EXPERIMENT NO. 1

Aim : To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE,Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

Theory:

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.

An EC2 instance is a virtual server in the AWS Cloud. When you launch an EC2 instance, the instance type that you specify determines the hardware available to your instance. Each instance type offers a different balance of compute, memory, network, and storage resources. For more information, see the Amazon EC2 Instance Types Guide.

Features of Amazon EC2 :
Amazon EC2 provides the following high-level features:
Instances Virtual servers.

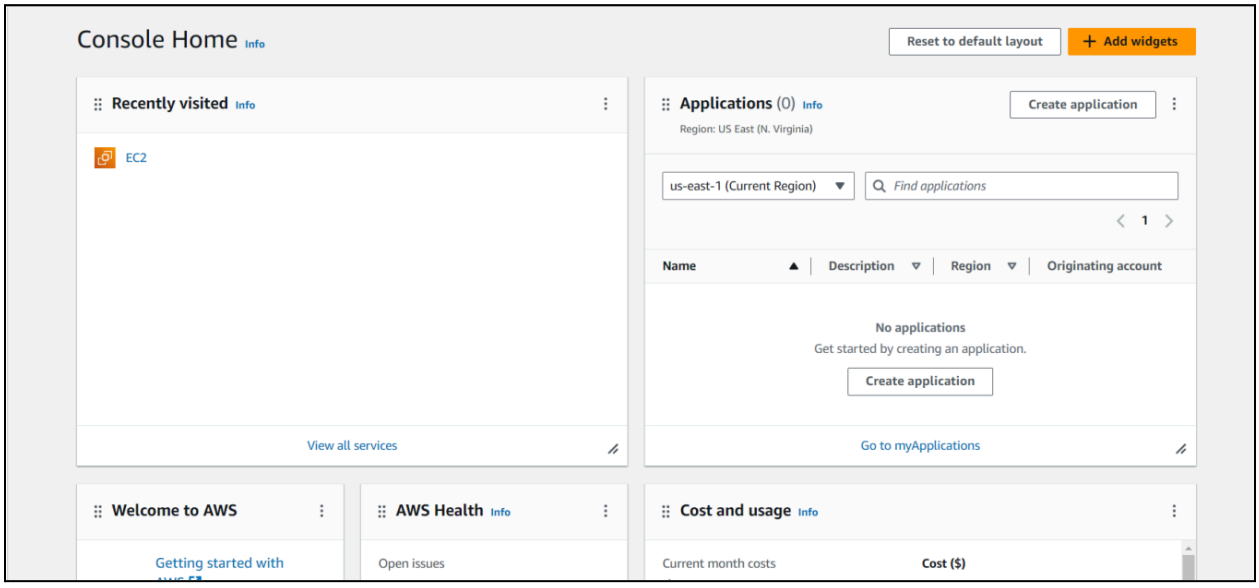
Amazon Machine Images (AMIs)
Preconfigured templates for your instances that package the components you need for your server (including the operating system and additional software).

Instance types
Various configurations of CPU, memory, storage, networking capacity, and graphics hardware for your instances

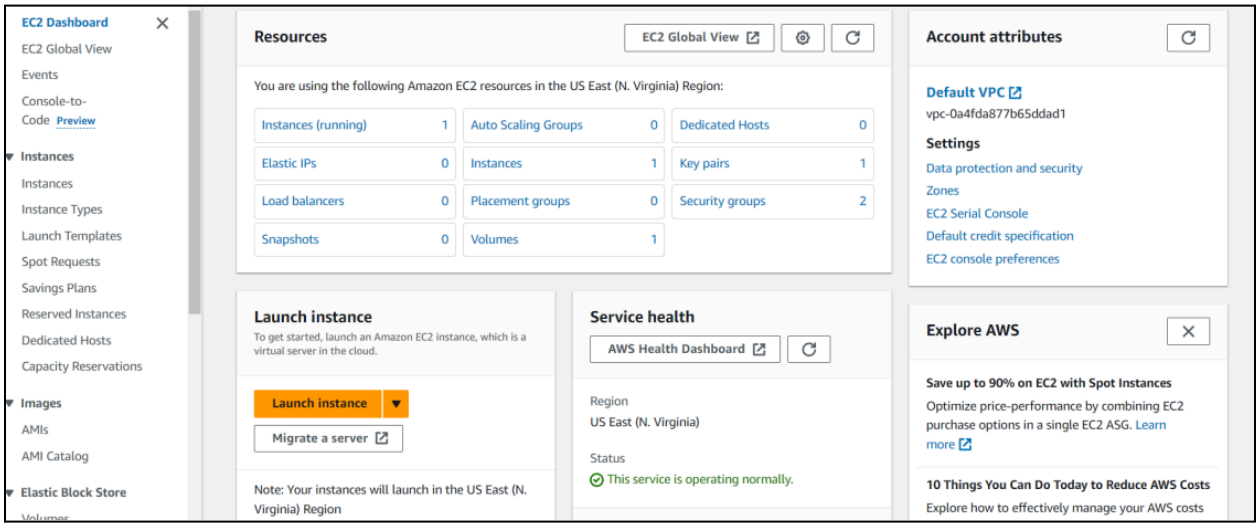
Amazon EBS volumes
Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS).

EC2 Instance Creation and static site hosting

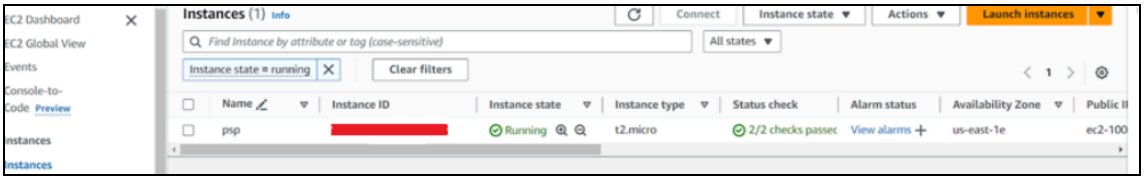
1) Login to your AWS account:



2) Click on EC2 and then create an instance by clicking on instances



3) After an instance is created successfully .



4) After that you wil go to command prompt and perform the following commands:

```
Last login: Thu Aug  8 08:40:05 2024 from 18.206.107.27
[ec2-user@ip-172-31-43-4 ~]$ wget https://github.com/ronak03rsk/IP_Lab_Exp1/archive/refs/heads/main.zip
--2024-08-08 08:55:27--  https://github.com/ronak03rsk/IP_Lab_Exp1/archive/refs/heads/main.zip
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/ronak03rsk/IP_Lab_Exp1/zip/refs/heads/main [following]
--2024-08-08 08:55:27--  https://codeload.github.com/ronak03rsk/IP_Lab_Exp1/zip/refs/heads/main
Resolving codeload.github.com (codeload.github.com)... 140.82.113.10
Connecting to codeload.github.com (codeload.github.com)|140.82.113.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'main.zip'
```

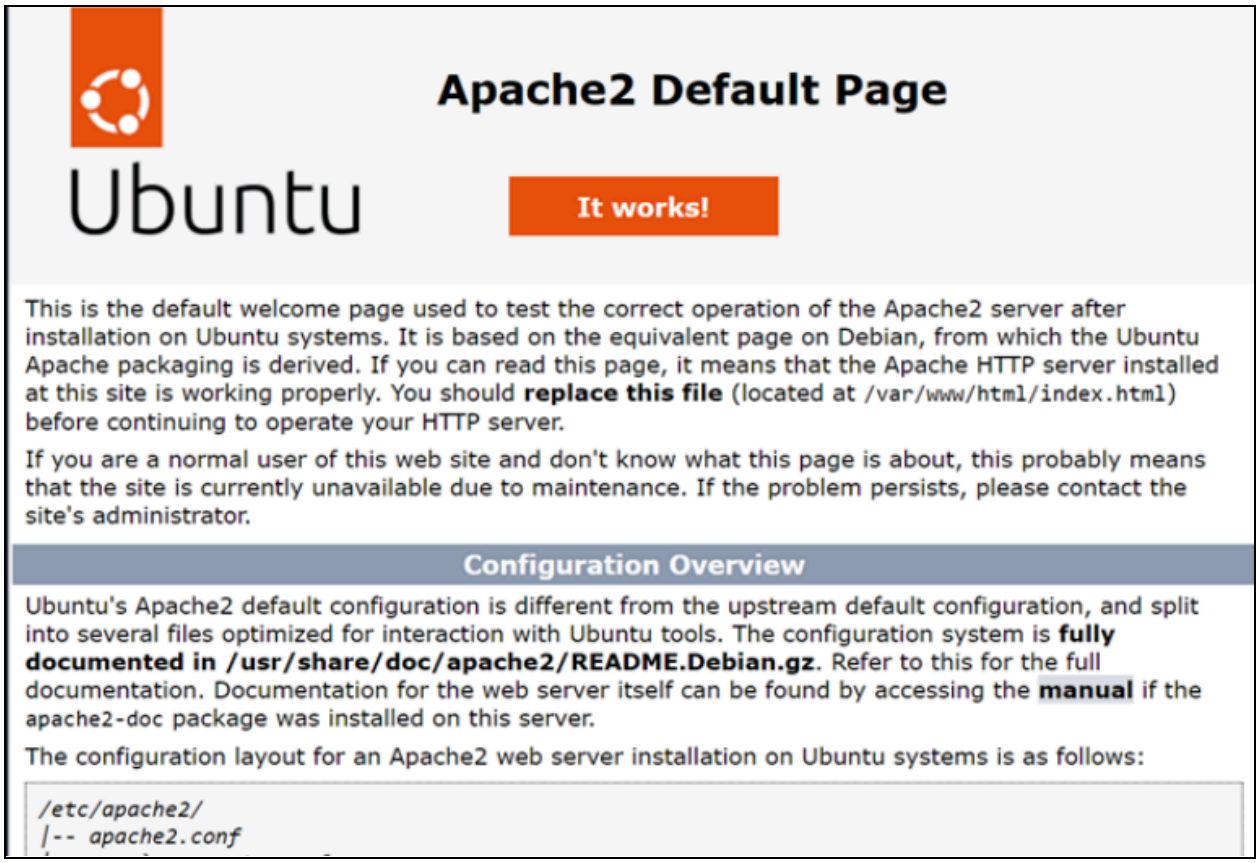
```
2024-08-08 08:55:27 (20.2 MB/s) - 'main.zip' saved [5978110]

[ec2-user@ip-172-31-43-4 ~]$ ls -lrt
total 5840
-rw-r--r--. 1 ec2-user ec2-user 5978110 Aug  8 08:55 main.zip
[ec2-user@ip-172-31-43-4 ~]$ cd aws_exp1a
-bash: cd: aws_exp1a: No such file or directory
[ec2-user@ip-172-31-43-4 ~]$ sudo su -
Last login: Thu Aug  8 08:44:56 UTC 2024 on pts/1
[root@ip-172-31-43-4 ~]# ls -lrt
total 0
drwxr-xr-x. 2 root root 29 Aug  8 08:51 aws_exp1a
[root@ip-172-31-43-4 ~]# cd aws_exp1a
[root@ip-172-31-43-4 aws_exp1a]# wget https://github.com/ronak03rsk/IP_Lab_Exp1/archive/refs/heads/main.zip
--2024-08-08 08:58:02--  https://github.com/ronak03rsk/IP_Lab_Exp1/archive/refs/heads/main.zip
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/ronak03rsk/IP_Lab_Exp1/zip/refs/heads/main [following]
--2024-08-08 08:58:02--  https://codeload.github.com/ronak03rsk/IP_Lab_Exp1/zip/refs/heads/main
Resolving codeload.github.com (codeload.github.com)... 140.82.114.9
Connecting to codeload.github.com (codeload.github.com)|140.82.114.9|:443... connected.
```

```
Resolving codeload.github.com (codeload.github.com)... 140.82.114.9
Connecting to codeload.github.com (codeload.github.com)|140.82.114.9|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'main.zip'

main.zip                                [  <=>

2024-08-08 08:58:02 (16.7 MB/s) - 'main.zip' saved [5978110]

[root@ip-172-31-43-4 aws_exp1a]# ls -lrt
total 6108
-rw-r--r--. 1 root root  271422 Aug  8 08:51 IP_Lab_Exp1.git
-rw-r--r--. 1 root root 5978110 Aug  8 08:58 main.zip
[root@ip-172-31-43-4 aws_exp1a]# unzip main.zip
Archive:  main.zip
af1da3ee5df3966b9c50a2aa25b1bf6de214ef2f
   creating: IP_Lab_Exp1-main/
 extracting: IP_Lab_Exp1-main/README.md
```

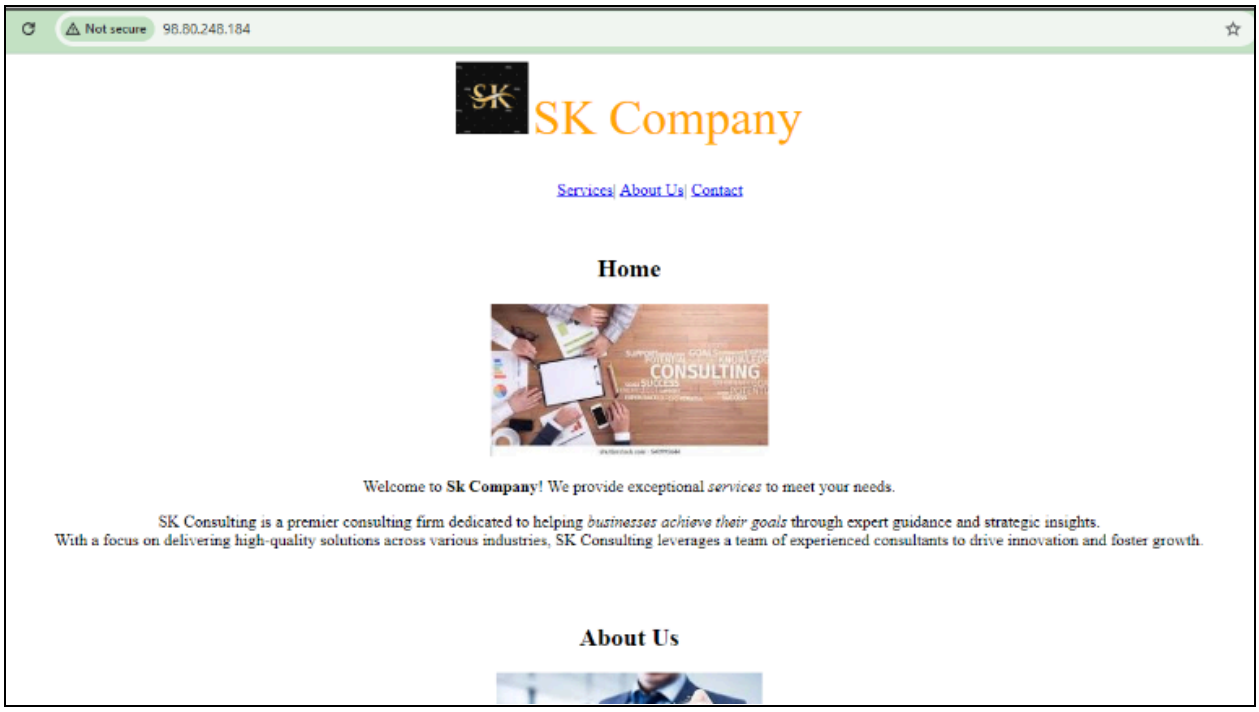
```
-rw-r--r--. 1 root root    7678 Aug  4 10:53 index.html
-rw-r--r--. 1 root root 5960098 Aug  4 10:53 heartbreak-piano-love-song-207235.mp3
drwxr-xr-x. 2 root root      94 Aug  4 10:53 assets
-rw-r--r--. 1 root root      13 Aug  4 10:53 README.md
[root@ip-172-31-43-4 IP_Lab_Exp1-main]# mv * /vat/www/html/
mv: target '/vat/www/html/' is not a directory
[root@ip-172-31-43-4 IP_Lab_Exp1-main]# mv * /var/www/html/
[root@ip-172-31-43-4 IP_Lab_Exp1-main]# cd /var/www/html
[root@ip-172-31-43-4 html]# ls -lrt
total 5836
-rw-r--r--. 1 root root    7678 Aug  4 10:53 index.html
-rw-r--r--. 1 root root 5960098 Aug  4 10:53 heartbreak-piano-love-song-207235.mp3
drwxr-xr-x. 2 root root      94 Aug  4 10:53 assets
-rw-r--r--. 1 root root      13 Aug  4 10:53 README.md
[root@ip-172-31-43-4 html]# systemctl status httpd
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-172-31-43-4 html]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/htt
[root@ip-172-31-43-4 html]# systemctl start httpd
```

```
[root@ip-172-31-43-4 aws_exp1a]# ls -lrt
total 6108
-rw-r--r--. 1 root root  271422 Aug  8 08:51 IP_Lab_Exp1.git
-rw-r--r--. 1 root root 5978110 Aug  8 08:58 main.zip
[root@ip-172-31-43-4 aws_exp1a]# unzip main.zip
Archive:  main.zip
af1da3ee5df3966b9c50a2aa25b1bf6de214ef2f
   creating: IP_Lab_Exp1-main/
 extracting: IP_Lab_Exp1-main/README.md
   creating: IP_Lab_Exp1-main/assets/
  inflating: IP_Lab_Exp1-main/assets/consulting.jpg
  inflating: IP_Lab_Exp1-main/assets/consulting1.jpg
  inflating: IP_Lab_Exp1-main/assets/consulting2.jpg
 extracting: IP_Lab_Exp1-main/assets/download.png
  inflating: IP_Lab_Exp1-main/heartbreak-piano-love-song-207235.mp3
  inflating: IP_Lab_Exp1-main/index.html
[root@ip-172-31-43-4 aws_exp1a]# ls -lrt
total 6108
drwxr-xr-x. 3 root root     100 Aug  4 10:53 IP_Lab_Exp1-main
-rw-r--r--. 1 root root  271422 Aug  8 08:51 IP_Lab_Exp1.git
-rw-r--r--. 1 root root 5978110 Aug  8 08:58 main.zip
[root@ip-172-31-43-4 aws_exp1a]# cd IP_Lab_Exp1-main
[root@ip-172-31-43-4 IP_Lab_Exp1-main]# ls -lrt
```

5) After that the ip-address which was given while running the instance, copy that and paste that on chrome, make sure that it is http and not https

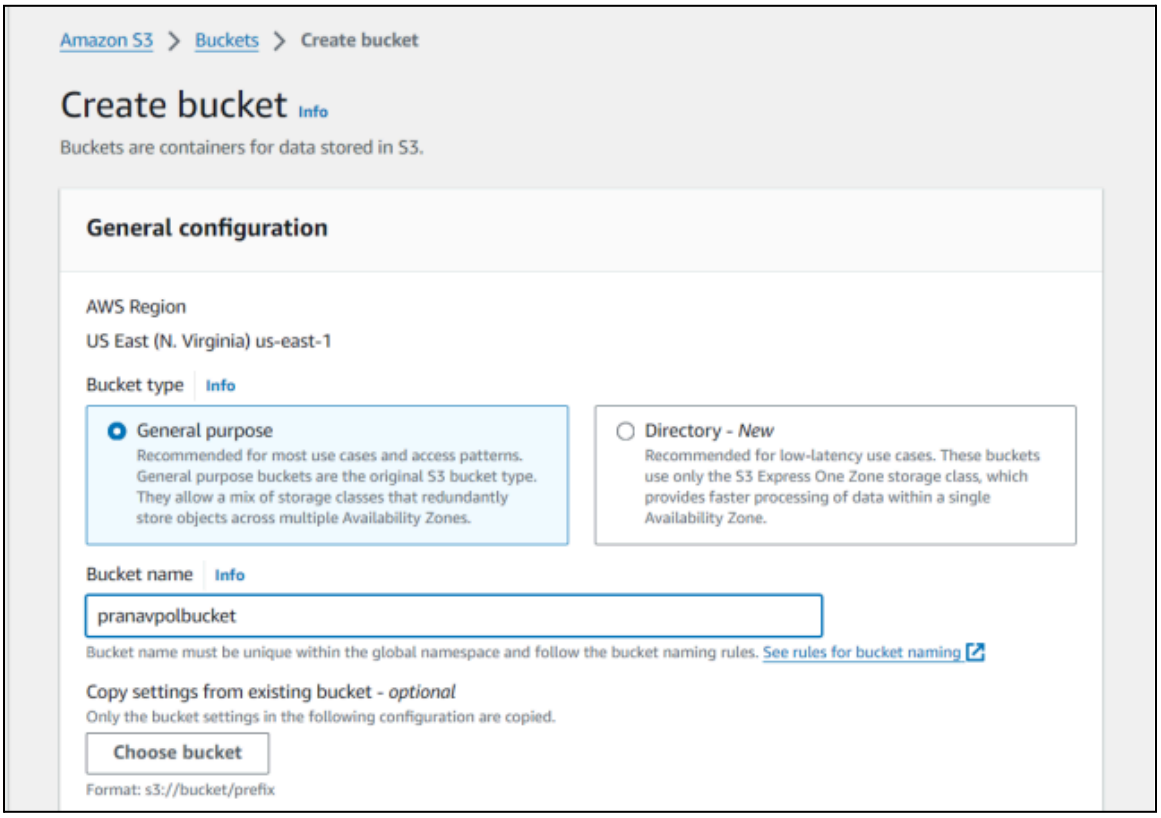


6) After the commands being successfully executed and everything is perfectly fine you will see the website you wanted on the same IP address.

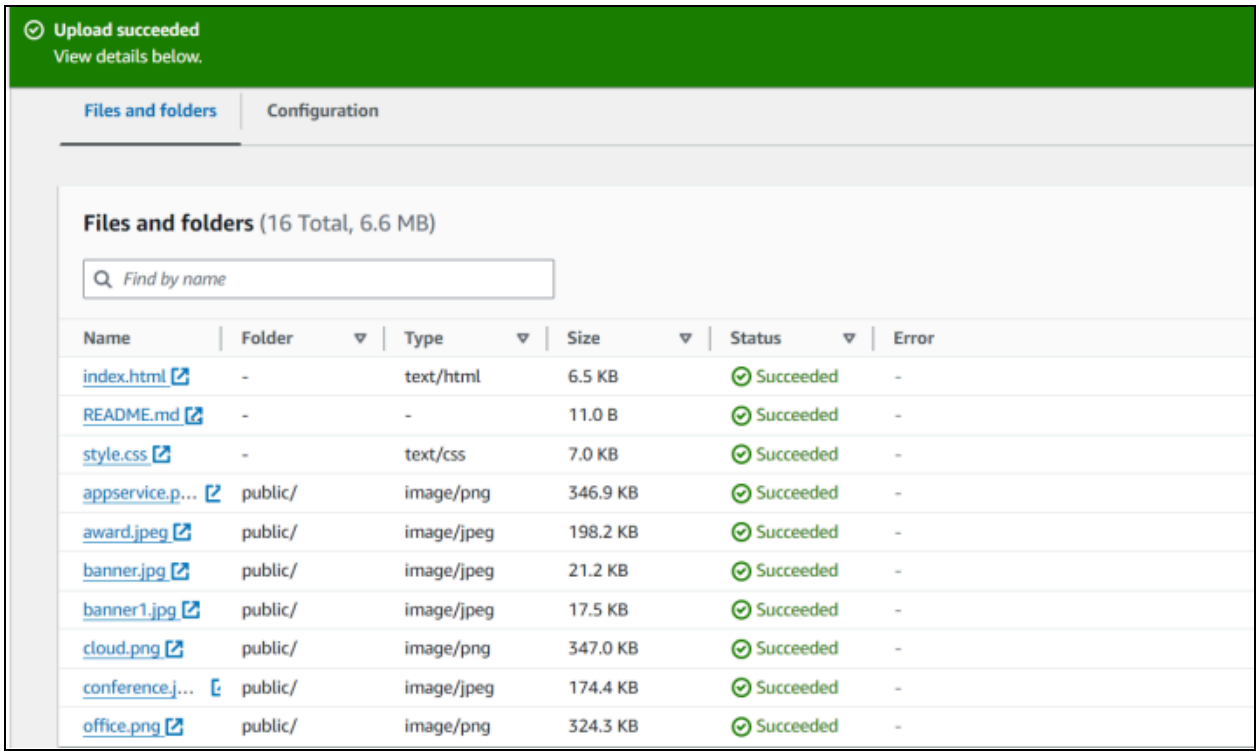


Static Site Hosting using S3 bucket:

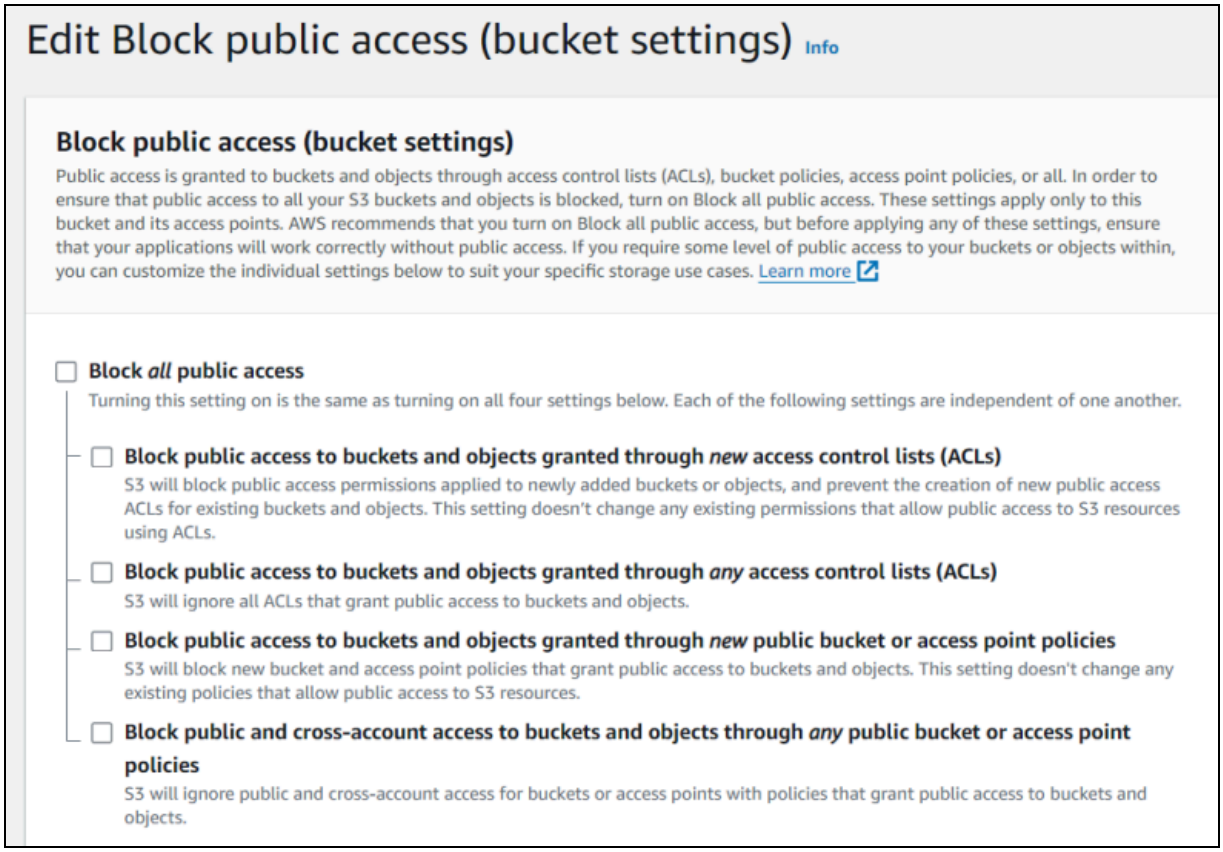
Step1: Create bucket

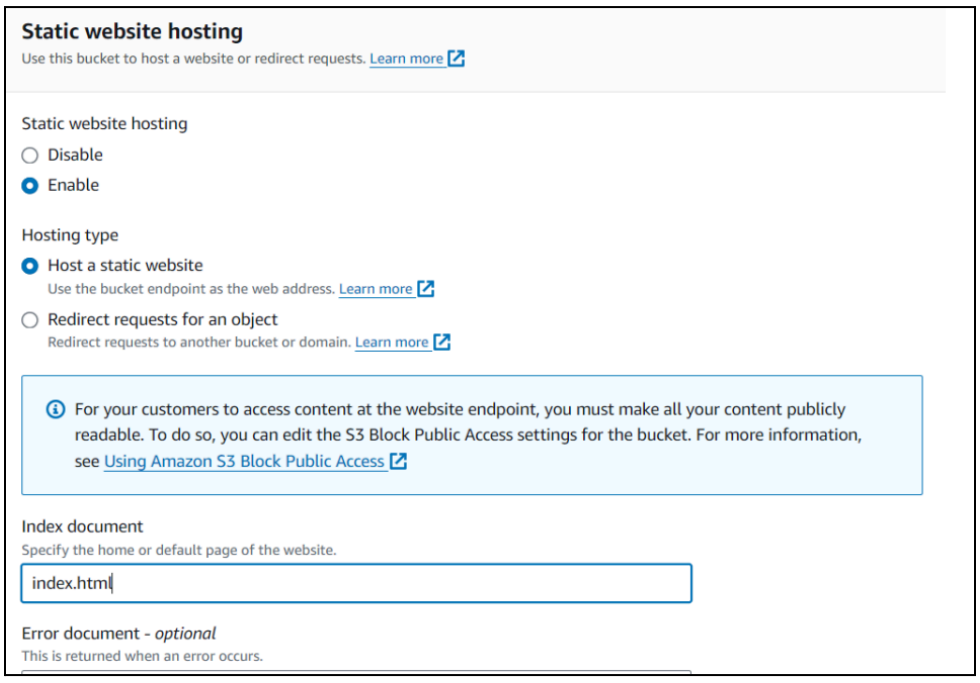


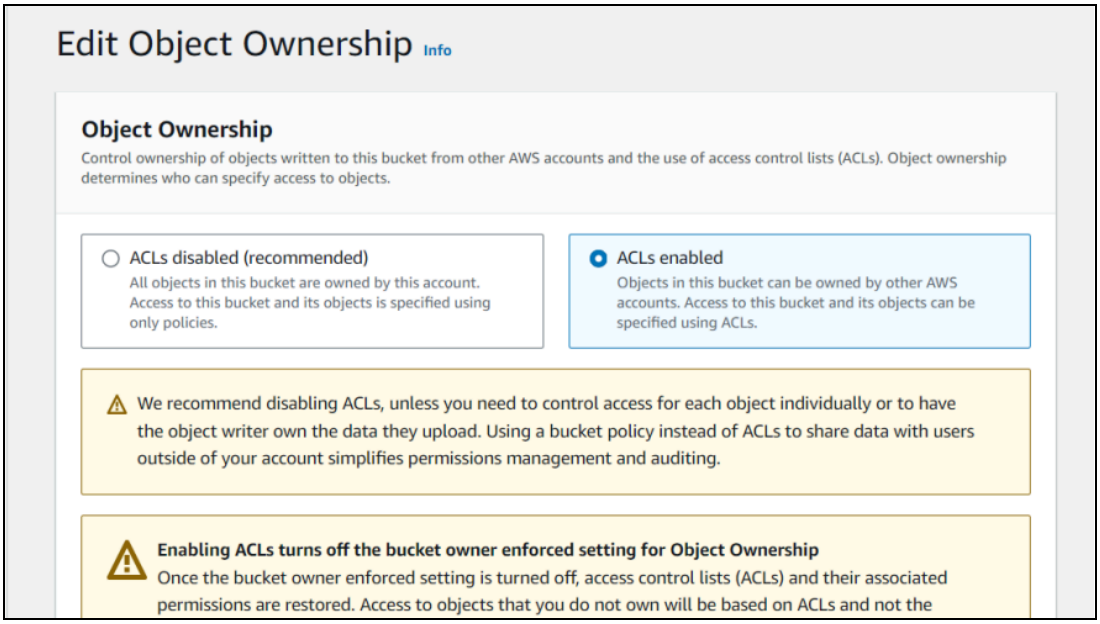
Step 2: Add resources

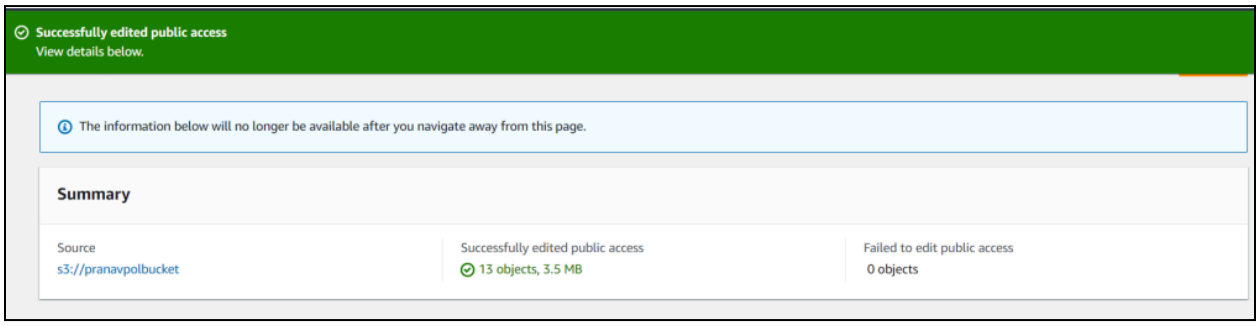


Step 3 : Provide public access

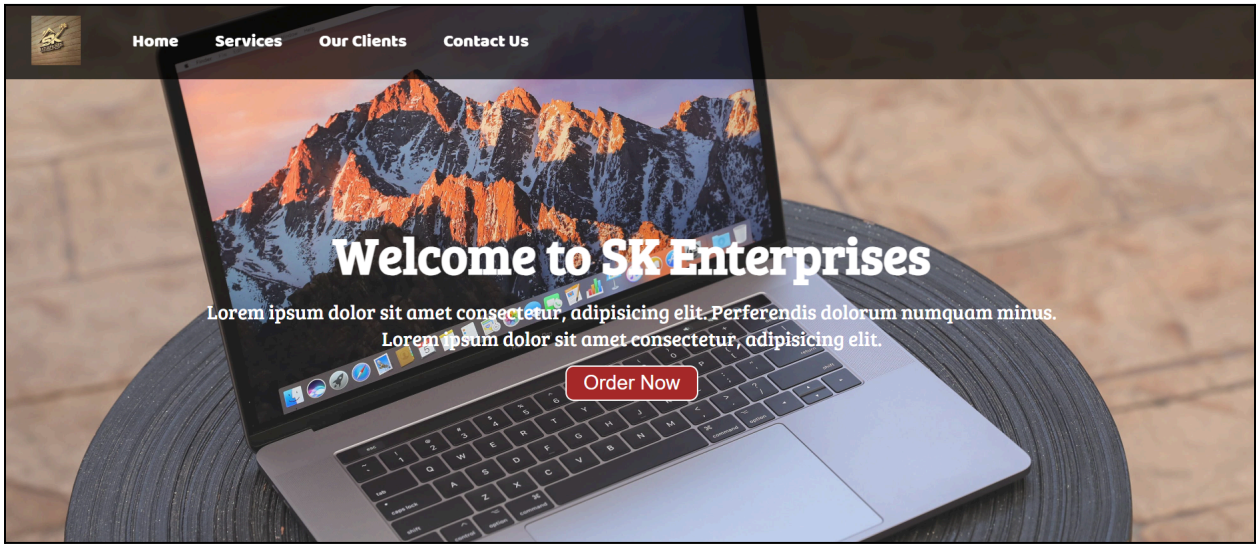

Edit Block public access (bucket settings) Info
Block public access (bucket settings)
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
Block public access to buckets and objects granted through new access control lists (ACLs)
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.


Static website hosting
Use this bucket to host a website or redirect requests. Learn more
Static website hosting
Disable
Enable
Hosting type
Host a static website
Use the bucket endpoint as the web address. Learn more
Redirect requests for an object
Redirect requests to another bucket or domain. Learn more
For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access
Index document
Specify the home or default page of the website.
index.html
Error document - optional
This is returned when an error occurs.





Step 4 : visit hosted website



EC2 Dynamic Site Hosting:

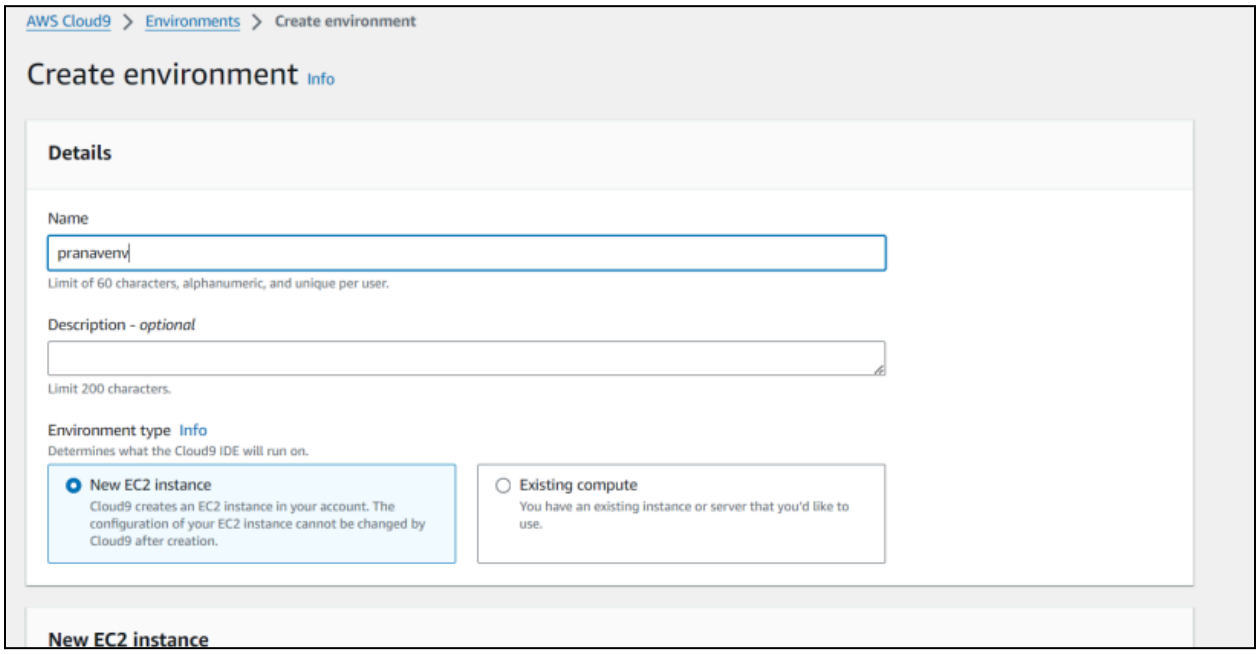
Step 1 : Open Console and clone the github repository

```
root@ip-172-31-55-145:/home/ubuntu/dynamic/dyanamic_site# npm i
(                    ) ⸩ reify:define-data-property: http fetch GET 200 https://registry.npmjs.org/define-data-propert

added 93 packages, and audited 94 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@ip-172-31-55-145:/home/ubuntu/dynamic/dyanamic_site# npm start

> hosting-dynamic-website@1.0.0 start
> nodemon index.js

[nodemon] 3.1.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server is running on port 3000
```

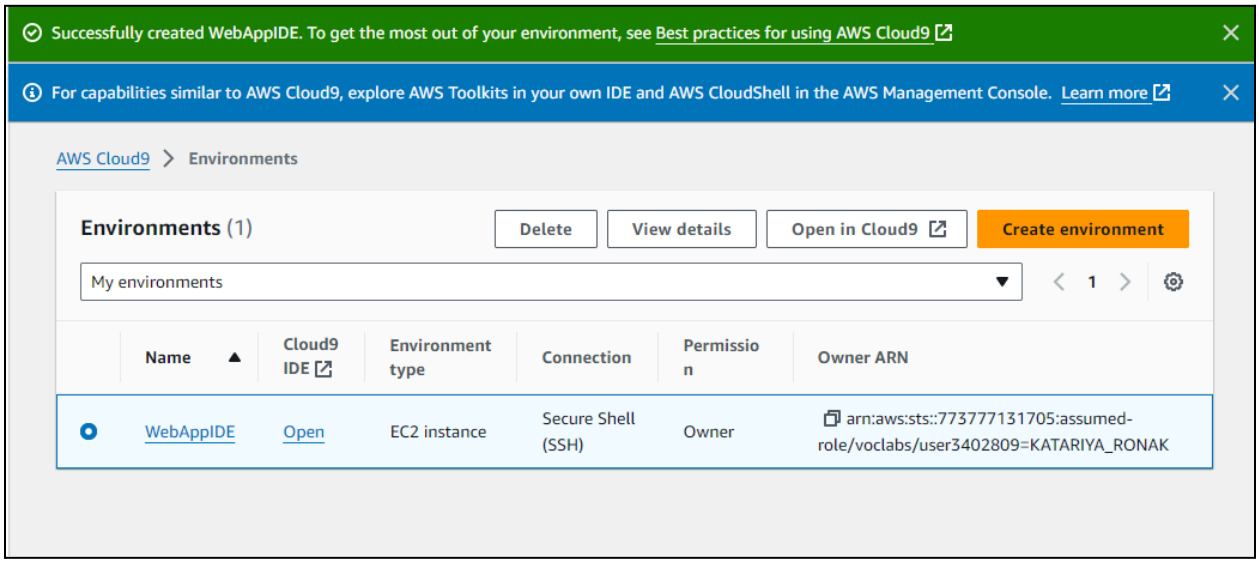
Step 2 : Install necessary Packages and run website on port 3000

Cloud 9 IDE Site Hosting:

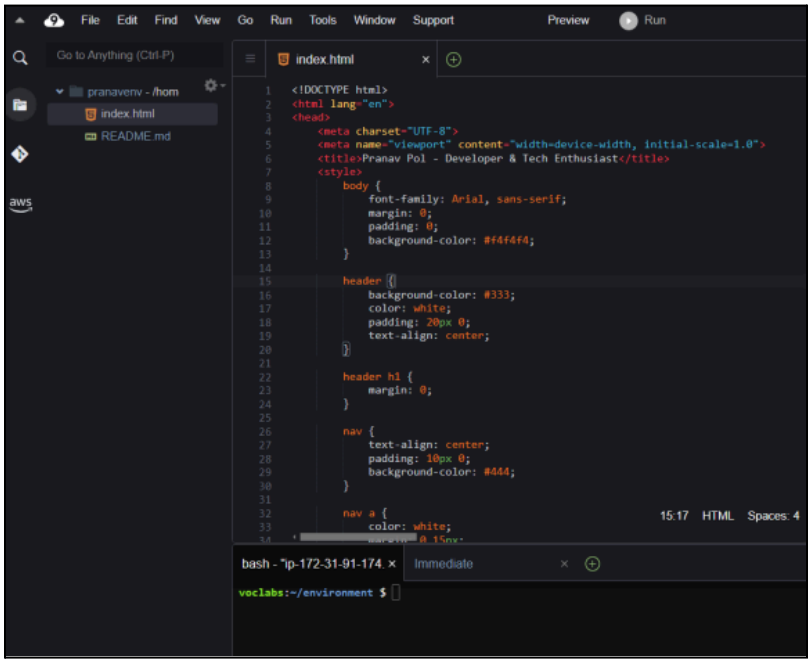
Step 1: Create Environment



Step 2 :Open the Environment IDE

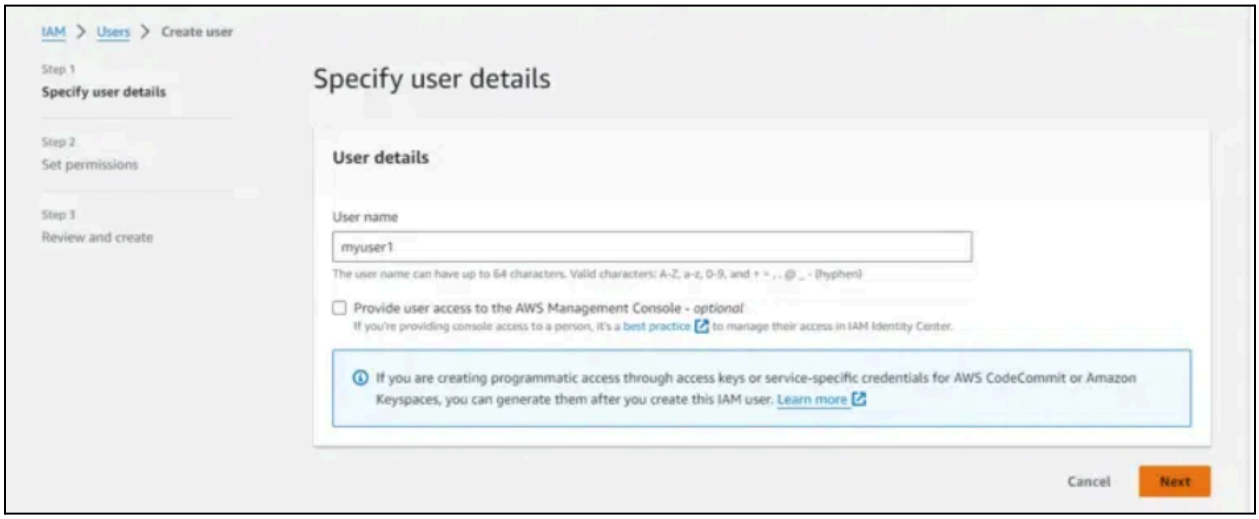


Step 3: Add the code and preview the website

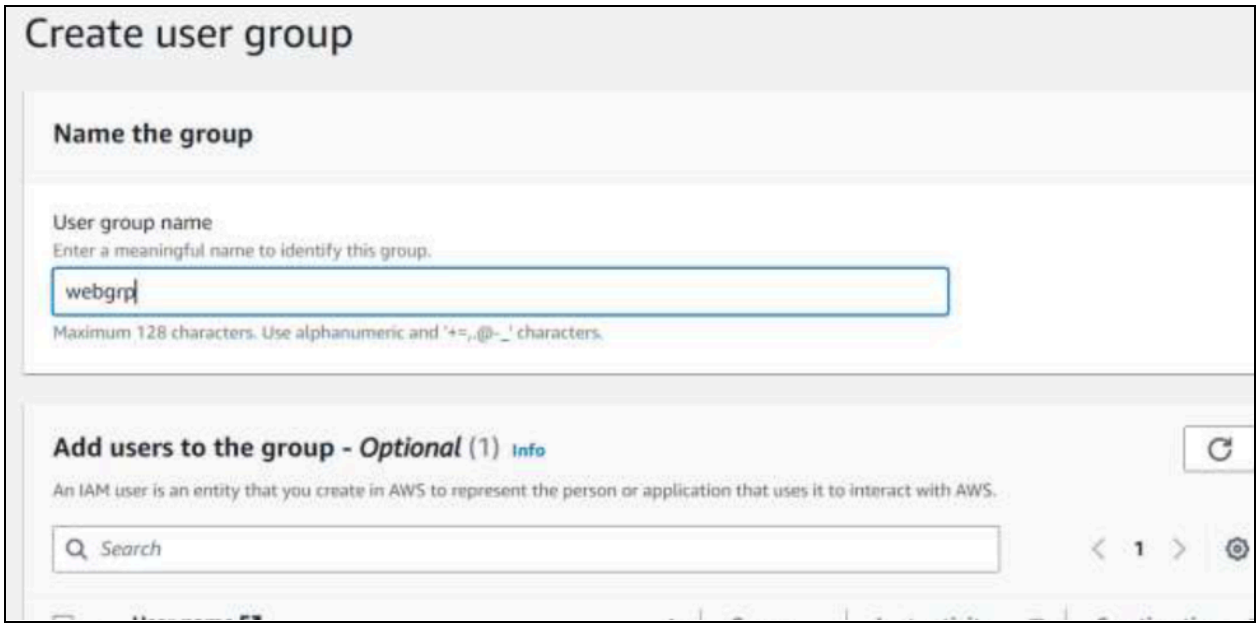


IAM:

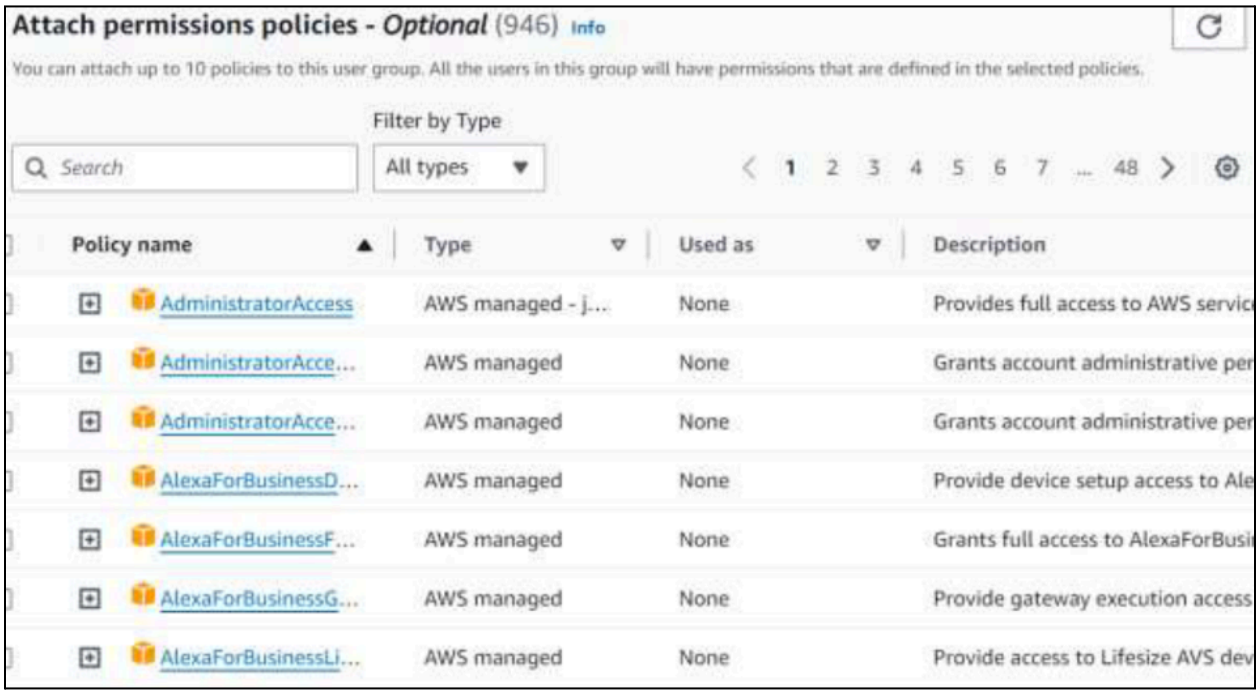
Step 1: Create a user.



