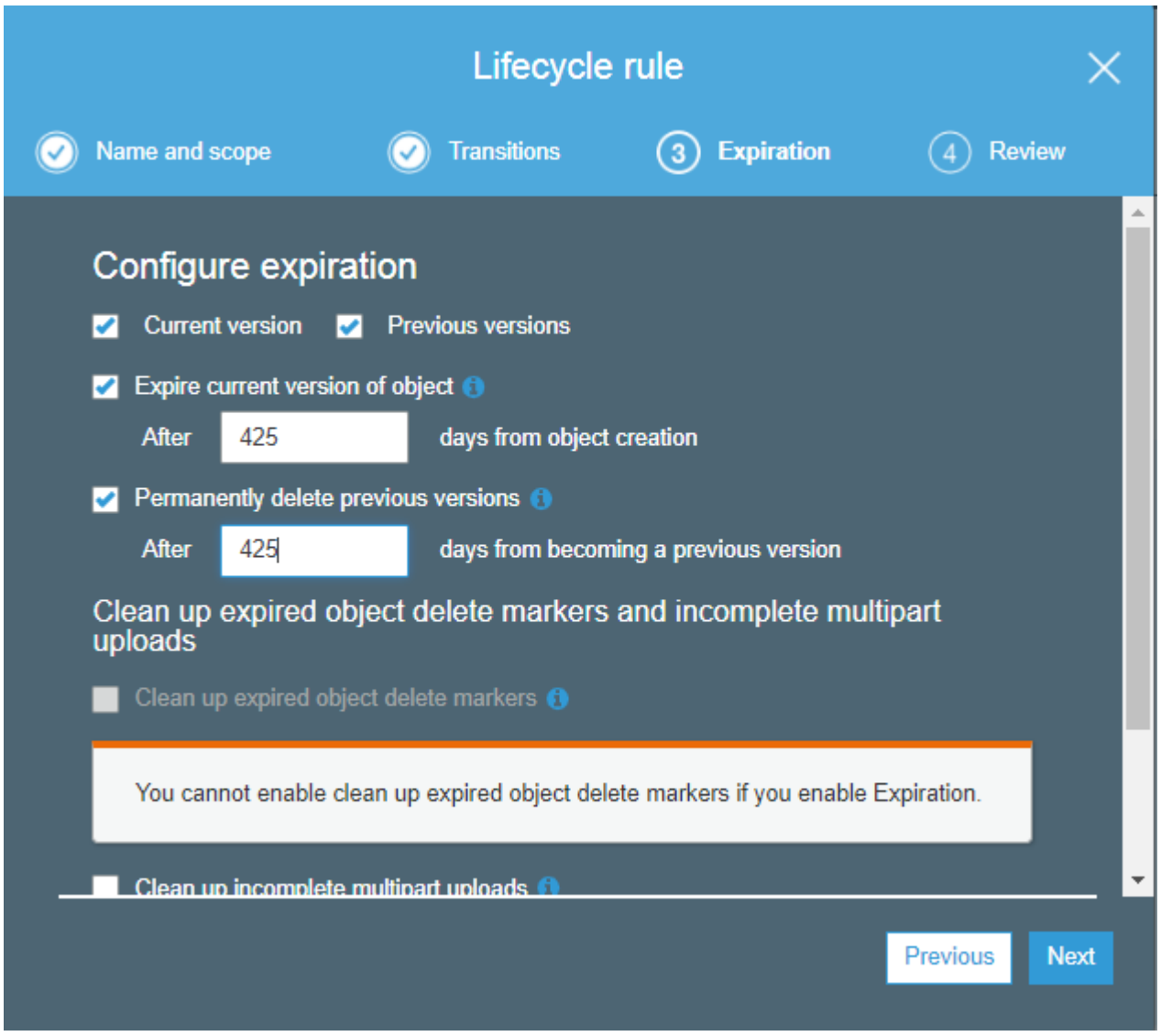
Second transition: 60 days after the creation of an object, the object's storage class is converted to Glacier storage class.



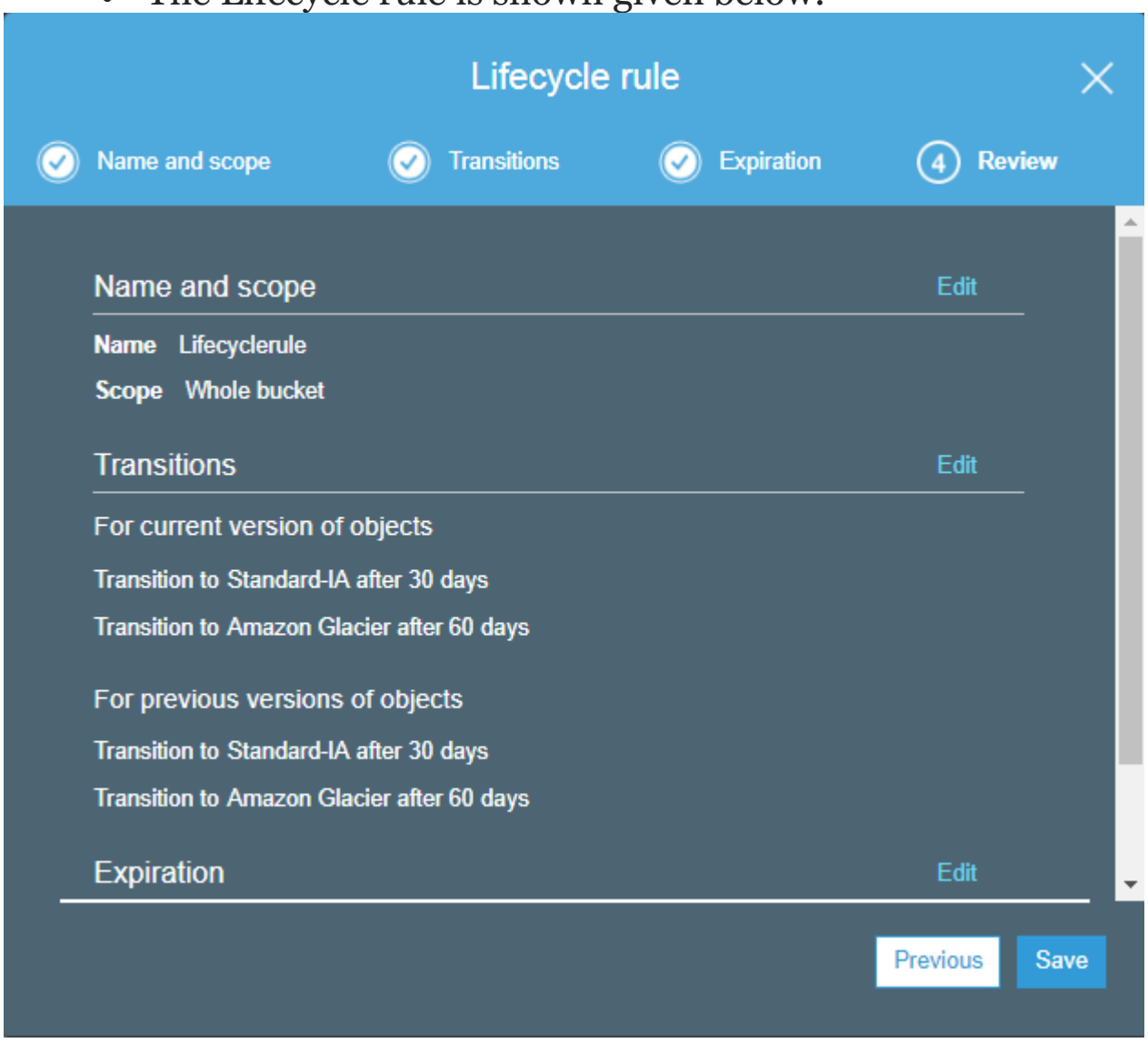
- Similarly, we can do with the previous version objects. Check the **"previous version"** and then **"Add transitions"**. Click on the **Next**.



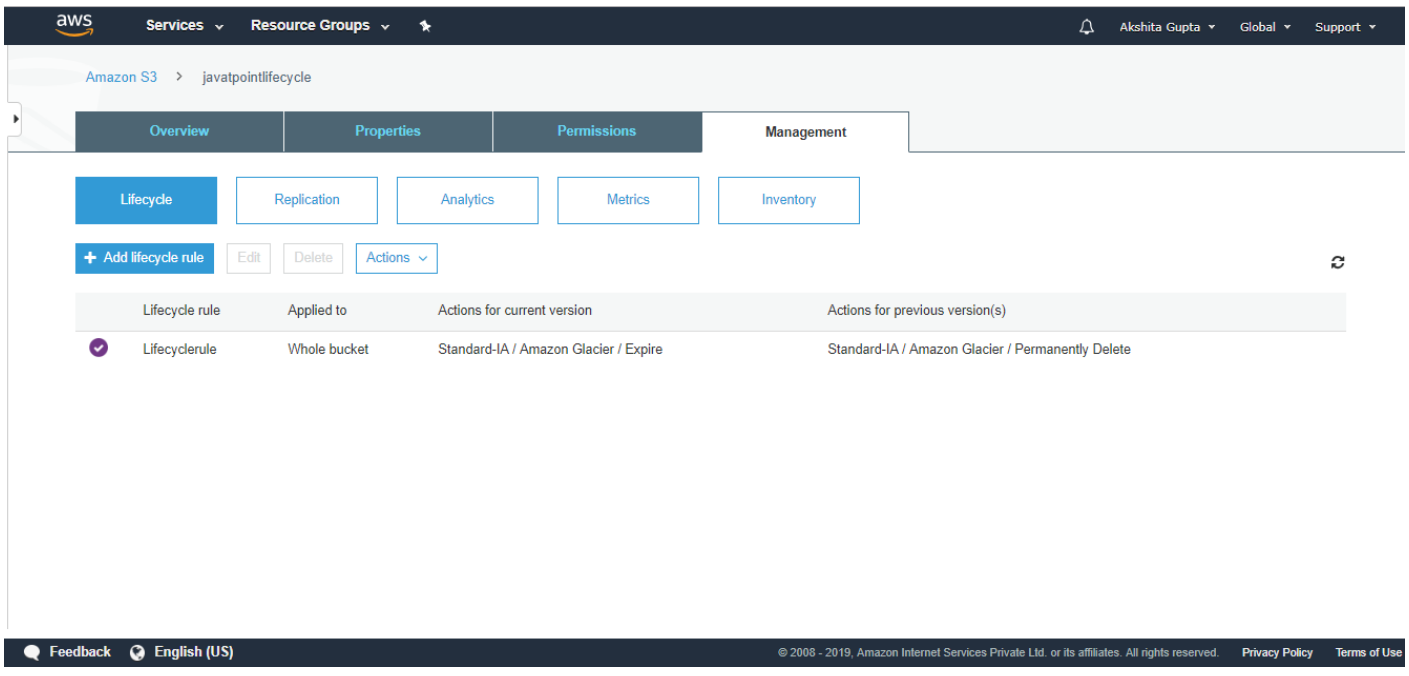
- Now, we expire the object after its creation. Suppose we expire the current and previous version objects after 425 days of their creation. Click on the **Next**.



- The Lifecycle rule is shown given below:



- Click on the **Save**.



The above screen shows that **"Lifecyclerule"** has been created.

Creating and Deleting Buckets

Step 1

Sign in to the AWS Management Console using your credentials. Once you login successfully, you will be displayed the home page of AWS Management Console.

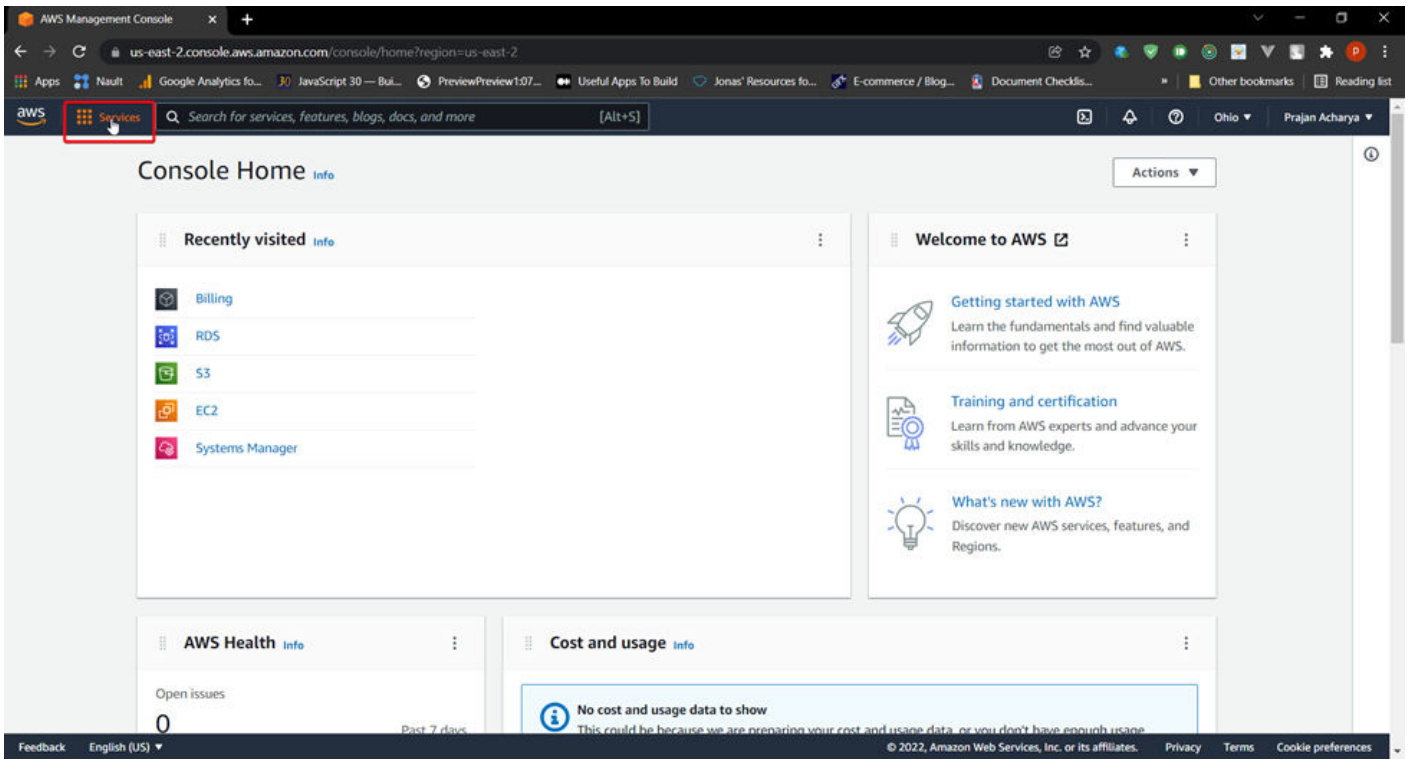
For this step, having an AWS account is mandatory. If you don't have an AWS account till now, please create one. For that, you can refer to my article on creating an AWS account, using th link below:

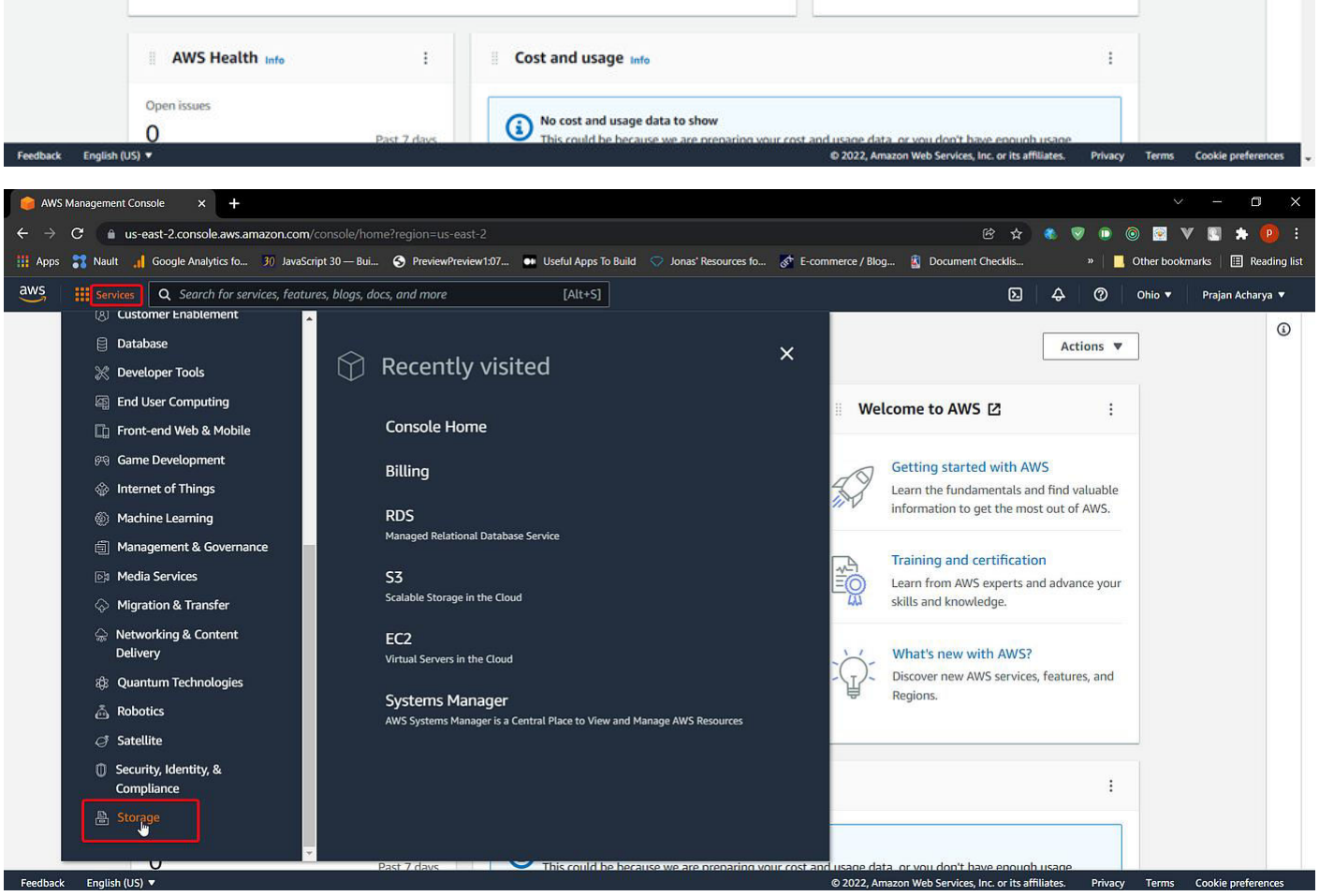
How to Create an AWS Account
A step-by-step guide for creating an AWS Account.
aws.plainenglish.io

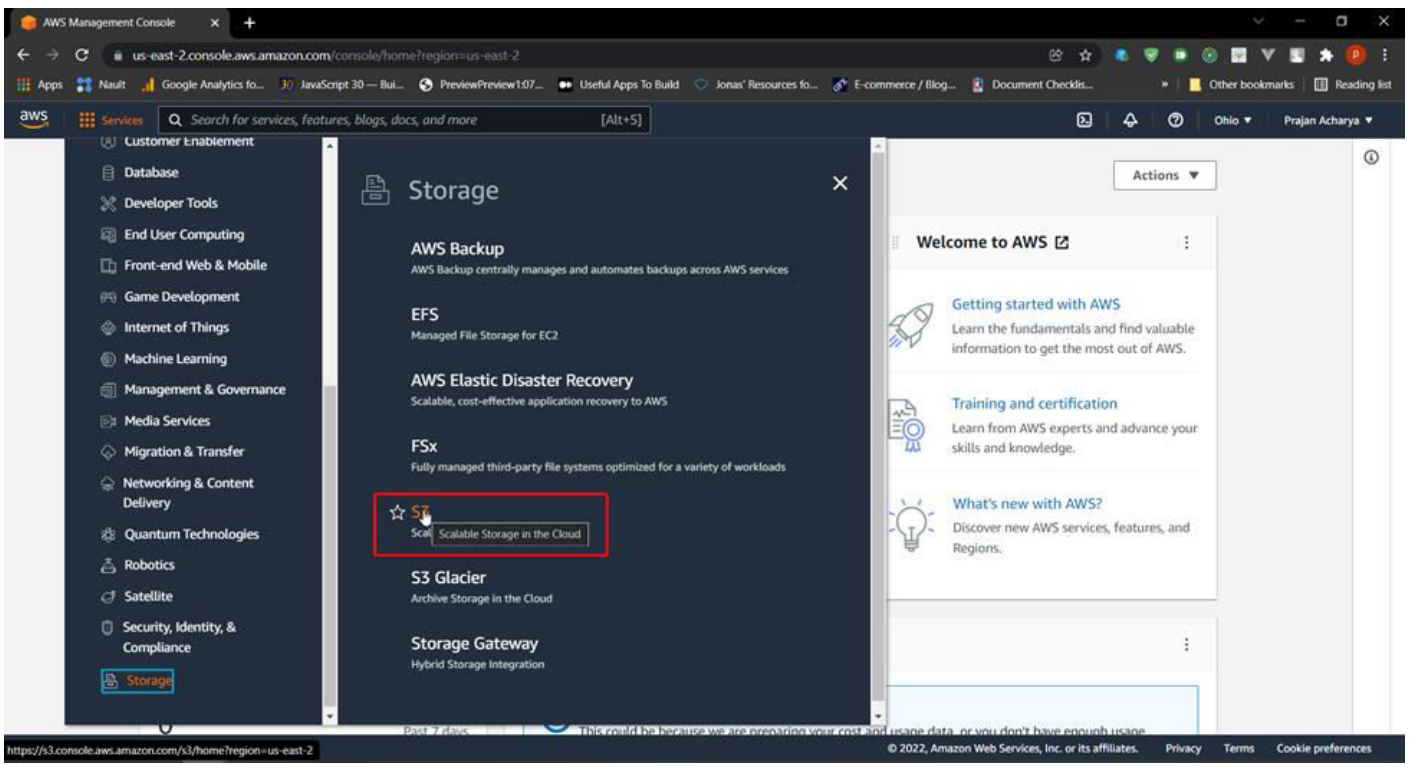
Step 2

Click on "**Services**" present in top-left corner of the Console Home.

- You will be displayed a menu of services.
- From the menu of services, find and click "Storage" menu.
- You will be displayed a list of storage services provided by AWS, on the right hand side of menu of services.
- Finally, from the list of storage services, click on "S3".

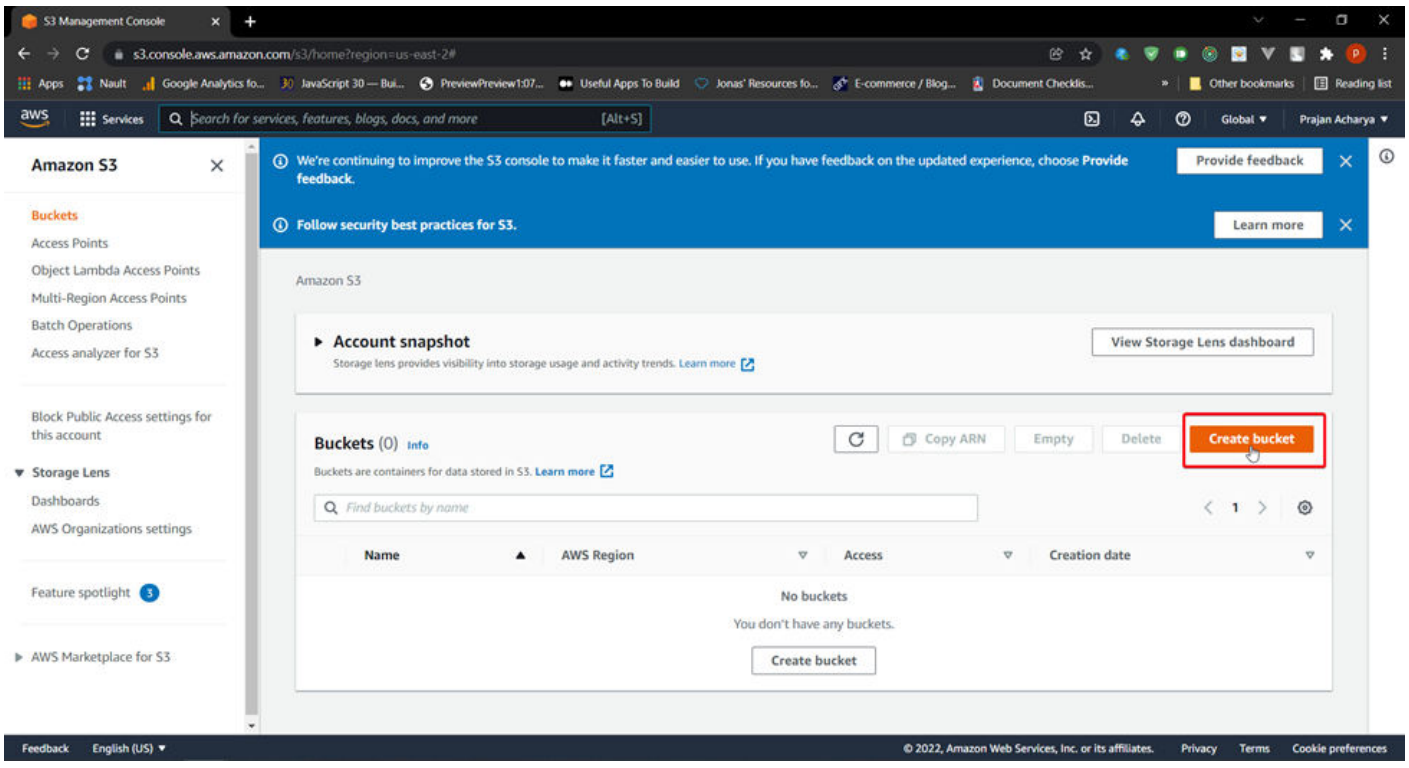


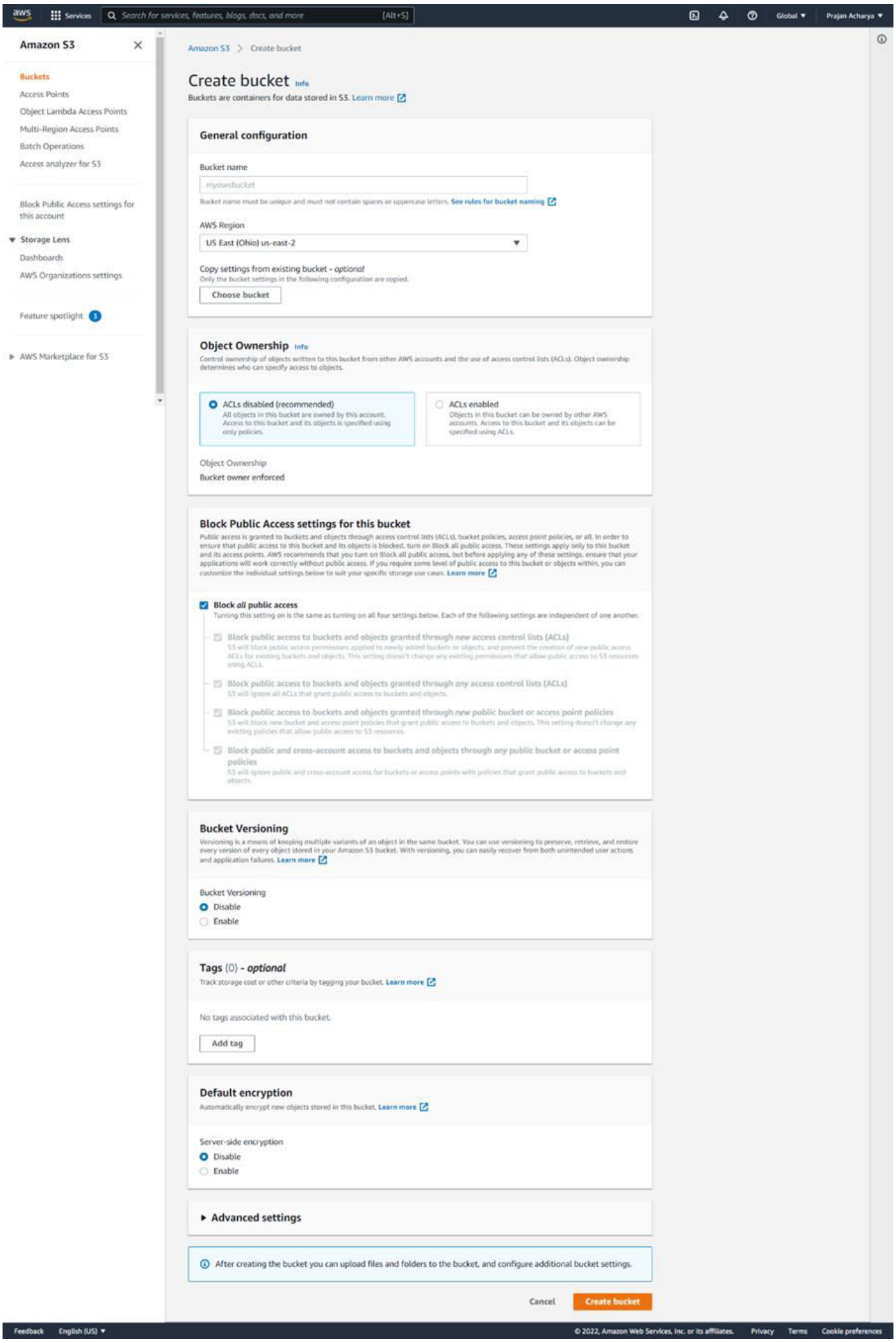




Step 3

To create a bucket, search for a yellowish “Create Bucket” bucket in the right hand side of Console Home page and click on it. You will be displayed a wizard for creating bucket, which comprises of number of sections, where each section is intended to collect several information from us for creating a S3 bucket.


S3 Management Console
s3.console.aws.amazon.com/s3/home?region=us-east-2#
Search for services, features, blogs, docs, and more [Alt+S]
Global
Prajan Acharya
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.
Provide feedback
Follow security best practices for S3.
Learn more
Amazon S3
Account snapshot
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
Buckets (0) Info
Copy ARN
Empty
Delete
Create bucket
Buckets are containers for data stored in S3. Learn more
Find buckets by name
Name
AWS Region
Access
Creation date
No buckets
You don't have any buckets.
Create bucket
Feedback
English (US)
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Step 4

Fill up the required information in all the sections of the wizard. Each section has been provided with the information explaining what it is and it's purpose. Go through those information in case of any confusion and fill up all the sections carefully.

Before moving any further, lets have a clear picture on what are we supposed to do in each section of the wizard.

- In “**General Configuration**” section, enter the bucket name and select an AWS region where the bucket would be created and stored.

✓ *While entering the name for the bucket, please make sure that the name entered by you satisfies all the Bucket naming rules. Also, after we create a bucket, we cannot change its name. Hence, it is good to have concise knowledge on Bucket naming rules before giving a name to a bucket. For gaining information on naming bucket, please see [Bucket naming rules.]*

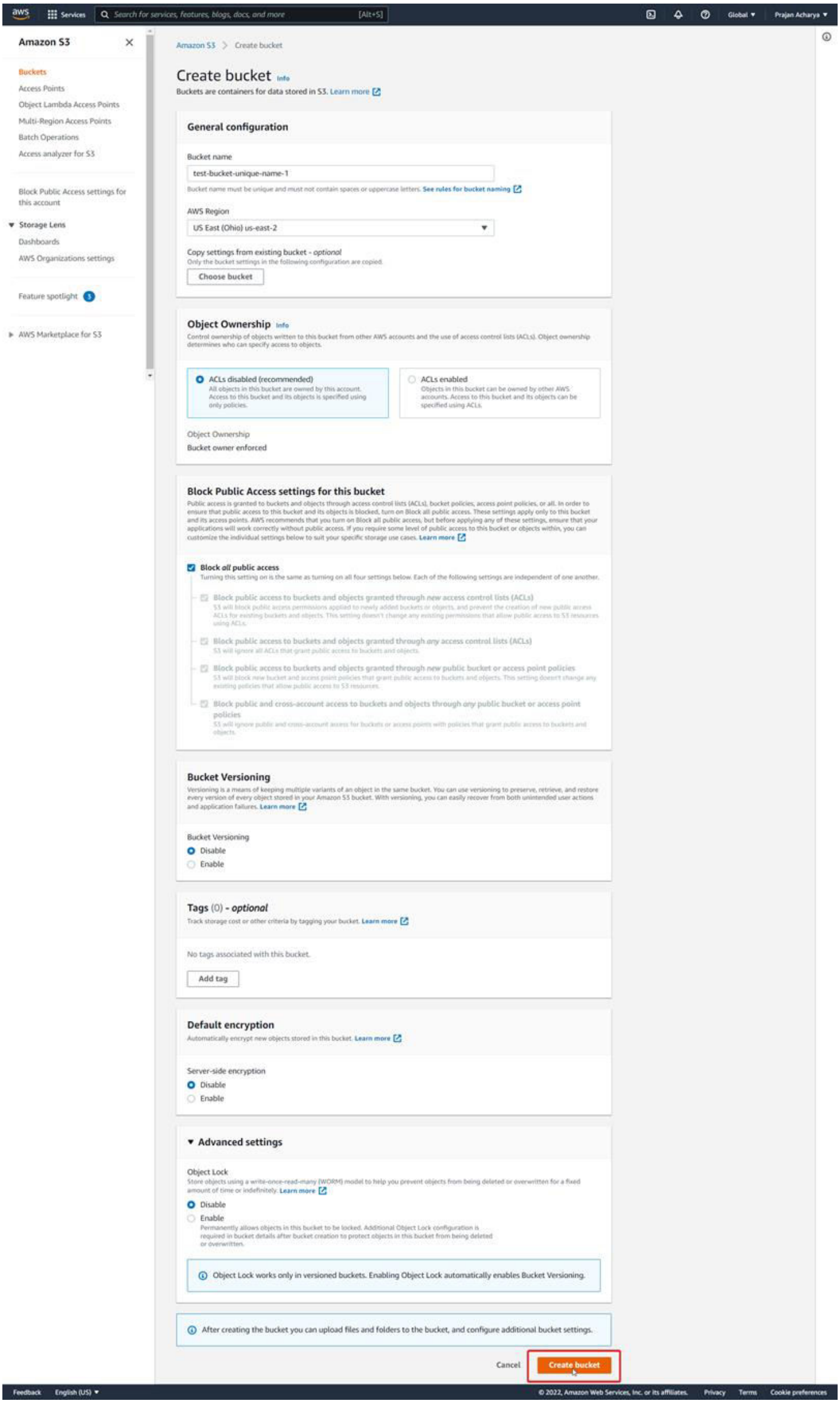
✓ *While choosing an AWS Region, choose a region close to you or your target audience, for minimizing latency and costs and address regulatory requirements.*

- In “**Object Ownership**” section, you can either disable or enable ACLs and control ownership of the objects uploaded to the bucket. For that, you have to choose a setting from two provided settings i.e. choose either “**ACLs disabled**” setting or choose “**ACLs enabled**” setting. If you have any doubt on what each setting means, please go through the description provided under each setting.
- In “**Block Public Access settings for this bucket**” section, choose the Block Public Access settings that you want to apply to the bucket. By default, all the four available setting are enabled and AWS recommends that you keep all those setting enables unless turning off one or more of them is required.

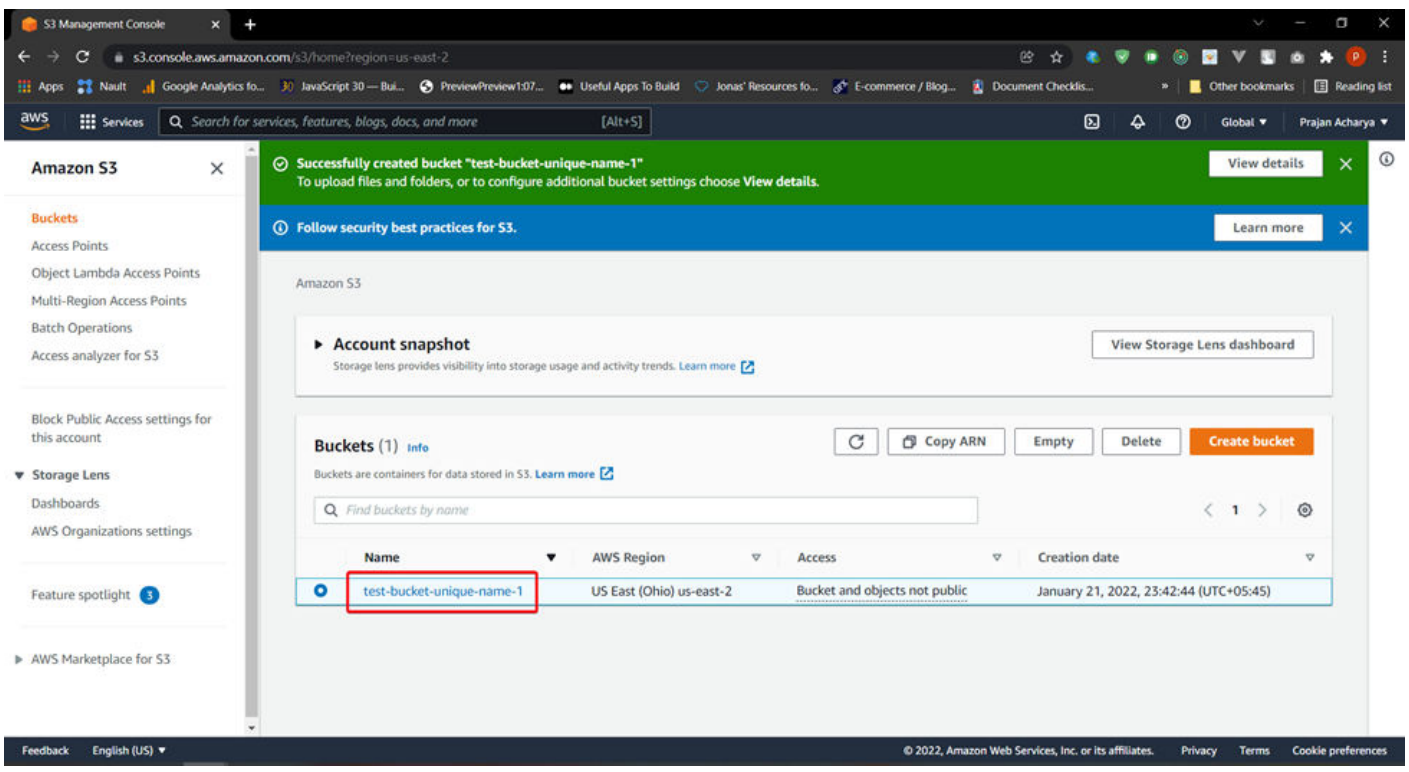
For more information on blocking public access, please see “[Blocking public access to your Amazon S3 storage]”.

- In “**Bucket Versioning**” section, you can either enable or disable versioning for the bucket. By default, versioning is disabled. Enable it only if you require to keep multiple versions of an object in the same bucket.
- In “**Tags**” section, you can add tags to your bucket for various purposes such as tracking storage costs, grouping resources and so on. This section is optional.
- In “**Default encryption**” section, you can either enable or disable server-side encryption for the objects to be stored in the bucket. By default, it is disabled.
- In “**Advanced Setting**” section, you can either enable or disable object lock property for the objects to be stored in the bucket. By default, it is disabled. Please read the information provided in this section and enable it if required.

Now that you have a clear idea on what are you supposed to do in each section, fill each of the section carefully reading the information provided, and finally, click “**Create bucket**” button to submit the wizard.



After clicking on “**Create bucket**” button, a bucket would be created and you will be displayed the home page of Amazon S3 with the newly created bucket listed under “**Buckets**” sections as shown in the screenshot below:



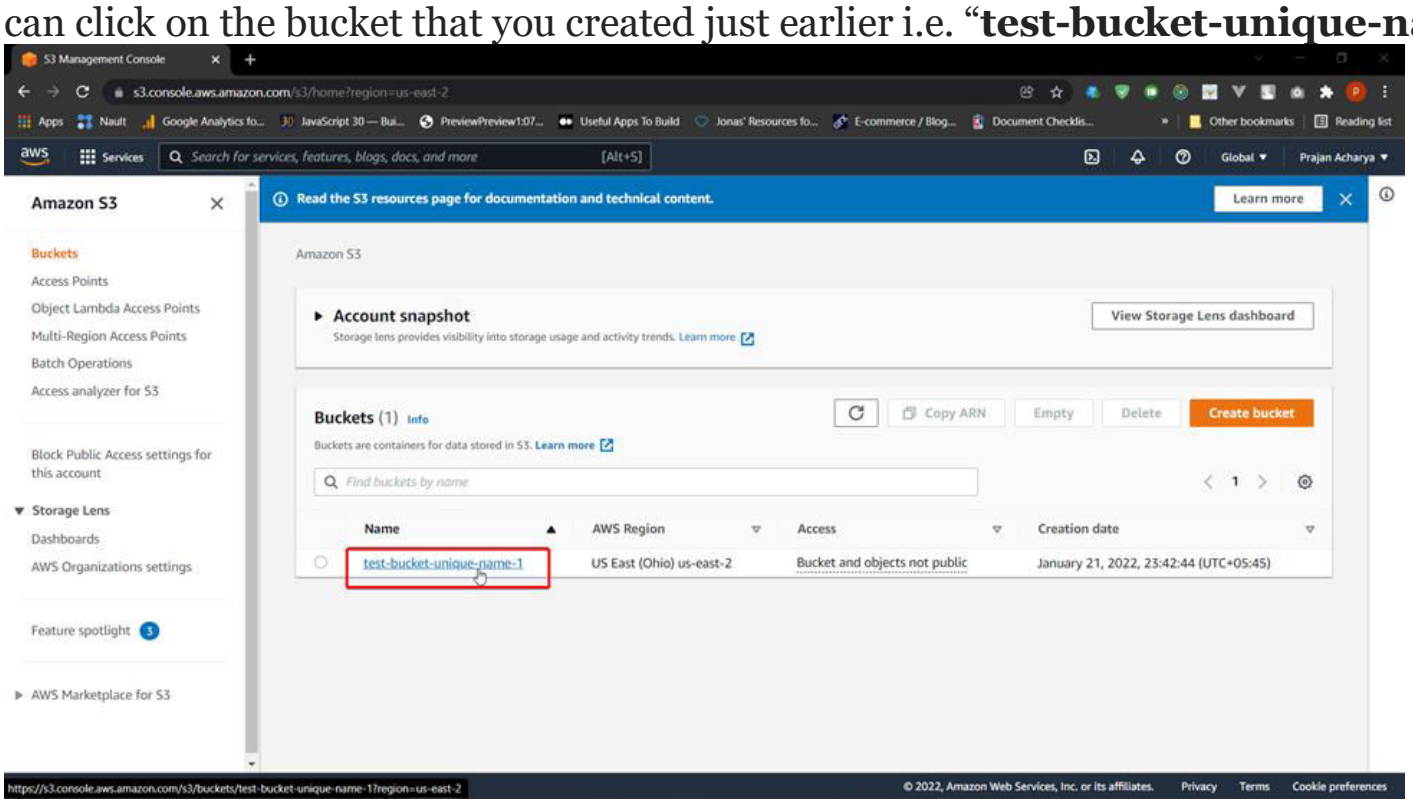
Congratulations! You have created a bucket in Amazon S3.

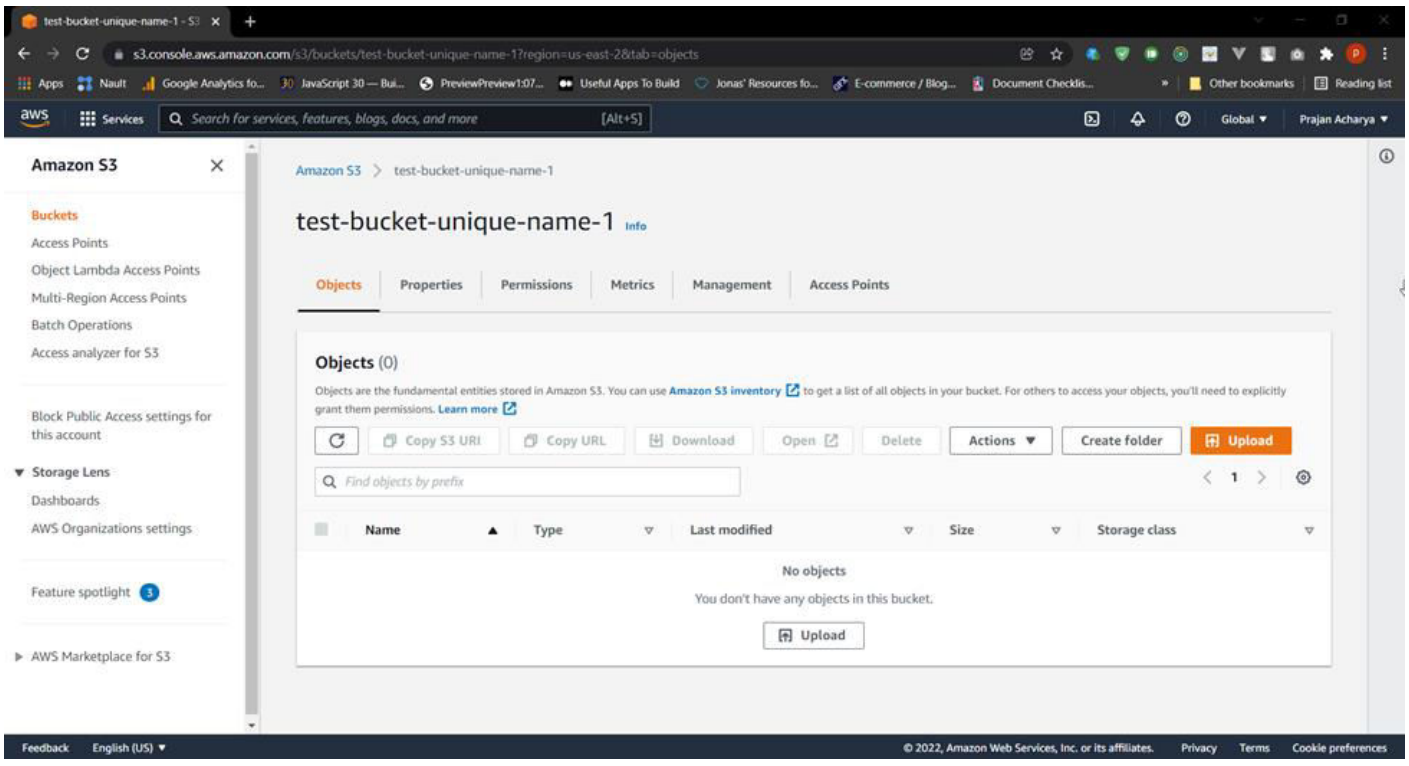
How to upload object(s) to Amazon S3 bucket

Amazon S3 bucket stores the data uploaded by us in the form of object. Hence, the object can be anything from document, image files, video files, folders and so on. In order to upload object(s) to Amazon S3 bucket, you carry out following steps:

Step 1

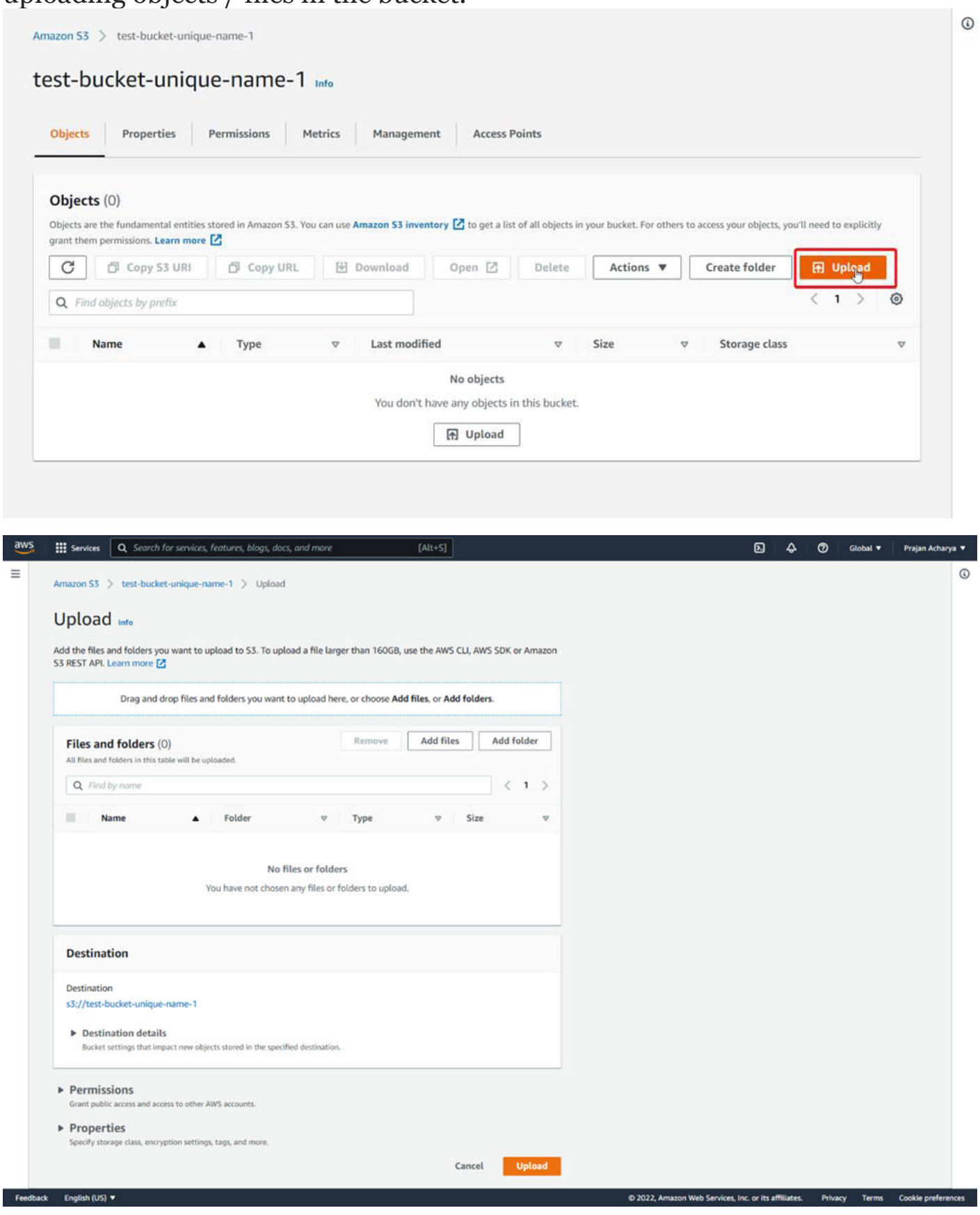
In the home page of Amazon S3, under **Buckets** section present in the middle of the page, you will be displayed a list of Amazon S3 buckets. You have to find and click on the bucket, inside which you want to upload object(s). After you click on the bucket, you will be displayed a page for the bucket you clicked. You can click on the bucket that you created just earlier i.e. "**test-bucket-unique-name-1**"





Step 2

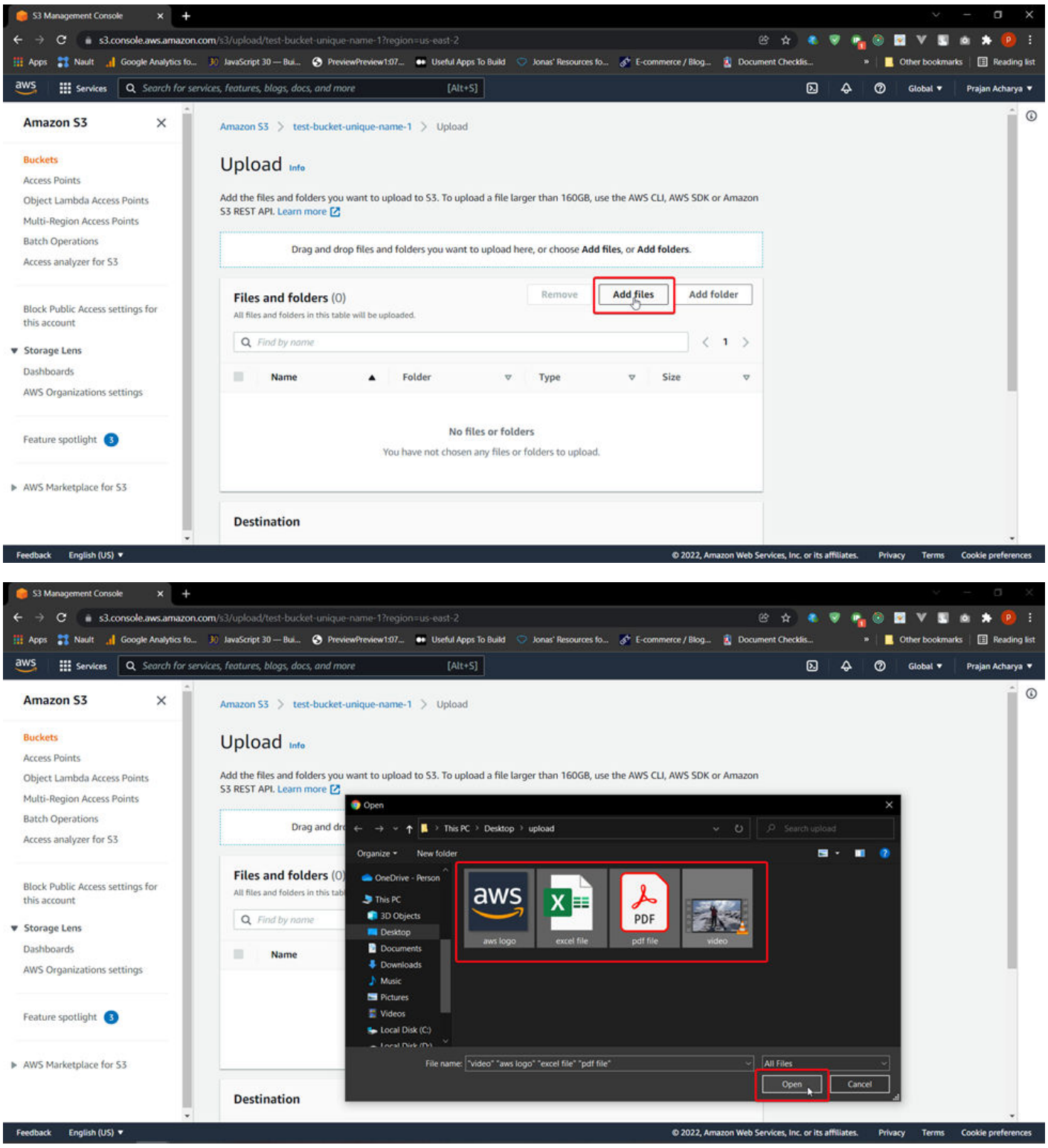
Click on “**Upload**” button present in the right hand side of the screen. You will be displayed a page for uploading objects / files in the bucket.

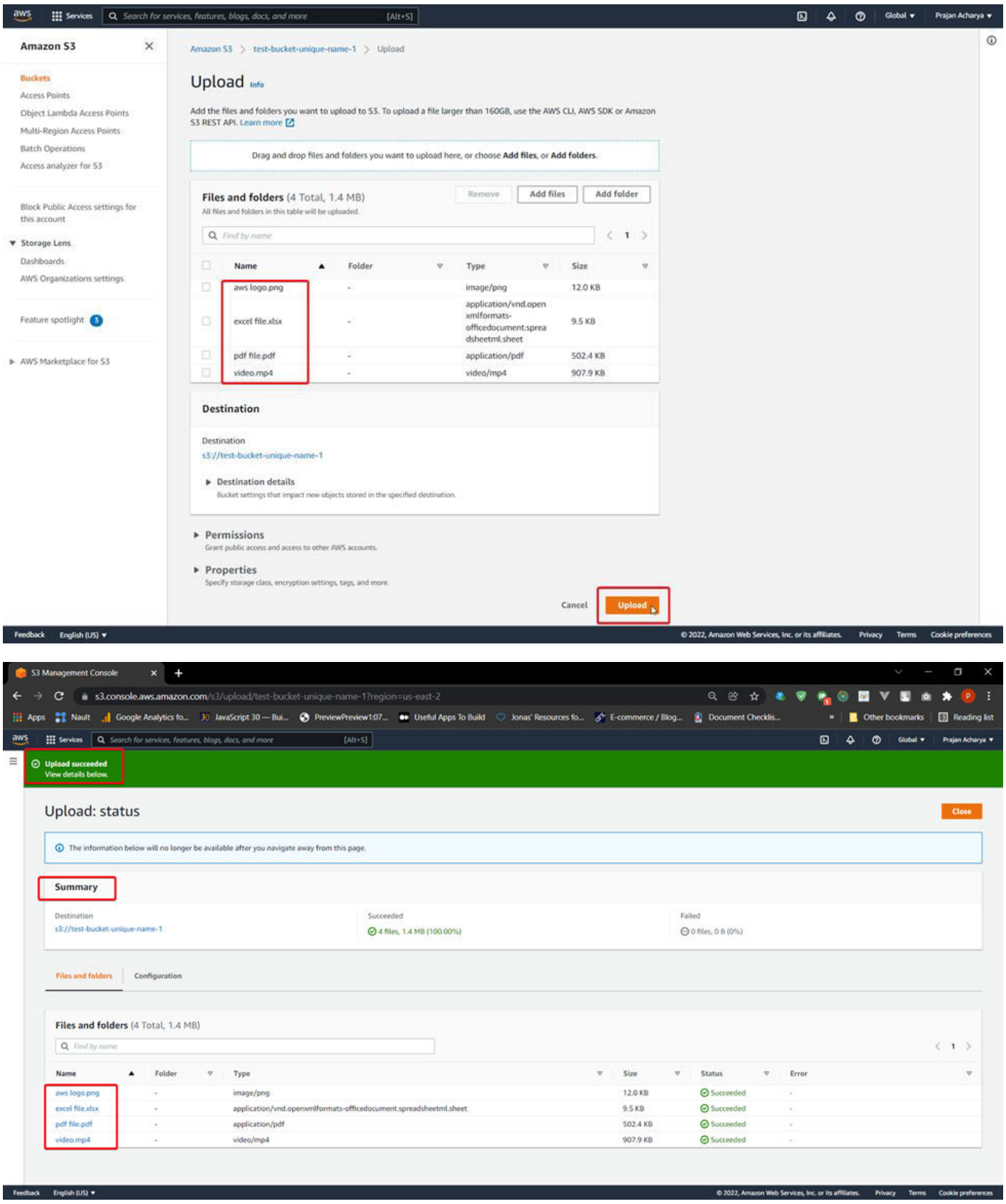


Step 3

Click on the “**Add files**” button present under “Files and folders” section. You will be displayed a file explorer from where you can select file(s) and upload. Alternatively, you can drag and drop file(s) directly into the page for uploading.

After selecting and confirming the files to be uploaded via. File explorer or via. Drag and drop, the selected files would be displayed under “**Files and Folders**” section. Finally, after clicking “Upload” button present at the bottom of the page, the selected files would be uploaded to the particular Amazon S3 bucket and you will be then displayed a success message, summary of success message and the list of uploaded files under “**Files and Folders**” section.



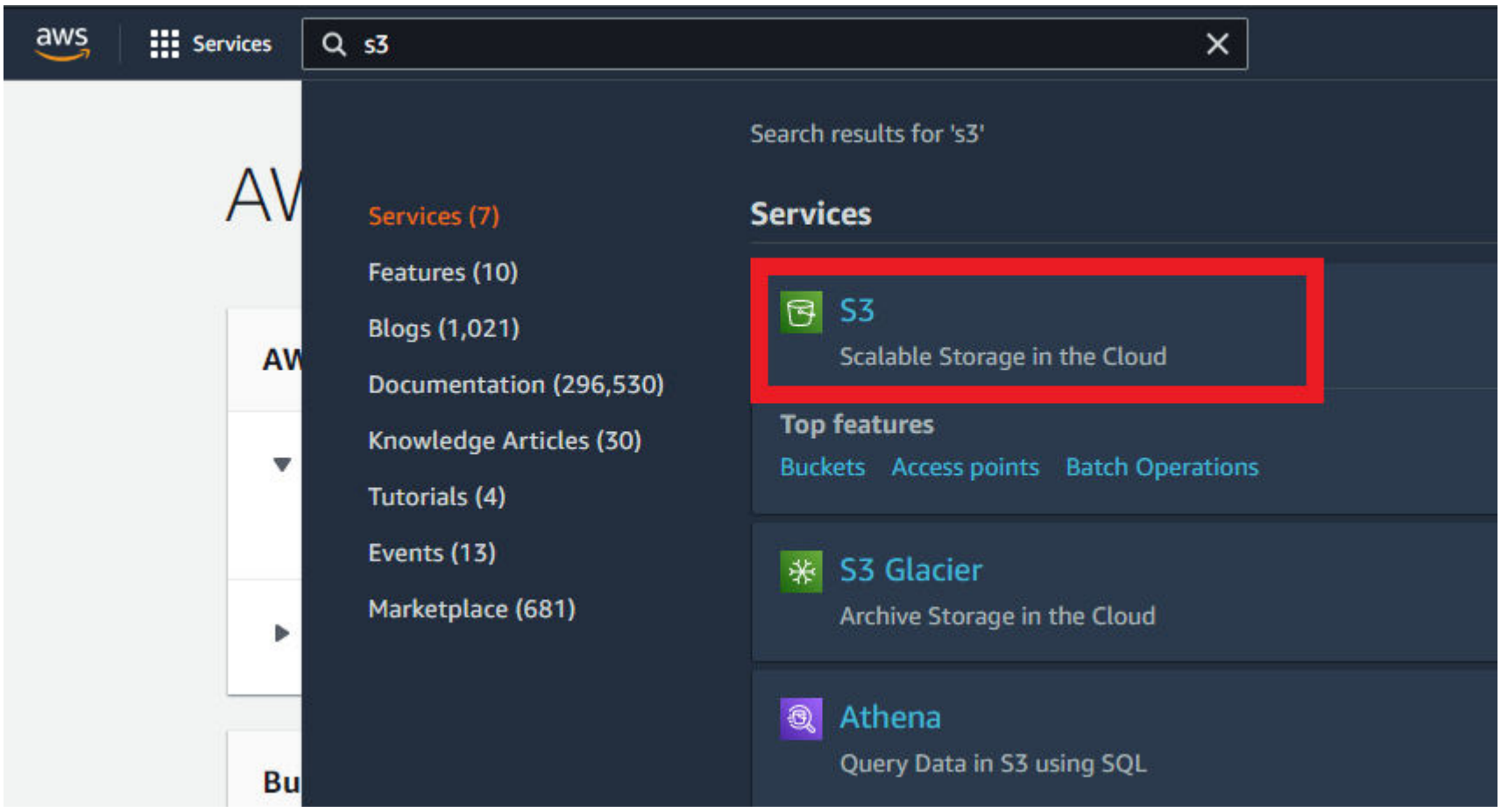

Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3 > test-bucket-unique-name-1 > Upload
Upload Info
Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more
Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.
Files and folders (4 Total, 1.4 MB)
All files and folders in this table will be uploaded.
Remove
Add files
Add folder
Find by name
Name
Folder
Type
Size
aws logo.png
image/png
12.0 KB
excel file.xlsx
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
9.5 KB
pdf file.pdf
application/pdf
502.4 KB
video.mp4
video/mp4
907.9 KB
Destination
s3://test-bucket-unique-name-1
Destination details
Bucket settings that impact new objects stored in the specified destination.
Permissions
Grant public access and access to other AWS accounts.
Properties
Specify storage class, encryption settings, tags, and more.
Cancel
Upload

S3 Management Console
s3.console.aws.amazon.com/s3/upload/test-bucket-unique-name-1?region=us-east-2
Upload succeeded
View details below.
Upload: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Destination
s3://test-bucket-unique-name-1
Succeeded
4 files, 1.4 MB (100.00%)
Failed
0 files, 0 B (0%)
Files and folders
Configuration
Files and folders (4 Total, 1.4 MB)
Find by name
Name
Folder
Type
Size
Status
Error
aws logo.png
image/png
12.0 KB
Succeeded
excel file.xlsx
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
9.5 KB
Succeeded
pdf file.pdf
application/pdf
502.4 KB
Succeeded
video.mp4
video/mp4
907.9 KB
Succeeded

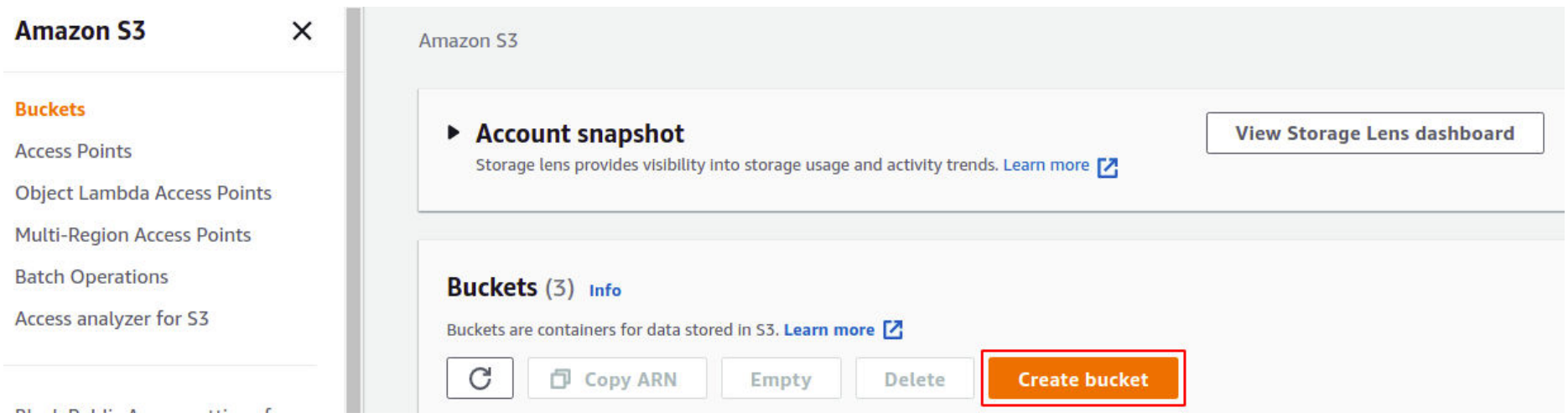
Hosting Static Website using S3

The first step to hosting a static website on AWS S3 is to create an S3 bucket in your account. After creating the bucket, we will upload the website contents and files in our bucket. The website content will then be assigned specific permissions to be accessible to the public.

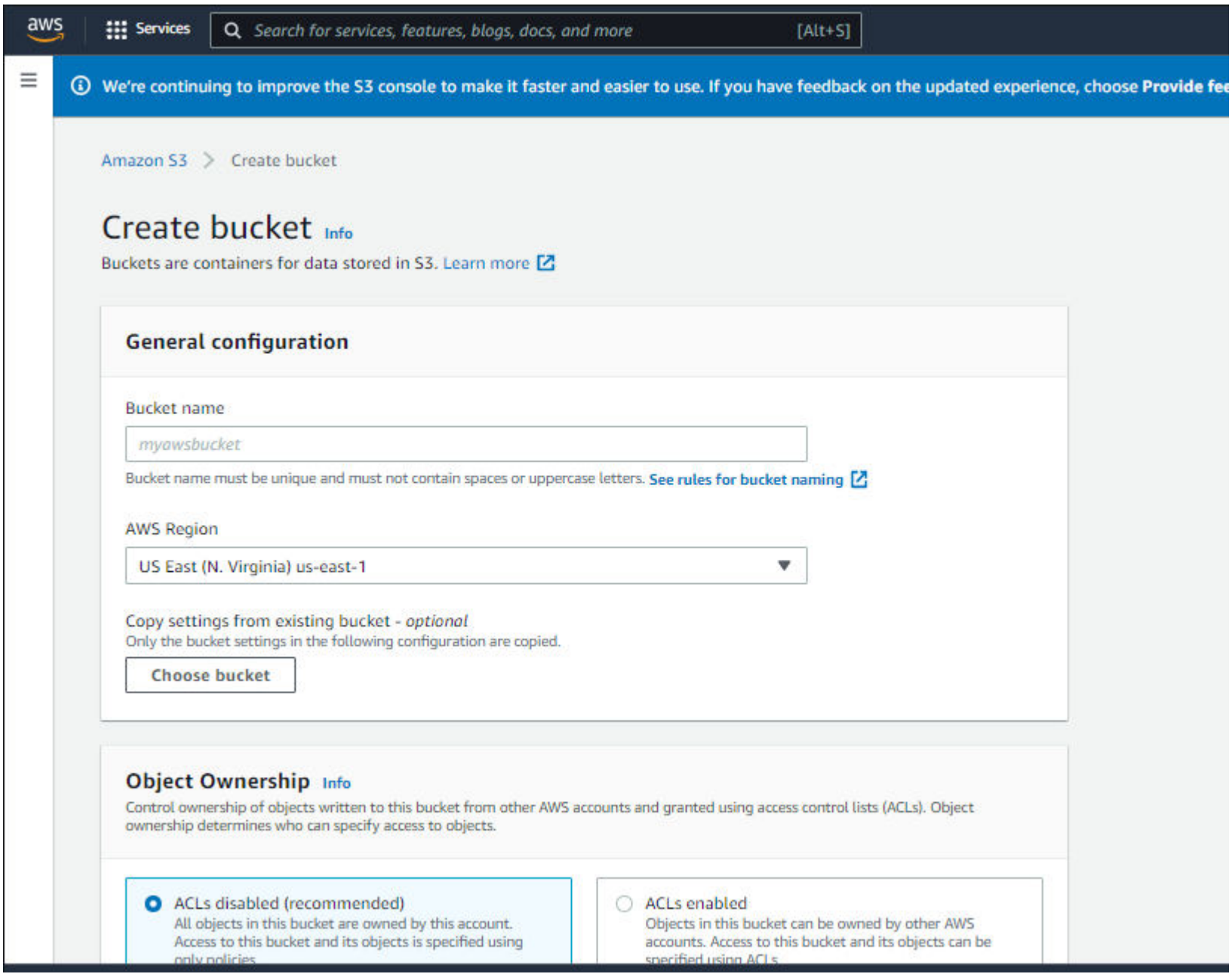
Login to your AWS management console and go to the search bar and search for **S3** there. This will lead you to your S3 dashboard:



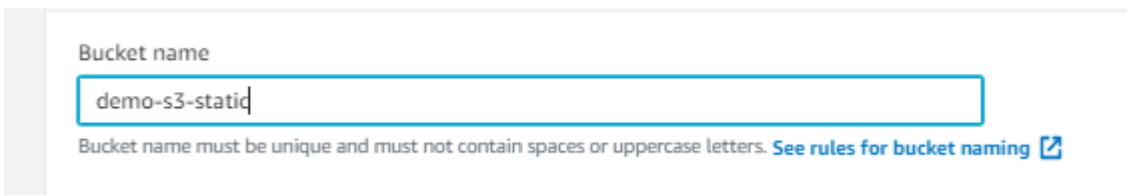
Click on Create Bucket at the right corner of the S3 console:



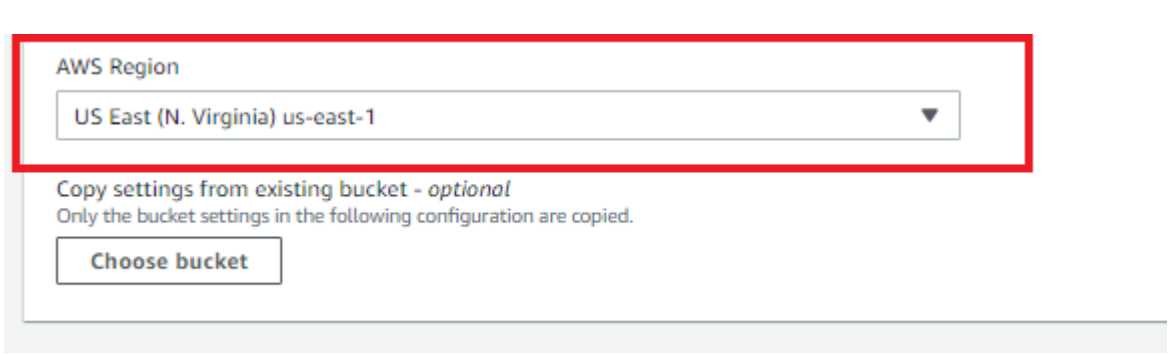
Next, you need to provide your S3 bucket name, the region where you want your bucket to be created, and then configure your bucket's security and privacy setting:



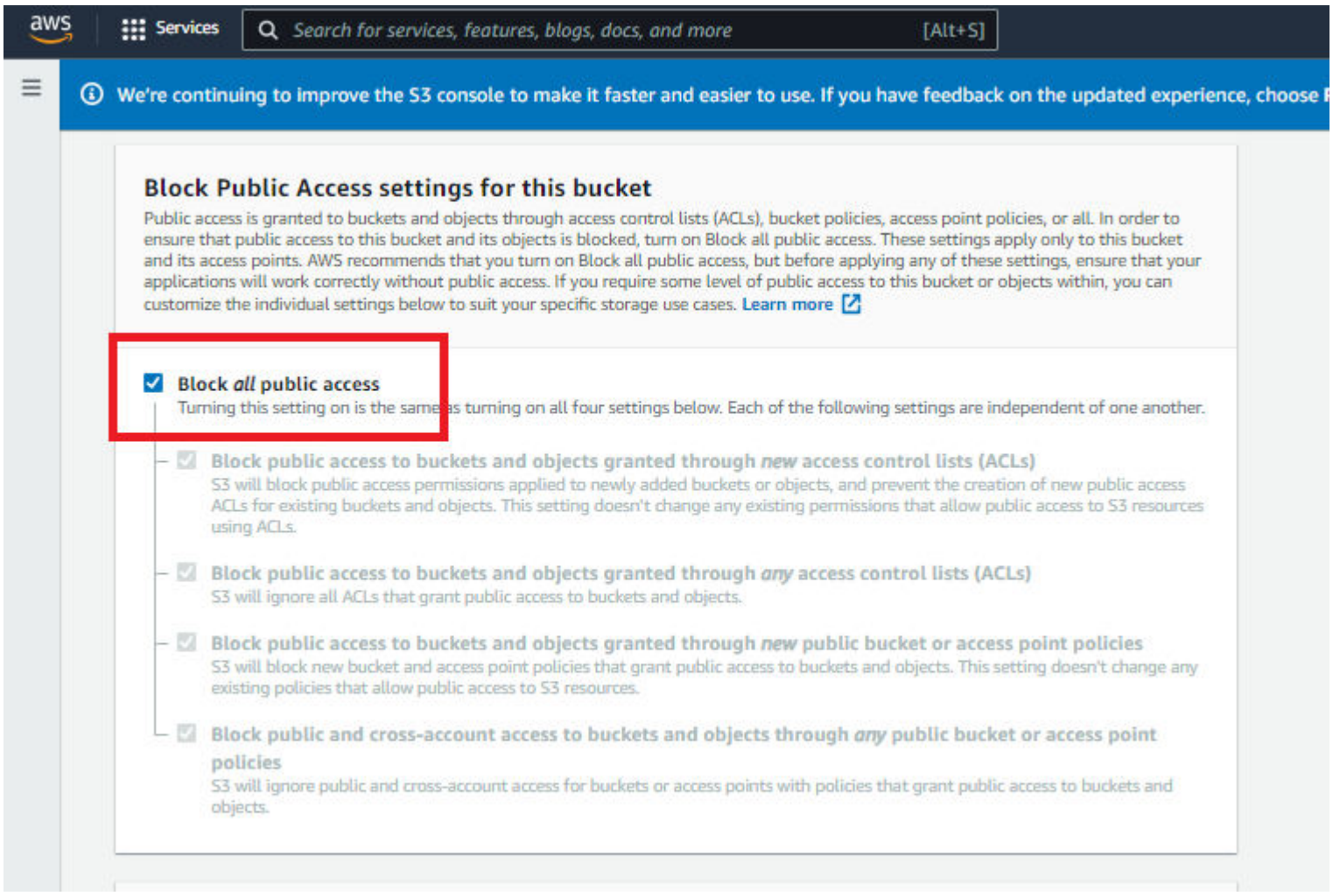
Enter Bucket Name, try to make it look like your domain. The bucket name should be unique for all AWS accounts around the world:



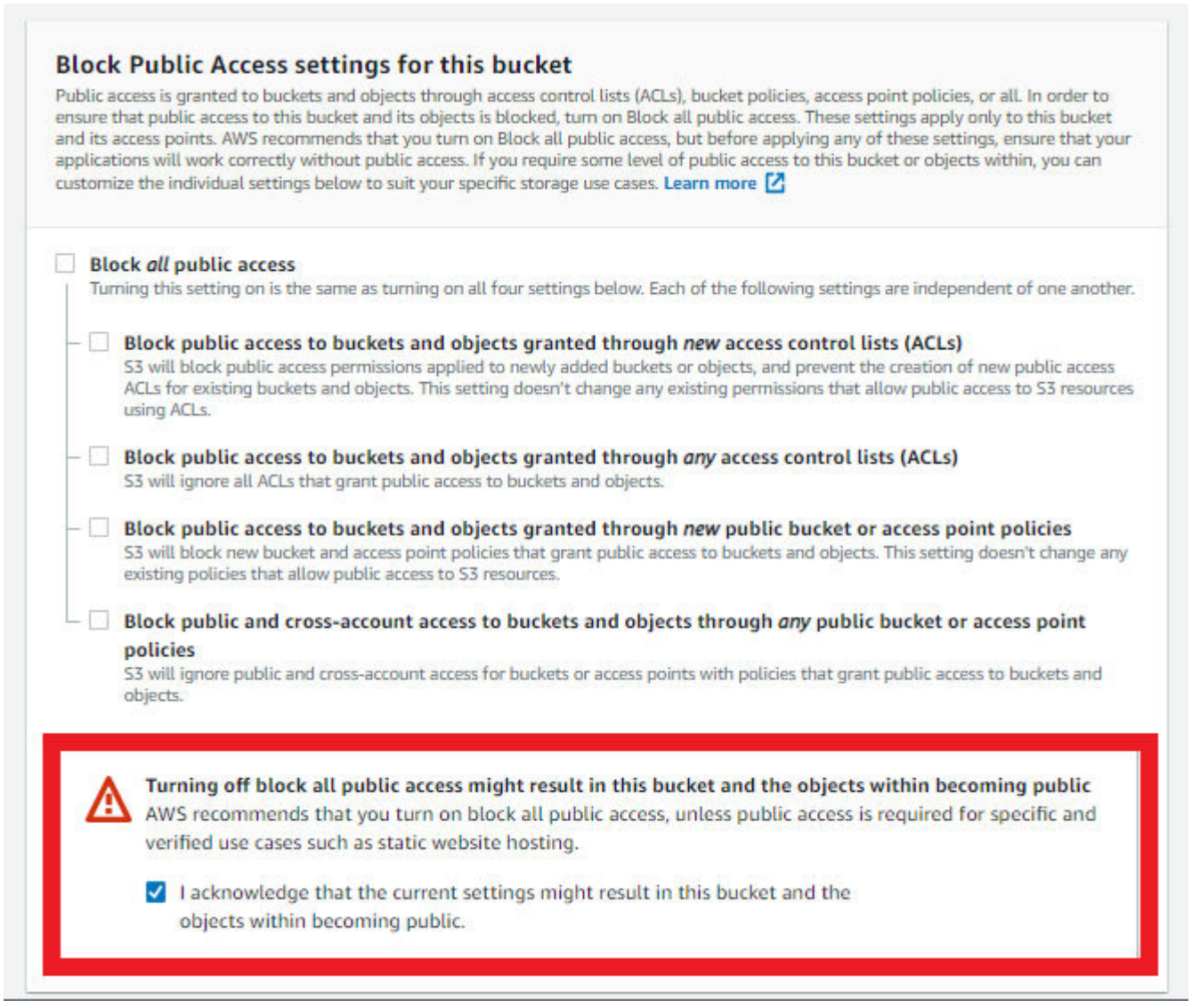
Select the region in which the S3 bucket will be created. Try to select a region near the public that will access the website:



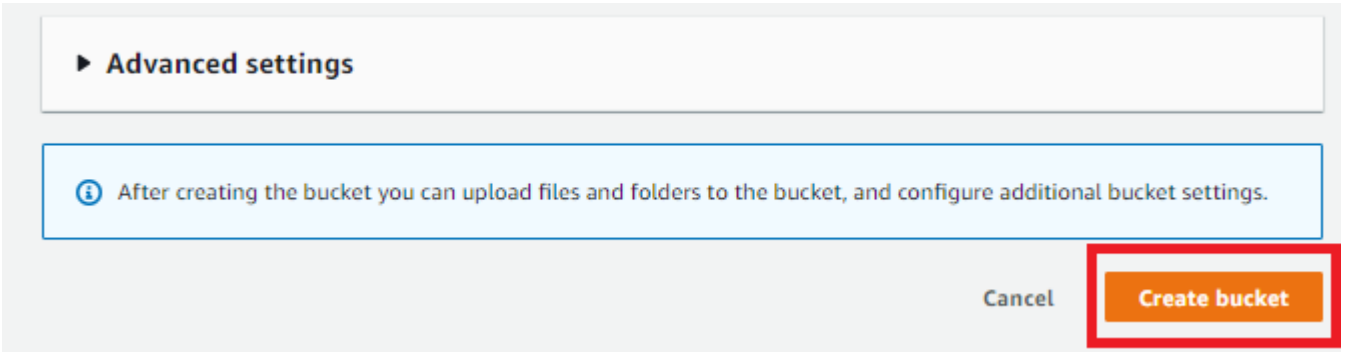
Since we wanted the website to be accessible to the audience, we had to grant the public access to the objects of this S3 bucket. For that, uncheck the Block all public access checkbox in the “Block Public Access setting for this bucket” section:



After configuring the public access settings, a section will appear to acknowledge the S3 bucket and its content being made public. Check the box to acknowledge it:



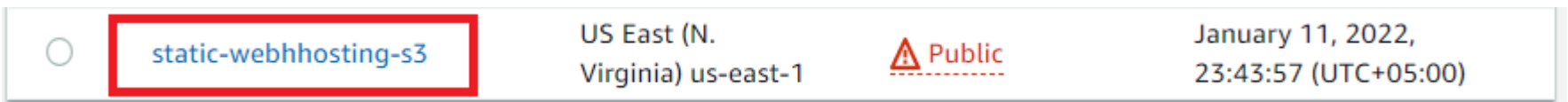
Now, you have finished setting up your bucket, leave other options and settings as it is, and just click on the **Create Bucket** button at the bottom right corner:



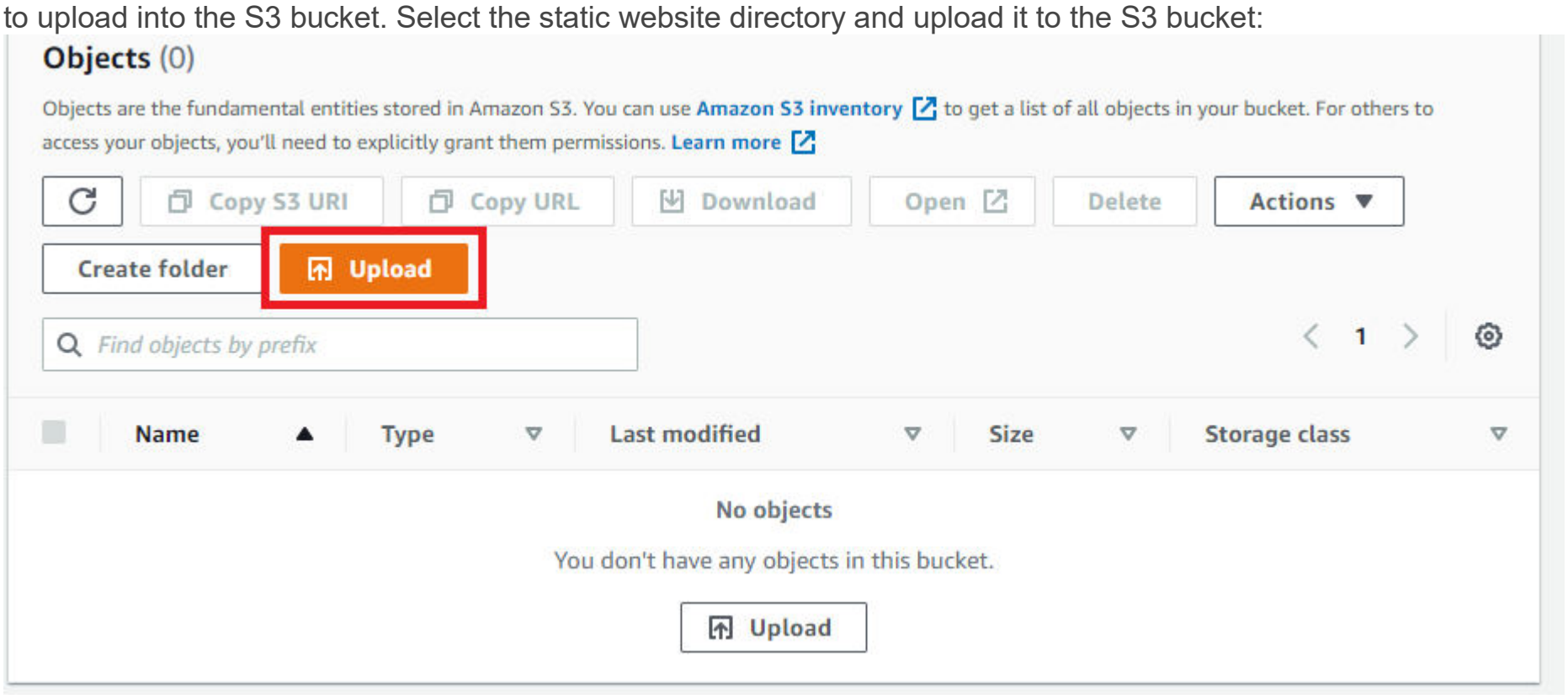
If the bucket name you specified is unique, the S3 bucket will be created. Otherwise, you will get an error, and you have to change the bucket name.

