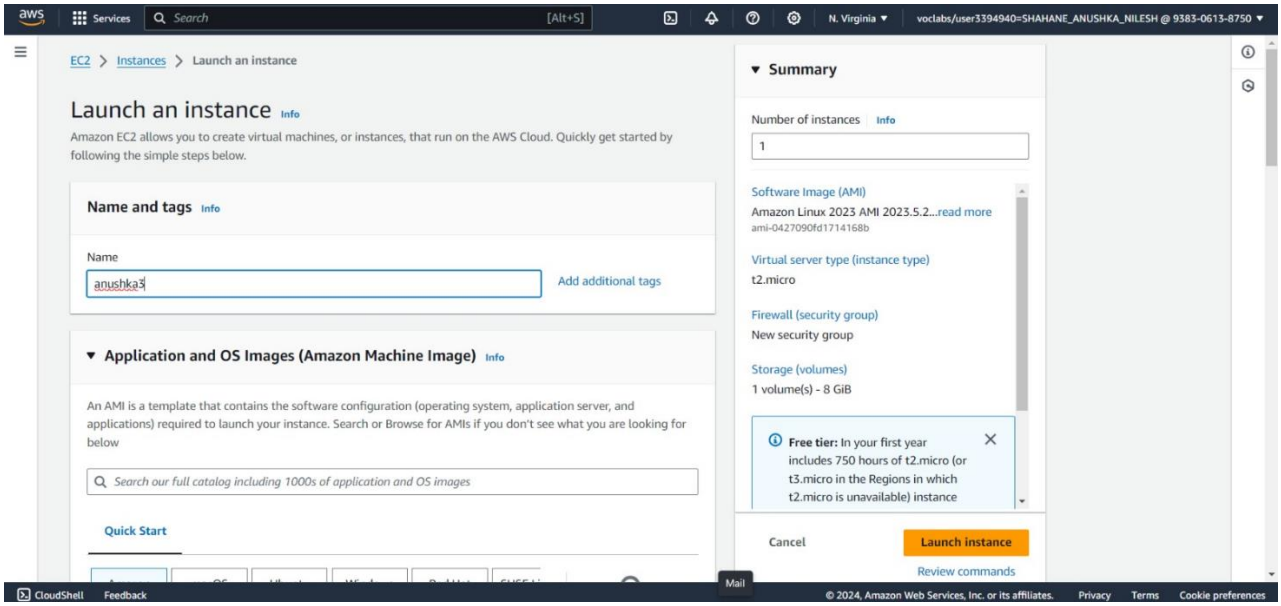
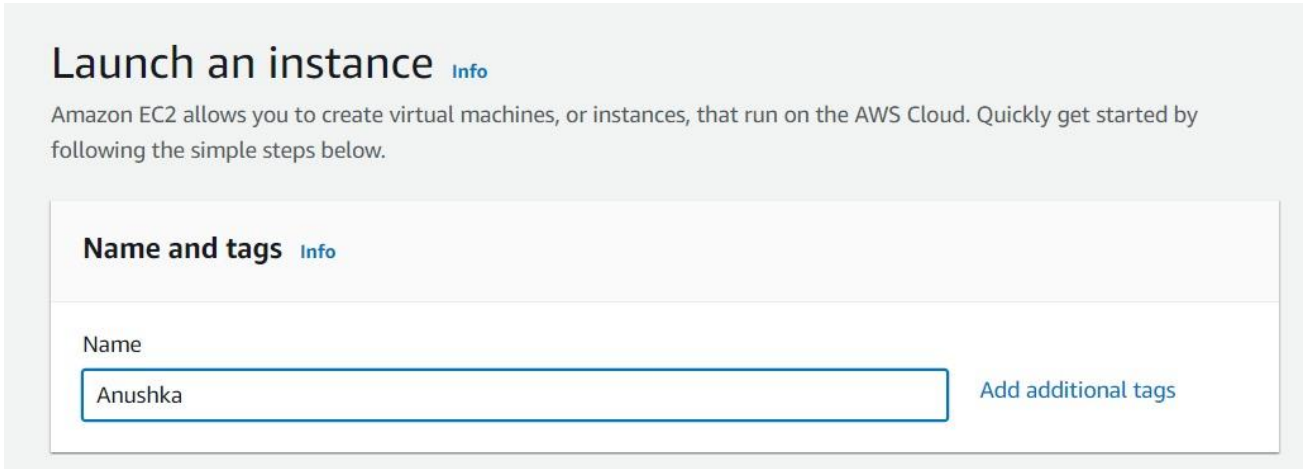
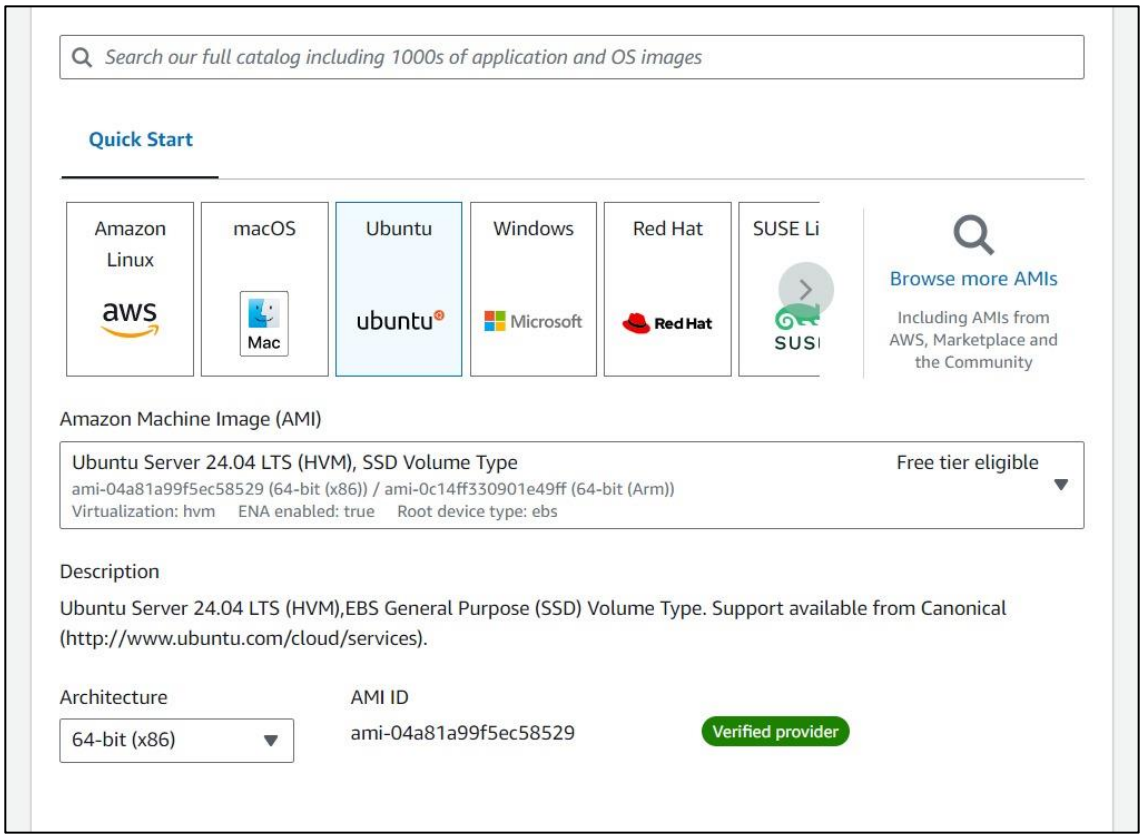
ADVANCE DEVOPS EXPERIMENT 1

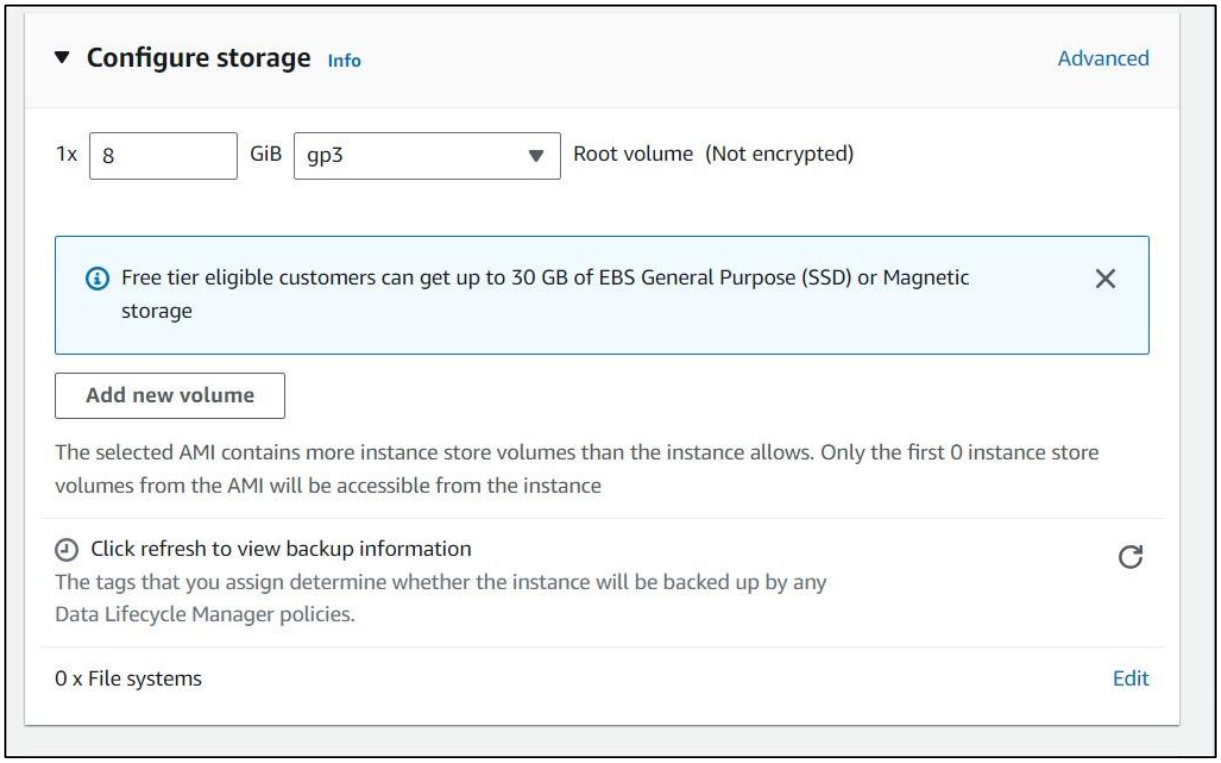
Aim : To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.



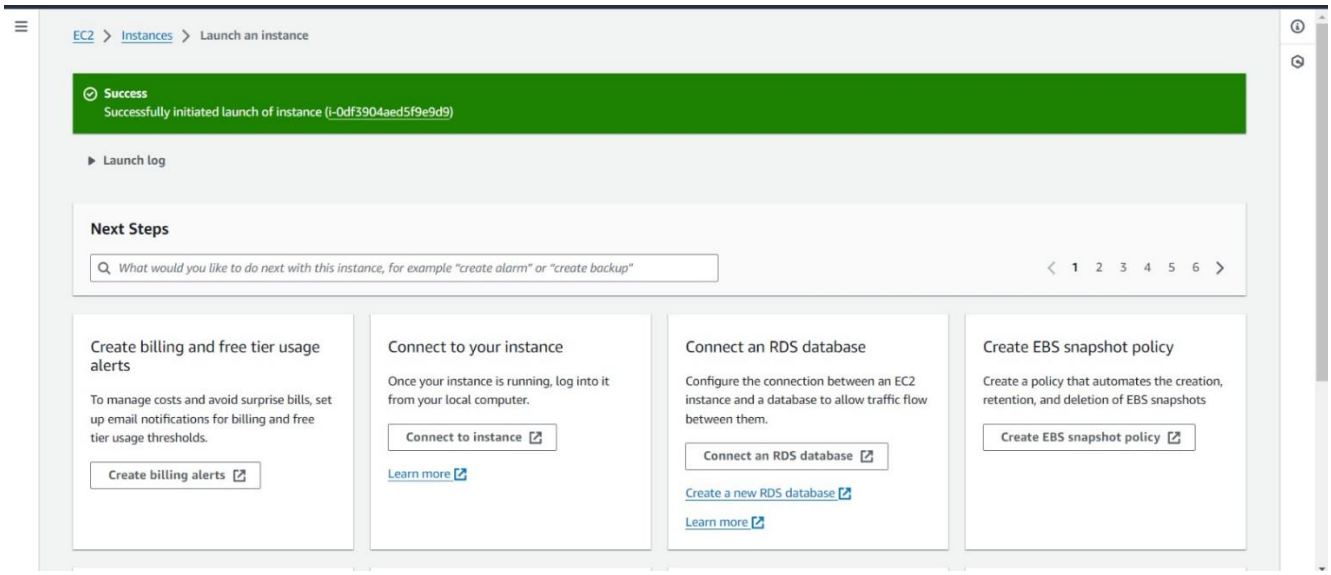


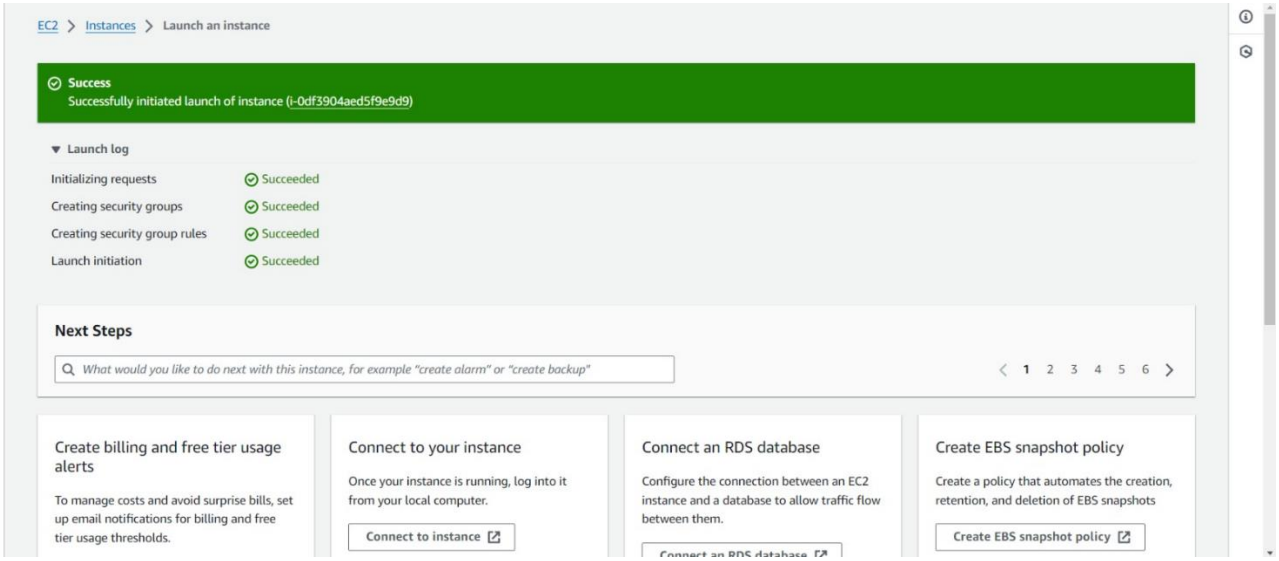
Anushka Shahane D15A 55



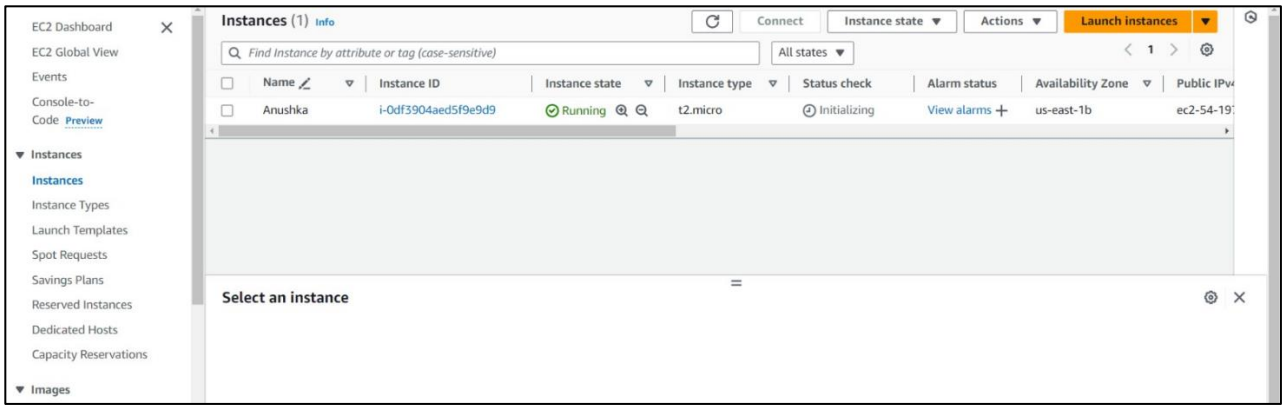


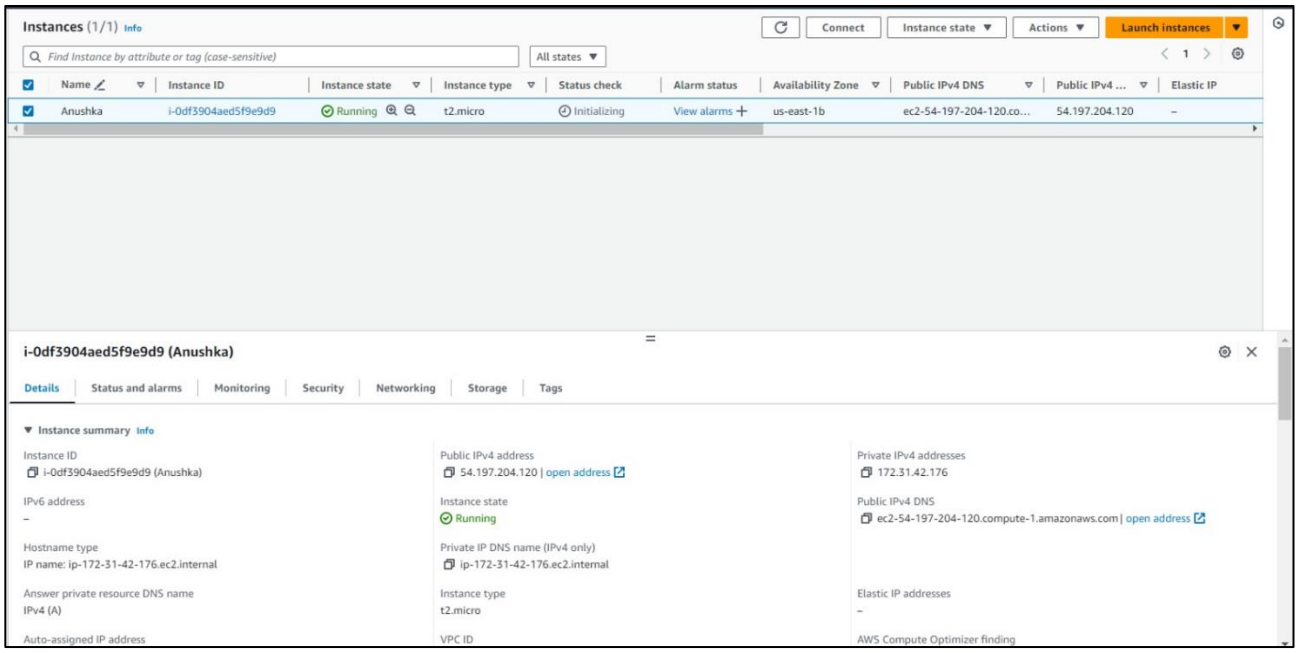
Anushka Shahane D15A 55





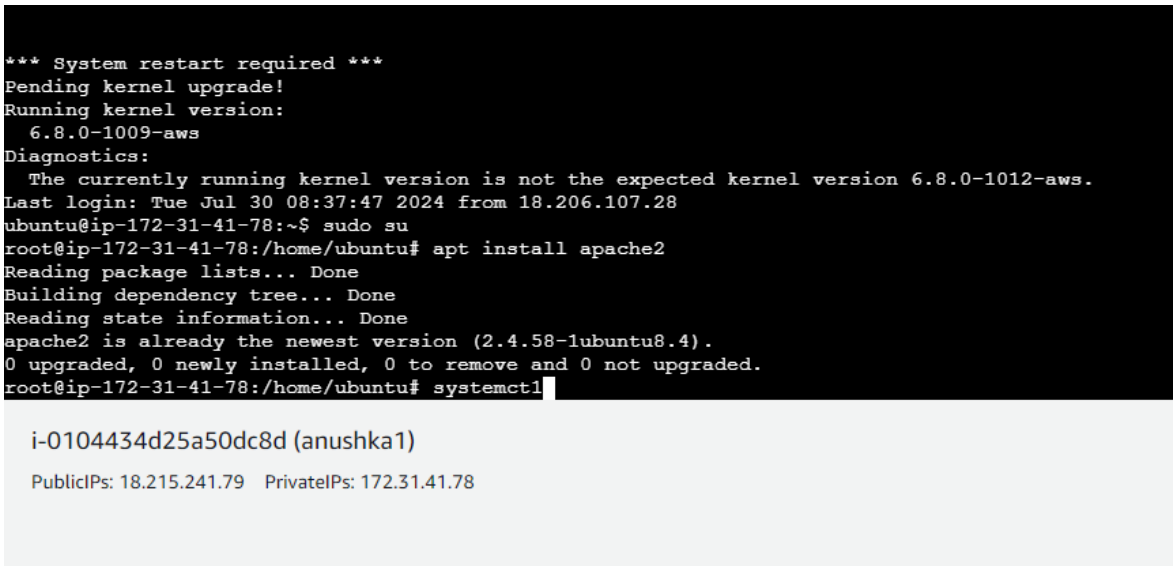
Anushka Shahane D15A 55

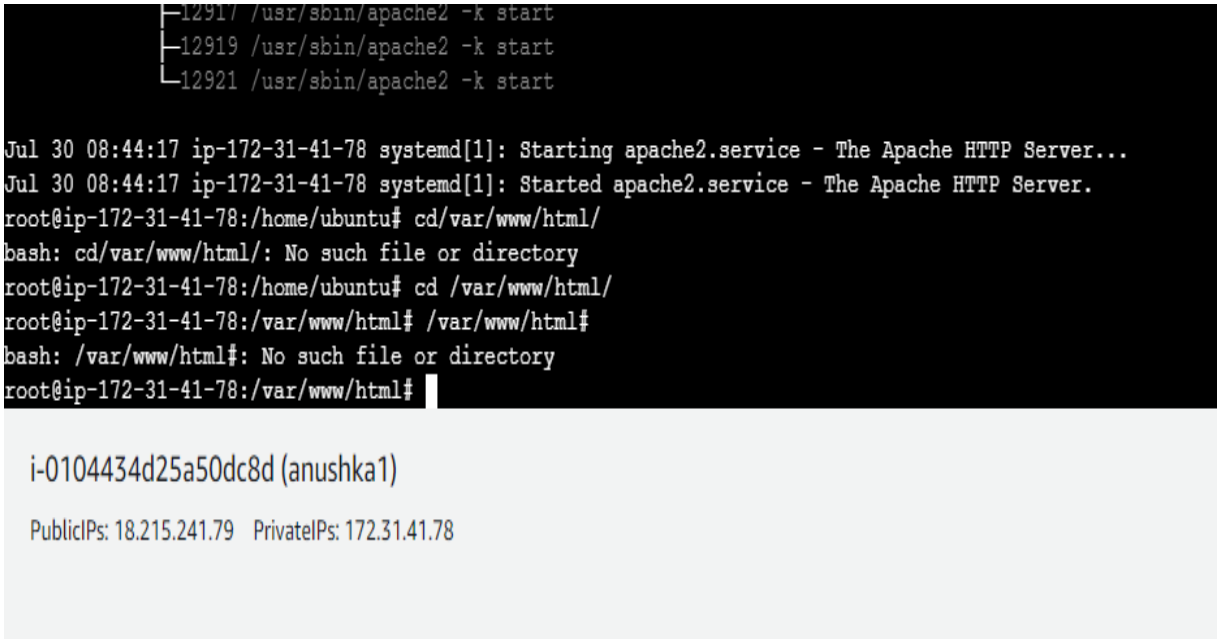





```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-42-176:~$ ls
ubuntu@ip-172-31-42-176:~$ echo "hello"
hello
ubuntu@ip-172-31-42-176:~$ cat > myfile.txt
This is Advance devops lab
^C
ubuntu@ip-172-31-42-176:~$ cat myfile
cat: myfile: No such file or directory
ubuntu@ip-172-31-42-176:~$ cat myfile.txt
This is Advance devops lab
ubuntu@ip-172-31-42-176:~$
```

Hosting a static website using EC2 instance:



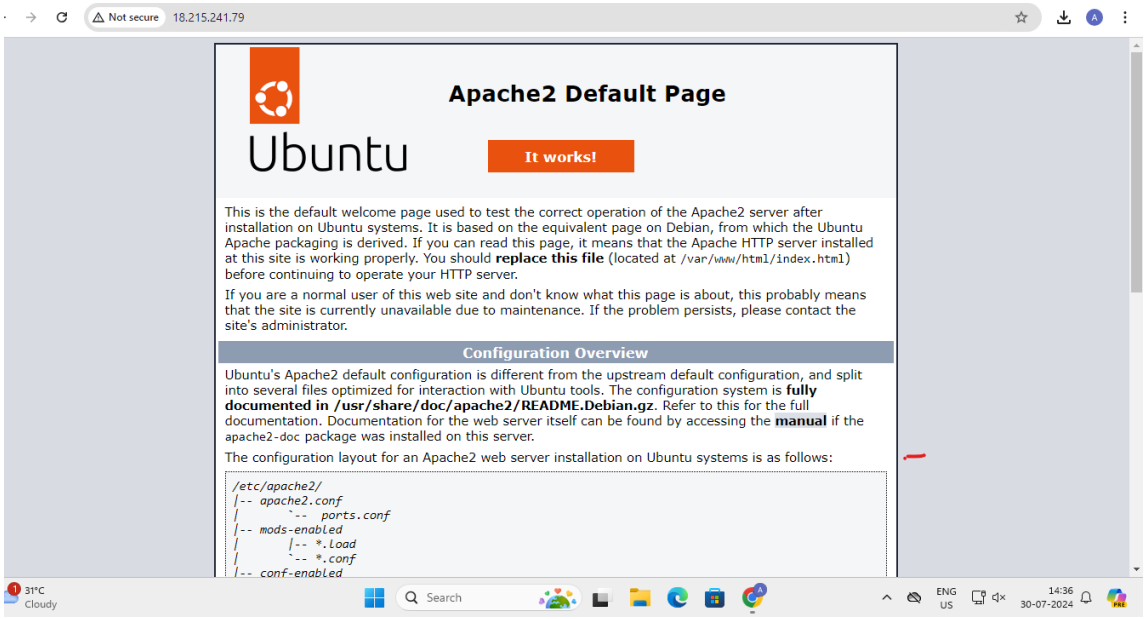


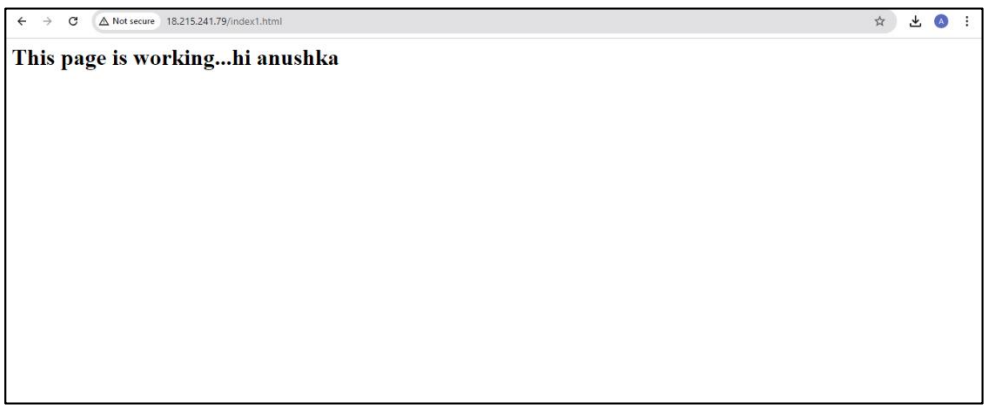
Anushka Shahane D15A 55

```
  command 'systemctl' from deb systemctl (1.4.4181-1.1)
Try: apt install <deb name>
root@ip-172-31-41-78:/home/ubuntu# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-07-30 08:44:17 UTC; 12min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 12917 (apache2)
      Tasks: 55 (limit: 1130)
     Memory: 5.3M (peak: 5.4M)
        CPU: 74ms
     CGroup: /system.slice/apache2.service
             ├─12917 /usr/sbin/apache2 -k start
             ├─12919 /usr/sbin/apache2 -k start
             └─12921 /usr/sbin/apache2 -k start

Jul 30 08:44:17 ip-172-31-41-78 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Jul 30 08:44:17 ip-172-31-41-78 systemd[1]: Started apache2.service - The Apache HTTP Server.
root@ip-172-31-41-78:/home/ubuntu#
```

i-0104434d25a50dc8d (anushka1)

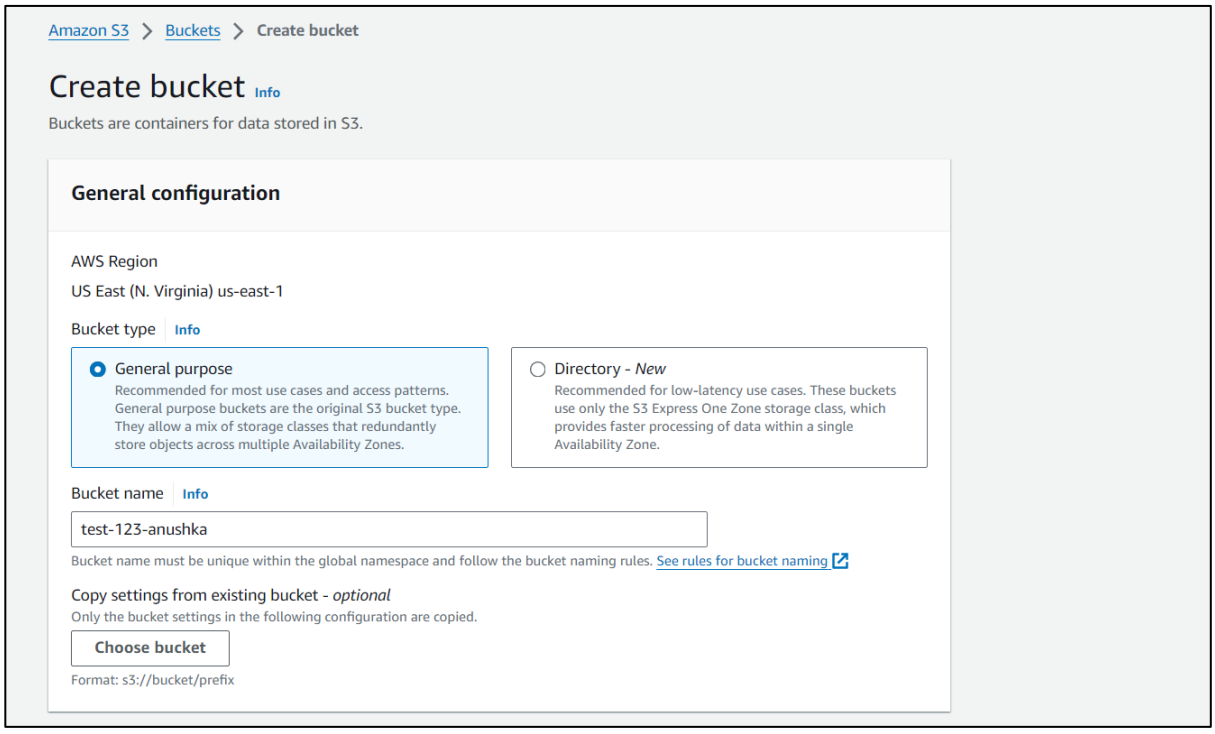
PublicIPs: 18.215.241.79 PrivateIPs: 172.31.41.78

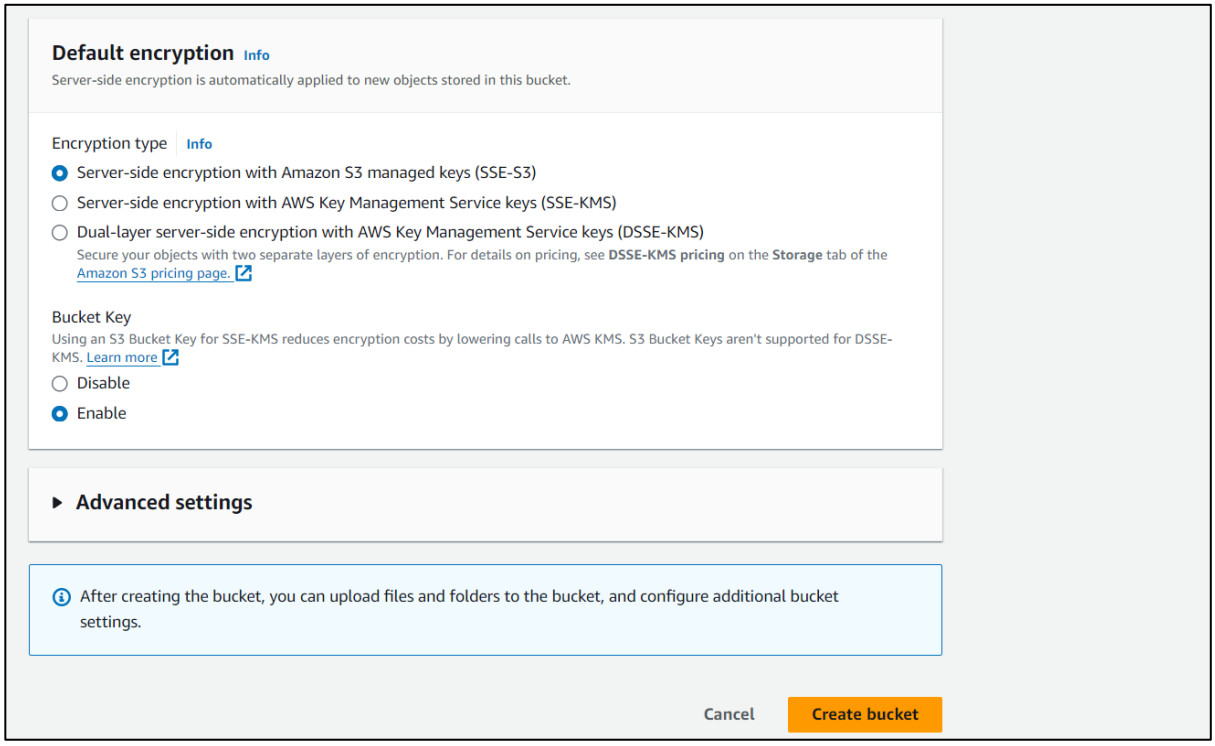




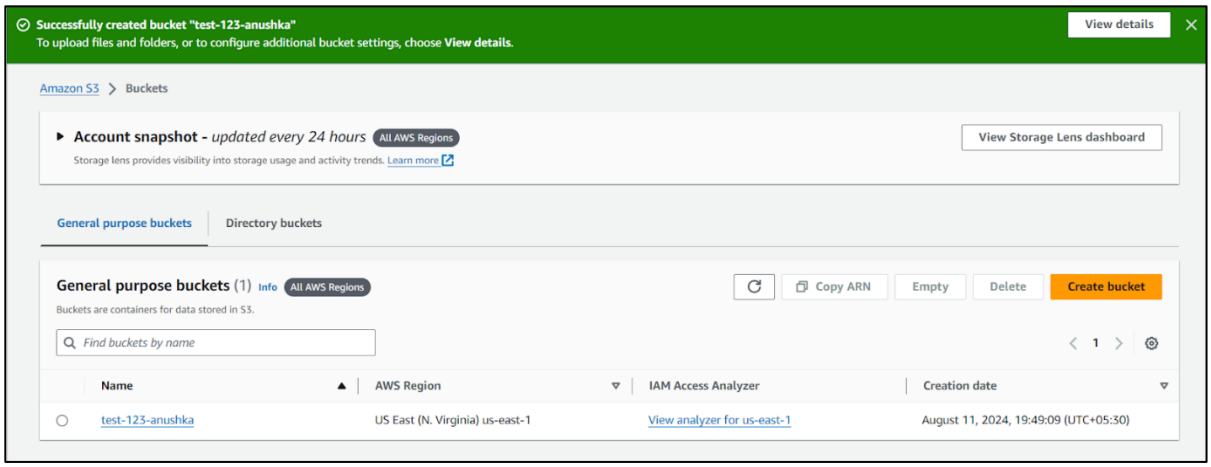
Anushka Shahane D15A 55

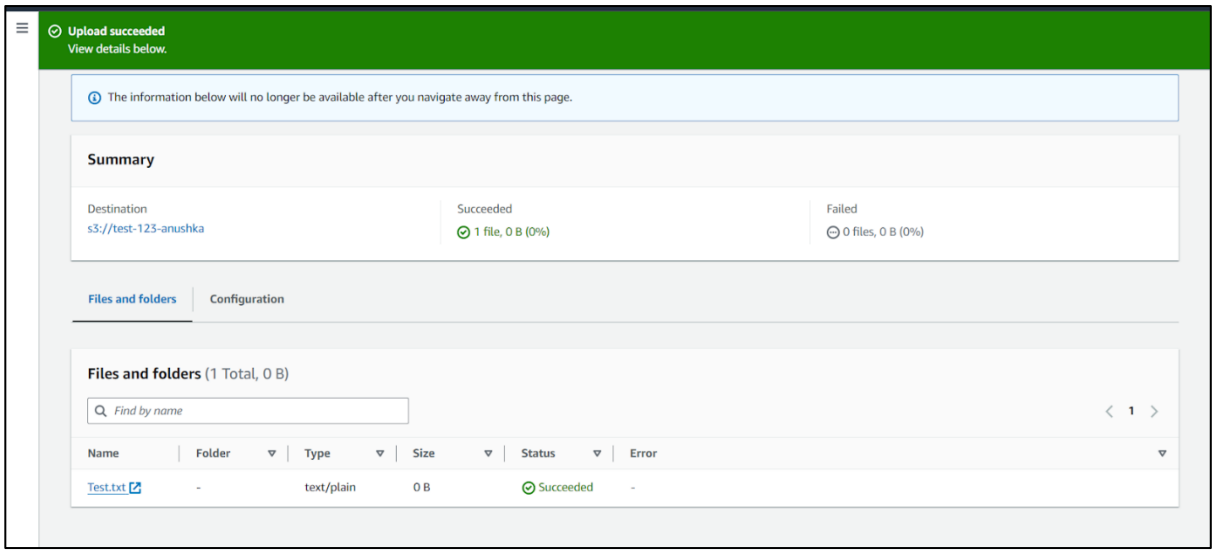
Hosting using S3 bucket :