Step 2: Create a group.



Step 3: Give some role to the user.



Experiment No: 2

Aim :To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Theory:

With Elastic Beanstalk you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. Elastic Beanstalk also supports Docker platforms. With Docker containers you can choose your own programming language and application dependencies that may not be supported by the other Elastic Beanstalk platforms. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.

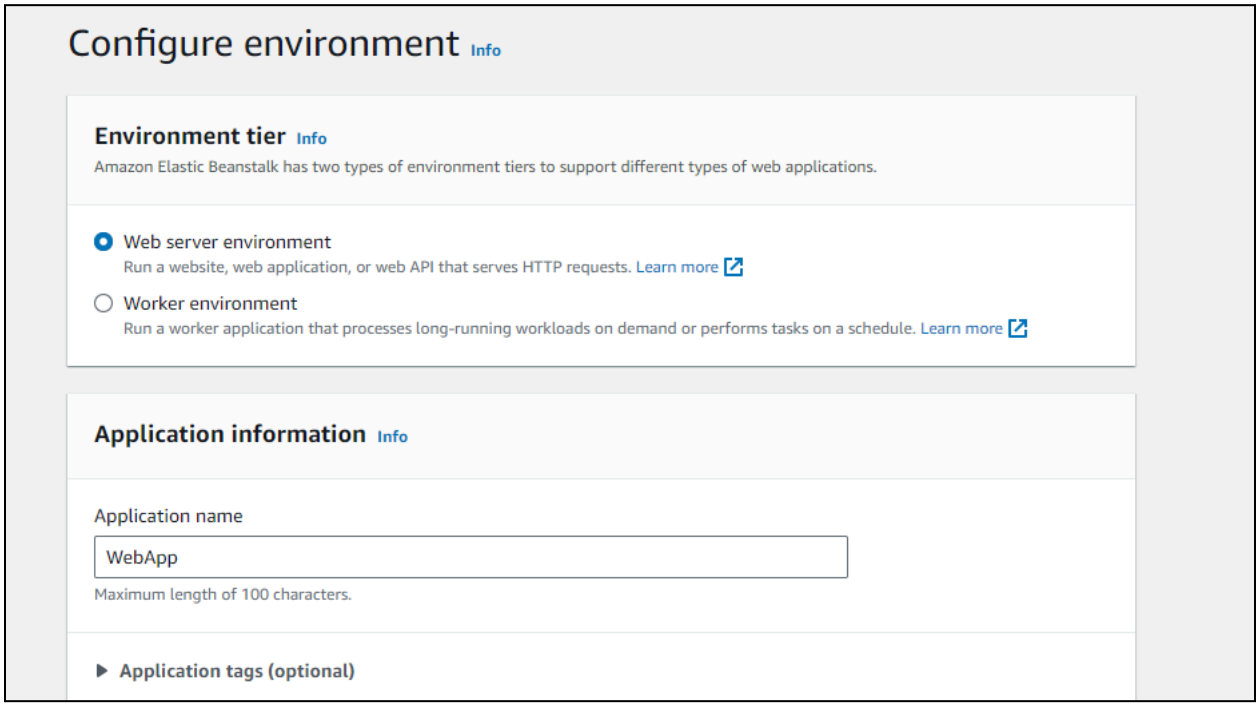
You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or eb, a high-level CLI designed specifically for Elastic Beanstalk.

You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the Elastic Beanstalk web interface (console). To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions. The following diagram illustrates the workflow of Elastic Beanstalk.

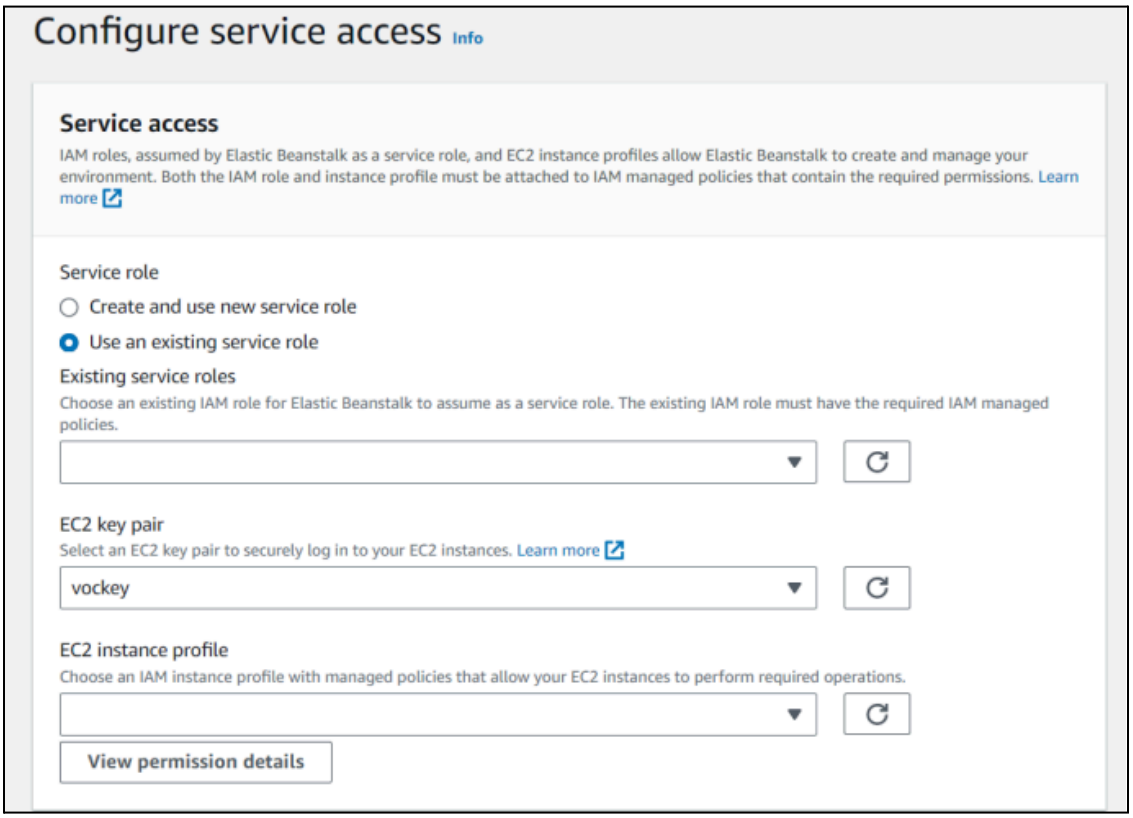
After you create and deploy your application, information about the application—including metrics, events, and environment status—is available through the Elastic Beanstalk console, APIs, or Command Line Interfaces, including the unified AWS CLI.

Elastic Beanstalk

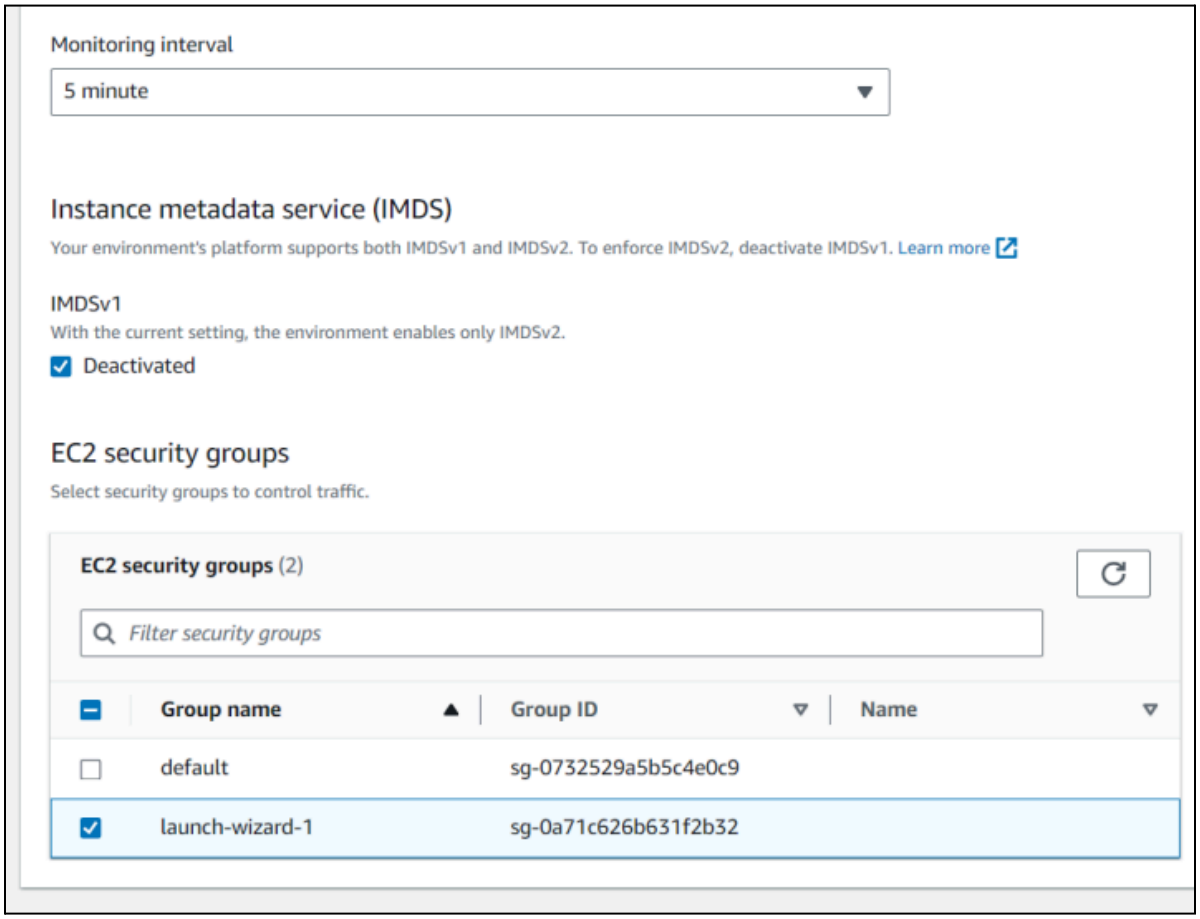
Step 1: create environment

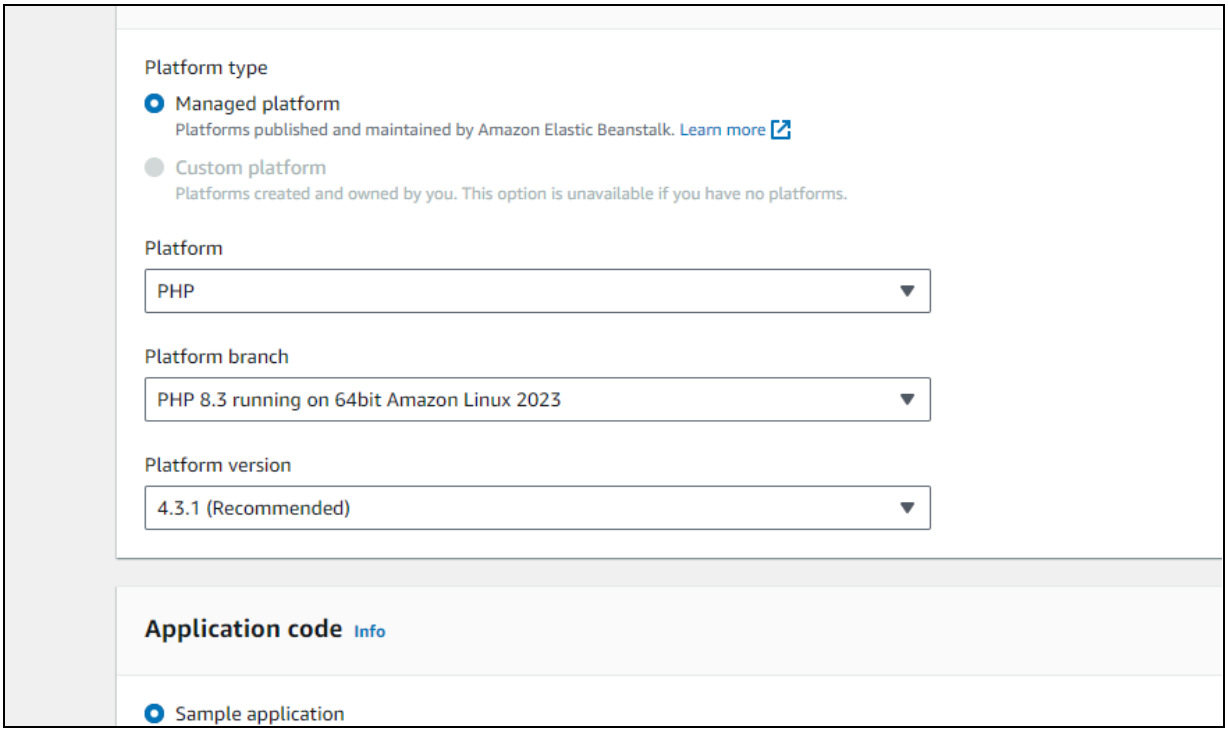


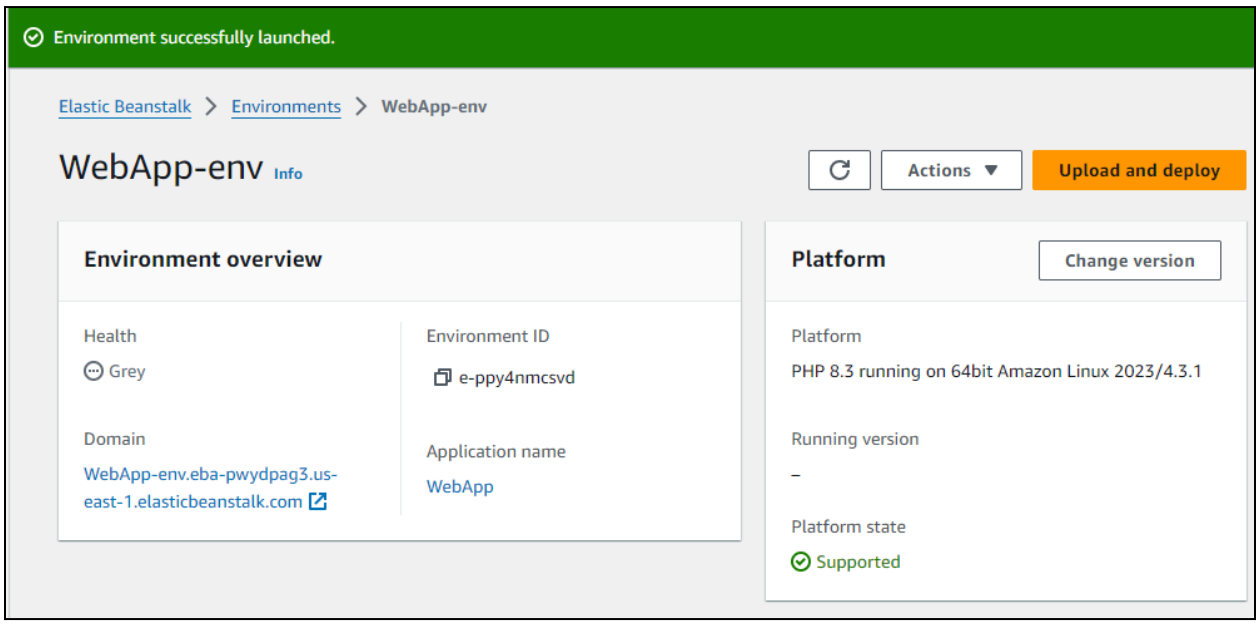
Step 2 : add your Ec2 key pair and instance profile



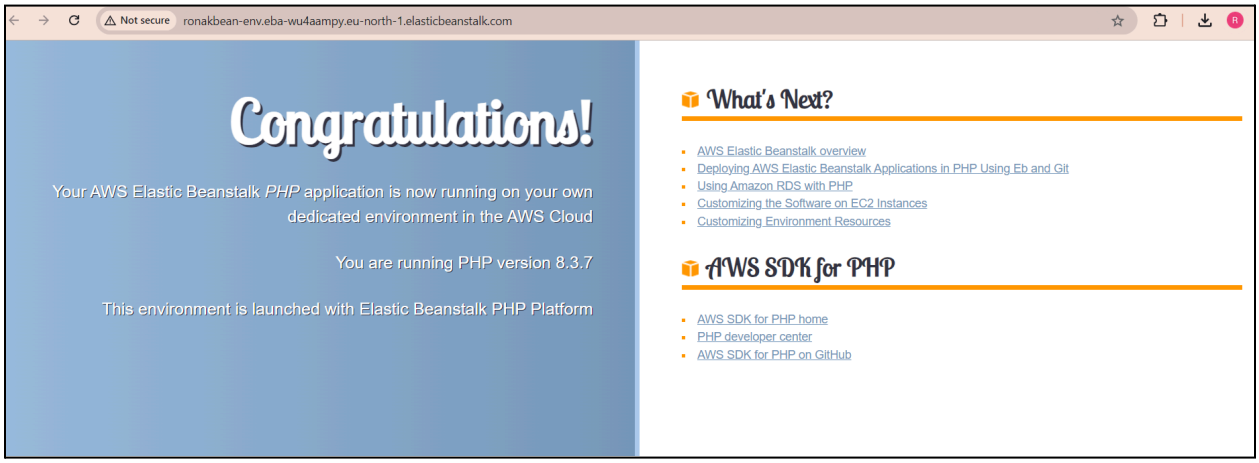
Step 3 : add security config and review all settings




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
PHP
Platform branch
PHP 8.3 running on 64bit Amazon Linux 2023
Platform version
4.3.1 (Recommended)
Application code Info
Sample application

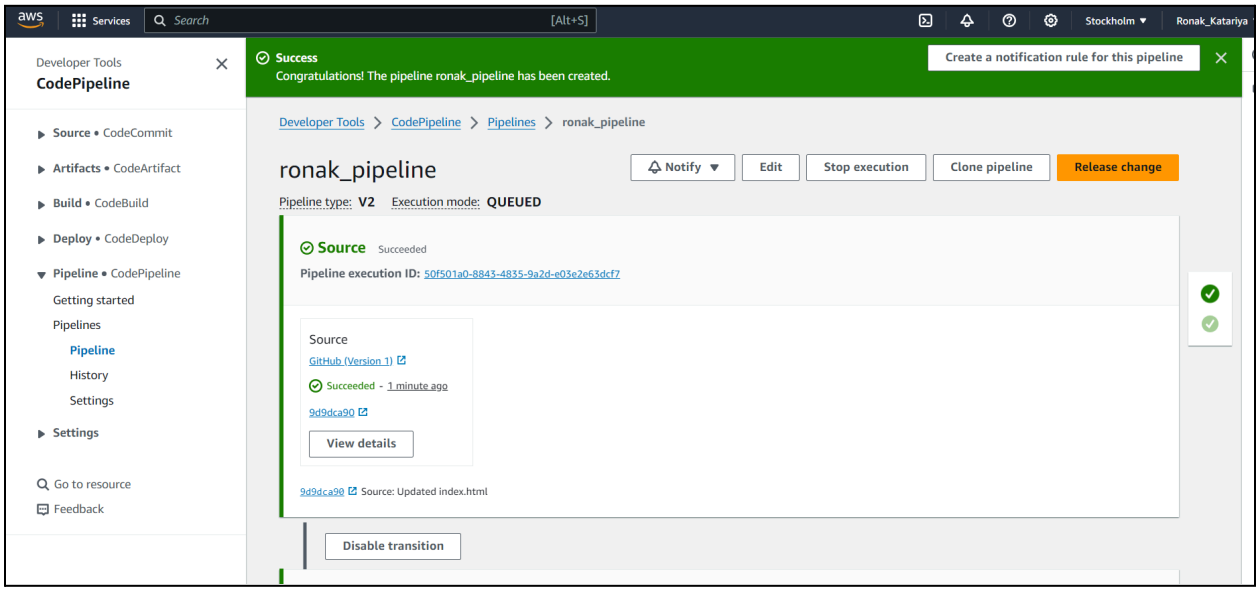

Environment successfully launched.
Elastic Beanstalk > Environments > WebApp-env
WebApp-env Info
Actions
Upload and deploy
Environment overview
Health
Grey
Environment ID
e-ppy4nmcsvd
Domain
WebApp-env.eba-pwydpag3.us-east-1.elasticbeanstalk.com
Application name
WebApp
Platform
Change version
Platform
PHP 8.3 running on 64bit Amazon Linux 2023/4.3.1
Running version
–
Platform state
Supported

Step 4 : Beanstalk environment is created

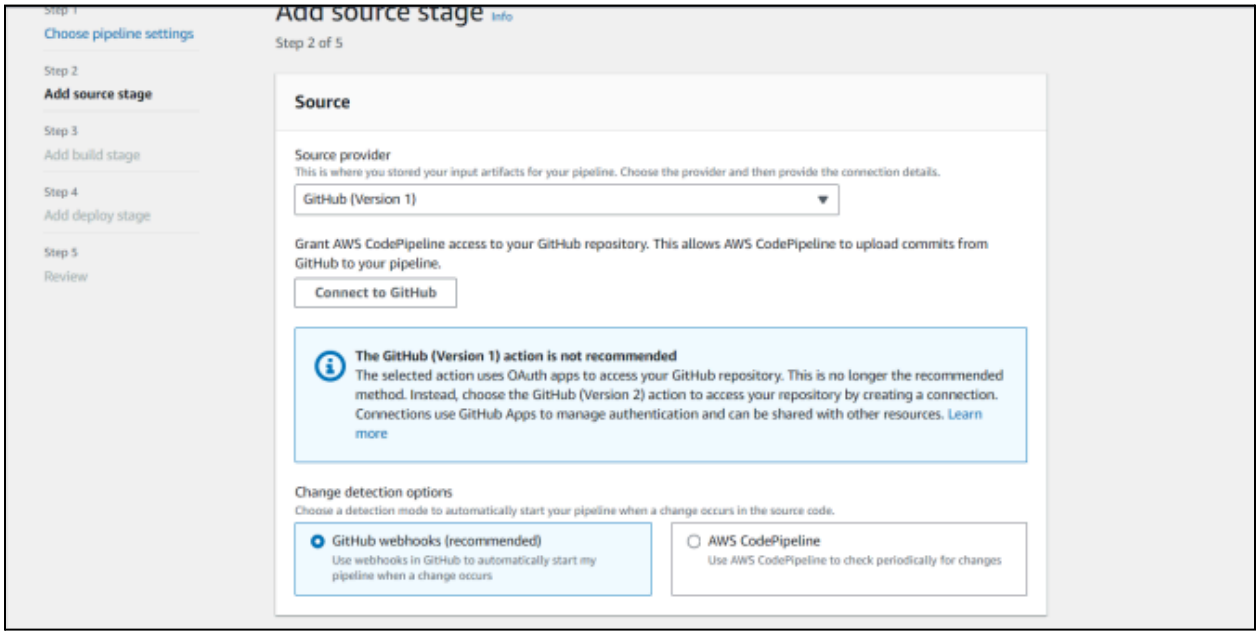


Pipeline Creation:

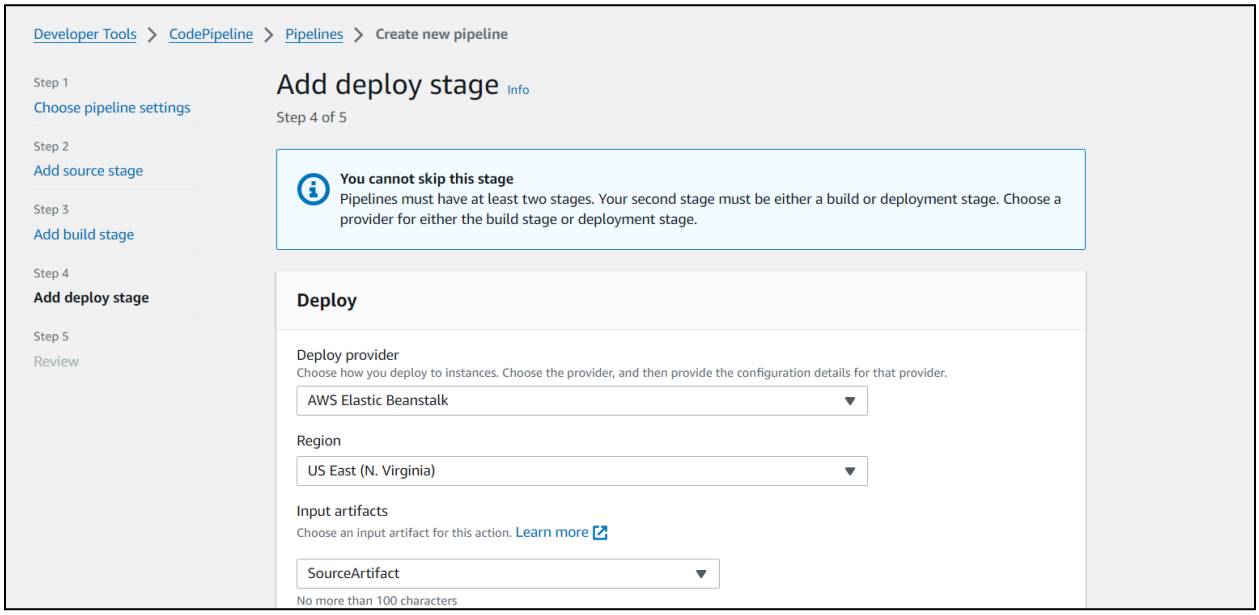
Step 1 : click on create pipeline and give name



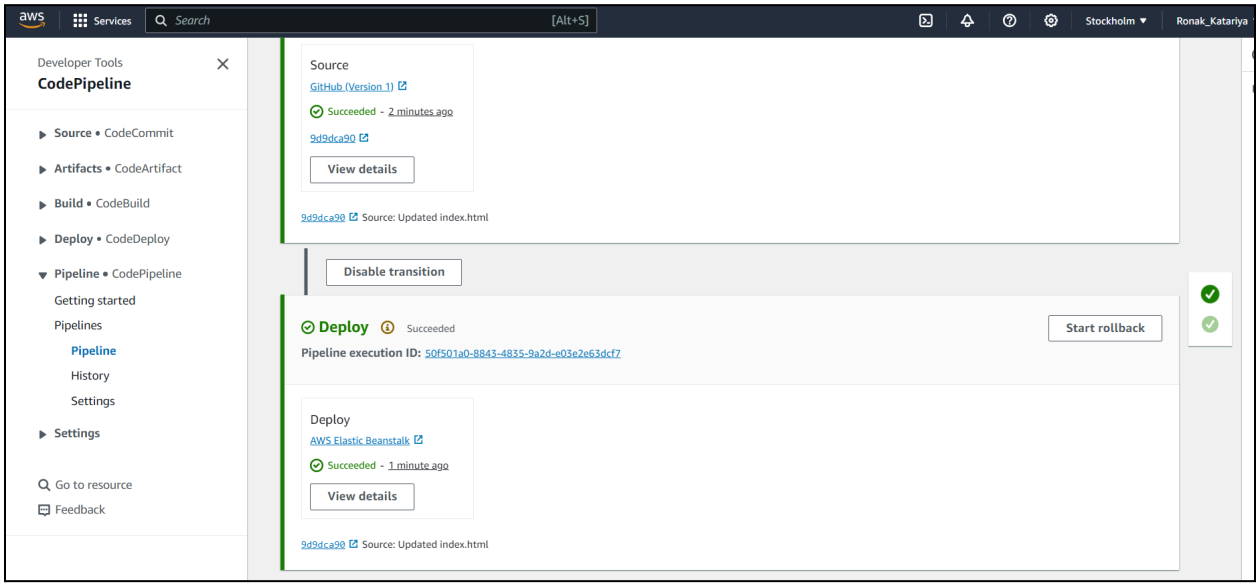
Step 2 : Add Your github account and add the file to add to pipeline deployment



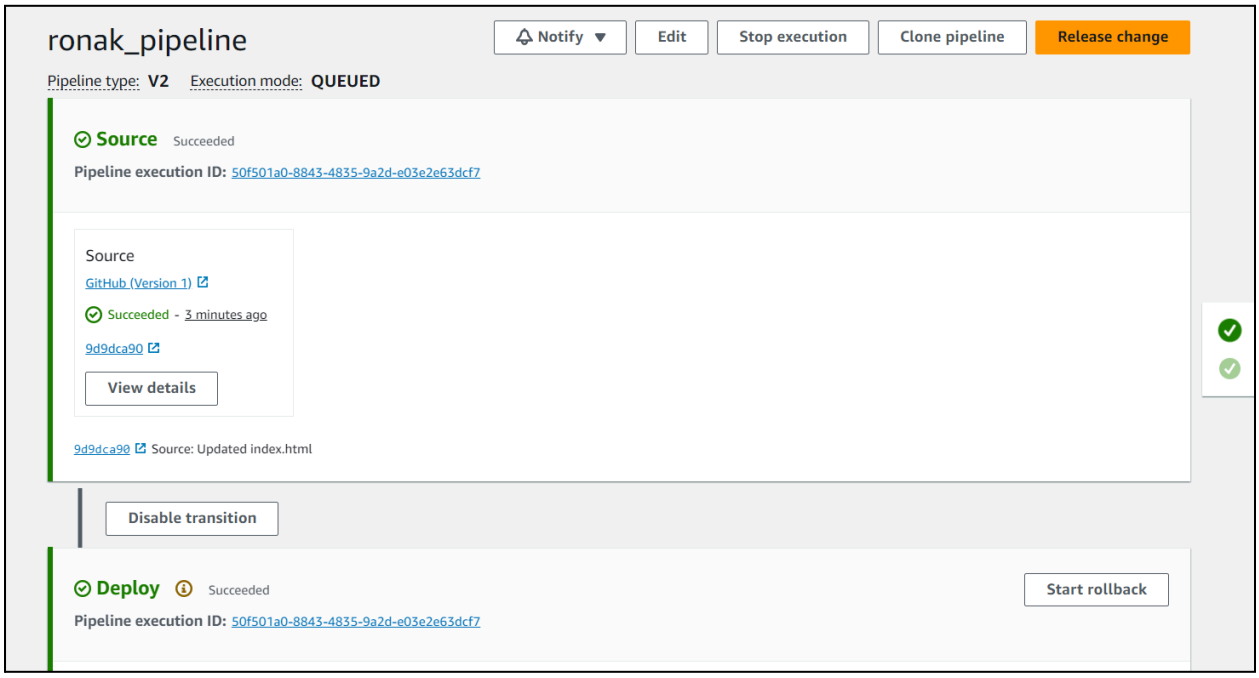
Step 3 : Add deploy config choosing the elastic beanstalk



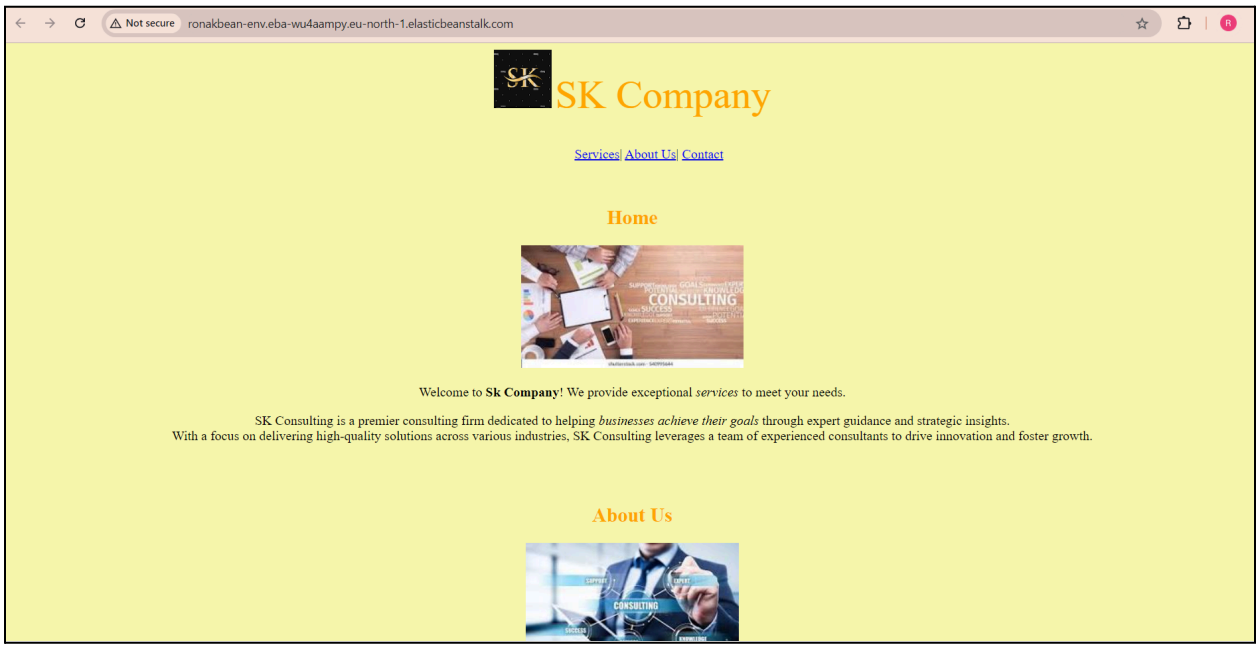
Step 4 : review changes and submit



Step 5 : view the pipeline build and deployment



Step 6 : Check the deployed website at beanstalk link



Advanced DevOps Lab
Experiment:3

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Theory:

To understand Kubernetes Cluster Architecture and how to install and spin up a Kubernetes cluster on Linux machines or cloud platforms, it's essential to grasp the fundamental components and design principles of Kubernetes.

Overview of Kubernetes

Kubernetes is an open-source container orchestration platform developed by Google, designed to automate the deployment, scaling, and management of containerized applications. It provides a robust infrastructure that supports microservices architecture, offering features such as self-healing, scaling, and zero-downtime deployments. Kubernetes can run on various environments, including public clouds (like AWS and Azure), private clouds, and bare metal servers.

Kubernetes Architecture

Kubernetes architecture is primarily composed of two main components: the Control Plane and the Data Plane.

Control Plane

The Control Plane manages the overall state of the Kubernetes cluster and includes several key components:

- kube-apiserver: The API server acts as the gateway for all interactions with the cluster, processing REST requests and managing the state of the cluster.
- etcd: A distributed key-value store that holds the configuration data and state of the cluster, ensuring consistency and availability.
- kube-scheduler: Responsible for assigning Pods to worker nodes based on resource availability and other constraints.
- kube-controller-manager: Manages controllers that regulate the state of the cluster, ensuring that the desired state matches the actual state.
- cloud-controller-manager (optional): Integrates with cloud provider APIs to manage resources specific to the cloud environment.

Data Plane

The Data Plane consists of the worker nodes that run the containerized applications. Each worker node includes:

- kubelet: An agent that ensures containers are running in Pods. It communicates with the Control Plane to receive instructions.
- kube-proxy: Maintains network rules and facilitates communication between Pods and services.
- Container Runtime: Software responsible for running containers, such as Docker or containerd.

Core Concepts

Key concepts in Kubernetes include:

- Pods: The smallest deployable units in Kubernetes, which can contain one or more containers.
- Services: Abstracts a set of Pods, providing a stable network endpoint for accessing them.
- Deployments: Define the desired state for Pods and manage their lifecycle, including scaling and updates.

Installing and Spinning Up a Kubernetes Cluster

To install and set up a Kubernetes cluster, follow these general steps:

1. Choose an Environment: Decide whether to deploy on local machines or a cloud platform. For cloud platforms, services like Google Kubernetes Engine (GKE), Amazon EKS, or Azure AKS can simplify the process.
2. Install Prerequisites: Ensure that you have the necessary tools installed, such as kubectl (the command-line tool for interacting with the cluster) and a container runtime.
3. Set Up the Control Plane: This can be done using tools like kubeadm, which helps bootstrap the cluster by initializing the Control Plane components.
4. Join Worker Nodes: Once the Control Plane is set up, you can join worker nodes to the cluster using the token generated during the initialization.
5. Deploy Applications: After the cluster is up and running, you can deploy your applications using YAML configuration files that define the desired state of your Pods and Services.

Best Practices

When setting up a Kubernetes cluster, consider the following best practices:

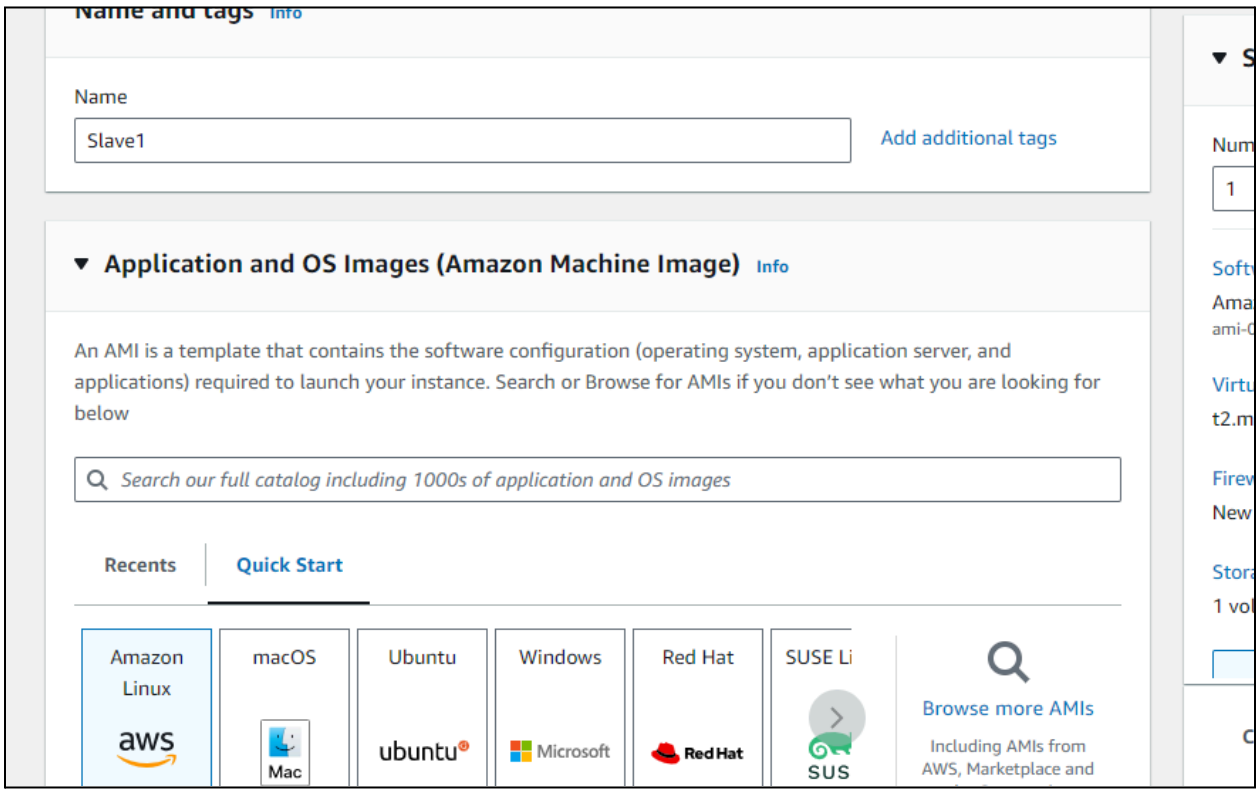
- Resource Management: Define resource requests and limits for Pods to ensure efficient utilization of cluster resources.
- High Availability: Use multiple Control Plane nodes to avoid single points of failure.
- Networking: Implement network policies to secure communication between Pods and manage external access.
- Monitoring and Logging: Integrate monitoring tools and logging solutions to keep track of cluster performance and troubleshoot issues.

By understanding the architecture and following the installation steps and best practices, you can effectively manage a Kubernetes cluster, enabling efficient deployment and scaling of containerized applications.