Upload Your Website to the S3 Bucket

After creating the S3 bucket, it is time to upload website content to the S3 bucket. From the S3 console, select the S3 bucket you just created:



Go to the **Objects** section, and then Click on the upload button. Now, browse your system for the directory you want to upload into the S3 bucket. Select the static website directory and upload it to the S3 bucket:

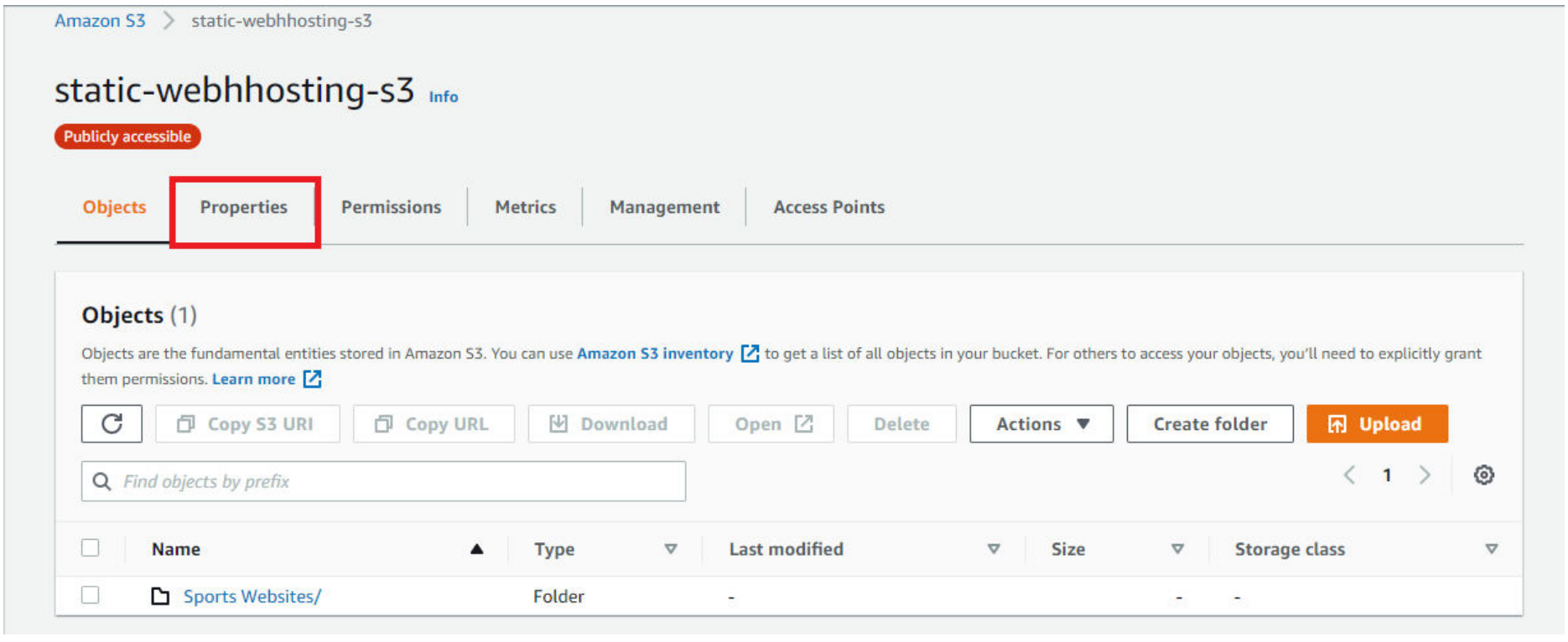


Uploading the static site content may take some time depending upon the size of the folder:

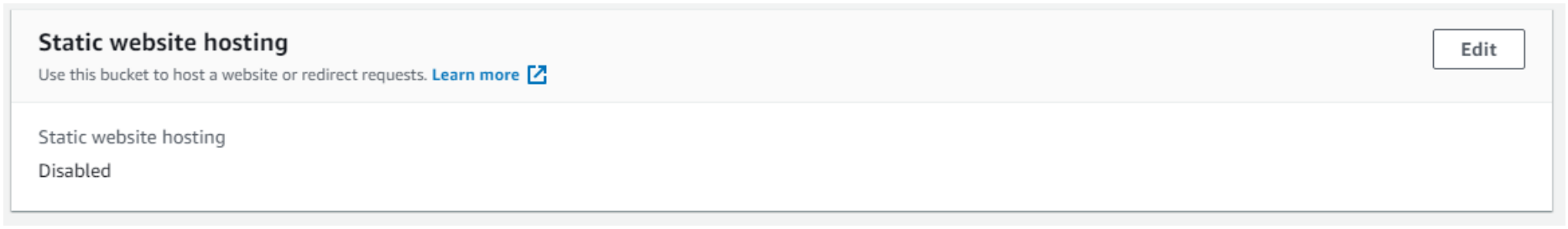
After a successful upload, click close at the right corner. You will be directed back to the object section.

Setting up Static Web Hosting in S3 Bucket

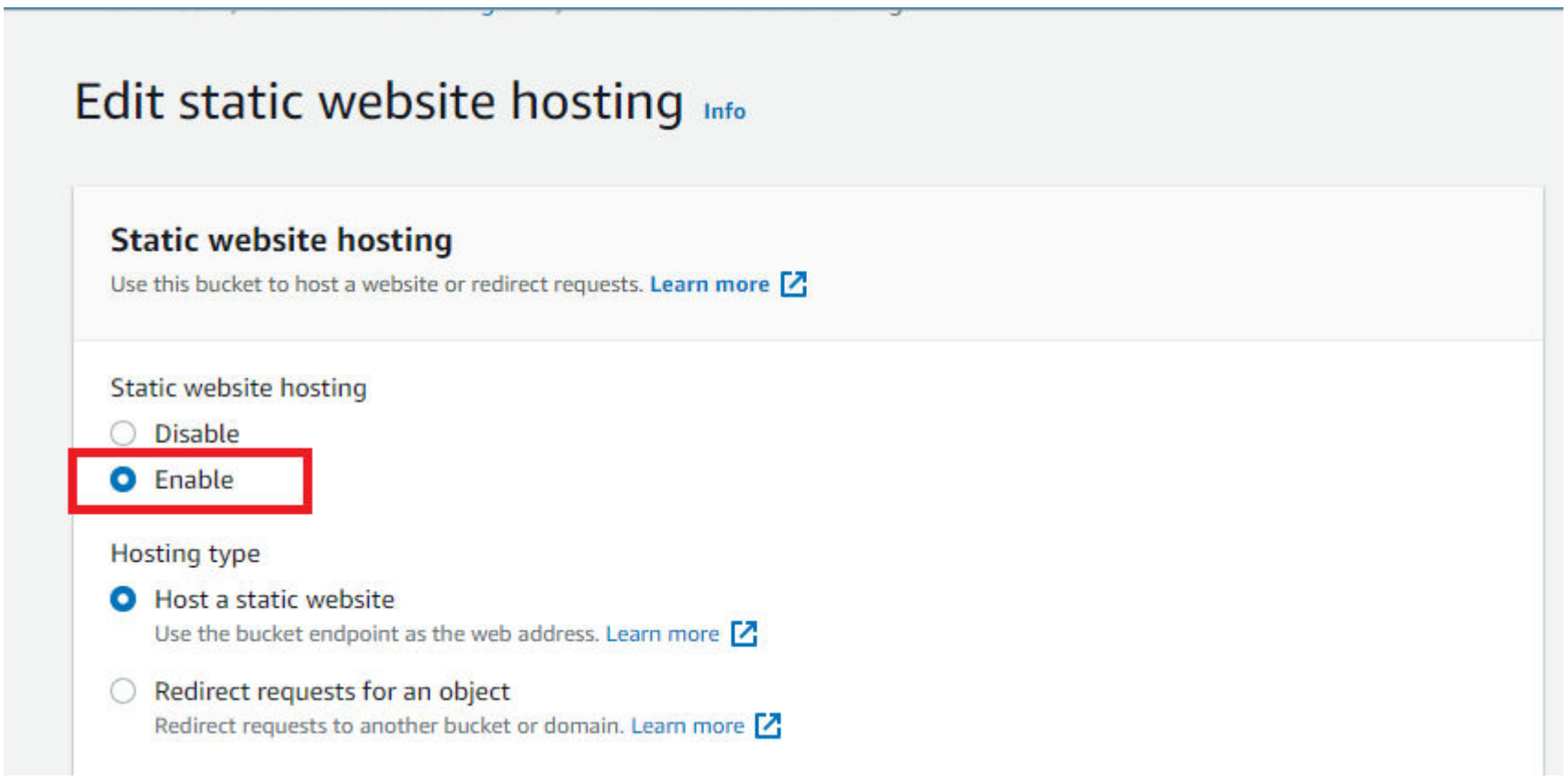
After uploading the static site content, enable hosting on your S3 bucket. In order to allow static website hosting on your S3 bucket, go to the properties tab from the top menu in the S3 bucket:



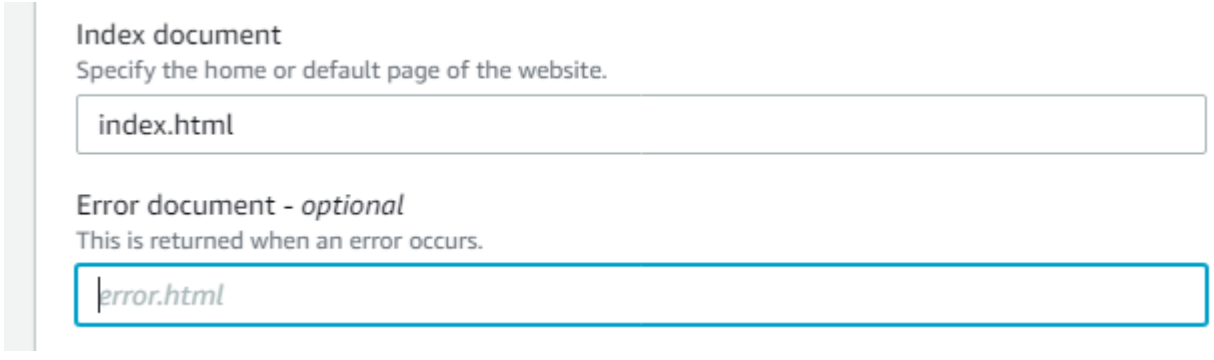
Scroll down in properties tab and look for the Static Website Hosting section:



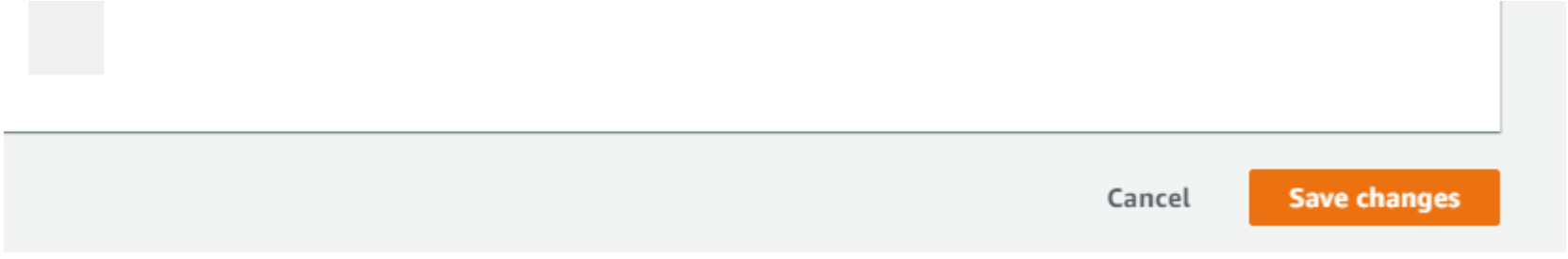
Click on the Edit button in the Static website hosting section and enable the hosting:



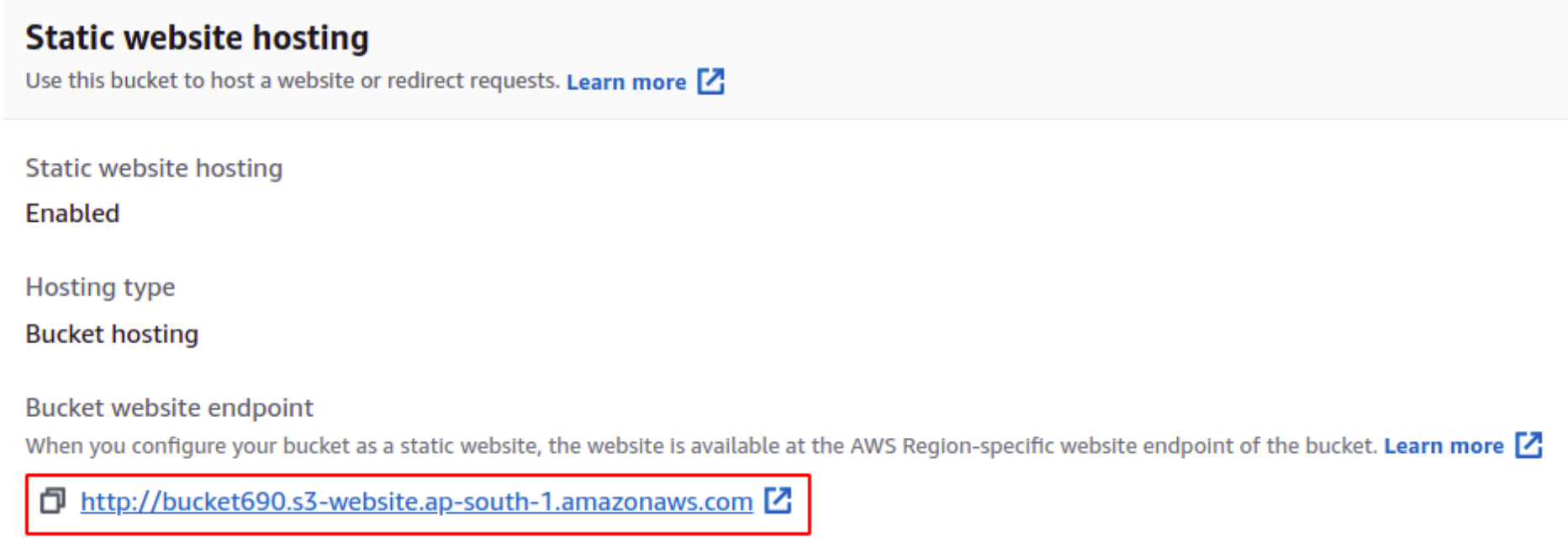
After enabling static website hosting, specify the index file of your project (the opening page of your website or web application). In this case, it is index.html:



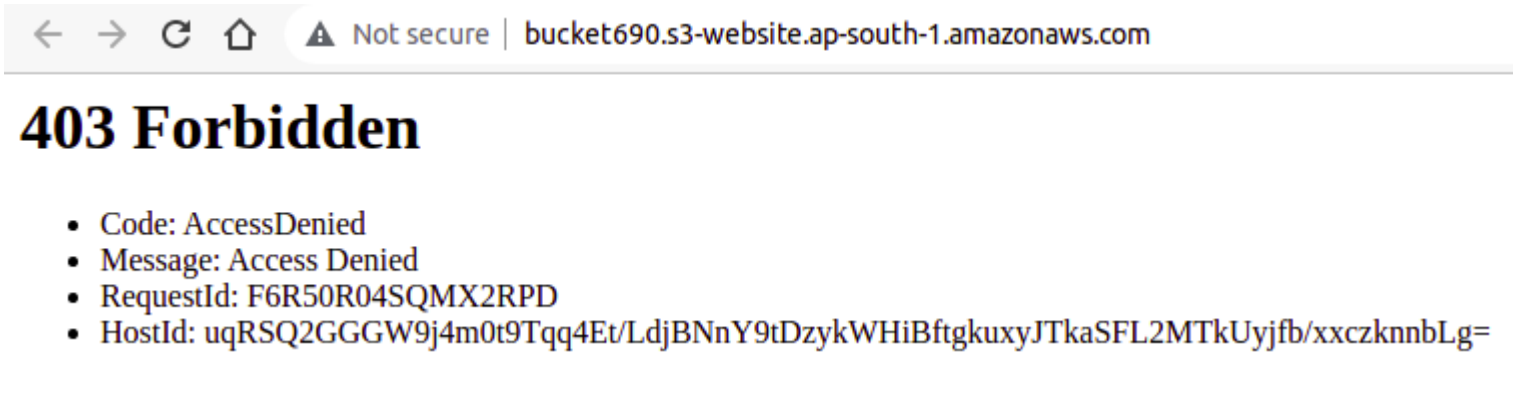
Also, if there is an error file in your project, you must specify it in the error document field. This will appear in case your actual web page is not reachable. Now. click on the **Save changes** button to apply the changes to your S3 bucket:



Now, our S3 bucket is hosting the website content uploaded to it and is publicly accessible. In order to access the website, we need a public URL that AWS itself provides. This URL can be seen in the static website hosting section of the S3 bucket:



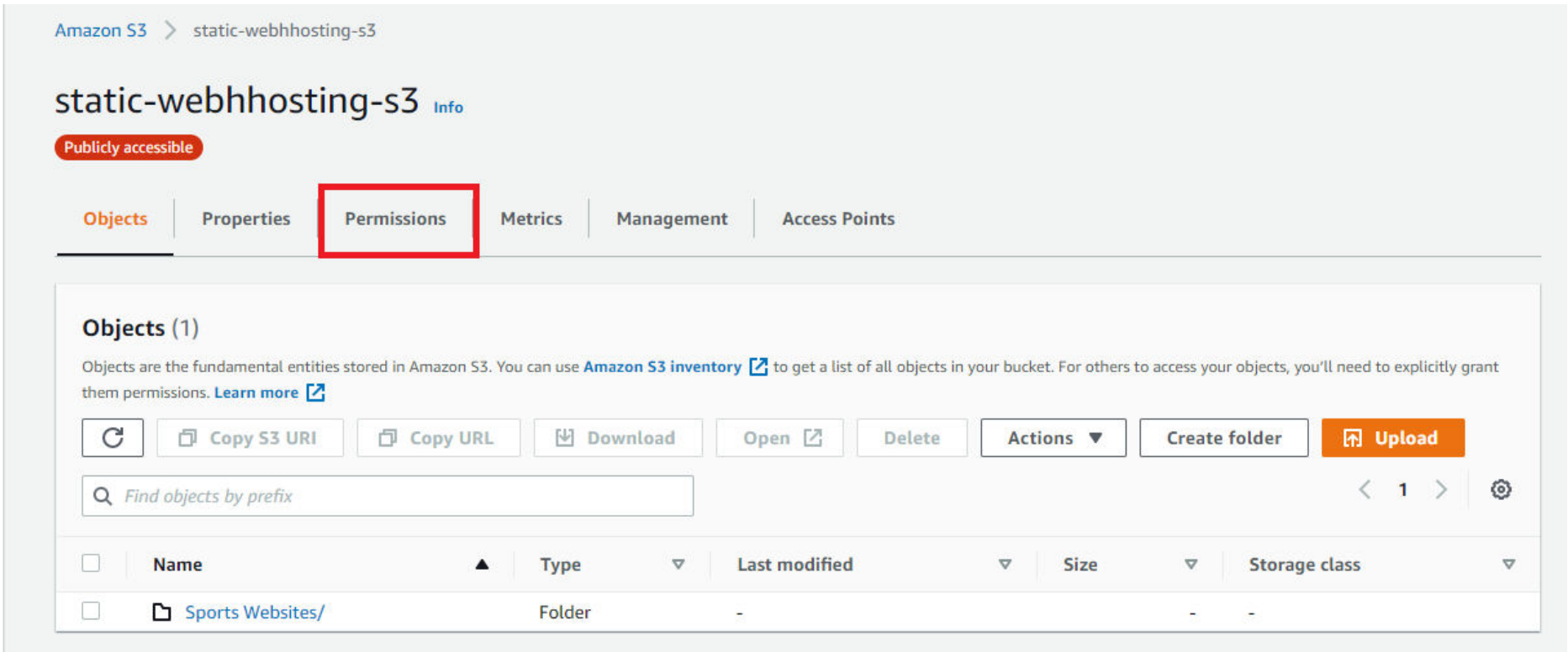
Go to the URL provided by S3, and the website will not be accessible because we have made the S3 bucket public, but the objects inside the S3 bucket are not public yet:



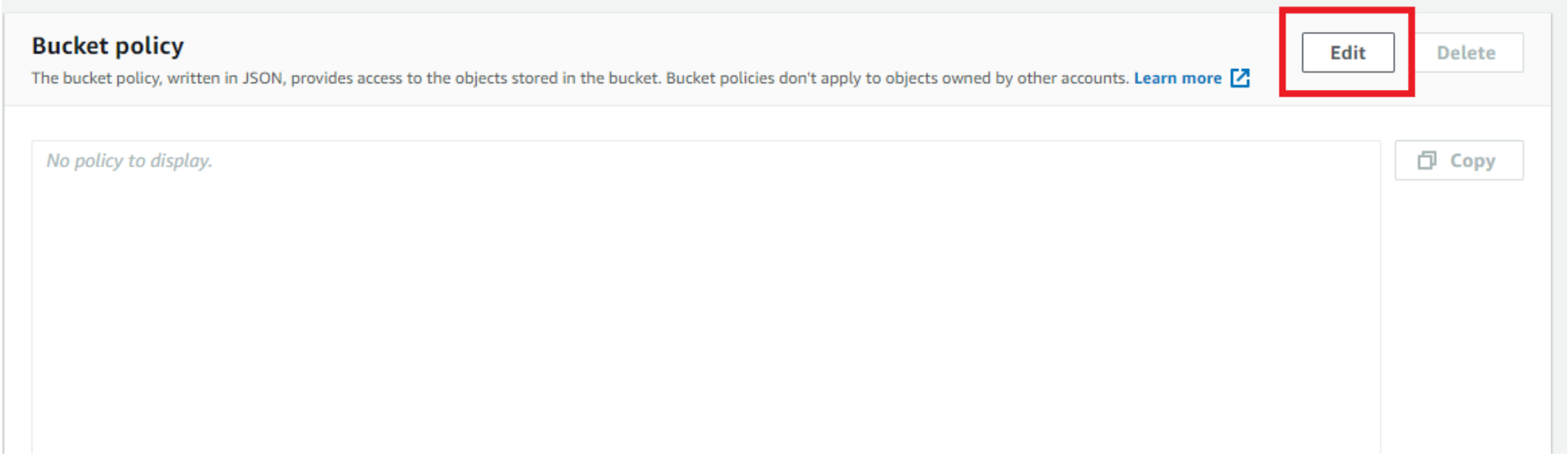
This problem can be solved by using the S3 bucket policies.

Setting up Permissions in S3 Bucket

To make our content accessible publicly, we need to add a bucket policy for which we have to go to the permissions tab of our S3 bucket to make some changes to the permissions of our S3 bucket:



Now, move to the bucket policy section and click on the **Edit** button:



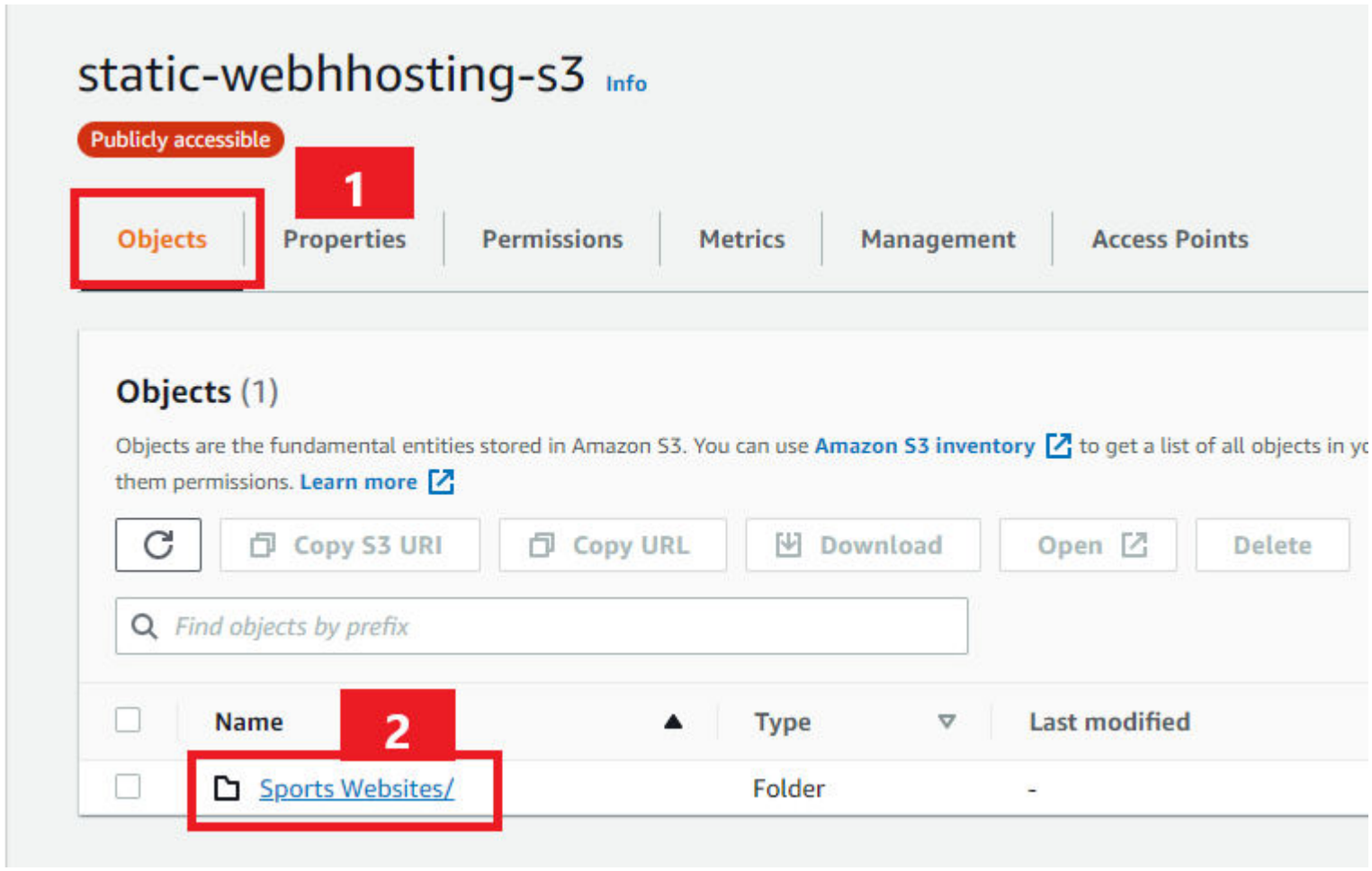
Paste the following JSON in the editor to allow the public to read files from the bucket:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::YOUR-S3-BUCKETNAME/*"
        }
    ]
}
```

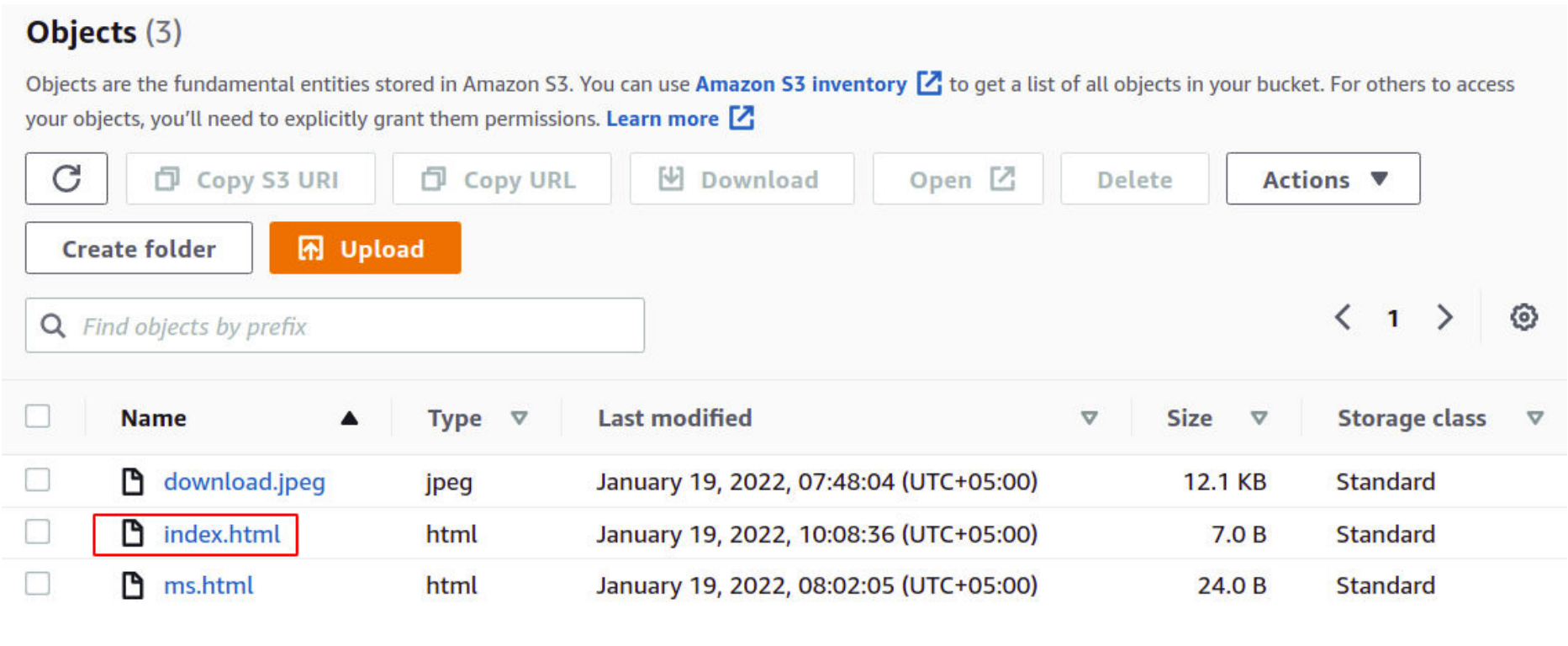
Make sure to replace “**YOUR-S3-BUCKETNAME**” with your S3 bucket name in the JSON policy.

Accessing the Website Through URL

After setting the permissions for the bucket, it’s time to access the webpage through the URL. For this, go to the **Objects** tab of the S3 bucket and go to the static site directory:



Look for the index.html file in the folder, which you defined as the index document for this project. Click on the index.html file:



Now, in the object overview section under the properties tab, you can find the URL of the static website:

Object overview

Owner

1919cfead8d7e1eb706a1357f05c31ec44cf6f0708473f6cd613a4838e476d2e

AWS Region

Asia Pacific (Mumbai) ap-south-1

Last modified

January 19, 2022, 10:41:21 (UTC+05:00)

Size

7.0 B

Type

html

S3 URI

s3://bucket690/Sports Websites/index.html

Amazon Resource Name (ARN)

arn:aws:s3:::bucket690/Sports Websites/index.html

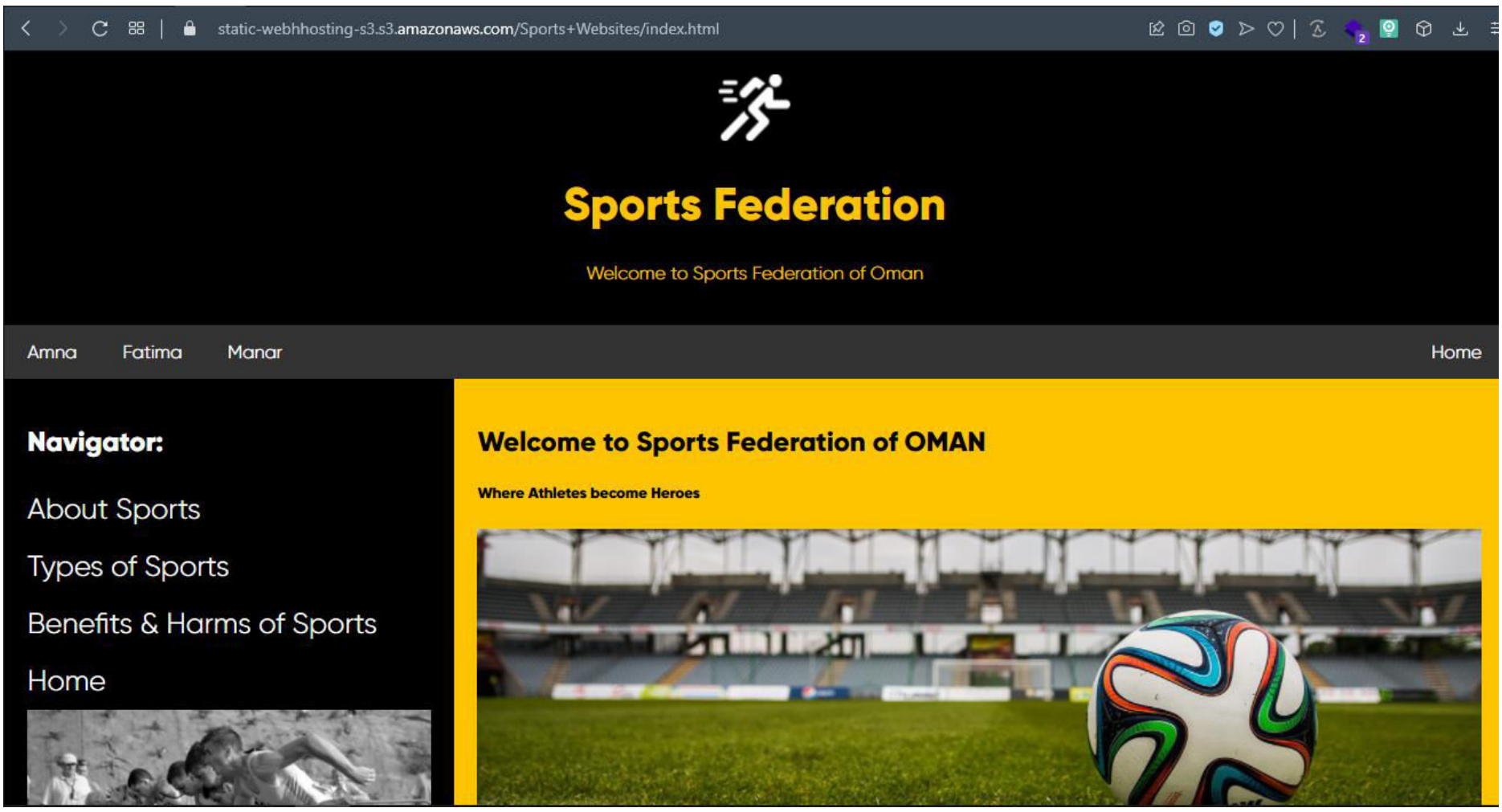
Entity tag (Etag)

db163818a96b68134f42b6d10dd2c88d

Object URL

https://bucket690.s3.ap-south-1.amazonaws.com/Sports+Websites/index.html

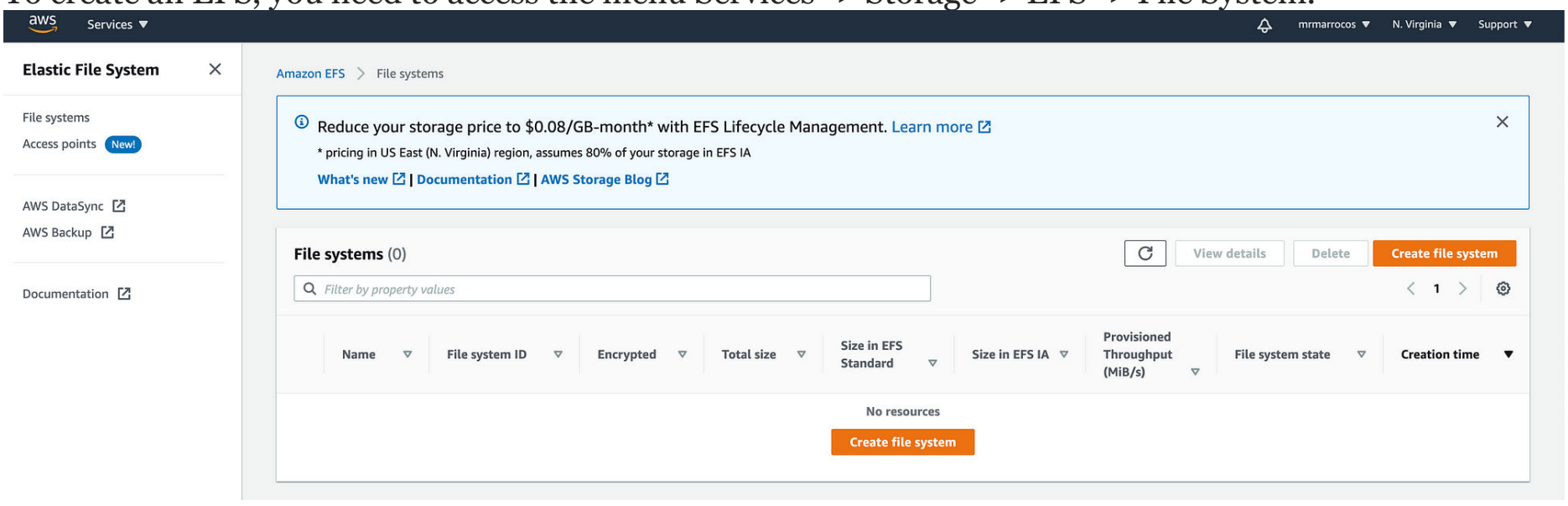
Go to this URL, and the static website hosted on the AWS S3 bucket will be accessible via browser:



10. Elastic File System (EFS)

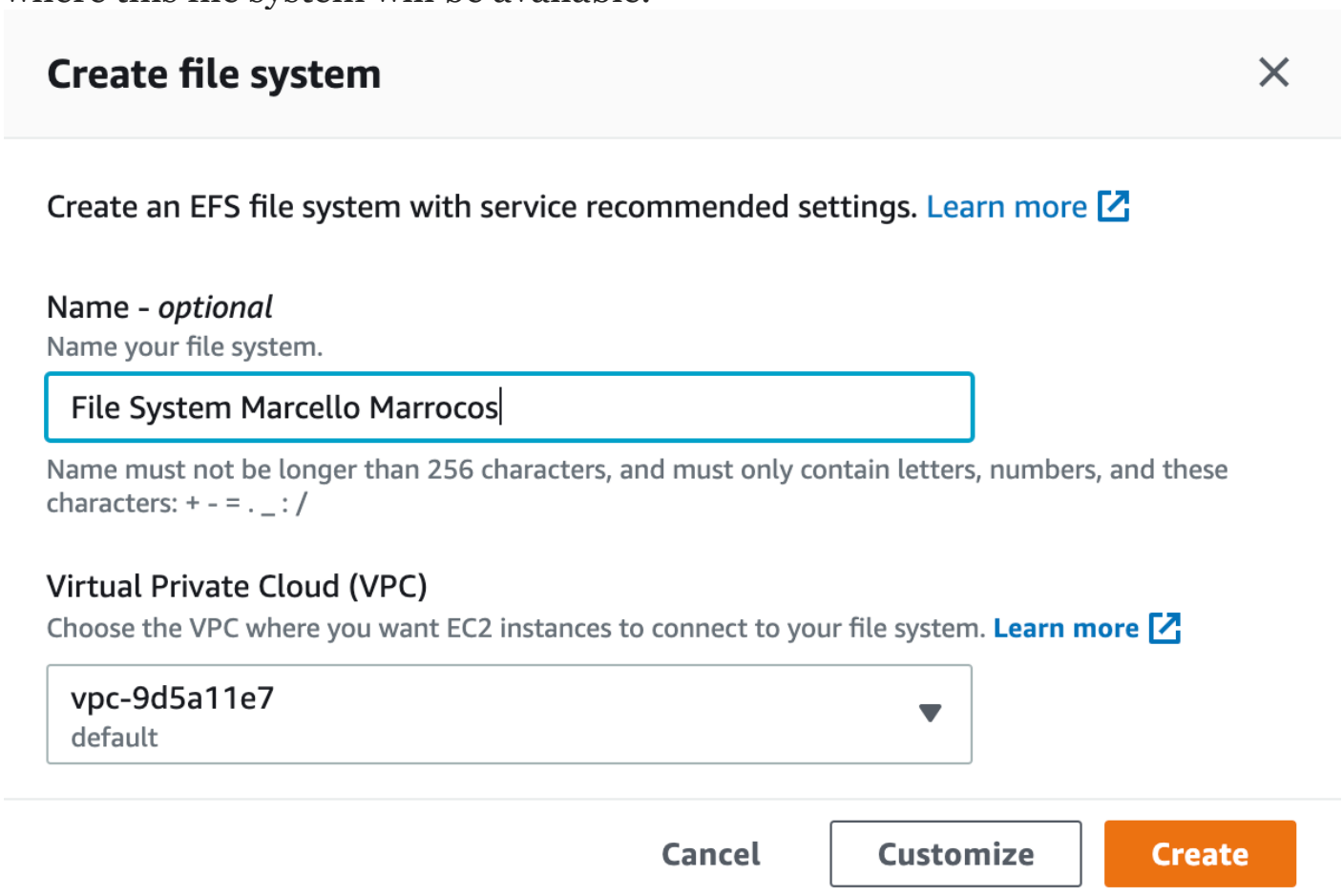
Creating & Deleting EFS

To create an EFS, you need to access the menu Services -> Storage -> EFS -> File System.



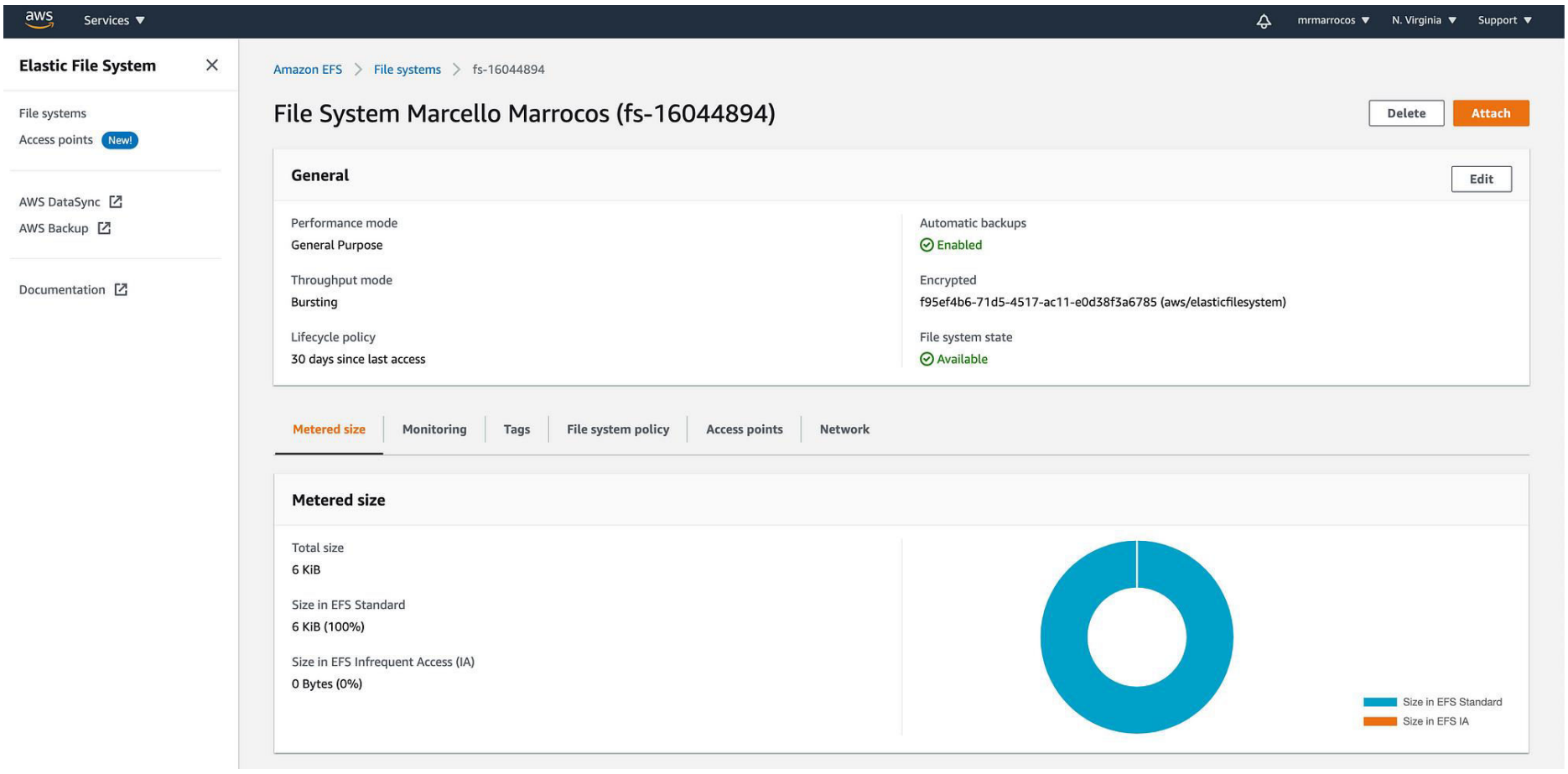
Screenshot by the Author

Click on the button "Create file system" to open the dialog. The field "name" is optional. However, I encourage you to fill out for easy identification. It is also required to select the VPC (Virtual Private Cloud) where this file system will be available.



Screenshot by the Author

After hitting the "Create" button, your file system will be available within a few seconds:



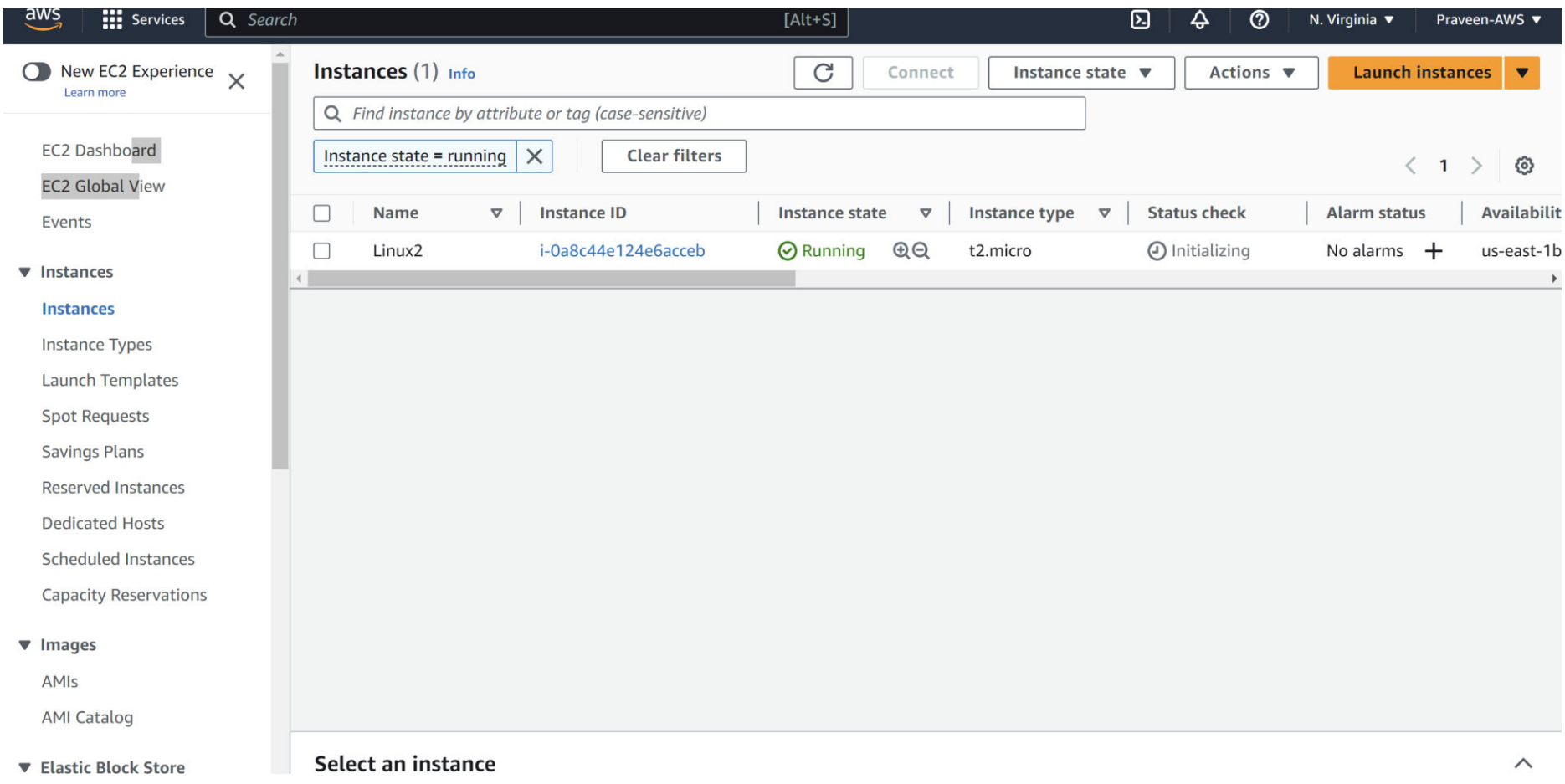
Screenshot by the Author

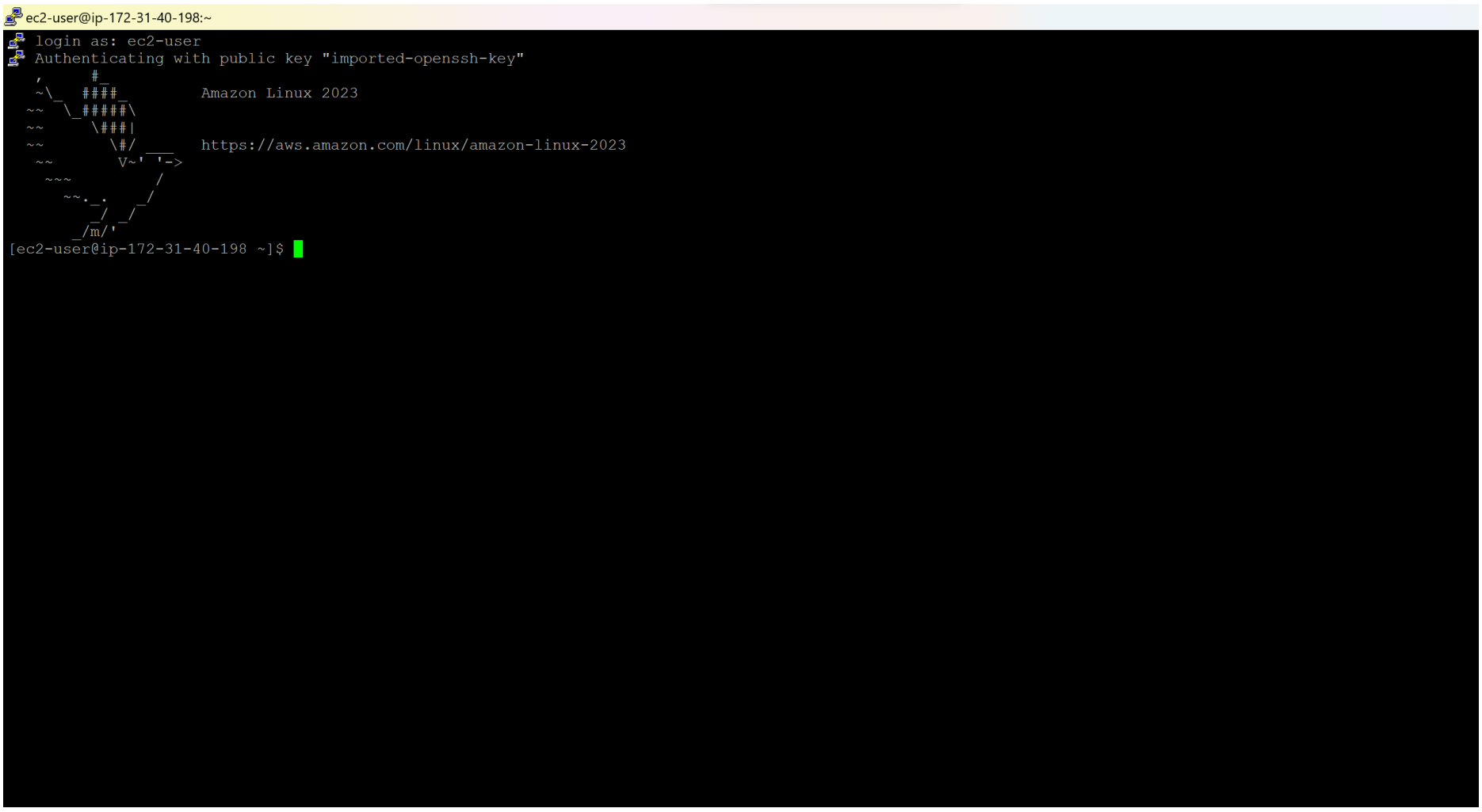
When clicking on your file system ID or file system name, it takes you to the details page of your EFS:

Attaching & Detaching EFS to Instance

Setup EC2 Instance

- Create an EC2 instance in AWS Linux 2 image in AWS console. The Instance must have an SSH connection allowed in its security group.
- Then connect your Instance via SSH from the local terminal using the IP address of the instance.

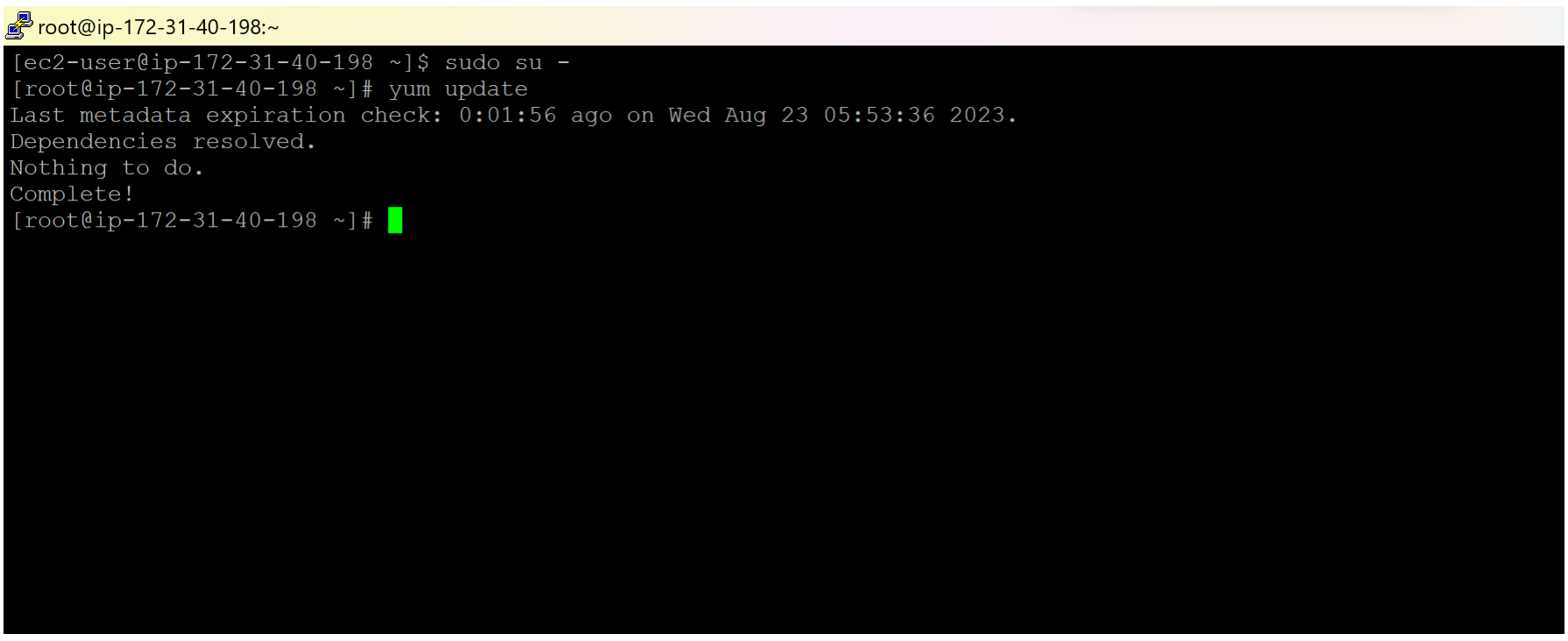




- Run the following commands to login as a **root** user and update the system.

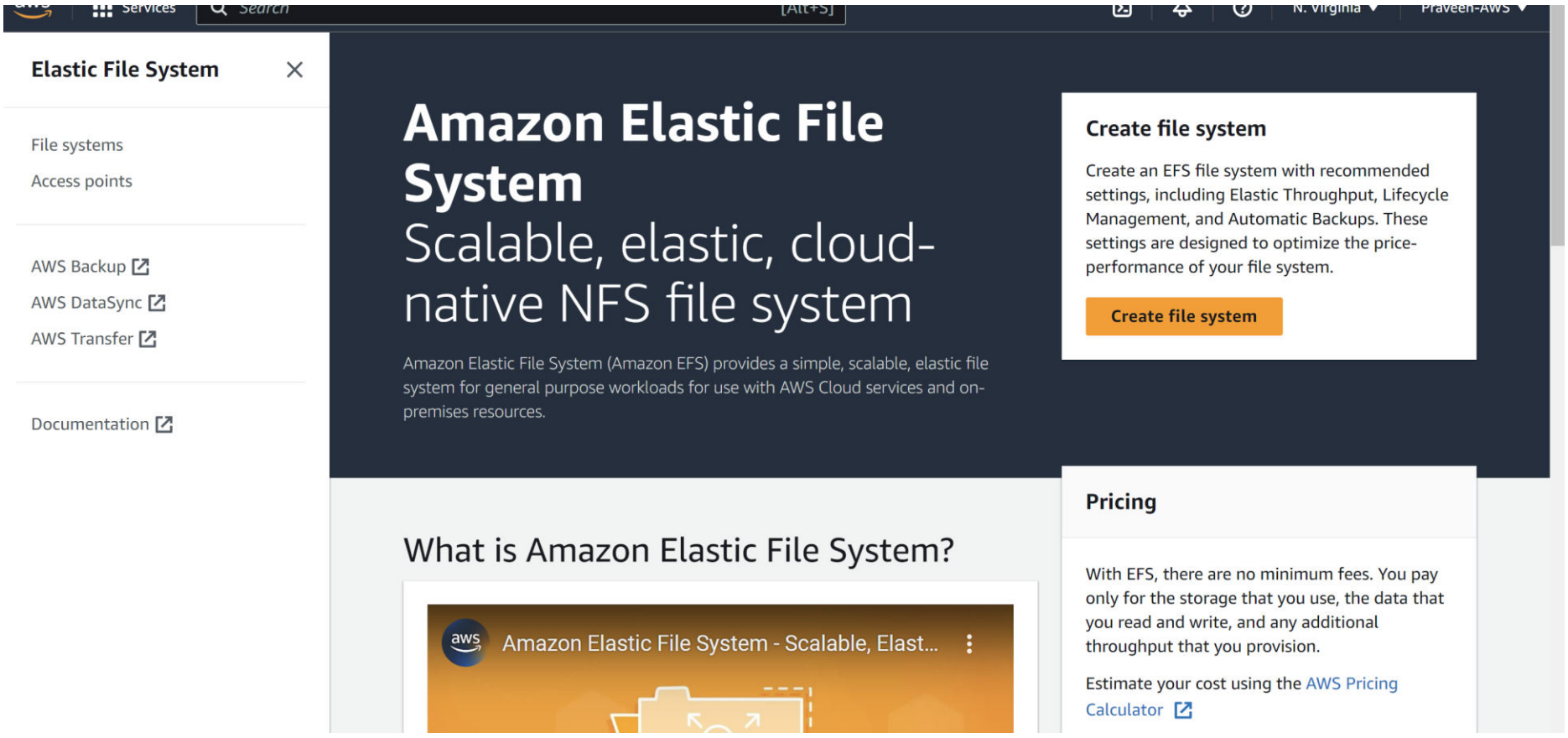
```
sudo su -
```

```
yum update
```

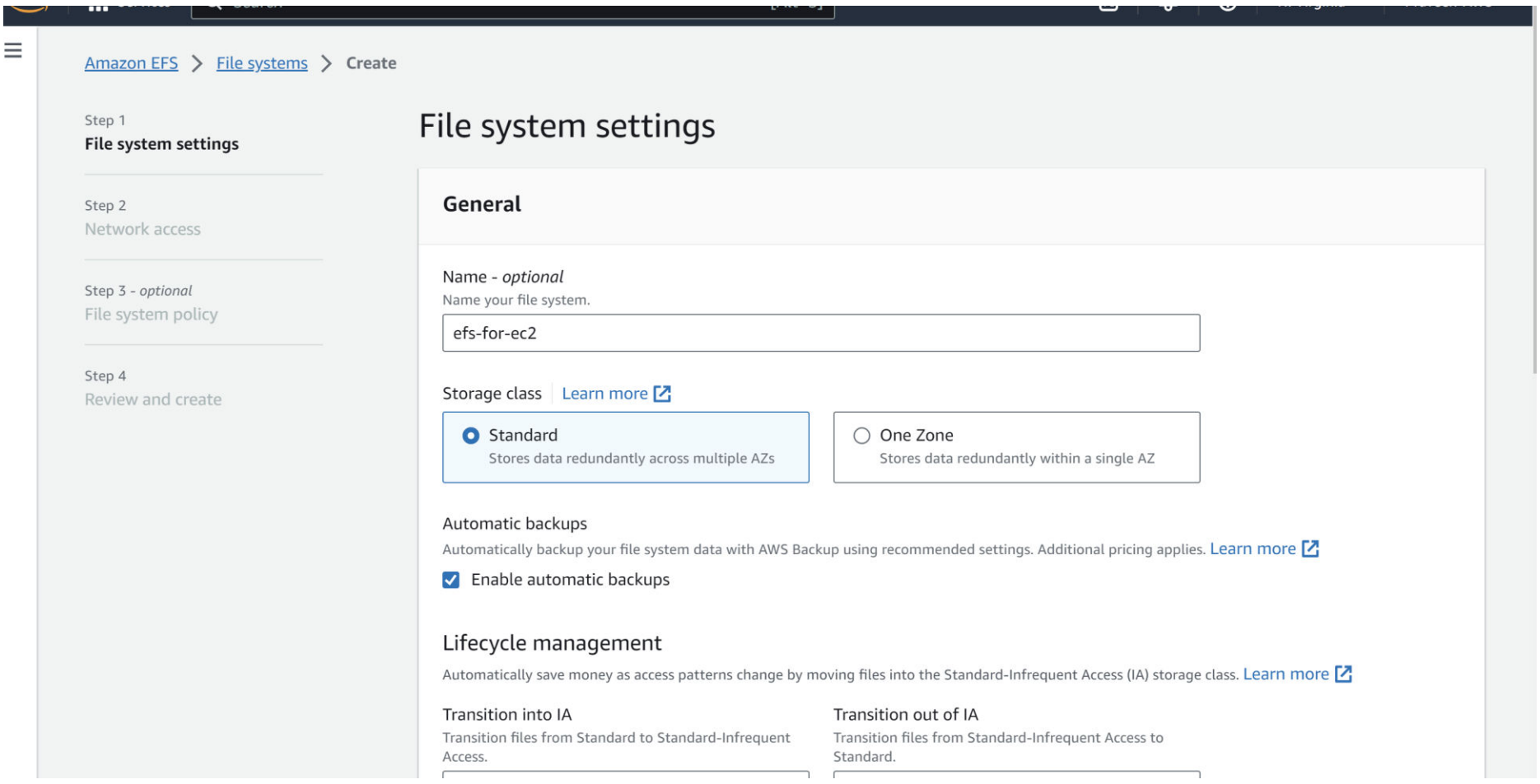


Creating Amazon Elastic File System

Open your AWS console and go to **All Services→ Storage→** select **EFS** under storage (direct link). Then Click **"Create file system."**



- Provide a Name for your file system.
- Select a VPC for EFS. For this tutorial we can select the default VPC.
- Select the storage class that you want to. As I mentioned in Introduction, the **Standard** type stores the data across multiple AZs, and the **One Zone** type stores them in a single AZ.
- Then click the **Create** button.



Lifecycle management

Automatically save money as access patterns change by moving files into the Standard-Infrequent Access (IA) storage class. Learn more

Transition into IA
Transition files from Standard to Standard-Infrequent Access.
30 day(s) since last access

Transition out of IA
Transition files from Standard-Infrequent Access to Standard.
None

Encryption
Choose to enable encryption of your file system's data at rest. Uses the AWS KMS service key (aws/elasticfilesystem) by default. Learn more

- [x] Enable encryption of data at rest

▶ Customize encryption settings

Performance settings

Throughput mode
Choose a method for your file system's throughput limits. Learn more

- (•) Enhanced
  Provides more flexibility and higher throughput levels for workloads with a range of performance requirements.
- ( ) Bursting
  Provides throughput that scales with the amount of storage for workloads with basic performance requirements.

- (•) Elastic (Recommended)

Performance settings

Throughput mode
Choose a method for your file system's throughput limits. Learn more

- (•) Enhanced
  Provides more flexibility and higher throughput levels for workloads with a range of performance requirements.
- ( ) Bursting
  Provides throughput that scales with the amount of storage for workloads with basic performance requirements.

- (•) Elastic (Recommended)
  Use this mode for workloads with unpredictable I/O. With Elastic mode, your throughput scales automatically and you only pay for what you use.
- ( ) Provisioned
  Use this mode if you can estimate your workload's throughput requirements. With Provisioned mode, you configure your file system's throughput and pay for throughput provisioned.

▶ Additional settings

▶ **Tags** optional

Cancel Next

Step 1
File system settings

Step 2
Network access

Step 3 - *optional*
File system policy

Step 4
Review and create

Network access

Network

Virtual Private Cloud (VPC) | Learn more

Choose the VPC where you want EC2 instances to connect to your file system.

vpc-0e5961525db3f7396
default

Mount targets

A mount target provides an NFSv4 endpoint at which you can mount an Amazon EFS file system. We recommend creating one mount target per Availability Zone. Learn more

Availability zone	Subnet ID	IP address	Security groups	
us-east-1a	subnet-08ac3…	*Automatic*	*Choose securi…*	Remove
			sg-09cf98f904b02ff60 default	
us-east-1b	subnet-04bc…	*Automatic*	*Choose securi…*	Remove
			sg-09cf98f904b02ff60 default	
us-east-1e	subnet-0b29…	*Automatic*	*Choose securi…*	Remove
			sg-09cf98f904b02ff60 default	
us-east-1f	subnet-04ec1…	*Automatic*	*Choose securi…*	Remove
			sg-09cf98f904b02ff60 default	

Add mount target

You can only create one mount target per Availability Zone.

Cancel | Previous | Next

Step 1
File system settings

Step 2
Network access

Step 3 - *optional*
File system policy

Step 4
Review and create

File system policy - *optional*

Policy options

Select one or more of these common policy options, or create a custom policy using the editor. Learn more

- [ ] Prevent root access by default*
- [ ] Enforce read-only access by default*
- [ ] Prevent anonymous access
- [ ] Enforce in-transit encryption for all clients

* Identity-based policies can override these default permissions.

▶ **Grant additional permissions**

Policy editor {JSON}

Clear

1

Manual changes will prevent the use of the policy options on the left until the editor is cleared.

Cancel Previous Next

Step 1
File system settings

Step 2
Network access

Step 3 - *optional*
File system policy

Step 4
Review and create

Review and create

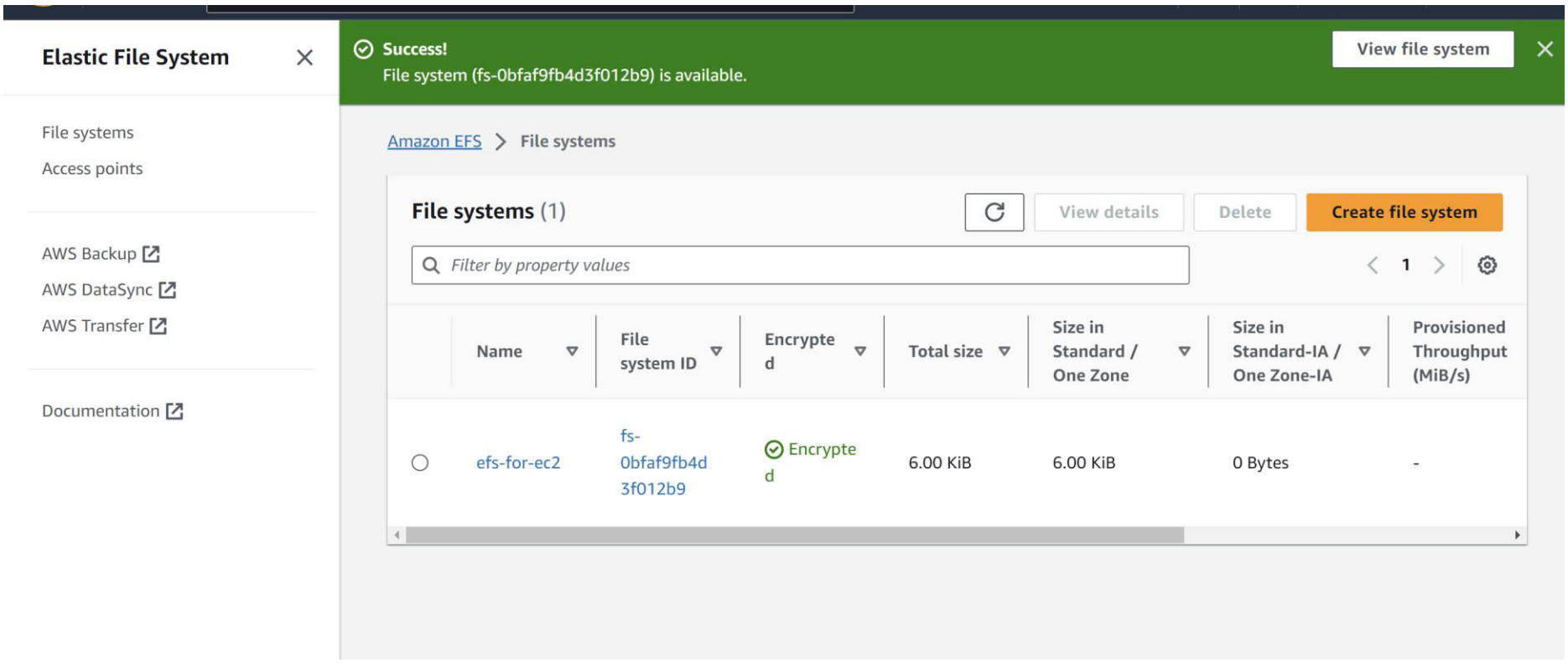
Step 1: File system settings

Edit

File system

Field	Value	Is editable?
Name	efs-for-ec2	Yes
Performance mode	General Purpose	No
Throughput mode	Elastic	Yes
Encrypted	Yes	No
KMS Key ID	-	No
Lifecycle management	Transition into IA: 30 day(s) since last access Transition out of IA: None	Yes

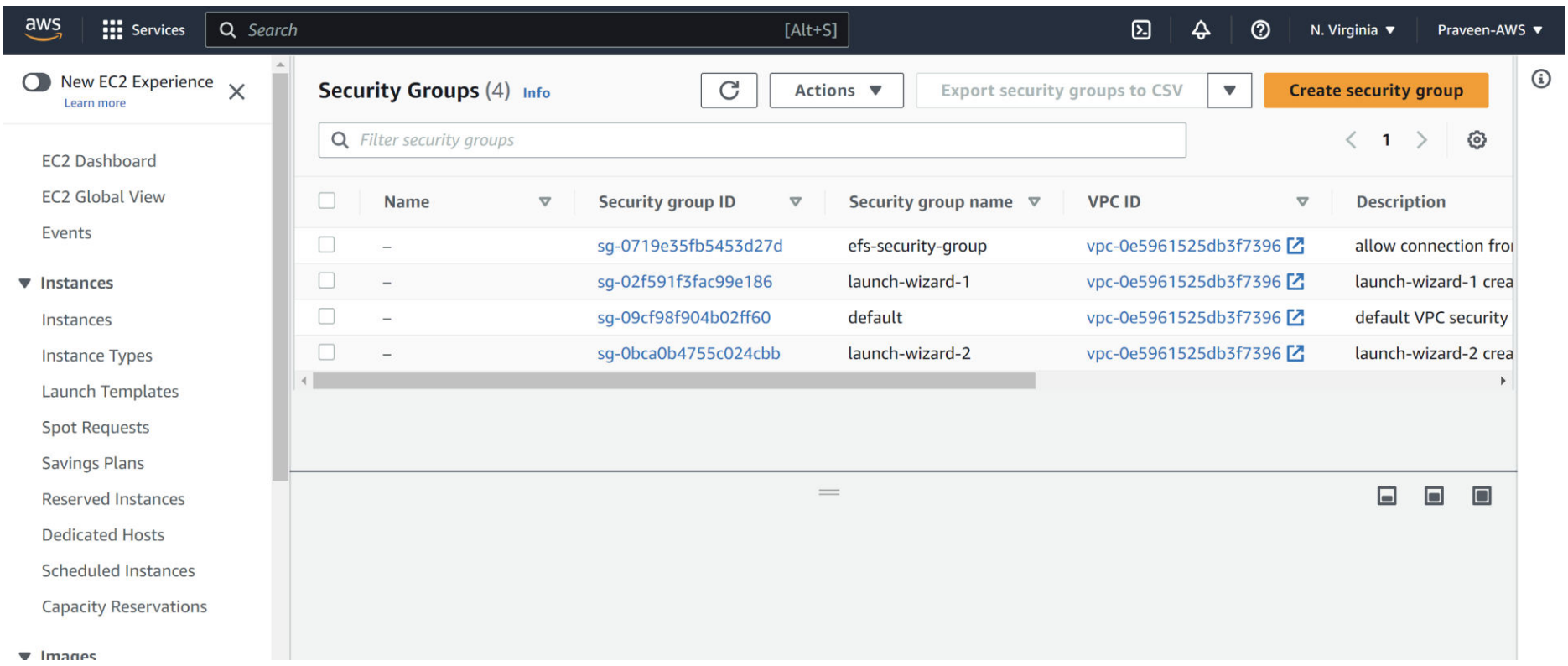
Cancel Previous Create



Creating Security Group for EFS

Once you complete creating EFS, create the security group of EFS as shown in the below steps.

Navigate to EC2, find **Security Groups** under the **Network & Security**. Click it and then click the **Create Security Group** button.



- Provide a name for your Security group for EFS.
- Select VPC which one you previously created for your **EFS file system**. EFS and its Security group must be in the same VPC.
- Scroll down and under the inbound rules section click the **Add rule** button, and select the Type NFS. Select source to **Anywhere**.

EC2 > Security Groups > Create security group

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name Info

efs-security-group

Name cannot be edited after creation.

Description Info

allow connection from ec2 to efs

VPC Info

vpc-0e5961525db3f7396

Inbound rules Info

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
NFS	TCP	2049	Anywh... 0.0.0.0/0		Delete

Add rule

Outbound rules Info

Type Info	Protocol Info	Port range Info	Destination Info	Description - optional Info	
All traffic	All	All	Custom 0.0.0.0/0		Delete

Add rule

Tags - *optional*

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.

No tags associated with the resource.

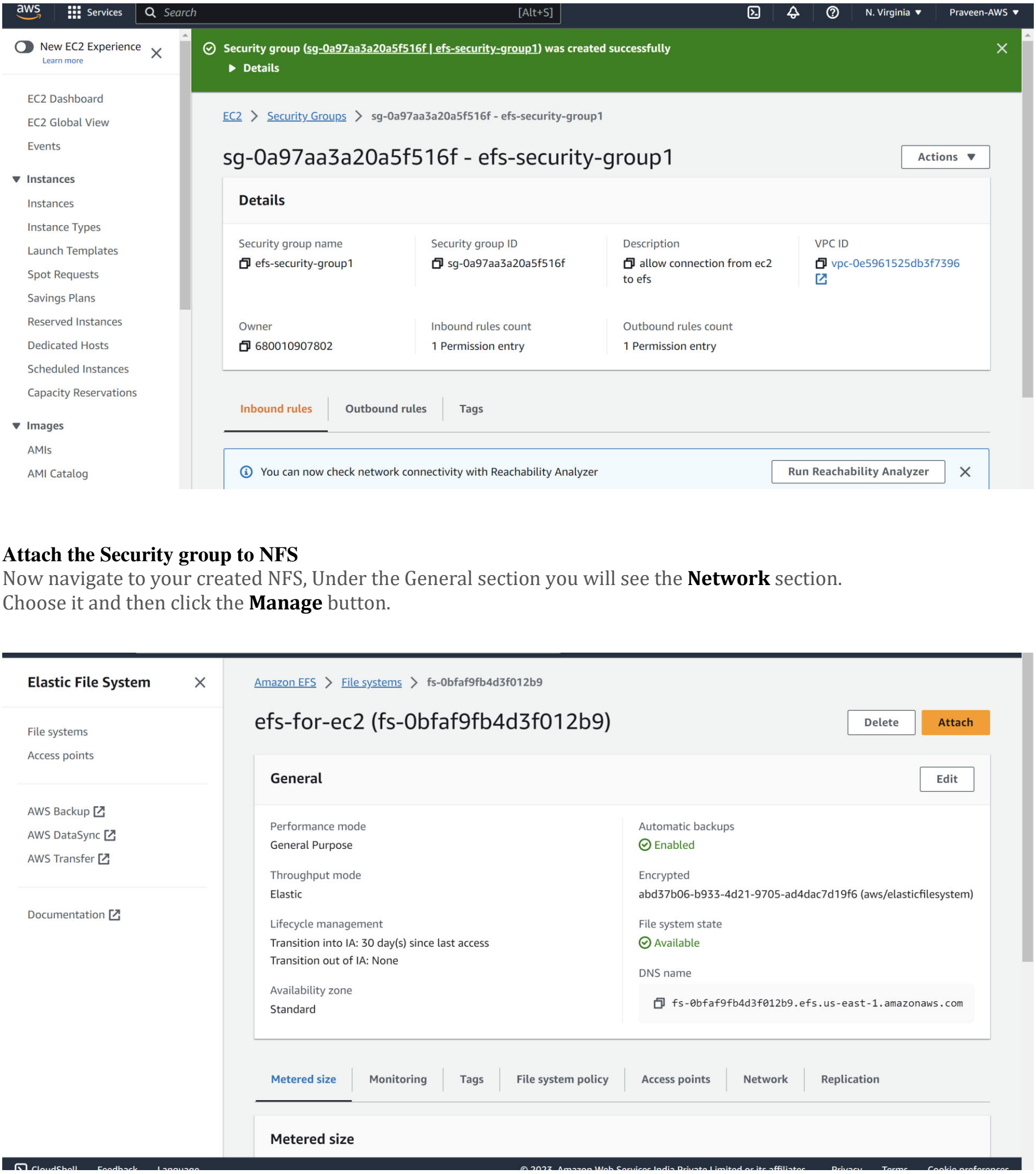
Add new tag

You can add up to 50 more tags

Cancel Create security group

Attach the Security group to NFS

Now navigate to your created NFS, Under the General section you will see the **Network** section. Choose it and then click the **Manage** button.



File systems
Access points

AWS Backup
AWS DataSync
AWS Transfer

Documentation

Metered size | Monitoring | Tags | File system policy | Access points | Network | Replication

Network

Manage

Availability zone	Mount target ID	Subnet ID	Mount target state	IP address	Network interface ID	Security groups
us-east-1a	fsmt-030895103cab72c34	subnet-08ac375e9fbfcb4b9	Available	172.31.24.235	eni-092a1a7635299b0df	sg-09cf98f904b02ff60 (default)
us-east-1b	fsmt-0b86c3cf5293dcbb4	subnet-04bcb85fec602cb59	Available	172.31.42.171	eni-09e60c451f077845a	sg-09cf98f904b02ff60 (default)
us-east-1c	fsmt-0162035fefeb3d52a	subnet-0b8bca925b733e383	Available	172.31.9.0	eni-0939d14fabcc25921	sg-09cf98f904b02ff60 (default)
us-east-1d	fsmt-0113aefa8568fa77a	subnet-0ada16d35362e7851	Available	172.31.80.254	eni-028b6df595de9c4d2	sg-09cf98f904b02ff60 (default)

Elastic File System

File systems
Access points

AWS Backup
AWS DataSync
AWS Transfer

Documentation

Virtual Private Cloud (VPC)
Choose the VPC where you want EC2 instances to connect to your file system.

vpc-0e5961525db3f7396
default

You must delete all existing mount targets in order to change the VPC of your file system.

Mount targets

A mount target provides an NFSv4 endpoint at which you can mount an Amazon EFS file system. We recommend creating one mount target per Availability Zone. Learn more

Availability zone	Subnet ID	IP address	Security groups	
us-east-1a	subnet-08ac375e9fb	172.31.24.235	Choose securit… ▼ sg-09cf98f904b02ff60 default	Remove
us-east-1b	subnet-04bcb85fec6	172.31.42.171	Choose securit… ▼ sg-09cf98f904b02ff60 default	Remove

Elastic File System

File systems
Access points

AWS Backup
AWS DataSync
AWS Transfer

Documentation

You must delete all existing mount targets in order to change the VPC of your file system.

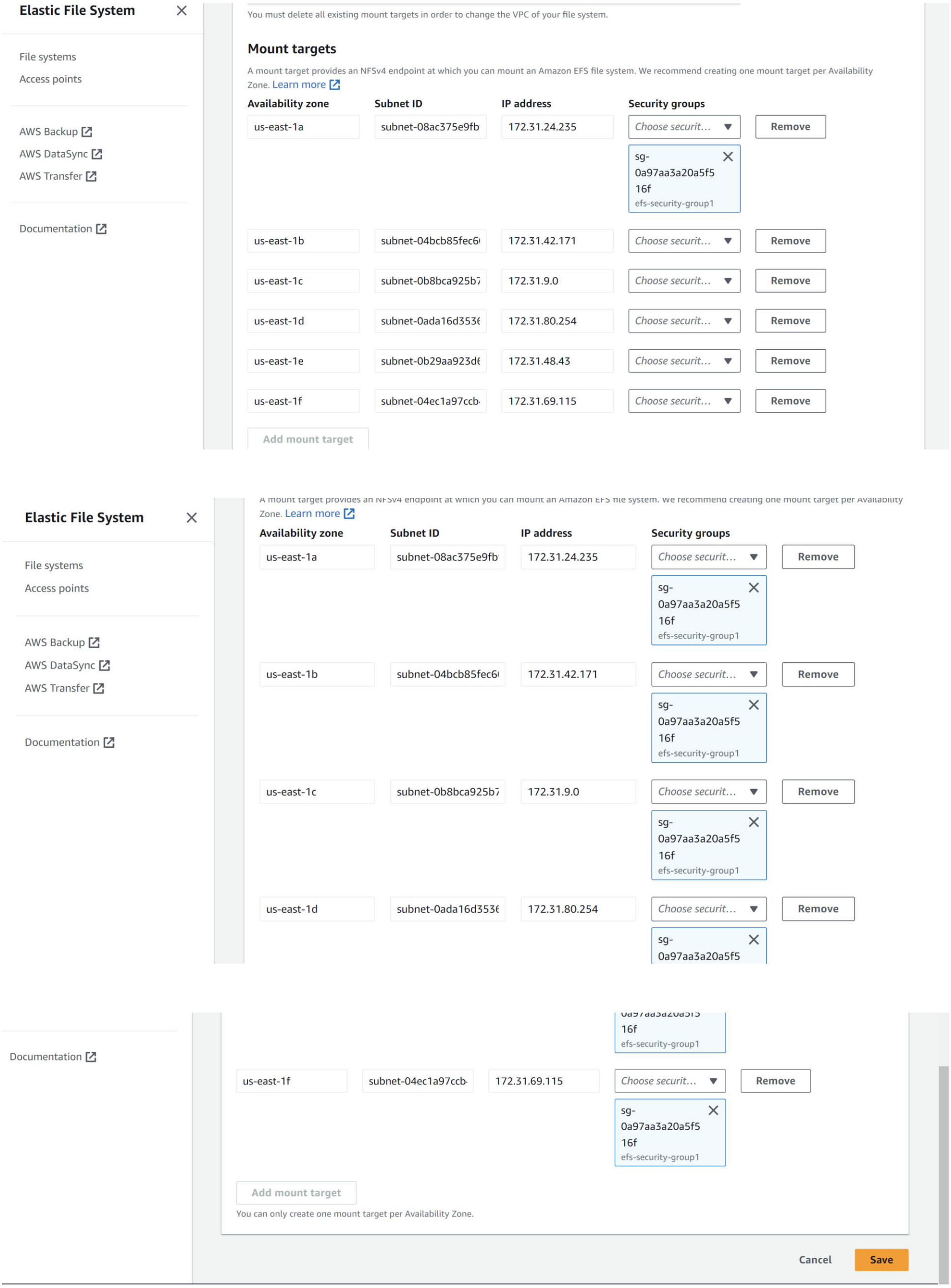
Mount targets

A mount target provides an NFSv4 endpoint at which you can mount an Amazon EFS file system. We recommend creating one mount target per Availability Zone. Learn more

Availability zone	Subnet ID	IP address	Security groups	
us-east-1a	subnet-08ac375e9fb	172.31.24.235	Choose securit… ▼	Remove
us-east-1b	subnet-04bcb85fec6	172.31.42.171	Choose securit… ▼	Remove
us-east-1c	subnet-0b8bca925b7	172.31.9.0	Choose securit… ▼	Remove
us-east-1d	subnet-0ada16d353€	172.31.80.254	Choose securit… ▼	Remove
us-east-1e	subnet-0b29aa923d€	172.31.48.43	Choose securit… ▼	Remove
us-east-1f	subnet-04ec1a97ccb·	172.31.69.115	Choose securit… ▼	Remove

Add mount target

You can only create one mount target per Availability Zone.



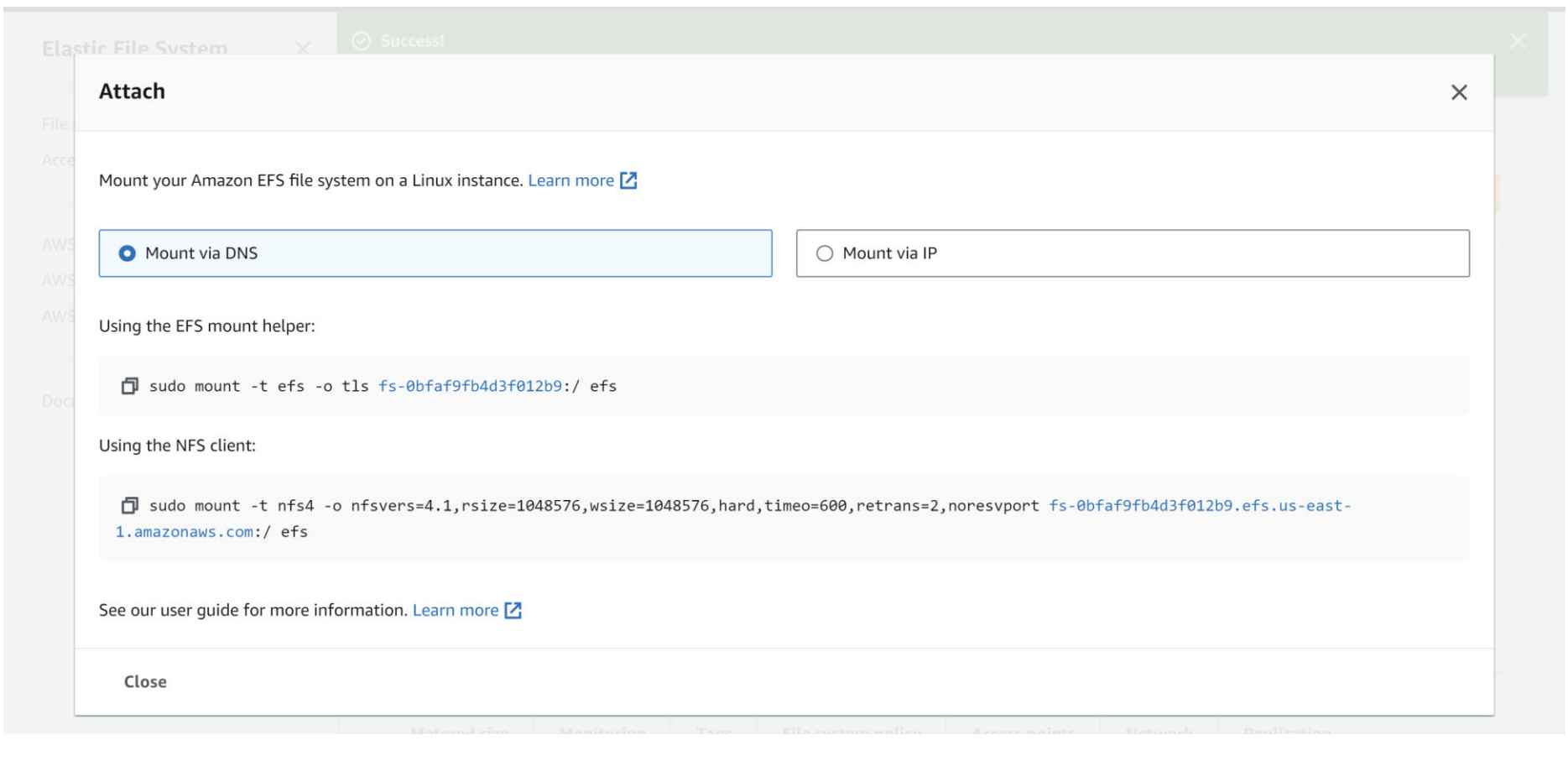
Attaching Elastic File System (EFS) to EC2 instance

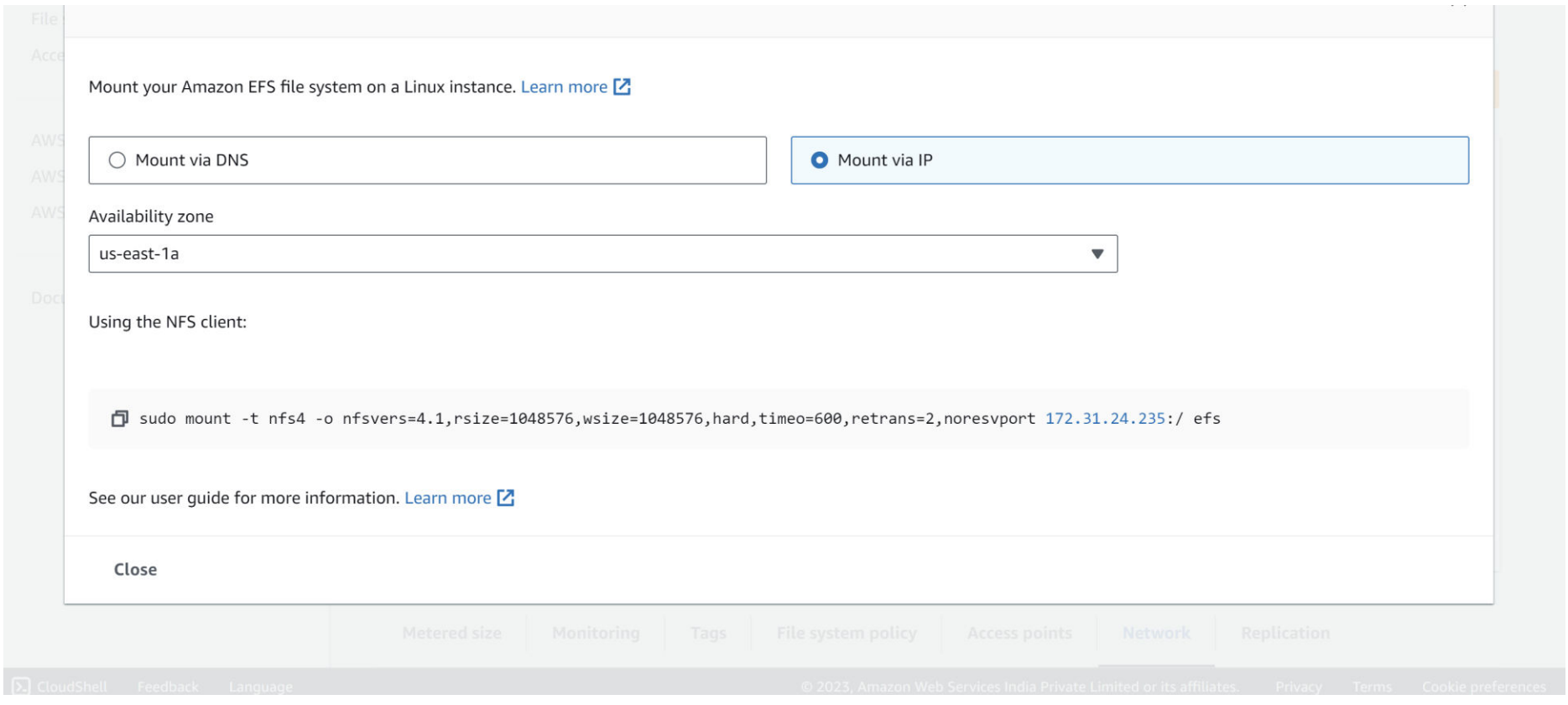
Now your EFS is ready to be mounted in an EC2 instance.

Use one of the following method to mount the EFS in your desired folder. For example, let us mount our EFS under **/efs**

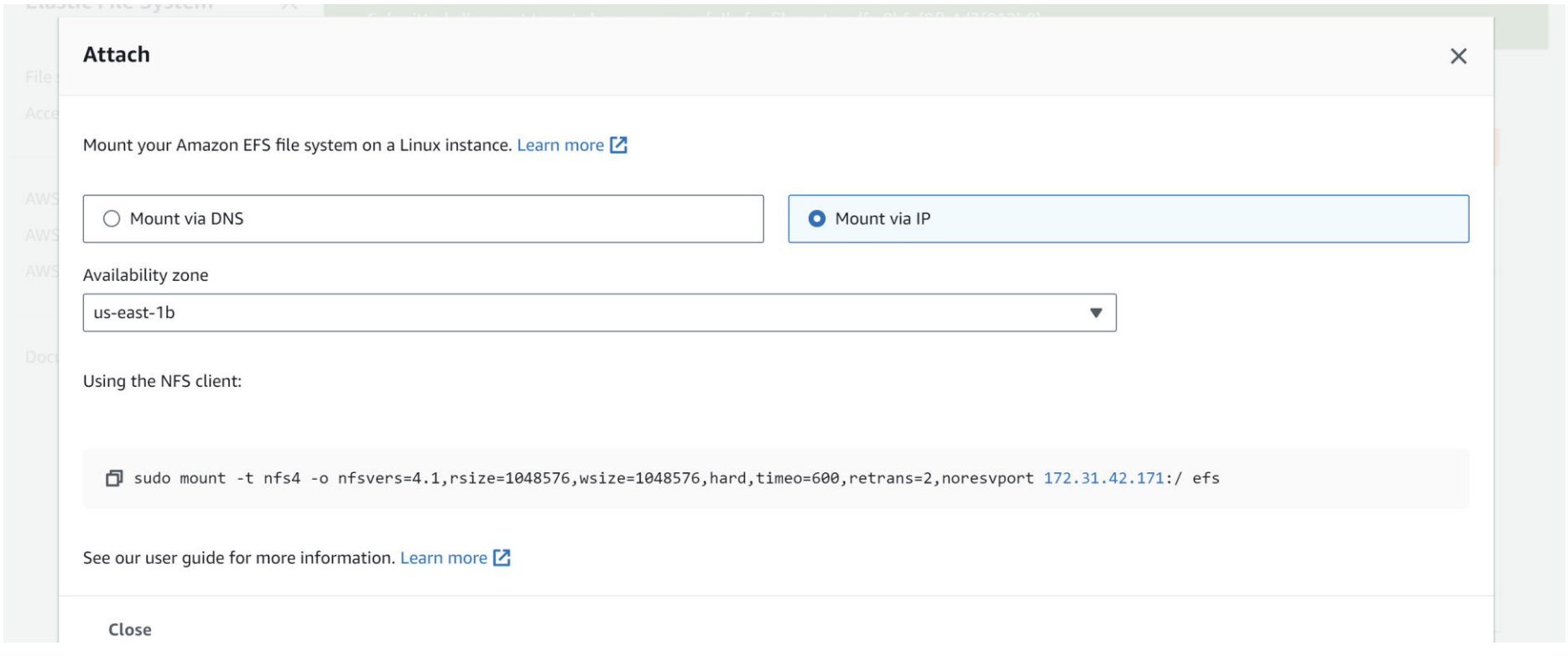
Go to your terminal and type '**mkdir efs** command.

In your File system's General page you can see the **Attach** button in the top right corner. Click the button and you can see two options. Either one of them you can use.



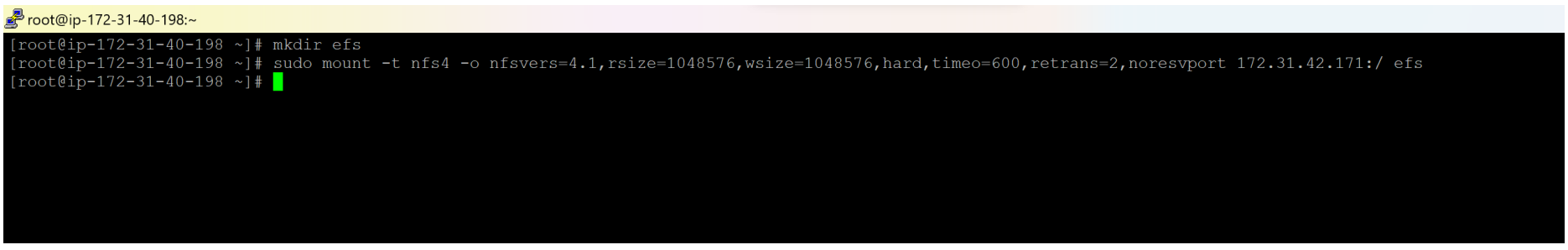


Select the second option and under the NFS Client you can see the command for mount EFS with IP address. Before that we have to select an availability zone like the picture below.



Then you are able to see the command with an IP Address. So copy the command and paste it in your instance terminal.

sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport 172.31.42.171:/ efs



You have successfully mounted the EFS to your EC2 instance. you can check your output using the **'df -h'** command from the console.

root@ip-172-31-40-198:~