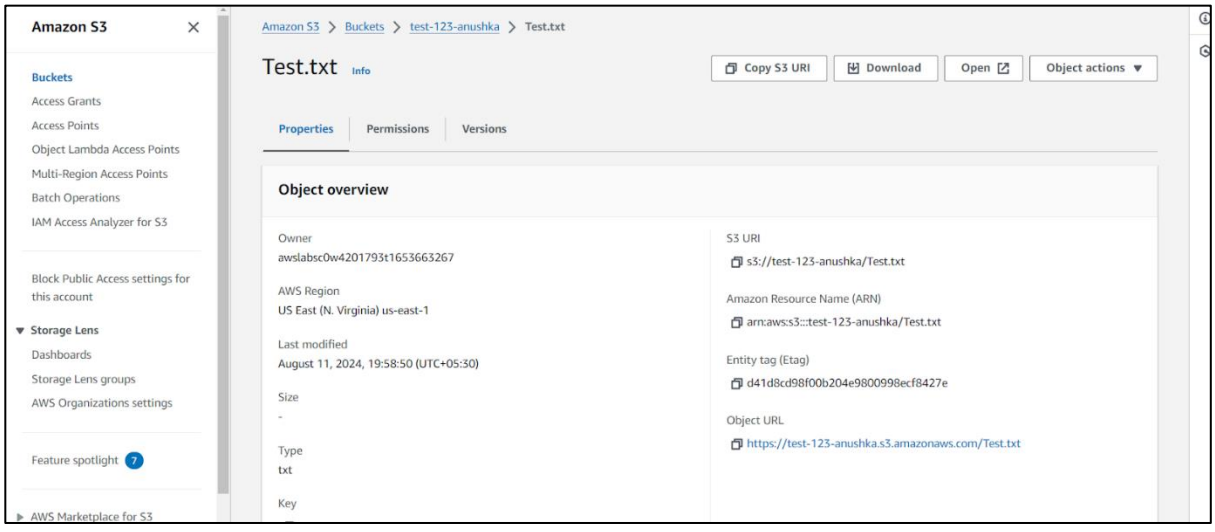


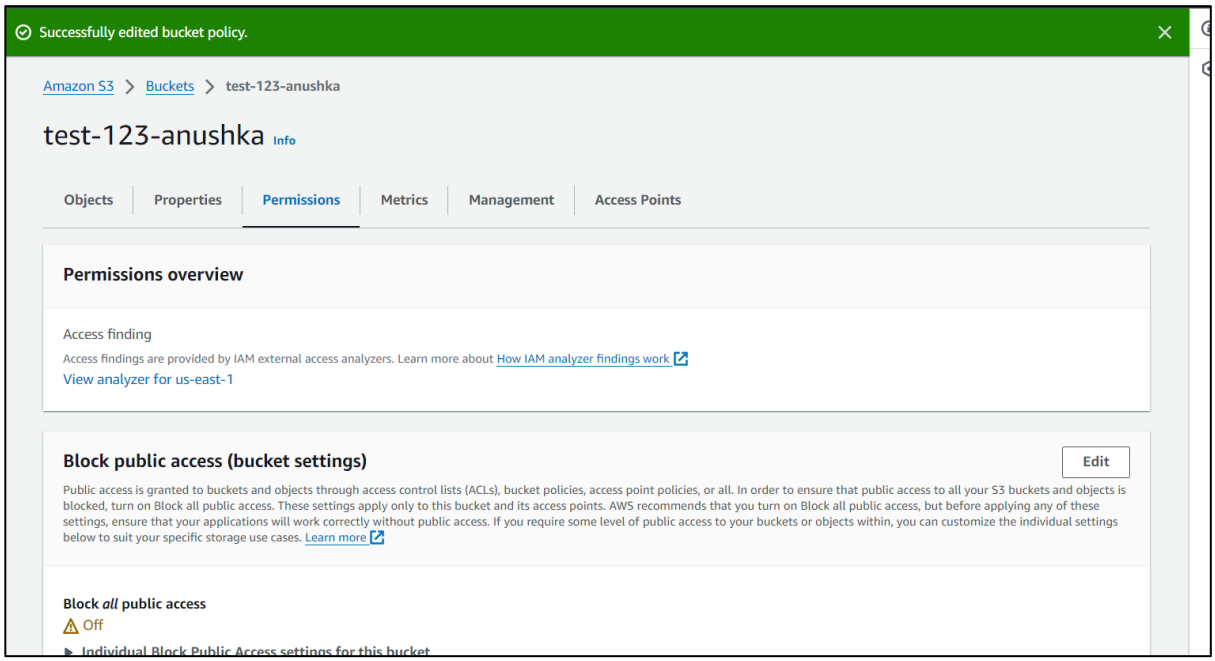


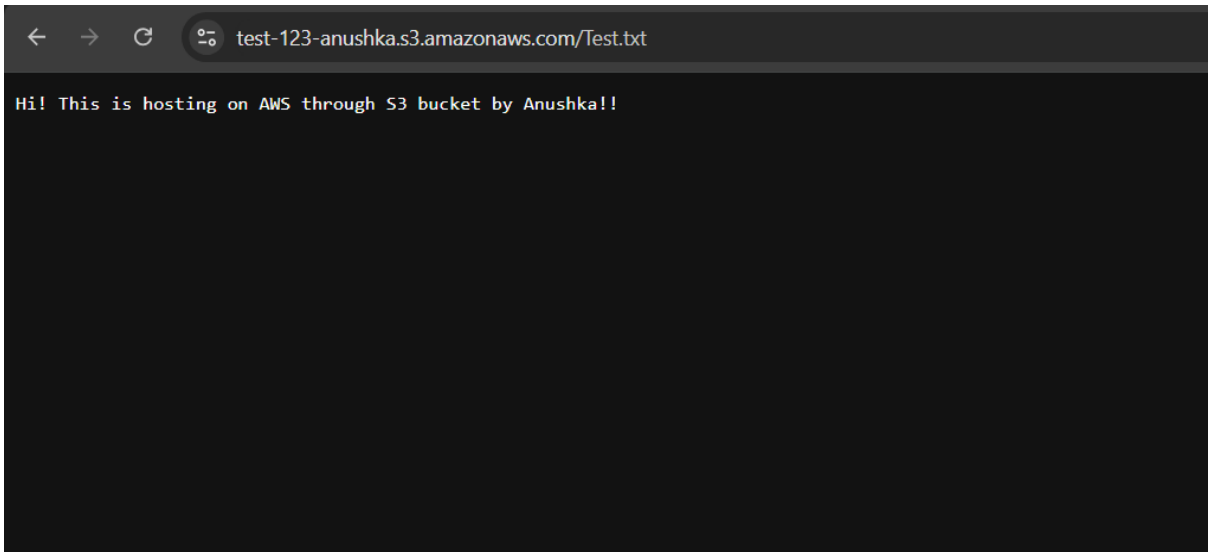
Anushka Shahane D15A 55



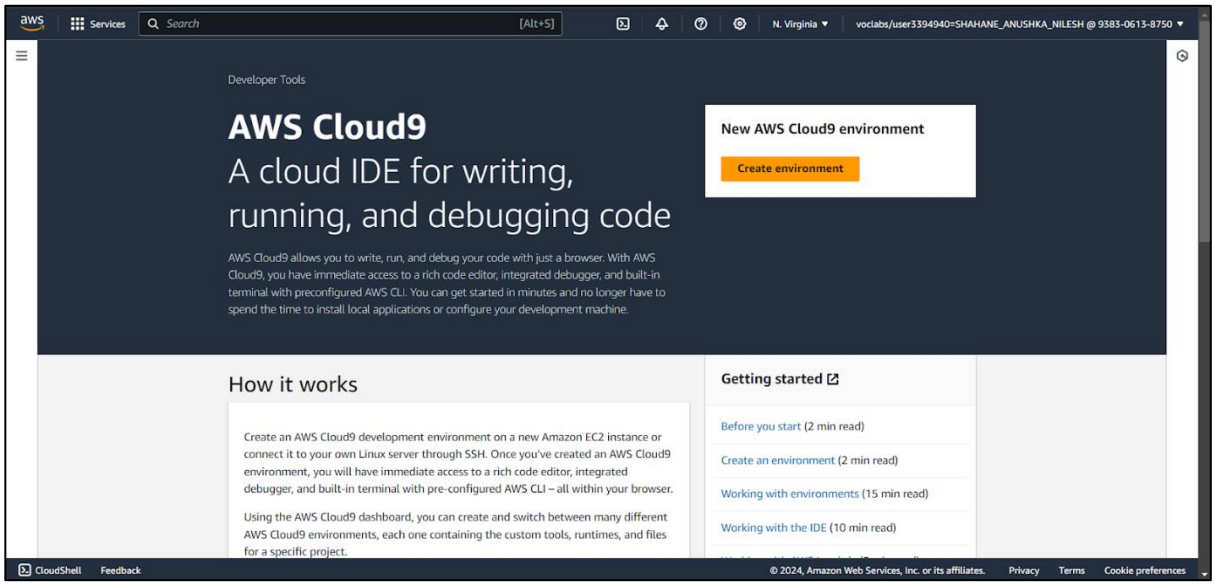


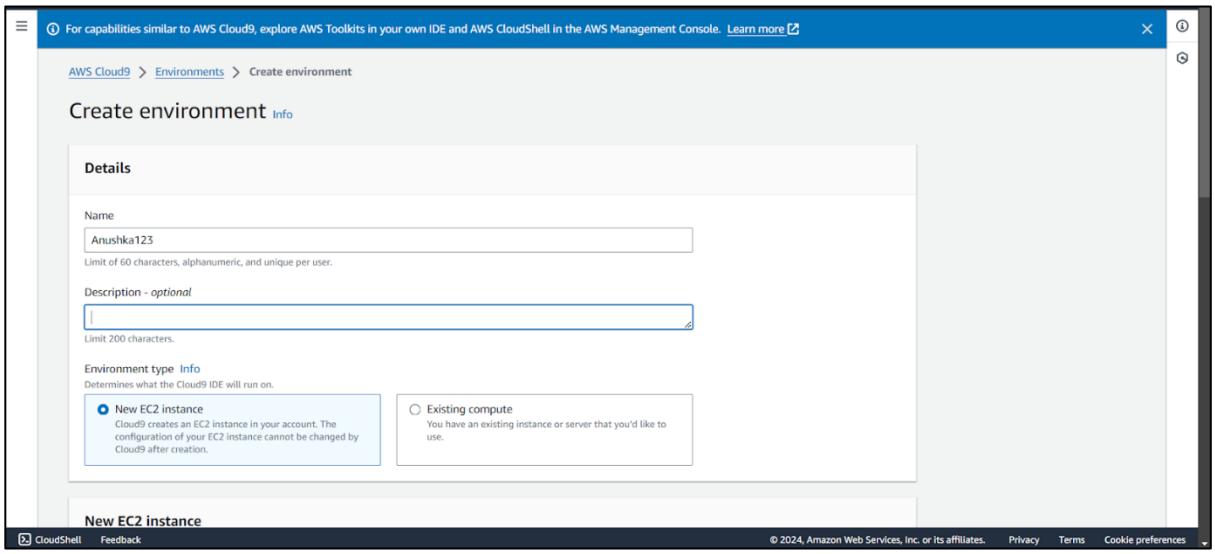


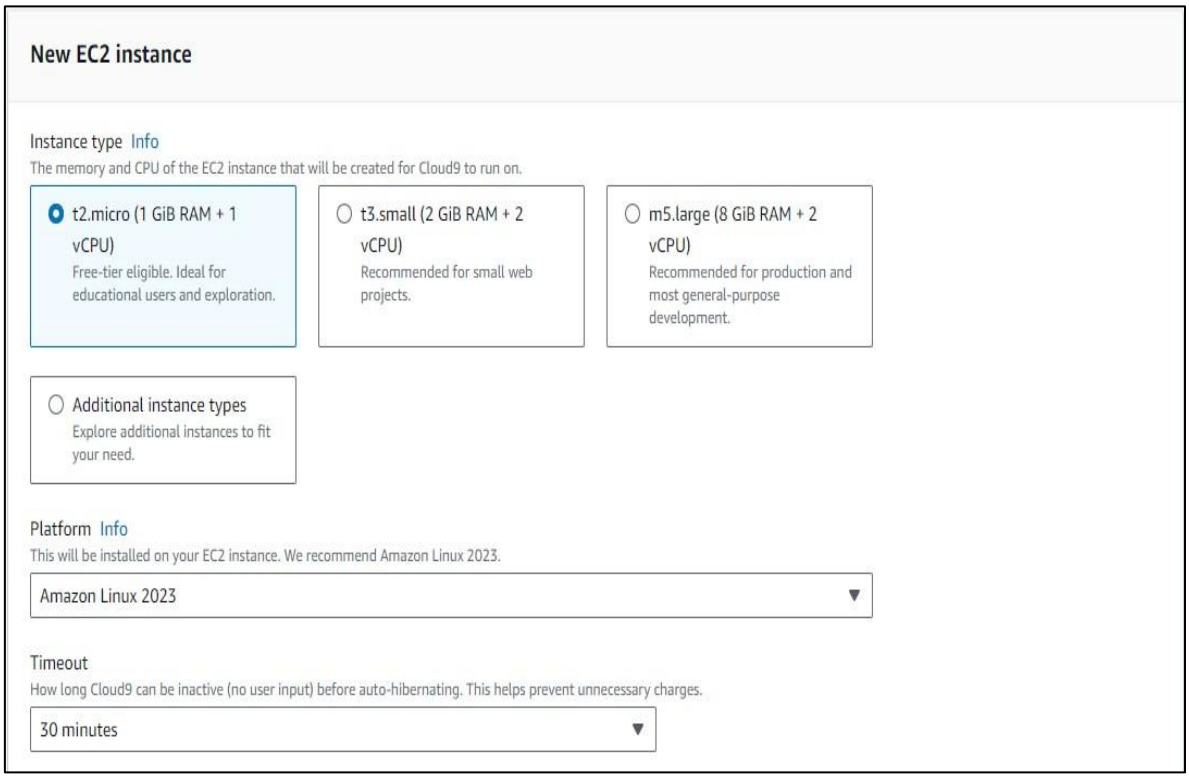

Successfully edited bucket policy.
Amazon S3 > Buckets > test-123-anushka
test-123-anushka Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Permissions overview
Access finding
Access findings are provided by IAM external access analyzers. Learn more about How IAM analyzer findings work
View analyzer for us-east-1
Block public access (bucket settings)
Edit
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Off
Individual Block Public Access settings for this bucket

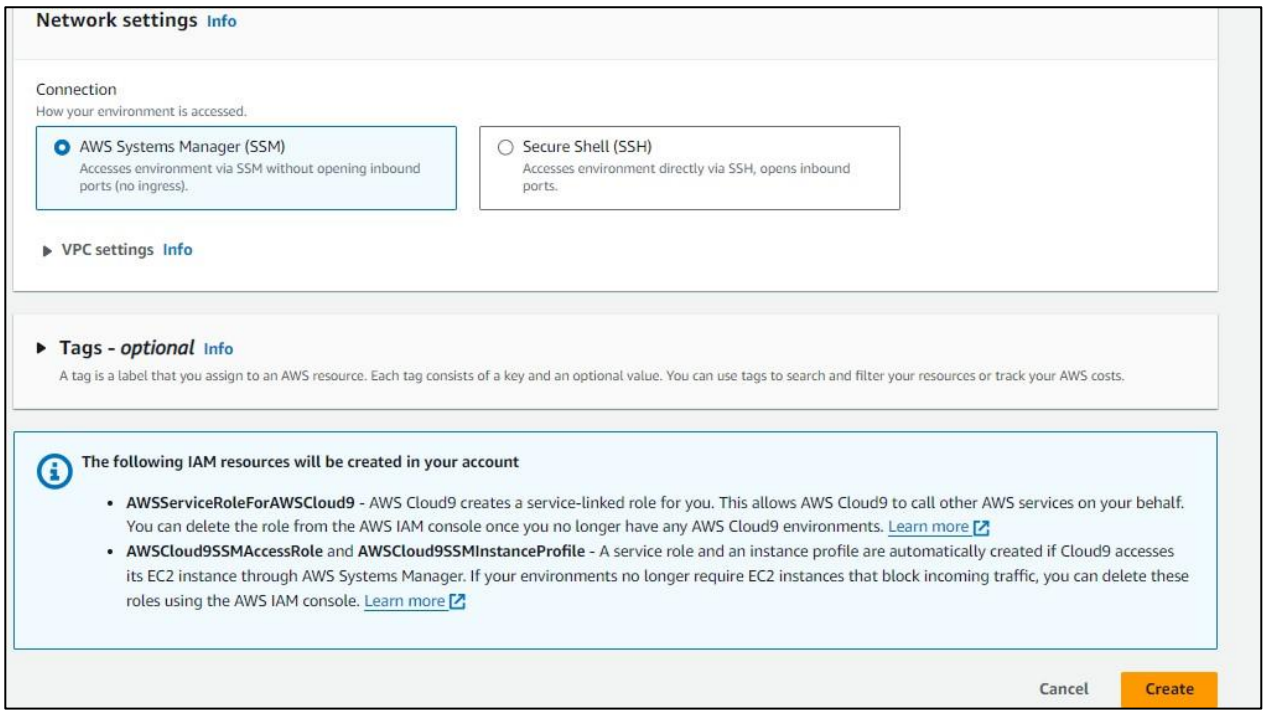

test-123-anushka.s3.amazonaws.com/Test.txt
Hi! This is hosting on AWS through S3 bucket by Anushka!!

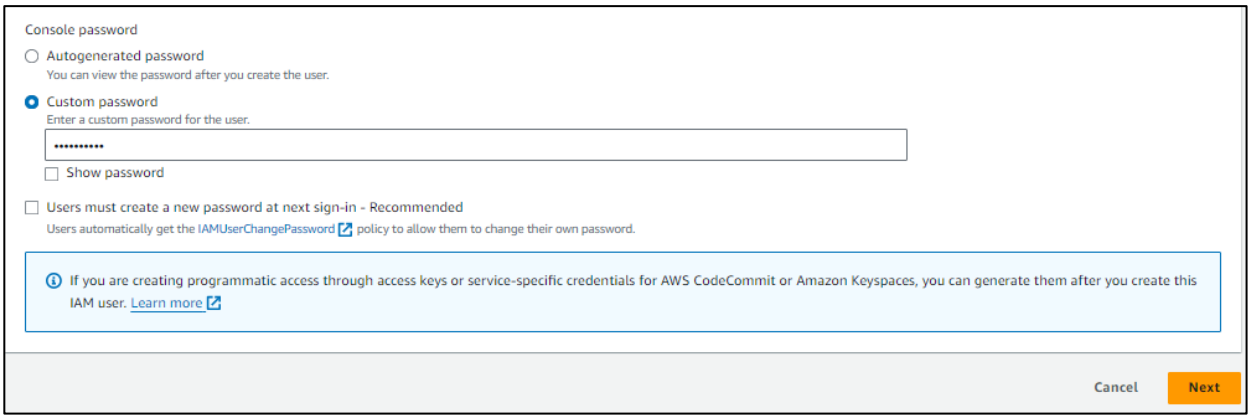
Hosting using Cloud 9 :

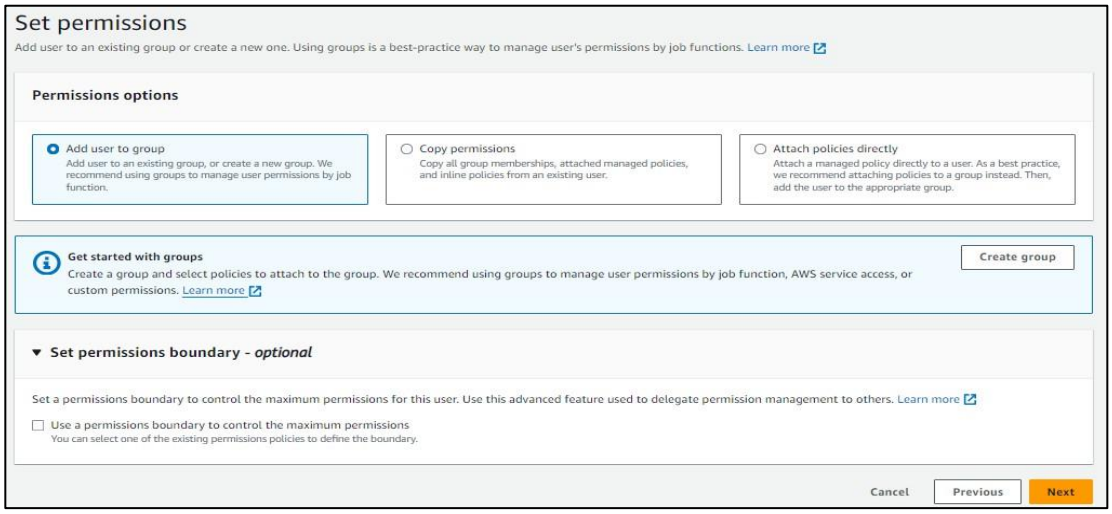




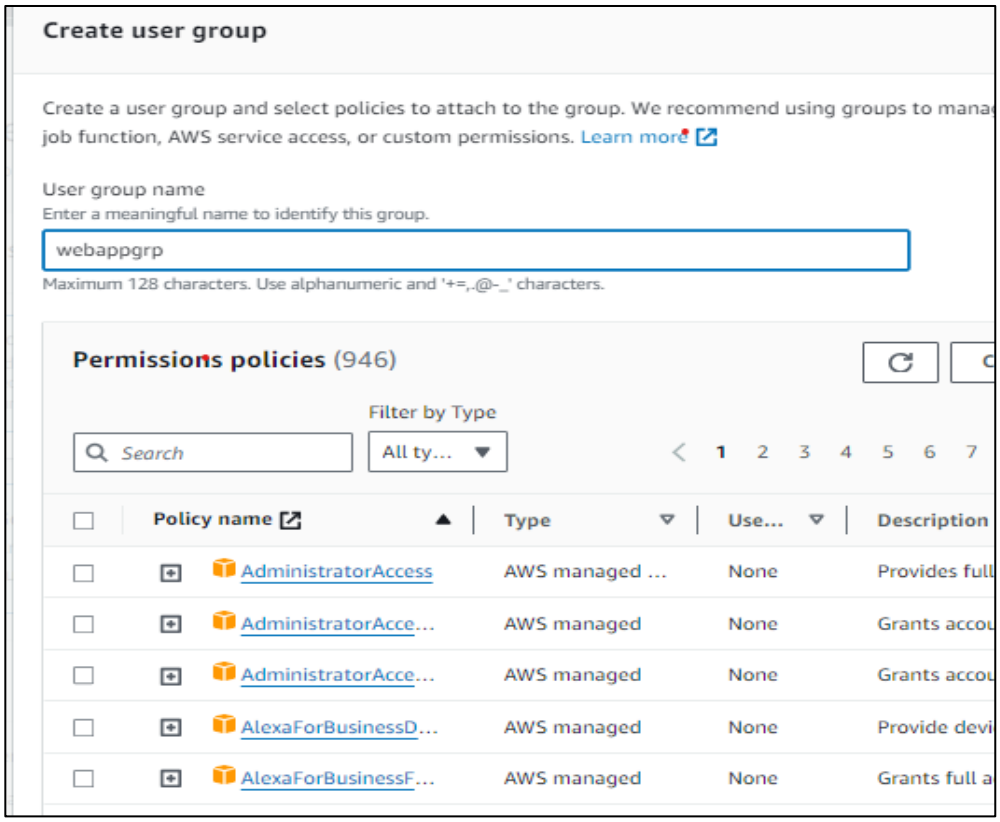

New EC2 instance
Instance type Info
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.
t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.
t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.
m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.
Additional instance types
Explore additional instances to fit your need.
Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2023.
Amazon Linux 2023
Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes

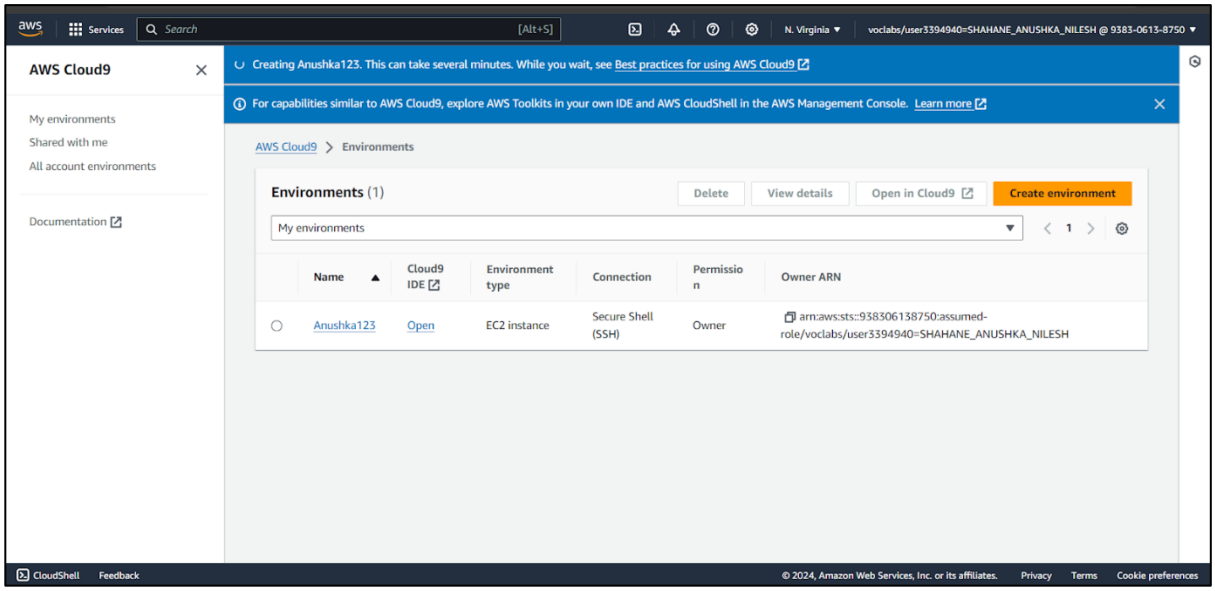

Network settings Info
Connection
How your environment is accessed.
AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.
VPC settings Info
Tags - optional Info
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
The following IAM resources will be created in your account
AWSServiceRoleForAWSCloud9 - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. Learn more
AWSCloud9SSMAccessRole and AWSCloud9SSMInstanceProfile - A service role and an instance profile are automatically created if Cloud9 accesses its EC2 instance through AWS Systems Manager. If your environments no longer require EC2 instances that block incoming traffic, you can delete these roles using the AWS IAM console. Learn more
Cancel
Create


Console password
Autogenerated password
You can view the password after you create the user.
Custom password
Enter a custom password for the user.
Show password
Users must create a new password at next sign-in - Recommended
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.
If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more
Cancel
Next

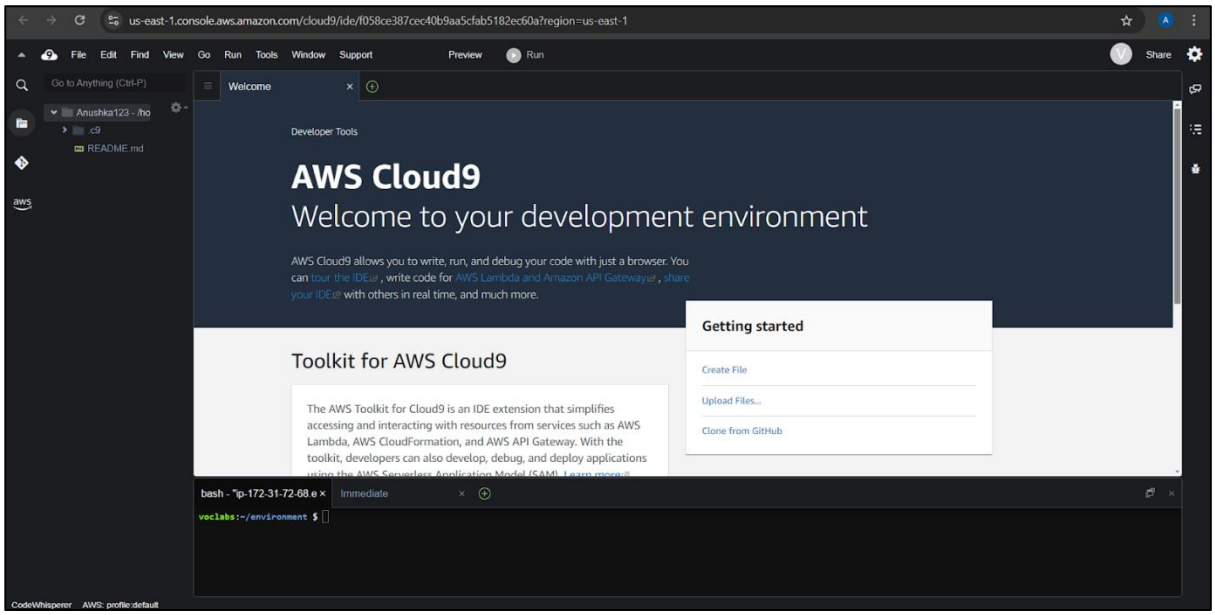

Set permissions
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more
Create group
Set permissions boundary - optional
Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. Learn more
Use a permissions boundary to control the maximum permissions
You can select one of the existing permissions policies to define the boundary.
Cancel
Previous
Next