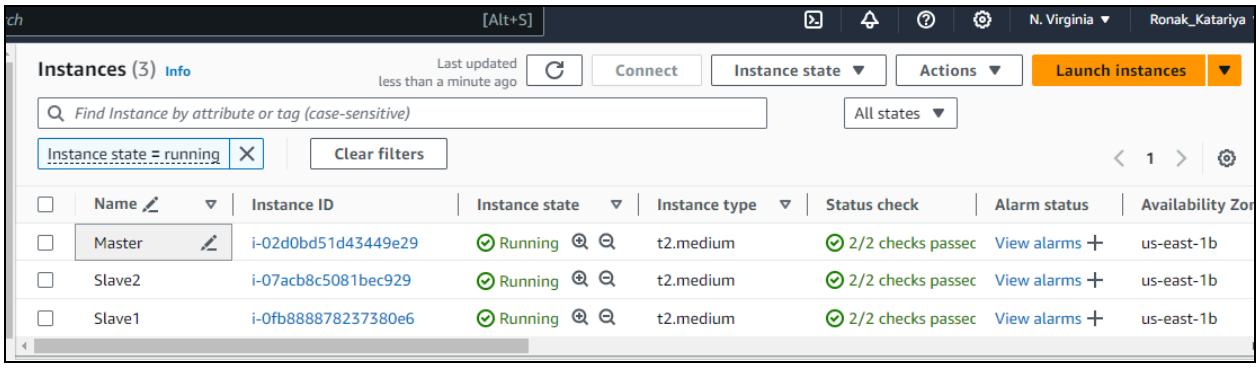
Steps:

1. Create 3 EC2 Ubuntu Instances on AWS.

Extra:
When we select ubuntu we have to select an older version - 22.04.
(Name 1 as Master, the other 2 as Slave1 and Slave2)

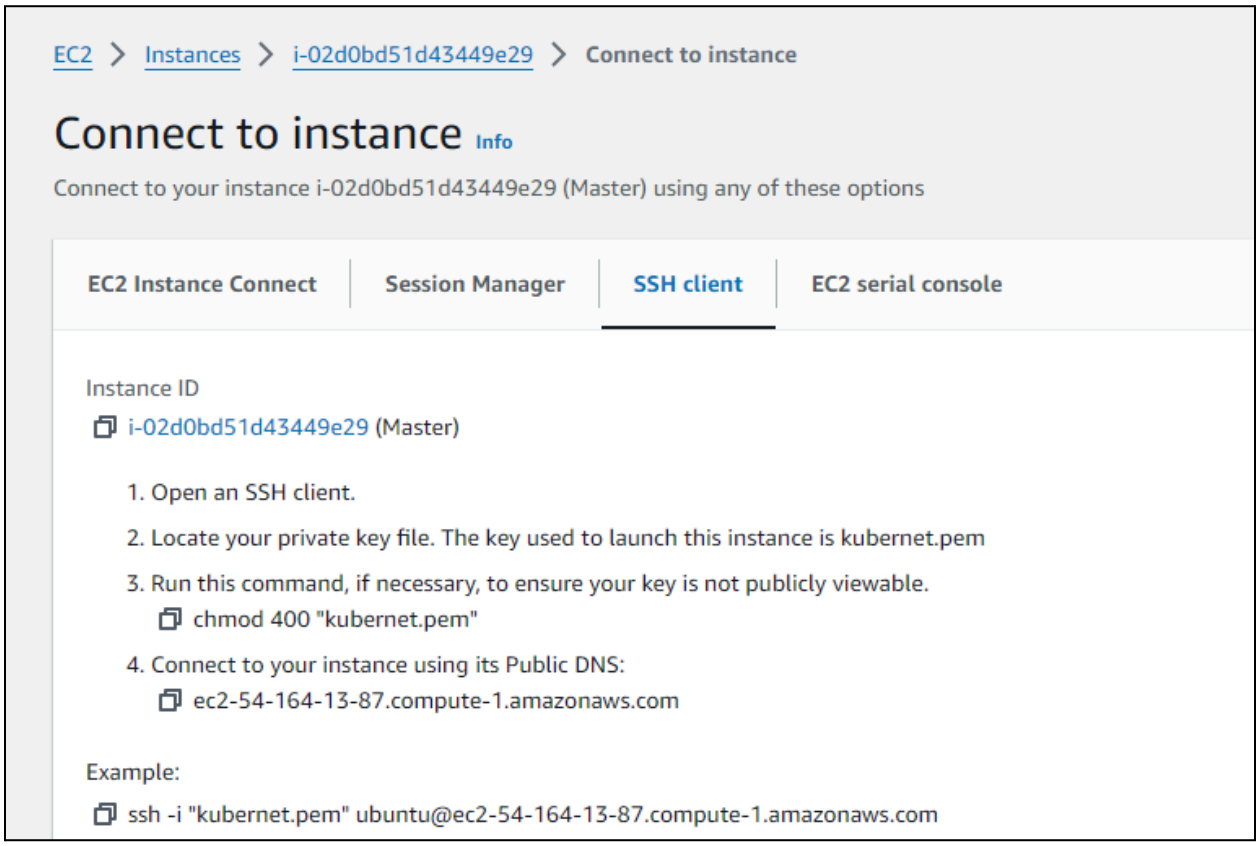


Created a master and 2 slaves:



2. Now click on connect to instance, then click on SSH client.

3. Now copy the ssh from the example and paste it on command prompt.(I used gitbash)



Commands:

4. Now since you are on GitBash, first type sudo su to perform the command as a root user.

5. After this type on all 3 machines
Yum install docker -y

```
[ec2-user@ip-172-31-84-37 ~]$ sudo su
[root@ip-172-31-84-37 ec2-user]# yum install docker -y
Last metadata expiration check: 0:18:22 ago on Thu Aug 29 08:52:52 2024.
Dependencies resolved.
==========================================================================================================================
 Package                     Architecture         Version                        Repository                  Size
==========================================================================================================================
Installing:
 docker                      x86_64               25.0.6-1.amzn2023.0.1          amazonlinux                 44 M
Installing dependencies:
 containerd                  x86_64               1.7.20-1.amzn2023.0.1          amazonlinux                 35 M
 iptables-libs               x86_64               1.8.8-3.amzn2023.0.2           amazonlinux                401 k
 iptables-nft                x86_64               1.8.8-3.amzn2023.0.2           amazonlinux                183 k
 libcgroup                   x86_64               3.0-1.amzn2023.0.1             amazonlinux                 75 k
 libnetfilter_conntrack      x86_64               1.0.8-2.amzn2023.0.2           amazonlinux                 58 k
 libnfnetlink                x86_64               1.0.1-19.amzn2023.0.2          amazonlinux                 30 k
 libnftnl                    x86_64               1.2.2-2.amzn2023.0.2           amazonlinux                 84 k
 pigz                        x86_64               2.5-1.amzn2023.0.3             amazonlinux                 83 k
 runc                        x86_64               1.1.11-1.amzn2023.0.1          amazonlinux                3.0 M
```

```
  Running scriptlet: docker-25.0.6-1.amzn2023.0.1.x86_64
  Installing       : docker-25.0.6-1.amzn2023.0.1.x86_64
  Running scriptlet: docker-25.0.6-1.amzn2023.0.1.x86_64
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

  Verifying        : containerd-1.7.20-1.amzn2023.0.1.x86_64
  Verifying        : docker-25.0.6-1.amzn2023.0.1.x86_64
  Verifying        : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  Verifying        : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
  Verifying        : libcgroup-3.0-1.amzn2023.0.1.x86_64
  Verifying        : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  Verifying        : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
  Verifying        : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  Verifying        : pigz-2.5-1.amzn2023.0.3.x86_64
  Verifying        : runc-1.1.11-1.amzn2023.0.1.x86_64

Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64      docker-25.0.6-1.amzn2023.0.1.x86_64      iptables-libs-1.8.8-3.amzn2023.0.2.x86_6
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64     libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64    libnftnl-1.2.2-2.amzn2023.0.2.x86_64     pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.1.11-1.amzn2023.0.1.x86_64

Complete!
```

6. To start the docker on master and slave perform this command:

Systemctl start docker

Extra

7. To check if docker is Installed successfully:

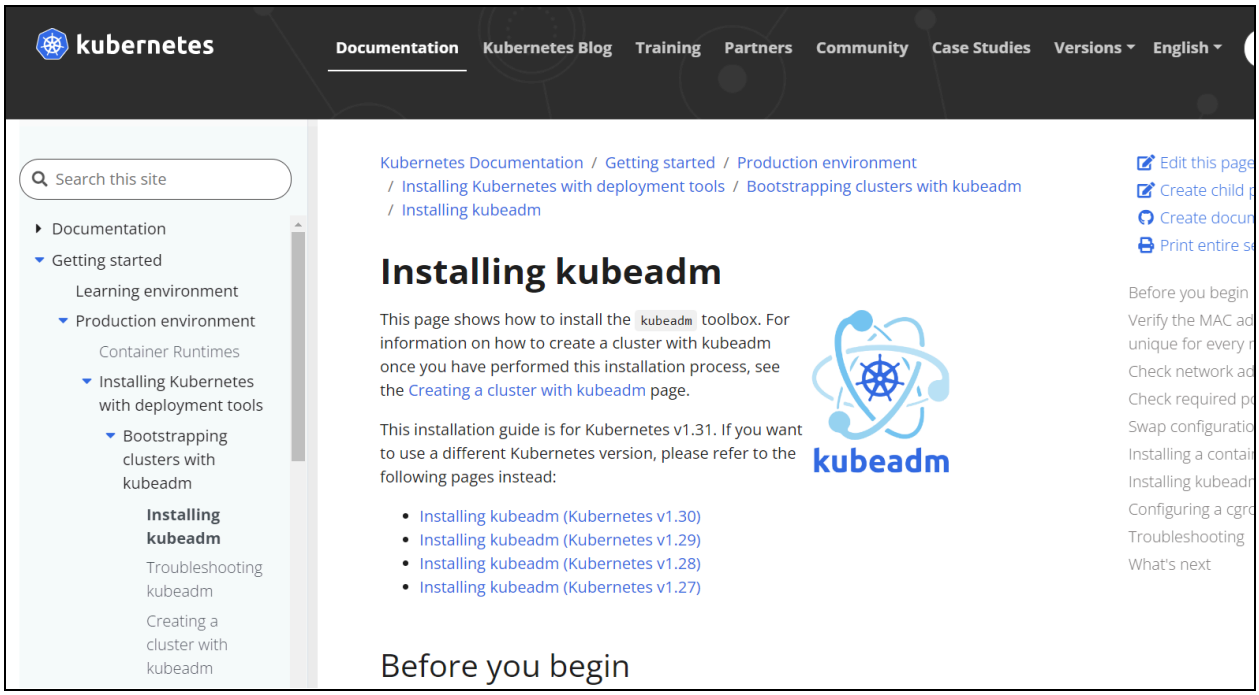
Docker -v or Docker –version

```
[root@ip-172-31-84-37 ec2-user]# systemctl start docker
[root@ip-172-31-84-37 ec2-user]# sudo su
[root@ip-172-31-84-37 ec2-user]# yum repolist
repo id                                                 repo name
amazonlinux                                             Amazon Linux 2023 repository
kernel-livepatch                                        Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-84-37 ec2-user]# docker --version
Docker version 25.0.5, build 5dc9bcc
```

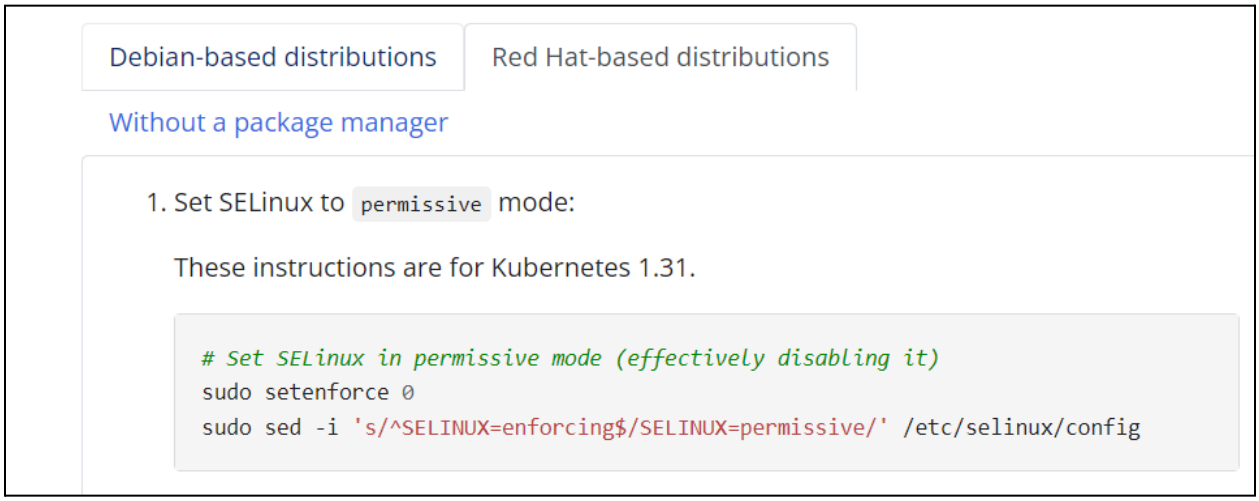
8. Now to install kubeadm on master and slaves :

Installing kubeadm:

Go the official documentation off kubeadm.



9. Scroll down and select Red Hat based distributions:



10. Now copy the command on all 3 machines:

Set SELinux to permissive mode:

These instructions are for Kubernetes 1.31.

```
# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

11. Now copy all the commands on the GitBash on all the 3 machines:

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

#Install kubelet, kubeadm and kubectl:

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

#(Optional) Enable the kubelet service before running kubeadm:

sudo systemctl enable --now kubelet
```

```
  Installing       : kubeadm-1.31.0-150500.1.1.x86_64
  Installing       : kubectl-1.31.0-150500.1.1.x86_64
  Running scriptlet: kubectl-1.31.0-150500.1.1.x86_64
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
  Verifying        : socat-1.7.4.2-1.amzn2023.0.2.x86_64
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64
  Verifying        : kubeadm-1.31.0-150500.1.1.x86_64
  Verifying        : kubectl-1.31.0-150500.1.1.x86_64
  Verifying        : kubelet-1.31.0-150500.1.1.x86_64
  Verifying        : kubernetes-cni-1.5.0-150500.2.1.x86_64

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                              cr
  kubeadm-1.31.0-150500.1.1.x86_64                                         ku
  kubelet-1.31.0-150500.1.1.x86_64                                         ku
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                       li
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                           so

Complete!
[root@ip-172-31-84-37 ec2-user]# sudo systemctl enable --now kubelet
```

12. Type yum repolist to check the repository of kubernetes

```
[root@ip-172-31-84-143 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                 Kubernetes
```

EXTRA

Got an error in initialization kubeadm

```
[root@ip-172-31-31-240 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0908 11:25:45.820964    2320 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking t
RI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containe
ble desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/containerd/containerd.sock:
        [WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

Error was resolved:
(after again starting from scratch)

13. Initialize the kubeadm by the command kubeadm init only on master:

Kubeadm initialized successfully:

```
[root@ip-172-31-26-66 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your interr
[preflight] You can also perform this action beforehand using 'kubeadm config imag
W0912 06:07:49.475553   28037 checks.go:846] detected that the sandbox image "regi
 that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-26-66.ec2.interr
efault.svc.cluster.local] and IPs [10.96.0.1 172.31.26.66]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
```

14. After this we will get 3 things:
    - The directory
    - Some export Statement
    - The most important thing - the token to connect the slaves with the master.

15. Copy them one by one and paste it on the slaves:

```
To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.26.66:6443 --token grw4r4.gb3kkhb7392dnvjp \
        --discovery-token-ca-cert-hash sha256:b61f1de7eedb2c0dc0cc237d4629e9631920b63dd6634c3e22e76aaa36d01920
```

16. After pasting type kubectl get nodes:

The nodes are connected successfully:

```
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-17-23    Ready    control-plane   3m56s   v1.29.0
ip-172-31-18-12    Ready    <none>          37s     v1.29.0
ip-172-31-26-153   Ready    <none>          24s     v1.29.0
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-17-23    Ready    control-plane   9m34s   v1.29.0
ip-172-31-18-12    Ready    <none>          6m15s   v1.29.0
ip-172-31-26-153   Ready    <none>          6m2s    v1.29.0
ubuntu@ip-172-31-17-23:~$
```

Advanced DevOps Lab
Experiment 4

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Theory:

Overview of Kubernetes and Kubectl

What is Kubernetes?

Kubernetes, often referred to as K8s, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Originally developed by Google, it has become the industry standard for managing container workloads due to its flexibility and robust features.

Core Concepts of Kubernetes

1. Containers: These are lightweight, portable packages that include everything needed to run an application, ensuring consistency across different environments.
2. Pods: The smallest deployable units in Kubernetes, pods can contain one or more containers that share storage and network resources.
3. Nodes: A node is a worker machine in the Kubernetes cluster that runs at least one pod. Nodes can be either physical or virtual machines.
4. Clusters: A cluster comprises multiple nodes that run containerized applications. The control plane manages the cluster's state.
5. Services: Services provide stable endpoints for accessing pods and facilitate load balancing and service discovery.
6. Deployments: A deployment manages the lifecycle of pods, allowing users to specify the number of replicas and facilitating rolling updates and rollbacks.

Architecture of Kubernetes

Kubernetes follows a client-server architecture consisting of:

- Control Plane: Manages the cluster and includes components like the API server (the front-end for the control plane), scheduler (assigns pods to nodes), controller manager (regulates cluster state), and etcd (a distributed key-value store for cluster data).
- Worker Nodes: Each node runs components like kubelet (ensures containers are running), kube-proxy (manages network communication), and a container runtime (e.g., Docker).

Role of Kubectl in Kubernetes

What is Kubectl?

Kubectl is the command-line interface used to interact with the Kubernetes API server. It enables users to manage resources within a Kubernetes cluster effectively.

Key Functions of Kubectl

1. Resource Management: Users can create, update, delete, and retrieve information about various resources such as deployments, services, and pods.
2. Configuration Management: Users can apply configuration files written in YAML or JSON format to define resource structures and behaviors.
3. Monitoring and Debugging: Kubectl allows users to inspect resource statuses, view logs from containers, and describe resource configurations for troubleshooting.
4. Access Control: Supports role-based access control (RBAC) to define permissions for users interacting with the cluster.
5. Namespace Management: Facilitates the creation and management of namespaces to organize resources across teams or projects.

Configuration Files

Configuration files are essential for defining how resources should be created or modified within Kubernetes. Users can employ declarative configurations (using YAML/JSON files) or imperative commands directly in the terminal.

Deploying Applications on Kubernetes

Application Deployment Lifecycle

1. Define Application Requirements: Identify necessary resources such as CPU, memory, storage, etc.
2. Create Deployment Configurations: Write deployment manifests specifying container images, replicas for scaling, health checks, etc.
3. Deploying with Kubectl: Use kubectl commands like kubectl apply to deploy applications based on these configurations.
4. Monitoring and Scaling Applications: Monitor performance metrics and adjust deployments based on traffic demands.
5. Updating Applications: Modify deployment configurations for updates; Kubernetes supports rolling updates by default.
6. Rollback Capabilities: If an update causes issues, kubectl allows easy rollback to previous versions using commands like kubectl rollout undo.

Best Practices for Application Deployment

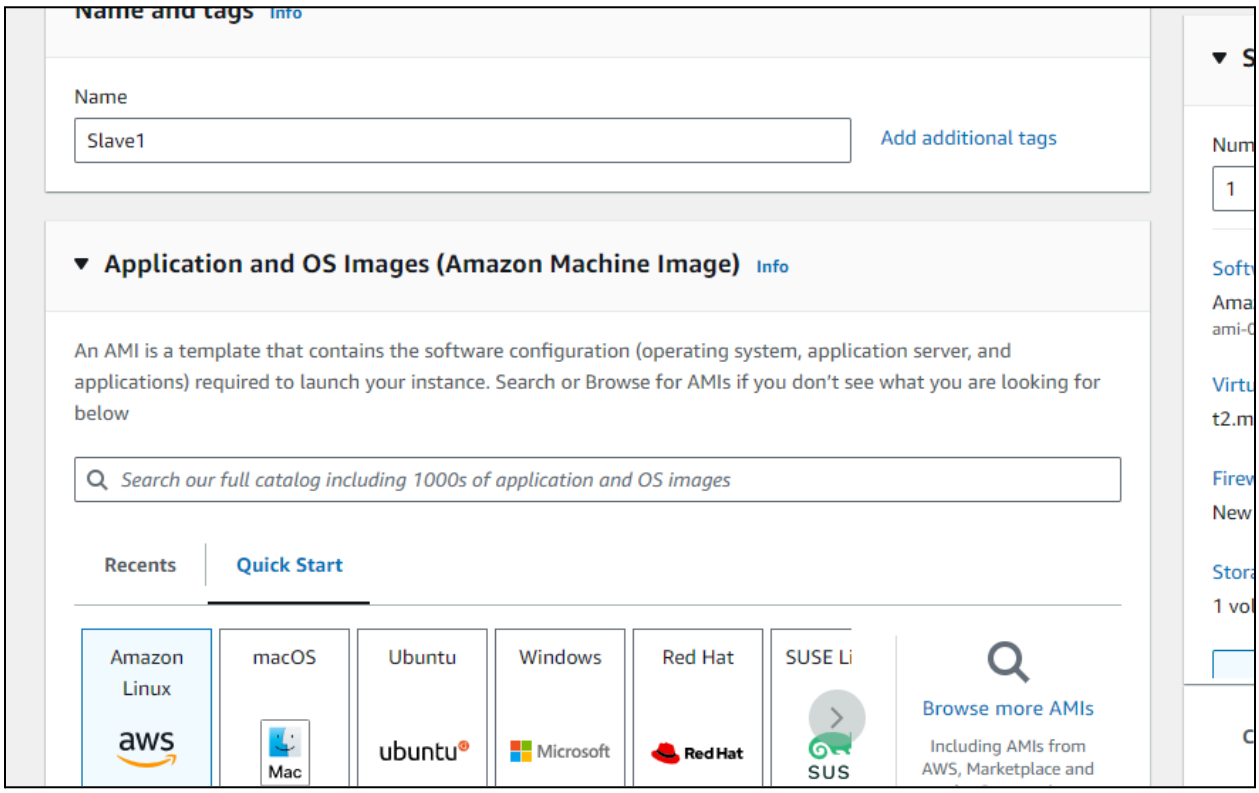
- Use versioned images for consistency.
- Implement health checks to manage application availability.
- Utilize namespaces for better organization.

- Regularly monitor resource usage and adjust accordingly.
- Automate deployment processes using CI/CD pipelines integrated with kubectl commands.

Steps:

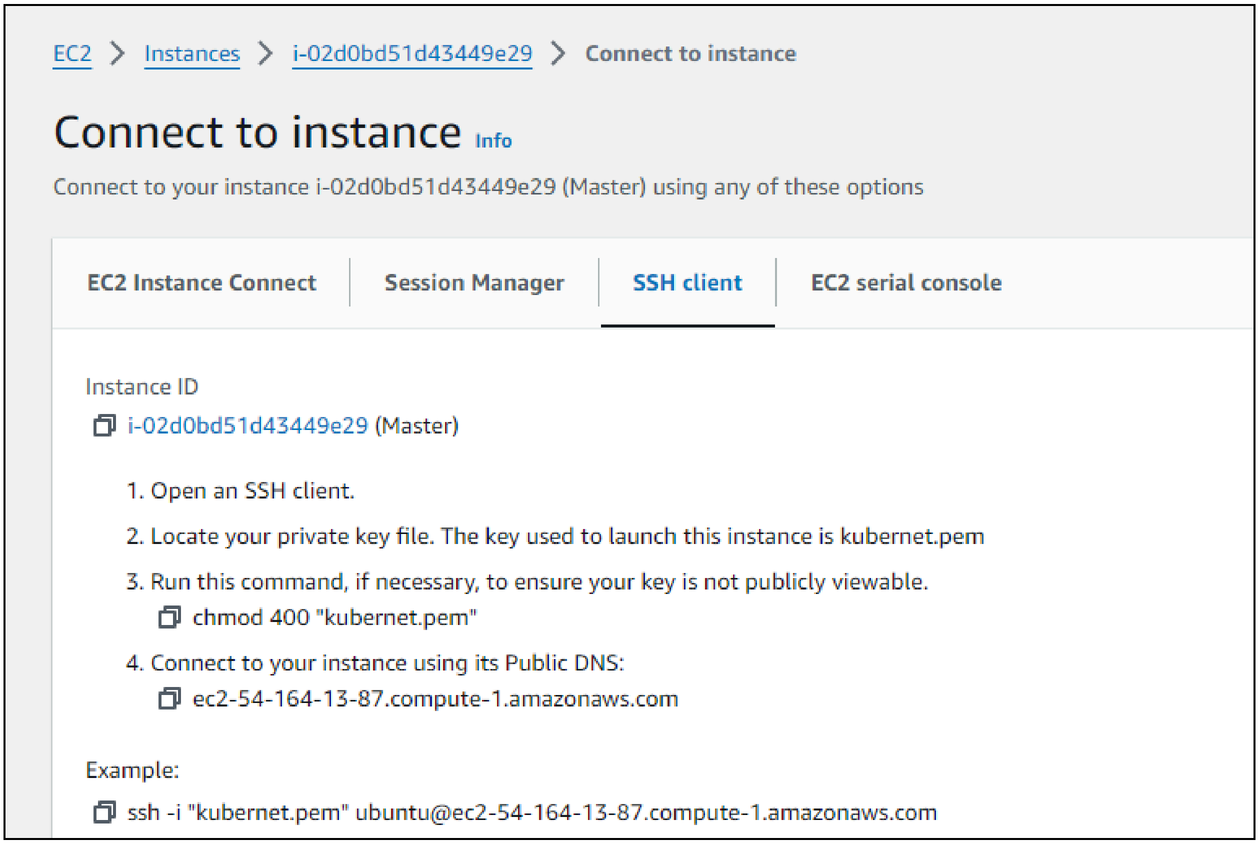
1. Create 3 EC2 Ubuntu Instances on AWS.

Extra:
When we select ubuntu we have to select an older version - 22.04.
(Name 1 as Master, the other 2 as Slave1 and Slave2)



2. Now click on connect to instance, then click on SSH client.

3. Now copy the ssh from the example and paste it on command prompt.(I used gitbash)



Commands:

4. Now since you are on GitBash, first type sudo su to perform the command as a root user.

5. After this type on GitBash
Yum install docker -y

```
[ec2-user@ip-172-31-84-37 ~]$ sudo su
[root@ip-172-31-84-37 ec2-user]# yum install docker -y
Last metadata expiration check: 0:18:22 ago on Thu Aug 29 08:52:52 2024.
Dependencies resolved.
==========================================================================================================================
 Package                     Architecture       Version                          Repository                      Size
==========================================================================================================================
Installing:
 docker                      x86_64             25.0.6-1.amzn2023.0.1            amazonlinux                     44 M
Installing dependencies:
 containerd                  x86_64             1.7.20-1.amzn2023.0.1            amazonlinux                     35 M
 iptables-libs               x86_64             1.8.8-3.amzn2023.0.2             amazonlinux                    401 k
 iptables-nft                x86_64             1.8.8-3.amzn2023.0.2             amazonlinux                    183 k
 libcgroup                   x86_64             3.0-1.amzn2023.0.1               amazonlinux                     75 k
 libnetfilter_conntrack      x86_64             1.0.8-2.amzn2023.0.2             amazonlinux                     58 k
 libnfnetlink                x86_64             1.0.1-19.amzn2023.0.2            amazonlinux                     30 k
 libnftnl                    x86_64             1.2.2-2.amzn2023.0.2             amazonlinux                     84 k
 pigz                        x86_64             2.5-1.amzn2023.0.3               amazonlinux                     83 k
 runc                        x86_64             1.1.11-1.amzn2023.0.1            amazonlinux                    3.0 M
```

```
  Running scriptlet: docker-25.0.6-1.amzn2023.0.1.x86_64
  Installing       : docker-25.0.6-1.amzn2023.0.1.x86_64
  Running scriptlet: docker-25.0.6-1.amzn2023.0.1.x86_64
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

  Verifying        : containerd-1.7.20-1.amzn2023.0.1.x86_64
  Verifying        : docker-25.0.6-1.amzn2023.0.1.x86_64
  Verifying        : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  Verifying        : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
  Verifying        : libcgroup-3.0-1.amzn2023.0.1.x86_64
  Verifying        : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  Verifying        : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
  Verifying        : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  Verifying        : pigz-2.5-1.amzn2023.0.3.x86_64
  Verifying        : runc-1.1.11-1.amzn2023.0.1.x86_64

Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64       docker-25.0.6-1.amzn2023.0.1.x86_64      iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64      libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.0
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64     libnftnl-1.2.2-2.amzn2023.0.2.x86_64     pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.1.11-1.amzn2023.0.1.x86_64

Complete!
```

6. To start the docker perform this command:

Systemctl start docker

Extra

7. To check if docker is Installed successfully:

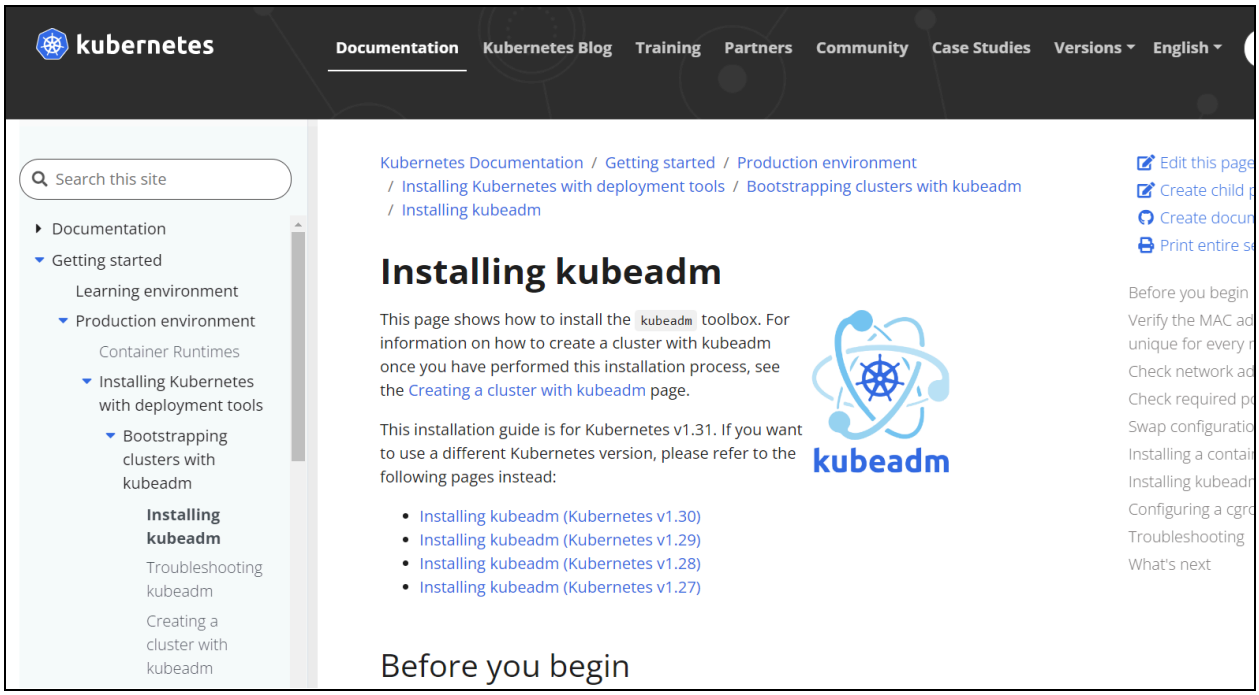
Docker -v or Docker –version

```
[root@ip-172-31-84-37 ec2-user]# systemctl start docker
[root@ip-172-31-84-37 ec2-user]# sudo su
[root@ip-172-31-84-37 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-84-37 ec2-user]# docker --version
Docker version 25.0.5, build 5dc9bcc
```

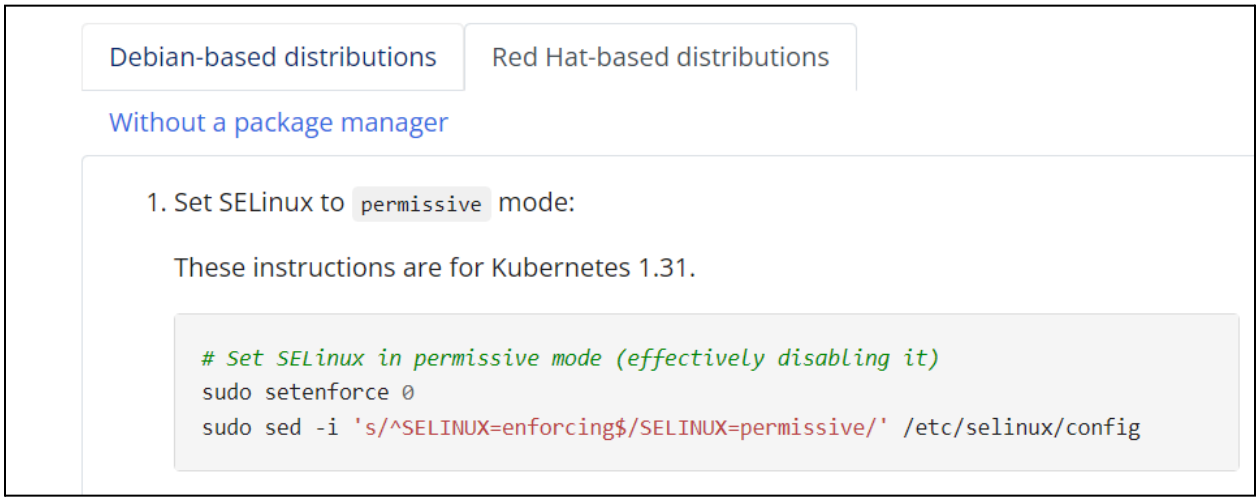
8. Now to install kubeadm :

Installing kubeadm:

Go the official documentation off kubeadm.



9. Scroll down and select Red Hat based distributions:



10. Now copy the command:

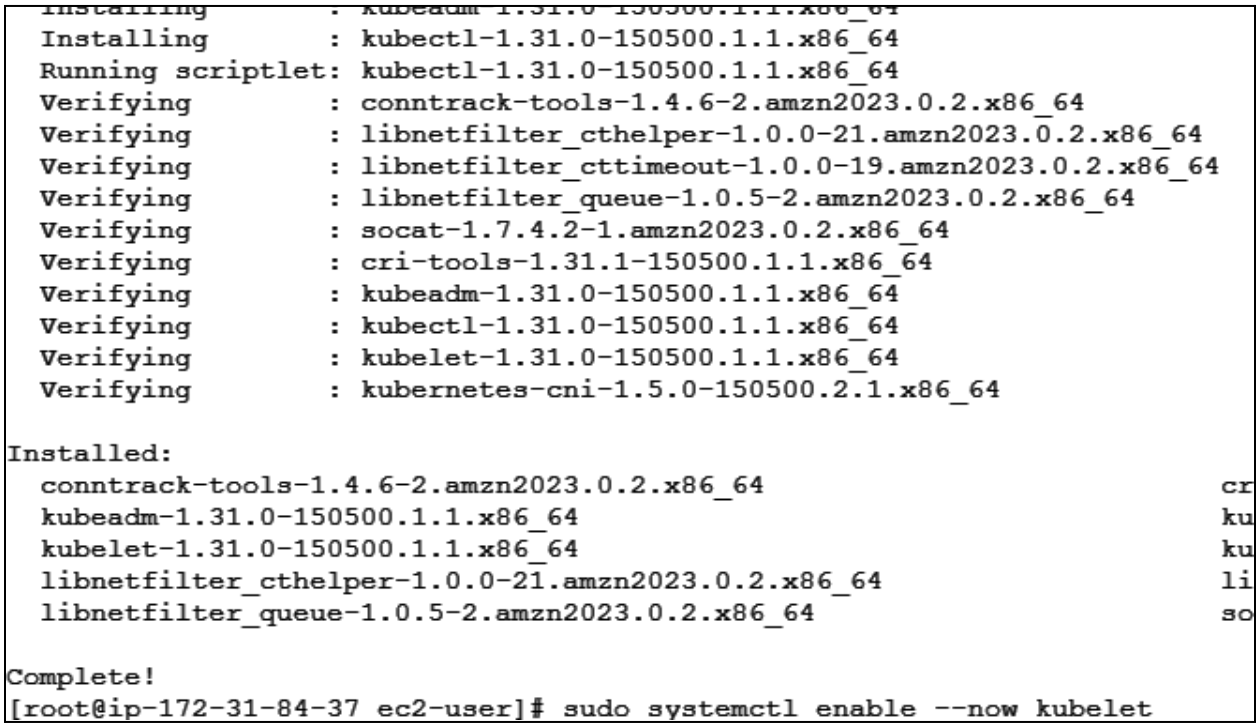
Set SELinux to permissive mode:

These instructions are for Kubernetes 1.31.

```
# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

11. Now copy all the commands on the GitBash:

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

#Install kubelet, kubeadm and kubectl:

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

#(Optional) Enable the kubelet service before running kubeadm:

sudo systemctl enable --now kubelet
```



12. Type yum repolist to check the repository of kubernetes

```
[root@ip-172-31-84-143 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                 Kubernetes
```

EXTRA

Got an error in initialization kubeadm

```
[root@ip-172-31-31-240 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0908 11:25:45.820964    2320 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking t
RI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containe
ble desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/containerd/containerd.sock:
        [WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

Error was resolved:
(after again starting from scratch)

13. Initialize the kubeadm by the command kubeadm init :

Kubeadm initialized successfully:

```
[root@ip-172-31-26-66 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your interr
[preflight] You can also perform this action beforehand using 'kubeadm config imag
W0912 06:07:49.475553   28037 checks.go:846] detected that the sandbox image "regi
 that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-26-66.ec2.interr
efault.svc.cluster.local] and IPs [10.96.0.1 172.31.26.66]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
```

14. After this we will get 3 things:
    - The directory
    - Some export Statement
    - The most important thing - the token to connect the slaves with the master.

15. Copy them

```
To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.26.66:6443 --token grw4r4.gb3kkhb7392dnvjp \
        --discovery-token-ca-cert-hash sha256:b61f1de7eedb2c0dc0cc237d4629e9631920b63dd6634c3e22e76aaa36d01920
```

16. After pasting type kubectl get nodes:

The nodes are connected successfully:

```
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-17-23    Ready    control-plane   3m56s   v1.29.0
ip-172-31-18-12    Ready    <none>          37s     v1.29.0
ip-172-31-26-153   Ready    <none>          24s     v1.29.0
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-17-23    Ready    control-plane   9m34s   v1.29.0
ip-172-31-18-12    Ready    <none>          6m15s   v1.29.0
ip-172-31-26-153   Ready    <none>          6m2s    v1.29.0
ubuntu@ip-172-31-17-23:~$
```

17. Create two YAML files named nginx-deployment.yaml and nginx-service.yaml
(I used nano editor for the same)

```
ubuntu@ip-172-31-17-23:~$ nano nginx-deployment.yaml
ubuntu@ip-172-31-17-23:~$ nano nginx-service.yaml
```

18. Then add the deployment and service configuration in it, respectively:

Deployment:

```
  GNU nano 6.2                        nginx-deployment.yaml *
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21.3
        ports:
        - containerPort: 80


^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

Service:

```
  GNU nano 6.2                         nginx-service.yaml *
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer




^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

19. Now since we have configured our files we would now proceed for applying both the deployment and the service files.

Deployment :

```
ubuntu@ip-172-31-17-23:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
```

Service:

```
ubuntu@ip-172-31-17-23:~$ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

20. After deployment its time for verifying the same:

For deployment:

```
ubuntu@ip-172-31-17-23:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx              1/1     1            1           14m
nginx-deployment   2/2     2            2           39s
```

For services:

```
ubuntu@ip-172-31-17-23:~$ kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT
(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/
TCP        70m
nginx           NodePort       10.109.245.143   <none>        80:3
0306/TCP   37m
nginx-service   LoadBalancer   10.99.247.105    <pending>     80:3
1130/TCP   36s
```

For pods:

```
ubuntu@ip-172-31-17-23:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AG
E
nginx-7854ff8877-mxrqg              1/1     Running   0          15
m
nginx-deployment-6b4d6fdbf-5rb6h    1/1     Running   0          65
s
nginx-deployment-6b4d6fdbf-6q2jj    1/1     Running   0          65
s
```

Extra:

```
ubuntu@ip-172-31-17-23:~$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   55m
kube-node-lease   Active   55m
kube-public       Active   55m
kube-system       Active   55m
```

21. Now Lastly, port forward the deployment to your localhost so that you can view it.

```
ubuntu@ip-172-31-17-23:~$ kubectl port-forward service/nginx 8080:
80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

22. You can open the browser and check on

    http://localhost:8080

Experiment 5

AIM: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine and Windows.

Theory:

Terraform is an infrastructure as code (IaC) tool developed by HashiCorp that allows users to define and manage cloud and on-premises resources using declarative configuration files. Understanding its lifecycle and core concepts is essential for effectively utilizing Terraform in managing infrastructure.

Terraform Lifecycle

Terraform manages resources through a defined lifecycle that includes three primary stages:

1. Apply (Create): This stage involves creating the resource as defined in the configuration files. Terraform communicates with the cloud provider's API to provision the resource.
2. Update: In this stage, Terraform makes modifications to existing resources. This can involve incremental changes or complete replacements, depending on the nature of the update.
3. Destroy: This final stage removes the resource from the environment, ensuring that it is no longer managed by Terraform.

Terraform also provides a lifecycle meta-argument that allows users to control these stages more granularly. Options within this meta-argument include:

- create_before_destroy: Ensures a new instance is created before the old one is destroyed, which is useful for maintaining service availability during updates.
- prevent_destroy: Prevents accidental deletion of critical resources.
- ignore_changes: Allows Terraform to ignore changes made outside of its management for specified fields or entire objects.

Core Concepts

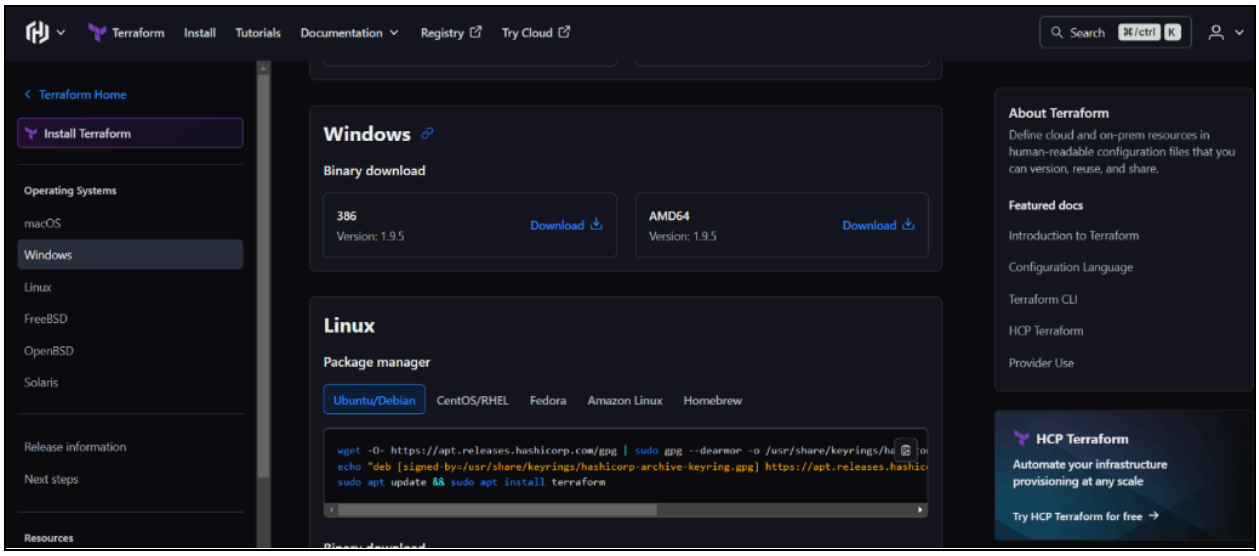
Several key concepts underpin Terraform's functionality:

- Configuration Files: These files, written in HashiCorp Configuration Language (HCL), define the desired state of the infrastructure. They describe resources, their properties, and relationships.
- Resources: The fundamental building blocks in Terraform, representing various infrastructure components like virtual machines, storage accounts, and networking configurations.
- Data Sources: These provide external data to Terraform configurations, allowing users to reference existing resources or configurations from other projects.
- Modules: Modules are reusable packages of Terraform configurations that enable users to group related resources for better organization and reusability. They can be shared across projects and teams, promoting best practices and consistency.
- State Management: Terraform maintains a state file that acts as a source of truth for the infrastructure. This file tracks the current state of resources, enabling Terraform to determine the necessary changes during the apply phase.
- Execution Plan: Before applying changes, Terraform generates an execution plan that outlines what actions will be taken, allowing users to review and approve changes before they are executed.

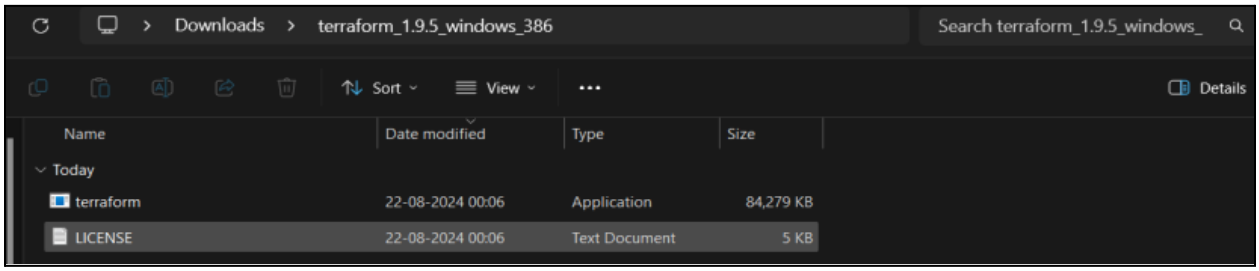
By mastering these concepts and understanding the lifecycle of resources, users can effectively leverage Terraform to automate and manage their infrastructure in a safe and efficient manner.

Implementation:

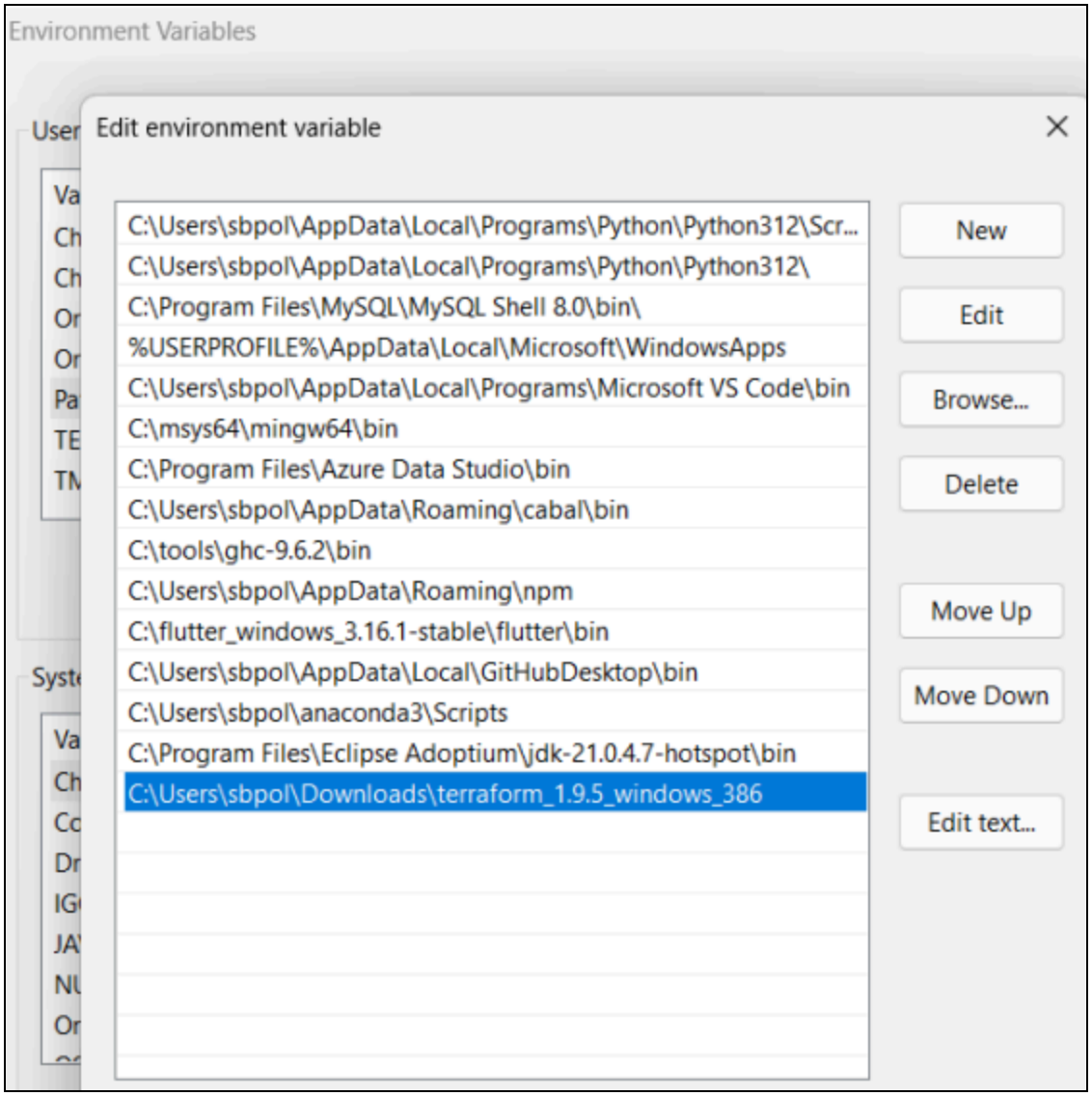
Step 1: Go to Terraform website and download 386 .



Step 2 : Extract the zip file and copy the path of the file.



Step 3: Edit environment variables and paste copied path.



Step 4 : Make sure Terraform is installed successfully.

```
(base) PS C:\Users\sbpol> terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
```

```
Install the latest PowerShell for new features an

Loading personal and system profiles took 1476ms.
(base) PS C:\Users\sbpol> terraform --version
Terraform v1.9.5
on windows_386
```

Experiment 6

Aim :To Build, change, and destroy AWS infrastructure Using Terraform (S3 bucket or Docker) .

Theory :

Terraform is an open-source tool that enables developers and operations teams to define, provision, and manage cloud infrastructure through code. It uses a declarative language to specify the desired state of infrastructure, which can include servers, storage, networking components, and more. With Terraform, infrastructure changes can be automated, versioned, and tracked efficiently.

Building Infrastructure

When you build infrastructure using Terraform, you define the desired state of your infrastructure in configuration files. For example, you may want to create an S3 bucket or deploy a Docker container on an EC2 instance. Terraform reads these configuration files and, using the specified cloud provider (such as AWS), it provisions the necessary resources to match the desired state.

- S3 Buckets: Terraform can create and manage S3 buckets, which are used to store and retrieve data objects in the cloud. You can define the properties of the bucket, such as its name, region, access permissions, and versioning.

- Docker on AWS: Terraform can deploy Docker containers on AWS infrastructure. This often involves setting up an EC2 instance and configuring it to run Docker containers, which encapsulate applications and their dependencies.

Changing Infrastructure

As your needs evolve, you may need to modify the existing infrastructure. Terraform makes it easy to implement changes by updating the configuration files to reflect the new desired state. For instance, you might want to change the storage settings of an S3 bucket, add new security policies, or modify the Docker container's configuration.

Terraform's "plan" command helps you preview the changes that will be made to your infrastructure before applying them. This step ensures that you understand the impact of your changes and can avoid unintended consequences.

Destroying Infrastructure

When certain resources are no longer needed, Terraform allows you to destroy them in a controlled manner. This might involve deleting an S3 bucket or terminating an EC2 instance running Docker containers. By running the "destroy" command, Terraform ensures that all associated resources are properly de-provisioned and removed.

Destroying infrastructure with Terraform is beneficial because it helps avoid unnecessary costs associated with unused resources and ensures that the environment remains clean and free of clutter.

Benefits of Using Terraform for AWS Infrastructure

1.Consistency: Terraform ensures that infrastructure is consistent across environments by applying the same configuration files.

2.Automation: Manual processes are reduced, and infrastructure is provisioned, updated, and destroyed automatically based on code.

3.Version Control: Infrastructure configurations can be stored in version control systems (like Git), allowing teams to track changes, collaborate, and roll back if necessary.

4.Scalability: Terraform can manage complex infrastructures, scaling them up or down as needed, whether for small projects or large-scale applications.

5.Modularity: Terraform configurations can be broken down into reusable modules, making it easier to manage and scale infrastructure.