```
[root@ip-172-31-40-198 ~]# mkdir efs
[root@ip-172-31-40-198 ~]# sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport 172.31.42.171:/ efs
[root@ip-172-31-40-198 ~]# df -h
Filesystem       Size  Used Avail Use% Mounted on
devtmpfs         4.0M     0  4.0M   0% /dev
tmpfs            475M     0  475M   0% /dev/shm
tmpfs            190M  2.8M  188M   2% /run
/dev/xvda1        10G  1.5G  8.5G  16% /
tmpfs            475M     0  475M   0% /tmp
tmpfs             95M     0   95M   0% /run/user/1000
172.31.42.171:/  8.0E     0  8.0E   0% /root/efs
[root@ip-172-31-40-198 ~]#
```

11. Route 53

DNS Records overview

As we all use internet, so obviously we have used DNS. DNS is used to convert domain names (like http://medium.com) into an Internet Protocol IP address such as (http://87.193.63.1)

We have two types of IP addresses IPv4 (32 bit field, apx 4 billion different addresses)& IPv6 (128bits).

if you want to know how DNS name resolves query, please check my below blog on same.

Let's head over to our main topic i.e Route53.

In AWS, Route53 is global managed DNS (Domain Name System) & we already know DNS is a collection of rules and records which helps clients understand how to reach a server through URLs.

DNS operates on port 53. Amazon decided to call it route 53 so that's where the name comes from.

It's a global service. You need to buy a domain in order to work with Route53, Go to Route53 Service & Click on register domain. Enter the domain name & check availability, Add to cart & click on continue.

Route53 can use basically:

- Public domain names you own (or buy) or Private domain names that can be resolved by your instances in your VPCs.
- Route53 has many features such as **Load balancing, Health checks, Routing policy like Simple, Failover, Geolocation, Latency, Weighted, Multi value.**
- You pay $0.50 per month per hosted zone.

In AWS Route53, we have many types of records. Let's talk about various records.

1- SOA (Start of Authority Records)

Basic SOA stores information about below things.

- Name of Server that supplied the data for zone.
- The administrator of that zone & current version of data file.

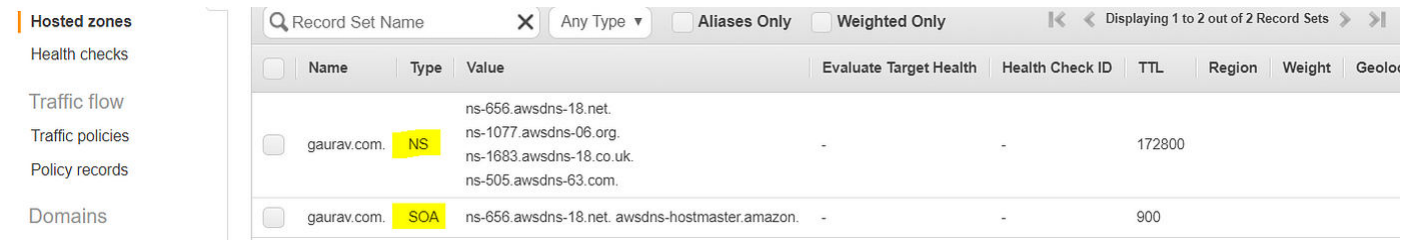
Eg:

ns-2048.awsdns-64.net. hostmaster.example.com. 1 7200 900 1209600 86400
Route53 Name server that create SOA record: ns-2048.awsdns-64.net.
Email Address of Administrator: *hostmaster.example.com*

2- NS Record (Name Server Records)

NS records is basically your name server records which are used by top level domain servers to direct traffic to content DNS server which contains the authoritative records.

So whenever we create a hosted zone in Route53, Two types of records automatically created, one is SOA & second is NS.



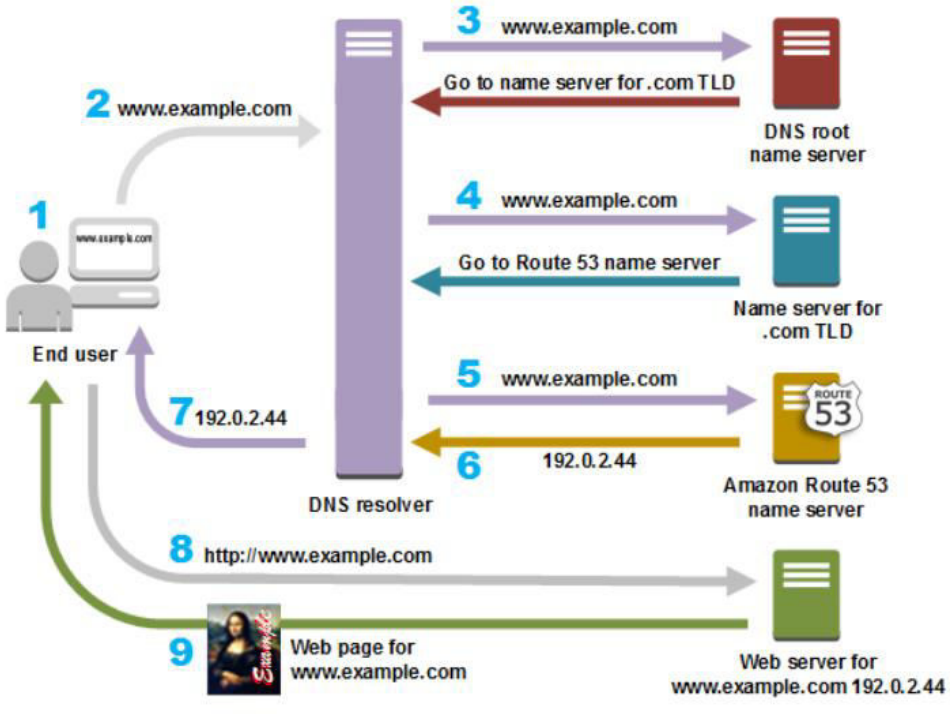
Hosted Zone(SOA & NS Records)

Before we head over to important type of records used in AWS, let's talk about one important concept TTL(Time to Live)

TTL (Time to Live):

TTL is mandatory for each DNS record. So TTL is length that a DNS records is cached on either the resolving server or user own Laptop. The Lower the TTL, the faster changes to DNS records. Whenever you created record set, you need to define TTL for it.

TTL(Time to Live)



How Amazon Route 53 Routes Traffic for Your Domain (Source AWS Official)

Let’s talk about the most common records in AWS.

When you create basic records, you specify the following values.

- Name
- Type
- Alias
- TTL (Time to Live)
- Value
- Routing Policy

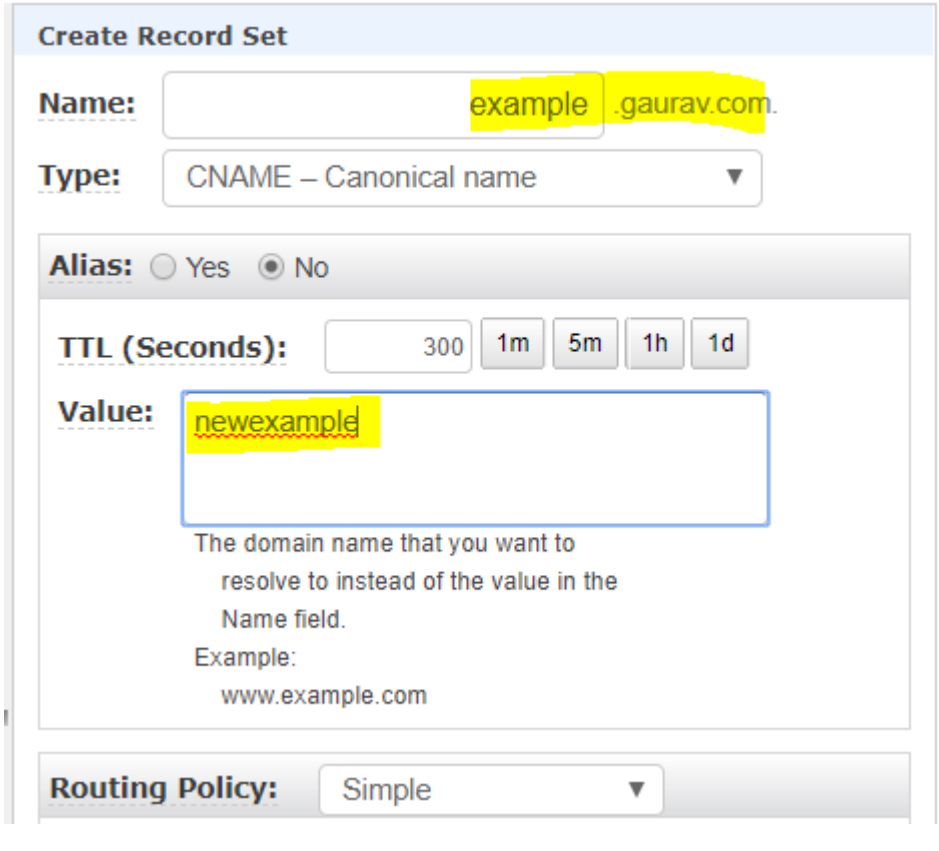
1- A Record (URL to IPv4)

The “A” record stands for Address record. The A record is used by computer to translate the name of the domain to an IP address.

Eg: (http://medium.com might point to [http://126.78.98.90)](http://126.78.98.90)

2- CNAME (Canonical Records- URL to URL)

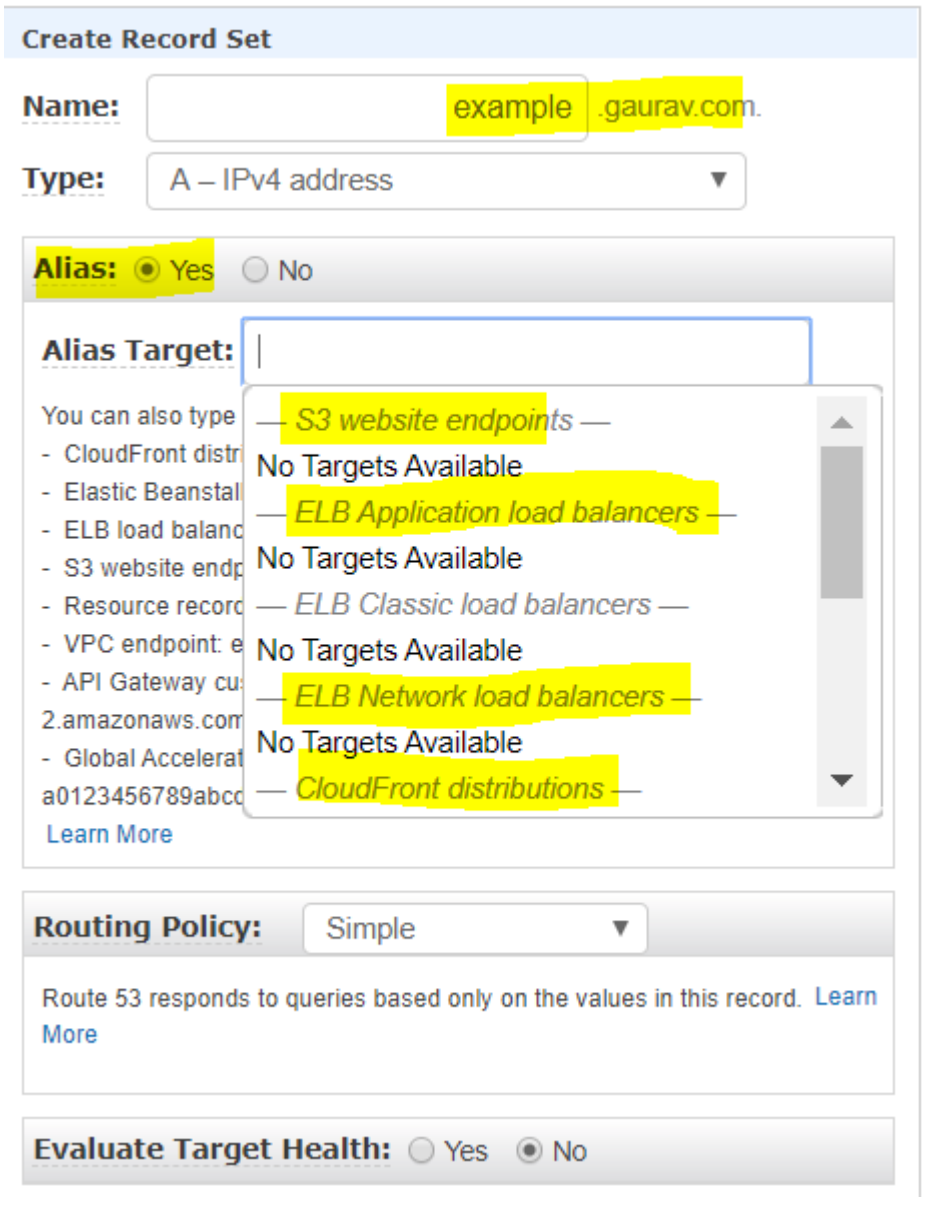
CNAME Points a URL to any other URL. (gaurav.gupta.com => gkg.example.com), We use it only for Non-Root Domain(aka. something.mydomain.com)



Resolving example.gaurav.com to newexample.gaurav.com

3- Alias Record:

Alias record points a URL to an AWS Resource, Alias record are used to map resource record sets in your hosted zone to Elastic Load Balancer, CloudFront or S3 Buckets websites.



Alias Record

4- AAAA: (URL to IPv6)

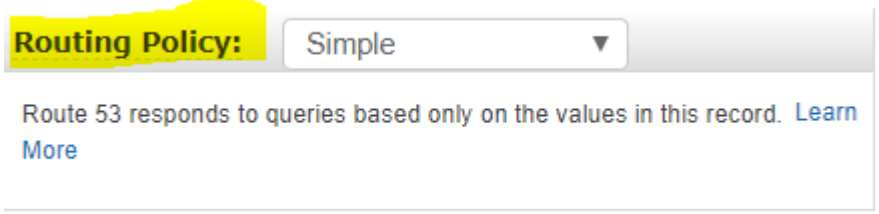
An **AAAA record** maps a domain name to the IP address (Version 6) of the computer hosting the domain. An **AAAA record** is used to find the IP address of a computer connected to the internet from a name.

5- MX Record (Main Exchange Record)

A mail Exchanger **record** (**MX record**) specifies the mail server responsible for accepting email messages on behalf of a domain name. It is a resource **record** in the Domain Name System (**DNS**). It is possible to configure several **MX records**, typically pointing to an array of mail servers for load balancing and redundancy.

This is all about various type records, let's talk about Routing policy. So what is routing policy.

When you create a record, you choose a routing policy, which determines how Amazon Route 53 responds to queries.



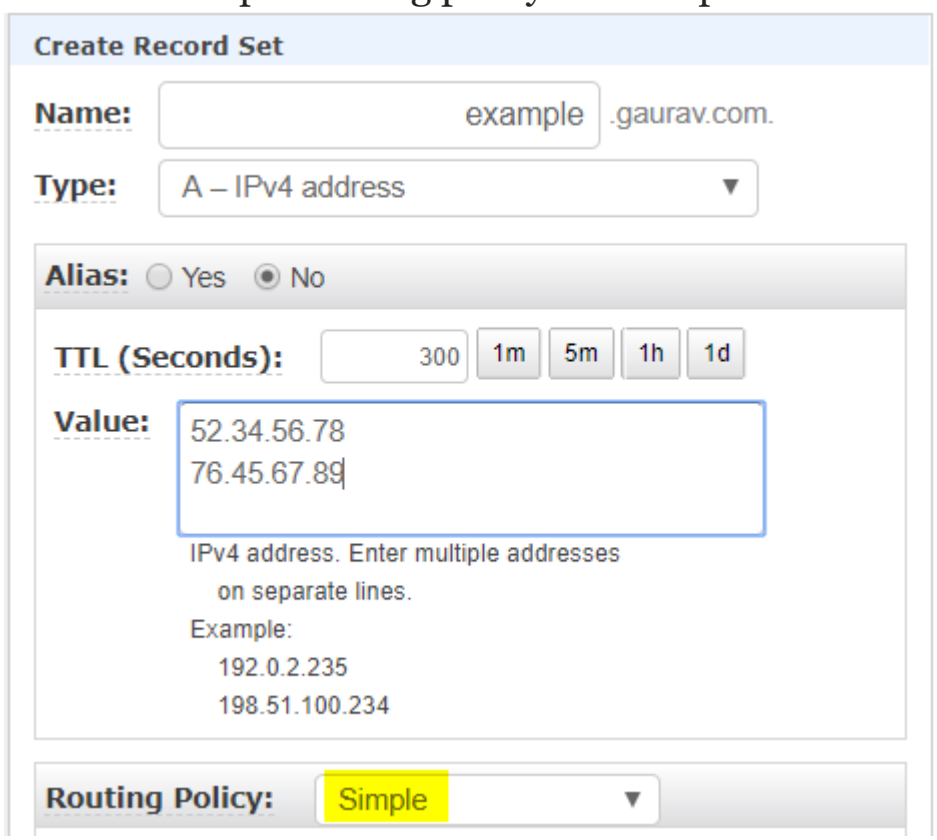
Routing Policy

There is total 6 types of routing policy in Route53, let's talk about one by one.

1- Simple Routing Policy:

In case of simple routing policy, you can have only one record with multiple IP addresses. If you specify multiple values in record, Route53 returns all values in random order to the user.

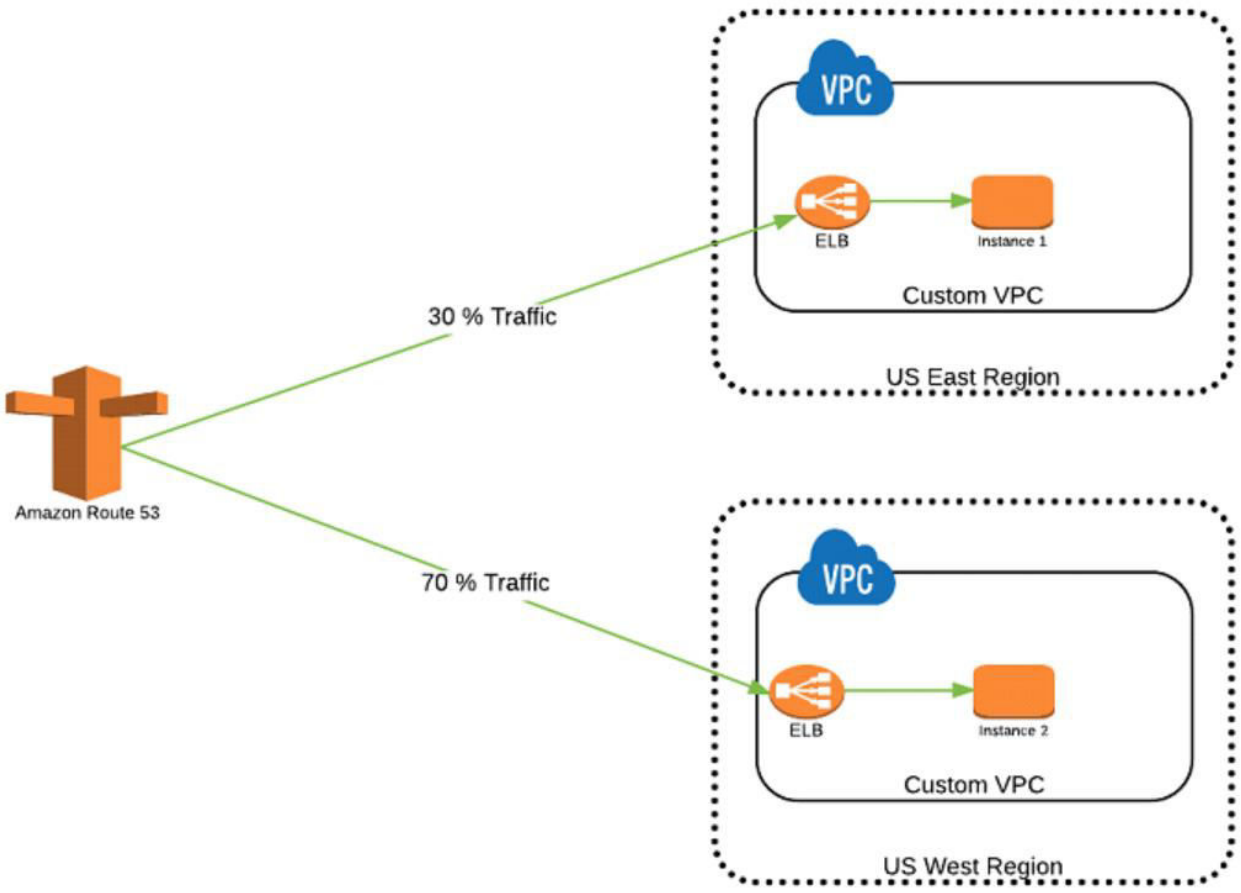
Maps a domain to one URL, Use when you need to redirect to a single resource. You can't attach health checks to simple routing policy. If multiple values are returned, a random one is chosen by the client.



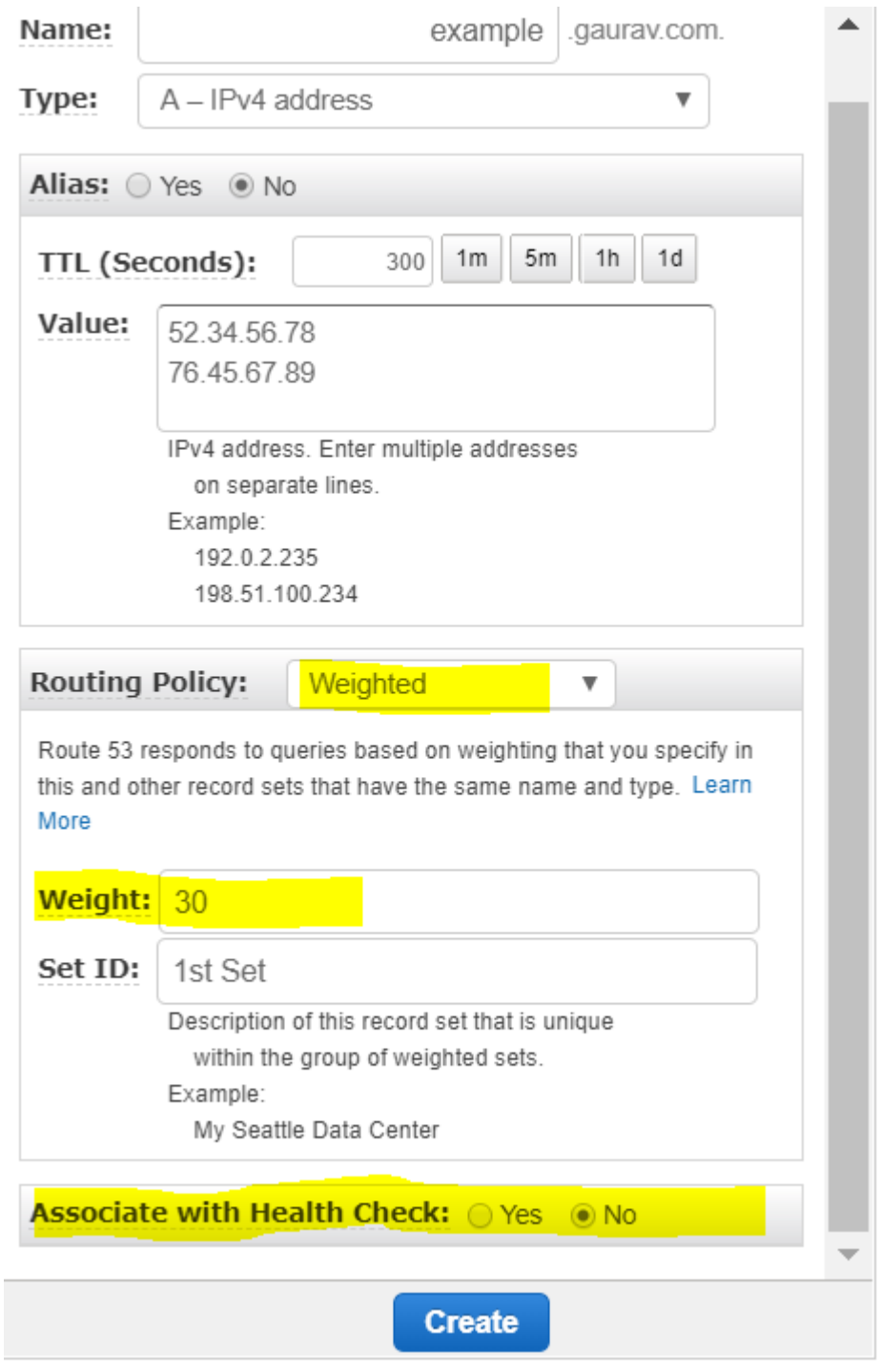
Simple Routing Policy

2- Weighted Routing Policy

Weighted Routing Policy controls the what percentage % of the requests that go to specific endpoint. It's helpful to test 1% of traffic on new app version. It is also helpful to split traffic between two regions. We can associate Health checks with it.

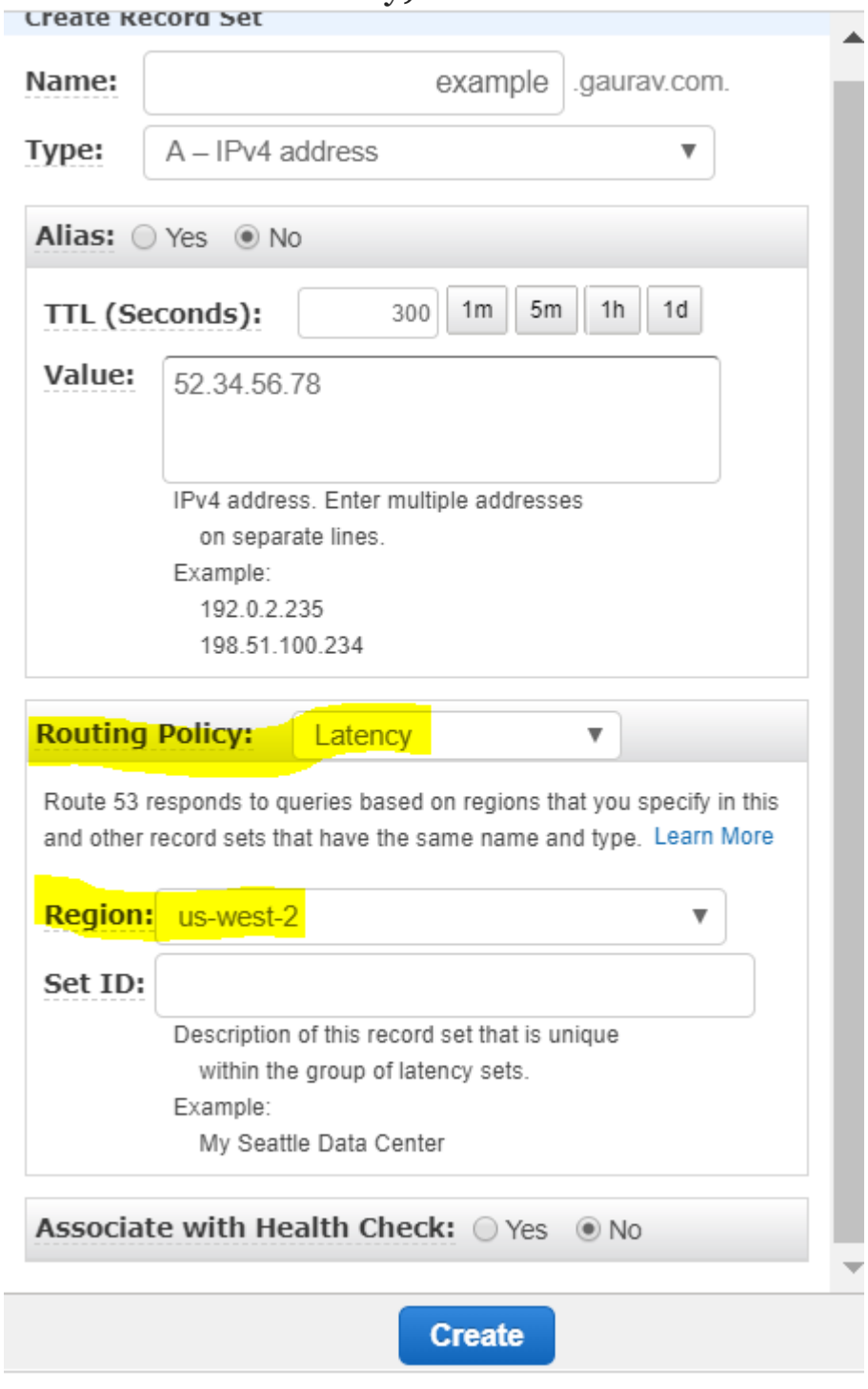


Weighted Policy



3- Latency Routing Policy

It allows you to route your traffic based on lowest network latency for your end user. It redirects to the server that has the least latency close to us also helpful when latency of users is a priority. Latency is evaluated in terms of user to designated AWS Region. For example: Germany may be directed to the US (if that's the lowest latency)

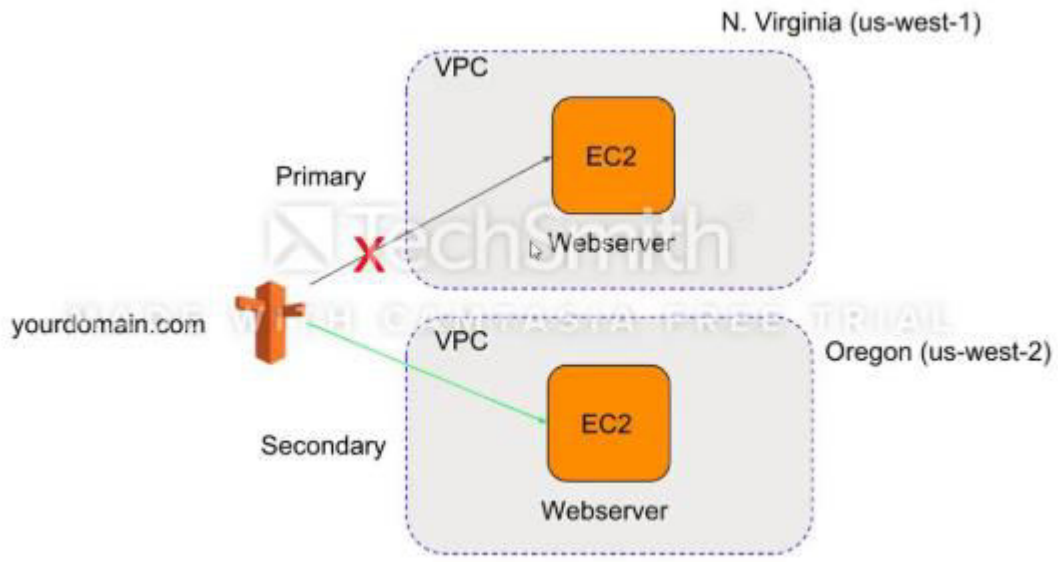


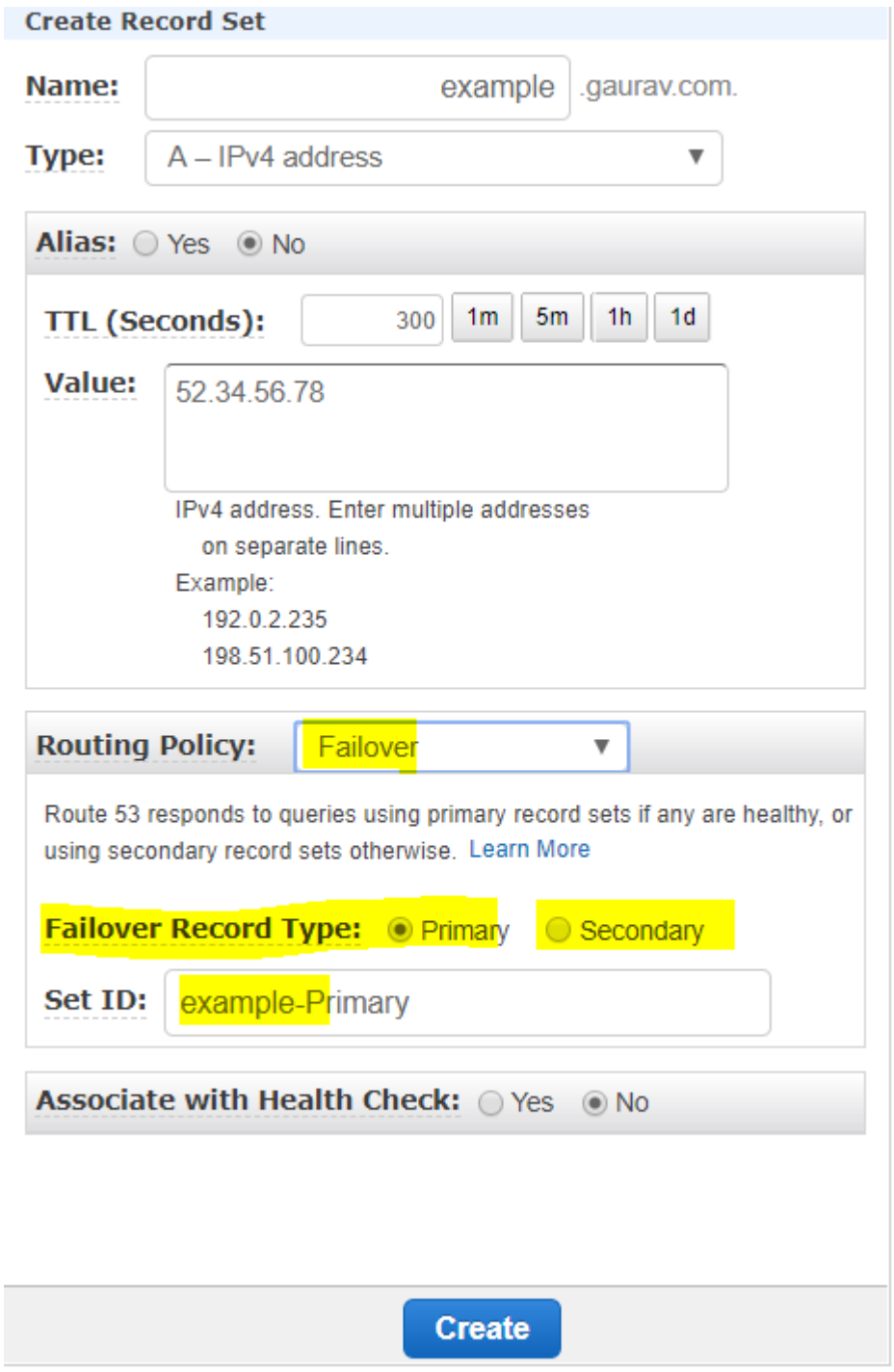
Latency Routing Polciy

4- Failover Routing Policy

Failover routing lets you **route** traffic to a resource when the resource is healthy or to a different resource when the first resource is unhealthy. The primary and secondary records can **route** traffic to anything from an Amazon S3 bucket that is configured as a website to a complex tree of records.

You can associate health check with this type of policy.



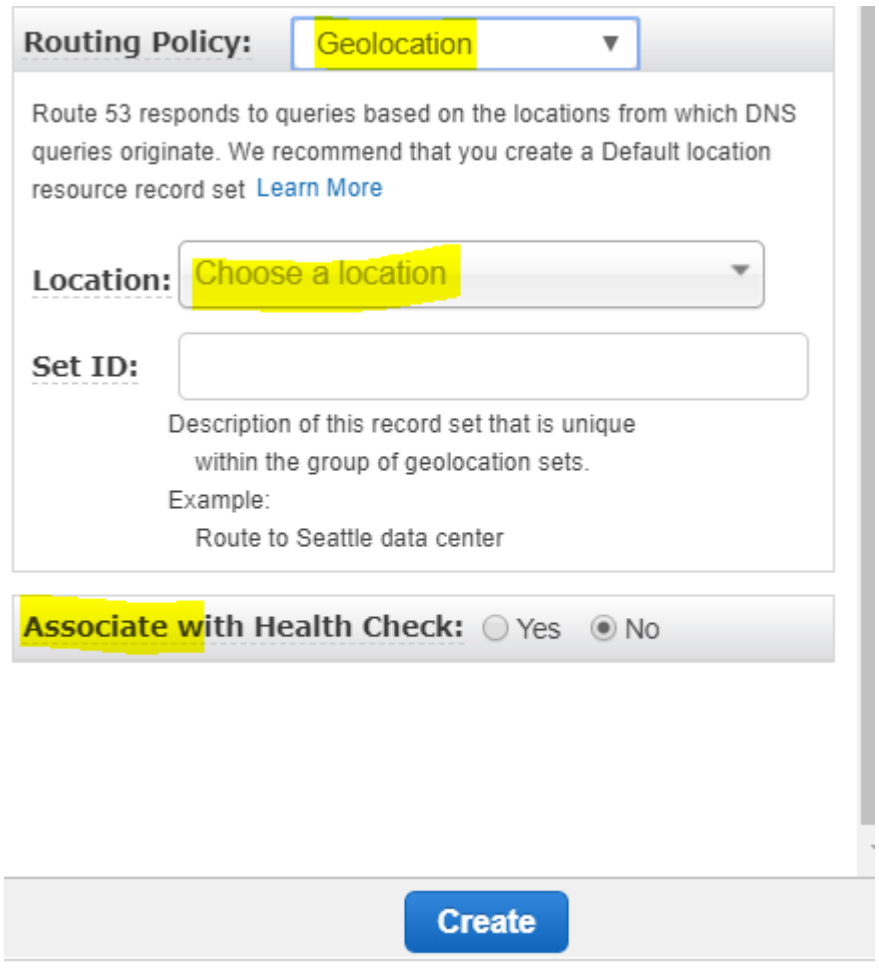


Failover Routing Policy

5- Geo Location Routing Policy

Geolocation routing lets you choose the resources that serve your traffic based on the geographic location of your users, meaning the location that DNS queries originate from. For example, you might want all queries from Europe to be **routed** to an ELB load balancer in the Frankfurt region.

This is routing based on user location. Health check associated.

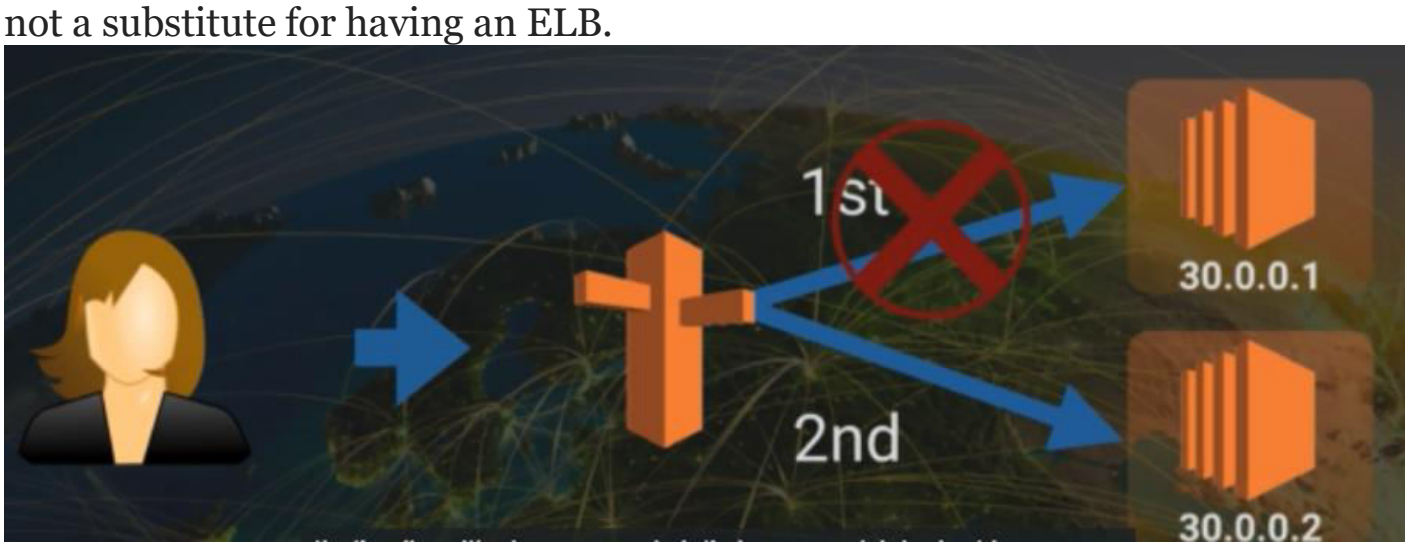


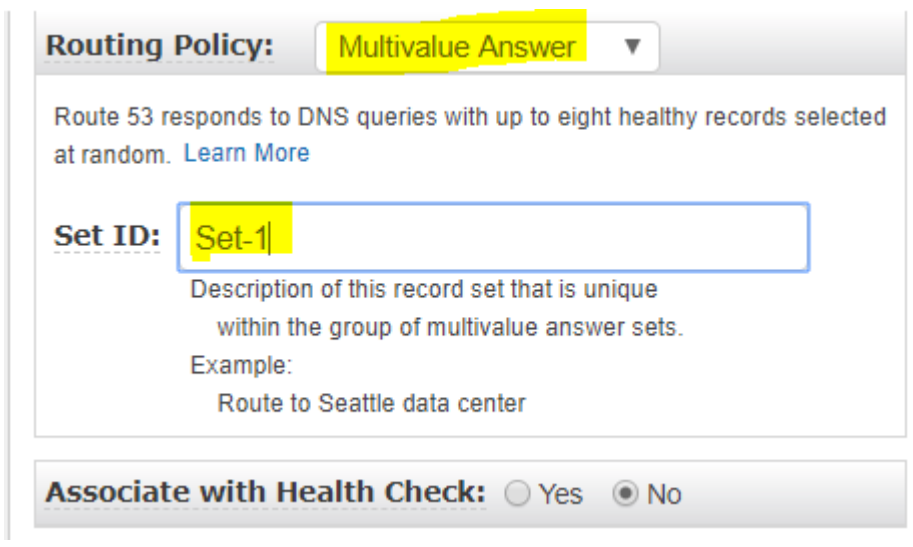
Geolocation Policy

6- Multi Value Routing Policy

It helps distribute DNS responses across **multiple** resources. For example,
use **multivalue** answer **routing** when you want to associate your **routing** records with a **Route** 53 health check.

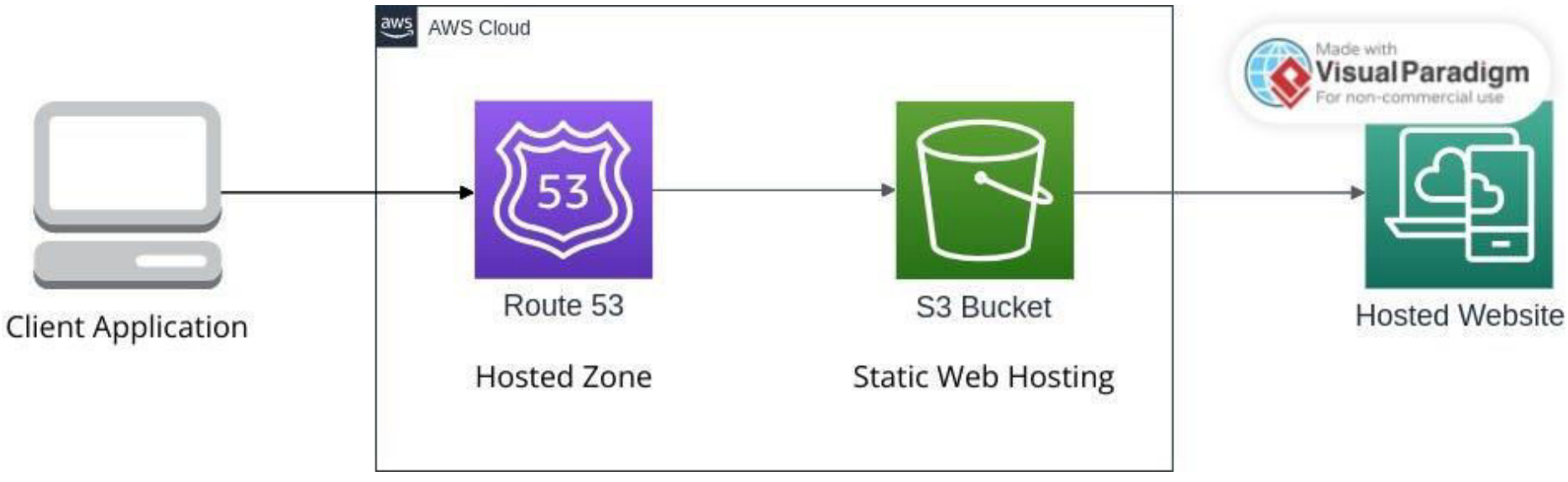
Use multivalue answer routing when you need to return multiple values for a DNS query and route traffic to multiple IP addresses. Up to 8 healthy records are returned for each Multi Value query. Multi Value is not a substitute for having an ELB.





MultiValue Routig

Hosting sample Website and configuring Policies



What is a domain name?

A domain name is a distinctive, spelt-out name that identifies a particular website on the internet. Without having to memorize long IP addresses, it is used to quickly find and access websites.

The actual name and the domain extension make up the two main components of a domain name. The name “example” and the domain extension “.com” are both present in the domain name “example.com,” for instance. The domain extension also referred to as the top-level domain (TLD), identifies the website’s category or type.

What is a static site?

A static site is a website made up of web pages with predetermined content that is always the same for every visitor. In other words, a static site’s content is pre-rendered and presented to users in its current state without any dynamic or in-the-moment processing.

Static sites can include multimedia components like images and videos and are frequently created using HTML, CSS, and JavaScript. The main distinction between a static site and one that changes in response to user input or interaction is that the content on a static site is fixed.

Simple Storage Service (S3)

S3 is a cloud-based object storage provided by AWS which offers industry-leading scalability, data visibility, security and performance. It allows you to store and retrieve large amounts of data over the internet. You can reduce costs, organize data, and set up precise access controls to satisfy unique business, organizational, and compliance requirements using cost-effective storage classes and simple management tools.

Moreover, it can also be used to host static websites. And the best part is S3 manages all the hosting for you so that you can focus on your sites rather than worrying about web servers, IPs and so on.

Here’s what AWS says about S3:

> “Object storage built to retrieve any amount of data from anywhere.”

Route 53

Route 53 is a domain name system (DNS) service offered by Amazon Web Services (AWS) that is extremely scalable. You can control how internet traffic is directed to your services, resources, and applications.

Simply put, Route 53 facilitates the connection between domain names (like example.com) and the associated IP addresses or AWS resources that house your website or application. It serves as an internet directory by converting domain names that can be read by humans into IP addresses that can be read by computers.

Here’s what AWS says about Route 53:

> “A reliable and cost-effective way to route end users to Internet applications.”

Configure Amazon Route 53 to route traffic to the S3 bucket

Now that we got basic definitions and what each service is out of the way, let’s start configuring our basic architecture.

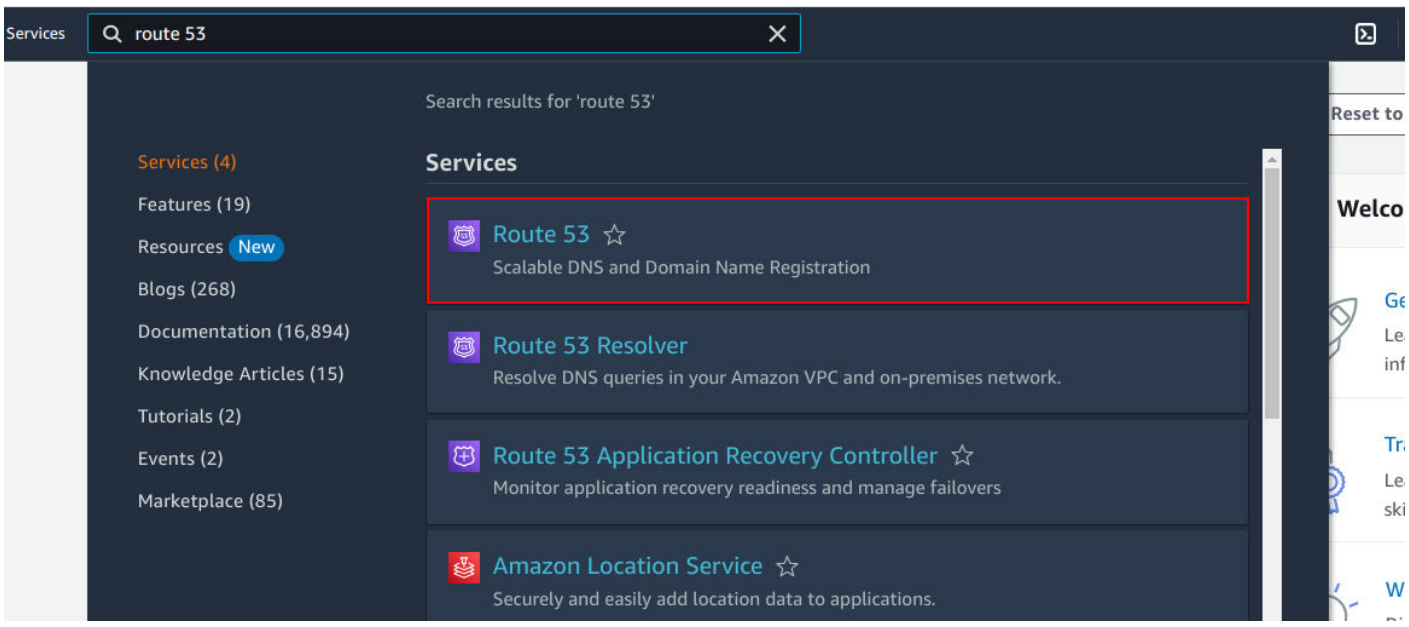
To configure Amazon Route 53 to route traffic to the S3 bucket i.e., host a static site, we’ll perform the following steps.

1. Creating a hosted zone
2. Host a static site in the root domain S3 bucket
3. Configure subdomain bucket for website redirect
4. Configure Route 53 to point to the S3 bucket

Creating a hosted zone

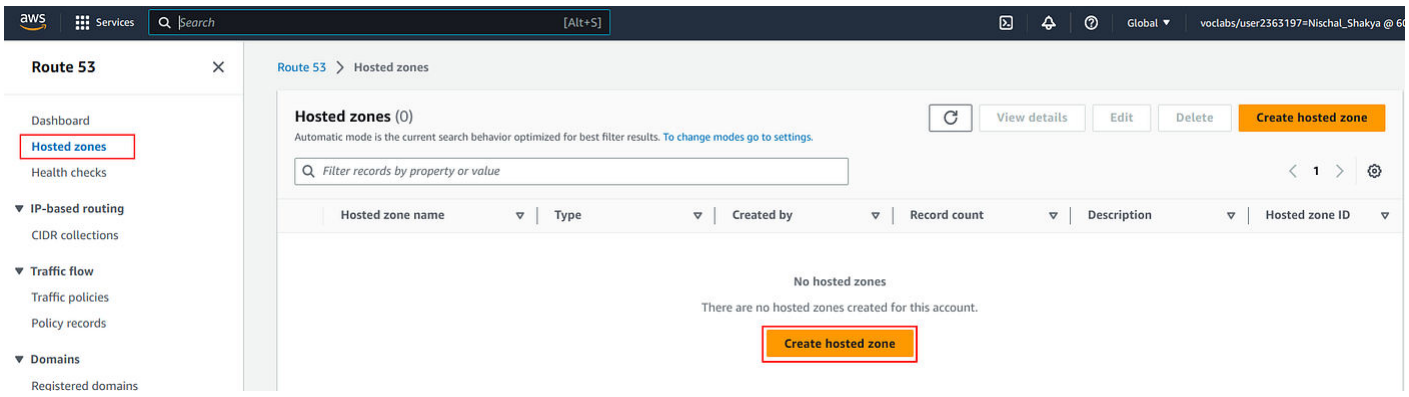
First, you must visit the Route 53 service which can be easily found using the bar.

Simply type **Route 53,** and you should see the service



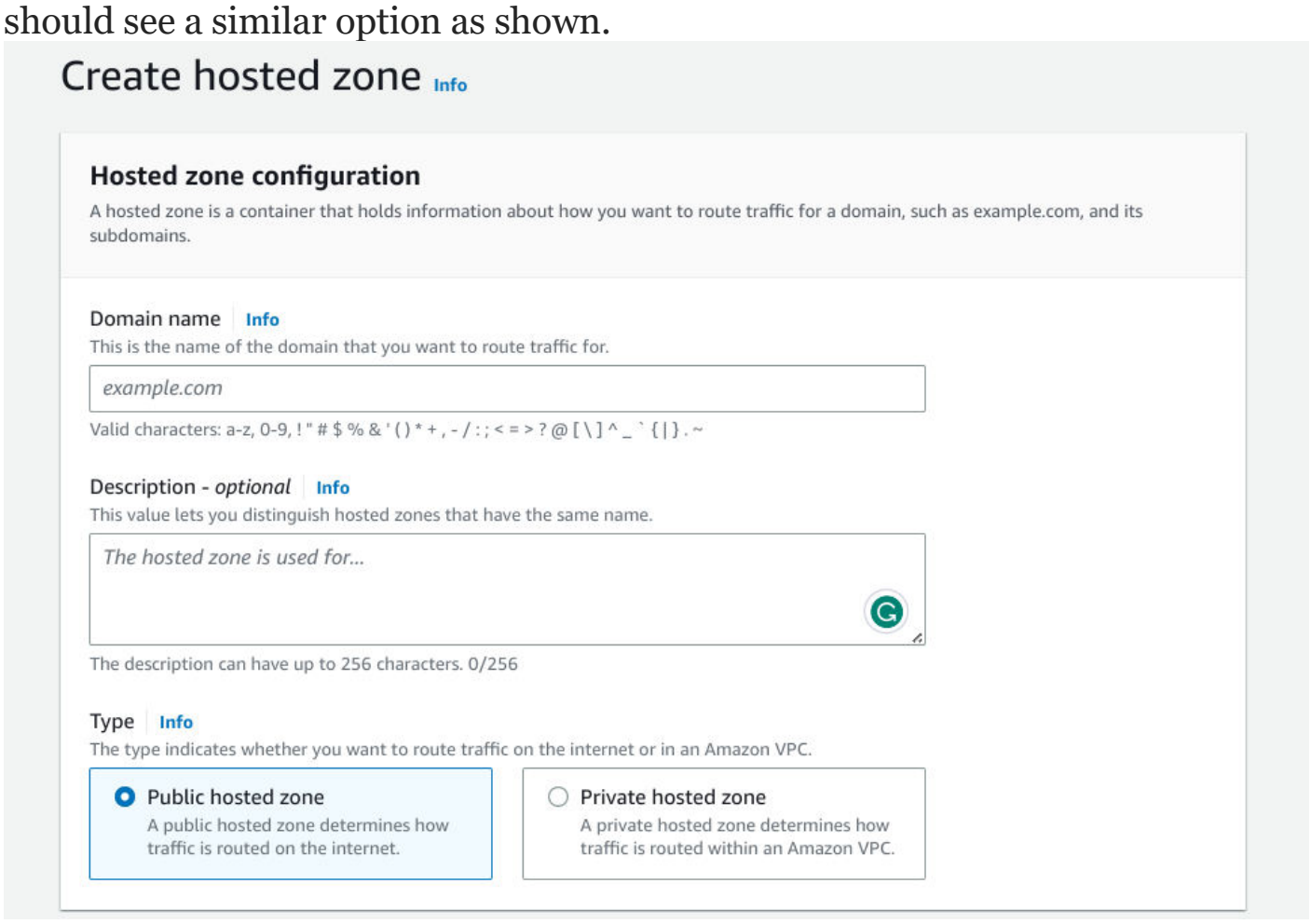
Route 53

Click on the service and then visit the **Hosted Zone** on the sidebar. You should see a UI similar to this.



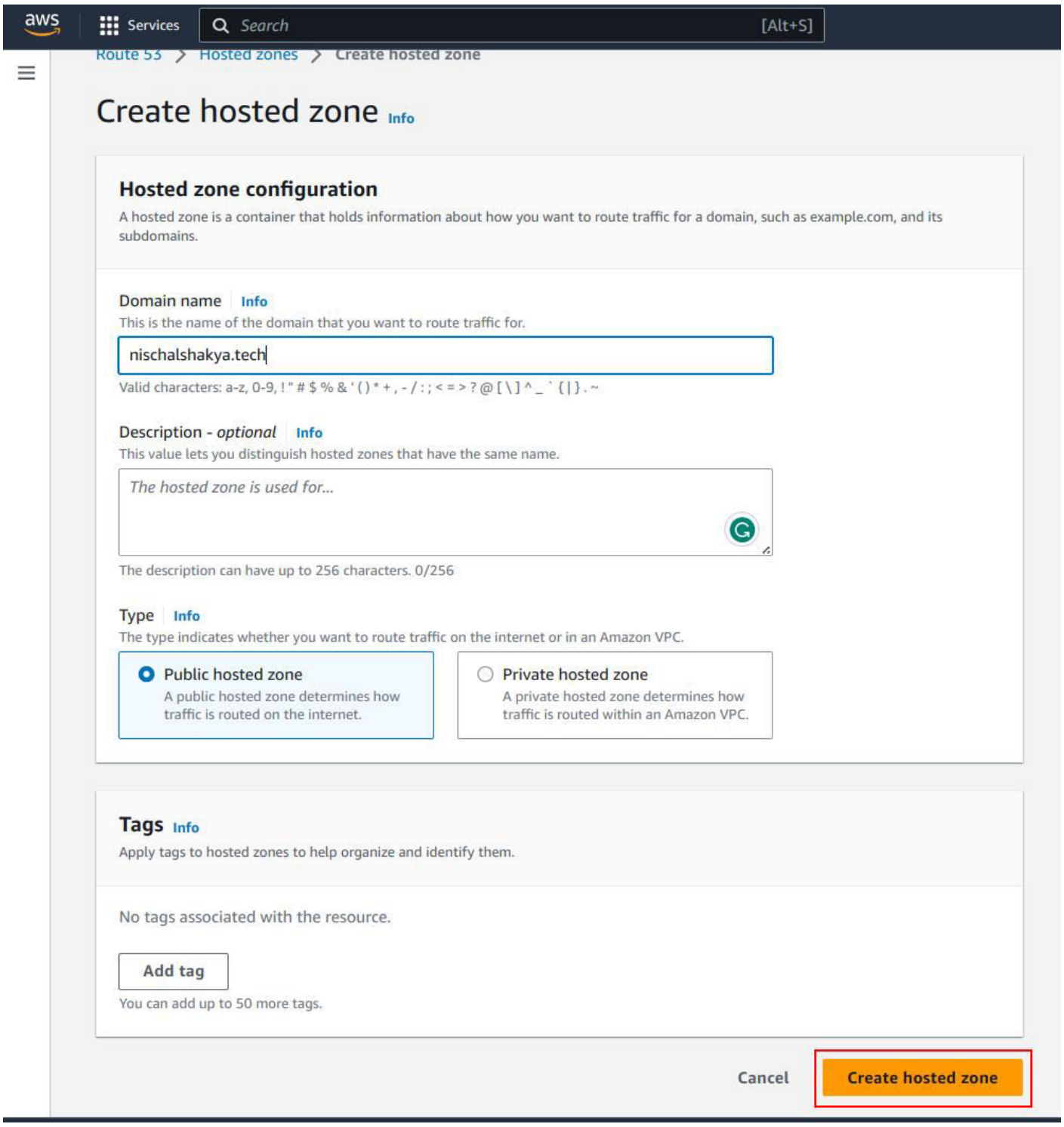
Route 53 Hosted zones

Notice how don't have any hosted zones. To create the hosted zone click on **Created hosted zone**. You should see a similar option as shown.



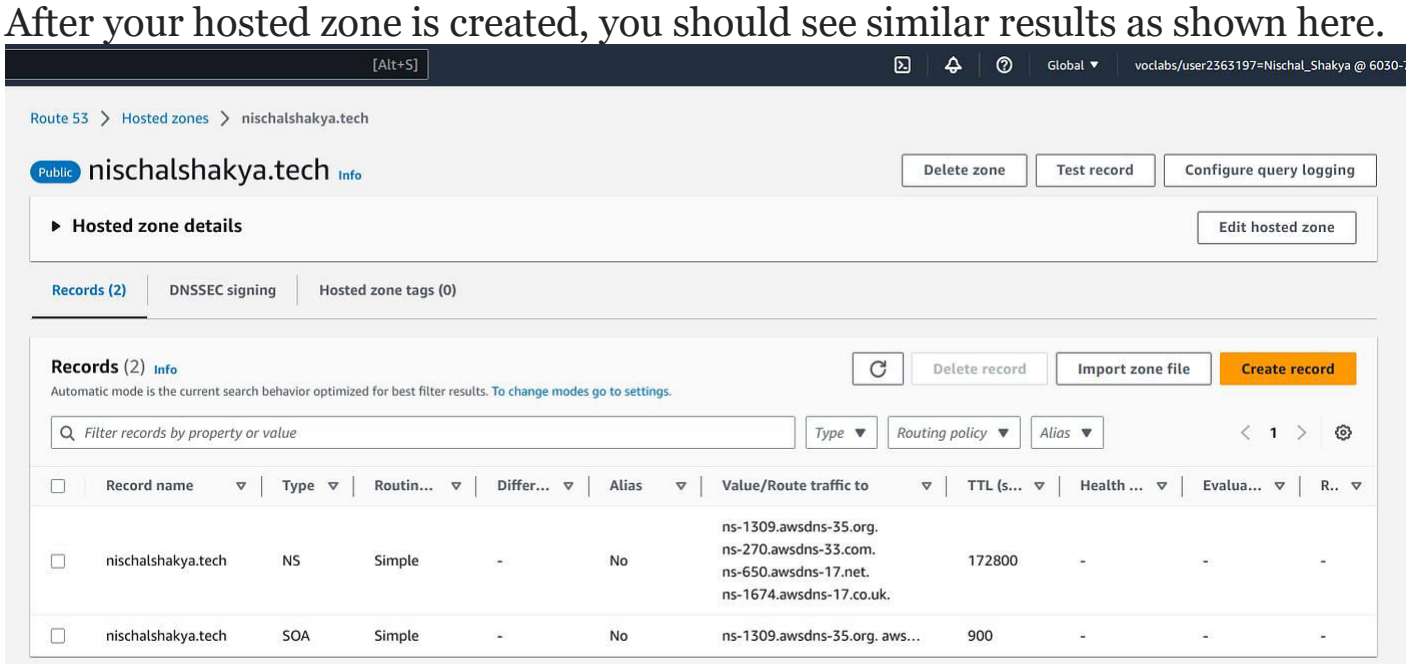
Create hosted zone

Simply type the Domain name. I have a domain named *nischalshakya.tech*. But make sure you type the domain that you own. Click on **Create hosted zone** and your hosted zone will be created.



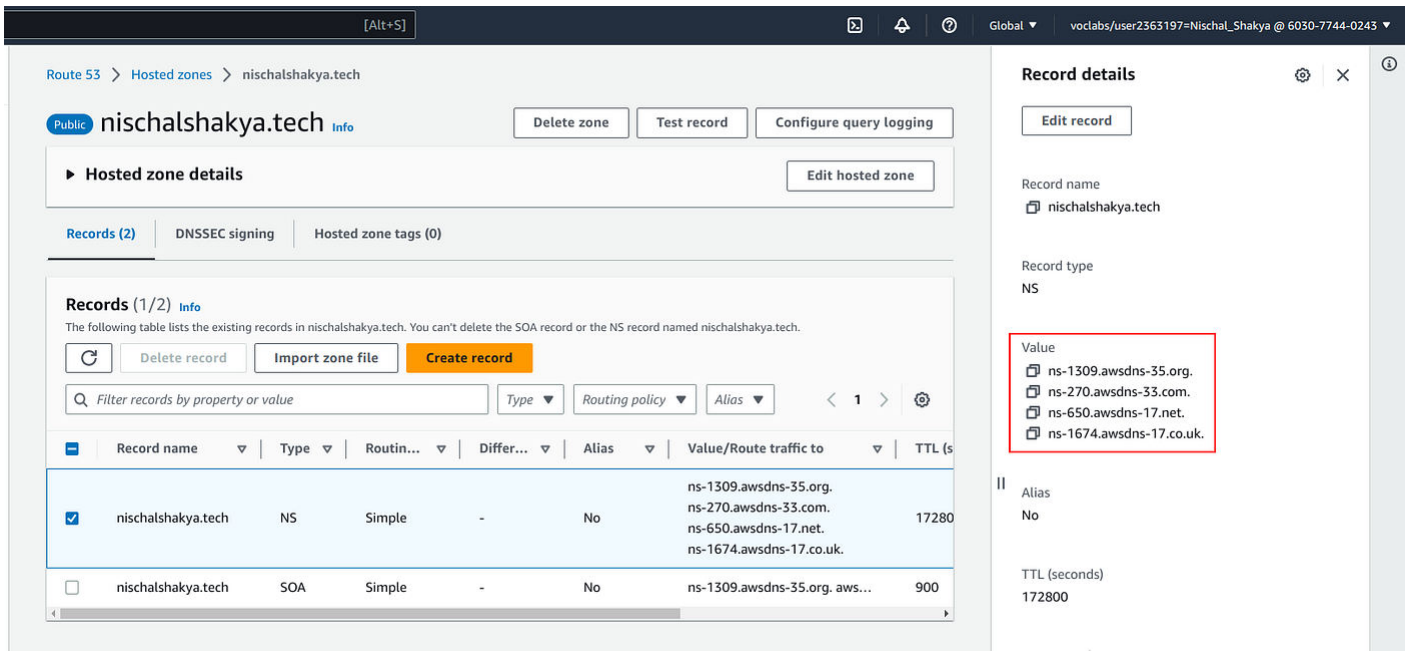
Specifying domain name for the hosted zone

After your hosted zone is created, you should see similar results as shown here.



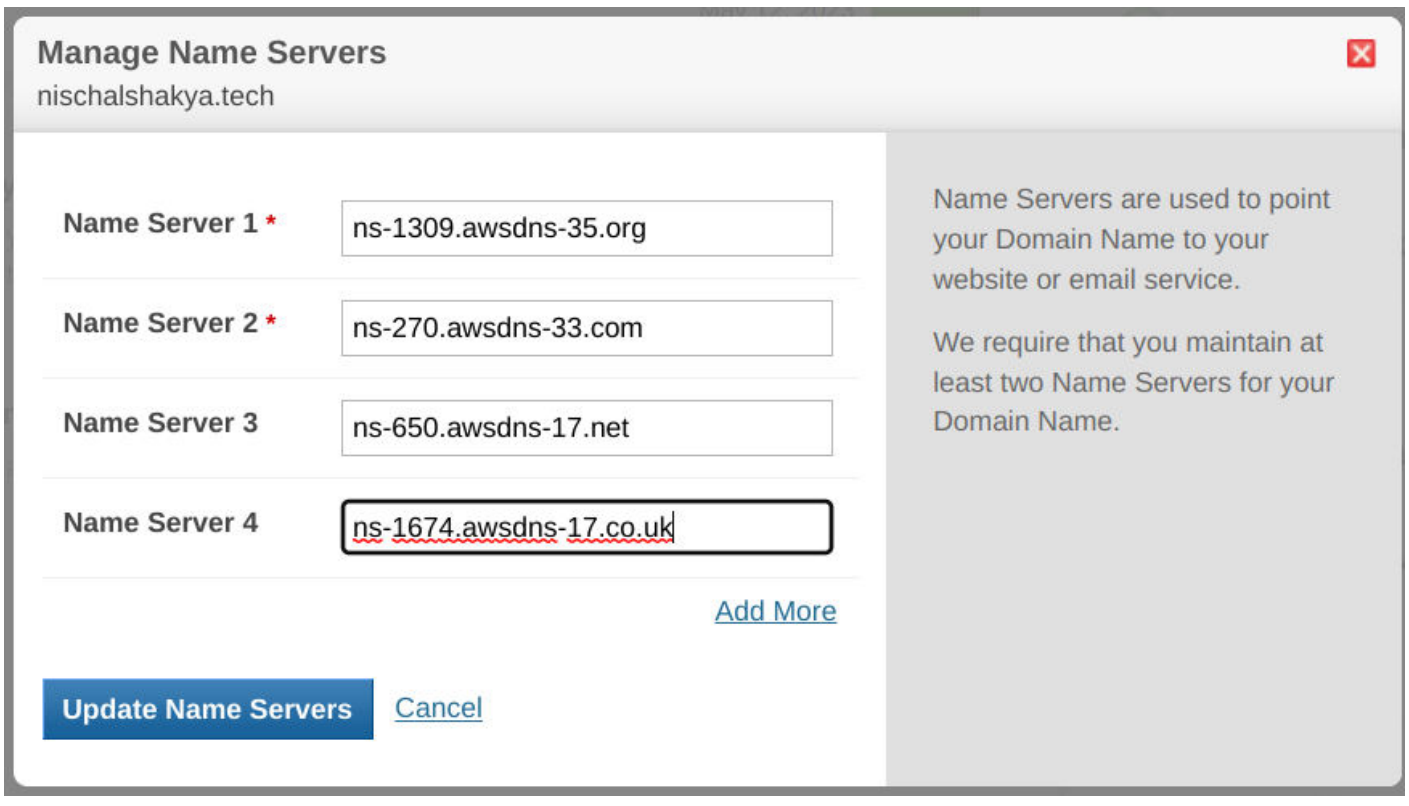
Notice how the hosted zone contains 2 records of Type `NS` and `SOA`. Our concern, in this case, is only with the `NS` type.

Click on the `NS` record and copy all the values provided by it.



Nameserver Values

The values must be updated in the nameserver section of your domain name provider similar to what is shown here. Your domain provider may look different depending upon the provider but the idea is the same.



Configuring nameservers

Once you have updated the nameservers, we are ready to move on to the next section.

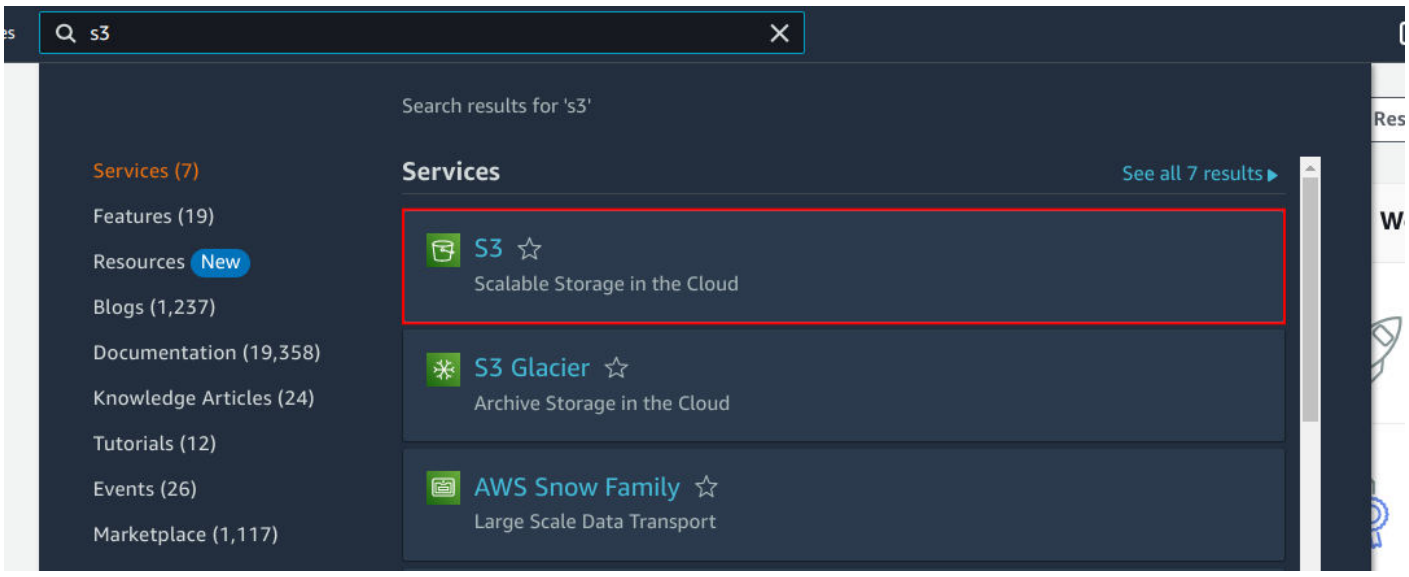
Host a static site in the root domain S3 bucket

You need a static site in the first place to host a site. If you have a static site then it's all good. But if you don't have a site, you can use an example site from here. Simply clone the site and you are good to go.

1. Create a root domain bucket

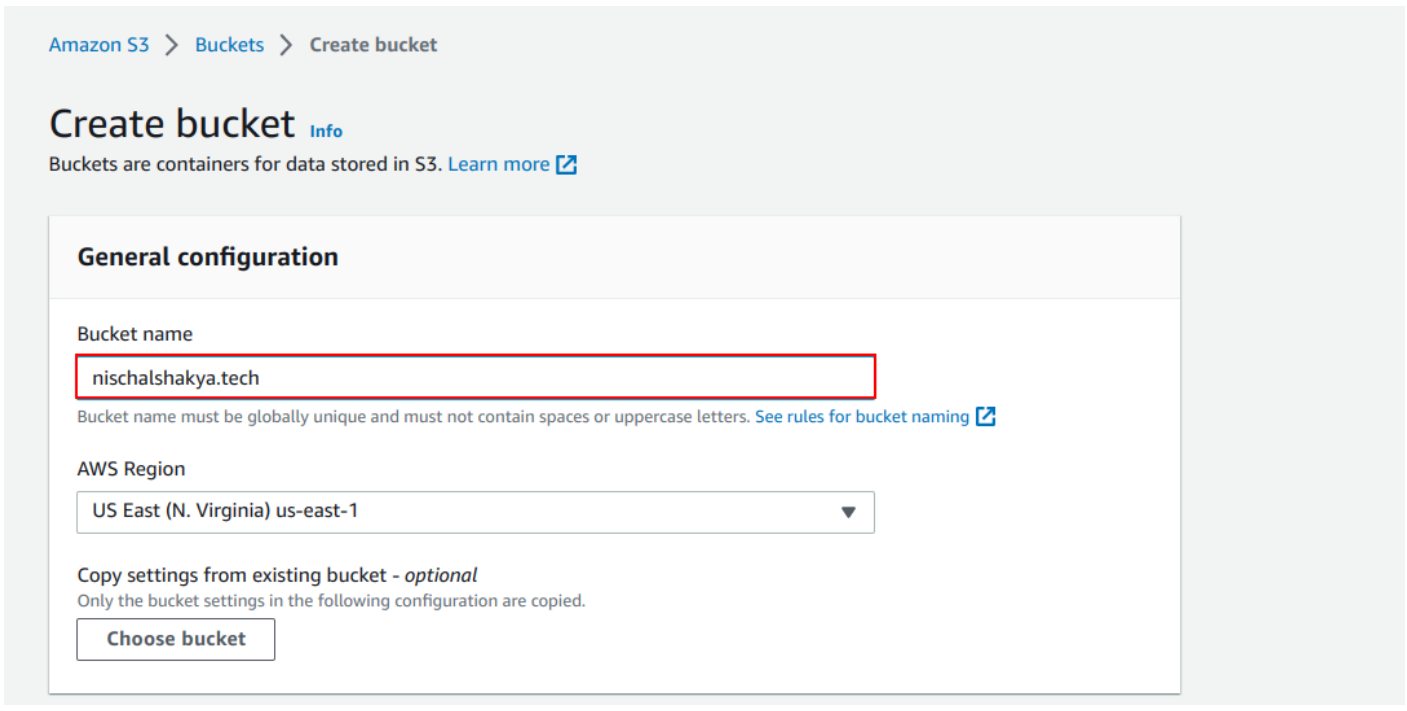
First, create a bucket named ***<domain-name>*.** Since my domain name is **nischalshakya.tech**, I'll be creating my bucket named **nischalshakya.tech**

To create the buckets, simply search for **S3** in the search bar and click on **S3**.



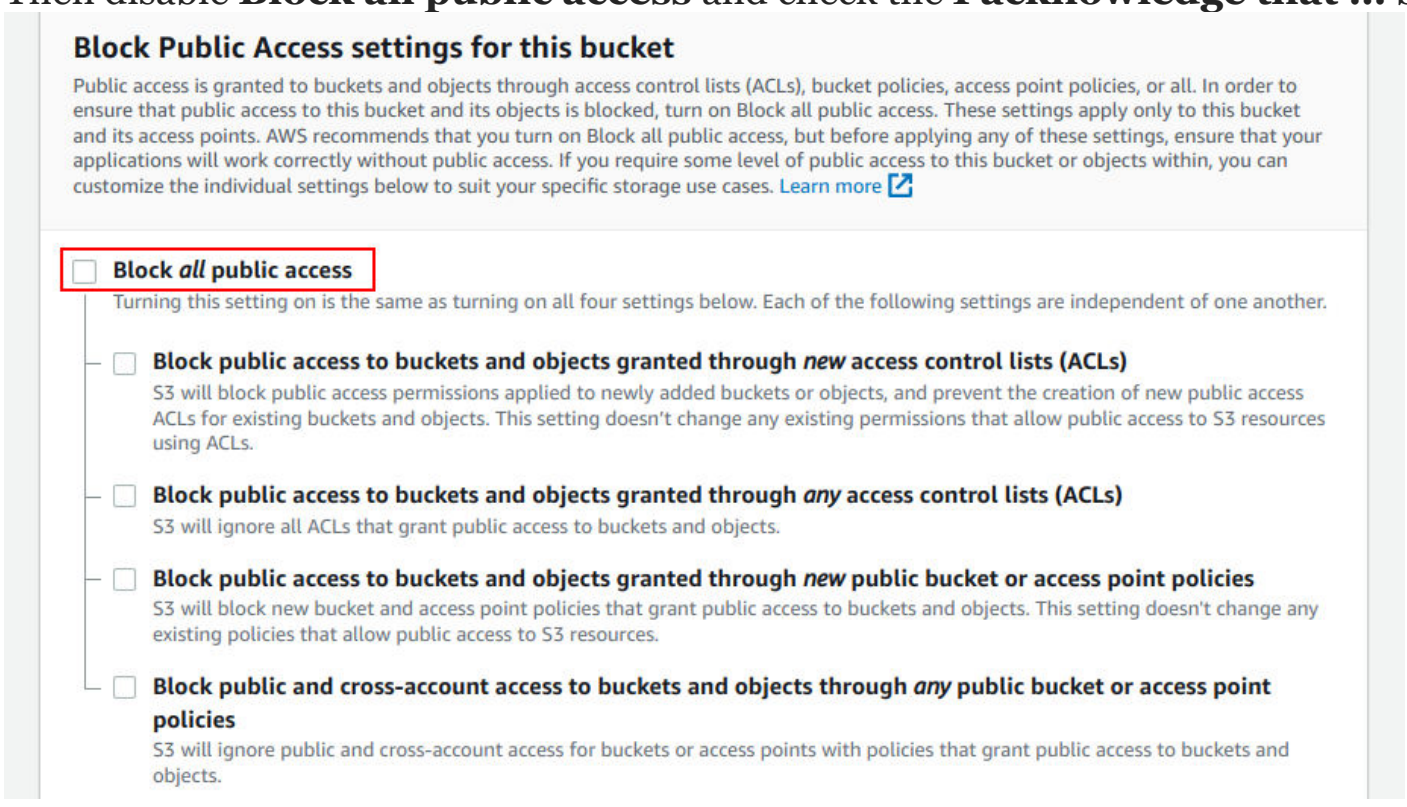
S3 Bucket

Click on **Create bucket** and enter the bucket name.



2. Disable Block all public access

Then disable **Block all public access** and check the **I acknowledge that ...** box.

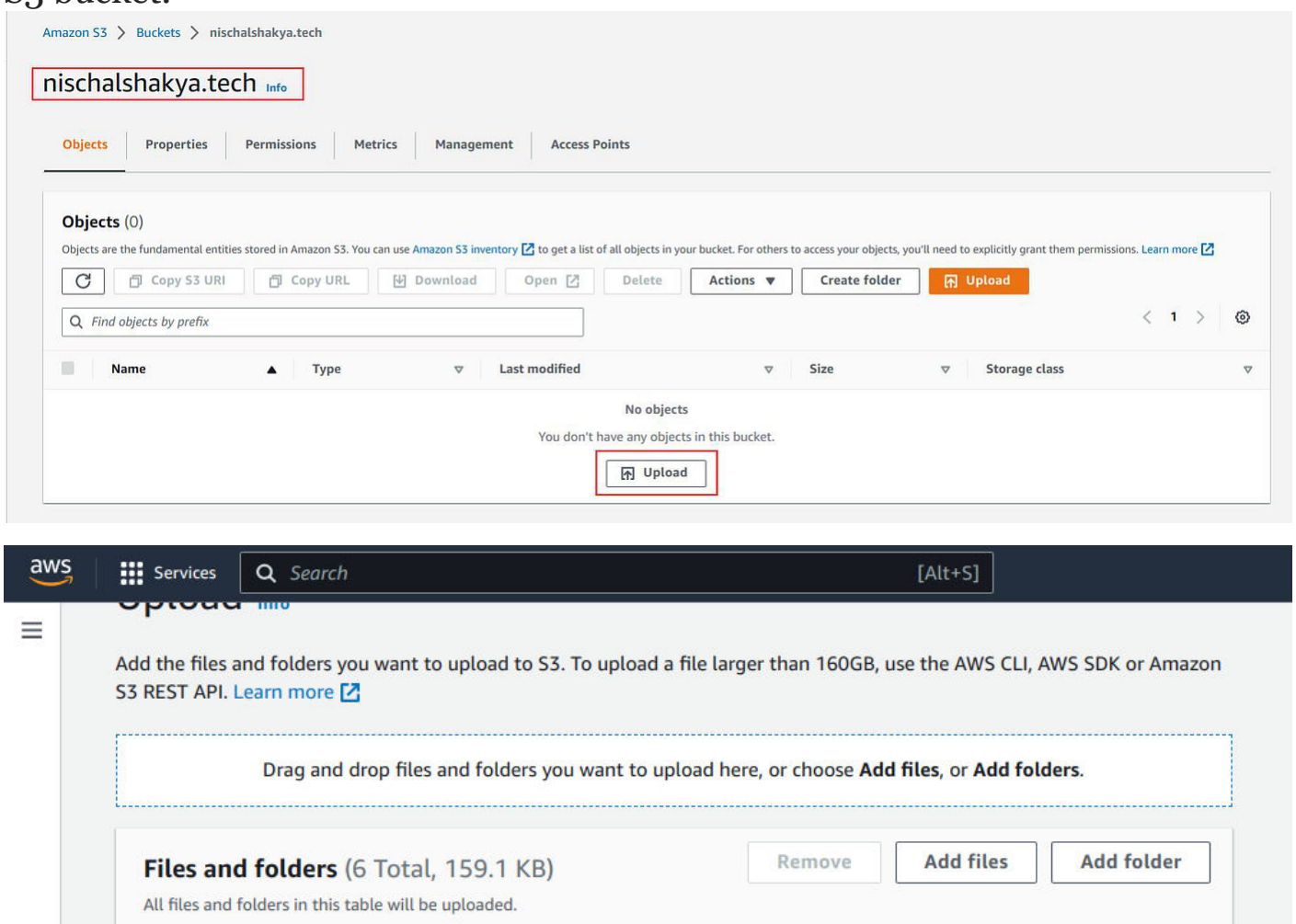


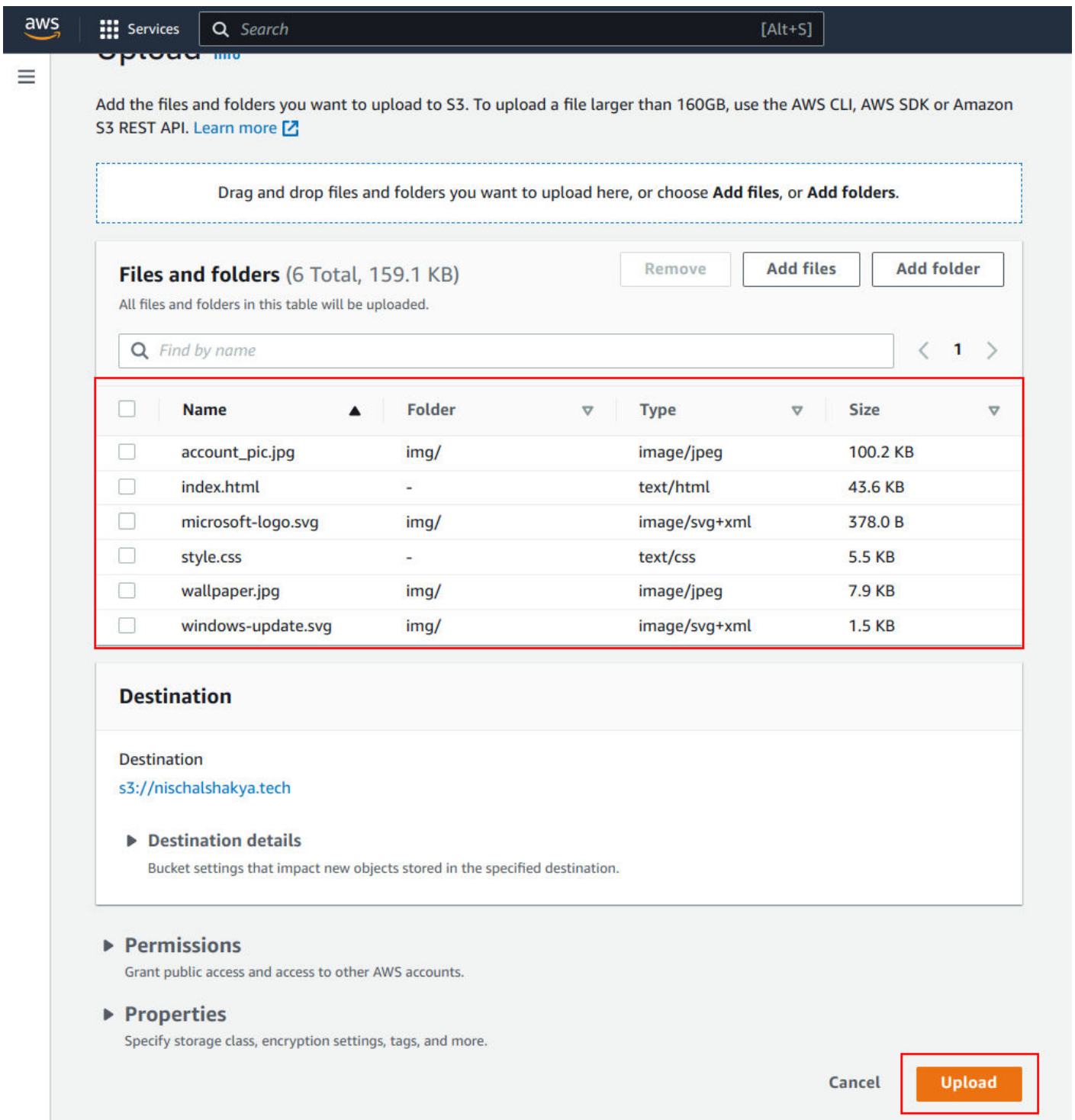
Disable Block all public access

After that simply click on **Create bucket**

The next step is to upload the previously cloned repo in ***<your-domain> (nischalshakya.tech)*** bucket.

Click **Upload** and then simply drag all the folders and files except `.git` folder inside the cloned repo to the S3 bucket.



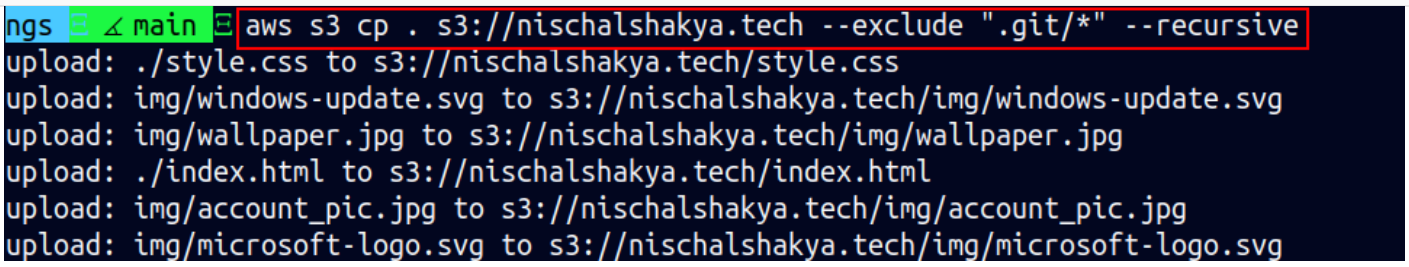


Click on **Upload**

This may take a little while.

If you don't want to perform all the drag and drop, you can use `aws cli` to perform the same action. Simply change your directory where the files are located and write the following code.

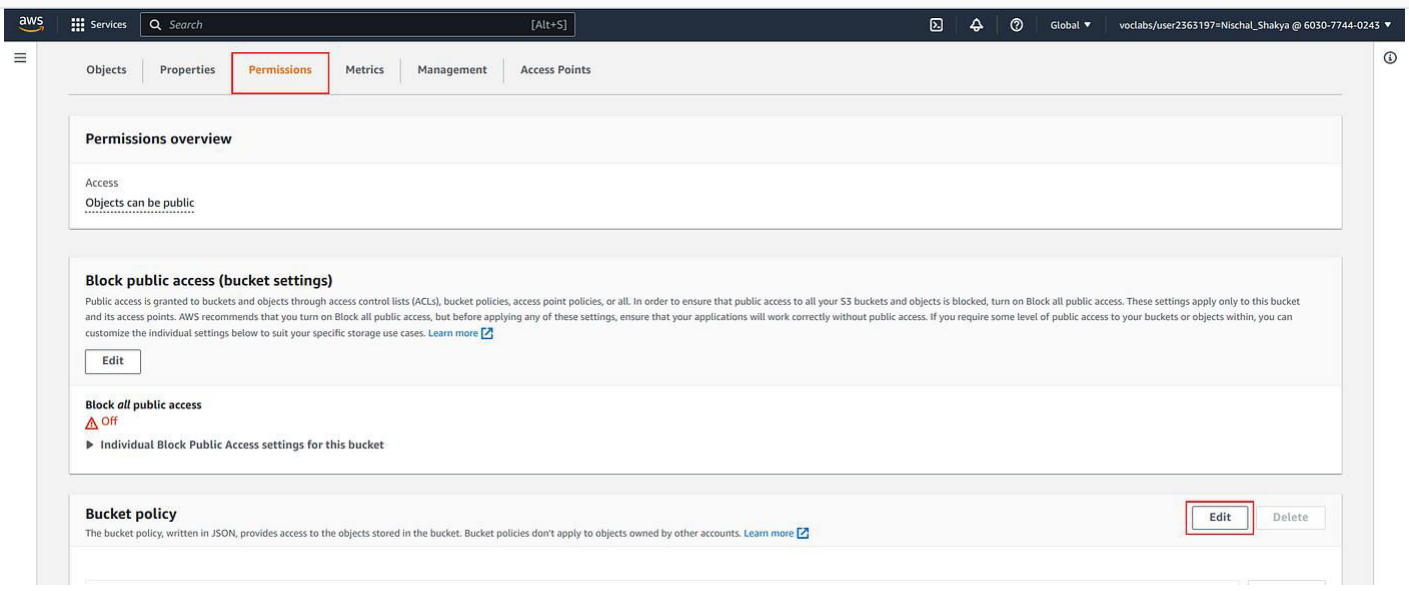
```
aws s3 cp . s3://<domain-name> --exclude ".git/*" --recursive
```



3. Attach bucket policy

You need to make this bucket public by writing a bucket rule. To make the bucket public, write the following code inside the bucket policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1405592139000",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::<domain-name>/*",
            ]
        }
    ]
}
```



