EXPERIMENT NO. 1

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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

Theory :

EC2 Hosting

Amazon Elastic Compute Cloud (EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers by providing virtual servers, known as instances, to run applications.

Key Concepts of EC2:

1. **Instances:**
   An instance is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications. Instances can be launched on demand and scaled according to the needs of the application.
2. **AMI (Amazon Machine Image):**
   AMIs are pre-configured templates for instances. They include the operating system, application server, and applications themselves, allowing for quick and consistent instance launches.
3. **Instance Types:**
   EC2 offers a variety of instance types optimized for different use cases, such as compute-optimized, memory-optimized, and storage-optimized instances. Each instance type offers different combinations of CPU, memory, storage, and networking capacity.
4. **Elastic IP Addresses:**
   Elastic IPs are static IP addresses that can be associated with an EC2 instance. They are particularly useful for maintaining a consistent IP address even if the underlying instance changes.
5. **Security Groups:**
   Security groups act as virtual firewalls that control the traffic to and from an EC2 instance. You can configure rules to allow or deny traffic based on IP addresses, ports, and protocols.
6. **Auto Scaling:**
   Auto Scaling allows you to automatically adjust the number of EC2 instances in your application environment according to the current demand, ensuring optimal performance and cost-efficiency.

Amazon S3 (Simple Storage Service)

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance. S3 is used to store and retrieve any amount of data, at any time, from anywhere on the web.

Key Concepts of S3:

1. **Buckets:**
   Buckets are containers for storing objects in S3. Each object is stored in a bucket, and each bucket is unique across the AWS environment. Buckets are used to organize and manage the data stored in S3.
2. **Objects:**
   Objects are the fundamental entities stored in S3. Each object consists of data, metadata, and a unique identifier (key). Objects can be any type of data, including images, videos, documents, or binary files.
3. **Keys:**
   Keys are unique identifiers for objects within a bucket. Each object in S3 is assigned a unique key that can be used to access and manage the object.
4. **Versioning:**
   S3 allows you to maintain multiple versions of an object within a bucket. Versioning helps protect against accidental overwrites and deletions by preserving older versions of objects.
5. **Access Control:**
   S3 provides several mechanisms for controlling access to data, including bucket policies, access control lists (ACLs), and IAM (Identity and Access Management) policies. These controls help ensure that only authorized users can access and manage the data.
6. **Lifecycle Management:**
   Lifecycle policies in S3 allow you to automate the transition of objects between different storage classes or delete them after a specified period. This helps optimize storage costs by moving data to less expensive storage as it ages.
7. **Storage Classes:**
   S3 offers various storage classes designed for different use cases, such as S3 Standard for frequently accessed data, S3 Intelligent-Tiering for automatically optimizing costs, and S3 Glacier for long-term archival storage.

AWS Cloud9 Infrastructure

AWS Cloud9 is a cloud-based integrated development environment (IDE) that allows you to write, run, and debug code with just a browser. It supports multiple programming languages, including Python, JavaScript, and more. AWS Cloud9 comes pre-packaged with essential tools and libraries, making it easier for developers to start coding without the need for complex setup processes.

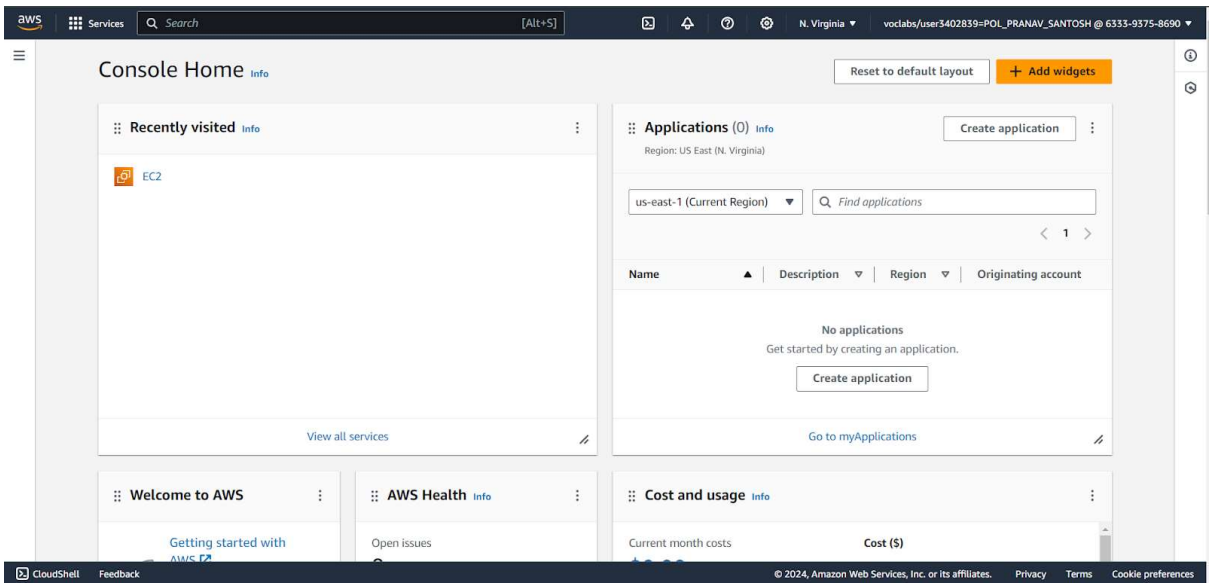
Key Features of AWS Cloud9:

1. **Cloud-Based IDE:**
   AWS Cloud9 provides a full-featured development environment accessible through a web browser. This eliminates the need for local IDE installations and configurations.
2. **Collaborative Development:**
   Developers can collaborate in real-time, with multiple users able to work on the same project simultaneously. It includes features like chat and simultaneous editing, making it ideal for pair programming and team collaborations.
3. **Pre-configured Environment:**
   Cloud9 is pre-configured with essential tools and libraries for various languages and frameworks. This allows developers to start coding immediately without worrying about environment setup.
4. **Seamless Integration with AWS Services:**
   AWS Cloud9 integrates seamlessly with other AWS services like EC2, S3, and Lambda, allowing developers to easily deploy and manage their applications directly from the IDE.
5. **Terminal Access:**
   Cloud9 provides full terminal access to the underlying instance, giving developers the ability to run shell commands and manage their environment directly.

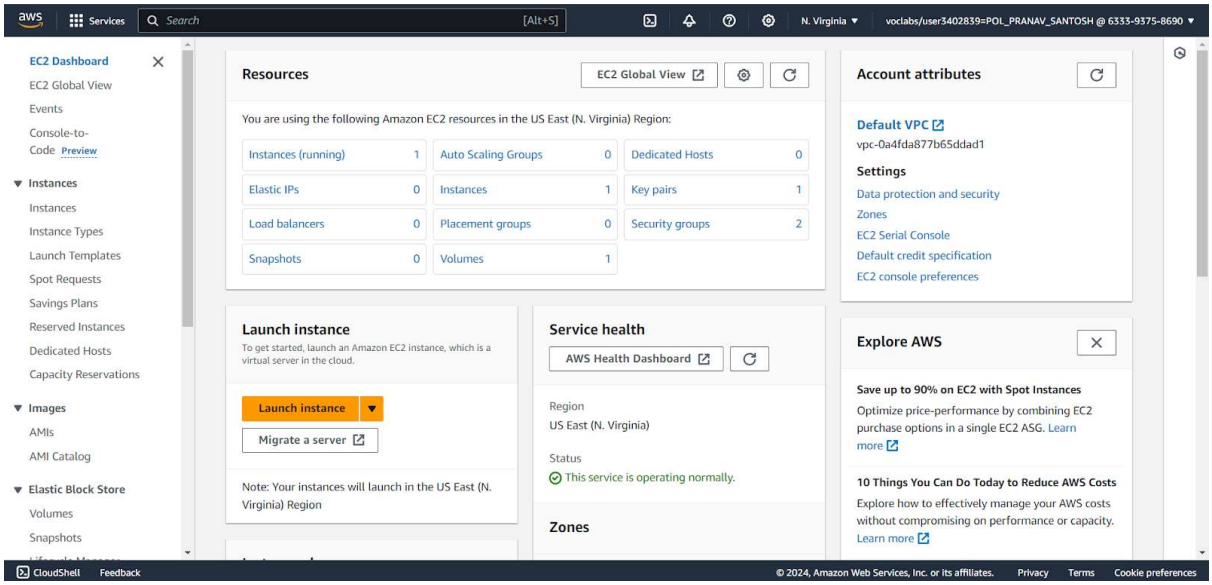
Implementation :

EC2 Instance Creation and static site hosting

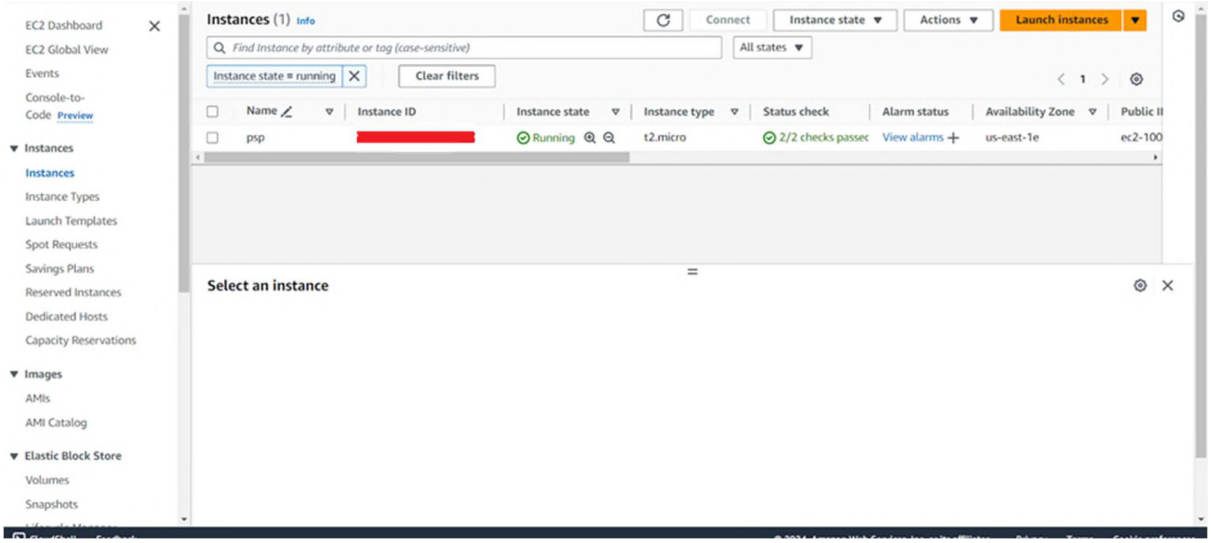
1) Login to your AWS account



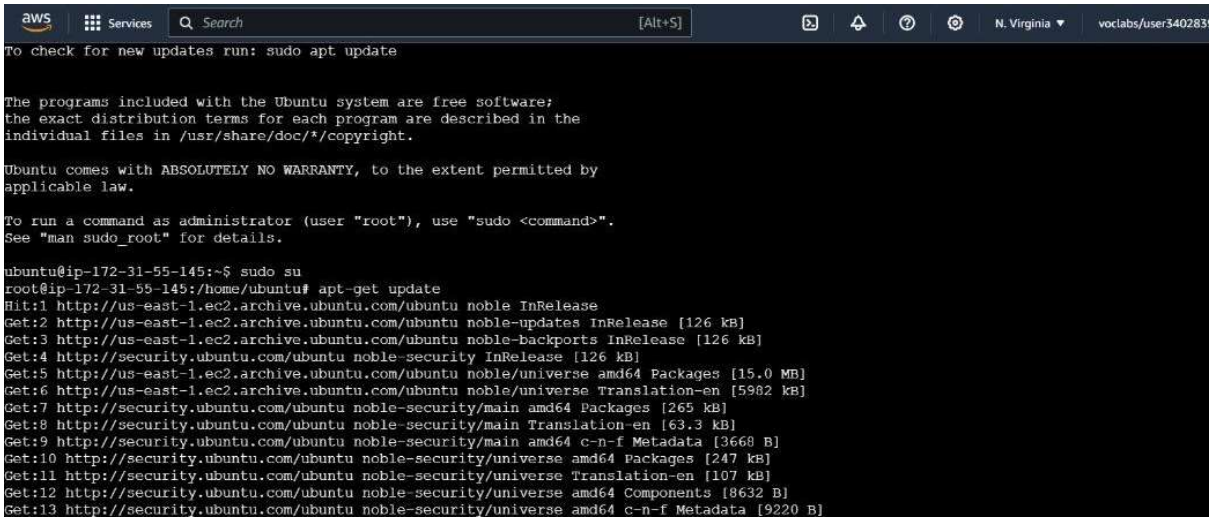
2) Click on EC2 and then create an instance by clicking on instances



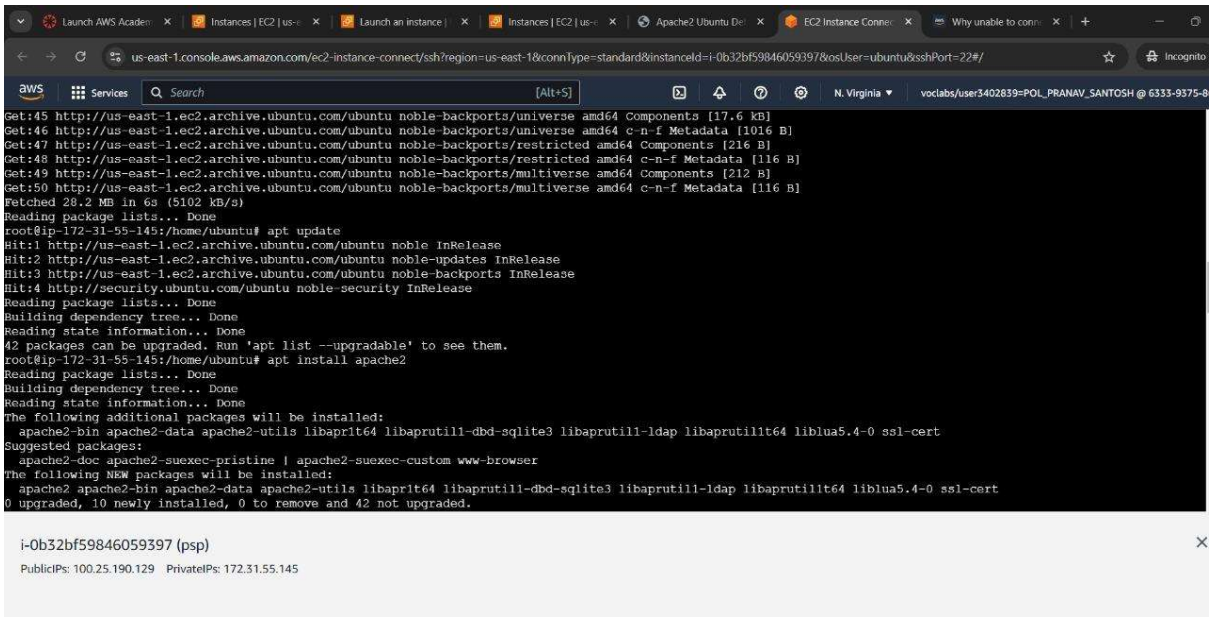
3) After an instance is created wait for it to come to Running state

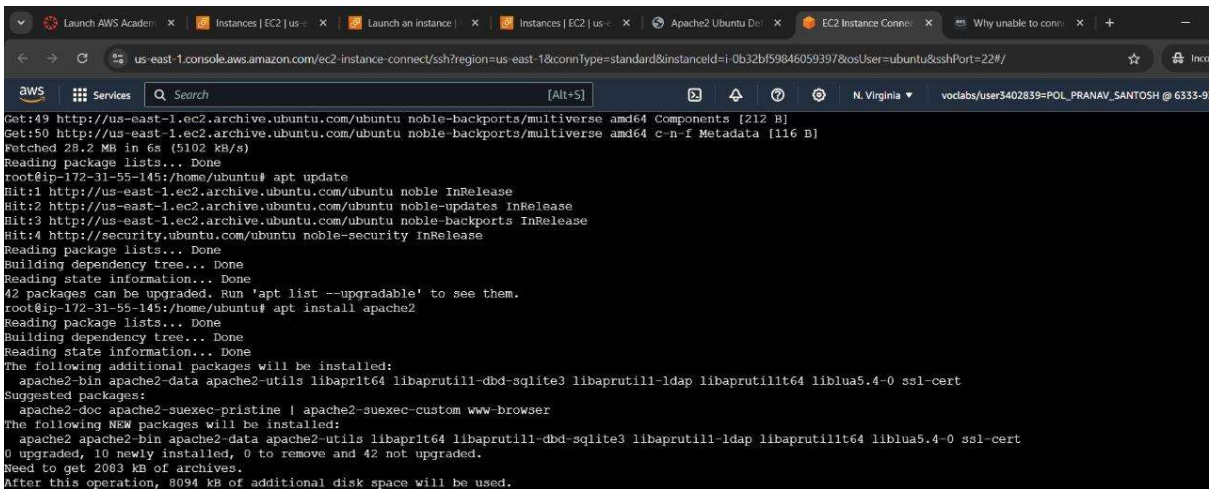


4) After doing that you will see this UI

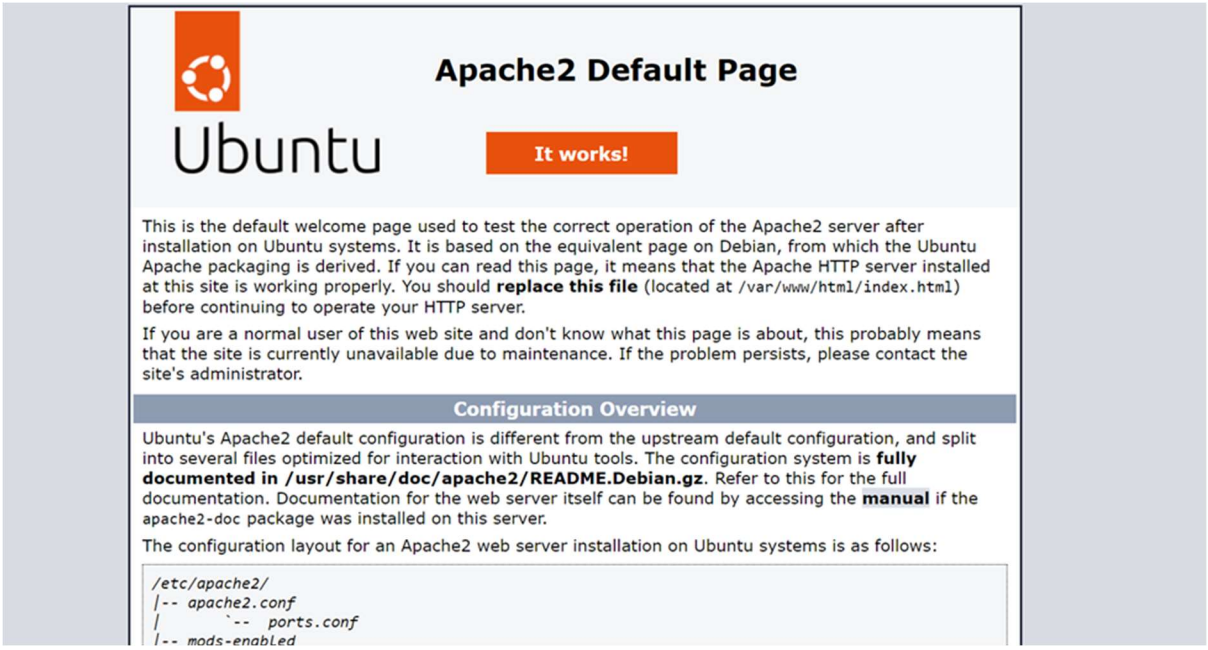


5) Follow these steps and then run these commands

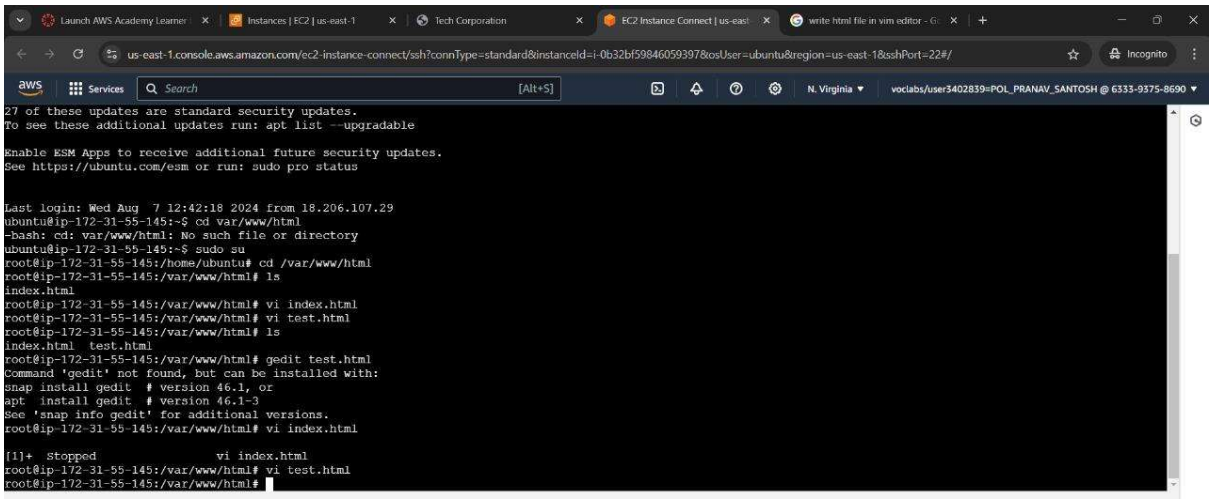




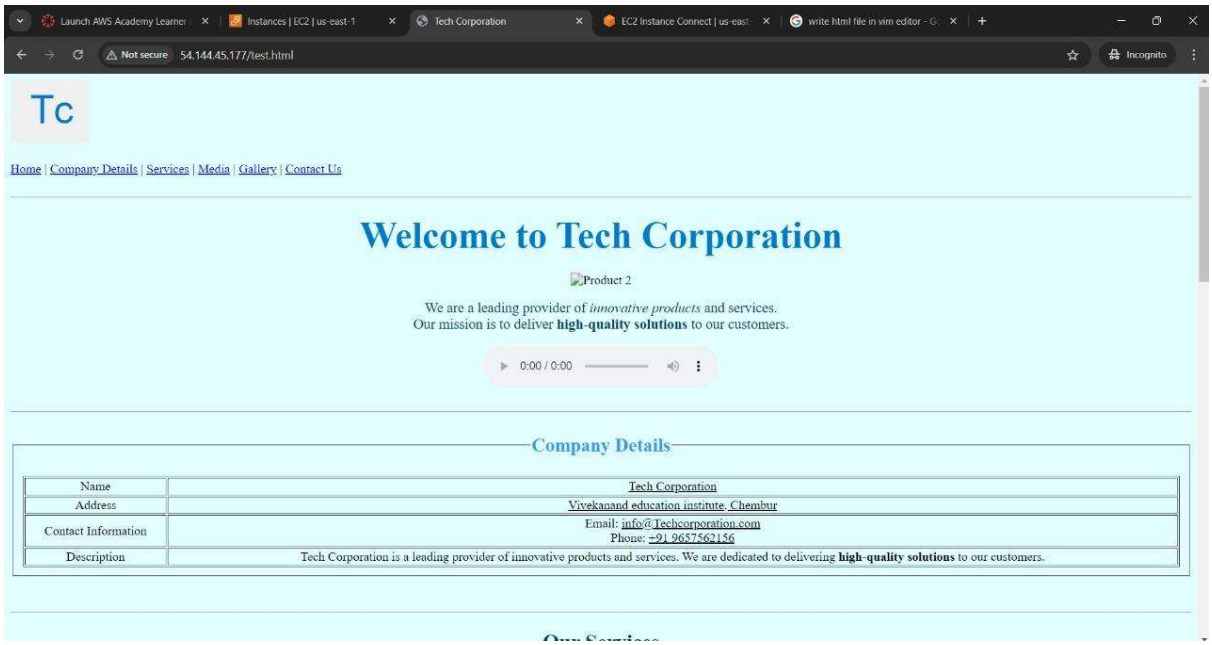
6) After that the ip-address which was given while running the instance, copy that and paste that on chrome, make sure that it is http and not https



7) Create a file using vi command and save it using :wq

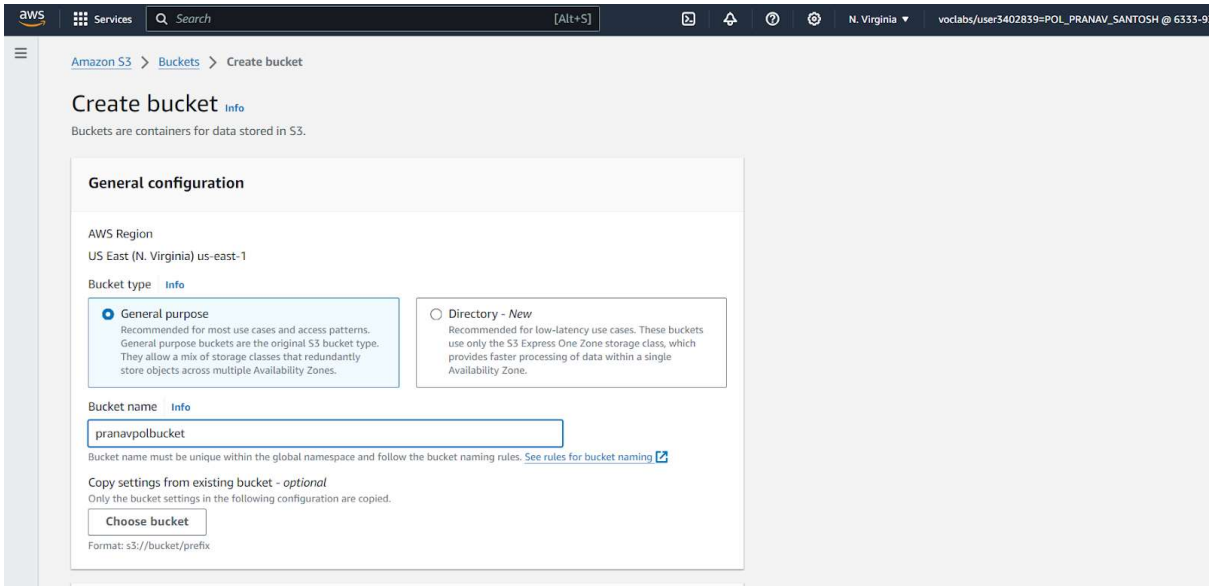


8) After saving that file go that page where ubuntu is listed and then on the link add “/your_file_name.html” and then whatever you saved on that file will be displayed

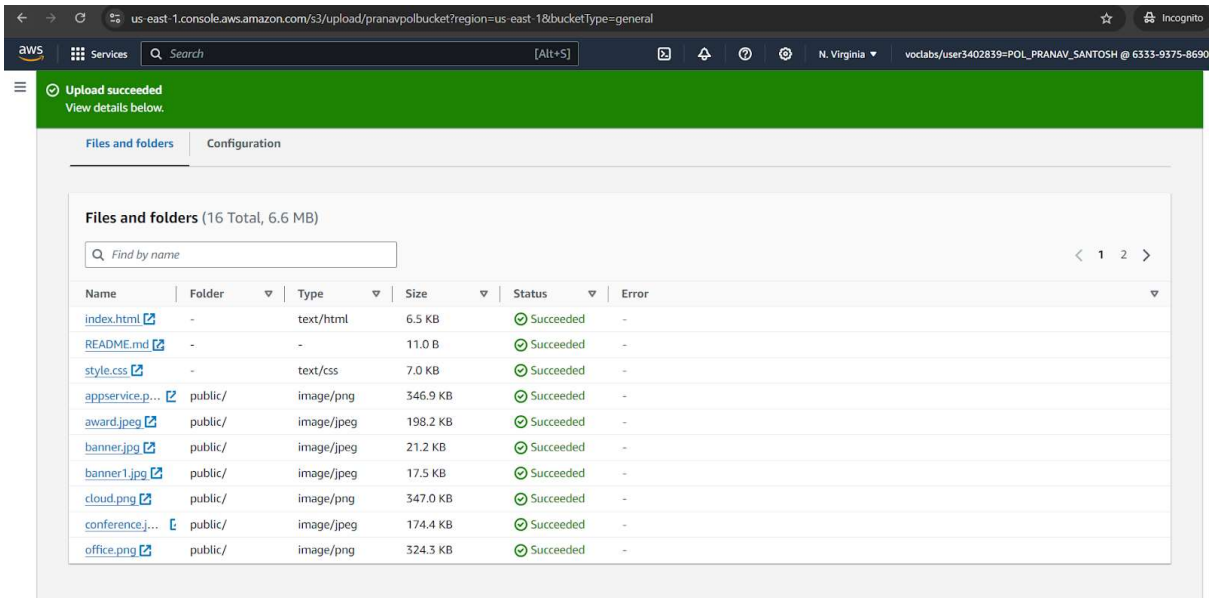


Static Site Hosting using S3 bucket

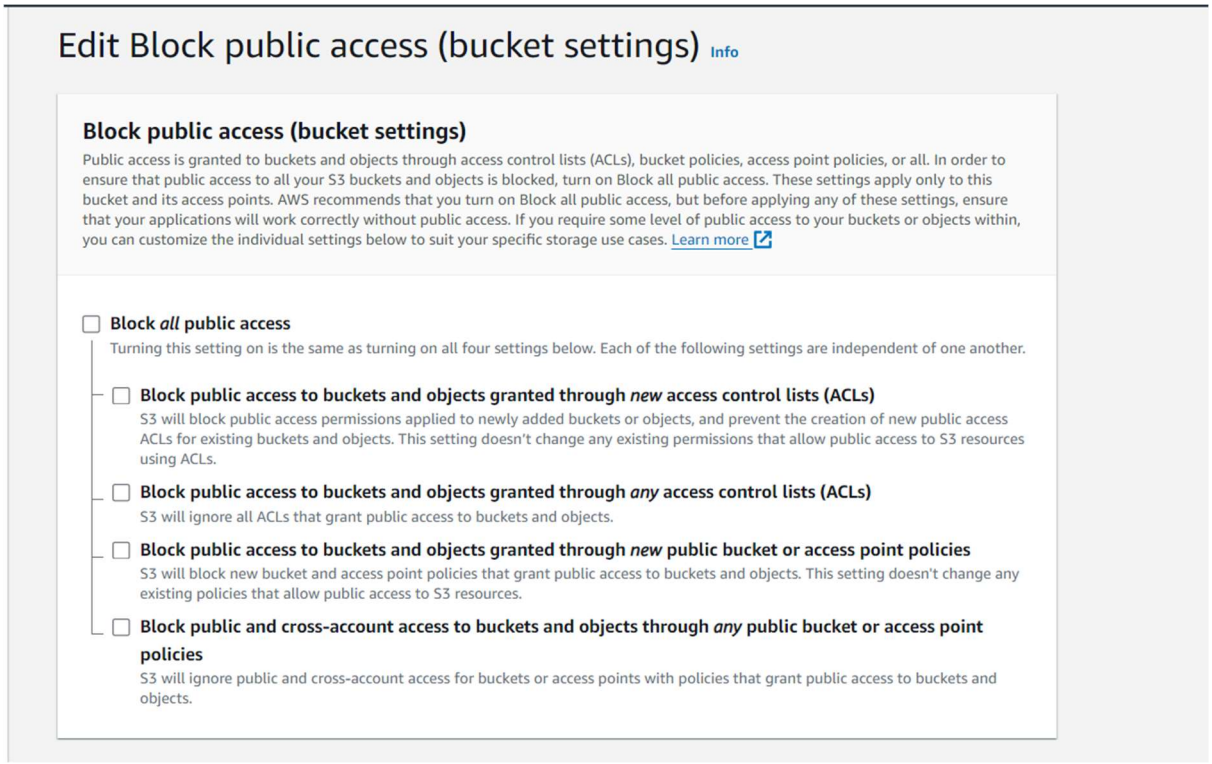
Step1: Create bucket

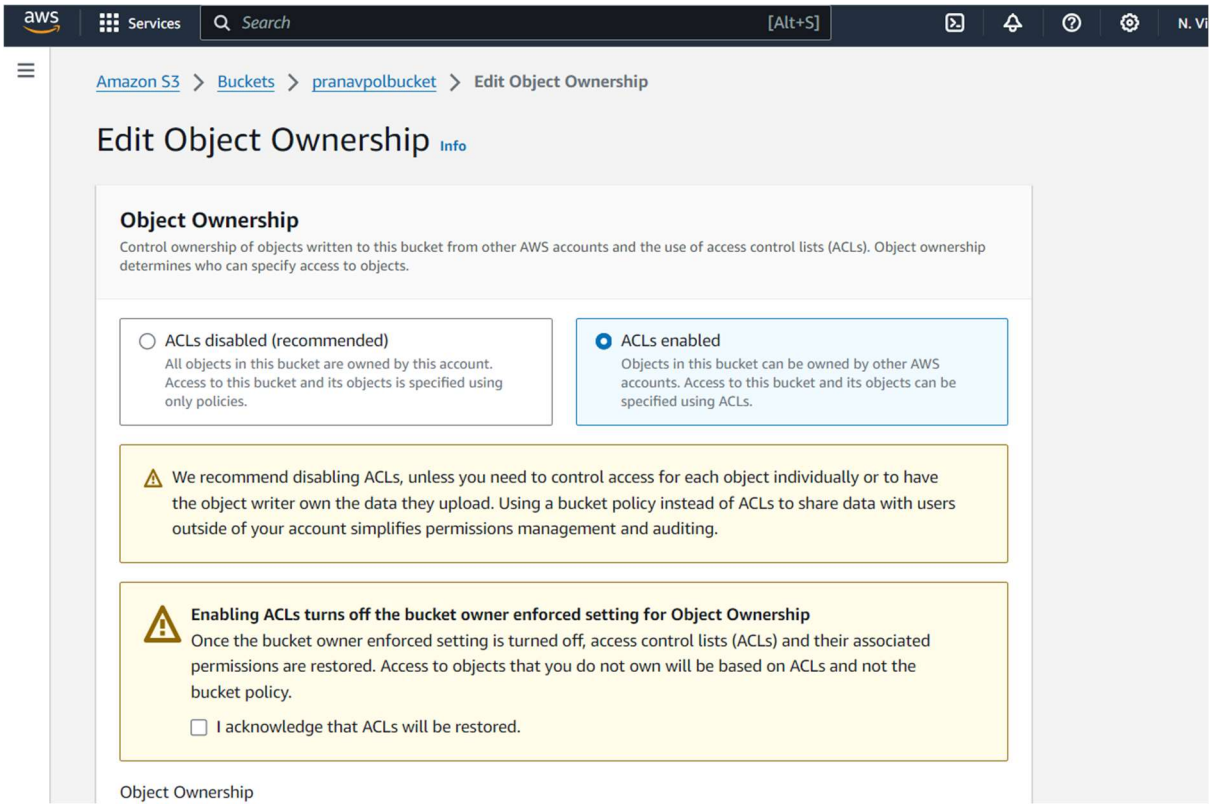


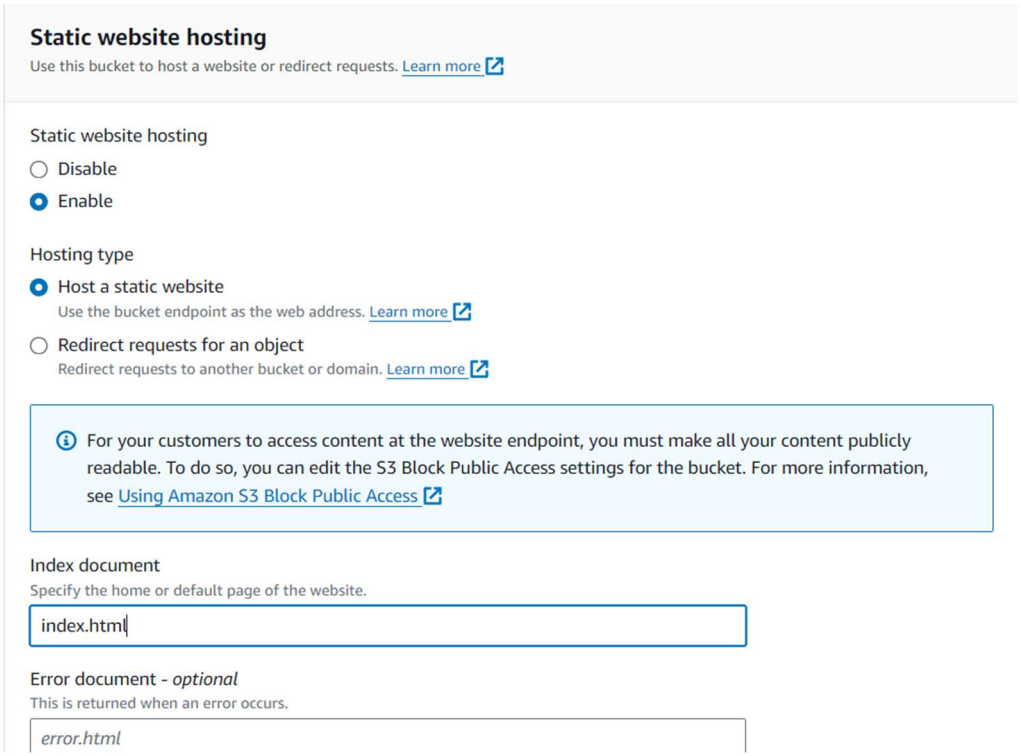
Step 2: Add resources

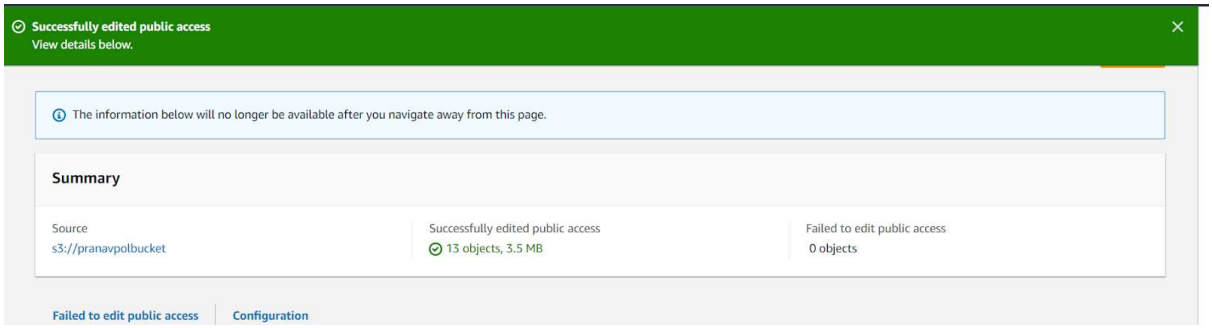


Step 3 : Provide public access

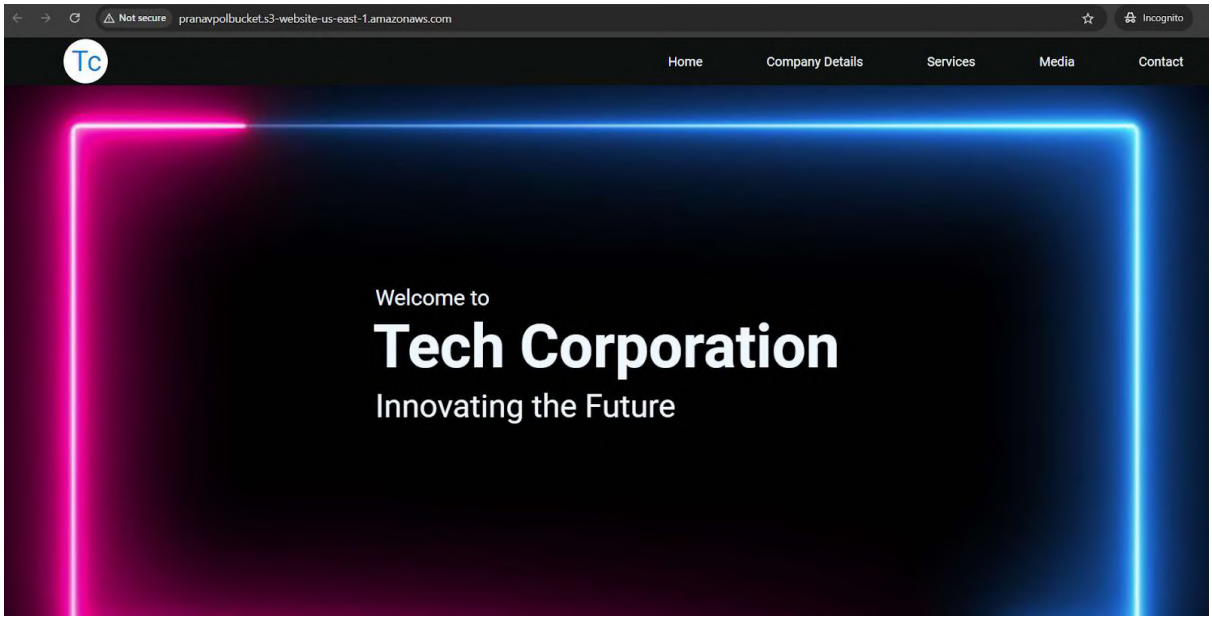



aws
Services
Search
[Alt+S]
N. Vi
Amazon S3 > Buckets > pranavpolbucket > Edit Object Ownership
Edit Object Ownership Info
Object Ownership
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.
We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.
Enabling ACLs turns off the bucket owner enforced setting for Object Ownership
Once the bucket owner enforced setting is turned off, access control lists (ACLs) and their associated permissions are restored. Access to objects that you do not own will be based on ACLs and not the bucket policy.
I acknowledge that ACLs will be restored.
Object Ownership