Implementation :

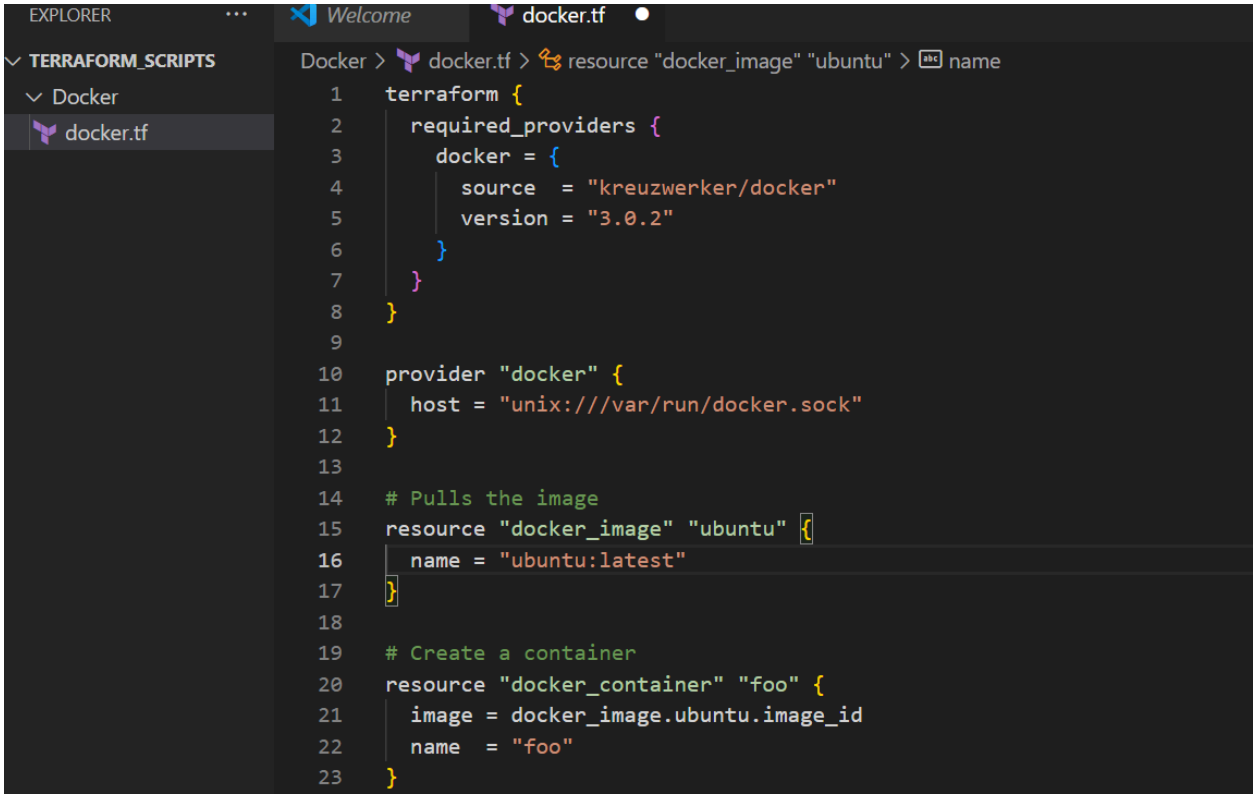
Terraform and Docker -

Step 1: Check the docker functionality:

```
PS C:\Users\272241> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image
```

```
PS C:\Users\272241> docker --version
Docker version 27.0.3, build 7d4bcd8
PS C:\Users\272241>
```

Step 2:



Step 3: Executed the terraform init command.

```
PS C:\Users\272241\Terraform_Scripts> cd Docker
PS C:\Users\272241\Terraform_Scripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step 4: Execute the terraform plan to see the resources.

```
PS C:\Users\272241\Terraform_Scripts\Docker> terraform plan

Terraform used the selected providers to generate the following execution plan
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
```

```
      + stdin_open                                  = false
      + stop_signal                                 = (known after apply)
      + stop_timeout                                = (known after apply)
      + tty                                         = false
      + wait                                        = false
      + wait_timeout                                = 60

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + name        = "ubuntu:latest"
      + repo_digest = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

(Extra)

Step 5: Use terraform validate to check if the code is perfect to build further and apply the changes.

```
PS C:\Users\272241\Terraform_Scripts\Docker> terraform validate
Success! The configuration is valid.
```

Step 6: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration.

Using command :“terraform apply”

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
  + create

Terraform will perform the following actions:

  # docker_container.nginx will be created
  + resource "docker_container" "nginx" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
      + init                                        = (known after apply)
      + ipc_mode                                    = (known after apply)
      + log_driver                                  = (known after apply)
      + logs                                        = false
      + must_run                                    = true
      + name                                        = "tutorial"
      + network_data                                = (known after apply)
      + read_only                                   = false
      + remove_volumes                              = true
      + restart                                     = "no"
      + rm                                          = false
      + runtime                                     = (known after apply)
      + security_opts                               = (known after apply)
      + shm_size                                    = (known after apply)
      + start                                       = true
      + stdin_open                                  = false
      + stop_signal                                 = (known after apply)
      + stop_timeout                                = (known after apply)
      + tty                                         = false
      + wait                                        = false
      + wait_timeout                                = 60
```

```
      + tty                                    = false
      + wait                                   = false
      + wait_timeout                           = 60

      + healthcheck (known after apply)

      + labels (known after apply)

      + ports {
          + external = 8000
          + internal = 80
          + ip       = "0.0.0.0"
          + protocol = "tcp"
        }
    }

  # docker_image.nginx will be created
  + resource "docker_image" "nginx" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + keep_locally = false
      + name        = "nginx:latest"
      + repo_digest = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

docker_image.nginx: Creating...
docker_image.nginx: Still creating... [10s elapsed]
docker_image.nginx: Creation complete after 19s [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cng
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=c25805e4484164520912c50ac3080526c9926219c98c673021078772eb484357]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Step 7: Docker images before applying the changes

```
PS C:\Users\272241\Terraform_Scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Step 8: Docker images after applying the changes

```
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago    78.1MB
```

Step 9: Now Terraform Destroy to delete the image ubuntu

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform destroy
docker_image.nginx: Refreshing state... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:la
docker_container.nginx: Refreshing state... [id=c648cc3dd8129abf9acb7cb06dfdd0aa9bafb0c7973f16695cd06a7ad447c631]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the fol
  - destroy

Terraform will perform the following actions:

  # docker_container.nginx will be destroyed
  - resource "docker_container" "nginx" {
      - attach                                      = false -> null
      - command                                     = [
          - "nginx",
          - "-g",
          - "daemon off;",
        ] -> null
      - container_read_refresh_timeout_milliseconds = 15000 -> null
      - cpu_shares                                  = 0 -> null
      - dns                                         = [] -> null
      - dns_opts                                    = [] -> null
      - dns_search                                  = [] -> null
      - entrypoint                                  = [
          - "/docker-entrypoint.sh",
        ] -> null
      - env                                         = [] -> null
      - group_add                                   = [] -> null
      - hostname                                    = "c648cc3dd812" -> null
      - id                                          = "c648cc3dd8129abf9acb7cb06dfdd0aa9bafb0c7973f16695cd06a7ad447c631" ->
      - image                                       = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda
      - init                                        = false -> null
      - ipc_mode                                    = "private" -> null
      - log_driver                                  = "json-file" -> null
      - log_opts                                    = {} -> null
      - logs                                        = false -> null
      - max_retry_count                             = 0 -> null
      - memory                                      = 0 -> null
      - memory_swap                                 = 0 -> null
      - stop_timeout                                = 0 -> null
      - storage_opts                                = {} -> null
      - sysctls                                     = {} -> null
      - tmpfs                                       = {} -> null
      - tty                                         = false -> null
      - wait                                        = false -> null
      - wait_timeout                                = 60 -> null
        # (7 unchanged attributes hidden)

      - ports {
          - external = 8000 -> null
          - internal = 80 -> null
          - ip       = "0.0.0.0" -> null
          - protocol = "tcp" -> null
        }
    }

  # docker_image.nginx will be destroyed
  - resource "docker_image" "nginx" {
      - id           = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest" ->
      - image_id     = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c" -> null
      - keep_locally = false -> null
      - name         = "nginx:latest" -> null
      - repo_digest  = "nginx@sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add" -> null
    }

Plan: 0 to add, 0 to change, 2 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

docker_container.nginx: Destroying... [id=c648cc3dd8129abf9acb7cb06dfdd0aa9bafb0c7973f16695cd06a7ad447c631]
docker_container.nginx: Destruction complete after 1s
docker_image.nginx: Destroying... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cngi
docker_image.nginx: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

Step 10 : Docker after terraform destroy command.

```
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
```

Terraform and S3 -

Step 1: Open VS Code and also log in to your aws account.

Step 2 : Type below code in main.tf in editor for aws and terraform connection and environment creation .

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
     random = {
      source = "hashicorp/random"
      version = "3.6.2"
    }
  }
}

resource "random_id" "ran_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "demo-bucket" {
  bucket = "my-demo-bucket-${random_id.ran_id.hex}"
}

resource "aws_s3_object" "bucket-data" {
  bucket = aws_s3_bucket.demo-bucket.bucket
  source = "./myfile.txt"
```

```
  key = "newfile.txt"

}
```

aws-s3 > main.tf > terraform > required_providers > random >

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
    random = {
      source = "hashicorp/random"
      version = "3.6.2"
    }
  }
}

resource "random_id" "ran_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "demo-bucket" {
  bucket = "my-demo-bucket-${random_id.ran_id.hex}"
}

resource "aws_s3_object" "bucket-data" {
  bucket = aws_s3_bucket.demo-bucket.bucket
  source = "./myfile.txt"
  key = "newfile.txt"

}
```

Step 3 : Type terraform init command in powershell.

```
C:\Users\272241\New folder\aws-s3>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.64.0"...
- Installing hashicorp/aws v5.64.0...
- Installed hashicorp/aws v5.64.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record th
rovider
selections it made above. Include this file in your version contro
epository
so that Terraform can guarantee to make the same selections by def
t when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform p
" to see
any changes that are required for your infrastructure. All Terrafo
commands
should now work.

If you ever set or change modules or backend configuration for Ter
orm,
rerun this command to reinitialize your working directory. If you
get  other
```

Step 4 : Type terraform plan command in powershell.

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.terr will be created
  + resource "aws_s3_bucket" "terr" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "my-tf-test-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + tags_all                    = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)
```

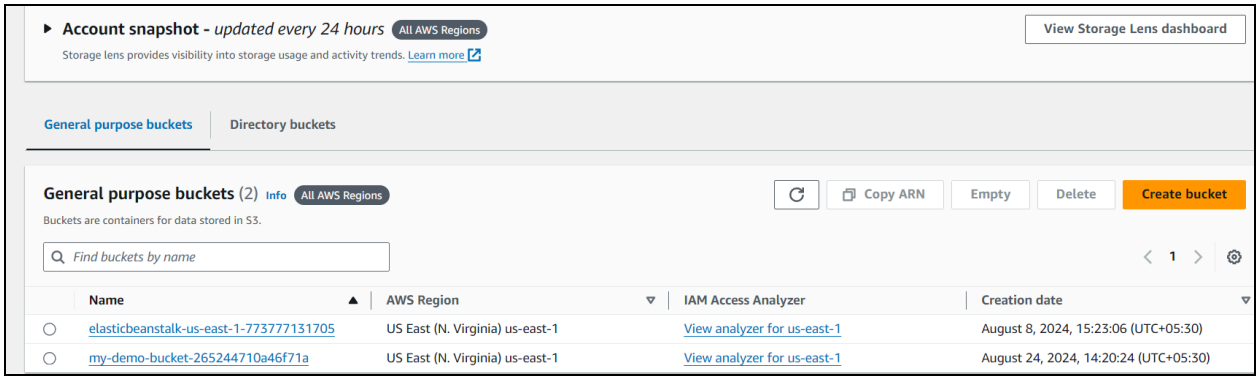
Step 5 Type terraform validate

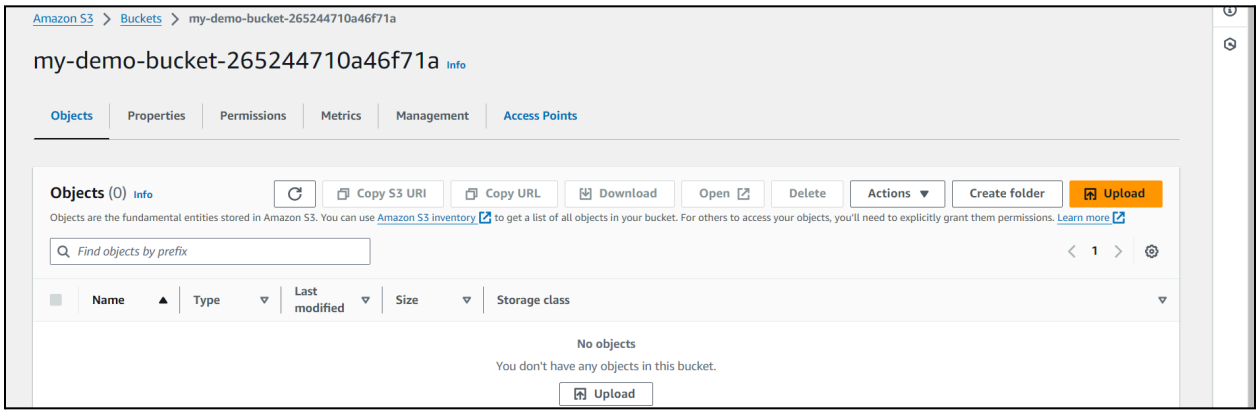
```
C:\Users\272241\New folder\aws-ec2>terraform validate
Success! The configuration is
valid.
```

Step 6 : Type terraform apply command in powershell

```
      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after ap

      + versioning (known after apply)

      + website (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket.demo-bucket: Creating...
aws_s3_bucket.demo-bucket: Creation complete after 7s [id=my
ket-265244710a46f71a]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Step 7 : Now check AWS to see the bucket created:





Step 8(EXTRA) :Terraform plan and apply command to apply the changes for file.

```
      + key                    = "newfile.txt"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "./myfile.txt"
      + storage_class          = (known after apply)
      + tags_all               = (known after apply)
      + version_id             = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_object.bucket-data: Creating...
aws_s3_object.bucket-data: Creation complete after 3s [id=newfile.txt
]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Step 9: Check the s3 bucket created before and after uploading the file.

Step 10 : Terraform destroy command to destroy the s3 bucket.

```
Plan: 3 to add, 0 to change, 2 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_object.bucket-data: Destroying... [id=newfile.txt]
aws_s3_object.bucket-data: Destruction complete after 1s
aws_s3_bucket.demo-bucket: Destroying... [id=my-demo-bucket-265244710
a46f71a]
aws_s3_bucket.demo-bucket: Destruction complete after 2s
random_id.ran_id: Creating...
random_id.ran_id: Creation complete after 0s [id=vIyCbfGVC9A]
aws_s3_bucket.demo-bucket: Creating...
aws_s3_bucket.demo-bucket: Creation complete after 7s [id=my-demo-buc
ket-bc8c826df1950bd0]
aws_s3_object.bucket-data: Creating...
aws_s3_object.bucket-data: Creation complete after 2s [id=newfile.txt
]

Apply complete! Resources: 3 added, 0 changed, 2 destroyed.
```

(EXTRA)

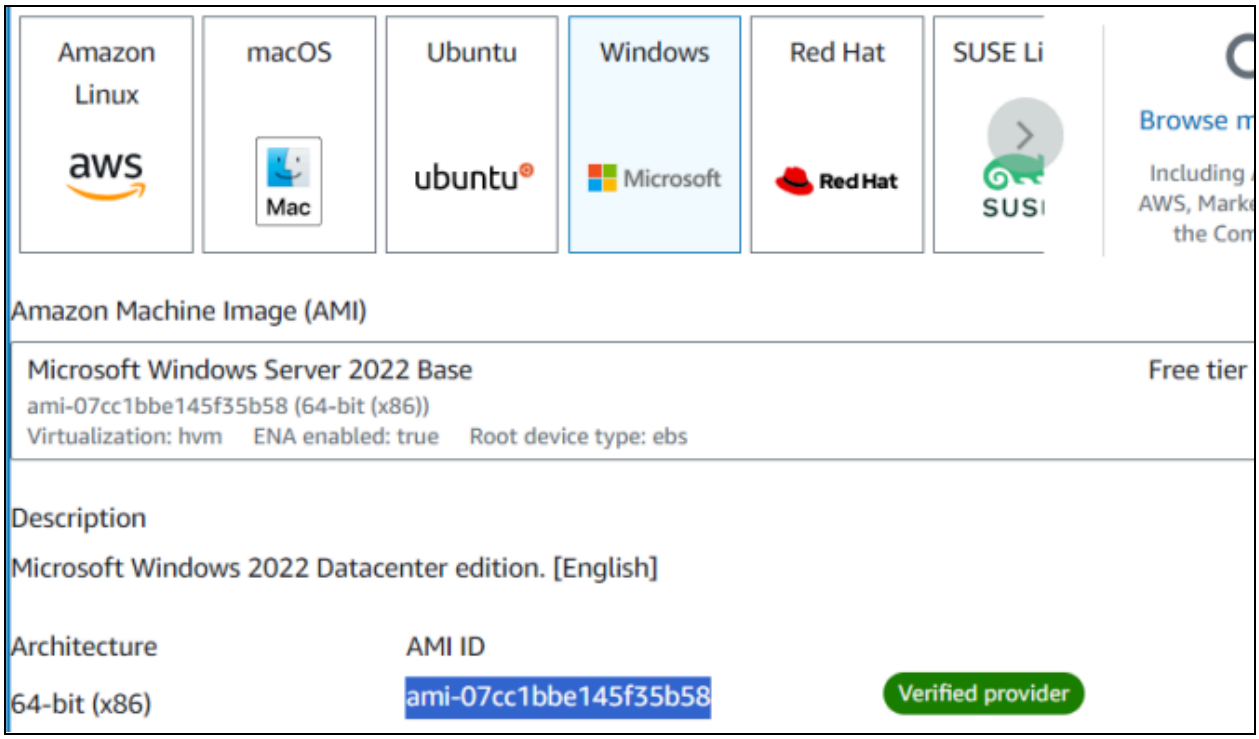
EC2:

Creating EC2 instance using Terraform

Step 1 : connect the aws academy and terraform using the credentials

```
eee_W_3413358@runweb131733:~$ ^V
bash: $'\026': command not found
eee_W_3413358@runweb131733:~$ export AWS_ACCESS_KEY_ID="ASIAZG6JVYHRLQ7XABVF"
eee_W_3413358@runweb131733:~$ export AWS_SECRET_ACCESS_KEY="FV+B+/JDLgRHpPs2bLr9jB+835PQ4cyz7HQ4LAzR"
eee_W_3413358@runweb131733:~$ export AWS_SESSION_TOKEN= "IQoJb3JpZ2luX2VjELT//////////wEaCXVzLXdlc3QtMiJGMEQCIGM45rz6G
OsZBjBcMcCWfAJetwP1F2qgToQCSoJbLE+HAiB2t1XfLcQY0BFOSBsbvJwCmQQ1vQ6/5m4YmzBC1rRelCq1Agi9//////////8BEAIaDDYzMzM5Mzc1ODY
5MCIM3vgTOnS9B6JyQQmeKokCJkhMaeK5NcXazpFuqObvIOQpIjKOVtHR/NwxdQCrfqPa2qbn+VsG9i7tF0pvxniO/OQmqxXXaNlRjnq2QomydAte/91VX
J1cqT7R7k/06ISBc2AVcSAJfgAYEIB7kKVF2UkY01VJ845VjTPnER7O4enKd5jYyHakuOkj29olSph1sjrq6VFYBo0foLgLJcDsL/QbipTk8HXX7XT8f/G
h8jGKfUjy2CUvJfuAAX3zvsTFjSsGEb69J1pZd0sQfoBGi6Mv0vezW+1jWX+dLdpnzDEJrnk0x7g6po1uXrCjDF6+pB+5QwPhI78D2lF/tcLahLbr5E16r
i2DXv0eQ0woOaL6u0xsKDPvwzDCkqe2BjqeAYi5Fs7WB0Ei5FiAqHdJEzXcQZI18JX5H59W3p+v71sN7sGLxJYrXoMmFLH7amaZxQ7r5xkn9/is6Ge3Zcu
xROIy5GOLuqoHVsNRxCRQ83ZoIewd32TRN8h3uRLQnE7ZMf6ggljBvqvT1e2IlA+YcdeWrkeM/fCXJ0g7kKEcnkNgBMv+W9LXi2P8DMsm0AnP6jhFK5R6C
```

Step 2 : copy the AMI ID from the EC2



Step 3 : Create the main.tf and provider.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
  }
}

resource "aws_instance" "myserver" {
    ami = "ami-066784287e358dad1"
    instance_type = "var.instance_type"
    tags = {
      Name="SampleServer"
    }
}
```

```
provider "aws" {
  access_key="ASIA3IKFWBC4XD2NUOGT"
```

```
  secret_key="HV1nehMF9eHDuMPb3kffqN4S9FgWiuyRt0FtKMN7"

token="IQoJb3JpZ2luX2VjELD/////////wEaCXVzLXdlc3QtMiJIMEYCIQDUStpPY4WH1rL
xLjK+gWmrGVsyUjHDszjELoj/+ODmOwIhAJXxnswL/ZId9I3+CvTGQhOaUaRaA03FF/e2QqXOP
FLJKrECCLj/////////wEQABoMNzczNzc3MTMxNzA1IgzAQdzV3pYlWwUUeVsqhQLrEFCk8N6
Y0xynEV4qLqSbfQ3gS074l976p9R7hyn15nT+PkAR2uytbSQfDD2XceXD0KTSF0F1GqHEtrTyI
RM3y5wbWdHj/3X7WPSgMa2b04vln+9LJehMT3naBzqtUxO3qauygsxlrgnhKF3Necr4jTjy5kU
ioPh3rm53pNh07nXAXH2W1WB9HUeHWS8Fp2x698cN2pTINjzjJ5UaO8ouuSOfeknDZweadm2u2
SPA5UjDk8xjsHc/RWXrrVZ8RTMYI6yBEml2NStiR2txQXNT8g0zf/rgJz6gWSLVfvW3Vk6SCI1
iMupxPYE+JpFdsyt4+AL/MLxFpQ12jg5rQJGbhhUnOYww9JKmtgY6nAHrmQPgQaSyPFsJETANO
kOUV/zjwEVL8KlzEHip2u22rYivpsTyoOkfKZibT9CtycdSMaOQgjFj5kX1ASnxIxocdVMVMSj
2cWn2JwskGcOcJjuvL5ZbKmh/T8cAKgDJTTaFdPKZCO6yVGQ/RB3REeCKmvkAE9yPjjcHclGcf
DJnaVS8neVyky2xwGBNGhBwnoWc3Ol+LQ22V9dmFWs="
  region = "us-east-1"
}
```

Step 4 : Execute terraform init , terraform plan and terraform apply command

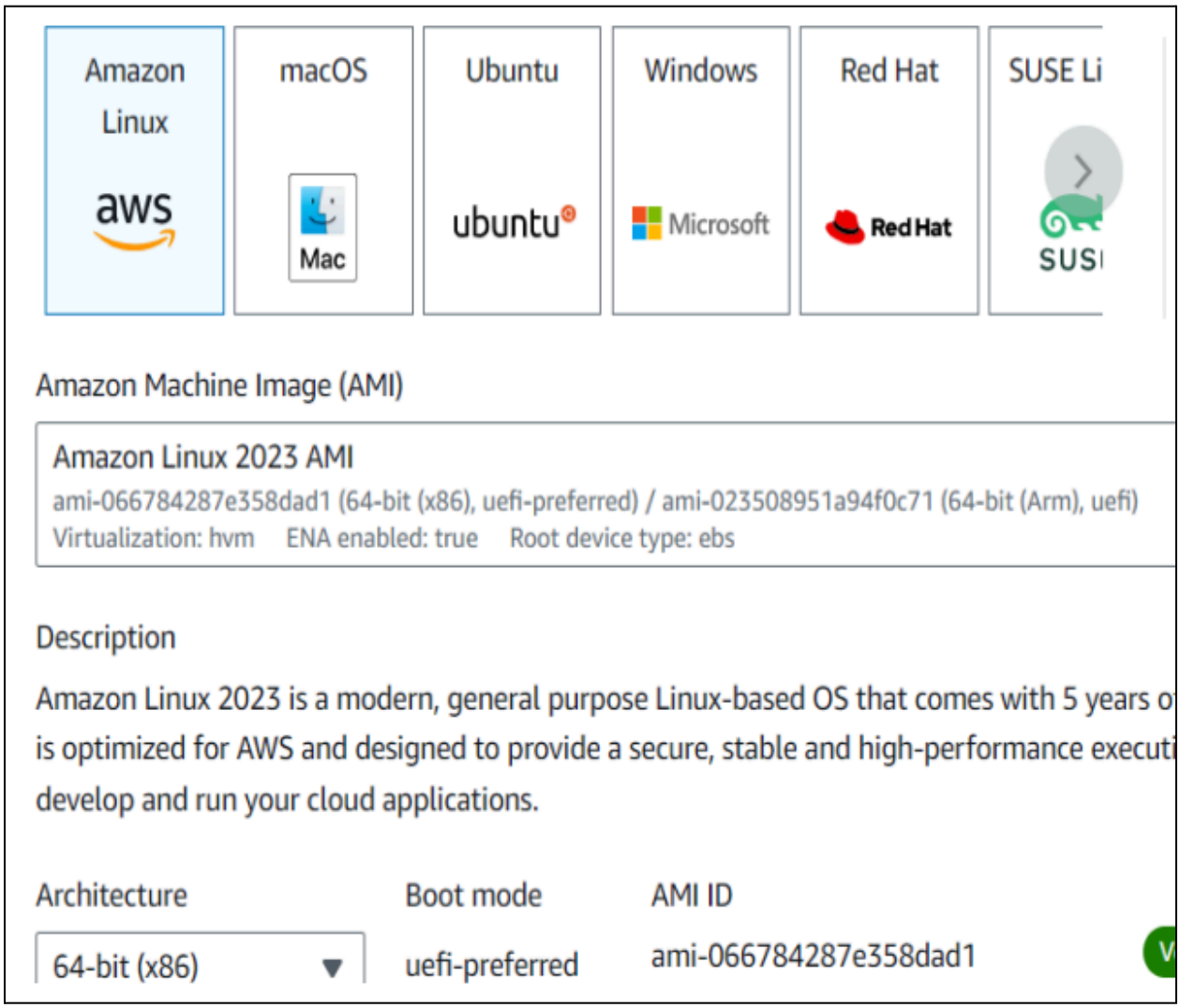
```
C:\Users\272241\New folder\aws-ec2>terraform plan

Terraform used the selected providers to generate the following
execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.myserver will be created
  + resource "aws_instance" "myserver" {
      + ami                                  = "ami-07cc1bbe145f35b58
"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
```

```
      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
```

```
      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
aws_instance.myserver: Still creating... [20s elapsed]
aws_instance.myserver: Still creating... [30s elapsed]
aws_instance.myserver: Still creating... [40s elapsed]
aws_instance.myserver: Creation complete after 47s [id=i-0eaff679364
b7d49]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

```
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 1
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 2
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 3
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 4
0s elapsed]
aws_instance.myserver: Destruction complete after 43s
aws_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
aws_instance.myserver: Still creating... [20s elapsed]
aws_instance.myserver: Creation complete after 26s [id=i-02c812193590
baf79]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
```

Step 5 : Ec2 before and after instance creation .
Step 6 : Copy AWS AMI ID and change it in code



Step 9 : Destroy the instance using terraform destroy

```
Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as sho
ove.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.myserver: Destroying... [id=i-02c812193590baf79]
aws_instance.myserver: Still destroying... [id=i-02c812193590baf
```

```
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
aws_instance.myserver: Still creating... [20s elapsed]
aws_instance.myserver: Still creating... [30s elapsed]
aws_instance.myserver: Creation complete after 37s [id=i-02652447
6f71a]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

ec2_instance_type = "t2.micro"
ec2_public_ip = "44.203.12.232"
```

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as s
ove.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.myserver: Destroying... [id=i-0265244710a46f71a]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Destruction complete after 43s

Destroy complete! Resources: 1 destroyed.
```

(EXTRA)

Hosting Website on s3 using Terraform-
Static website hosting:

Step 1 : create main.tf and write following code
Code -

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
     random = {
```

```
      source = "hashicorp/random"
      version = "3.6.2"
    }
  }
}

resource "random_id" "ran_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "mywebapp-bucket" {
  bucket = "my-mywebapp-bucket-${random_id.ran_id.hex}"
}

resource "aws_s3_object" "index_html" {
  bucket = aws_s3_bucket.mywebapp-bucket.bucket
  source = "./index.html"
  key = "index.html"
  content_type = "text/html"

}

resource "aws_s3_object" "style_css" {
  bucket = aws_s3_bucket.mywebapp-bucket.bucket
  source = "./styles.css"
  key = "styles.css"
  content_type = "text/css"

}

resource "aws_s3_bucket_public_access_block" "example" {
  bucket = aws_s3_bucket.mywebapp-bucket.id
  block_public_acls       = false
  block_public_policy     = false
  ignore_public_acls      = false
  restrict_public_buckets = false
}

resource "aws_s3_bucket_policy" "staticwebnew" {
   bucket = aws_s3_bucket.mywebapp-bucket.id
    policy = jsonencode(
```

```
        {
    Version = "2012-10-17",
    Statement = [
        {
            Sid = "PublicReadGetObject",
            Effect = "Allow",
            Principal = "*",
            Action = "s3:GetObject",
                                                        Resource     =
"arn:aws:s3:::${aws_s3_bucket.mywebapp-bucket.id}/*"
        }
    ]
}
    )
}

resource "aws_s3_bucket_website_configuration" "example" {
  bucket = aws_s3_bucket.mywebapp-bucket.id

  index_document {
    suffix = "index.html"
  }
}

output "website_endpoint" {
  value = aws_s3_bucket_website_configuration.example.website_endpoint

}
```

Step 2 : Create Provider.tf and write following code

Code:

```
provider "aws" {
  access_key="ASIA3IKFWBC4XD2NUOGT"
  secret_key="HV1nehMF9eHDuMPb3kffqN4S9FgWiuyRt0FtKMN7"

token="IQoJb3JpZ2luX2VjELD/////////wEaCXVzLXdlc3QtMiJIMEYCIQDUStpPY4WH1rL
xLjK+gWmrGVsyUjHDszjELoj/+ODmOwIhAJXxnswL/ZId9I3+CvTGQhOaUaRaA03FF/e2QqXOP
FLJKrECCLj/////////wEQABoMNzczNzc3MTMxNzA1IgzAQdzV3pYlWwUUeVsqhQLrEFCk8N6
Y0xynEV4qLqSbfQ3gS074l976p9R7hyn15nT+PkAR2uytbSQfDD2XceXD0KTSF0F1GqHEtrTyI
```

```
RM3y5wbWdHj/3X7WPSgMa2b04vln+9LJehMT3naBzqtUxO3qauygsxlrgnhKF3Necr4jTjy5kU
ioPh3rm53pNh07nXAXH2W1WB9HUeHWS8Fp2x698cN2pTINjzjJ5UaO8ouuSOfeknDZweadm2u2
SPA5UjDk8xjsHc/RWXrrVZ8RTMYI6yBEml2NStiR2txQXNT8g0zf/rgJz6gWSLVfvW3Vk6SCI1
iMupxPYE+JpFdsyt4+AL/MLxFpQ12jg5rQJGbhhUnOYww9JKmtgY6nAHrmQPgQaSyPFsJETANO
kOUV/zjwEVL8KlzEHip2u22rYivpsTyoOkfKZibT9CtycdSMaOQgjFj5kX1ASnxIxocdVMVMSj
2cWn2JwskGcOcJjuvL5ZbKmh/T8cAKgDJTTaFdPKZCO6yVGQ/RB3REeCKmvkAE9yPjjcHclGcf
DJnaVS8neVyky2xwGBNGhBwnoWc3Ol+LQ22V9dmFWs="
  region = "us-east-1"
}
```

Step 3: Execute Terraform init command.

```
C:\Users\272241\New folder\tf-backend>cd//
The system cannot find the path specified.

C:\Users\272241\New folder\tf-backend>cd..

C:\Users\272241\New folder>cd static-website-hosting

C:\Users\272241\New folder\static-website-hosting>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.64.0"...
- Finding hashicorp/random versions matching "3.6.2"...
- Installing hashicorp/aws v5.64.0...
```

```
- Finding hashicorp/aws versions matching "5.64.0"...
- Finding hashicorp/random versions matching "3.6.2"...
- Installing hashicorp/aws v5.64.0...
- Installed hashicorp/aws v5.64.0 (signed by HashiCorp)
- Installing hashicorp/random v3.6.2...
- Installed hashicorp/random v3.6.2 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the
rovider
selections it made above. Include this file in your version control
epository
so that Terraform can guarantee to make the same selections by defa
t when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform pl
" to see
any changes that are required for your infrastructure. All Terrafor
commands
should now work.

If you ever set or change modules or backend configuration for Terr
orm,
rerun this command to reinitialize your working directory. If you f
get, other
commands will detect it and remind you to do so if necessary.
```

Step 4: Terraform plan and terraform apply:

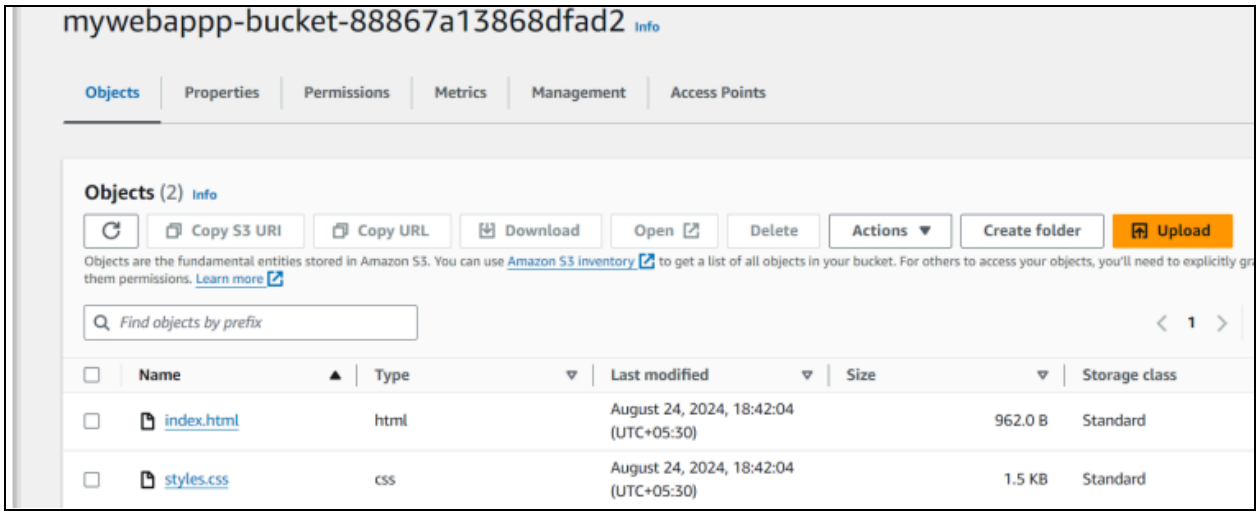
```
      + checksum_sha1           = (known after apply)
      + checksum_sha256         = (known after apply)
      + content_type            = (known after apply)
      + etag                    = (known after apply)
      + force_destroy           = false
      + id                      = (known after apply)
      + key                     = "styles.css"
      + kms_key_id              = (known after apply)
      + server_side_encryption  = (known after apply)
      + source                  = "./styles.css"
      + storage_class           = (known after apply)
      + tags_all                = (known after apply)
      + version_id              = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_object.style_css: Creating...
aws_s3_object.style_css: Creation complete after 3s [id=styles.css]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

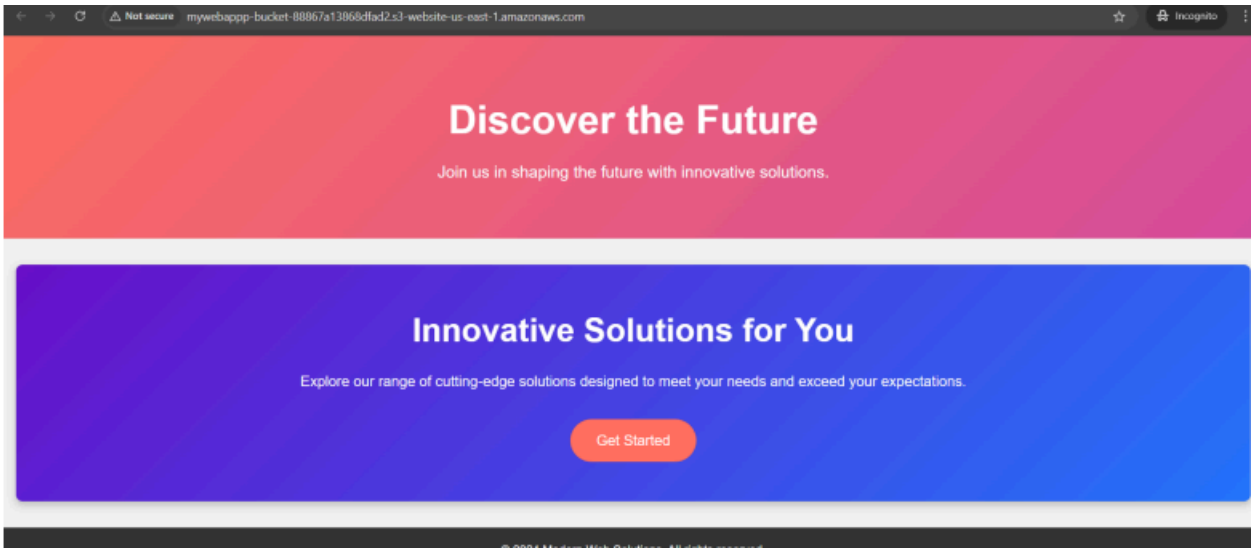
```
Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + website_endpoint = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket_policy.staticwebnew: Creating...
aws_s3_bucket_website_configuration.example: Creating...
aws_s3_bucket_policy.staticwebnew: Creation complete after 3s [id=my-
mywebapp-bucket-6beb0443d9758340]
aws_s3_bucket_website_configuration.example: Creation complete after
3s [id=my-mywebapp-bucket-6beb0443d9758340]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

website_endpoint = "my-mywebapp-bucket-6beb0443d9758340.s3-website-us
-east-1.amazonaws.com"
```

```
        # (23 unchanged attributes hidden)
    }

Plan: 0 to add, 2 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_object.index_html: Modifying... [id=index.html]
aws_s3_object.style_css: Modifying... [id=styles.css]
aws_s3_object.style_css: Modifications complete after 2s [id=styles.c
ss]
aws_s3_object.index_html: Modifications complete after 2s [id=index.h
tml]

Apply complete! Resources: 0 added, 2 changed, 0 destroyed.

Outputs:

website_endpoint = "my-mywebapp-bucket-6beb0443d9758340.s3-website-us
-east-1.amazonaws.com"

C:\Users\272241\New folder\static-website-hosting>
```

```
                    },
                ]
              + Version   = "2012-10-17"
            }
        )
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket_policy.staticwebnew: Creating...
aws_s3_bucket_policy.staticwebnew: Creation complete after 2s [id=my-
mywebapp-bucket-c00793cfc7eca1f6]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

website_endpoint = "my-mywebapp-bucket-c00793cfc7eca1f6.s3-website-us
-east-1.amazonaws.com"
```

Step 4 : check bucket for if files are uploaded and if the site is hosted correctly at the website_endpoint given in cmd Outputs





Step 5 : terraform destroy to destroy the bucket

```
  Enter a value: yes

aws_s3_bucket_public_access_block.example: Destroying... [id=my
app-bucket-6beb0443d9758340]
aws_s3_bucket_policy.staticwebnew: Destroying... [id=my-mywebap
et-6beb0443d9758340]
aws_s3_object.style_css: Destroying... [id=styles.css]
aws_s3_bucket_website_configuration.example: Destroying... [id=
ebapp-bucket-6beb0443d9758340]
aws_s3_object.index_html: Destroying... [id=index.html]
aws_s3_object.index_html: Destruction complete after 2s
aws_s3_object.style_css: Destruction complete after 2s
aws_s3_bucket_website_configuration.example: Destruction comple
er 2s
aws_s3_bucket_policy.staticwebnew: Destruction complete after 2
aws_s3_bucket_public_access_block.example: Destruction complete
 2s
aws_s3_bucket.mywebapp-bucket: Destroying... [id=my-mywebapp-bu
beb0443d9758340]
aws_s3_bucket.mywebapp-bucket: Destruction complete after 1s
random_id.ran_id: Destroying... [id=a-sEQ9l1g0A]
random_id.ran_id: Destruction complete after 0s