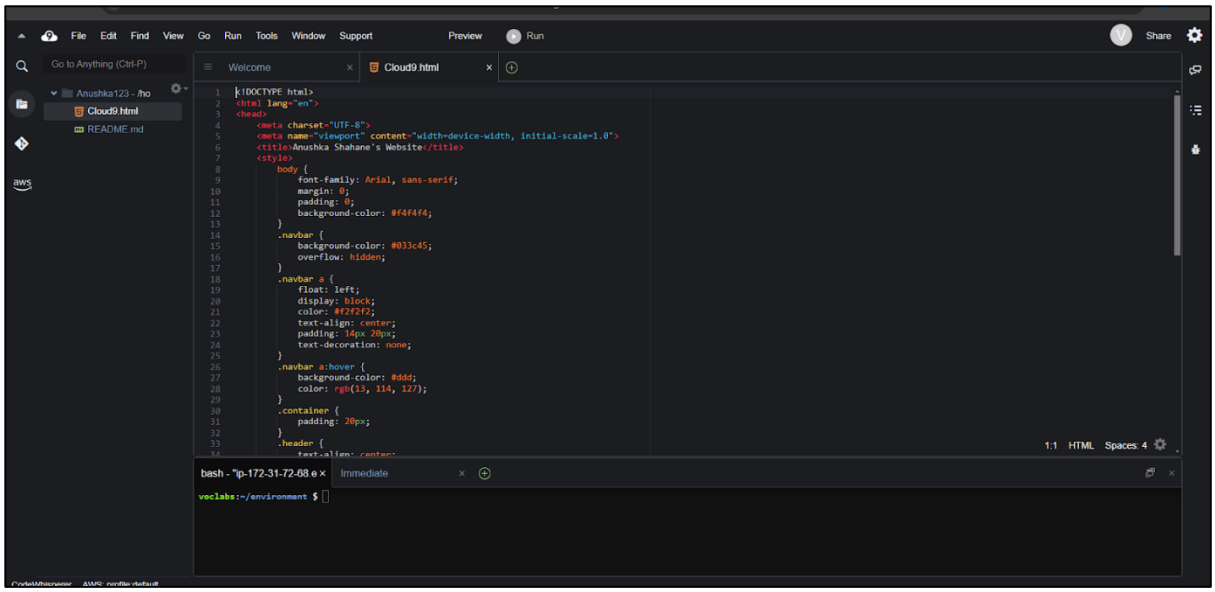
Anushka Shahane D15A 55



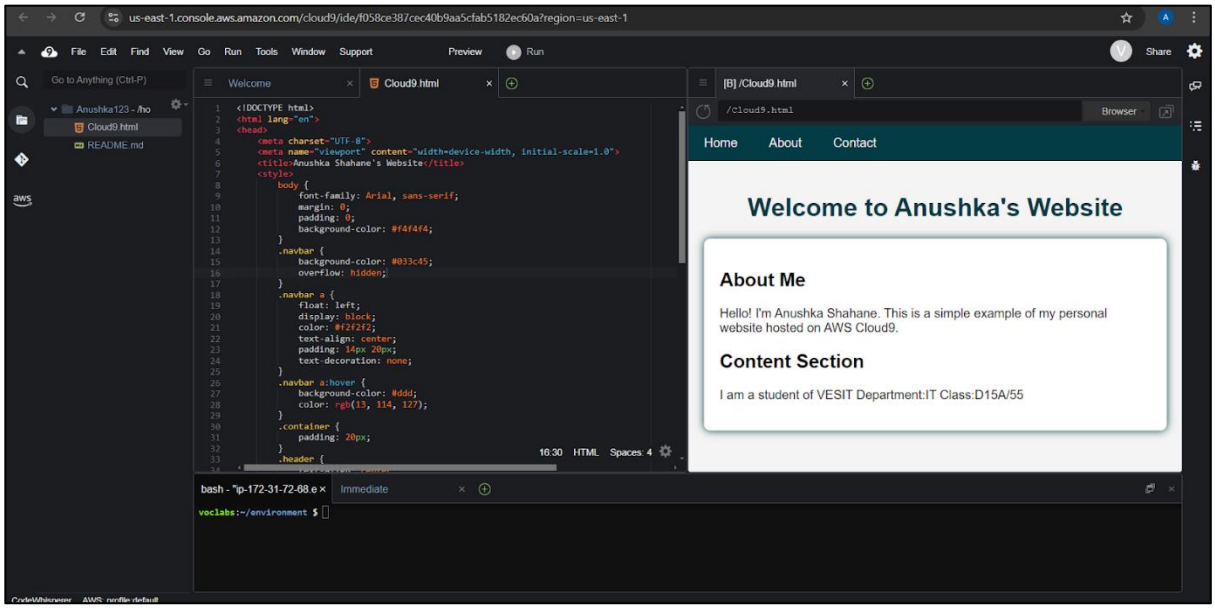


Anushka Shahane D15A 55

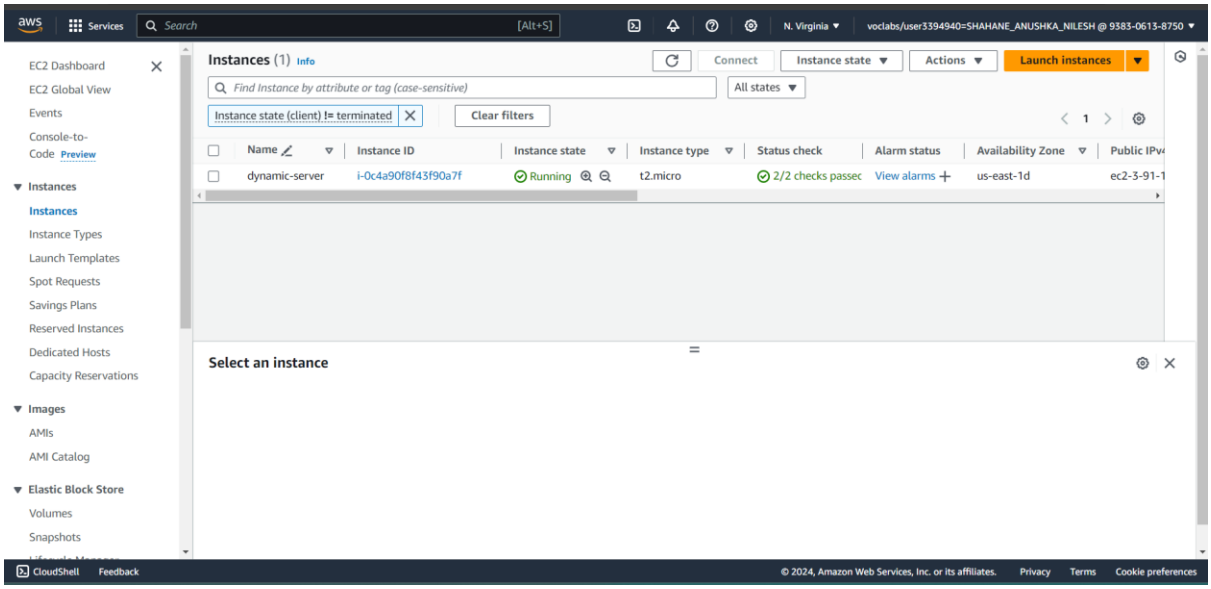




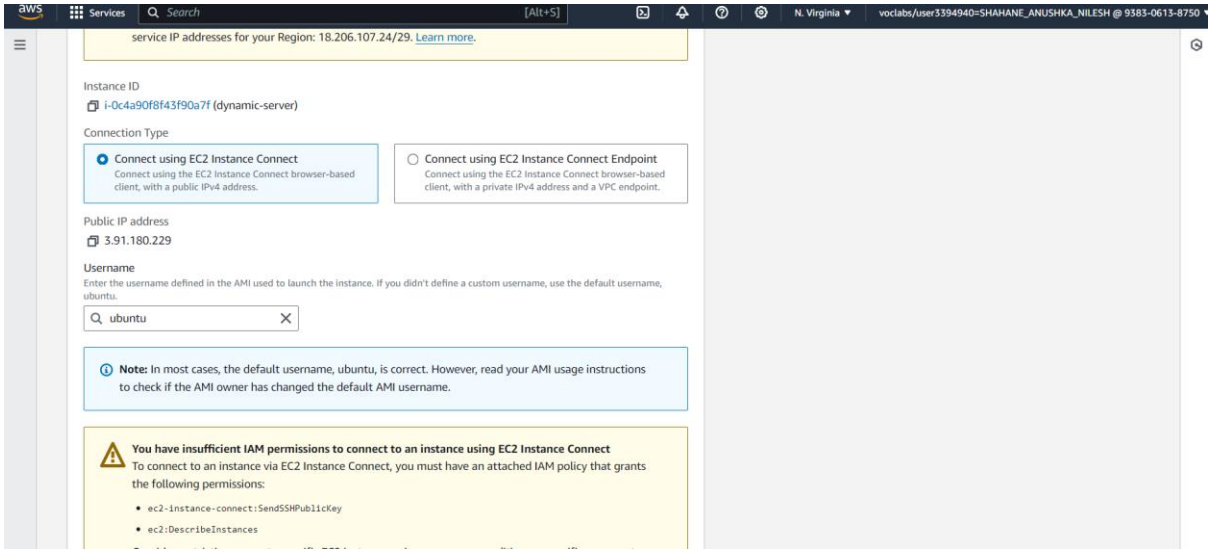
Anushka Shahane D15A 55



Dynamic Hosting using EC2 instance :



Anushka Shahane D15A 55

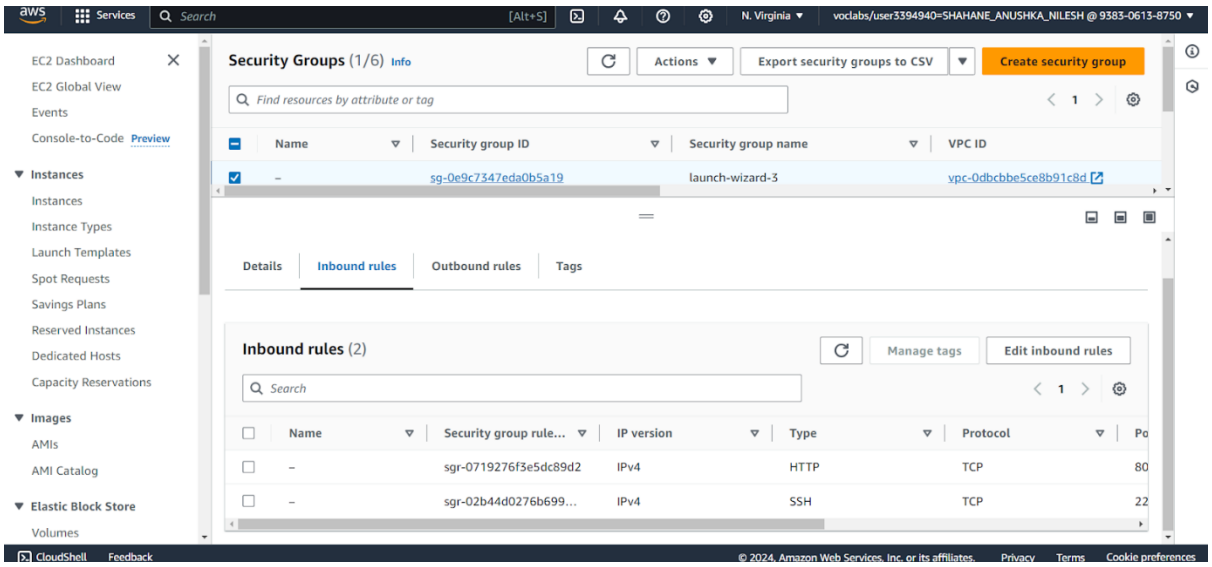


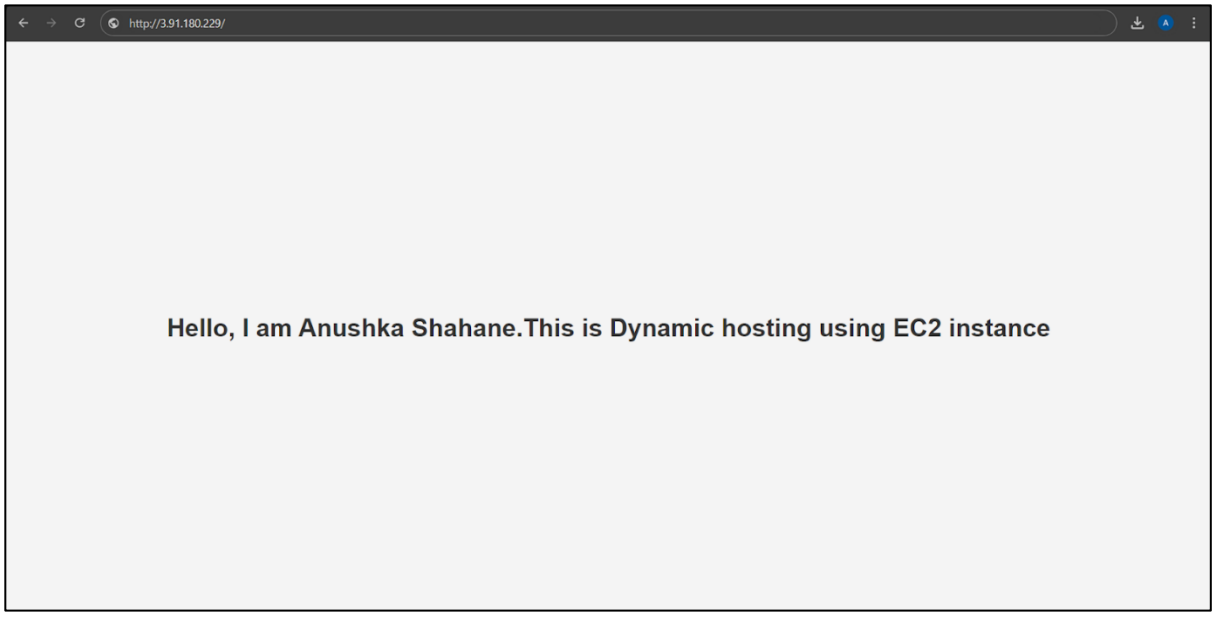

```
ubuntu@ip-172-31-83-228:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-83-228:~$ mkdir Anushka
ubuntu@ip-172-31-83-228:~$ cd Anushka
ubuntu@ip-172-31-83-228:~/Anushka$ git clone https://github.com/Anushka3204/Dynamic_hosting_EC2.git
Cloning into 'Dynamic_hosting_EC2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 8.39 KiB | 1.68 MiB/s, done.
ubuntu@ip-172-31-83-228:~/Anushka$ ls
Dynamic_hosting_EC2
ubuntu@ip-172-31-83-228:~/Anushka$ cd^C
ubuntu@ip-172-31-83-228:~/Anushka$ cd Dynamic_hosting_EC2
ubuntu@ip-172-31-83-228:~/Anushka/Dynamic_hosting_EC2$ ls
index.js  package-lock.json  package.json
```

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-83-228:~/Anushka/Dynamic_hosting_EC2/Dynamic_hosting_EC2$ npm i

added 64 packages, and audited 65 packages in 3s

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

found 0 vulnerabilities
ubuntu@ip-172-31-83-228:~/Anushka/Dynamic_hosting_EC2/Dynamic_hosting_EC2$
```

```
Dynamic_hosting_EC2  Dynamic_hosting_ec2  Hosting_dynamic_ec2  package-lock.json
ubuntu@ip-172-31-83-228:~/Anushka$ cd  Hosting_dynamic_ec2
ubuntu@ip-172-31-83-228:~/Anushka/Hosting_dynamic_ec2$ ls
index.js  package-lock.json  package.json
ubuntu@ip-172-31-83-228:~/Anushka/Hosting_dynamic_ec2$ npm i

added 64 packages, and audited 65 packages in 1s

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

found 0 vulnerabilities
ubuntu@ip-172-31-83-228:~/Anushka/Hosting_dynamic_ec2$ npm start