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.
error.html



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-property

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
```

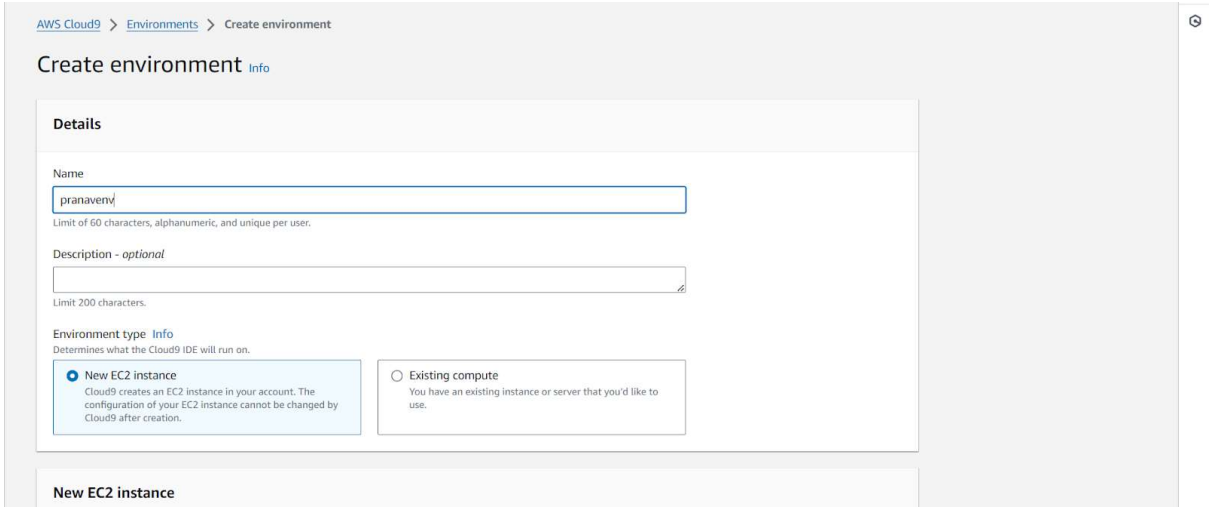
Step 2 : Install necessary Packages and run website on port 3000

Hey this is Dynamic Website.

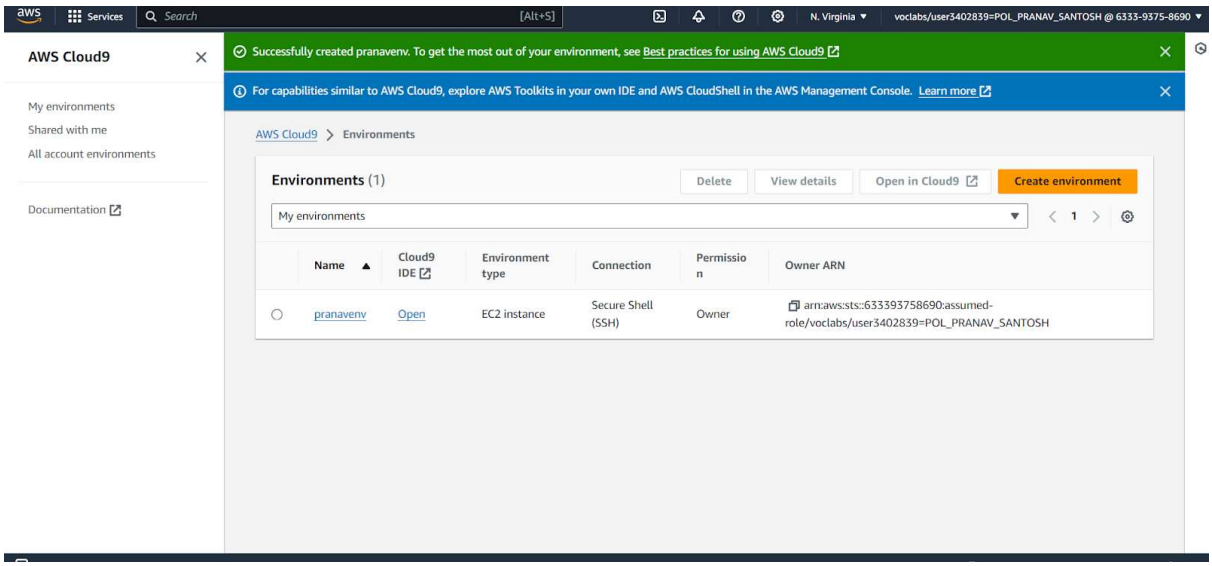
Hey this is about page.

Cloud 9 IDE Site Hosting

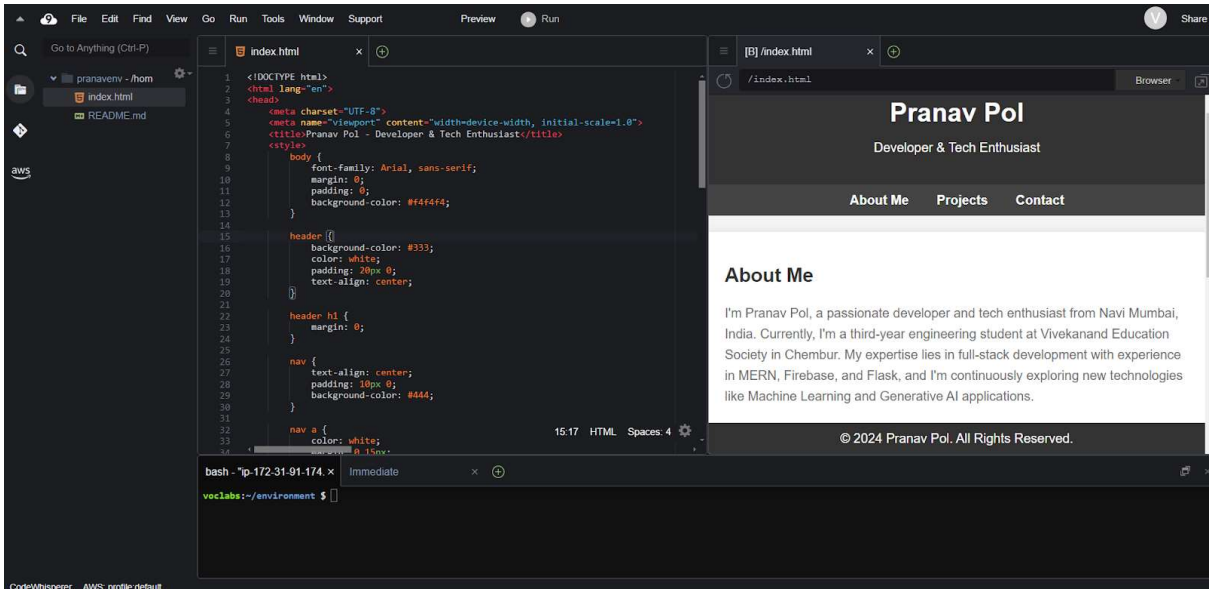
Step 1: Create Environment



Step 2 :Open the Environment IDE



Step 3: Add the code and preview the website



EXPERIMENT NO. 2

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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 :

AWS Elastic Beanstalk

AWS Elastic Beanstalk is a Platform as a Service (PaaS) offering from Amazon Web Services (AWS) that allows developers to deploy and manage applications in the AWS Cloud without needing to manage the underlying infrastructure. It automates the deployment process, including provisioning resources like EC2 instances, load balancers, and databases, making it easier to manage web applications and services.

Key Features of Elastic Beanstalk:

1. **Ease of Use:**
   Elastic Beanstalk simplifies the deployment process by automatically handling the infrastructure setup, deployment, monitoring, and scaling of your application. Developers can focus on writing code without worrying about the underlying infrastructure.
2. **Support for Multiple Languages and Frameworks:**
   Elastic Beanstalk supports a wide range of programming languages and frameworks, including Java, .NET, Node.js, PHP, Python, Ruby, Go, and Docker.
3. **Automatic Scaling:**
   Elastic Beanstalk automatically scales your application up or down based on the demand. It adjusts the number of instances running your application to meet traffic demands, ensuring optimal performance and cost-efficiency.
4. **Health Monitoring:**
   Elastic Beanstalk monitors the health of your applications and provides detailed logs and metrics. It automatically replaces any failed instances to maintain application availability.
5. **Environment Management:**
   Elastic Beanstalk allows you to manage multiple environments (such as development, testing, and production) for your application. You can easily deploy updates to specific environments without affecting others.

AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages ready for deployment. It allows you to automate the build process, ensuring that your code is compiled and tested consistently across all development environments.

Key Features of AWS CodeBuild:

1. **Fully Managed Build Service:**
   CodeBuild eliminates the need to set up, patch, update, and manage your own build servers. AWS manages the infrastructure, allowing you to focus on developing and testing your code.
2. **Scalability:**
   CodeBuild scales automatically to handle multiple builds concurrently, ensuring that your builds are processed quickly, even during peak times.
3. **Custom Build Environments:**
   CodeBuild allows you to define custom build environments using Docker images. This flexibility enables you to tailor the build environment to meet the specific needs of your application.
4. **Integration with Other AWS Services:**
   CodeBuild integrates seamlessly with other AWS services, such as CodePipeline, CodeCommit, and S3, allowing you to create a complete CI/CD pipeline.
5. **Pay-As-You-Go Pricing:**
   CodeBuild charges you only for the build time you use, making it a cost-effective solution for automating your build process.

Deploying on S3 Using AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that automates the build, test, and deployment phases of your release process. CodePipeline integrates with other AWS services, including CodeBuild and S3, to automate the entire application release process.

Key Steps to Deploy on S3 Using AWS CodePipeline:

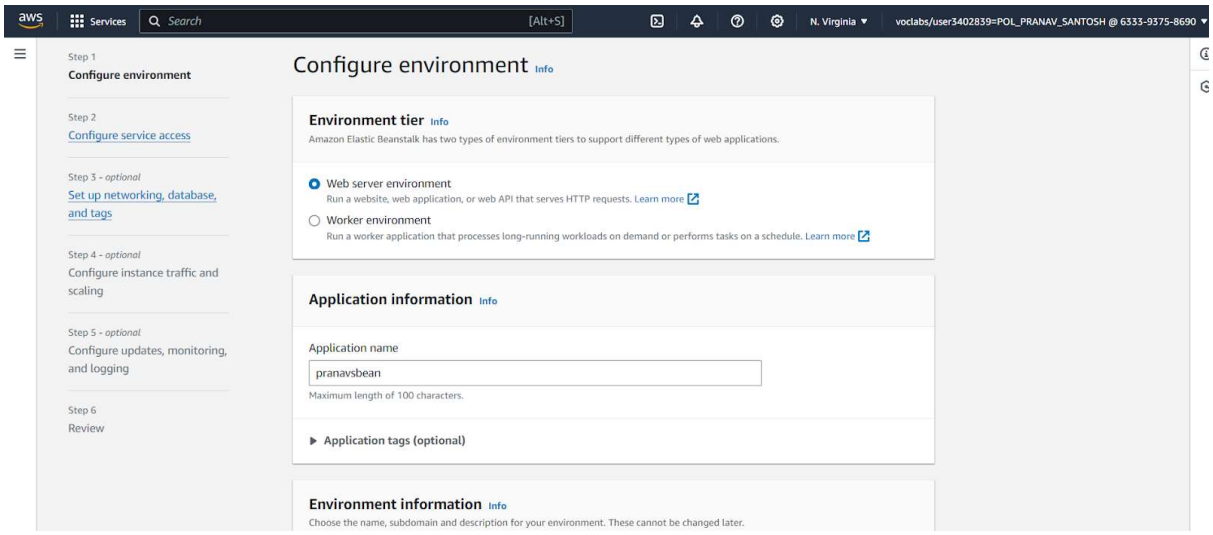
1. **Source Stage:**
   The first stage in the pipeline is typically the source stage, where the source code is retrieved from a repository, such as AWS CodeCommit, GitHub, or S3. This code serves as the input for the subsequent build and deployment stages.
2. **Build Stage (Using AWS CodeBuild):**
   In the build stage, CodePipeline triggers a build process using AWS CodeBuild. CodeBuild compiles the source code, runs tests, and packages the application. The output is an artifact that is stored in an S3 bucket, ready for deployment.
3. **Deploy Stage (Deploying to S3):**
   The final stage in the pipeline is the deploy stage. In this stage, CodePipeline

automatically deploys the artifacts generated in the build stage to an S3 bucket. The S3 bucket can serve as a static website hosting service or store files that are accessed by your application.

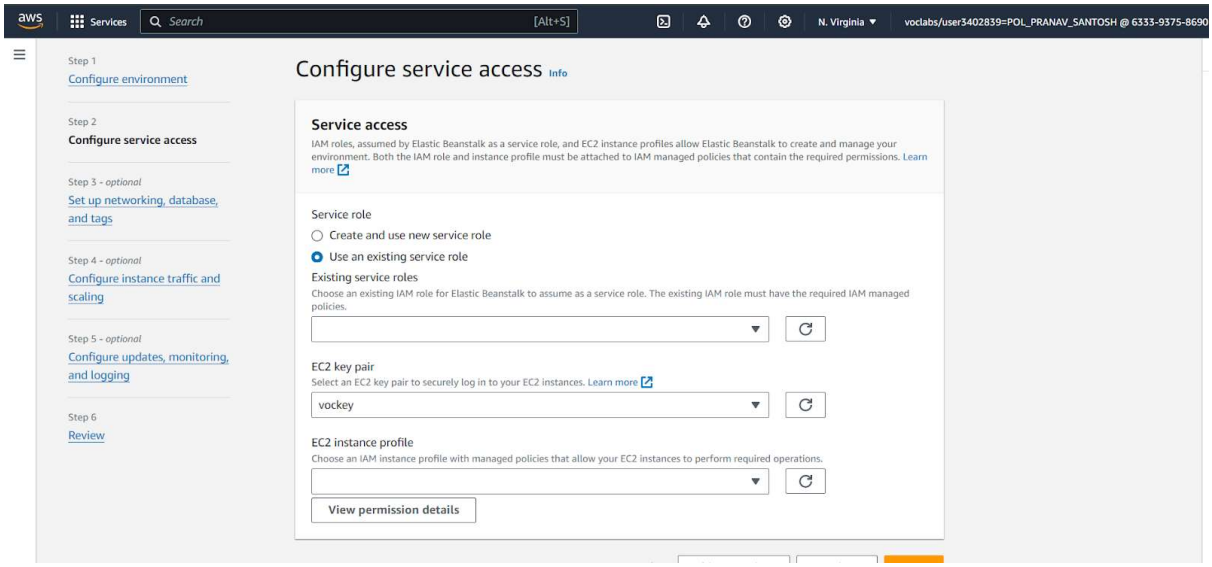
4. **Automation and Notifications:**
   CodePipeline can be configured to trigger automatic builds and deployments based on changes to the source code repository. It can also be integrated with Amazon SNS to send notifications about the status of the pipeline, allowing you to monitor the release process.
5. **Pipeline Monitoring:**
   CodePipeline provides visual monitoring and logging of each stage in the pipeline, making it easier to track the status of your builds and deployments. Any failures in the pipeline can be quickly identified and addressed.

Implementation ;
Elastic Beanstalk

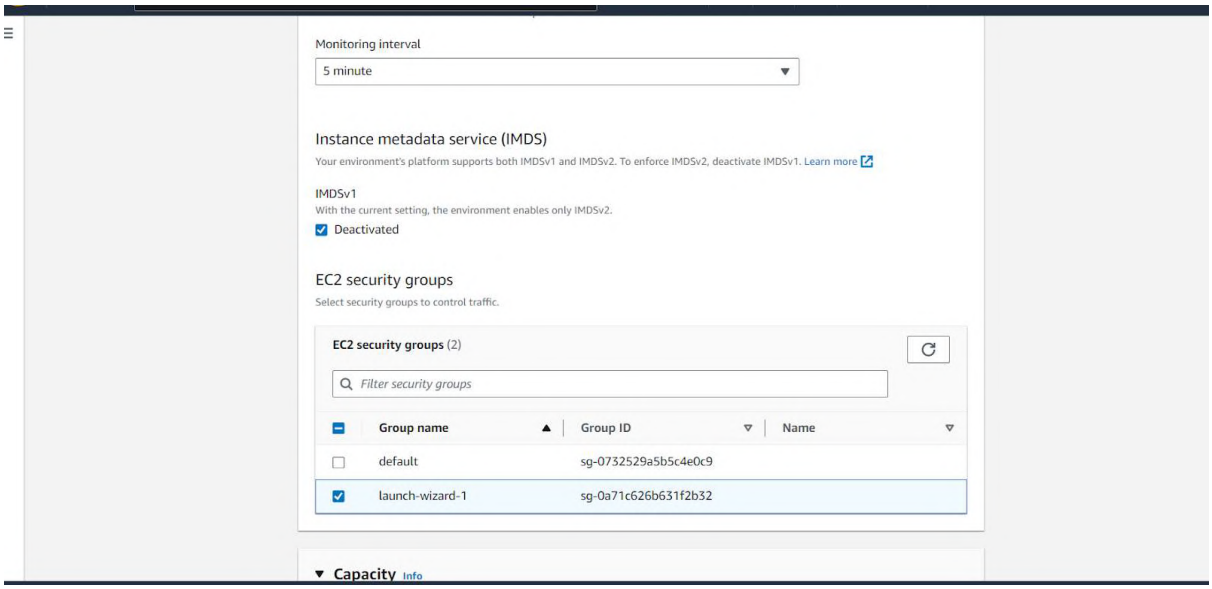
Step 1: create environment

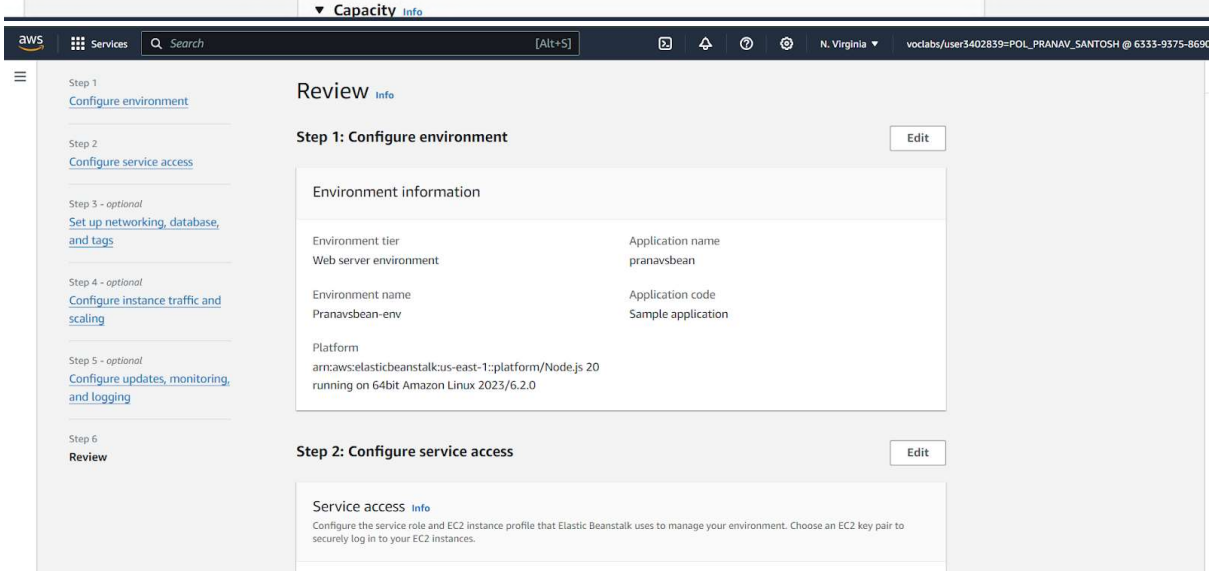


Step 2 : add your Ec2 key pair and instance profile

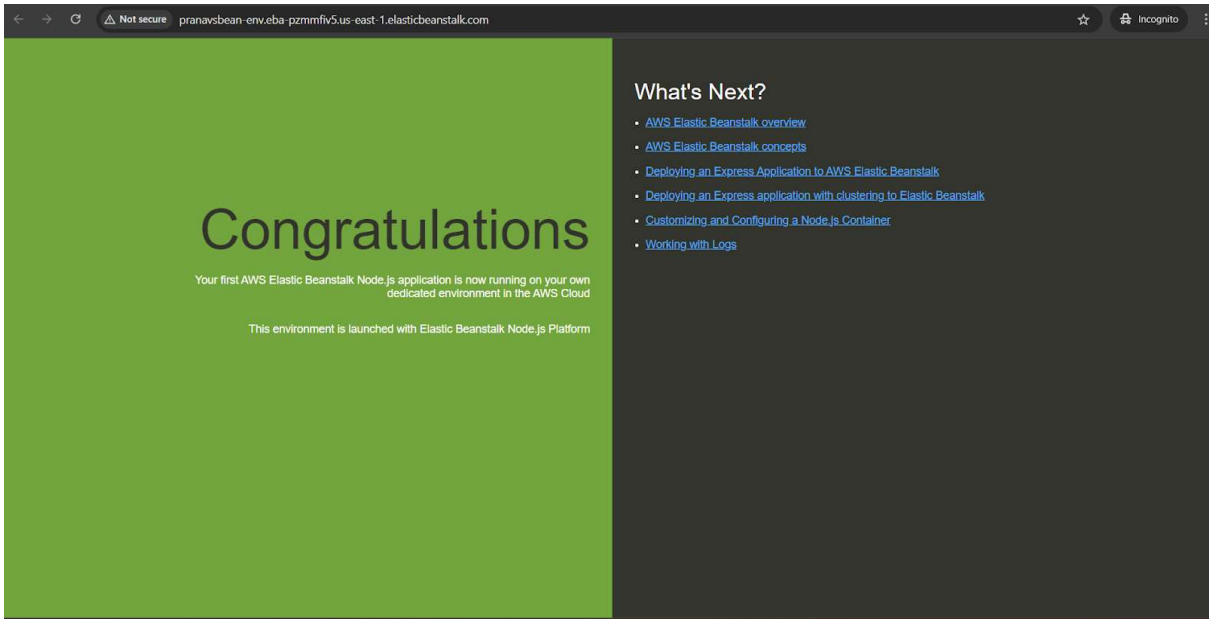


Step 3 : add security config and review all settings



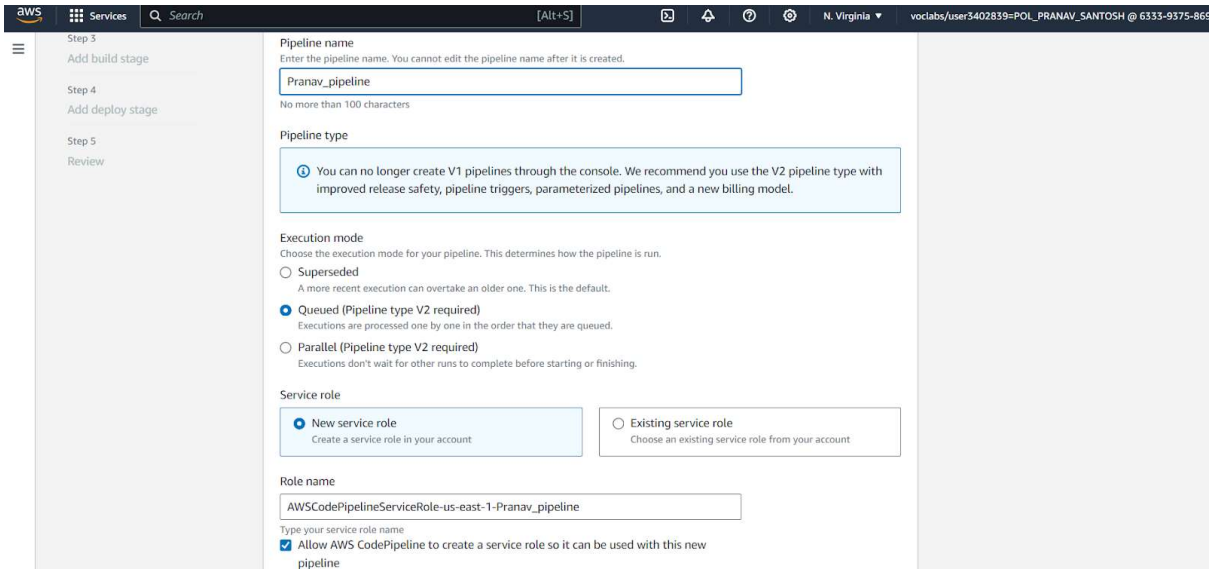


Step 4 : Beanstalk environment is created

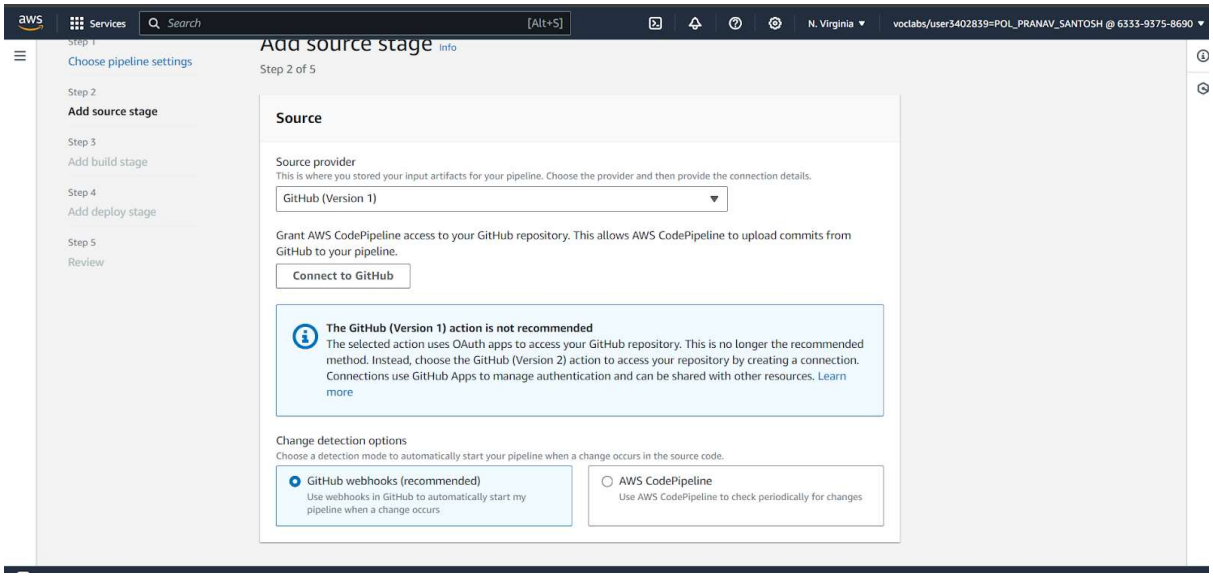


Pipeline Creation

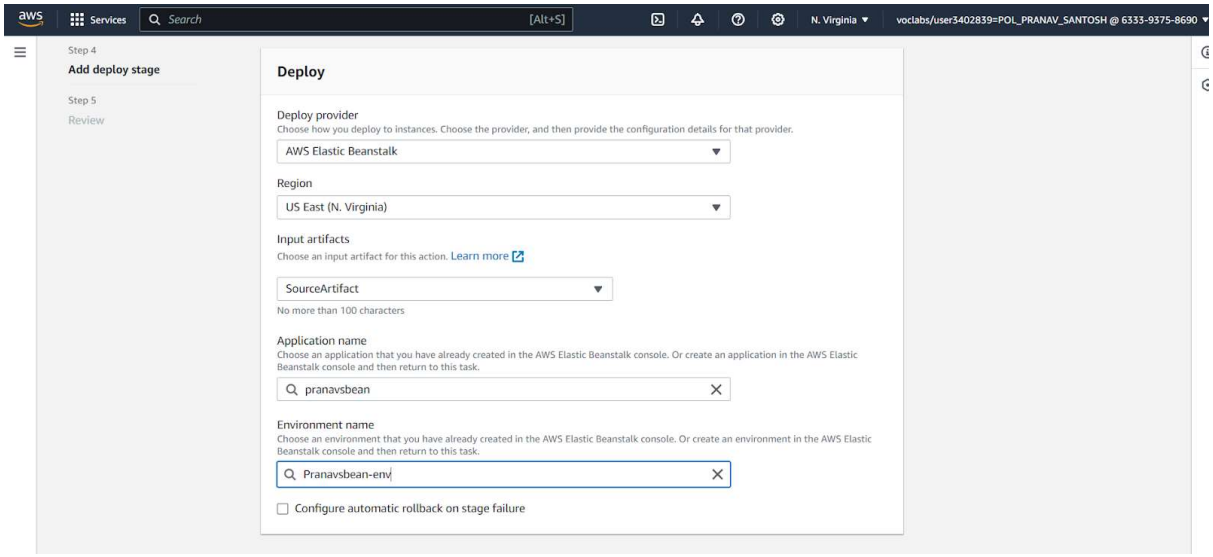
Step 1 : click on create pipeline and give name



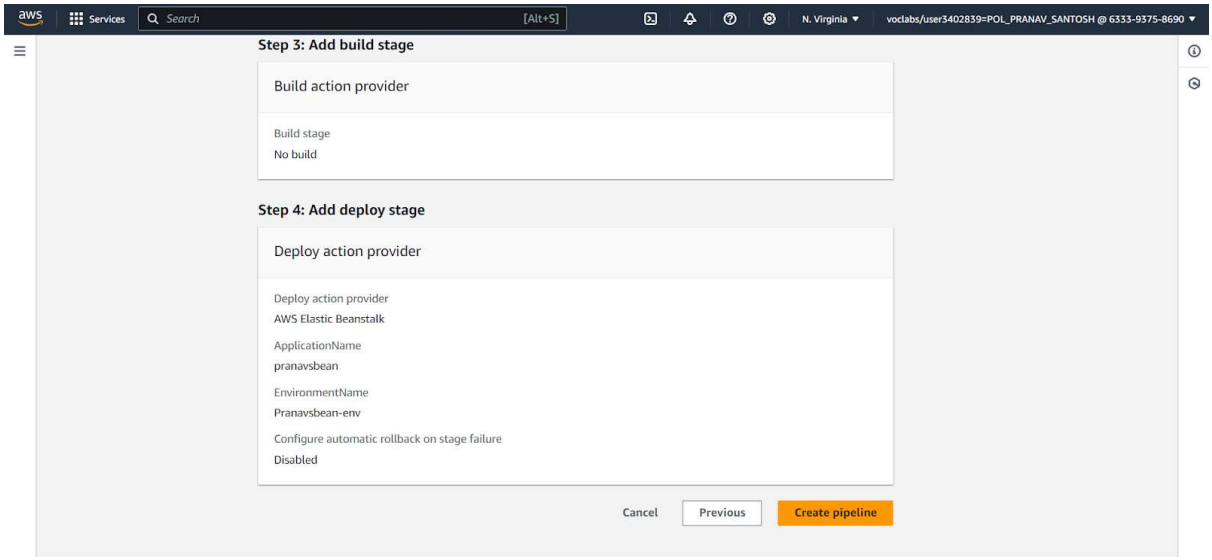
Step 2 : Add Your github account and add the file to add to pipeline deployment

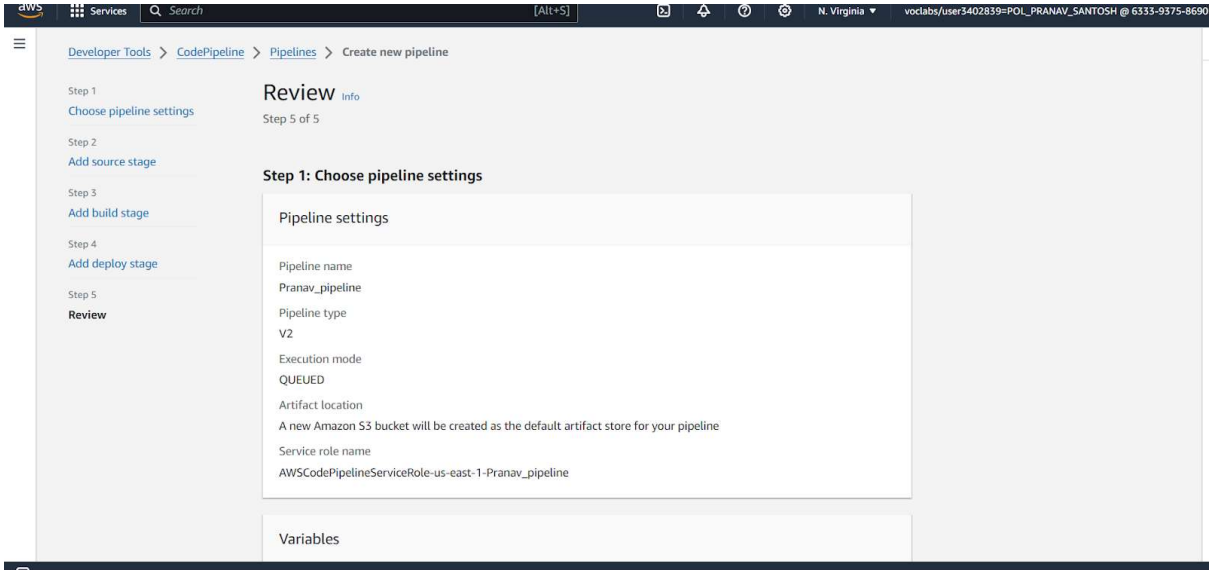


Step 3 : Add deploy config choosing the elastic beanstalk



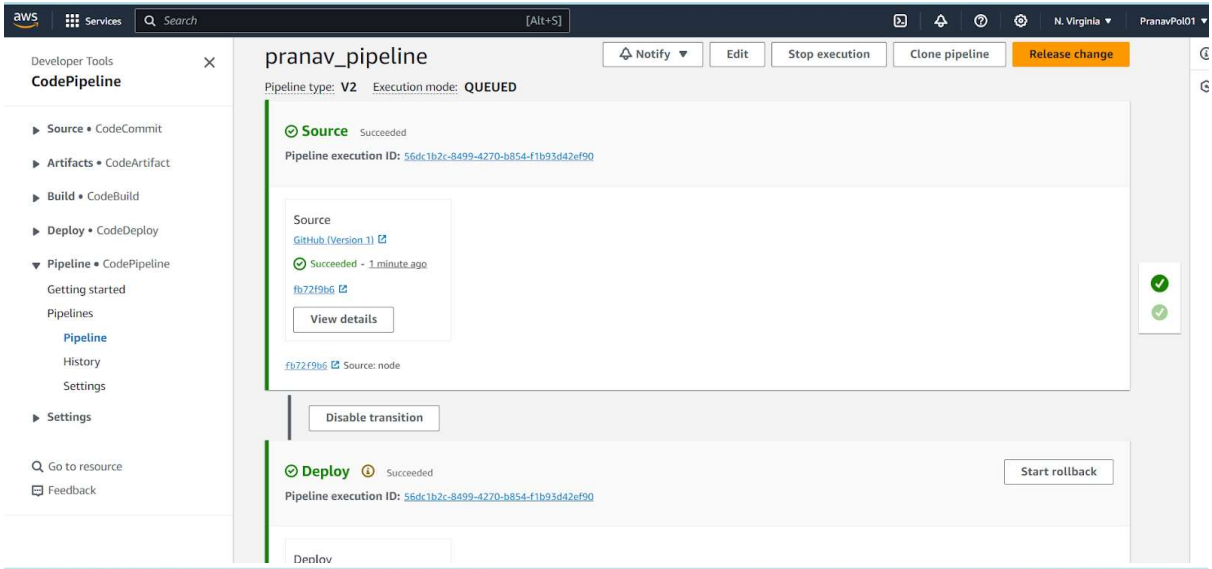
Step 4 : review changes and submit

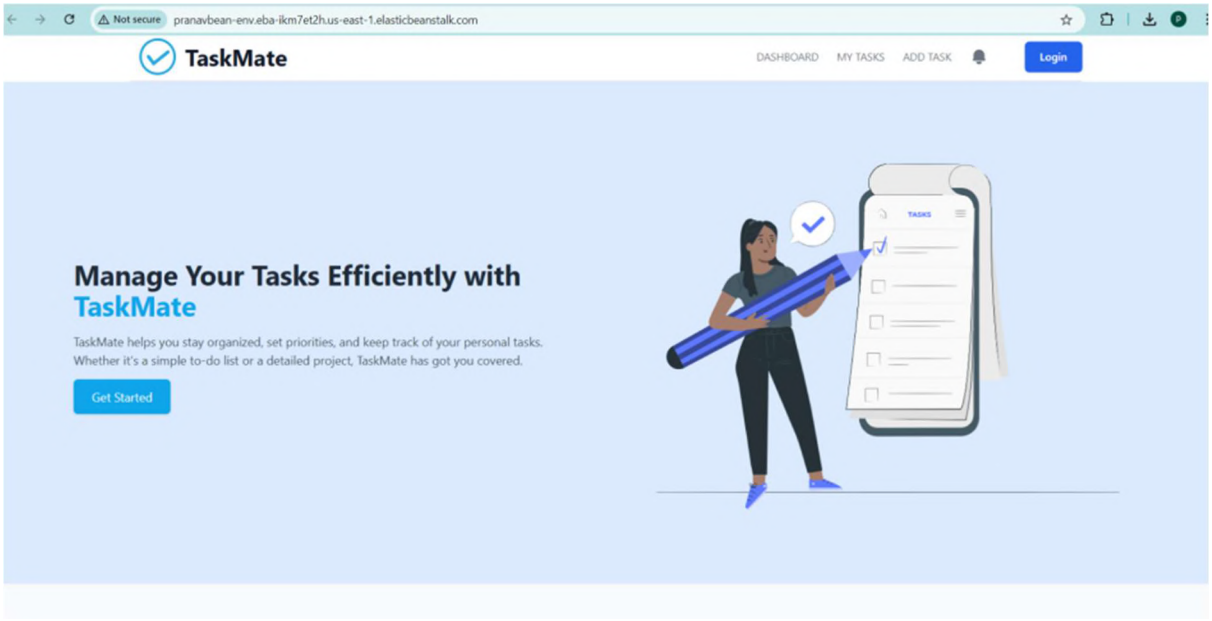




Step 5 : view the pipeline build and deployment

Step 6 : Check the deployed website at beanstalk link


aws
Services
Search
[Alt+S]
N. Virginia
PranavPol01
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
pranav_pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change
Pipeline type: V2
Execution mode: QUEUED
Source
Succeeded
Pipeline execution ID: 56dc1b2c-8499-4270-b854-f1b93d42ef90
Source
GitHub (Version 1)
Succeeded - 1 minute ago
fb72f9b6
View details
fb72f9b6 Source: node
Disable transition
Deploy
Succeeded
Start rollback
Pipeline execution ID: 56dc1b2c-8499-4270-b854-f1b93d42ef90
Deploy


Not secure
pranavbean-env.eba-ikm7et2h.us-east-1.elasticbeanstalk.com
TaskMate
DASHBOARD
MY TASKS
ADD TASK
Login
Manage Your Tasks Efficiently with
TaskMate
TaskMate helps you stay organized, set priorities, and keep track of your personal tasks.
Whether it's a simple to-do list or a detailed project, TaskMate has got you covered.
Get Started
TASKS

EXPERIMENT NO. 3

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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

Theory :

Kubernetes (often abbreviated as K8s) is a powerful open-source platform for managing containerized applications across multiple hosts. It provides an abstraction layer for deploying and scaling applications, ensuring high availability and fault tolerance. The **Kubernetes Cluster** architecture consists of several key components divided into two major categories:

1. **Master Node Components**
2. **Worker Node Components**

1. Master Node Components

The **Master Node** is responsible for managing the entire Kubernetes cluster. It acts as the brain of the cluster, orchestrating container deployments, scaling, and communication between the nodes.

a. API Server

- Acts as the front-end for the Kubernetes control plane.
- All internal and external communication with the cluster is through the API server.
- It validates and processes RESTful requests.

b. Scheduler

- Responsible for distributing workload across the cluster.
- Determines which worker node will host a newly created pod based on resource availability, policy constraints, and other factors.

c. Controller Manager

- Ensures that the desired state of the cluster matches the actual state.
- Examples of controllers include the **Node Controller**, **Replication Controller**, **Endpoint Controller**, and **Service Controller**.

d. etcd

- A consistent and highly-available key-value store used as Kubernetes' backing store.

- Stores the entire configuration and state of the Kubernetes cluster.
- etcd must be backed up regularly as it's critical to the integrity of the cluster.

e. Cloud Controller Manager (optional)

- Integrates cloud-specific APIs into Kubernetes.
- Handles cloud-related services like load balancing, managing cloud storage, and node lifecycle events in cloud platforms like AWS, GCP, Azure, etc.

2. Worker Node Components

The **Worker Nodes** are where your applications run. Each worker node communicates with the master node and executes the commands given.

a. Kubelet

- The primary agent that runs on each worker node.
- Ensures that the containers described in a pod are running and healthy.
- Communicates with the API server and ensures that the desired state is met on the node.

b. Kube-proxy

- Manages the networking for the worker node.
- Ensures that traffic is correctly routed to and from the pods running on the worker node.
- Facilitates communication between services within the cluster.

c. Container Runtime

- Responsible for pulling container images and running the containers.
- Common runtimes include Docker, containerd, or CRI-O.

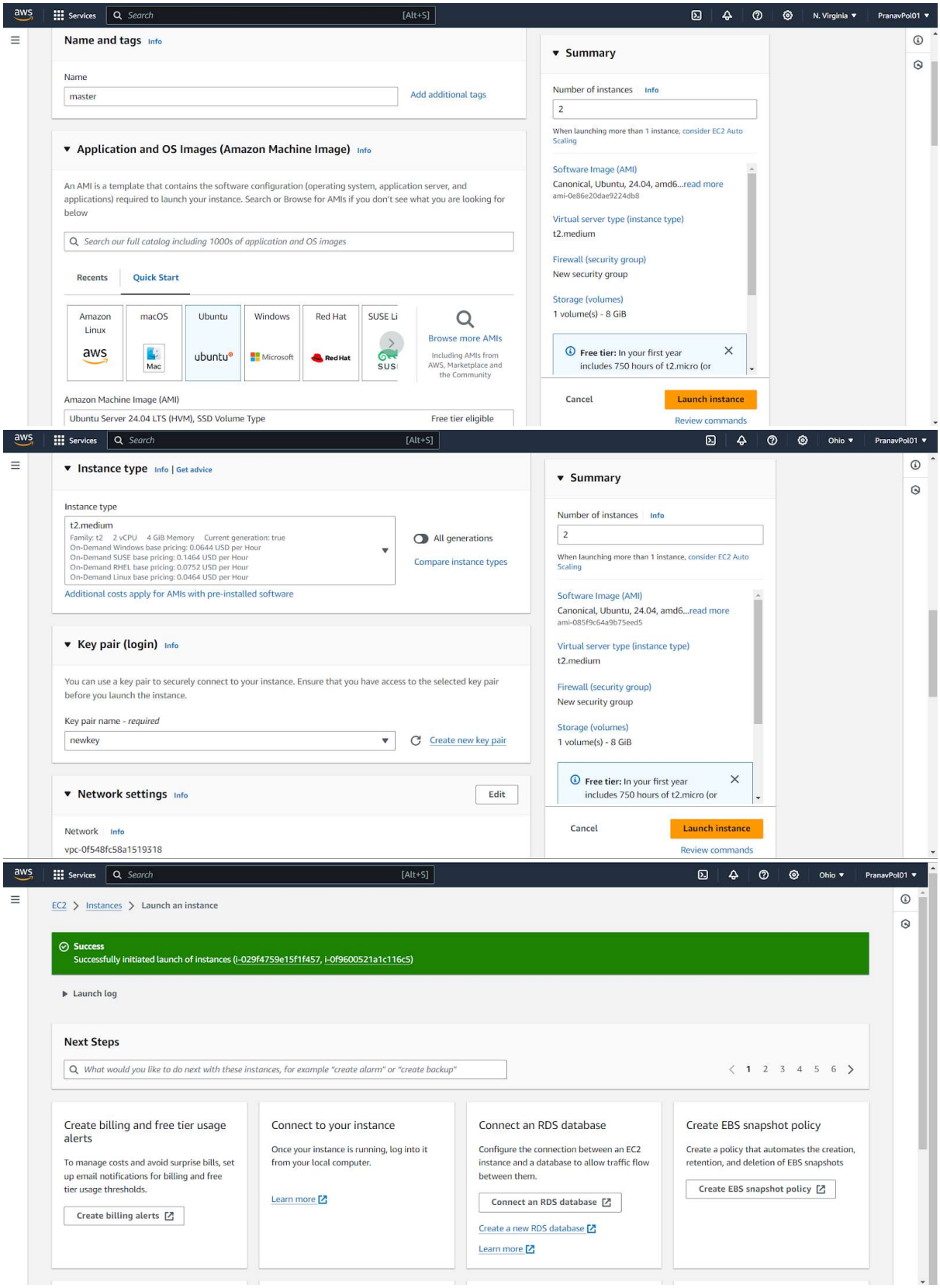
d. Pods

- The smallest deployable unit in Kubernetes.
- A pod encapsulates one or more containers and their shared storage, network, and configuration options.

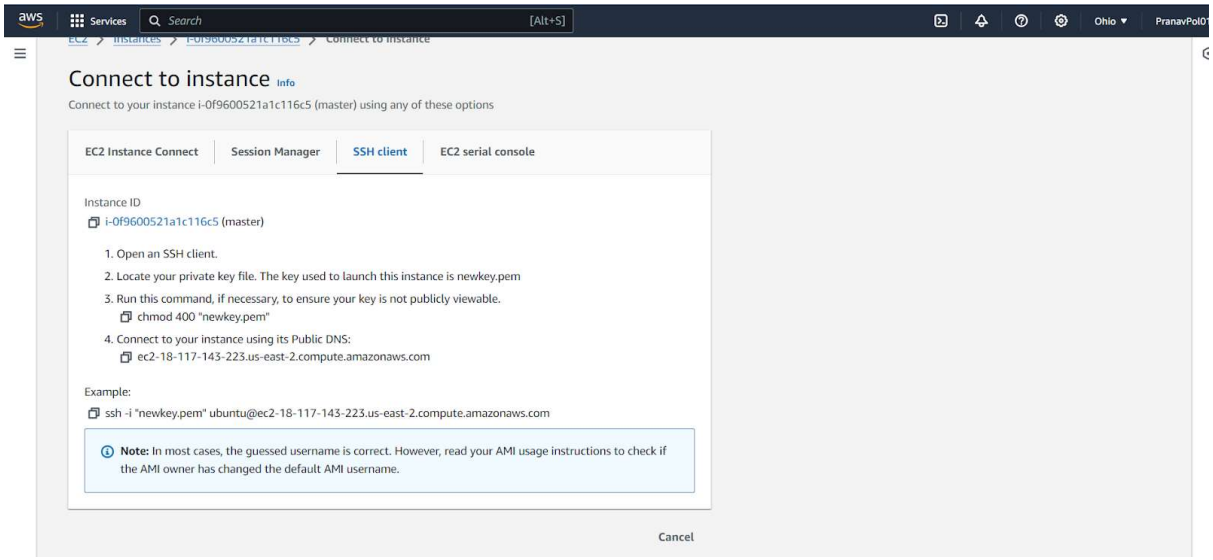
Implementation :

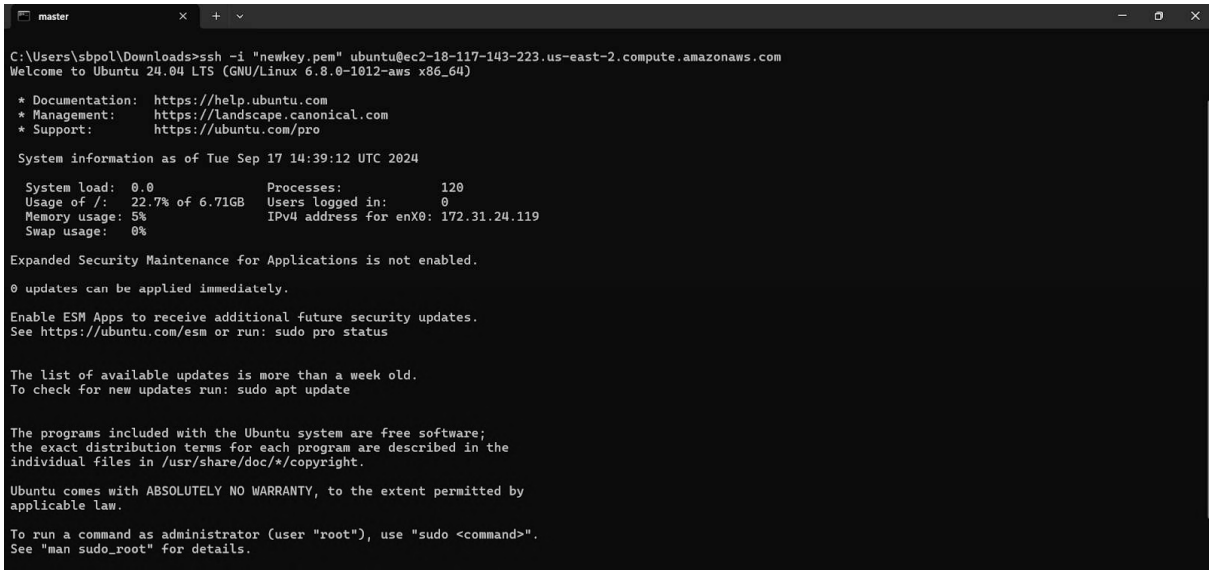
Step 1: Pre-requisites

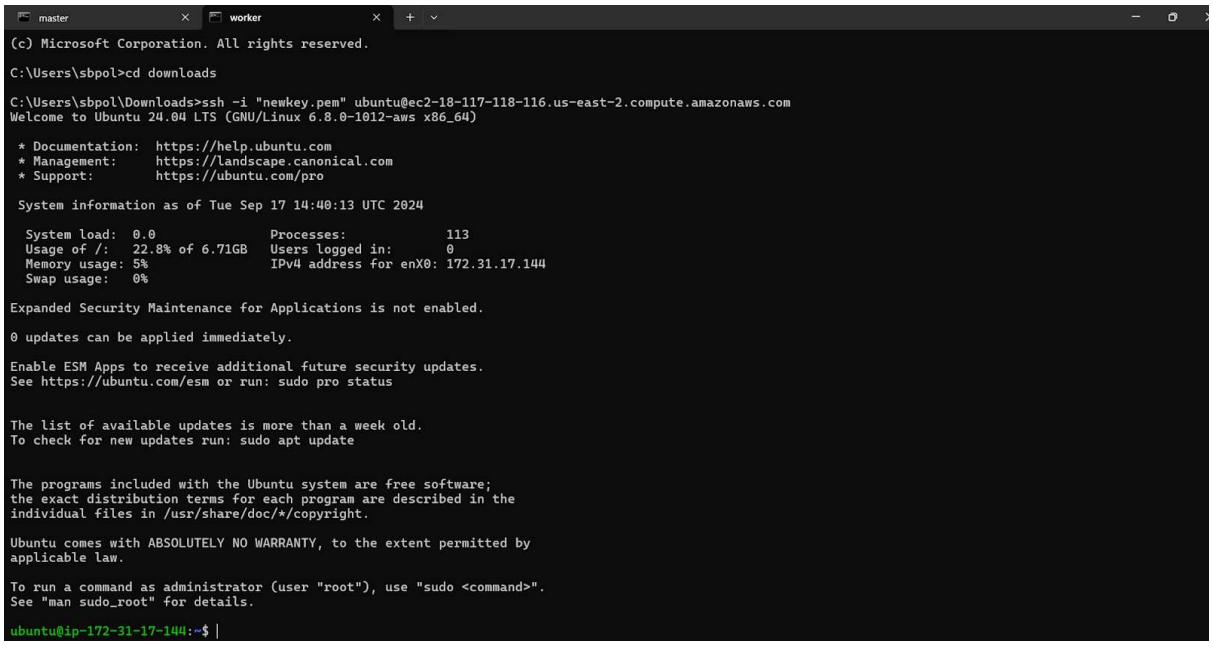
1.1. Create 2 EC2 instances,one for the master node and one for the worker nodes.



1.2 . After the instances have been created,copy the text given in the example part of each of the three instances into git bash.

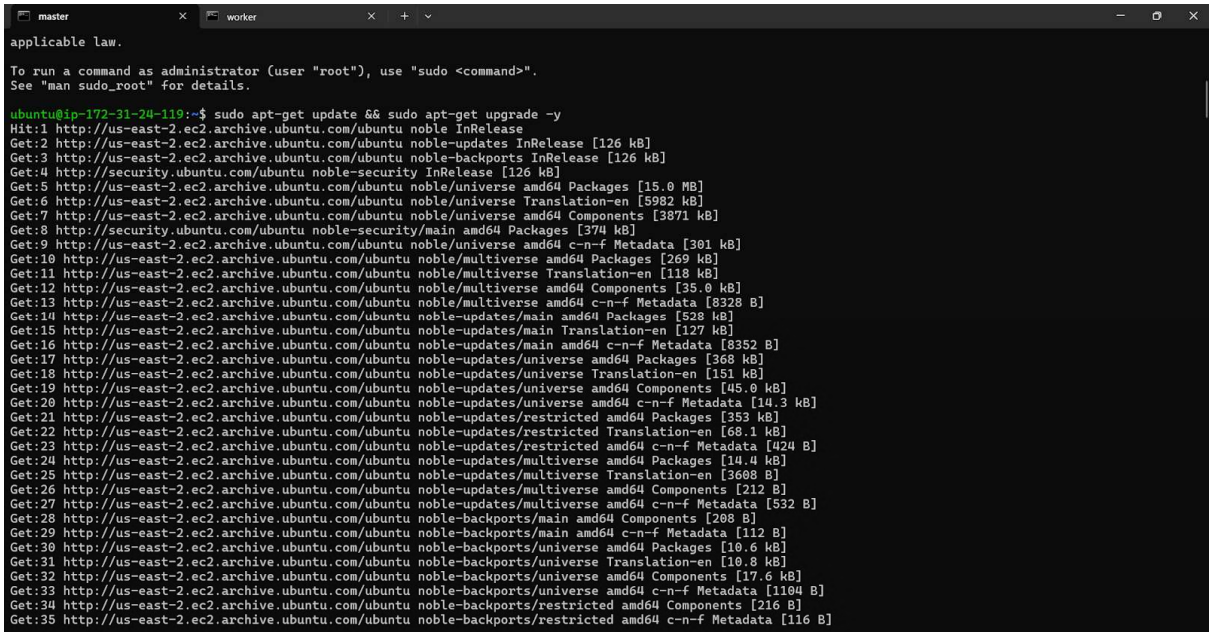

aws
Services
Search
[Alt+S]
Ohio
PranavPol0
Connect to instance
Info
Connect to your instance i-0f9600521a1c116c5 (master) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-0f9600521a1c116c5 (master)
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is newkey.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "newkey.pem"
4. Connect to your instance using its Public DNS:
ec2-18-117-143-223.us-east-2.compute.amazonaws.com
Example:
ssh -i "newkey.pem" ubuntu@ec2-18-117-143-223.us-east-2.compute.amazonaws.com
Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.
Cancel


master
C:\Users\sbpol\Downloads>ssh -i "newkey.pem" ubuntu@ec2-18-117-143-223.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Tue Sep 17 14:39:12 UTC 2024
System load: 0.0
Usage of /: 22.7% of 6.71GB
Memory usage: 5%
Swap usage: 0%
Processes: 120
Users logged in: 0
IPv4 address for enX0: 172.31.24.119
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.


master
worker
(c) Microsoft Corporation. All rights reserved.
C:\Users\sbpol>cd downloads
C:\Users\sbpol\Downloads>ssh -i "newkey.pem" ubuntu@ec2-18-117-118-116.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Tue Sep 17 14:40:13 UTC 2024
System load: 0.0
Usage of /: 22.8% of 6.71GB
Memory usage: 5%
Swap usage: 0%
Processes: 113
Users logged in: 0
IPv4 address for enX0: 172.31.17.144
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-17-144:~$

Step 2:Prepare Nodes

2.1. Update the package manager on all nodes:
Sudo apt-get update &7 sudo apt-get update -y



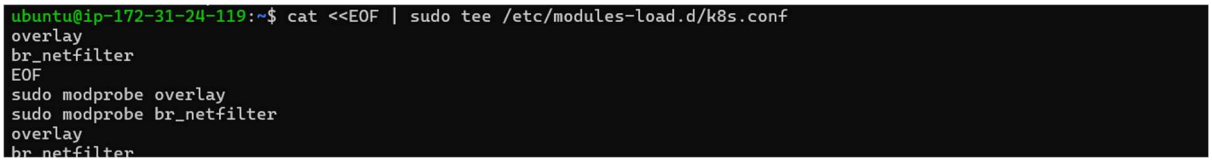
2.2. Disable Swap (Kubernetes requires swap to be off):

sudo swapoff -a

sudo sed -i '/ swap / s/^/#/' /etc/fstab

2.3. Load necessary kernel modules for networking and iptables: cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter



2.1. Configure sysctl settings for Kubernetes networking: cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1
EOF

sudo sysctl --system