Destroy complete! Resources: 7 destroyed.
```

Experiment 7

Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Theory:

Static Application Security Testing (SAST)

SAST is a method of security testing that analyzes source code to identify vulnerabilities without executing the program. It is also known as white-box testing. Here's a breakdown of the SAST process:
1. Code Parsing: The source code is parsed to create an abstract syntax tree (AST), which represents the code structure.
2. Pattern Matching: The AST is analyzed using predefined rules to detect patterns that may indicate security vulnerabilities.
3. Data Flow Analysis: This step examines how data moves through the code to identify potential security issues like SQL injection or cross-site scripting (XSS).
4. Control Flow Analysis: This involves analyzing the paths that the code execution might take to find logical errors or vulnerabilities.
5. Reporting: The tool generates a report highlighting the vulnerabilities found, their severity, and recommendations for fixing them.

Benefits of SAST

● Early Detection: Identifies vulnerabilities early in the development lifecycle, reducing the cost and effort required to fix them.
● Comprehensive Coverage: Can analyze 100% of the codebase, including all possible execution paths.
● Automated and Scalable: Suitable for large codebases and can be integrated into CI/CD pipelines for continuous monitoring.

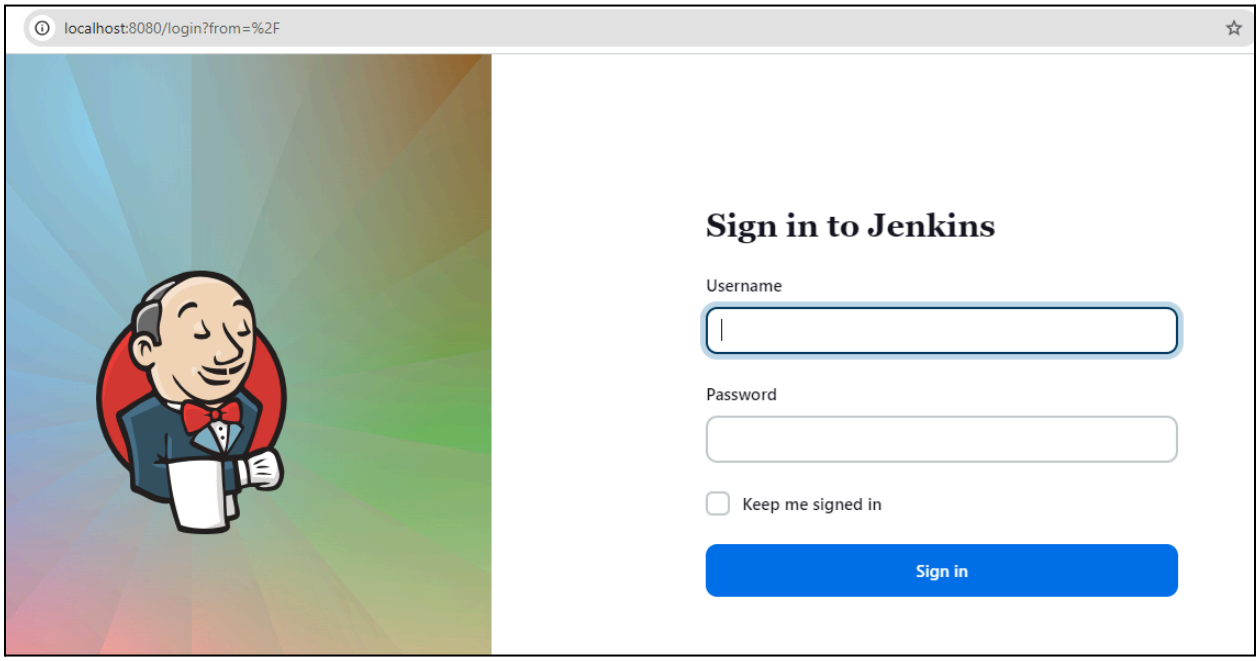
SonarQube and SAST

SonarQube is a popular tool that provides static code analysis to detect bugs, code smells, and security vulnerabilities. Here's how SonarQube fits into the SAST process:
1. Integration: SonarQube can be integrated into your CI/CD pipeline to automatically analyze code every time it is committed.
2. Rule Sets: It uses a comprehensive set of rules to detect security vulnerabilities, coding standards violations, and code quality issues.
3. Dashboards and Reports: SonarQube provides detailed dashboards and reports that help developers understand and fix issues.
4. Continuous Improvement: By continuously analyzing code, SonarQube helps maintain high code quality and security standards over time.

Implementation:

Steps:

1. Open Jenkins Dashboard
● Access your Jenkins Dashboard by navigating to http://localhost:8080 (or the port you have configured Jenkins to run on).



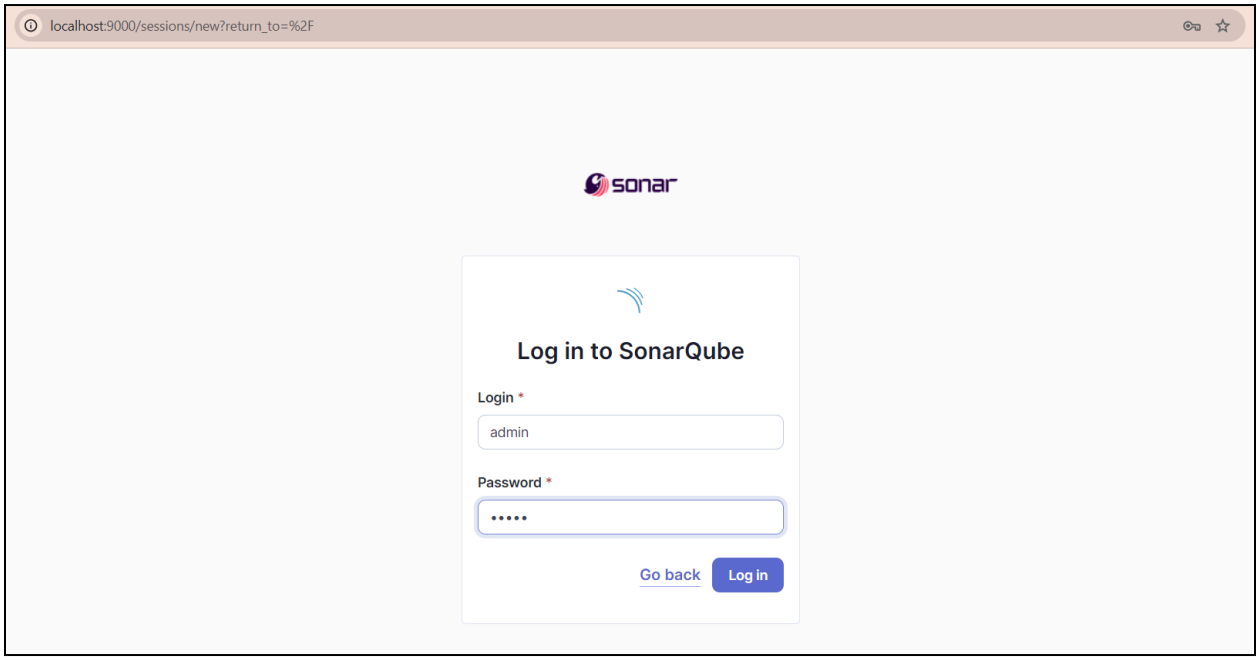
2. Run SonarQube in a Docker Container
    ● Open a terminal and run the following command to start SonarQube in a Docker container

Command -
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

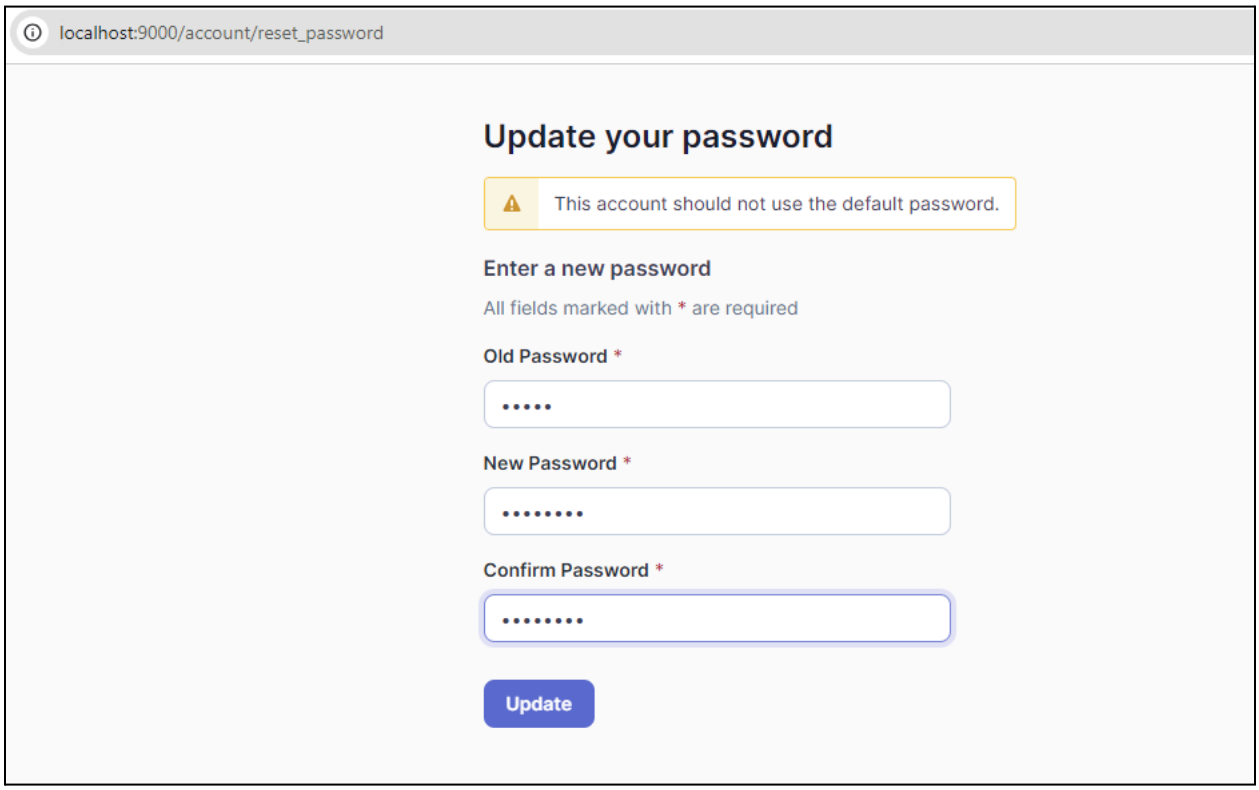
```
C:\Users\272241>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_
DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31ae
cde
Status: Downloaded newer image for sonarqube:latest
8b62aeca4d09887a0db32d349116529581a639b59222a8b20987b42d8cec6ef3
```

3. Check SonarQube Status
    - Once the container is up and running, check the status of SonarQube by navigating to http://localhost:9000

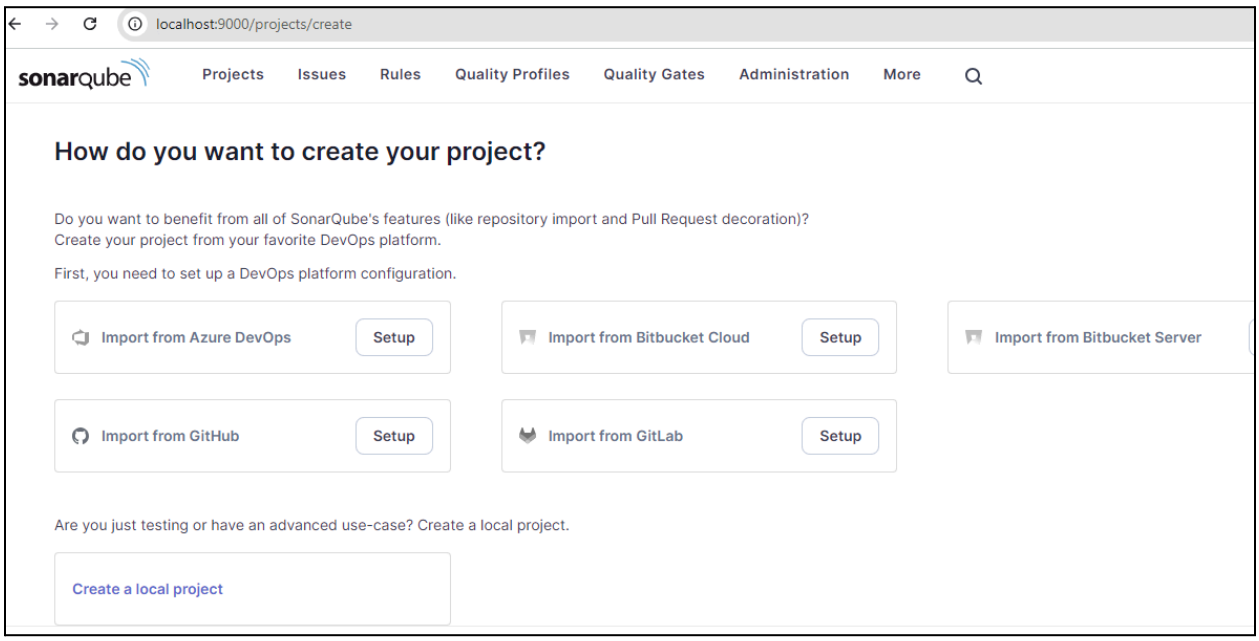
4. Login to SonarQube
    - Use the default credentials to log in:
        - [ ] Login: admin
        - [ ] Password: admin



5. Update the password:

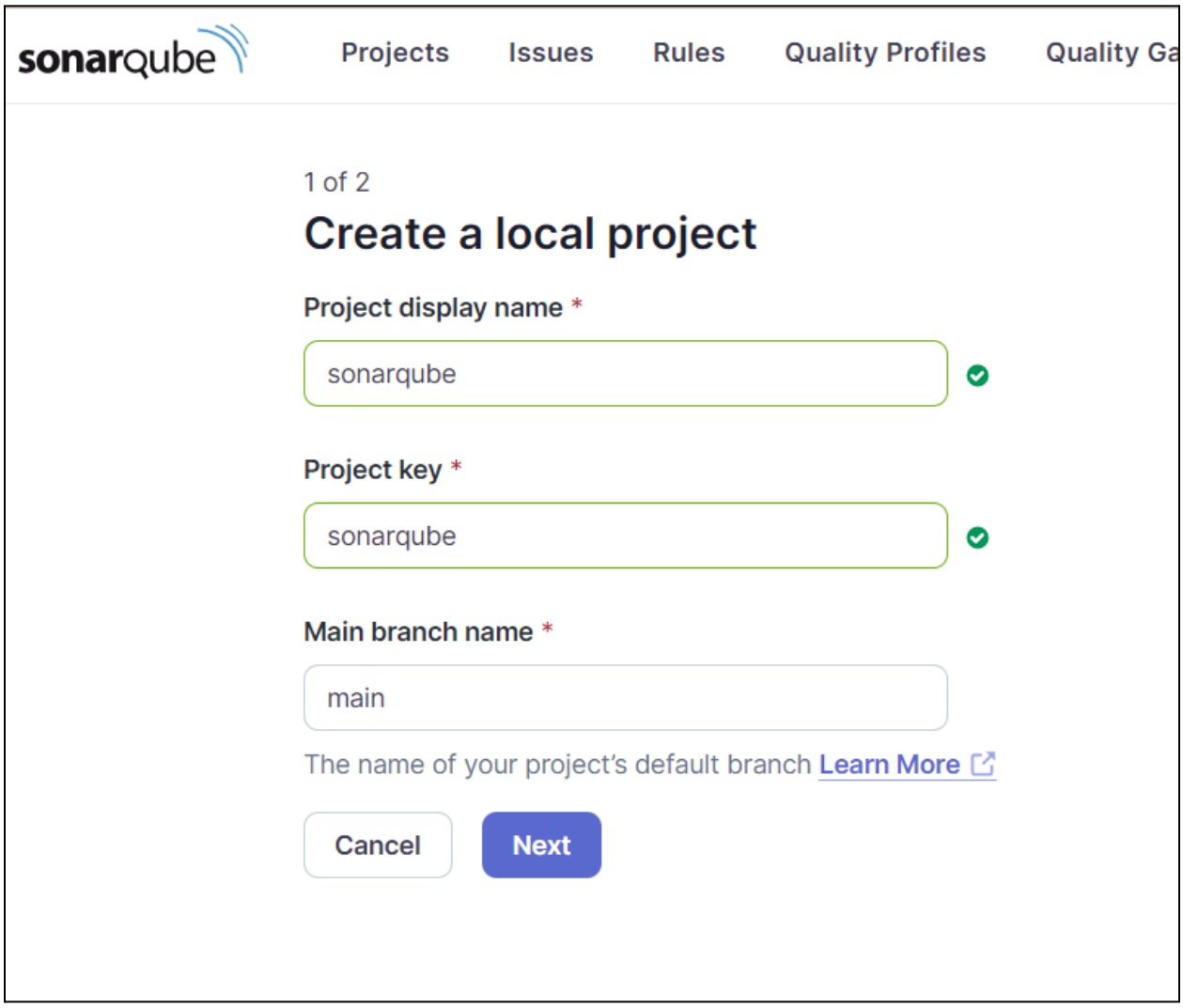


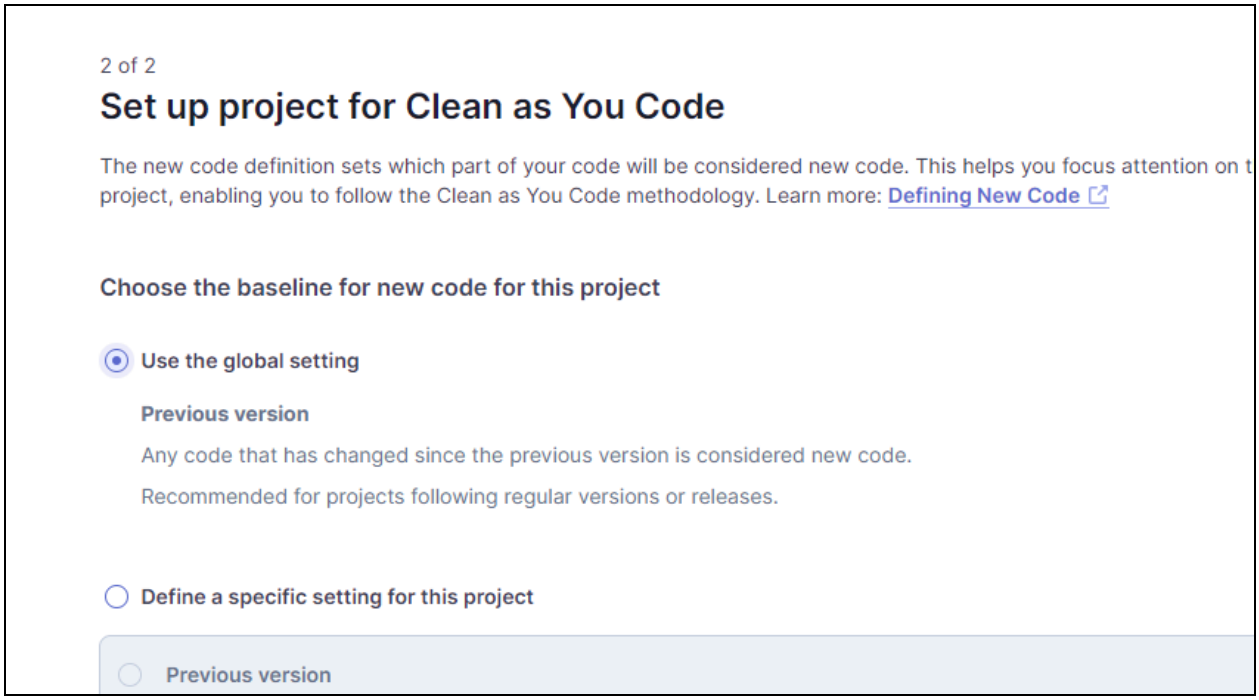
6. After Logging in you will see this:



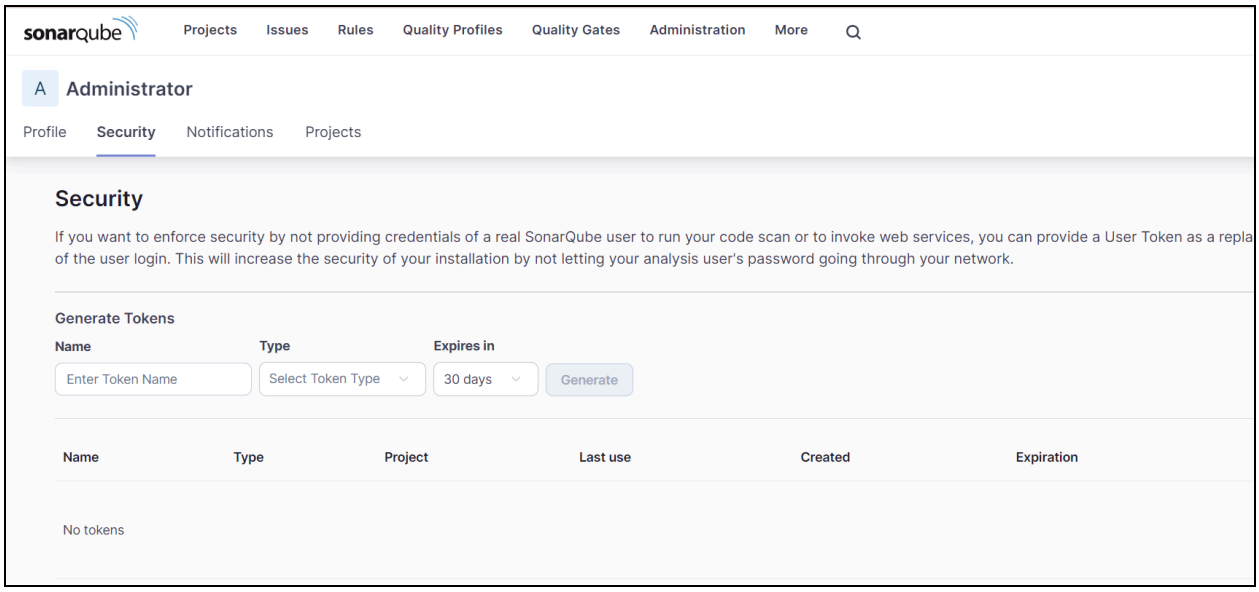
7. After that click on create a local project.

8. Then enter the name of the project:

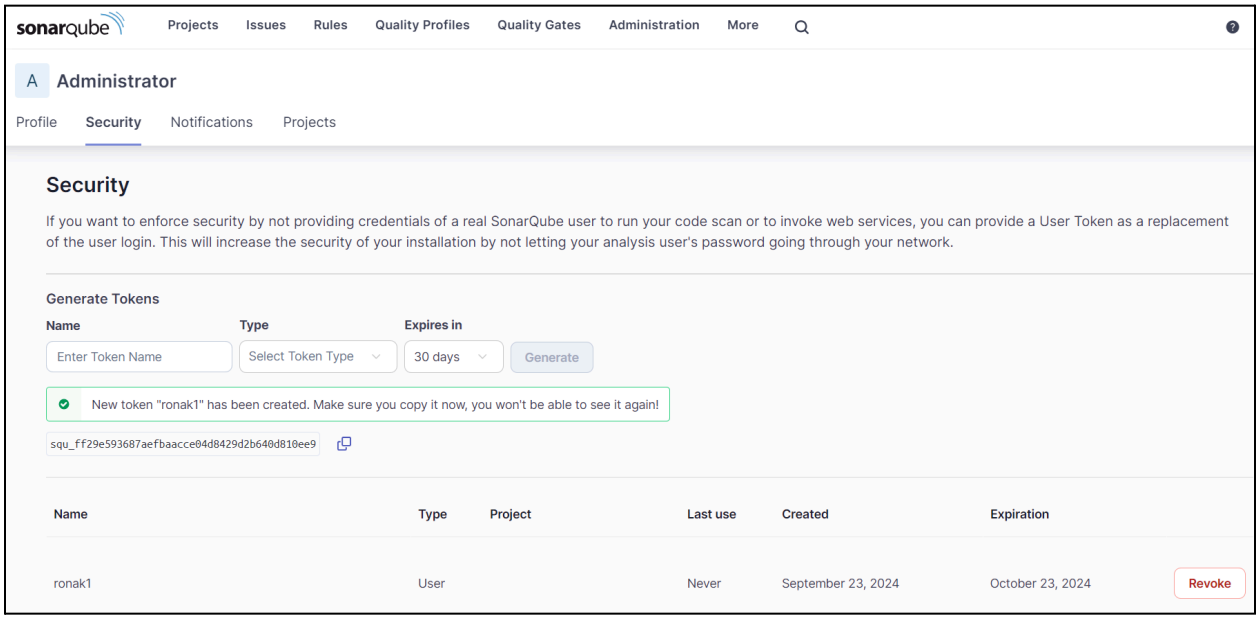




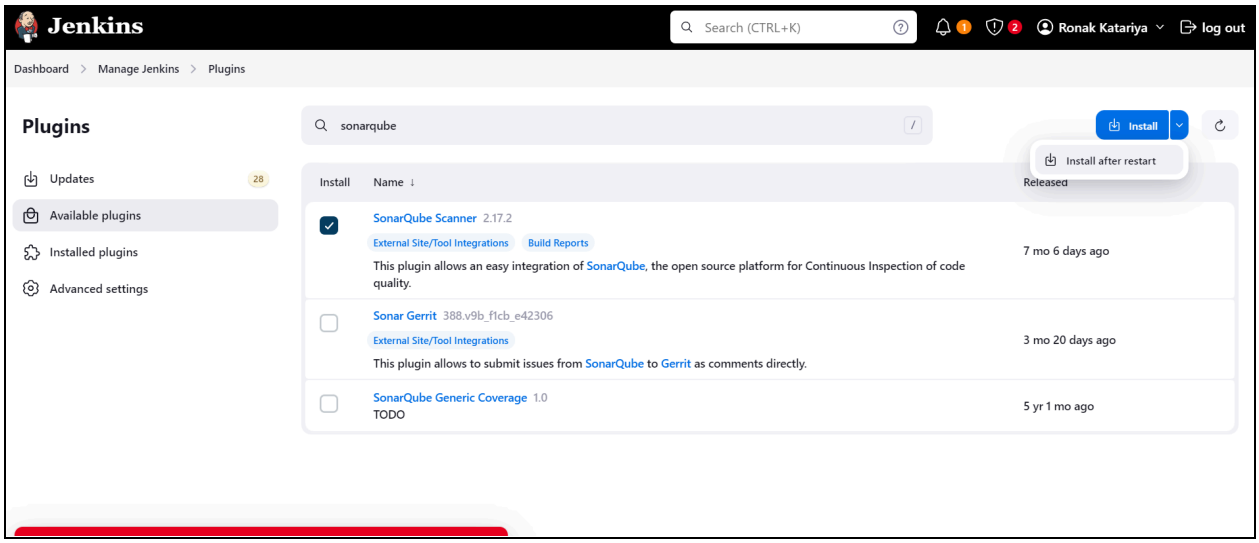
9. Then click on the ‘My Account’ tab:
    - Under that select security.
    - Then name a token you want to generate.
    - Make sure you save it in notepad or somewhere else.



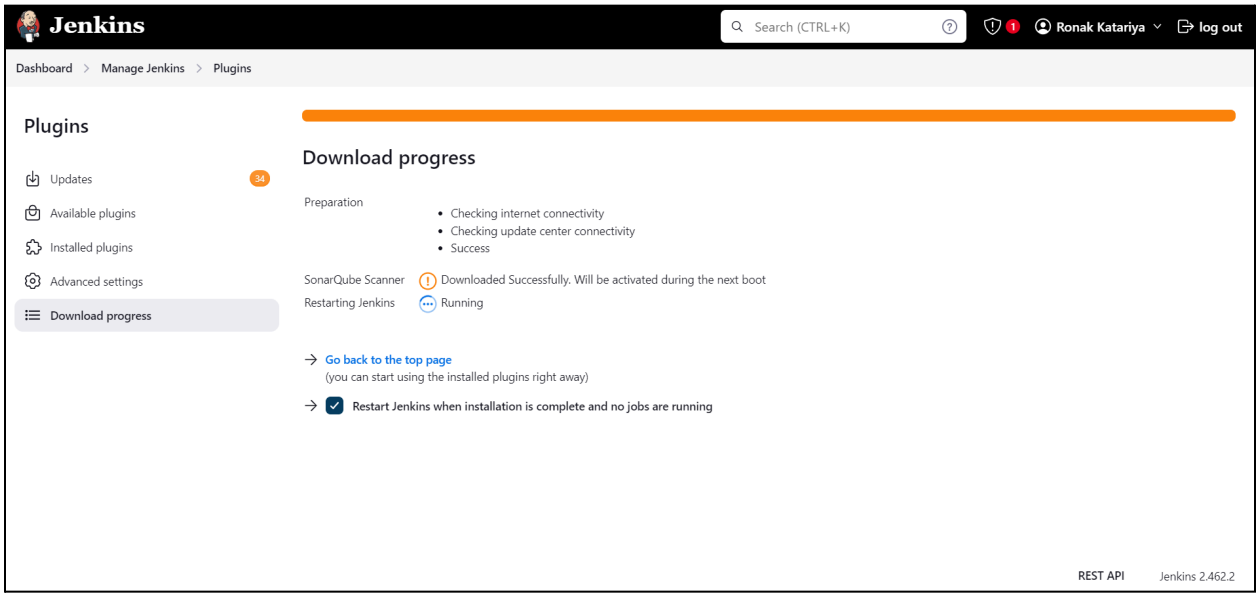
10. After adding the token you can see the list of tokens and the token you just generated:



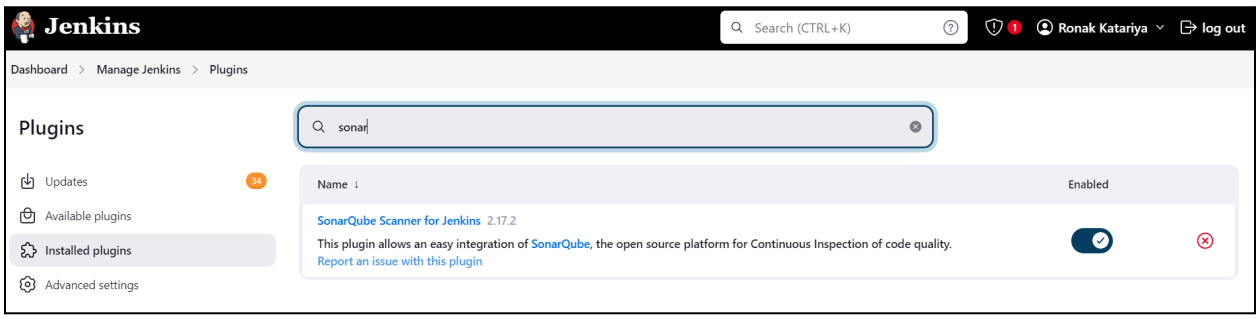
11. Install SonarQube Scanner for Jenkins
    - Go back to the Jenkins Dashboard.
    - Navigate to Manage Jenkins > Manage Plugins.
    - Search for SonarQube Scanner for Jenkins and install it.



12. You will see such window :



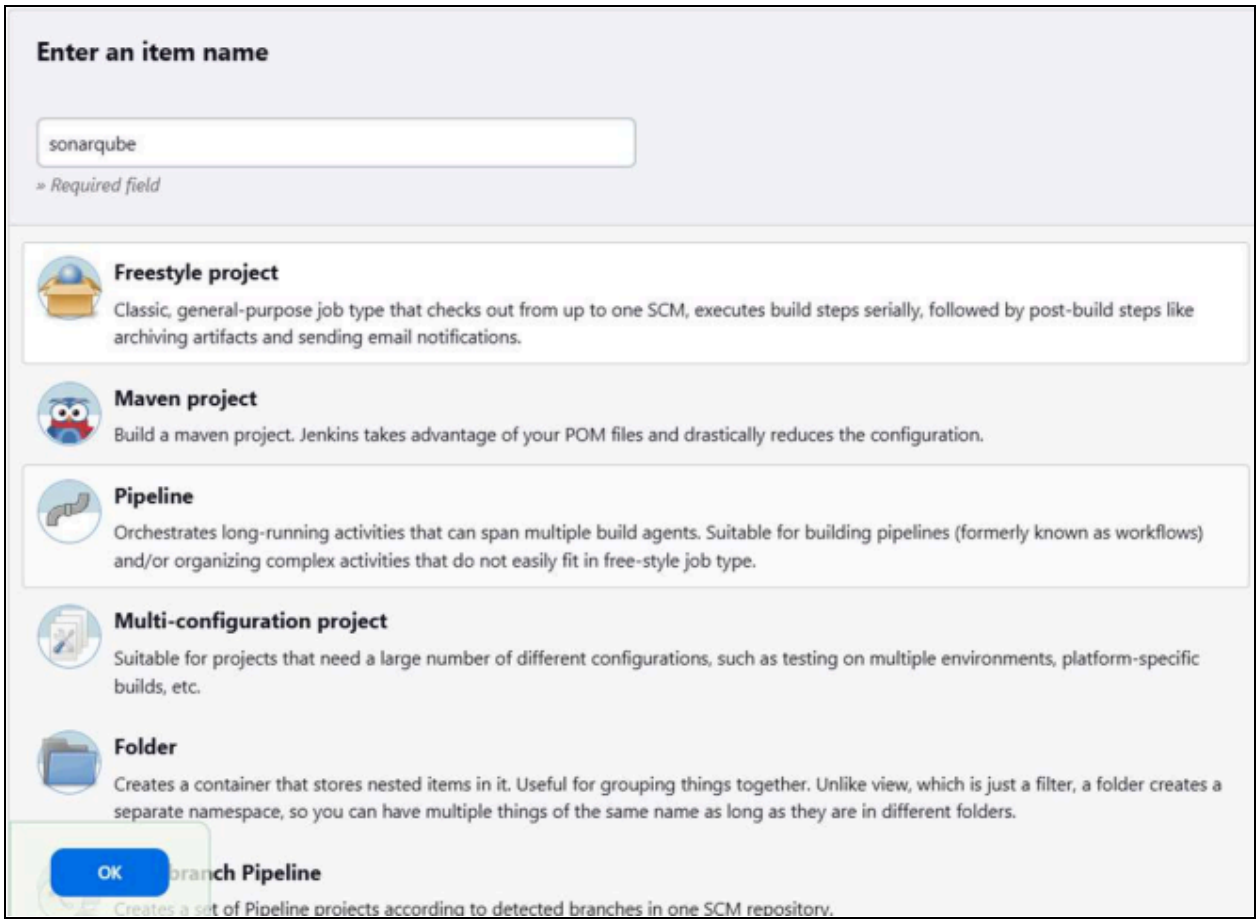
13. You can see the plugins has been downloaded:



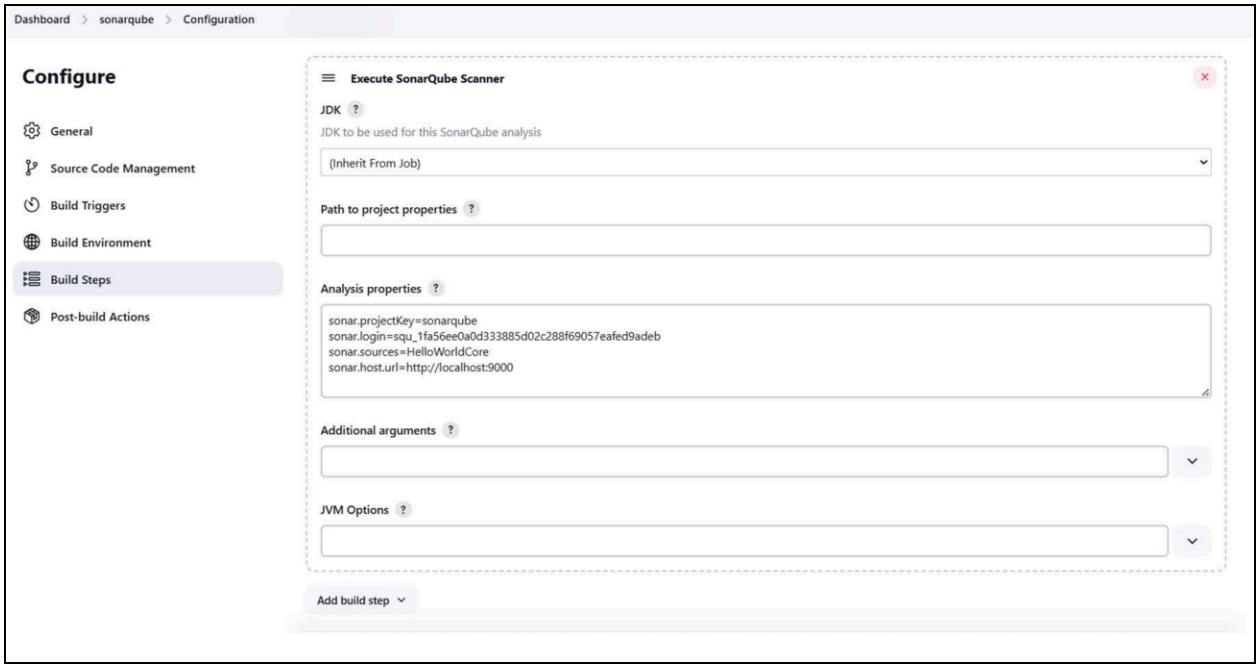
14. Configure SonarQube in Jenkins
    - Go to Manage Jenkins > Configure System.
    - Scroll down to the SonarQube Servers section and enter the required details:
        - Name: Any name you prefer.
        - Server URL: http://localhost:9000
        - Server Authentication Token: (Generate this token in SonarQube under My Account > Security > Generate Tokens).
        - Add Jenkins: Select Kind - Secret Text > Secret (Paste Generated Token)

15. Configure SonarQube Scanner in Jenkins
    - Go to Manage Jenkins > Global Tool Configuration.
    - Scroll down to SonarQube Scanner.
    - Choose the latest version and select Install automatically

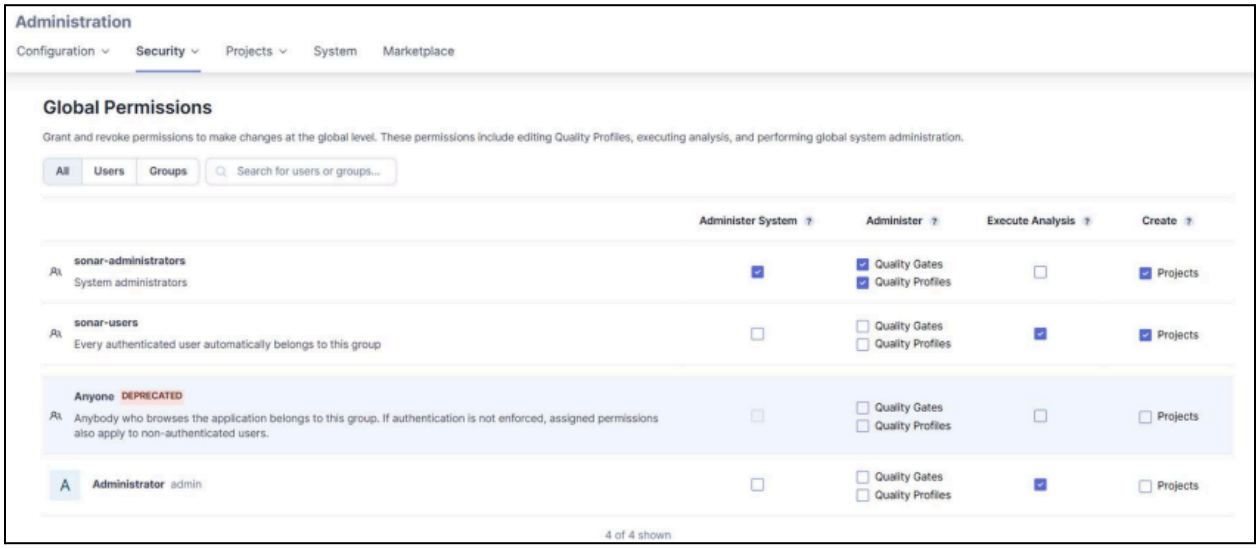
16. Create a New Jenkins Job
    - In Jenkins, create a new item and select Freestyle project.
    - Under Source Code Management, choose Git and enter the repository URL:
    - https://github.com/shazforiot/MSBuild_firstproject.git



17. Configure Build Steps
    - Under the Build section, add a build step to Execute SonarQube Scanner.
    - Enter the following analysis properties:
        - sonar.projectKey=my_project_name
        - sonar.login=your_generated_token
        - sonar.sources=HelloWorldCore
        - sonar.host.url=http://localhost:9000

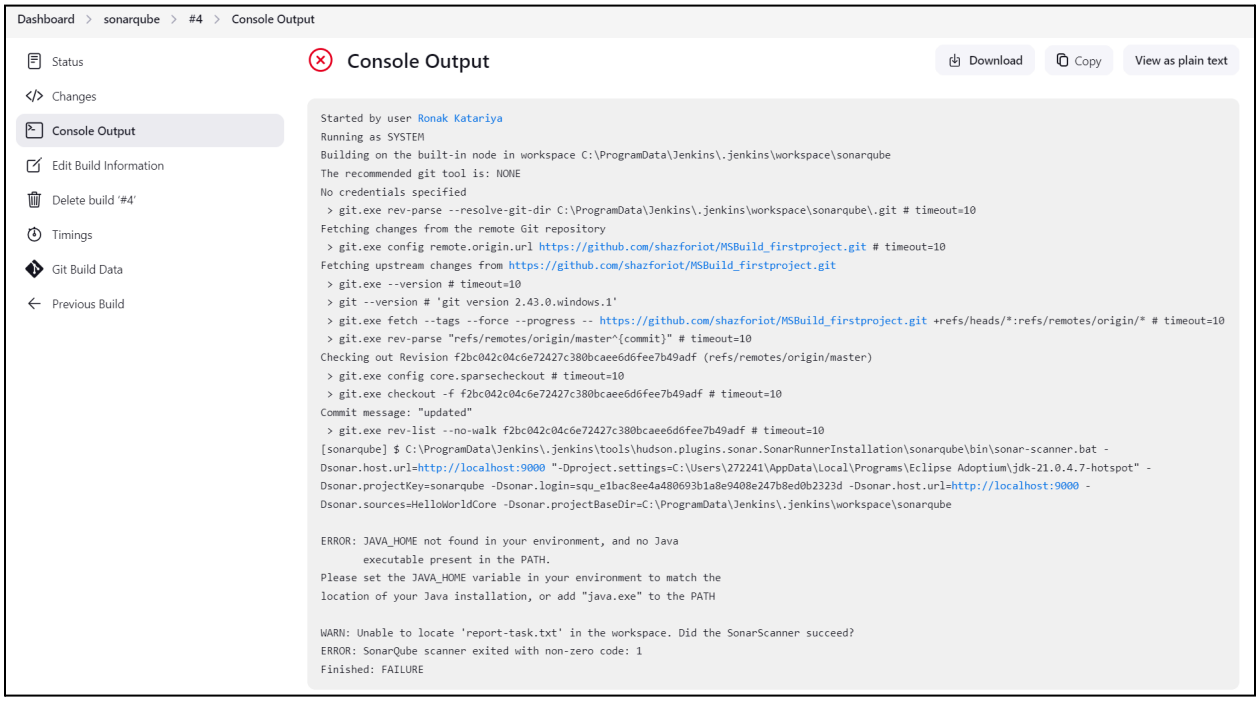


18. Set Permissions in SonarQube
    - Navigate to http://localhost:9000//permissions.
    - Allow Execute Permissions to the Admin user.

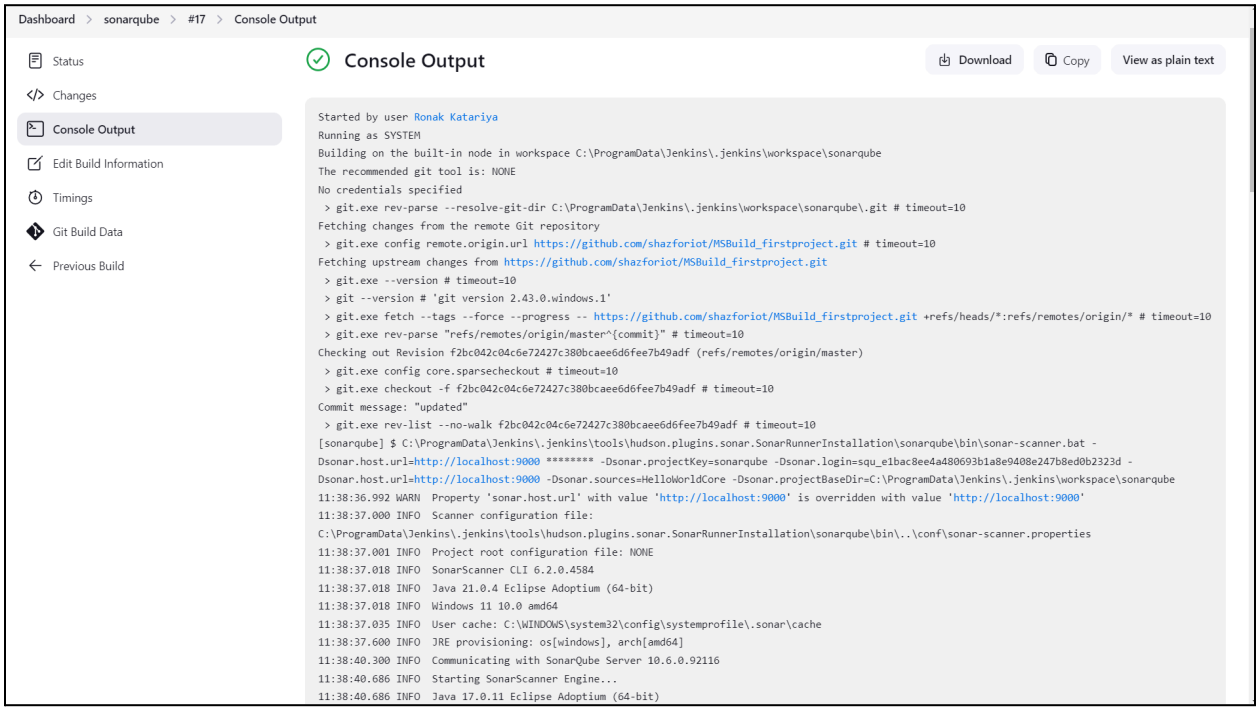


19. Run the Build
    - Go back to Jenkins and run the build.
    - Check the console output for any errors or issues.

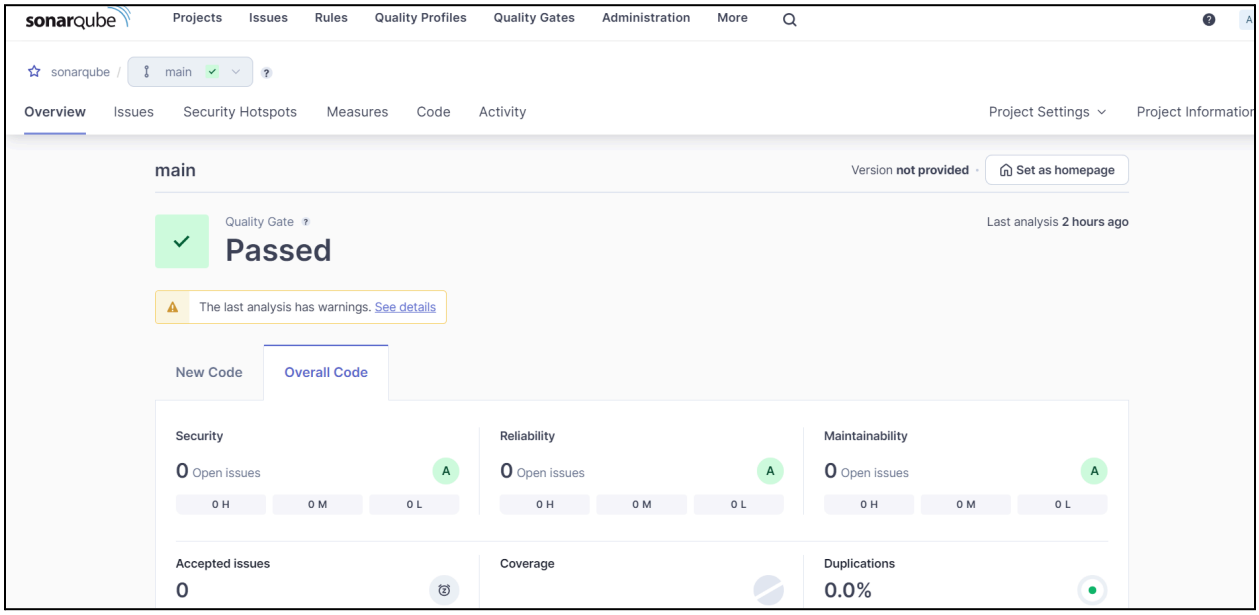
Failed output:



Successful Output:



20. Verify in SonarQube
    - Once the build is complete, check the project in SonarQube to see the analysis results.



Conclusion:

In this experiment, we have understood the importance of SAST and have successfully integrated Jenkins with SonarQube for Static Analysis and Code Testing.

Experiment 8

Aim:

Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Theory:

Static Application Security Testing (SAST)

SAST is a testing methodology that analyzes source code to find security vulnerabilities, making applications less susceptible to attacks. It scans the application before the code is compiled, also known as white-box testing.

Problems SAST Solves:

- Early Detection: Identifies vulnerabilities early in the SDLC, allowing developers to resolve issues without breaking builds or passing vulnerabilities to the final release.
- Real-Time Feedback: Provides developers with immediate feedback as they code, helping them fix issues before moving to the next phase.
- Graphical Representations: Offers visual aids to navigate code, pinpointing exact locations of vulnerabilities and providing guidance on fixes.
- Regular Scanning: Should be run regularly, such as during daily/monthly builds, code check-ins, or code releases.

Importance of SAST

- Resource Efficiency: Developers outnumber security staff, making it challenging to perform manual code reviews. SAST tools can analyze 100% of the codebase quickly.
- Speed: Can scan millions of lines of code in minutes, identifying critical vulnerabilities like buffer overflows, SQL injection, and cross-site scripting with high confidence.

CI/CD Pipeline

A CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline, which is the backbone of the DevOps approach. It involves a series of tasks connected in sequence to facilitate quick software releases. The pipeline is responsible for building code, running tests, and deploying new software versions.

SonarQube

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. It performs static code analysis, providing detailed reports on bugs, code smells, vulnerabilities, and code duplications. It supports over 25 major programming languages and can be extended with various plugins.

Benefits of SonarQube:

- Sustainability: Reduces complexity, vulnerabilities, and code duplications, optimizing application lifespan.
- Increased Productivity: Lowers maintenance costs and risks, reducing the need for extensive code changes.
- Quality Code: Ensures code quality control is an integral part of software development.
- Error Detection: Automatically detects errors and alerts developers to fix them before output submission.
- Consistency: Identifies code criteria breaches, enhancing overall code quality.
- Business Scaling: Supports scaling without restrictions.

Implementation:

Prerequisites
1. Jenkins installed on your machine.
2. Docker installed to run SonarQube.
3. SonarQube installed via Docker

1. Set Up Jenkins
    - Open Jenkins Dashboard on localhost:8080 or your configured port.
    - Install the necessary plugins:
        - SonarQube Scanner Plugin

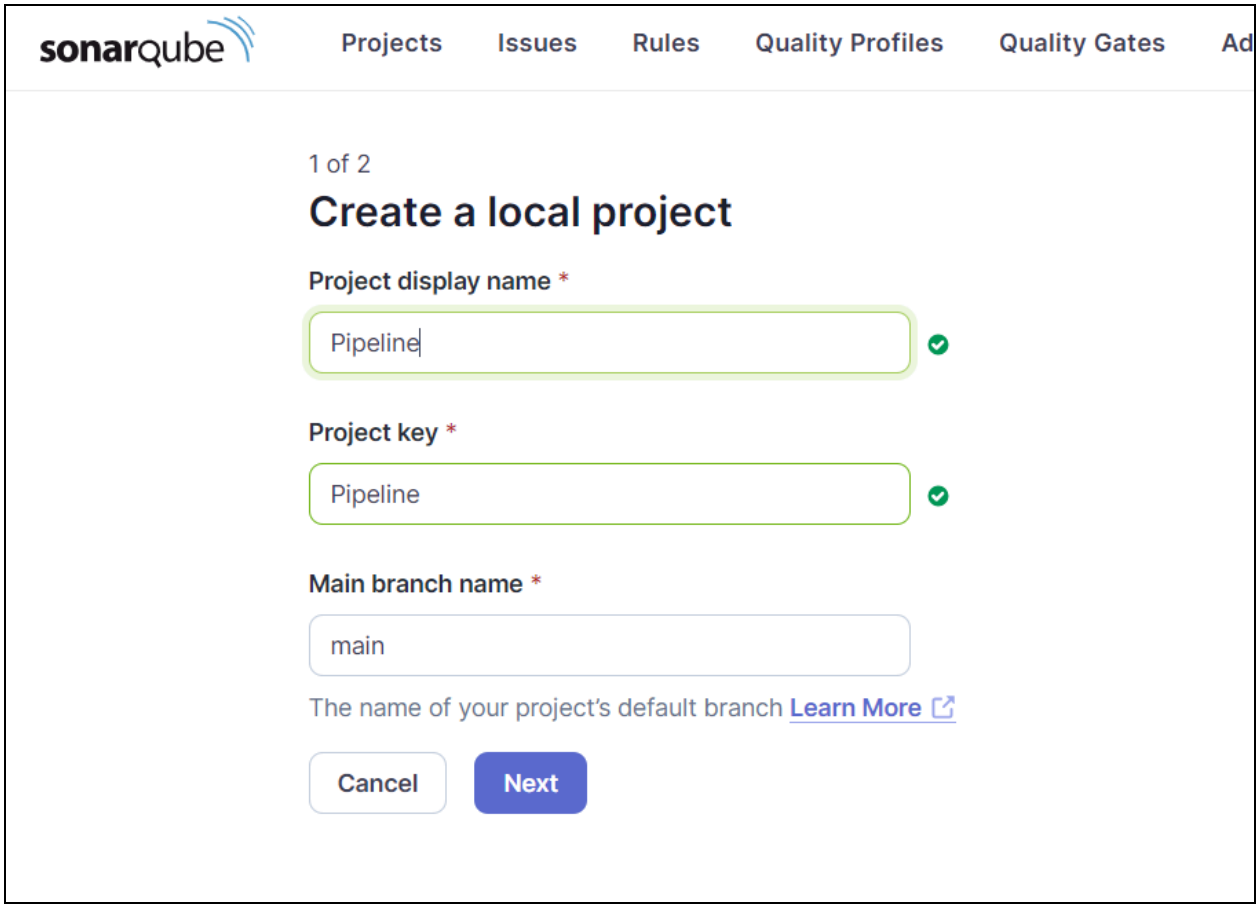
2. Run SonarQube in a Docker Container
    - Open a terminal and run the following command to start SonarQube in a Docker container

Command -
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

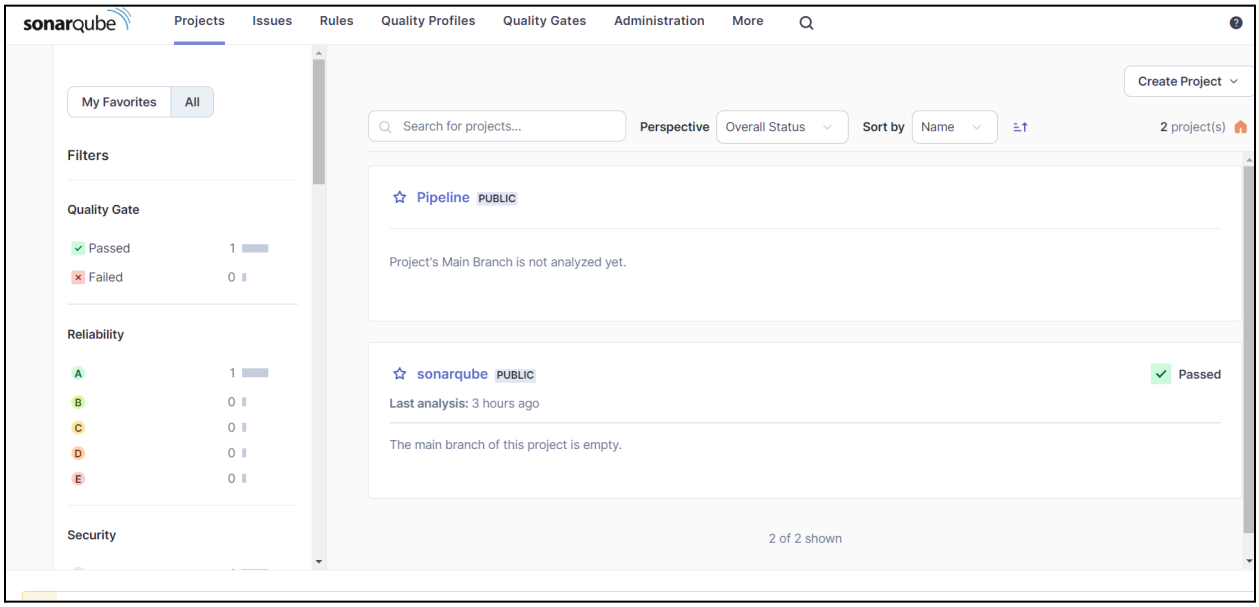
```
C:\Users\272241>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_
DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31ae
cde
Status: Downloaded newer image for sonarqube:latest
8b62aeca4d09887a0db32d349116529581a639b59222a8b20987b42d8cec6ef3
```

3. Check SonarQube Status
    - Once the container is up and running, check the status of SonarQube by navigating to http://localhost:9000

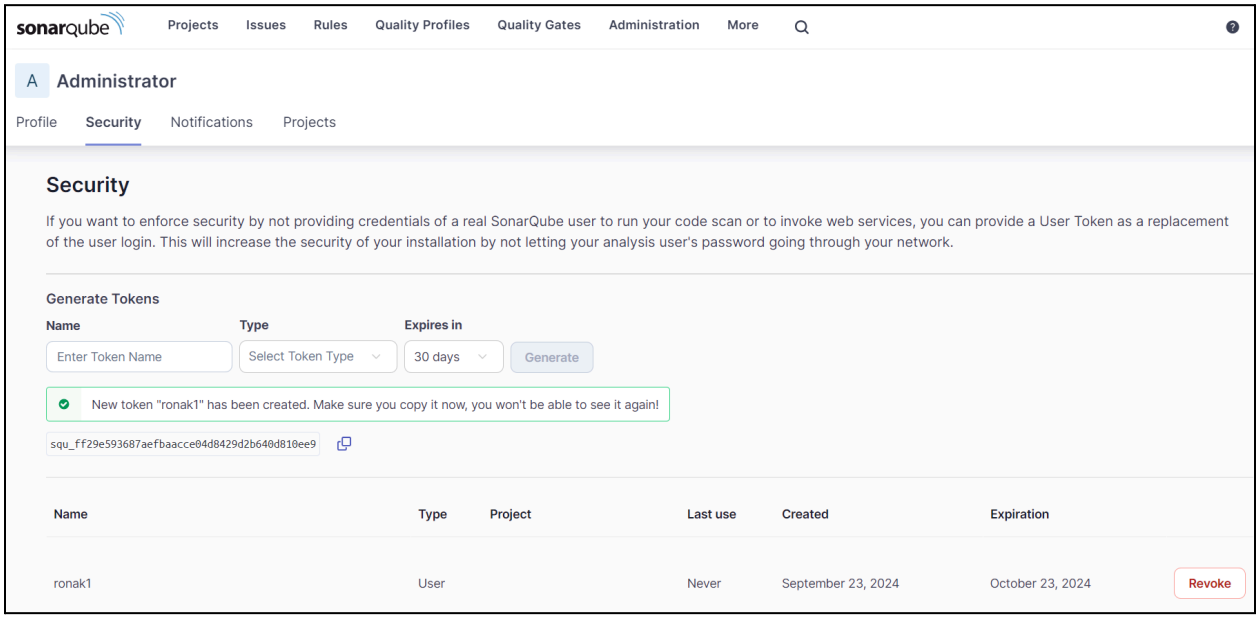
4. Login to SonarQube
    - Use the credentials to log in:



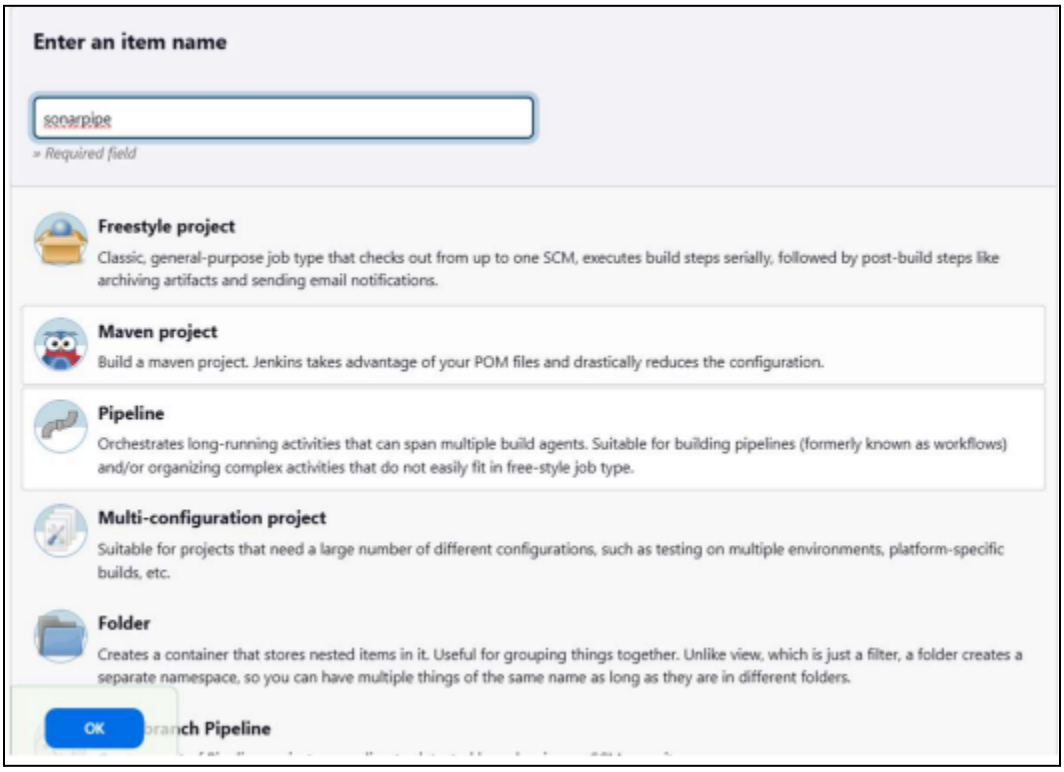
5. You can view the project:



6. Generate SonarQube Token
    - Go to My Account > Security > Generate Tokens.
    - Copy the generated token for later use.

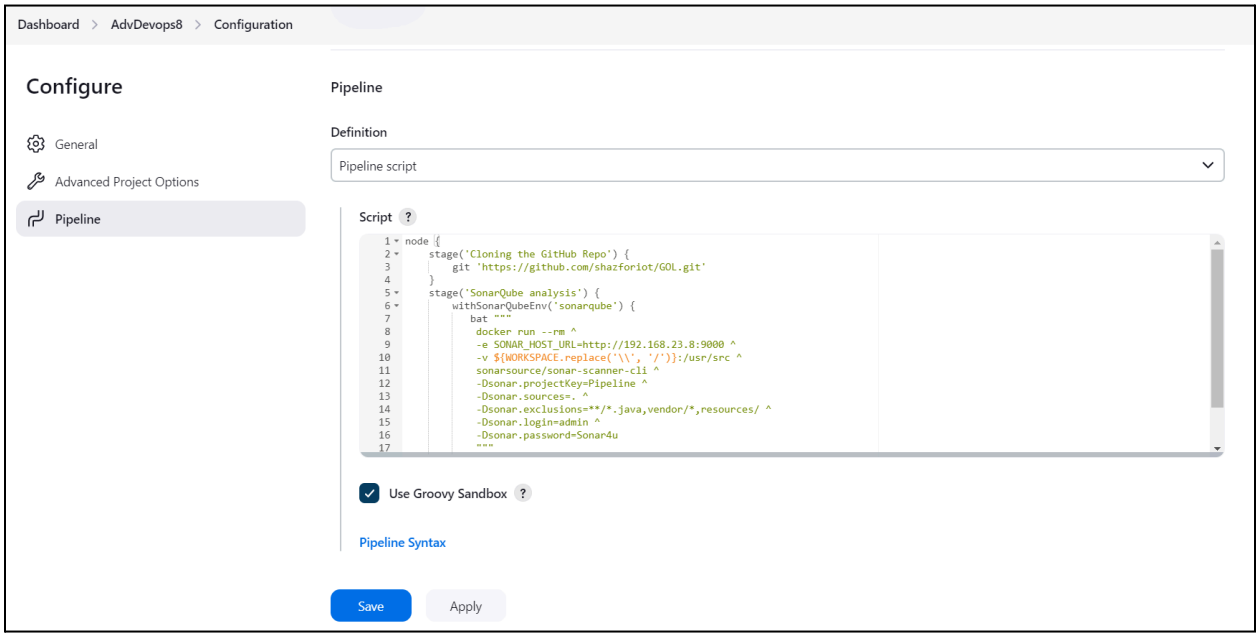


7. Create a Jenkins Pipeline
    - Go to Jenkins Dashboard, click New Item, and select Pipeline.



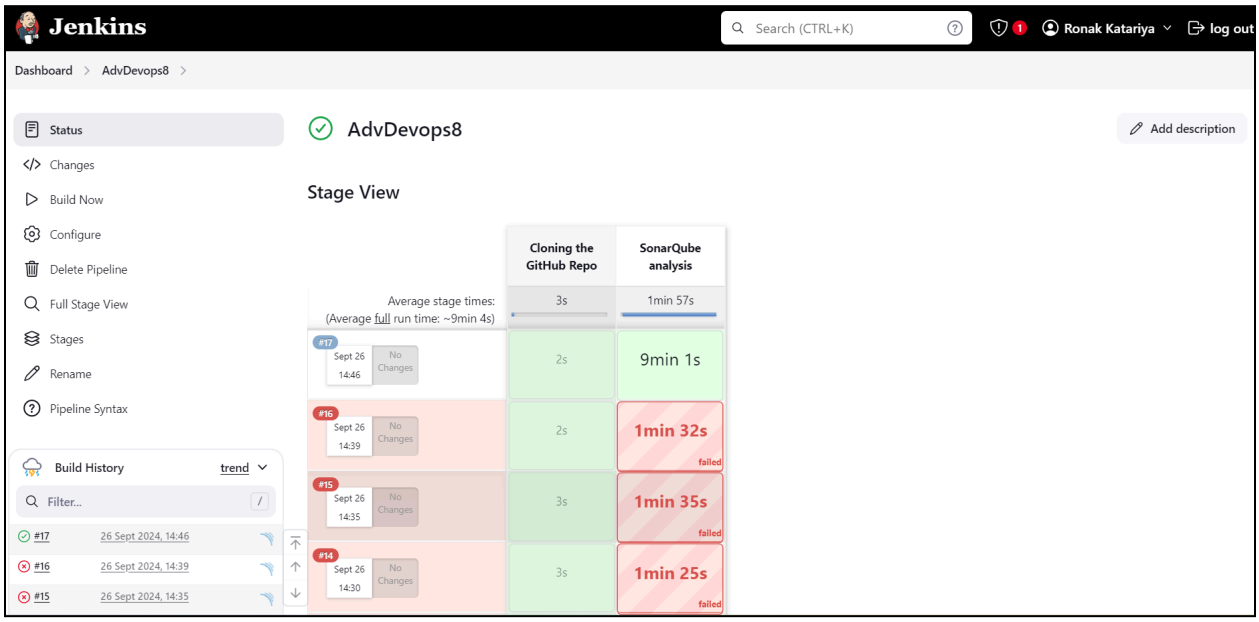
8. Under Pipeline Script, enter the following script:

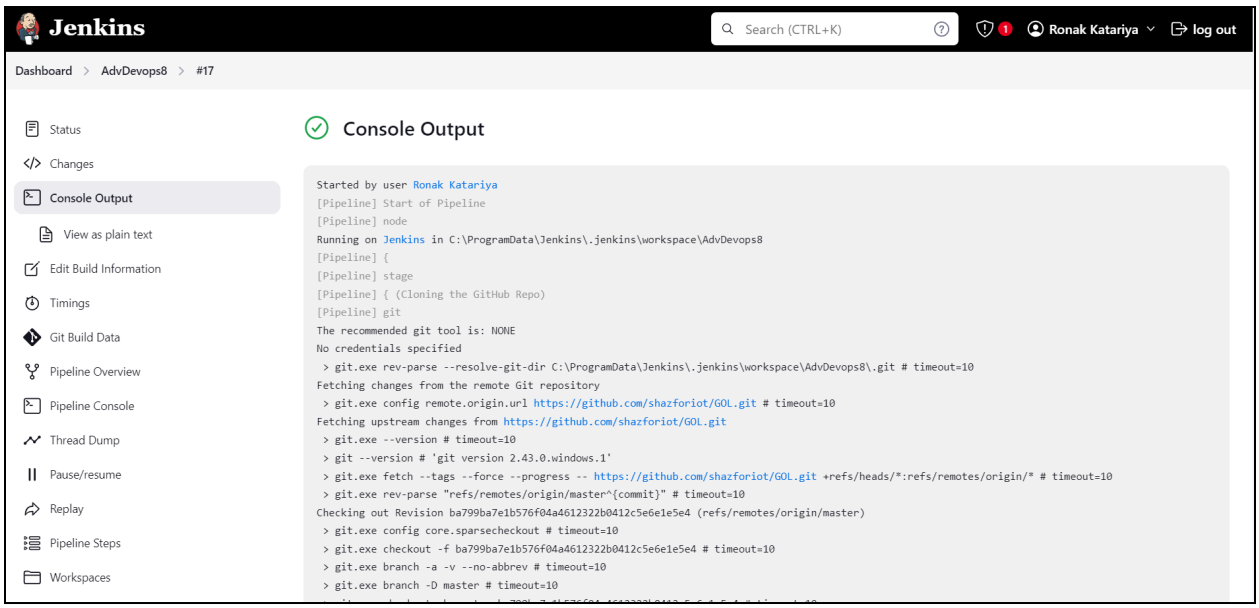
```
docker network create sonarnet
node {
 stage('Cloning the GitHub Repo') {
 git 'https://github.com/shazforiot/GOL.git'
 }
 stage('SonarQube analysis') {
 withSonarQubeEnv('sonarqube') {
 bat """
 docker run --rm --network host \
 -e SONAR_HOST_URL=http://<ip_address>:9000 \
 -e SONAR_LOGIN=admin \
 -e SONAR_PASSWORD=<Sonarqube_password> \
 -e SONAR_PROJECT_KEY=sonarqube-test \
 -v ${WORKSPACE.replace('\\', '/')}:/usr/src \
 sonarsource/sonar-scanner-cli \
 -Dsonar.projectKey=sonarqube-test \
 -Dsonar.exclusions=vendor/**,resources/**,**/*.java \
 -Dsonar.login=admin \
 -Dsonar.password=<Sonarqube_password>
 """
 }
 }
}
```



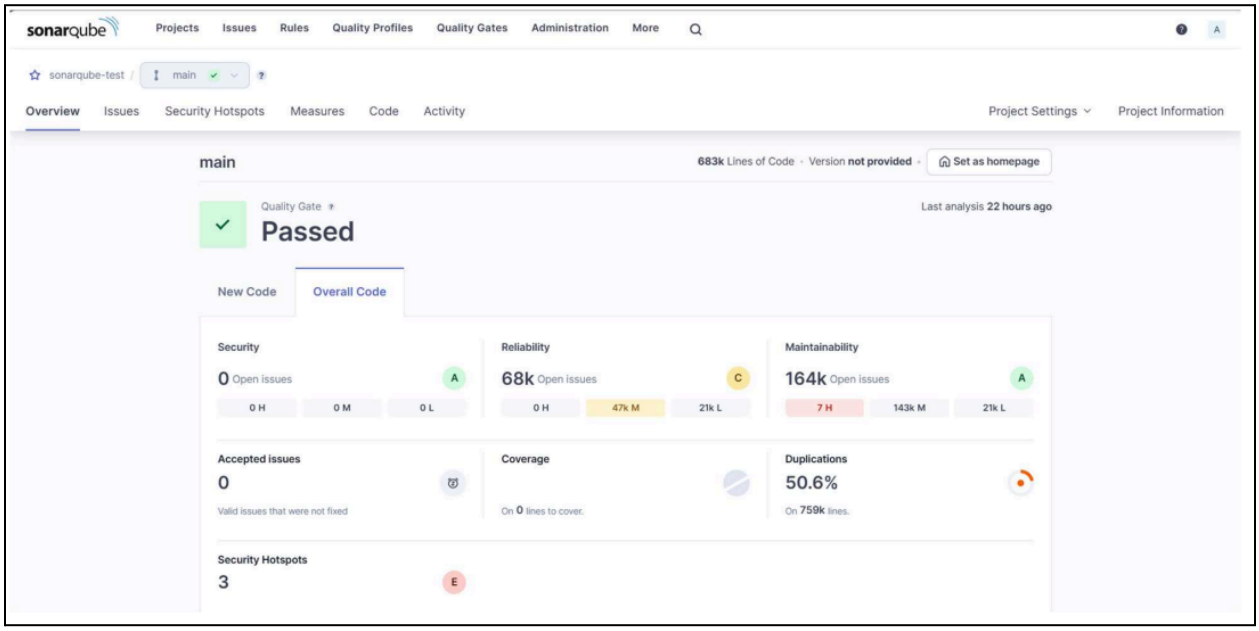
9. Run the Pipeline

- Save the pipeline and click Build Now.
- Monitor the console output for any errors.

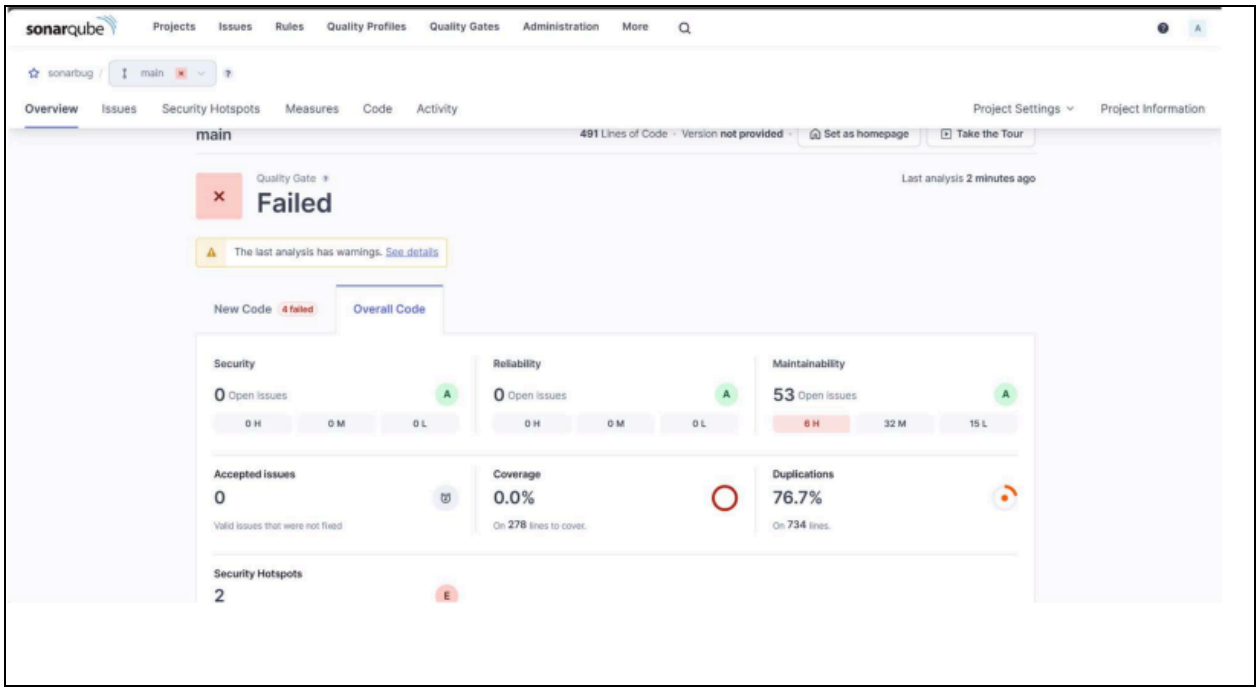




10. Check SonarQube for Analysis Results
    - Go to your SonarQube dashboard and check the project for issues such as bugs, code smells, and security vulnerabilities.



11. Checking SonarQube for Analysis Results of a Code File with Bugs , Code Smells, Security Vulnerabilities, Cyclomatic Complexities and Duplicates .
    - Overview -



Conclusion:

In this experiment, we performed a static analysis of the code to detect bugs, code smells, and security vulnerabilities on our sample codes.

Experiment 9

AIM:
To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Theory:

Introduction
Continuous monitoring is a critical aspect of IT infrastructure management, enabling organizations to maintain system reliability and performance. Nagios Core is a widely-used open-source monitoring system that provides comprehensive monitoring solutions for servers, network devices, and applications. This practical aims to understand the theoretical underpinnings of Nagios Core, its plugins, and the Nagios Remote Plugin Executor (NRPE).

Overview of Nagios Core

What is Nagios Core?

Nagios Core is an open-source application designed to monitor the status of various components in an IT environment. It allows administrators to track the health of servers, network devices, and applications in real-time, providing alerts and notifications for any issues that may arise.
The core functionality includes:
Host Monitoring: Monitoring the availability and performance of servers and network devices.
Service Monitoring: Keeping track of specific services (e.g., HTTP, FTP) running on hosts.
Alerting: Sending notifications via email or SMS when issues are detected.
Reporting: Generating reports on system performance and uptime.

Key Components

Nagios Core: The central engine that performs monitoring tasks.
Nagios Plugins: A set of scripts that extend Nagios's capabilities by allowing it to check various metrics (e.g., CPU load, disk usage).
NRPE (Nagios Remote Plugin Executor): A daemon that allows Nagios to execute plugins on remote hosts. This is essential for monitoring systems that cannot be directly accessed by the Nagios server.

Importance of Continuous Monitoring

Continuous monitoring is vital for several reasons:
Proactive Issue Detection: By continuously monitoring systems, organizations can identify potential problems before they escalate into significant outages.

Performance Optimization: Monitoring helps in understanding system performance trends, enabling optimization and resource allocation.
Compliance and Reporting: Many industries require compliance with regulations that mandate continuous monitoring and reporting of system health.
Improved Reliability: Continuous monitoring contributes to higher system availability, reducing downtime and improving user satisfaction.

Installation and Configuration of Nagios Core:

Installing Nagios Core involves setting up the core application on a server that will act as the monitoring host. This includes configuring a web interface for easy access to monitoring data, which allows administrators to visualize the status of their infrastructure.

Configuration of Nagios Plugins

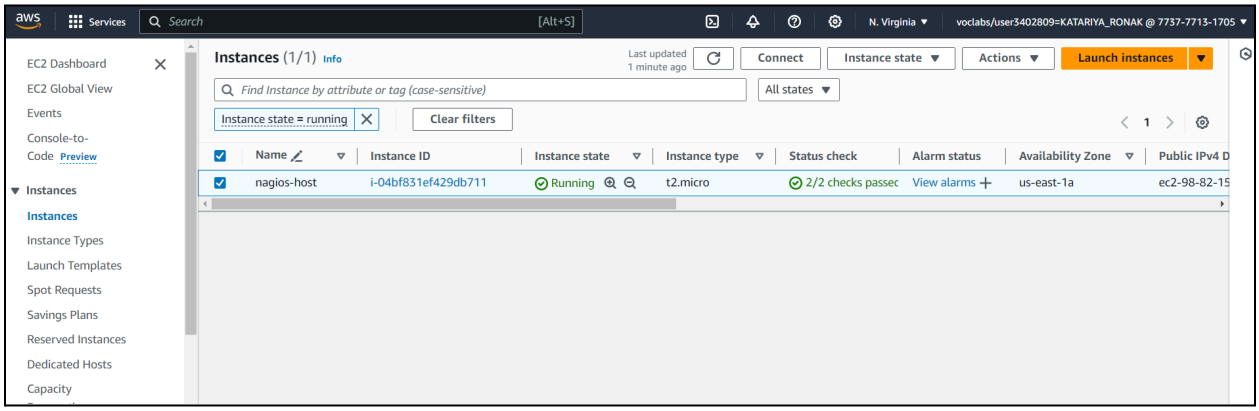
Nagios plugins are essential for extending the functionality of Nagios Core. These plugins perform specific checks on hosts and services, returning results back to the Nagios server. Proper configuration ensures that all necessary metrics are monitored effectively.

Deployment of NRPE

NRPE enables remote execution of plugins on monitored hosts. This is particularly useful for environments where direct access to servers is limited or where additional checks need to be performed on remote systems. Configuring NRPE involves setting up a daemon on each remote host and defining which checks can be executed remotely.

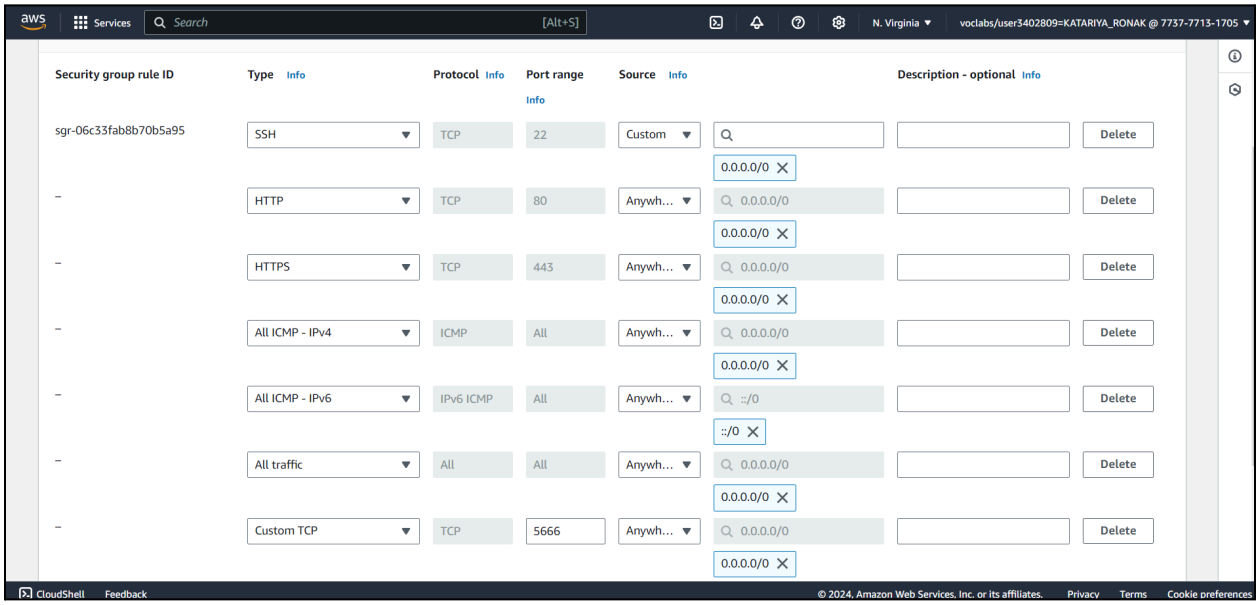
Implementation:

1. Create an Amazon Linux EC2 Instance
● Name it nagios-host.



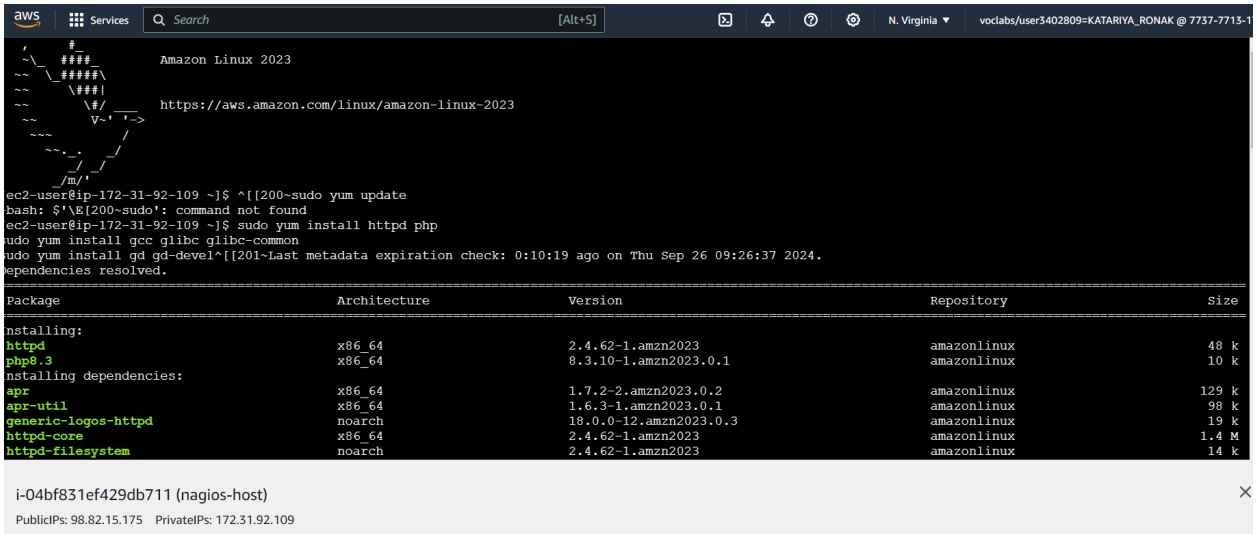
2. Configure Security Group

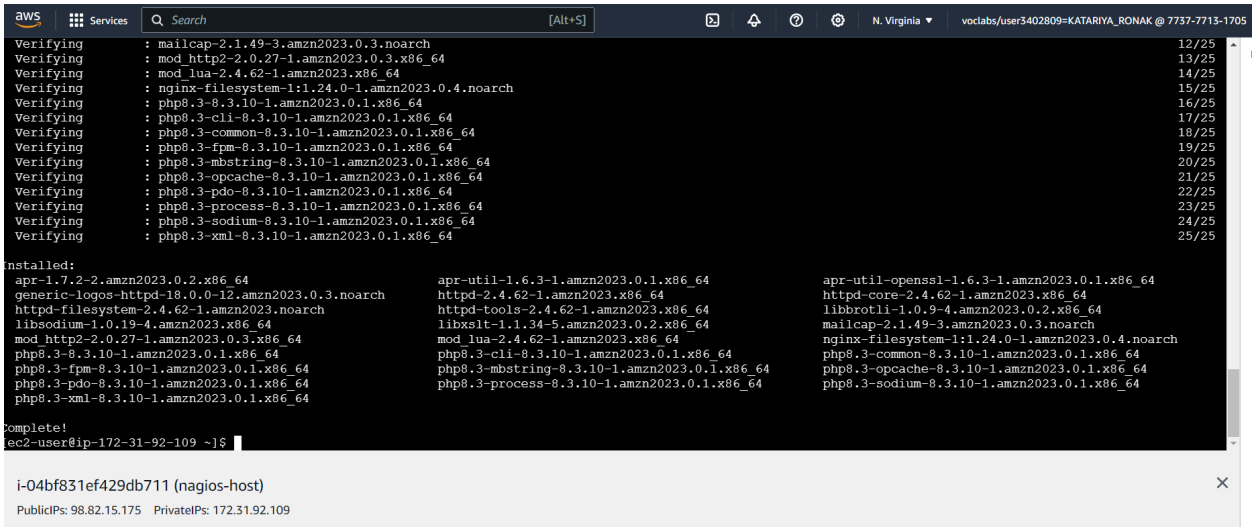
- Ensure HTTP, HTTPS, SSH, and ICMP are open from everywhere.
- Edit the inbound rules of the specified Security Group



3. Connect to Your EC2 Instance
- SSH into your EC2 instance or use EC2 Instance Connect from the browser

4. Update Package Indices and Install Required Packages
Commands -
sudo yum update
sudo yum install httpd php
sudo yum install gcc glibc glibc-common
sudo yum install gd gd-devel





5. Create a New Nagios User
Commands -
sudo adduser -m nagios
sudo passwd nagios