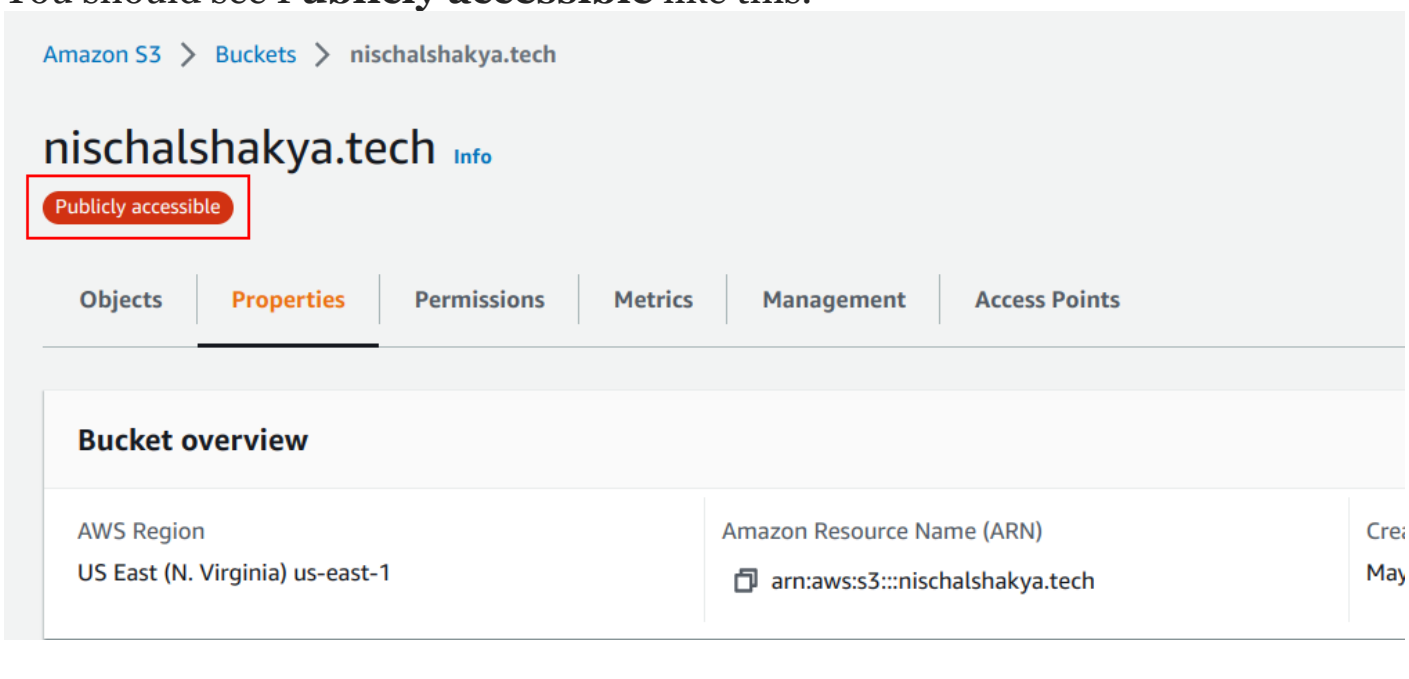
Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1405592139000",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::nischalshakya.tech/*"
            ]
        }
    ]
}
```

Make sure to replace the <domain-name> with your bucket name. In my case it is **nischalshakya.tech**

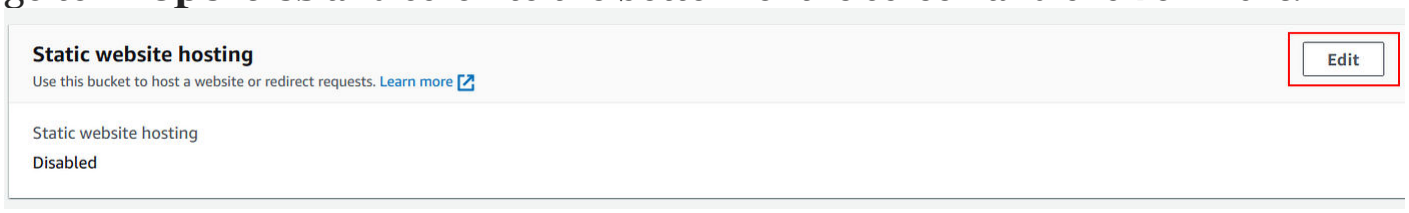
Then click on **Save changes**

You should see P**ublicly accessible** like this.

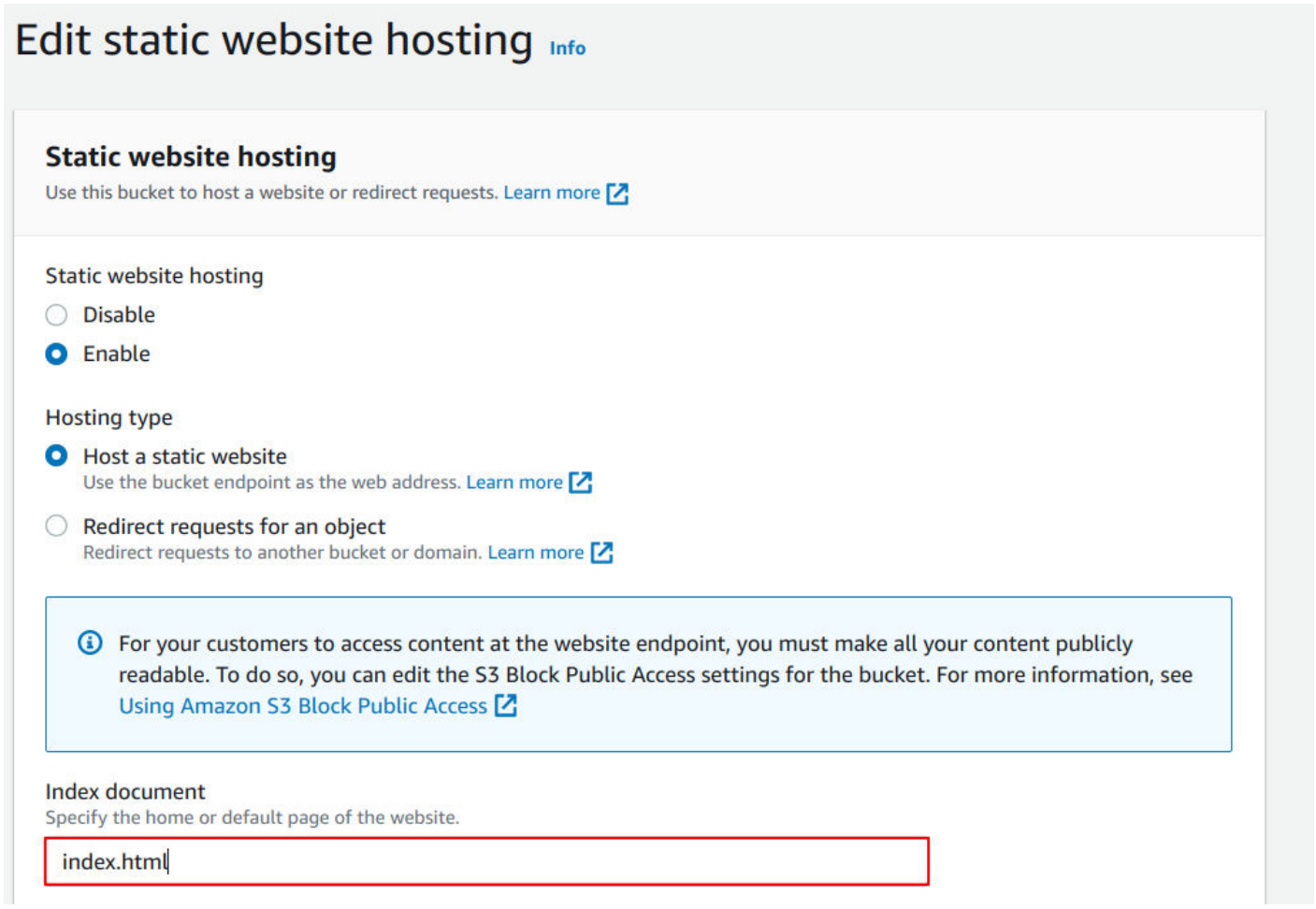


4. Enable Static website hosting

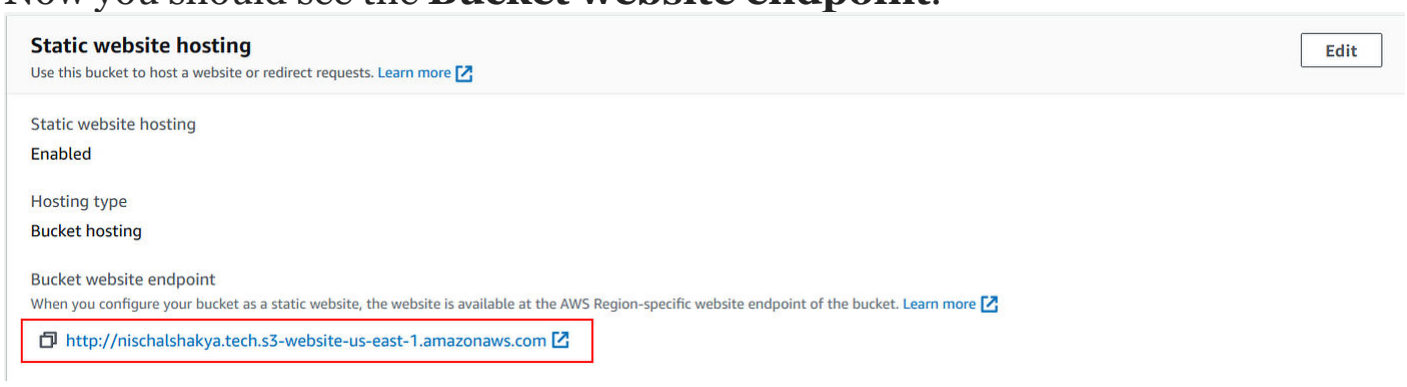
There is one more step that we need to perform in this bucket i.e., to enable static hosting. To perform this go to **Properties** and scroll to the bottom of the screen and click on **Edit**.



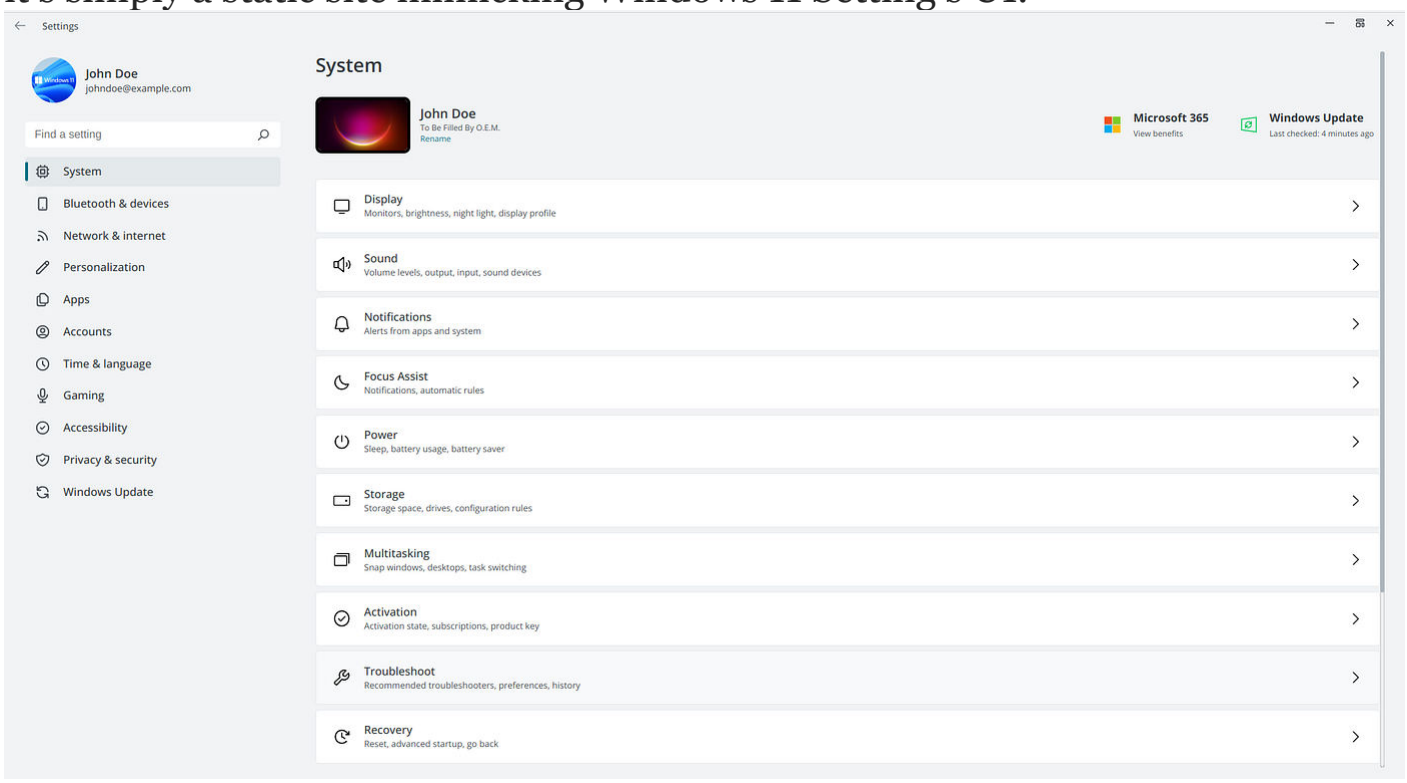
Enable **Static website hosting** and specify **index.html** inside **Index document**. Then simply click on **Save Changes**.



Now you should see the **Bucket website endpoint**.



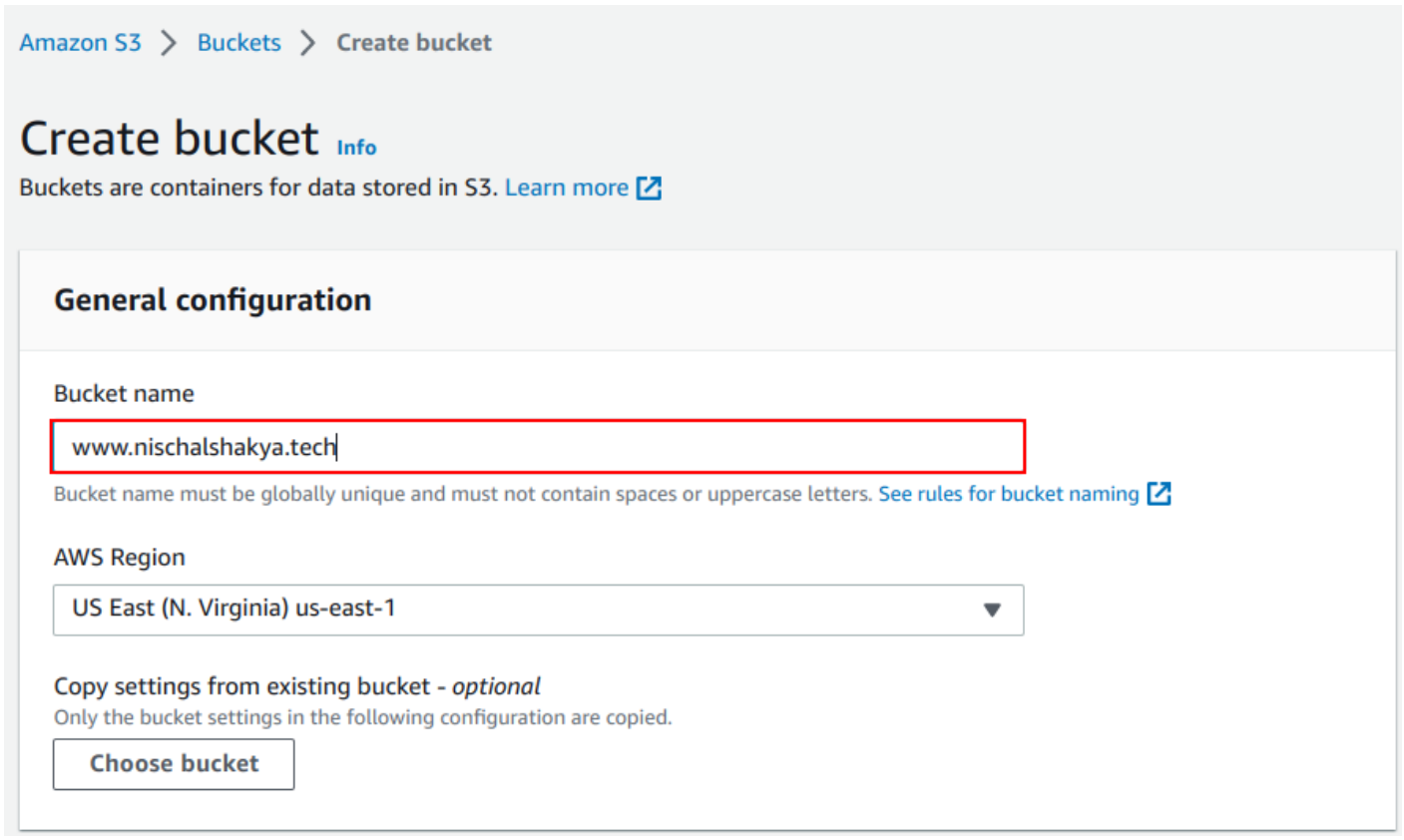
If you click this endpoint, you should see the beautiful site. Don't get confused. It's not Windows Settings, it's simply a static site mimicking Windows 11 Setting's UI.



Configure subdomain bucket for website redirect

1. Create a subdomain bucket

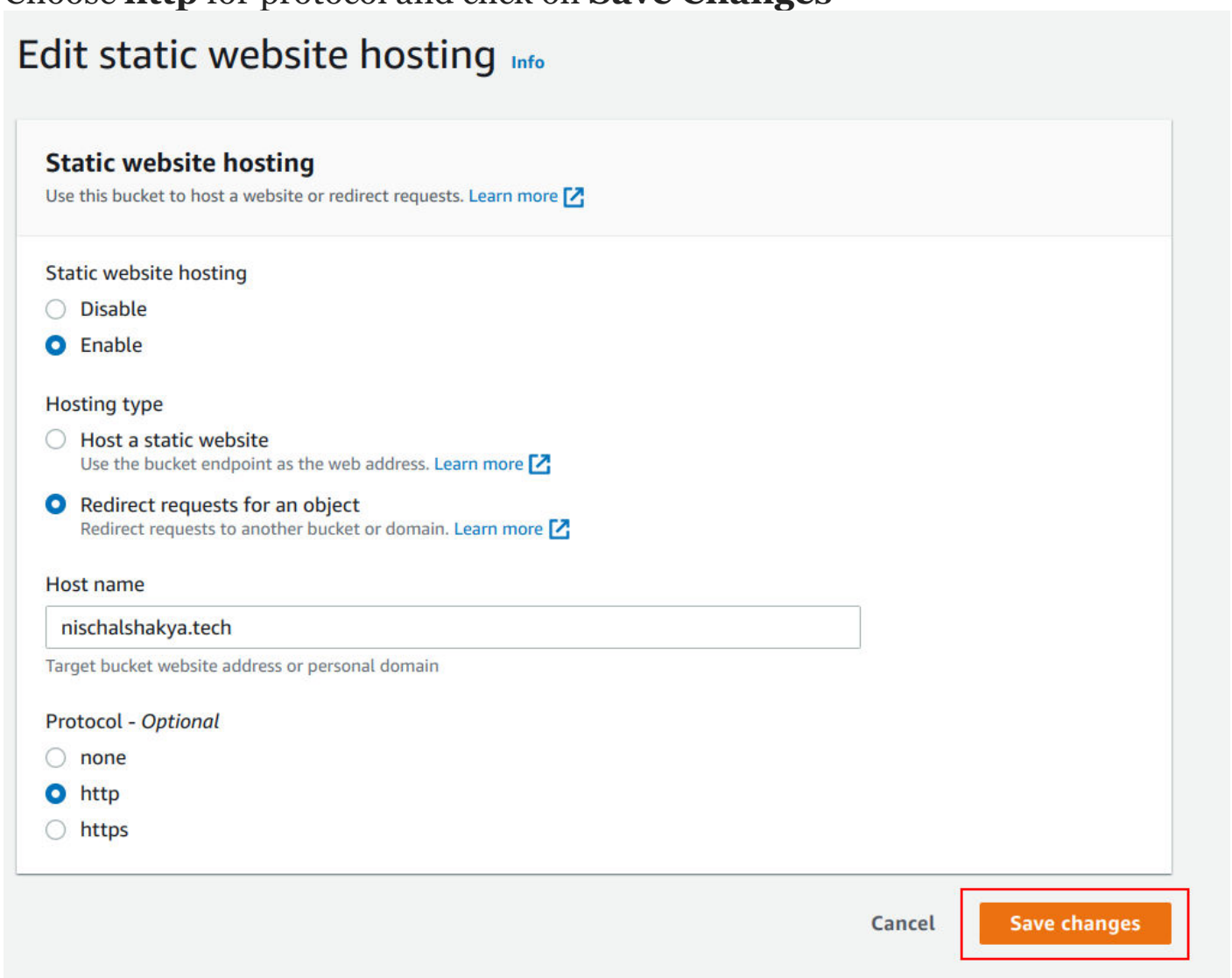
Create another bucket **www.<*domain-name*>**. In my case, I will create a bucket named **www.nischalshakya.tech**. But this time, simply create the bucket and leave all the settings as default.



2. Enable static hosting

Simply go to **Static website hosting** as before and **Enable** it. But this time, we'll choose **Redirect requests for an object**. For **Host name**, enter your **domain name (nischalshakya.tech)**

Choose **http** for protocol and click on **Save Changes**



If you visit this endpoint, you should be directed to your **domain name (nischalshakya.tech)**.

Yeah, I know it shows a **This site can't be reached** page, but we'll fix this in a moment.

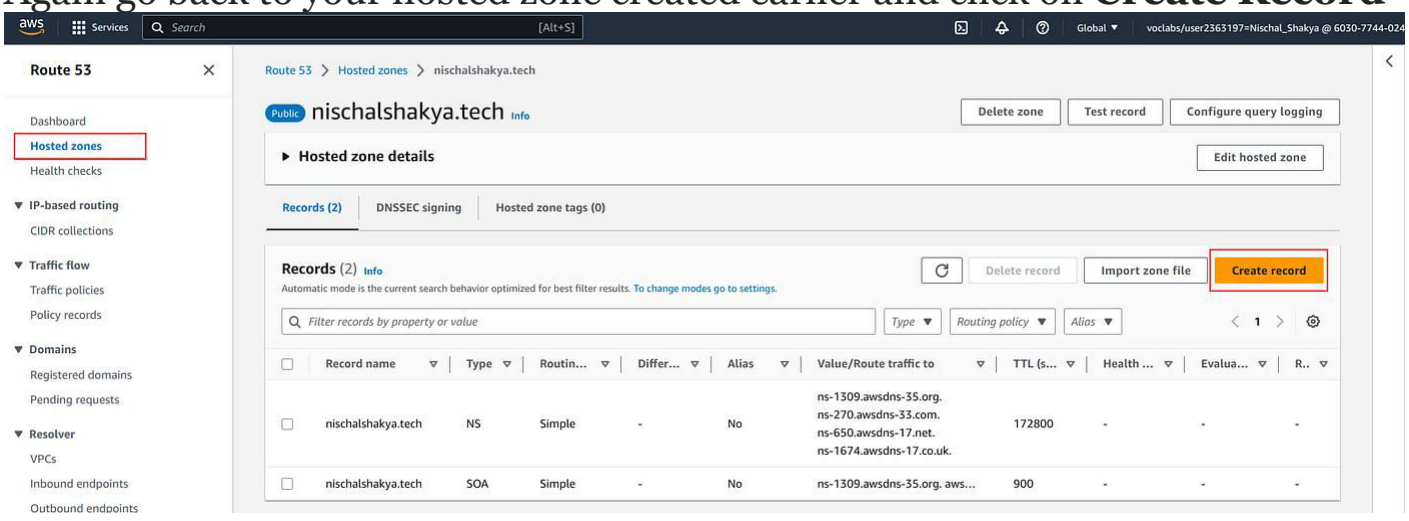
Configure Route 53 to point to the S3 endpoint

Now comes the most exciting part, i.e., configuring Route 53 to point to the S3 endpoint.

This might sound scary, but believe me, it's the easiest part.

1. Create a record

Again go back to your hosted zone created earlier and click on **Create Record**



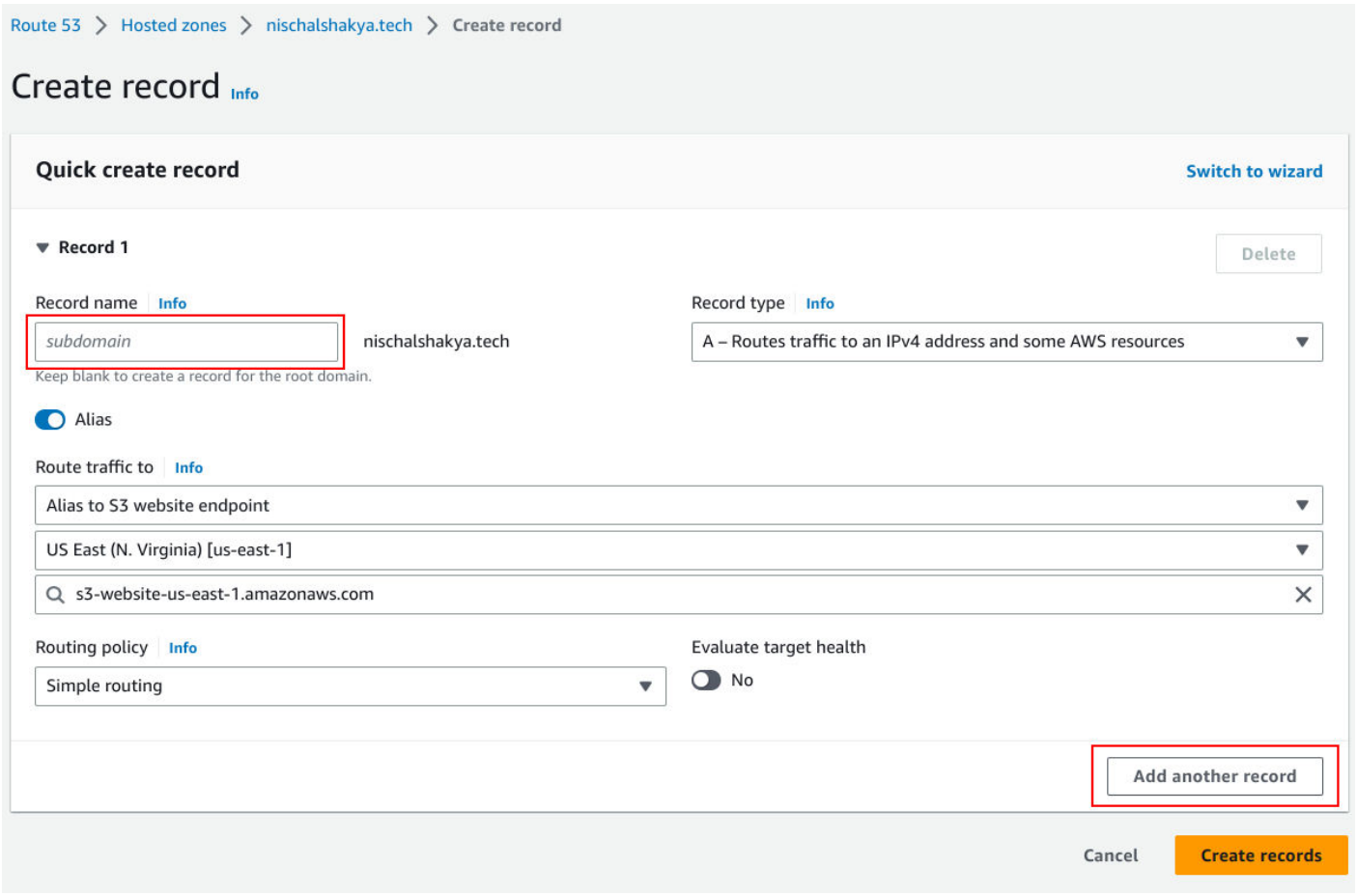
For **Record 1**, leave the **Record name** empty and enable **Alias**

Under **Route traffic to** choose **Alias to S3 website endpoint**

Under **Choose Region**, choose the region where you created the S3 bucket. In my case, it *us-east-1*

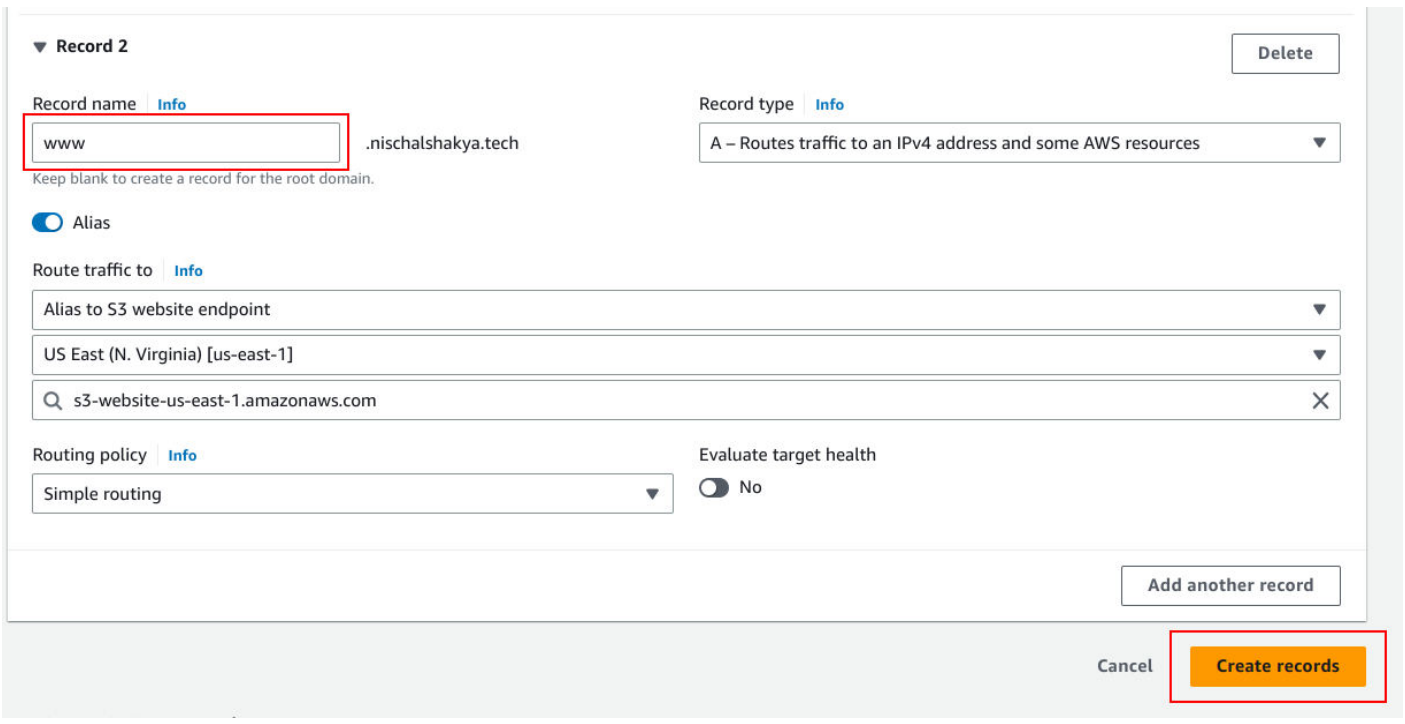
Now, under **Enter S3 endpoint**, you should see your endpoint with a **domain name (nischalshakya.tech)**. For some reason, if you don't see the endpoint listed, you can visit this site and copy the **Website Endpoint** based on your S3 Bucket deployed Region name.

For **Evaluate target health**, choose **No**.



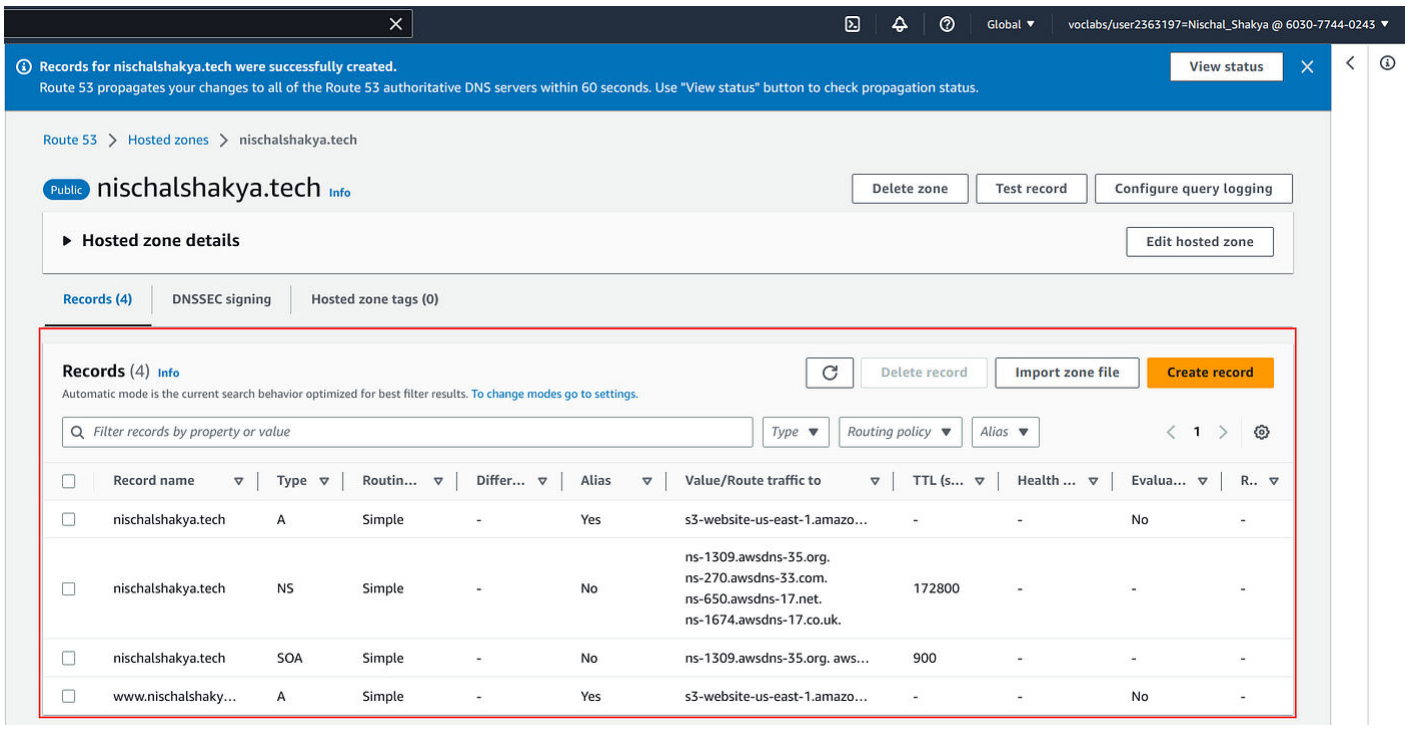
Now click on **Add another record** and perform the same action as before.

But enter the **Record name** as **www**



Then click on **Create records**

Now, you should see your record being added similar to this.



So now what?

Visit your **domain name (nischalshakya.tech)** and your site is running as before.

If you don't see anything immediately, wait for a minute or two as change propagation may take a little time.

It's that simple to host your site with a proper domain with the help of **S3** and **Route 53**.

12.Relational Database Services (RDS)

Data Base Instances

A *DB instance* is an isolated database environment running in the cloud. It is the basic building block of Amazon RDS. A DB instance can contain multiple user-created databases, and can be accessed using the same client tools and applications you might use to access a standalone database instance. DB instances are simple to create and modify with the AWS command line tools, Amazon RDS API operations, or the AWS Management Console.

Note

Amazon RDS supports access to databases using any standard SQL client application. Amazon RDS does not allow direct host access.

You can have up to 40 Amazon RDS DB instances, with the following limitations:

- 10 for each SQL Server edition (Enterprise, Standard, Web, and Express) under the "license-included" model
- 10 for Oracle under the "license-included" model
- 40 for MySQL, MariaDB, or PostgreSQL
- 40 for Oracle under the "bring-your-own-license" (BYOL) licensing model

Note

If your application requires more DB instances, you can request additional DB instances by using this form.

Each DB instance has a DB instance identifier. This customer-supplied name uniquely identifies the DB instance when interacting with the Amazon RDS API and AWS CLI commands. The DB instance identifier must be unique for that customer in an AWS Region.

The DB instance identifier forms part of the DNS hostname allocated to your instance by RDS. For example, if you specify `db1` as the DB instance identifier, then RDS will automatically allocate a DNS endpoint for your instance. An example endpoint is *`db1`*`.abcdefghijkl.us-east-1.rds.amazonaws.com`, where *`db1`* is your instance ID.

In the example endpoint *`db1`*`.abcdefghijkl.us-east-1.rds.amazonaws.com`, the string *`abcdefghijkl`* is a unique identifier for a specific combination of AWS Region and AWS account. The identifier *`abcdefghijkl`* in the example is internally generated by RDS and doesn't change for the specified combination of Region and account. Thus, all your DB instances in this Region share the same fixed identifier. Consider the following features of the fixed identifier:

- If you rename your DB instance, the endpoint is different but the fixed identifier is the same. For example, if you rename *`db1`* to *`renamed-db1`*, the new instance endpoint is *`renamed-db1`*`.abcdefghijkl.us-east-1.rds.amazonaws.com`.
- If you delete and re-create a DB instance with the same DB instance identifier, the endpoint is the same.
- If you use the same account to create a DB instance in a different Region, the internally generated identifier is different because the Region is different, as in *`db2`*`.mnopqrstuvwx.us-west-1.rds.amazonaws.com`.

Each DB instance supports a database engine. Amazon RDS currently supports MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and Amazon Aurora database engines.

When creating a DB instance, some database engines require that a database name be specified. A DB instance can host multiple databases, or a single Oracle database with multiple schemas. The database name value depends on the database engine:

- For the MySQL and MariaDB database engines, the database name is the name of a database hosted in your DB instance. Databases hosted by the same DB instance must have a unique name within that instance.
- For the Oracle database engine, database name is used to set the value of ORACLE_SID, which must be supplied when connecting to the Oracle RDS instance.
- For the Microsoft SQL Server database engine, database name is not a supported parameter.
- For the PostgreSQL database engine, the database name is the name of a database hosted in your DB instance. A database name is not required when creating a DB instance. Databases hosted by the same DB instance must have a unique name within that instance.

Amazon RDS creates a master user account for your DB instance as part of the creation process. This master user has permissions to create databases and to perform create, delete, select, update, and insert operations on tables the master user creates. You must set the master user password when you create a DB instance, but you can change it at any time using the AWS CLI, Amazon RDS API operations, or the AWS Management Console. You can also change the master user password and manage users using standard SQL commands.

Data Base Engine

A database engine is an underlying software that a database management system, in this case, Amazon RDS, uses to create, read, update, and delete data from your database. This is roughly analogous to the operating system on your computer, and your database engine represents a fundamental building block of your database.
Different database engines approach the method of organizing and accessing data in a slightly different way, or use a different querying language to do so. The most popular database engines today include MySQL, MariaDB, PostgreSQL, Aurora, Oracle, and Microsoft SQL.
Selecting the right database engine is a key step toward optimizing your Amazon RDS implementation.

Why Use Amazon RDS Over Other Solutions?

Amazon RDS is attractive to startups because of the flexibility inherent to cloud computing solutions. Rather than having to dedicate development resources to maintain complicated in-house infrastructure, startups can offload that responsibility to AWS.
Additionally, Amazon RDS enables startups to rapidly scale capacity up and down as needed. This allows you to meet sudden surges in demand, without needing to invest in infrastructure that might sit idle if demand drops off. This is particularly useful for businesses with “seasonal” patterns, like news outlets, streaming services, or e-commerce, but it can also prevent downtime if your app goes “viral”, providing your business with a vital ability to scale up.
Finally, startups can take advantage of flexible pricing based on the required performance. This enables startups to aggressively balance their budgets and minimize burn rate, freeing up resources to be spent on growth rather than maintenance.

What Database Engines does Amazon RDS Support?

Amazon RDS supports 7 different kinds of database engines out of the box. Users can select these engines whenever they create a new instance. Which engine you pick will be defined by your existing business requirements, infrastructure, and long-term growth plans.
In addition to the standard Amazon RDS database engines, startups can also take advantage of Amazon Aurora, which is specifically optimized for cloud database implementations.

Amazon RDS Database Engines

Amazon RDS is a managed SQL database service that supports the most popular varieties of database engines. Amazon RDS is a good choice for startups that already have some kind of database infrastructure, and wish to migrate to the cloud with minimal headaches. It supports:

MySQL

MySQL is one of the most popular database engines in the world. It is free and open-source, however, the main supporting company was acquired by Oracle in 2010… It’s typically fast, very robust and compatible with many different operating systems. Amazon RDS supports MySQL Community Edition versions 5.7 and 8.0.
Generally, MySQL solutions are used for databases that only require simple read-write functions, with strong performance.

MariaDB

MariaDB is also a free and open-source database engine. It began as a fork of the MySQL database management system, made by the original MySQL creators fearing possible development control from Oracle. It is often positioned as an improved version of MySQL, with better performance compared to its predecessor.
In general, MariaDB is a good option for startups who don’t need to access customer support and are confident that they can build their own database solution in-house, as there is no enterprise support option available.

PostgreSQL

PostgreSQL is an object-relational database system that is ACID-compliant. It is more advanced than MySQL based solutions, but this brings increased complexity that can impact performance.

Despite this, PostgreSQL is highly popular, stable, and good for startups who have complicated database requirements, or who need a low maintenance solution once the initial set-up is completed.

Oracle SQL

Oracle is a cross-platform management system built specifically for large commercial use-cases. Oracle is a “what you see is what you get” solution, and cannot be customized by the startup using it.
While it is a possible choice for companies that need to deal with large quantities of data, it is not useful for smaller companies.
If your organization has a specific reason to pick Oracle, then an advantage of using Oracle SQL is that you can choose from two different licensing models on RDS. You can either Bring Your Own License, or simply use Amazon’s “License Included” service model, which means you don’t need to separately procure an Oracle SQL license.

Microsoft SQL

SQL server is Microsoft’s proprietary database engine. As with Oracle, it’s generally best suited for larger companies with large static datasets who already use Microsoft SQL as a core part of their business. It is generally not a good choice for startups.
One advantage of using Microsoft SQL with RDS, is the “License Included” model, which means you don’t need to separately procure Microsoft SQL Server licenses when you’re using Amazon RDS.

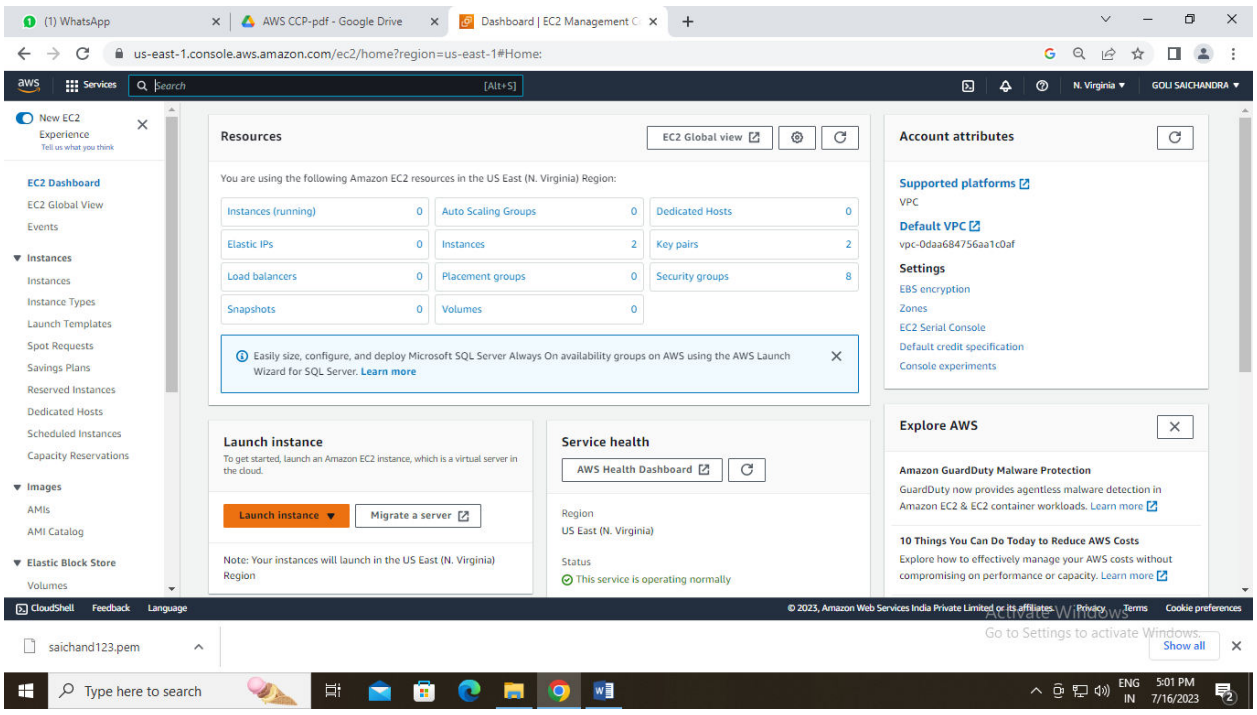
Amazon Aurora

Unlike Amazon RDS, Amazon Aurora is designed specifically with the cloud in mind. It is a fully managed MySQL- and PostgreSQL-compatible relational database that combines the performance and availability of traditional enterprise databases, with the simplicity, customizability, and cost-effectiveness of open source databases.

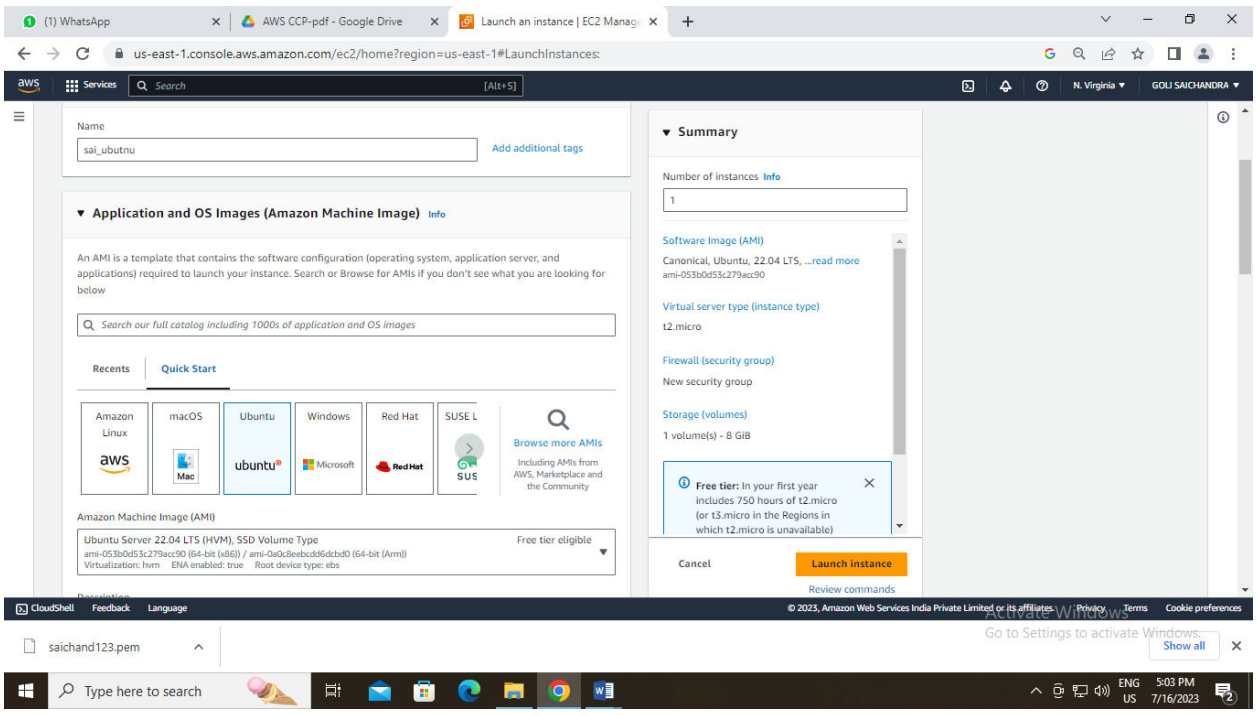
Launching a RDS Instances (MySQL, MSSQL & Aurora)

RDS MYSQL:

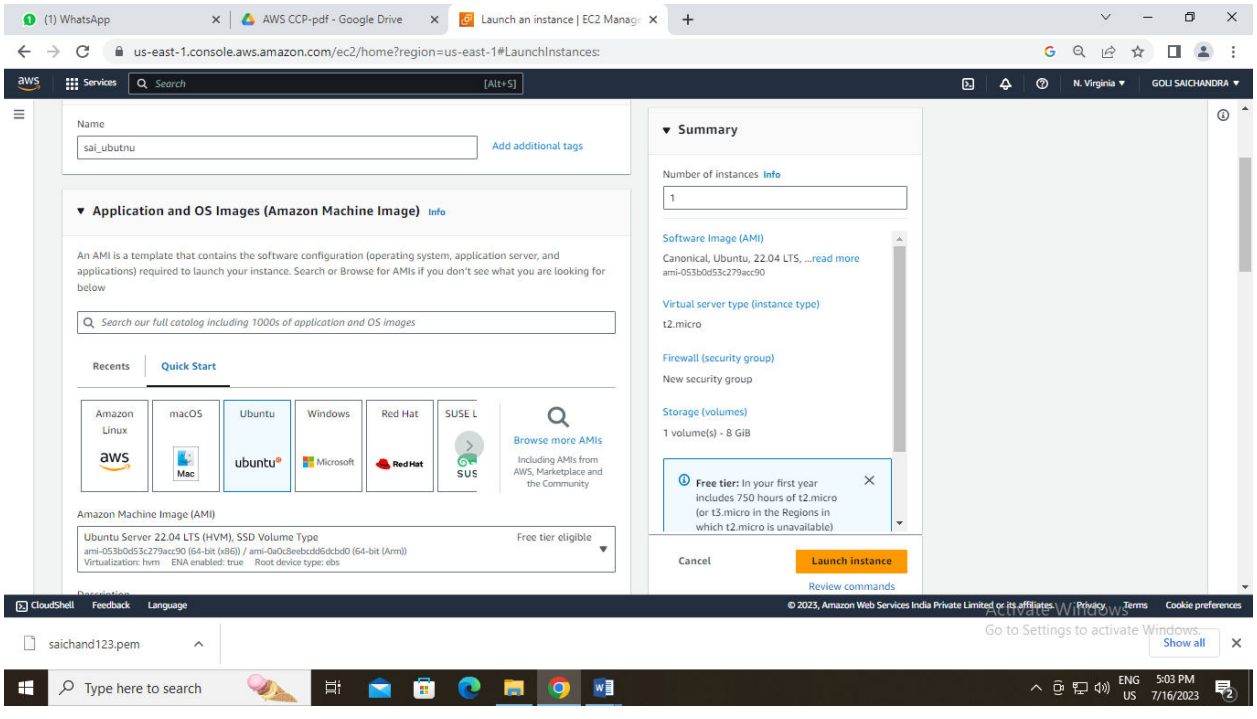
Click on EC2 and launch an instance of Ubuntu



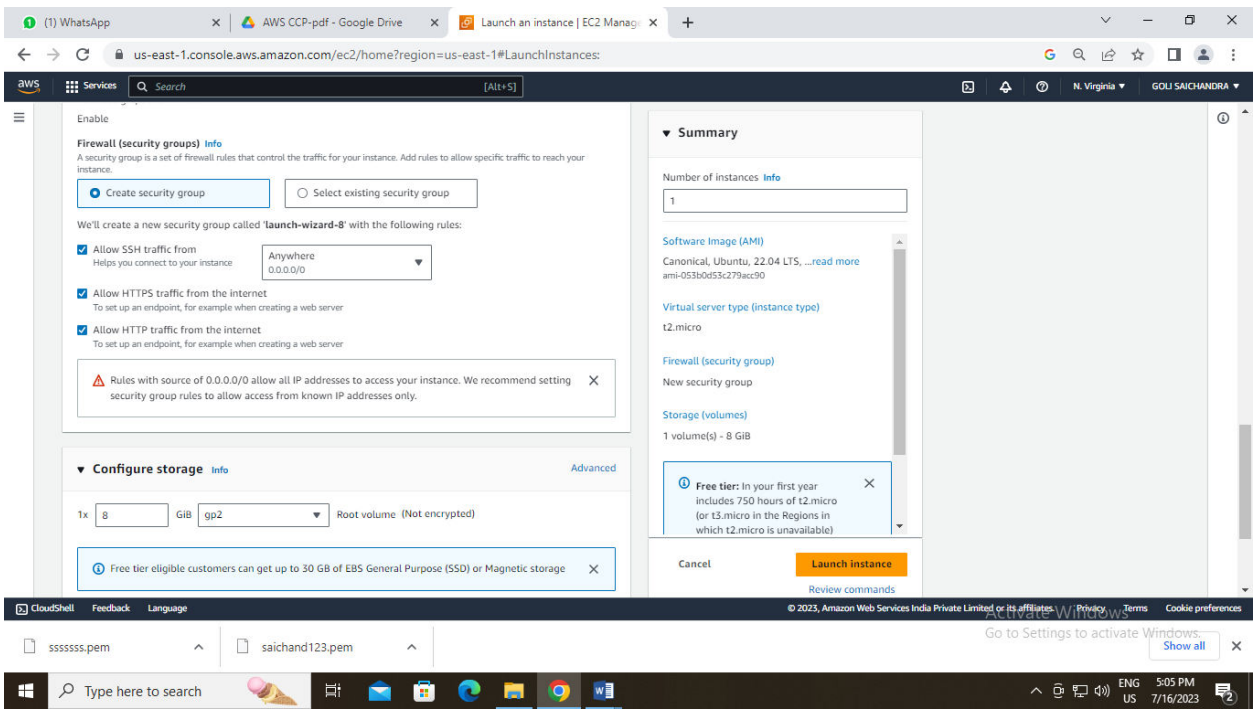
Now select a name and AMI of the instance as shown in the imahge



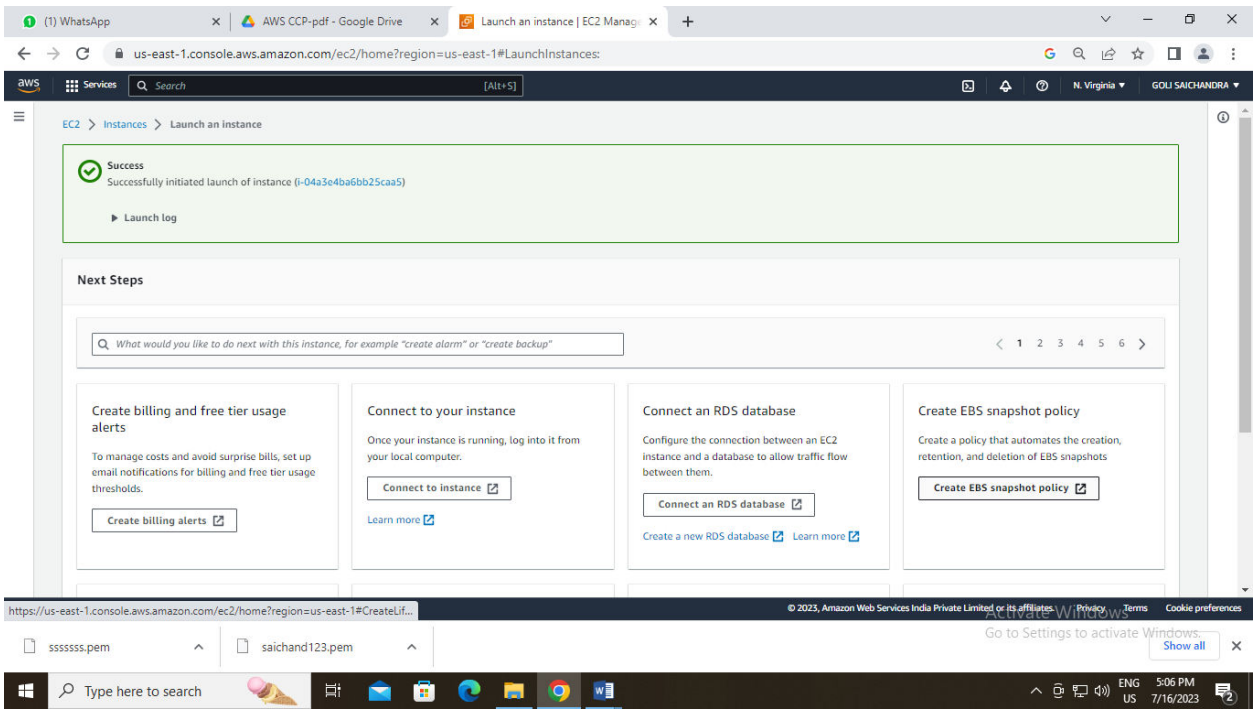
Now select the instance type and create keypair for the instance



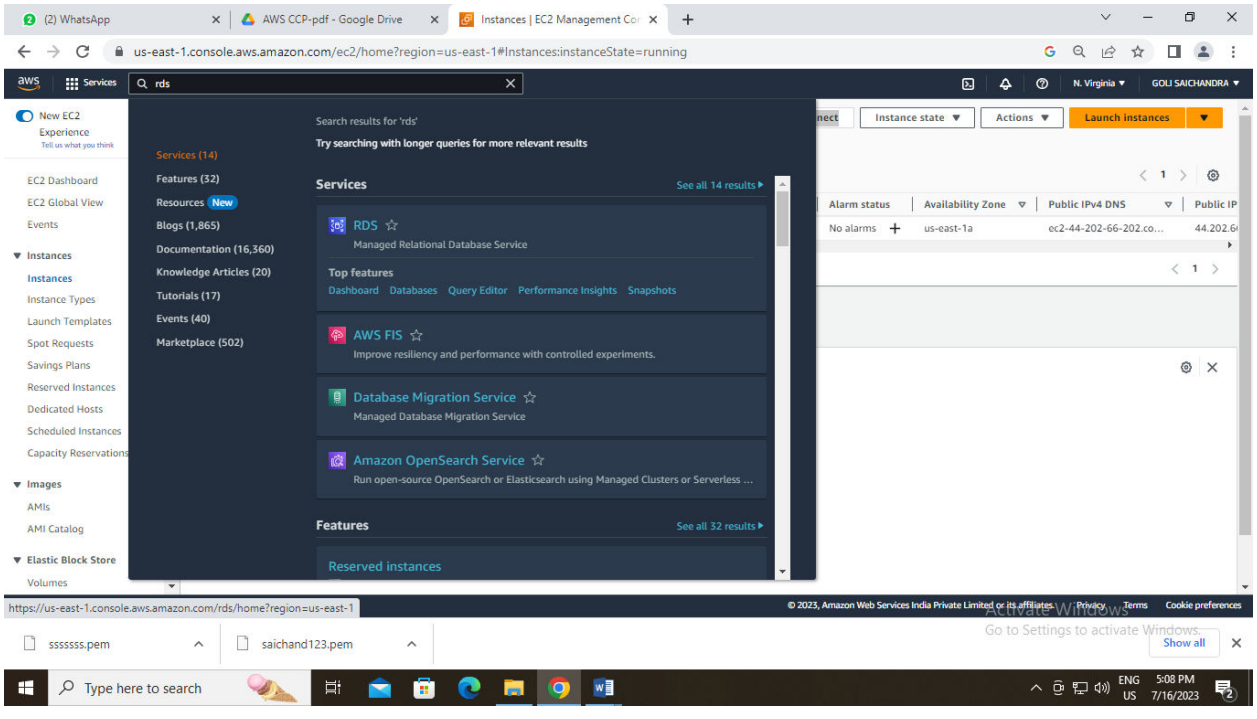
Select the security group and tick the boxes and click on launch the instance



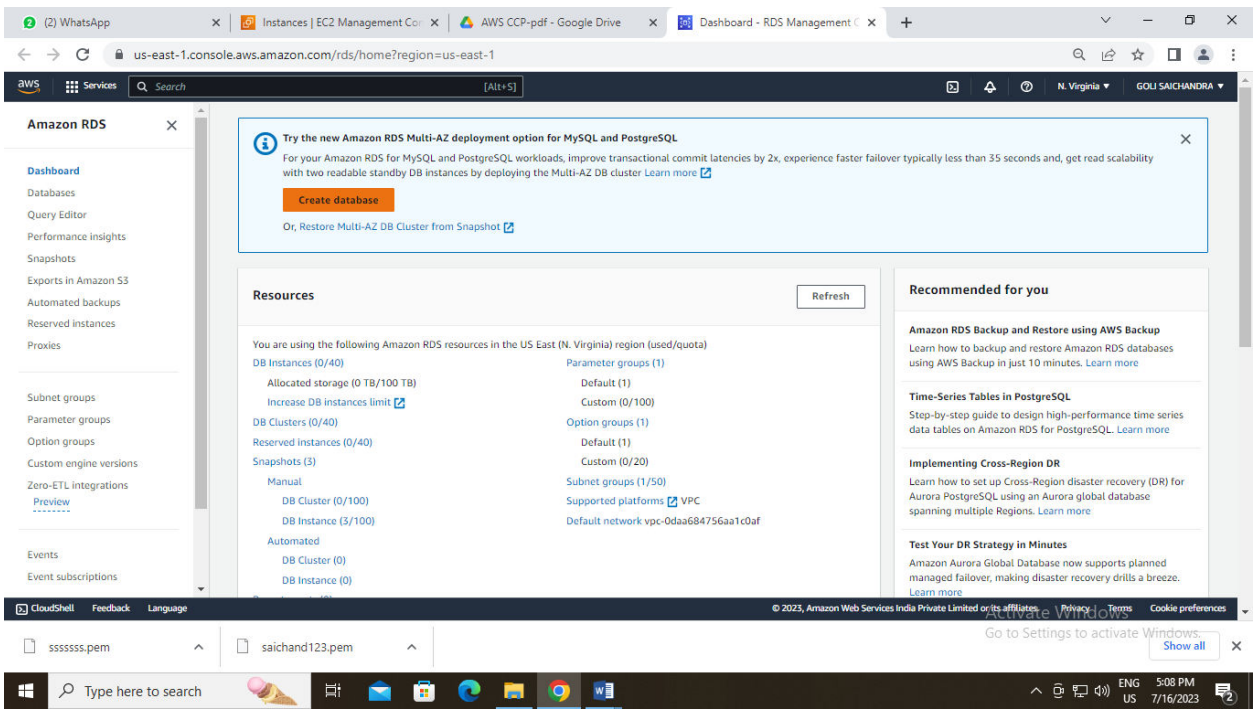
Now the instance has been launched as shown in the figure and click on instance



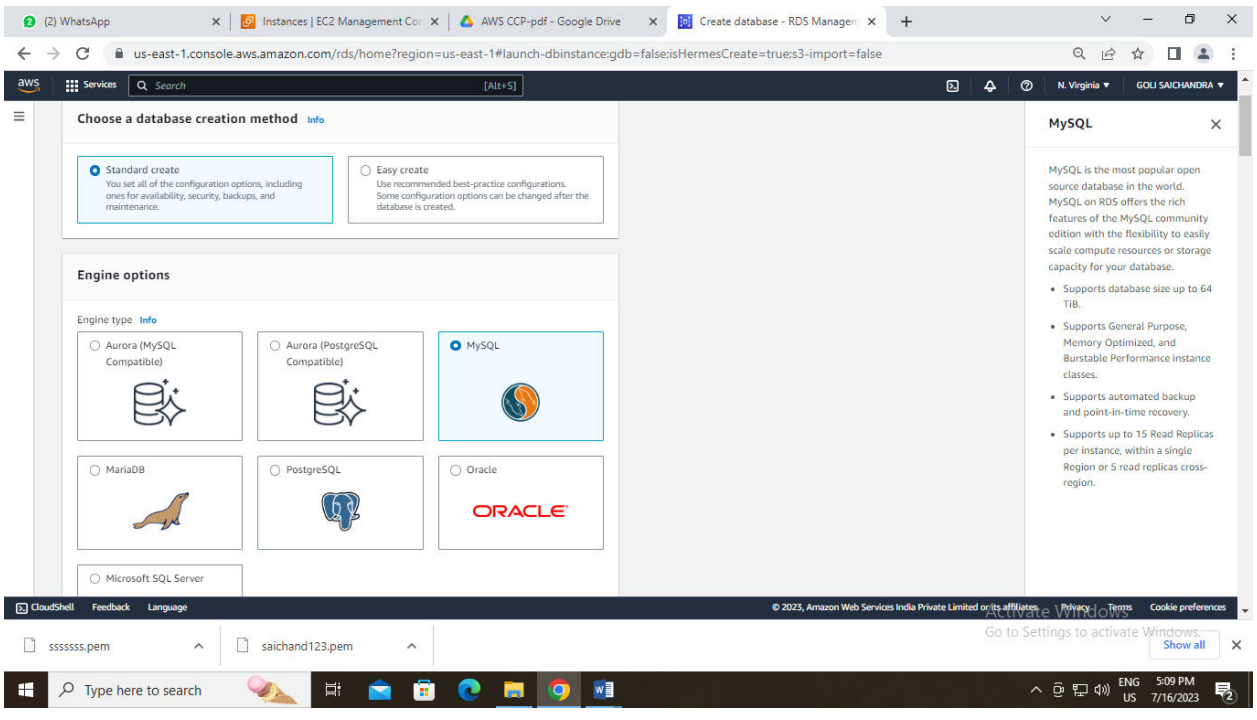
Now create RDS,



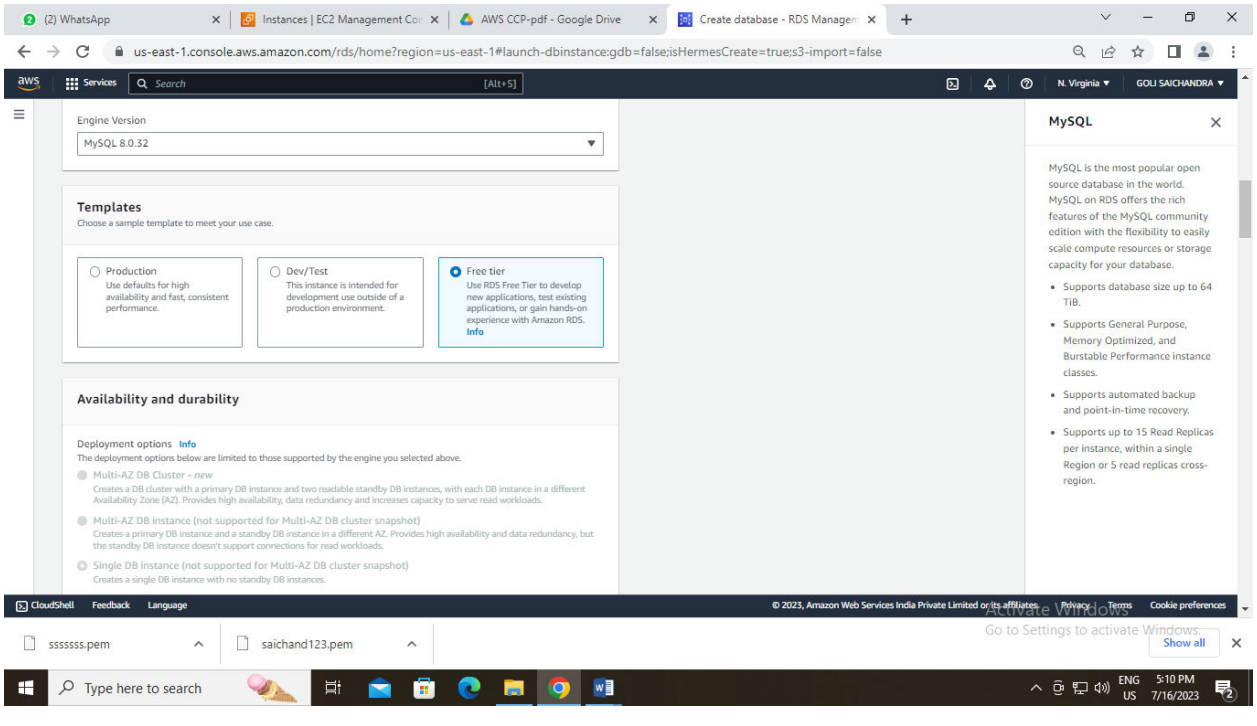
Click on create Data base



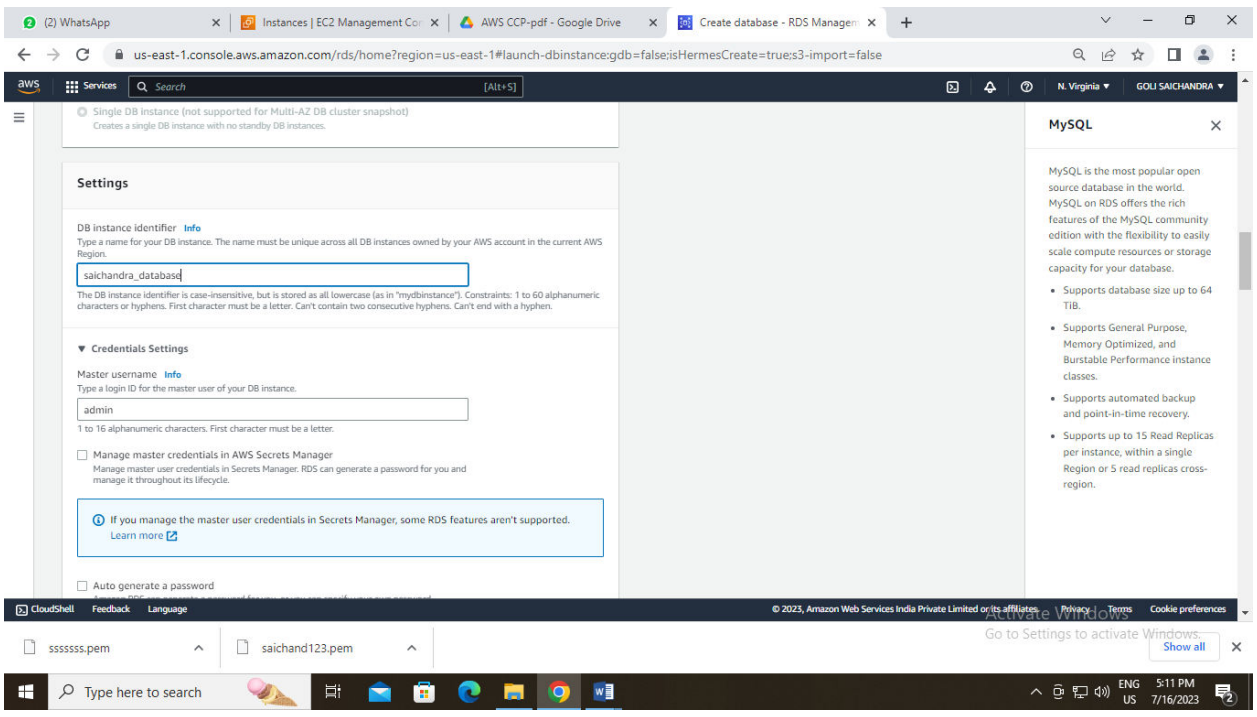
Now choose standard create and MYSQL as shown in the image



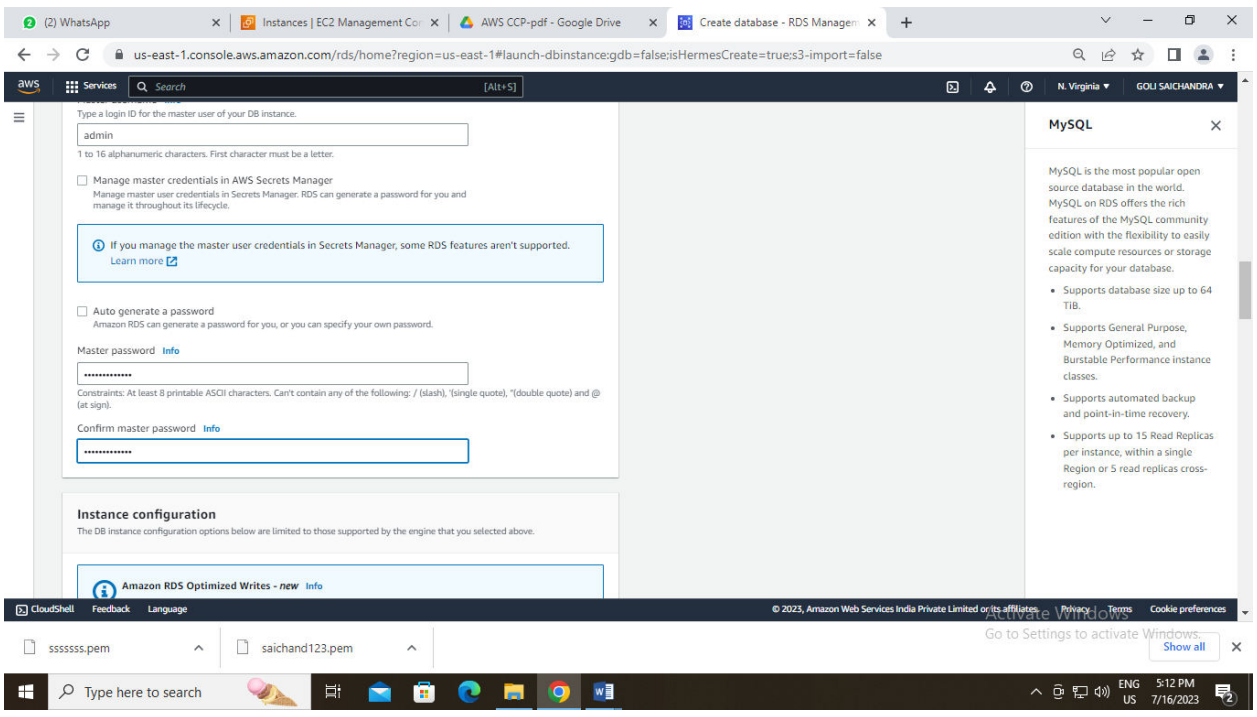
Now choose free tier



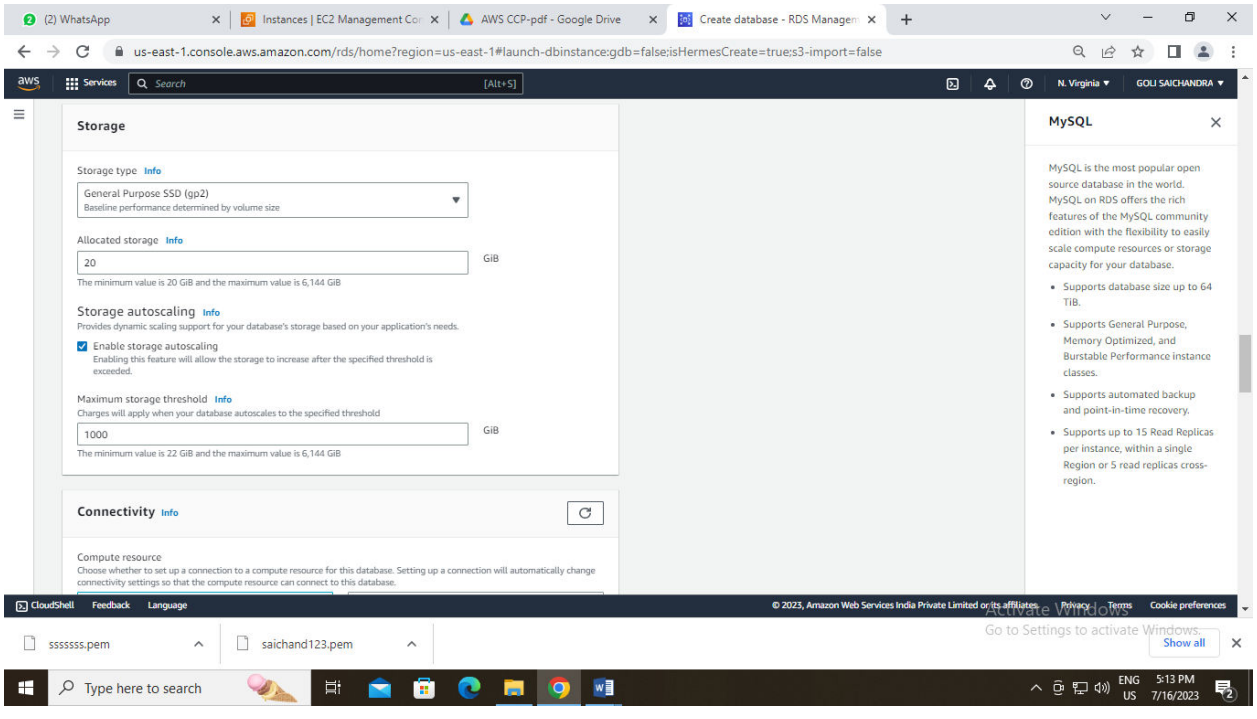
Now give a name to your data base



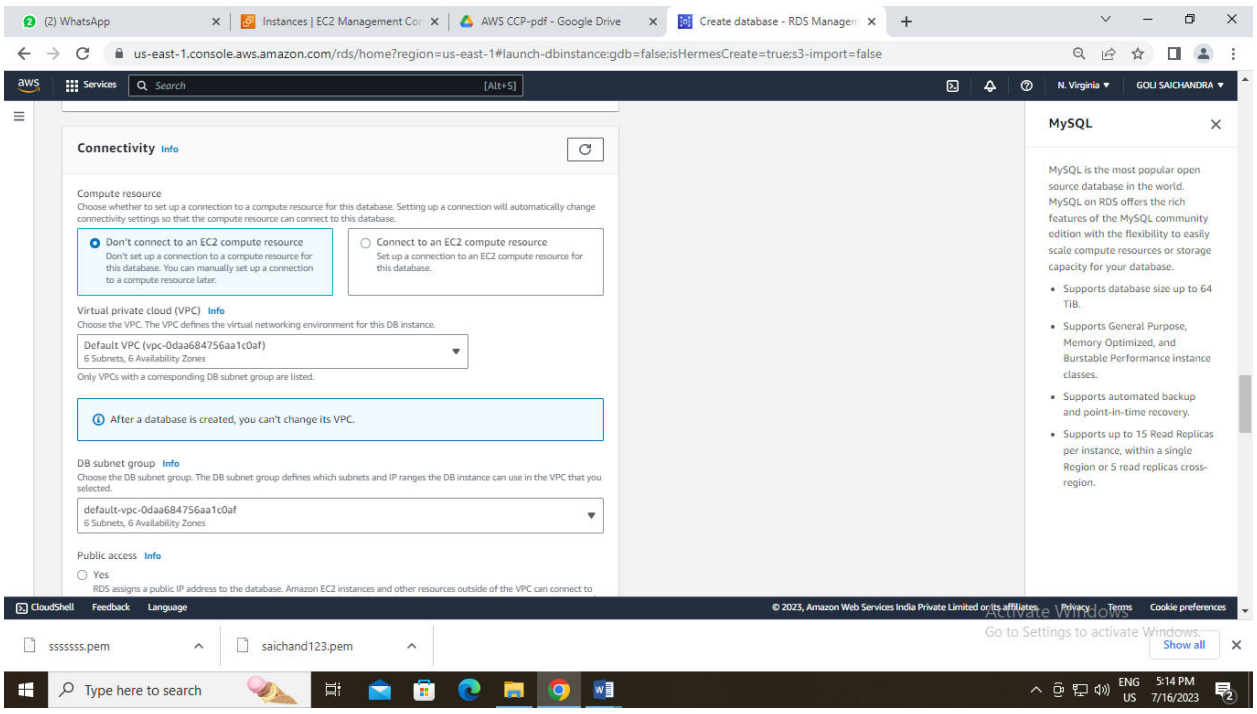
Now give a password as shown in the image as per ur wish



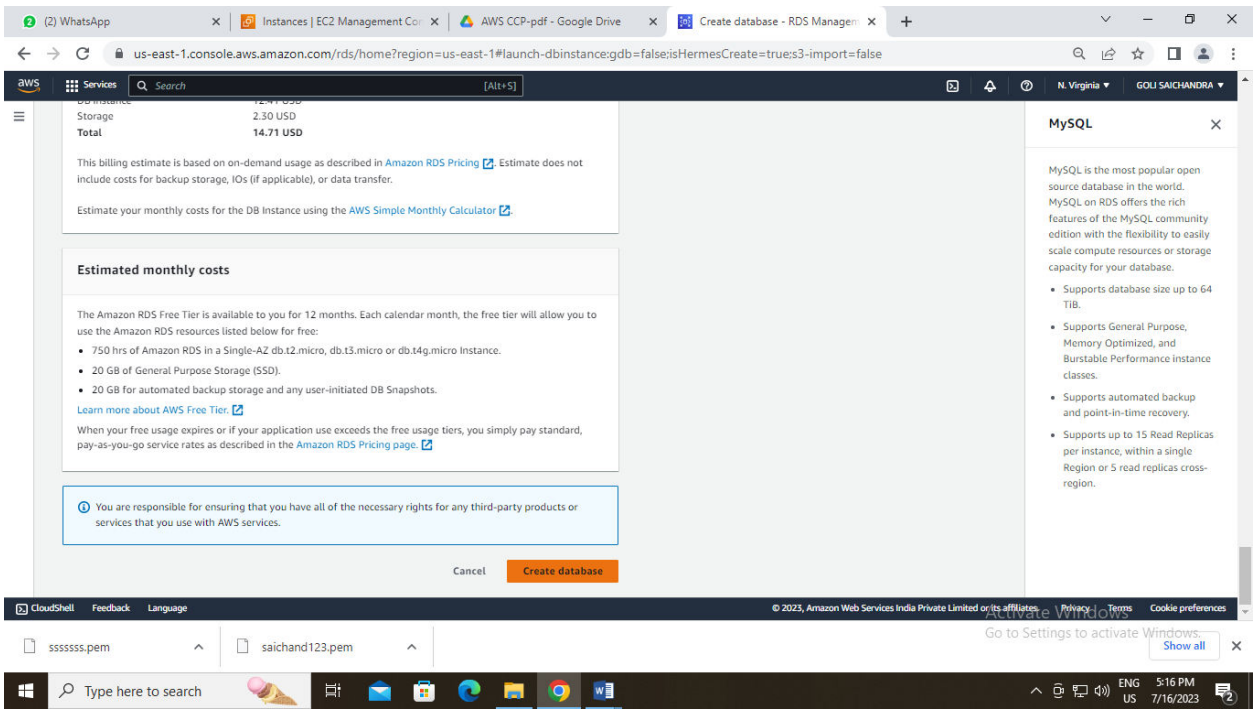
Select the features as per the requirement shown in the image



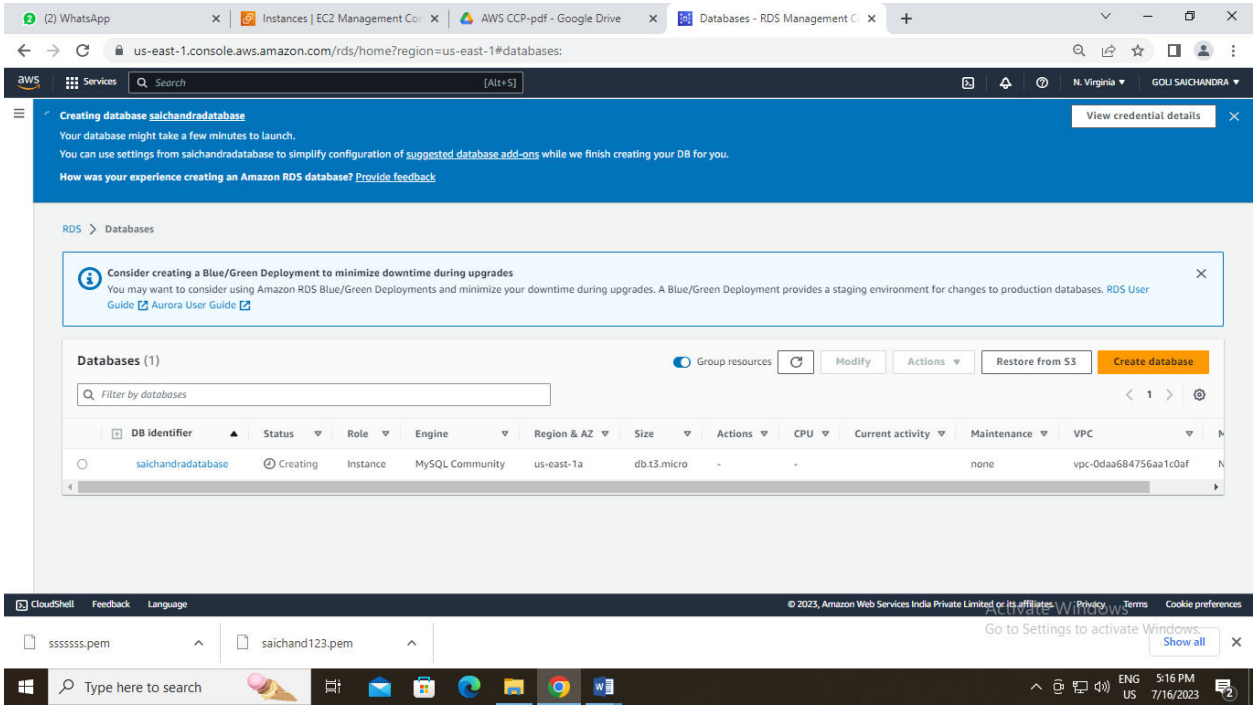
Select the below option shown in the image



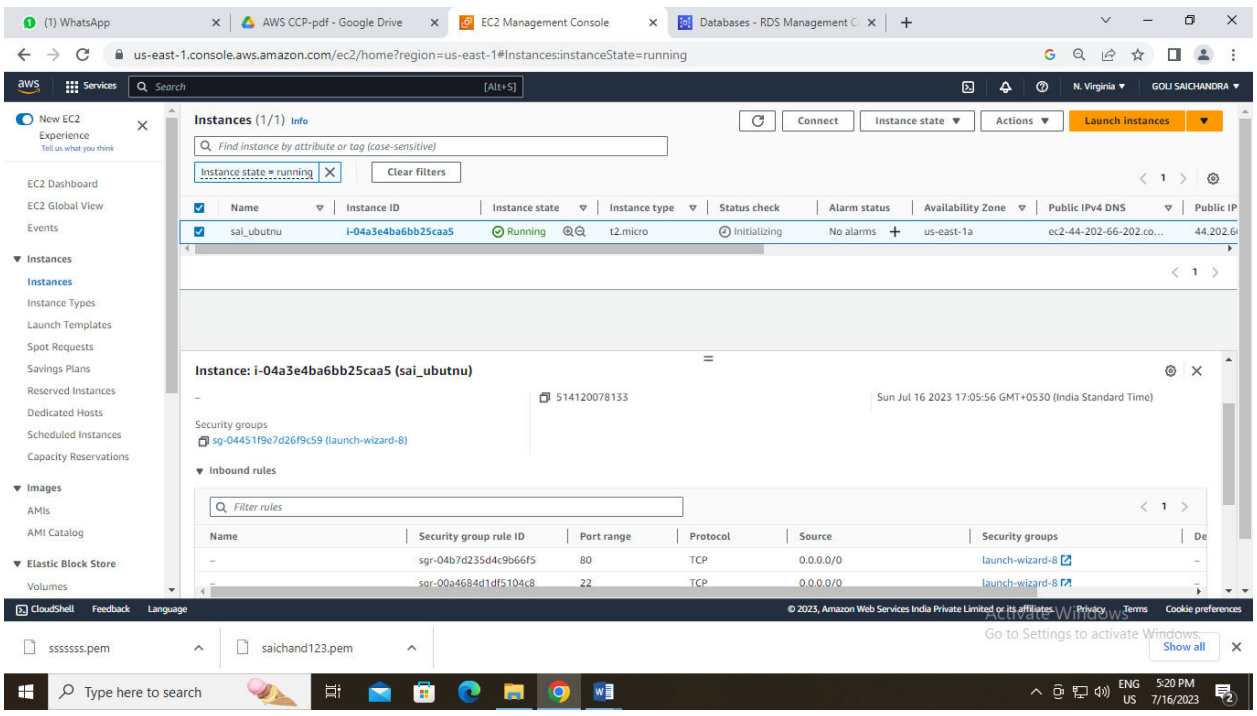
Now click on create database



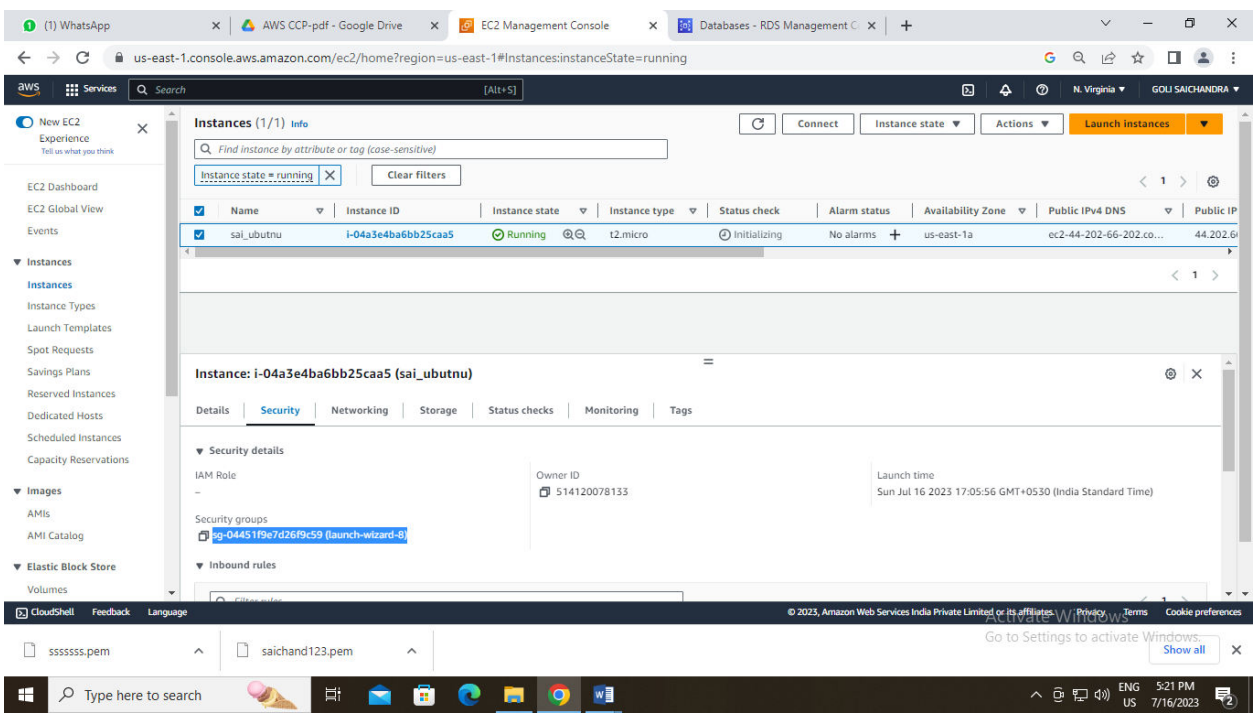
Now the RDS will be created with in 5-10 minutes



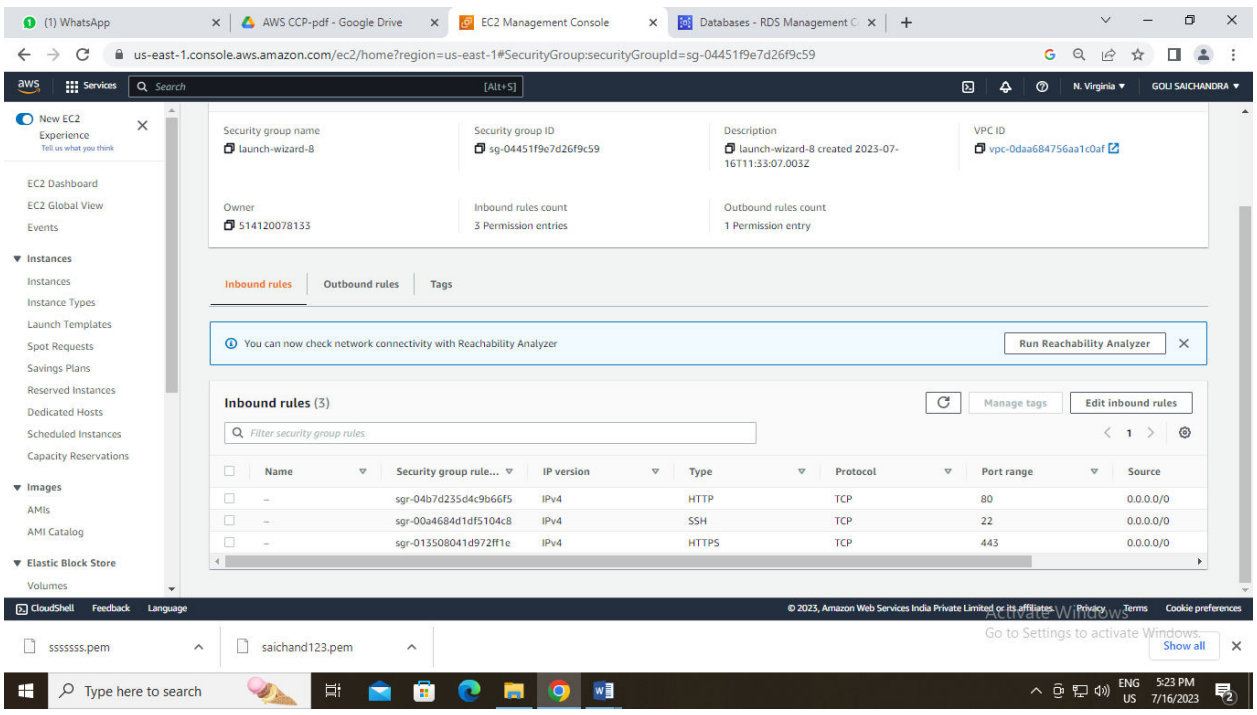
Mean while, go to EC2 and select the Ubuntu instance,

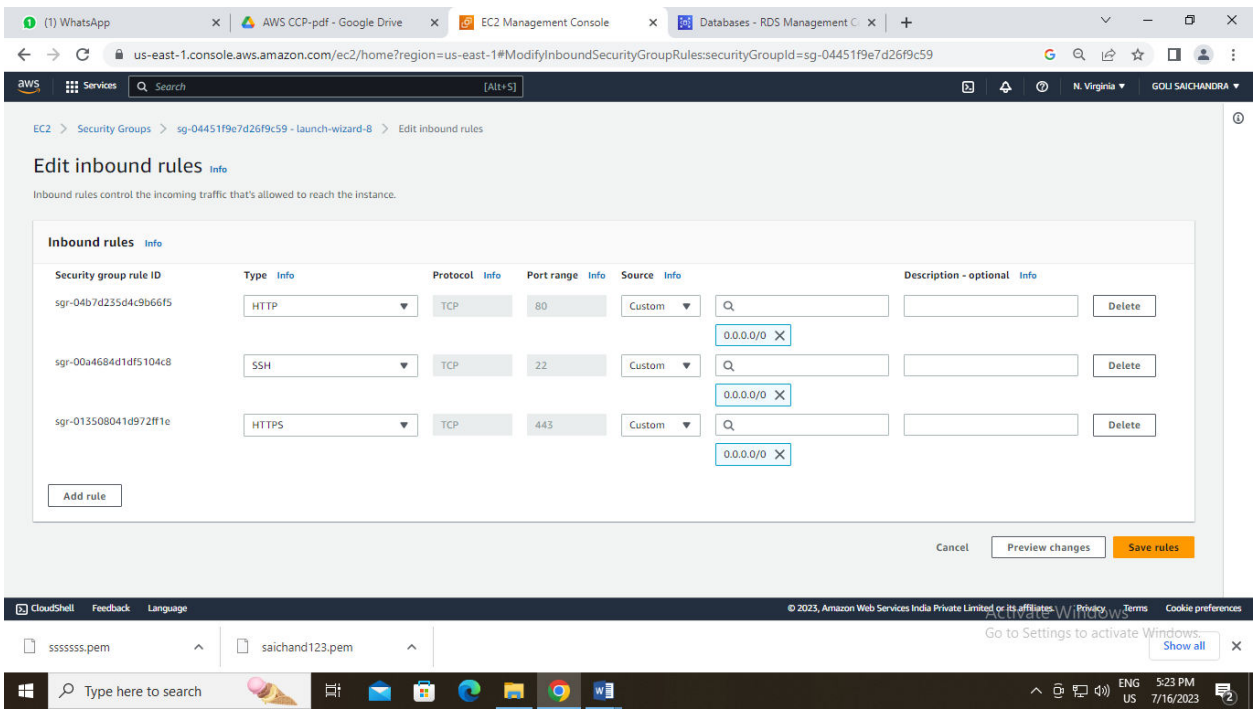


Now go to security and click on security groups as shown in the image

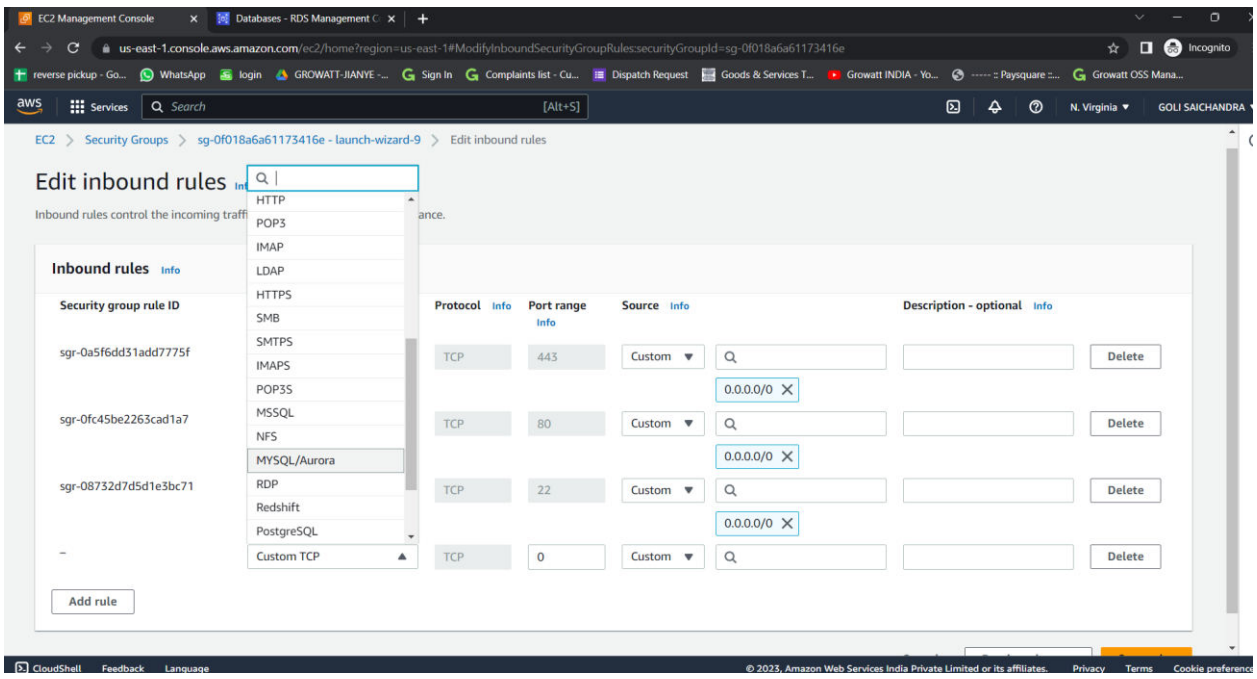


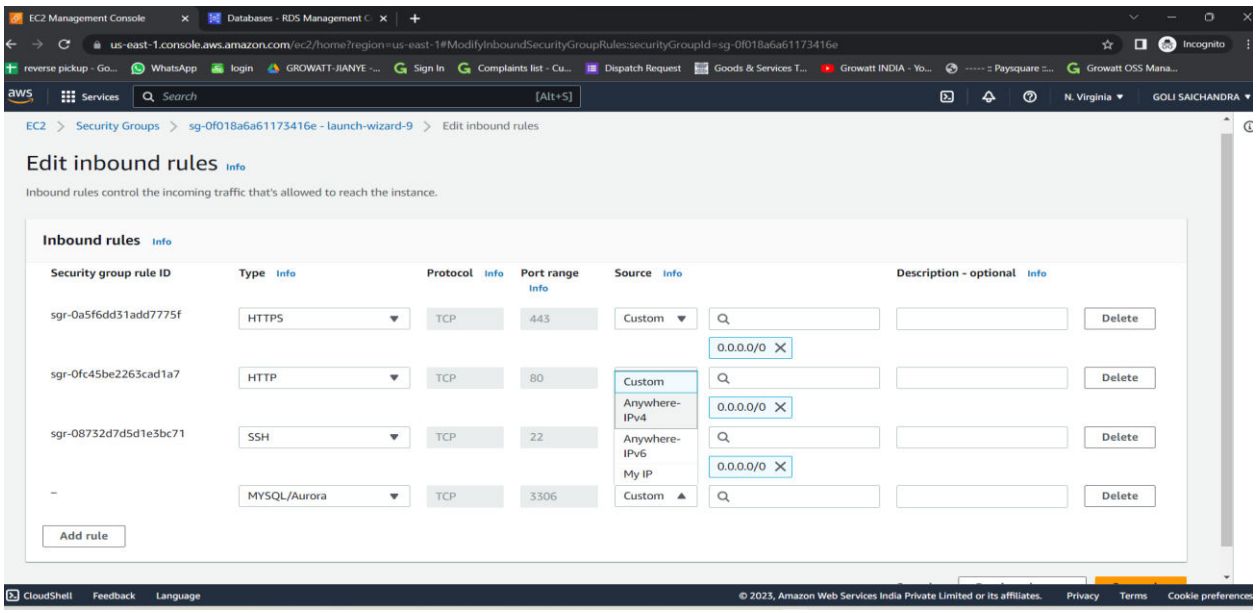
Now click on inbound rules and click on edit inbound rules

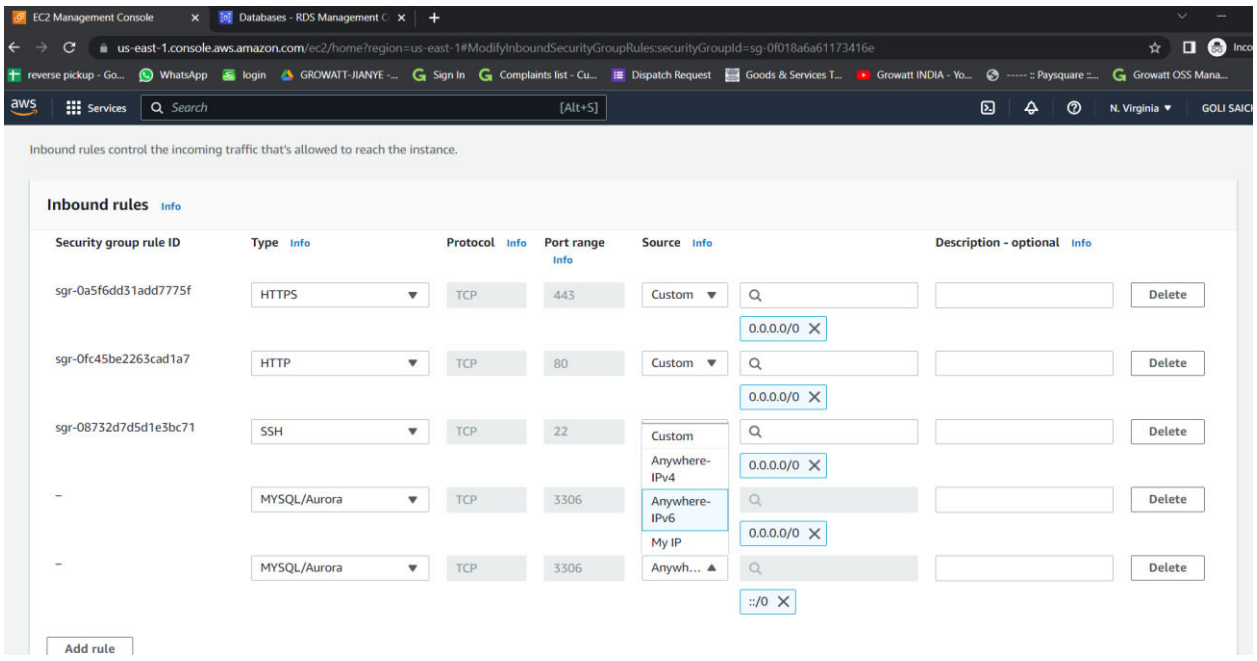


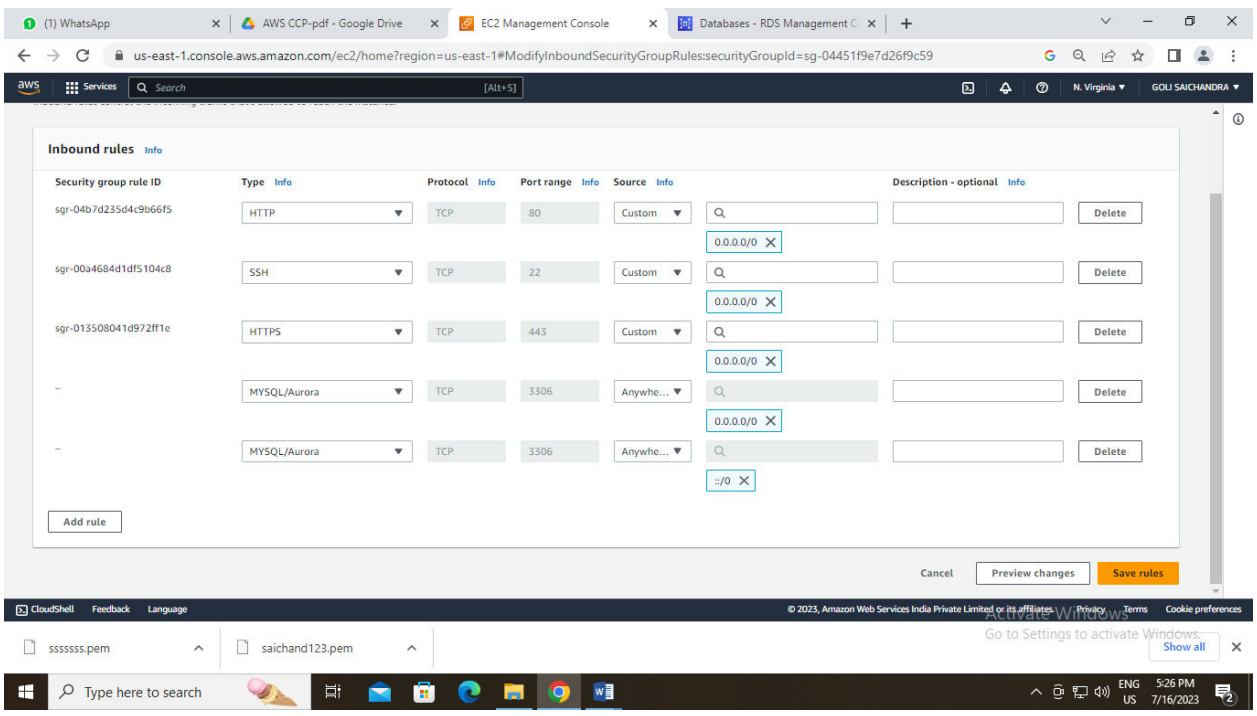


Here click on Add Rule and select MYSQL/Aurora and in the place of custom select IPV4 AND IPV6.Here I added twice once for IPV4 and once for IPV6 and click on save rules

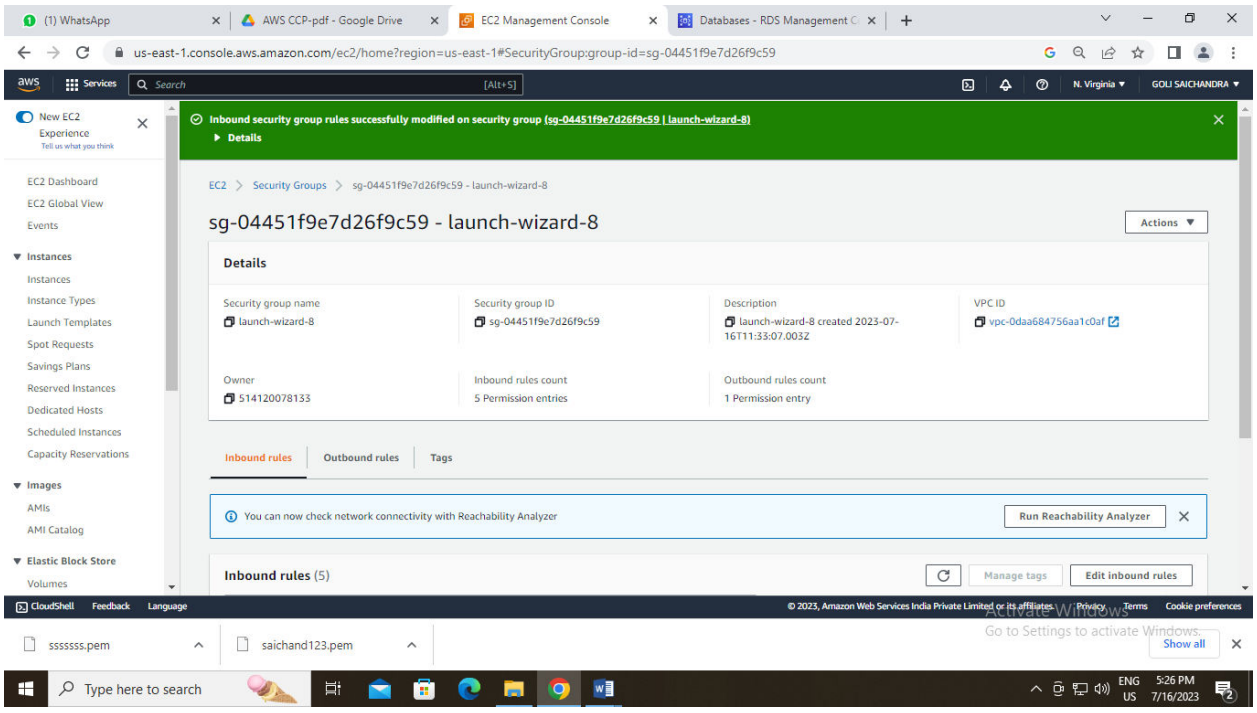



EC2 Management Console
Databases - RDS Management C
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-0f018a6a6117316e
Incognito
reverse pickup - Go...
WhatsApp
login
GROWATT-JIANYE -...
Sign In
Complaints list - Cu...
Dispatch Request
Goods & Services T...
Growatt INDIA - Yo...
----- :: Paysquare ::...
Growatt OSS Mana...
aws
Services
Search
[Alt+S]
N. Virginia
GOLI SAICHANDRA
EC2 > Security Groups > sg-0f018a6a61173416e - launch-wizard-9 > Edit inbound rules
Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-0a5f6dd31add7775f
HTTPS
TCP
443
Custom
0.0.0.0/0
Delete
sgr-0fc45be2263cad1a7
HTTP
TCP
80
Custom
Anywhere-IPv4
Anywhere-IPv6
My IP
0.0.0.0/0
Delete
sgr-08732d7d5d1e3bc71
SSH
TCP
22
0.0.0.0/0
Delete
-
MYSQL/Aurora
TCP
3306
Custom
0.0.0.0/0
Delete
Add rule
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

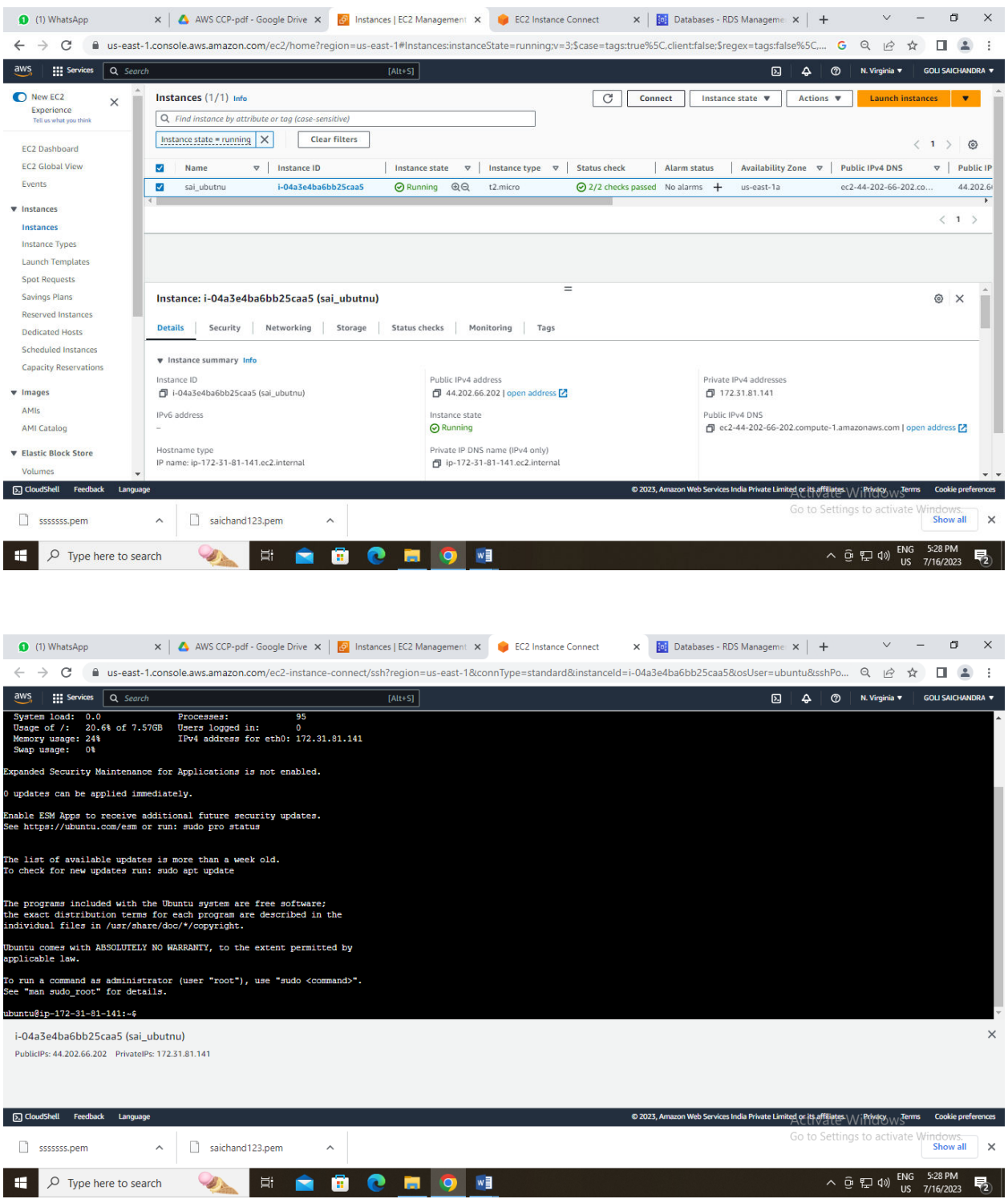

EC2 Management Console
Databases - RDS Management C
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-0f018a6a6117316e
aws
Services
Search
[Alt+S]
N. Virginia
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-0a5f6dd31add7775f
HTTPS
TCP
443
Custom
0.0.0.0/0
Delete
sgr-0fc45be2263cad1a7
HTTP
TCP
80
Custom
0.0.0.0/0
Delete
sgr-08732d7d5d1e3bc71
SSH
TCP
22
Custom
Anywhere-IPv4
Anywhere-IPv6
My IP
0.0.0.0/0
Delete
-
MYSQL/Aurora
TCP
3306
Anywhere-IPv6
0.0.0.0/0
Delete
-
MYSQL/Aurora
TCP
3306
Anywh...
::/0
Delete
Add rule


(1) WhatsApp
AWS CCP-pdf - Google Drive
EC2 Management Console
Databases - RDS Management C
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-04451f9e7d26f9c59
aws
Services
Search
[Alt+S]
N. Virginia
GOLI SAICHANDRA
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-04b7d235d4c9b66f5
HTTP
TCP
80
Custom
0.0.0.0/0
Delete
sgr-00a4684d1df5104c8
SSH
TCP
22
Custom
0.0.0.0/0
Delete
sgr-013508041d972ff1e
HTTPS
TCP
443
Custom
0.0.0.0/0
Delete
-
MYSQL/Aurora
TCP
3306
Anywhe...
0.0.0.0/0
Delete
-
MYSQL/Aurora
TCP
3306
Anywhe...
::/0
Delete
Add rule
Cancel
Preview changes
Save rules
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Activate Windows
Go to Settings to activate Windows.
ssssss.pem
saichand123.pem
Show all
Type here to search
ENG
US
5:26 PM
7/16/2023

And save it



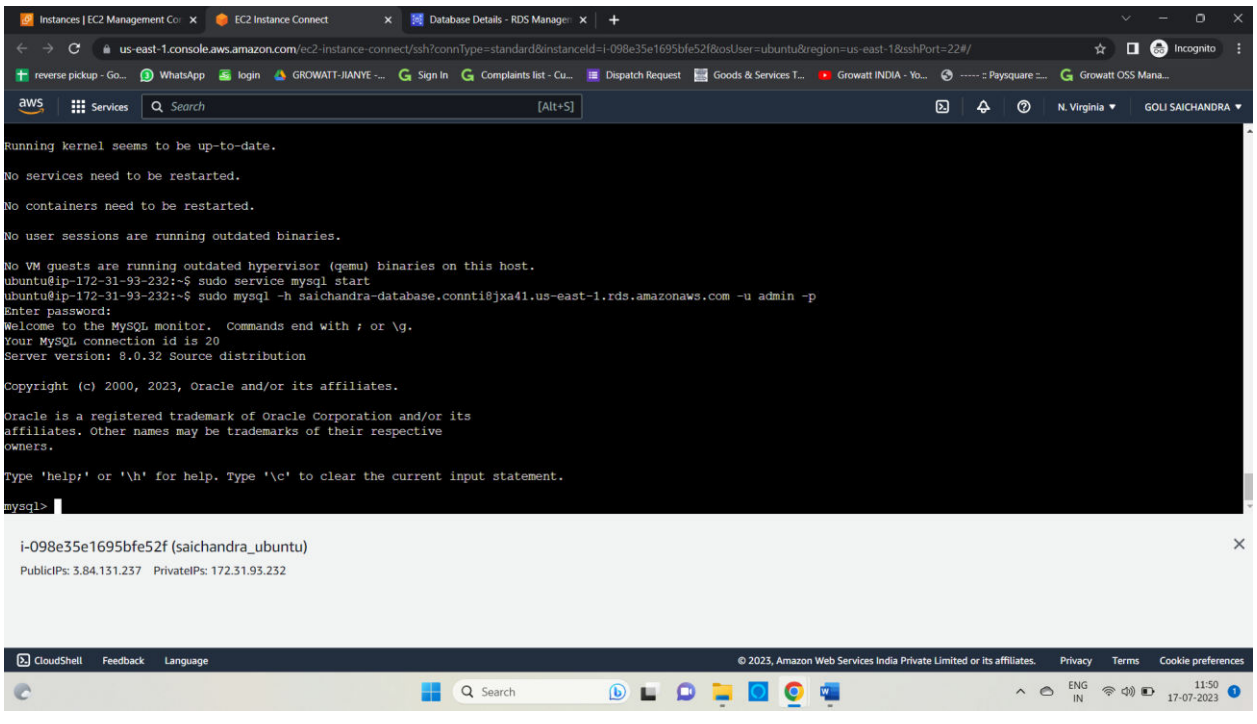
Mean while the RDS has been created and go to EC2 instance and connect to Ubuntu instance



Now write all the commands given below

Write a these commands on cmd pmt

- Sudo apt-get update
- Sudo apt-get install mysql-server
- Sudo service mysql start
- Sudo mysql –h (paste rds endpoint link) –u admin –p

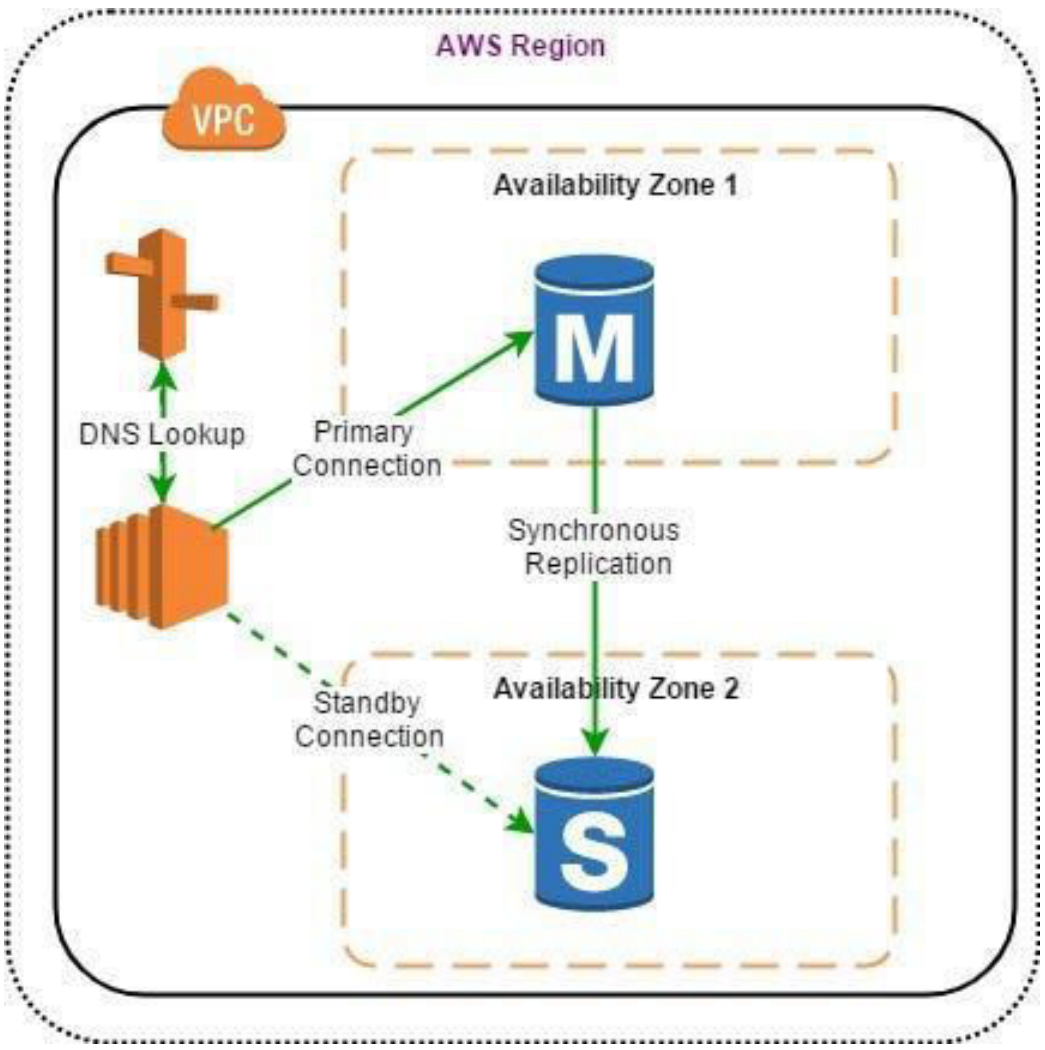


Once entering the commands followed by guidelines at the end of it we can see **mysql is been connected to the instance and started working.**

Multi-AZ & Read Replicas for RDS instances

Amazon RDS Multi-AZ and Read Replicas maintain a copy of database but they are different in nature. Use Multi-AZ deployments for High Availability/Failover and Read Replicas for read scalability.

Amazon RDS Multi-AZ



RDS Multi-AZ

Amazon RDS Multi-AZ deployments provide enhanced availability for database instances within a single AWS Region. With Multi-AZ, your data is synchronously replicated to a standby instance in a different AZ.

In the event of an infrastructure failure, Amazon RDS performs an automatic fail-over to the standby, minimizing disruption to your applications without administrative intervention.

Benefits of Multi-AZ deployment:

- Replication to a standby replica is synchronous which is highly durable.

□ Endpoint of DB instance remains the same after a failover, the application can resume database operations without manual intervention.

□ If a failure occurs, your availability impact is limited to time that automatic failover takes to complete. This helps to achieve increased availability.

□ It reduces the impact of maintenance. RDS performs maintenance on the standby first, promotes the standby to primary master, and then performs maintenance on the old master which is now a standby replica.

□ To prevent any negative impact of the backup process on performance, Amazon RDS creates a backup from the standby replica.

When a problem is detected on the primary instance, it will automatically failover to the standby in the following conditions: 1) The primary DB instance fails. 2) An Availability Zone outage. 3) The DB instance server type is changed. 4) The operating system of DB instance is undergoing software patching. 5) Manual failover of DB instance was initiated using reboot with failover.

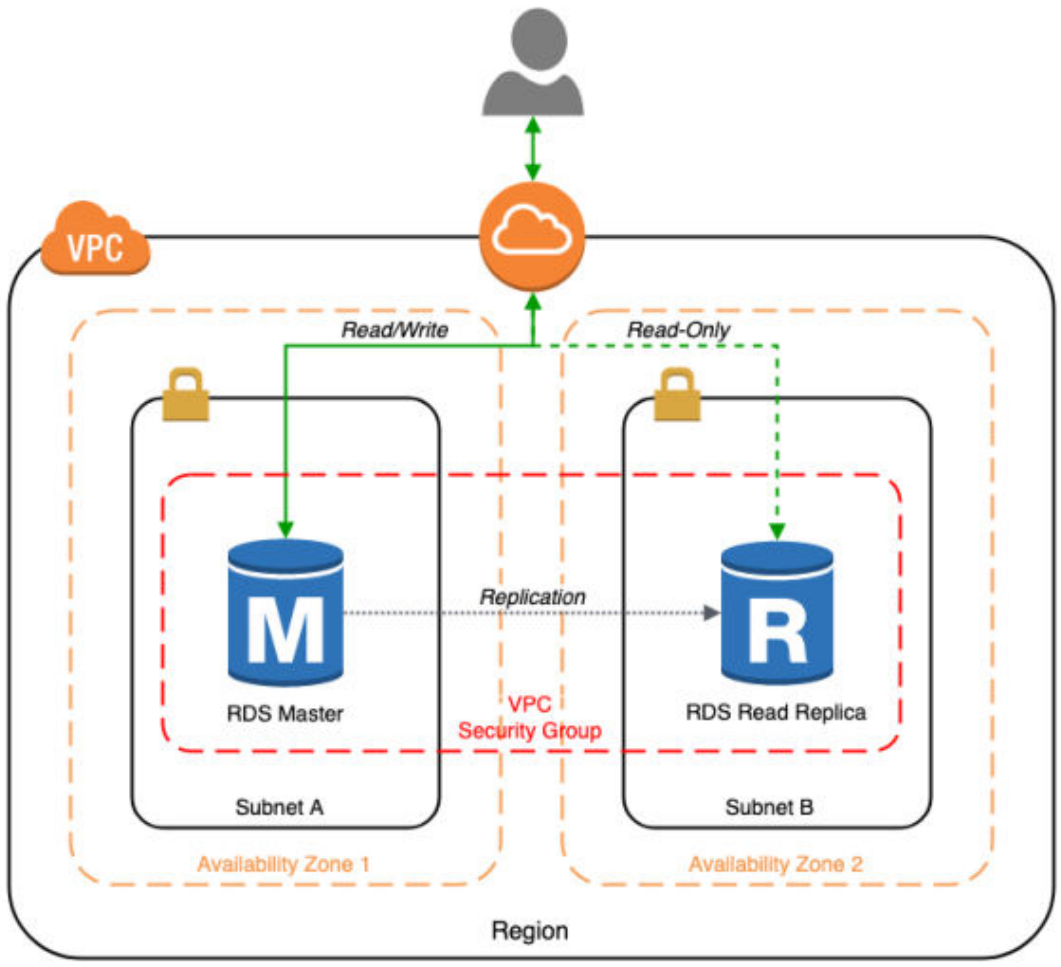
Cross-region Multi-AZ is not currently supported yet.

Multi-AZ Use Cases