```
ubuntu@ip-172-31-24-119:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...
* Applying /etc/sysctl.d/10-zeropage.conf ...
* Applying /etc/sysctl.d/50-cloudimg-settings.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
* Applying /etc/sysctl.conf ...
kernel.apparmor_restrict_unprivileged_userns = 1
kernel.printk = 4 4 1 7
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
kernel.kptr_restrict = 1
kernel.sysrq = 176
vm.max_map_count = 1048576
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.all.rp_filter = 2
kernel.yama.ptrace_scope = 1
vm.mmap_min_addr = 65536
```

Step 3: Install Docker

Kubernetes uses container runtimes like Docker. Install Docker on all nodes.
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu

$(lsb_release -cs) stable" sudo apt-get update sudo apt-get install -y docker-ce docker-ce-cli containerd.io

```
ubuntu@ip-172-31-24-119:~$
ubuntu@ip-172-31-24-119:~$ sudo rm /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
ubuntu@ip-172-31-24-119:~$ sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo a
pt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable" sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 126 kB in 0s (352 kB/s)
Reading package lists... Done
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
E: Unable to locate package https://download.docker.com/linux/ubuntu
E: Couldn't find any package by glob 'https://download.docker.com/linux/ubuntu'
E: Couldn't find any package by regex 'https://download.docker.com/linux/ubuntu'
gpg: no valid OpenPGP data found.
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble apt-get stable sudo update'
Description:
Archive for codename: noble components: apt-get,stable,sudo,update
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [13.8 kB]
Fetched 62.6 kB in 0s (127 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION s
ection in apt-key(8) for details.
W: Skipping acquire of configured file 'apt-get/binary-amd64/Packages' as repository 'https://download.docker.com/linux/ubuntu noble InRelease' doesn't have
```

Configure Docker for Kubernetes:

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",

  "log-opts": {

    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
 sudo systemctl restart docker
```

```
ubuntu@ip-172-31-24-119:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"], "log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
sudo systemctl restart docker

{
"exec-opts": ["native.cgroupdriver=systemd"], "log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
ubuntu@ip-172-31-24-119:~$
```

Step 4: Install kubeadm, kubelet, kubectl

Install Kubernetes tools on all nodes.

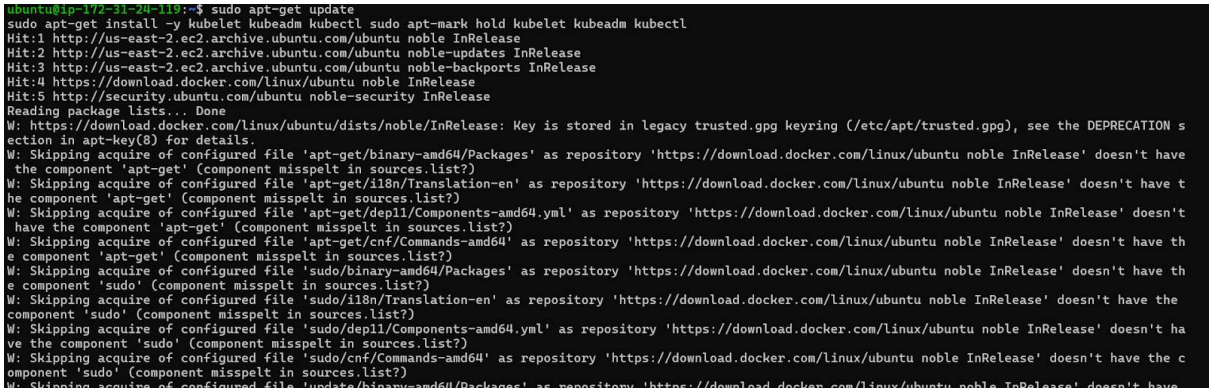
4.1. Add Kubernetes APT repository:

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee

/etc/apt/sources.list.d/kubernetes.list

4.2. Install kubeadm, kubelet, and kubectl: sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl



4.3 Install crio runtime
Using CRI-O can help streamline your container management processes, improve the security of your applications, and simplify the overall management of your infrastructure.

sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d
/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.99.48).
software-properties-common set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.4).
curl set to manually installed.
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
```

4.4 Add Kubernetes APT repository and install required packages
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-44-131:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyri
ng.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubern
etes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /
```

4.5 Install kubelet kubectl and kubeadm
sudo apt-get update -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
sudo apt-get update -y
sudo apt-get install -y jq

sudo systemctl enable --now kubelet
sudo systemctl start kubelet

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
sudo apt-get update -y
sudo apt-get install -y jq
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease [1189 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  Packages [14.0 kB]
Fetched 15.1 kB in 1s (23.9 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Selected version '1.29.0-1.1' (isv:kubernetes:core:stable:v1.29:pkgs.k8s.io [amd64]) for 'kubelet'
Selected version '1.29.0-1.1' (isv:kubernetes:core:stable:v1.29:pkgs.k8s.io [amd64]) for 'kubectl'
Selected version '1.29.0-1.1' (isv:kubernetes:core:stable:v1.29:pkgs.k8s.io [amd64]) for 'kubeadm'
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 7 not upgraded.
```

```
ubuntu@ip-172-31-44-131:~$
sudo systemctl enable --now kubelet
sudo systemctl start kubelet
```

Step 5 : Execute only on master node

5.1 : kudeadm config and init

```
ubuntu@ip-172-31-44-131:~$ sudo kubeadm config images pull
I0917 16:12:12.827368   12932 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.9
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.29.9
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.29.9
[config/images] Pulled registry.k8s.io/kube-proxy:v1.29.9
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
```

```
ubuntu@ip-172-31-44-131:~$ sudo kubeadm init
I0917 16:12:56.929652   13290 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.9
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0917 16:12:57.776009   13290 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.10" of the container runtime is inconsistent with that
 used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[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-172-31-44-131 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster
.local] and IPs [10.96.0.1 172.31.44.131]
[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
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-44-131 localhost] and IPs [172.31.44.131 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-44-131 localhost] and IPs [172.31.44.131 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
```

5.2 : install config from github

network plugin = calico

```
ubuntu@ip-172-31-44-131:~$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

5.3 Create token of kubeadm

```
ubuntu@ip-172-31-44-131:~$ kubeadm token create --print-join-command
kubeadm join 172.31.44.131:6443 --token zixj91.n5y95uhn5xyydmps --discovery-token-ca-cert-hash sha256:f48119eb49e6f94e34ef69d388d3b955a7b380d34a887e32494385
67341d9bcb
```

Step 6 : Execute on ALL of your Worker Node's

6.1 : perform pre- flight checks and Paste the join command you got from the master node and append --v=5 at the end

```
ubuntu@ip-172-31-40-114:~$ sudo kubeadm join 172.31.44.131:6443 --token zixj91.n5y95uhn5xyydmps --discovery-token-ca-cert-hash sha256:f48119eb49e6f94e34ef69d388d3b955a7b380d34a88
67341d9bcb --v=5
I0917 16:36:16.052568   13354 join.go:413] [preflight] found NodeName empty; using OS hostname as NodeName
I0917 16:36:16.052700   13354 initconfiguration.go:122] detected and using CRI socket: unix:///var/run/crio/crio.sock
[preflight] Running pre-flight checks
I0917 16:36:16.052757   13354 preflight.go:93] [preflight] Running general checks
I0917 16:36:16.052787   13354 checks.go:280] validating the existence of file /etc/kubernetes/kubelet.conf
I0917 16:36:16.052793   13354 checks.go:280] validating the existence of file /etc/kubernetes/bootstrap-kubelet.conf
I0917 16:36:16.052800   13354 checks.go:104] validating the container runtime
I0917 16:36:16.074072   13354 checks.go:639] validating whether swap is enabled or not
I0917 16:36:16.074157   13354 checks.go:370] validating the presence of executable crictl
I0917 16:36:16.074180   13354 checks.go:370] validating the presence of executable conntrack
I0917 16:36:16.074191   13354 checks.go:370] validating the presence of executable ip
I0917 16:36:16.074204   13354 checks.go:370] validating the presence of executable iptables
I0917 16:36:16.074216   13354 checks.go:370] validating the presence of executable mount
I0917 16:36:16.074229   13354 checks.go:370] validating the presence of executable nsenter
I0917 16:36:16.074260   13354 checks.go:370] validating the presence of executable ebtables
I0917 16:36:16.074291   13354 checks.go:370] validating the presence of executable ethtool
I0917 16:36:16.074305   13354 checks.go:370] validating the presence of executable socat
I0917 16:36:16.074328   13354 checks.go:370] validating the presence of executable tc
I0917 16:36:16.074343   13354 checks.go:370] validating the presence of executable touch
I0917 16:36:16.074386   13354 checks.go:516] running all checks
I0917 16:36:16.084808   13354 checks.go:401] checking whether the given node name is valid and reachable using net.LookupHost
I0917 16:36:16.086249   13354 checks.go:605] validating kubelet version
I0917 16:36:16.133477   13354 checks.go:130] validating if the "kubelet" service is enabled and active
I0917 16:36:16.145208   13354 checks.go:203] validating availability of port 10250
I0917 16:36:16.145467   13354 checks.go:280] validating the existence of file /etc/kubernetes/pki/ca.crt
I0917 16:36:16.145489   13354 checks.go:430] validating if the connectivity type is via proxy or direct
I0917 16:36:16.145528   13354 checks.go:329] validating the contents of file /proc/sys/net/bridge/bridge-nf-call-iptables
I0917 16:36:16.145575   13354 checks.go:329] validating the contents of file /proc/sys/net/ipv4/ip_forward
I0917 16:36:16.145610   13354 join.go:532] [preflight] Discovering cluster-info
I0917 16:36:16.145642   13354 token.go:80] [discovery] Created cluster-info discovery client, requesting info from "172.31.44.131:6443"
I0917 16:36:16.155196   13354 token.go:118] [discovery] Requesting info from "172.31.44.131:6443" again to validate TLS against the pinned public key
I0917 16:36:16.162805   13354 token.go:135] [discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "172
:6443"
I0917 16:36:16.162817   13354 discovery.go:52] [discovery] Using provided TLSBootstrapToken as authentication credentials for the join process
I0917 16:36:16.162825   13354 join.go:546] [preflight] Fetching init configuration
I0917 16:36:16.162830   13354 join.go:592] [preflight] Retrieving KubeConfig objects
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
I0917 16:36:16.171694   13354 kubeproxy.go:55] attempting to download the KubeProxyConfiguration from ConfigMap "kube-proxy"
I0917 16:36:16.174741   13354 kubelet.go:74] attempting to download the KubeletConfiguration from ConfigMap "kubelet-config"
I0917 16:36:16.178638   13354 initconfiguration.go:114] skip CRI socket detection, fill with the default CRI socket unix:///var/run/containerd/containerd.sock
I0917 16:36:16.178801   13354 interface.go:432] Looking for default routes with IPv4 addresses
I0917 16:36:16.178819   13354 interface.go:437] Default route transits interface "enX0"
```

Step 7 : Verify cluster connection on master node

```
ubuntu@ip-172-31-44-131:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-40-114  Ready    <none>          26s   v1.29.0
ip-172-31-44-131  Ready    control-plane   23m   v1.29.0
```

EXPERIMENT NO. 4

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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

Theory:

What is kubectl?

kubectl is the command-line interface (CLI) used to interact with a Kubernetes cluster. It allows users to manage cluster resources, deploy applications, inspect and manage cluster components, and much more. Using kubectl, you can communicate with the Kubernetes API server to issue commands and queries.

Common kubectl commands:

- kubectl get: View information about resources.
- kubectl describe: Detailed description of resources.
- kubectl create/apply: Create or update resources.
- kubectl delete: Delete resources.

kubectl plays a crucial role in the day-to-day operation of a Kubernetes cluster.

Basic Concepts in Kubernetes

Before diving into the application deployment process, it's important to understand a few key Kubernetes objects:

1. **Pods**: The smallest deployable unit in Kubernetes. A pod encapsulates one or more containers (usually a single container) that share the same network namespace and storage.
2. **Deployments**: A Kubernetes resource that defines how to create and manage pods. It ensures the specified number of pod replicas are running at any given time and handles updates and rollbacks.
3. **Services**: An abstraction that defines how to access the pods. A service allows you to expose your pods to internal or external clients.
4. **ReplicaSets**: Ensures that a specified number of pod replicas are running at all times. It is managed by a Deployment, but can also be used independently.

Step 1: Install Kubectl on Ubuntu

1.1 Add Kubernetes APT repository

First, add the Kubernetes repository to your system.

1. **Install prerequisites:**

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease
Reading package lists... Done
```

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get install -y apt-transport-https ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.7.14build2).
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.4).
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
```

2. **Add the GPG key for Kubernetes:**

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

3. **Add the Kubernetes repository:**

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee

/etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-44-131:~$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main" | sudo
  tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main
```

1.2 Install kubectl Now install kubectl: sudo apt-get update

sudo apt-get install -y kubectl

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:5 https://packages.cloud.google.com/apt kubernetes-focal InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-focal Release
  404  Not Found [IP: 142.250.76.206 443]
Reading package lists... Done
E: The repository 'https://apt.kubernetes.io kubernetes-focal Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get install -y kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubectl is already the newest version (1.29.0-1.1).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
ubuntu@ip-172-31-44-131:~$
```

Verify the installation(extra): kubectlversion --client

```
ubuntu@ip-172-31-44-131:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Step 2: Deploying Your Application on Kubernetes

2.1 Set up Kubernetes Cluster

1. If you haven't already set up a Kubernetes cluster (e.g., with kubeadm), use minikube or any managed Kubernetes service (like EKS, GKE, etc.) to get a cluster running.

2. Once your cluster is ready, verify the nodes:

kubectl get nodes

```
ubuntu@ip-172-31-44-131:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-40-114    Ready    <none>          9m55s   v1.29.0
ip-172-31-44-131    Ready    control-plane   33m     v1.29.0
```

Step 3:Create the Deployment YAML file

a) Create the YAML file: Use a text editor to create a file named nginx-deployment.yaml Add the Deployment Configuration: Copy and paste the following YAML content into the file. Save and exit the editor (Press Ctrl+X, then Y, and Enter).

```
 GNU nano 7.2                                  nginx-deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
Save modified buffer?
 Y Yes
 N No           ^C Cancel
```

Step

4:Create the Service YAML File

a) Create the YAML File: Create another file named nginx-service.yaml Add the Service Configuration: Copy and paste the following YAML content into the file given below.

Step 5:Apply the YAML Files

a) Deploy the Application: Use kubectl to create the Deployment and Service from the YAML files.

Verify the Deployment: Check the status of your Deployment,Pods and Services.

Describe the deployment(Extra)

```
ubuntu@ip-172-31-44-131:~$ ubuntu@ip-172-31-44-131:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-44-131:~$ kubectl apply -f nginx-service.yaml
service/my-nginx created
```

Step 6:Ensure Service is Running

6.1 **Verify Service**: Run the following command to check the services running in your cluster:

Kubectl get deployment

Kubectl get pods

kubectl get service

```
ubuntu@ip-172-31-44-131:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           74s
ubuntu@ip-172-31-44-131:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-8d7rx   1/1     Running   0          81s
nginx-deployment-86dcfdf4c6-bdbcm   1/1     Running   0          81s
ubuntu@ip-172-31-44-131:~$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   48m
my-nginx     ClusterIP   10.111.168.255   <none>        80/TCP    55s
```

Step 7:Forward the Service Port to Your Local Machine

kubectl port-forward allows you to forward a port from your local machine to a port on a service running in the Kubernetes cluster.

1. **Forward the Service Port**: Use the following command to forward a local port to the service's target port.

kubectl port-forward service/<service-name> <local-port>:<service-port>

This command will forward local port 8080 on your machine to port 80 of the service nginx-service running inside the cluster.

```
ubuntu@ip-172-31-44-131:~$ kubectl describe deployments
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Tue, 17 Sep 2024 17:00:22 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-deployment-86dcfdf4c6 (2/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  2m9s  deployment-controller  Scaled up replica set nginx-deployment-86dcfdf4c6 to 2
```

2. This means port forwarding is now active, and any traffic to localhost:8080 will be routed to the nginx-service on port 80.

```
ubuntu@ip-172-31-44-131:~$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   49m
my-nginx     ClusterIP   10.111.168.255   <none>        80/TCP    2m9s
ubuntu@ip-172-31-44-131:~$
```

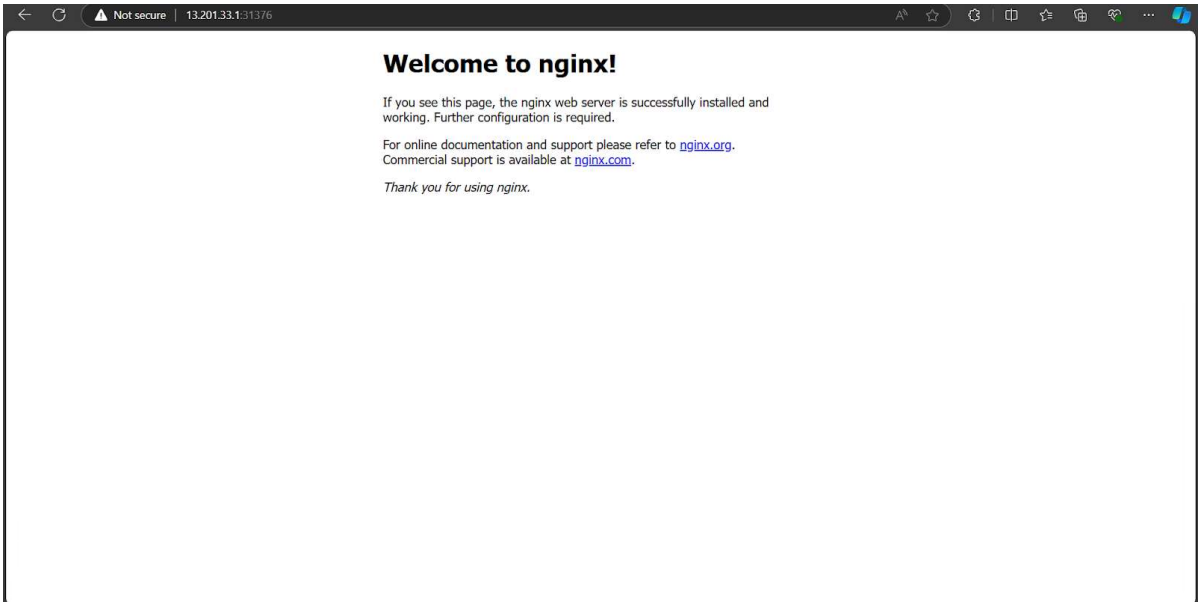
```
nginx-deployment   2/2     2            2           17m
ubuntu@ip-172-31-44-131:~$ nano nginx-services.yaml
ubuntu@ip-172-31-44-131:~$ nano nginx-service.yaml
ubuntu@ip-172-31-44-131:~$ kubectl apply -f nginx-service.yaml
service/nginx-service created
ubuntu@ip-172-31-44-131:~$ kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/TCP        71m
my-nginx        ClusterIP      10.111.168.255   <none>        80/TCP         23m
nginx-service   LoadBalancer   10.105.174.168   <pending>     80:31376/TCP   10s
ubuntu@ip-172-31-44-131:~$ kubectl port-forward service/nginx-service 8088:80
Forwarding from 127.0.0.1:8088 -> 80
Forwarding from [::1]:8088 -> 80
^Cubuntu@ip-172-31-44-131:~kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-8d7rx   1/1     Running   0          26m
nginx-deployment-86dcfdf4c6-bdbcm   1/1     Running   0          26m
ubuntu@ip-172-31-44-131:~$
```

Step 8: Access the Application Locally

1. **Open a Web Browser**: Now open your web browser and go to the following URL:

http://localhost:8080
You should see the application (in this case, Nginx) that you have deployed running in the Kubernetes cluster, served locally via port 8080.

In case the port 8080 is unavailable, try using a different port like 8081



EXPERIMENT NO. 5

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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

Theory :

Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It allows users to define and provision infrastructure using a high-level configuration language known as HashiCorp Configuration Language (HCL) or JSON. Terraform supports a wide range of cloud providers, such as AWS, Azure, Google Cloud, and on-premises solutions, enabling users to manage infrastructure across multiple environments consistently.

Core Concepts and Terminologies

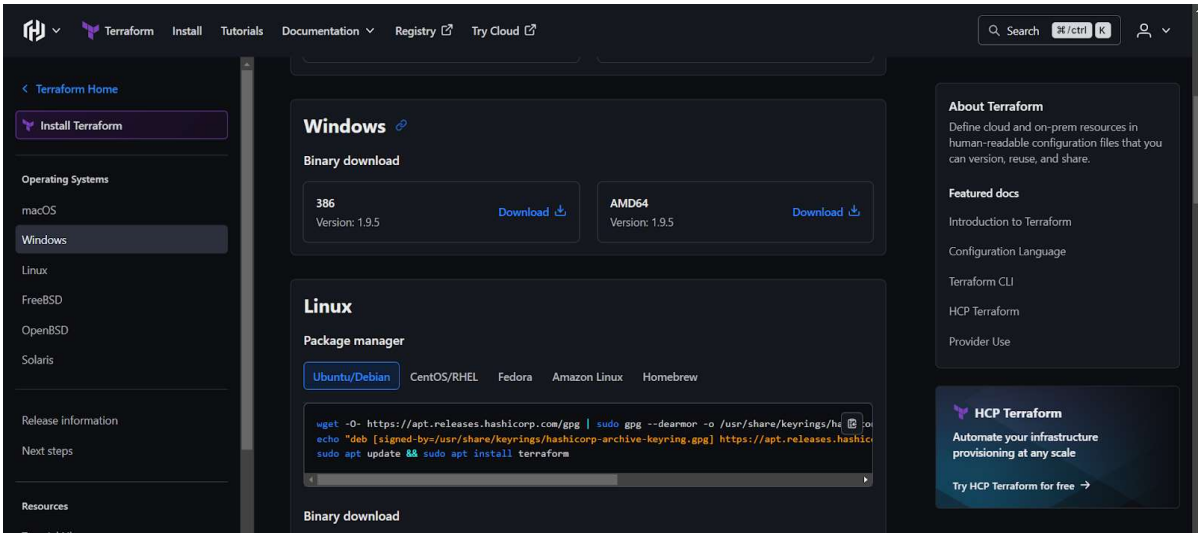
1. **Providers:**
   Providers are plugins that allow Terraform to interact with various APIs of cloud providers, SaaS providers, and other services. Each provider requires configuration and manages resources for that specific service.
2. **Resources:**
   Resources are the most fundamental elements in Terraform. They represent components of your infrastructure, such as virtual machines, databases, networks, and more.
3. **Modules:**
   Modules are containers for multiple resources that are used together. A module can call other modules, creating a hierarchical structure. This makes it easier to organize and reuse code.
4. **State:**
   Terraform maintains a state file that keeps track of the infrastructure managed by Terraform. The state file is crucial as it provides a mapping between the real-world resources and the configuration defined in Terraform.
5. **Variables:**
   Variables in Terraform are used to make configurations dynamic and reusable. They can be defined in the configuration files and assigned values at runtime.
6. **Outputs:**
   Outputs are used to extract information from the Terraform-managed infrastructure and display it after the execution of a Terraform plan or apply.

Terraform Lifecycle

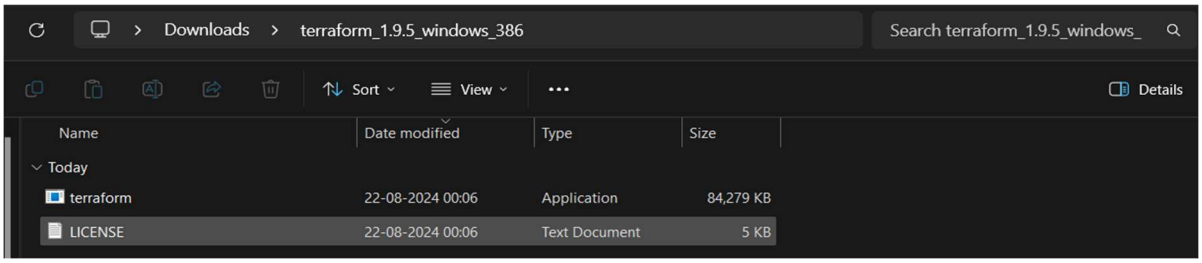
1. **Write:**
   Write the configuration file (typically with .tf extension) using HCL to describe the desired infrastructure.
2. **Initialize (terraform init):**
   Initialize the working directory containing the configuration files. This command downloads the necessary provider plugins and sets up the environment.
3. **Plan (terraform plan):**
   Terraform creates an execution plan based on the configuration files. It compares the current state with the desired state and shows the changes that will be made.
4. **Apply (terraform apply):**
   Apply the changes required to reach the desired state of the configuration. Terraform will prompt for confirmation before making any changes.
5. **Destroy (terraform destroy):**
   Destroy the infrastructure managed by Terraform. This command is used to remove all resources defined in the configuration files.

Implementation :

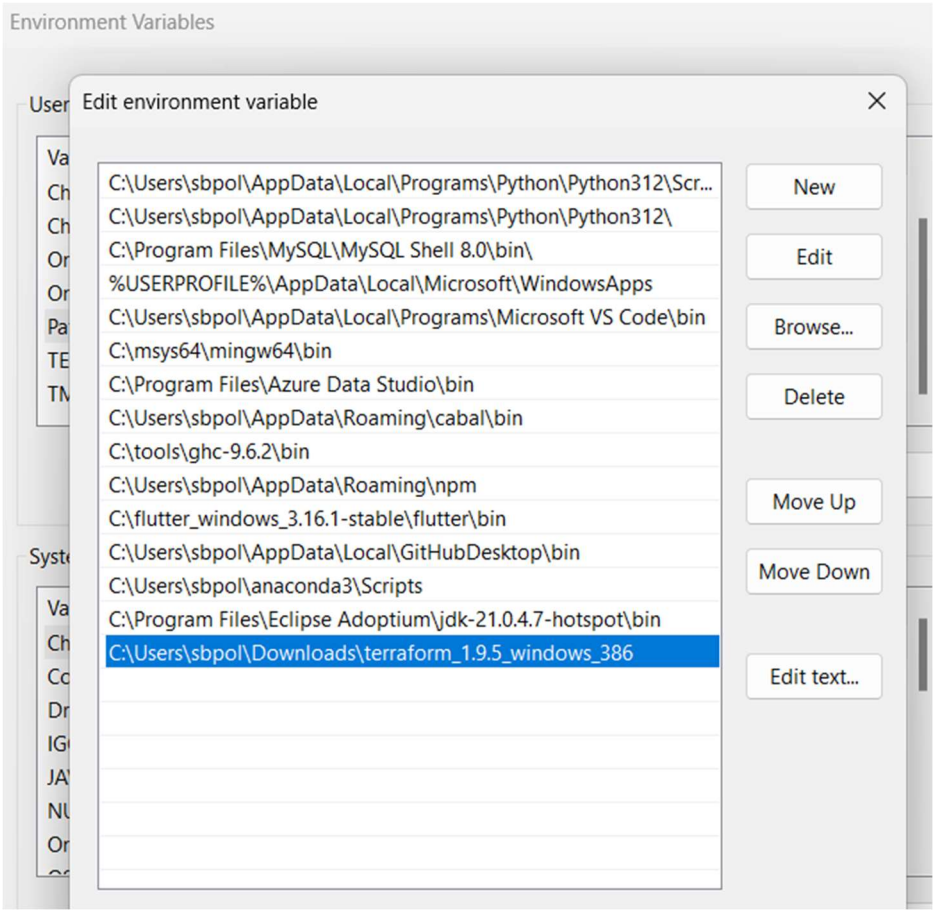
Step 1: Go to Terraform website and download 386 .



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



Step 3: Edit environment variables and paste copied path



Step 4 : check if the Terraform is installed correctly.