```
[ec2-user@ip-172-31-92-109 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password contains the user name in some form
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-92-109 ~]$
```

6. Create a New User Group
Commands -
sudo groupadd nagcmd

```
[ec2-user@ip-172-31-92-109 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-92-109 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-92-109 ~]$ mkdir ~/downloads
cd ~/downloads
```

7. Add Users to the Group
Commands -
sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-92-109 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-92-109 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-92-109 ~]$ mkdir ~/downloads
cd ~/downloads
```

8. Create a Directory for Nagios Downloads
Commands -
mkdir ~/downloads
cd ~/downloads

9. Download Nagios and Plugins Source Files
Commands -
wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz

```
[ec2-user@ip-172-31-92-109 downloads]$ wget
http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.
gz
wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try `wget --help' for more options.
-bash: http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.: No such file or directory
-bash: gz: command not found
--2024-09-26 09:42:16--  http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2659772 (2.5M) [application/x-gzip]
Saving to: 'nagios-plugins-2.0.3.tar.gz'

nagios-plugins-2.0.3.tar.gz             100%[=========================================================================>]   2.54M  8.63MB/s    in 0.3s

2024-09-26 09:42:16 (8.63 MB/s) - 'nagios-plugins-2.0.3.tar.gz' saved [2659772/2659772]
```

```
[ec2-user@ip-172-31-92-109 downloads]$  wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz
wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
--2024-09-26 09:44:10--  http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz
Resolving prdownloads.sourceforge.net (prdownloads.sourceforge.net)... 204.68.111.105
Connecting to prdownloads.sourceforge.net (prdownloads.sourceforge.net)|204.68.111.105|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz [following]
--2024-09-26 09:44:10--  http://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz
Resolving downloads.sourceforge.net (downloads.sourceforge.net)... 204.68.111.105
Reusing existing connection to prdownloads.sourceforge.net:80.
HTTP request sent, awaiting response... 302 Found
Location: http://psychz.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1 [following]
--2024-09-26 09:44:11--  http://psychz.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1
Resolving psychz.dl.sourceforge.net (psychz.dl.sourceforge.net)... 208.87.241.191
Connecting to psychz.dl.sourceforge.net (psychz.dl.sourceforge.net)|208.87.241.191|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1805059 (1.7M) [application/x-gzip]
Saving to: 'nagios-4.0.8.tar.gz'

nagios-4.0.8.tar.gz                     100%[=========================================================================>]   1.72M  2.60MB/s    in 0.7s

2024-09-26 09:44:12 (2.60 MB/s) - 'nagios-4.0.8.tar.gz' saved [1805059/1805059]

--2024-09-26 09:44:12--  http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
```

```
nagios-plugins-2.0.3/plugins-scripts/check_rpc.pl
nagios-plugins-2.0.3/plugins-scripts/check_oracle.sh
nagios-plugins-2.0.3/plugins-scripts/utils.pm.in
nagios-plugins-2.0.3/plugins-scripts/check_disk_smb.pl
nagios-plugins-2.0.3/plugins-scripts/t/
nagios-plugins-2.0.3/plugins-scripts/t/check_ifoperstatus.t
nagios-plugins-2.0.3/plugins-scripts/t/check_rpc.t
nagios-plugins-2.0.3/plugins-scripts/t/check_file_age.t
nagios-plugins-2.0.3/plugins-scripts/t/check_disk_smb.t
nagios-plugins-2.0.3/plugins-scripts/t/check_ifstatus.t
nagios-plugins-2.0.3/plugins-scripts/t/utils.t
nagios-plugins-2.0.3/plugins-scripts/check_mailq.pl
nagios-plugins-2.0.3/plugins-scripts/check_wave.pl
nagios-plugins-2.0.3/plugins-scripts/check_ircd.pl
nagios-plugins-2.0.3/plugins-scripts/utils.sh.in
nagios-plugins-2.0.3/plugins-scripts/check_ifstatus.pl
nagios-plugins-2.0.3/plugins-scripts/check_sensors.sh
nagios-plugins-2.0.3/pkg/
nagios-plugins-2.0.3/pkg/fedora/
nagios-plugins-2.0.3/pkg/fedora/requires
nagios-plugins-2.0.3/pkg/solaris/
nagios-plugins-2.0.3/pkg/solaris/preinstall
nagios-plugins-2.0.3/pkg/solaris/solpkg
nagios-plugins-2.0.3/pkg/solaris/pkginfo.in
nagios-plugins-2.0.3/pkg/solaris/pkginfo
nagios-plugins-2.0.3/pkg/redhat/
nagios-plugins-2.0.3/pkg/redhat/requires
[ec2-user@ip-172-31-92-109 downloads]$
```

i-04bf831ef429db711 (nagios-host)

PublicIPs: 98.82.15.175 PrivateIPs: 172.31.92.109

10. Extract the Nagios Source File
Commands -
tar zxvf nagios-4.4.6.tar.gz
cd nagios-4.4.6

```
[ec2-user@ip-172-31-92-109 downloads]$  cd nagios-4.0.8
```

11. Run the Configuration Script
Commands -
./configure --with-command-group=nagcmd

EXTRA (some packages were installed which when not installed were giving errors)

```
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ sudo yum update -y
Last metadata expiration check: 0:25:56 ago on Thu Sep 26 09:26:37 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ sudo yum groupinstall "Development Tools" -y
Last metadata expiration check: 0:26:09 ago on Thu Sep 26 09:26:37 2024.
No match for group package "system-rpm-config"
No match for group package "rcs"
No match for group package "pkgconfig"
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing group/module packages:				
autoconf	noarch	2.69-36.amzn2023.0.3	amazonlinux	666 k
automake	noarch	1.16.5-9.amzn2023.0.3	amazonlinux	677 k
bison	x86_64	3.7.4-2.amzn2023.0.2	amazonlinux	925 k
byacc	x86_64	2.0.20210109-2.amzn2023.0.3	amazonlinux	90 k
cscope	x86_64	15.9-15.amzn2023.0.3	amazonlinux	288 k
ctags	x86_64	5.9-1.20210725.0.amzn2023.0.2	amazonlinux	719 k
diffstat	x86_64	1.64-4.amzn2023.0.2	amazonlinux	43 k
doxygen	x86_64	2:1.9.4-1.amzn2023.0.3	amazonlinux	4.7 M
elfutils	x86_64	0.188-3.amzn2023.0.2	amazonlinux	525 k
flex	x86_64	2.6.4-7.amzn2023.0.2	amazonlinux	310 k

i-04bf831ef429db711 (nagios-host)

PublicIPs: 98.82.15.175 PrivateIPs: 172.31.92.109

```
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ gcc --version
gcc (GCC) 11.4.1 20230605 (Red Hat 11.4.1-2)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ which gcc
/usr/bin/gcc
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ nano ~/.bashrc
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ source ~/.bashrc
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
```

EXTRA done

Change directory to nagios4.08
Then run configure file:

```
[ec2-user@ip-172-31-92-109 downloads]$ cd nagios-4.0.8
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for sys/types.h... yes
```

Configuration 👍

```
*** Configuration summary for nagios 4.0.8 08-12-2014 ***:

 General Options:
 -------------------------
        Nagios executable:  nagios
        Nagios user/group:  nagios,nagios
       Command user/group:  nagios,nagcmd
             Event Broker:  yes
        Install ${prefix}:  /usr/local/nagios
    Install ${includedir}:  /usr/local/nagios/include/nagios
                Lock file:  ${prefix}/var/nagios.lock
   Check result directory:  ${prefix}/var/spool/checkresults
           Init directory:  /etc/rc.d/init.d
  Apache conf.d directory:  /etc/httpd/conf.d
             Mail program:  /bin/mail
                  Host OS:  linux-gnu
          IOBroker Method:  epoll

 Web Interface Options:
 ------------------------
                HTML URL:  http://localhost/nagios/
                 CGI URL:  http://localhost/nagios/cgi-bin/
 Traceroute (used by WAP):  /usr/bin/traceroute


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.
```

12. Compile the Source Code
Commands -
make all

```
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/base'
make -C ../lib
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/lib'
gcc -Wall -g -O2 -DHAVE_CONFIG_H -c squeue.c -o squeue.o
gcc -Wall -g -O2 -DHAVE_CONFIG_H -c kvvec.c -o kvvec.o
gcc -Wall -g -O2 -DHAVE_CONFIG_H -c iocache.c -o iocache.o
gcc -Wall -g -O2 -DHAVE_CONFIG_H -c iobroker.c -o iobroker.o
gcc -Wall -g -O2 -DHAVE_CONFIG_H -c bitmap.c -o bitmap.o
gcc -Wall -g -O2 -DHAVE_CONFIG_H -c dkhash.c -o dkhash.o
gcc -Wall -g -O2 -DHAVE_CONFIG_H -c runcmd.c -o runcmd.o
runcmd.c: In function ‘runcmd_open’:
runcmd.c:347:12: warning: ‘nonnull’ argument ‘cmd’ compared to NULL [-Wnonnull-compare]
  347 |         if (!cmd || !*cmd || !pfd || !pfderr)
      |            ^
runcmd.c:347:30: warning: ‘nonnull’ argument ‘pfd’ compared to NULL [-Wnonnull-compare]
  347 |         if (!cmd || !*cmd || !pfd || !pfderr)
      |                              ^~~~
runcmd.c:347:38: warning: ‘nonnull’ argument ‘pfderr’ compared to NULL [-Wnonnull-compare]
  347 |         if (!cmd || !*cmd || !pfd || !pfderr)
      |                                      ^~~~~~~
runcmd.c:389:12: warning: ‘nonnull’ argument ‘iobreg’ compared to NULL [-Wnonnull-compare]
```

```
[ec2-user@ip-172-31-92-109 nagios-4.0.8]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o broker.o broker.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o nebmods.o nebmods.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o nerd.o nerd.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o query-handler.o query-handler.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function ‘get_wproc_list’,
    inlined from ‘get_worker’ at workers.c:224:12:
workers.c:209:17: warning: ‘%s’ directive argument is null [-Wformat-overflow=]
  209 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/') ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o checks.o checks.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o config.o config.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o commands.o commands.c
commands.c: In function ‘process_passive_service_check’:
commands.c:2247:19: warning: assignment discards ‘const’ qualifier from pointer target type [-Wdiscarded-qualifiers]
 2247 |         cr.source = command_worker.source_name;
      |                   ^
```

```
*** Support Notes *******************************************

If you have questions about configuring or running Nagios,
please make sure that you:

     - Look at the sample config files
     - Read the documentation on the Nagios Library at:
          https://library.nagios.com

before you post a question to one of the mailing lists.
Also make sure to include pertinent information that could
help others help you.  This might include:

     - What version of Nagios you are using
     - What version of the plugins you are using
     - Relevant snippets from your config files
     - Relevant error messages from the Nagios log file

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

************************************************************

Enjoy.

[ec2-user@ip-172-31-81-173 nagios-4.4.6]$
```

13. Install Binaries, Init Script, and Sample Config Files

Commands -
./sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode

```
Enjoy.

[ec2-user@ip-172-31-81-173 nagios-4.4.6]$ ./sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
-bash: ./sudo: No such file or directory
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

[ec2-user@ip-172-31-81-173 nagios-4.4.6]$ ~
```

14. Edit the Config File to Change the Email Address

Commands -
sudo nano /usr/local/nagios/etc/objects/contacts.cfg

- Change the email address in the contacts.cfg file to your preferred email.

```
  GNU nano 5.8                                  /usr/local/nagios/etc/objects/contacts.cfg
#
###############################################################################

# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

define contact {

    contact_name            nagiosadmin             ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined above)
    alias                   Nagios Admin            ; Full name of user
    email                   masterincoding03@gmail.com ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}



###############################################################################
#
# CONTACT GROUPS
#
###############################################################################

# We only have one contact in this simple configuration file, so there is

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] T
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q WH
```

15. Configure the Web Interface

Commands -
sudo make install-webconf

```
[ec2-user@ip-172-31-81-173 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

[ec2-user@ip-172-31-81-173 nagios-4.4.6]$
```

16. Create a Nagios Admin Account

Commands -
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
● You will be prompted to enter and confirm the password for the nagiosadmin user.