> server@1.0.0 start
> node index.js

Server is running on port 3000
```

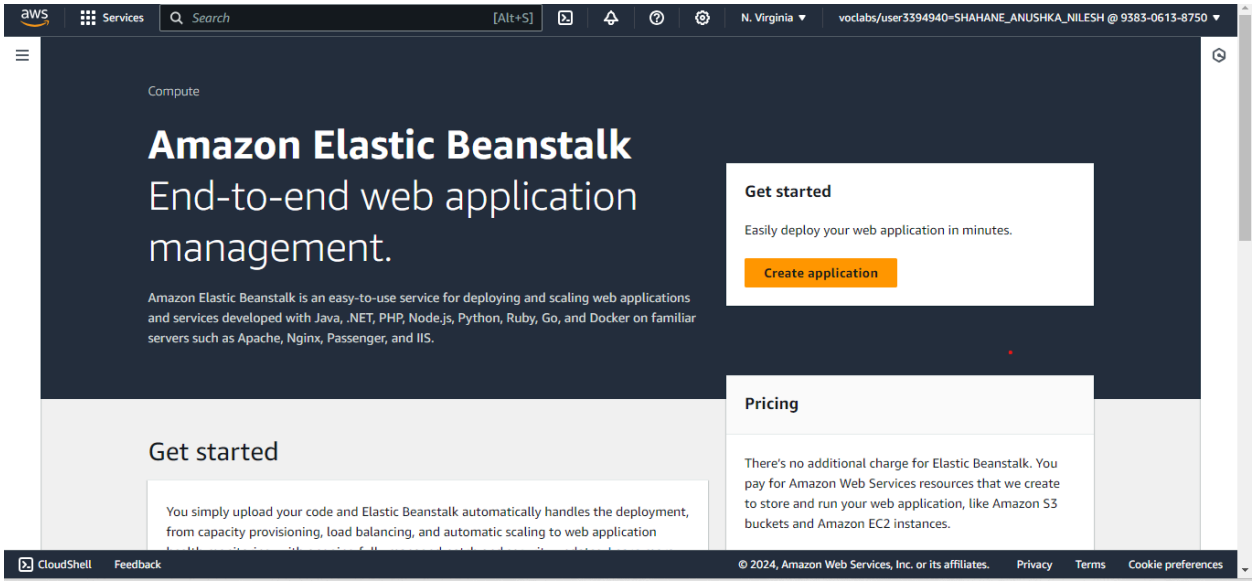
Anushka Shahane D15A 55

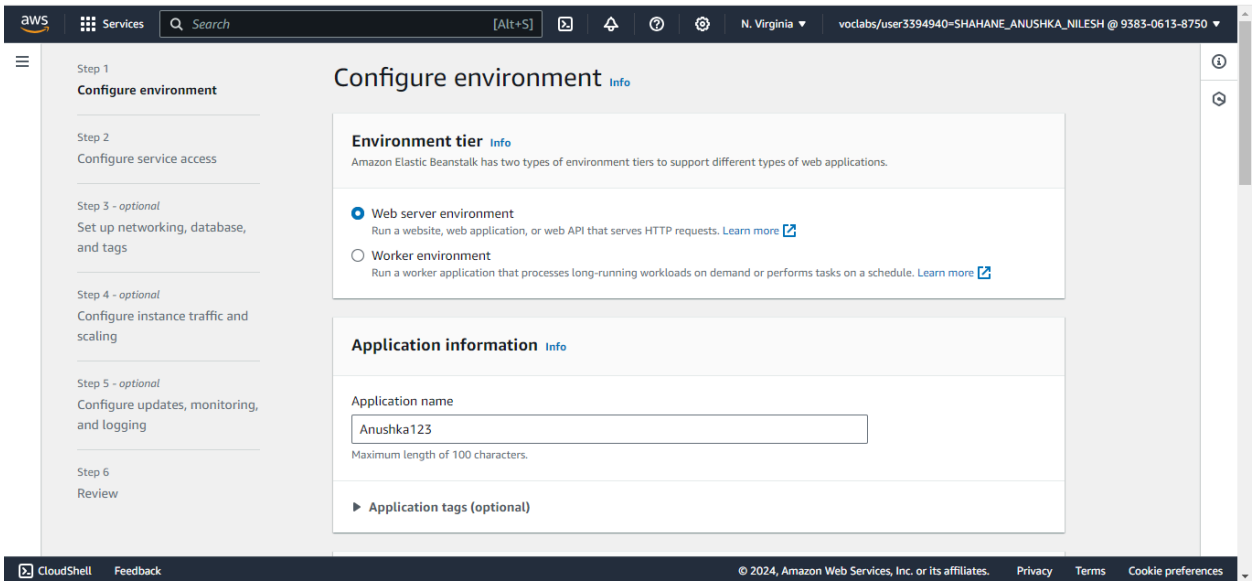




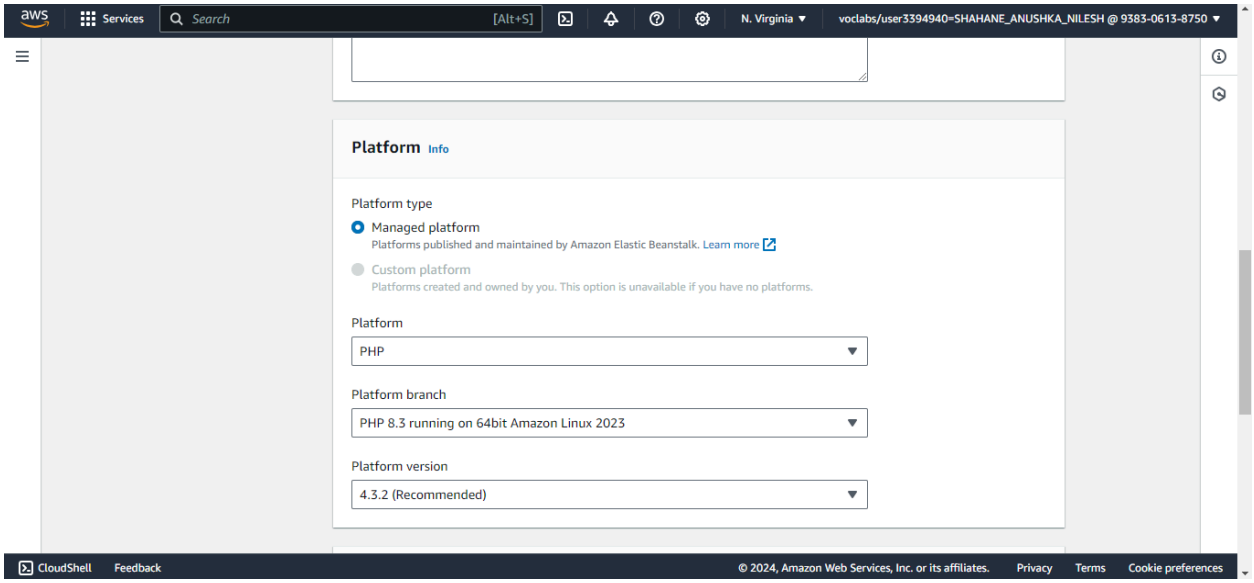
ADVANCE DEVOPS EXPERIMENT NO. 2

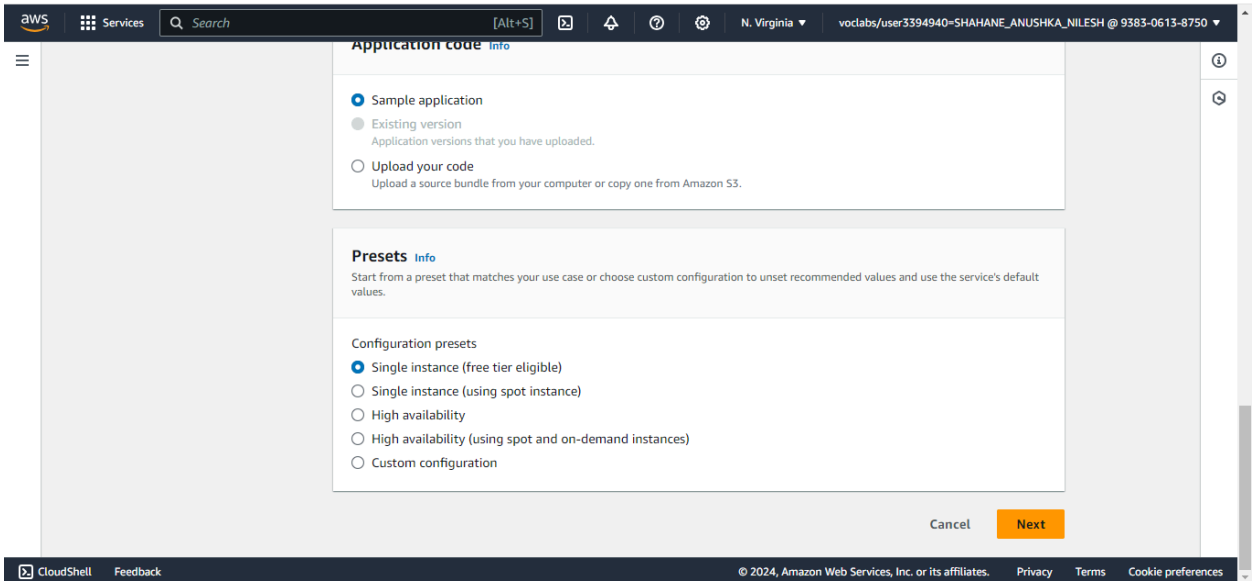
Aim : To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.



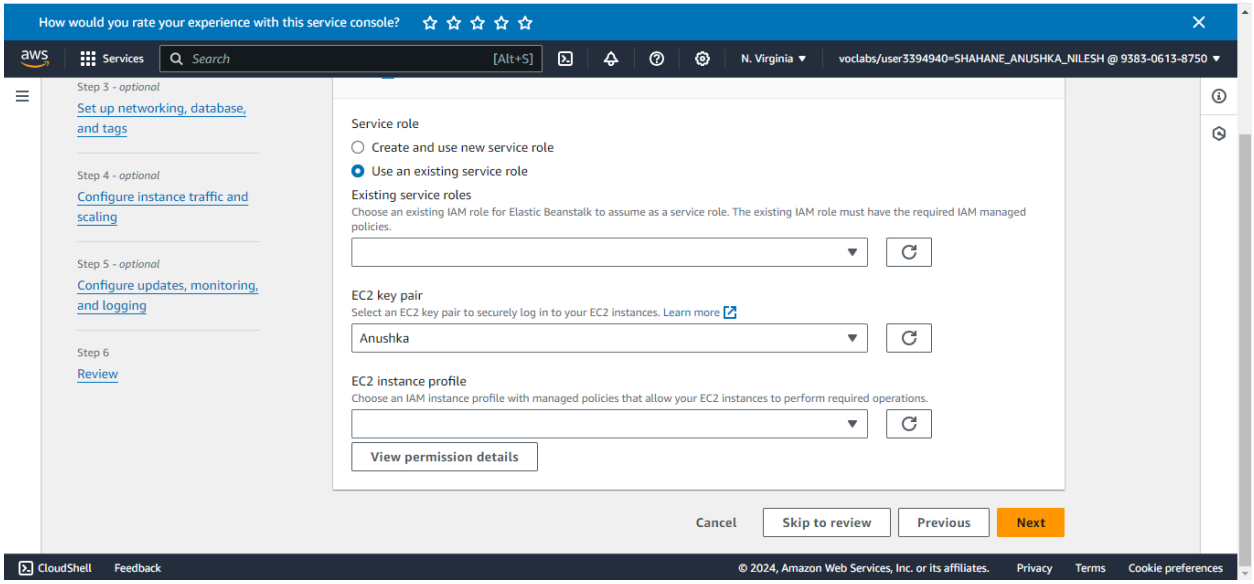


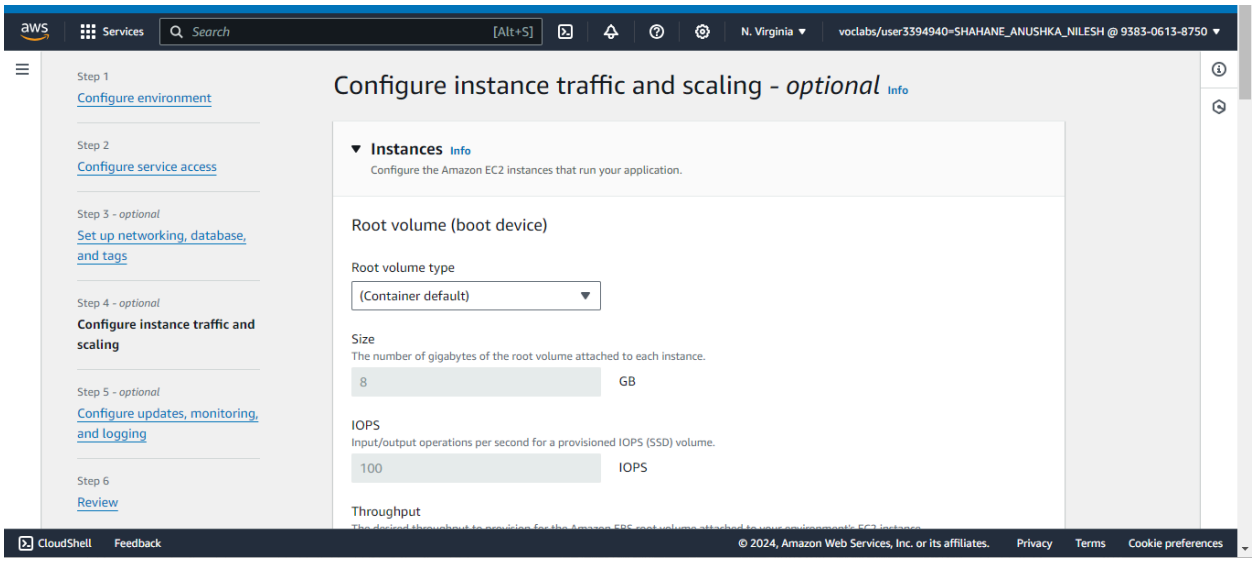
Anushka Shahane D15A 55



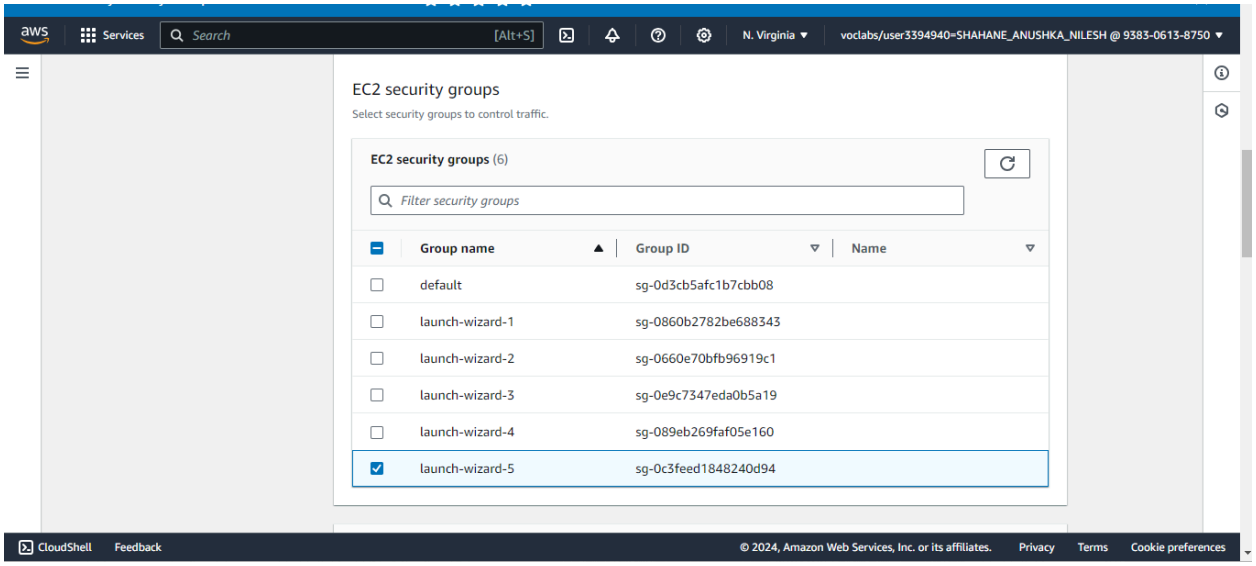


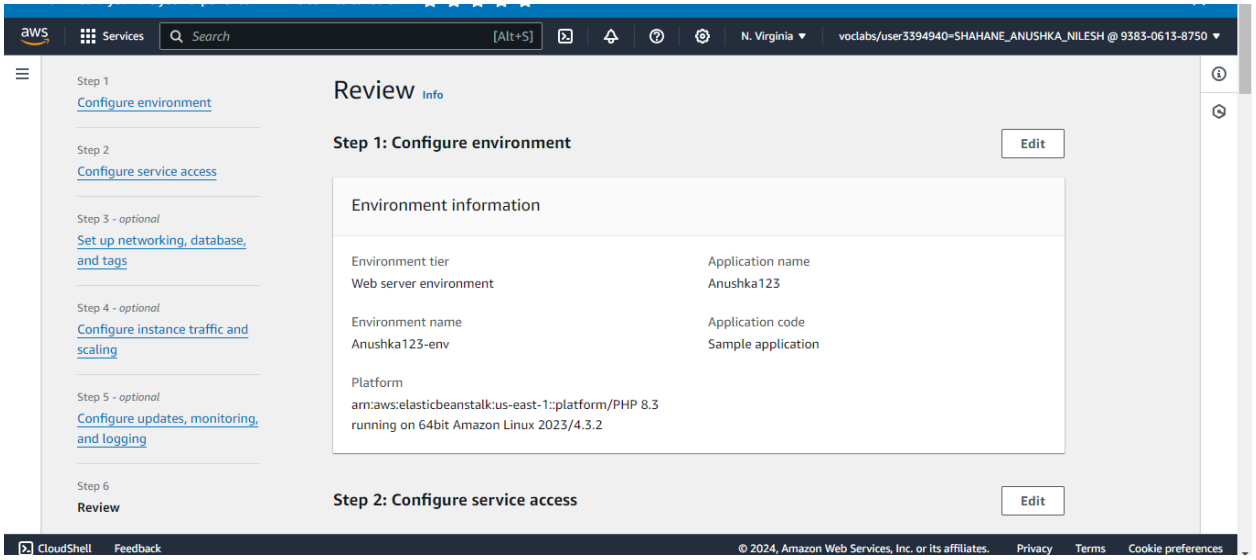
Anushka Shahane D15A 55



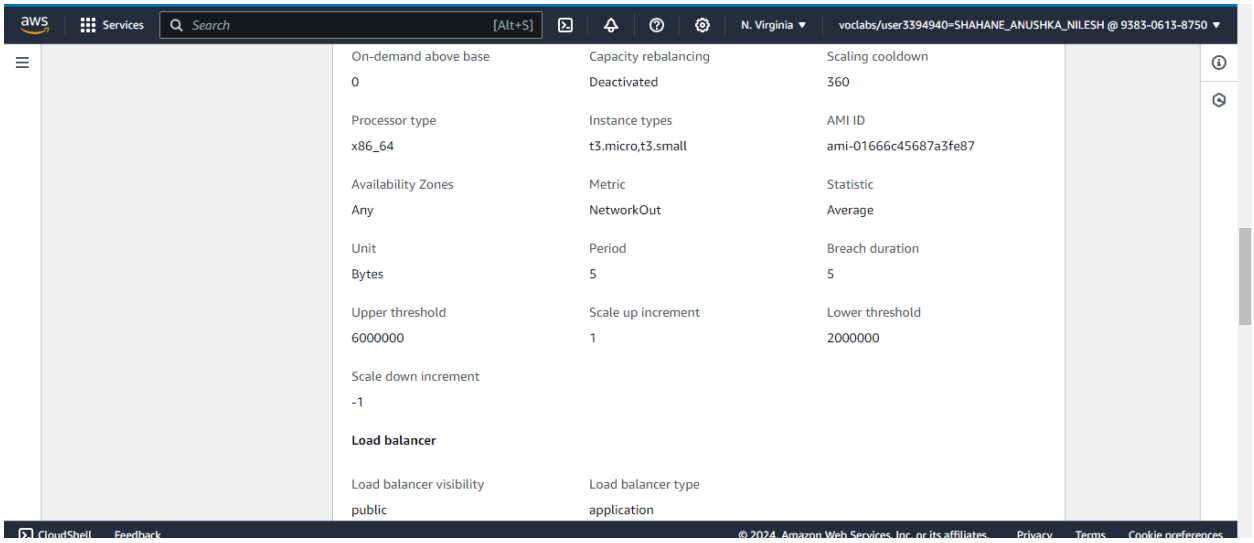


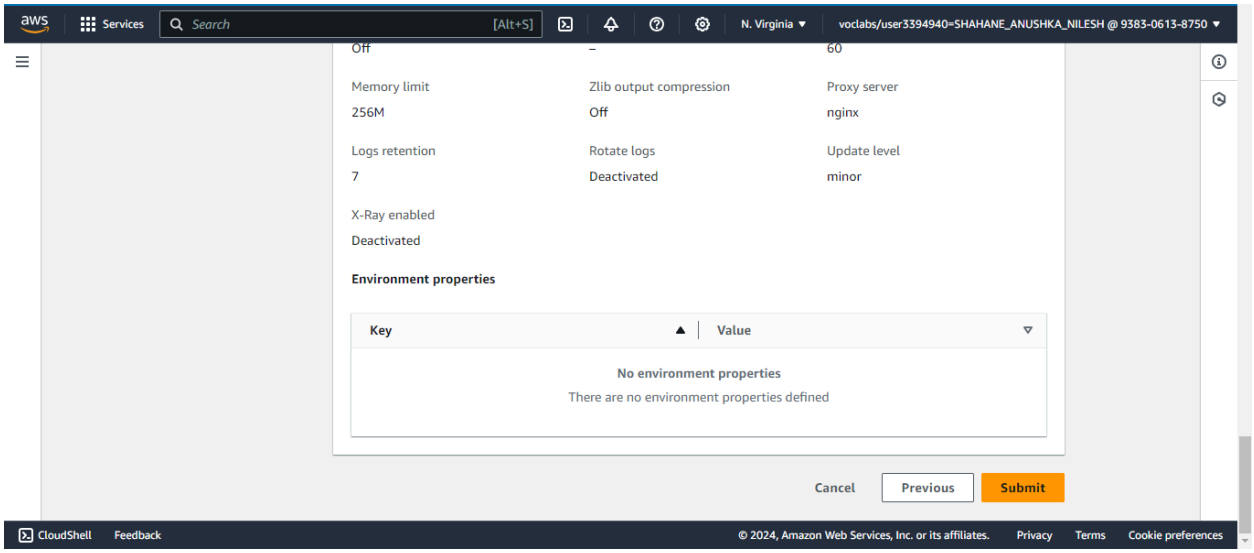
Anushka Shahane D15A 55



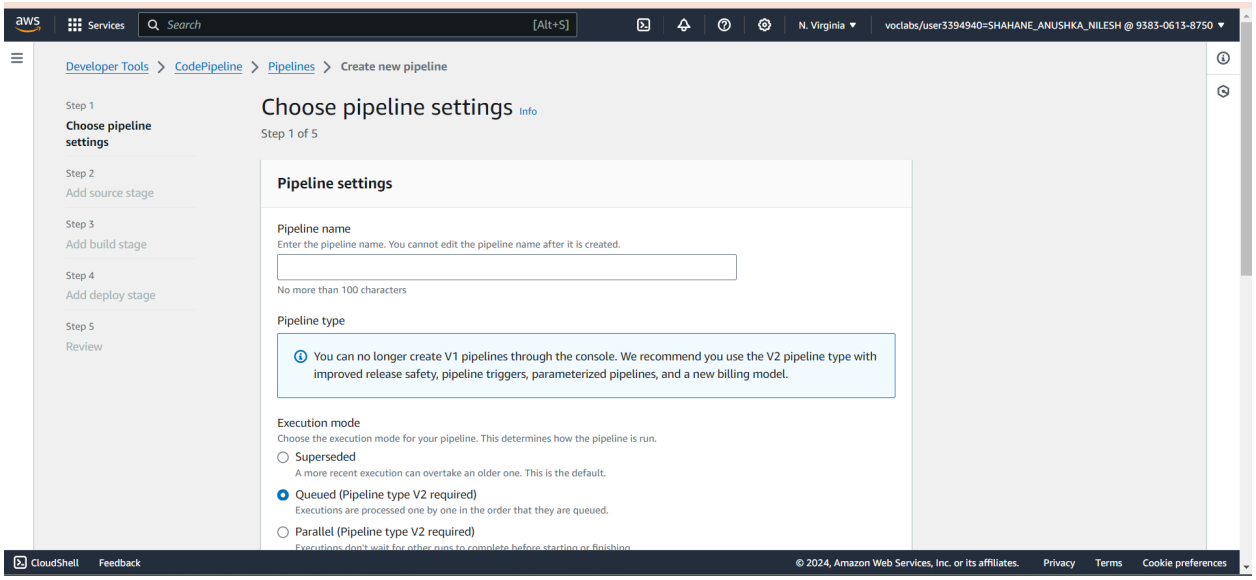


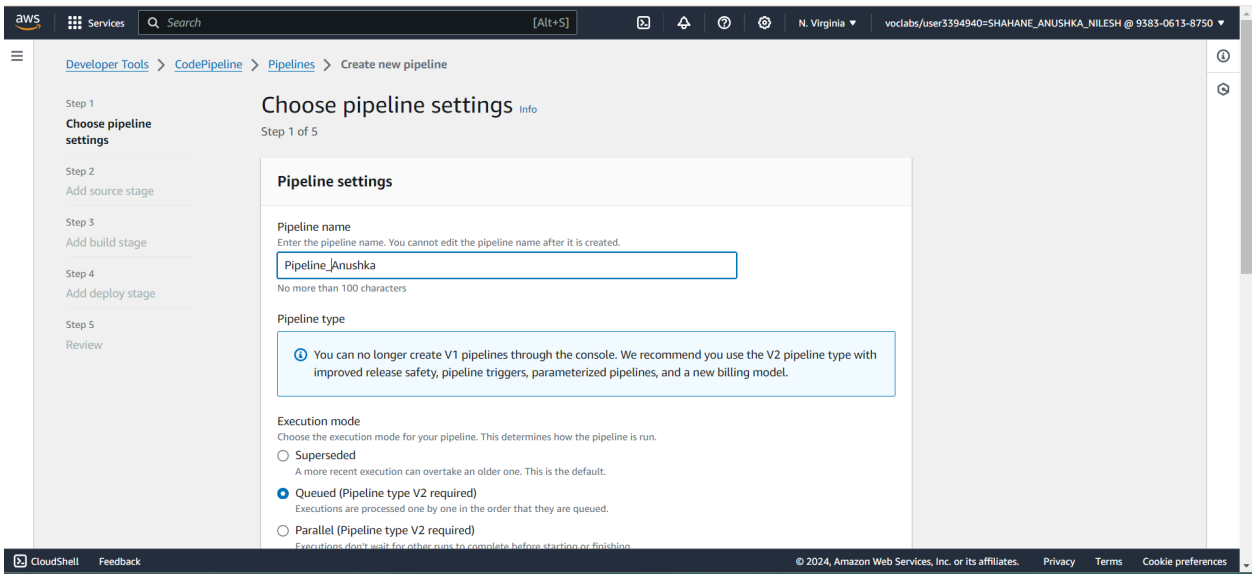
Anushka Shahane D15A 55



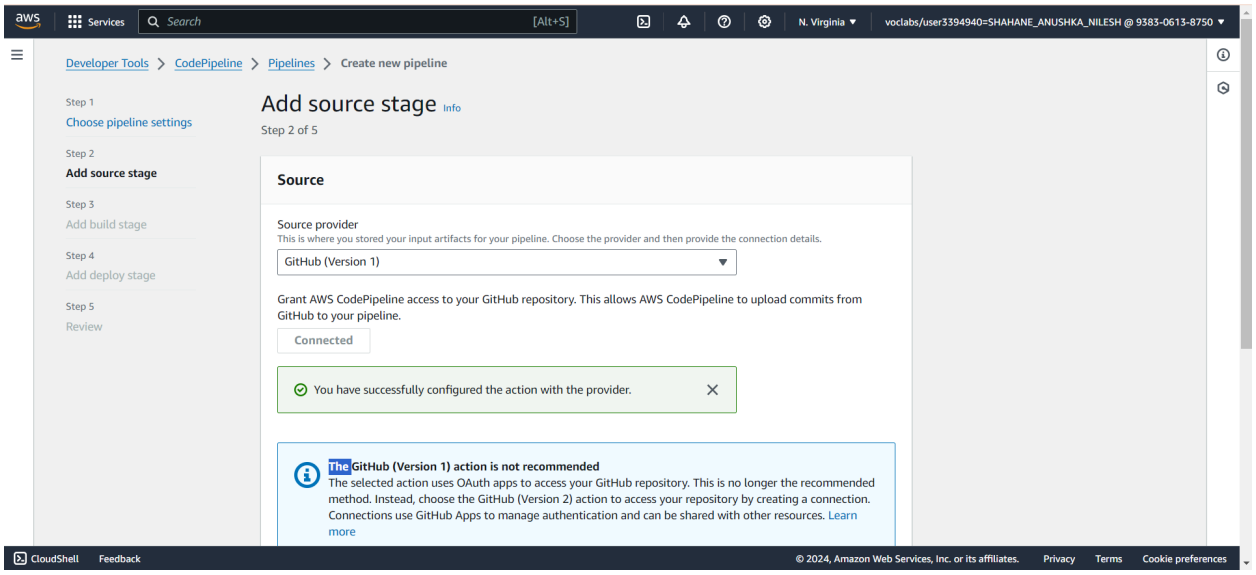


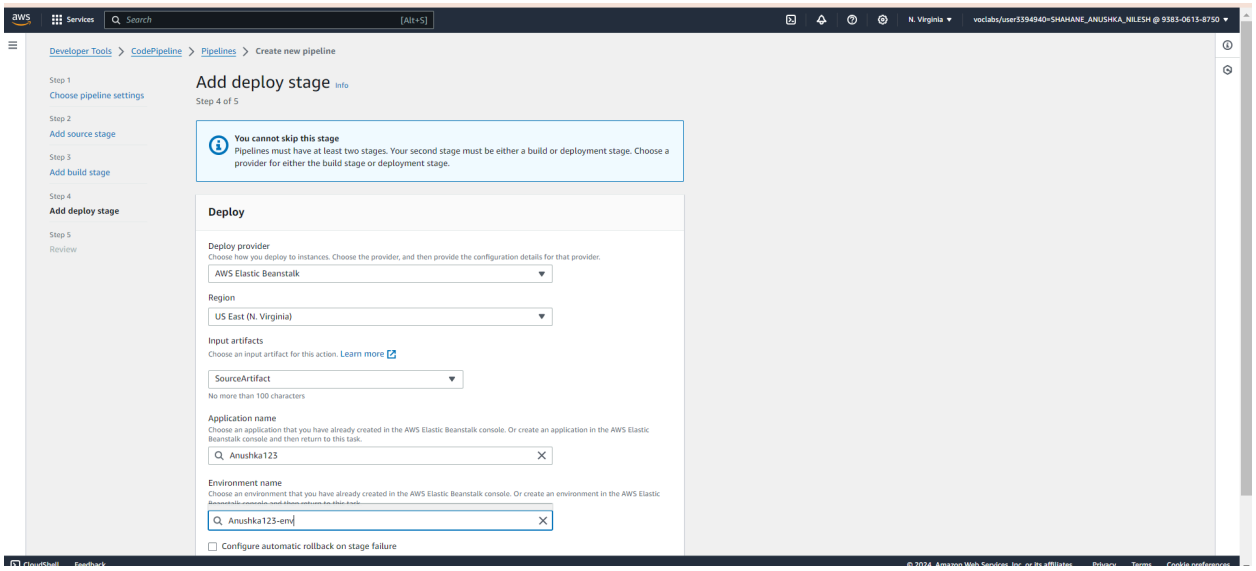
Anushka Shahane D15A 55



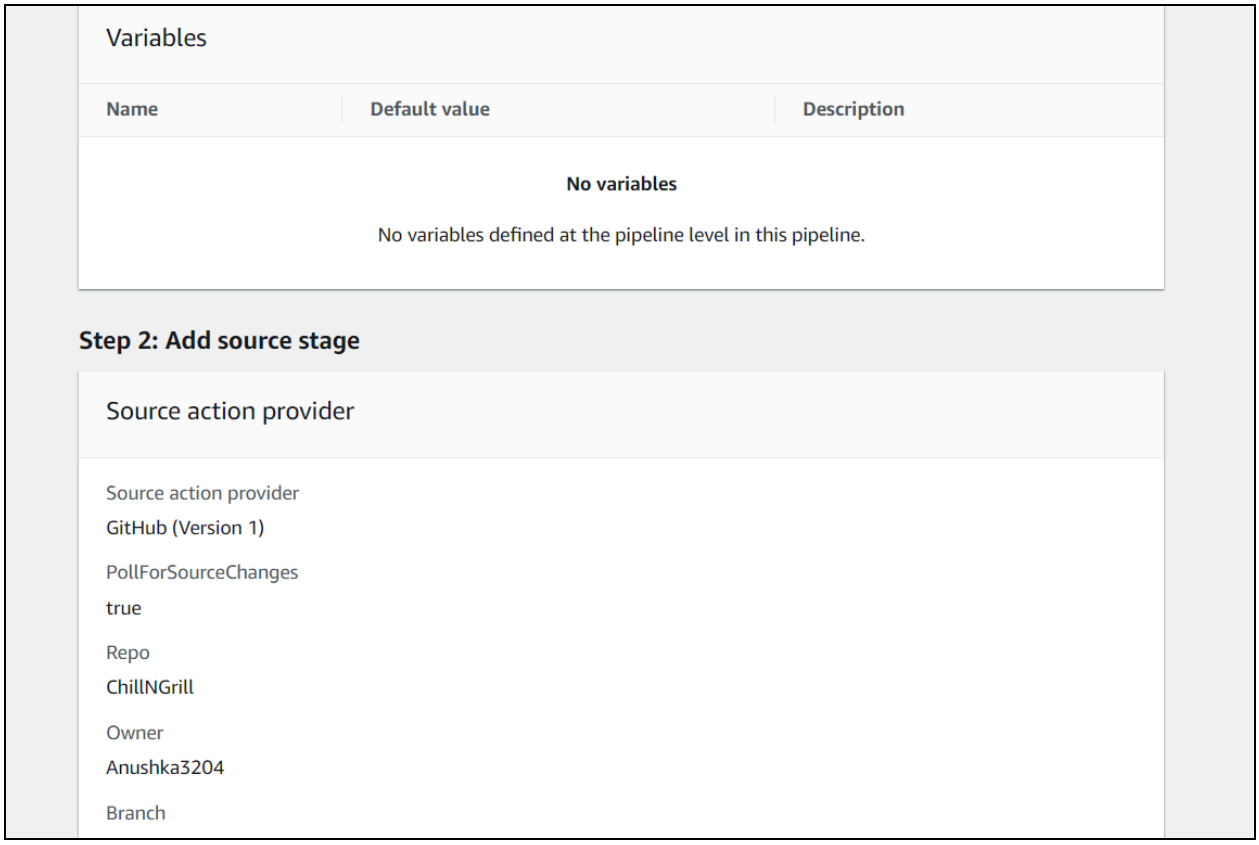


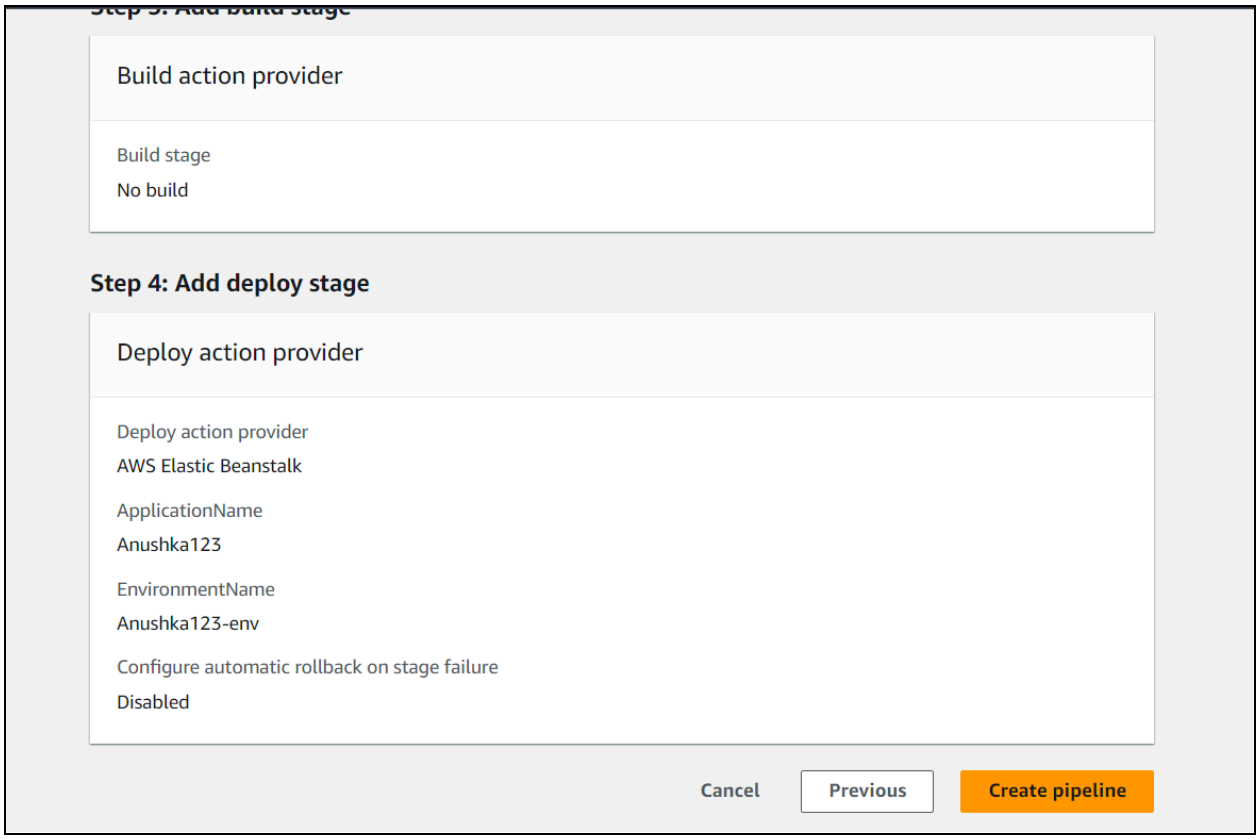
Anushka Shahane D15A 55

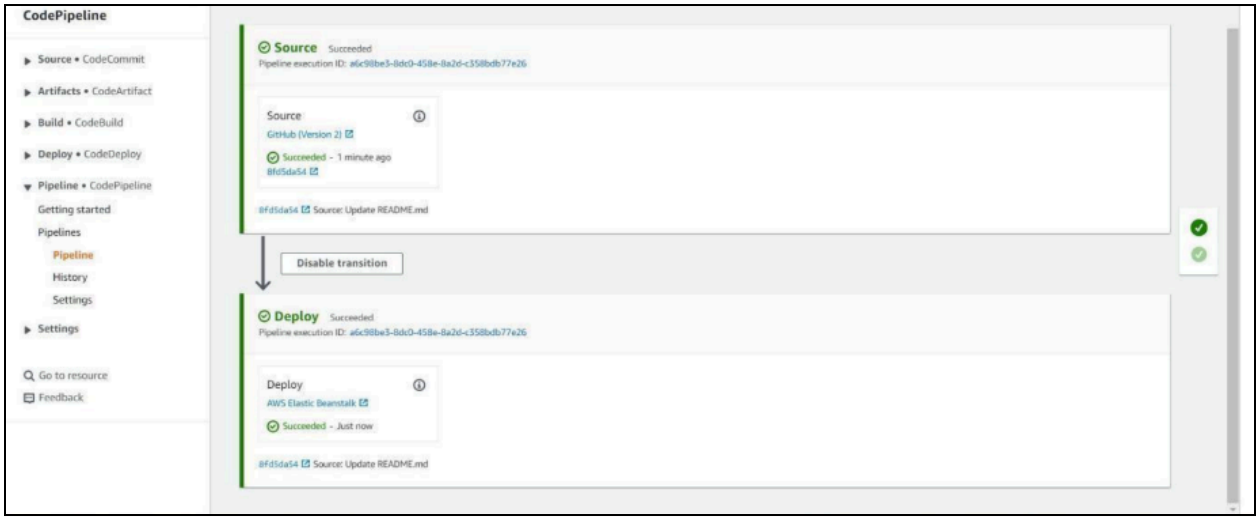




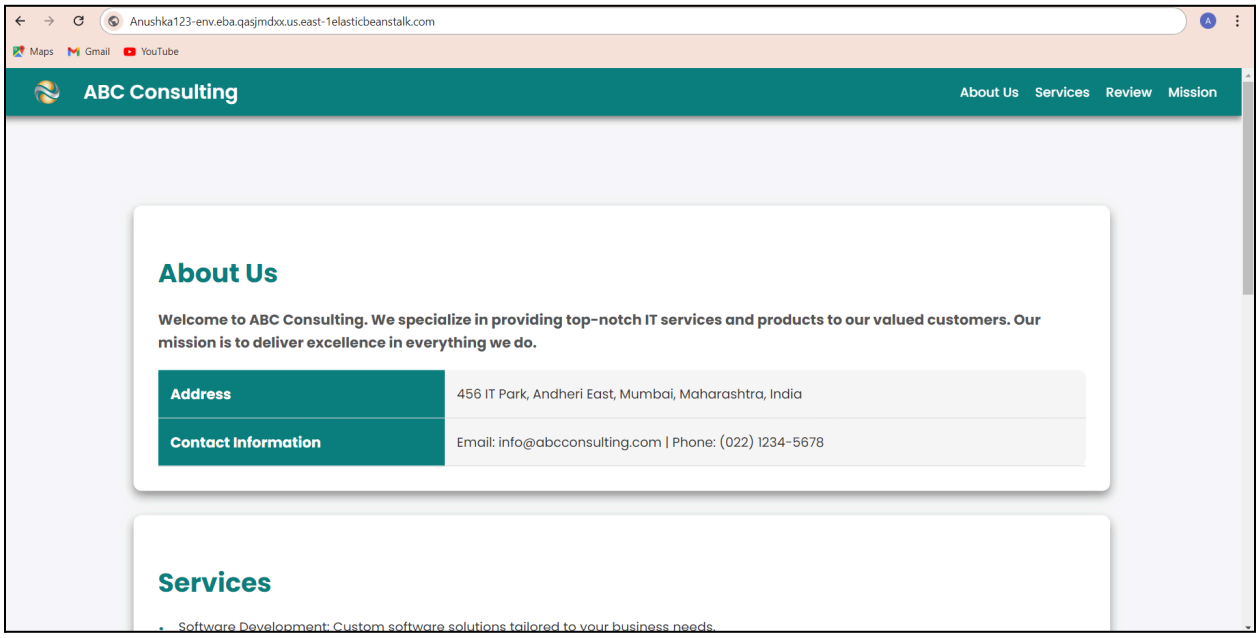
=


Variables
Name
Default value
Description
No variables
No variables defined at the pipeline level in this pipeline.
Step 2: Add source stage
Source action provider
Source action provider
GitHub (Version 1)
PollForSourceChanges
true
Repo
ChillNGrill
Owner
Anushka3204
Branch

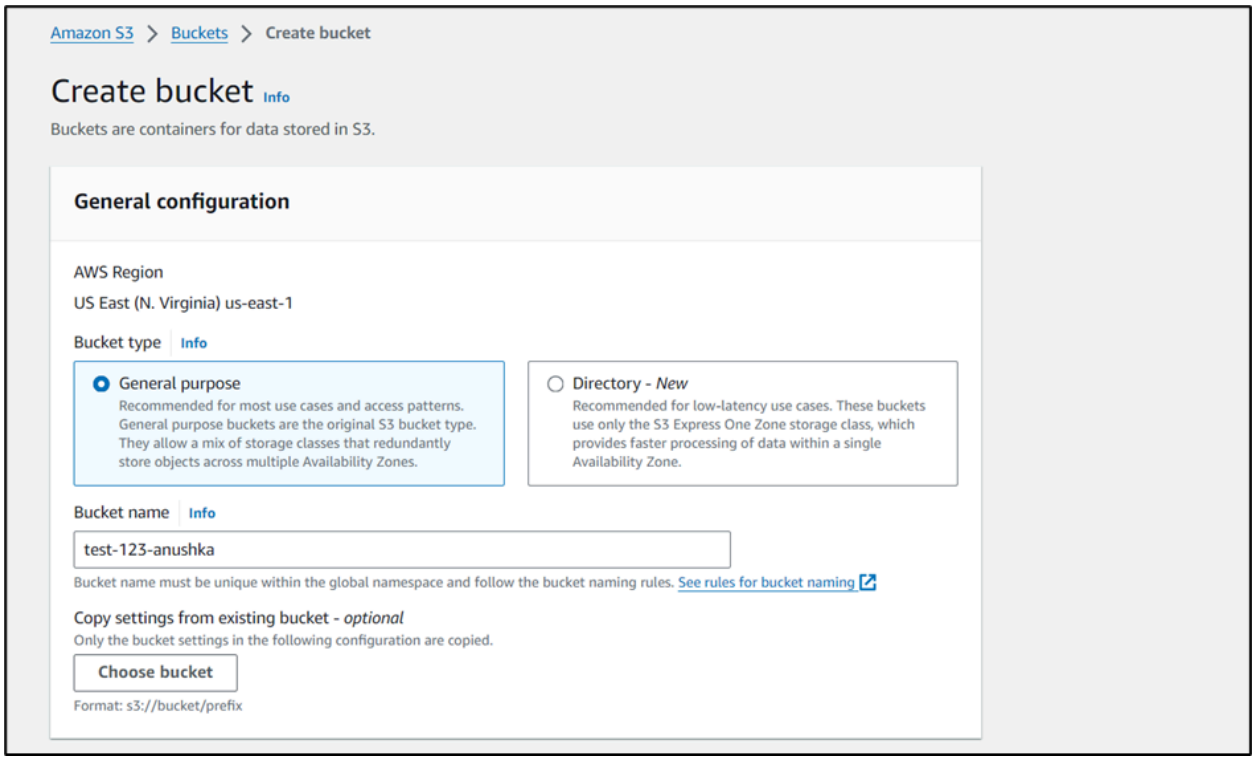

Build action provider
Build stage
No build
Step 4: Add deploy stage
Deploy action provider
Deploy action provider
AWS Elastic Beanstalk
ApplicationName
Anushka123
EnvironmentName
Anushka123-env
Configure automatic rollback on stage failure
Disabled
Cancel
Previous
Create pipeline

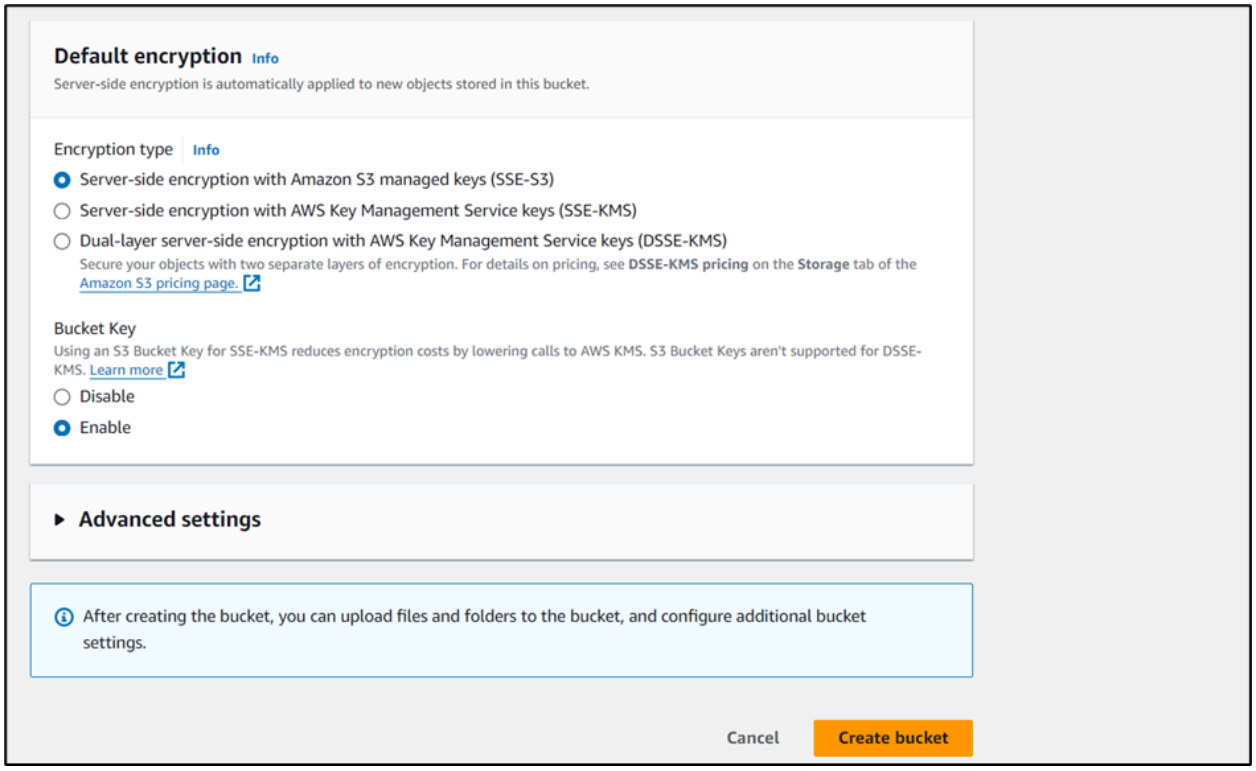

CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Source Succeeded
Source
GitHub (Version 2)
Succeeded - 1 minute ago
Disable transition
Deploy Succeeded
Deploy
AWS Elastic Beanstalk
Succeeded - Just now

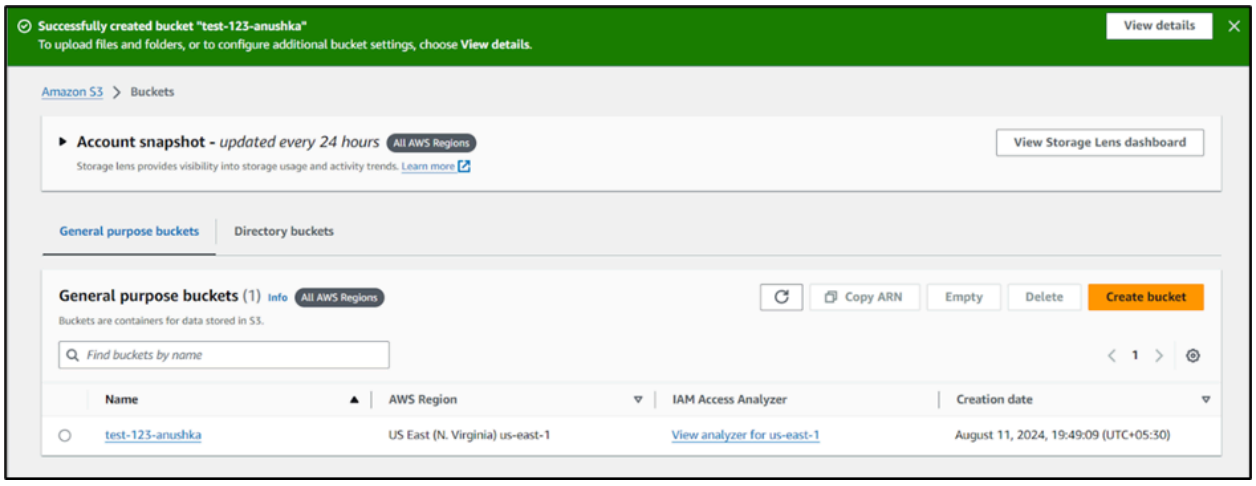
Anushka Shahane D15A 55



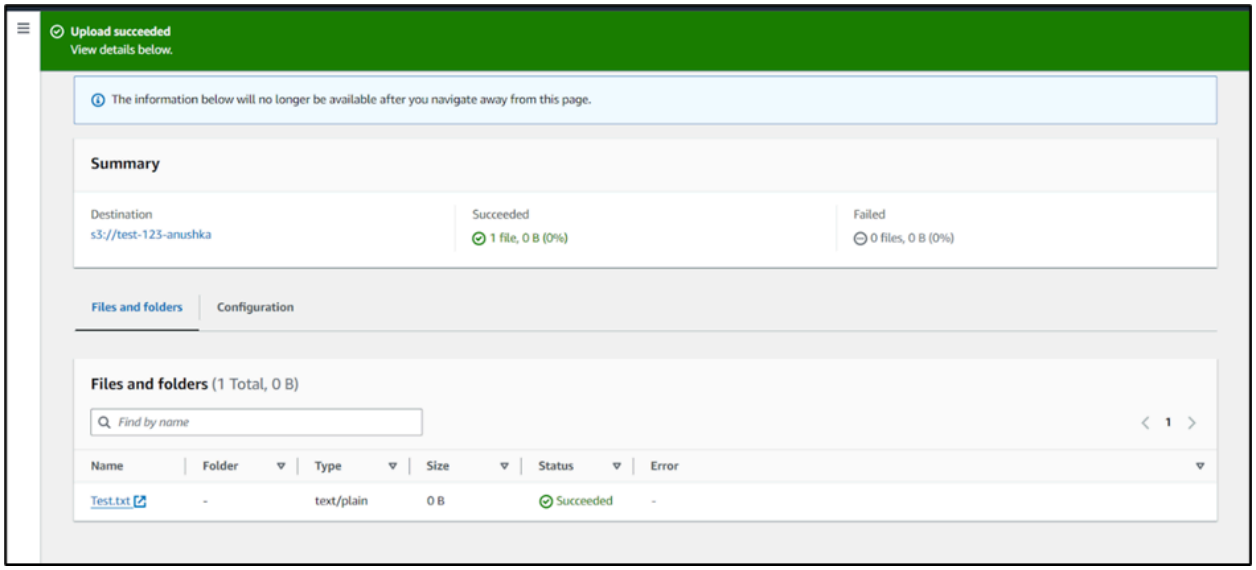
Using S3 bucket :

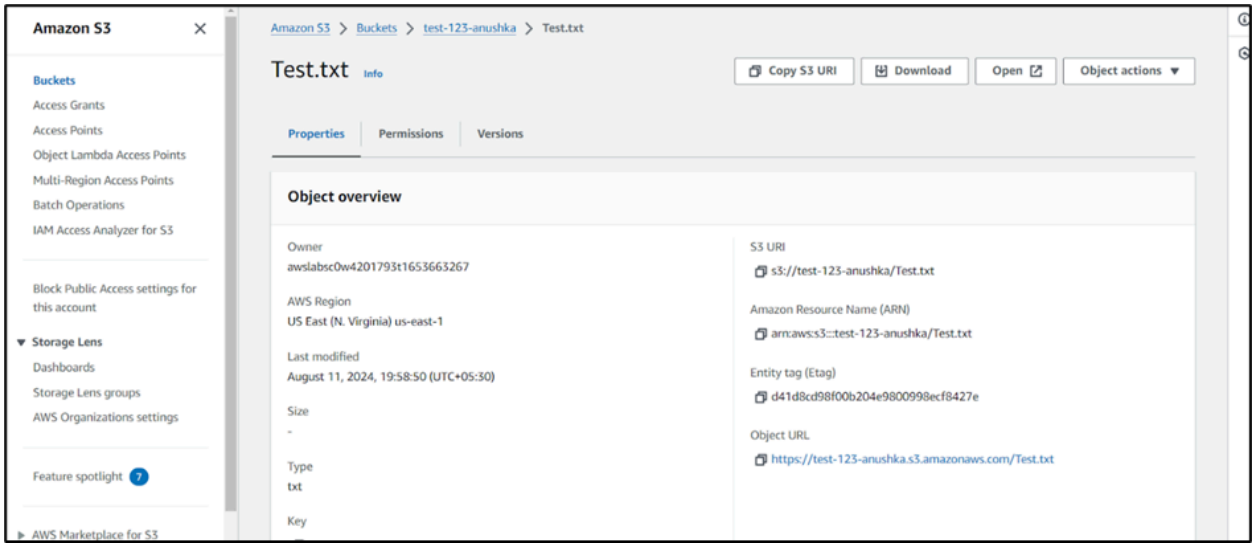



Default encryption Info
Server-side encryption is automatically applied to new objects stored in this bucket.
Encryption type Info
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the Amazon S3 pricing page.
Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more
Disable
Enable
Advanced settings
After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.
Cancel
Create bucket

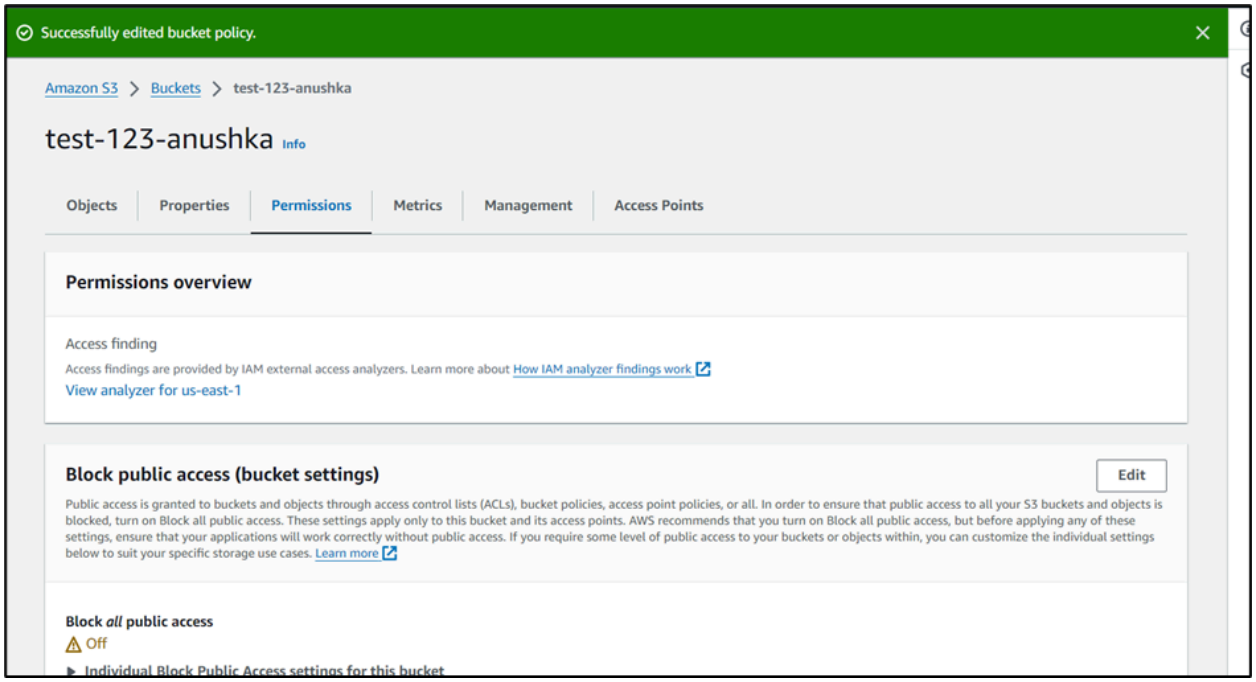

Successfully created bucket "test-123-anushka"
To upload files and folders, or to configure additional bucket settings, choose View details.
View details
Amazon S3 > Buckets
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (1) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
test-123-anushka
US East (N. Virginia) us-east-1
View analyzer for us-east-1
August 11, 2024, 19:49:09 (UTC+05:30)

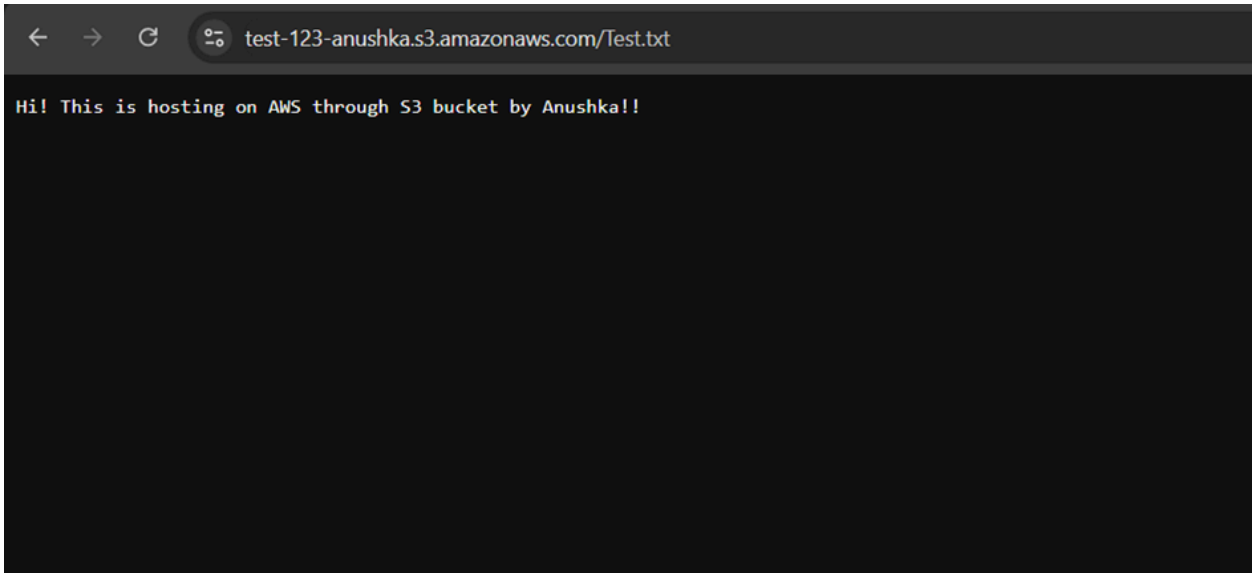
Anushka Shahane D15A 55





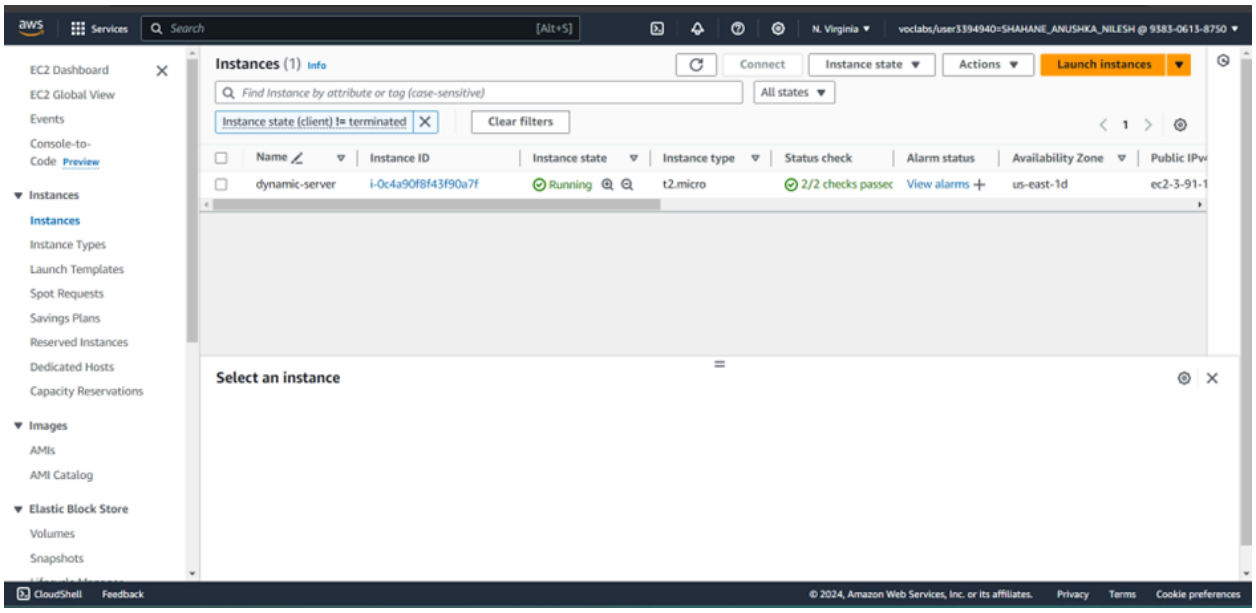
Anushka Shahane D15A 55

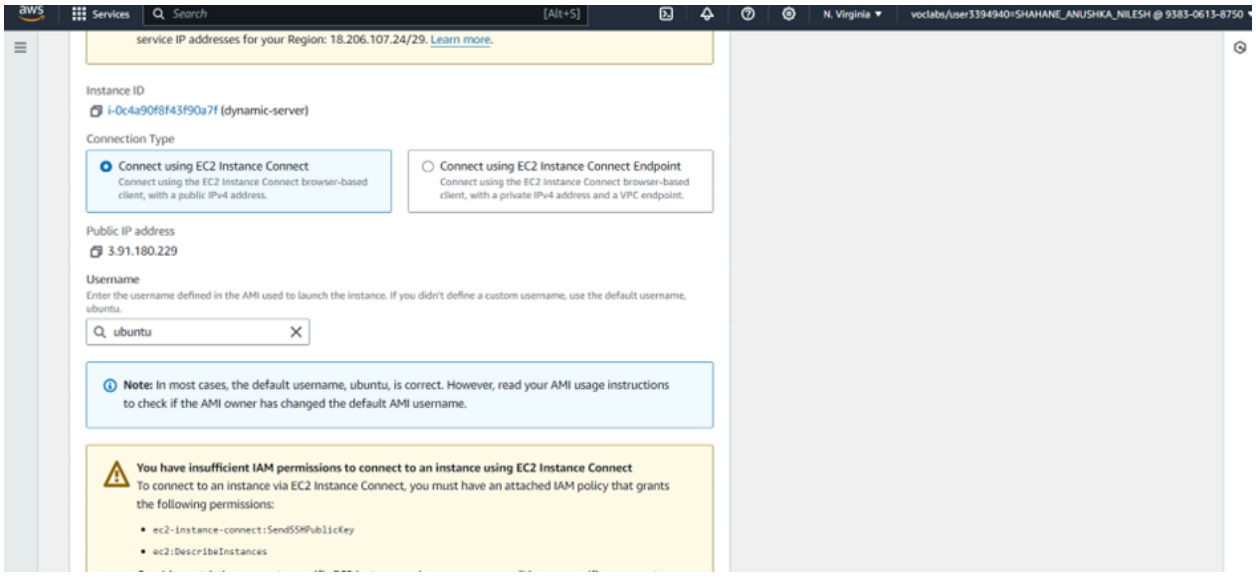




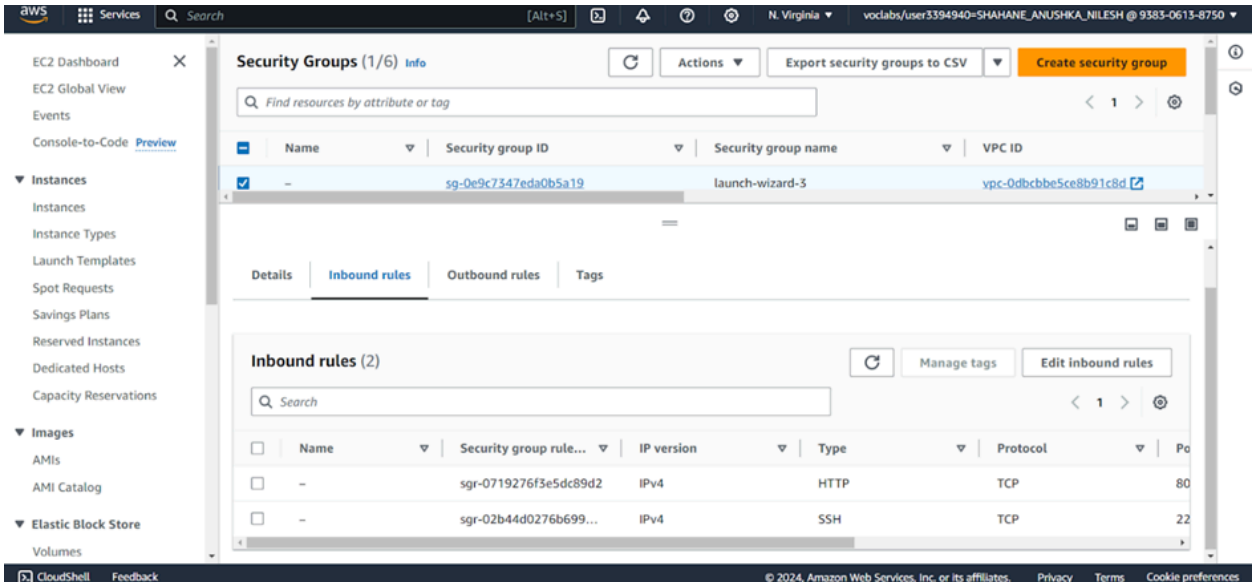
Anushka Shahane D15A 55

Using EC2 instance :

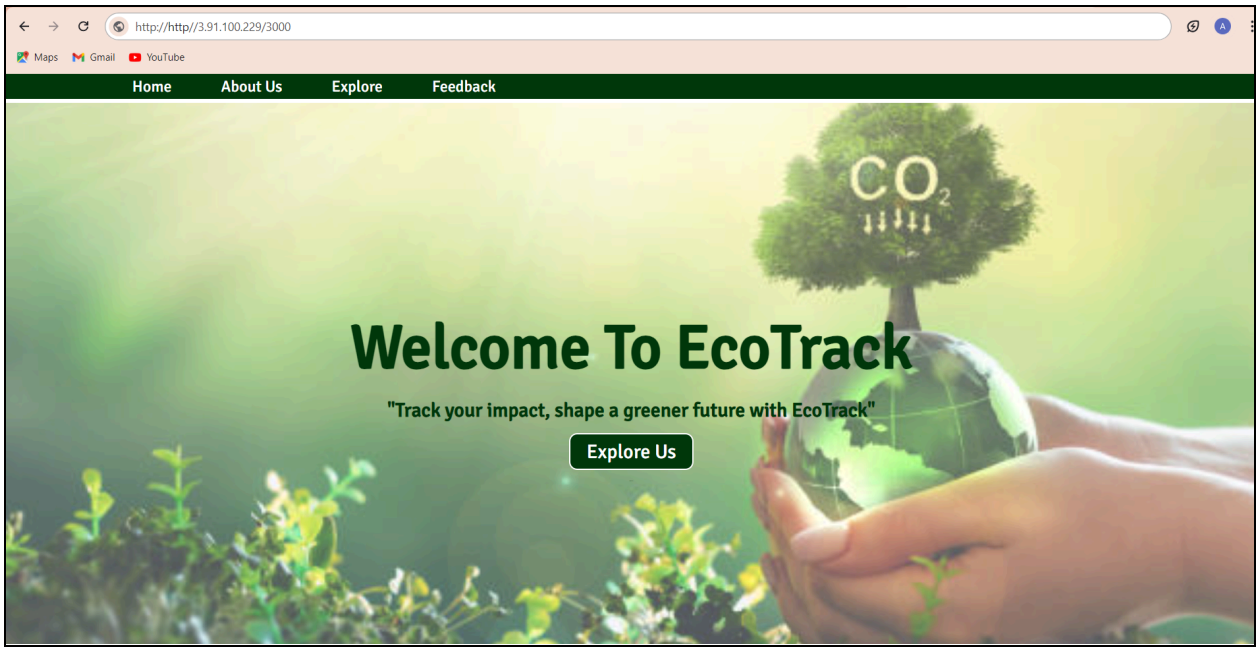