```
(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 and improvements! https://aka.ms/PSWindows

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

EXPERIMENT NO. 6

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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.

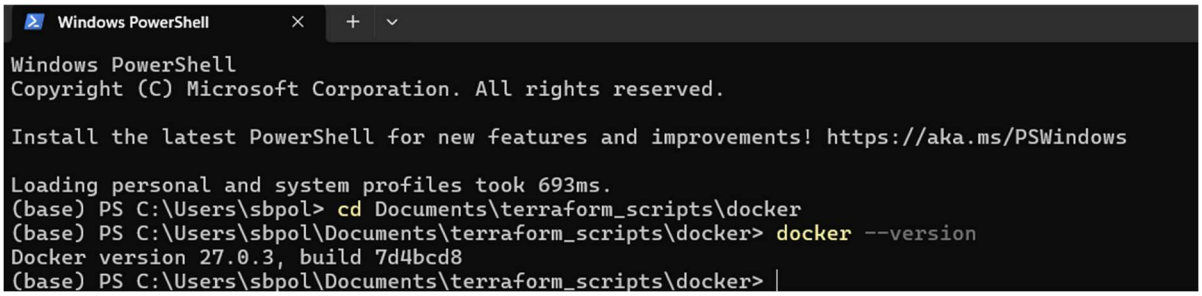
Benefits of Using Terraform for AWS Infrastructure

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 docker installation and version



Step 2 : create docker.tf file and write following code for terraform and docker

Code -

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}

provider "docker" {
  host    = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}
resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}
```

docker.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}

provider "docker" {
  host    = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}

```

Step 3 : Type terraform init command to initialize terraform backend

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 3.0.1"...
- Installing kreuzwerker/docker v3.0.2...
- Installed kreuzwerker/docker v3.0.2 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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(EXTRA) : type terraform fmt and validate commands .
The two Terraform commands – terraform validate and terraform fmt – are used to maintain a clean, error-free, and well-structured Terraform codebase.

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform fmt
docker.tf
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform validate
Success! The configuration is valid.
```

Step 5 : Type Terraform plan command to create execution plan .

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

  # 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
      + 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

      + 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.
```

Step 6 : Type terraform apply to apply changes .

```
(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 following symbols:
  + 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=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:lates
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=c25805e4484164520912c50ac3080526c9926219c98c673021078772eb484357]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker>
```

Step 7 : Docker container before and after step 6 execution

BEFORE -

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker container list
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform plan
```

AFTER -

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker container list
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                  NAMES
c25805e44841   5ef79149e0ec   "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:8000->80/tcp   tutorial
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
nginx        latest    5ef79149e0ec   6 days ago   188MB
```

Step 8 (EXTRA) : Execution of change .

docker.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}

provider "docker" {
  host = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8080
  }
}

```

```
          + publish_all_ports                            = (known after apply)
          + read_only                                    = (known after apply)
          + remove_volumes                               = (known after apply)
          + restart                                      = (known after apply)
          + rm                                           = (known after apply)
          + runtime                                      = (known after apply)
          + security_opts                                = (known after apply)
          + shm_size                                     = (known after apply)
          + start                                        = (known after apply)
          + stdin_open                                   = (known after apply)
          + stop_signal                                  = (known after apply)
          + stop_timeout                                 = (known after apply)
          + storage_opts                                 = (known after apply)
          + sysctls                                      = (known after apply)
          + tmpfs                                        = (known after apply)
          + tty                                          = (known after apply)
          + user                                         = (known after apply)
          + userns_mode                                  = (known after apply)
          + wait                                         = (known after apply)
          + wait_timeout                                 = (known after apply)
          + working_dir                                  = (known after apply)
        } -> (known after apply)

      ~ ports {
          ~ external = 8000 -> 8080 # forces replacement
            # (3 unchanged attributes hidden)
        }
    }

Plan: 1 to add, 0 to change, 1 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_container.nginx: Destroying... [id=c25805e44841645209l2c50ac3080526c9926219c98c6730210787721eb484357]
docker_container.nginx: Destruction complete after 1s
docker_container.nginx: Creating...
```

Step 9 : terraform destroy to destroy infrastructure.

```
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:latest]
docker_container.nginx: Refreshing state... [id=c648cc3dd8129abf9acb7cb06dfdd0aa9bafb0c7973f16695cd06a7ad447c631]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - 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" -> null
      - image                                       = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c" -> null
      - 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
      - must_run                                    = true -> null
```

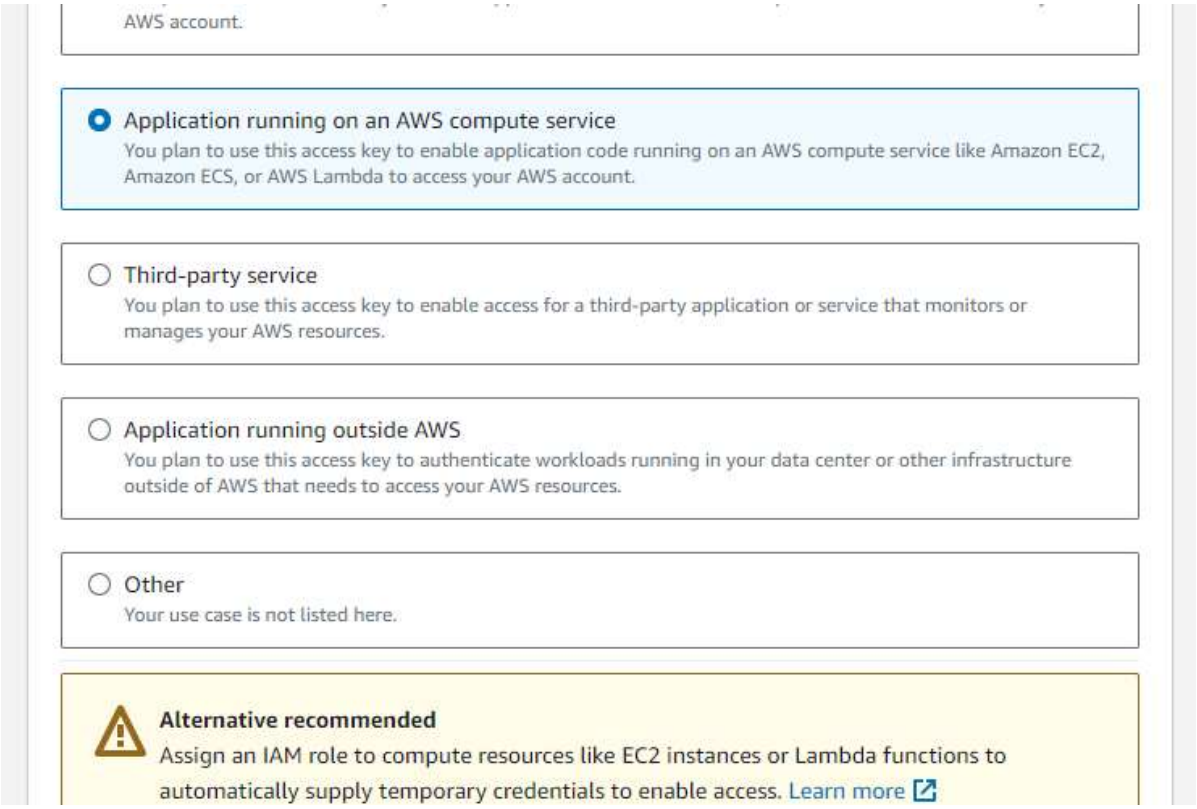
```
      - stop_timeout                              = 0 -> null
      - storage_opts                              = {} -> null
      - sysctls                                   = {} -> null
      - tmpfs                                     = {} -> null
      - tty                                       = 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" -> null
      - 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=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_image.nginx: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker>
```

Step 10 : Docker after destroy command.

```
Destroy complete! Resources: 2 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker container list
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker>
```

Terraform and S3 -

Step 1: Create access keys and secret key for IAM user



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

Code -

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  access_key = ""
  secret_key = ""
}
resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"
```

```
  tags = {
    Name = "My bucket"

  }
}
```

s3 > main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  access_key = ""
  secret_key = ""
}



resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"

  tags = {
    Name = "My bucket"

  }
}
```

Step 3 : Type terraform init command in powershell.

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.63.1...
- Installed hashicorp/aws v5.63.1 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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 : Type terraform plan command in powershell.

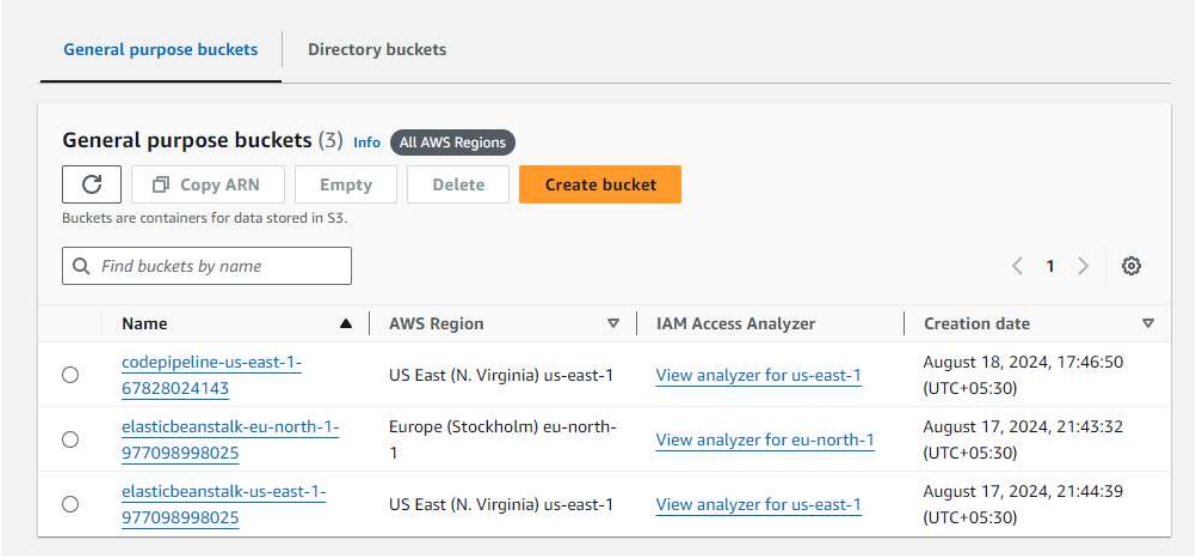
```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform plan

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

      + lifecycle_rule (known after apply)

      + logging (known after apply)
```

Step 5 : Type terraform apply command in powershell.

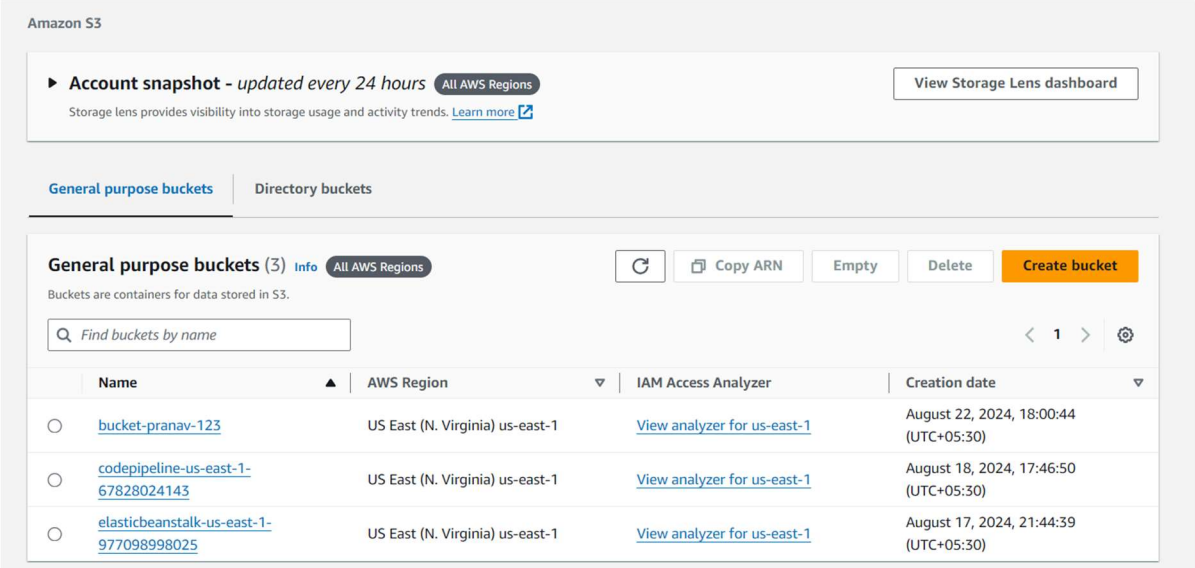
```
      + versioning (known after apply)

      + website (known after apply)
    }

Plan: 1 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.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform apply

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_s3_bucket.bucket will be created
  + resource "aws_s3_bucket" "bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "bucket-pranav-123"
      + 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                        = {
          + "Name" = "My bucket"
        }
      + tags_all                    = {
          + "Name" = "My bucket"
        }
      + website_domain              = (known after apply)
```

```
      }
    + tags_all                    = {
        + "Name" = "My bucket"
      }
    + website_domain              = (known after apply)
    + website_endpoint            = (known after apply)

    + cors_rule (known after apply)

    + grant (known after apply)

    + lifecycle_rule (known after apply)

    + logging (known after apply)

    + object_lock_configuration (known after apply)

    + replication_configuration (known after apply)

    + server_side_encryption_configuration (known after apply)

    + 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.bucket: Creating...
aws_s3_bucket.bucket: Creation complete after 5s [id=bucket-pranav-123]

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

Step 6 : AWS s3 before and after the bucket creation using terraform.
BEFORE -



AFTER -



Step 7(EXTRA) : Upload file to the bucket using terraform .

CODE -

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  access_key = ""
  secret_key = ""
}

resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"

  tags = {
```

```
    Name = "My bucket"

  }
}

resource "aws_s3_bucket_object" "file" {
  bucket = aws_s3_bucket.bucket.id
  key    = "hello.txt"
  source = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.txt"

}
```

```
resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"

  tags = {
    Name = "My bucket"

  }
}

resource "aws_s3_bucket_object" "file" {
  bucket = aws_s3_bucket.bucket.id
  key    = "hello.txt"
  source = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.txt"

}
```

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

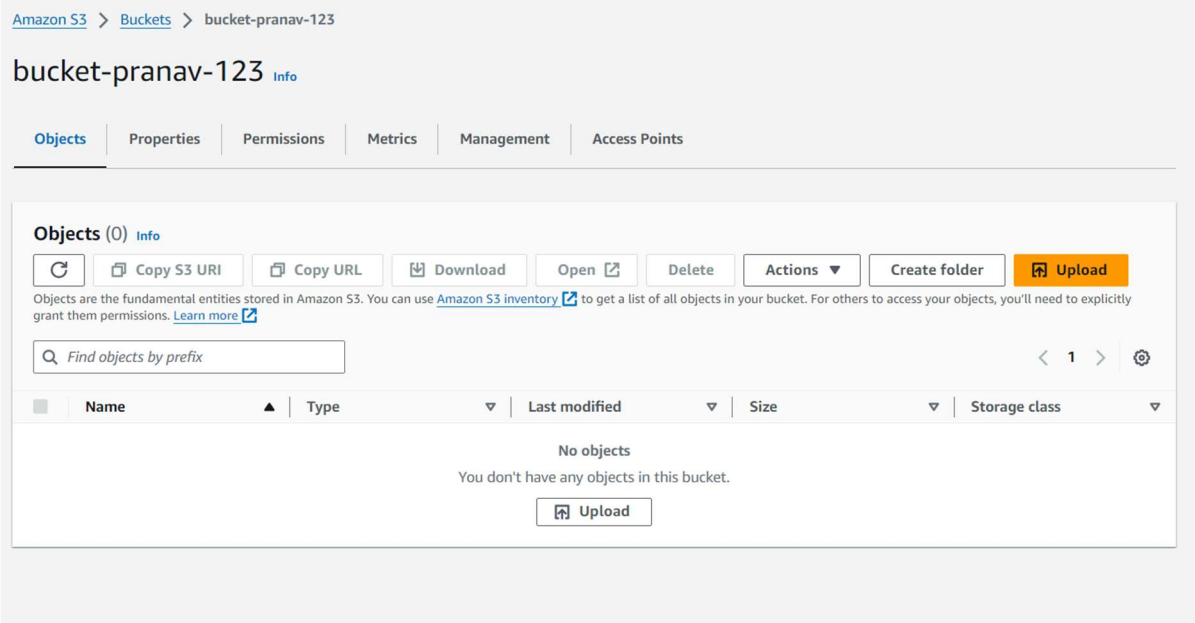
```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform plan
aws_s3_bucket.bucket: Refreshing state... [id=bucket-pranav-123]

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_object.file will be created
  + resource "aws_s3_bucket_object" "file" {
      + acl                    = "private"
      + arn                    = (known after apply)
      + bucket                 = "bucket-pranav-123"
      + bucket_key_enabled     = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "hello.txt"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.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.

  Warning: Deprecated Resource

    with aws_s3_bucket_object.file,
    on main.tf line 28, in resource "aws_s3_bucket_object" "file":
    28: resource "aws_s3_bucket_object" "file" {

  use the aws_s3_object resource instead

  (and one more similar warning elsewhere)
```

```
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" no
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform apply
aws_s3_bucket.bucket: Refreshing state... [id=bucket-pranav-123]

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_s3_bucket_object.file will be created
  + resource "aws_s3_bucket_object" "file" {
      + acl                    = "private"
      + arn                    = (known after apply)
      + bucket                 = "bucket-pranav-123"
      + bucket_key_enabled     = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "hello.txt"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.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.

  Warning: Deprecated Resource

    with aws_s3_bucket_object.file,
    on main.tf line 28, in resource "aws_s3_bucket_object" "file":
    28: resource "aws_s3_bucket_object" "file" {

  use the aws_s3_object resource instead

  (and one more similar warning elsewhere)
```

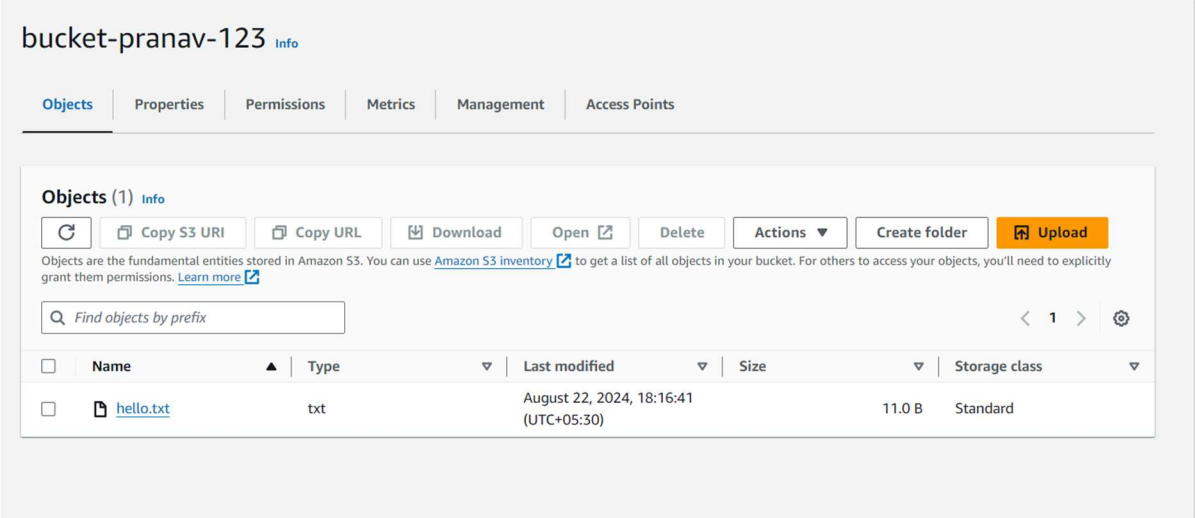
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket_object.file: Creating...
aws_s3_bucket_object.file: Creation complete after 1s [id=hello.txt]

Warning: Deprecated Resource

  with aws_s3_bucket_object.file,
  on main.tf line 28, in resource "aws_s3_bucket_object" "file":
  28: resource "aws_s3_bucket_object" "file" {

use the aws_s3_object resource instead

Warning: Argument is deprecated

  with aws_s3_bucket_object.file,
  on main.tf line 29, in resource "aws_s3_bucket_object" "file":
  29:   bucket = aws_s3_bucket.bucket.id

Use the aws_s3_object resource instead

(and one more similar warning elsewhere)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3>
```

Step 9(EXTRA) : s3 bucket before and after execution of upload

BEFORE -



AFTER -

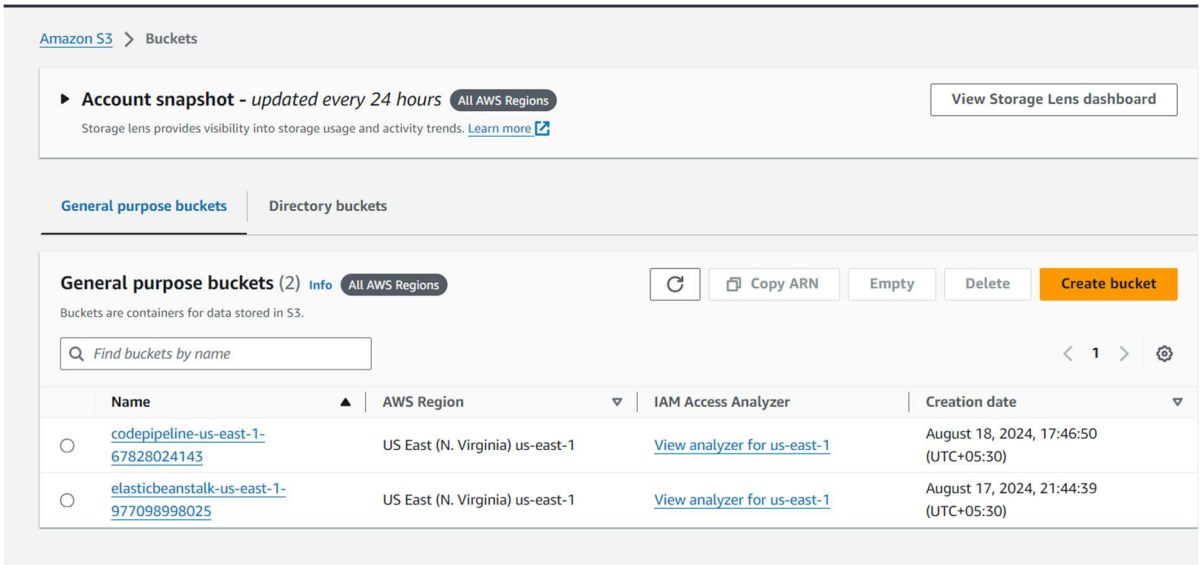


Step 10 : Terraform destroy command to destroy the s3 bucket.

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform destroy
aws_s3_bucket.bucket: Refreshing state... [id=bucket-pranav-123]
aws_s3_bucket_object.file: Refreshing state... [id=hello.txt]

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

Terraform will perform the following actions:

  # aws_s3_bucket.bucket will be destroyed
  - resource "aws_s3_bucket" "bucket" {
      - arn                         = "arn:aws:s3:::bucket-pranav-123" -> null
      - bucket                      = "bucket-pranav-123" -> null
      - bucket_domain_name          = "bucket-pranav-123.s3.amazonaws.com" -> null
      - bucket_regional_domain_name = "bucket-pranav-123.s3.us-east-1.amazonaws.com" -> null
      - force_destroy               = false -> null
      - hosted_zone_id              = "Z3AQBSTGFYJSTF" -> null
      - id                          = "bucket-pranav-123" -> null
      - object_lock_enabled         = false -> null
      - region                      = "us-east-1" -> null
      - request_payer               = "BucketOwner" -> null
      - tags                        = {
          - "Name" = "My bucket"
        } -> null
      - tags_all                    = {
          - "Name" = "My bucket"
        } -> null
        # (3 unchanged attributes hidden)

      - grant {
          - id          = "10def03d73e09d8adda11bfe68e632f70a83a37758b74ea6e933dafd0250c850" -> null
          - permissions = [
              - "FULL_CONTROL",
            ] -> null
          - type        = "CanonicalUser" -> null
            # (1 unchanged attribute hidden)
        }

      - server_side_encryption_configuration {
```

```
            }
        }

      - versioning {
          - enabled    = false -> null
          - mfa_delete = false -> null
        }
    }

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

  Warning: Deprecated Resource

    with aws_s3_bucket_object.file,
    on main.tf line 28, in resource "aws_s3_bucket_object" "file":
    28: resource "aws_s3_bucket_object" "file" {

  use the aws_s3_object resource instead


  Warning: Argument is deprecated

    with aws_s3_bucket_object.file,
    on main.tf line 30, in resource "aws_s3_bucket_object" "file":
    30:   key    = "hello.txt"

  Use the aws_s3_object resource instead


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

aws_s3_bucket.bucket: Destroying... [id=bucket-pranav-123]
aws_s3_bucket.bucket: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3>
```

Step 11: s3 after the destroy command execution .



Hosting Website on s3 using Terraform (EXTRA) -

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" "rand_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "mywebappp-bucket" {
  bucket = "mywebappp-bucket-${random_id.rand_id.hex}"
}

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

resource "aws_s3_object" "styles_css" {
  bucket       = aws_s3_bucket.mywebappp-bucket.bucket
  source       = "./styles.css"
  key          = "styles.css"
  content_type = "text/css"
```

```
}

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

resource "aws_s3_bucket_policy" "mywebappp" {
  bucket = aws_s3_bucket.mywebappp-bucket.id

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


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

  index_document {
    suffix = "index.html"
  }
}
```

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

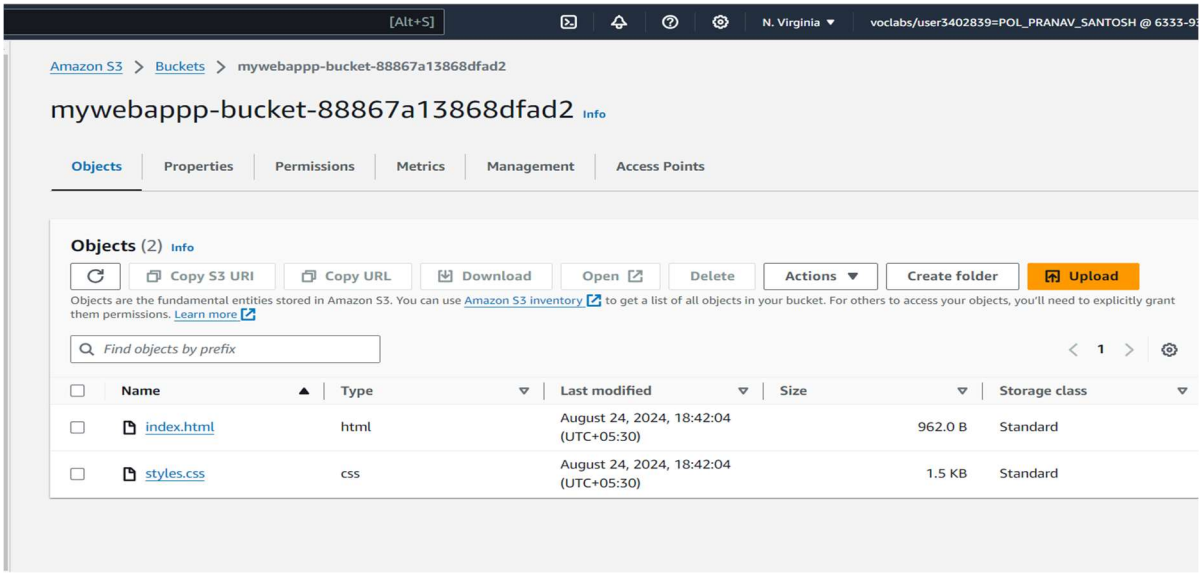
Step 2 : Create Provider.tf and write following code
Code -

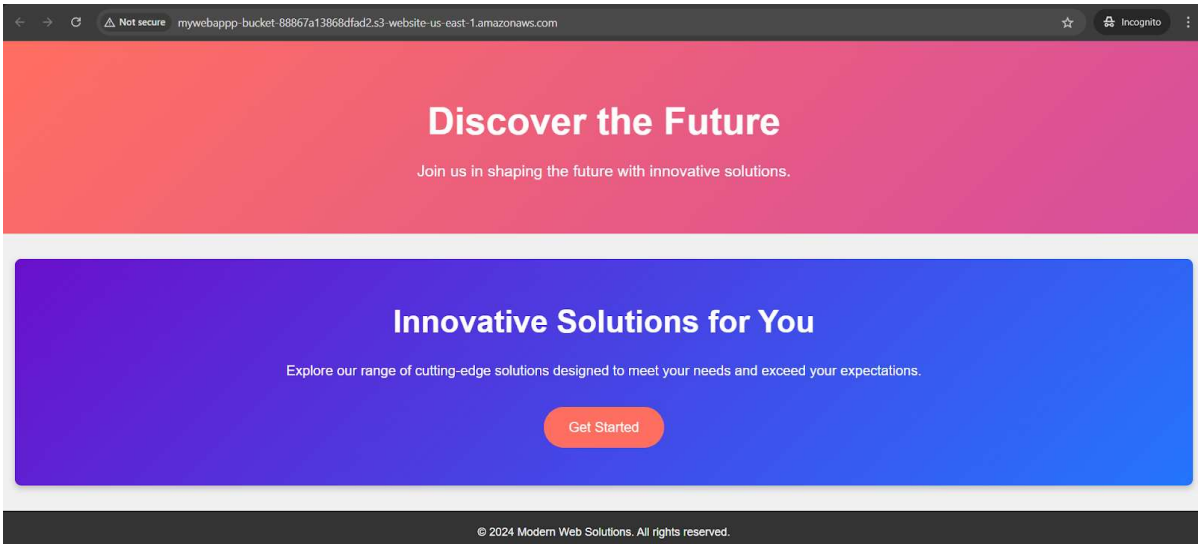
```
provider "aws" {
  access_key="ASIAZG6JVYHRLQ7XABVF"
  secret_key="FV+B+/JDLgRHpPs2bLr9jB+835PQ4cyz7HQ4LAzR"

token="IQoJb3JpZ2luX2VjELT/////////wEaCXVzLXdlc3QtMiJGMEQCIGM45rz6GOs
ZBjBcMcCWfAJetwP1F2qgToQCSoJbLE+HAiB2t1XfLcQY0BFOSBsbvJwCmQQ1vQ
6/5m4YmzBC1rRelCq1Agi9/////////8BEAIaDDYzMzM5Mzc1ODY5MCIM3vgTOnS9
B6JyQQmeKokCJkhMaeK5NcXazpFuqObvIOQpIjKOVtHR/NwxdQCrfqPa2qbn+VsG9
i7tF0pvxniO/OQmqxXXaNlRjnq2QomydAte/91VXJ1cqT7R7k/06lSBc2AVcSAJfgAY
EIB7kKVF2UkY01VJ845VjTPnER7O4enKd5jYyHakuOkj29olSph1sjrq6VFYBo0foLgL
JcDsL/QbipTk8HXX7XT8f/Gh8jGKfUjy2CUvJfuAAX3zvsTFjSsGEb69J1pZd0sQfoBG
i6Mv0vezW+IjWX+dLdpnzDEJrnk0x7g6po1uXrCjDF6+pB+5QwPhI78D2IF/tcLahLbr5
El6ri2DXv0eQ0woOaL6u0xsKDPvwzDCkqe2BjqeAYi5Fs7WB0Ei5FiAqHdJEzXcQZI1
8JX5H59W3p+v71sN7sGLxJYrXoMmFLH7amaZxQ7r5xkn9/is6Ge3ZcuxROIy5GOLu
qoHVsNRxCRQ83ZoIewd32TRN8h3uRLQnE7ZMf6ggljBvqvT1e2IlA+YcdeWrkeM/fC
XJ0g7kKEcnkNgBMv+W9LXi2P8DMsm0AnP6jhFK5R6Ch16JI+ePiL1"
  region="us-east-1"
}
```

Step 3: Execute Terraform init , terraform plan and terraform apply command.

```
Terraform will perform the following actions:

  # aws_s3_bucket_policy.mywebappp will be created
  + resource "aws_s3_bucket_policy" "mywebappp" {
      + bucket = "mywebappp-bucket-88867a13868dfad2"
      + id     = (known after apply)
      + policy = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "s3:GetObject"
                      + Effect    = "Allow"
                      + Principal = "*"
                      + Resource  = "arn:aws:s3:::mywebappp-bucket-88867a13868dfad2/*"
                      + Sid       = "PublicReadGetObject"
                    },
                ]
              + 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.mywebappp: Creating...
aws_s3_bucket_policy.mywebappp: Creation complete after 2s [id=mywebappp-bucket-88867a13868dfad2]

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

Outputs:

website_endpoint = "mywebappp-bucket-88867a13868dfad2.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

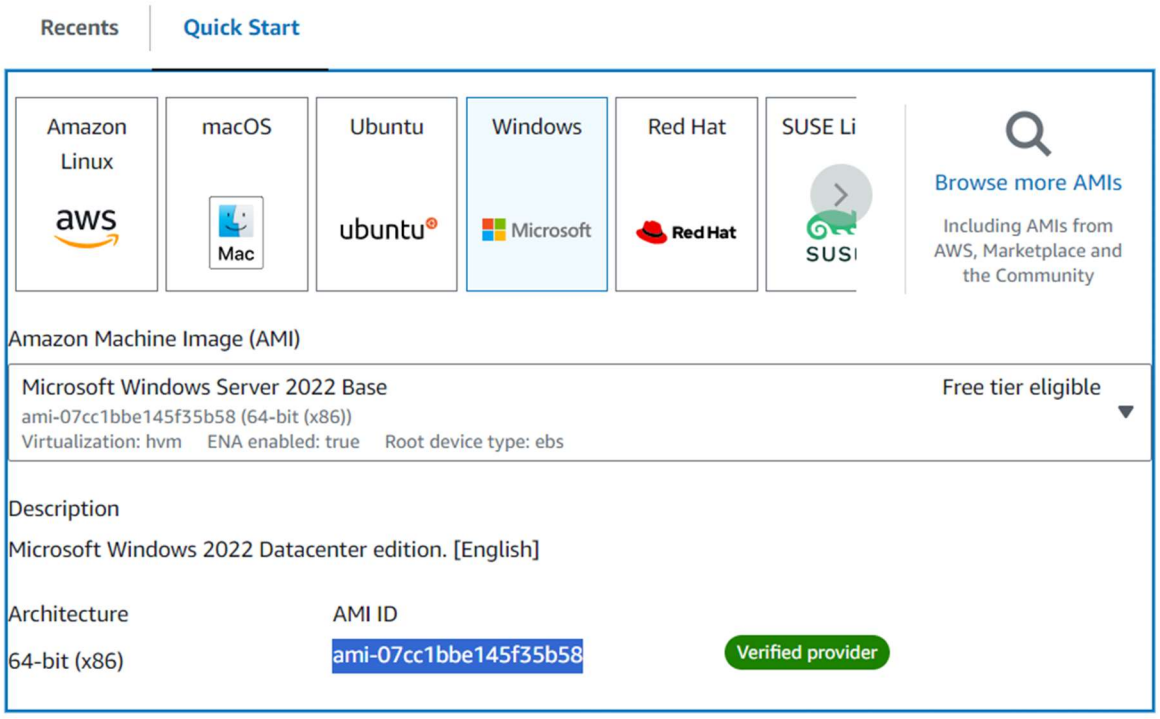
```
  # random_id.rand_id will be destroyed
  - resource "random_id" "rand_id" {
      - b64_std     = "iIZ6E4aN+tI=" -> null
      - b64_url     = "iIZ6E4aN-tI" -> null
      - byte_length = 8 -> null
      - dec         = "9837684660317846226" -> null
      - hex         = "88867a13868dfad2" -> null
      - id          = "iIZ6E4aN-tI" -> null
    }

Plan: 0 to add, 0 to change, 7 to destroy.

Changes to Outputs:
  - website_endpoint = "mywebappp-bucket-88867a13868dfad2.s3-website-us-east-1.amazonaws.com" -> null

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

aws_s3_bucket_policy.mywebappp: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_bucket_public_access_block.example: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_bucket_website_configuration.example: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_object.index_html: Destroying... [id=index.html]
aws_s3_object.styles_css: Destroying... [id=styles.css]
aws_s3_object.index_html: Destruction complete after 1s
aws_s3_object.styles_css: Destruction complete after 1s
aws_s3_bucket_website_configuration.example: Destruction complete after 1s
aws_s3_bucket_public_access_block.example: Destruction complete after 1s
aws_s3_bucket_policy.mywebappp: Destruction complete after 2s
aws_s3_bucket.mywebappp-bucket: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_bucket.mywebappp-bucket: Destruction complete after 0s
random_id.rand_id: Destroying... [id=iIZ6E4aN-tI]
random_id.rand_id: Destruction complete after 0s

Destroy complete! Resources: 7 destroyed.

C:\Users\sbpol\Documents\terraform_scripts\docker\siteHosting>
```

Creating EC2 instance using Terraform (EXTRA) -

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+ljWX+dLdpnzDEJrnk0x7g6po1uXrCjDF6+pB+5QwPhI78D2lF/tcLahLbr5El6r
i2DXv0eQ0woOaL6u0xsKDPvwzDCkqe2BjqeAYi5Fs7WB0Ei5FiAqHdJEzXcQZI18JX5H59W3p+v71sN7sGLxJYrXoMmFLH7amaZxQ7r5xkn9/is6Ge3Zcu
xROIy5GOLuqoHVsNRxCRQ83ZoIewd32TRN8h3uRLQnE7ZMf6ggljBvqvT1e2I1A+YcdeWrkeM/fCXJ0g7kKEcnkNgBMv+W9LXi2P8DMsm0AnP6jhFK5R6C
h16JI+ePiL1"
```

Step 2 : copy the AMI ID from the EC2



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

provider.tf
```
provider "aws" {
  access_key="ASIAZG6JVYHRLQ7XABVF"
  secret_key="FV+B+/JDLgRHpPs2bLr9jB+835PQ4cyz7HQ4LAzR"
  token="IQoJb3JpZ2luX2VjELT/////////wEaCXVzLXdlc3QtMiJGMEQCIGM45rz6GOsZBjBcMcCWfAJetwP1F2qgToQCSoJbLE+HAiB2t1XfLcQY0BFOSBsbvJwCmQQ1vQ6/5m4
  region="us-east-1"
}
```

main.tf
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}


resource "aws_instance" "myServer" {
  ami = "ami-07cc1bbe145f35b58"
  instance_type = "t2.micro"
    tags = {
        Name = "my Server"
    }
}
```

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

```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- 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 the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.