□ To get high availability, and enhance availability during planned system maintenance, and help protect databases against DB instance failure and Availability Zone disruption.

□ To get data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

Amazon RDS Read Replicas



Amazon RDS Read Replicas enable you to create one or more read-only copies of your database instance within the same AWS Region or in a different AWS Region to increase the scalability.
Updates made to source database are then asynchronously copied to Read Replicas. Writes can happen in main database only and reads can happen in Read replica database.

When you create a Read Replica, you first specify an existing DB instance as the source. Then Amazon RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot. The source DB must have automatic backups enabled for setting up read replica.

Benefits of Read Replicas

□ Read Replicas helps in decreasing load on the primary DB by serving read-only traffic.

□ You can create Read Replicas within AZ, Cross-AZ or Cross-Region.

□ Read Replica can be manually promoted as a standalone database instance.

□ Read Replicas support Multi-AZ deployments.

□ You can use Read Replicas to take logical backups, if you want to store the backups externally to RDS.

□ You can have Read Replicas of Read Replicas.

□ Read Replica helps to maintain a copy of databases in a different region for disaster recovery.

□ You can have up to five Read Replicas per master, each with own DNS endpoint. Unlike a Multi-AZ standby replica, you can connect to each Read Replica and use them for read scaling.

Business reporting or data warehousing scenarios where you might want business reporting queries to run against a read replica, rather than your production DB instance.

□ Implementing disaster recovery. You can promote a read replica to a standalone instance as a disaster recovery solution if the primary DB instance fails.

□ Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. You can direct this excess read traffic to one or more read replicas.

□ Serving read traffic while the source DB instance is unavailable. In some cases, source DB instance might not be able to take I/O requests, for example due to I/O suspension for backups or scheduled maintenance. In these cases, you can direct read traffic to your read replicas.

13. Elastic Beanstalk

Deploy, manage, scale an application

AWS Elastic Beanstalk (EB) is a function for application management provided by Amazon.com as one of Amazon Web Services. With just uploading code, EB automatically handles deployments ranging from capacity provisioning, load balancing and automatic scaling to application health monitoring. The environment necessary for application operation, without having to make various settings it will be able to run on the cloud.
Elastic Beanstalk can be used from '**Amazon Management Console**' which uses AWS from a Web browser and '**AWS Toolkit for Eclipse**' (AWS Toolkit) which is a plug-in that uses AWS from Eclipse.
AWS Elastic Beanstalk deploys and scales web applications and services developed in Java , .NET , PHP , Node.js , Python , Ruby , Go , and Docker on familiar servers such as Apache, Nginx, and Passenger It is an easy-to-use service.
EB is a higher level managed "**Platform as Platform (PaaS)**" for hosting web applications. Instead of directly processing low-level AWS resources, EB creates an application environment using the web interface, selects the platform used by the application, creates and uploads source bundles, and EB handles the rest Platform management platform.
With EB you can take advantage of all sorts of built-in functions to monitor the application environment and introduce new versions of the application. In EB, cloud formation is used to create and manage various AWS resources of the application. One can customize and extend the default EB environment by adding the cloud formation resource to the EB configuration file deployed in the application.

Benefits of AWS Elastic Beanstalk

- Best and simplest way to deploy your application on Amazon Web Services
- No need to spend time on developing expertise
- Automatically scales the application up and down
- Freedom to choose AWS resources necessary for the application

What is CloudFormation (CFn)

CloudFormation (CFn) is a lighter, lower level abstraction than the existing AWS API. AWS CFn allows developers and system administrators to easily create, manage, and provision and update collections of related AWS resources in an orderly, predictable way. Use a static JSON / YAML template document to declare a set of Resources (such as EC2 instance and S3 bucket) corresponding to the CRUD operation of the AWS API.
When you create the CFn stack, CFn calls the corresponding API to create the associated resources, and when you delete the stack it calls the corresponding API to delete it. Most AWS APIs (not all) are supported.

Main Features of AWS Elastic Beanstalk

Immediate Launch Application – Just deploy the application from Eclipse or web browser to launch the application. Like the world's PaaS, you can easily run applications.
Monitoring Function – One can easily check the CPU utilization, the number of requests, and even the log of the Tomcat server from the screen. In addition, it is also possible to notify by e-mail at the timing of state changes such as the addition and deletion of application servers.
Selection of Database – You can select 'Amazon RDS ', 'Amazon SimpleDB', 'Microsoft SQL Server', 'Oracle' as the database server to use. You can select the database according to the application.
Loose Limit – The entity on Elastic Beanstalk is a virtual machine that runs on Amazon EC2. Site access, multithreading, process calls etc are not subject to limitations.

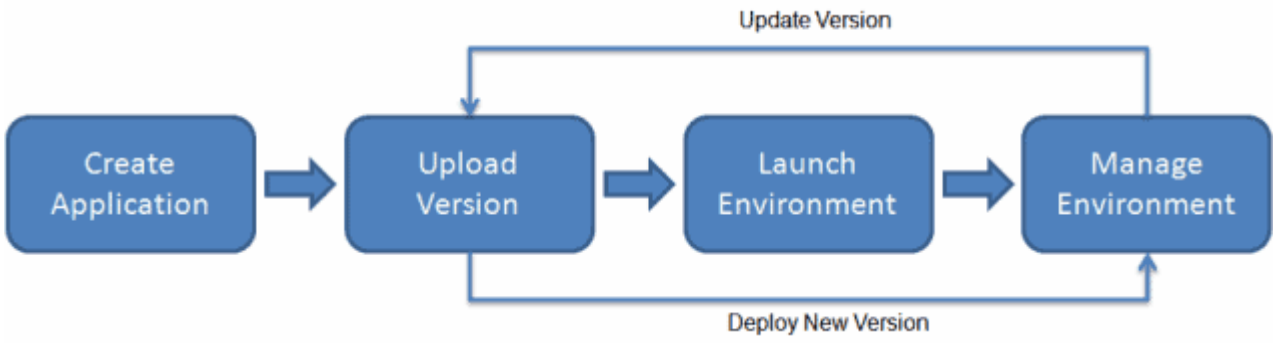
Basically, it's Free – Although Elastic Beanstalk's own features are available for free, instances, storage, and network traffic are charged as with regular AWS. EB's addition is free, so there is no hand to use.

Conclusion

AWS Elastic Beanstalk can be used if your application is a standard Web-tier application that uses one of EB's supported platforms, and you want to manage applications easily and require highly scalable hosting.

Workflow of Elastic Beanstalk

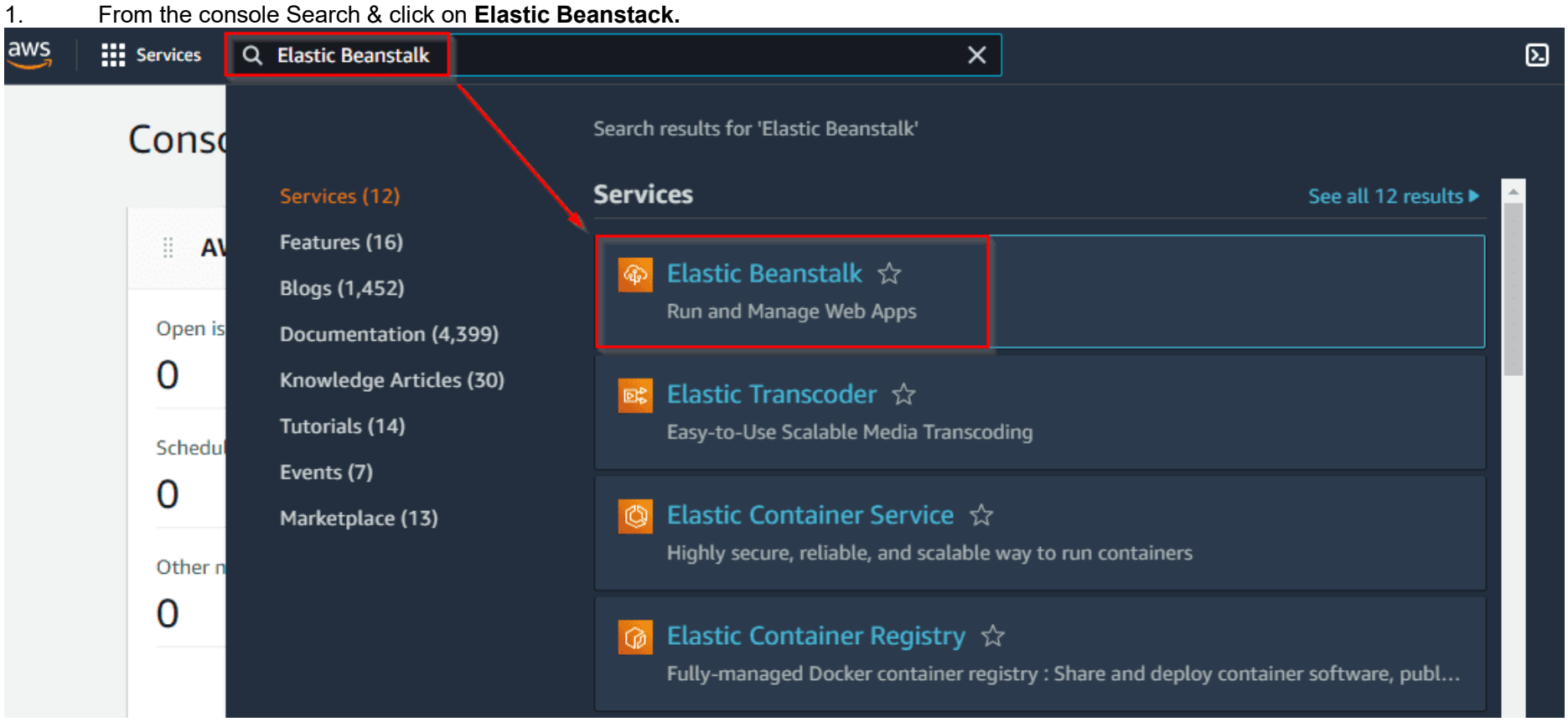
You can construct an application using Elastic Beanstalk, upload an application version in the form of an application code bundle (for instance, a Python.war file), and then provide some information about the program. The AWS resources required to run your code are automatically created and configured by Elastic Beanstalk. **You can manage your environment and roll out new application versions once your environment has launched**. The workflow of Elastic Beanstalk is shown in the diagram below.



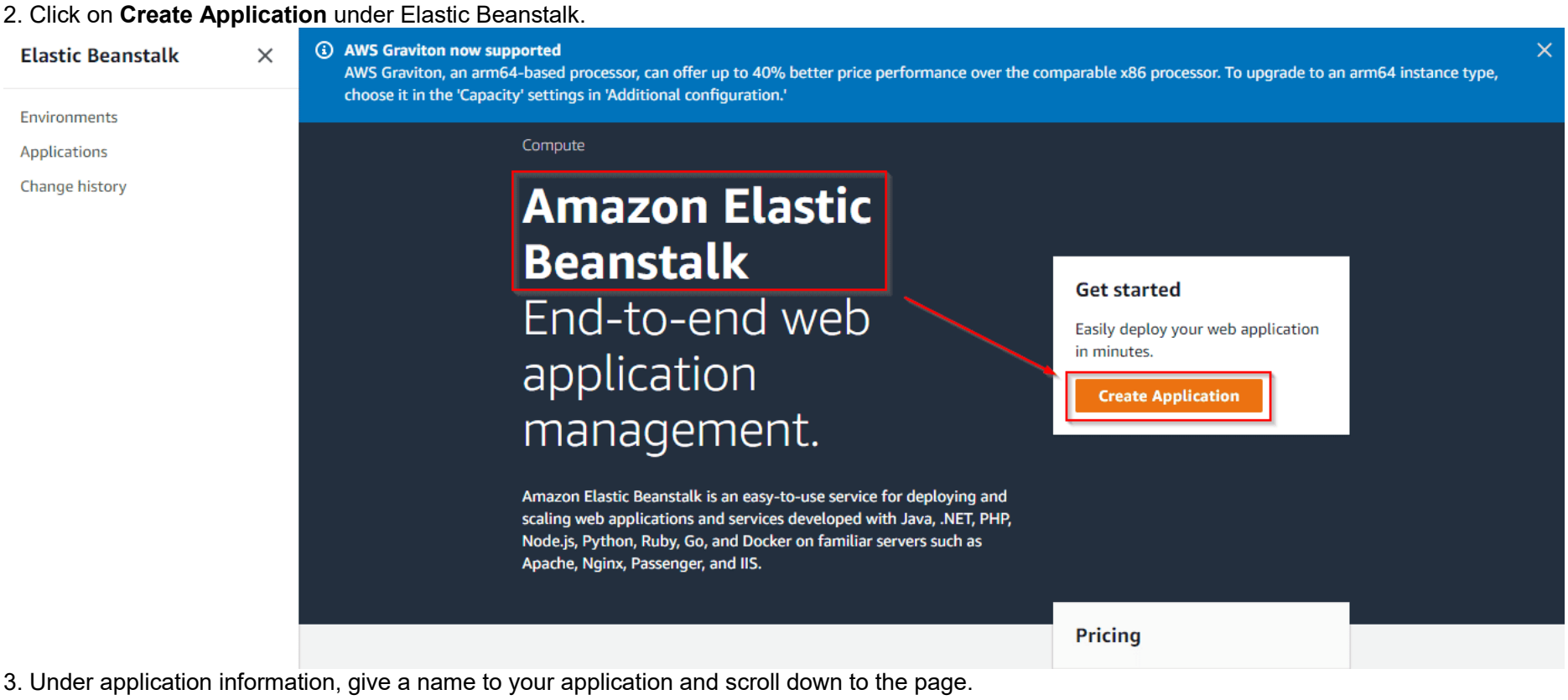
It supports the DevOps practice name **"rolling deployments."** When **enabled, your configuration deployments work hand in hand with Auto Scaling** to ensure there are always a defined number of instances available as configuration changes are made. It gives you control as Amazon EC2 instances are updated

Create Application

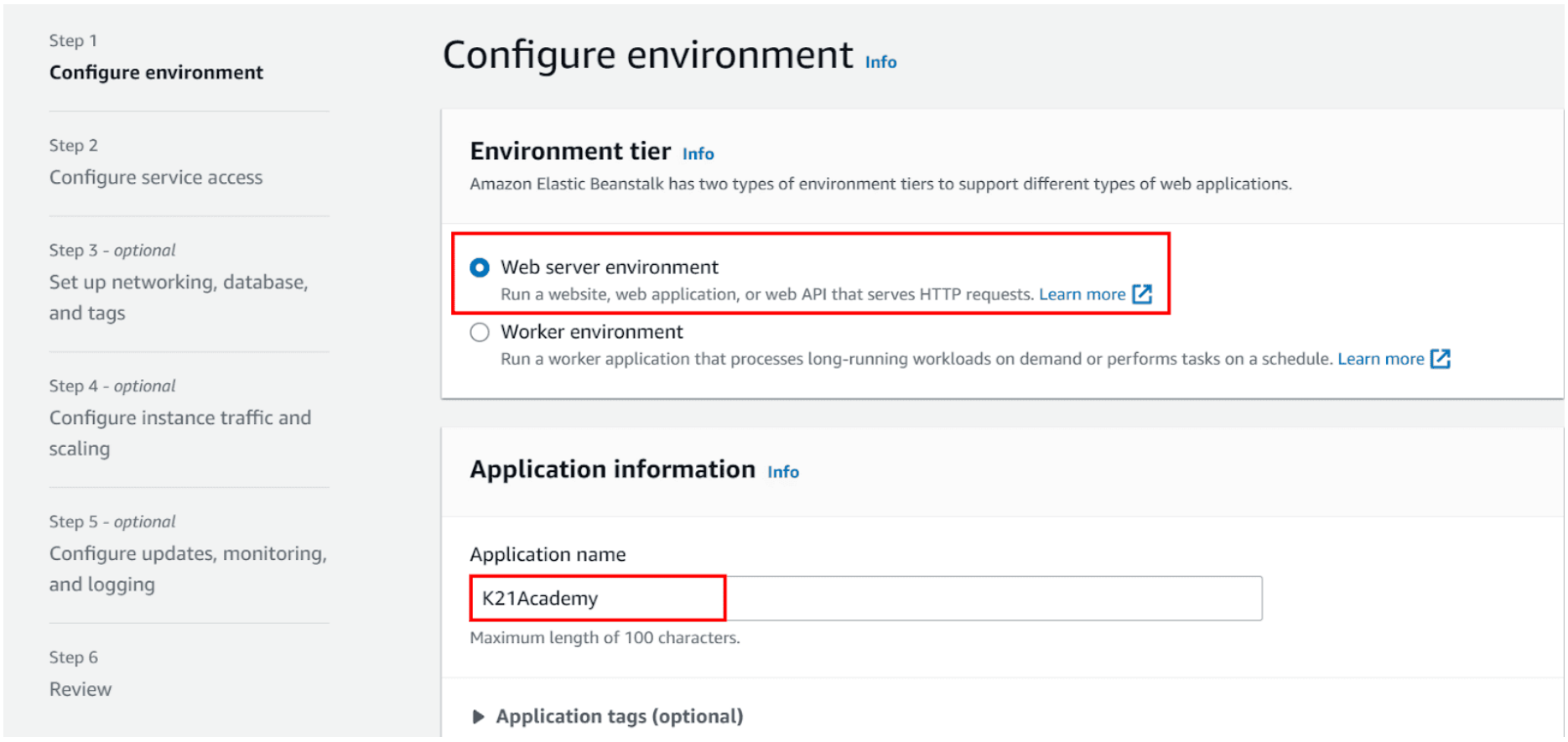
1. From the console Search & click on **Elastic Beanstack.**



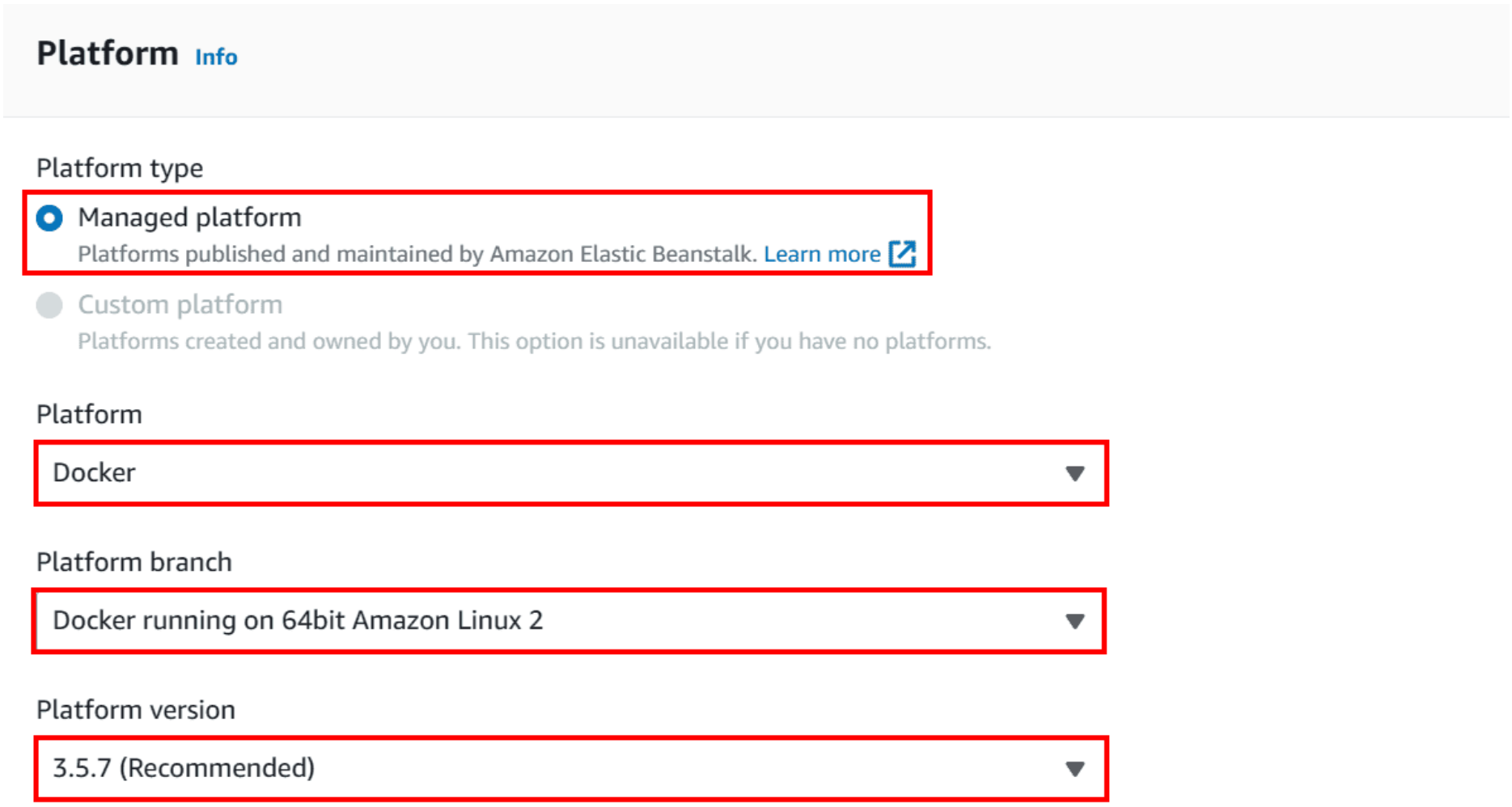
2. Click on **Create Application** under Elastic Beanstalk.



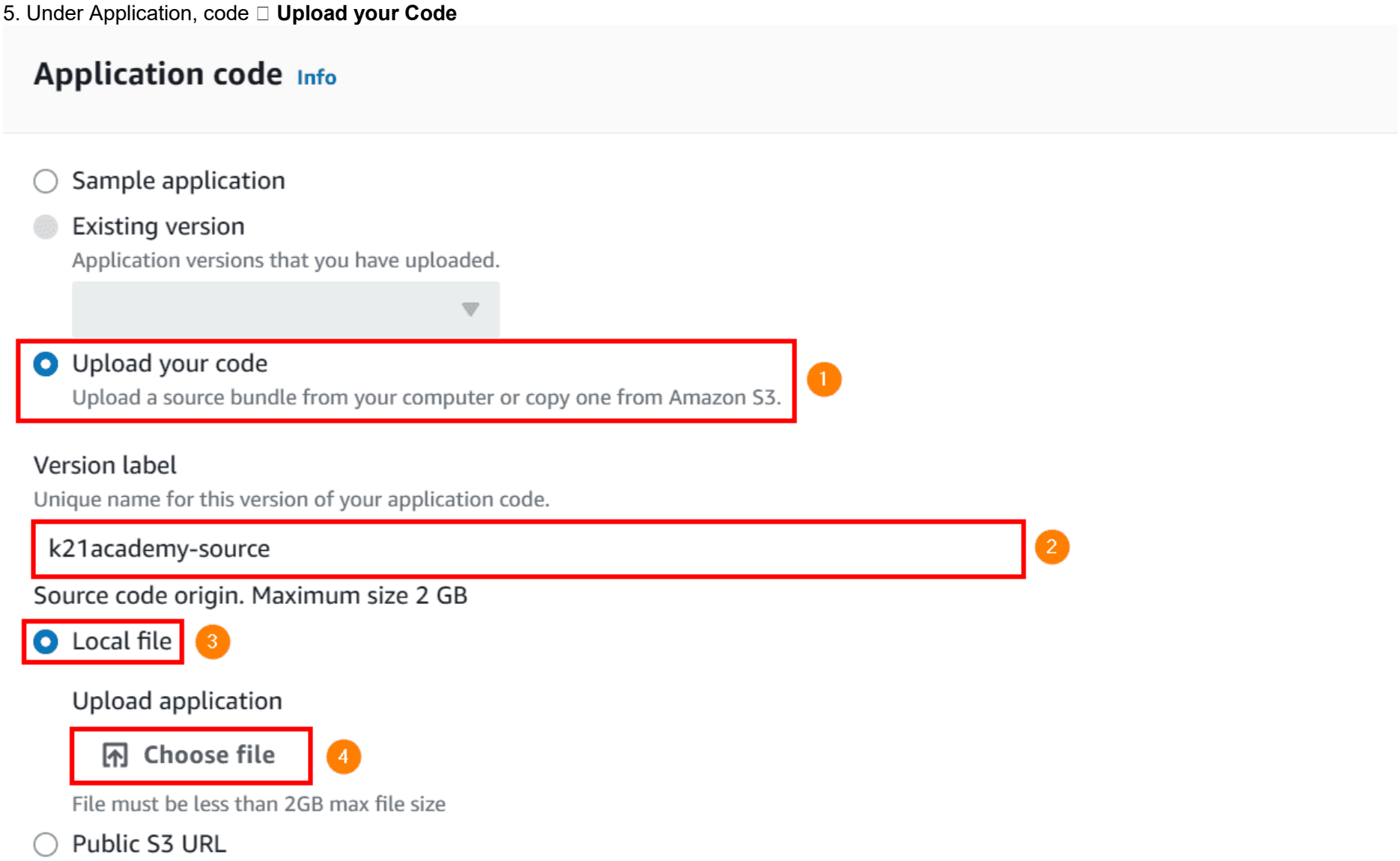
3. Under application information, give a name to your application and scroll down to the page.



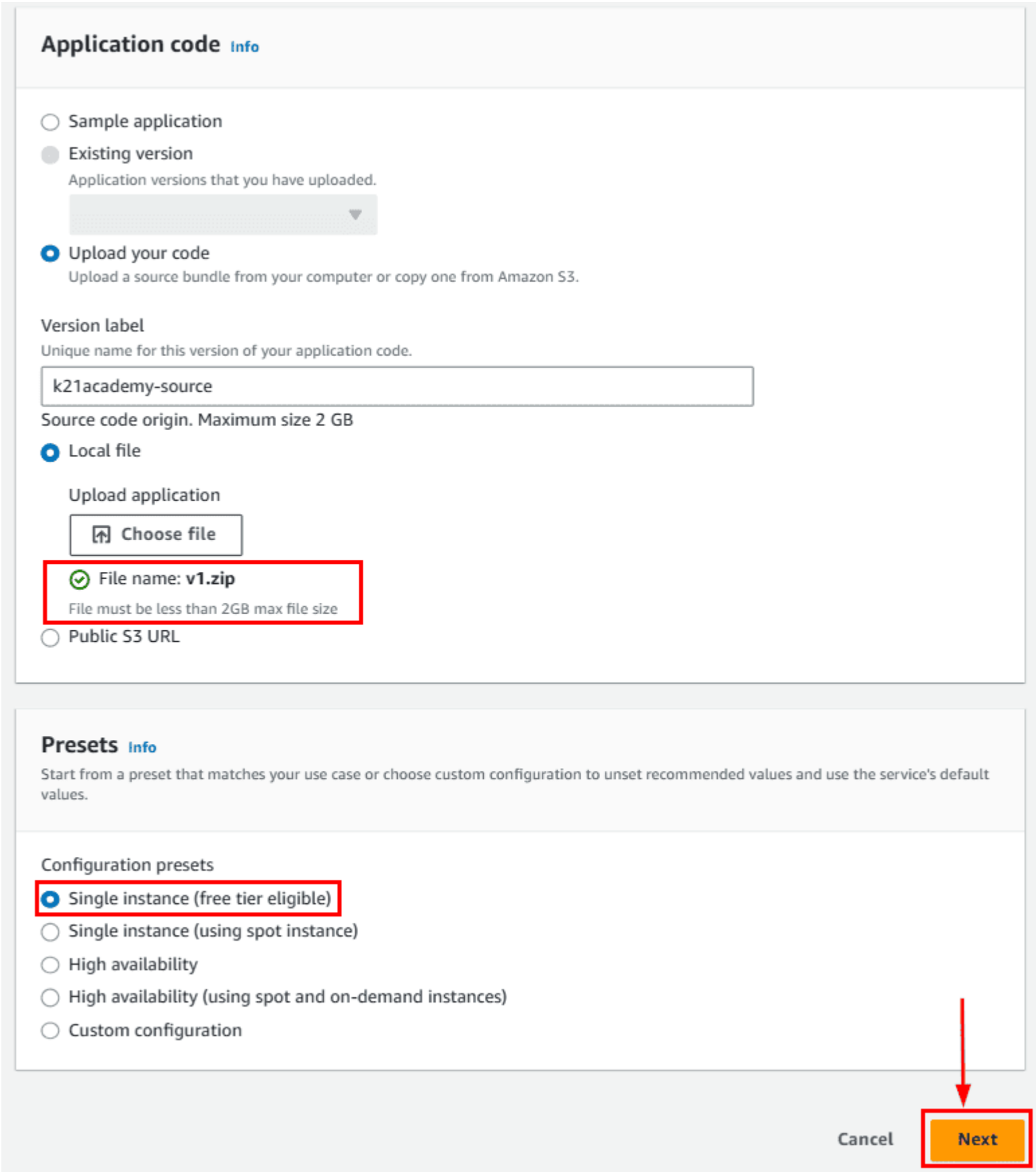
4. Under platform select the following configuration and click on Create application.
Platform **Docker**
Platform Branch **Docker running on 64bit Amazon Linux**
Platform version Set it to default



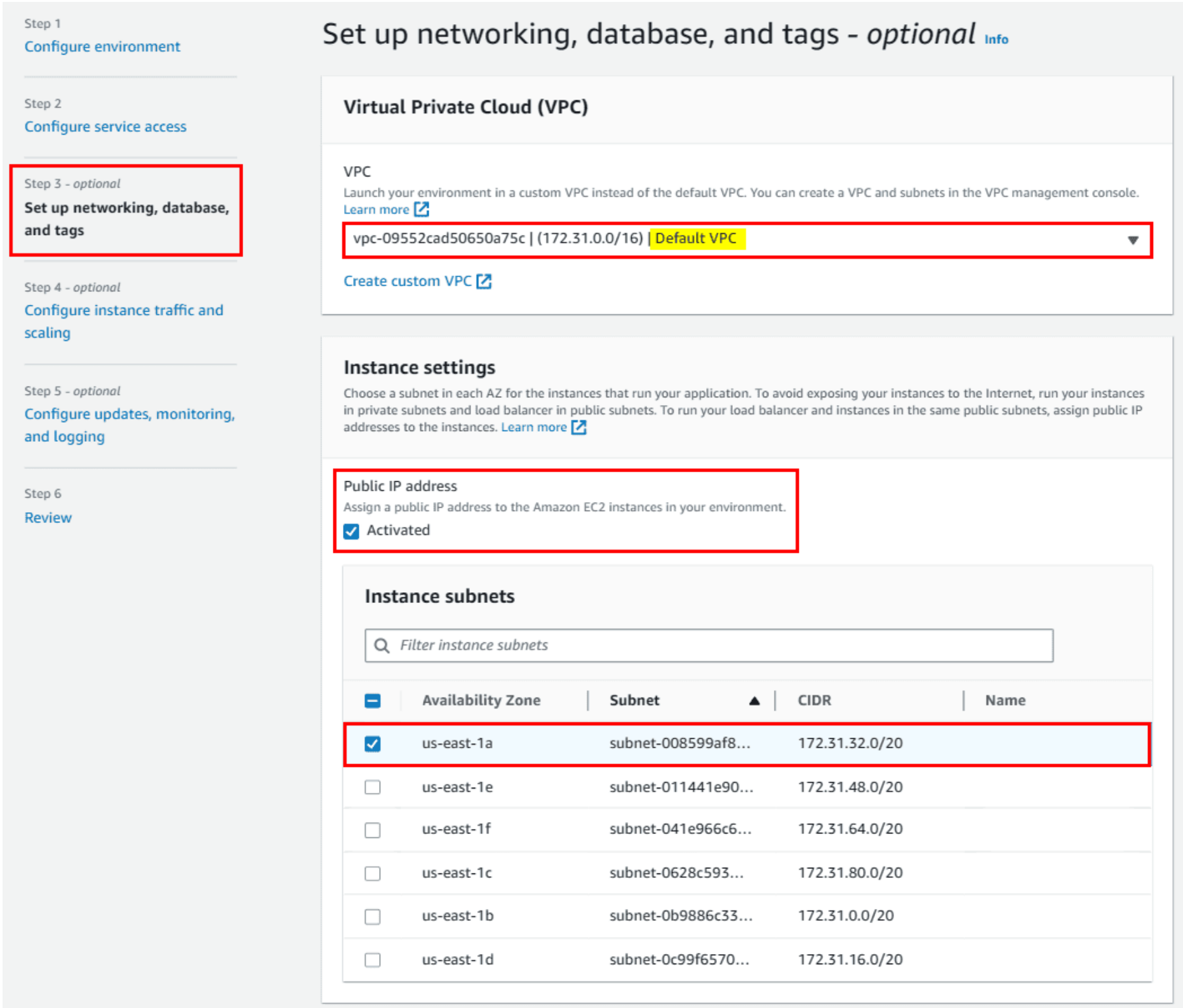
5. Under Application, code □ **Upload your Code**



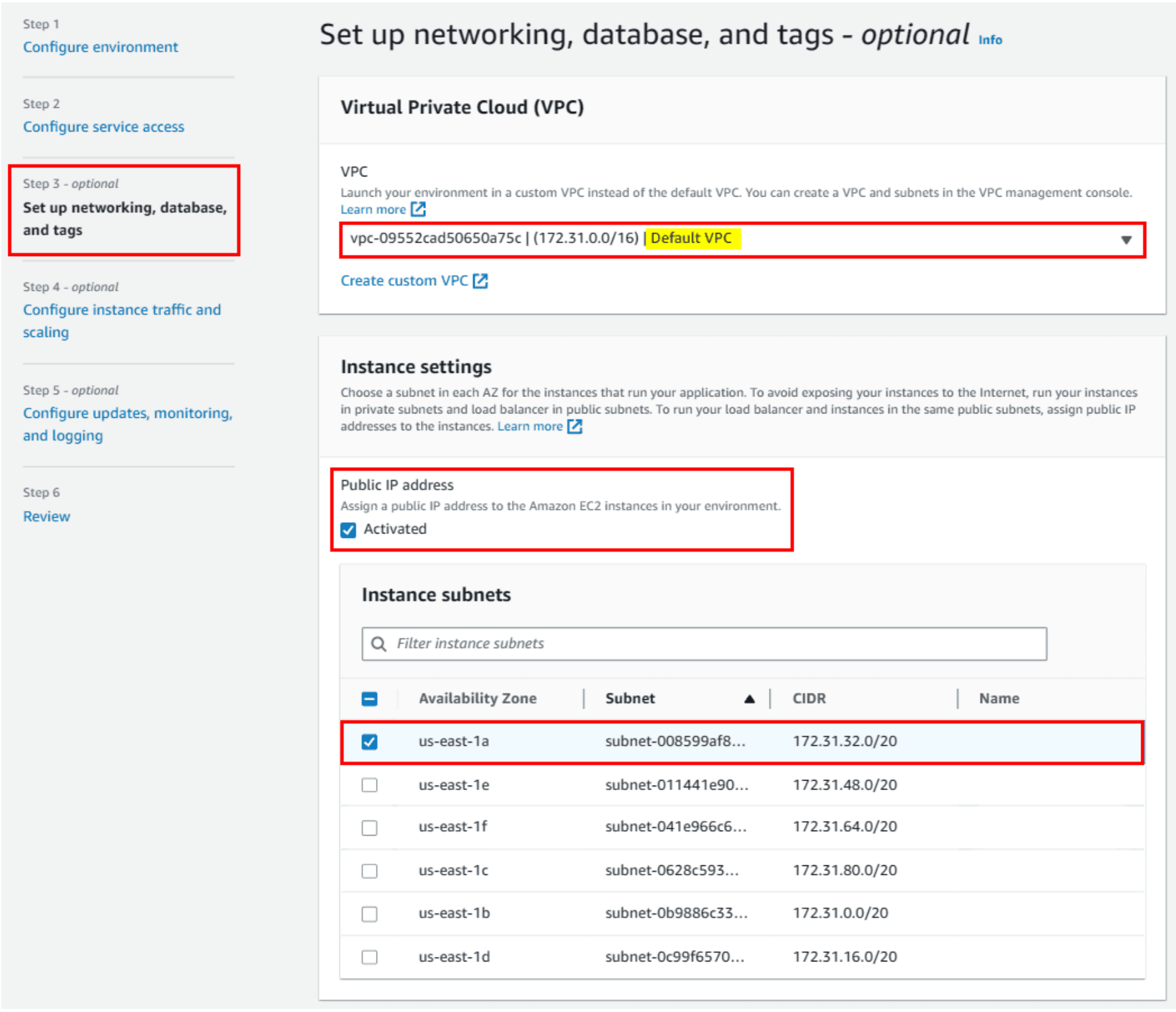
6. Under source code origin click on choose file then select the zip file which we downloaded in the previous section
Link: https://k21academy.s3.us-west-2.amazonaws.com/AWS+DevOps+Professional/Activity_Guides/Code+/Deploy+An+Application+In+Beanstalk+Using+Docker+Code/v1.zip
7. Then choose **Single instance** under Presets and Click on **Next**



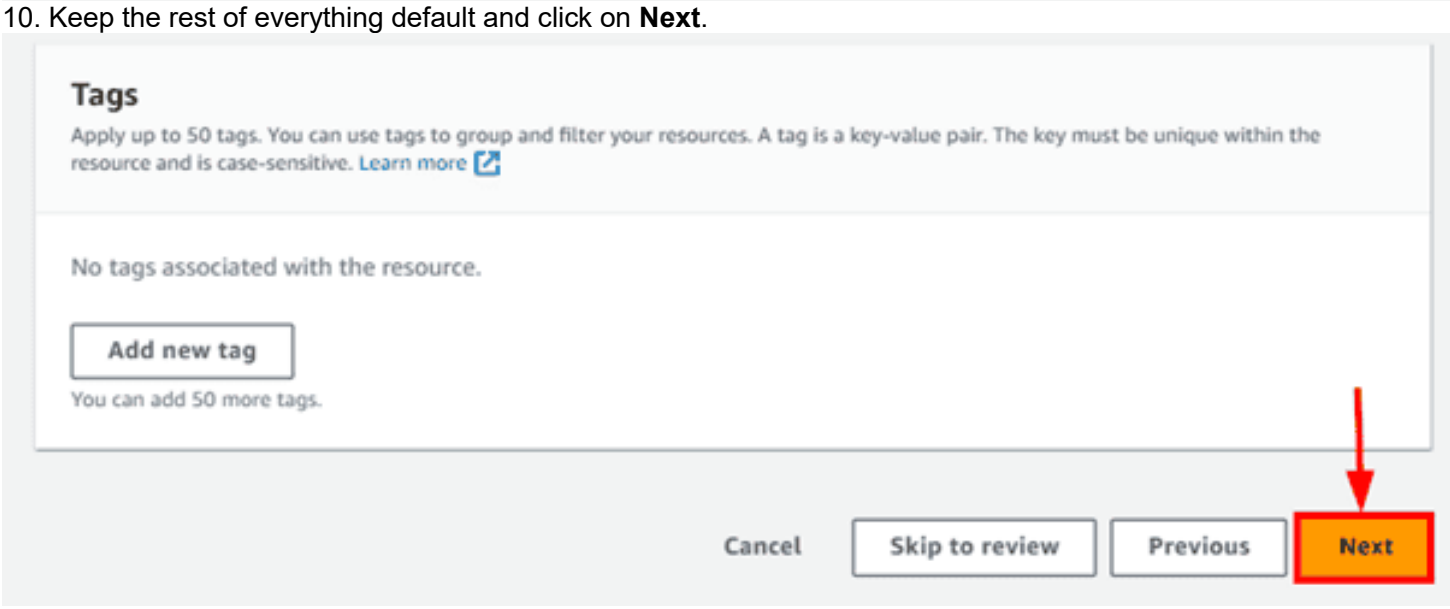
8. Under **Configure service access**, choose to **Create and Use new service role** and click on
Note: We need to configure service access and create a role for Elastic Beanstalk so that it can assume this role when calling services on our behalf. This allows Elastic Beanstalk to automatically deploy new versions of our application



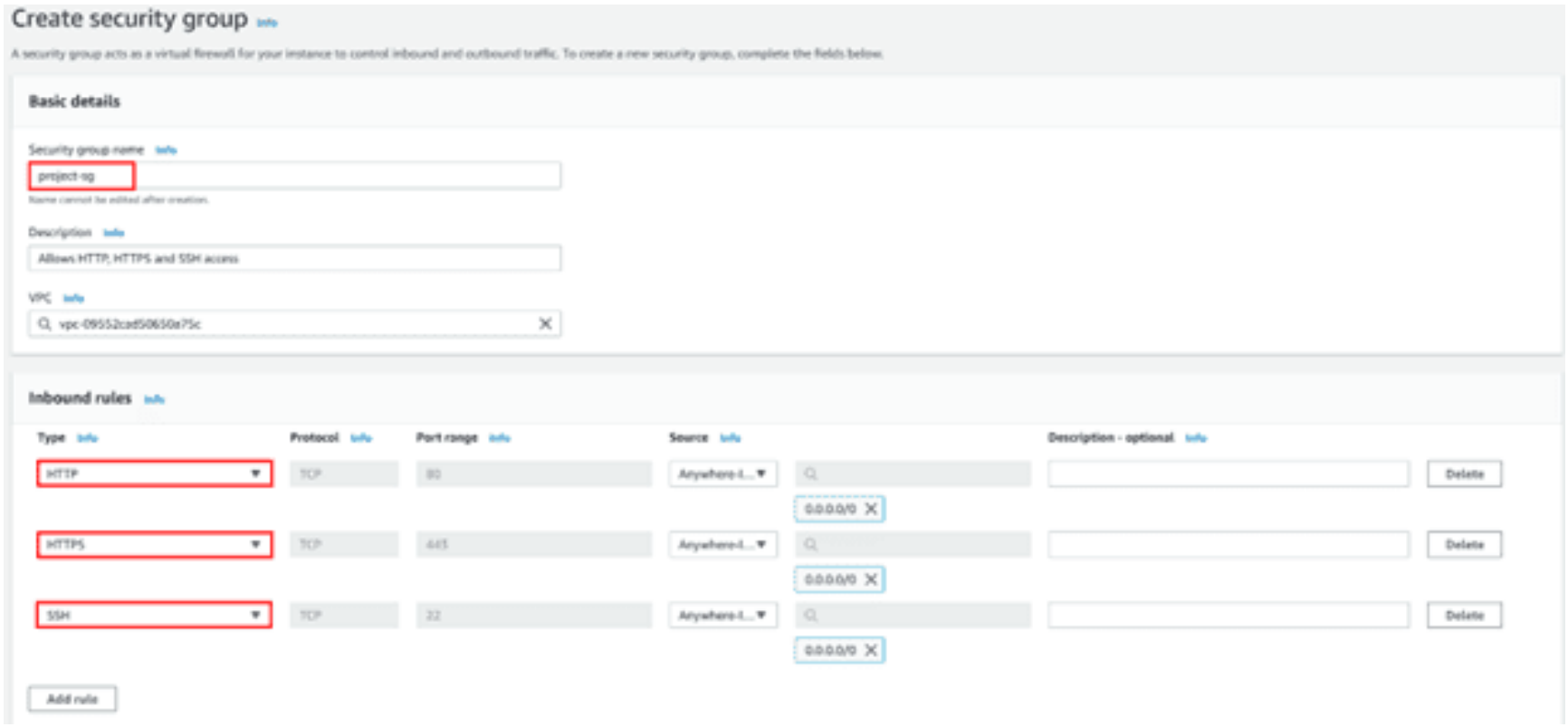
9. In **the networking, database, and tags section,** choose **Default VPC, Enable** public IP address, and select availability zone as **us-east-1a.**



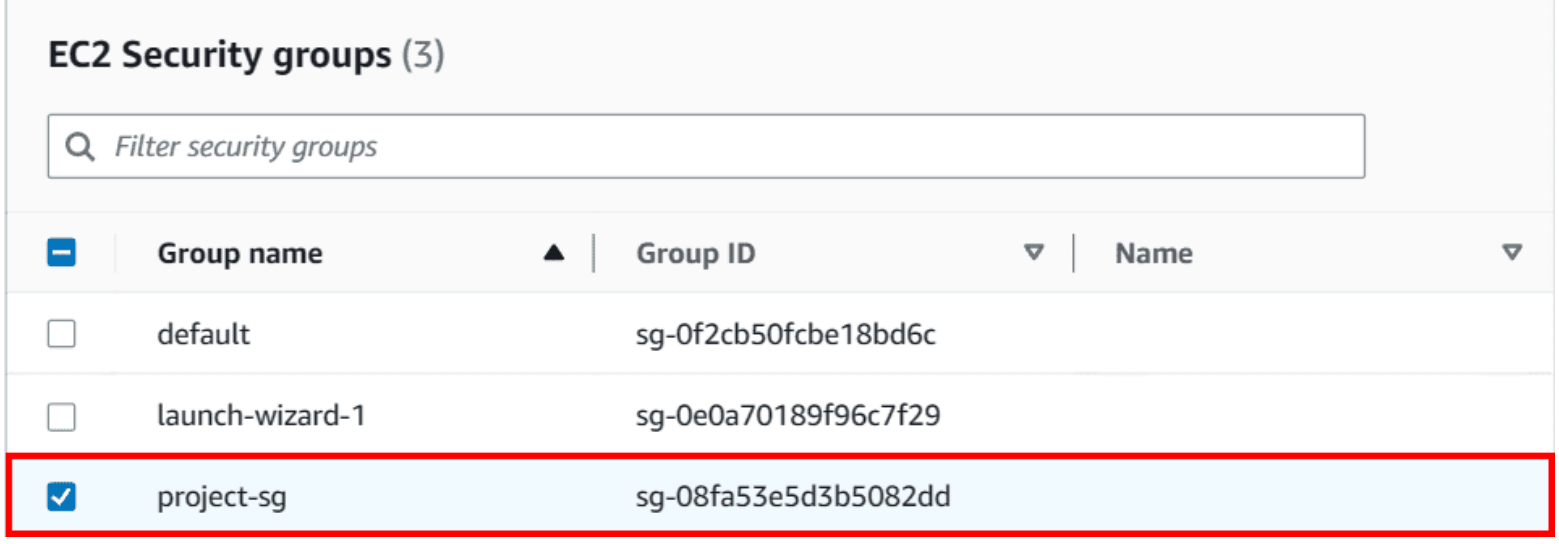
10. Keep the rest of everything default and click on **Next**.



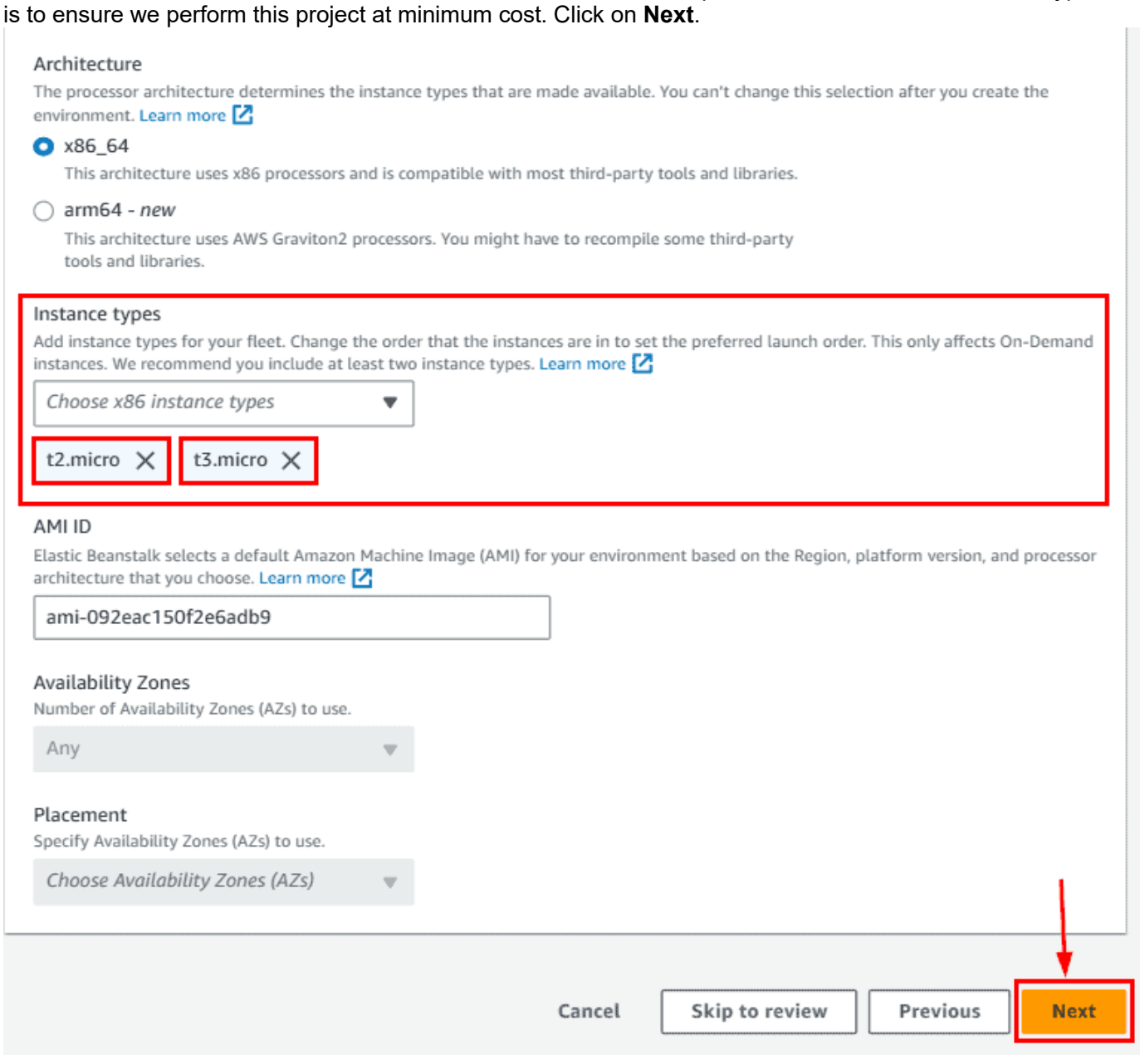
11. Create a security group by going to **EC2 Dashboard > Security Groups > Create security group.** Create a security group that allows **HTTP, HTTPS,** and **SSH** traffic from **ANYWHERE.**



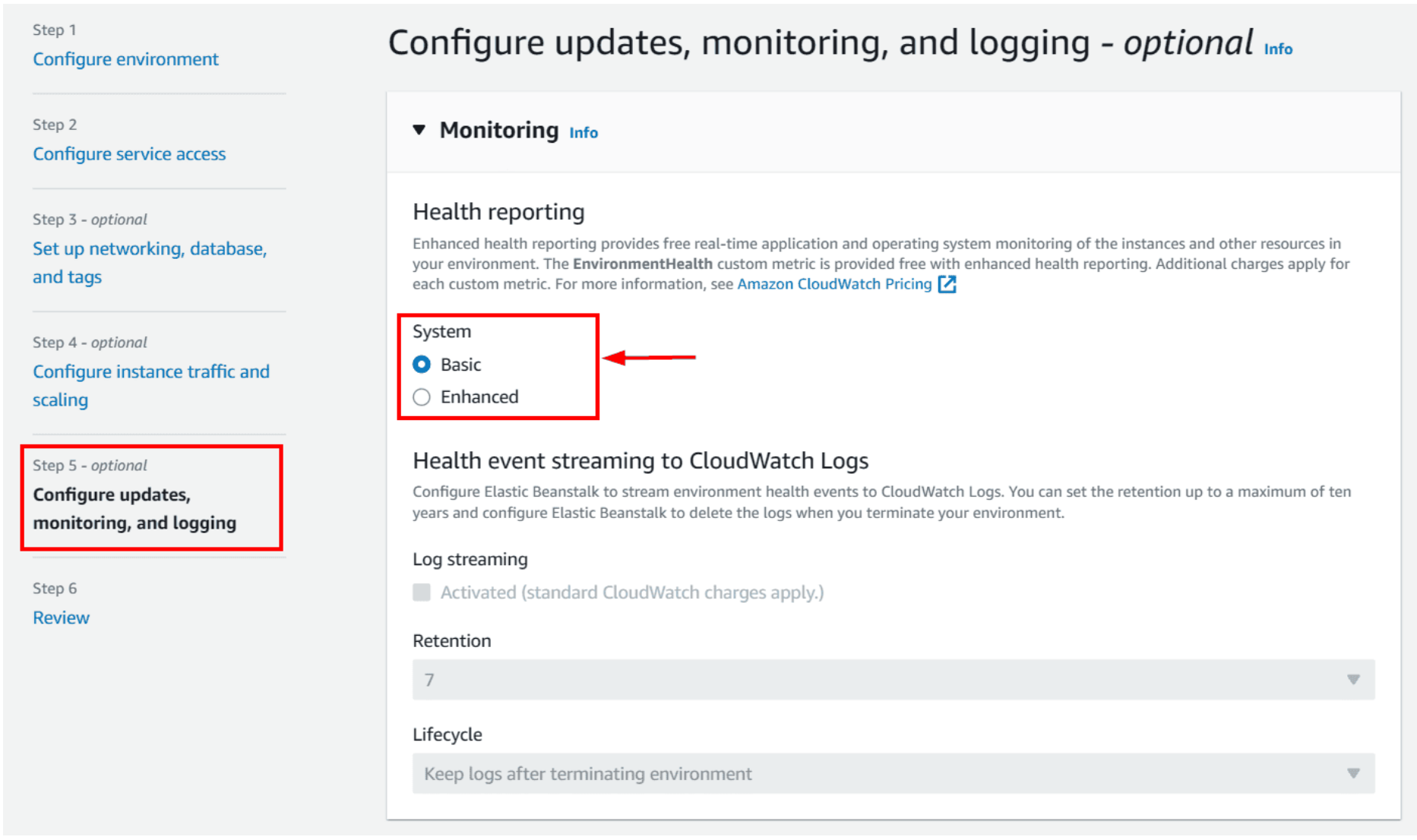
12. Under **Configure instance traffic and scaling,** choose a security group that we just created.



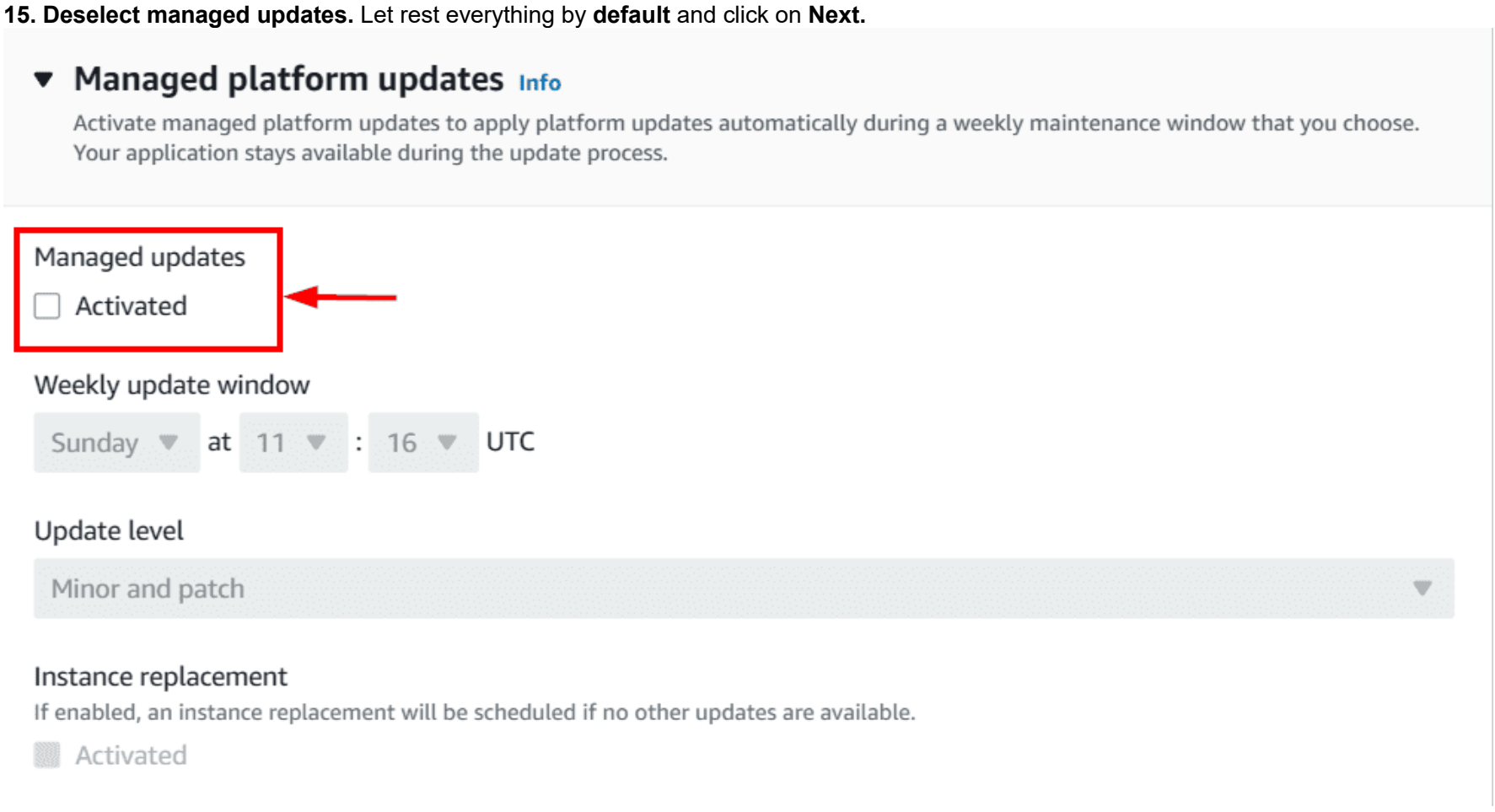
13. Now, scroll down and select **t2.micro** and **t3.micro** from the dropdown menu as desired instance types. Remove any existing instance types. This is to ensure we perform this project at minimum cost. Click on **Next**.



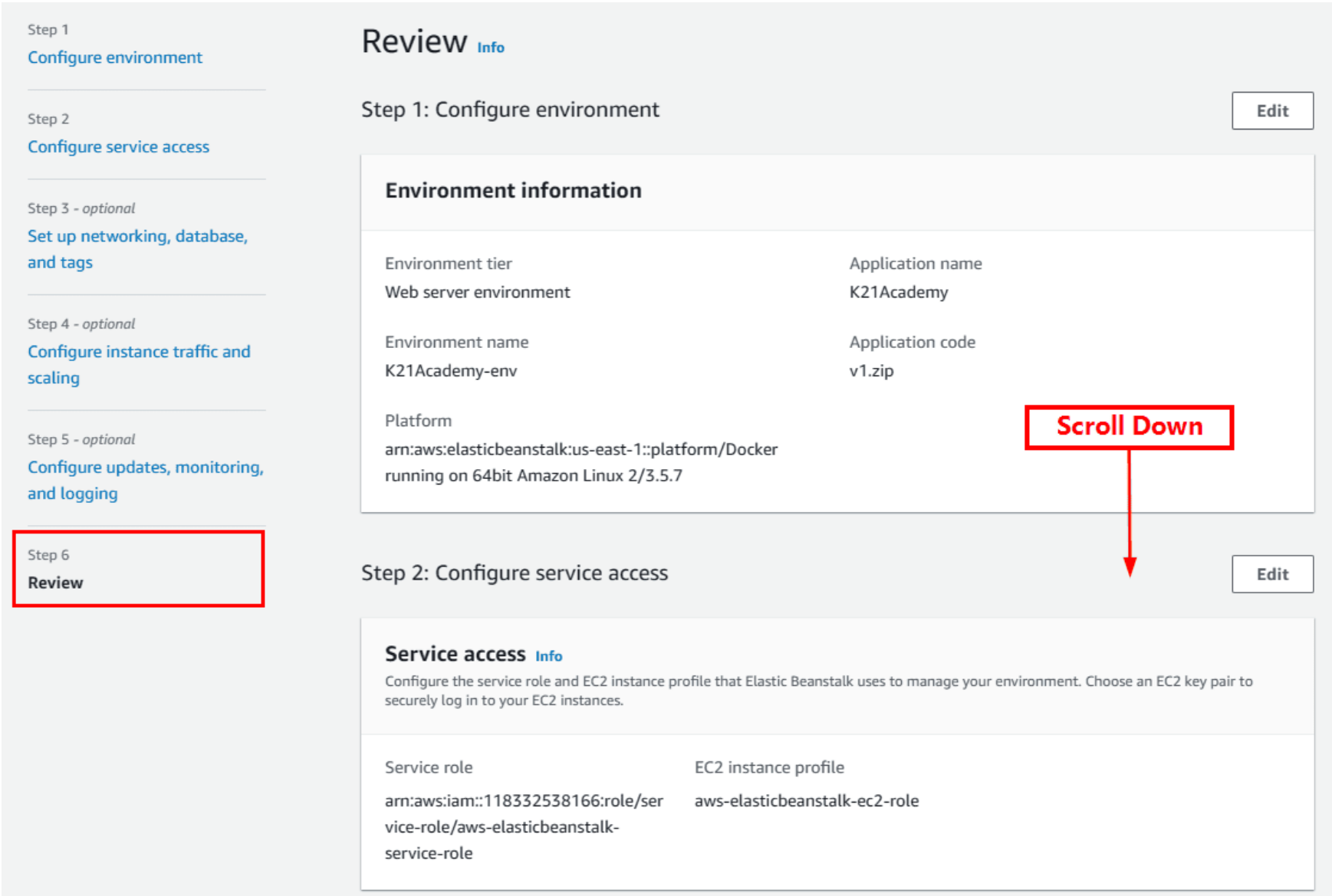
14. Under Configure updates, monitoring, and logging, choose Basic Health Reporting



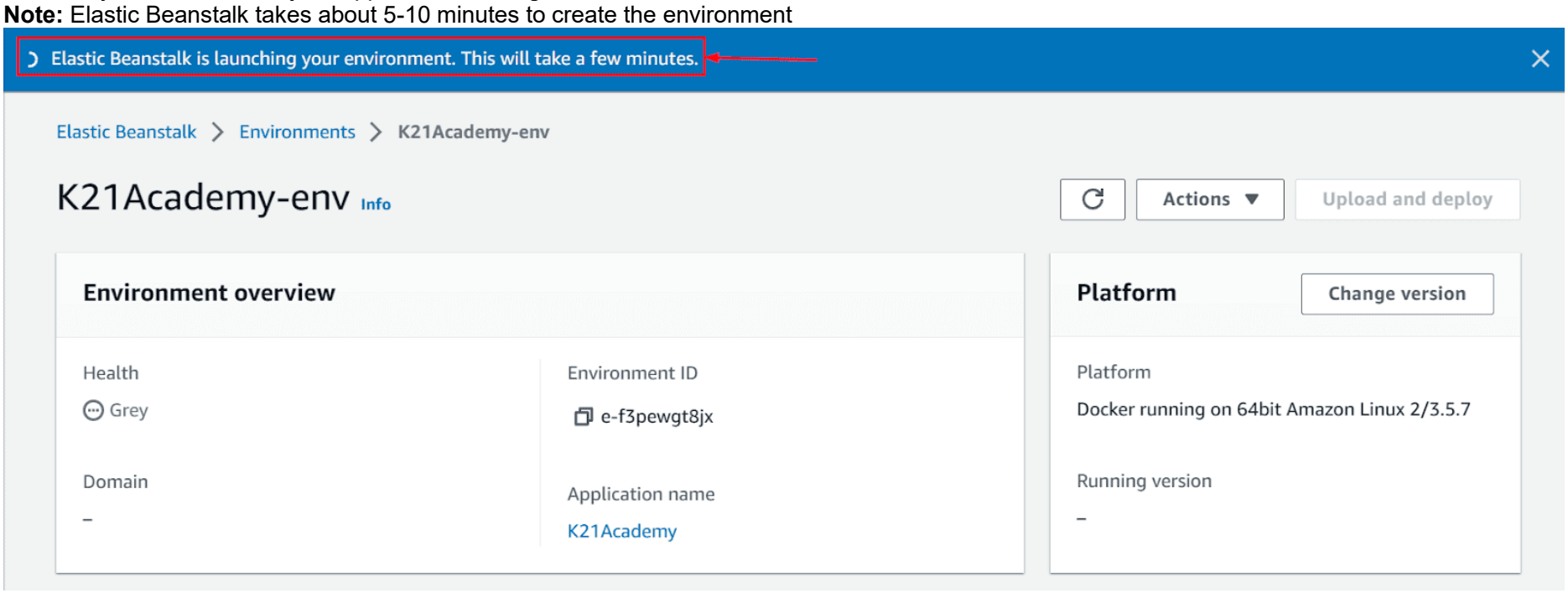
15. Deselect managed updates. Let rest everything by **default** and click on **Next.**



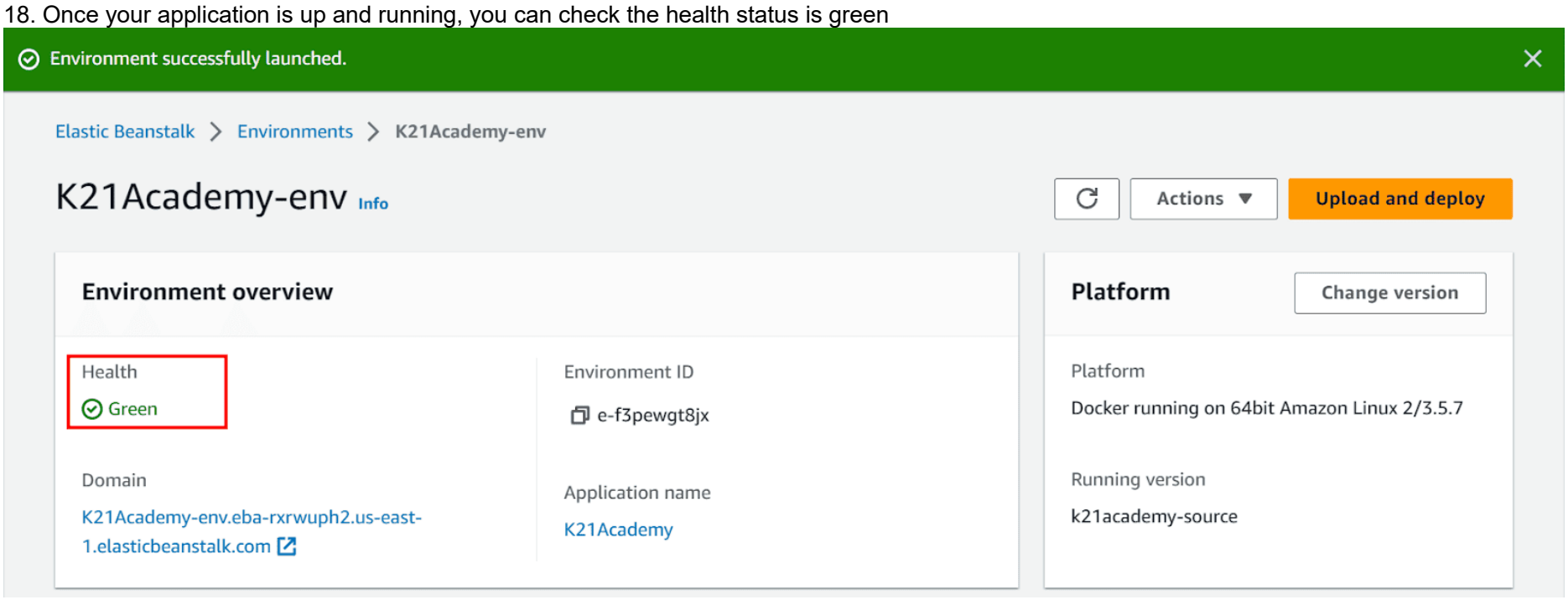
16. Review everything and click on submit



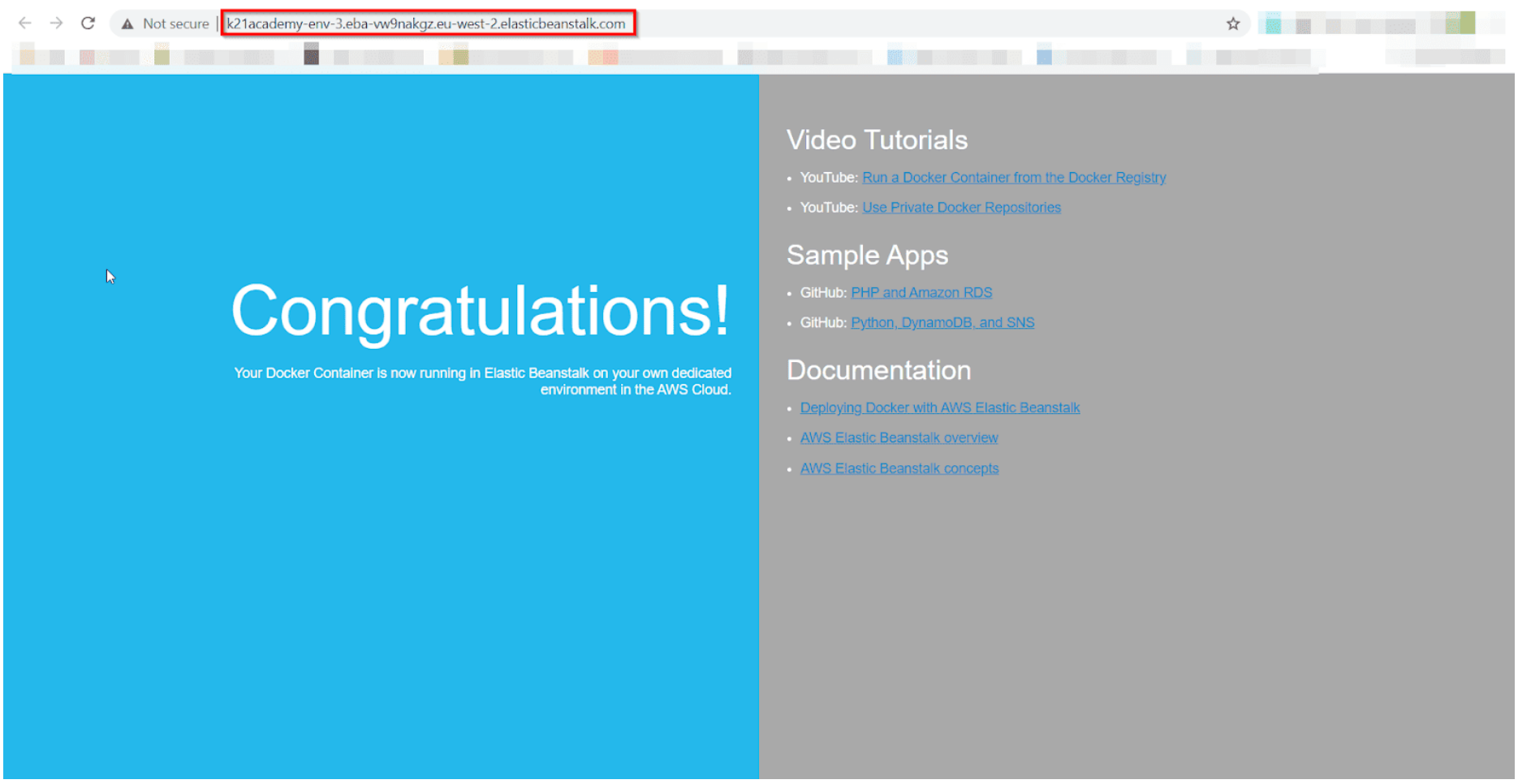
17. Here you shall see that your application is being created.
Note: Elastic Beanstalk takes about 5-10 minutes to create the environment



18. Once your application is up and running, you can check the health status is green



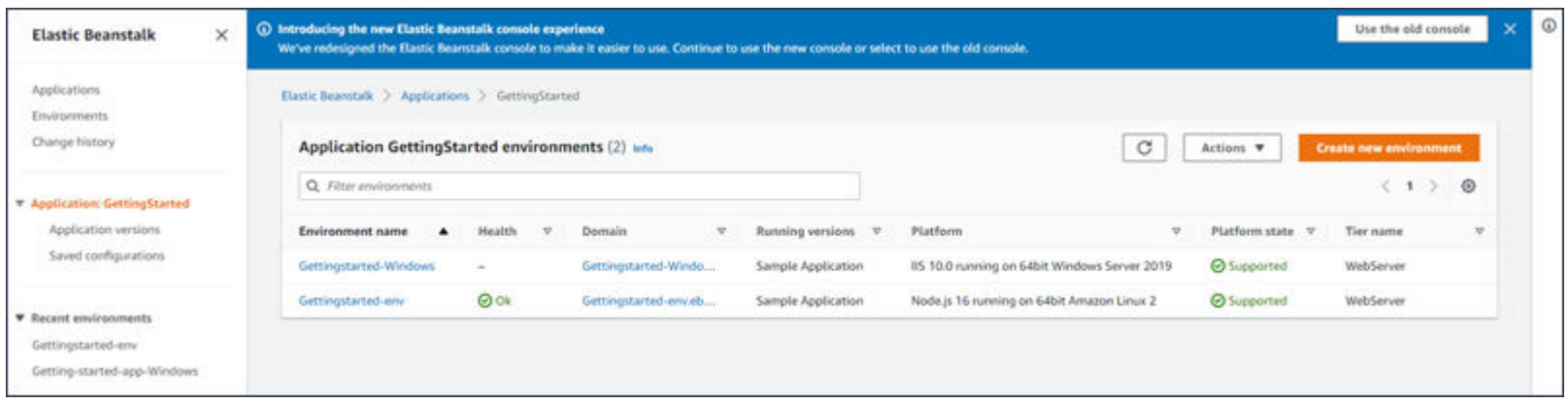
19. Now, click the URL it will redirect to the sample website which is served from your docker container



Launch Environment

To launch an environment with a sample application (console)

1. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.
2. In the navigation pane, choose **Applications**, and then choose an existing application's name in the list or create one.
3. On the application overview page, choose **Create new environment**.



This launches the **Create environment** wizard. The wizard provides a set of steps for you to create a new environment.

Step 1
Configure environment

Step 2
Configure service access

Step 3 - *optional*
Configure instance traffic and scaling

Step 4 - *optional*
Set up networking, database, and tags

Step 5 - *optional*
Configure updates, monitoring, and logging

Step 6
Review

Configure environment Info

Environment tier Info

Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

- (•) Web server environment
  Run a website, web application, or web API that serves HTTP requests. Learn more
- ( ) Worker environment
  Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. Learn more

Application information Info

Application name

GettingStarted

Maximum length of 100 characters.

▶ Application tags (optional)

Environment information Info

Choose the name, subdomain and description for your environment. These cannot be changed later.

Environment name

GettingStarted-env

Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.

Domain name

Leave blank for autogenerated value .us-east-1.elasticbeanstalk.com Check availability

Environment description

Platform Info

Platform type

- (•) Managed platform
  Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
- ( ) Custom platform
  Platforms created and owned by you. This option is unavailable if you have no platforms.

Platform

Choose a platform

Platform branch

Choose a platform branch

Platform version

Choose a platform version

Application code Info

- (•) Sample application
- ( ) Existing version
  Application versions that you have uploaded.
  Sample Application
- ( ) Upload your code
  Upload a source bundle from your computer or copy one from Amazon S3.

Presets Info

Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.

Configuration presets

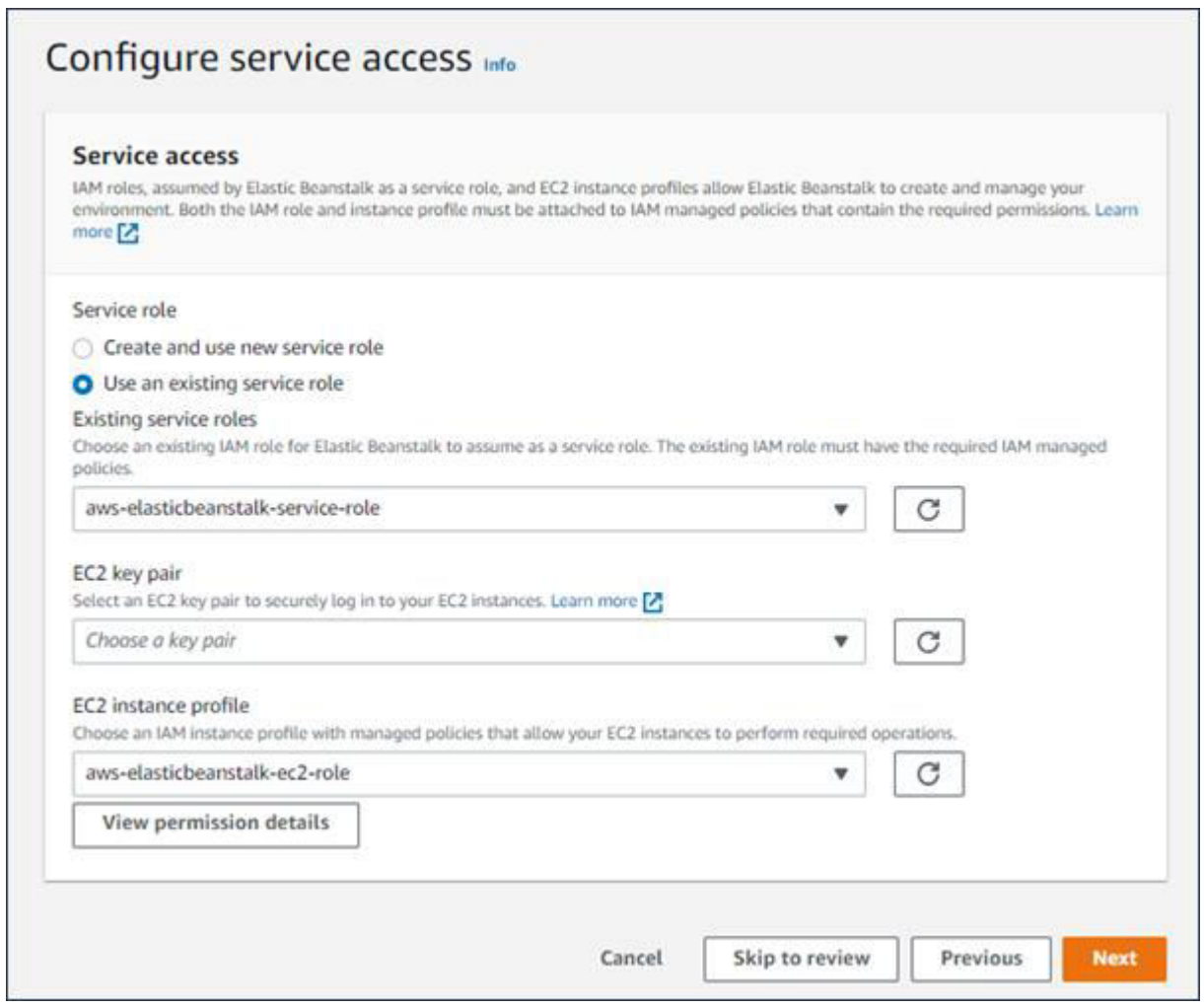
- (•) Single instance (free tier eligible)
- ( ) Single instance (using spot instance)
- ( ) High availability
- ( ) High availability (using spot and on-demand instances)
- ( ) Custom configuration

Cancel Next

4. For environment tier, choose the **Web server environment** or **Worker environment** environment tier. You can't change an environment's tier after creation.
   Note

   The .NET on Windows Server platform doesn't support the worker environment tier.
5. For **Platform**, select the platform and platform branch that match the language your application uses.
   Note

   Elastic Beanstalk supports multiple versions for most of the platforms that are listed. By default, the console selects the recommended version for the platform and platform branch you choose. If your application requires a different version, you can select it here. For information about supported platform versions, see Elastic Beanstalk supported platforms.
6. For **Application code**, choose **Sample application**.
7. For **Configuration presets**, choose **Single instance**.
8. Choose **Next**.
9. The **Configure service access** page displays.



10. Choose **Use an existing service role** for **Service Role**.
11. Next, we'll focus on the **EC2 instance profile** dropdown list. The values displayed in this dropdown list may vary, depending on whether you account has previously created a new environment.

    Choose one of the following, based on the values displayed in your list.

    - If `aws-elasticbeanstalk-ec2-role` displays in the dropdown list, select it from the **EC2 instance profile** dropdown list.
    - If another value displays in the list, and it's the default EC2 instance profile intended for your environments, select it from the **EC2 instance profile** dropdown list.
    - If the **EC2 instance profile** dropdown list doesn't list any values to choose from, expand the procedure that follows, *Create IAM Role for EC2 instance profile*.

      Complete the steps in *Create IAM Role for EC2 instance profile* to create an IAM Role that you can subsequently select for the **EC2 instance profile**. Then return back to this step.

      Now that you've created an IAM Role, and refreshed the list, it displays as a choice in the dropdown list. Select the IAM Role you just created from the **EC2 instance profile** dropdown list.
12. Choose **Skip to Review** on the **Configure service access** page.

    This will select the default values for this step and skip the optional steps.
13. The **Review** page displays a summary of all your choices.

    To further customize your environment, choose **Edit** next to the step that includes any items you want to configure. You can set the following options only during environment creation:
    - Environment name
    - Domain name
    - Platform version

- Processor
- VPC
- Tier

You can change the following settings after environment creation, but they require new instances or other resources to be provisioned and can take a long time to apply:

- Instance type, root volume, key pair, and AWS Identity and Access Management (IAM) role
- Internal Amazon RDS database
- Load balancer

For details on all available settings, see The create new environment wizard.

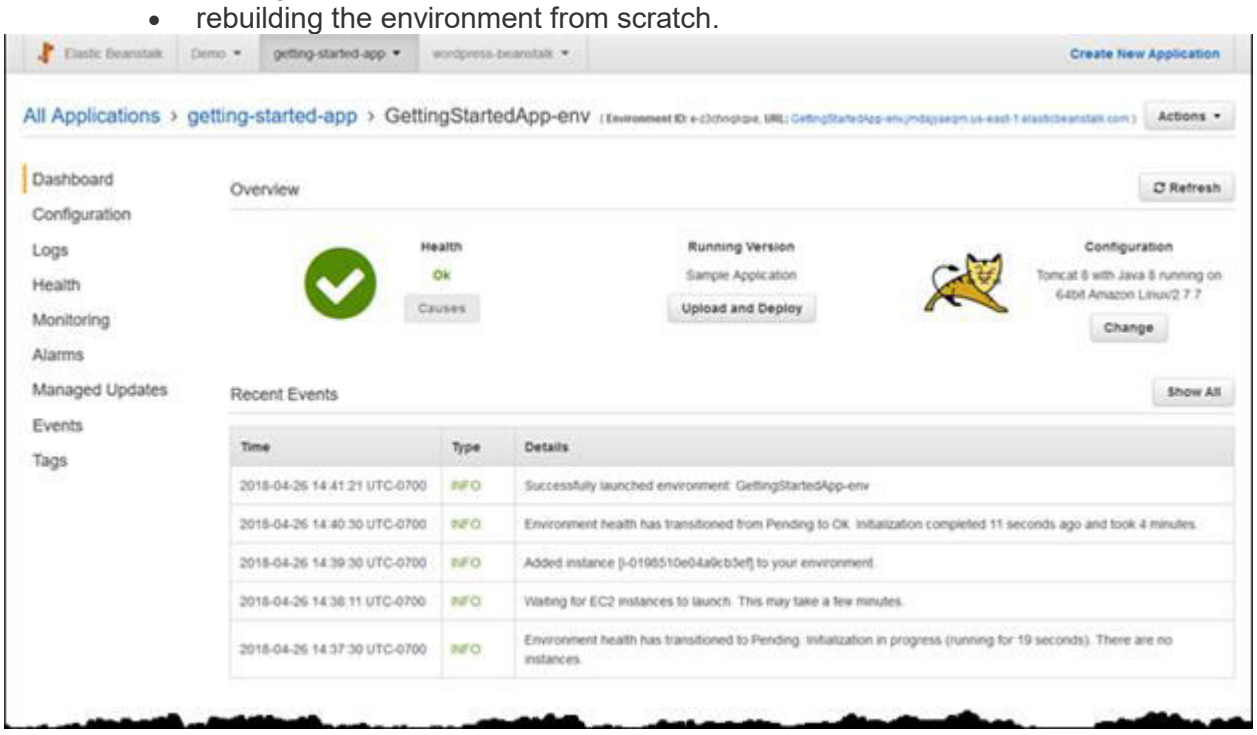
14. Choose **Submit** at the bottom of the page to initialize the creation of your new environment.

Manage Environment

- Elastic Beanstalk manage separate environments for
  - Development
  - Testing
  - production use
- Deploy any version of application to any environment.
- Environments can be long-running or temporary.
- If terminating an environment, you can save its configuration to recreate later.
- During application development, it is deployed often to different environments for different purposes.
- Elastic Beanstalk easily configure how deployments are performed.
- Either deploy to all of the instances simultaneously or split deployment into batches with rolling deployments.
- Select platform version when launching an environment
- Can also split application into multiple components, each running in a separate environment.

Elastic Beanstalk Environment Management Console:

- provides a management page for each Elastic Beanstalk environments
- manage environment's configuration
- perform common actions like
  - restarting the web servers running
  - cloning the environment
  - rebuilding the environment from scratch.



Environment Dashboard

- main view of the environment management console is a dashboard
- To view
  - choose Dashboard on the navigation pane.
- environment management dashboard shows
  - environment's health
  - application version
  - information about the in-use platform
  - list of recent events generated by the environment.

Various sections of Dashboard are:

Health

- Shows the overall health of the environment.
- environment status is shown with a Causes button

- choose to view more information about current status.

Recent Events

- Shows most recent events emitted by environment.
- List is updated in real time when environment is being updated.

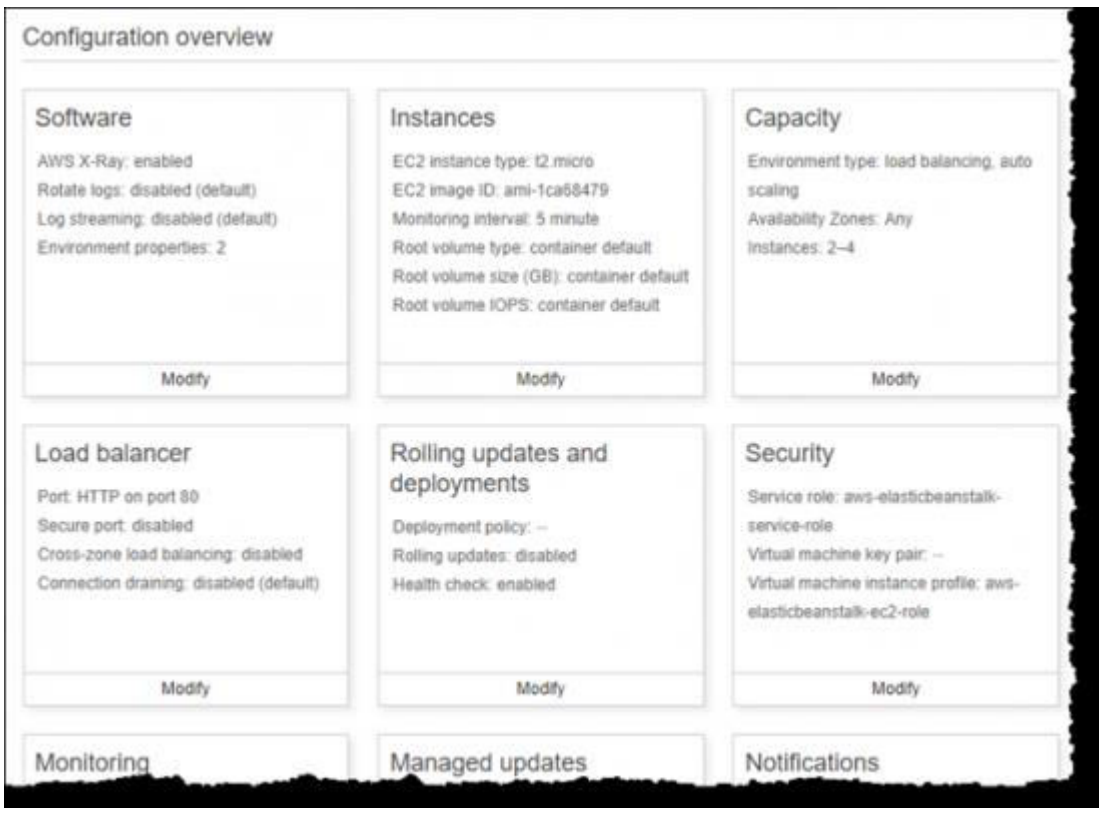
Environment Management Actions

- Actions menu perform common operations on environment.
- This menu is on right side of environment header under the Create New Environment option.
- Actions include
    - Load Configuration – Load a previously saved configuration.
    - Save Configuration – Save current configuration of environment to application.
    - Swap Environment URLs – Swap CNAME of current environment with a new environment.
    - Clone Environment – Launch a new environment with the same configuration as currently running environment.
    - Abort Current Operation – Stop an in-progress environment update.
    - Restart App Servers – Restart the web server running on environment's instances.
    - Rebuild Environment – Terminate all resources in the running environment and build a new environment with the same settings.
    - Terminate Environment – Terminate all resources in the running environment, and remove the environment from the application.
    - Restore Environment – If the environment has been terminated in the last hour, restore it from this page. After an hour, you can restore it from the application overview page.

Configuration

- It shows current configuration of environment and resources,
- It lists
    - EC2 instances
    - load balancer
    - notifications
    - health monitoring settings
- It helps to customize
    - behavior of environment during deployments
    - enable additional features

modify the instance type and other settings chosen during environment creation.



Logs

- lets you retrieve logs from EC2 instances in environment.
- When logs are requested,
    - Elastic Beanstalk sends a command to the instances
    - which then upload logs to Elastic Beanstalk storage bucket in S3.
    - When requesting logs on this page, Elastic Beanstalk automatically deletes them from S3 after 15 minutes.

Environment Types

- Can create one of environment types, as
    - load-balancing
    - autoscaling environment
    - single-instance
- environment type depends on application to deploy.

Load-balancing, Autoscaling Environment

- uses ELB and EC2 Auto Scaling
- EC2 Auto Scaling starts additional instances
- If need to running in multiple Availability Zones, use a load-balancing, autoscaling environment. Can switch environment type later.

Single-Instance Environment

- It contains one EC2 instance with an Elastic IP address.
- It doesn't have a load balancer
- It uses EC2 Auto Scaling service but, minimum, maximum and desired capacity are set to 1. Hence, , new instances are not started if increased load.