```
Dynamic_hosting_EC2  Dynamic_hosting_ec2  Hosting_dynamic_ec2  package-lock.json
ubuntu@ip-172-31-83-228:~/Anushka$ cd  Hosting_dynamic_ec2
ubuntu@ip-172-31-83-228:~/Anushka/Hosting_dynamic_ec2$ ls
index.js  package-lock.json  package.json
ubuntu@ip-172-31-83-228:~/Anushka/Hosting_dynamic_ec2$ npm i

added 64 packages, and audited 65 packages in 1s

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

found 0 vulnerabilities
ubuntu@ip-172-31-83-228:~/Anushka/Hosting_dynamic_ec2$ npm start

> server@1.0.0 start
> node index.js

Server is running on port 3000
```



Anushka Shahane D15A 55



Anushka Shahane D15A 55

Exp 3 Advance Devops

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

Theory:

Container-based microservices architectures have revolutionized how development and operations teams test and deploy modern software. Containers allow companies to scale and deploy applications more efficiently, but they also introduce new challenges, adding complexity by creating a whole new infrastructure ecosystem.

Today, both large and small software companies are deploying thousands of container instances daily.

Managing this level of complexity at scale requires advanced tools. Enter Kubernetes.

Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications.

Kubernetes has quickly become the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), supported by major players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes simplifies the deployment and operation of applications in a microservice architecture by providing an abstraction layer over a group of hosts. This allows development teams to deploy their applications while Kubernetes takes care of key tasks, including:

- Managing resource consumption by applications or teams
- Distributing application load evenly across the infrastructure
- Automatically load balancing requests across multiple instances of an application
- Monitoring resource usage to prevent applications from exceeding resource limits and automatically restarting them if needed
- Moving application instances between hosts when resources are low or if a host fails
- Automatically utilizing additional resources when new hosts are added to the cluster
- Facilitating canary deployments and rollbacks with ease

Necessary Requirements:

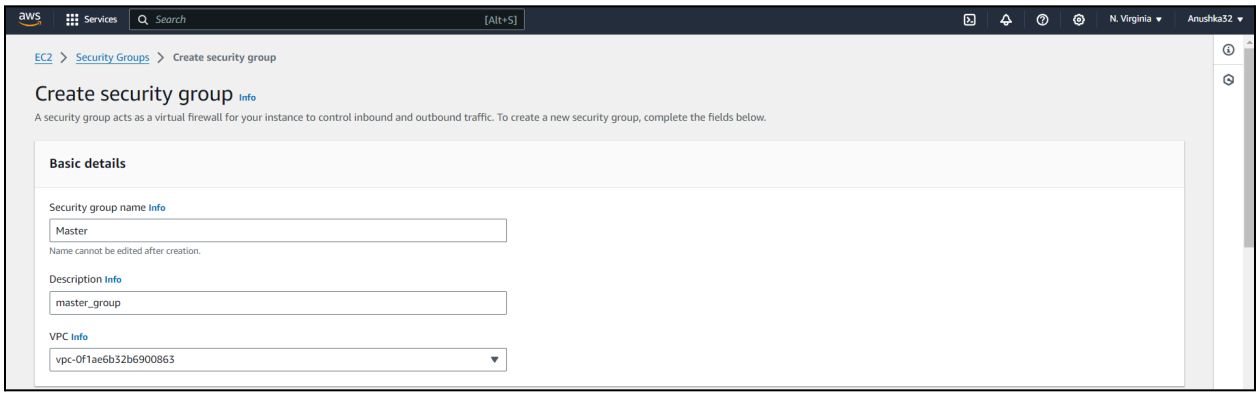
- EC2 Instance: The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.
- Minimum Requirements:
  - Instance Type: t2.medium
  - CPUs: 2
  - Memory: Adequate for container orchestration.

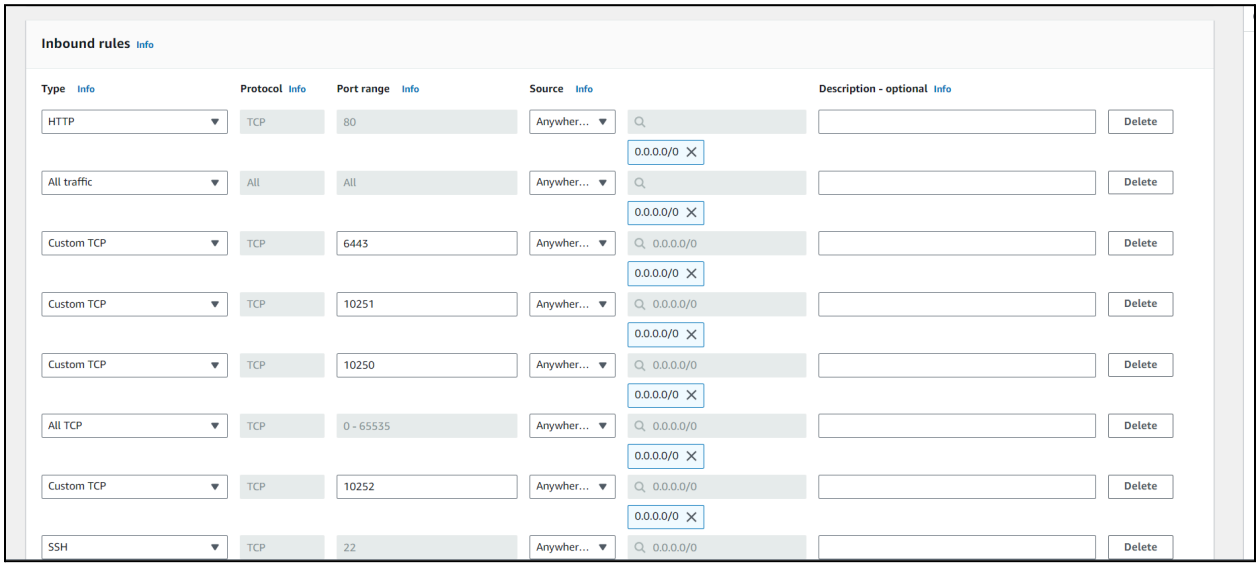
Anushka Shahane D15A 55

This ensured that the Kubernetes cluster had the necessary resources to function smoothly

Prerequisites :
Create 2 Security Groups for Master and Nodes and add the following rules inbound rules in those Groups.

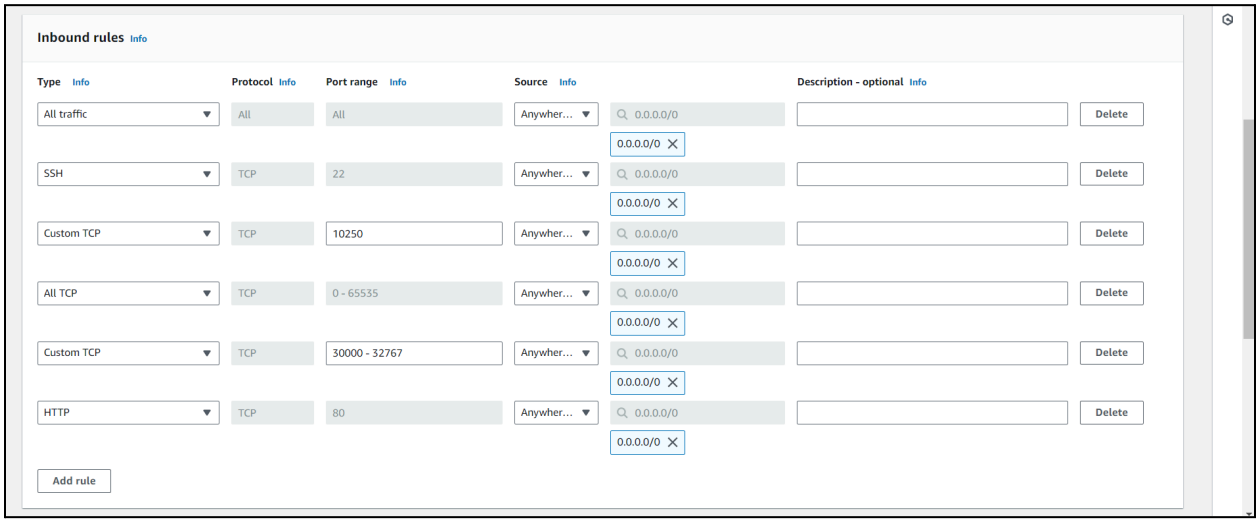
master





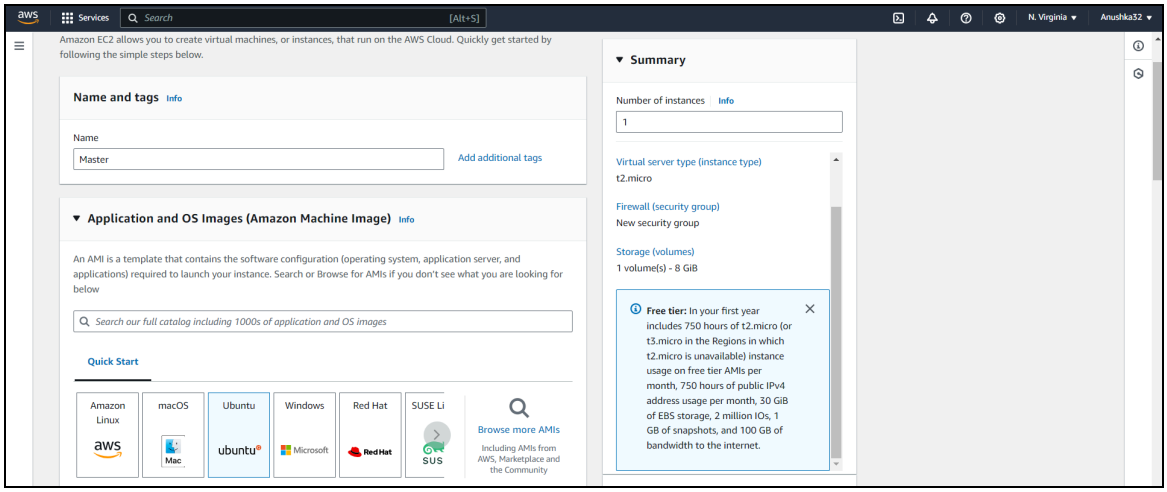
Anushka Shahane D15A 55

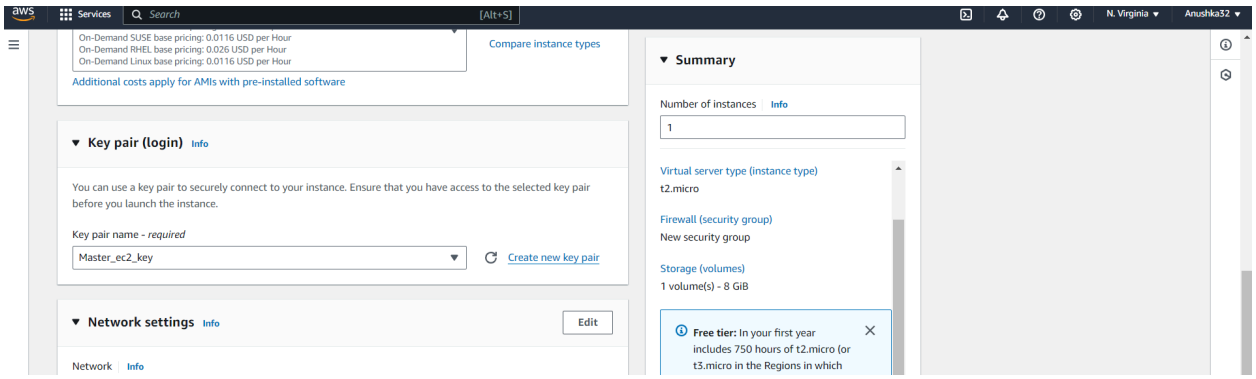
Node

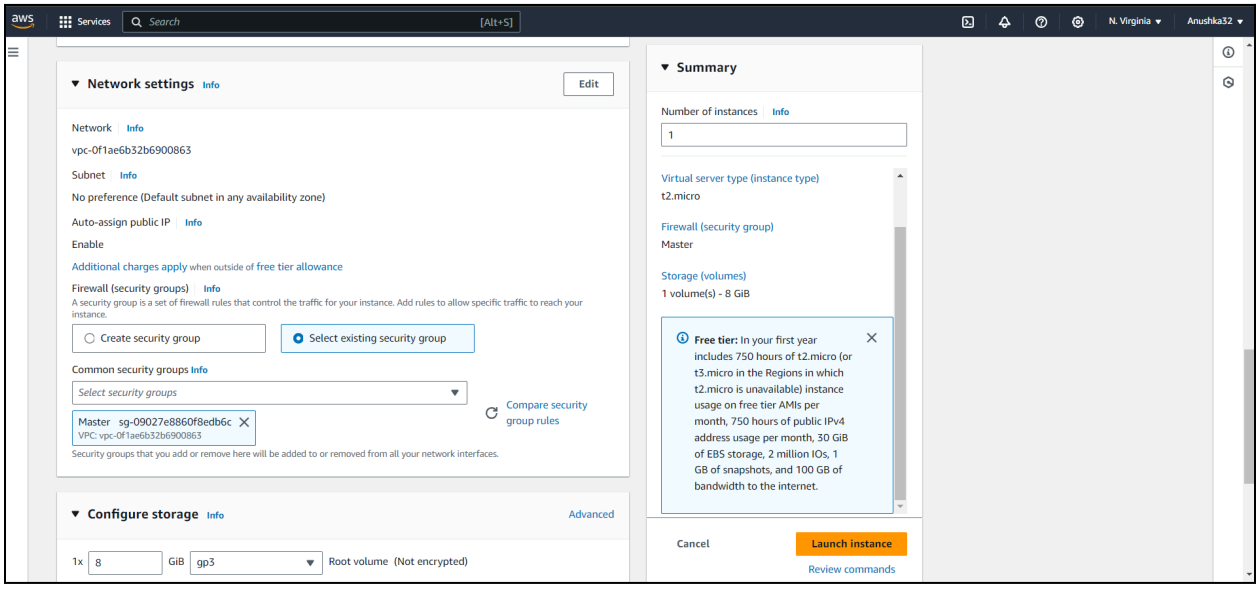


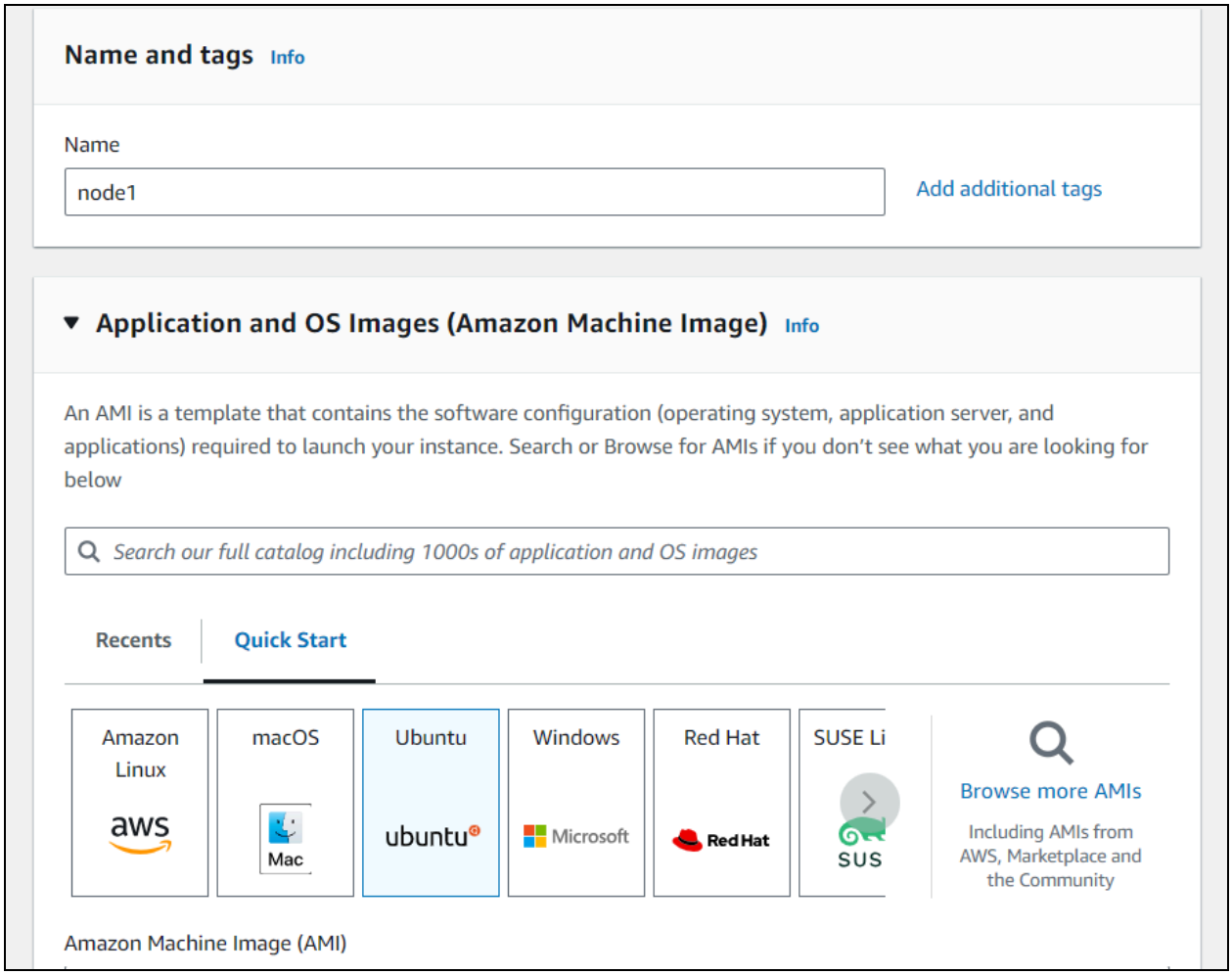
Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances. Select Ubuntu as AMI and t2.medium as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 3 Different keys or 1 common
key also.
Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the
instance after the experiment because it is not available in the free tier.
Also Select Security groups from existing.
Master:

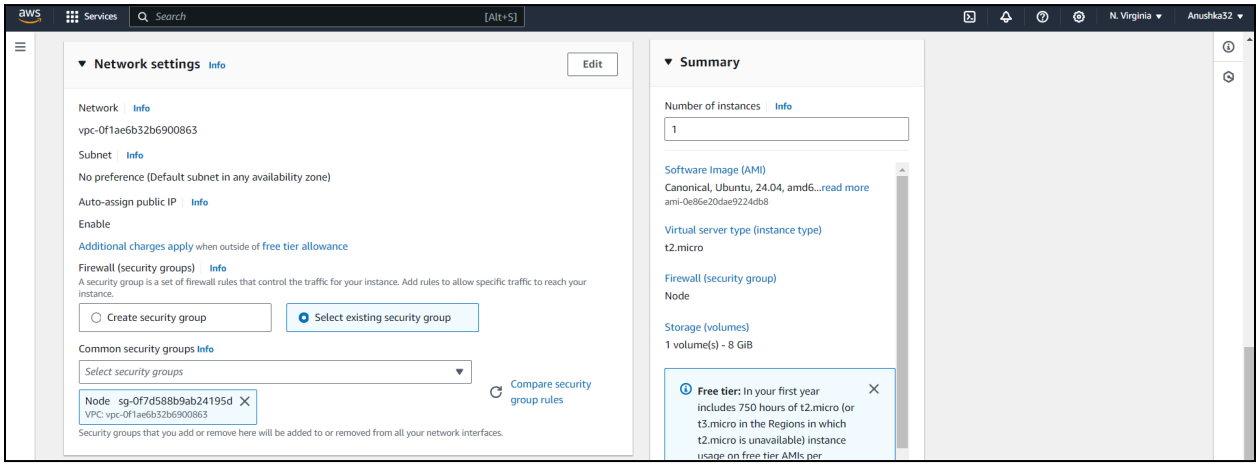
Anushka Shahane D15A 55





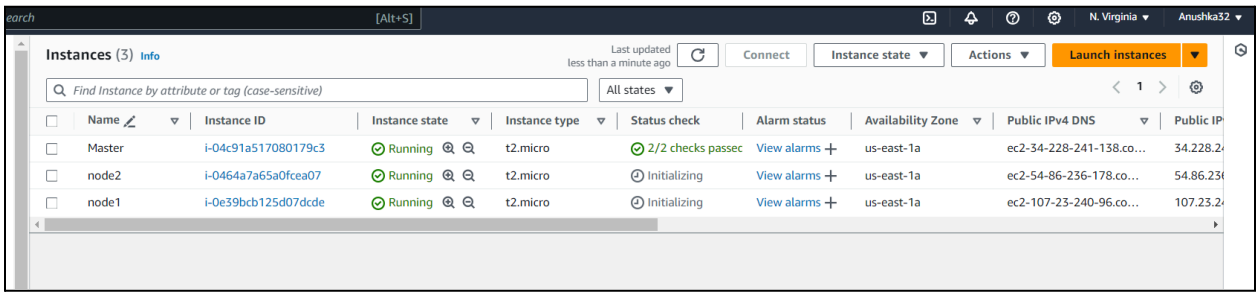


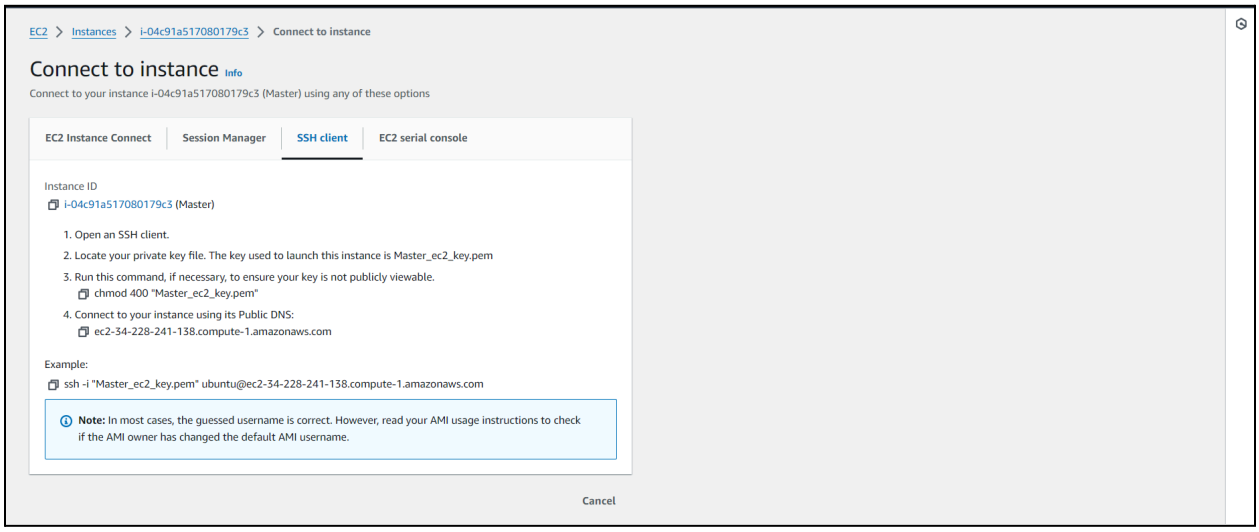

Name and tags Info
Name
node1
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)


Network settings Info
Edit
Network Info
vpc-0f1ae6b32b6900863
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Common security groups Info
Select security groups
Node sg-0f7d588b9ab24195d
VPC: vpc-0f1ae6b32b6900863
Compare security group rules
Security groups that you add or remove here will be added to or removed from all your network interfaces.
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8
Virtual server type (instance type)
t2.micro
Firewall (security group)
Node
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

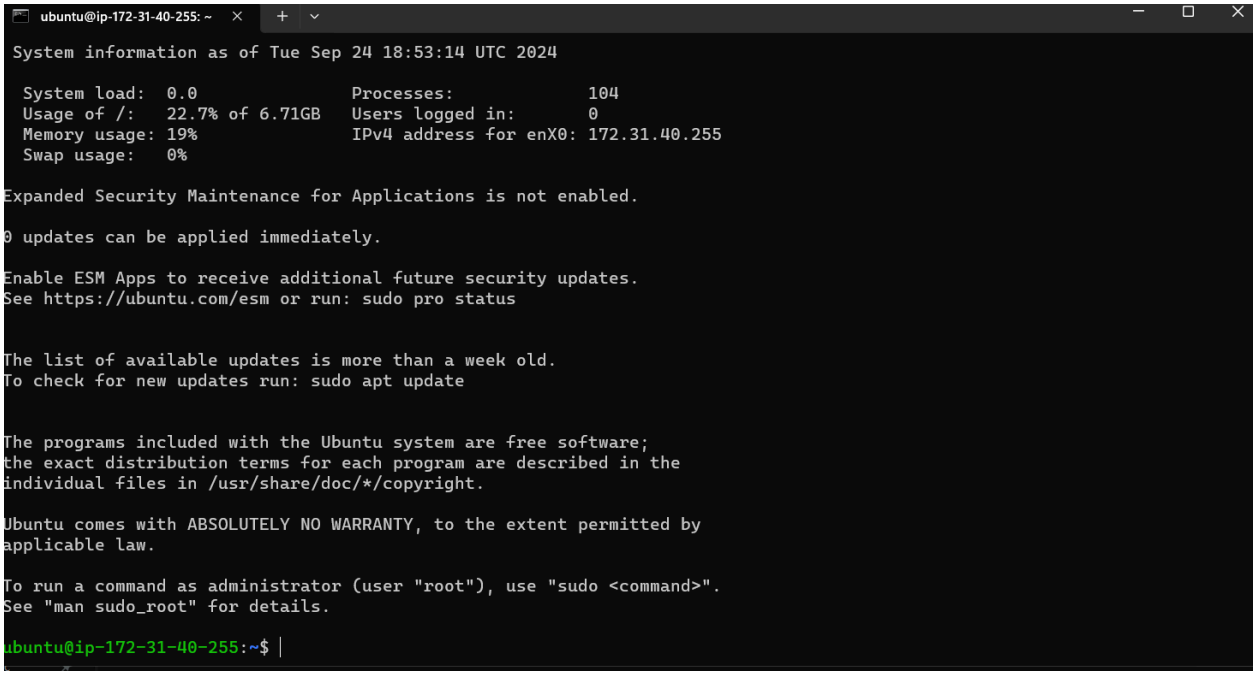
Anushka Shahane D15A 55

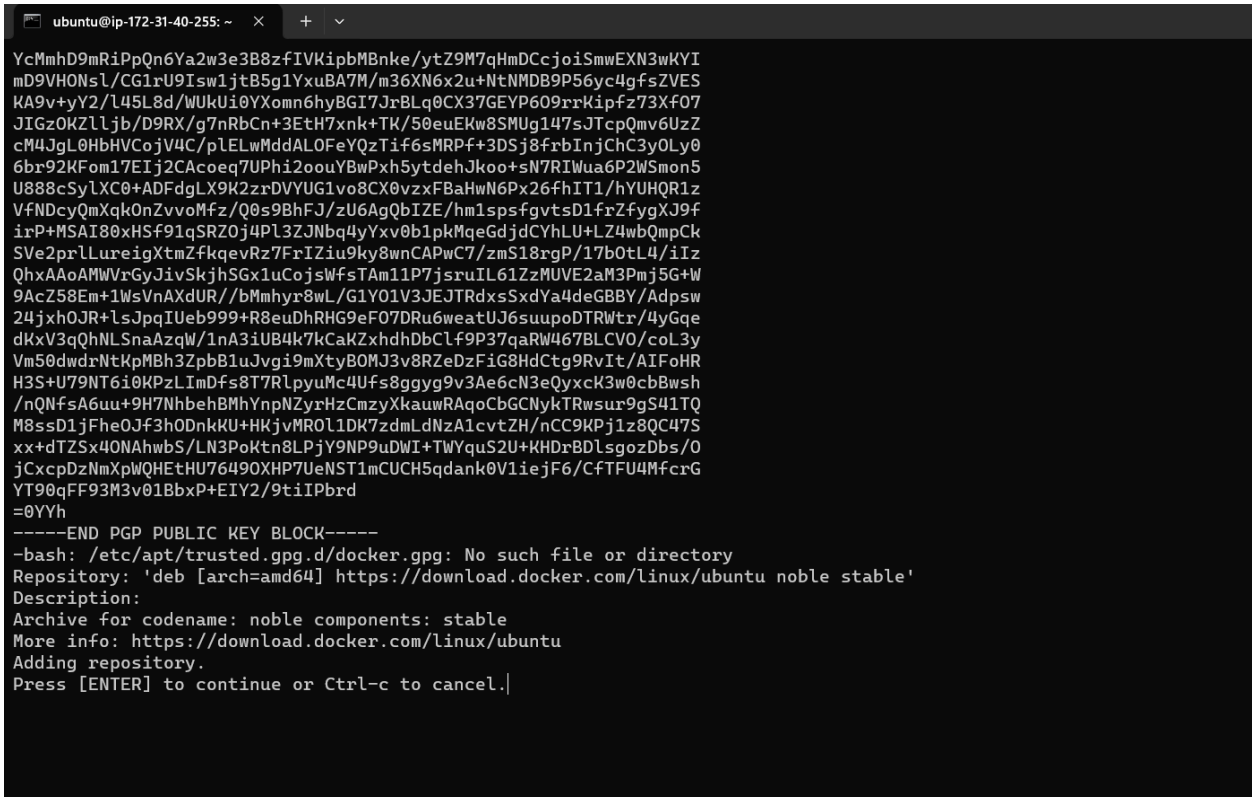
Do Same for 2 Nodes and use security groups of Node for that.
Step 2: After creating the instances click on Connect & connect all 3 instances and navigate to SSH
Client.



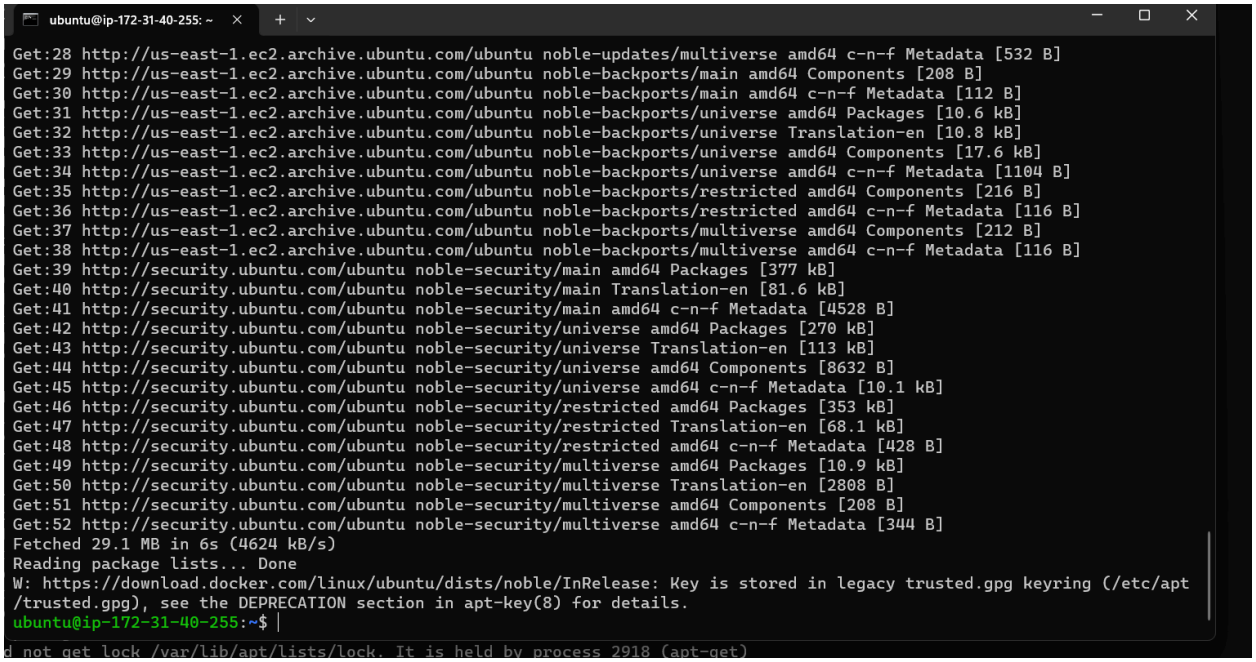


Step 3: Now open the folder in the terminal 3 times for Master, Node1& Node 2 where our .pem key is
stored and paste the Example command (starting with ssh -i .....) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)
Master:

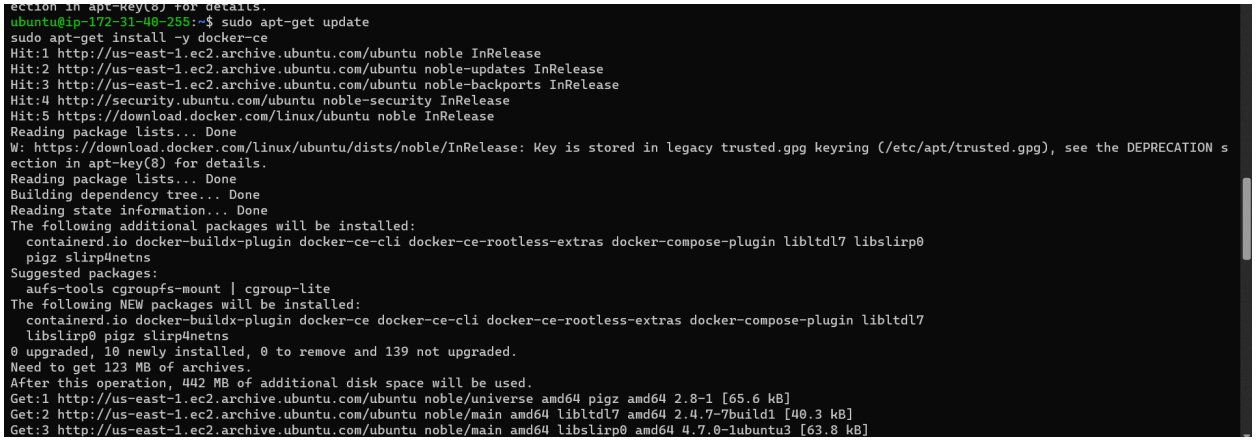

ubuntu@ip-172-31-40-255: ~
System information as of Tue Sep 24 18:53:14 UTC 2024
System load: 0.0
Usage of /: 22.7% of 6.71GB
Memory usage: 19%
Swap usage: 0%
Processes: 104
Users logged in: 0
IPv4 address for enX0: 172.31.40.255
Expanded Security Maintenance for Applications is not enabled.
0 updates 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-172-31-40-255:~$


ubuntu@ip-172-31-40-255: ~
YcMmhD9mRiPpQn6Ya2w3e3B8zfIVKipbMBnke/ytZ9M7qHmDCcjoiSmwEXN3wKYI
mD9VHONsl/CG1rU9Isw1jtB5g1YxuBA7M/m36XN6x2u+NtNMDB9P56yc4gfsZVES
KA9v+yY2/l45L8d/WUkUi0YXomn6hyBGI7JrBLq0CX37GEYP6O9rrKipfz73XfO7
JIGzOKZlljb/D9RX/g7nRbCn+3EtH7xnk+TK/50euEKw8SMUg147sJTcpQmv6UzZ
cM4JgL0HbHVCojV4C/plELwMddALOFeYQzTif6sMRPf+3DSj8frbInjChC3yOLy0
6br92KFom17EIj2CAcoeq7UPhi2oouYBwPxh5ytdehJkoo+sN7RIWua6P2WSmon5
U888cSylXC0+ADFdgLX9K2zrDVYUG1vo8CX0vzxFBaHwN6Px26fhIT1/hYUHQR1z
VfNDcyQmXqkOnZvvoMfz/Q0s9BhFJ/zU6AgQbIZE/hm1spsfgvtsD1frZfygXJ9f
irP+MSAI80xHSf91qSRZOj4Pl3ZJNbq4yYxv0b1pkMqeGdjdCYhLU+LZ4wbQmpCk
SVe2prlLureigXtmZfkqevRz7FrIZiu9ky8wnCAPwC7/zmS18rgP/17bOtL4/iIz
QhxAAoAMWVrGyJivSkjhSGx1uCojsWfsTAm11P7jsruIL61ZzMUVE2aM3Pmj5G+W
9AcZ58Em+1WsVnAXdUR//bMmhyr8wL/G1YO1V3JEJTRdxsSxdYa4deGBBY/Adpsw
24jxhOJR+lsJpqIUeb999+R8euDhRHG9eFO7DRu6weatUJ6suupoDTRWtr/4yGqe
dKxV3qQhNLSnaAzqW/1nA3iUB4k7kCaKZxhdhDbClf9P37qaRW467BLCVO/coL3y
Vm50dwdrNtKpMBh3ZpbB1uJvgi9mXtyBOMJ3v8RZeDzFiG8HdCtg9RvIt/AIFoHR
H3S+U79NT6i0KPzLImDfs8T7RlpyuMc4Ufs8ggyg9v3Ae6cN3eQyxcK3w0cbBwsh
/nQNfsA6uu+9H7NhbehBMhYnpNZyrHzCmzyXkauwRAqoCbGCNykTRwsur9gS41TQ
M8ssD1jFheOJf3hODnkKU+HKjvMROl1DK7zdmLdNzA1cvtZH/nCC9KPj1z8QC47S
xx+dTZSx4ONAhwbS/LN3PoKtn8LPjY9NP9uDWI+TWYquS2U+kHDrBDlsgozDbs/O
jCxcpDzNmXpWQHEtHU7649OXHP7UeNST1mCUCH5qdank0V1iejF6/CfTFU4MfcrG
YT90qFF93M3v01BbxP+EIY2/9tiIPbrd
=0YYh
-----END PGP PUBLIC KEY BLOCK-----
-bash: /etc/apt/trusted.gpg.d/docker.gpg: No such file or directory
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.

Anushka Shahane D15A 55



sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectL



```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-255:~$
```

sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

```
}
ubuntu@ip-172-31-40-255:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-255:~$
```

Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master,
Node1, and Node2.
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-255:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/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:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-40-255:~$
```

Anushka Shahane D15A 55

sudo apt-get update
sudo apt-get install -y docker-ce

Error

```
ubuntu@ip-172-31-40-255:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
ubuntu@ip-172-31-40-255:~$
```

To solve

```
ubuntu@ip-172-31-40-255:~$ sudo mkdir -p /etc/apt/keyrings
ubuntu@ip-172-31-40-255:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
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]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8564 B]
Get:8 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 141 kB in 1s (141 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.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 139 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
79% [5 kubernetes-cni 33.9 MB/33.9 MB 100%]
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-40-255:~$
```

sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

```
ubuntu@ip-172-31-40-255:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-40-255:~$
```

Step 5: Run the below command to install Kubernets.
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/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:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-40-255:~$ sudo systemctl enable --now kubelet
sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 139 not upgraded.
Need to get 47.2 MB of archives.
```

sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

Anushka Shahane D15A 55

```
ubuntu@ip-172-31-40-255:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
```

sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd

```
  uid = 0
ubuntu@ip-172-31-40-255:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-09-24 19:23:12 UTC; 420ms ago
       Docs: https://containerd.io
   Main PID: 8629 (containerd)
      Tasks: 7
     Memory: 15.8M (peak: 16.0M)
        CPU: 82ms
     CGroup: /system.slice/containerd.service
             └─8629 /usr/bin/containerd

Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.238510263Z" level=info msg=serving... address=/run/containerd/containerd.sock.>
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.238584725Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240040850Z" level=info msg="Start subscribing containerd event"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240176953Z" level=info msg="Start recovering state"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240253922Z" level=info msg="Start event monitor"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240273184Z" level=info msg="Start snapshots syncer"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240286270Z" level=info msg="Start cni network conf syncer for default"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240299291Z" level=info msg="Start streaming server"
Sep 24 19:23:12 ip-172-31-40-255 systemd[1]: Started containerd.service - containerd container runtime.
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.244849418Z" level=info msg="containerd successfully booted in 0.070201s"
ubuntu@ip-172-31-40-255:~$
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-40-255:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 139 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (11.6 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-255:~$
```

Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-255:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR NumCPU]: the number of available CPUs 1 is less than the required 2
        [ERROR Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-40-255:~$
```

```
ubuntu@ip-172-31-40-255:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU --ignore-preflight-errors=Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
[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 beforehand using 'kubeadm config images pull'
W0924 19:29:57.933799    9246 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that
used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" 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-172-31-40-255 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster
.local] and IPs [10.96.0.1 172.31.40.255]
[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-172-31-40-255 localhost] and IPs [172.31.40.255 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-40-255 localhost] and IPs [172.31.40.255 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

```
Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.40.255:6443 --token l2izt1.mt5iy3g7o0yhjft7 \
        --discovery-token-ca-cert-hash sha256:39c290262a4af785e7629a945f25514226b3f65234f280fe02b033f0f9924cfc
ubuntu@ip-172-31-40-255:~$
```

Run this command on master and also copy and save the Join command from above.
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
        --discovery-token-ca-cert-hash sha256:39c290262a4af785e7629a945f25514226b3f65234
ubuntu@ip-172-31-40-255:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-40-255:~$
```

Step 7: Now Run the command kubectl get nodes to see the nodes before executing Join command on nodes.

```
ubuntu@ip-172-31-40-255:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE    VERSION
ip-172-31-40-255    NotReady   control-plane   113s   v1.31.1
ubuntu@ip-172-31-40-255:~$
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.

Anushka Shahane D15A 55

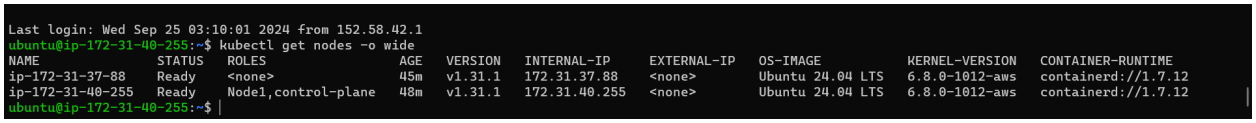
sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \
--discovery-token-ca-cert-hash
sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6
Node 1:

```
ubuntu@ip-172-31-40-255:~$ sudo kubeadm join 172.31.40.255:6443 --token l2izt1.mt5iy3g7o0yhjft7 --discovery-token-ca-cer
t-hash sha256:39c290262a4af785e7629a945f25514226b3f65234f280fe02b033f0f9924cfc
[preflight] Running pre-flight checks
[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'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.509666981s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Step 9: Now Run the command kubectl get nodes to see the nodes after executing Join command on nodes.

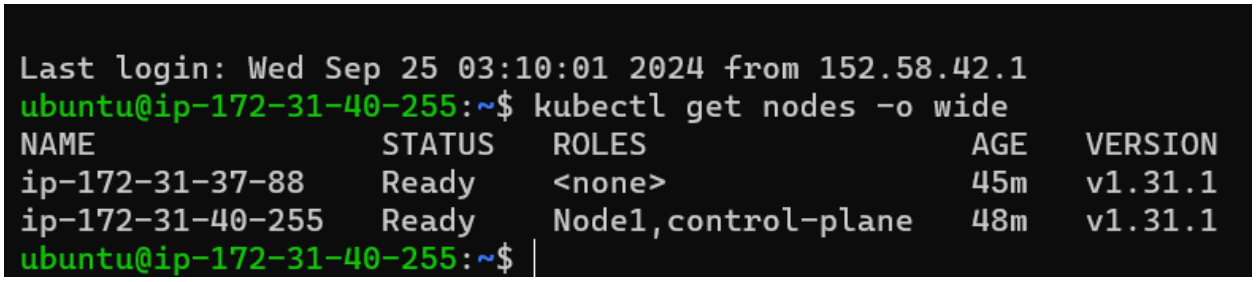
```
Last login: Wed Sep 25 03:09:14 2024 from 152.58.42.1
ubuntu@ip-172-31-37-88:~$ sudo kubeadm join 172.31.40.255:6443 --token zo9fea.16bddwnc11vq1qso \
    --discovery-token-ca-cert-hash sha256:a92bc7fadc6f973441bb6c3278fbd5f33e67ed5c9dc5d7b83aa2aaf2b56b0510
[preflight] Running pre-flight checks
[preflight] 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'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.636003ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@ip-172-31-37-88:~$ client_loop: send disconnect: Connection reset

C:\Users\Admin\Desktop\Node1_key>
```

Now Run command kubectl get nodes -o wide we can see Status is ready.



Step 11: Run command kubectl get nodes -o wide . And Hence we can see we have Successfully
connected Node 1 and Node 2 to the Master.

Or run kubectl get nodes



Anushka Shahane D15A 55

Advance Devops Experiment 4

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 tool for interacting with Kubernetes clusters. It allows you to manage Kubernetes resources by creating, updating, and deleting pods, deployments,
services, and more.

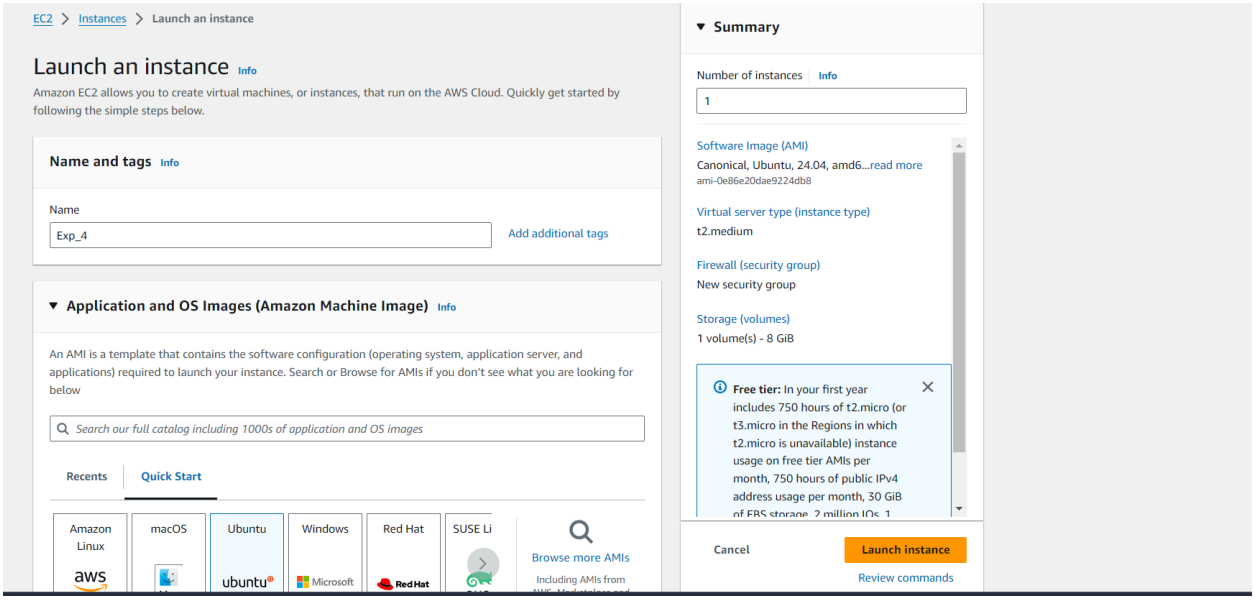
Prerequisites

A Kubernetes cluster running either locally (e.g., with Minikube, Kind, or Docker Desktop) or remotely (cloud-based, such as Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), or Azure Kubernetes Service (AKS)).

kubectl installed on your local machine to interact with the cluster.

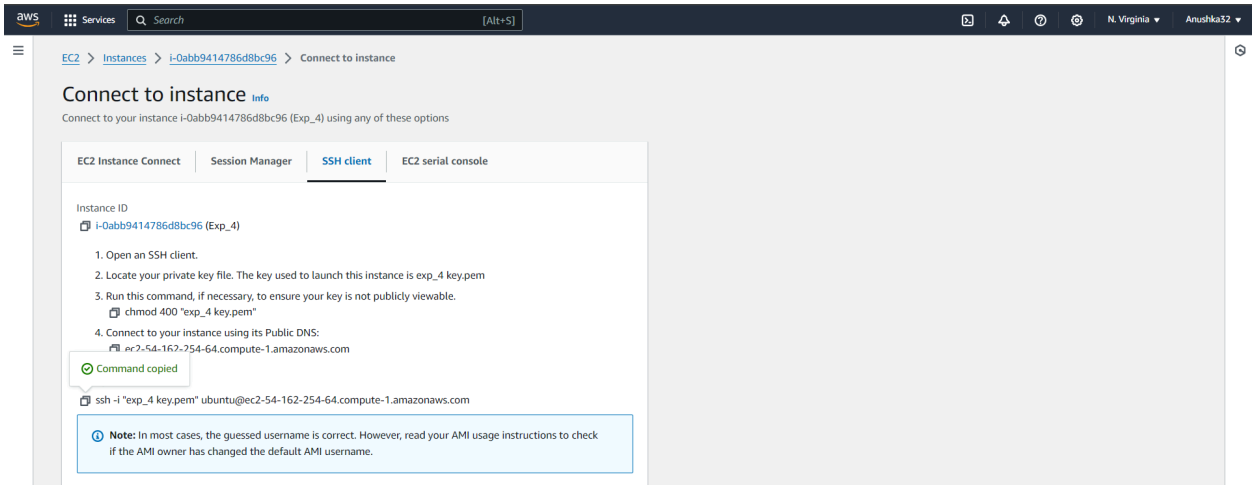
Step 1:

Go to AWS Academia in services select EC2 and create 3 instance with instance type t2.medium and names as node1, node2 and master

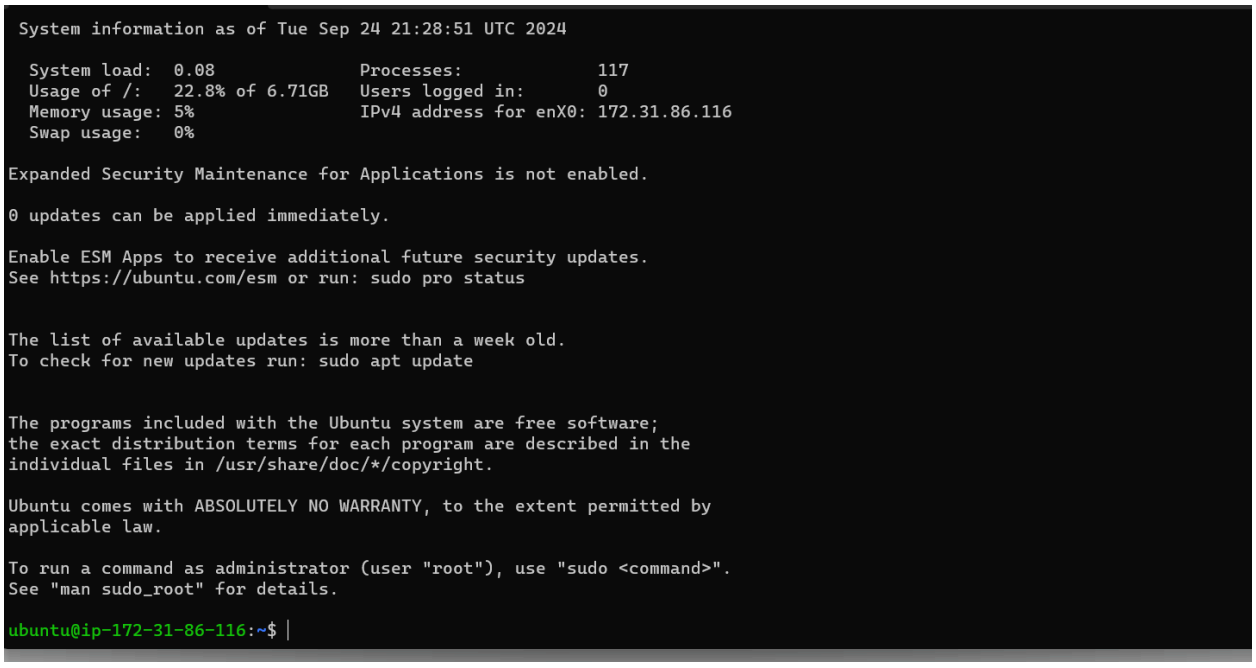


Anushka Shahane D15A 55

Step 2: Create a new key pair and name it as myKey1 and download as .pem file.
Open command promt run the following command



Step 3: Now open the folder in the terminal where our .pem key is stored and paste the Example
command (starting with ssh -i …..) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)



Anushka Shahane D15A 55

Step 4: Run the below commands to install and setup Docker.
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable”

```
ubuntu@ip-172-31-86-116:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
XYukHjlxxEgBdC/J3cMMNRE1F4NCA3ApfV1Y7/hTeOnmDuDYwr9/obA8t016Yljj
q5rdkywPf4JF8mXUW5eCN1vAFHxeg9ZWemhBtQmGxXnw9M+z6hWwc6ahmwARAQAB
tCtEb2NrZXIgUmVsZWFzZSAoQ0UgZGViKSA8ZG9ja2VyQGRvY2tlci5jb20+iQI3
BBMBCgAhBQJYrefAAhsvBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEI2BgDwO
```

sudo apt-get update
sudo apt-get install -y docker-ce

```
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-86-116:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
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 section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0
  pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 139 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
```

Anushka Shahane D15A 55

sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

```
ubuntu@ip-172-31-86-116:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-86-116:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-86-116:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/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:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-86-116:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
```

sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-86-116:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-86-116:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/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:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-86-116:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
ubuntu@ip-172-31-86-116:~$
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

Step 5: Run the below command to install Kubernets.
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/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:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-86-116:~$ sudo mkdir -p /etc/apt/keyrings
ubuntu@ip-172-31-86-116:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (11.3 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 section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 139 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7
 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 M
```

```
ubuntu@ip-172-31-86-116:~$ kubectl describe pod nginx-deployment-d556bf558-dmdsr
Name:             nginx-deployment-d556bf558-dmdsr
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           app=nginx
                  pod-template-hash=d556bf558
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Controlled By:    ReplicaSet/nginx-deployment-d556bf558
```

```
ubuntu@ip-172-31-86-116:~$ kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
error: flag needs an argument: 'f' in -f
See 'kubectl apply --help' for usage.
-bash: https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml: No such file or directory
ubuntu@ip-172-31-86-116:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-f
lannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-86-116:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-86-116:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-dmdsr   0/1     Pending   0          10s
nginx-deployment-d556bf558-kf9l4   0/1     Pending   0          10s
ubuntu@ip-172-31-86-116:~$
POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8081:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-86-116:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untaint
ed
error: at least one taint update is required
ubuntu@ip-172-31-86-116:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-86-116   Ready    control-plane   4m15s   v1.31.1
ubuntu@ip-172-31-86-116:~$
```

sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-86-116:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 139 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (14.0 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
```

sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd

```
ubuntu@ip-172-31-86-116:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-09-24 21:37:08 UTC; 220ms ago
       Docs: https://containerd.io
   Main PID: 4764 (containerd)
      Tasks: 7
     Memory: 13.4M (peak: 14.0M)
        CPU: 55ms
     CGroup: /system.slice/containerd.service
             └─4764 /usr/bin/containerd

Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.940185812Z" level=info msg="Start subscrib>
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.940228819Z" level=info msg="Start recoveri>
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.940274910Z" level=info msg="Start event mo>
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.940284757Z" level=info msg="Start snapshot>
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.940292060Z" level=info msg="Start cni netw>
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.940298061Z" level=info msg="Start streamin>
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.940208336Z" level=info msg=serving... addr>
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.941160760Z" level=info msg=serving... addr>
Sep 24 21:37:08 ip-172-31-86-116 systemd[1]: Started containerd.service - containerd container runtime.
Sep 24 21:37:08 ip-172-31-86-116 containerd[4764]: time="2024-09-24T21:37:08.941884864Z" level=info msg="containerd suc>
```

Step 6: Initialize the Kubecluster

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-86-116:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[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 beforehand using 'kubeadm config images pull'
W0924 21:37:32.938984    4937 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
dbox 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-172-31-86-116 kubernetes kubernetes.default kubernetes.defaul
t.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.86.116]
[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-172-31-86-116 localhost] and IPs [172.31.86.116 127.0.0.1 :
:1]
[certs] Generating "etcd/peer" certificate and key
```

Copy the mkdir and chown commands from the top and execute them.

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
ubuntu@ip-172-31-86-116:~$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-86-116:~$ kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
error: flag needs an argument: 'f' in -f
See 'kubectl apply --help' for usage.
-bash: https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml: No such file or directory
ubuntu@ip-172-31-86-116:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-f
lannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-86-116:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-86-116:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-dmdsr   0/1     Pending   0          10s
nginx-deployment-d556bf558-kf9l4   0/1     Pending   0          10s
ubuntu@ip-172-31-86-116:~$
POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
```

Add a common networking plugin called flannel as mentioned in the code.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
ubuntu@ip-172-31-86-116:~$ kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
error: flag needs an argument: 'f' in -f
See 'kubectl apply --help' for usage.
-bash: https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml: No such file or directory
ubuntu@ip-172-31-86-116:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-f
lannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-86-116:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-86-116:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-dmdsr   0/1     Pending   0          10s
nginx-deployment-d556bf558-kf9l4   0/1     Pending   0          10s
ubuntu@ip-172-31-86-116:~$
POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8081:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-86-116:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untaint
ed
error: at least one taint update is required
ubuntu@ip-172-31-86-116:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-86-116   Ready    control-plane   4m15s   v1.31.1
ubuntu@ip-172-31-86-116:~$
```

Step 7: Now that the cluster is up and running, we can deploy our nginx server on this cluster.Apply this deployment file using this command to create a deployment
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

Anushka Shahane D15A 55

```
ubuntu@ip-172-31-86-116:~$
POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8081:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-86-116:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untaint
ed
error: at least one taint update is required
ubuntu@ip-172-31-86-116:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-86-116   Ready    control-plane   4m15s   v1.31.1
ubuntu@ip-172-31-86-116:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-dmdsr   0/1     Pending   0          2m52s
nginx-deployment-d556bf558-kf9l4   0/1     Pending   0          2m52s
ubuntu@ip-172-31-86-116:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
```

```
heduling.
ubuntu@ip-172-31-86-116:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/ip-172-31-86-116 untainted
ubuntu@ip-172-31-86-116:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-dmdsr   1/1     Running   0          6m29s
nginx-deployment-d556bf558-kf9l4   1/1     Running   0          6m29s
ubuntu@ip-172-31-86-116:~$
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

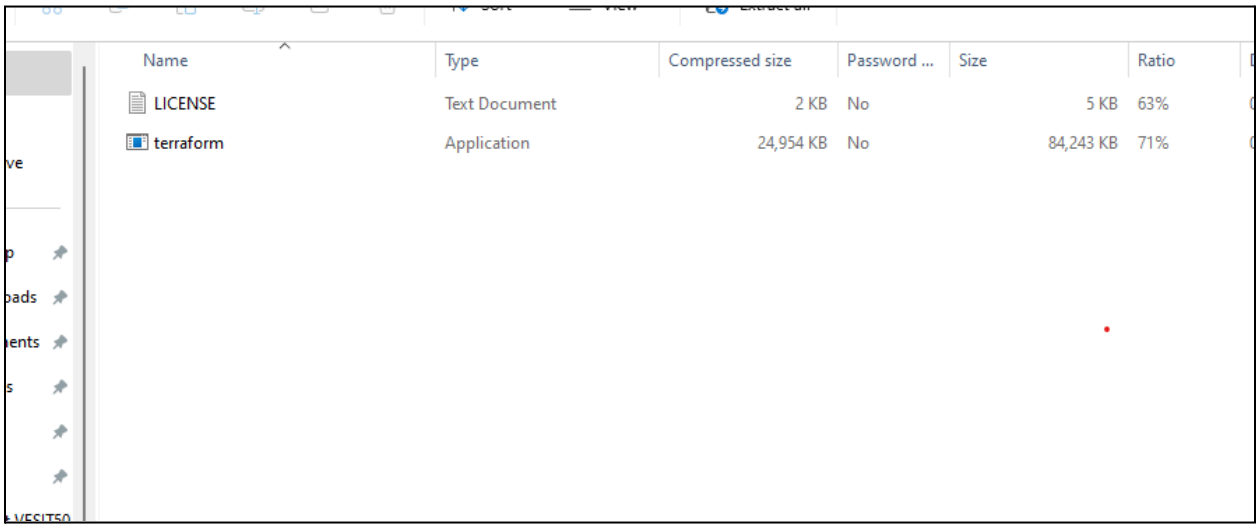
```
ubuntu@ip-172-31-86-116:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

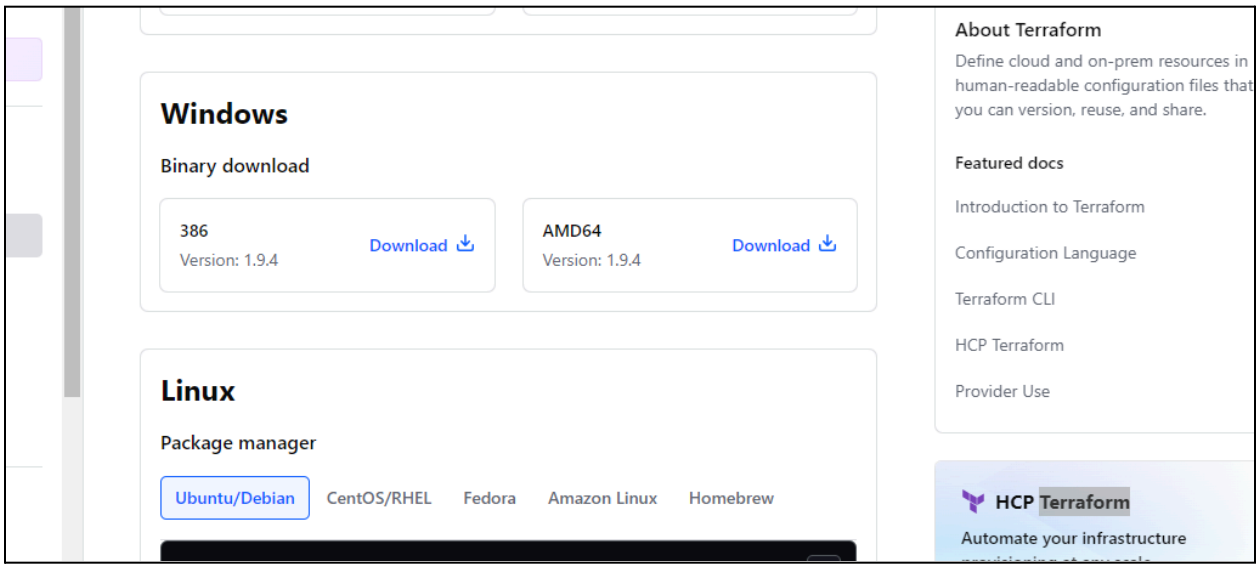
kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted
kubectl get nodes