C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>
```

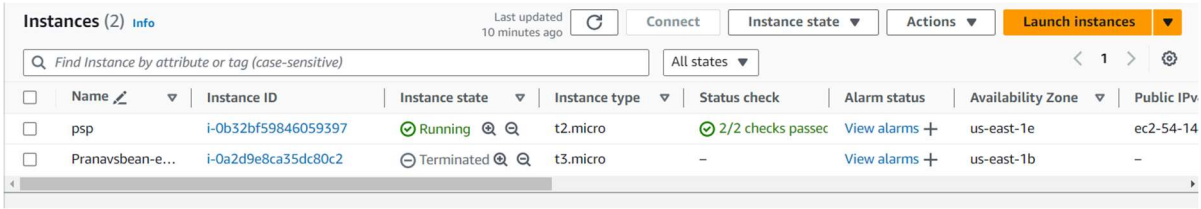
```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform plan

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)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
```

```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the follo
  + 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)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
```

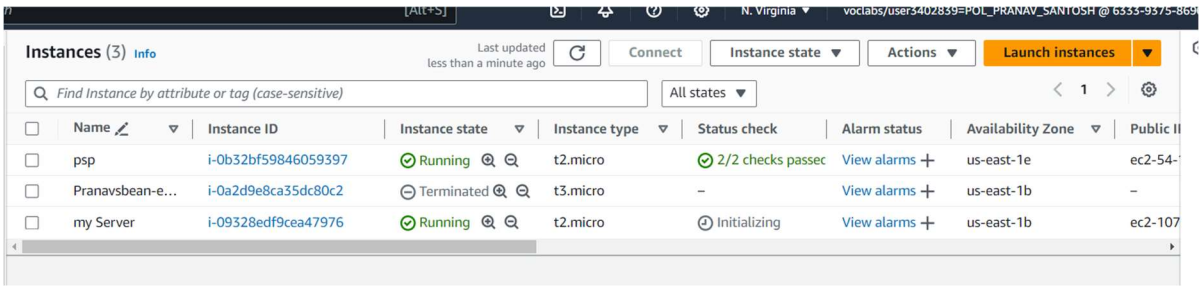
```
Plan: 1 to add, 0 to change, 0 to destroy.

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: Creation complete after 18s [id=i-09328edf9cea47976]

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

Step 5 : Ec2 before and after instance creation .

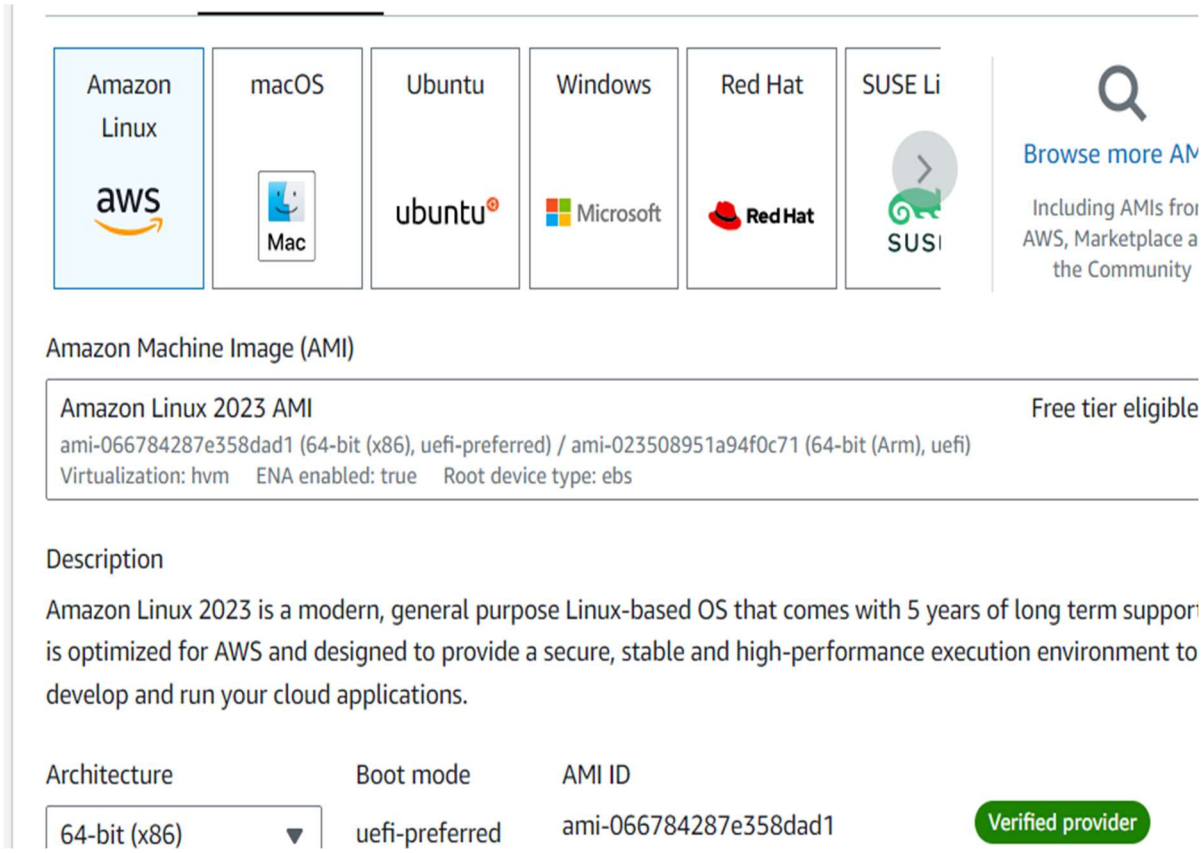
BEFORE -



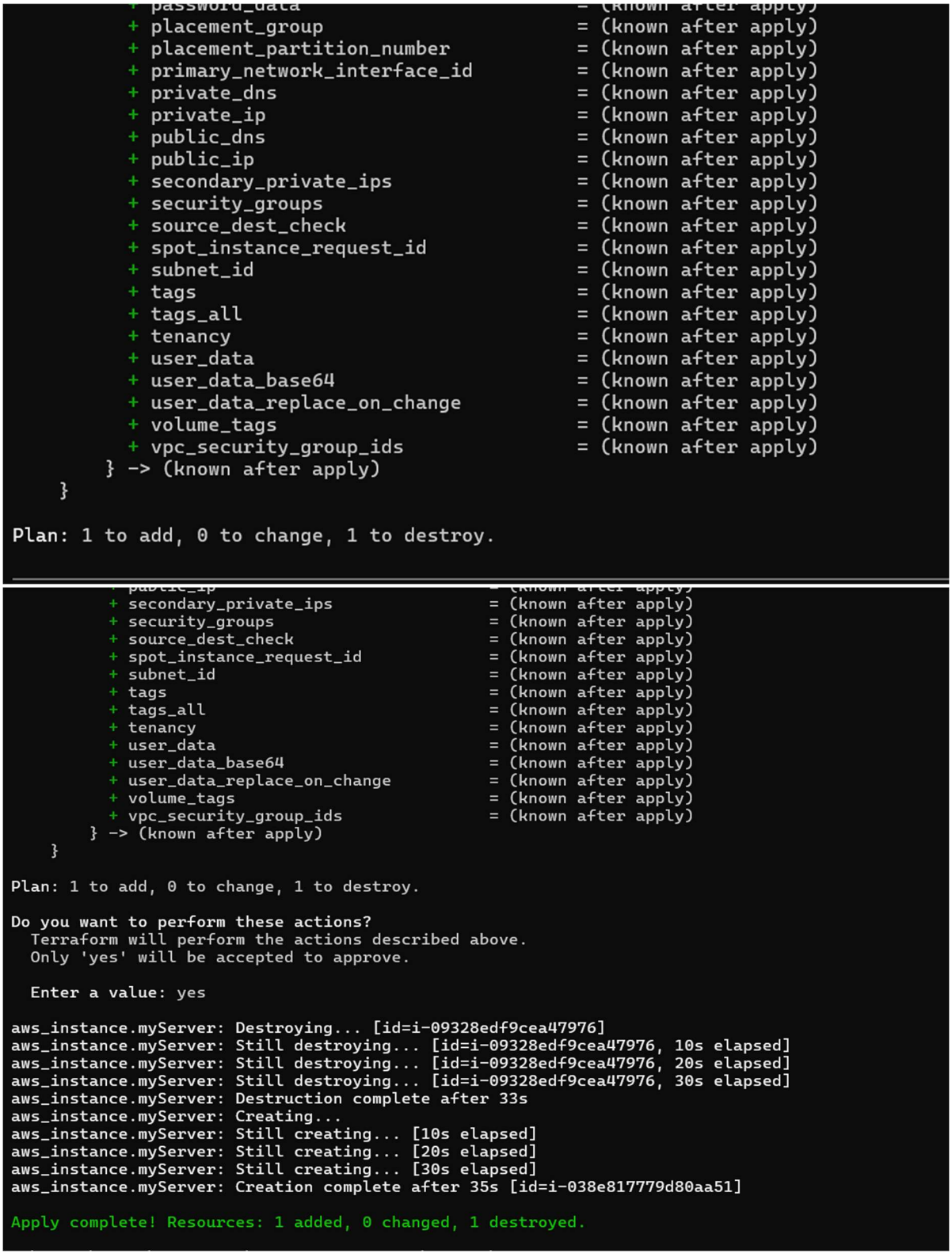
AFTER -



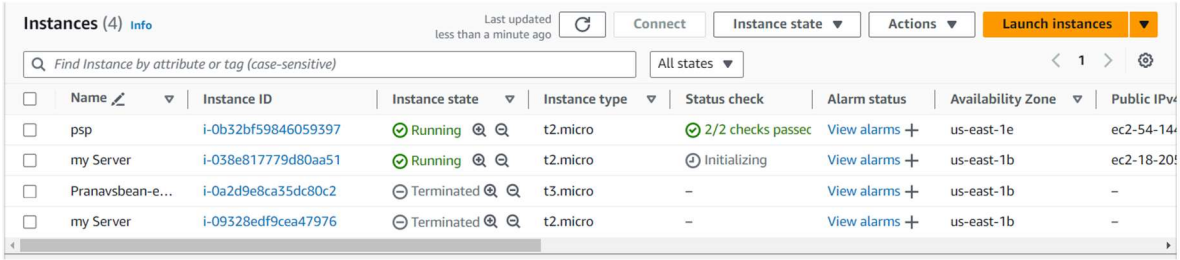
Step 6 : Copy AWS AMI ID and change it in code



Step 7 : Type terraform plan and terraform apply command.



Step 8 : Instances after deleting window instance and creating AWS instance



Step 9 : Destroy the instance using terraform destroy

```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform destroy
aws_instance.myServer: Refreshing state... [id=i-038e817779d80aa51]

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

Terraform will perform the following actions:

  # aws_instance.myServer will be destroyed
  - resource "aws_instance" "myServer" {
      - ami                                  = "ami-066784287e358dad1" -> null
      - arn                                  = "arn:aws:ec2:us-east-1:633393758690:instance/i-038e817779d80aa51" -
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "us-east-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-038e817779d80aa51" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-0c93e7a6f650aaacb" -> null
      - private_dns                          = "ip-172-31-84-36.ec2.internal" -> null
      - private_ip                           = "172.31.84.36" -> null
      - public_dns                           = "ec2-18-205-116-164.compute-1.amazonaws.com" -> null
      - public_ip                            = "18.205.116.164" -> null
      - secondary_private_ips                = [] -> null
      - security_groups                      = [
          - "default",
        ] -> null
      - source_dest_check                    = true -> null
    }

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 shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.myServer: Destroying... [id=i-038e817779d80aa51]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 10s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 20s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 30s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 40s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 50s elapsed]
aws_instance.myServer: Destruction complete after 53s

Destroy complete! Resources: 1 destroyed.
```

EXPERIMENT NO. 7

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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:

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

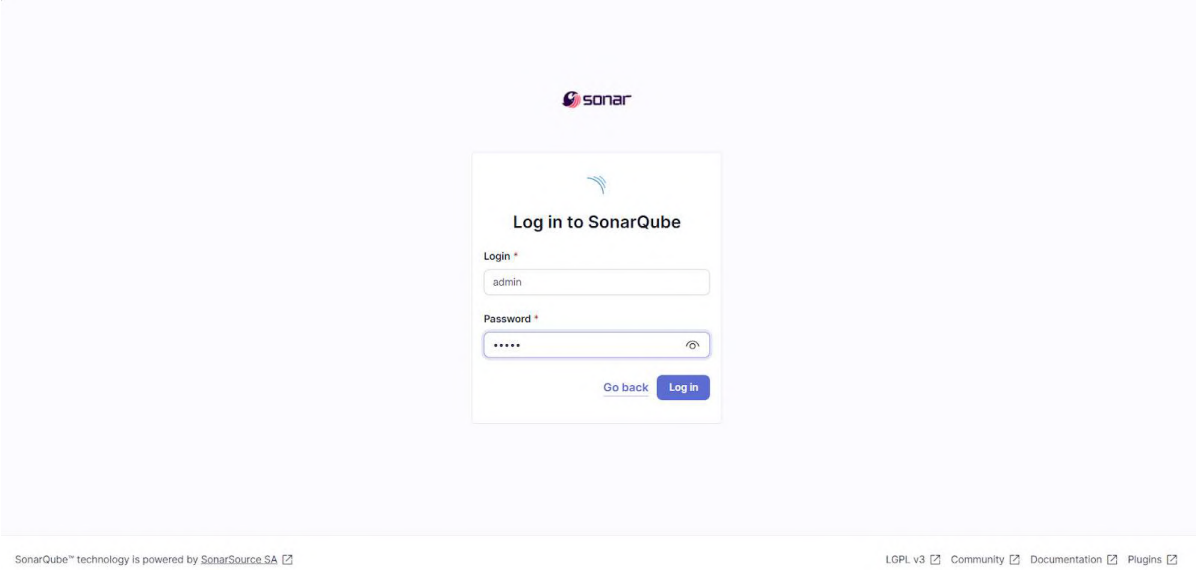
3. Check SonarQube Status

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

4. Login to SonarQube

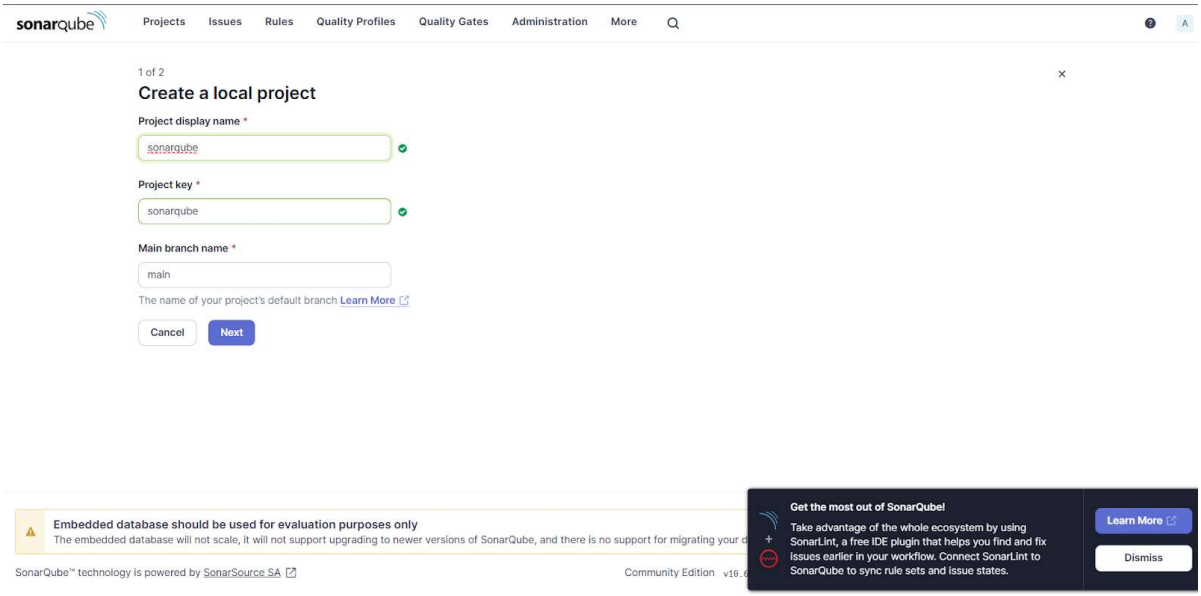
- Use the default credentials to log in:
    - **Username:** admin

- **Password:** admin



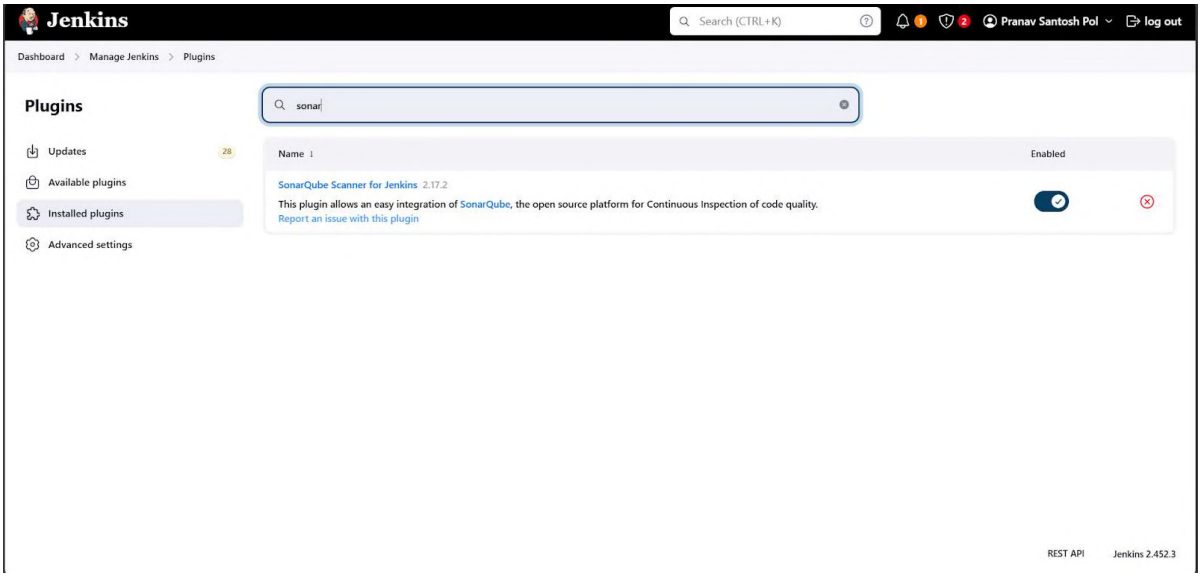
5. Create a Project in SonarQube

- Create a new project manually in SonarQube and name it sonarqube.



6. 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.



7. Configure SonarQube in Jenkins

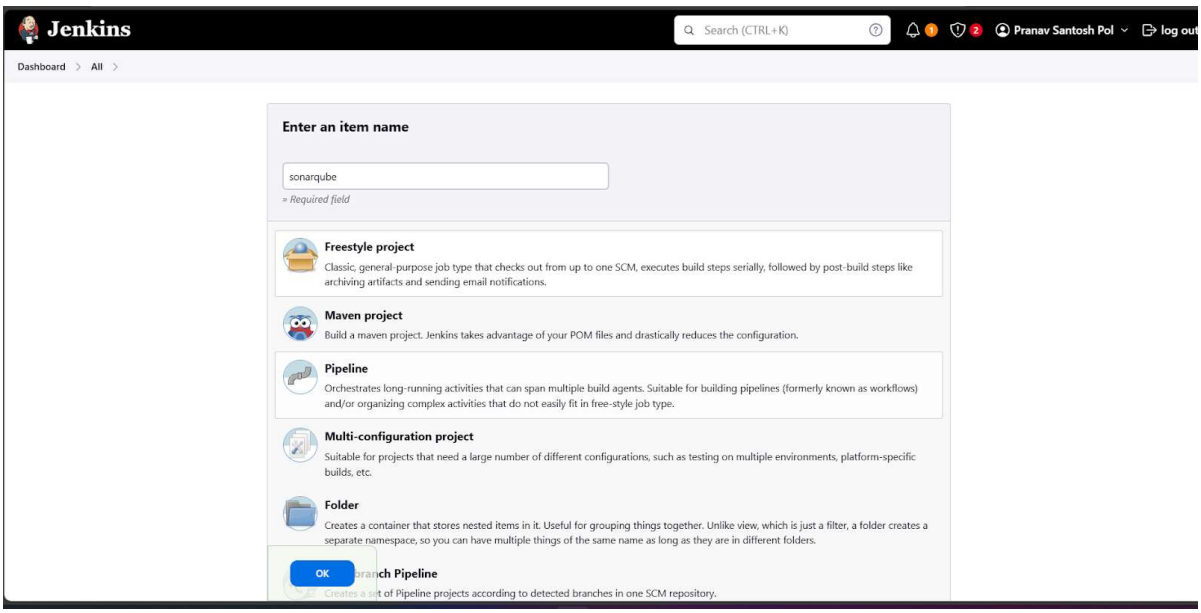
- Go to Manage Jenkins > Configure System.
- Scroll down to the SonarQube Servers section and enter the required details:
    - **Name:** Any name you prefer.
    - **Server URL:** http://localhost:9000
    - **Server Authentication Token:** (Generate this token in SonarQube under My Account > Security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

8. 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

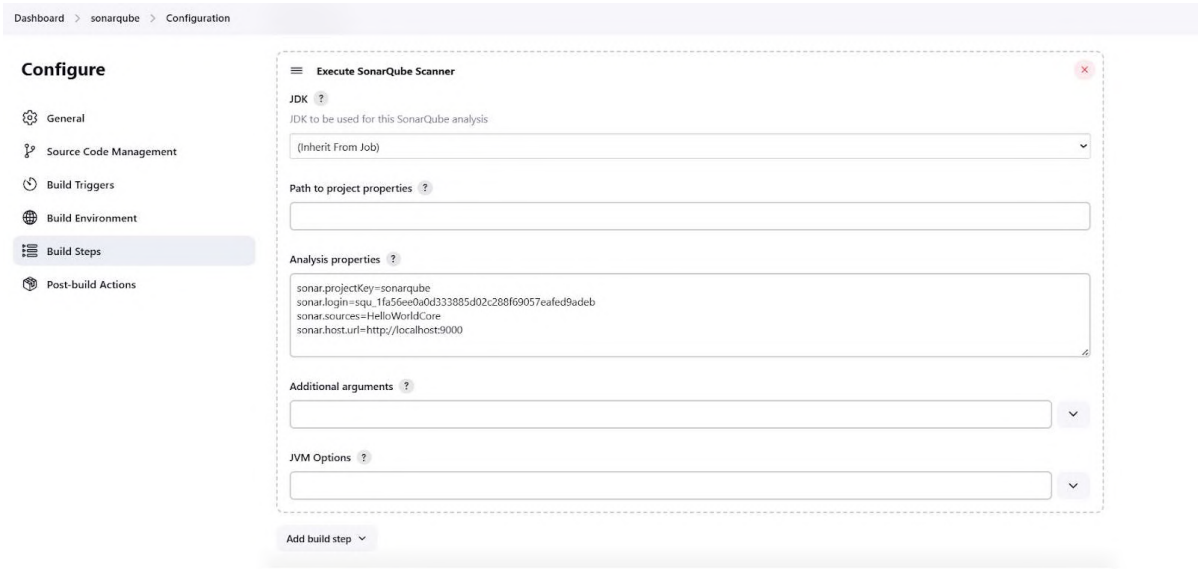
9. Create a New Jenkins Job

- 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



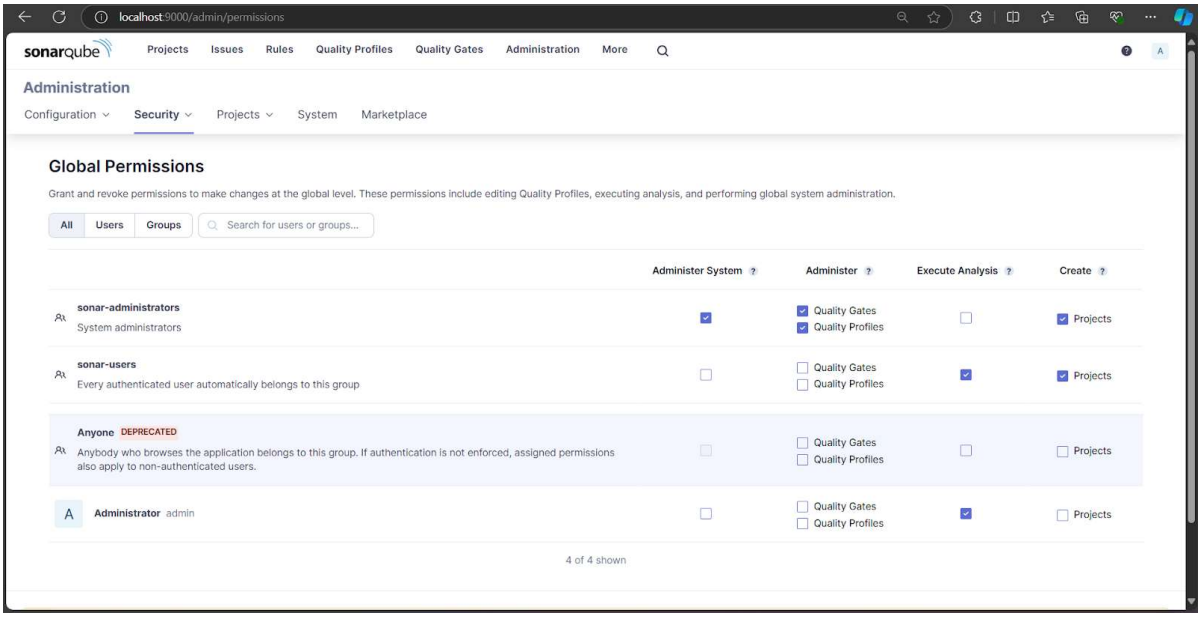
10. Configure Build Steps

- 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



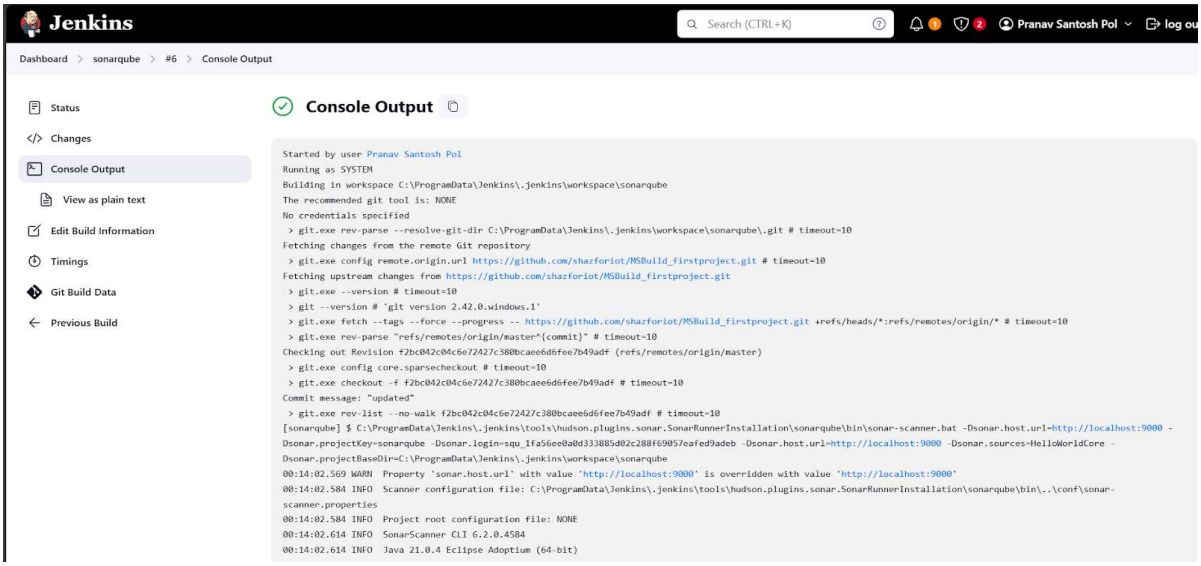
11. Set Permissions in SonarQube

- Navigate to http://localhost:9000/<user_name>/permissions.
- Allow Execute Permissions to the Admin user.



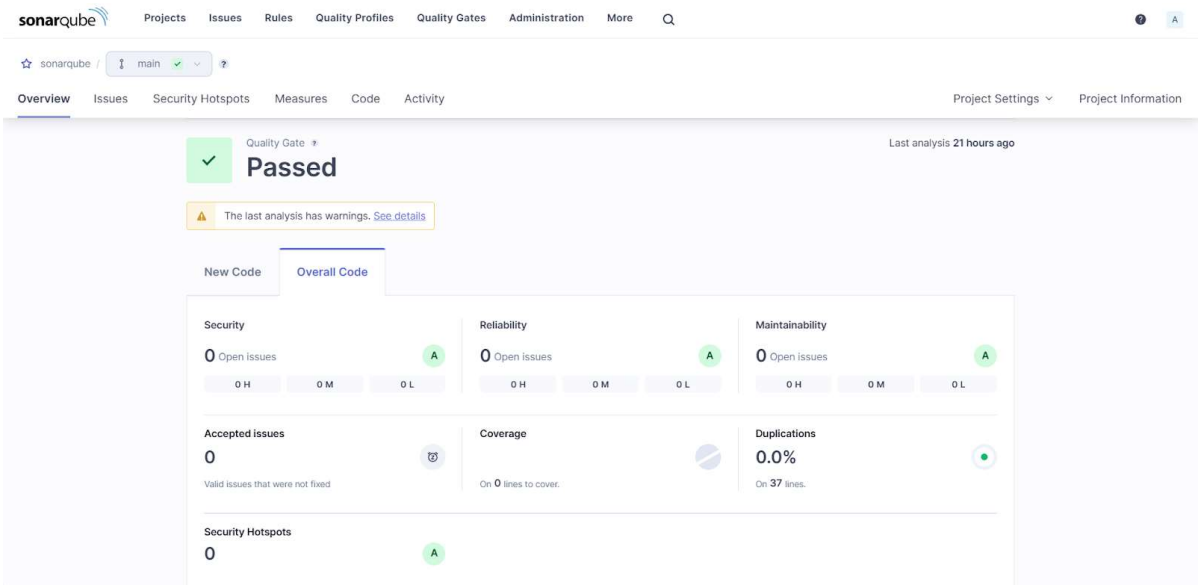
12. Run the Build

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



13. 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 NO. 8

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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 Docker

Run the following command to start SonarQube in a Docker container:

command :

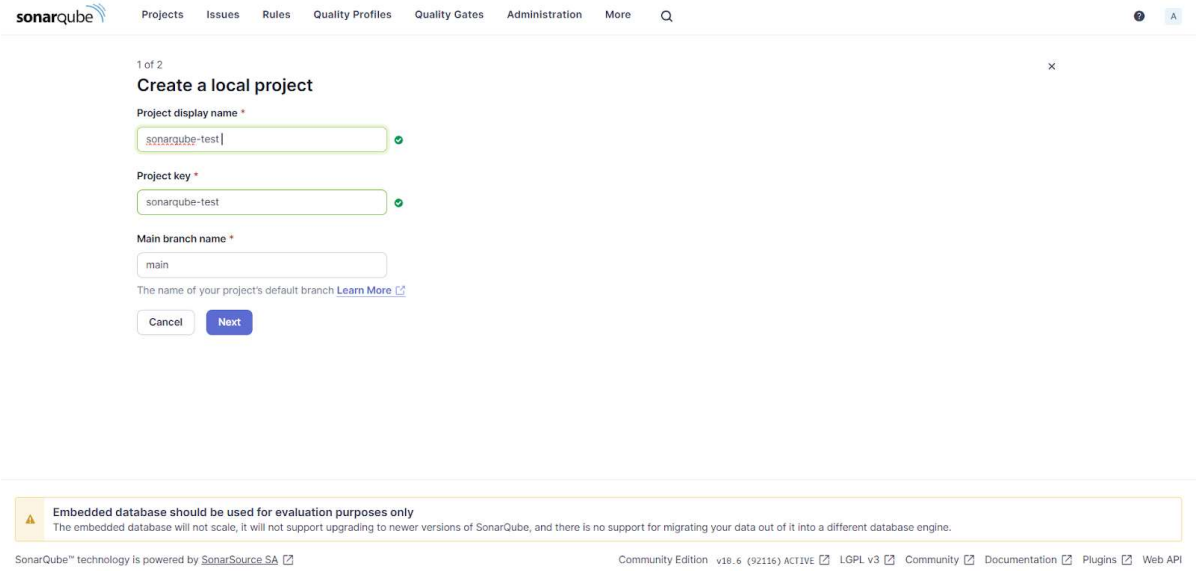
```
docker run -d --name sonarqube -e
SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

- Check SonarQube status at http://localhost:9000.
- Login with your credentials:

```
C:\Users\sbpol>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: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
2f213117ce50f08304d681a60dc0e2a4dd6c3c8e46f5725be7fb40fd0d48bb5d
```

3. Create a Project in SonarQube

- Go to **Projects** > **Create Project**.
- Name the project (e.g., sonarqube-test).

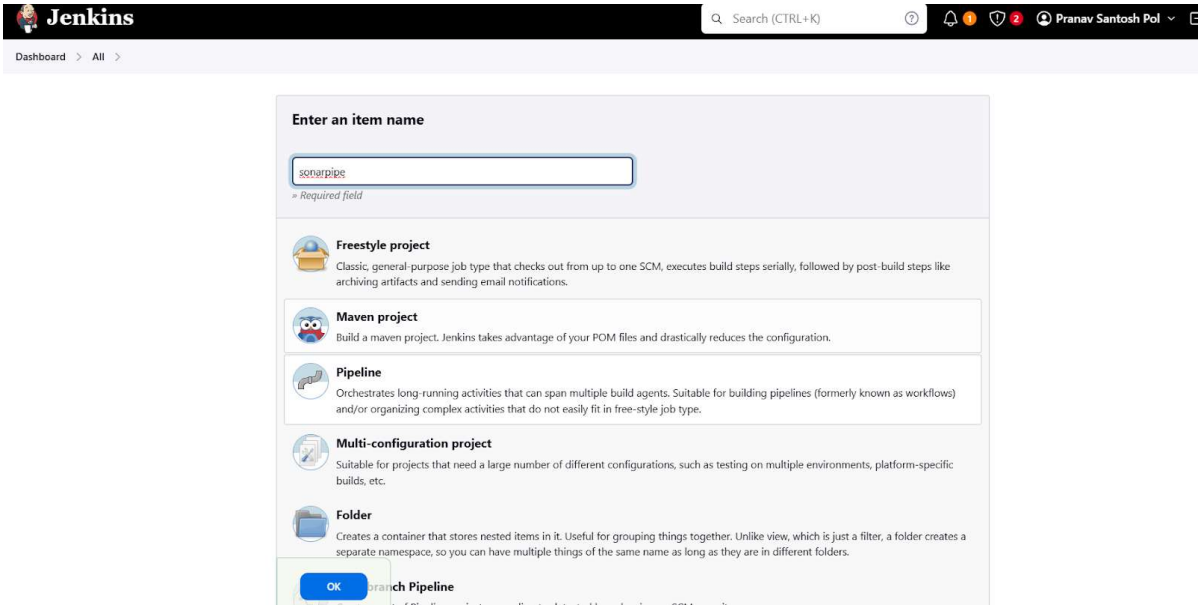


4. Generate SonarQube Token

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

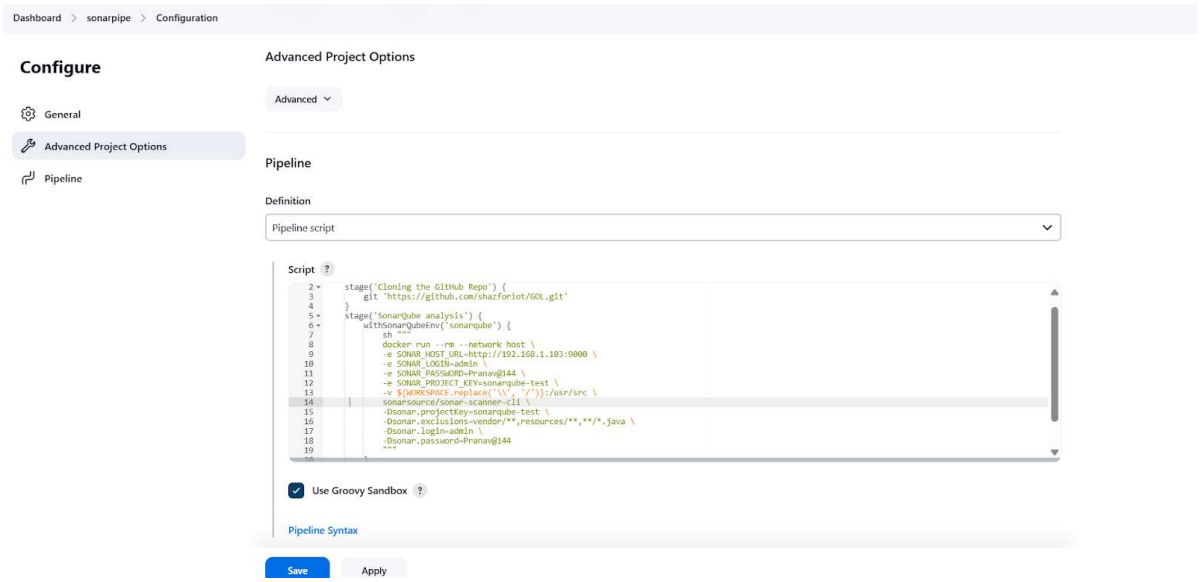
5. Create a Jenkins Pipeline

- Go to Jenkins Dashboard, click **New Item**, and select **Pipeline**.



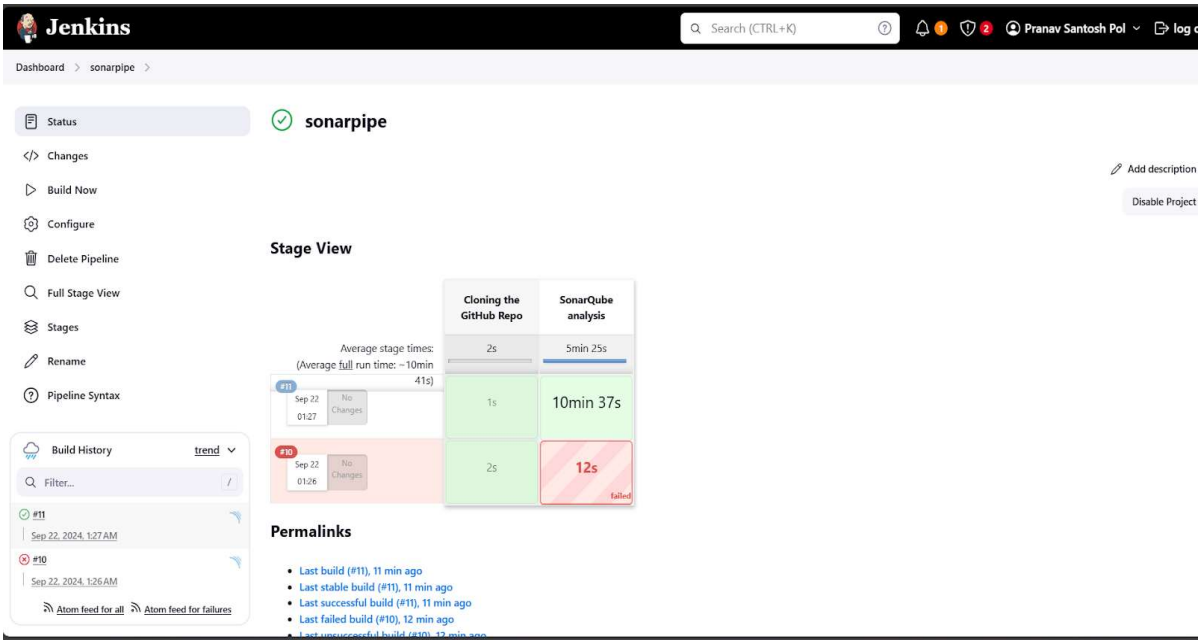
6. Under Pipeline Script, enter the following script:

```
docker network create sonarnet

node {
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
   }
   stage('SonarQube analysis') {
      withSonarQubeEnv('sonarqube') {
         sh """
         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>
         """
      }
   }
}
```



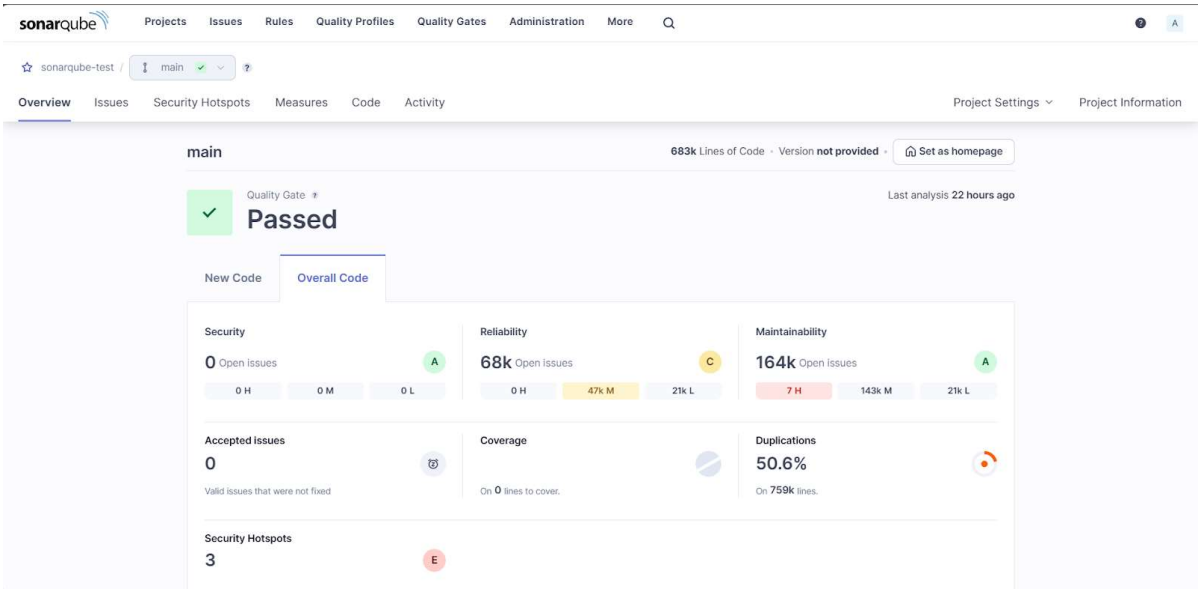
7. Run the Pipeline

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



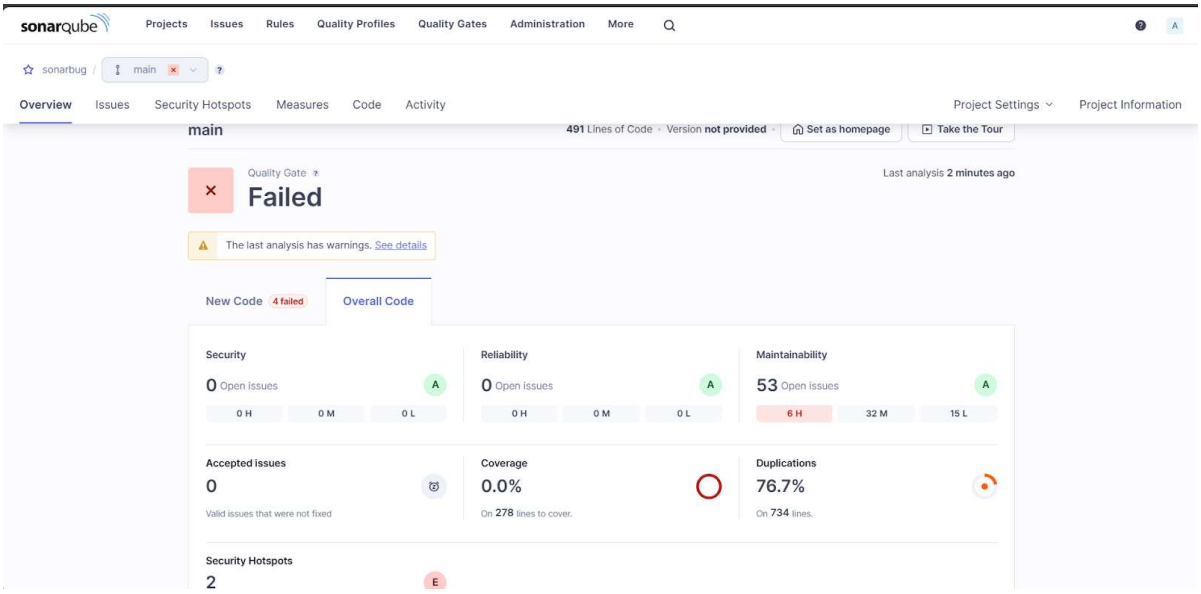
9. Check SonarQube for Analysis Results

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

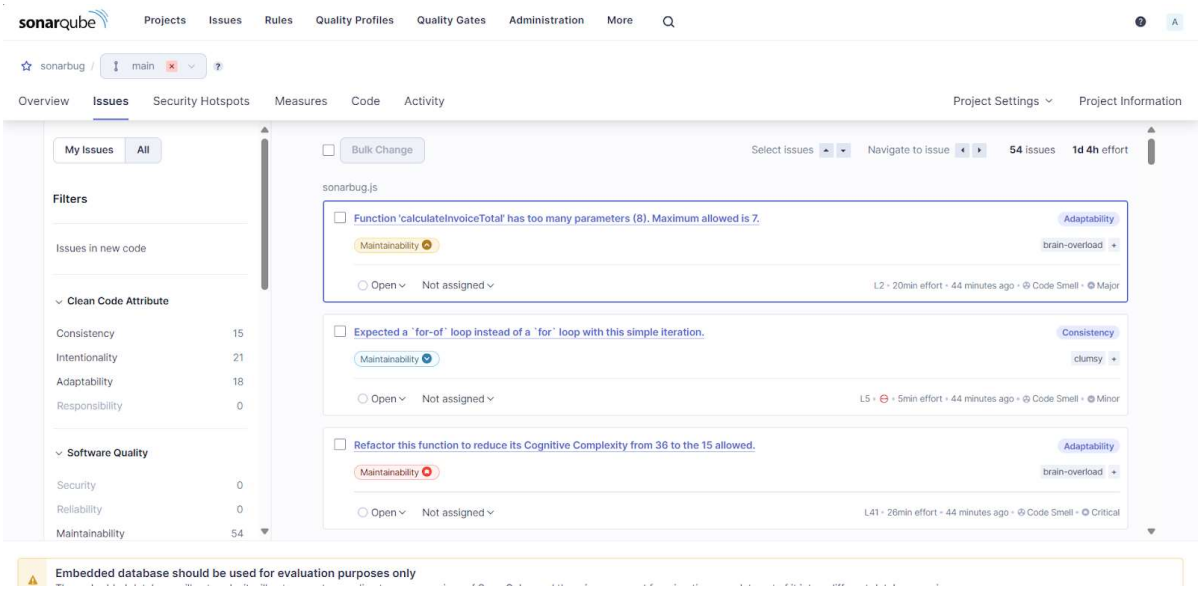


10. Checking SonarQube for Analysis Results of a File with Bugs , Code Smells, Security Vulnerabilities, Cyclomatic Complexities and Duplicates .

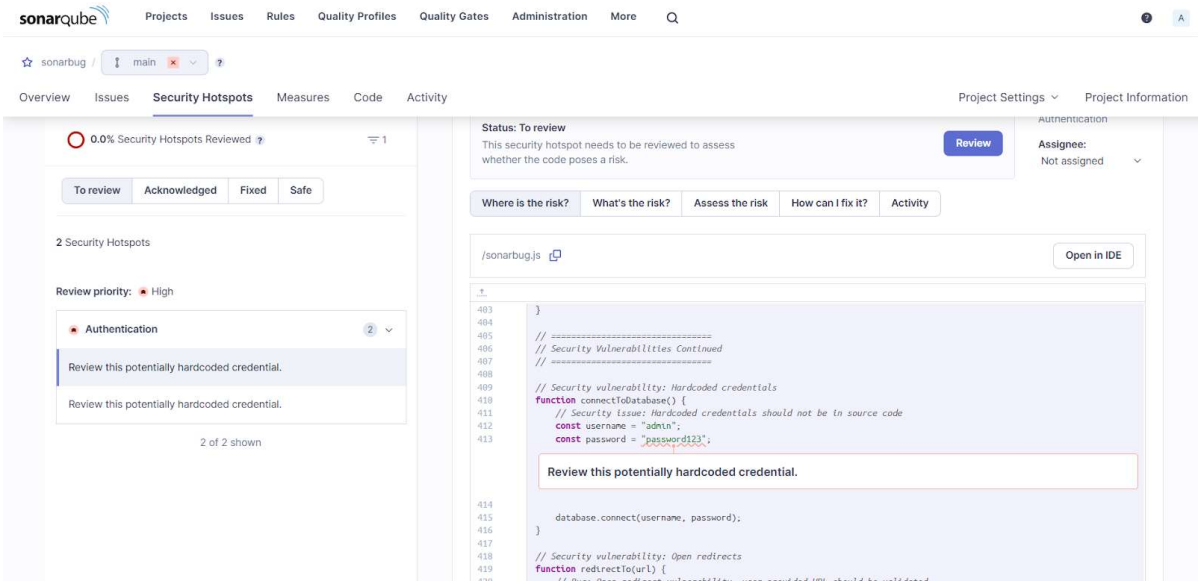
- Overview -



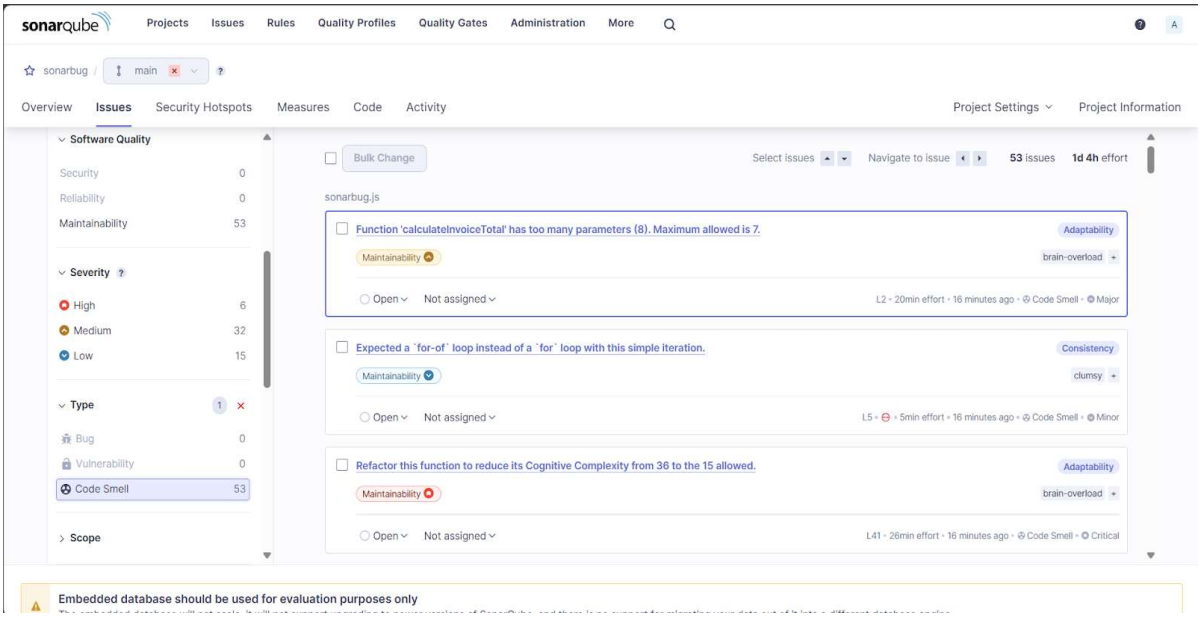
- Issues -

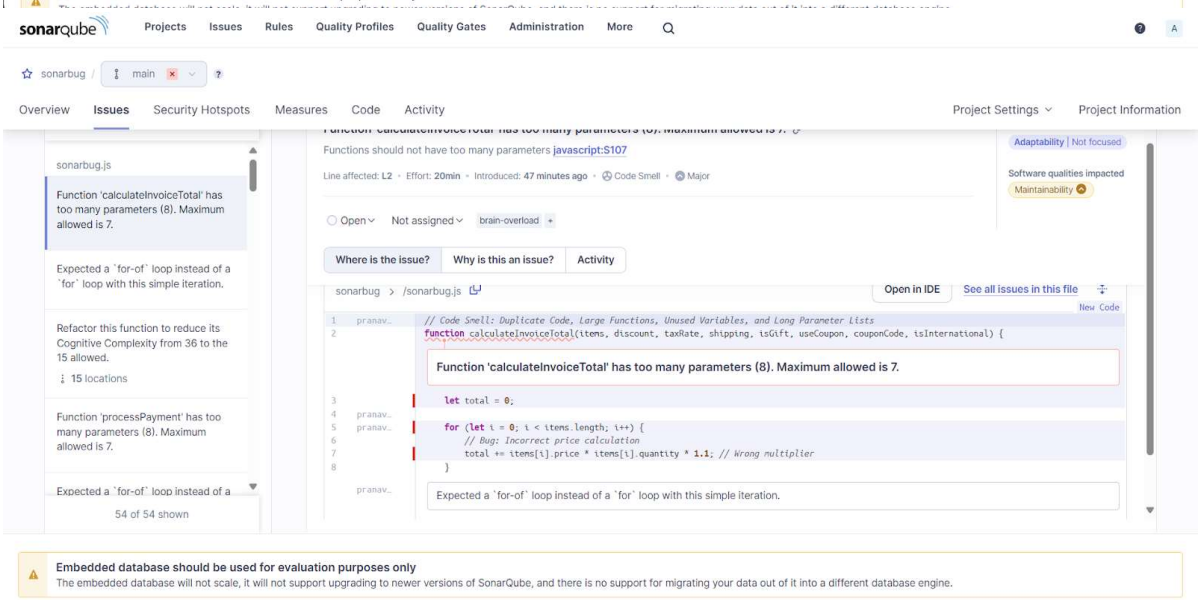


- Security Hotspot (Security Vulnerabilities) -

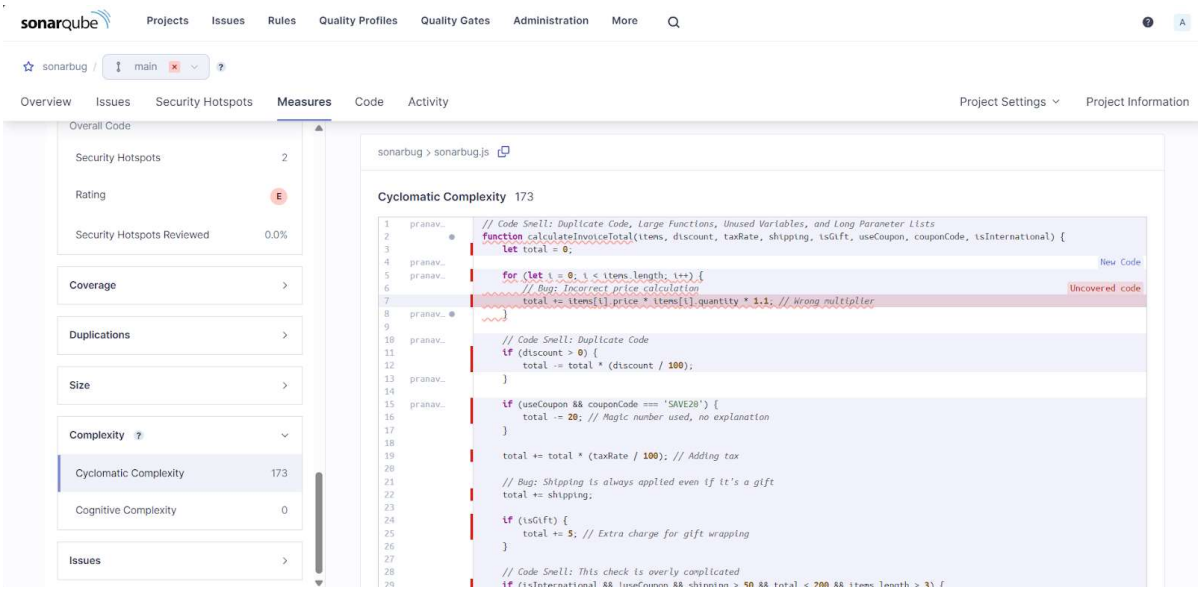


- Codesmells -

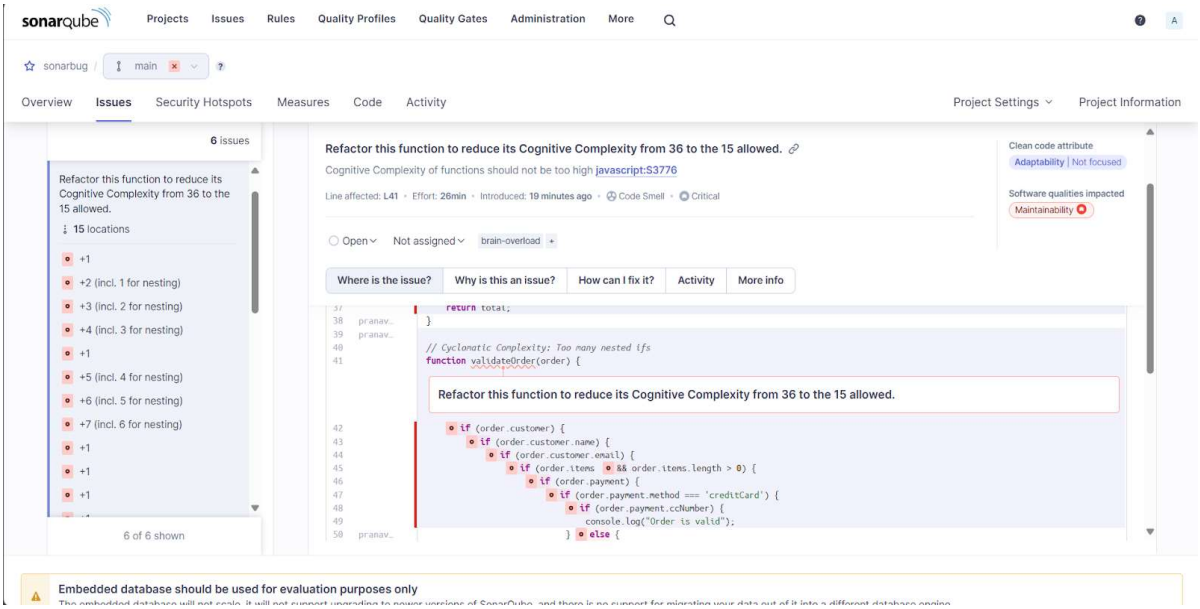

sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarbug
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
Software Quality
Security 0
Reliability 0
Maintainability 53
Severity
High 6
Medium 32
Low 15
Type 1
Bug 0
Vulnerability 0
Code Smell 53
Scope
Bulk Change
Select issues
Navigate to issue
53 issues
1d 4h effort
sonarbug.js
Function 'calculateInvoiceTotal' has too many parameters (8). Maximum allowed is 7.
Adaptability
Maintainability
brain-overload
Open
Not assigned
L2 • 20min effort • 16 minutes ago • Code Smell • Major
Expected a `for-of` loop instead of a `for` loop with this simple iteration.
Consistency
Maintainability
clumsy
Open
Not assigned
L5 • 5min effort • 16 minutes ago • Code Smell • Minor
Refactor this function to reduce its Cognitive Complexity from 36 to the 15 allowed.
Adaptability
Maintainability
brain-overload
Open
Not assigned
L41 • 26min effort • 16 minutes ago • Code Smell • Critical
Embedded database should be used for evaluation purposes only


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarbug
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
sonarbug.js
Function 'calculateInvoiceTotal' has too many parameters (8). Maximum allowed is 7.
Expected a `for-of` loop instead of a `for` loop with this simple iteration.
Refactor this function to reduce its Cognitive Complexity from 36 to the 15 allowed.
15 locations
Function 'processPayment' has too many parameters (8). Maximum allowed is 7.
54 of 54 shown
Functions should not have too many parameters javascript:S107
Line affected: L2 • Effort: 20min • Introduced: 47 minutes ago • Code Smell • Major
Adaptability | Not focused
Software qualities impacted
Maintainability
Open
Not assigned
brain-overload
Where is the issue?
Why is this an issue?
Activity
sonarbug > /sonarbug.js
Open in IDE
See all issues in this file
New Code
// Code Smell: Duplicate Code, Large Functions, Unused Variables, and Long Parameter Lists
function calculateInvoiceTotal(items, discount, taxRate, shipping, isGift, useCoupon, couponCode, isInternational) {
Function 'calculateInvoiceTotal' has too many parameters (8). Maximum allowed is 7.
let total = 0;
for (let i = 0; i < items.length; i++) {
// Bug: Incorrect price calculation
total += items[i].price * items[i].quantity * 1.1; // Wrong multiplier
}
Expected a `for-of` loop instead of a `for` loop with this simple iteration.
Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

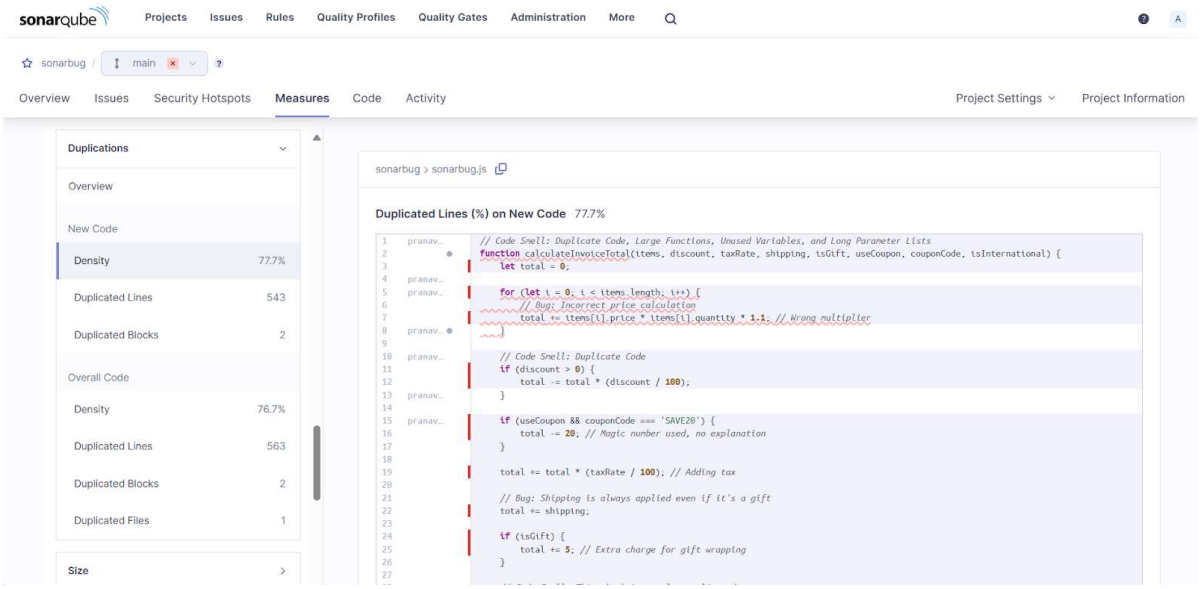
- Cyclomatic Complexity -



- Cognitive Complexity -



- Duplications -



Conclusion:

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

EXPERIMENT NO. 9

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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

Theory:

What is Nagios?

Nagios is an open-source monitoring tool designed to monitor systems, networks, and infrastructure. It helps organizations identify and resolve IT infrastructure issues before they affect critical business processes. Nagios provides monitoring and alerting services for servers, switches, applications, and services.

Key Features of Nagios

1. **Monitoring**: Nagios can monitor a wide range of network services (HTTP, SMTP, POP3, etc.), host resources (processor load, disk usage, system logs, etc.), and environmental factors (temperature, humidity, etc.).
2. **Alerting**: When an issue is detected, Nagios can send alerts via email, SMS, or custom scripts to notify administrators.
3. **Reporting**: Nagios provides detailed reports and logs of outages, events, notifications, and alert responses, helping in historical analysis and SLA compliance.
4. **Scalability**: Nagios is designed to scale and can monitor large, complex environments.
5. **Flexibility**: With a wide range of plugins and add-ons, Nagios can be customized to meet specific monitoring needs.

How Nagios Works

1. **Configuration**: Administrators configure Nagios to monitor specific services and hosts. This involves defining what to monitor, how to monitor it, and what actions to take when issues are detected.
2. **Plugins**: Nagios uses plugins to gather information about the status of various services and hosts. These plugins can be custom scripts or pre-built ones available in the Nagios community.
3. **Scheduling**: Nagios schedules regular checks of the defined services and hosts using the configured plugins.
4. **Alerting**: If a check indicates a problem, Nagios triggers an alert. Alerts can be configured to escalate if not acknowledged within a certain timeframe.

5. **Web Interface**: Nagios provides a web interface for viewing the status of monitored services and hosts, acknowledging alerts, and generating reports.

Continuous Monitoring

Continuous monitoring is a process that involves constantly tracking and analyzing the performance and security of IT systems. This practice is crucial for identifying and responding to issues in real-time, ensuring system reliability, and maintaining security. Key benefits include:

- **Real-time insights** into system performance.
- **Early detection** of issues to prevent downtime.
- **Enhanced security** through continuous threat detection.
- **Improved compliance** with regulatory standards.

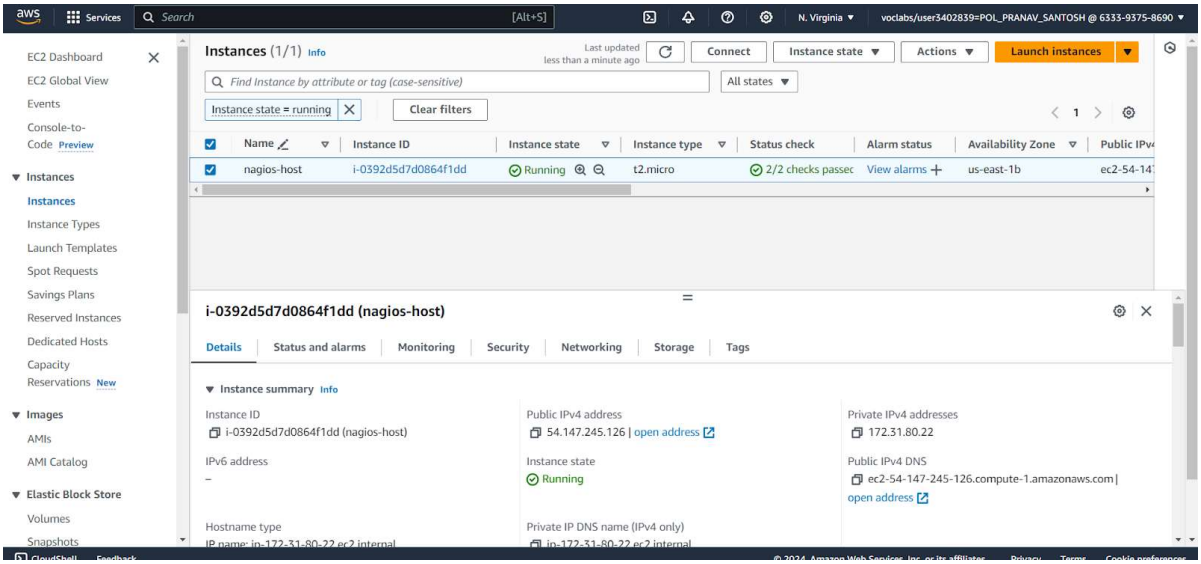
Setting Up Nagios

1. **Installation**: Install Nagios on a server, typically a Linux-based system.
2. **Configuration Files**: Edit configuration files to define what to monitor and how to monitor it. This includes defining hosts, services, contacts, and notification methods.
3. **Plugins**: Install and configure necessary plugins to monitor specific services and hosts.
4. **Web Interface**: Set up the web interface to allow easy access to monitoring data and alert management.
5. **Testing**: Test the configuration to ensure that Nagios is correctly monitoring the defined services and hosts and that alerts are being sent as expected.

Implementation :

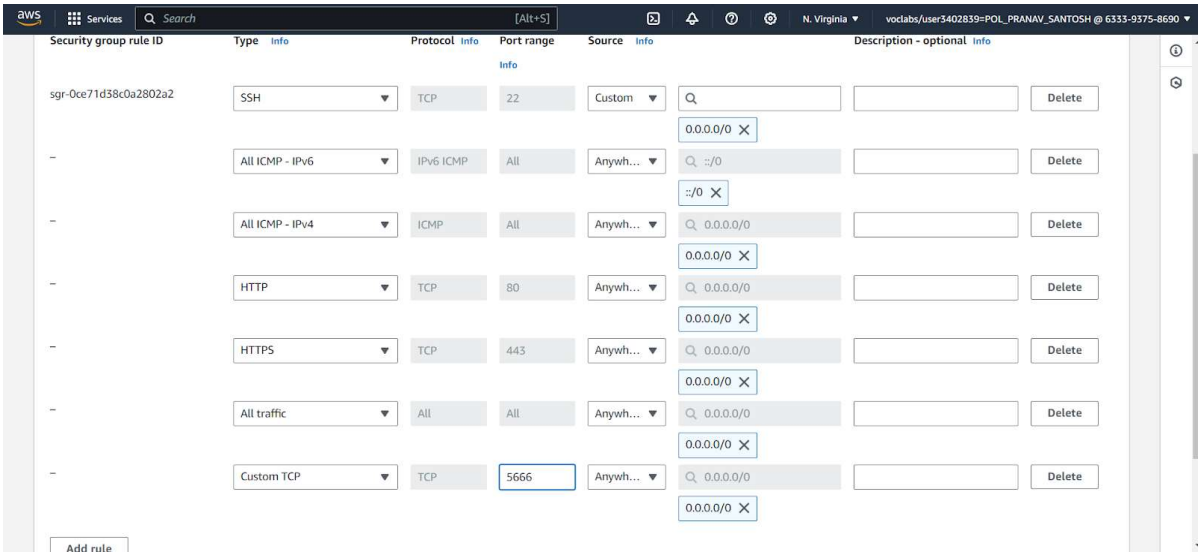
1. Create an Amazon Linux EC2 Instance

- Name it nagios-host.



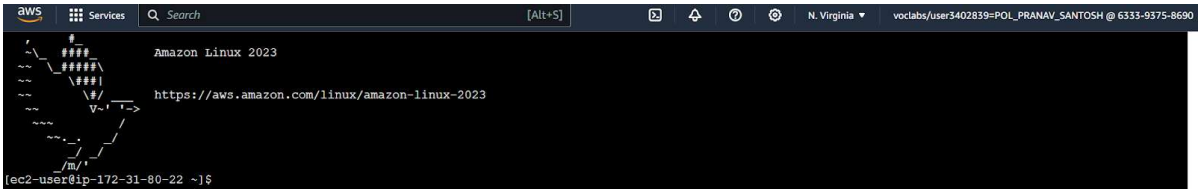
2. Configure Security Group

- 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

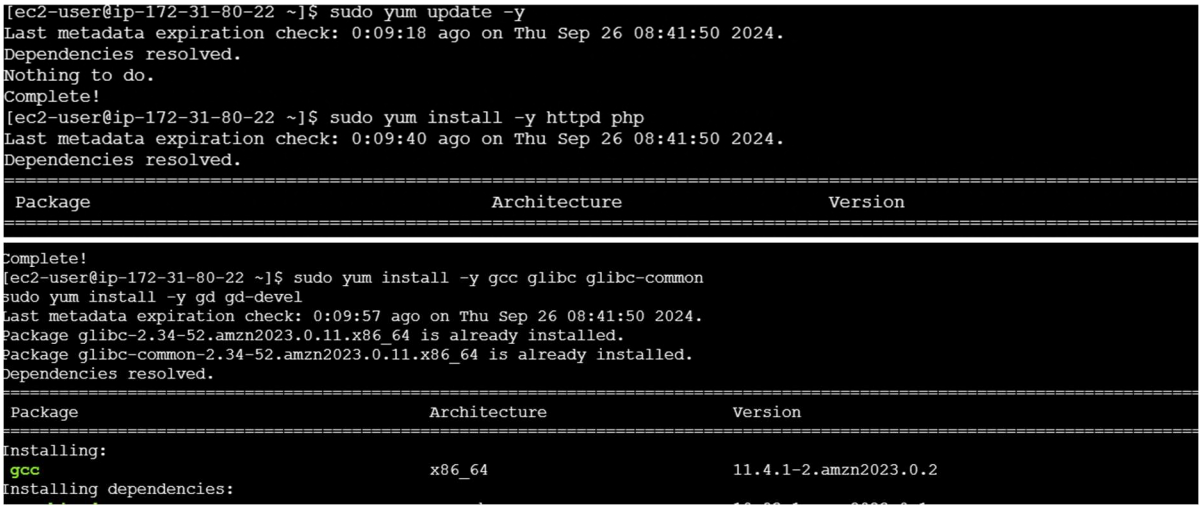
Commands -

sudo yum update

sudo yum install httpd php

sudo yum install gcc glibc glibc-common

sudo yum install gd gd-devel

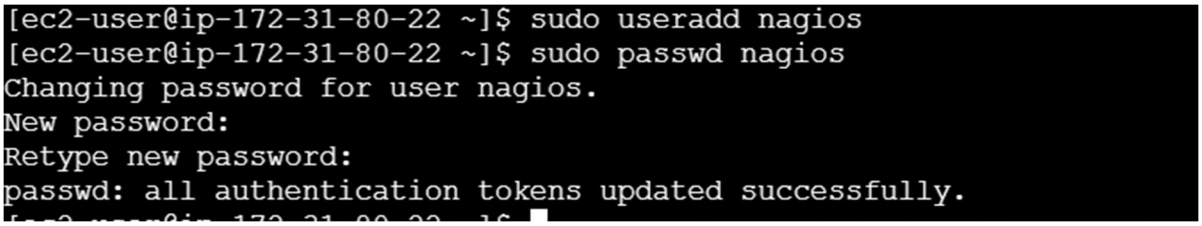


5. Create a New Nagios User

Commands -

sudo adduser -m nagios

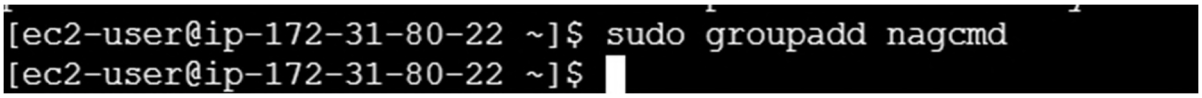
sudo passwd nagios



6. Create a New User Group

Commands -

sudo groupadd nagcmd

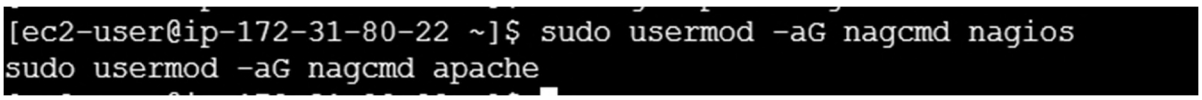


7. Add Users to the Group

Commands -

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache



8. Create a Directory for Nagios Downloads

Commands -

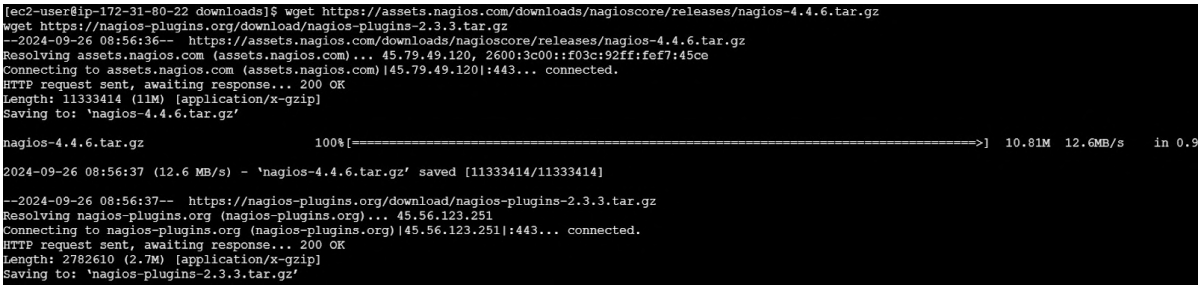
mkdir ~/downloads
cd ~/downloads

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



10. Extract the Nagios Source File

Commands -

tar zxvf nagios-4.4.6.tar.gz

cd nagios-4.4.6

11. Run the Configuration Script

Commands -

./configure --with-command-group=nagcmd

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
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 whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
```

12. Compile the Source Code

Commands -

make all

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/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 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:277:12:
workers.c:253:17: warning: '%s' directive argument is null [-Wformat-overflow=]
  253 |                 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
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o events.o events.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o flapping.o flapping.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o logging.o logging.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o macros-base.o ../common/macros.c
```

```
*** 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.
```

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

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.4.6/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
       /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
```

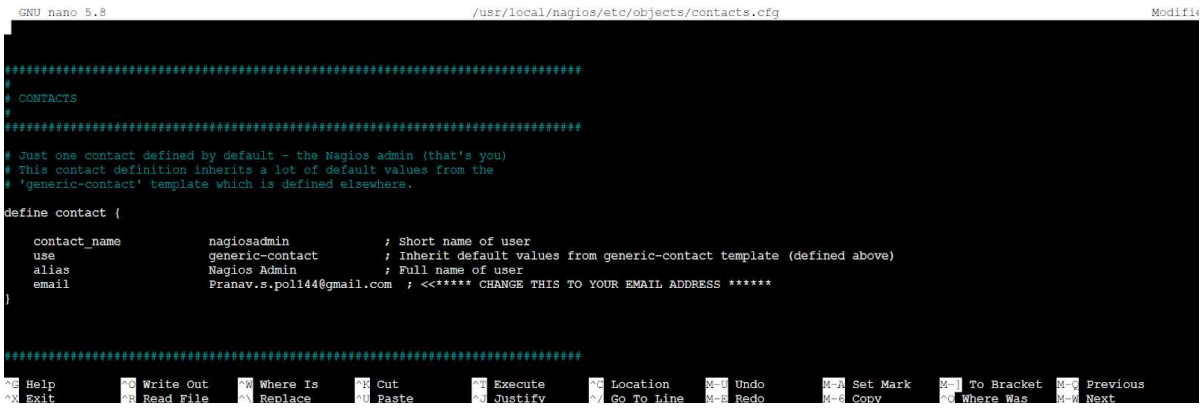
```
*** Config files installed ***

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 ***
```

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.



15. Configure the Web Interface

Commands -

sudo make install-webconf

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
[ec2-user@ip-172-31-80-22 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-172-31-80-22 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-172-31-80-22 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
```

17. Restart Apache

Commands -

sudo systemctl restart httpd

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$
```

18. Extract the Plugins Source File

Commands -

cd ~/downloads

tar zxvf nagios-plugins-2.3.3.tar.gz

cd nagios-plugins-2.3.3

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ cd ~/downloads
tar zxvf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3
nagios-plugins-2.3.3/
nagios-plugins-2.3.3/perlmods/
nagios-plugins-2.3.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.3.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.3.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.in
nagios-plugins-2.3.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.am
nagios-plugins-2.3.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.3.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.3.3/perlmods/Class-Accessor-0.34.tar.gz
nagios-plugins-2.3.3/perlmods/Try-Tiny-0.18.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Implementation-0.07.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile
nagios-plugins-2.3.3/perlmods/Perl-OSType-1.003.tar.gz
nagios-plugins-2.3.3/perlmods/install_order
nagios-plugins-2.3.3/perlmods/Nagios-Plugin-0.36.tar.gz
nagios-plugins-2.3.3/perlmods/Math-Calc-Units-1.07.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Build-0.4007.tar.gz
nagios-plugins-2.3.3/ABOUT-NLS
nagios-plugins-2.3.3/configure.ac
nagios-plugins-2.3.3/Makefile.in
```

19. Compile and Install Plugins

Commands -

./configure --with-nagios-user=nagios --with-nagios-group=nagios

make

sudo make install

```
[ec2-user@ip-172-31-80-22 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
```

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-172-31-80-22 nagios-plugins-2.3.3]$ 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
Error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
       Checked 8 services.
       Checked 1 hosts.
       Checked 1 host groups.
```

21. Check the Status of Nagios

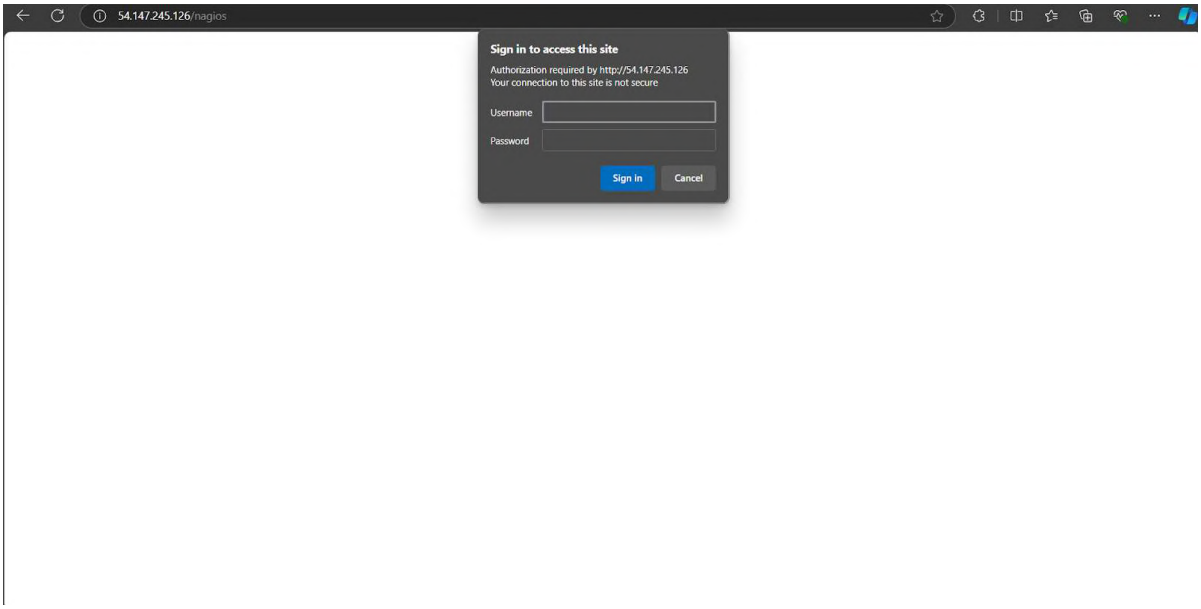
Commands -

sudo systemctl status nagios

```
[ec2-user@ip-172-31-80-22 nagios-plugins-2.3.3]$ 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 Thu 2024-09-26 09:09:51 UTC; 1min 34s ago
       Docs: https://www.nagios.org/documentation
    Process: 68229 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 68230 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 68231 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.3M
        CPU: 33ms
     CGroup: /system.slice/nagios.service
             ├─68231 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─68232 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68233 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68234 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─68235 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─68236 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: qh: core query handler registered
Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: qh: echo service query handler registered
Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: qh: help for the query handler registered
Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: wproc: Successfully registered manager as @wproc with query handler
Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: wproc: Registry request: name=Core Worker 68234;pid=68234
Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: wproc: Registry request: name=Core Worker 68235;pid=68235
Sep 26 09:09:51 ip-172-31-80-22.ec2.internal nagios[68231]: wproc: Registry request: name=Core Worker 68233;pid=68233
```

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.