```
[ec2-user@ip-172-31-81-173 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-81-173 nagios-4.4.6]$
```

17. Restart Apache

Commands -
sudo systemctl restart httpd

```
[ec2-user@ip-172-31-81-173 nagios-4.4.6]$ sudo systemctl restart httpd
```

18. Extract the Plugins Source File

Commands -
cd ~/downloads
tar zxvf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3

19. Compile and Install Plugins

Commands -
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install

```
[ec2-user@ip-172-31-81-173 nagios-plugins-2.3.3]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to disable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for Minix Amsterdam compiler... no
checking for ar... ar
```

i-02099de677d2ddadf (nagios-host)

PublicIPs: 34.226.142.160 PrivateIPs: 172.31.81.173

20. Start Nagios

Commands -
sudo chkconfig --add nagios
sudo chkconfig nagios on
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo systemctl start nagios

```
[ec2-user@ip-172-31-81-173 nagios-4.4.6]$ sudo systemctl enable httpd
sudo systemctl start httpd
sudo systemctl enable nagios
sudo systemctl start nagios
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-81-173 nagios-4.4.6]$
```

```
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-root'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-root'
Making install in po
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/po'
/usr/bin/mkdir -p /usr/local/nagios/share
installing fr.gmo as /usr/local/nagios/share/locale/fr/LC_MESSAGES/nagios-plugins.mo
installing de.gmo as /usr/local/nagios/share/locale/de/LC_MESSAGES/nagios-plugins.mo
if test "nagios-plugins" = "gettext-tools"; then \
  /usr/bin/mkdir -p /usr/local/nagios/share/gettext/po; \
  for file in Makefile.in.in remove-potcdate.sin    Makevars.template; do \
    /usr/bin/install -c -o nagios -g nagios -m 644 ./$file \
                  /usr/local/nagios/share/gettext/po/$file; \
  done; \
  for file in Makevars; do \
    rm -f /usr/local/nagios/share/gettext/po/$file; \
  done; \
else \
  : ; \
fi
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/po'
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3'
[ec2-user@ip-172-31-81-173 nagios-plugins-2.3.3]$
```

i-02099de677d2ddadf (nagios-host)

21. Check the Status of Nagios

Commands -
sudo systemctl status nagios

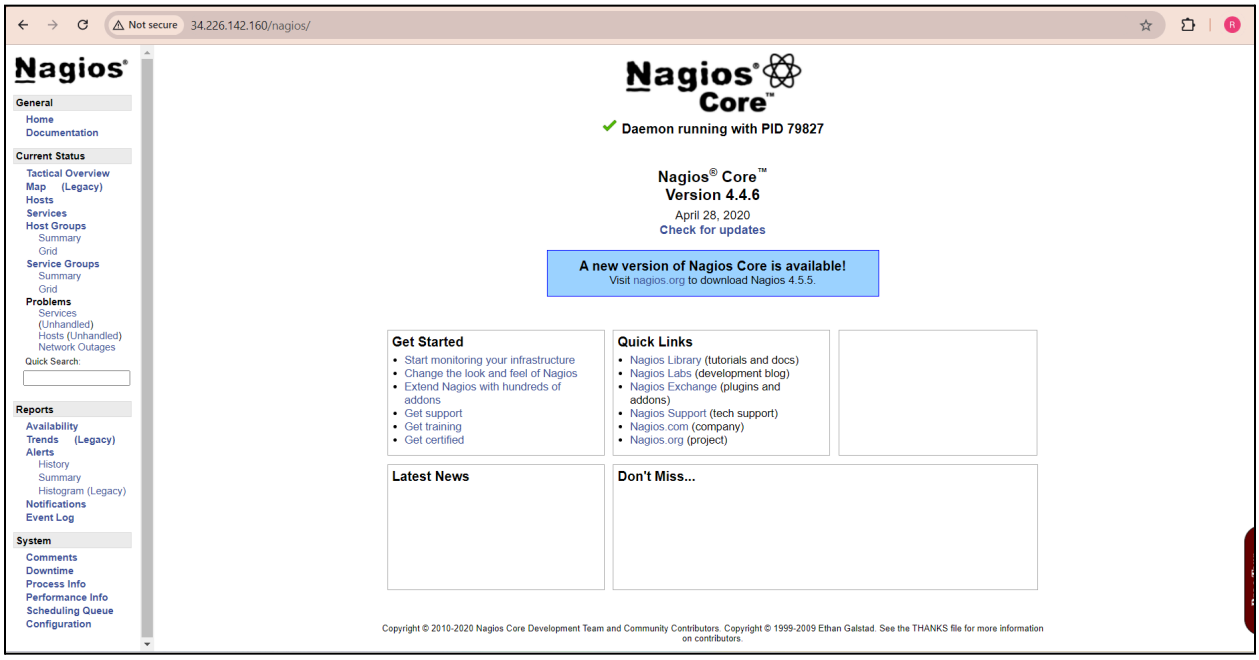
```
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Tue 2024-10-01 16:44:47 UTC; 26s ago
       Docs: https://www.nagios.org/documentation
    Process: 79825 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 79826 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 79827 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.1M
        CPU: 21ms
     CGroup: /system.slice/nagios.service
             ├─79827 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─79828 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─79829 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─79830 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─79831 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─79832 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: qh: core query handler registered
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: qh: echo service query handler registered
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: qh: help for the query handler registered
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: wproc: Successfully registered manager as @wproc with query handler
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: wproc: Registry request: name=Core Worker 79830;pid=79830
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: wproc: Registry request: name=Core Worker 79831;pid=79831
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: wproc: Registry request: name=Core Worker 79829;pid=79829
Oct 01 16:44:47 ip-172-31-81-173.ec2.internal nagios[79827]: wproc: Registry request: name=Core Worker 79828;pid=79828
lines 1-27
```

i-02099de677d2ddadf (nagios-host)

22. Access Nagios Web Interface

- Copy the Public IP address of your EC2 instance.
- Open your browser and navigate to http://<your_public_ip_address>/nagios.

- Enter the username nagiosadmin and the password you set in Step 16.



EXTRA:

Downloaded extra packages:

```
[ec2-user@ip-172-31-81-173 nagios-plugins-2.3.3]$ sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
Last metadata expiration check: 0:33:33 ago on Tue Oct  1 15:46:18 2024.
[MIRROR] epel-release-latest-7.noarch.rpm: Status code: 404 for https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm (IP: 38.145.60.23)
[MIRROR] epel-release-latest-7.noarch.rpm: Status code: 404 for https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm (IP: 38.145.60.23)
[MIRROR] epel-release-latest-7.noarch.rpm: Status code: 404 for https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm (IP: 38.145.60.23)
[MIRROR] epel-release-latest-7.noarch.rpm: Status code: 404 for https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm (IP: 38.145.60.23)
[FAILED] epel-release-latest-7.noarch.rpm: Status code: 404 for https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm (IP: 38.145.60.23)
Status code: 404 for https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm (IP: 38.145.60.23)
[ec2-user@ip-172-31-81-173 nagios-plugins-2.3.3]$
```

EXTRA:

```
[ec2-user@ip-172-31-81-173 nagios-plugins-2.3.3]$ sudo yum install -y gcc glibc glibc-common perl httpd php
sudo yum install -y gd gd-devel
Last metadata expiration check: 0:53:46 ago on Tue Oct  1 15:46:18 2024.
Package gcc-11.4.1-2.amzn2023.0.2.x86_64 is already installed.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package httpd-2.4.62-1.amzn2023.x86_64 is already installed.
Package php8.3-8.3.10-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
================================================================================================================
 Package                   Architecture      Version                          Repository               Size
================================================================================================================
Installing:
 perl                      x86_64            4:5.32.1-477.amzn2023.0.6        amazonlinux              13 k
Installing dependencies:
 gcc-c++                   x86_64            11.4.1-2.amzn2023.0.2            amazonlinux              12 M
 libdatrie                 x86_64            0.2.13-1.amzn2023.0.2            amazonlinux              33 k
 libstdc++-devel           x86_64            11.4.1-2.amzn2023.0.2            amazonlinux             2.2 M
 libthai                   x86_64            0.1.28-6.amzn2023.0.2            amazonlinux             209 k
 perl-Algorithm-Diff       noarch            1.2010-2.amzn2023.0.2            amazonlinux              47 k
 perl-Archive-Tar          noarch            2.40-1.amzn2023.0.2              amazonlinux              72 k
```

Experiment 10

Aim:
To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Theory:

Introduction

Monitoring network ports and services is essential for maintaining the health and security of IT infrastructure. Nagios is a powerful open-source monitoring tool that enables organizations to track the availability and performance of their servers, services, and network devices. This practical focuses on the theoretical aspects of performing port and service monitoring, as well as monitoring Windows and Linux servers using Nagios.

Importance of Port and Service Monitoring

Ports are communication endpoints for applications on a server, identified by port numbers. Monitoring these ports is crucial for several reasons:

- Security: Open ports can be potential entry points for unauthorized access. Monitoring helps identify any unauthorized or unexpected open ports, reducing the risk of security breaches.
- Availability: Ensuring that critical services are running on their designated ports is vital for maintaining application availability. If a service goes down, it can lead to significant disruptions in business operations.
- Performance: Monitoring ports allows administrators to track the performance of applications. High latency or failure to respond on specific ports can indicate underlying issues that need to be addressed.

Nagios Architecture

Nagios operates on a client-server model:

- Nagios Core: The central monitoring engine that performs checks on hosts and services.
- Plugins: Scripts that extend Nagios's functionality by performing specific checks (e.g., checking if a port is open or if a service is running).
- NRPE (Nagios Remote Plugin Executor): A daemon that allows Nagios to execute plugins on remote hosts, enabling monitoring of systems not directly accessible by the Nagios server.

Monitoring Ports with Nagios

Nagios can monitor both TCP and UDP ports using various plugins. The process involves:

1. Plugin Selection: Administrators can choose from numerous community-provided plugins designed for port monitoring. For example, check_open_port is a plugin that checks specified ports on a host and alerts if any unauthorized ports are found open1.

2. Configuration: Each plugin must be configured with the necessary parameters, such as the IP address of the host and the specific ports to monitor. This configuration ensures that Nagios can accurately check the status of each port.
3. Alerting Mechanism: When a monitored port becomes unavailable or an unexpected port opens, Nagios triggers alerts via email or SMS, allowing administrators to take immediate action.

Monitoring Services

In addition to port monitoring, Nagios also allows for service monitoring:

- Service Checks: Nagios can check whether specific services (e.g., HTTP, FTP) are running on designated ports. This is typically done using plugins like check_http or check_ftp, which attempt to connect to the service and verify its operational status.
- Custom Checks: Administrators can create custom checks tailored to their environment's needs, ensuring comprehensive monitoring across all critical services.

Windows and Linux Server Monitoring

Nagios supports monitoring across different operating systems, including Windows and Linux:

- Linux Server Monitoring: Using NRPE or SSH, administrators can perform checks on Linux servers remotely. Common checks include CPU load, disk usage, memory consumption, and service status.
- Windows Server Monitoring: For Windows environments, Nagios uses agents like NSClient++ to facilitate communication between the Nagios server and Windows hosts. This allows for checks similar to those performed on Linux systems but tailored for Windows-specific metrics.

Implementation:

Prerequisites

- AWS Free Tier
- Nagios Server running on an Amazon Linux Machine

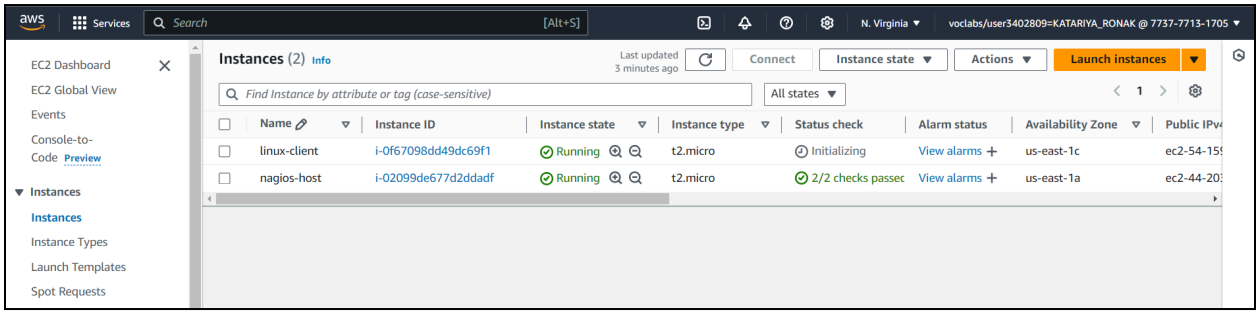
Steps:

1. Confirm Nagios is Running on the Server
Commands -
sudo systemctl status nagios

- Proceed if you see that Nagios is active and running.

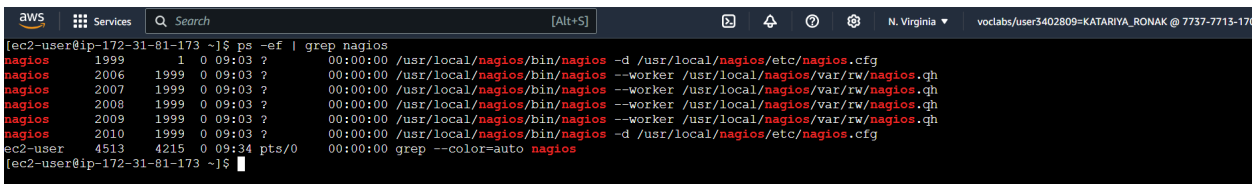
```
● ip-172-31-81-173.ec2.internal
    State: running
    Units: 296 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2024-10-02 09:03:18 UTC; 26min ago
  systemd: 252.23-2.amzn2023
   CGroup: /
           ├─init.scope
           │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize=32
           ├─system.slice
           │ ├─acpid.service
           │ │ ├─1955 /usr/bin/systemd-inhibit --what=handle-suspend-key:handle-hibernate-key --who=noah "
           │ │ └─1996 /usr/sbin/acpid -f
           │ ├─amazon-ssm-agent.service
           │ │ └─2341 /usr/bin/amazon-ssm-agent
           │ ├─atd.service
           │ │ └─2352 /usr/sbin/atd -f
           │ ├─auditd.service
           │ │ └─1778 /sbin/auditd
           │ ├─chronyd.service
           │ │ └─2375 /usr/sbin/chronyd -F 2
           │ ├─dbus-broker.service
           │ │ ├─1963 /usr/bin/dbus-broker-launch --scope system --audit
           │ │ └─1971 dbus-broker --log 4 --controller 9 --machine-id ec2c59ef5fdf3c9248d24ff5801dc348 --ma
           │ ├─gssproxy.service
           │ │ └─1998 /usr/sbin/gssproxy -D
lines 1-27
```

2. Create an Ubuntu 20.04 Server EC2 Instance
    - Name it linux-client.
    - Use the same security group as the Nagios Host.



3. Verify Nagios Process on the Server

Commands -
ps -ef | grep nagios



4. Become Root User and Create Directories

Commands -
sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-81-173 ~]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-81-173 ec2-user]# ls
```

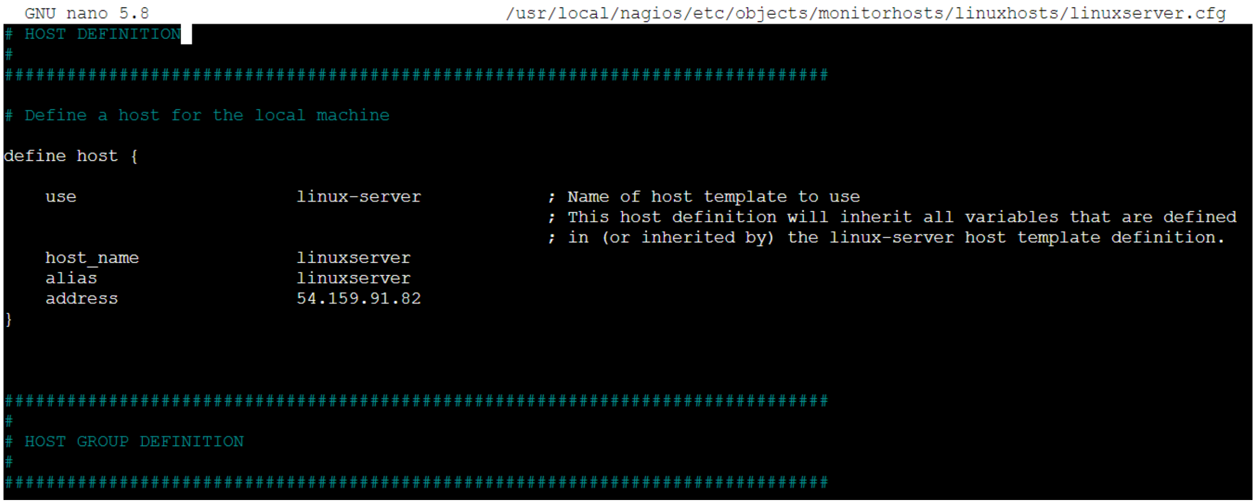
5. Copy Sample Configuration File
Commands -
cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

6. Edit the Configuration File

Commands -
sudo nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

- Change hostname to linuxserver everywhere in the file.
- Change address to the public IP address of your linux-client.

```
  GNU nano 5.8                          /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linux
}



###############################################################################
#
# SERVICE DEFINITIONS
#
###############################################################################

# Define a service to "ping" the local machine

define service {

    use                     local-service           ; Name of service template to use
    host_name               linuxserver
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
}



# Define a service to check the disk space of the root partition
# on the local machine.  Warning if < 20% free, critical if

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo
```



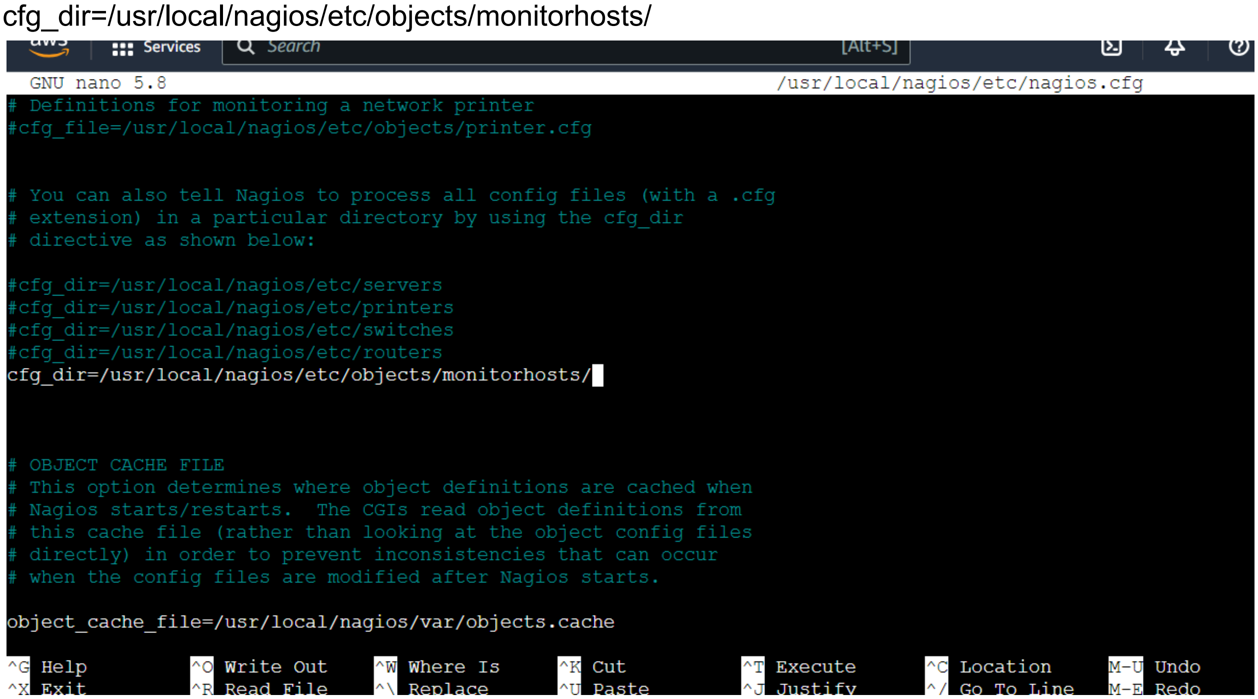
- Change hostgroup_name under hostgroup to linux-servers1.

7. Update Nagios Configuration

Commands -
sudo nano /usr/local/nagios/etc/nagios.cfg

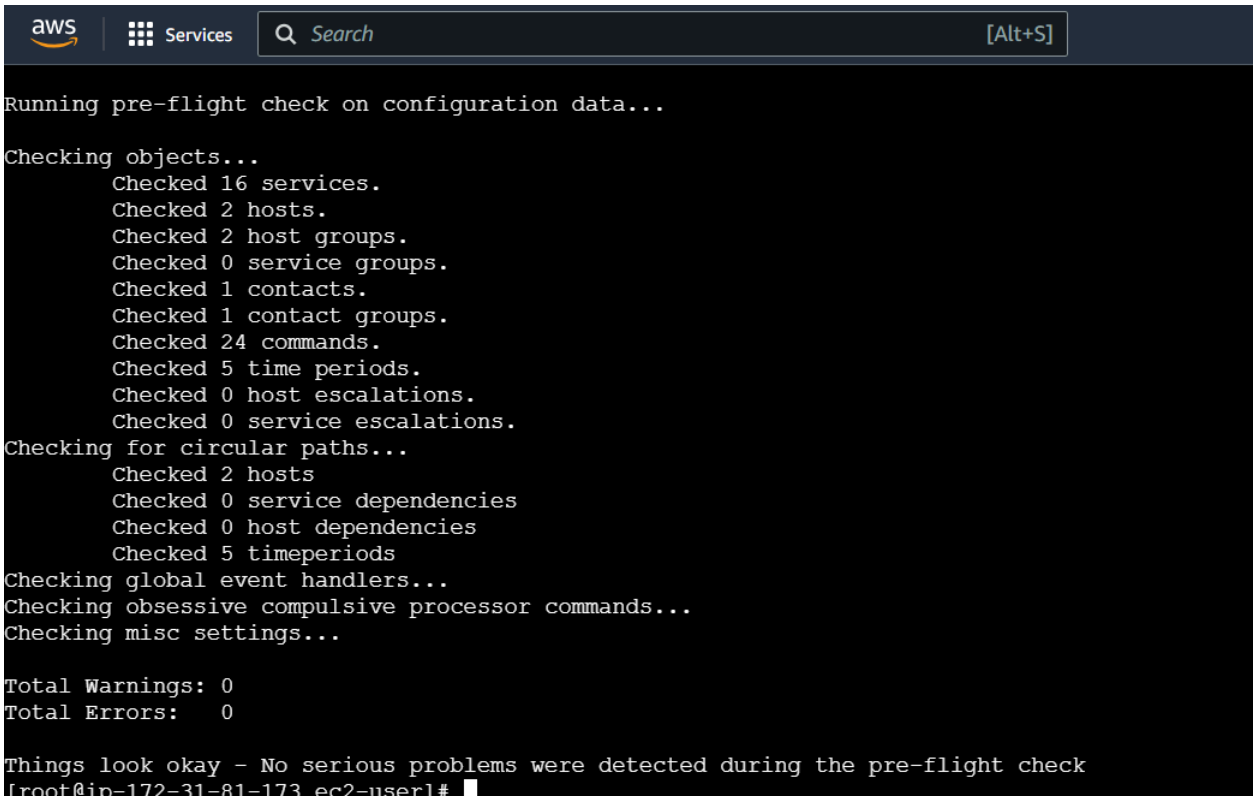
- Add the following line:

cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/



8. Verify Configuration Files

Commands -
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
● Ensure there are no errors.



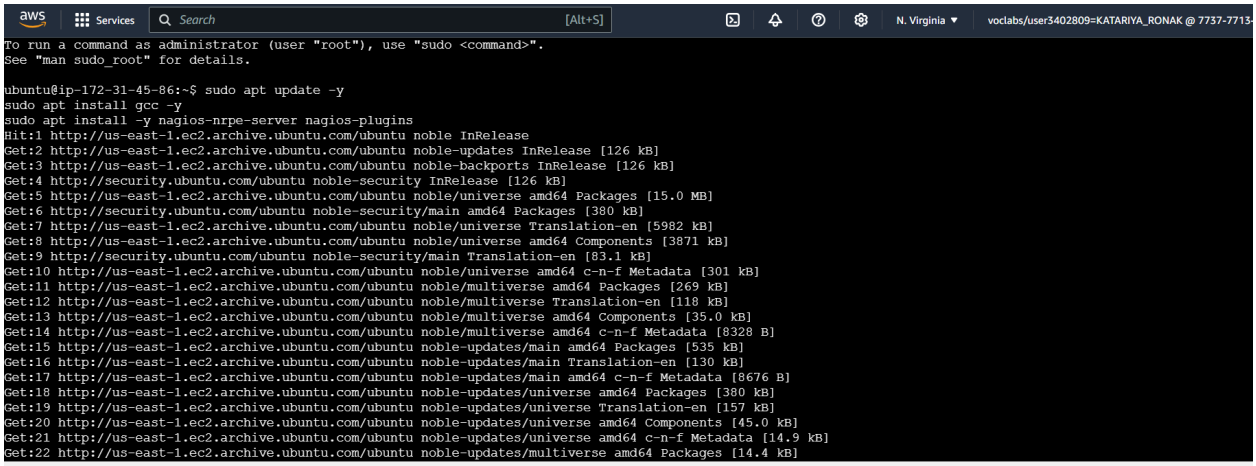
9. Restart Nagios Service

Commands -
sudo systemctl restart nagios

10. SSH into the Client Machine
    ● Use SSH or EC2 Instance Connect to access the linux-client.

11. Update Package Index and Install Required Packages

Commands -
sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins

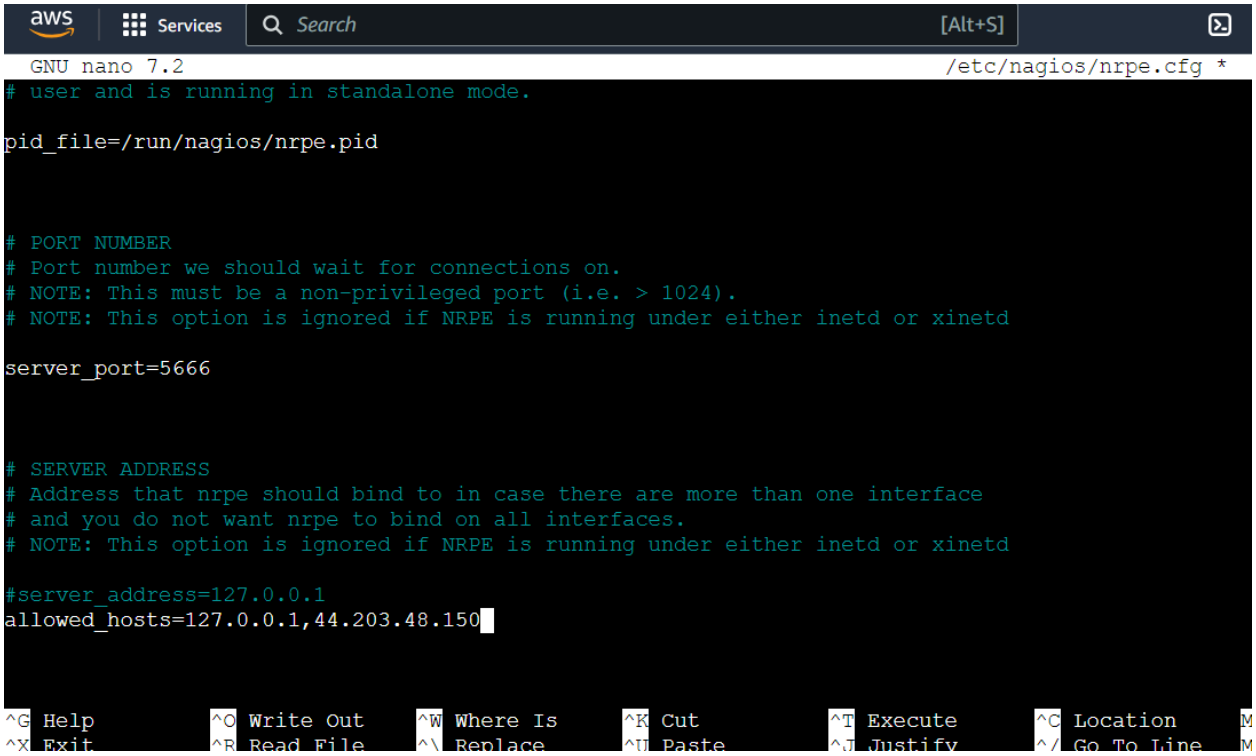


12. Edit NRPE Configuration File

Commands -
sudo nano /etc/nagios/nrpe.cfg

- Add your Nagios host IP address under allowed_hosts:

allowed_hosts=<Nagios_Host_IP>

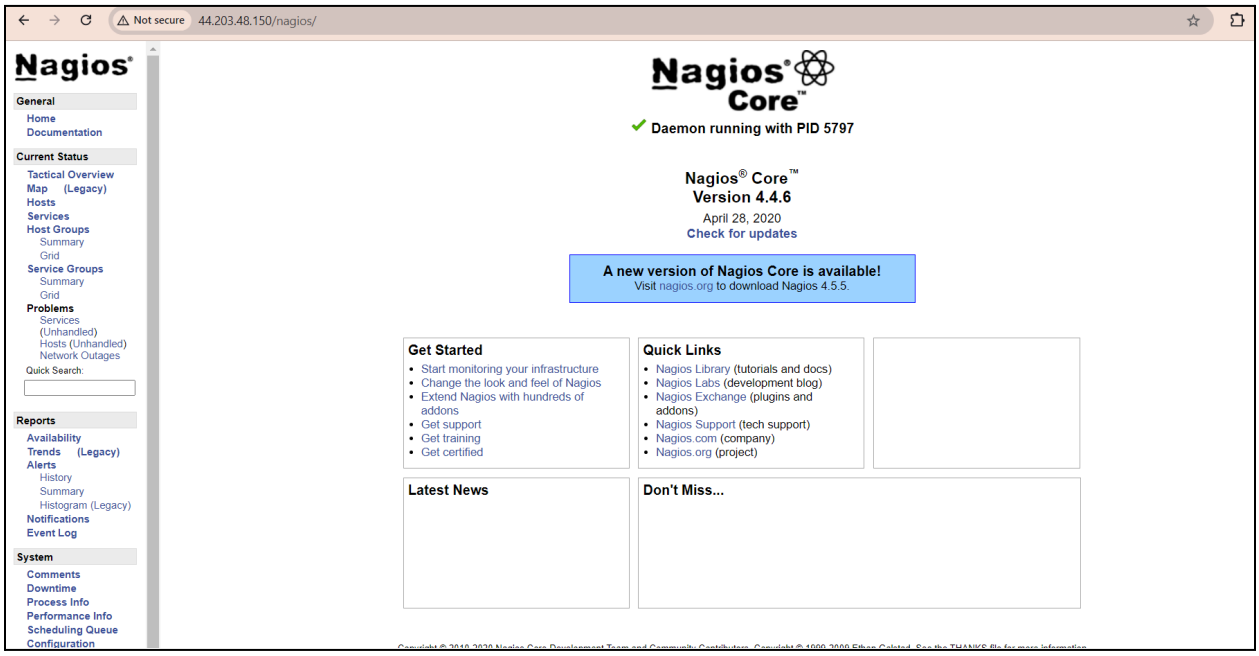


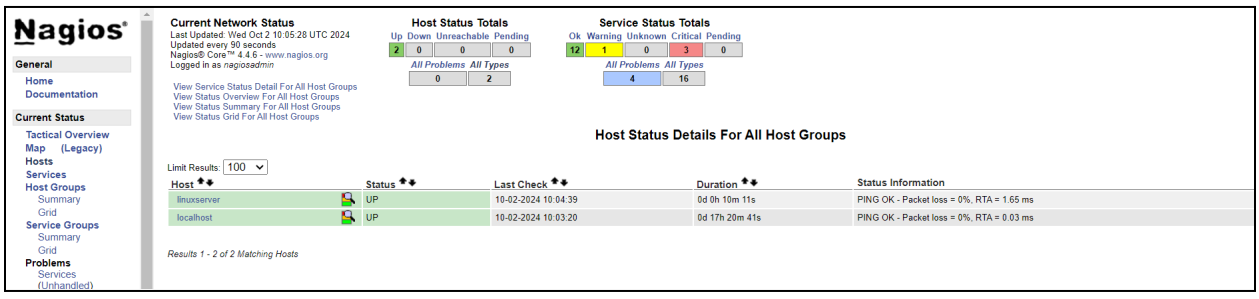
13. Restart NRPE Server

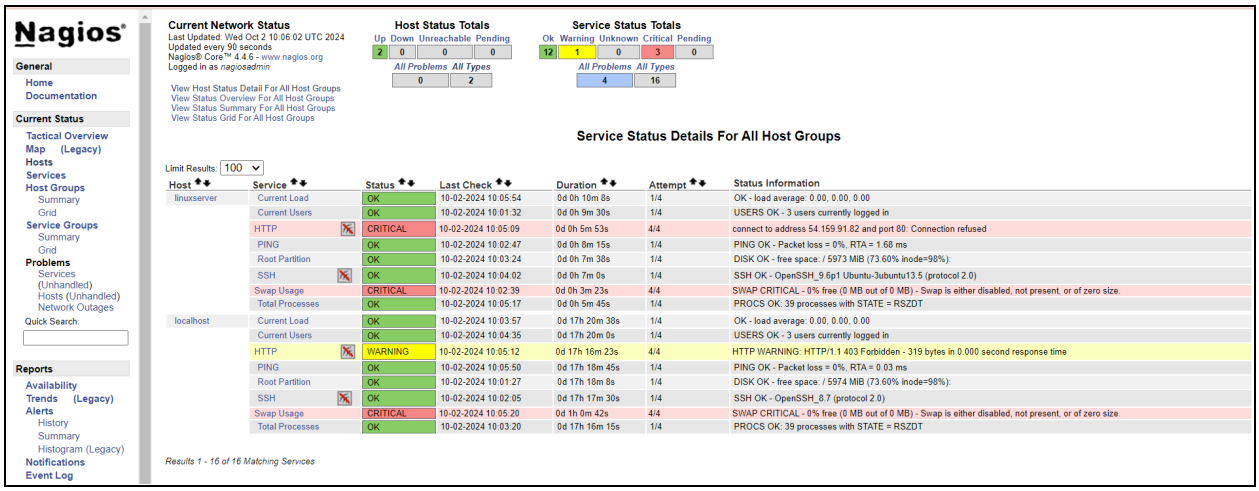
Commands -
sudo systemctl restart nagios-nrpe-server

14. Check Nagios Dashboard

- Open your browser and navigate to http://<Nagios_Host_IP>/nagios.
- Log in with nagiosadmin and the password you set earlier.
- You should see the new host linuxserver added.
- Click on Hosts to see the host details.
- Click on Services to see all services and ports being monitored






Nagios
General
Home
Documentation
Current Status
Tactical Overview
Map (Legacy)
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
Hosts (Unhandled)
Network Outages
Quick Search:
Reports
Availability
Trends (Legacy)
Alerts
History
Summary
Histogram (Legacy)
Notifications
Event Log
Current Network Status
Last Updated: Wed Oct 2 10:06:02 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as nagiosadmin
View Host Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups
Host Status Totals
Up Down Unreachable Pending
2 0 0 0
All Problems All Types
0 2
Service Status Totals
Ok Warning Unknown Critical Pending
12 1 0 3 0
All Problems All Types
4 16
Service Status Details For All Host Groups
Limit Results: 100
Host Service Status Last Check Duration Attempt Status Information
linuxserver Current Load OK 10-02-2024 10:05:54 0d 0h 10m 8s 1/4 OK - load average: 0.00, 0.00, 0.00
Current Users OK 10-02-2024 10:01:32 0d 0h 9m 30s 1/4 USERS OK - 3 users currently logged in
HTTP CRITICAL 10-02-2024 10:05:09 0d 0h 5m 53s 4/4 connect to address 54.159.91.82 and port 80: Connection refused
PING OK 10-02-2024 10:02:47 0d 0h 8m 15s 1/4 PING OK - Packet loss = 0%, RTA = 1.68 ms
Root Partition OK 10-02-2024 10:03:24 0d 0h 7m 38s 1/4 DISK OK - free space: / 5973 MiB (73.60% inode=98%):
SSH OK 10-02-2024 10:04:02 0d 0h 7m 0s 1/4 SSH OK - OpenSSH_9.6p1 Ubuntu-3ubuntu13.5 (protocol 2.0)
Swap Usage CRITICAL 10-02-2024 10:02:39 0d 0h 3m 23s 4/4 SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
Total Processes OK 10-02-2024 10:05:17 0d 0h 5m 45s 1/4 PROCS OK: 39 processes with STATE = RSZDT
localhost Current Load OK 10-02-2024 10:03:57 0d 17h 20m 38s 1/4 OK - load average: 0.00, 0.00, 0.00
Current Users OK 10-02-2024 10:04:35 0d 17h 20m 0s 1/4 USERS OK - 3 users currently logged in
HTTP WARNING 10-02-2024 10:05:12 0d 17h 16m 23s 4/4 HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.000 second response time
PING OK 10-02-2024 10:05:50 0d 17h 18m 45s 1/4 PING OK - Packet loss = 0%, RTA = 0.03 ms
Root Partition OK 10-02-2024 10:01:27 0d 17h 18m 8s 1/4 DISK OK - free space: / 5974 MiB (73.60% inode=98%):
SSH OK 10-02-2024 10:02:05 0d 17h 17m 30s 1/4 SSH OK - OpenSSH_8.7 (protocol 2.0)
Swap Usage CRITICAL 10-02-2024 10:05:20 0d 1h 0m 42s 4/4 SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
Total Processes OK 10-02-2024 10:03:20 0d 17h 16m 15s 1/4 PROCS OK: 39 processes with STATE = RSZDT
Results 1 - 16 of 16 Matching Services

Experiment 11

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Theory:

AWS Lambda

- AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.
- The Lambda functions can perform any kind of computing task, from serving web pages
and processing streams of data to calling APIs and integrating with other AWS services. The concept of “serverless” computing refers to not needing to maintain your own servers to run these functions.
- AWS Lambda is a fully managed service that takes care of all the infrastructure for you.
And so “serverless” doesn’t mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

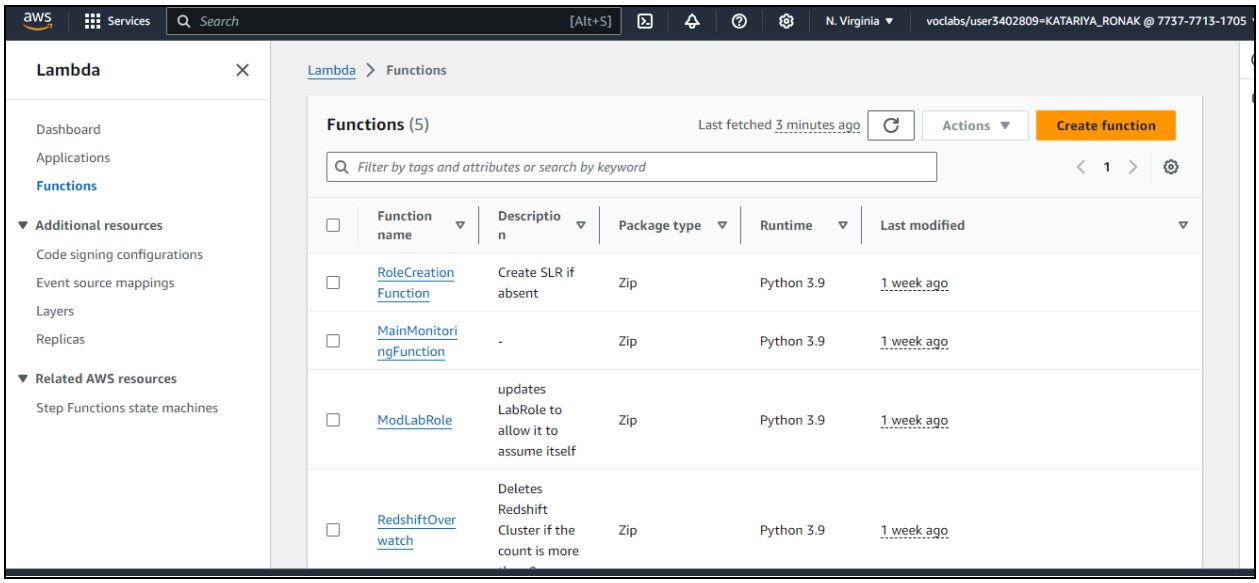
Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don’t need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
- It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down

Implementation:

Steps to create an AWS Lambda function

1. Open up the Lambda Console and click on the Create button. Be mindful of where you create your functions since Lambda is region-dependent.

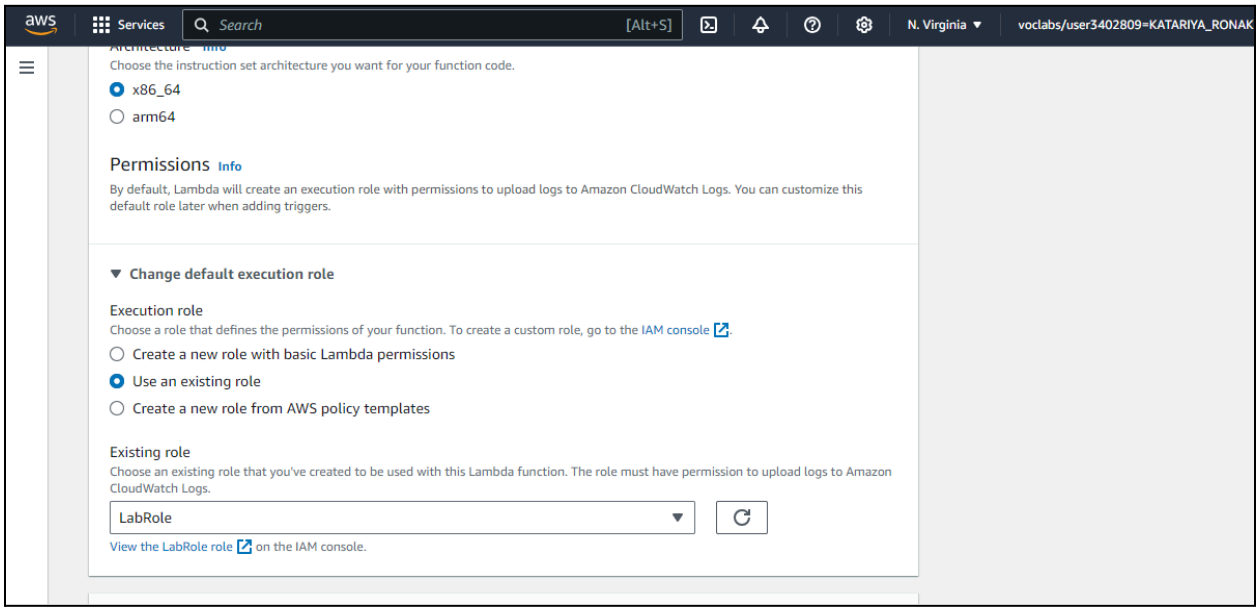


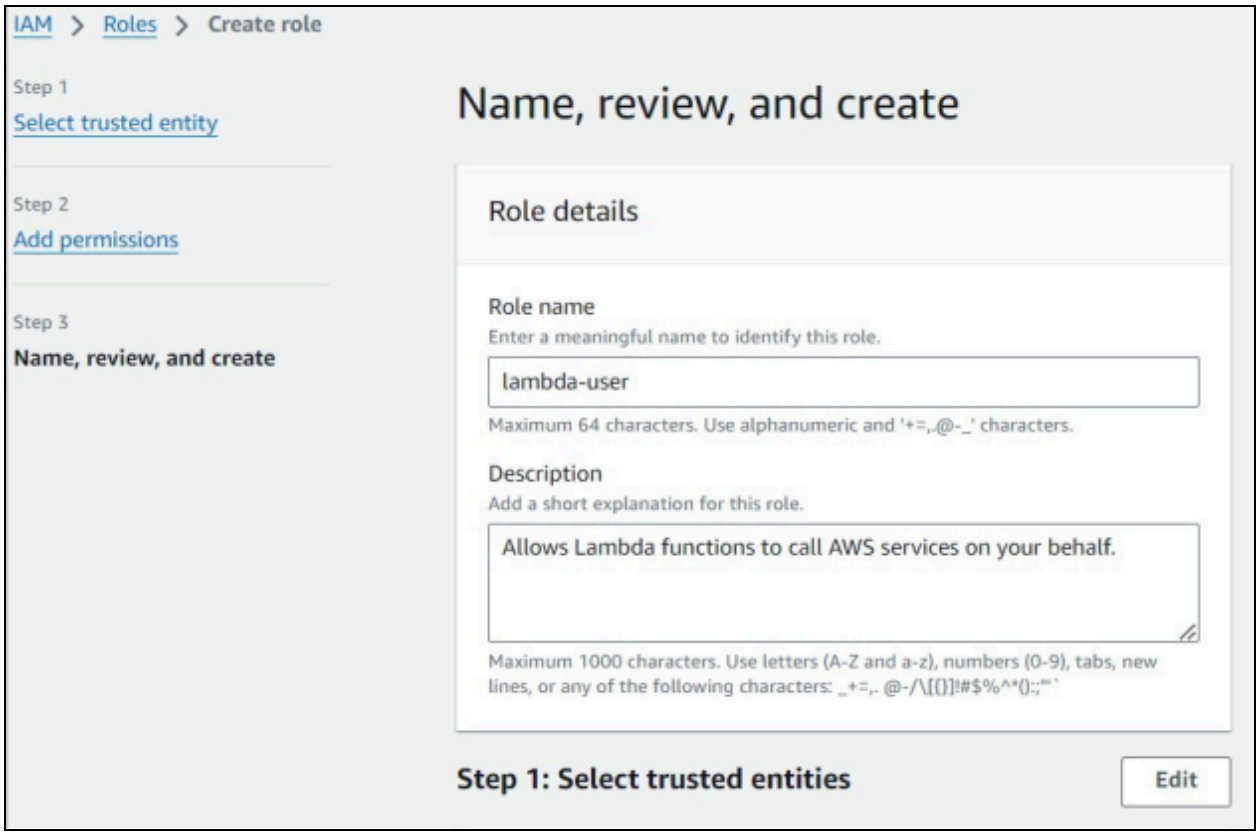
2. Attach CloudWatch Logs permissions:

● In the "Permissions" step, search for the policy called AWSLambdaBasicExecutionRole.

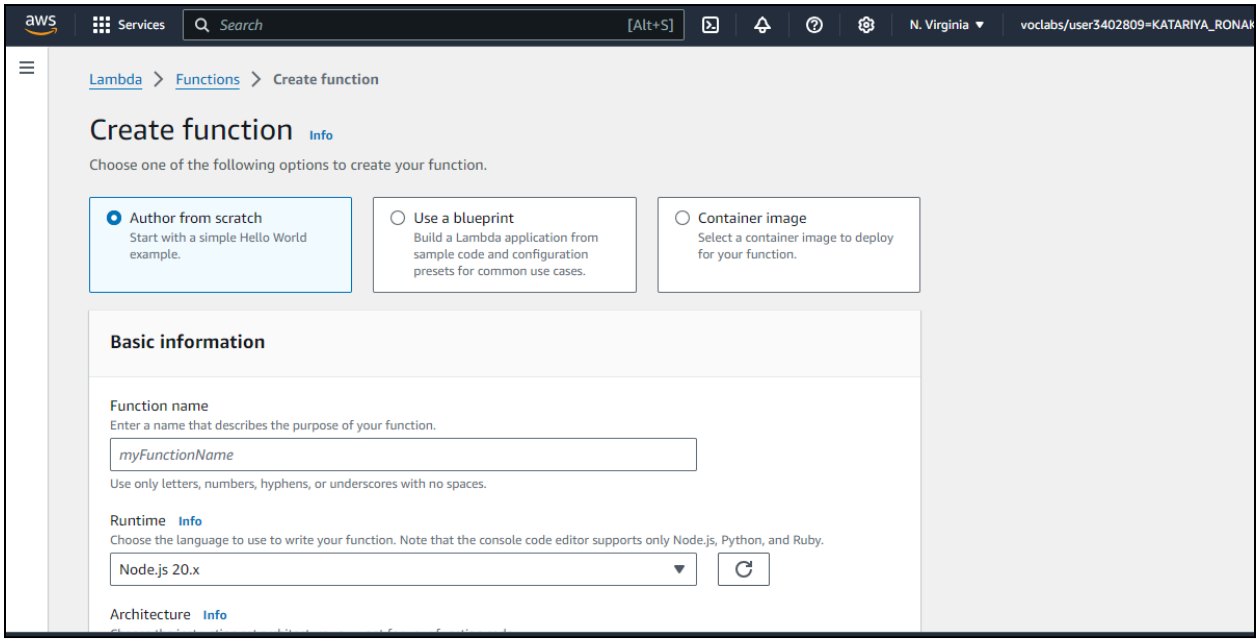
● Select this policy, which gives your Lambda function permission to write logs to CloudWatch.