```
Last login: Tue Sep 24 21:28:52 2024 from 103.88.83.126
ubuntu@ip-172-31-86-116:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Tue, 24 Sep 2024 21:48:41 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes

ubuntu@ip-172-31-86-116:~$
```

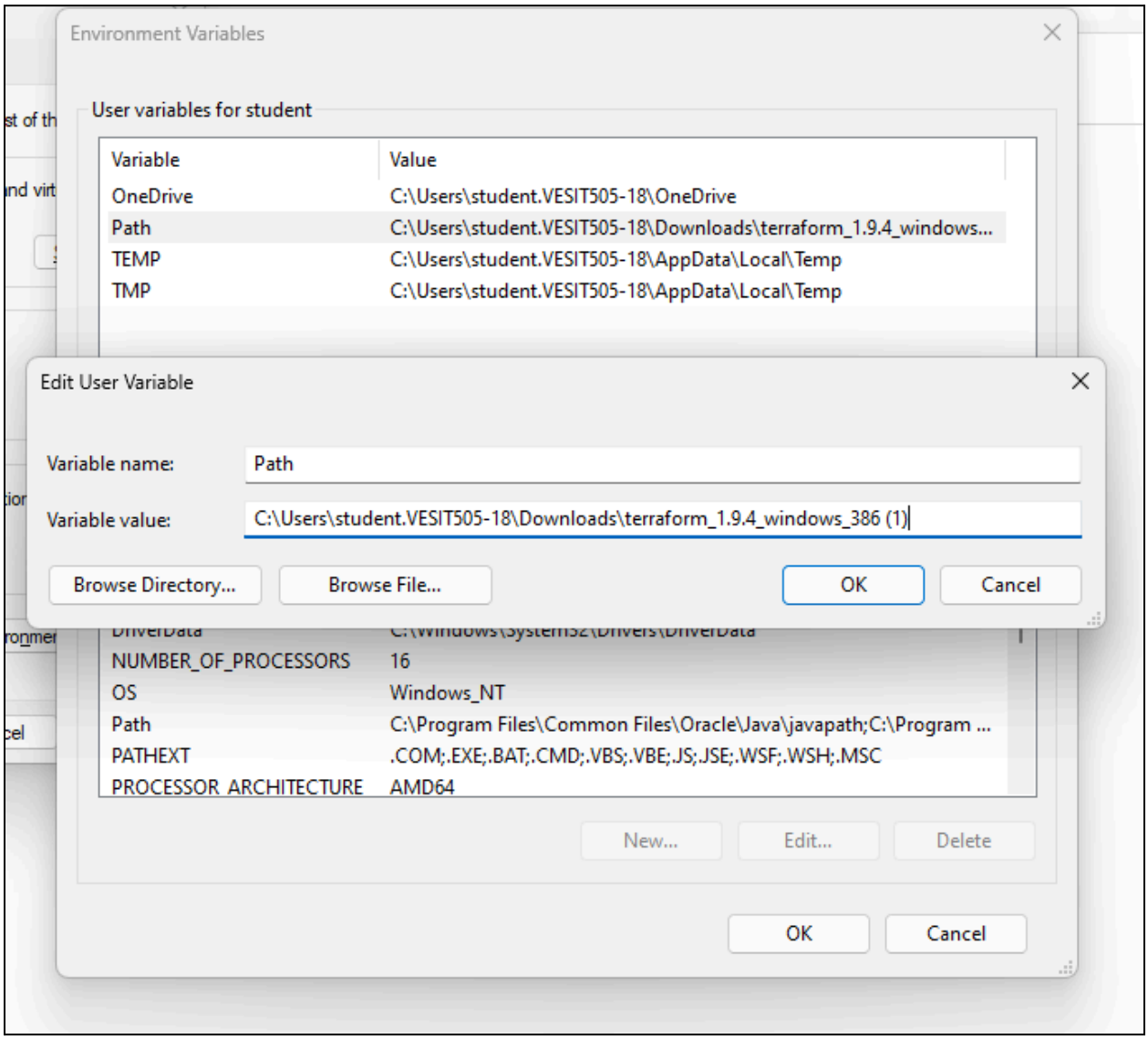
Experiment no. 5

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





Anushka Shahane D15A 55



```
C:\Users\student.VESIT505-18>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
  taint         Mark a resource instance as not fully functional
  test          Execute integration tests for Terraform modules
  untaint       Remove the 'tainted' state from a resource instance
```

```
C:\Users\student.VESIT505-18>terraform --version
Terraform v1.9.4
on windows_386

C:\Users\student.VESIT505-18>
```

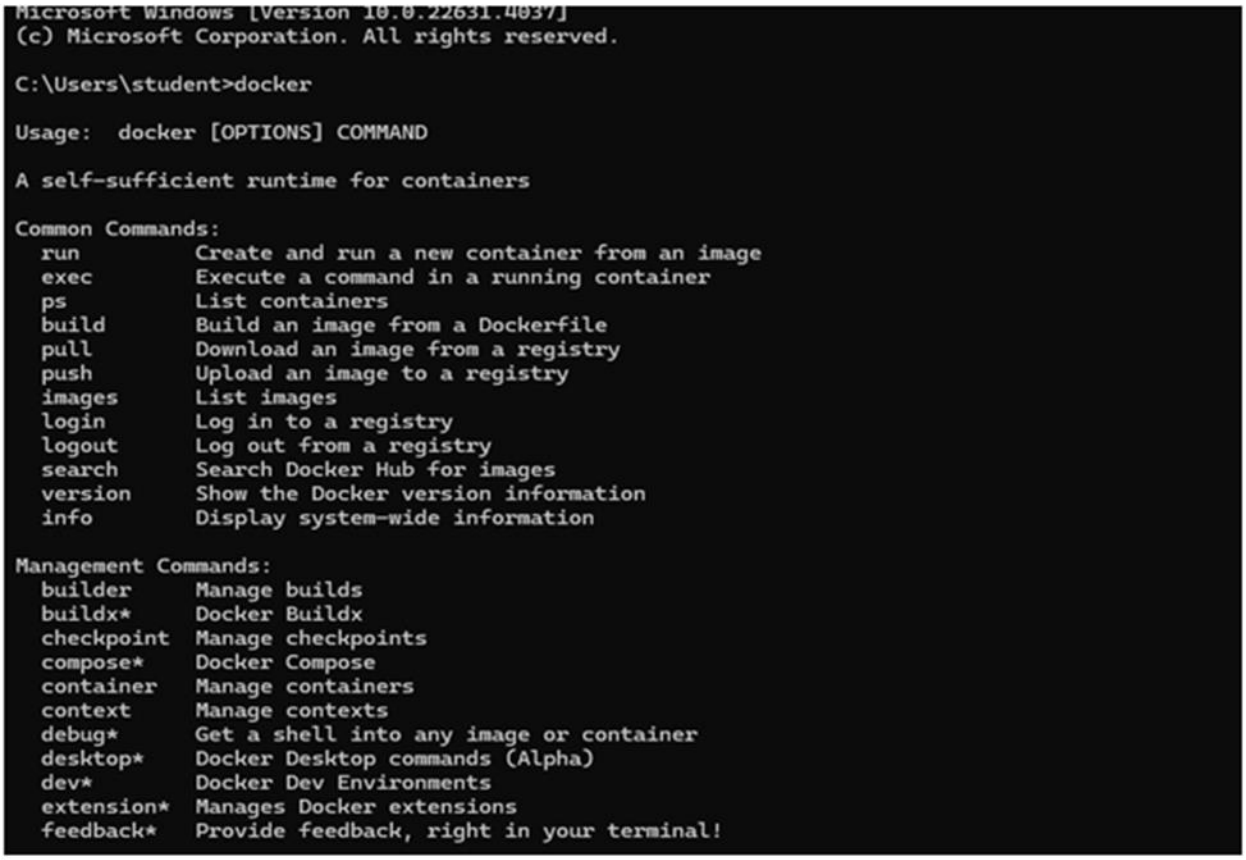
Anushka Shahane D15A 55

ADVANCE DEVOPS EXPERIMENT 6

Aim : To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform. (S3 bucket or Docker) fdp.

Part A: Creating docker image using terraform

Step 1:Check Docker functionality



Check for the docker version with the following command.

```
C:\Users\student>docker --version
Docker version 27.1.1, build 6312585

C:\Users\student>
```

Create a folder named ‘Terraform Scripts’ in which we save our different types of scripts which will be further used in this experiment.

Step 2:

Creating a new folder named ‘Docker’ in the ‘TerraformScripts’ folder.
Creating a new docker.tf file using Atom editor and write the following contents into.

This will create a Ubuntu Linux container

docker.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

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

# Pull the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
  command = ["sleep", "3600"]
}
```

Step 3: Execute Terraform Init command to initialize the resources

```
PS C:\Users\Admin\TerraformScripts> cd Docker
PS C:\Users\Admin\TerraformScripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (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: Execute Terraform plan to see the available resources

```
PS C:\Users\Admin\TerraformScripts\Docker> terraform plan

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "3600",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + 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

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

Step 5: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration. Using command : “terraform apply”

```
PS C:\Users\Admin\TerraformScripts\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.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "3600",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
```

Anushka Shahane D15A 55

```
      + 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

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + 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.ubuntu: Creating...
docker_image.ubuntu: Creation complete after 9s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 2s [id=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]

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

Before Executing Apply step:

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

After Executing Apply step:

```
PS C:\Users\Admin\TerraformScripts\Docker> docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
ubuntu                              latest    edbfe74c41f8   3 weeks ago     78.1MB
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\Users\Admin\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]

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.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach           = false -> null
      - command          = [
          - "sleep",
          - "3600",
        ] -> null
      - cpu_shares       = 0 -> null
      - dns              = [] -> null
      - dns_opts         = [] -> null
      - dns_search       = [] -> null
      - entrypoint       = [] -> null
      - env              = [] -> null
      - gateway          = "172.17.0.1" -> null
      - group_add        = [] -> null
      - hostname         = "01adf07e5918" -> null
      - id               = "01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24" -> null
      - image            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init             = false -> null
      - ip_address       = "172.17.0.2" -> null
      - ip_prefix_length = 16 -> null
      - ipc_mode         = "private" -> null
      - links            = [] -> 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
      - name             = "foo" -> null
      - network_data     = [
          - {
              - gateway                   = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                = "172.17.0.2"
              - ip_prefix_length          = 16
              - network_name              = "bridge"
                # (2 unchanged attributes hidden)
            },
        ] -> null
      - network_mode     = "default" -> null
      - privileged       = false -> null
      - publish_all_ports = false -> null
      - read_only        = false -> null
      - remove_volumes   = true -> null
      - restart          = "no" -> null
      - rm               = false -> null
      - runtime          = "runc" -> null
      - security_opts    = [] -> null
      - shm_size         = 64 -> null
      - start            = true -> null
      - stdin_open       = false -> null
      - stop_timeout     = 0 -> null
      - storage_opts     = {} -> null
      - sysctls          = {} -> null
      - tmpfs            = {} -> null
      - tty              = false -> null
        # (8 unchanged attributes hidden)
    }
```