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
System
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Configuration
Nagios Core
Daemon running with PID 68231
Nagios® Core™
Version 4.4.6
April 28, 2020
Check for updates
A new version of Nagios Core is available!
Visit nagios.org to download Nagios 4.5.5.
Get Started
Start monitoring your infrastructure
Change the look and feel of Nagios
Extend Nagios with hundreds of addons
Get support
Get training
Get certified
Quick Links
Nagios Library (tutorials and docs)
Nagios Labs (development blog)
Nagios Exchange (plugins and addons)
Nagios Support (tech support)
Nagios.com (company)
Nagios.org (project)
Latest News
Don't Miss...
Page Tour
Copyright © 2010-2020 Nagios Core Development Team and Community Contributors. Copyright © 1999-2009 Ethan Galstad. See the THANKS file for more information on contributors.

EXPERIMENT NO. 10

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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

Theory:

Nagios is a comprehensive monitoring and alerting platform designed to keep track of IT infrastructure, networks, and applications. It provides real-time monitoring, alerting, and reporting capabilities to ensure the health and performance of critical systems.

Key Components of Nagios

1. **Nagios Core**: The open-source foundation of the Nagios monitoring system. It provides the basic framework for monitoring and alerting.
2. **Nagios XI**: A commercial version of Nagios that offers advanced features, a more user-friendly interface, and additional support options.
3. **Nagios Log Server**: A tool for centralized log management, allowing you to view, analyze, and archive logs from various sources.
4. **Nagios Network Analyzer**: Provides detailed insights into network traffic and bandwidth usage.
5. **Nagios Fusion**: Centralizes monitoring data from multiple Nagios instances, providing a unified view of the entire network.

Monitoring Capabilities

1. **Port Monitoring**: Nagios can monitor specific network ports to ensure they are open and responsive. This is crucial for services that rely on these ports.
2. **Service Monitoring**: Nagios checks the status of various services (e.g., web servers, databases) to ensure they are running smoothly.
3. **Server Monitoring**: Nagios can monitor both Windows and Linux servers using agents like NSClient++ for Windows and NRPE for Linux. This includes metrics like CPU usage, memory usage, disk space, and more.

How Nagios Works

1. **Configuration**: Administrators define what to monitor and how to monitor it using configuration files.
2. **Plugins**: Nagios uses plugins to gather information about the status of various services and hosts. These plugins can be custom scripts or pre-built ones.
3. **Scheduling**: Nagios schedules regular checks of the defined services and hosts using the configured plugins.

4. **Alerting**: If a check indicates a problem, Nagios triggers an alert. Alerts can be configured to escalate if not acknowledged within a certain timeframe.
5. **Log Management**: Centralizing and analyzing logs from various sources to detect issues and ensure compliance.

Implementation :

Prerequisites

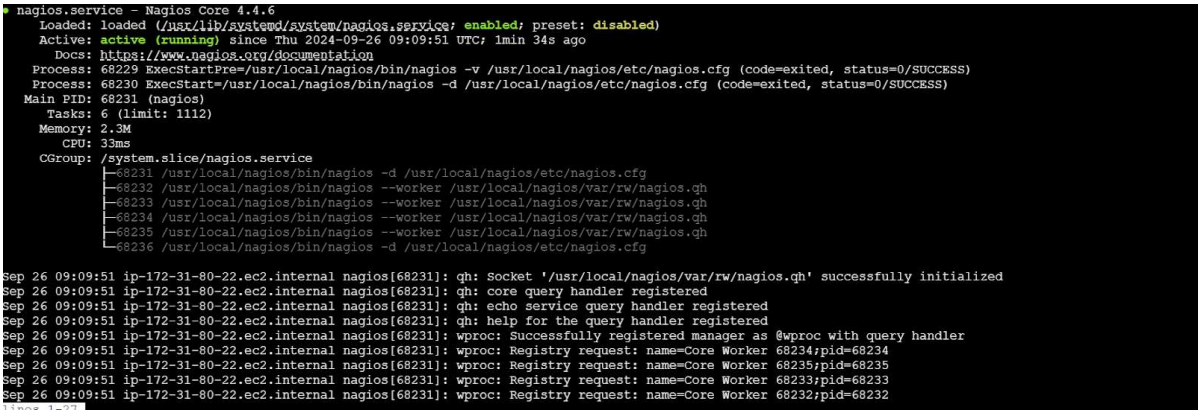
- AWS Free Tier
- Nagios Server running on an Amazon Linux Machine

1. Confirm Nagios is Running on the Server

Commands -

sudo systemctl status nagios

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



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

5. Copy Sample Configuration File

Commands -

cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

```
[root@ip-172-31-80-22 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-80-22 ec2-user]# sudo nano /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.

```
# HOST DEFINITION
#
###############################################################################

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   linuxserver
    address                 3.85.25.81
}


###############################################################################
```

- Change hostgroup_name under hostgroup to linux-servers1.

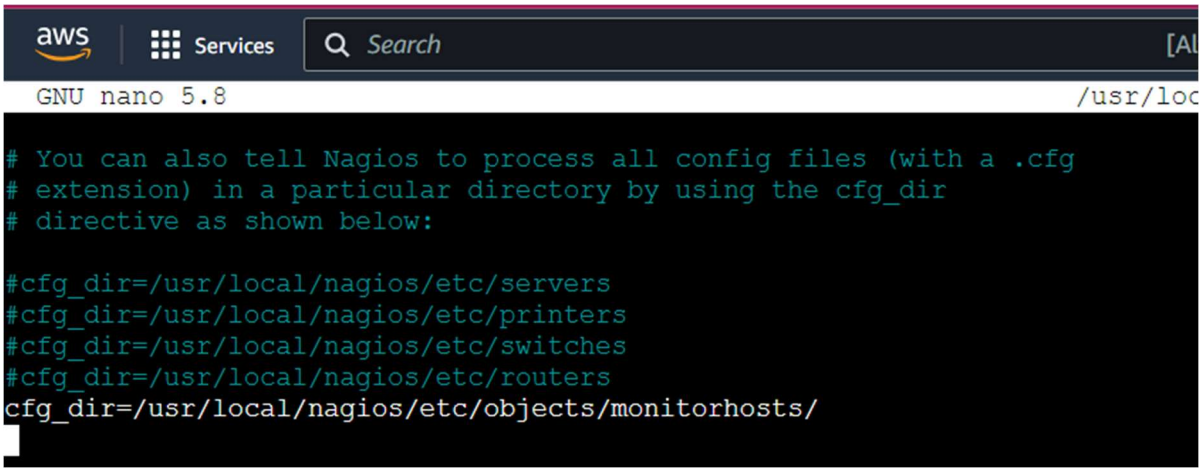
```
###############################################################################
#
# HOST GROUP DEFINITION
#
###############################################################################

# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 linuxserver             ; Comma separated list of hosts that belong to this group
}
```

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/

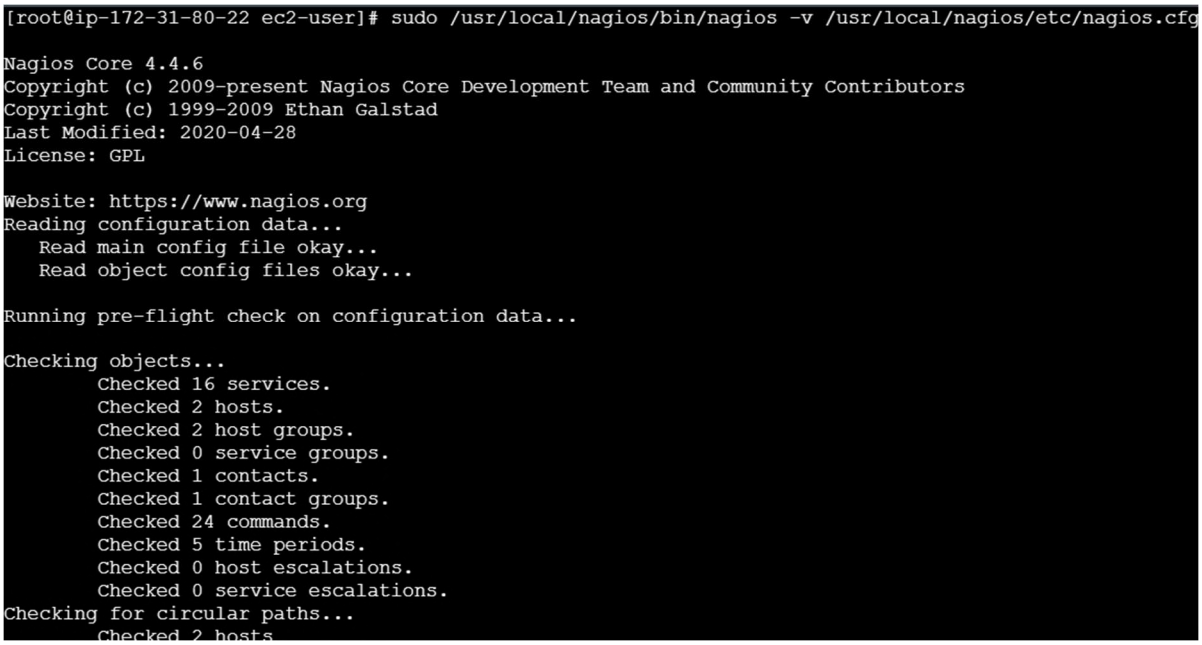


8. Verify Configuration Files

Commands -

sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

- Ensure there are no errors.



```
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

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

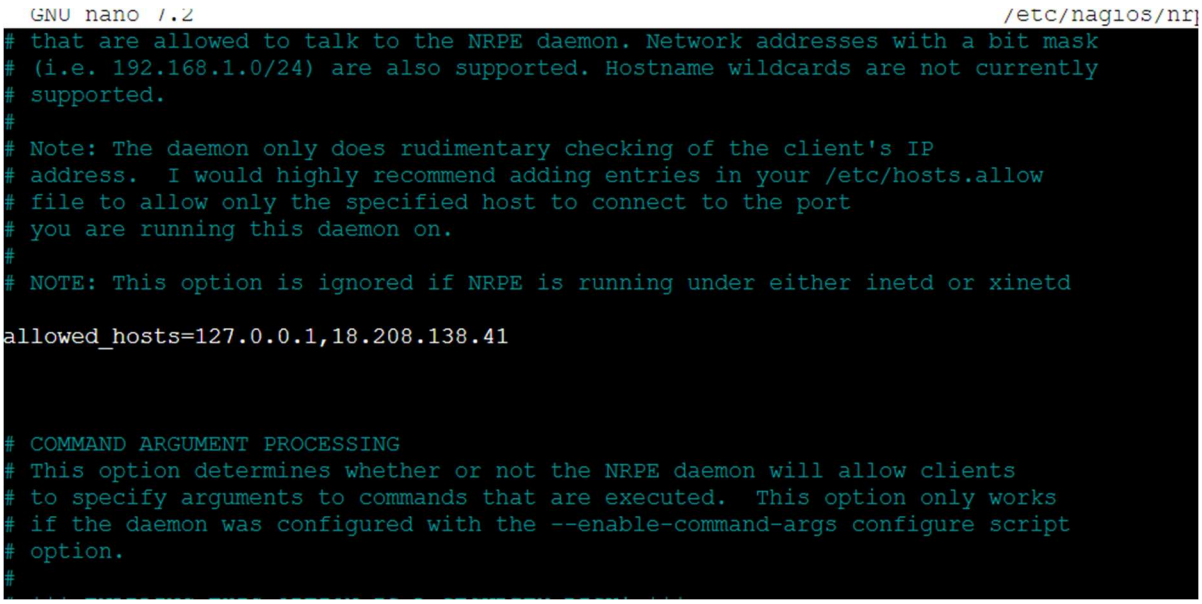
```
ubuntu@ip-172-31-88-112:~$ sudo apt update -y
sudo apt install gcc -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [378 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.0 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4548 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [271 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [115 kB]
```

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 -

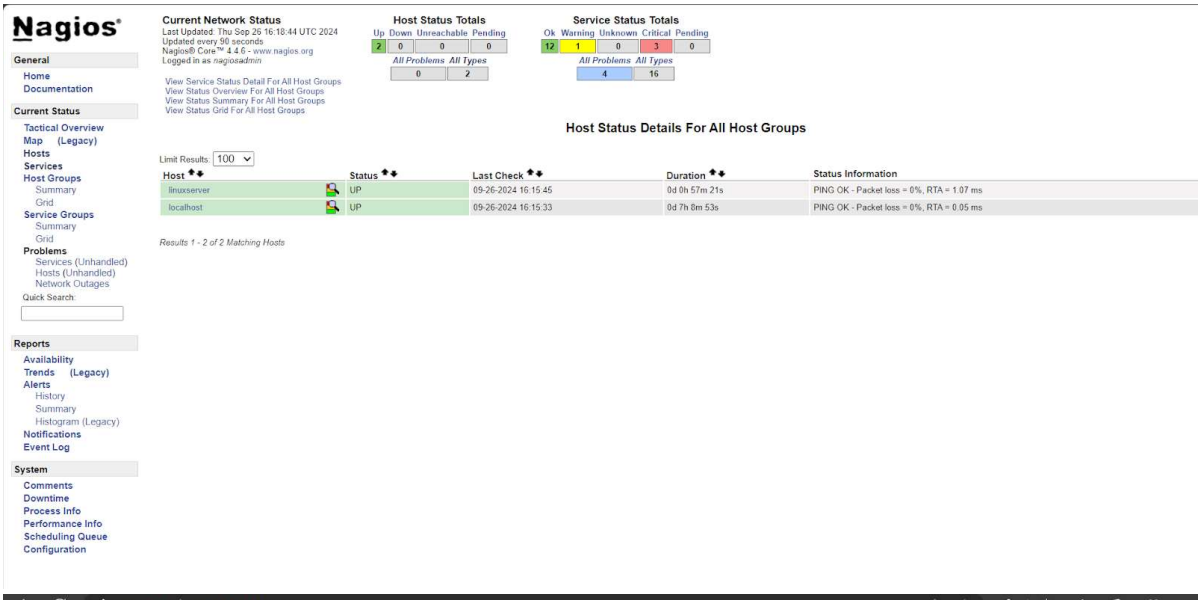
sudo systemctl restart nagios-nrpe-server

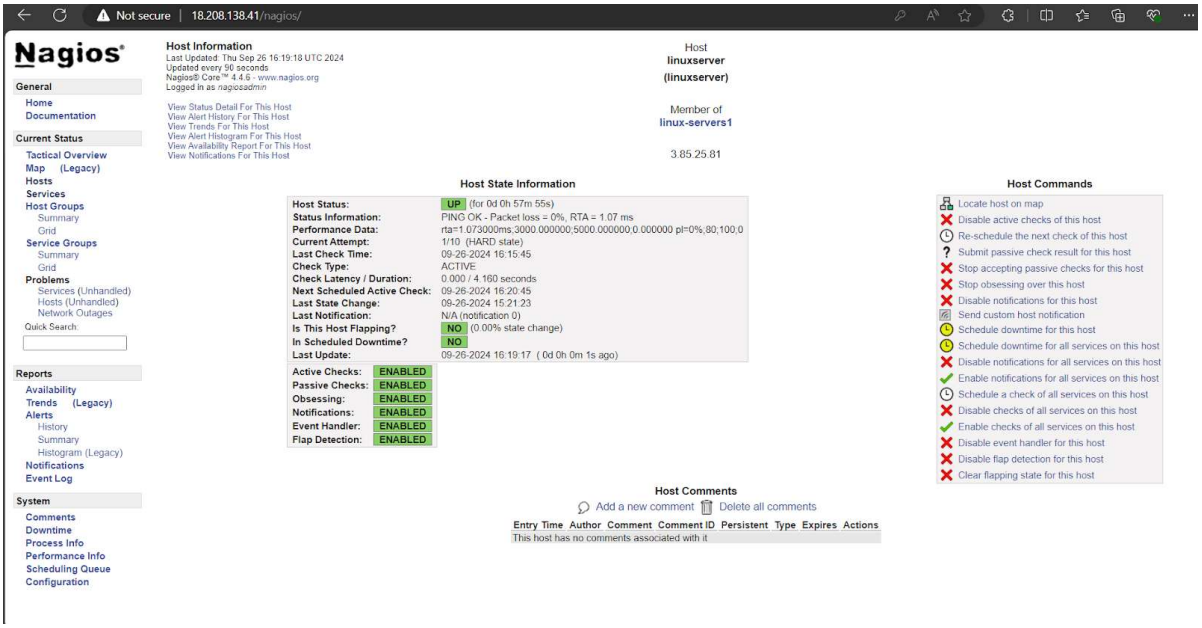
14. Check Nagios Dashboard

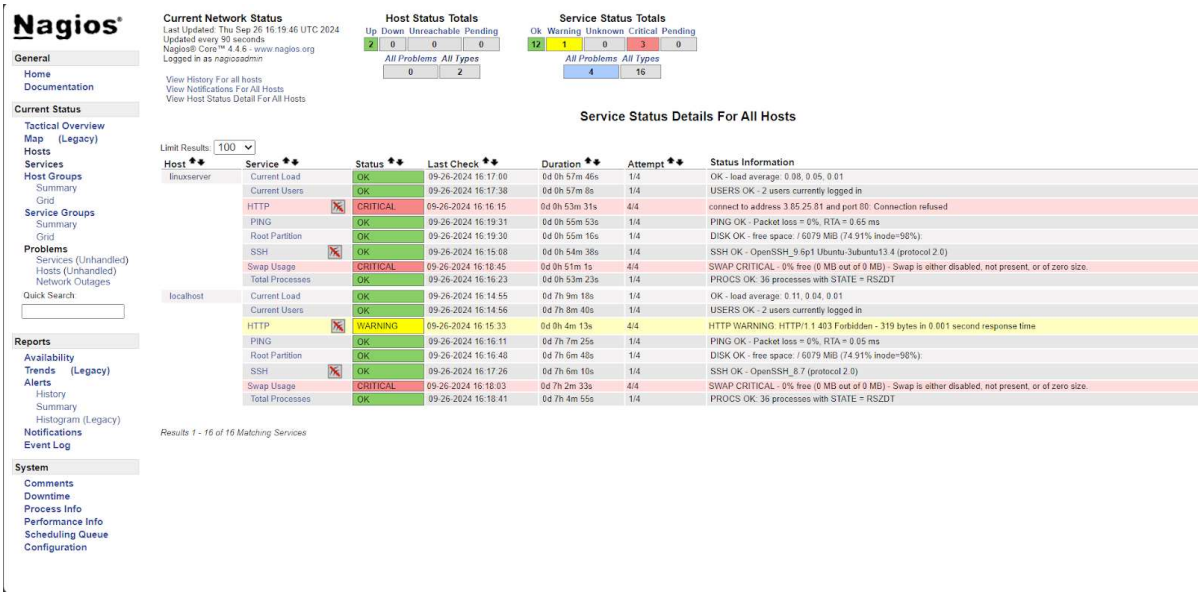
- Open your browser and navigate to http://<Nagios_Host_IP>/nagios.

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
Current Network Status
Last Updated: Thu Sep 26 16:18:44 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as nagiosadmin
View Service 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
Host Status Details For All Host Groups
Limit Results: 100
Host Status Last Check Duration Status Information
linuxserver UP 09-26-2024 16:15:45 0d 0h 57m 21s PING OK - Packet loss = 0%, RTA = 1.07 ms
localhost UP 09-26-2024 16:15:33 0d 7h 8m 53s PING OK - Packet loss = 0%, RTA = 0.05 ms
Results 1 - 2 of 2 Matching Hosts
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
System
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Configuration


Not secure | 18.208.138.41/nagios/
Nagios
Host Information
Last Updated: Thu Sep 26 16:19:18 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as nagiosadmin
View Status Detail For This Host
View Alert History For This Host
View Trends For This Host
View Alert Histogram For This Host
View Availability Report For This Host
View Notifications For This Host
Host
linuxserver
(linuxserver)
Member of
linux-servers1
3.85.25.81
Host State Information
Host Status: UP (for 0d 0h 57m 55s)
Status Information: PING OK - Packet loss = 0%, RTA = 1.07 ms
Performance Data: rta=1.073000ms;3000.000000;5000.000000;0.000000 pl=0%;80;100;0
Current Attempt: 1/10 (HARD state)
Last Check Time: 09-26-2024 16:15:45
Check Type: ACTIVE
Check Latency / Duration: 0.000 / 4.160 seconds
Next Scheduled Active Check: 09-26-2024 16:20:45
Last State Change: 09-26-2024 15:21:23
Last Notification: N/A (notification 0)
Is This Host Flapping? NO (0.00% state change)
In Scheduled Downtime? NO
Last Update: 09-26-2024 16:19:17 (0d 0h 0m 1s ago)
Active Checks: ENABLED
Passive Checks: ENABLED
Obsessing: ENABLED
Notifications: ENABLED
Event Handler: ENABLED
Flap Detection: ENABLED
Host Commands
Locate host on map
Disable active checks of this host
Re-schedule the next check of this host
Submit passive check result for this host
Stop accepting passive checks for this host
Stop obsessing over this host
Disable notifications for this host
Send custom host notification
Schedule downtime for this host
Schedule downtime for all services on this host
Disable notifications for all services on this host
Enable notifications for all services on this host
Schedule a check of all services on this host
Disable checks of all services on this host
Enable checks of all services on this host
Disable event handler for this host
Disable flap detection for this host
Clear flapping state for this host
Host Comments
Add a new comment
Delete all comments
Entry Time Author Comment Comment ID Persistent Type Expires Actions
This host has no comments associated with it
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
System
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Configuration


Nagios
Current Network Status
Last Updated: Thu Sep 26 16:19:46 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as nagiosadmin
View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts
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 Hosts
Limit Results: 100
Host Service Status Last Check Duration Attempt Status Information
linuxserver Current Load OK 09-26-2024 16:17:00 0d 0h 57m 46s 1/4 OK - load average: 0.08, 0.05, 0.01
Current Users OK 09-26-2024 16:17:38 0d 0h 57m 8s 1/4 USERS OK - 2 users currently logged in
HTTP CRITICAL 09-26-2024 16:16:15 0d 0h 53m 31s 4/4 connect to address 3.85.25.81 and port 80: Connection refused
PING OK 09-26-2024 16:19:31 0d 0h 55m 53s 1/4 PING OK - Packet loss = 0%, RTA = 0.65 ms
Root Partition OK 09-26-2024 16:19:30 0d 0h 55m 16s 1/4 DISK OK - free space: / 6079 MiB (74.91% inode=98%):
SSH OK 09-26-2024 16:15:08 0d 0h 54m 38s 1/4 SSH OK - OpenSSH_9.6p1 Ubuntu-3ubuntu13.4 (protocol 2.0)
Swap Usage CRITICAL 09-26-2024 16:18:45 0d 0h 51m 1s 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 09-26-2024 16:16:23 0d 0h 53m 23s 1/4 PROCS OK: 36 processes with STATE = RSZDT
localhost Current Load OK 09-26-2024 16:14:55 0d 7h 9m 18s 1/4 OK - load average: 0.11, 0.04, 0.01
Current Users OK 09-26-2024 16:14:56 0d 7h 8m 40s 1/4 USERS OK - 2 users currently logged in
HTTP WARNING 09-26-2024 16:15:33 0d 0h 4m 13s 4/4 HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.001 second response time
PING OK 09-26-2024 16:16:11 0d 7h 7m 25s 1/4 PING OK - Packet loss = 0%, RTA = 0.05 ms
Root Partition OK 09-26-2024 16:16:48 0d 7h 6m 48s 1/4 DISK OK - free space: / 6079 MiB (74.91% inode=98%):
SSH OK 09-26-2024 16:17:26 0d 7h 6m 10s 1/4 SSH OK - OpenSSH_8.7 (protocol 2.0)
Swap Usage CRITICAL 09-26-2024 16:18:03 0d 7h 2m 33s 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 09-26-2024 16:18:41 0d 7h 4m 55s 1/4 PROCS OK: 36 processes with STATE = RSZDT
Results 1 - 16 of 16 Matching Services
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
System
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Configuration

EXPERIMENT NO. 11

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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.

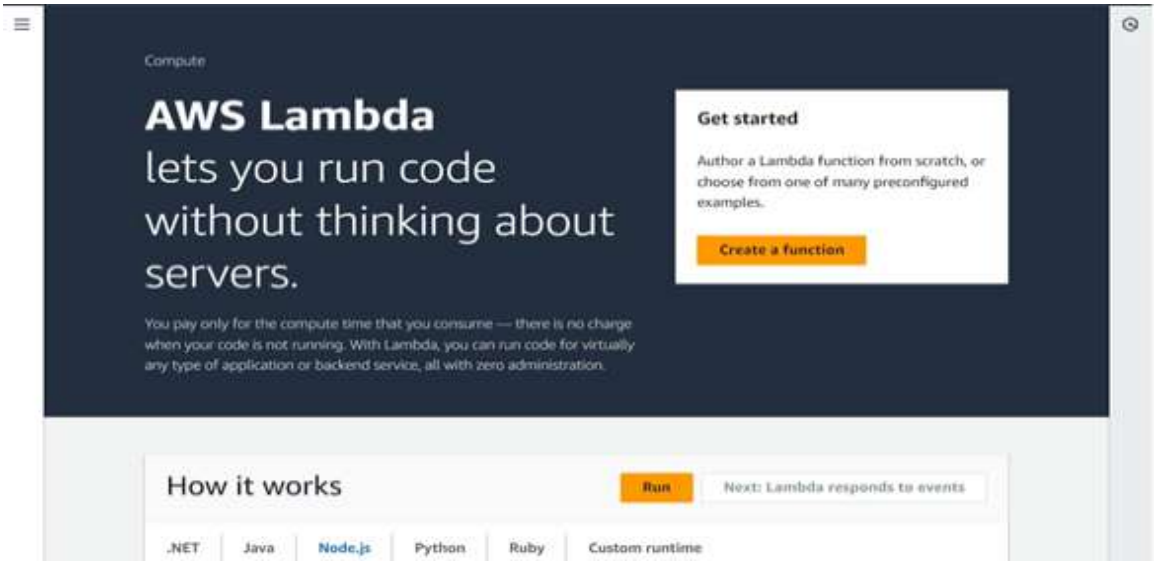
Features of AWS Lambda

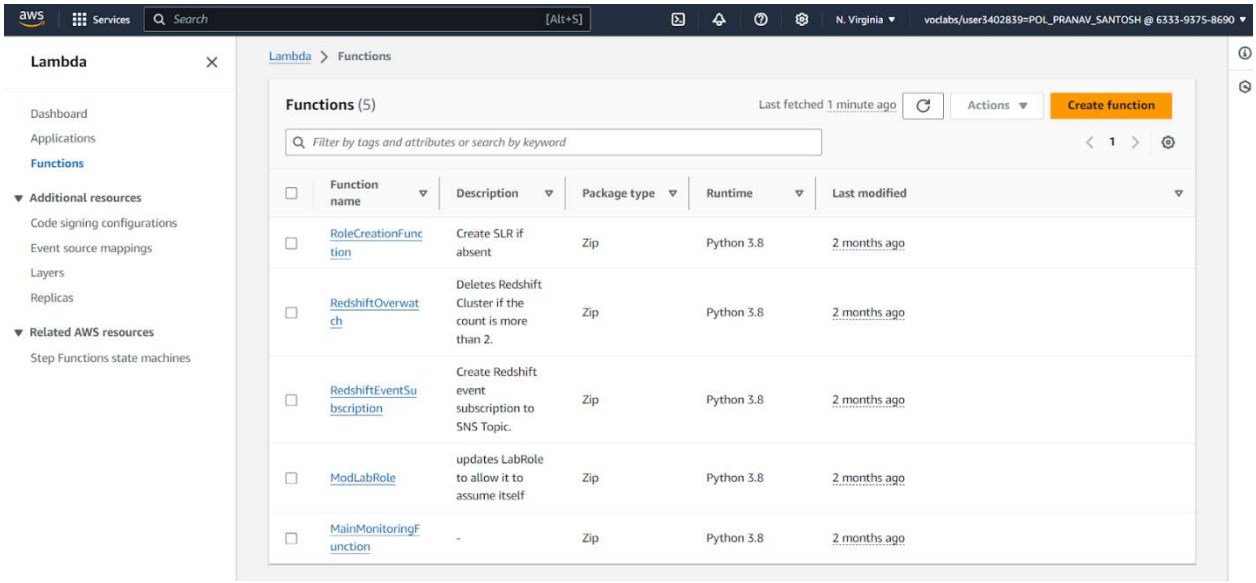
- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don't need to invest upfront. You pay 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

Steps to create an AWS Lambda function

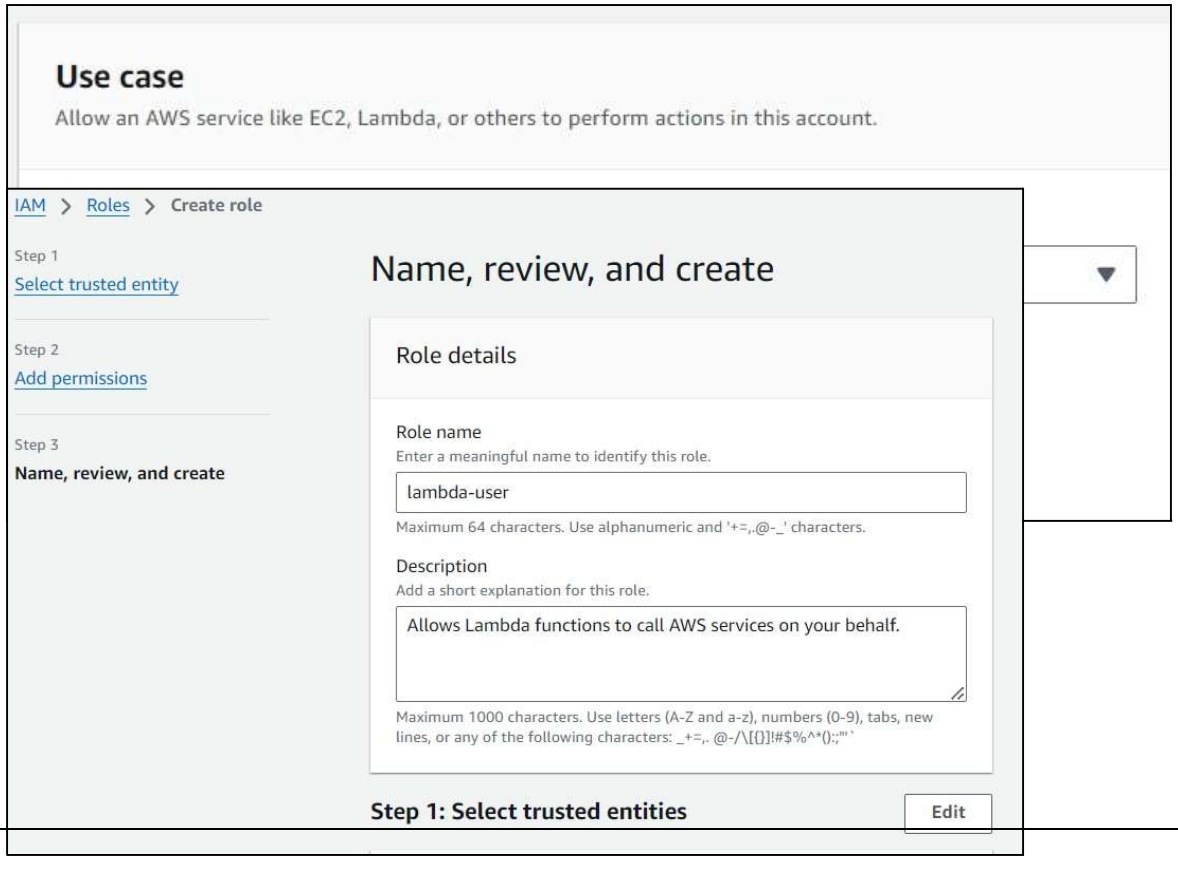
1. **Open up the Lambda Console and click on the Create button. Be mindful of where youcreate your functions since Lambda is region-dependent.**





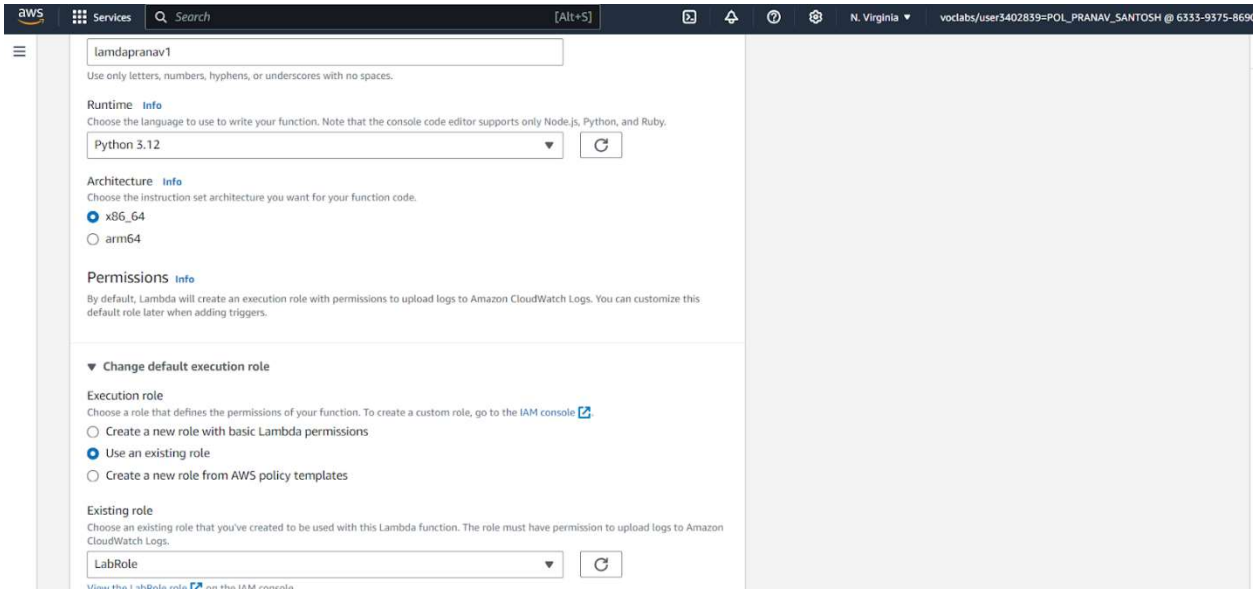
2. Attach CloudWatch Logs permissions:

- In the "Permissions" step, search for the policy called AWSLambdaBasicExecutionRole.
- Select this policy, which gives your Lambda function permission to write logs toCloudWatch.
- Click **Next**.

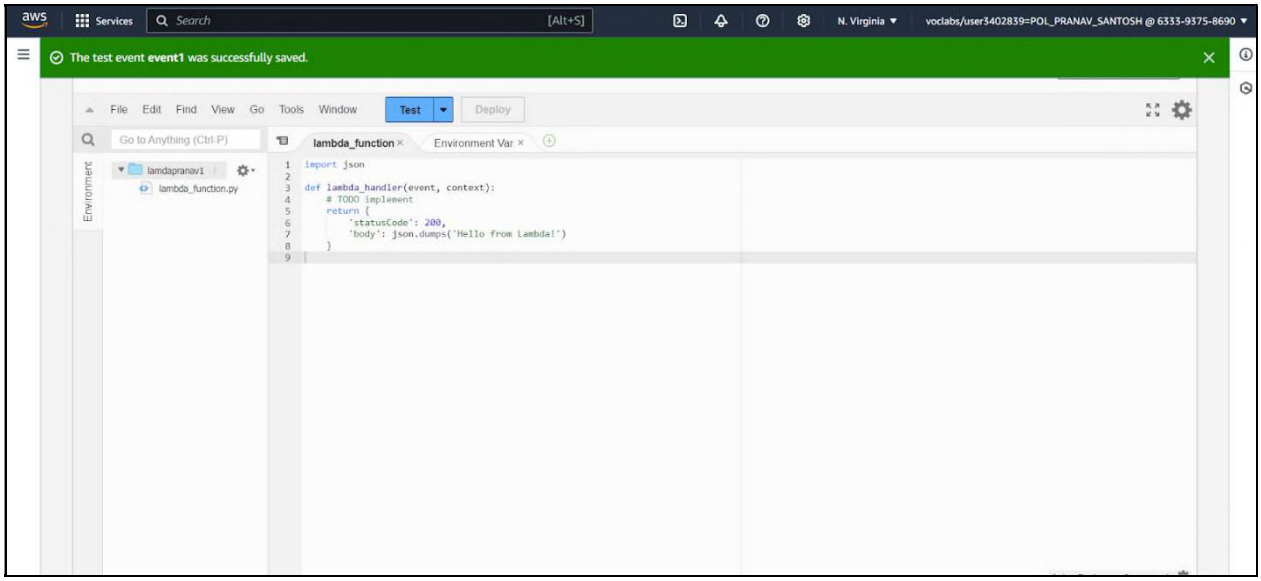


2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.

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.