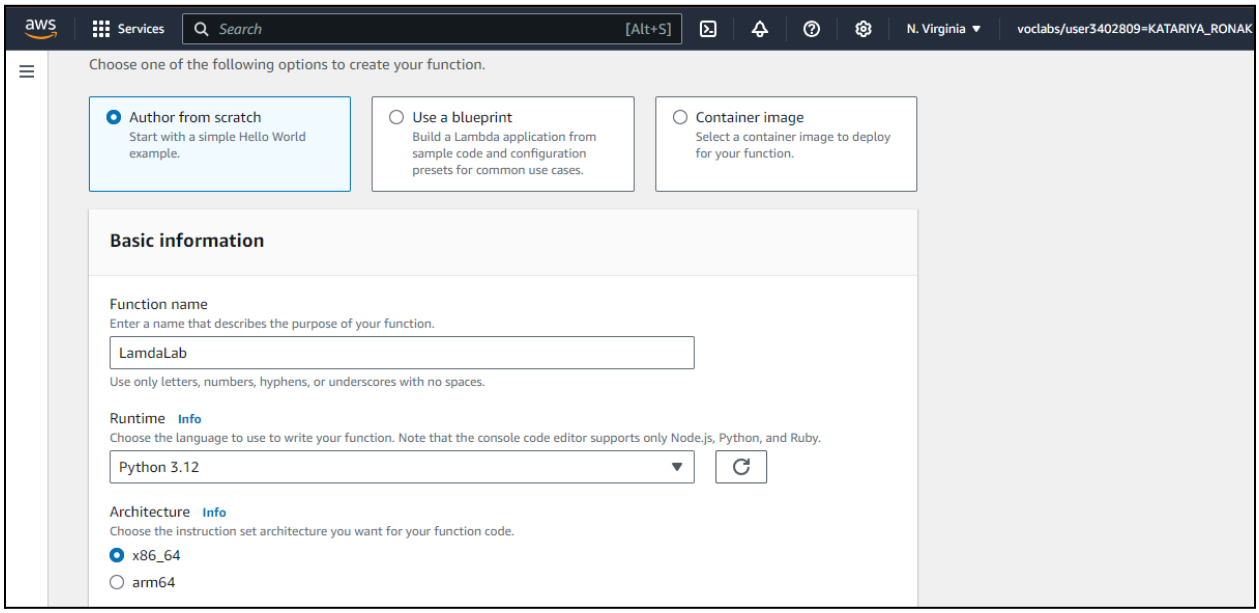
● Click Next.





2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.
Then, choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones.
After that, choose to create a new role with basic Lambda permissions if you don't have an existing one.



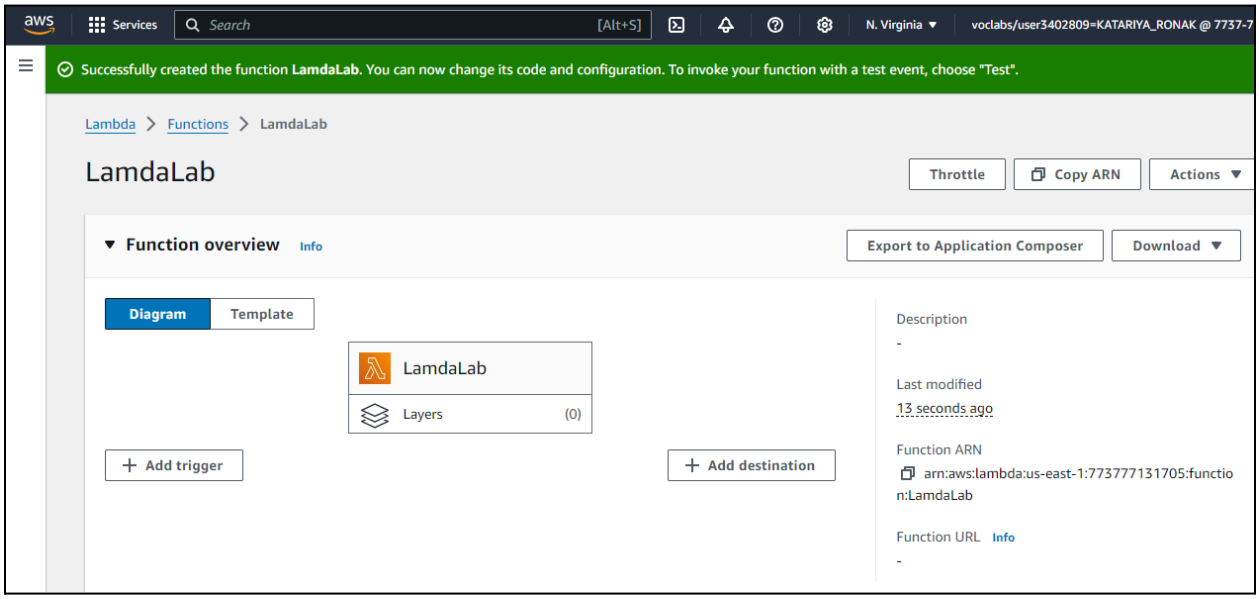


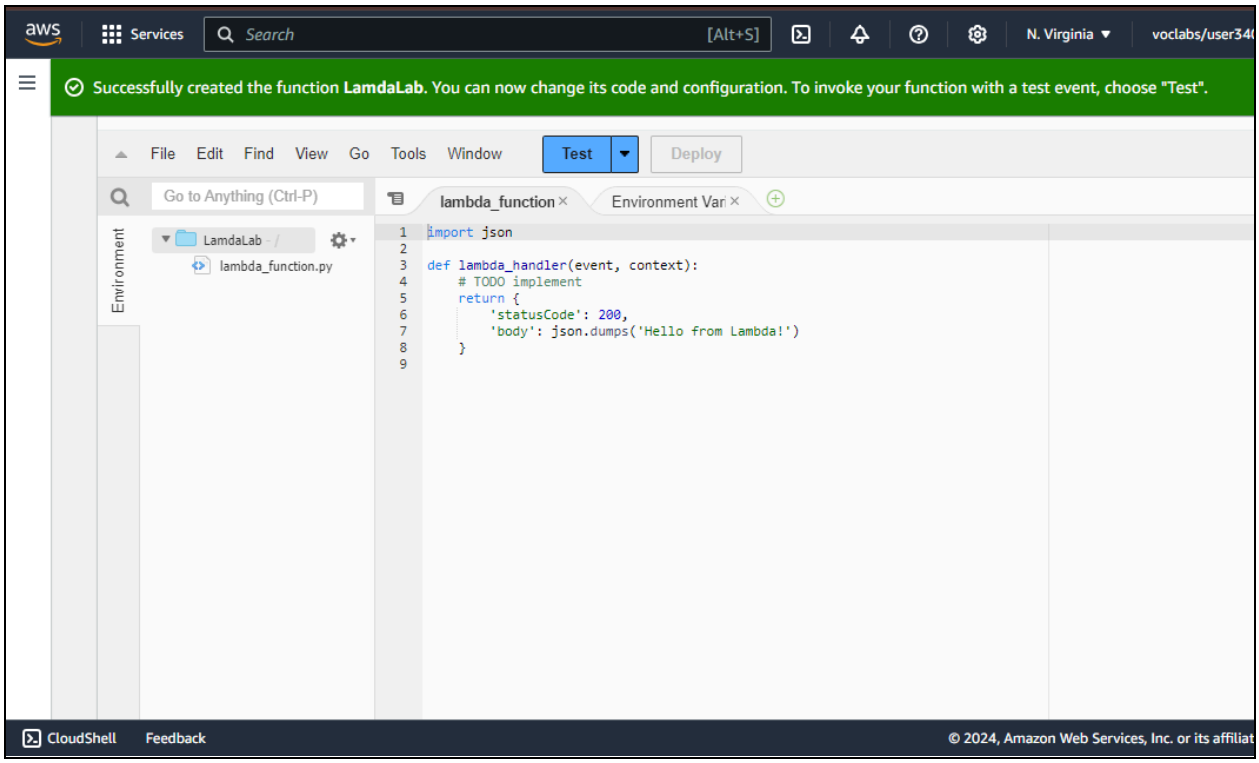
2. This process will take a while to finish and after that, you'll get a message that your function was successfully created

Edit Basic Settings:

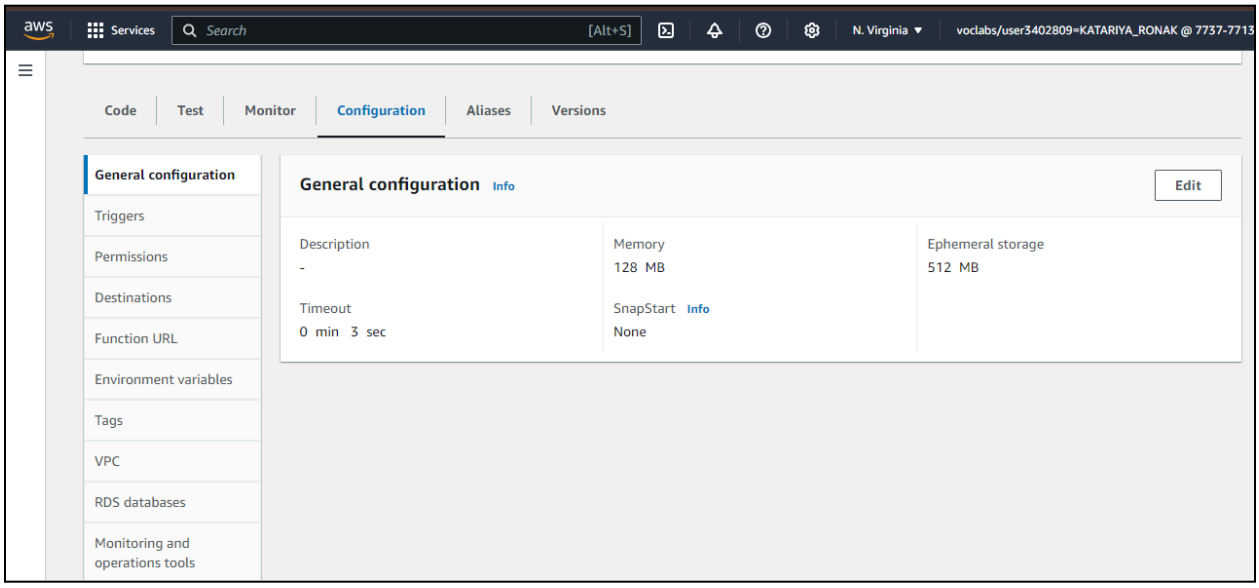
● On the function's Configuration tab, locate the Basic settings section

Configuring test event which tiggers when the function is tested

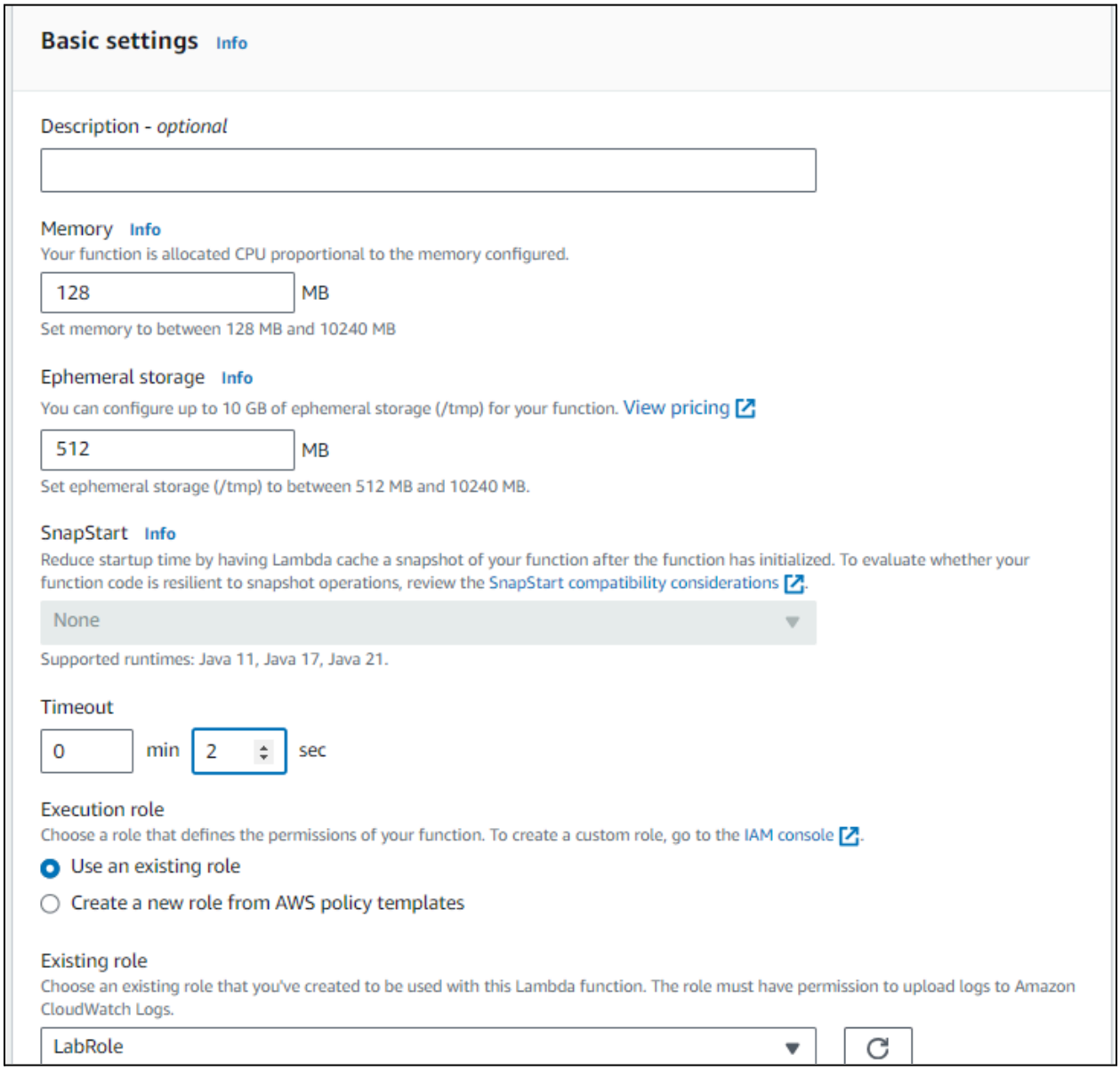




Before changing the configuration settings:



After:



Conclusion:

AWS Lambda is a serverless computing service that allows you to run code without managing servers, making it highly scalable, cost-effective, and easy to use. It automatically manages the compute resources, executes your code in response to specific events such as API calls, file uploads, or database updates, and scales based on the demand.

Experiment 12

Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Theory:

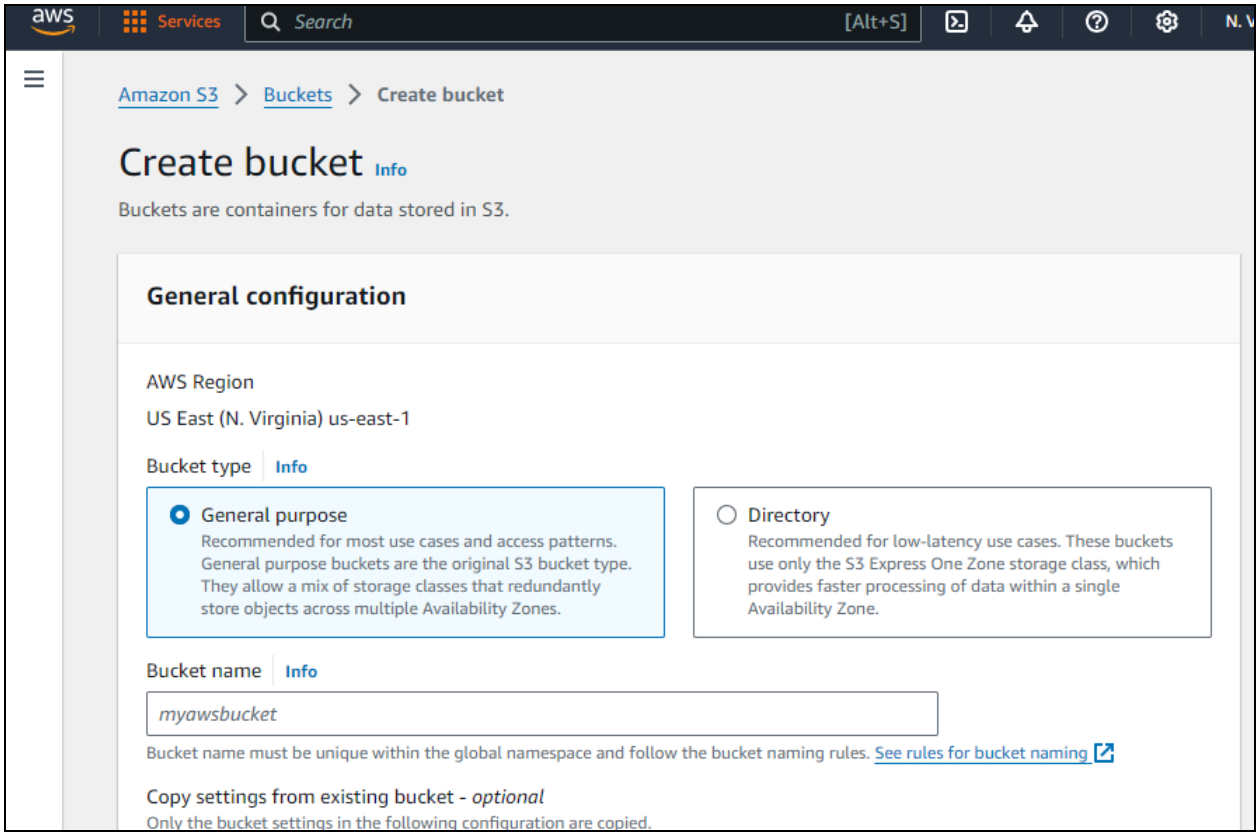
Creating a system to log activities when an image is added to an S3 bucket involves integrating Amazon S3 with AWS Lambda.

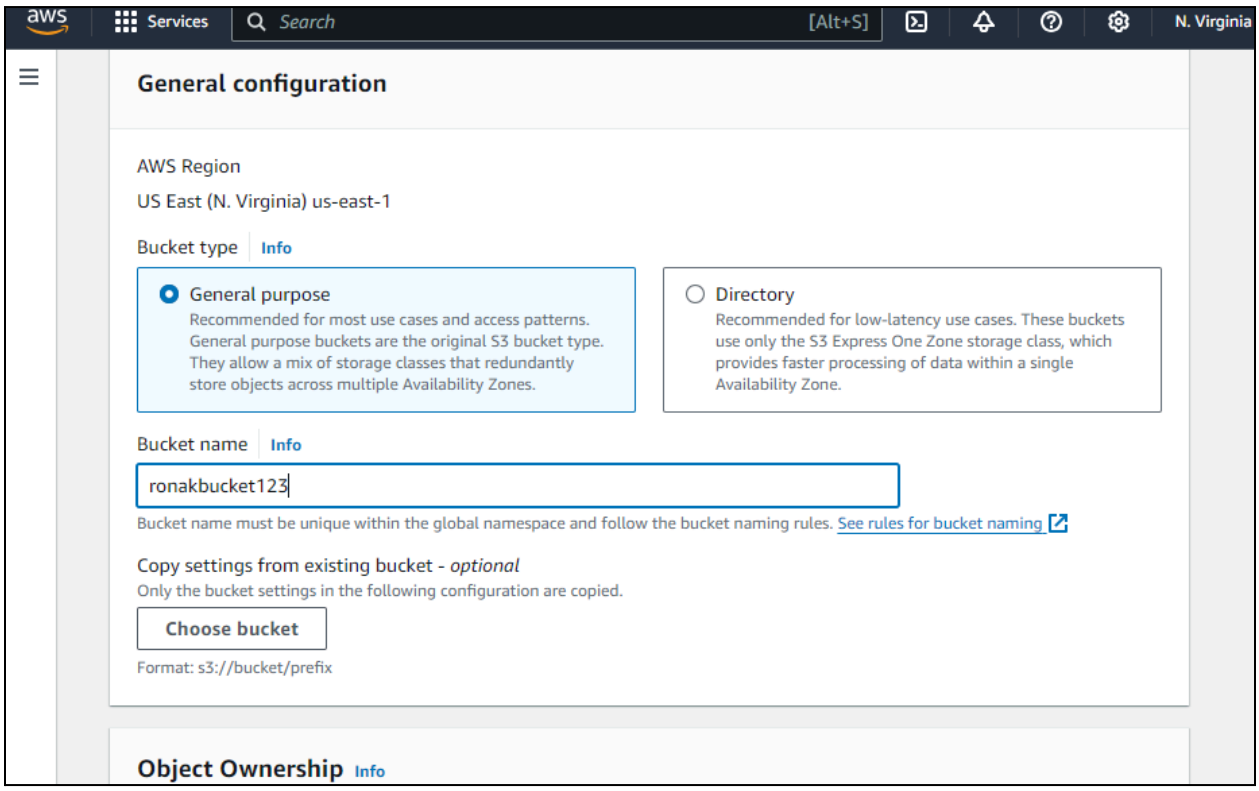
Amazon S3:
Amazon S3 (Simple Storage Service) is a service offered by AWS that provides object storage through a web service interface. It's designed to store and retrieve any amount of data from anywhere. You can use S3 to store images, videos, backups, data logs, and more.

AWS Lambda:
AWS Lambda is a serverless compute service that allows you to run code in response to events without provisioning or managing servers. You write your code and set up a trigger, and Lambda takes care of the rest. This means that when a specified event occurs, such as an object being added to an S3 bucket, the Lambda function is automatically invoked.
By setting up a Lambda function to trigger on new uploads to a specific S3 bucket, you can automate logging activities. This function will capture the event, process it, and log the message "An Image has been added." It ensures that every new upload is tracked efficiently and that you have a record of these actions.
This kind of setup is highly scalable, reliable, and costeffective, leveraging AWS's robust infrastructure. It's particularly useful for applications that require automated monitoring and logging of uploads for auditing or notification purposes.

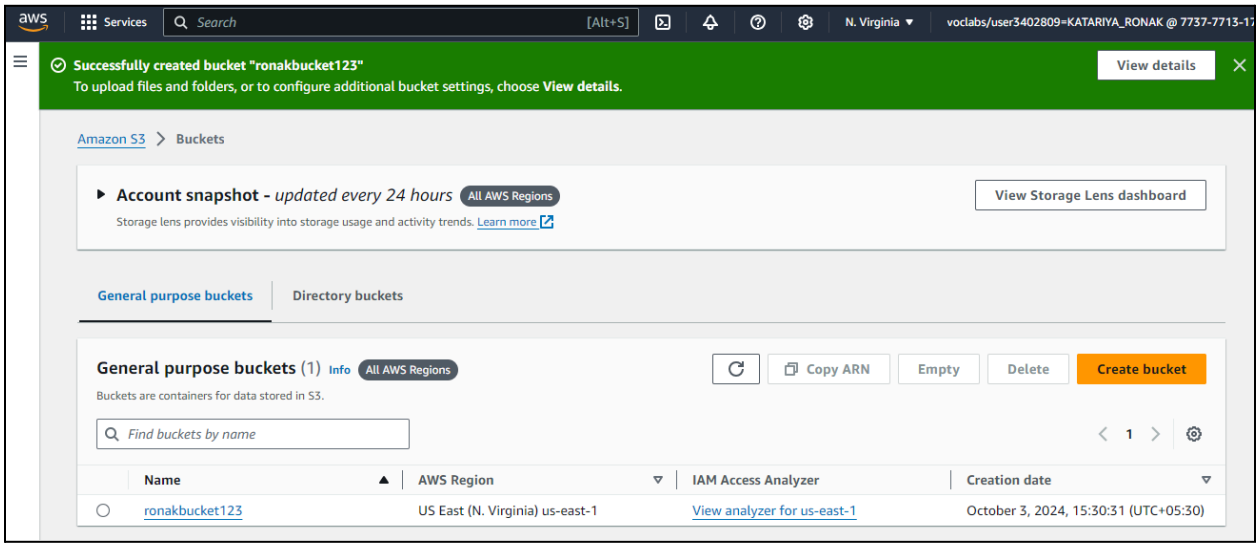
Implementation:

1. Create an S3 Bucket: First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

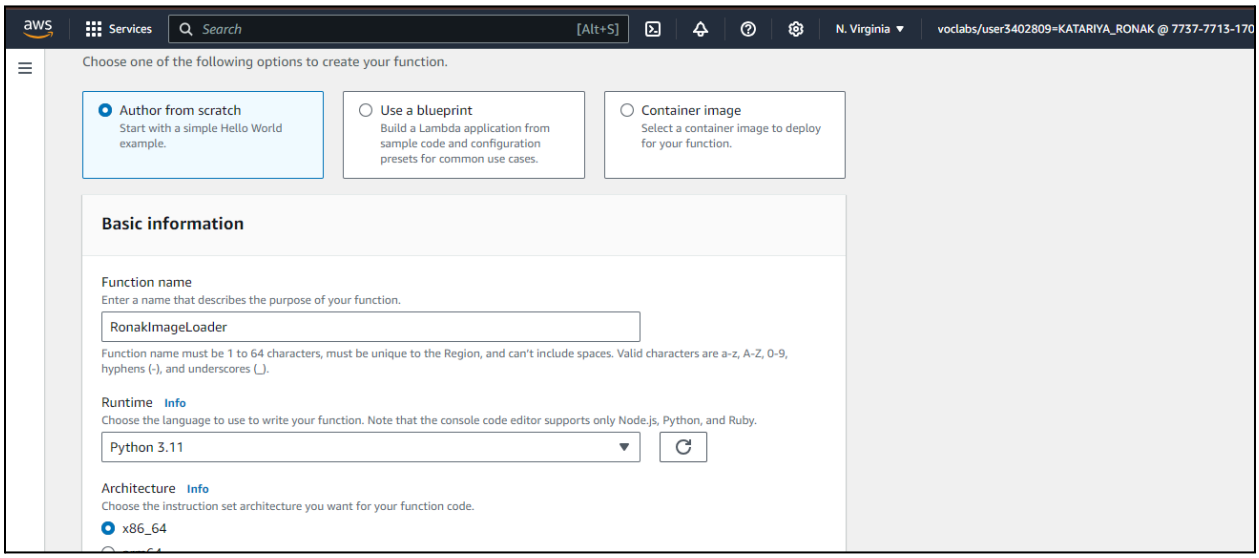

aws
Services
Search
[Alt+S]
N. V
Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
US East (N. Virginia) us-east-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
myawsbucket
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.


aws
Services
Search
[Alt+S]
N. Virginia
General configuration
AWS Region
US East (N. Virginia) us-east-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
ronakbucket123
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix
Object Ownership Info

Bucket created successfully:



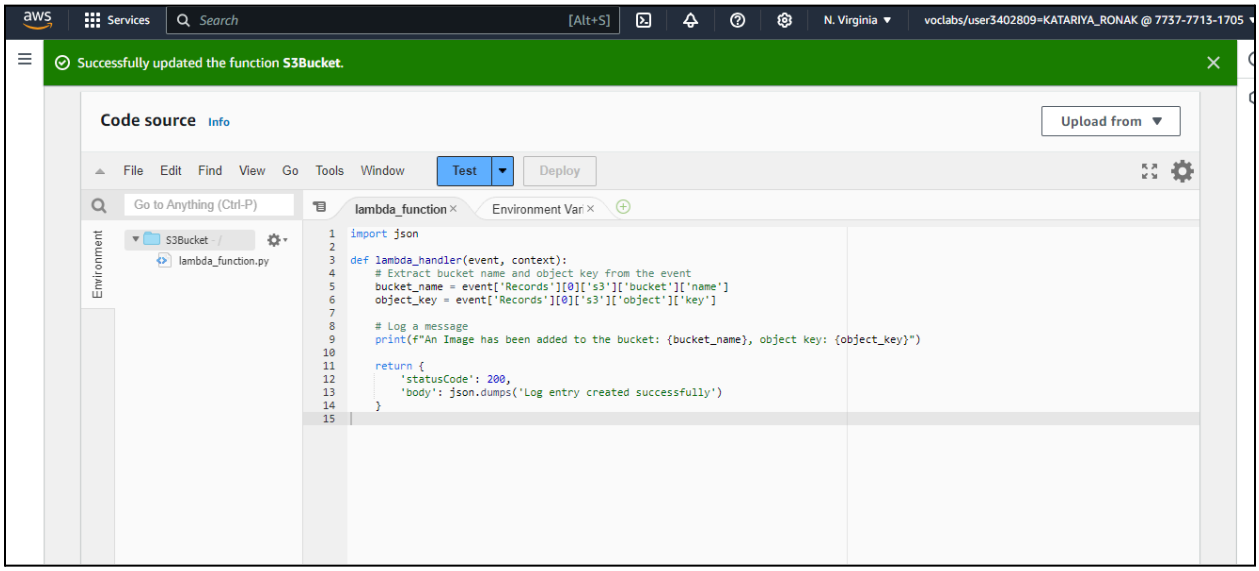
2. Create the Lambda Function: Set up a new Lambda function using AWS Lambda's console. You can choose a runtime environment like Python, Node.js, or Java.

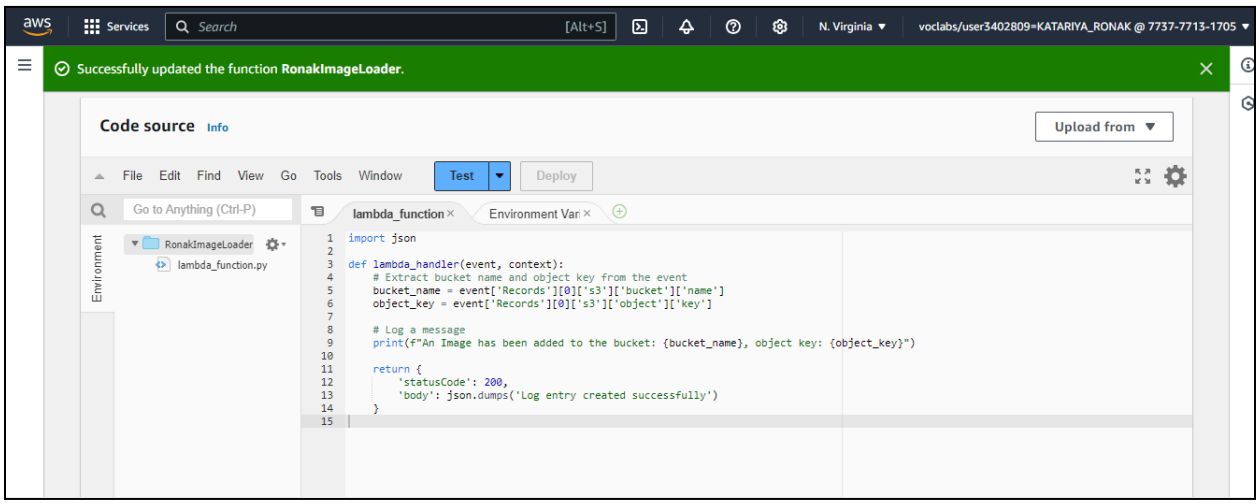


Write code that logs a
message like "An Image has been added" when triggered
This is the code we have to add:

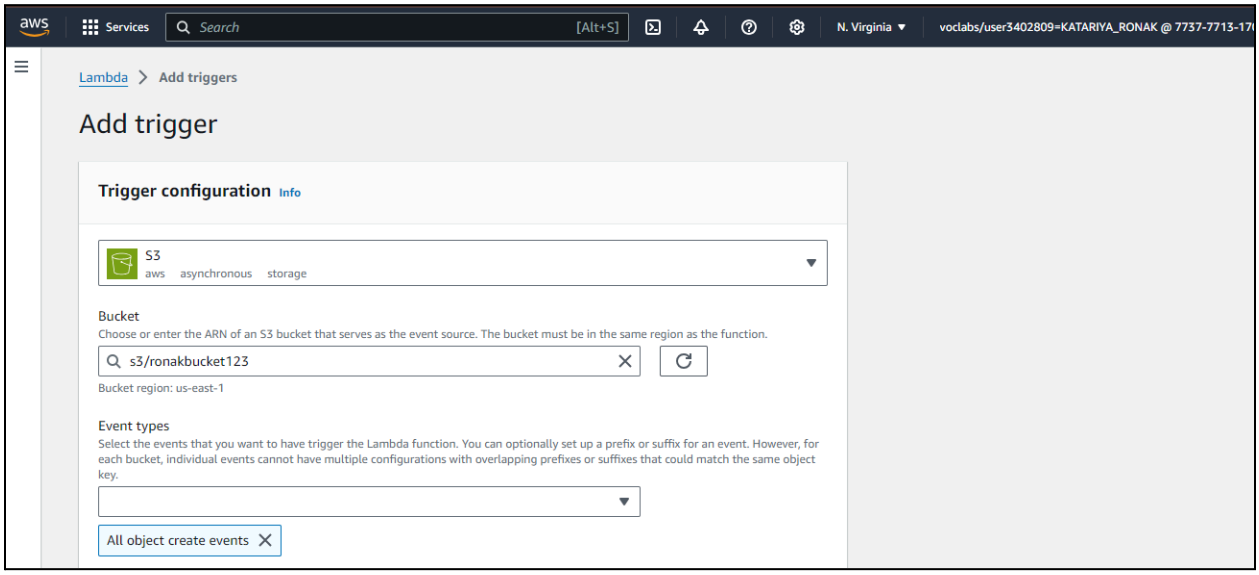
```
import json
import
logging
# Set up logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
```

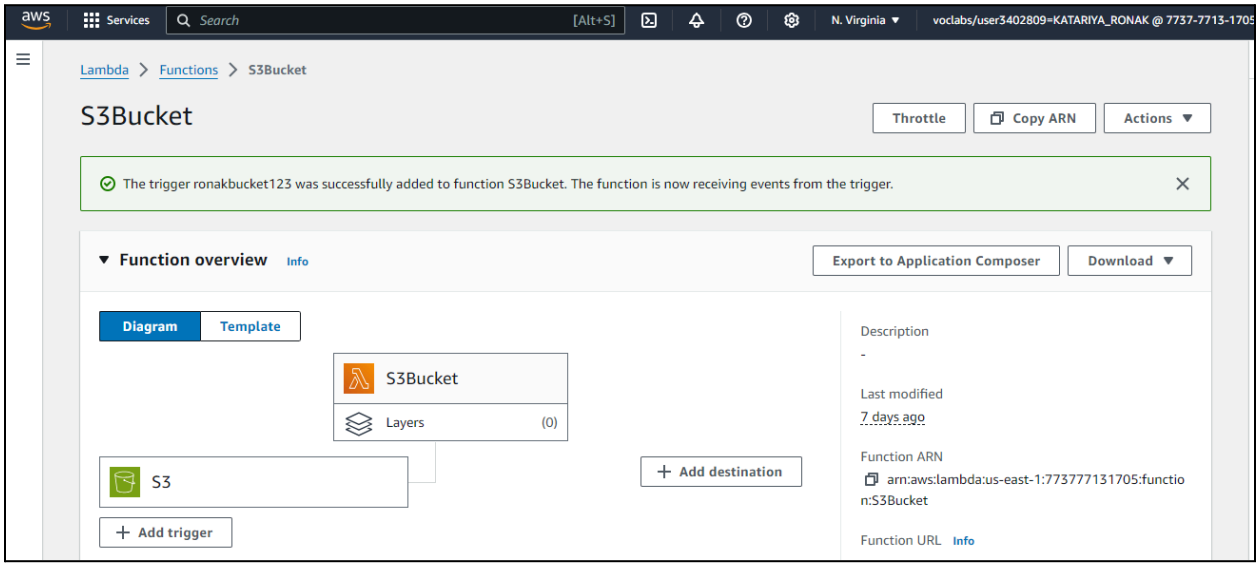
```
# Extract bucket name and object key from the S3
event for record in event['Records']:
bucket = record['s3']['bucket']['name']
key = record['s3']['object']['key']
# Log a message
logger.info(f"An image has been added to bucket {bucket}, object key: {key}")
# Check if the uploaded file is an image (you can adjust the file types
here) if key.lower().endswith(('.png', '.jpg', '.jpeg', '.gif')):
logger.info("Image Uploaded
successfully") else:
logger.info("A non-image file has been added")
return {
'statusCode': 200,
'body': json.dumps('Event processed')
}
```

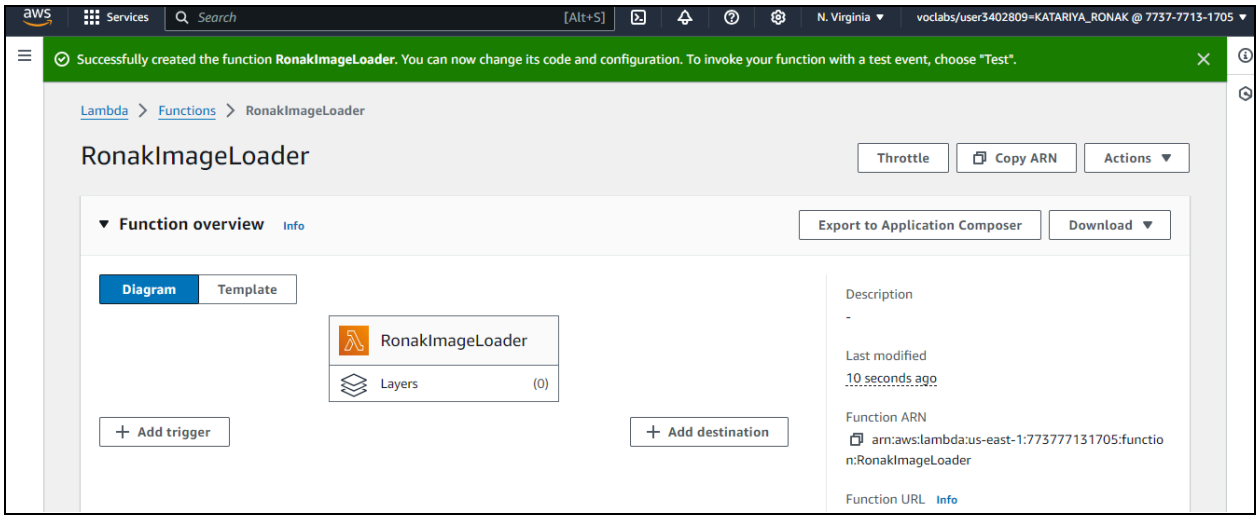




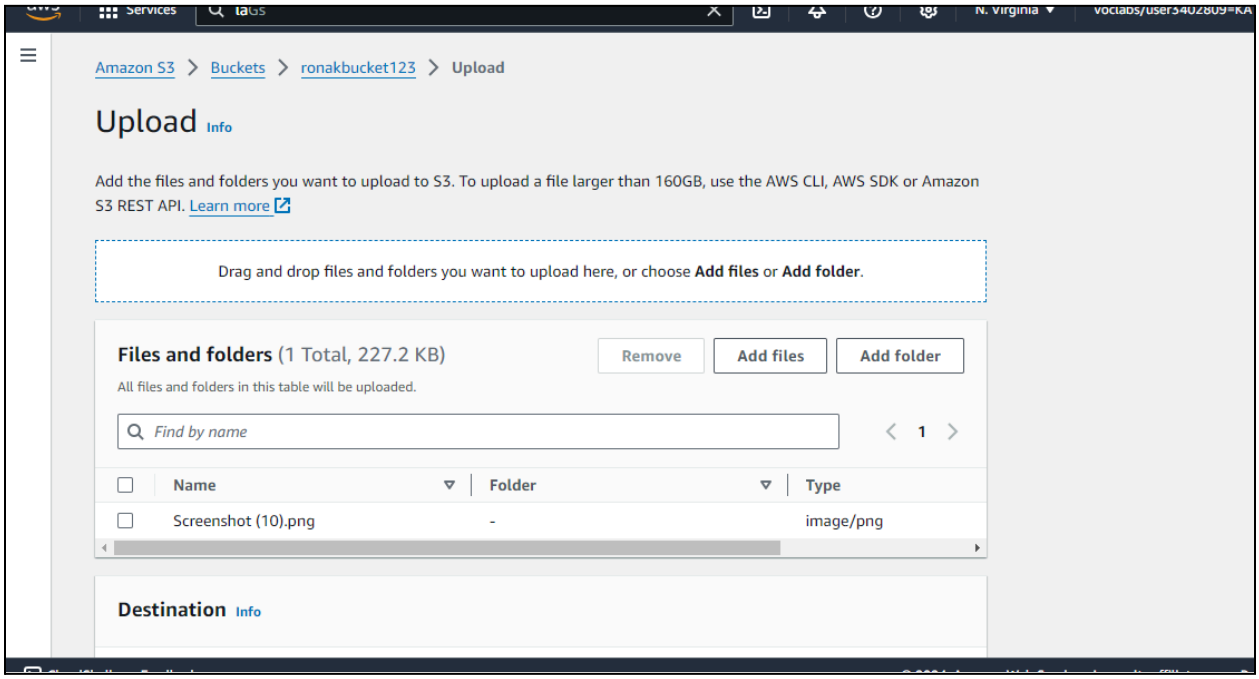
Configure S3 Trigger: Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

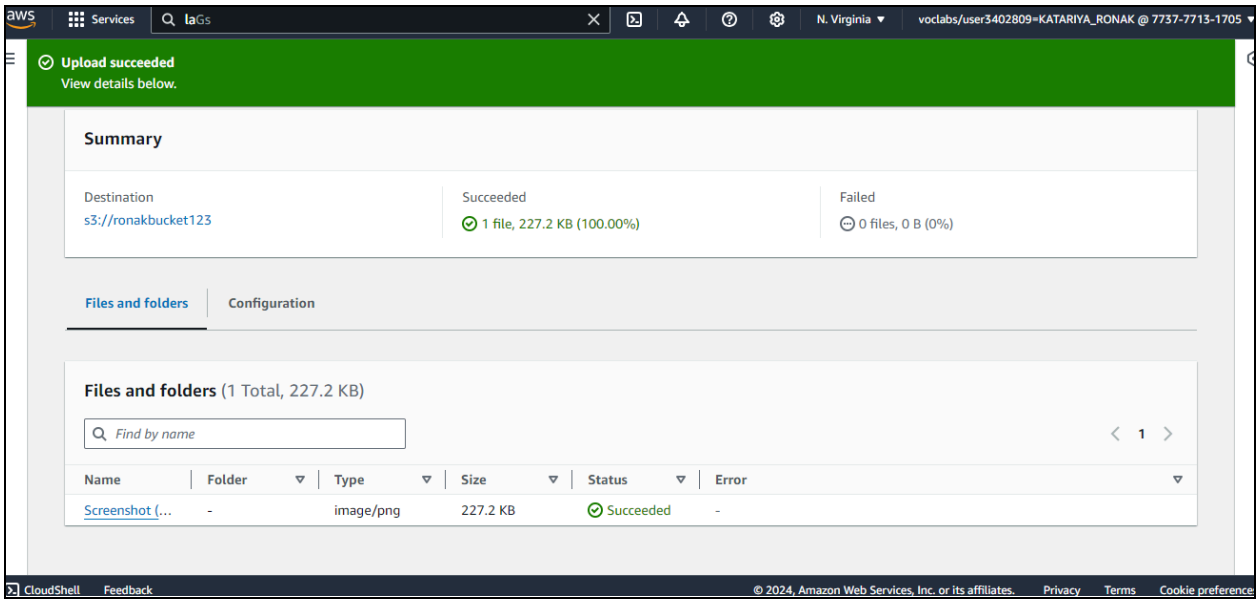




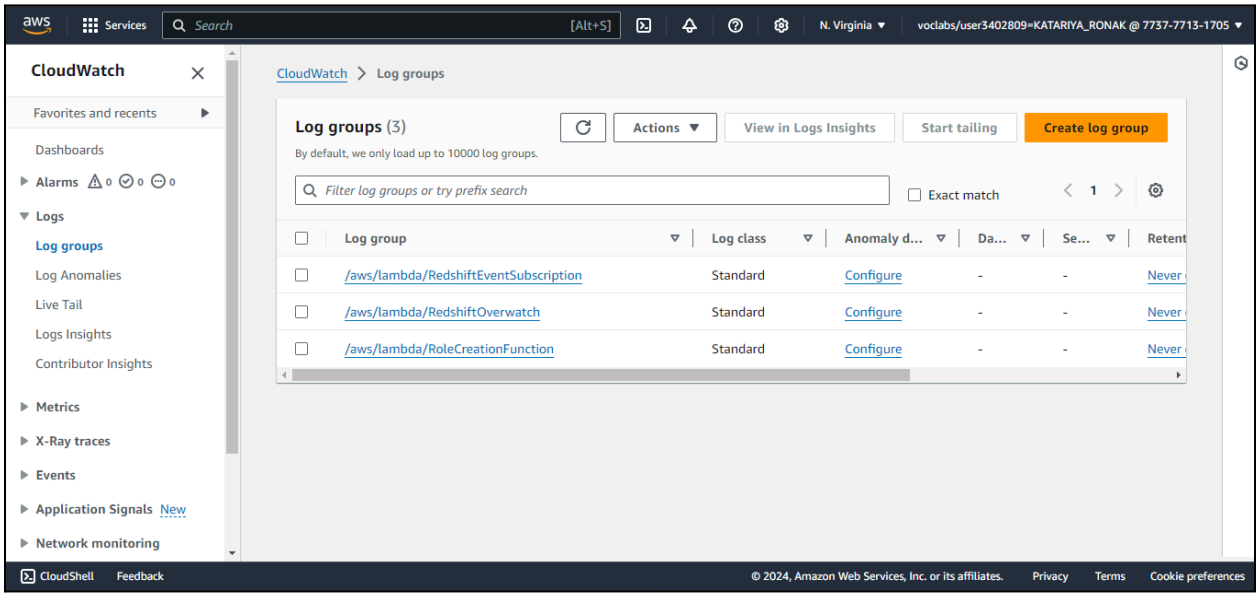


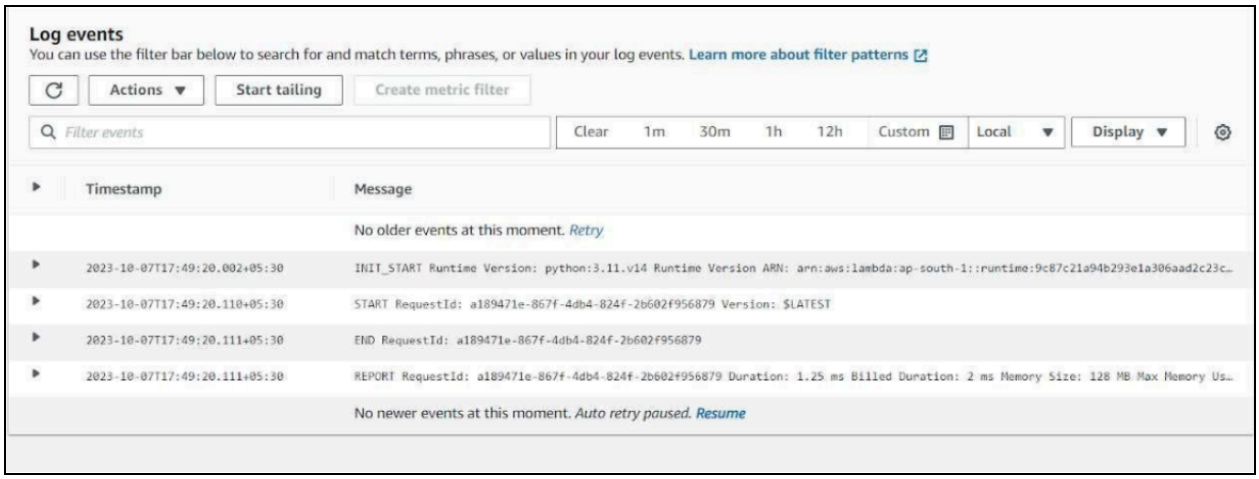
Upload an object (e.g., an image) to the S3 bucket to test the trigger





Test the Setup: Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs





Conclusion:

Creating a Lambda function to log “An Image has been added” upon S3 uploads showcases AWS's serverless capabilities. This integration automates event handling, enhances operational efficiency, and reduces manual oversight, allowing developers to focus on application logic while ensuring responsive and scalable cloud applications.