- Use it, if application to have low traffic or doing remote development.

To change an environment's type

- Open the Elastic Beanstalk console.
- Navigate to the management page for your environment.
- Choose Configuration.
- In the Capacity category, choose Modify.

Recommended values

The recommended values set in API, are

Elastic Beanstalk console

- Namespace: aws:autoscaling:launchconfiguration

Option Names: IamInstanceProfile, EC2KeyName, InstanceType

- Namespace: aws:autoscaling:updatepolicy:rollingupdate

Option Names: RollingUpdateType and RollingUpdateEnabled

- Namespace: aws:elasticbeanstalk:application

Option Name: Application Healthcheck URL

- Namespace: aws:elasticbeanstalk:command

Option Name: DeploymentPolicy, BatchSize and BatchSizeType

- Namespace: aws:elasticbeanstalk:environment

Option Name: ServiceRole

- Namespace: aws:elasticbeanstalk:healthreporting:system

Option Name: SystemType and HealthCheckSuccessThreshold

- Namespace: aws:elasticbeanstalk:sns:topics

Option Name: Notification Endpoint

- Namespace: aws:elasticbeanstalk:sqsd

Option Name: HttpConnections

- Namespace: aws:elb:loadbalancer

Option Name: CrossZone

- Namespace: aws:elb:policies

Option Names: ConnectionDrainingTimeout and ConnectionDrainingEnabled

EB CLI

- Namespace: aws:autoscaling:launchconfiguration

Option Names: IamInstanceProfile, InstanceType

- Namespace: aws:autoscaling:updatepolicy:rollingupdate

Option Names: RollingUpdateType and RollingUpdateEnabled

- Namespace: aws:elasticbeanstalk:command

Option Name: BatchSize and BatchSizeType

- Namespace: aws:elasticbeanstalk:environment

Option Name: ServiceRole

- Namespace: aws:elasticbeanstalk:healthreporting:system

Option Name: SystemType

- Namespace: aws:elb:loadbalancer

Option Name: CrossZone

- Namespace: aws:elb:policies
- Option Names: ConnectionDrainingEnabled

Creating application source bundle

Elastic Beanstalk provides a way to set up and deploy applications in the AWS cloud in an easy manner.

We have to focus on the application code and upload it to the Beanstalk environment which automatically scales the infra depending on the incoming traffic.

There are two ways apps can be deployed into the Elastic Beanstalk environment.

● Using AWS Console

● Using EB CLI (a command-line utility)

Supported Programming Languages

The following platforms are supported by Elastic beanstalk.

● PHP

● Docker

● Python

● Java

● Node.js

● Ruby

● GlassFish

● Go

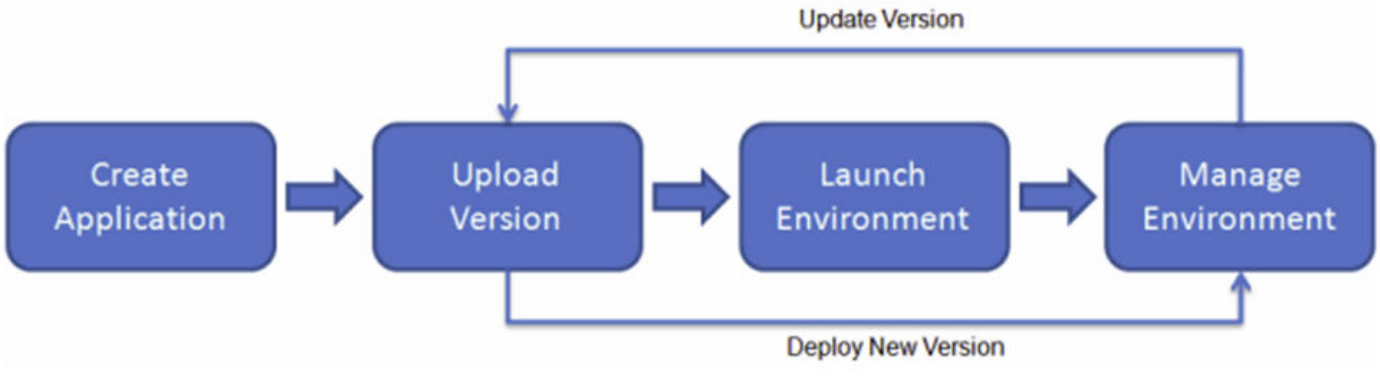
● Tomcat

● .NET on Windows and Linux

Choose the platform using which the application is built.

How it works

The below diagram represents the workflow of deploying applications into Elastic beanstalk



● Application setup with the preferred platform along with the source bundle will be created
● The Environment such as Instance, Load Balancing, Auto Scaling will be created to deploy the applications
● Versioning will be maintained for redeployment and rollback

Deploying Applications

Let’s go ahead and deploy a sample Node.js application into Elastic Beanstalk,

Sign in to ***AWS Console***,
Under Computer, Select Elastic Beanstalk
Click Create Application

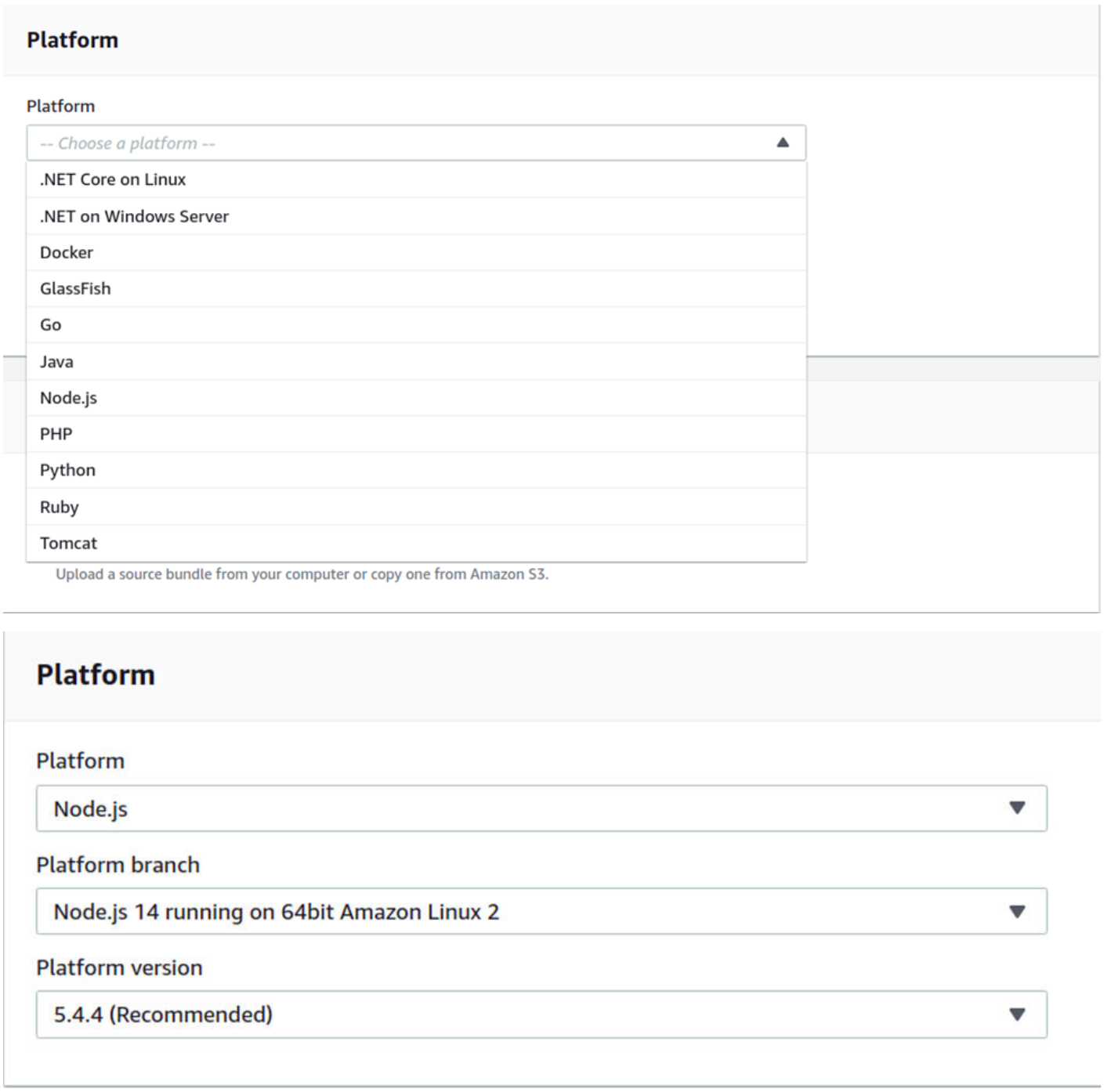
Creating a Web Application

In this step, We will set up an application by choosing the preferred application platform.

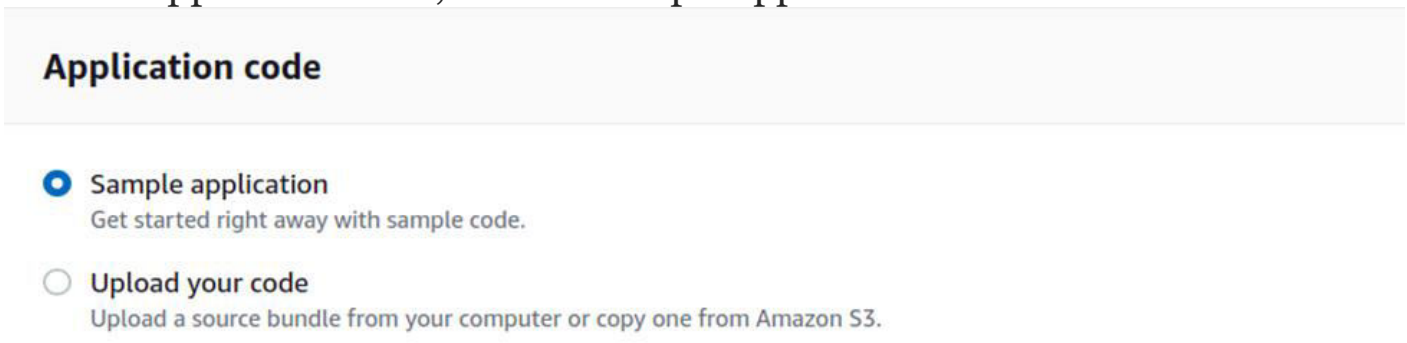
Enter the application name

For Platform, Select Node.js which will automatically select the Platform branch and the Platform version

You May Also Like: AWS vs Azure vs Google Cloud



For the Application code, Choose Sample application



Configuring Environment

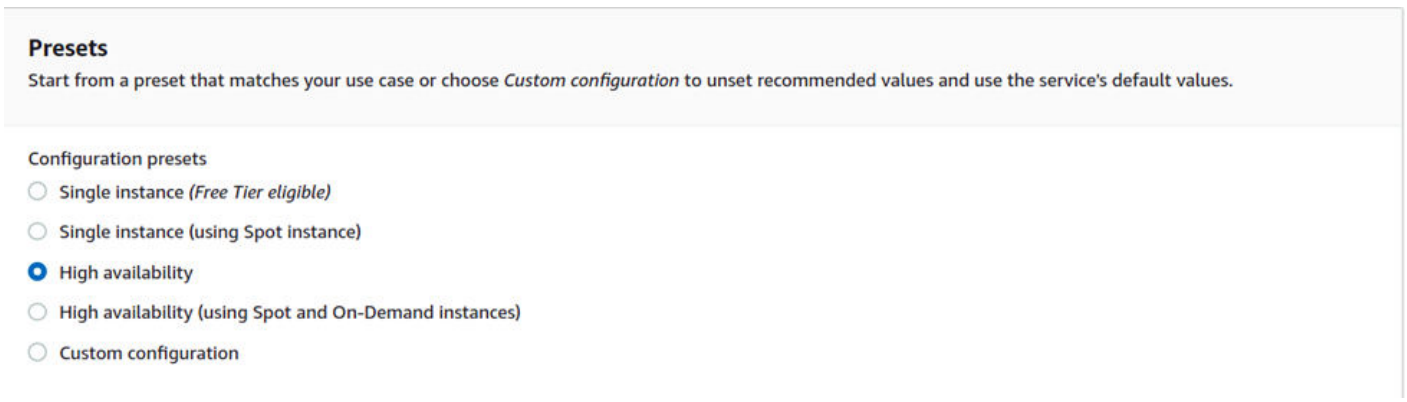
We must configure the environment where the application code will be deployed.

Environmental setups such as Instance, Security Groups, Load Balancing, VPC Network, and Database will be configured.

Click **Configure more options**

Now it’s time to configure the environment for the Zenesys web application

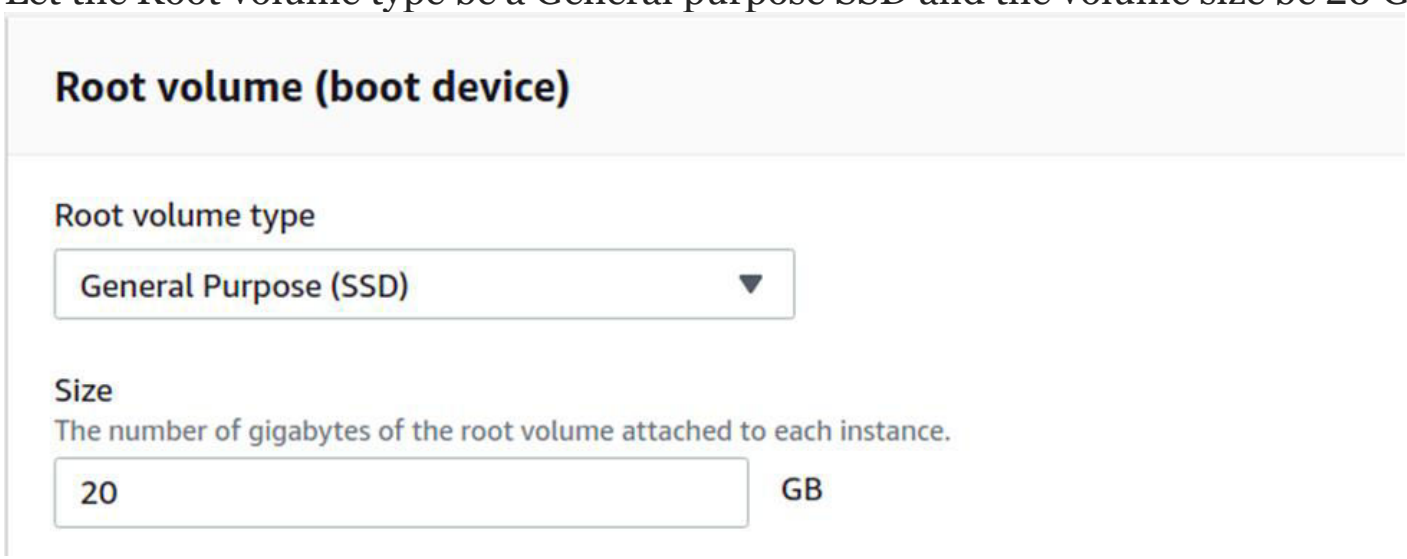
For this guide, We will choose to deploy the applications in the High Availability setup.



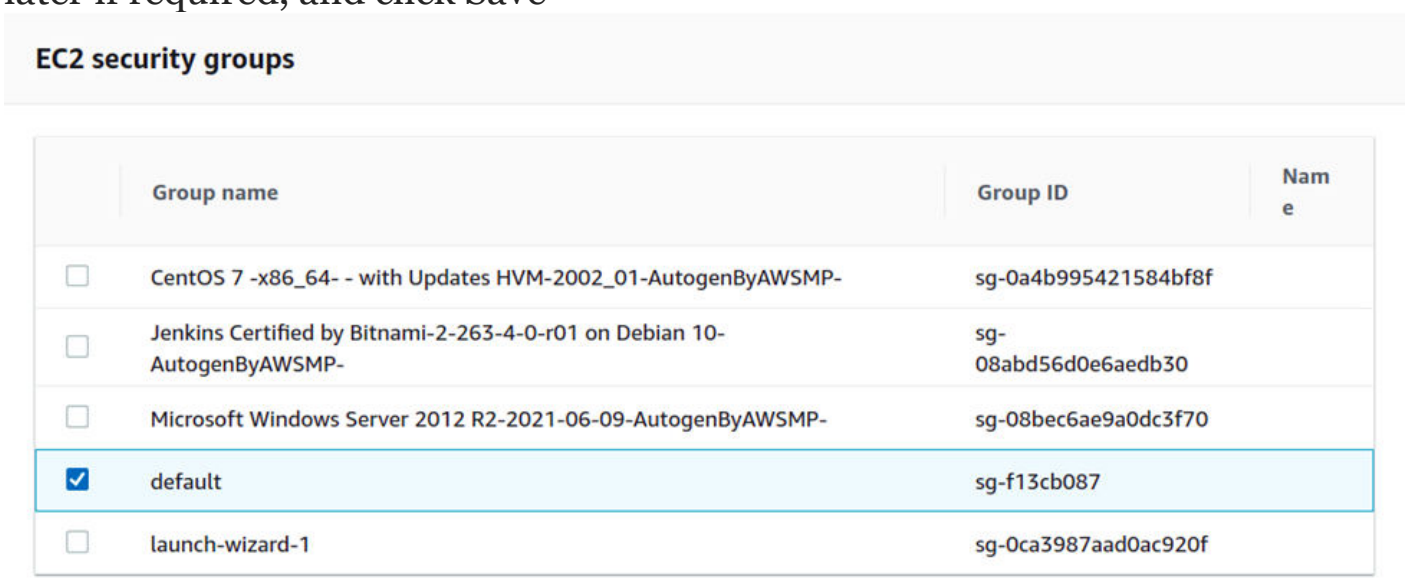
For Instances, Click Edit,

We should configure the EBS storage size and the storage type.

Let the Root volume type be a General purpose SSD and the volume size be 20 GB.



Under EC2 security groups, Choose the preferred security group name, where we can configure the ports later if required, and click Save

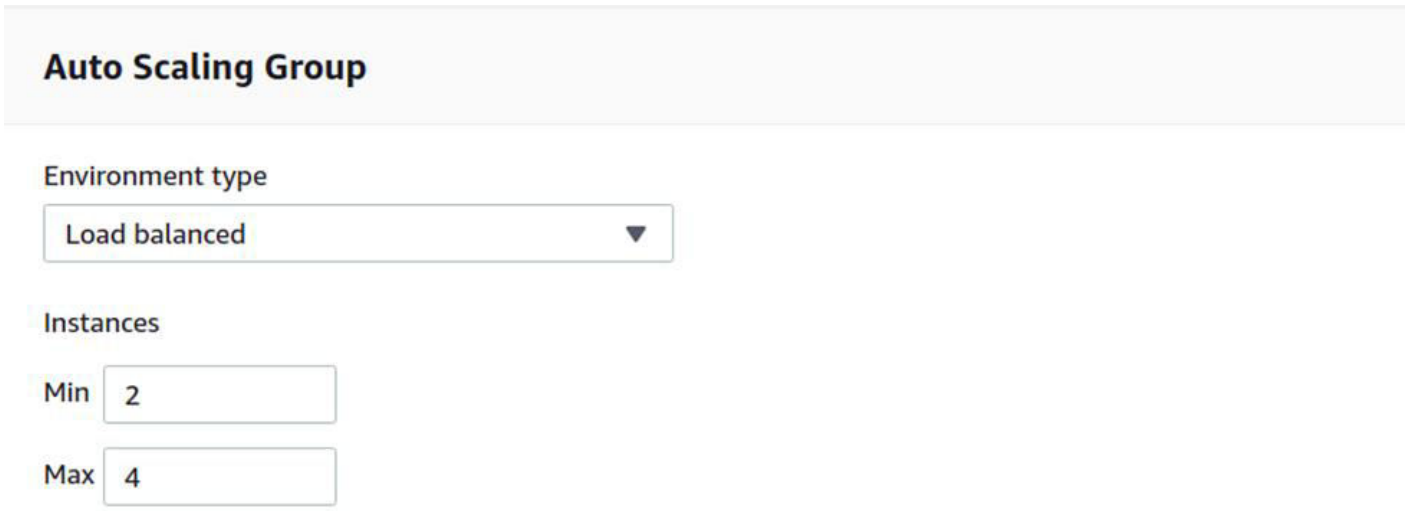


Under Capacity, Click Edit

The Environment Type will be Load balanced.

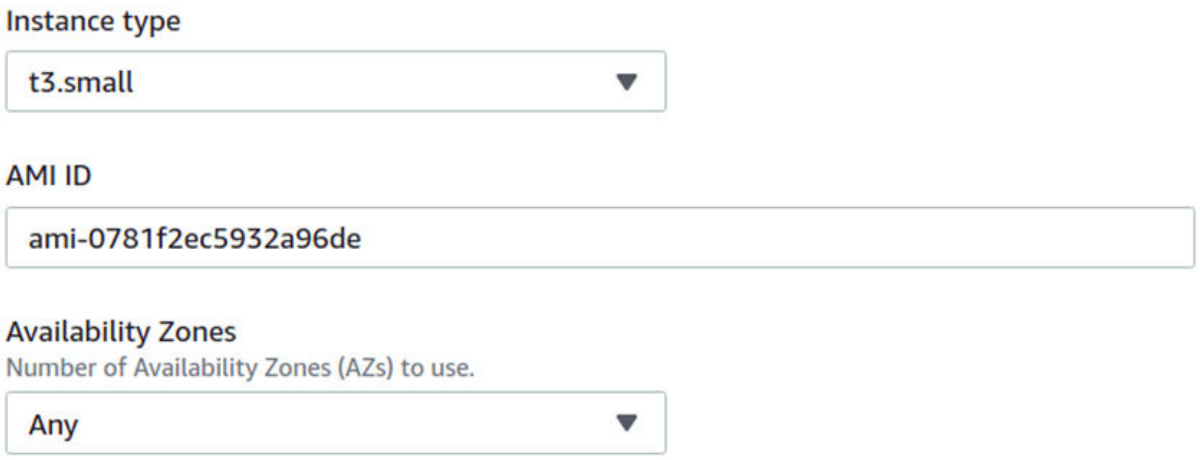
Here we have to specify the minimum and maximum number of instances required during Scale-in and Scale-out.

Let the min be 2 and the Max be 4



Choose the preferred Instance type.

And the AMI ID from which you want to launch the EC2 instances.



And the Availability Zones are Any.

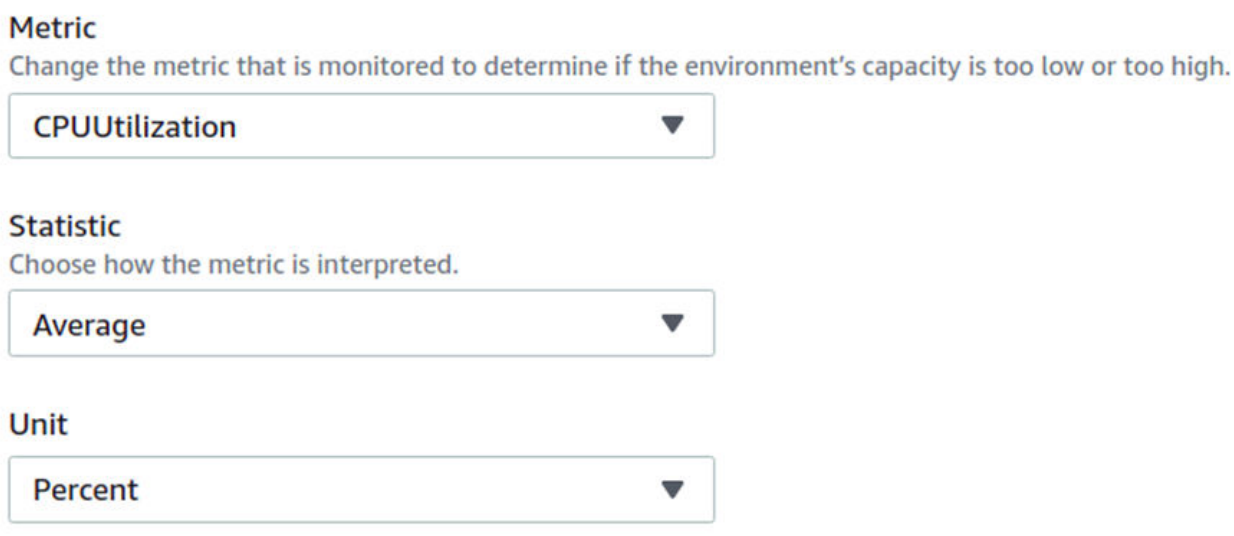
Next, we have to configure, Based on what parameter, the autoscaling on instances should work.

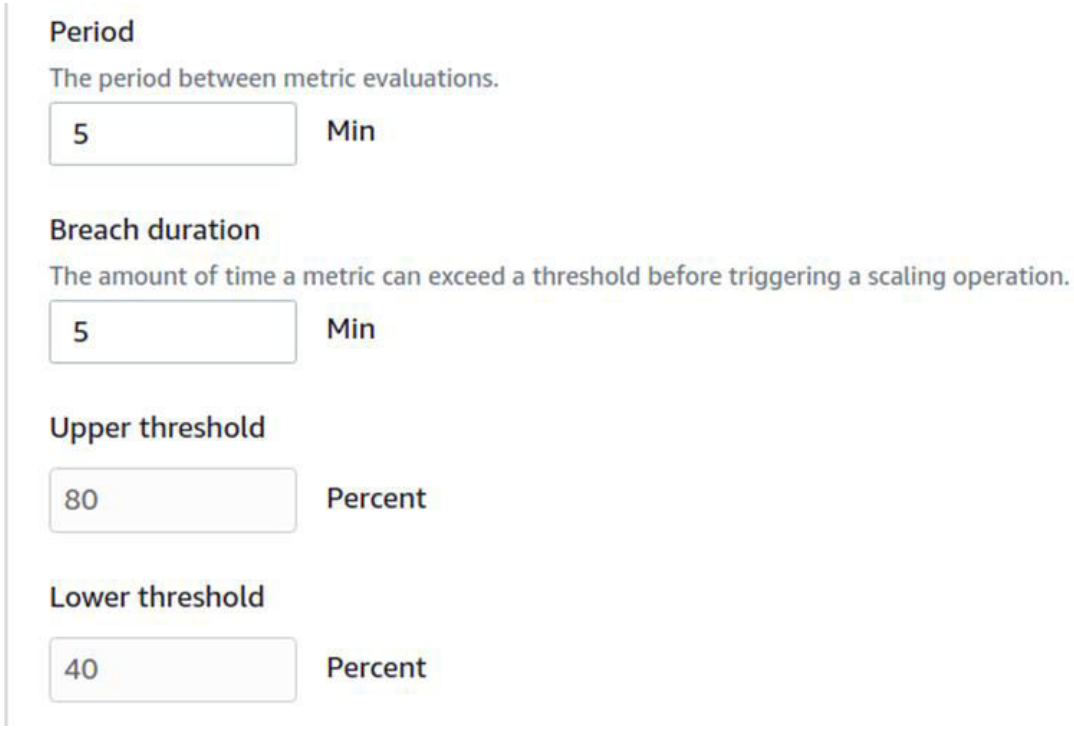
Under Scaling triggers, Let the

Metric be CPU Utilization

Statistics be Average and Unite be Percent.

And set the Upper and lower threshold.

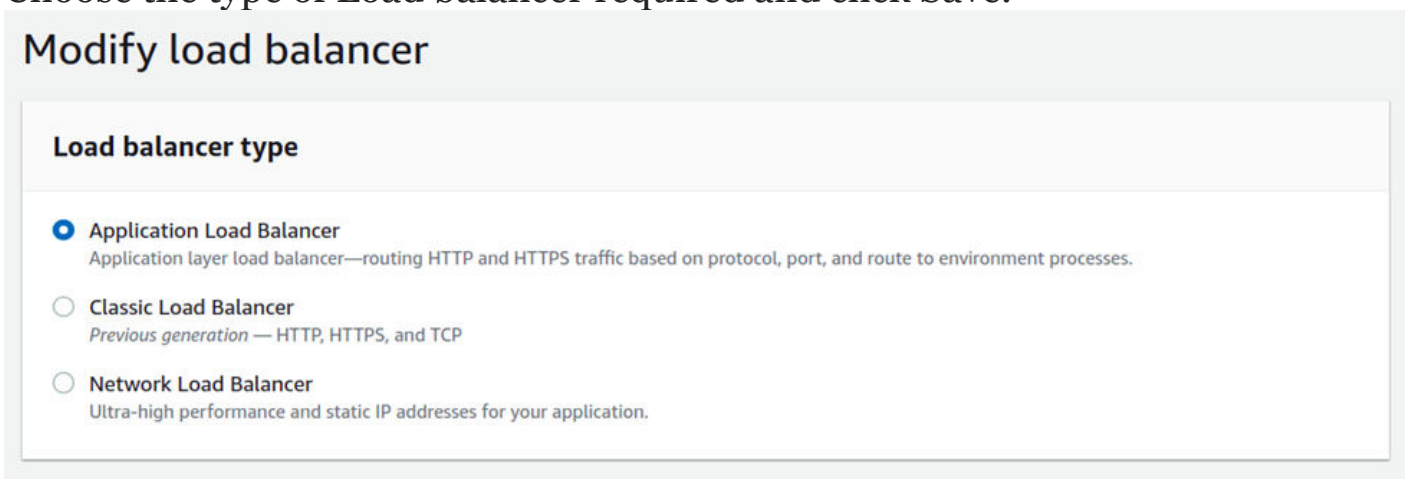




And click Save.

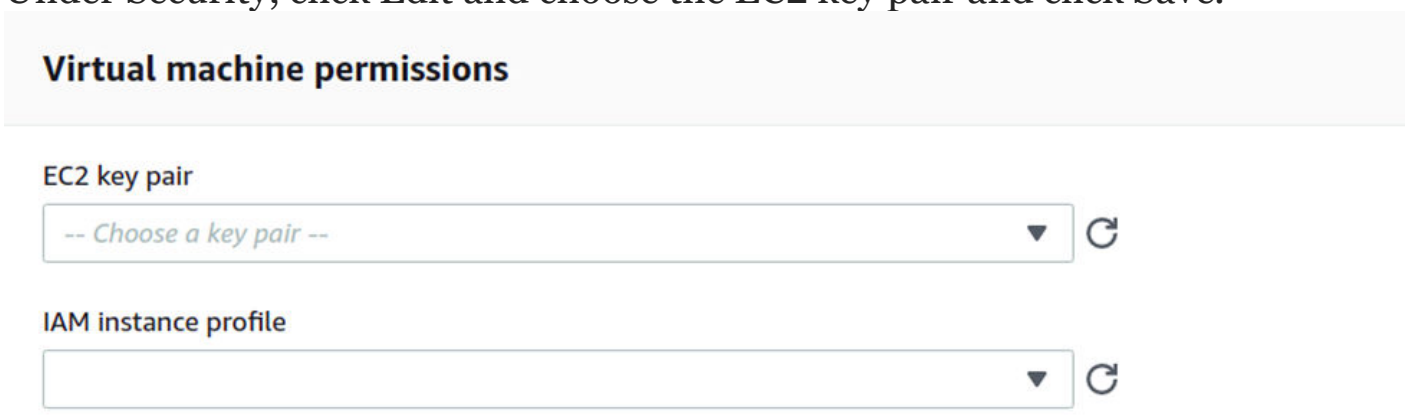
Under Load balancer, Click Edit

Choose the type of Load balancer required and click Save.



To SSH into the EC2 instances launched as part of the Elastic Beanstalk environment, We need to configure the SSH key pair.

Under Security, click Edit and choose the EC2 key pair and click Save.



Under VPC, click Edit

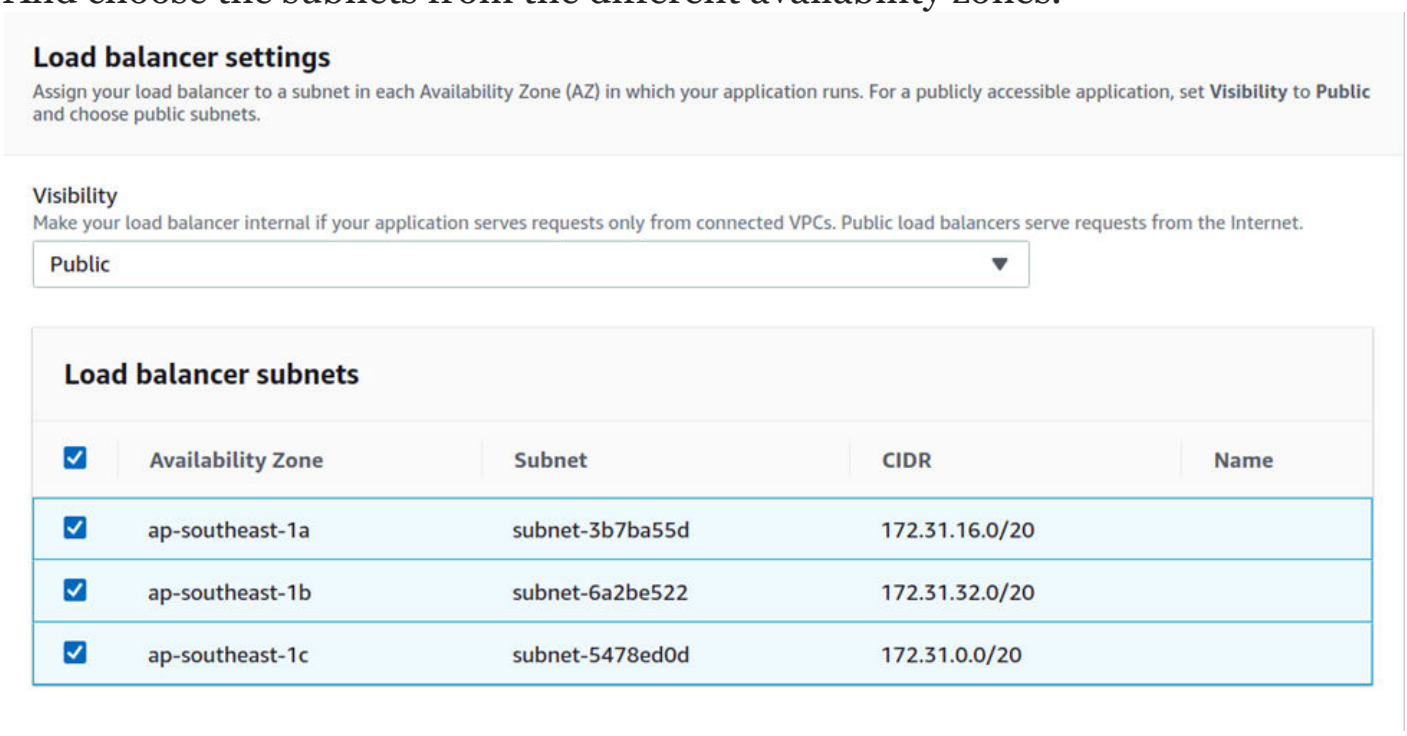
We have to choose the VPC and the subnets where the environment setup for the application should be created.

Here we have to configure subnets for the Load Balancer and the EC2 instances.

Select the VPC and Under the Load balancer settings,

The Visibility of the load balancer is Public.

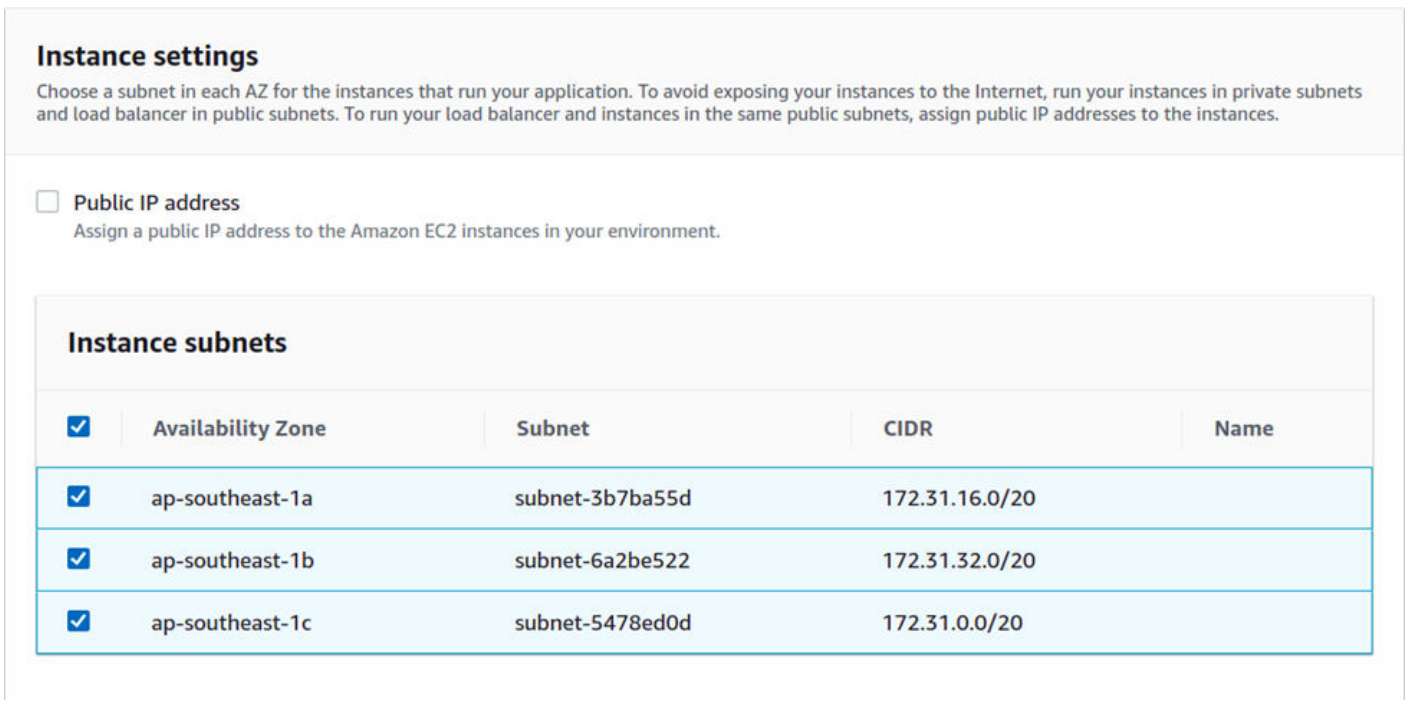
And choose the subnets from the different availability zones.



For Instance settings, As the traffic to the EC2 instances is routed using the public Load balancer.

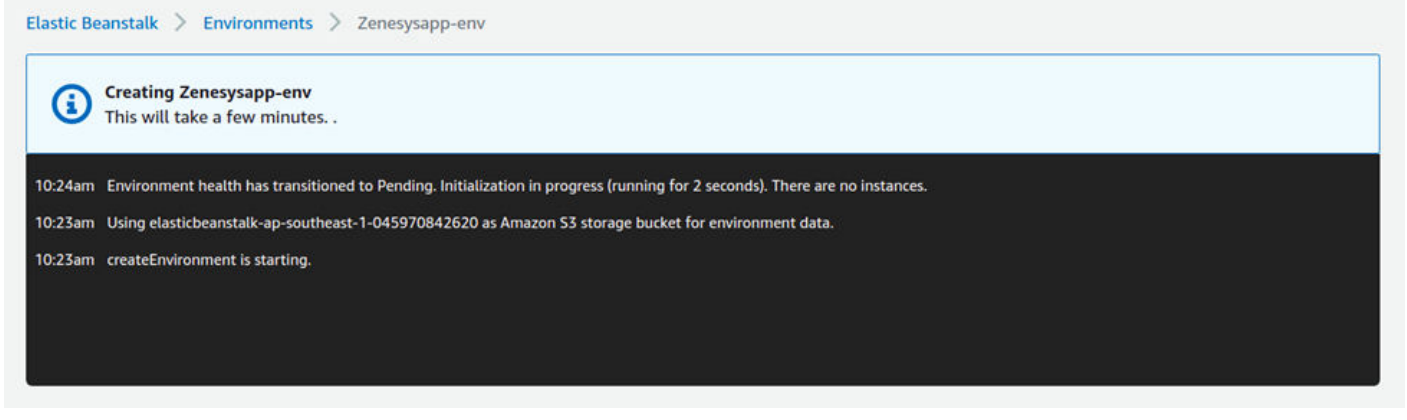
We don’t have to enable the Public IP address for the EC2 instances.

Choose the subnets from Different Az for the EC2 instances and click Save.



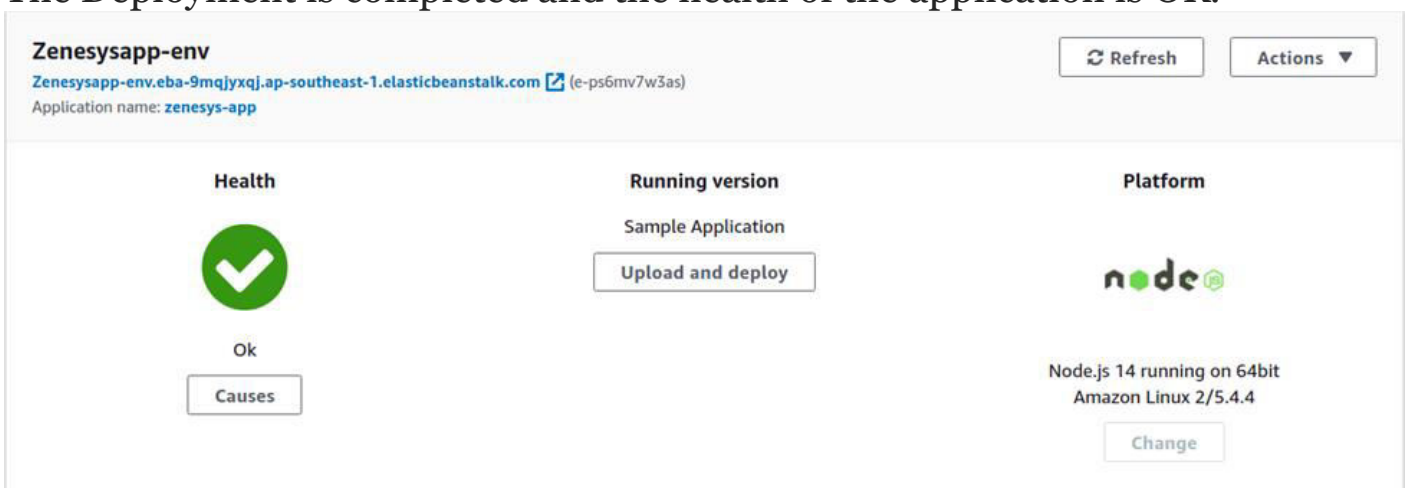
Now it’s time to click Create an app to deploy the application into the Elastic Beanstalk environment.

The Environment has started to create.

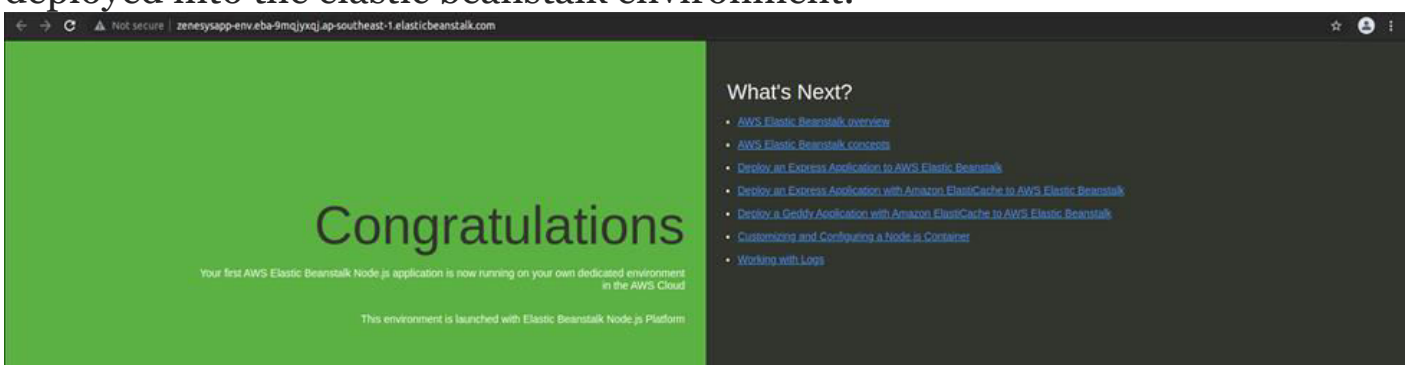


It will take a few minutes for the Environment and the application to be successfully deployed.

The Deployment is completed and the health of the application is OK.



Click the link, which will display the following page, which shows that the sample **Node.js application** is deployed into the elastic beanstalk environment.



Conclusion

We have successfully deployed a Node.js application into the Elastic Beanstalk environment with High availability and an Autoscaling setup.

Modifying the properties of the deployment

AWS Elastic Beanstalk provides several options for how deployments are processed, including deployment policies (*All at once*, *Rolling*, *Rolling with additional batch*, *Immutable*, and *Traffic splitting*) and options that let you configure batch size and health check behavior during deployments. By default, your environment uses all-at-once deployments. If you created the environment with the EB CLI and it's a scalable environment (you didn't specify the `--single` option), it uses rolling deployments.

With *rolling deployments*, Elastic Beanstalk splits the environment's Amazon EC2 instances into batches and deploys the new version of the application to one batch at a time. It leaves the rest of the instances in the environment running the old version of the application. During a rolling deployment, some instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version. For details, see How rolling deployments work.

To maintain full capacity during deployments, you can configure your environment to launch a new batch of instances before taking any instances out of service. This option is known as a *rolling deployment with an additional batch*. When the deployment completes, Elastic Beanstalk terminates the additional batch of instances.

Immutable deployments perform an immutable update to launch a full set of new instances running the new version of the application in a separate Auto Scaling group, alongside the instances running the old version. Immutable deployments can prevent issues caused by partially completed rolling deployments. If the new instances don't pass health checks, Elastic Beanstalk terminates them, leaving the original instances untouched.

Traffic-splitting deployments let you perform canary testing as part of your application deployment. In a traffic-splitting deployment, Elastic Beanstalk launches a full set of new instances just like during an immutable deployment. It then forwards a specified percentage of incoming client traffic to the new application version for a specified evaluation period. If the new instances stay healthy, Elastic Beanstalk forwards all traffic to them and terminates the old ones. If the new instances don't pass health checks, or if you choose to abort the deployment, Elastic Beanstalk moves traffic back to the old instances and terminates the new ones. There's never any service interruption. For details, see How traffic-splitting deployments work.

Warning

Some policies replace all instances during the deployment or update. This causes all accumulated Amazon EC2 burst balances to be lost. It happens in the following cases:

- Managed platform updates with instance replacement enabled
- Immutable updates
- Deployments with immutable updates or traffic splitting enabled

If your application doesn't pass all health checks, but still operates correctly at a lower health status, you can allow instances to pass health checks with a lower status, such as `Warning`, by modifying the **Healthy threshold** option. If your deployments fail because they don't pass health checks and you need to force an update regardless of health status, specify the **Ignore health check** option.

When you specify a batch size for rolling updates, Elastic Beanstalk also uses that value for rolling application restarts. Use rolling restarts when you need to restart the proxy and application servers running on your environment's instances without downtime.

Configuring application deployments

In the environment management console, enable and configure batched application version deployments by editing **Updates and Deployments** on the environment's **Configuration** page.

To configure deployments (console)

1. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.
2. In the navigation pane, choose **Environments**, and then choose the name of your environment from the list.
   Note

   If you have many environments, use the search bar to filter the environment list.
3. In the navigation pane, choose **Configuration**.
4. In the **Rolling updates and deployments** configuration category, choose **Edit**.
5. In the **Application Deployments** section, choose a **Deployment policy**, batch settings, and health check options.
6. To save the changes choose **Apply** at the bottom of the page.

The **Application deployments** section of the **Rolling updates and deployments** page has the following options for application deployments:

- **Deployment policy** – Choose from the following deployment options:
  - **All at once** – Deploy the new version to all instances simultaneously. All instances in your environment are out of service for a short time while the deployment occurs.
  - **Rolling** – Deploy the new version in batches. Each batch is taken out of service during the deployment phase, reducing your environment's capacity by the number of instances in a batch.
  - **Rolling with additional batch** – Deploy the new version in batches, but first launch a new batch of instances to ensure full capacity during the deployment process.
  - **Immutable** – Deploy the new version to a fresh group of instances by performing an immutable update.
  - **Traffic splitting** – Deploy the new version to a fresh group of instances and temporarily split incoming client traffic between the existing application version and the new one.

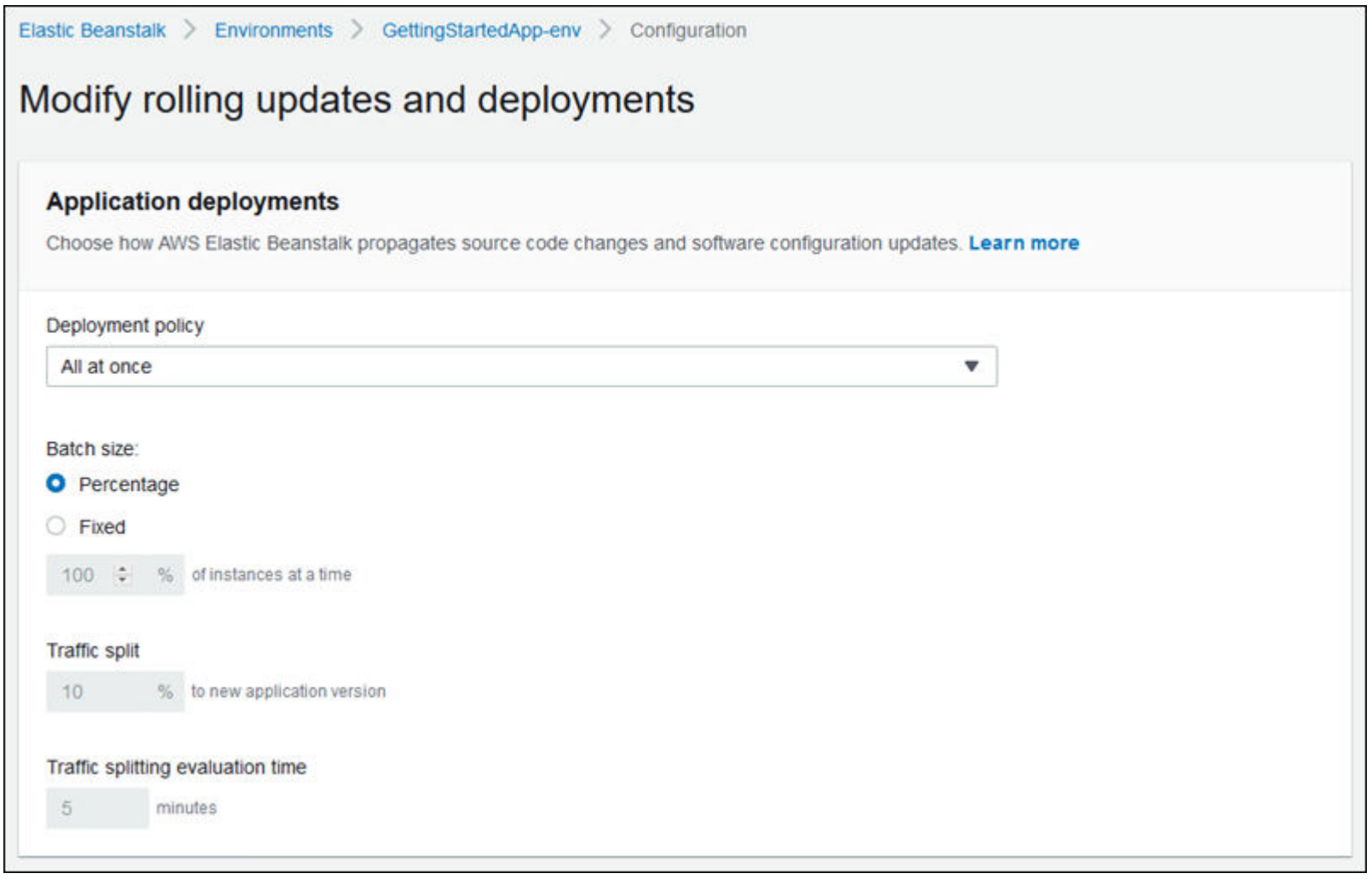
For the **Rolling** and **Rolling with additional batch** deployment policies you can configure:

- **Batch size** – The size of the set of instances to deploy in each batch.

  Choose **Percentage** to configure a percentage of the total number of EC2 instances in the Auto Scaling group (up to 100 percent), or choose **Fixed** to configure a fixed number of instances (up to the maximum instance count in your environment's Auto Scaling configuration).

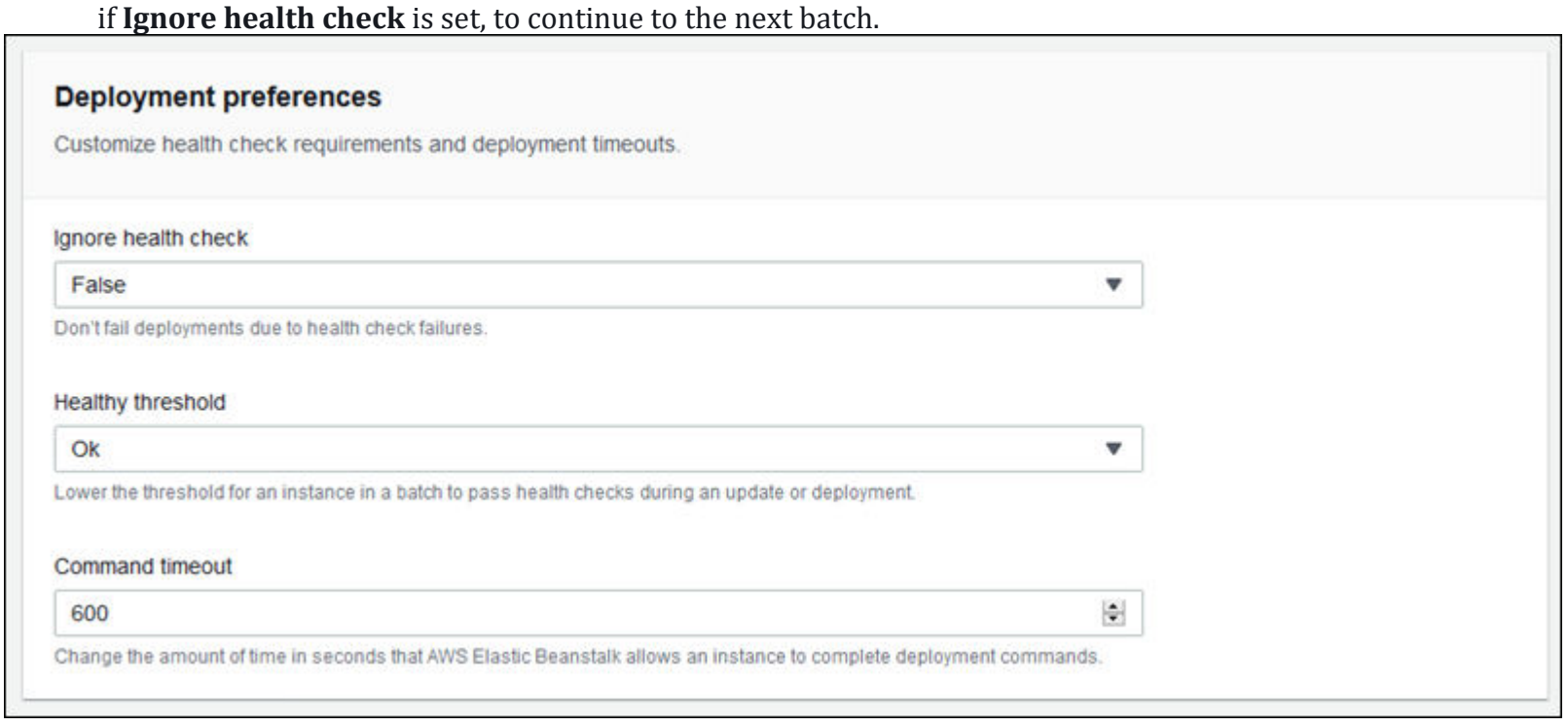
For the **Traffic splitting** deployment policy you can configure the following:

- **Traffic split** – The initial percentage of incoming client traffic that Elastic Beanstalk shifts to environment instances running the new application version you're deploying.
- **Traffic splitting evaluation time** – The time period, in minutes, that Elastic Beanstalk waits after an initial healthy deployment before proceeding to shift all incoming client traffic to the new application version that you're deploying.



The **Deployment preferences** section contains options related to health checks.

- **Ignore health check** – Prevents a deployment from rolling back when a batch fails to become healthy within the **Command timeout**.
- **Healthy threshold** – Lowers the threshold at which an instance is considered healthy during rolling deployments, rolling updates, and immutable updates.
- **Command timeout** – The number of seconds to wait for an instance to become healthy before canceling the deployment or, if **Ignore health check** is set, to continue to the next batch.



How rolling deployments work

When processing a batch, Elastic Beanstalk detaches all instances in the batch from the load balancer, deploys the new application version, and then reattaches the instances. If you enable connection draining, Elastic Beanstalk drains existing connections from the Amazon EC2 instances in each batch before beginning the deployment.

After reattaching the instances in a batch to the load balancer, Elastic Load Balancing waits until they pass a minimum number of Elastic Load Balancing health checks (the **Healthy check count threshold** value), and then starts routing traffic to them. If no health check URL is configured, this can happen very quickly, because an instance will pass the health check as soon as it can accept a TCP connection. If a health check URL is configured, the load balancer doesn't route traffic to the updated instances until they return a `200 OK` status code in response to an `HTTP GET` request to the health check URL.

Elastic Beanstalk waits until all instances in a batch are healthy before moving on to the next batch. With basic health reporting, instance health depends on the Elastic Load Balancing health check status. When all instances in the batch pass enough health checks to be considered healthy by Elastic Load Balancing, the batch is complete. If enhanced health reporting is enabled, Elastic Beanstalk considers several other factors, including the result of incoming requests. With enhanced health reporting, all instances must pass 12 consecutive health checks with an OK status within two minutes for web server environments, and 18 health checks within three minutes for worker environments.

If a batch of instances does not become healthy within the command timeout, the deployment fails. After a failed deployment, check the health of the instances in your environment for information about the cause of the failure. Then perform another deployment with a fixed or known good version of your application to roll back.

If a deployment fails after one or more batches completed successfully, the completed batches run the new version of your application while any pending batches continue to run the old version. You can identify the version running on the instances in your environment on the health page in the console. This page displays the deployment ID of the most recent deployment that executed on each instance in your environment. If you terminate instances from the failed deployment, Elastic Beanstalk replaces them with instances running the application version from the most recent successful deployment.

How traffic-splitting deployments work

Traffic-splitting deployments allow you to perform canary testing. You direct some incoming client traffic to your new application version to verify the application's health before committing to the new version and directing all traffic to it.

During a traffic-splitting deployment, Elastic Beanstalk creates a new set of instances in a separate temporary Auto Scaling group. Elastic Beanstalk then instructs the load balancer to direct a certain percentage of your environment's incoming traffic to the new instances. Then, for a configured amount of time, Elastic Beanstalk tracks the health of the new set of instances. If all is well, Elastic Beanstalk shifts remaining traffic to the new instances and attaches them to the environment's original Auto Scaling group, replacing the old instances. Then Elastic Beanstalk cleans up—terminates the old instances and removes the temporary Auto Scaling group.

Note

The environment's capacity doesn't change during a traffic-splitting deployment. Elastic Beanstalk launches the same number of instances in the temporary Auto Scaling group as there are in the original Auto Scaling group at the time the deployment starts. It then maintains a constant number of instances in both Auto Scaling groups for the deployment duration. Take this fact into account when configuring the environment's traffic splitting evaluation time.

Rolling back the deployment to the previous application version is quick and doesn't impact service to client traffic. If the new instances don't pass health checks, or if you choose to abort the deployment, Elastic Beanstalk moves traffic back to the old instances and terminates the new ones. You can abort any deployment by using the environment overview page in the Elastic Beanstalk console, and choosing **Abort current operation** in **Environment actions**. You can also call the AbortEnvironmentUpdate API or the equivalent AWS CLI command.

Traffic-splitting deployments require an Application Load Balancer. Elastic Beanstalk uses this load balancer type by default when you create your environment using the Elastic Beanstalk console or the EB CLI.

Deployment option namespaces

You can use the configuration options in the aws:elasticbeanstalk:command namespace to configure your deployments. If you choose the traffic-splitting policy, additional options for this policy are available in the aws:elasticbeanstalk:trafficsplitting namespace.

Use the `DeploymentPolicy` option to set the deployment type. The following values are supported:

- `AllAtOnce` – Disables rolling deployments and always deploys to all instances simultaneously.
- `Rolling` – Enables standard rolling deployments.
- `RollingWithAdditionalBatch` – Launches an extra batch of instances, before starting the deployment, to maintain full capacity.
- `Immutable` – Performs an immutable update for every deployment.
- `TrafficSplitting` – Performs traffic-splitting deployments to canary-test your application deployments.

When you enable rolling deployments, set the `BatchSize` and `BatchSizeType` options to configure the size of each batch. For example, to deploy 25 percent of all instances in each batch, specify the following options and values.

Example .ebextensions/rolling-updates.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Rolling
    BatchSizeType: Percentage
    BatchSize: 25
```