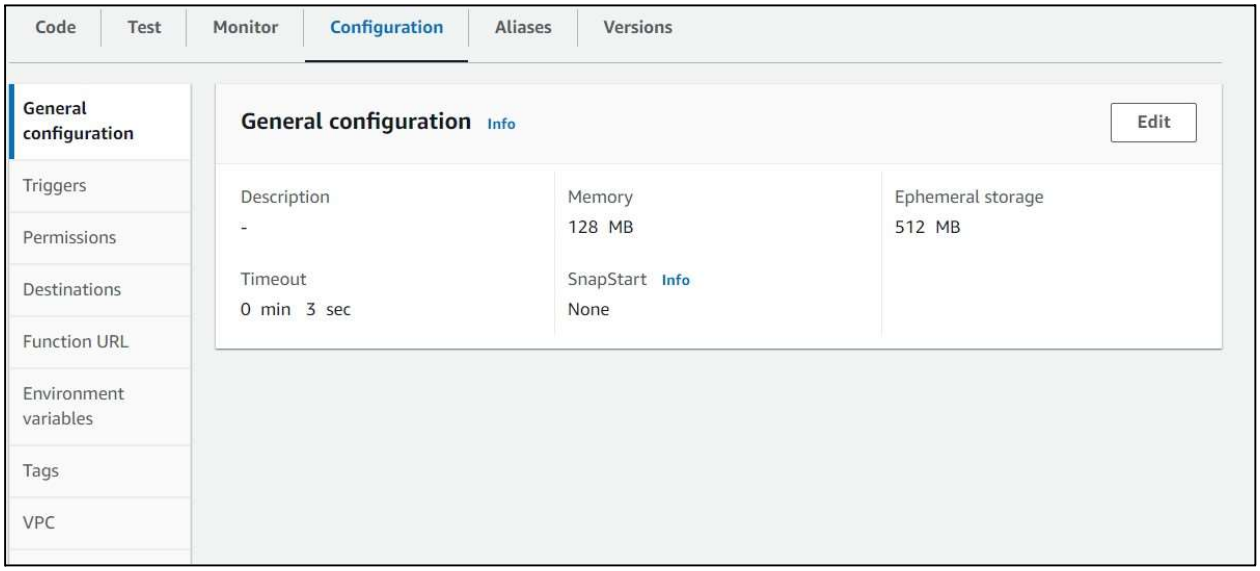


2. This process will take a while to finish and after that, you'll get a message that your function was successfully created

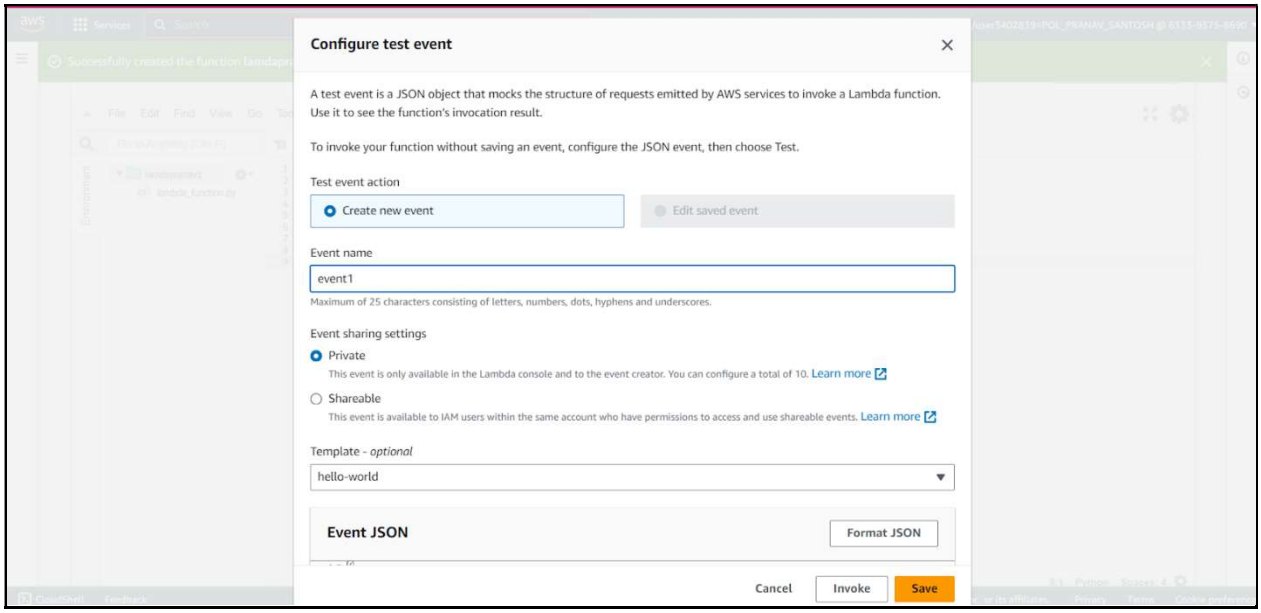


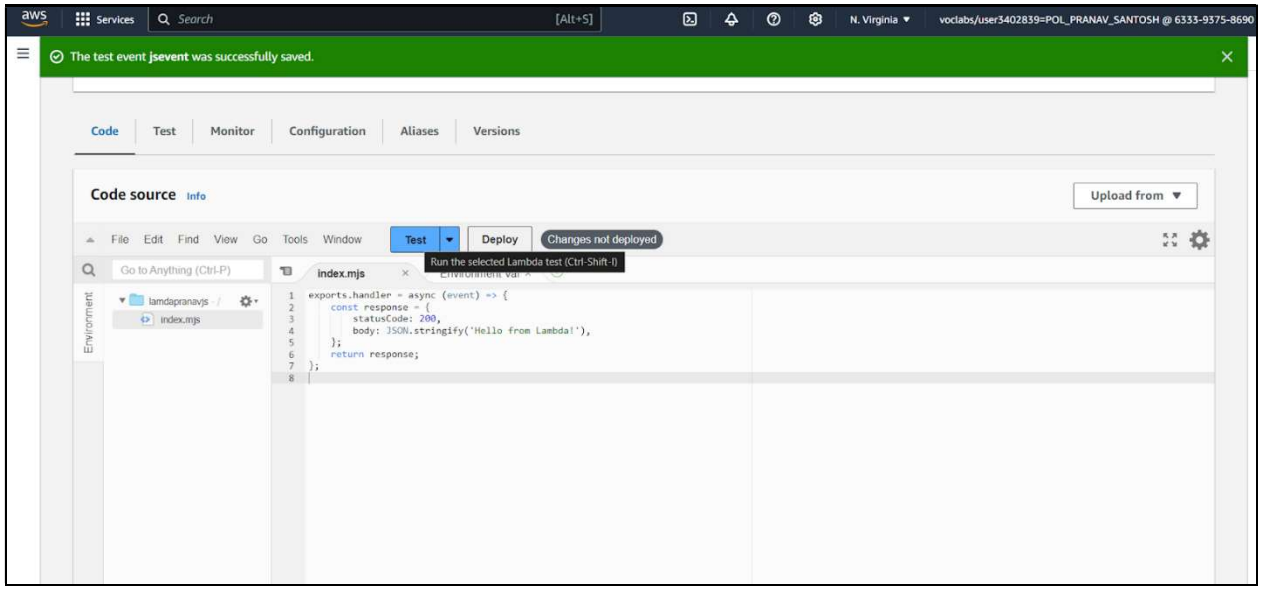
Edit Basic Settings:

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



Configuring test event which tiggers when the function is tested


Configure test event
A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, configure the JSON event, then choose Test.
Test event action
Create new event
Edit saved event
Event name
event1
Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.
Event sharing settings
Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more
Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more
Template - optional
hello-world
Event JSON
Format JSON
Cancel
Invoke
Save


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402839=POL_PRANAV_SANTOSH @ 6333-9375-8690
The test event jsevent was successfully saved.
Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Changes not deployed
Run the selected Lambda test (Ctrl-Shift-I)
Go to Anything (Ctrl-P)
lamdapranavjs
index.mjs
Environment
index.mjs
exports.handler = async (event) => {
const response = {
statusCode: 200,
body: JSON.stringify('Hello from Lambda!'),
};
return response;
};

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 NO. 12

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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.

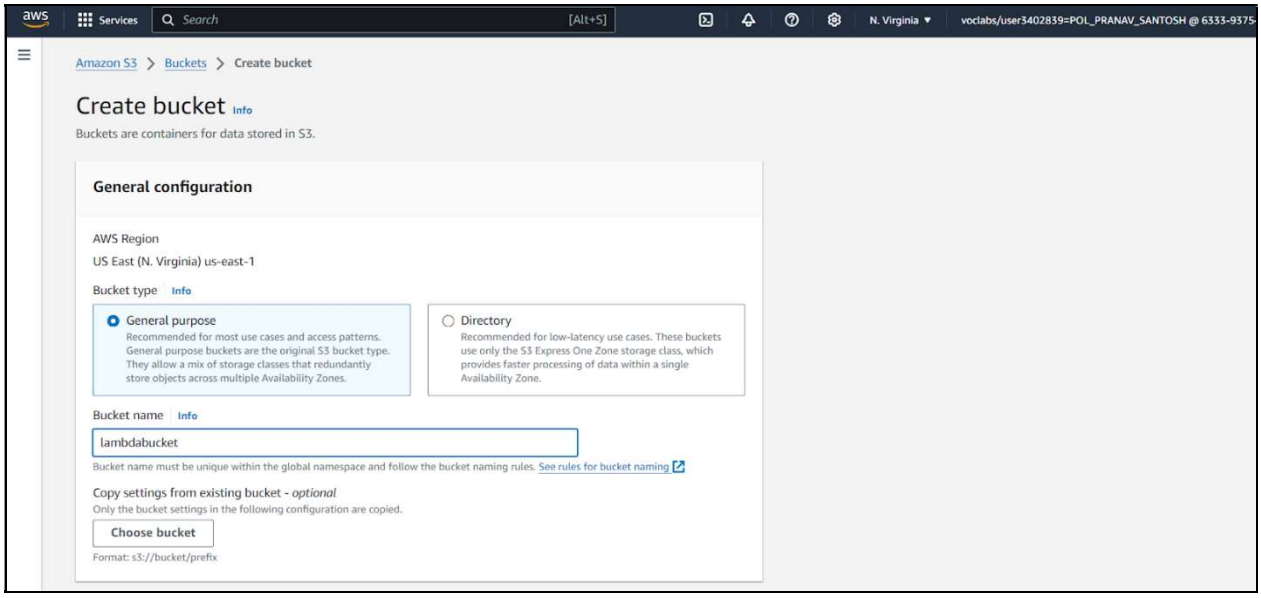
Amazon S3 (Simple Storage Service) is a service offered by AWS that provides object storage through a web service interface. It's designed to store and retrieve any amount of data from anywhere. You can use S3 to store images, videos, backups, data logs, and more.

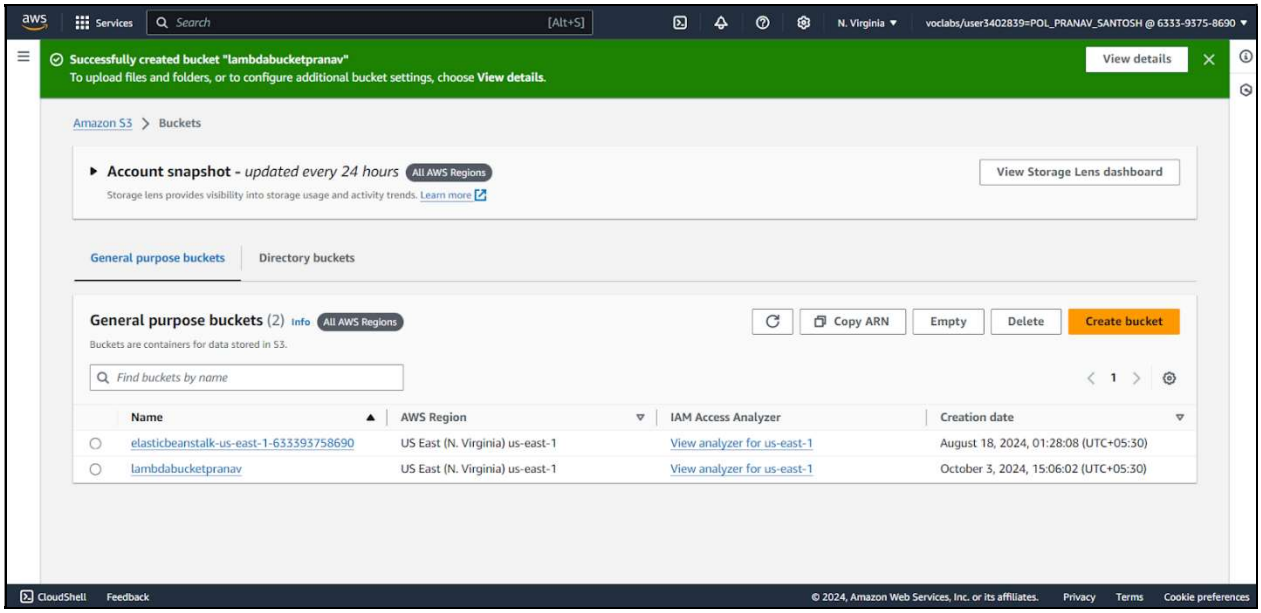
AWS Lambda is a serverless compute service that allows you to run code in response to events without provisioning or managing servers. You write your code and set up a trigger , and Lambda takes care of the rest. This means that when a specified event occurs, such as an object being added to an S3 bucket, 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.

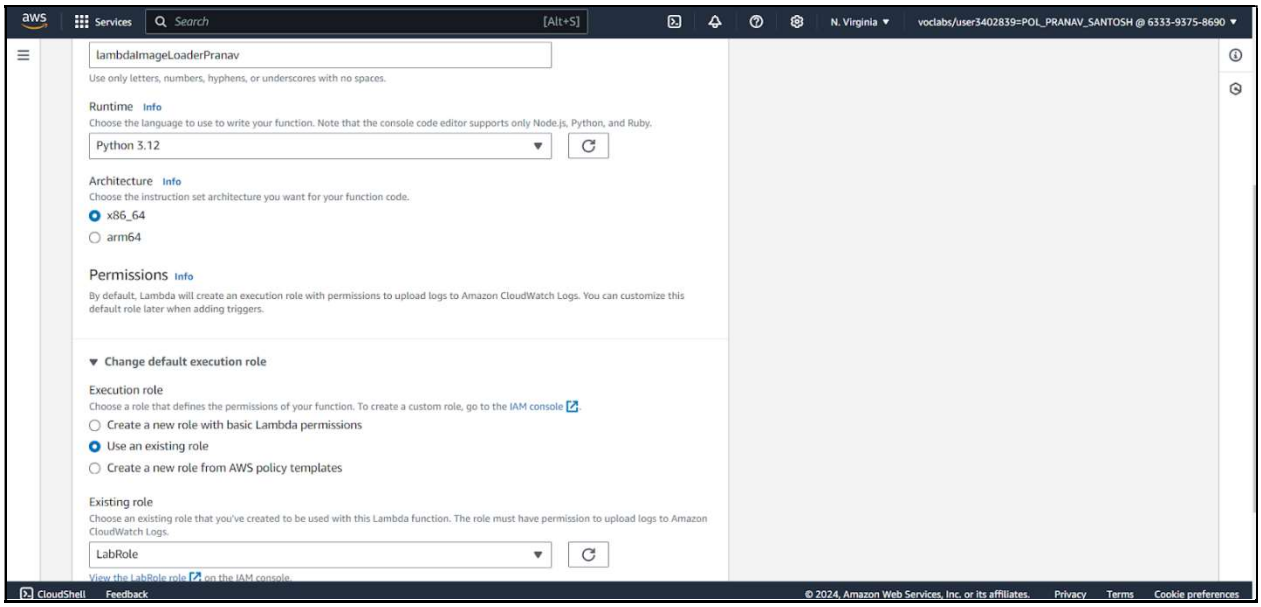
This kind of setup is highly scalable, reliable, and costeffective, leveraging AWS's robust infrastructure. It's particularly useful for applications that require automated monitoring and logging of uploads for auditing or notification purposes.

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.**



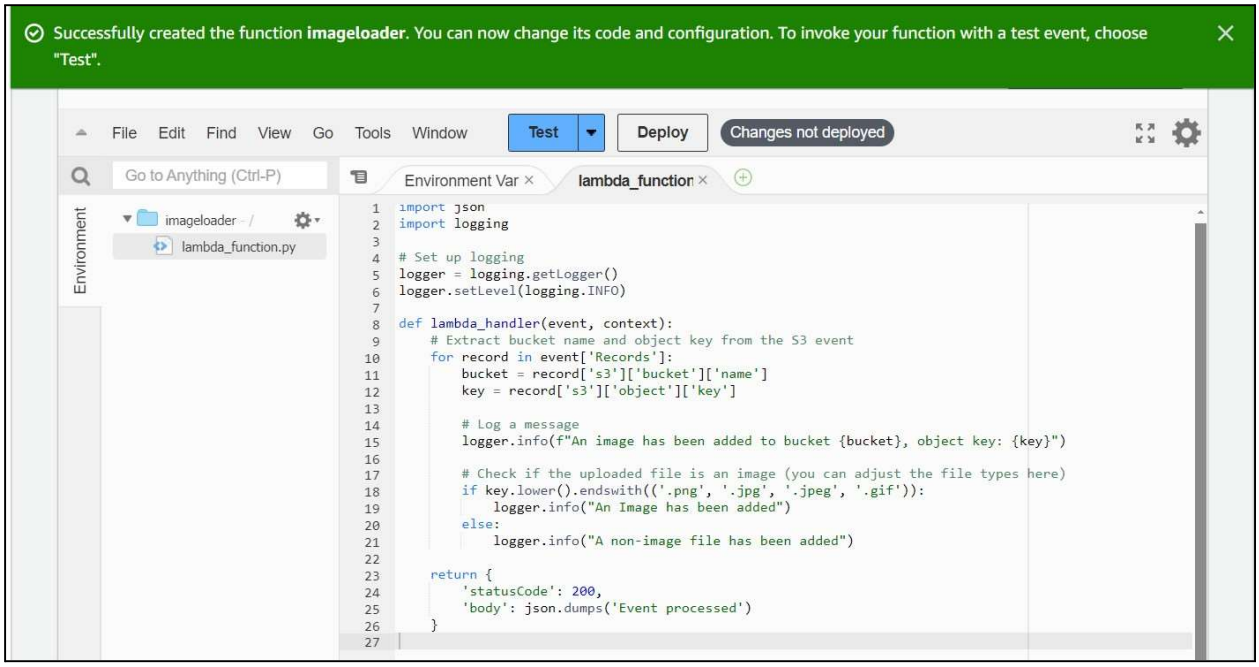


2. Create the Lambda Function: Set up a new Lambda function using AWS Lambda's console. You can choose a runtime environment like Python, Node.js, or Java. Write code that logs a message like "An Image has been added" when triggered

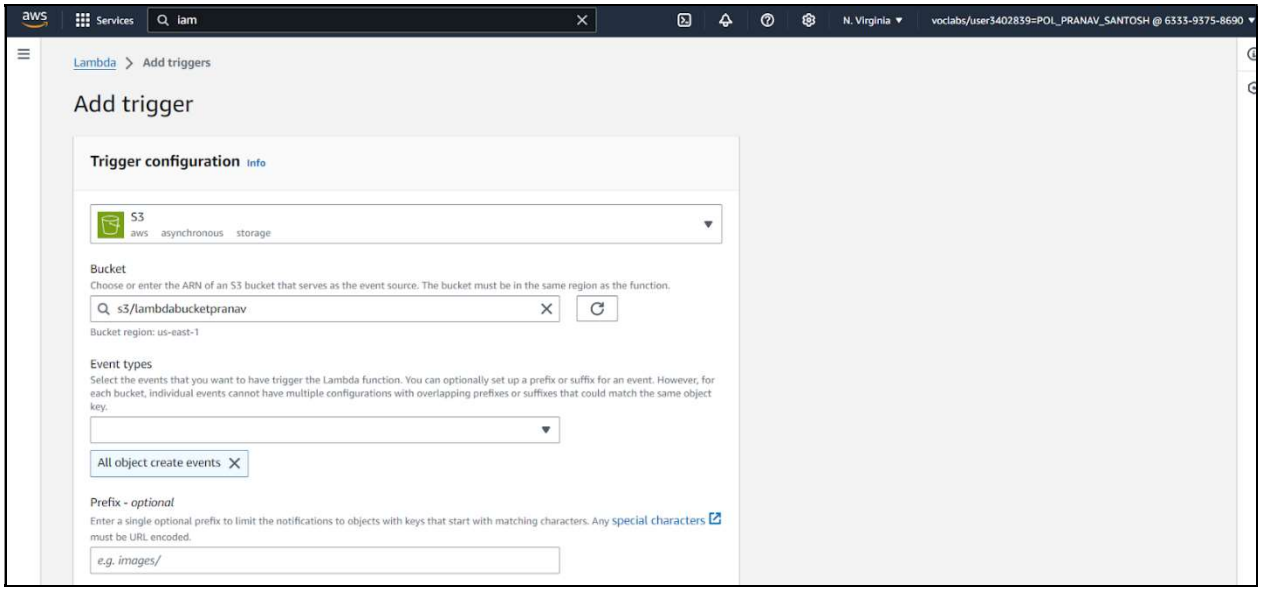


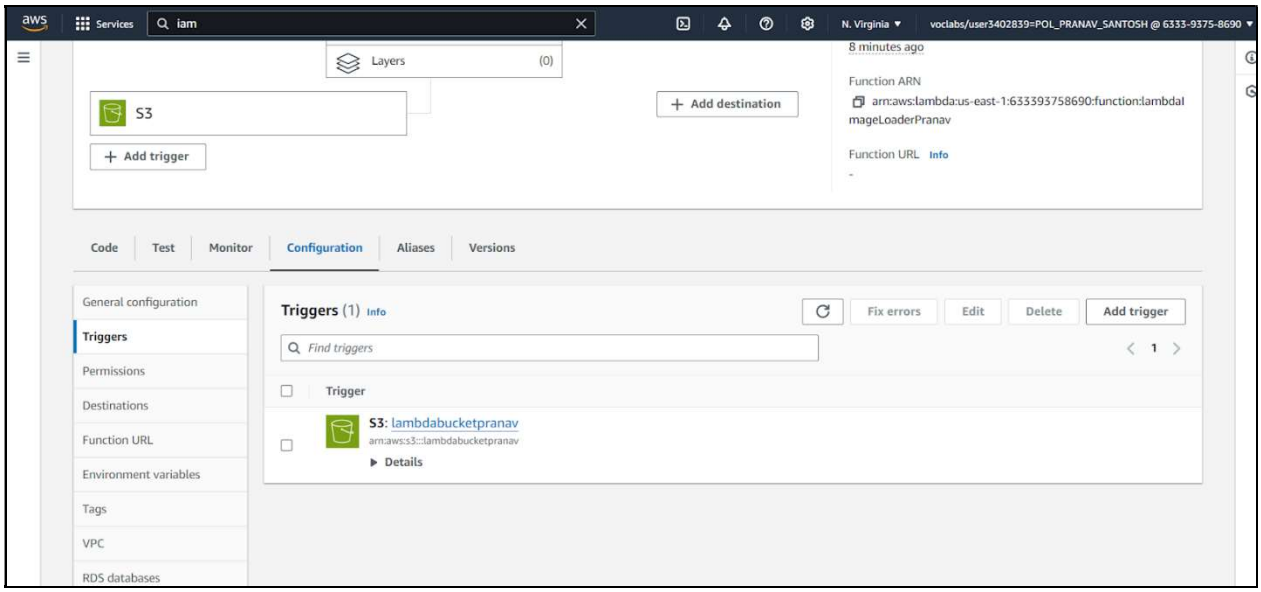

```
import json
import
logging

# 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')
    }
```

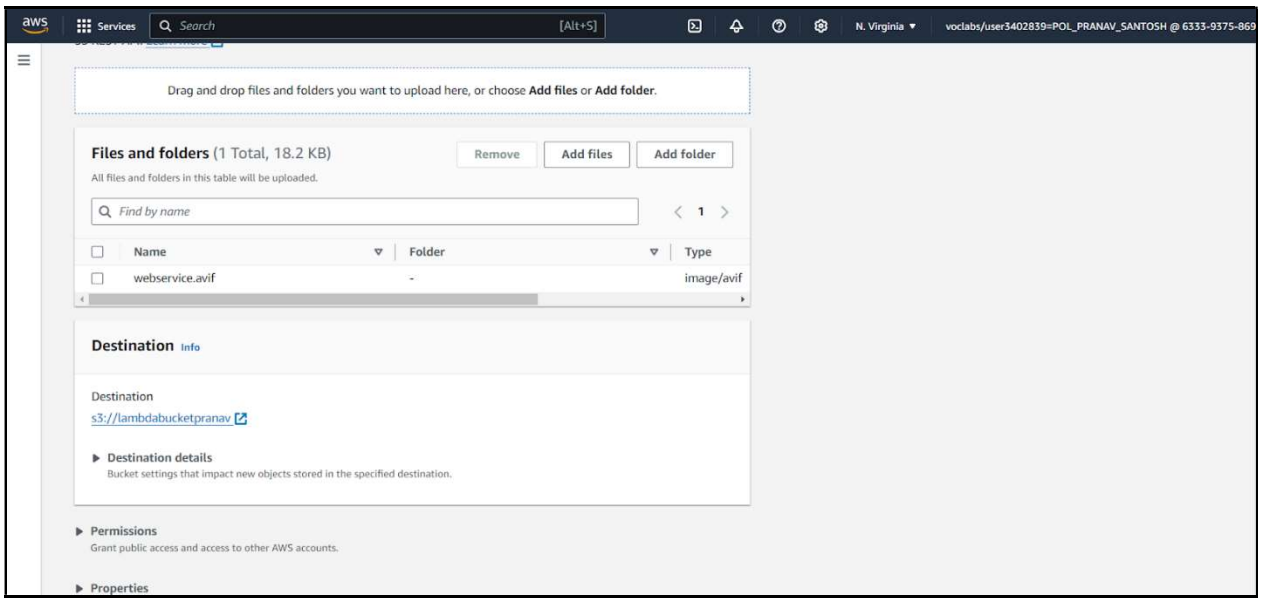


Configure S3 Trigger: Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

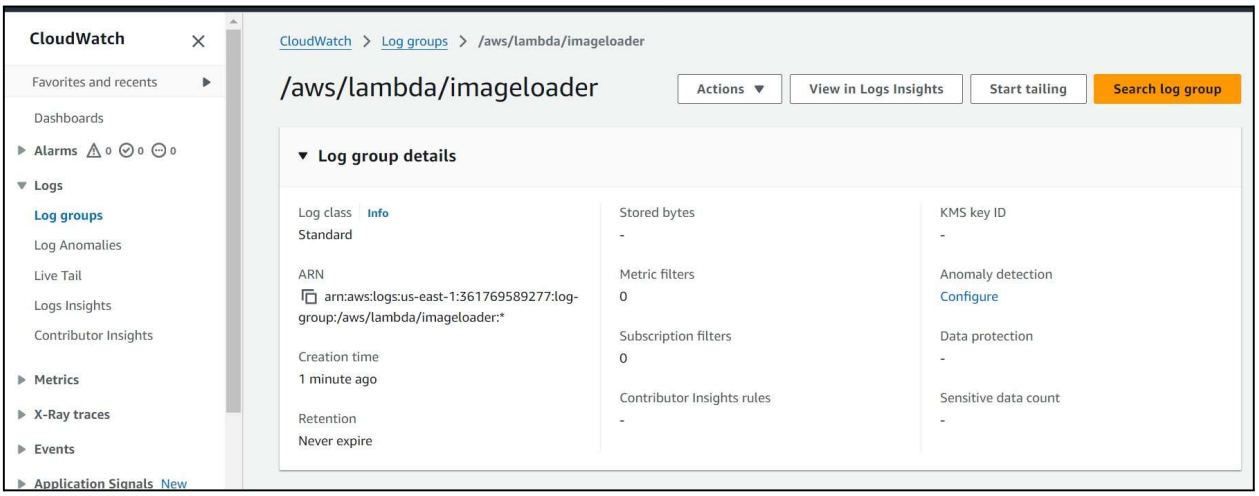


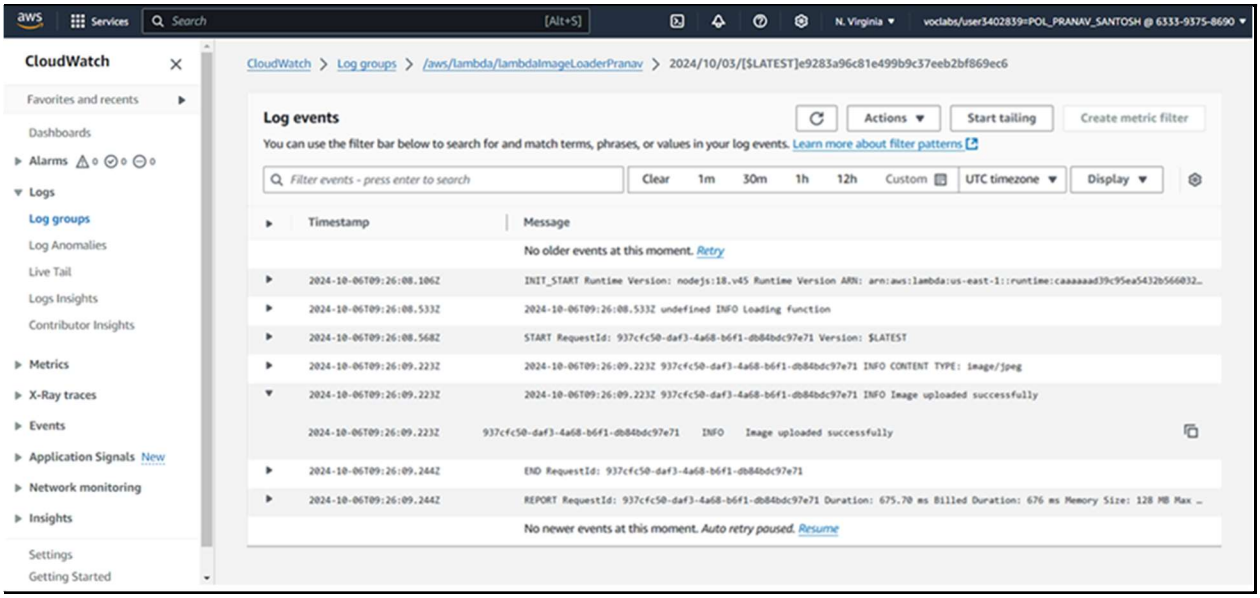


Upload an object (e.g., an image) to the S3 bucket to test the trigger



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 Log


CloudWatch
Favorites and recents
Dashboards
Alarms 0 0 0
Logs
Log groups
Log Anomalies
Live Tail
Logs Insights
Contributor Insights
Metrics
X-Ray traces
Events
Application Signals New
CloudWatch > Log groups > /aws/lambda/imageloader
/aws/lambda/imageloader
Actions
View in Logs Insights
Start tailing
Search log group
Log group details
Log class Info
Standard
ARN
arn:aws:logs:us-east-1:361769589277:log-group:/aws/lambda/imageloader:*
Creation time
1 minute ago
Retention
Never expire
Stored bytes
-
Metric filters
0
Subscription filters
0
Contributor Insights rules
-
KMS key ID
-
Anomaly detection
Configure
Data protection
-
Sensitive data count
-


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402839=POL_PRANAV_SANTOSH @ 6333-9375-8690
CloudWatch
Favorites and recents
Dashboards
Alarms 0 0 0
Logs
Log groups
Log Anomalies
Live Tail
Logs Insights
Contributor Insights
Metrics
X-Ray traces
Events
Application Signals New
Network monitoring
Insights
Settings
Getting Started
CloudWatch > Log groups > /aws/lambda/lambdaImageLoaderPranav > 2024/10/03/[$LATEST]e9283a96c81e499b9c37eeb2bf869ec6
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-10-06T09:26:08.106Z INIT_START Runtime Version: nodejs:18.v45 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:caaaaaad39c95ea5432b566032...
2024-10-06T09:26:08.533Z 2024-10-06T09:26:08.533Z undefined INFO Loading function
2024-10-06T09:26:08.568Z START RequestId: 937cfc50-daf3-4a68-b6f1-db84bdc97e71 Version: $LATEST
2024-10-06T09:26:09.223Z 2024-10-06T09:26:09.223Z 937cfc50-daf3-4a68-b6f1-db84bdc97e71 INFO CONTENT TYPE: image/jpeg
2024-10-06T09:26:09.223Z 2024-10-06T09:26:09.223Z 937cfc50-daf3-4a68-b6f1-db84bdc97e71 INFO Image uploaded successfully
2024-10-06T09:26:09.223Z 937cfc50-daf3-4a68-b6f1-db84bdc97e71 INFO Image uploaded successfully
2024-10-06T09:26:09.244Z END RequestId: 937cfc50-daf3-4a68-b6f1-db84bdc97e71
2024-10-06T09:26:09.244Z REPORT RequestId: 937cfc50-daf3-4a68-b6f1-db84bdc97e71 Duration: 675.70 ms Billed Duration: 676 ms Memory Size: 128 MB Max ...
No newer events at this moment. Auto retry paused. Resume

37

Pranav Pol D15A

05/05

ADV. DevOps

Assignment - 1

Q.1 Use S3 bucket and host video Streaming

Step 1: Create an S3 bucket

1. Sign in to AWS management Console
2. Navigate to S3:
   - In AWS Management Console, Select S3.
3. Create a Bucket
   - Click on Create bucket
   - Enter a unique bucket name.

Step 2: Upload video to S3 Bucket.

1. Open your Bucket by clicking on bucket name your Created.
2. upload Files
   - Click on upload
   - Drag and drop your files and Click upload.
3. Set permissions
   - For public access under permissions, Check Grant Public read access.

Step 3: Create a Cloud Front Distribution.

1. Navigate to Cloud Front from AWS Console.
2. Click on Create distribution.
   - Choose web as delivery method
3. Configure the distribution.
   - Origin Domain name: select your S3 bucket
   - Viewer Protocol Policy: Choose Redirect HTTP to HTTPS for secure access.

- Cache Behaviour settings : Configure caching
- Click Create distribution.

Step 4: Configure CloudFront for secure Access.

1. Create an origin access Identity (OAI)
   - In CloudFront Origin Console, go to distribution settings.
   - under origins and origin group, click Edit
   - Create a new origin access Identity
2. Update S3 bucket policy
   - Go to your S3 bucket.
   - Click on Permission and then bucket policy
   - Add to policy to grant access to OAI.

Step 5: Access the Video through CloudFront.

1. Get the CloudFront URL
   - In CloudFront Console, Go to your distribution.
   - Copy the Domain Name.
2. Use the URL.
   - Use this URL in your web application to Stream the Video.

Q.2) Discuss BMW and Hot Star Case studies using AWS.

BMW Case Study

BMW Group has been using AWS to innovate and scale its operations globally.

- Cloud Data Hub (CDH): BMW migrated its on premises data lake to AWS, creating a Cloud data hub that Processes and Combines anonymiz data from vehicle sensors and other sources. This helps internal teams develop Customer-facing and internal application more efficiently.
- Generative By using AWS Services like Amazon Sage maker and Amazon Bedrock,

The Cloud Data Hub Processes terabytes of telemetry data from millions of vehicles daily. Providing real-time insights from that help BMW resolve issues before they impact Customer.

HotStar Case Study

HotStar, one of India's leading Streaming platforms, utilizes AWS to handle its massive user base and streaming demands. During high-traffic events like the IPL, HotStar relies on AWS to dynamically scale its infra, ensuring a Seamless viewing experience for millions of Concurrent users. This Scalability is Crucial for maintaining Performance and reliability during peak times.

Hotstar also leverages AWS analytic services to gain insights to user behaviour and preferences. These insights enables Hotstar to to personalize Content and improve user engagement, enhancing overall user experience. This Combination of Scalability, analytics, and security has been instrumental in Hotstar sucess as a leading Stream servie.

Q.3) Why kubernetes and advantages and disadvantages of kubernates. Explain How adidas uses kubernetes.

kubernetes is an open-source Container orchestration platform that customates the deployment, scaling and management of Containerized applications.

1. Scalability : kubernetes Can automatically scale applications up or down based on demand, ensuring optimal resource utilization.
2. Portability : It allows application to run Consistently across different environments, whether on premises, in the cloud or hybrid
3. High availability : kubernetes ensures that application are always available by automatic manging failures and distributing workloads

Advantages

- Kubernetes automates many operational tasks such as deployment, scaling and updates.
- It optimizes resource usage by efficiently managing Containerized applications.
- Kubernetes Can automatically restart failed Containers, replace them and reschedule them when node die.
- It provides load balancing to manage micro services.

DisAdvantages.

- Kubernetes has stepper learning curve And Can be Complex to set up and manage.
- Running Kubernetes Can be resource-intensive requiring significant Computation resources

How addidas uses Kubernetes.

Adidas Faced a significant Challenges with their existing ifrastructure. They say that they were happy with software Choices but accessing all tools was a problem. This hindered their development and deployment processes. To overcomes these issues they adopted a Cloud-native approach using kubernetes. This allowed them to Containarize their application leading to greater scalabilit and flexiblity.

By integrating agile development and

Continous delivery process, adidas accelerate their release cycles from 4-6 weeks to multip times a day. The migration resulted in a 50% reduction in load time for their e-commerce site, significantly enhancing the user experience. Currently adidas operates 4000 pods and 200 nodes, handling 80000 builds per month, supporting critical system and global customer base.

Q.4) What are nagios and explain how Nagios are used in E Services.

Nagios is open source monitoring and alerting system widely used for tracking the health and performance of IT infrastructure, network and applications. It provides a flexible and extensible platform for monitoring various components within an organisation.

Nagios are used in e services to ensure the reliability and performance of critical infrastructure. For example, an eCommerce company might use Nagios to monitor health of its server tracking metrics like CPU usage, memory utilization, and disk space to prevent any potential issues. It also keeps an eye on network devices such as routers and switches, ensuring smooth data flow

and detecting anamolies. By monitoring essential services like web services and data bases Nagios can quickly ~~alter~~ alert administrator to any outages or performance problem allowing for immediate intervention. Additionally nagios integrates with log files and security system to detect and alert on suspicious activities, enhancing overall security. By providing Comprehensive monitoring and alerting capabilities, nagios ensures that e-services remain highly avilable, performant, and secure, thereby delivering a seamless experience to users and supporting buisness experience.

ADV. DevOps

Roll No. 42

NAME: Pranav Pol STD.: DIV.: D15 B

DATE :

PAGE :

Assignment No. 2

Q.1 Create a REST API with the serverless Framework.

The serverless Framework is an open source that simplifies the deployment and management of serverless applications. It allows developers to build and deploy applications on Cloud platform like AWS without having to manage the underlying infrastructure.

Step 1 : Install the serverless Framework.
First ensure you have Nodejs and npm installed. Then ensure install the serverless Framework globally
npm install -g serverless

Step 2 : Create a new Serverless Service.
serverless Create --template aws-nodejs --path my-service cd my-service.

Step 3 : Define Function in serverless.yml
open serverless.yml file in project directory.
In this file we define our Service Configuration including the functions and their triggers / events.
This file Contains Service, Provider it name and runtime environment. It also Contains Functions which Contains Create, read, update and delete methods.

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

Step 4: Implement lamda function.
Create a Handler.js file in your project directory. In this file we write the Code for our lambda functions.
This file Contains export function of modules of various CRUD methods.

Step 5: Deploy your Servic and AWS Config
- Configure AWS Credentials using aws Configure.
- Deploy your Servic to aws using Serverless deploy.

This Command packages our services, uploads it to AWS, and Setup necessary infrastructure.

Step 6: Testing our API.
After deployment, the Serverless framework will provide us with the endpoint for our API. We Can test these endpoint using tools like Postman or CURL.
For example, To test the Create function we Can Send the POST request to the provided URL

NAME:_____ STD.:_____ DIV.:_____ DATE : PAGE :

Q.2. Case Study for Sonarqube.

- Create your own profile in Sonarqube for testing project Quality.

Open sonarqube and log into as an administrator Then ~~to~~ Go to administration of Quality profiles Create and name your profile. Based on project requirements add or remove rules and save your new quality Projects.

- Using Sonarcloud for Github Code Analysis

Go to Sonarcloud website and sign up then link your github account to Sonarcloud. ~~go to~~ Then Choose the repositories to analyse then Follow the prompts to set up and run the analysis. By linking our Github repositories directly we achieved Continous Code quality checks with each Commit. The main challenge was Configuring the permissions Correctly to ensure all repositories were accessible.

- Install Sonarlint in your Intellij IDE and analyse your Java Code.

For installing Sonarlint in Intellij Go to file settings ~~and~~ then navigate to plugins and click on market place. In marketplace search

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

for Sonarlint and install it. Then we have restart IntelliJ IDEA to activate the plugin. Installing Sonarlint in our IDE's provided us real-time feedback on Code quality issues as we developed.

- Analyse a python Project with SonarQube.

For this we need sonar-scanner-cli we are installing it as a docker image Create sonar-Project.properties in your Project directory and add the properties this includes project key, host URL, and login token. then analyze python project via Command line and integrate this step into our CI pipeline.

- Analyze Node.js project with SonarQube.

As we have already got image then we need to install npm package of Sonar Scanner in our Node.js project. The main Challenge was Sonar-project.properties file to ensure all necessary parameter were included.

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

Q.3.

A Self Serve infrastructure model allows individu product teams within an organisation to provision and manage their infrastructure without the need for a Centralised operations team. This is done by Creating reusable Terraform modules that encapsulates the organisation's standard and best practices. Terraform enables Teams to define their infrastructure as Code, automating the provisioning and life cycle management of Cloud resources.

Terraform modules are reusable, versioned, and shareable Components that define infrastructured resources. They encapsulate best practices and standards for deploying and managing Services. For example, an organisation may require all its resources to have Specific tagging, use Certain Certain group, or enforce encryption on Storage resources. By using Terraform modules, these requirements can be Codified, ensuring Compliance. Terraform Cloud is a managed Service that provid for running Terraform Commands remotely, enabling infrastructure as Code workflows. With integration into ticketing Services System like ServiceNow, infrastructure request Can be automatically generated based on Service tickets, Streamlining the provisioning process.

NAME:_____ STD.:_____ DIV.:_____ DATE : PAGE :

The ticketing system integration works as A user submits a request for infrastructure through a ServiceNow Catalog form item. The Catalog form item include fields for necessary details such as the type of infrastructure, Configuration option. Service now automatically generates a ticket based on the user request This ticket goes through the approval workflow It Can be automated based on predifined rules. once approved the ticket trigger a Terraform run. Service Now uses the information in ticket to execute the Corresponding Terraform module. After this terrayerm make provision for requested modules. as Specified in Ticket. If the infrastructure is Successfully provisioned the ticket is marked as Completed. If there are error the ticket is updated with the error details for further action.