To deploy to five instances in each batch, regardless of the number of instances running, and to bring up an extra batch of five instances running the new version before pulling any instances out of service, specify the following options and values.

Example .ebextensions/rolling-additionalbatch.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: RollingWithAdditionalBatch
    BatchSizeType: Fixed
    BatchSize: 5
```

To perform an immutable update for each deployment with a health check threshold of **Warning**, and proceed with the deployment even if instances in a batch don't pass health checks within a timeout of 15 minutes, specify the following options and values.

Example .ebextensions/immutable-ignorehealth.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Immutable
    HealthCheckSuccessThreshold: Warning
    IgnoreHealthCheck: true
    Timeout: "900"
```

To perform traffic-splitting deployments, forwarding 15 percent of client traffic to the new application version and evaluating health for 10 minutes, specify the following options and values.

Example .ebextensions/traffic-splitting.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: TrafficSplitting
  aws:elasticbeanstalk:trafficsplitting:
    NewVersionPercent: "15"
    EvaluationTime: "10"
```

The EB CLI and Elastic Beanstalk console apply recommended values for the preceding options. You must remove these settings if you want to use configuration files to configure the same. See Recommended values for details.

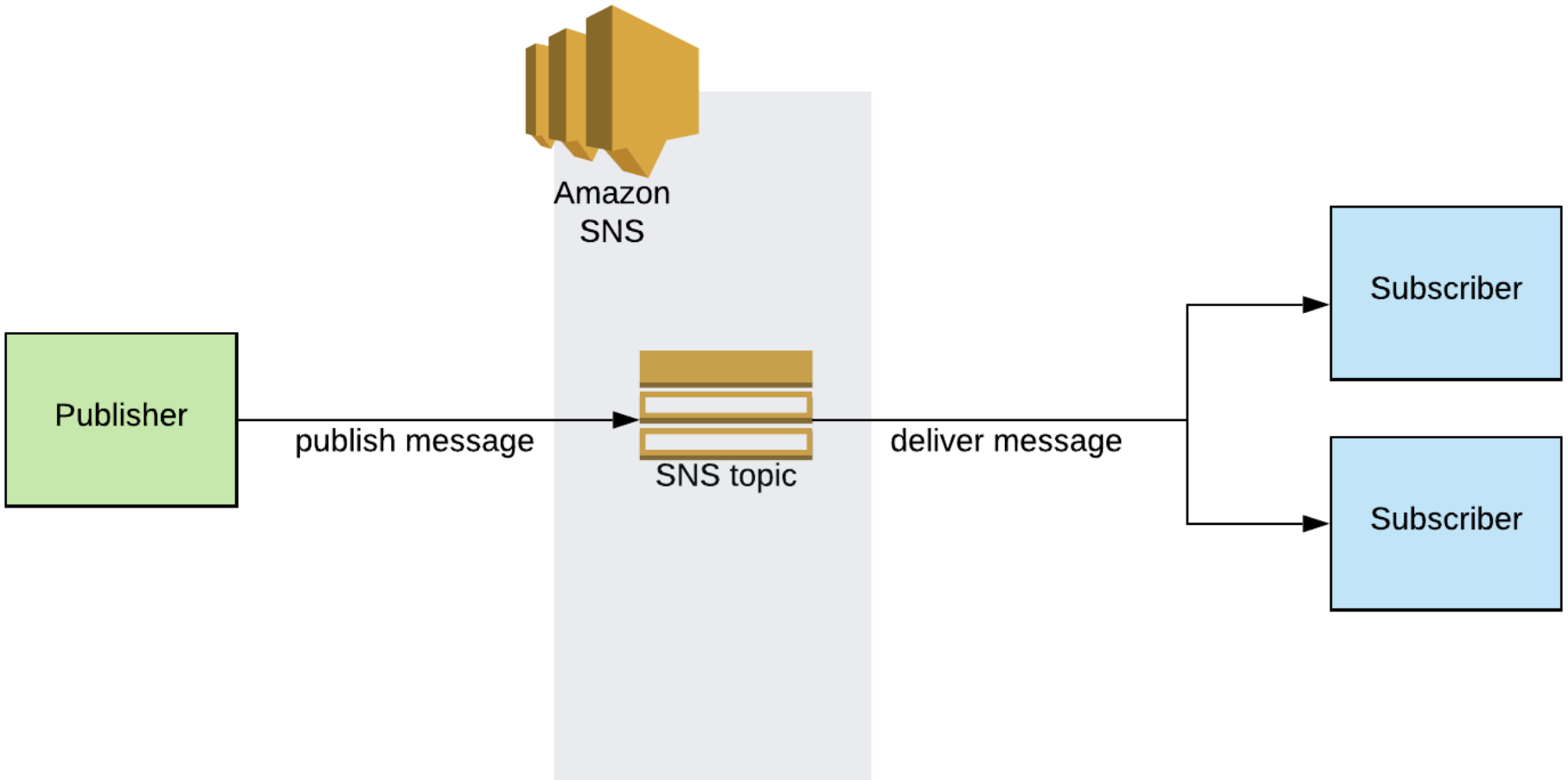
14. AWS Monitoring & Notification services

Amazon Cloud Watch – Create Topics & Set Alarms

n this article you'll discover how to setup the AWS Simple Notification Service (SNS) to send you emails whenever a CloudWatch alarm gets triggered. We'll run through a full working example, setting up the alarm and SNS resources, and demonstrating the notifications coming through in real time.

What is AWS SNS?

SNS (Simple Notification Service) is a highly available publish/subscribe messaging service. The ***publish-subscribe*** pattern allows publishers to send messages, without knowledge of who they need to be send to. Instead, it's up to the subscriber to register itself to receive messages from specific topics it's interested in.



An ***SNS topic*** is a communication channel to link together publishers and subscribers. In this diagram, you can see that a publisher is sending messages to a specific topic, and these messages are then being delivered to two subscribers.

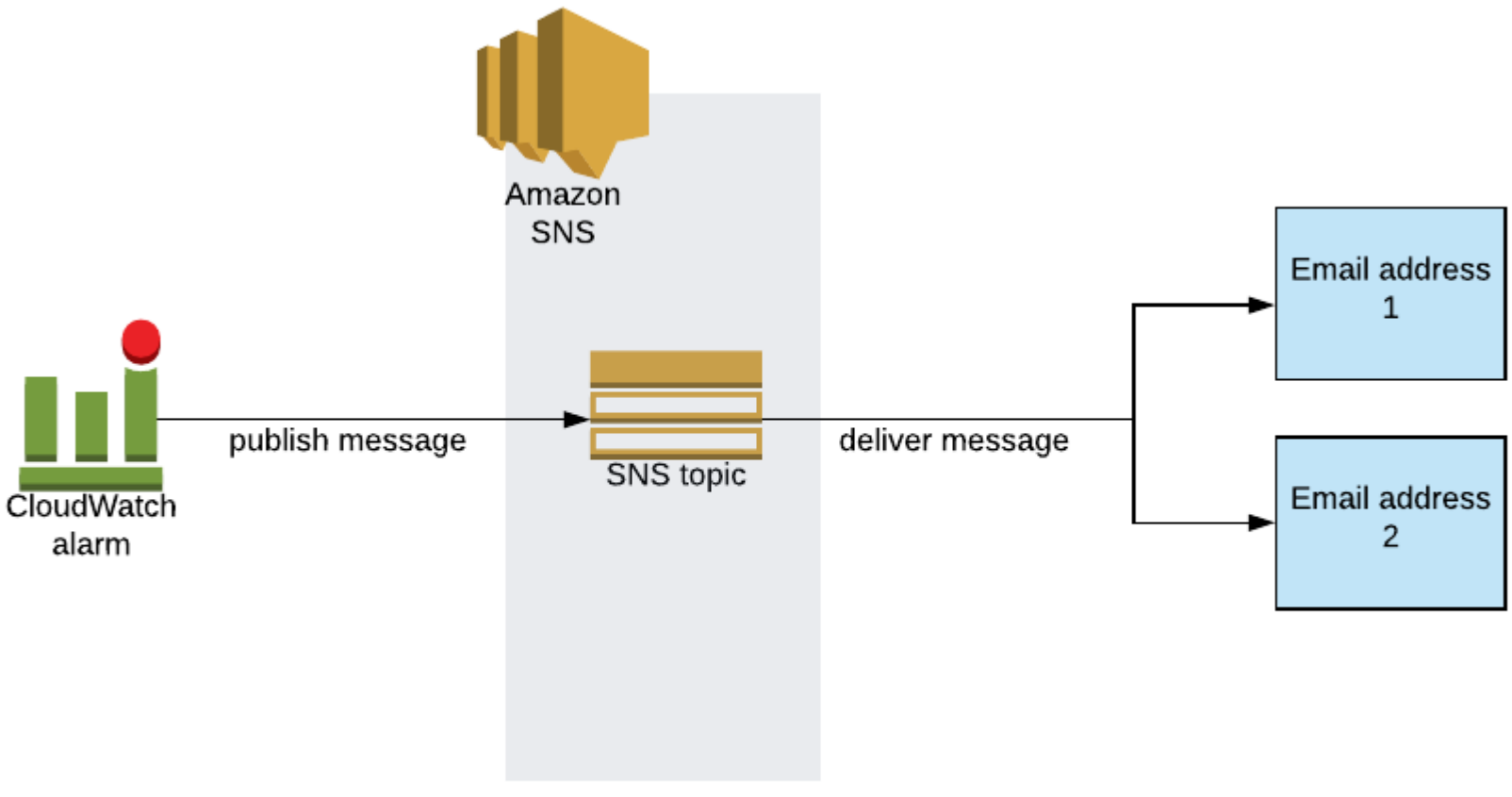
The key benefits of using SNS are:

1. publishers don't need to know about their subscribers, allowing for decoupling of services
2. one message can be sent to many subscribers, allowing for parallel processing
3. it's highly available (would we expect anything less from AWS?)
4. it's compatible with many AWS services to use for the message source and destination (including CloudWatch and emails)

How can we use SNS with CloudWatch alarms?

CloudWatch is an AWS monitoring service that allows you to ingest logs and metrics from your applications or servers. ***CloudWatch Alarms*** can be configured to be raised when metrics meet certain criteria. For example, you could set up alarms to be notified about low disk space or high CPU usage.

Fortunately CloudWatch can send messages directly to an SNS topic and SNS allows us to setup an email addresses for the message to be sent to.



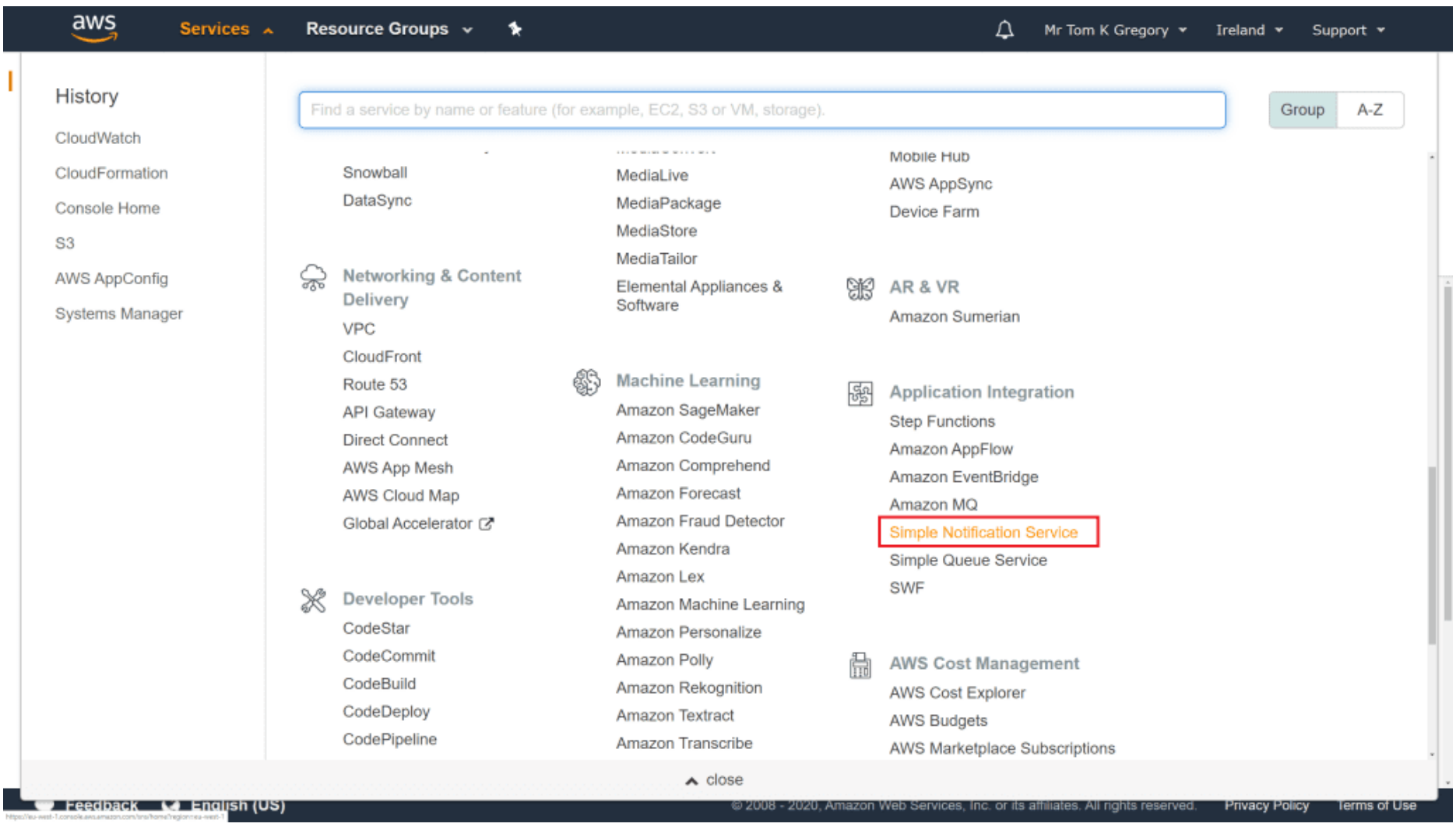
You can see from this diagram that effectively CloudWatch will now act as the publisher, and the email accounts as the subscribers. This allows CloudWatch alarms to send us email notifications.

Setting up SNS notifications for an existing CloudWatch alarm in the AWS console

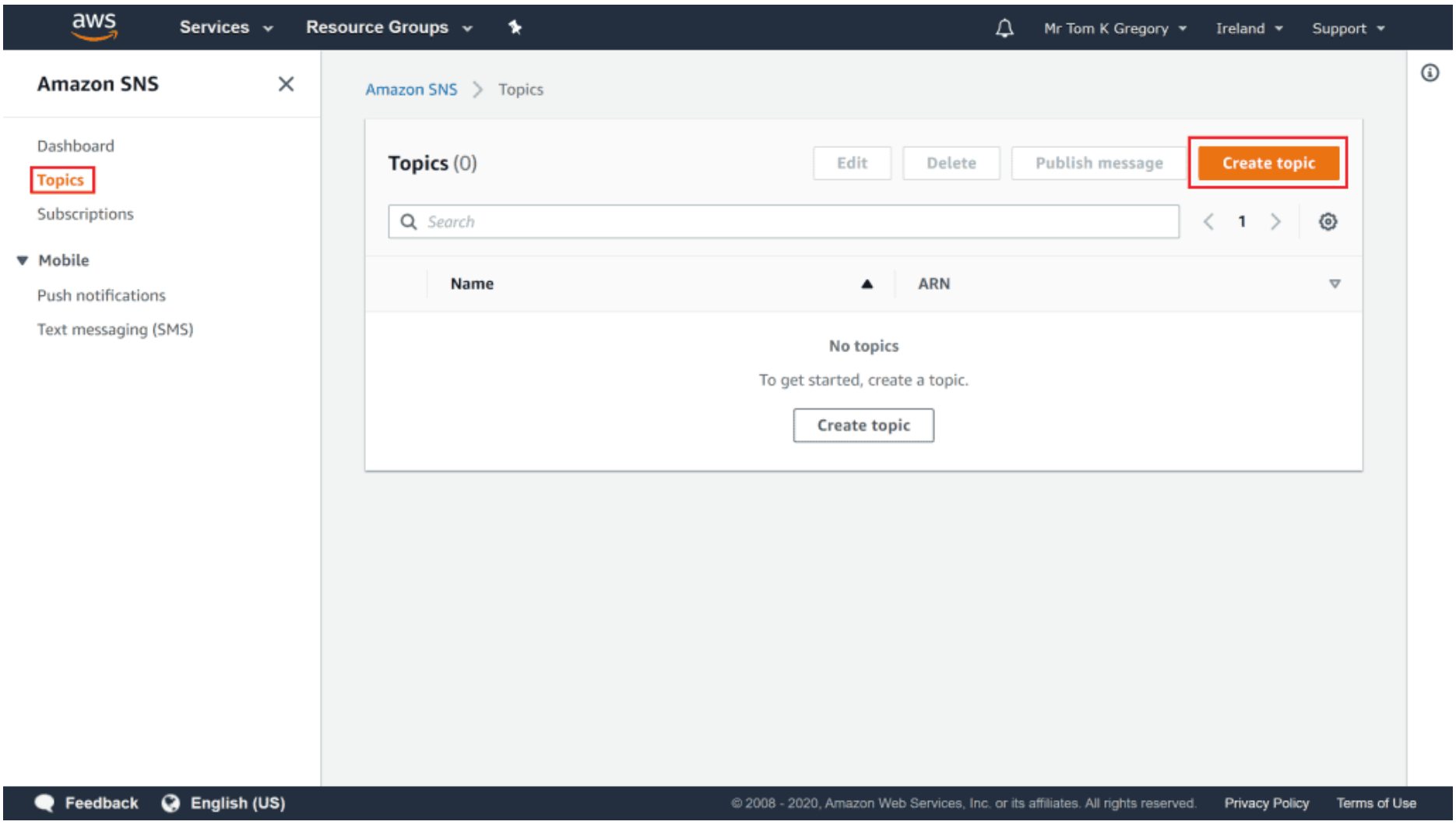
If you already have an alarm, follow these steps to setup an email notification from the AWS console. For a full end-to-end working example, including creating the alarm itself, see the next section.

Create a topic

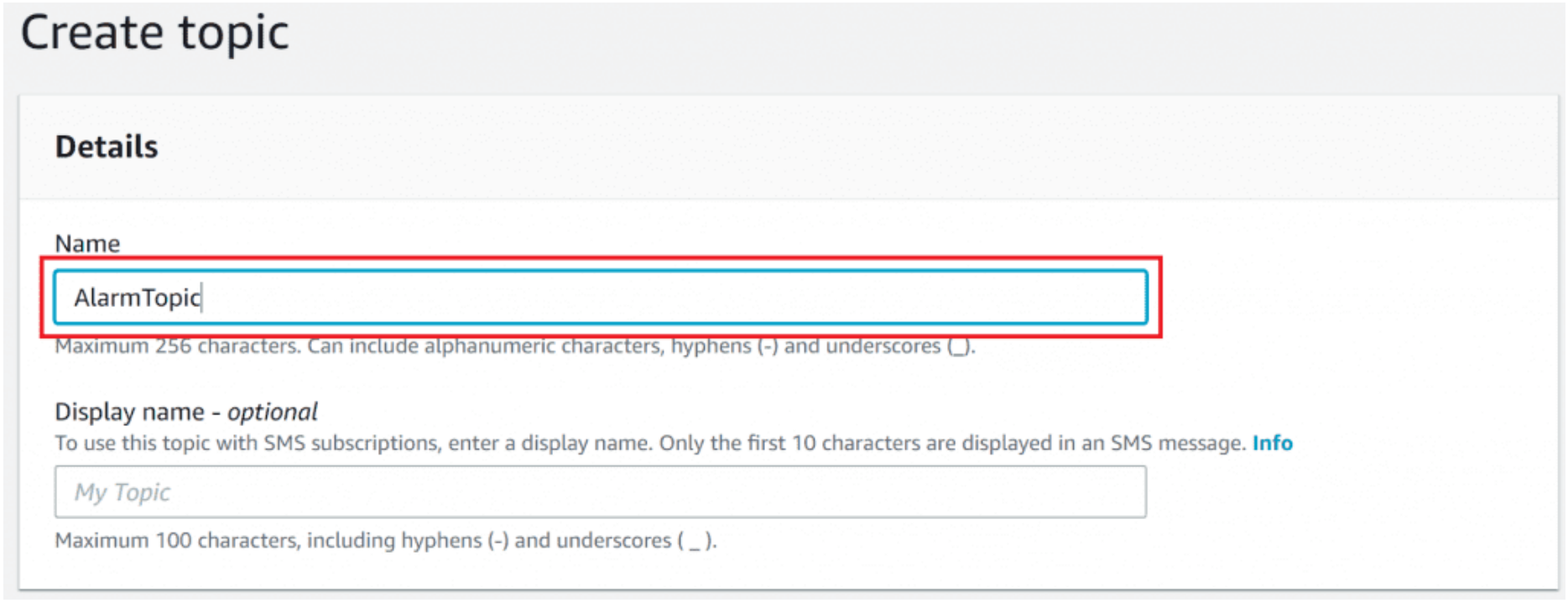
Select *Services* then type or select *Simple Notification Service*:



Select *Topics* from the left hand navigation. This page is where any existing topics are shown. Let's create a new one, so select *Create topic*:

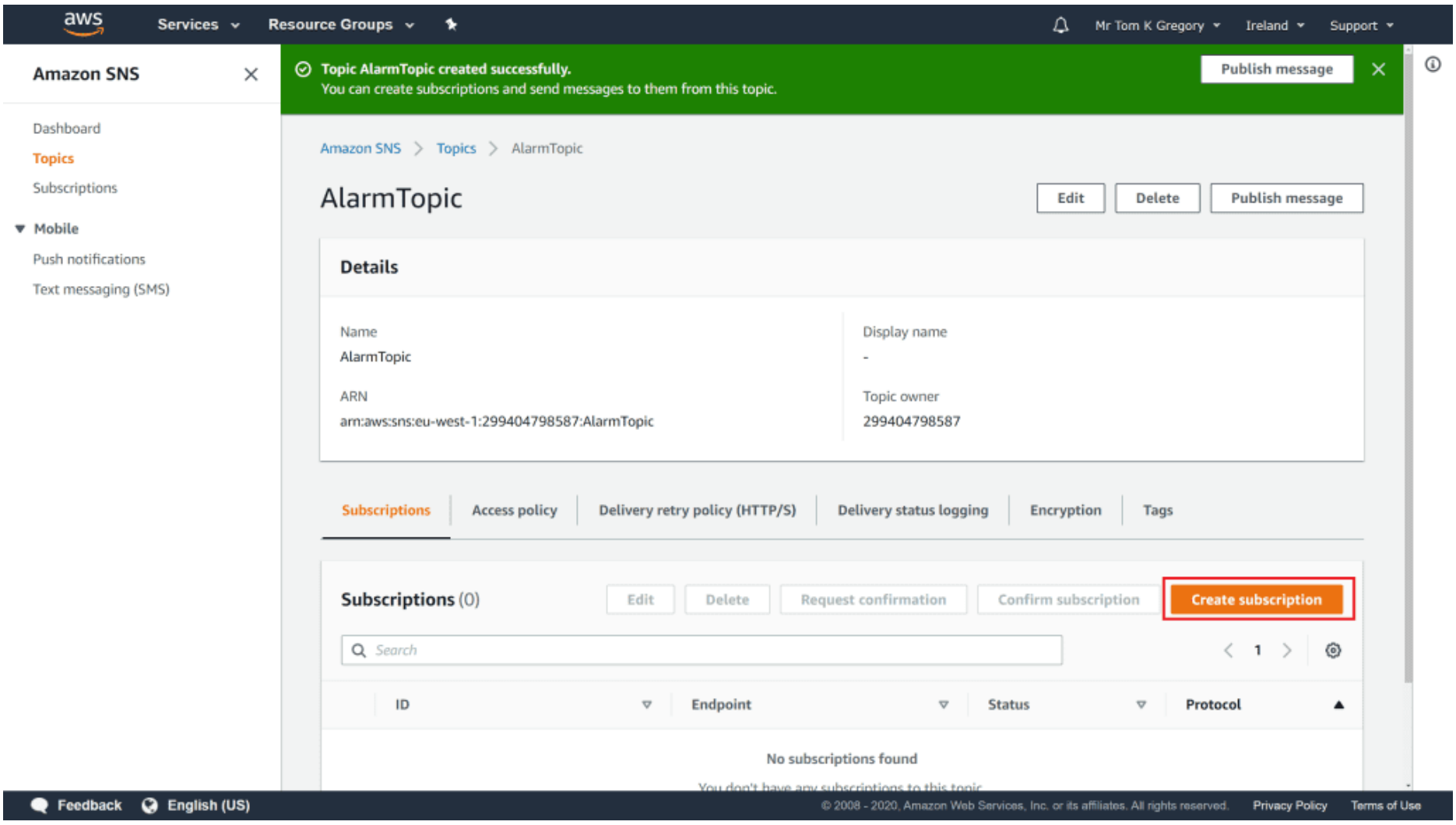


On the *Create topic* page you just need to specify a topic name. Type in something appropriate, then select the *Create topic* button at the bottom:

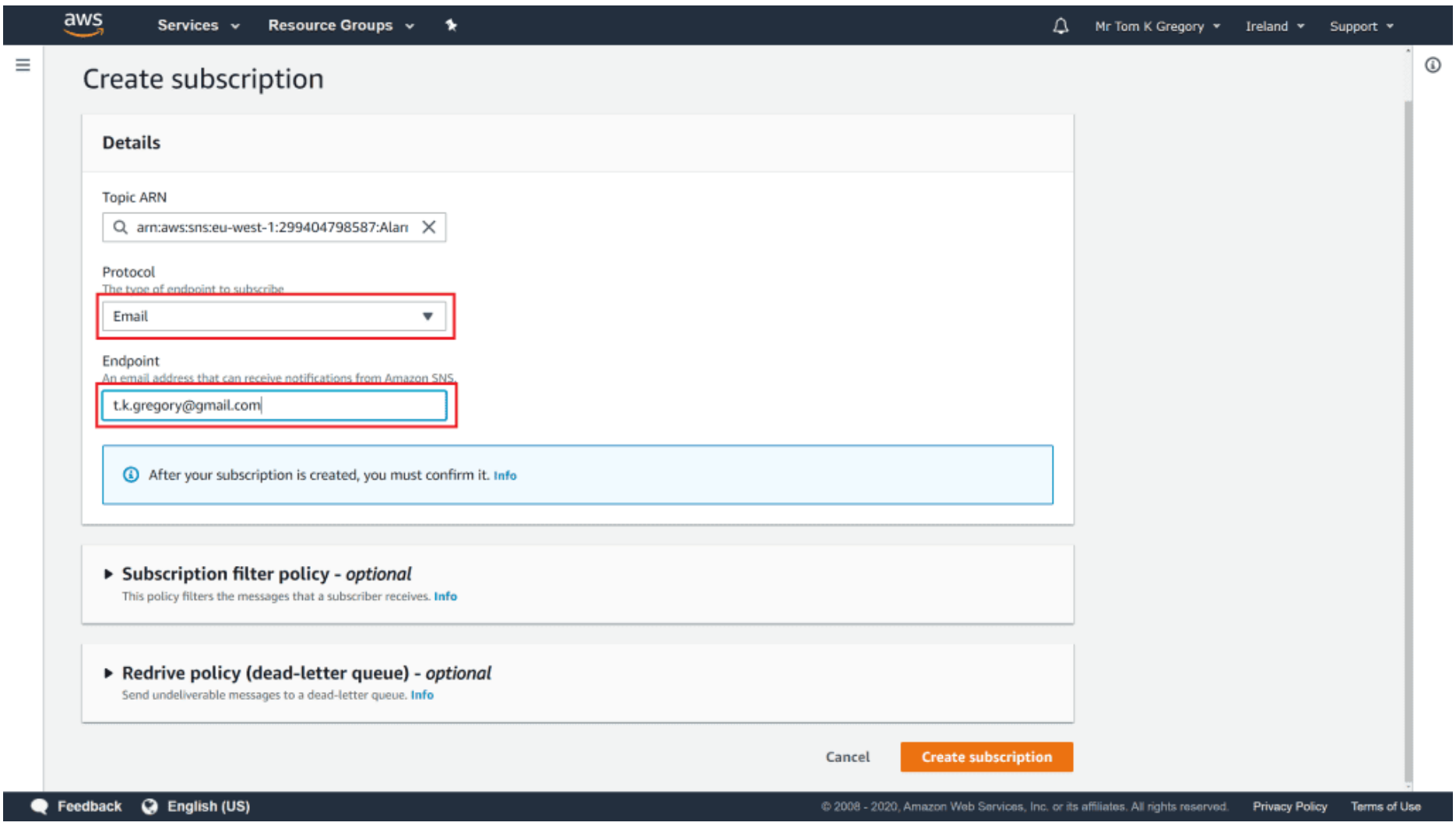


Add an email subscription to the topic

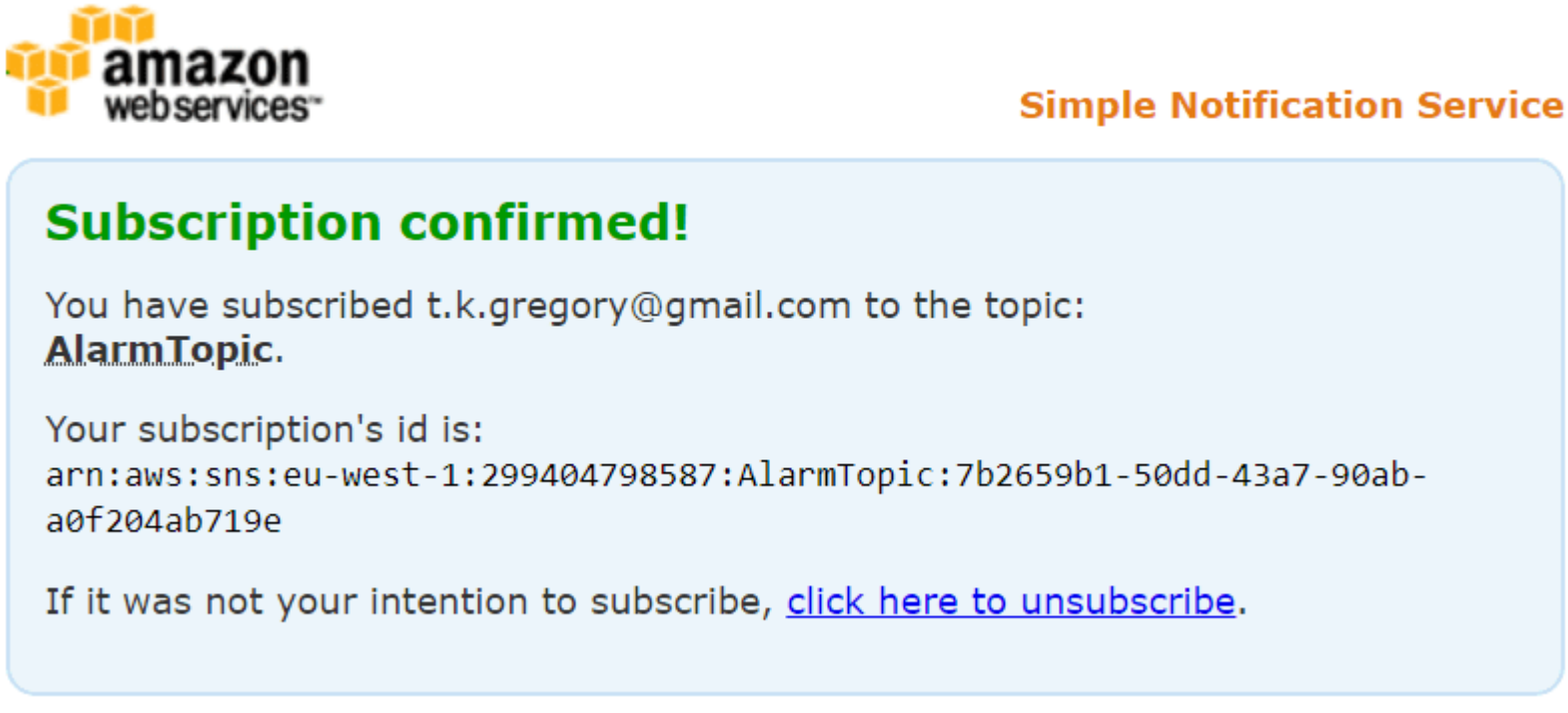
Now that we have a topic, let's subscribe an email address to it. On the topic details page, select *Create subscription*:



On the *Create subscription* page you need to select *Email* from the *Protocol* drop down list. Type your email address into the *Endpoint* field, then select *Create subscription* at the bottom:



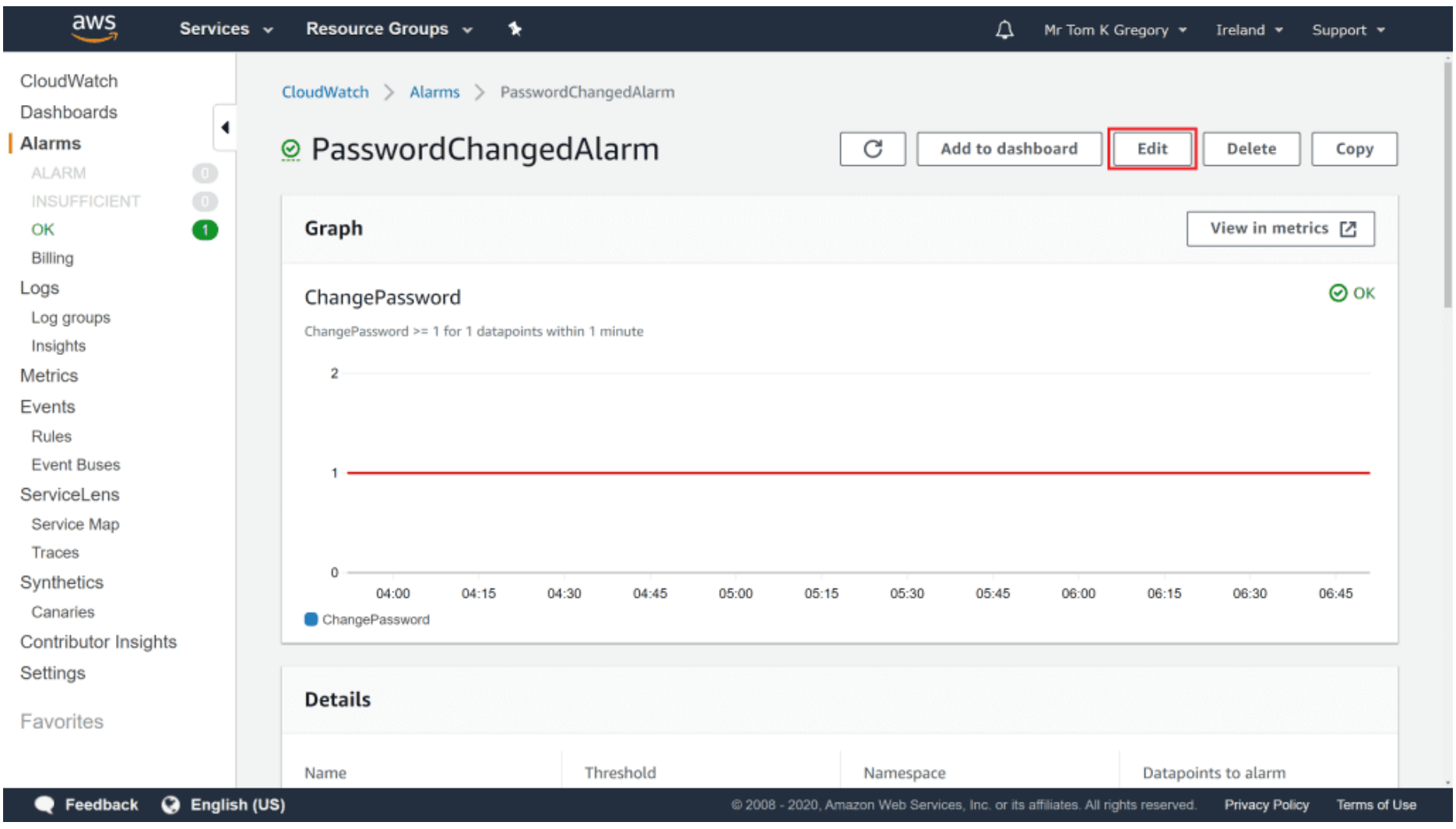
You'll get an email from AWS. Open it up and click the confirm link to confirm the subscription:



Create the alarm action

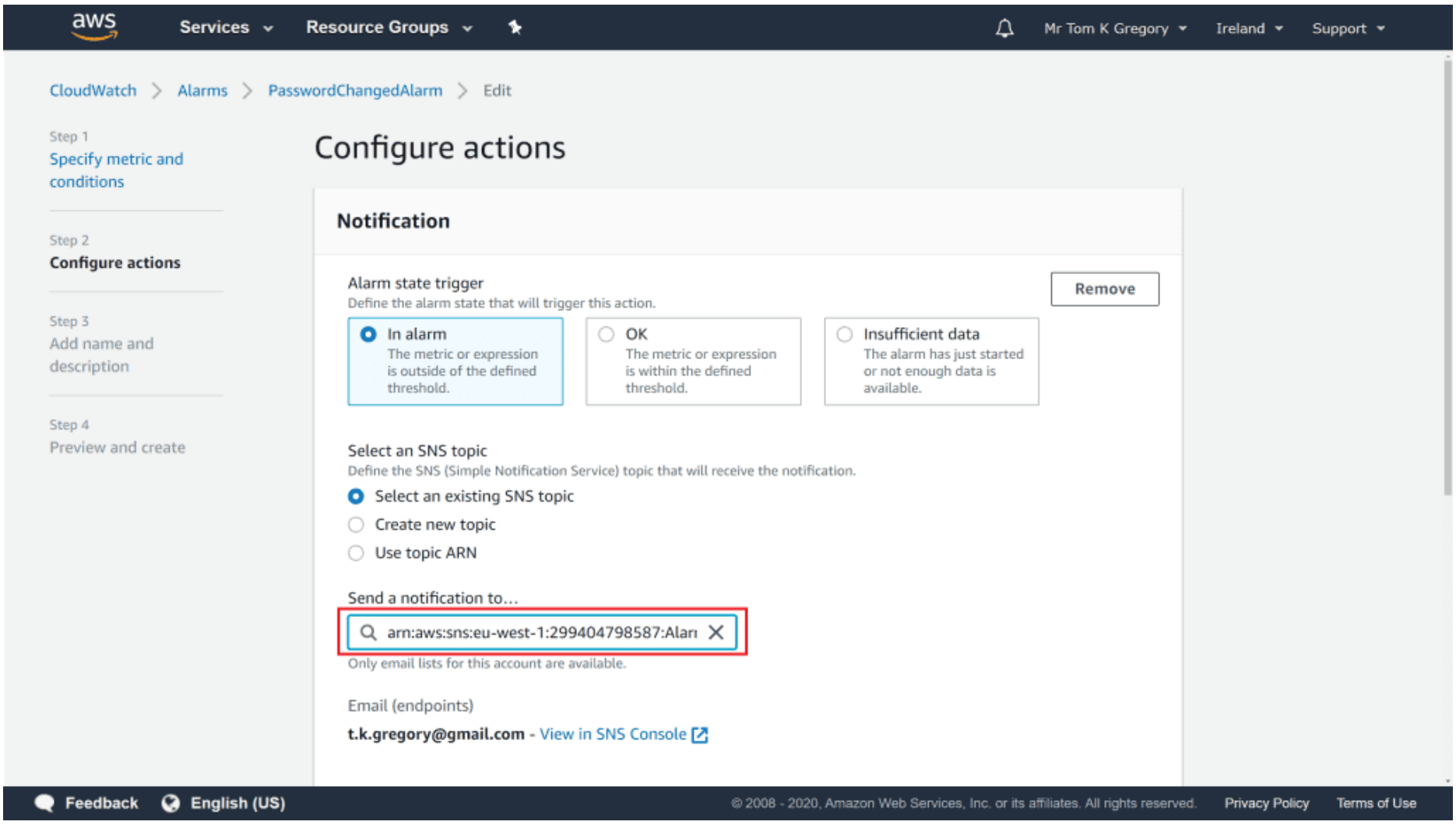
The last thing to do here is create an action in the alarm so that it sends a notification to SNS whenever it enters the *ALARM* status.

In CloudWatch, navigate to the alarm details for an existing alarm (if you don't have one follow the full example in the next section). Select *Edit*:



You'll now be on a four step edit page. Click *Next* to go to *Step 2*. This is the *Configure actions* page where we can add the notification.

Leave all the defaults as they are, including *Select an existing SNS topic*. Select the topic you added earlier from the *Send a notification to* input, then click *Update alarm* at the bottom of the page:



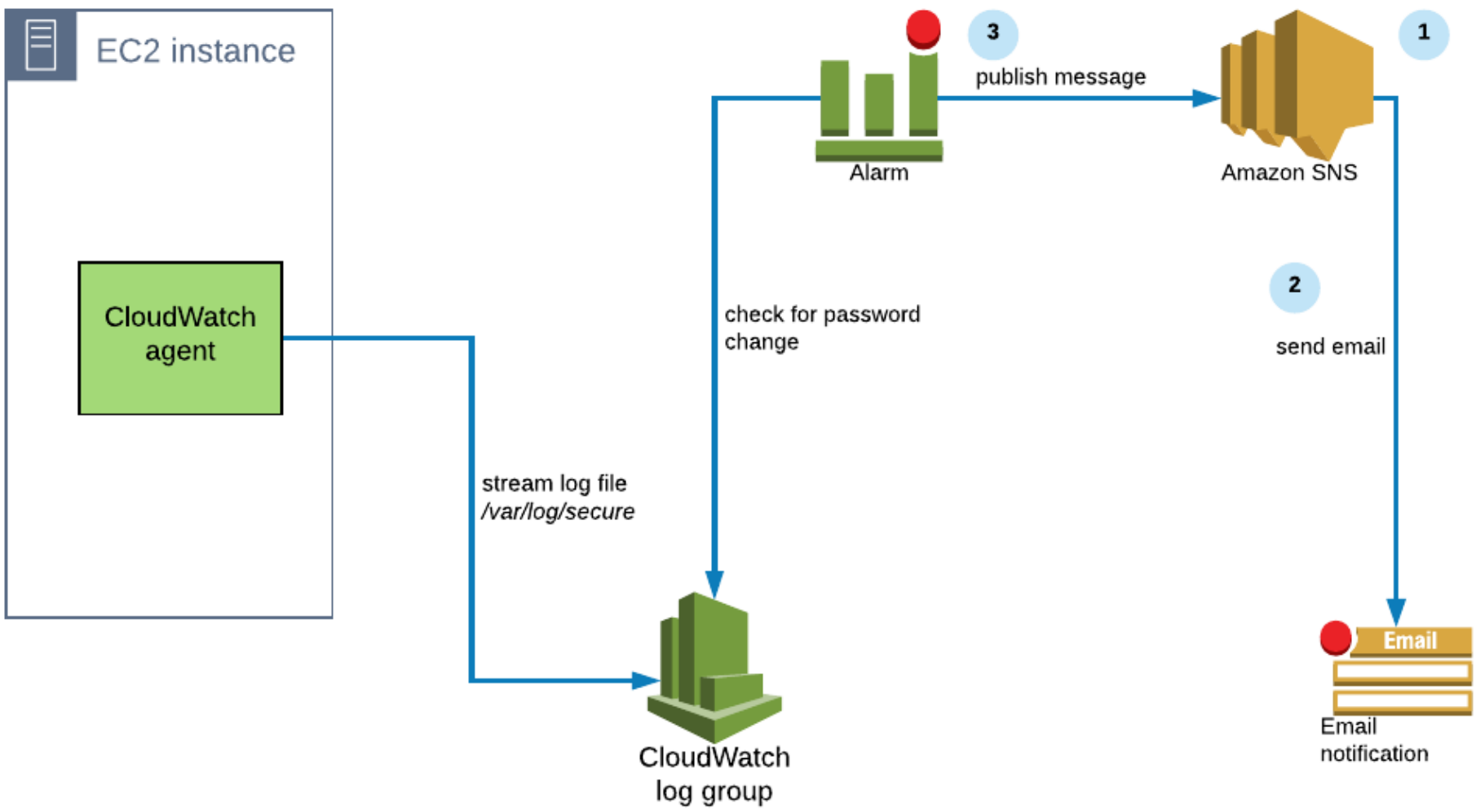
That's it. You'll now start receiving email notifications whenever the alarm is raised.

Full working example in CloudFormation

In the article *Shipping AWS EC2 logs to CloudWatch with the CloudWatch agent* we setup the CloudWatch agent on an EC2 instance to stream the */var/log/secure* log file to CloudWatch. We also included an alarm which notified us whenever someone tried to change their password on the EC2 instance.

Now let's extend this to:

1. create an SNS topic
2. subscribe an email address to the SNS topic, to receive an email alert about the alarm
3. publish a message to the SNS topic whenever the CloudWatch alarm gets raised

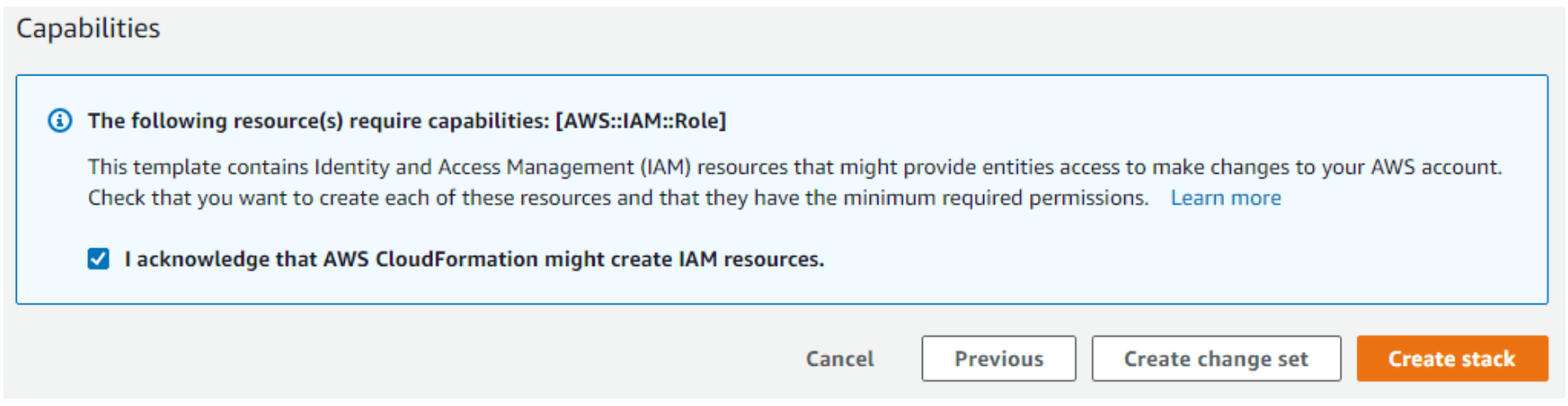


Launch the example with CloudFormation

You can go ahead and setup today's working example by clicking the Launch Stack button below, or download the template directly:

You can accept all the default options, but provide an email address to which the message will be sent from SNS:

Keep clicking the *Next* button, then on the last page before you click *Create stack* select the checkbox to allow this stack to create IAM resources:



Once the stack has been created, it will include all the resources from the CloudWatch agent article, with the following additions and changes.

Alarm SNS topic

We need to add the following `AWS::SNS::Topic` CloudFormation resource:

```
AlarmSNSTopic:
  Type: AWS::SNS::Topic
  Properties:
    TopicName: AlarmTopic
    Subscription:
      - Protocol: email
        Endpoint: !Ref EmailAddress
```

- ***TopicName*** must be provided, so let's just call it *AlarmTopic*
- ***Subscription*** can contain a list of subscribers that will get notified of any messages sent to this topic. We include a subscriber of type email, and reference the *EmailAddress* parameter.

Alarm actions

The `AWS::CloudWatch::Alarm` CloudFormation resource must now include the `AlarmActions` property:

```
Alarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName: PasswordChangedAlarm
    AlarmActions:
      - !Ref AlarmSNSTopic
    ...
```

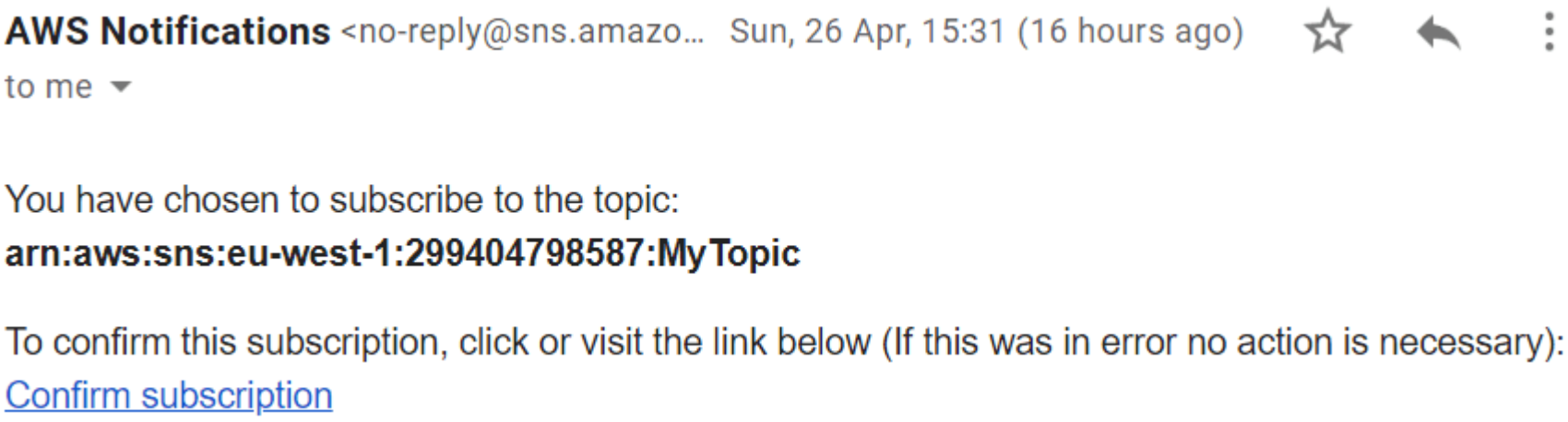
- ***AlarmActions*** allows us to specify the ARN (Amazon Resource Name) of the SNS topic that the alarm should be sent to

Alarm actions
There are all sorts of alarm actions that you can configure, including stopping, terminating, and rebooting EC2 instances. For more details, see these docs.

Confirming your SNS subscription

Because AWS wants to protect itself against spamming email addresses that it's users might configure, we have to confirm our subscription to SNS through a confirmation email.

Once the CloudFormation stack has been completed, you'll get an email like this:



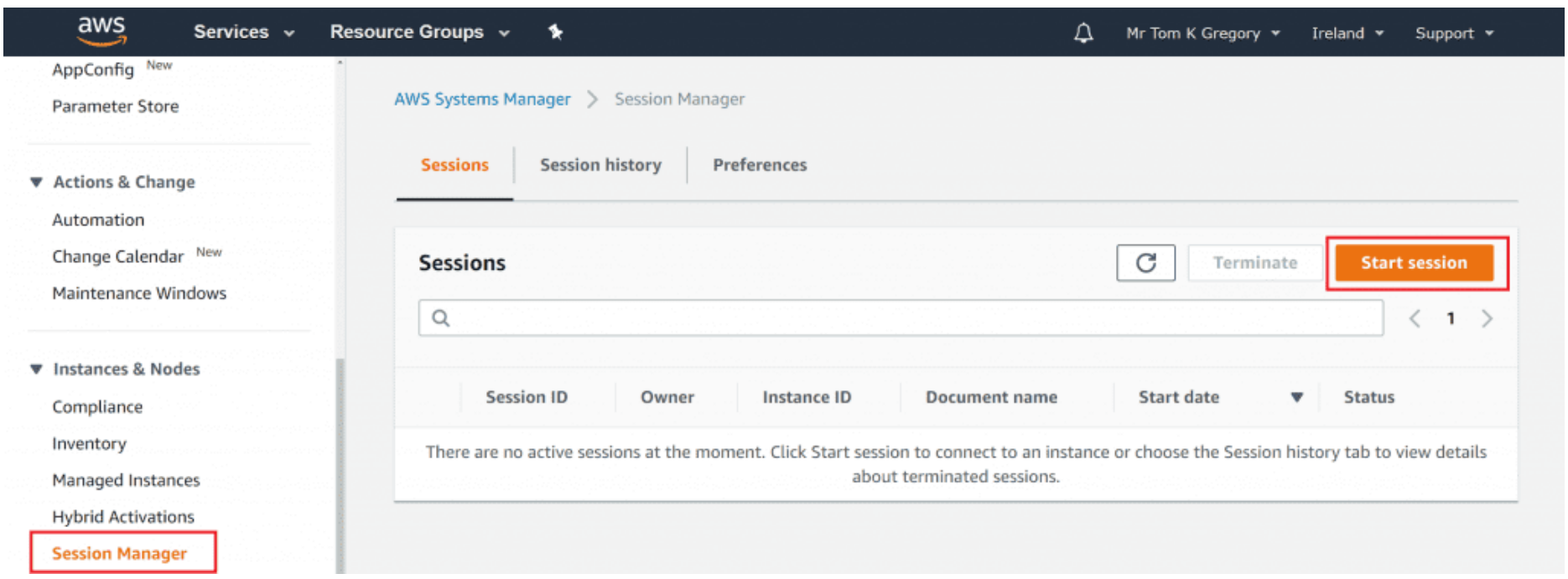
Click the *Confirm subscription* link, then you'll be ready to start receiving notifications. ✅

Raise the alarm!

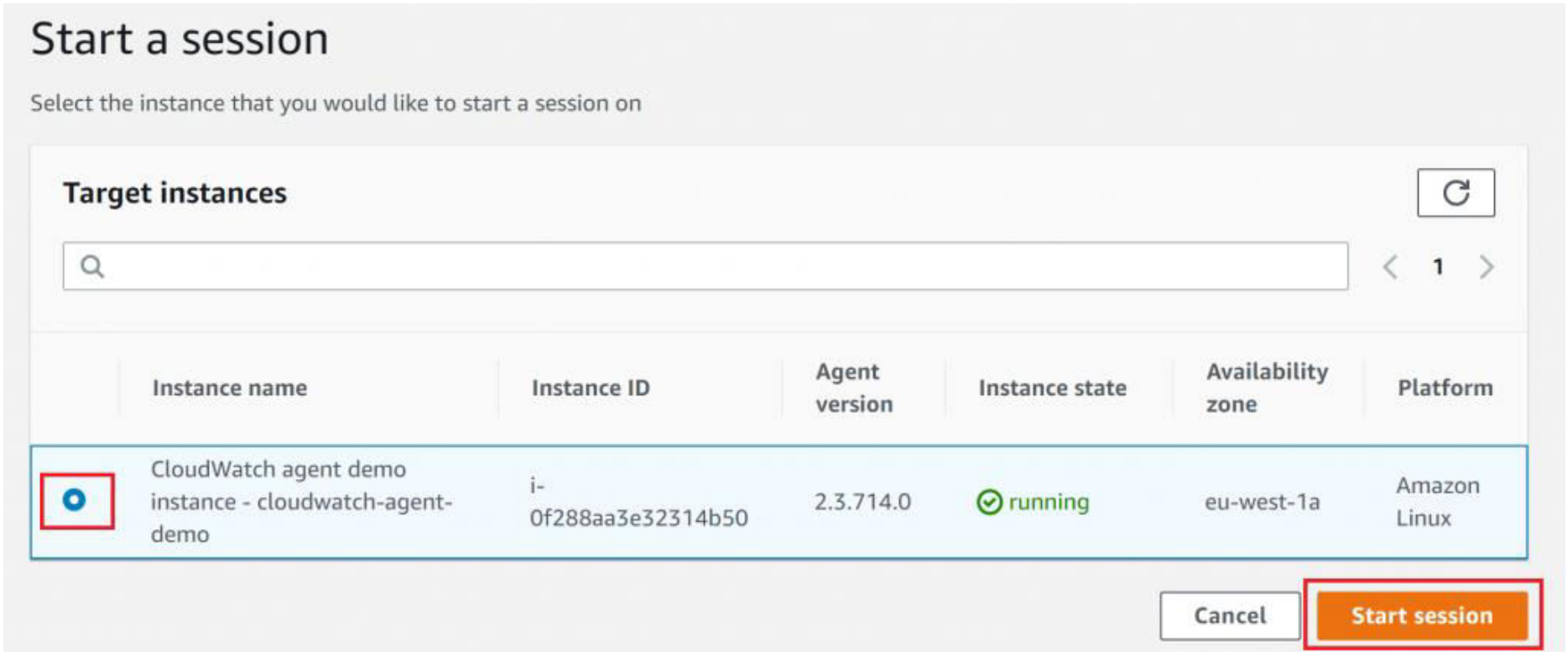
This alarm gets triggered whenever anyone changes a user password on the EC2 instance. We'll need to start a session in that instance, which we'll do using AWS Systems Manager Session Manager.

In the AWS console, go to *Services* > *Systems Manager*.

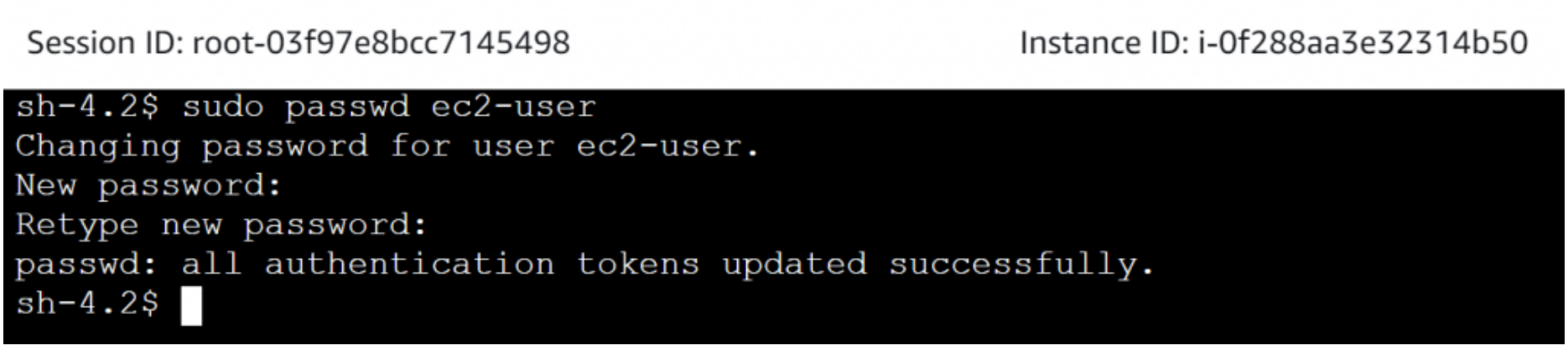
Then in the left hand navigation select *Session Manager*, then click the orange *Start session* button:



Now we can select the instance to connect to. Select the *CloudWatch agent demo instance*, then click *Start session*:



In the terminal window that appears, we'll run `sudo passwd ec2-user`. You'll then need to enter the password twice. To generate a random password try this site:



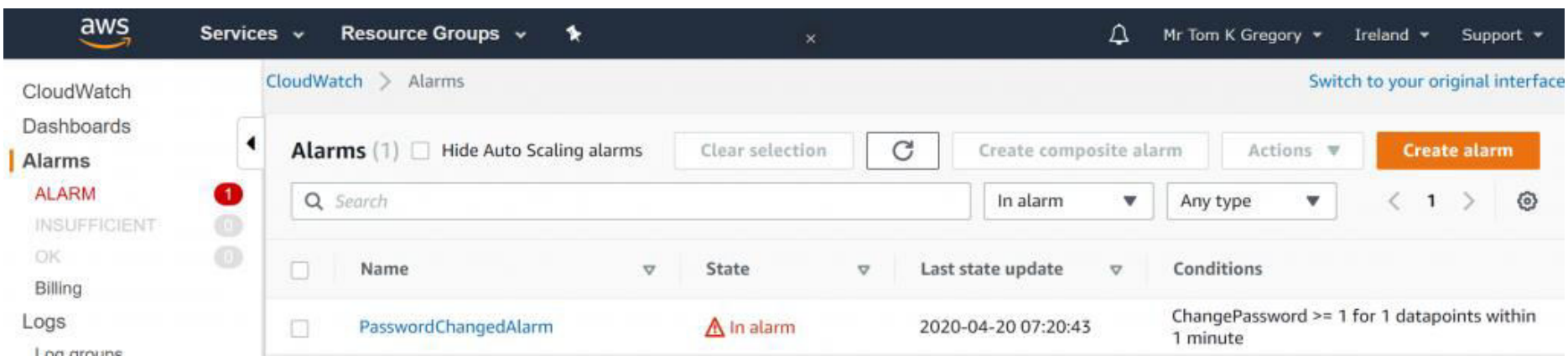
The follow sequence of events will now occur automatically:

1. The change password event will be logged in */var/log/secure*
2. The CloudWatch agent will stream that log to CloudWatch
3. The metric filter will generate a metric when it matches the text pattern *password changed* (see this article to learn about metric filters)
4. The alarm will detect the metric value increase and get raised
5. The alarm will then send a message to SNS
6. SNS will deliver that message to any subscribers, in this case the configured email account

Checking our notification

First off, let's see that the alarm has been raised correctly in CloudWatch.

Go to *Services* > *CloudWatch*, and you should see an alarm in the *ALARM* status on the left hand side:



If you click on the alarm you'll see it's reporting the password has been changed.

Now all that's left to do is check your inbox for the notification: ✉

AWS Notifications <no-reply@sns.amazo… Sun, 26 Apr, 16:17 (16 hours ago)

to me

You are receiving this email because your Amazon CloudWatch Alarm "PasswordChangedAlarm" in the EU (Ireland) region has entered the ALARM state, because "Threshold Crossed: 1 datapoint [1.0 (26/04/20 15:13:00)] was greater than or equal to the threshold (1.0)." at "Sunday 26 April, 2020 15:17:38 UTC".

View this alarm in the AWS Management Console:
https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#s=Alarms&alarm=PasswordChangedAlarm

Alarm Details:
- Name: PasswordChangedAlarm
- Description:
- State Change: INSUFFICIENT_DATA -> ALARM
- Reason for State Change: Threshold Crossed: 1 datapoint [1.0 (26/04/20 15:13:00)] was greater than or equal to the threshold (1.0).
- Timestamp: Sunday 26 April, 2020 15:17:38 UTC
- AWS Account: 299404798587
- Alarm Arn: arn:aws:cloudwatch:eu-west-1:299404798587:alarm:PasswordChangedAlarm

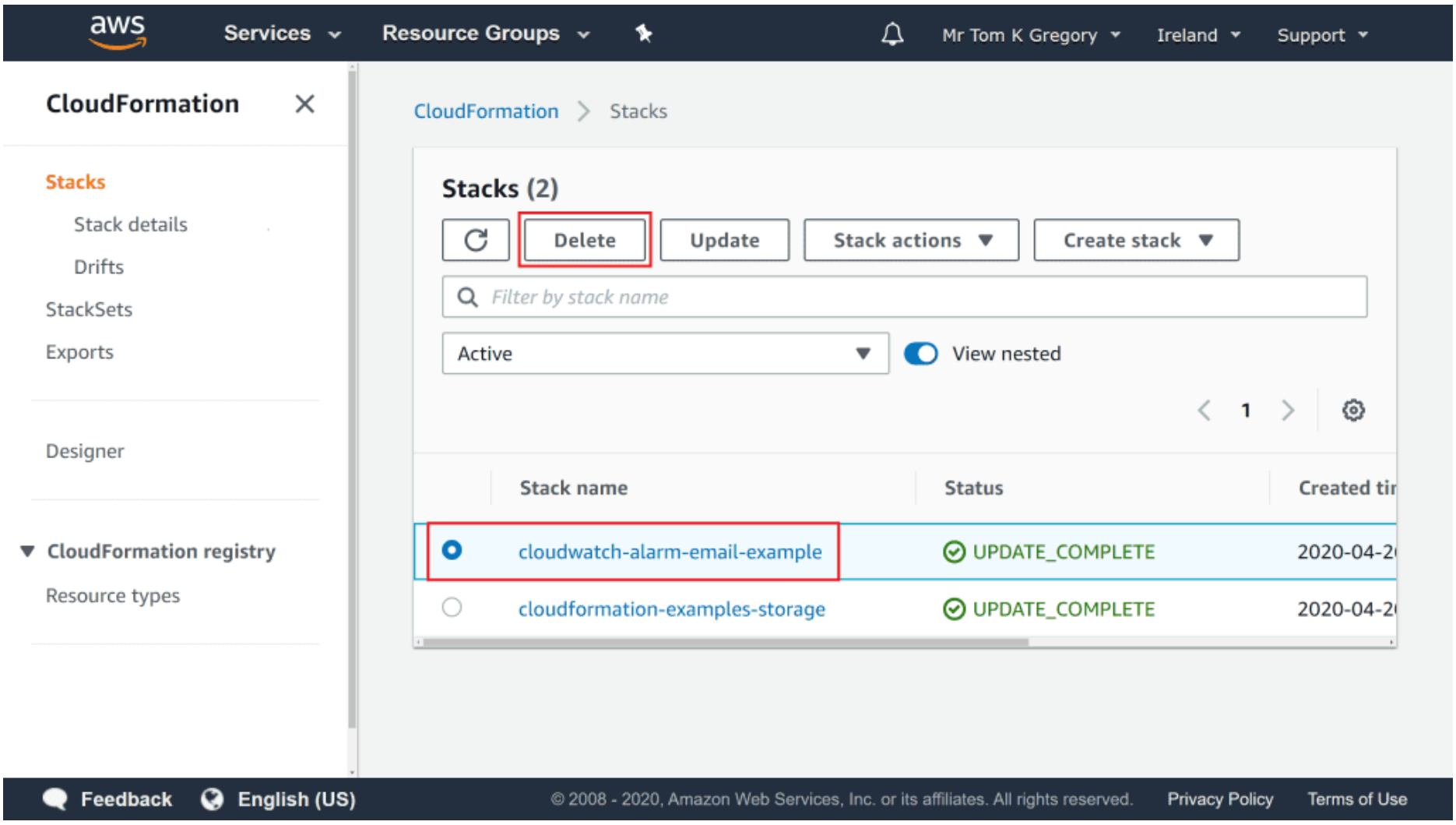
Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanOrEqualToThreshold 1.0 for 60 seconds.

All good! After a few minutes the alarm will go back to the OK status.

Clean up after yourself

Don't forget to delete your CloudFormation stack once you're done with it, to avoid incurring extra charges.

Go to *Services* > *CloudFormation*. Select the *cloudwatch-alarm-email-example* stack, then click *Delete*. On the confirmation dialog, select *Delete stack*:


aws
Services
Resource Groups
Mr Tom K Gregory
Ireland
Support
CloudFormation
Stacks
Stack details
Drifts
StackSets
Exports
Designer
CloudFormation registry
Resource types
CloudFormation > Stacks
Stacks (2)
Delete
Update
Stack actions
Create stack
Filter by stack name
Active
View nested
1
Stack name
Status
Created tir
cloudwatch-alarm-email-example
UPDATE_COMPLETE
2020-04-2
cloudformation-examples-storage
UPDATE_COMPLETE
2020-04-2
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

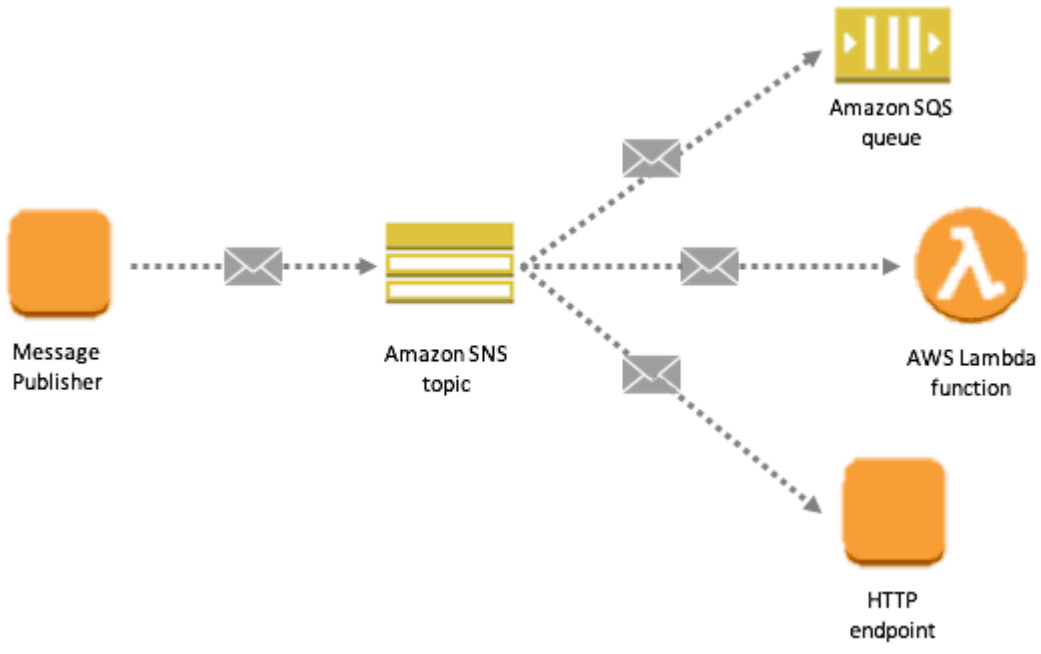
Simple Notification Services

Simple Notification Service is referred to as SNS. As a web service that coordinates and maintains the delivery and sending of messages to subscribing endpoints or clients, SNS is a quick, versatile, fully managed push notification service. It enables the delivery of a single message, a fan-out message, or other distributed services to a large number of recipients. SNS has push notification support as well. Simple Notification Service supports both First-in, First-out, and Standard services, and it has a unique encryption feature that enables you to save sensitive data in encrypted topics.

- **Topic**: Provide a name for the topic, which will appear as the topic's display name.
- The **subscribers** will receive the message that has been published on SNS subjects right away in this affordable pay-as-you-go model with an upfront cost.
- **Reliable –** Multiple AZs are used to keep at least three copies of the data in the same location.
- When you use autoscaling, it activates an SNS service that will send you an email to let you know that "**YOUR EC2 INSTANCE"** is expanding.
- Choose an endpoint type like HTTP OR HTTPS for the protocol.

Publisher: Messages are created or produced by publishers, also referred to as producers, and sent to the SNS, which serves as a logical access point.

Subscriber:– Web servers, email addresses, Amazon SQS queue, and AWS Lambda are a few examples of subscribers who get messages or notifications for SNS from (Amazon SQS, E-mail, Lamda, HTTPS, and SMS).



Types of Topics:-

Once a subject has been created, the topic type cannot be changed.

Standard:-

- Message ordering using best efforts.
- A message is transmitted at least once.
- Highest publish/second throughput.
- Subscription protocols include SQS, Lambda, HTTP, SMS, email, and endpoints for mobile applications.

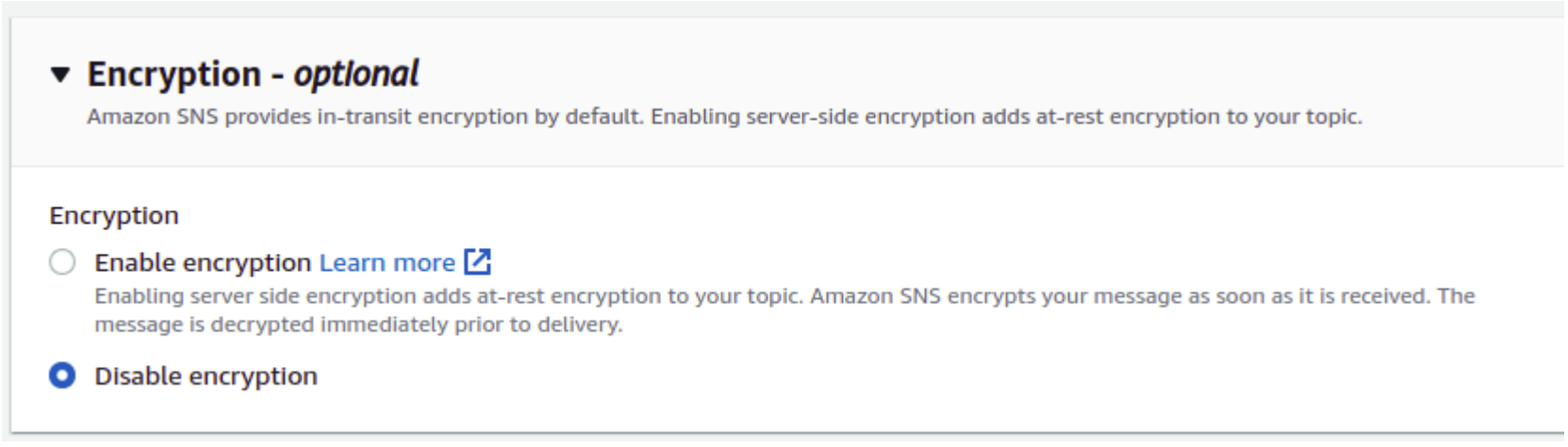
FIFO (First-in, First-out):-

- Ordering of strictly maintained messages
- delivery of messages only once.
- high rate, up to 300 publishes per second
- Subscription guidelines:
- SQS:-

Encryption:-

Enable Encryption:-You can save sensitive data in encrypted topics thanks to server-side encryption. Using a key controlled by the AWS key management service, SSE secures the contents of messages posted to Amazon SNS topics. Your communication is encrypted by Amazon SNS as soon as it is received, and it is promptly decrypted before being delivered.

Disable Encryption:- Disable the encryption by clicking on it if you don't want to use any kind of encryption.



Access Policy:-

BASIC:-

- A fundamental access policy is defined using uncomplicated criteria.
- Only the topic owner can publish the topic
- Everyone and Anybody can publish
- Only those AWS account IDs are allowed to publish the topic.

ADVANCED:-

- An advanced access policy is defined using a JSON object.
- The only person who can subscribe to a topic is its owner.
- Any AWS account and everyone can subscribe to the subject.
- Only the AWS account IDs mentioned may subscribe to the subject.
- Only users with specific endpoints may request.

```
{
 "Version": "2008-10-17",
 "Id": "__default_policy_ID",
 "Statement": [
  {
   "Sid": "__default_statement_ID",
   "Effect": "Allow",
   "Principal": {
    "AWS": "*"
   },
   "Action": [
    "SNS:Publish",
    "SNS:RemovePermission",
    "SNS:SetTopicAttributes",
    "SNS:DeleteTopic",
    "SNS:ListSubscriptionsByTopic",
    "SNS:GetTopicAttributes",
    "SNS:AddPermission",
    "SNS:Subscribe"
   ],
  "Resource": "arn:aws:sns:us-east-1:141837438119:my test",
  "Condition": {
    "StringEquals": {
     "AWS:SourceOwner": "141837438119"
```

```
      }
    }
  }
 ]
}
```


Delivery retry policy (HTTP/HTTPS):-

- This policy outlines the procedures for Amazon SNS's unsuccessful HTTP/S endpoint delivery retries.
- Sometimes multiple tries are necessary for an Amazon SNS delivery to an HTTP/HTTPS endpoint to be successful.
- The number of retries, the delay time, and the retry-backoff mechanism are all configured for you by Amazon SNS when you utilise the delivery retry policy default settings.

In JSON


```
{
 "http": {
  "defaultHealthyRetryPolicy": {
   "numRetries": 3,
   "numNoDelayRetries": 0,
   "minDelayTarget": 20,
   "maxDelayTarget": 20,
   "numMinDelayRetries": 0,
   "numMaxDelayRetries": 0,
   "backoffFunction": "linear"
  },
  "disableSubscriptionOverrides": false
 }
}
```


Delivery status logging:-

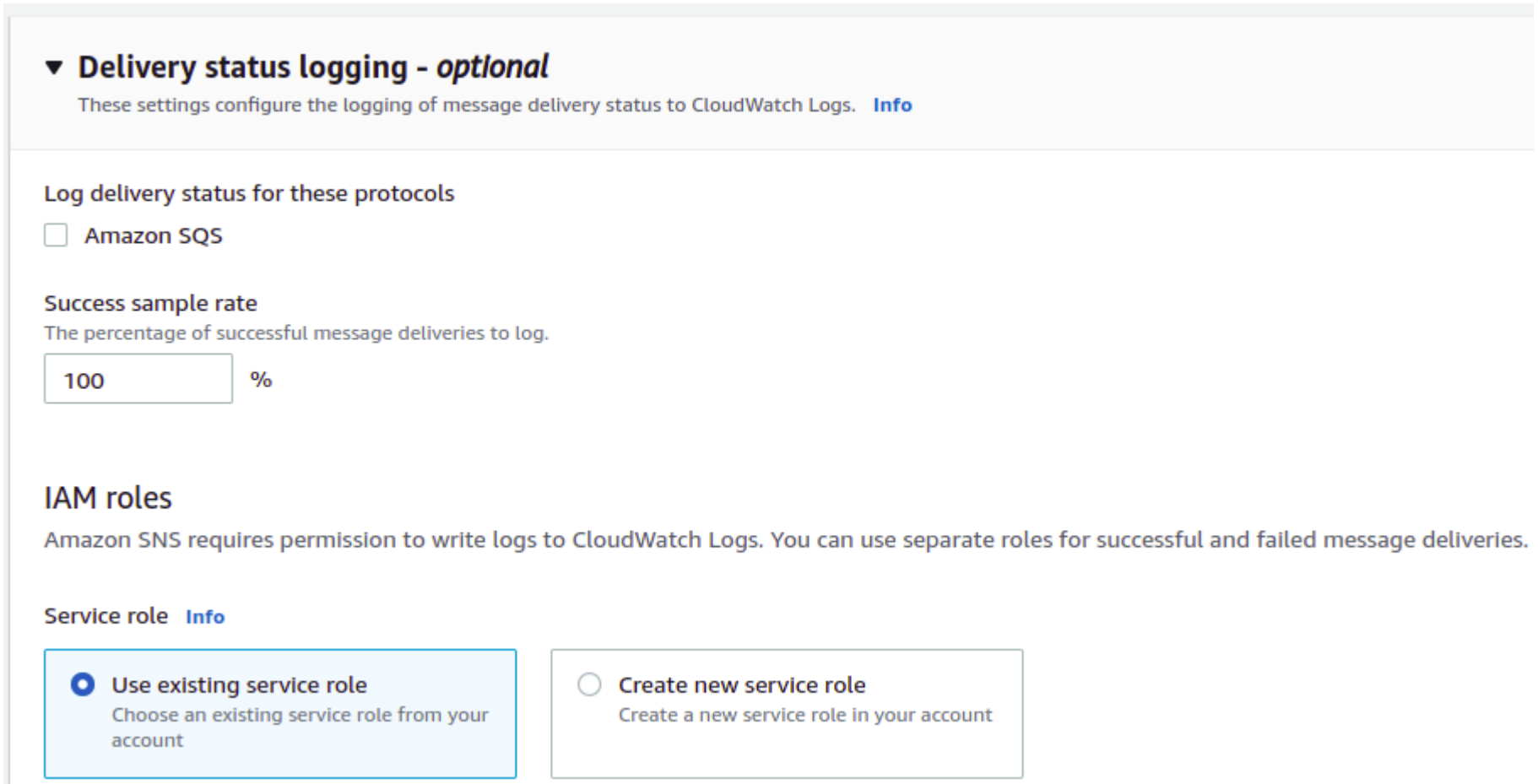
The following Amazon SNS endpoints are supported by Amazon SNS for logging the delivery status of notification messages issued to topics:

- HTTP
- Lambda
- Amazon SQS
- Application
- Platform application endpoint
- Amazon Kinesis Data Firehose.

You can use it as IAM roles in this case just as you can employ different roles for **successful and unsuccessful message deliveries.** given that there are two categories:-

Use an existing service role – For this, we must choose an existing service role from your account.

Create a new service role -Your account needs a new service role to be created in order to accomplish this.



Support Push Notification Platform:

Subscribers to SNS Topics can come from any push notification platform that is supported as well as from other endpoint types like SMS or email. Several platforms are:

- Amazon Device Messaging
- Google Device Messaging
- Google Cloud Messaging
- Windows Push Notification Service

Amazon SNS Alternative:-

- Amazon Kinesis Data Stream
- Aws Managed queue Service
- Apache Kafka
- Twilio
- Pusher

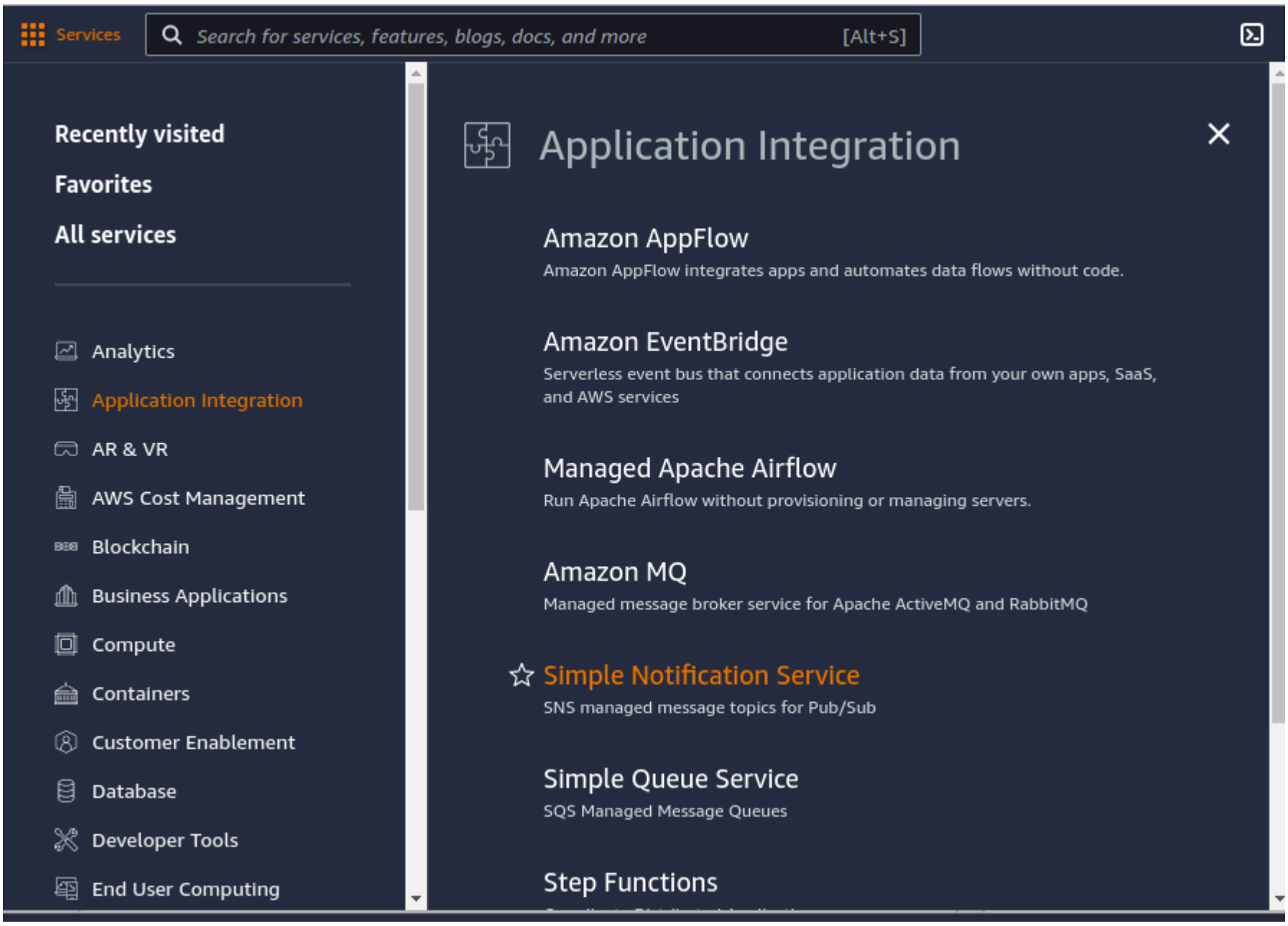
Amazon SNS Pricing:-

- **Publish Action:**-A request payload of 256 KB will be charged as four requests because each request payload of 64 KB counts as one request.
- **Mobile Push Notification:**-0.50/Millions Request
- **E-Mail:**– $2/100,000
- **SMS**:- Price Depend on the Country
- **HTTPS Notific**ation :- $ 60 /Million

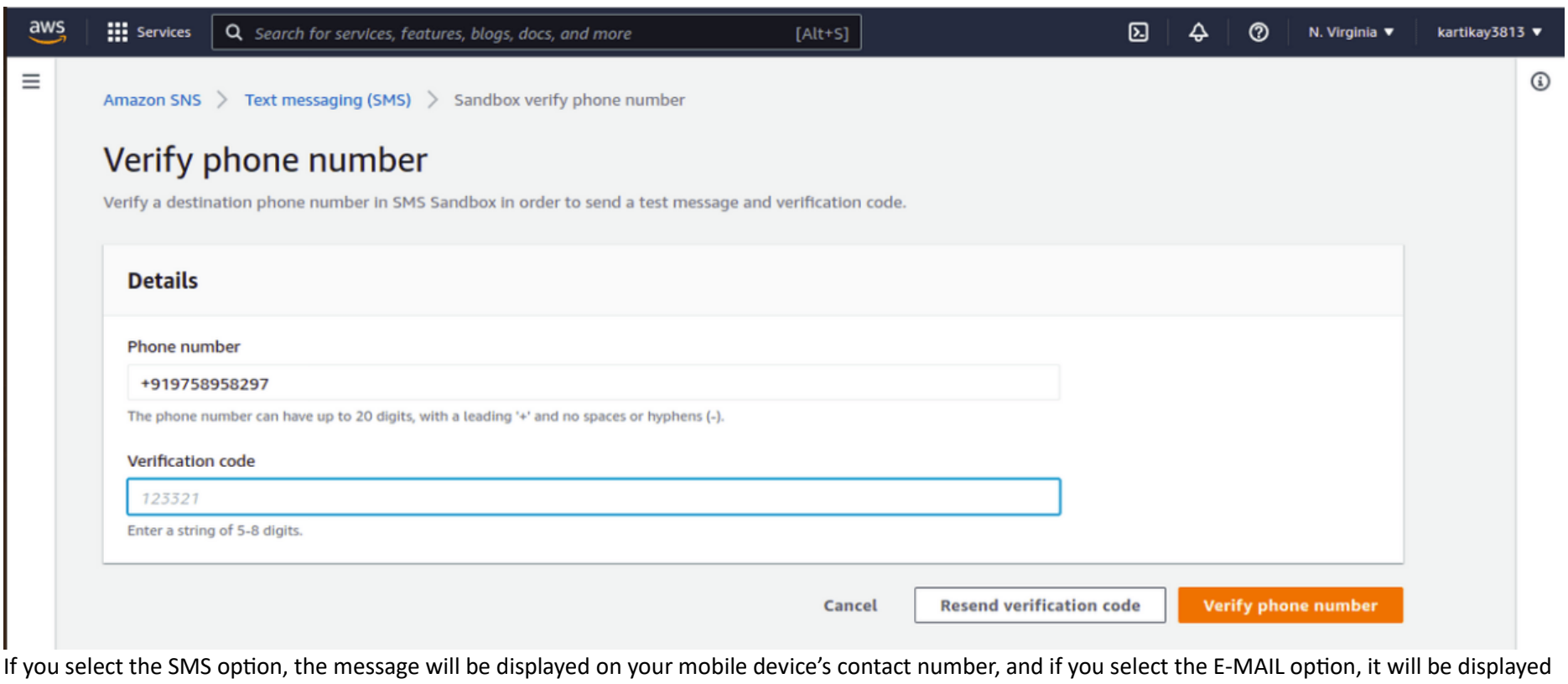
How does it Work?

- The console
- Select Simple Notification Service by clicking on services.
- Develop Topic -> Instance: (Covid caution) (Lack of Education) (Lack of Education)
- Go to Subscription and select the method for sharing messages (E-mail, Phone, etc)
- If you decide to email someone, be sure to include their email address.
- Select “send”
- Whatever message was put in the email that was sent to the mail ID was received.
- Visit Mail and choose CONFIRM.

First, pick SIMPLE NOTIFICATION SERVICE from the AWS dashboard.



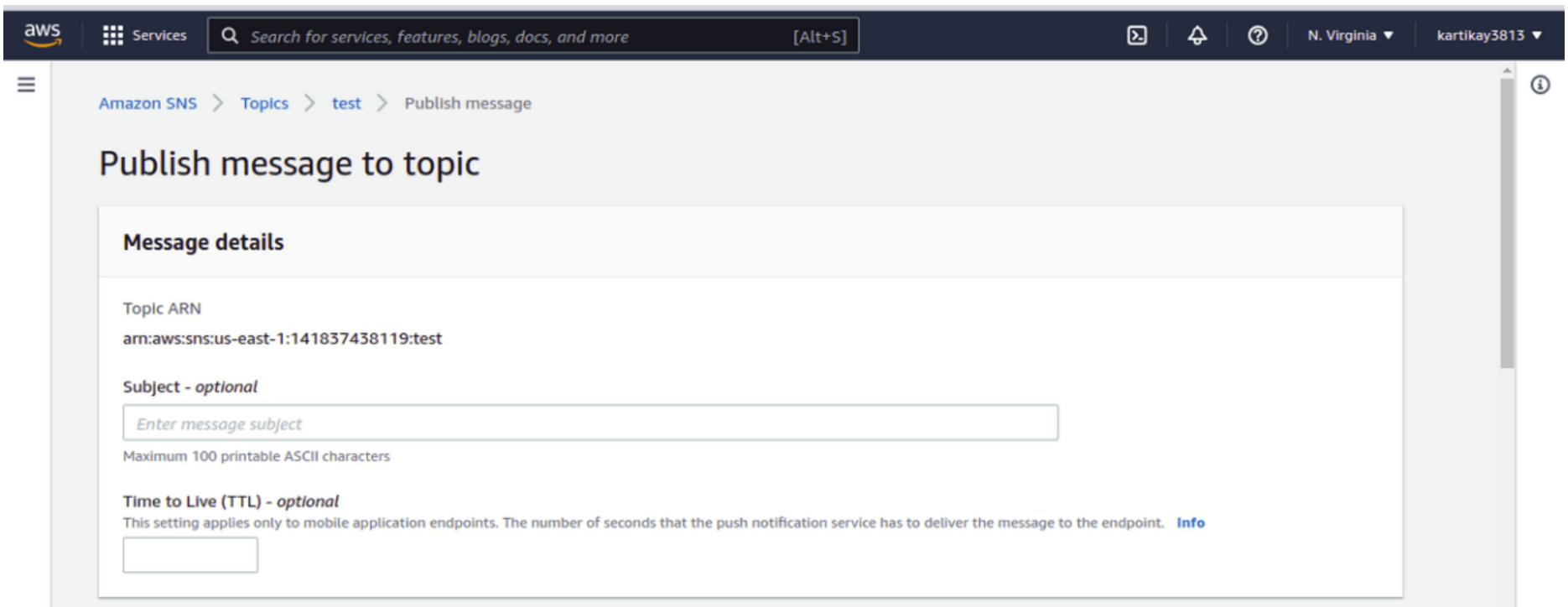
After selecting an SNS Simply create a topic, type your message, and then pick whether to send it via email or SMS. For eg: – A **Covid warning** will appear with the phrase "**Follow social distance.**"



If you select the SMS option, the message will be displayed on your mobile device's contact number, and if you select the E-MAIL option, it will be displayed on your email address.

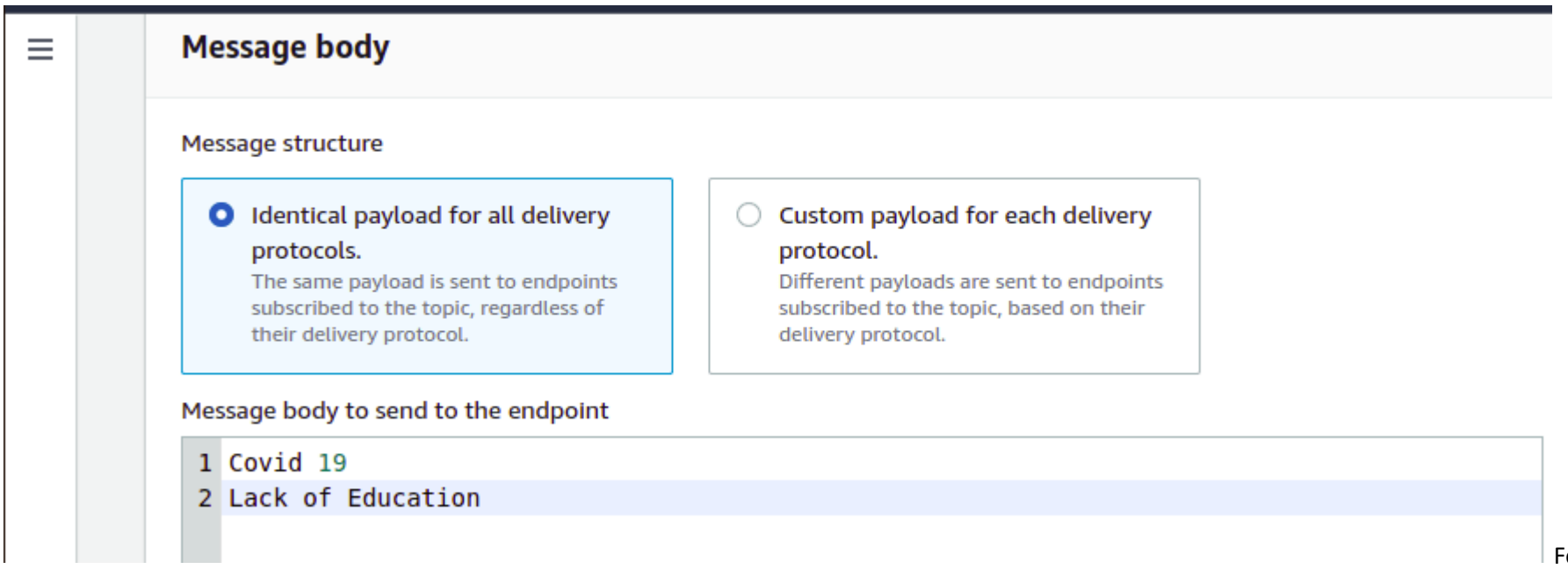
Publish message to topic:-

Navigate to Amazon SNS > Topics > Test > Create Message



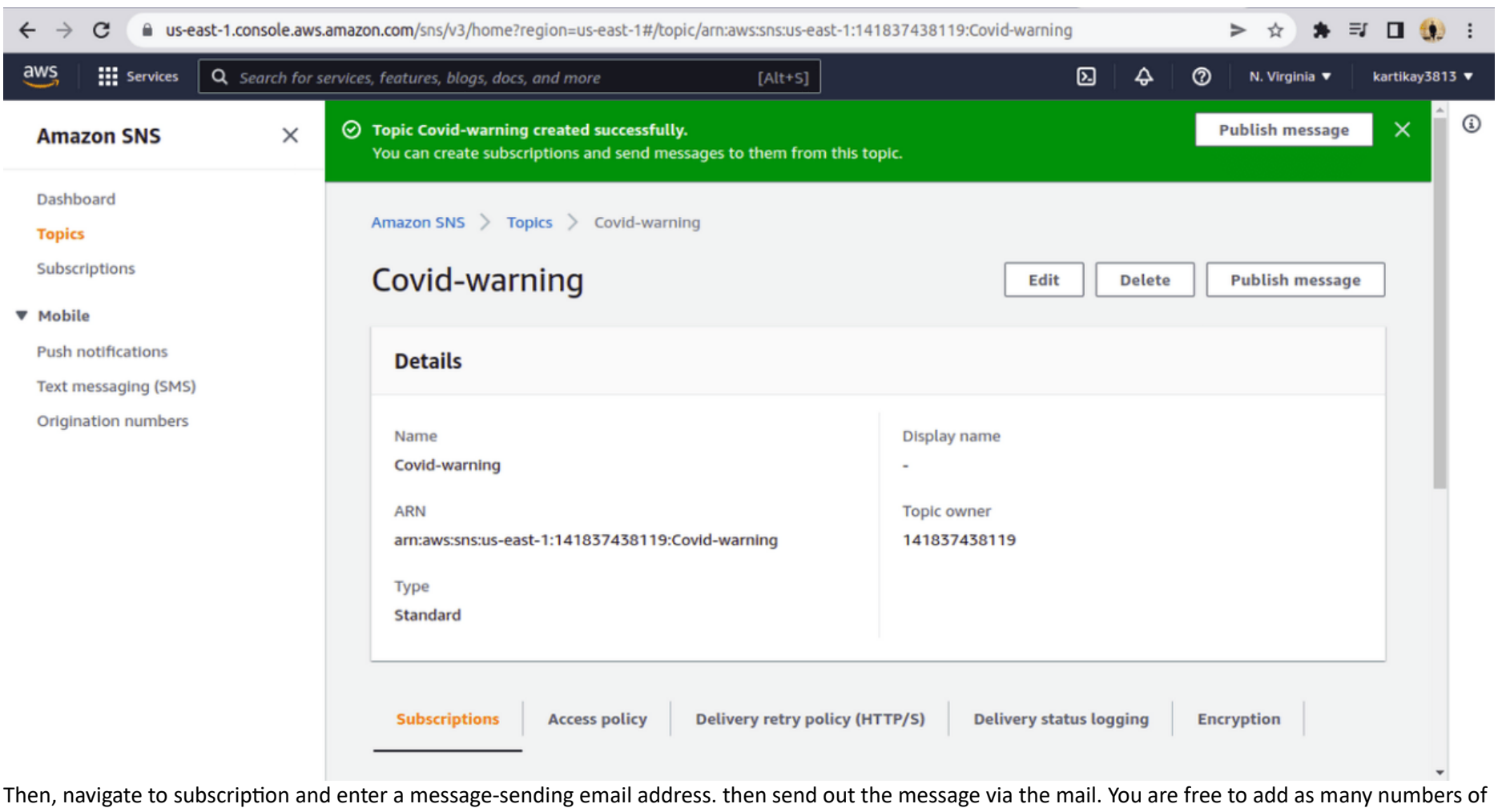
Select The subject **ARN** and then click on the **Subject.** The message will then be displayed, and if you wish to select the topic ARN, do so.
TTL (**Time To Live)**:- The number of seconds that the push notification service has to deliver the message to the endpoint.

Once it's finished, simply type the message.



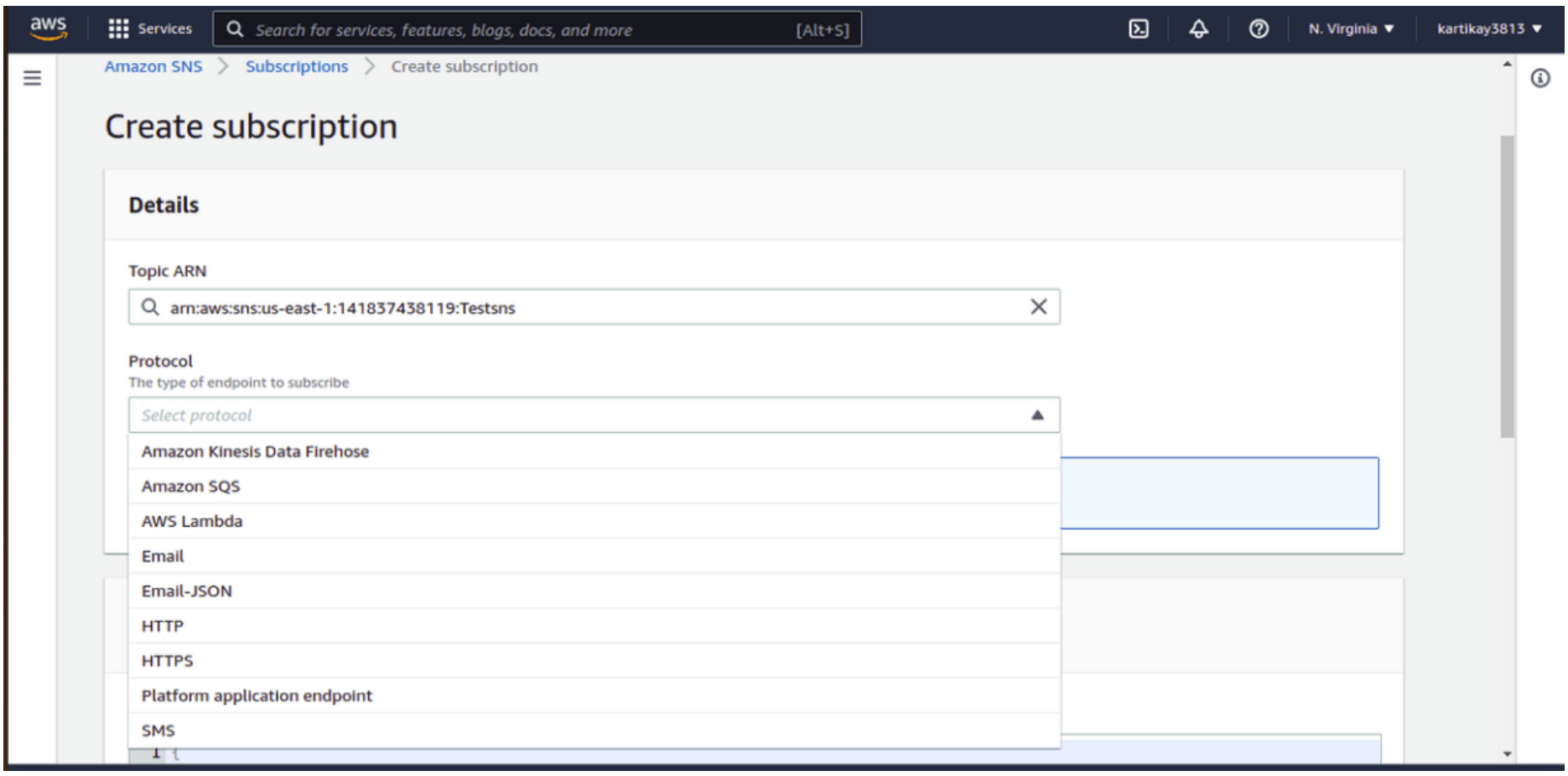
For same payload for all delivery protocols and custom payload for each delivery protocol, the message must be entered into the message body.

After typing the message in the box labeled "what to send," click "publish message."



Then, navigate to subscription and enter a message-sending email address. then send out the message via the mail. You are free to add as many numbers of emails as you like and send them all at once.

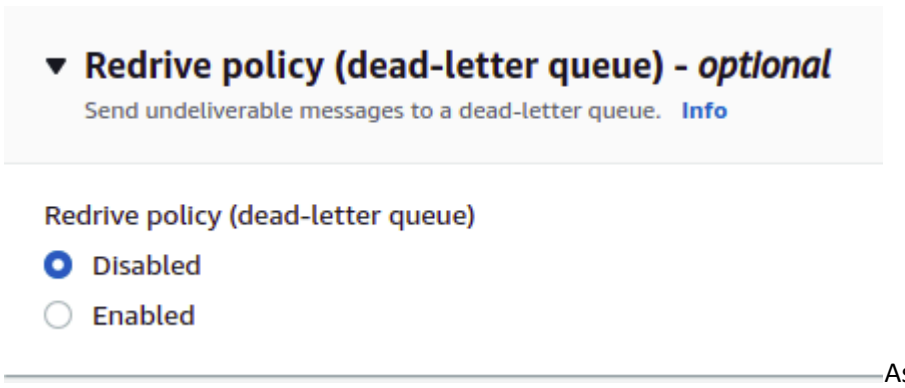
Subscription:-



After selecting the topic, select the protocol that will be used to subscribe to the desired endpoint type. Then, as it filters the messages a subscriber receives, you can select the subscription policy in it.

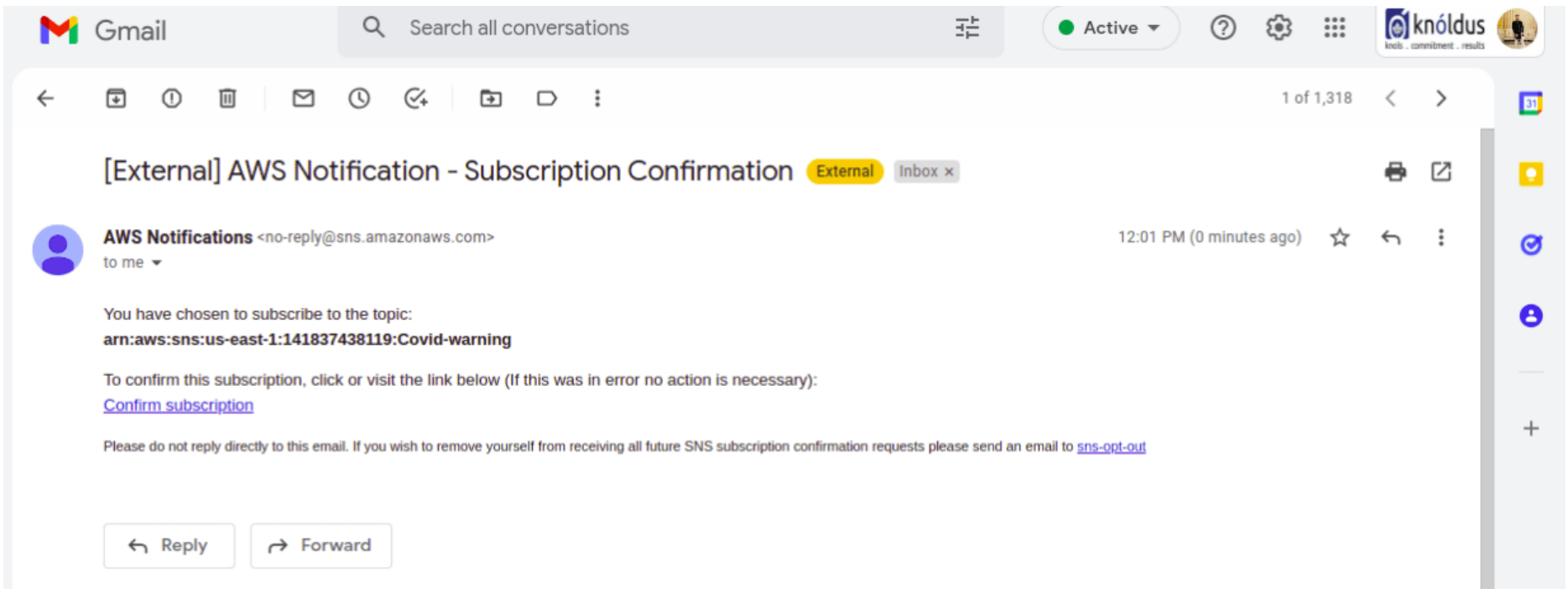
Redrive policy (dead-letter queue):-

By defining the Amazon SQS queue that holds messages that are unsuccessfully delivered to subscribers, we can apply a Redrive policy to Amazon SNS subscriptions.

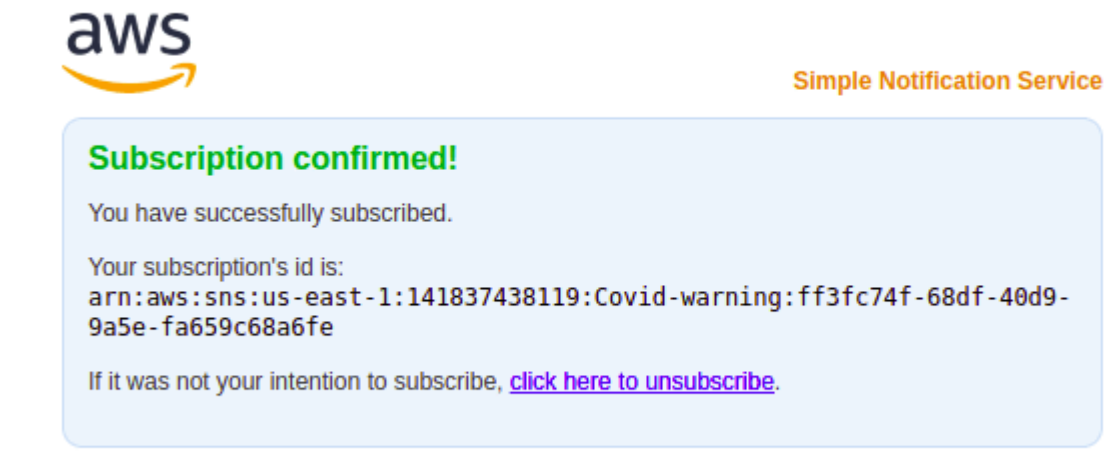


As a result, we have a policy that allows both Enabled and Disabled.

Just send the Mail once the entire process has been done.



When the procedure is complete, open the email you received and click the **Confirm subscription button.**



After receiving subscription confirmation, a message indicating the preferred method will be displayed in the email address. This method can either be done via email or mobile phone.

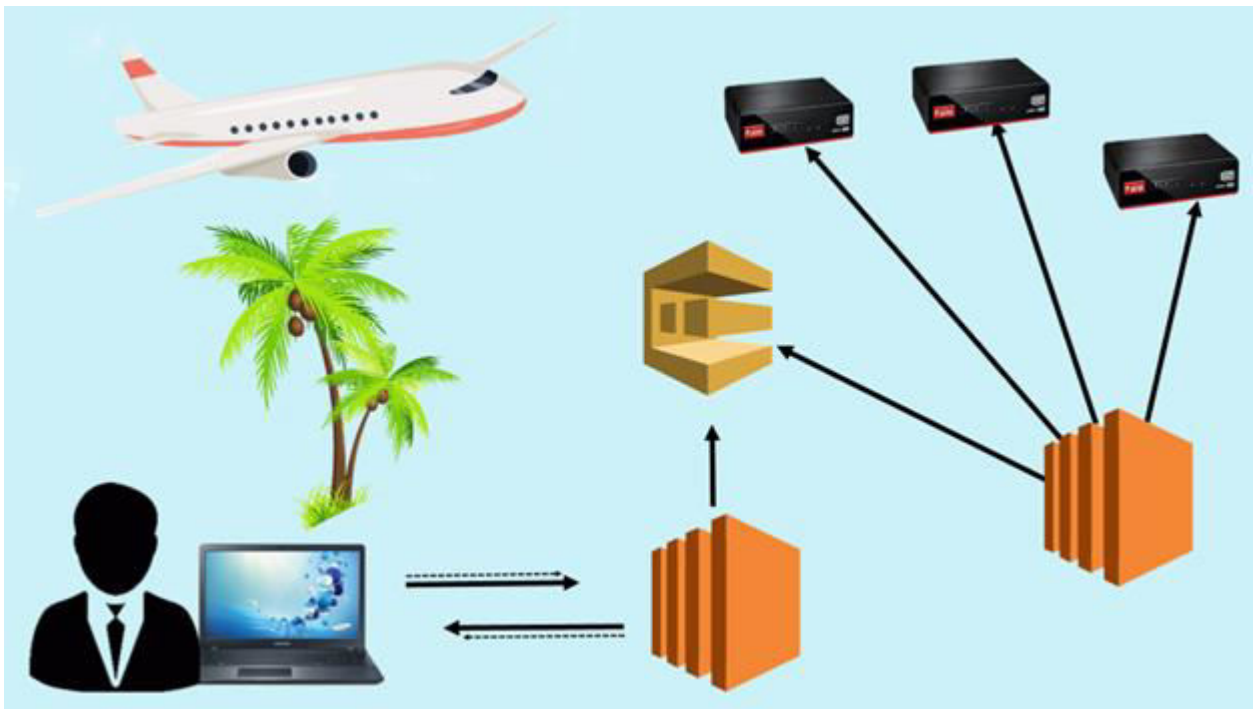
Simple Queue Service

- SQS stands for **Simple Queue Service**.
- SQS was the first service available in AWS.
- Amazon SQS is a web service that gives you access to a message queue that can be used to store messages while waiting for a computer to process them.
- Amazon SQS is a distributed queue system that enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component where a queue is a temporary repository for messages that are awaiting processing.
- With the help of SQS, you can send, store and receive messages between software components at any volume without losing messages.
- Using Amazon sqs, you can separate the components of an application so that they can run independently, easing message management between components.
- Any component of a distributed application can store the messages in the queue.
- Messages can contain up to 256 KB of text in any format such as json, xml, etc.
- Any component of an application can later retrieve the messages programmatically using the Amazon SQS API.
- The queue acts as a buffer between the component producing and saving data, and the component receives the data for processing. This means that the queue resolves issues that arise if the producer is producing work faster than the consumer can process it, or if the producer or consumer is only intermittently connected to the network.
- If you got two EC2 instances which are pulling the SQS Queue. You can configure the autoscaling group if a number of messages go over a certain limit. Suppose the number of messages exceeds 10, then you can add additional EC2 instance to process the job faster. In this way, SQS provides elasticity.

Let's understand through an example.

Let's look at a website that generates a Meme. Suppose the user wants to upload a photo and wants to convert into Meme. User uploads a photo on a website and website might store a photo in s3. As soon as it finished uploads, it triggers a Lambda function. Lambda analyzes the data about this particular image to SQS, and this data can be "what the top of the meme should say", "what the bottom of the meme should say", the location of the S3 bucket, etc. The data sits inside the SQS as a message. An EC2 instance looks at the message and performs its job. An EC2 instance creates a Meme and stores it in S3 bucket. Once the EC2 instance completed its job, it moves back to the SQS. The best thing is that if you lose your EC2 instance, then also you would not lose the job as the job sits inside the S3 bucket.

Let's look at another example of SQS, i.e., Travel Website.

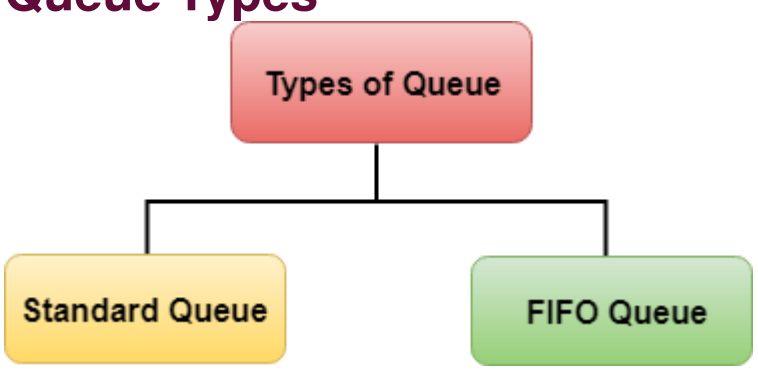


Suppose the user wants to look for a package holiday and wants to look at the best possible flight. AUser types a query in a browser, it then hits the EC2 instance. An EC2 instance looks "What the user is looking for?", it then puts the message in a queue to the SQS. An EC2 instance pulls queue. An EC2 instance continuously pulling the queue and looking for the jobs to do. Once it gets the job, it then processes it. It interrogates the Airline service to get all the best possible flights. It sends the result to the web server, and the web server sends back the result to the user. A User then selects the best flight according to his or her budget.

If we didn't have SQS, then what happened?

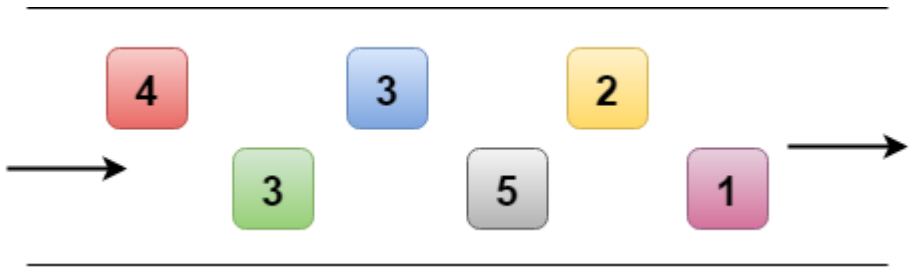
A web server passes the information to an application server and then application server queried an Airline service. If an Application server crashes, then a user loses its query. One of the great thing about SQS is that data is queued in the SQS even if the application server crashes, the message in the queue is marked as an invisible in a timeout interval window. When the timeout runs out, message reappears in the queue; then a new EC2 instance can use this message to perform its job. Therefore, we can say that SQS removes the application server dependency.

Queue Types



There are two types of Queue:

- **Standard Queues (default)**
- **FIFO Queues (First-In-First-Out)**
- **Standard Queue**



- SQS offers a standard queue as the default queue type.
- It allows you to have an unlimited number of transactions per second.
- It guarantees that a message is delivered at least once. However, sometime, more than one copy of a message might be delivered out of order.
- It provides best-effort ordering which ensures that messages are generally delivered in the same order as they are sent but it does not provide a guarantee.

- **FIFO Queue**

- The FIFO Queue complements the standard Queue.
- It guarantees ordering, i.e., the order in which they are sent is also received in the same order.
- The most important features of a queue are FIFO Queue and exactly-once processing, i.e., a message is delivered once and remains available until consumer processes and deletes it.
- FIFO Queue does not allow duplicates to be introduced into the Queue.
- It also supports message groups that allow multiple ordered message groups within a single Queue.
- FIFO Queues are limited to 300 transactions per second but have all the capabilities of standard queues.

SQS Visibility Timeout

- The visibility timeout is the amount of time that the message is invisible in the SQS Queue after a reader picks up that message.
- If the provided job is processed before the visibility time out expires, the message will then be deleted from the Queue. If the job is not processed within that time, the message will become visible again and another reader will process it. This could result in the same message being delivered twice.
- The Default Visibility Timeout is 30 seconds.
- Visibility Timeout can be increased if your task takes more than 30 seconds.
- The maximum Visibility Timeout is 12 hours.

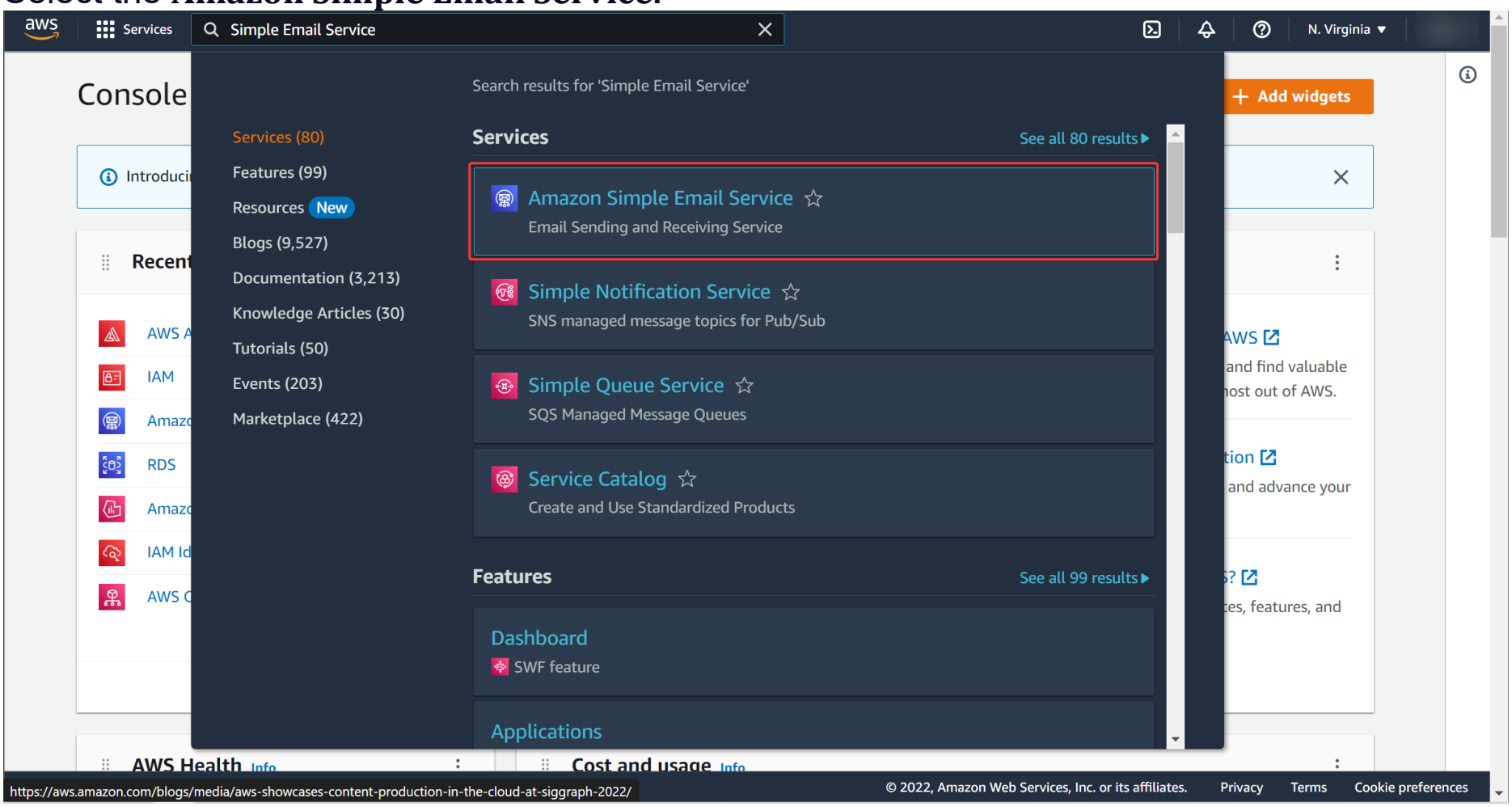
Important points to remember:

- SQS is pull-based, not push-based.
- Messages are 256 KB in size.
- Messages are kept in a queue from 1 minute to 14 days.
- The default retention period is 4 days.
- It guarantees that your messages will be processed at least once.

Simple Email Service

Step 1 – Verify Identities

First, login into your AWS Management Console account and search for Simple Email Service. Select the **Amazon Simple Email Service.**

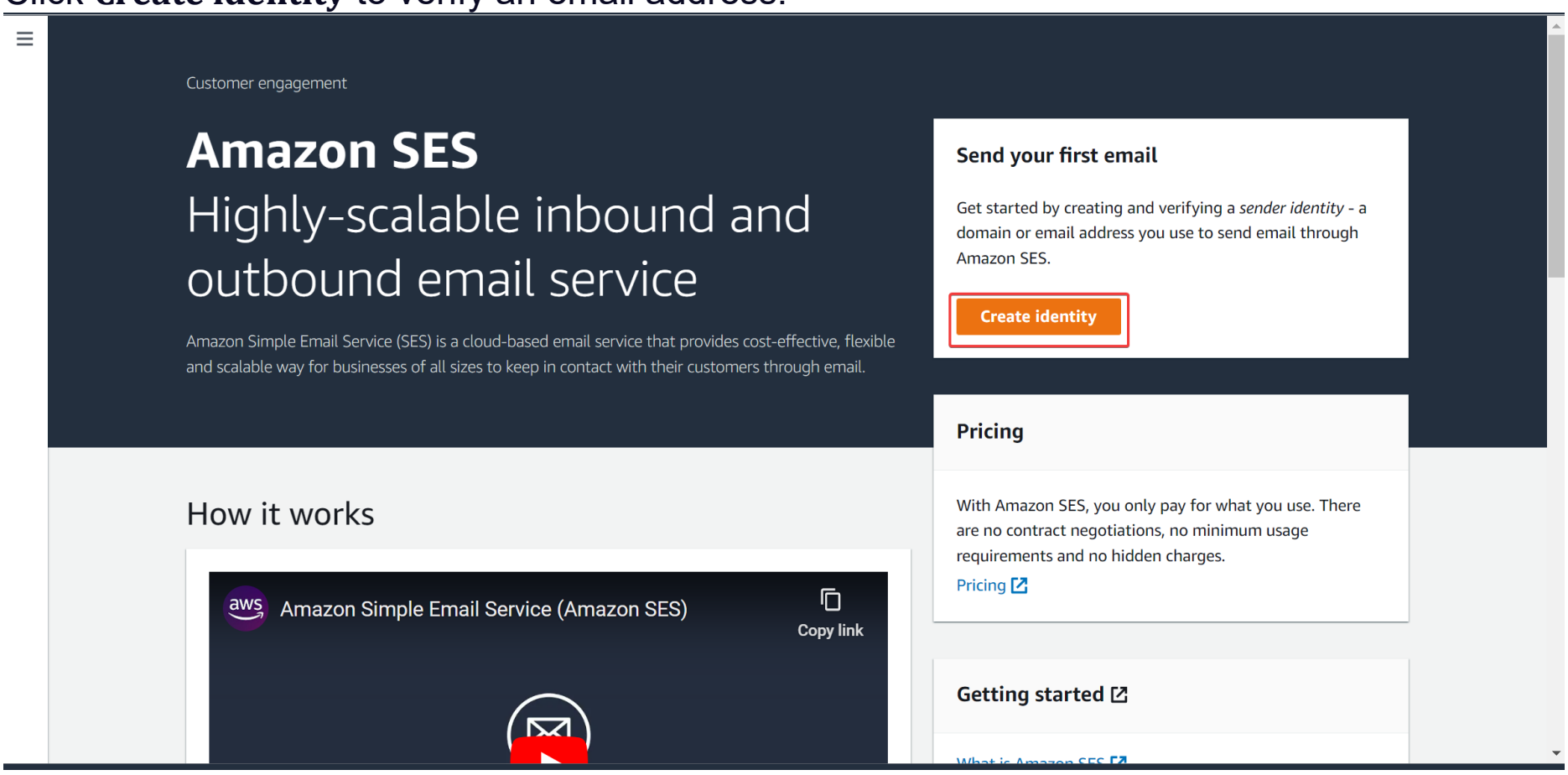


Searching for AWS SES

This will lead you to an SES Console.

To start sending emails, you'll need to create an identity. This involves verifying the email address you would use to send emails. If you do not verify the email address, you can't use the email to perform any action on SES.

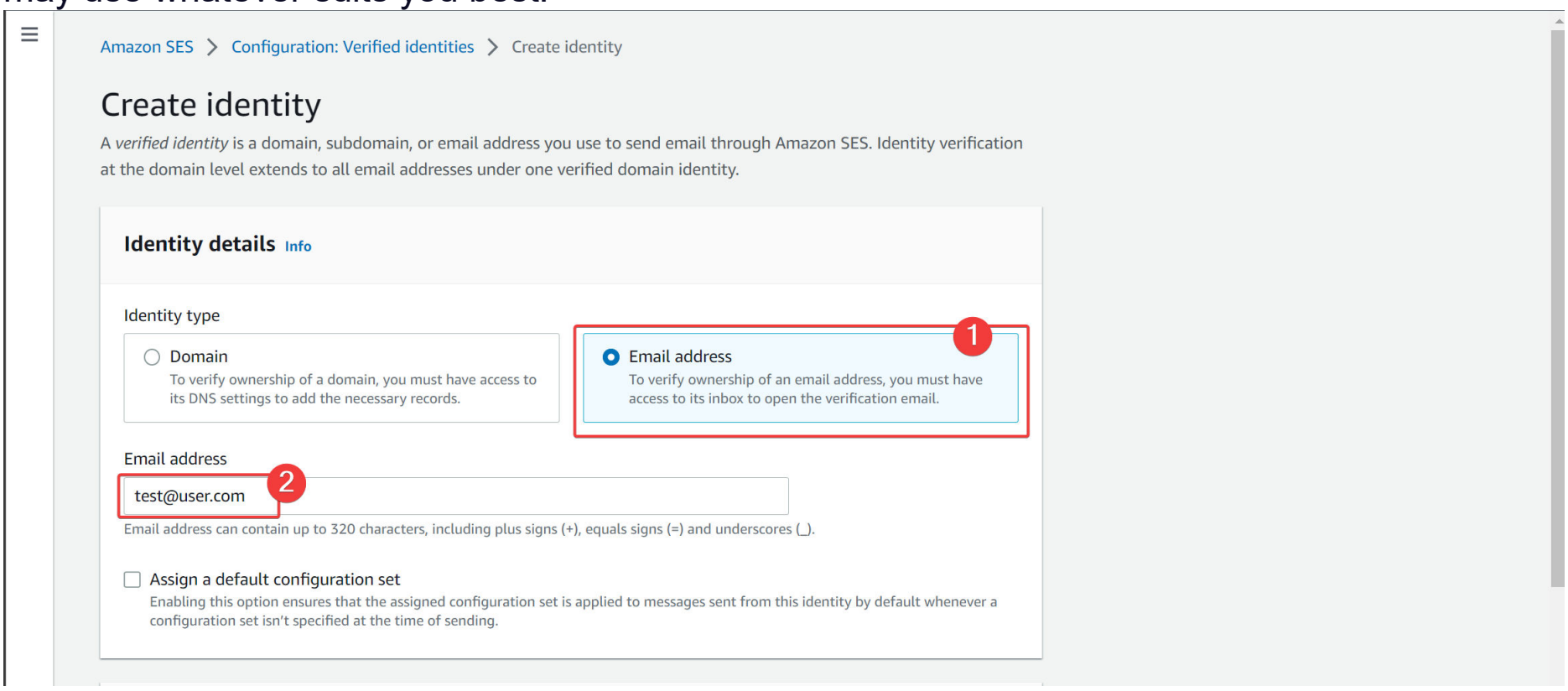
Note you can add a domain as an identity, but we'll use an email address for this guide. Click **Create identity** to verify an email address.



Creating an Identity

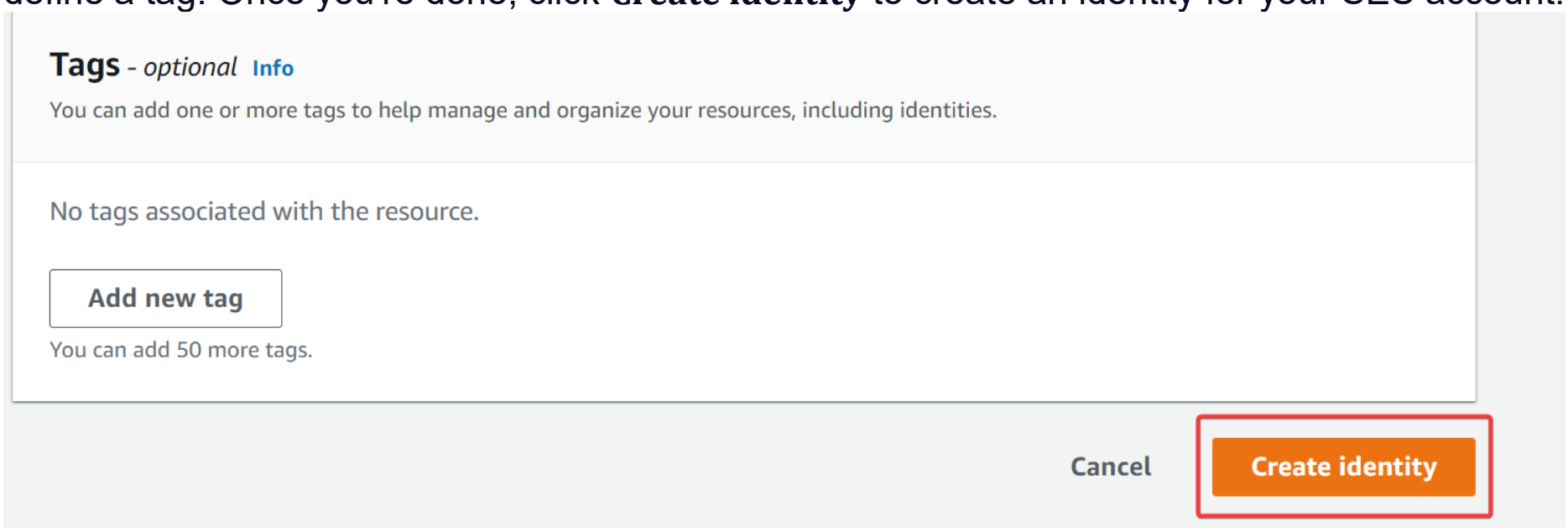
Next, select the **Email address** option and enter the email address you wish to use.

In Amazon SES, you can use a domain, subdomain, or email address as a *verified* identity. You may use whatever suits you best.



Verifying an Identity

We use tags to manage identities on Amazon SES. We'll skip this here, but if you wish, you can define a tag. Once you're done, click **Create identity** to create an identity for your SES account.



Creating an Identity for Amazon SES

Now, an email will be sent to the email address you used to create the identity. Click the link in the email to verify your email.

Amazon Web Services <no-reply-aws@amazon.com>
To [redacted]

Reply Reply all Forward Delete

AS

Dear Amazon Web Services Customer,

We have received a request to authorize this email address for use with Amazon SES and Amazon Pinpoint in region US East (N. Virginia). If you requested this verification, please go to the following URL to confirm that you are authorized to use this email address:

[redacted]

Your request will not be processed unless you confirm the address using this URL. This link expires 24 hours after your original verification request.

If you did NOT request to verify this email address, do not click on the link. Please note that many times, the situation isn't a phishing attempt, but either a misunderstanding of how to use our service, or someone setting up email-sending capabilities on your behalf as part of a legitimate service, but without having fully communicated the procedure first. If you are still concerned, please forward this notification to aws-email-domain-verification@amazon.com and let us know in the forward that you did not request the verification.

To learn more about sending email from Amazon Web Services, please refer to the Amazon SES Developer Guide at http://docs.aws.amazon.com/ses/latest/DeveloperGuide/Welcome.html and Amazon Pinpoint Developer Guide at http://docs.aws.amazon.com/pinpoint/latest/userguide/welcome.html.

Verifying your email address

Once you've done that, you will see your email address on your SES account's list of verified identities.

Amazon SES > Configuration: Verified identities

Verified identities

A *verified identity* is a domain, subdomain, or email address you use to send email through Amazon SES. Learn more

New identity status update
The **Identity status** now represents the explicit verification of the identity itself. For the domain identities this means verifying ownership through updates in the DNS records, and for the email address identities, this means opening the verification email from *no-reply-aws@amazon.com* and selecting the link to complete the verification process. Learn more

Identities (2) Info

Send test email | Delete | Create identity

Search all domain and email address identities

1

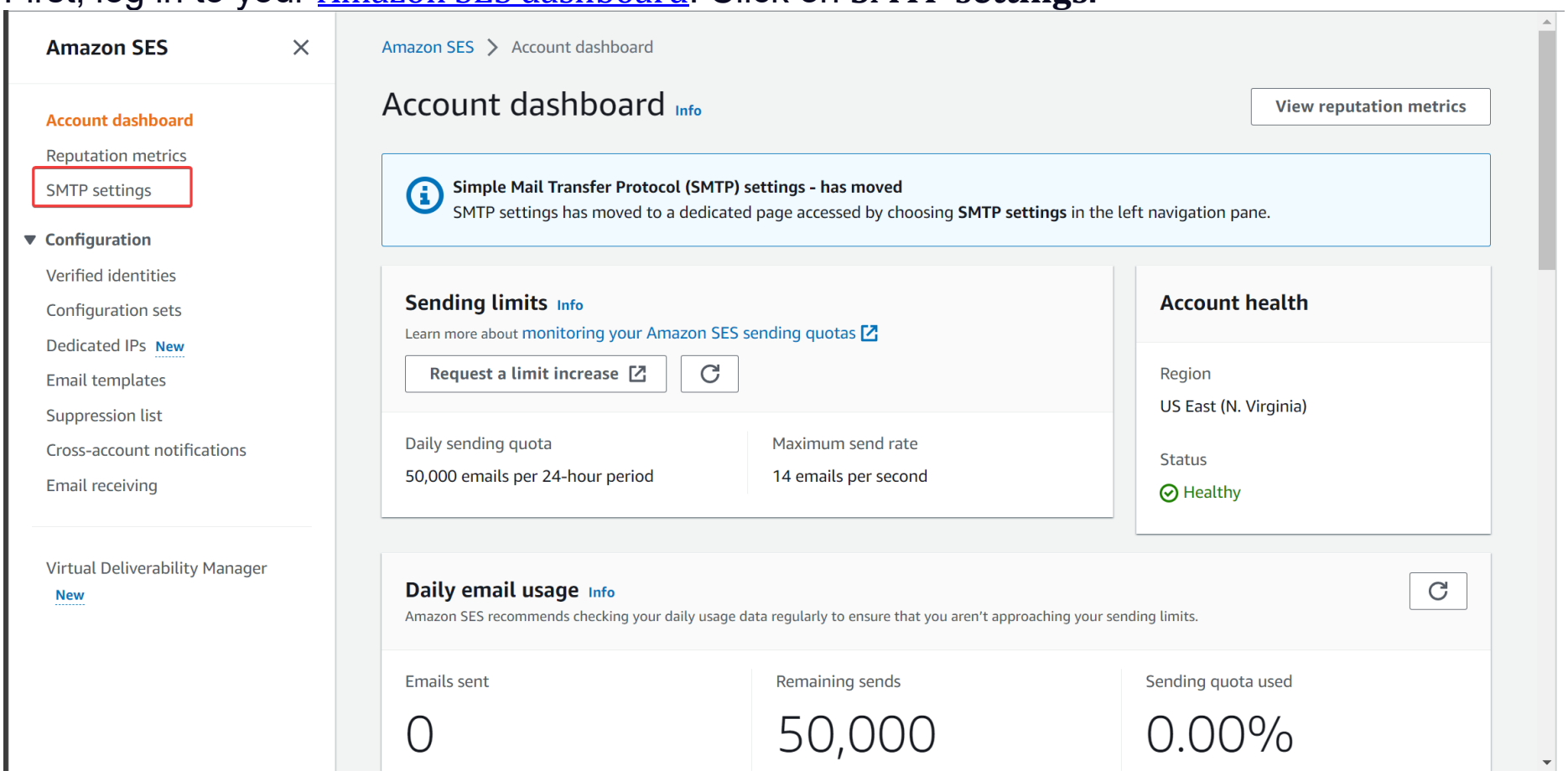
Identity	Identity type	Identity status
[redacted]	Email address	Verified
[redacted]	Email address	Verified

List of verified identities

How to Create SMTP Credentials

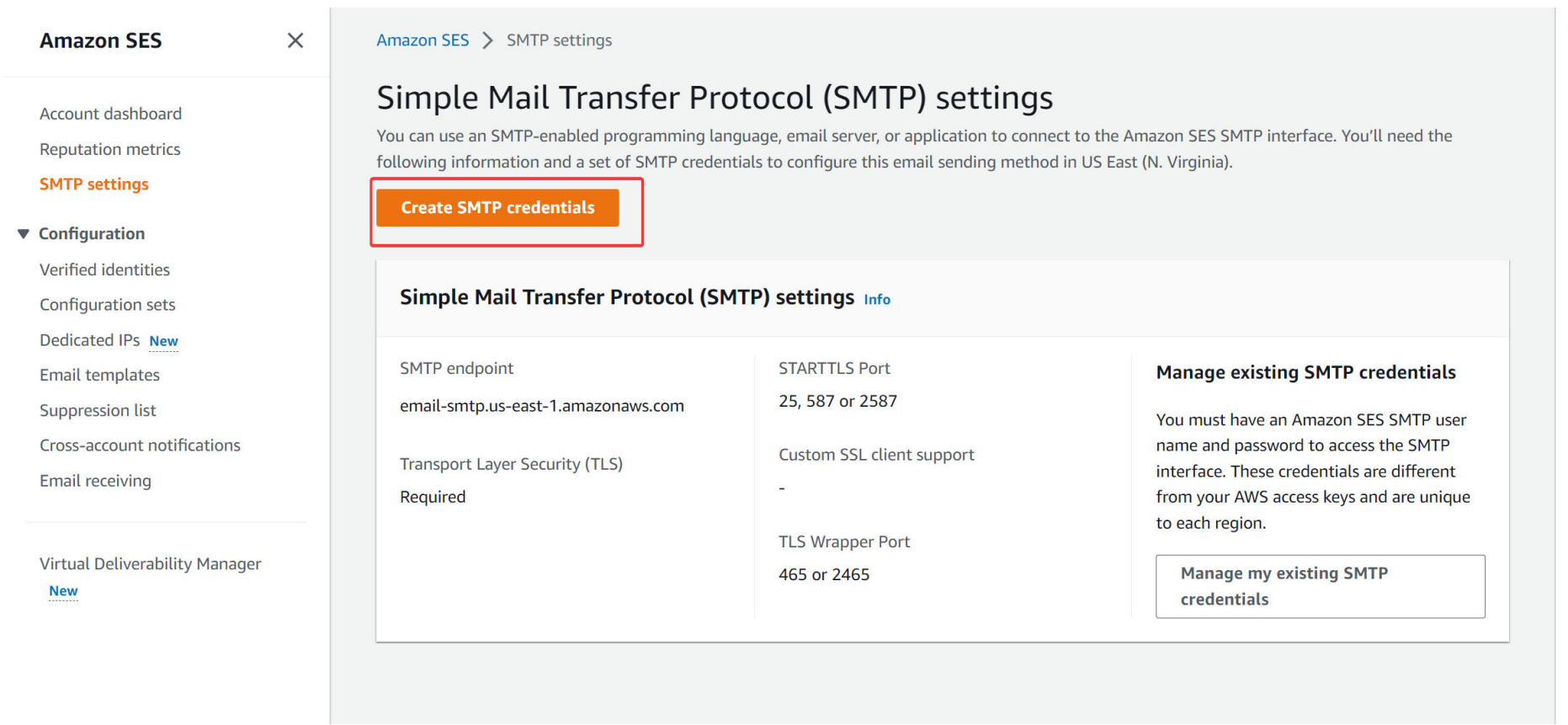
A Simple Mail Transfer Protocol (SMTP) sends and receives messages through a mail server. In this section, you will learn how to create credentials that grant you access to the SES mail server to send and receive mail.

First, log in to your Amazon SES dashboard. Click on **SMTP settings.**



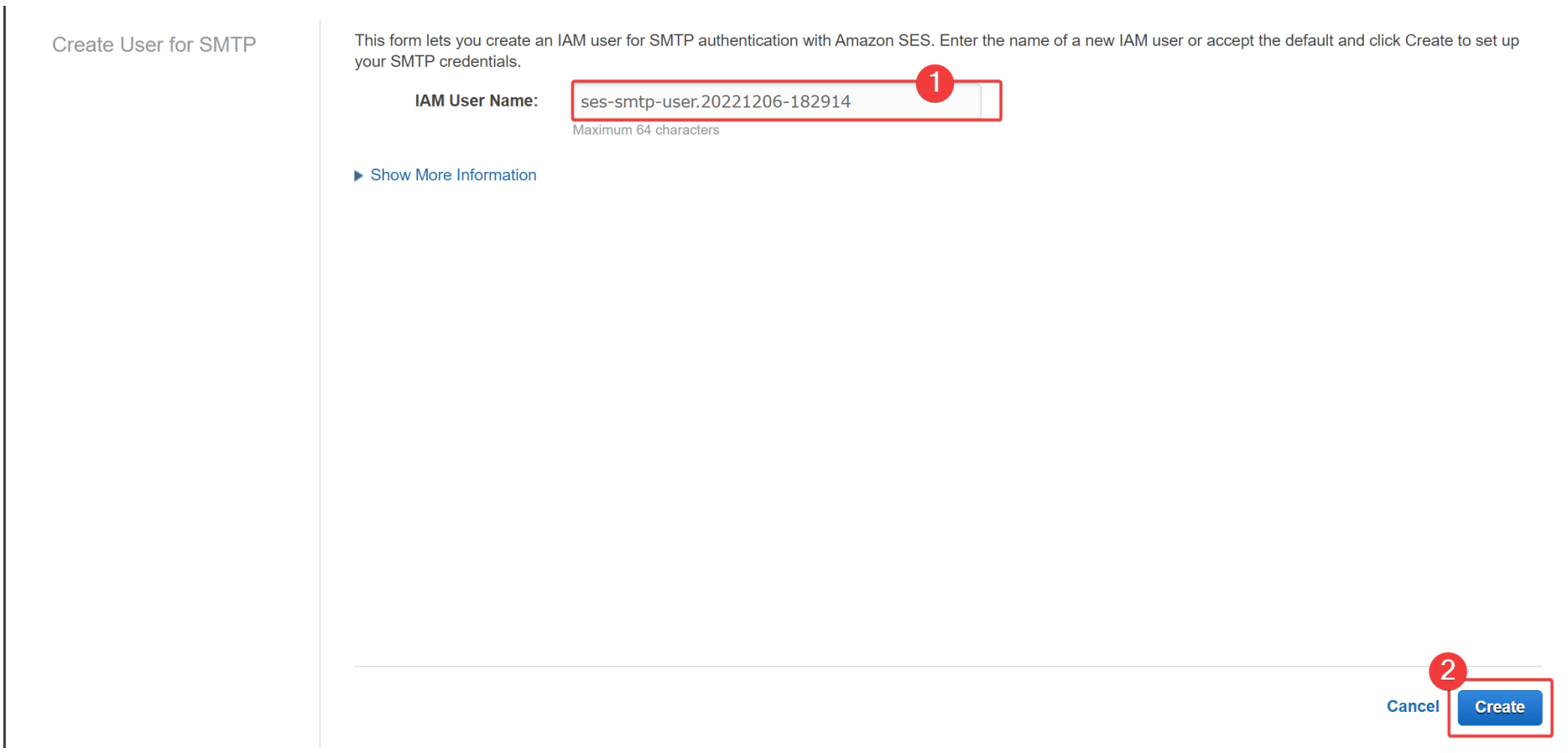
Amazon SES Dashboard

Then click on **Create SMTP credentials** to create login details to your SMTP account under Amazon SES.



Creating SMTP Credentials

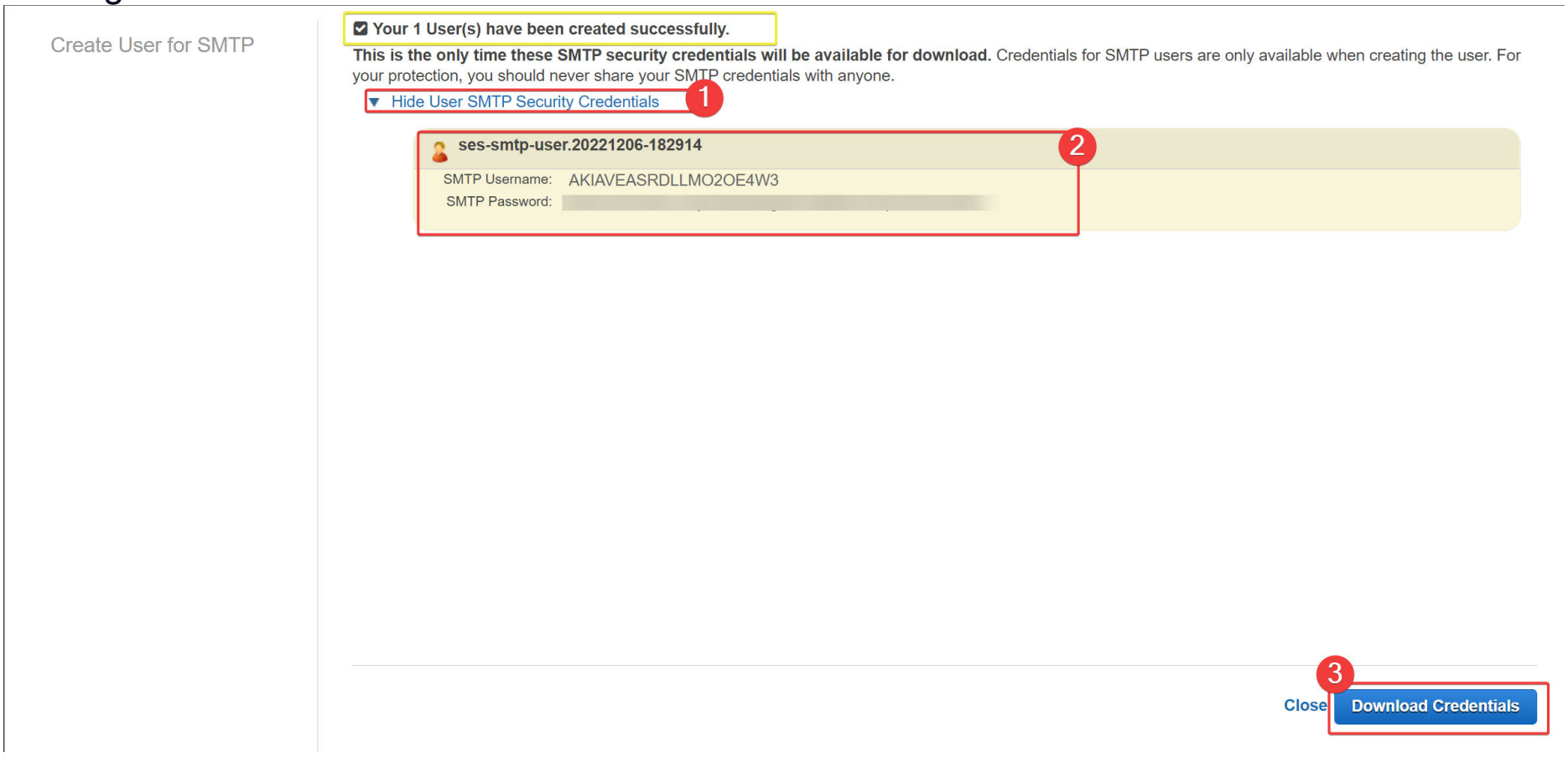
You can choose to define an IAM username or use the default. Once you've done that, click **Create**.



Creating IAM user for SMTP

Once you create an IAM user, your SMTP details will be displayed alongside your IAM username.

A notification telling you your user has been created will be displayed at the top. Make sure you download the credentials since it is a One-Time display detail. You can download them by clicking on **Download Credentials**.



Created SMTP Credentials

Great! You have access to AWS Simple Email Service SMTP credentials. You can use the credentials to connect your backend to the Amazon SES server to send emails.