```
  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
      - image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - latest      = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - name        = "ubuntu:latest" -> null
      - repo_digest = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> 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.foo: Destroying... [id=01adf07e5918931fee9b90073726a03671037923dd92032ce0e15bbb764a6f24]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
```

Docker images After Executing Destroy step

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

ADVANCE DEVOPS EXPERIMENT 7

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

Theory: Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being
considered an afterthought.
SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the

codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence. Thus, integrating static analysis into the SDLC can yield dramatic results in the overall quality of the
code developed.

What are the key steps to run SAST effectively?

There are six simple steps needed to perform SAST efficiently in organizations that have a very large number of applications built with different languages, frameworks, and platforms.

1. **Finalize the tool.** Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.

2. **Create the scanning infrastructure, and deploy the tool.** This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.

3. **Customize the tool.** Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build
   environment, create dashboards for tracking scan results, and build custom reports.

4. **Prioritize and onboard applications.** Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with
   application scans synced with release cycles, daily or monthly builds, or code check-ins.

5. **Analyze scan results.** This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.

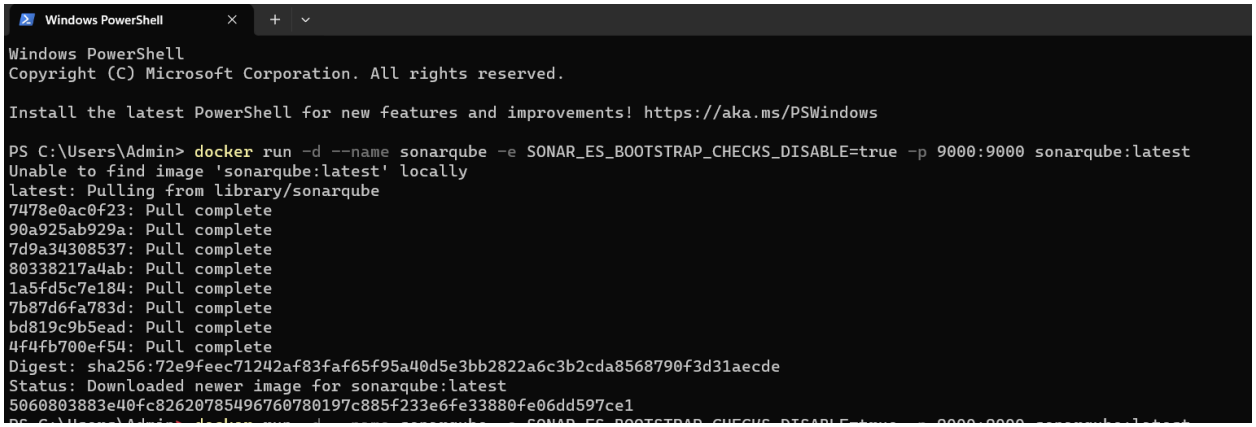
6. **Provide governance and training**

Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints
should be present within the SDLC. SAST should be incorporated as part of your application development and deployment process.

Steps to integrate Jenkins with SonarQube

1. Open up Jenkins Dashboard on localhost, port 8080 .

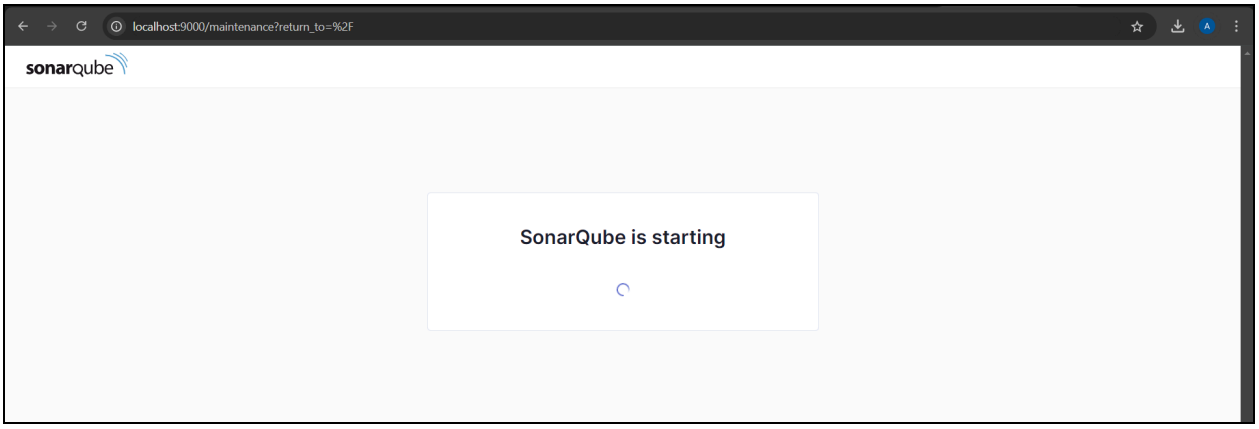
2. Run SonarQube in a Docker container using this command -



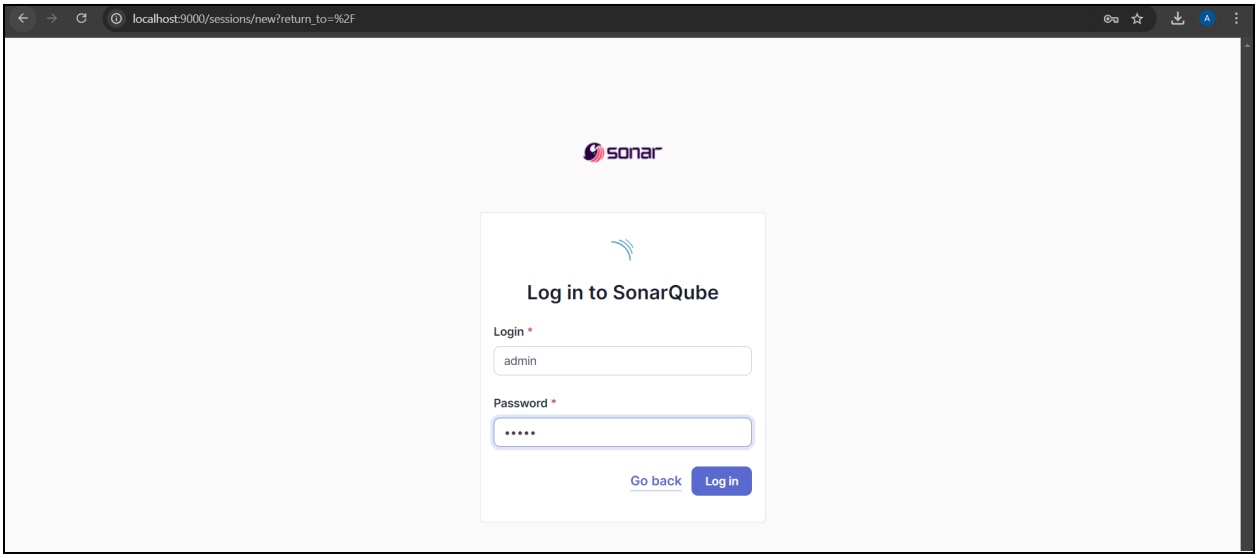
Warning: run below command only once

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

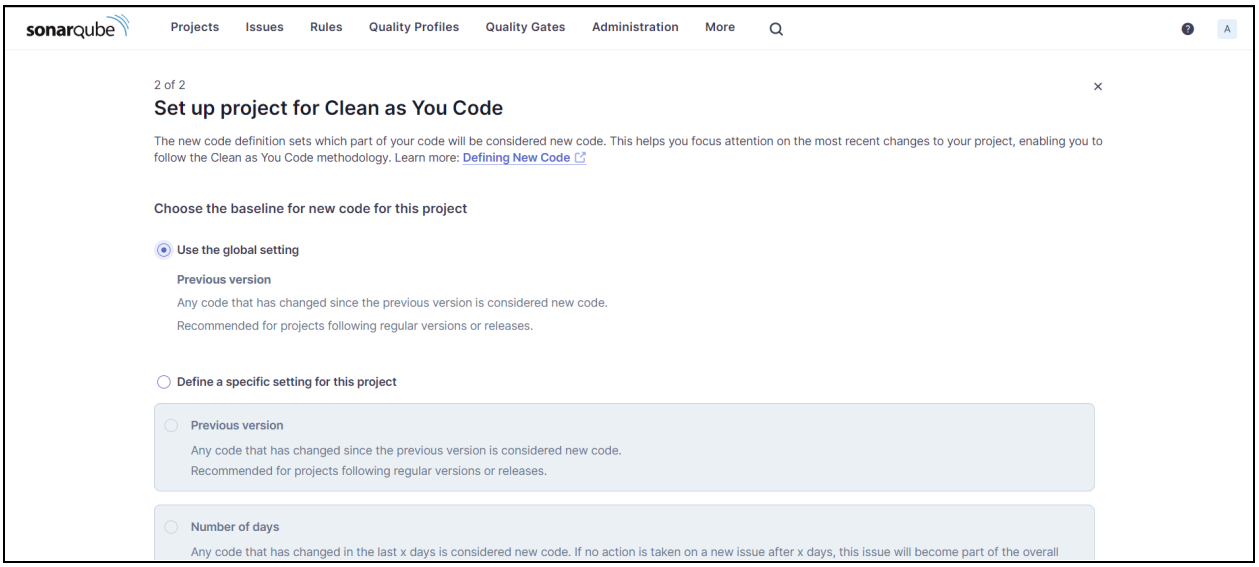
3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



4. Login to SonarQube using username admin and password admin.

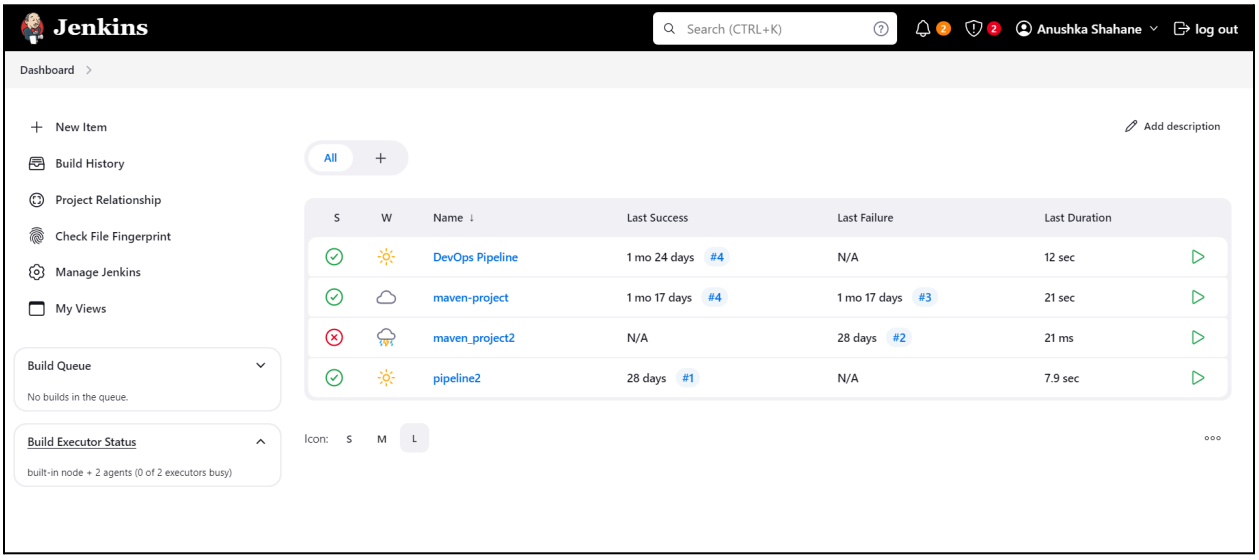


5. Create a manual project in SonarQube with the name sonarqube

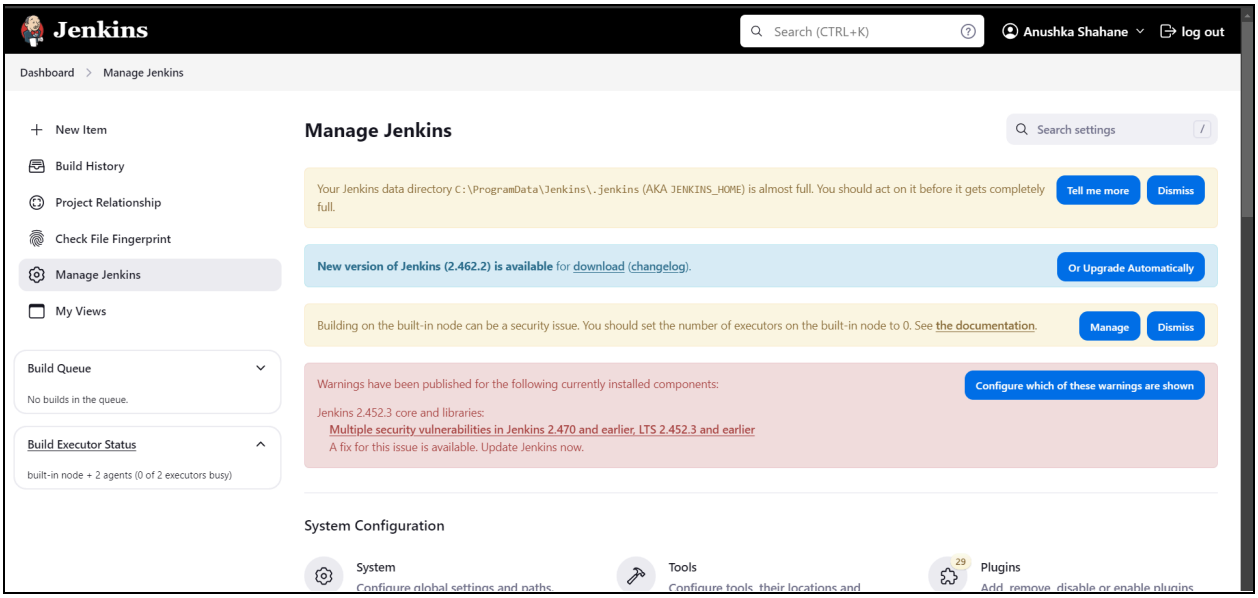


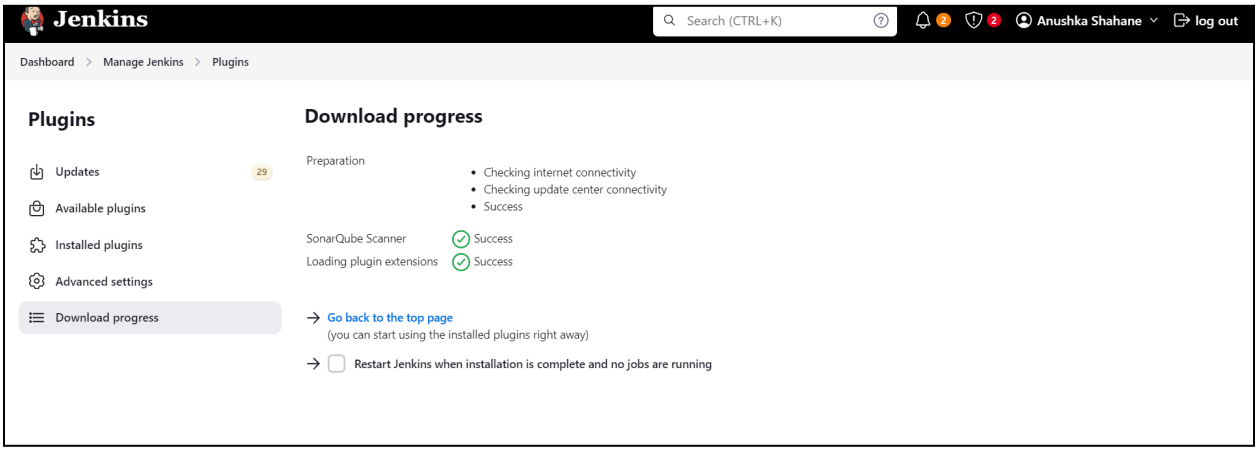
Setup the project and come back to Jenkins Dashboard.

Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it

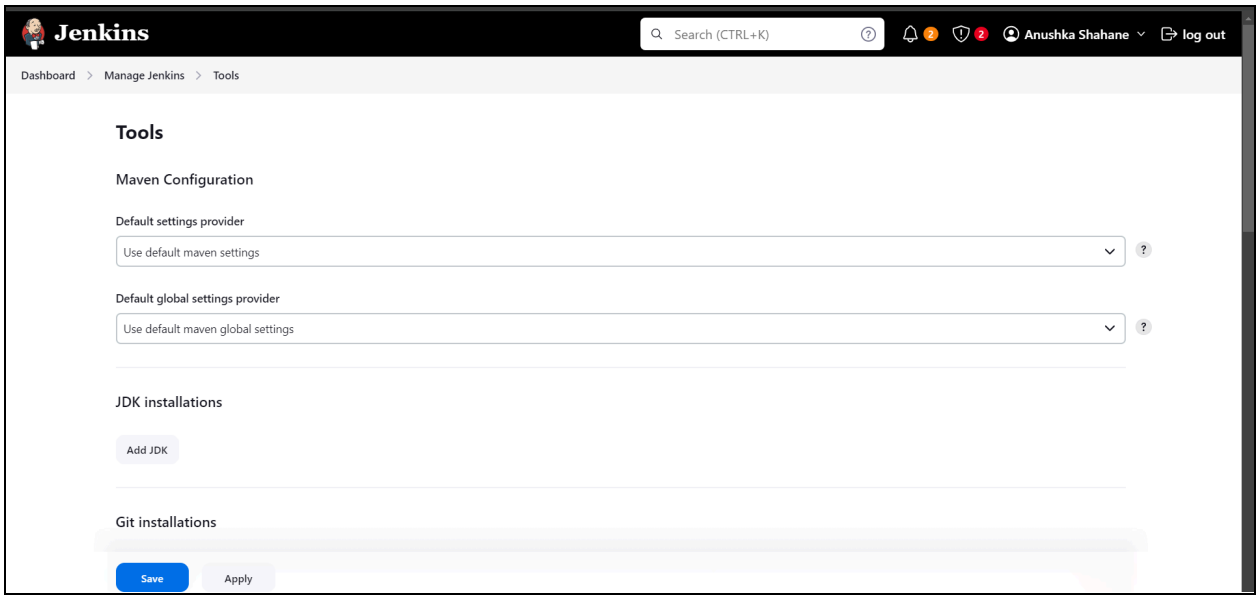


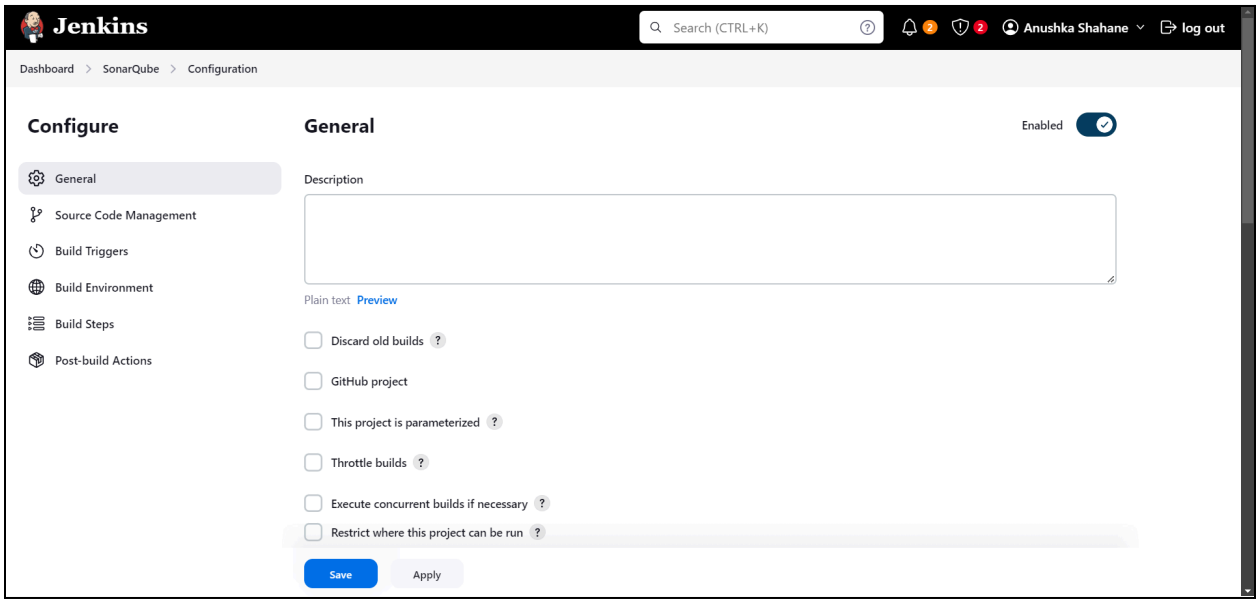
7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.





8. After the configuration, create a New Item in Jenkins, choose a freestyle project.



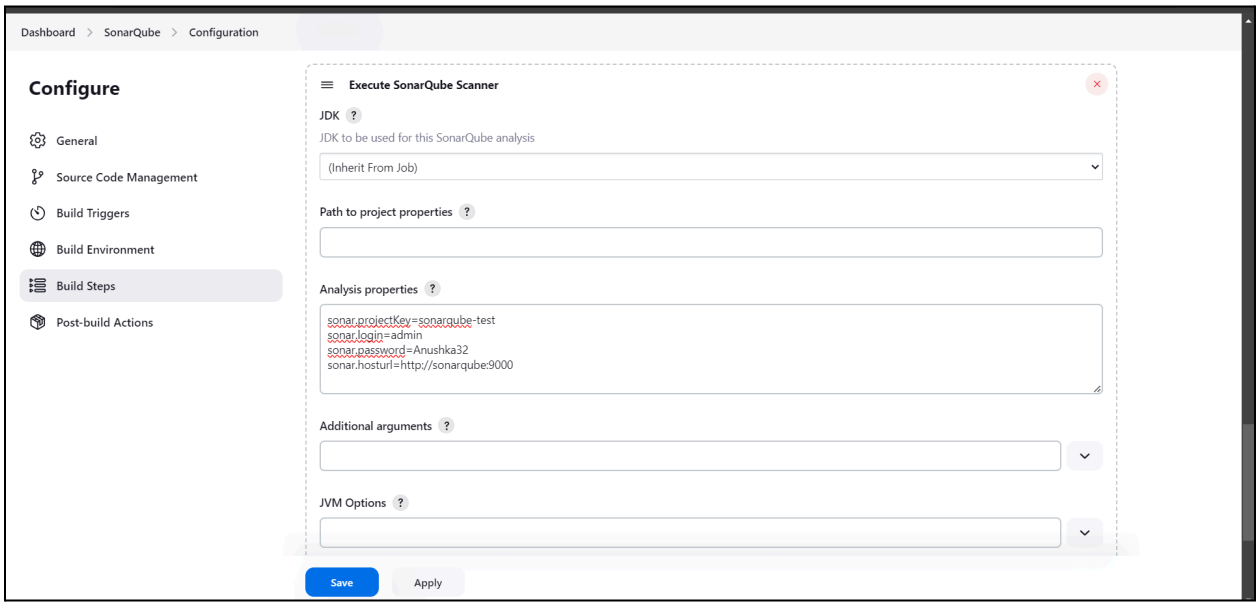


9. Choose this GitHub repository in Source Code

Management.

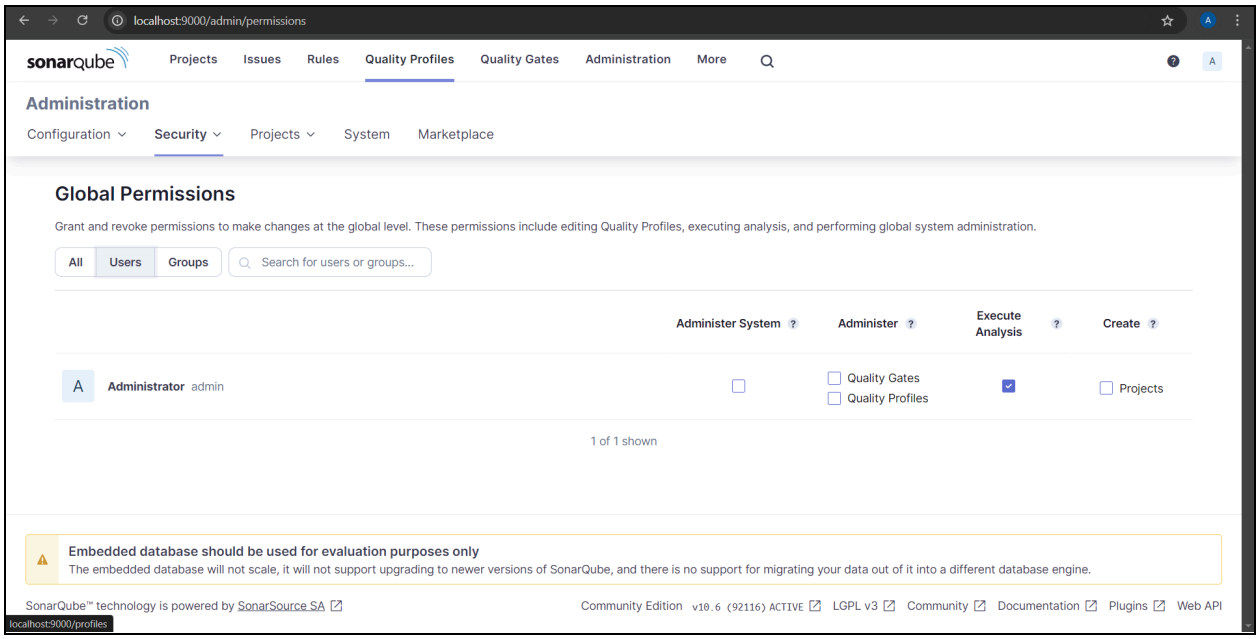
https://github.com/shazforiot/MSBuild_firstproject.git

It is a sample hello-world project with no vulnerabilities and issues,

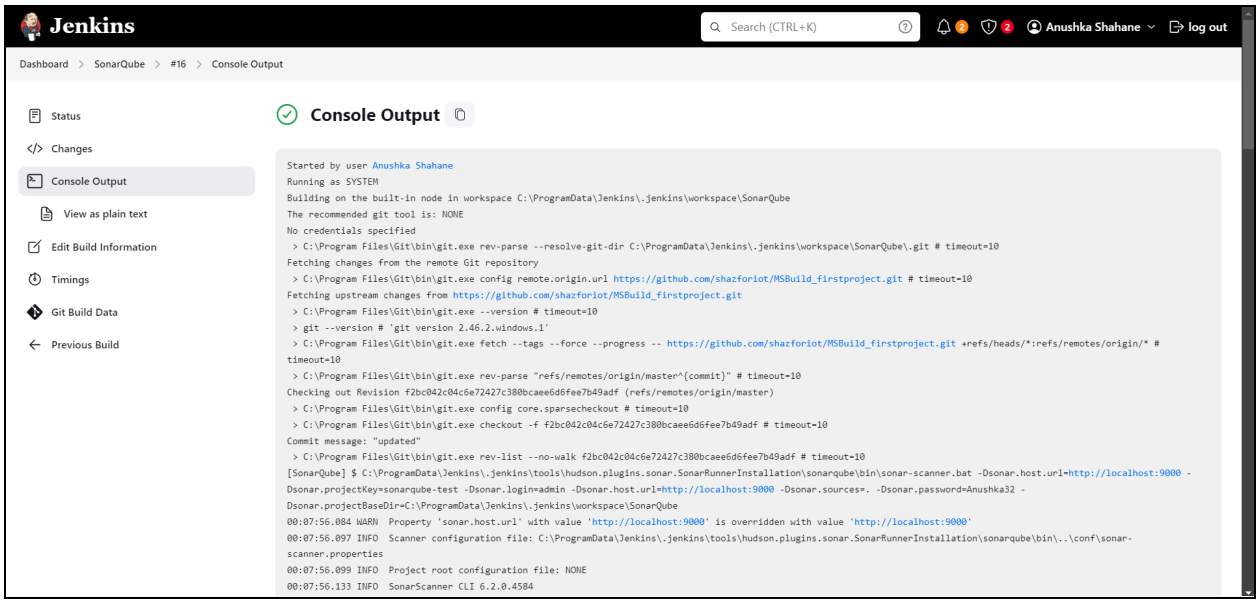


10. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.
11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.

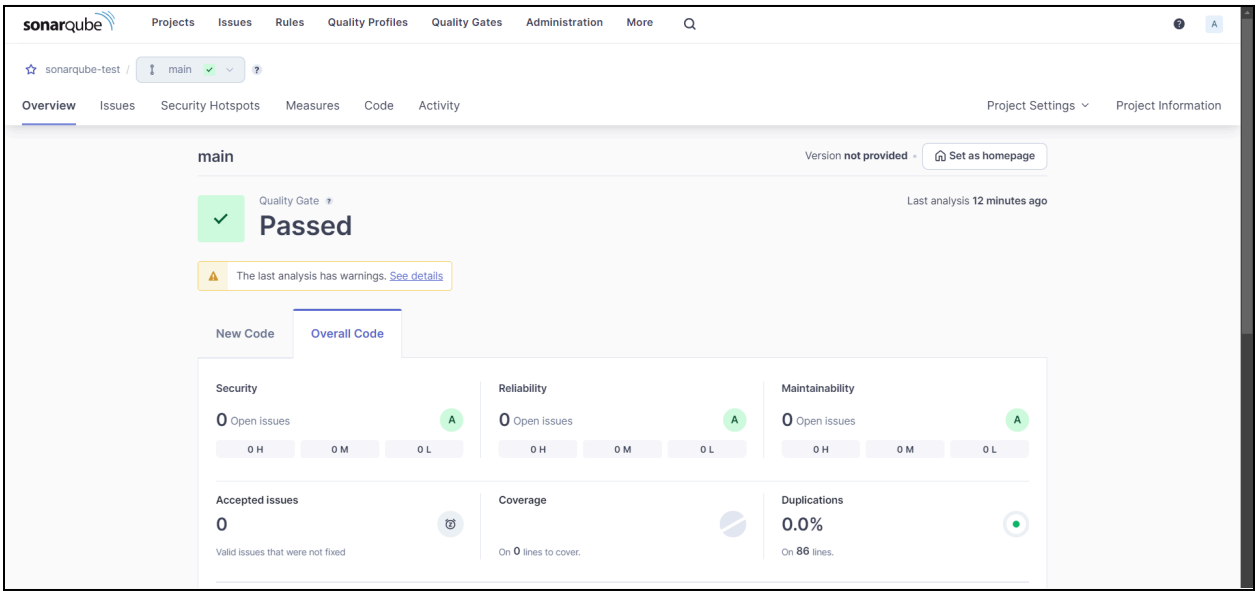
Anushka Shahane D15A 55



Check the console output.



13. Once the build is complete, check the project in SonarQube.



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

Anushka Shahane D15A 55

EXPERIMENT 8 - ADVANCE DEVOPS

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 /

Theory:

What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools pointout the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for anorganization to
find the resources to perform code reviews on even a fraction of its applications. A key strength
of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster
than manual secure code reviews performed by humans. These tools can scan millionsof lines of
code in a matter of minutes. SAST tools automatically identify criticalvulnerabilities—such as
buffer overflows, SQL injection, cross-site scripting, and others—with high confidence.What is a CI/CD Pipeline?

.

What is SonarQube

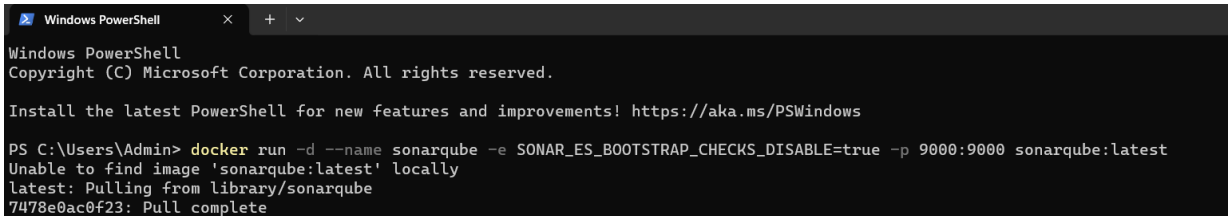
SonarQube is an open-source platform developed by SonarSource for continuous inspection of
code quality. Sonar does static code analysis, which provides a detailed report of bugs, code
smells, vulnerabilities, code duplications.

It supports 25+ major programming languages through built-in rulesets and can also be extended
with various plugins.

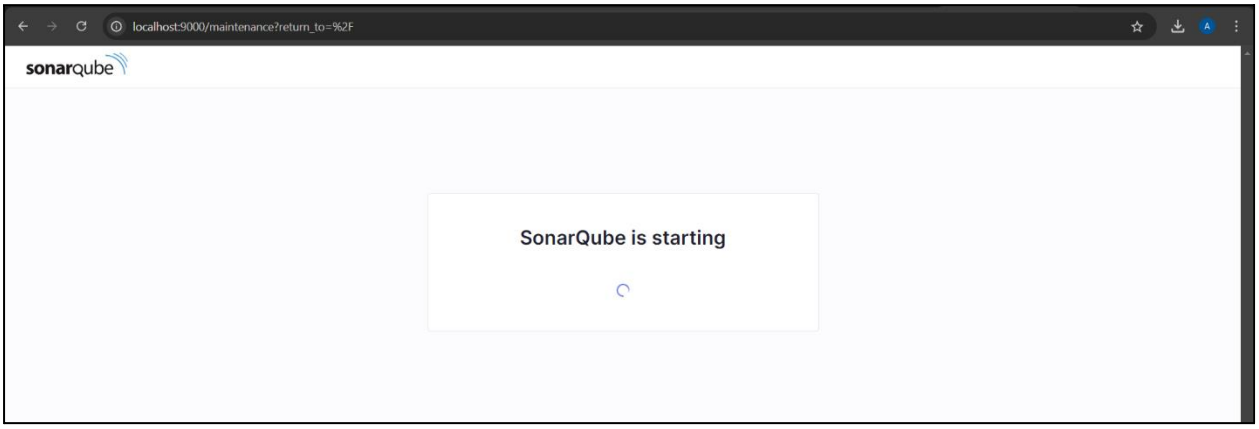
Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.

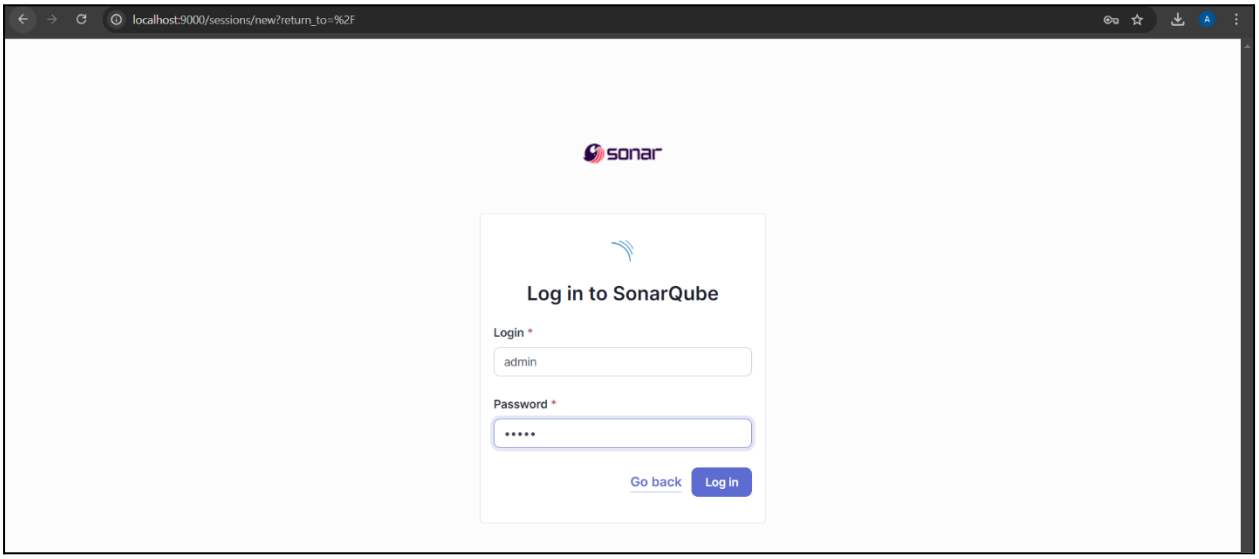
2. Run SonarQube in a Docker container using this command -



3. Once the container is up and running, you can check the status of SonarQube at localhost
port 9000.

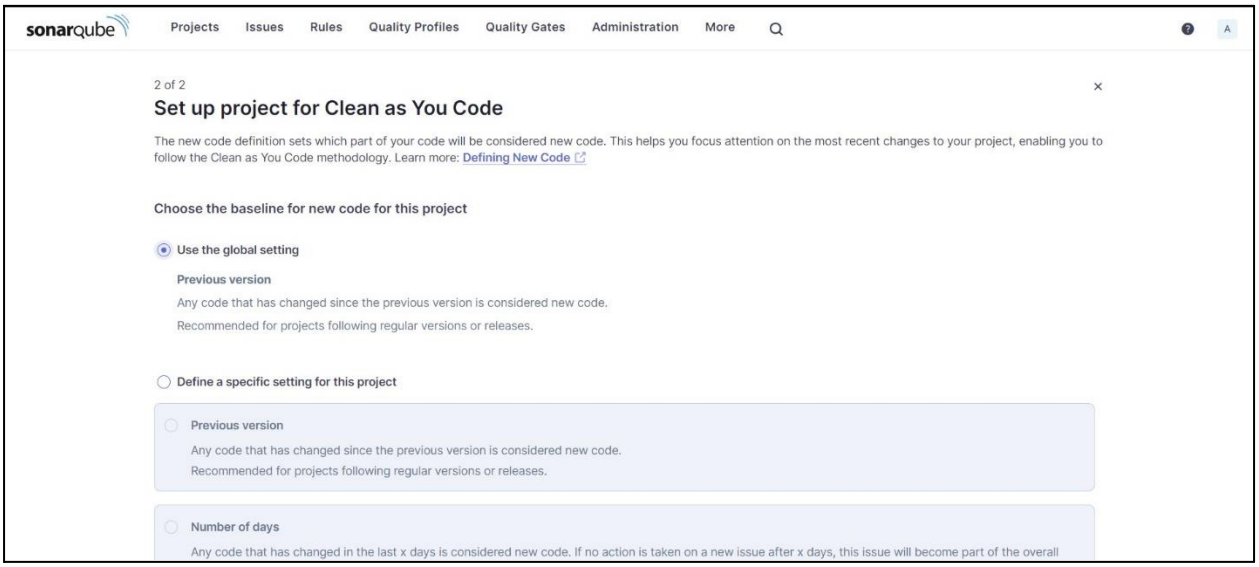


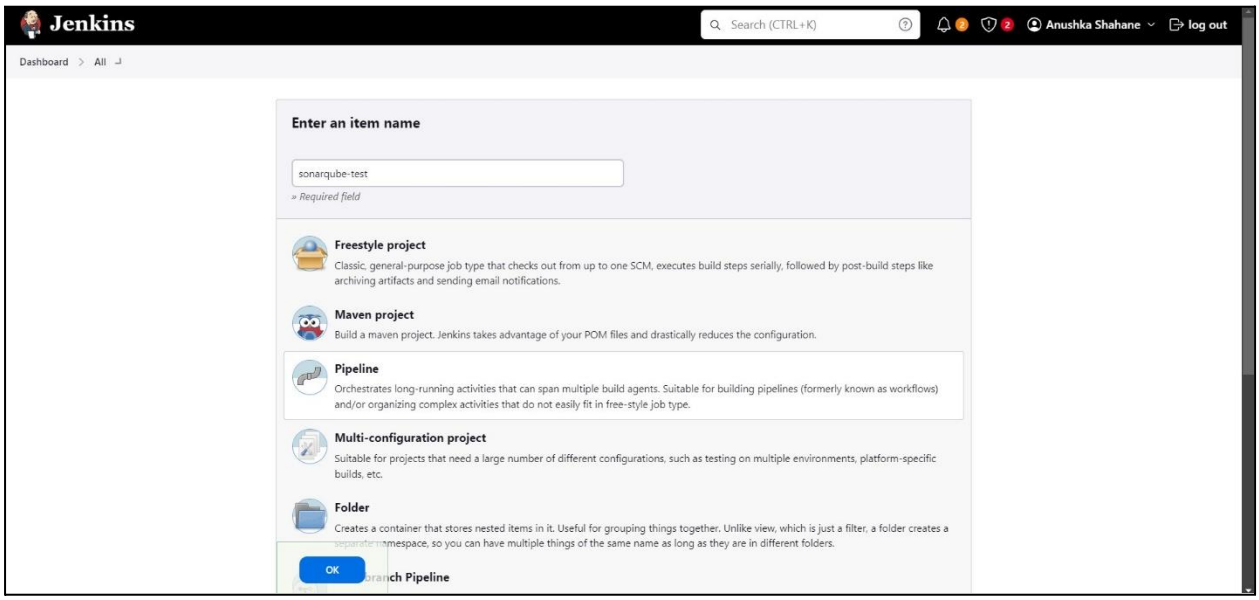
4. Login to SonarQube using username admin and password admin.



5. Create a manual project in SonarQube with the name sonarqube-test

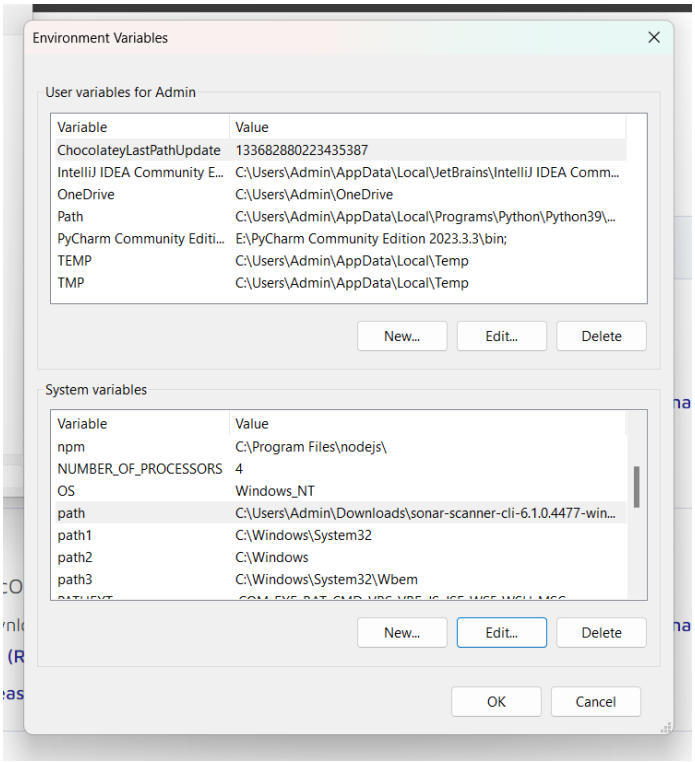
Anushka Shahane D15A 55





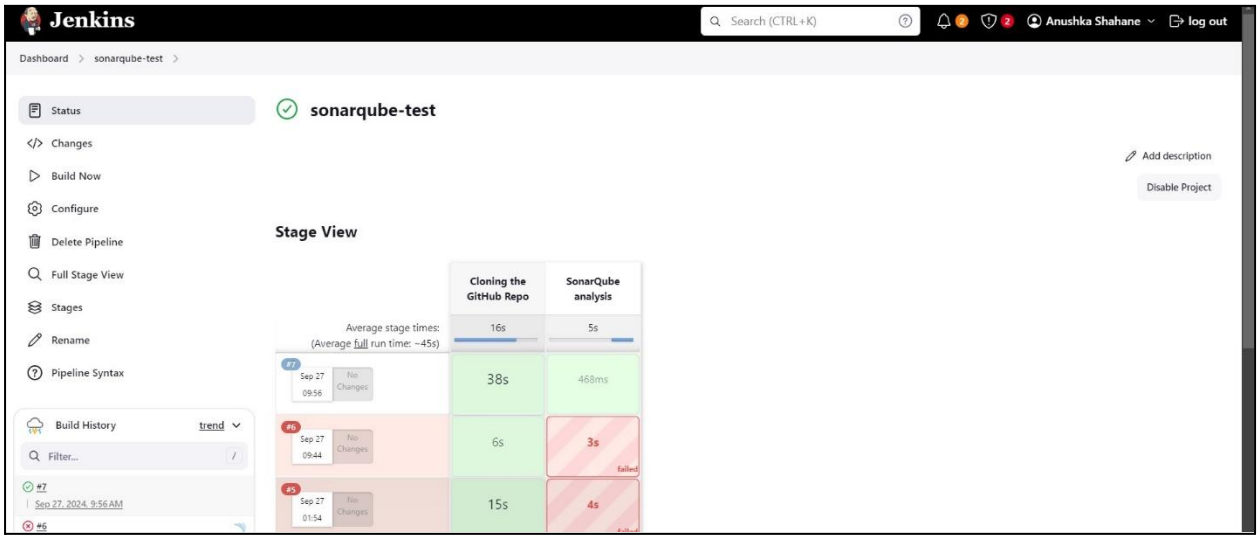
Step 6 : Go to download sonarscanner to download sonar scanner

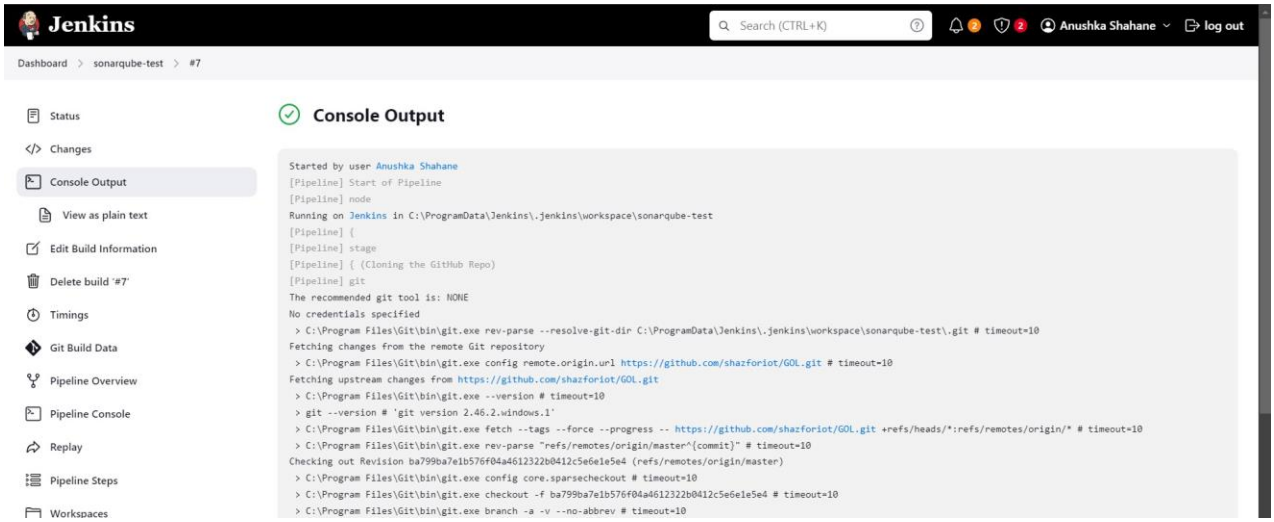
Step 7: After the download is complete, extract the file and copy the path to bin folder
Go to environment variables, system variables and click on path
Add a new path, paste the path copied earlier.



Step 8 : Save the pipeline and build it.

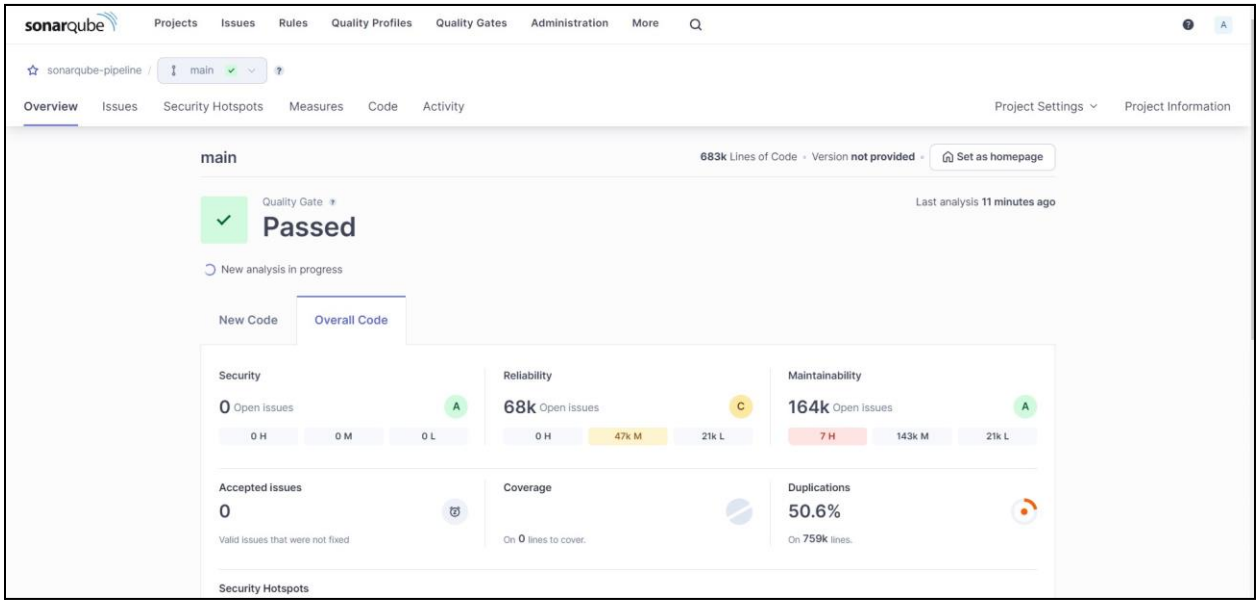
Output :



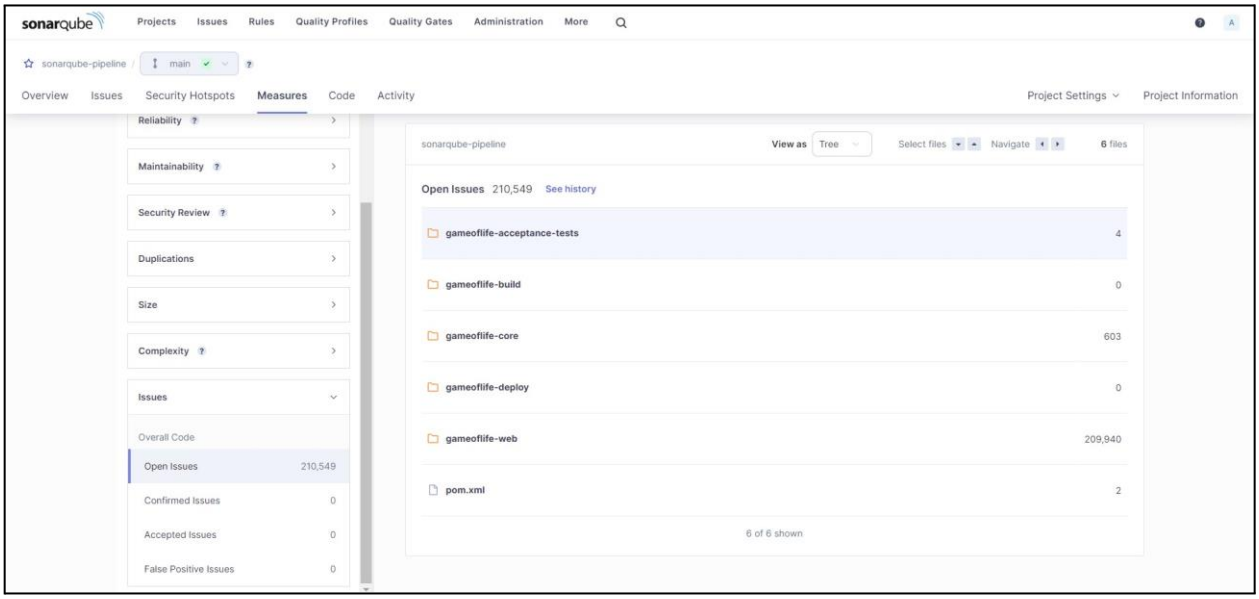



```
20:50:01.832 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-pipeline
20:50:01.832 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
20:50:01.832 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=159a9d05-1f5f-4e17-bd27-3643a32a836a
20:50:12.108 INFO  Analysis total time: 7:37.235 s
20:50:12.110 INFO  SonarScanner Engine completed successfully
20:50:12.849 INFO  EXECUTION SUCCESS
20:50:12.851 INFO  Total time: 7:44.878s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

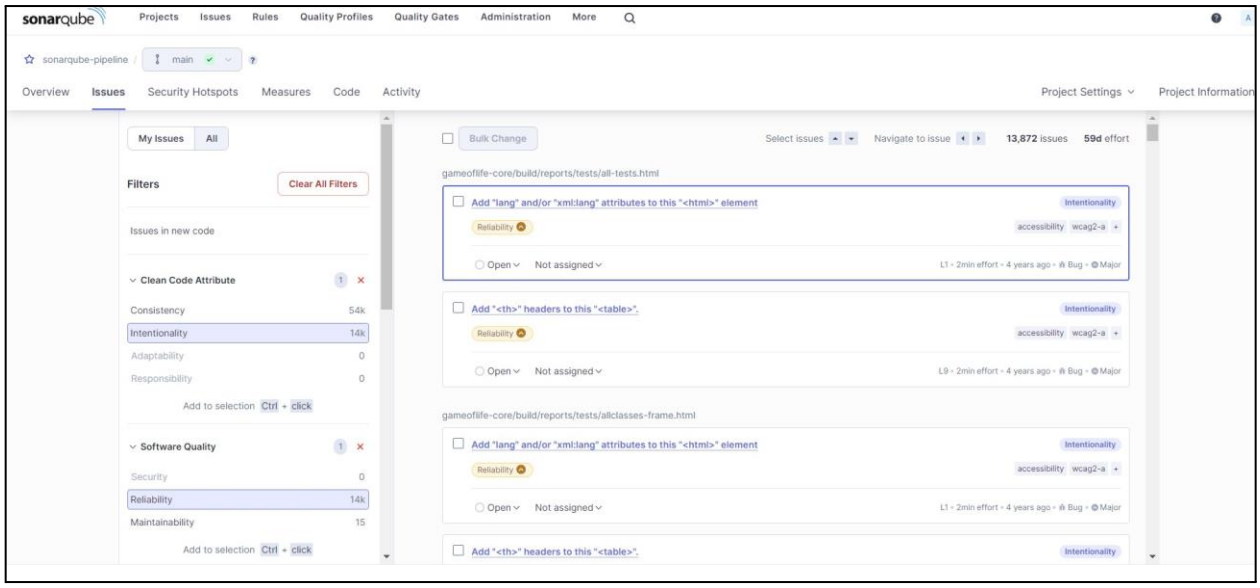
Step 9 : Check the project in SonarQube

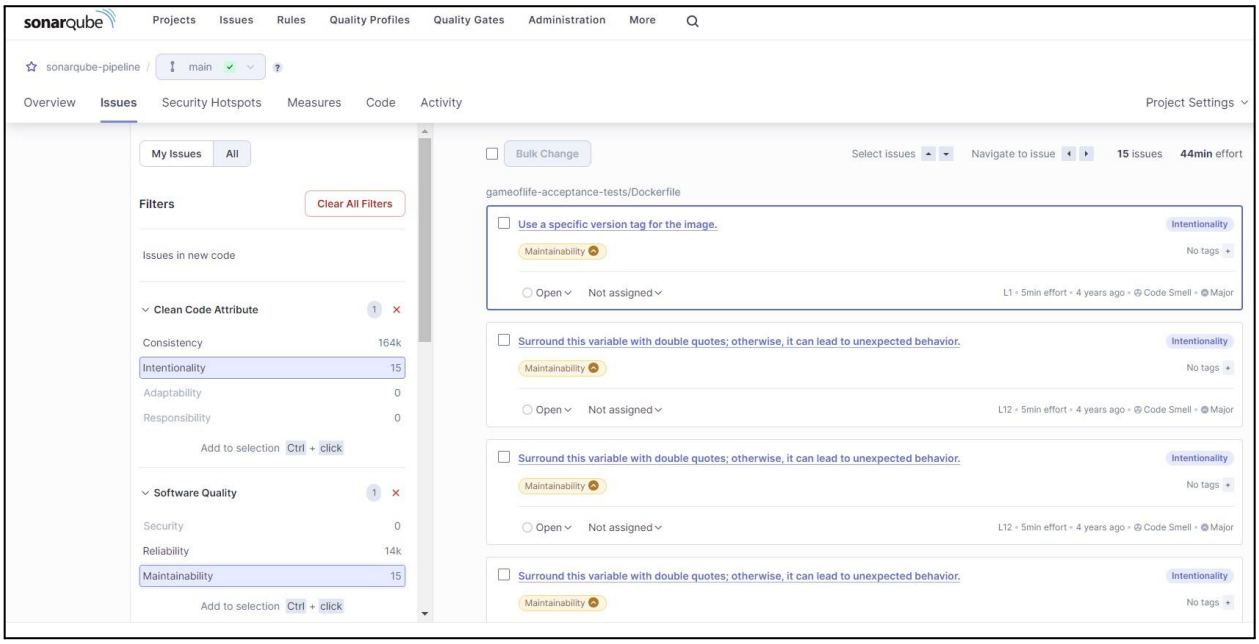


Under different tabs, check all different issues with the code.

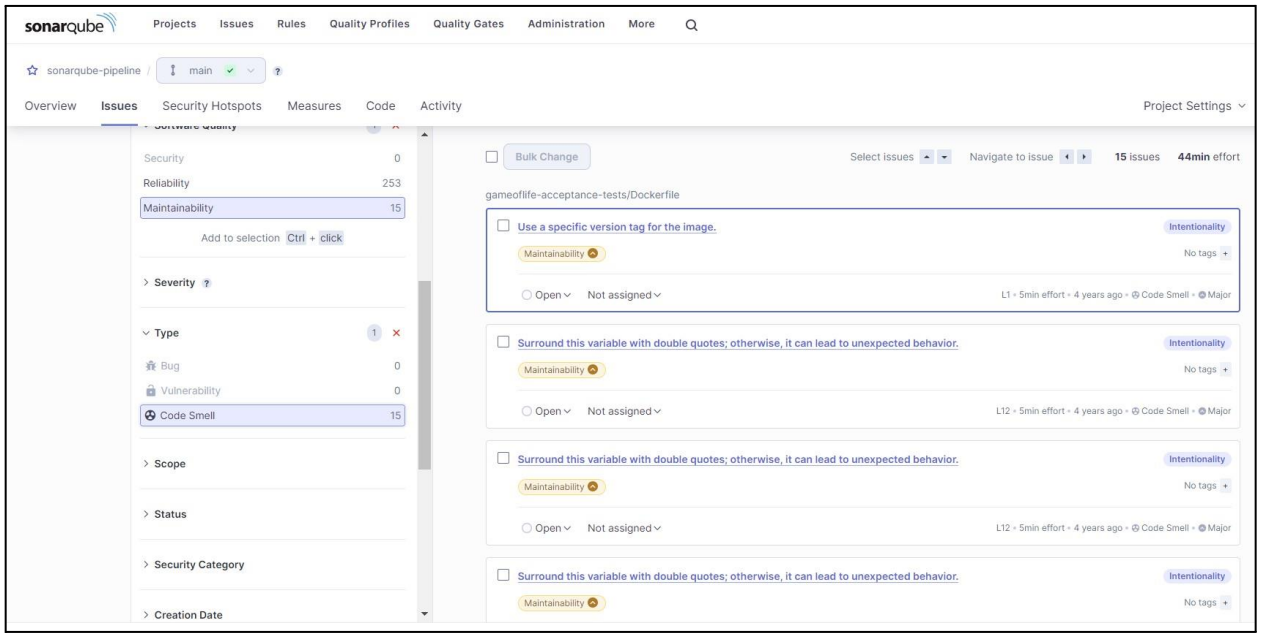


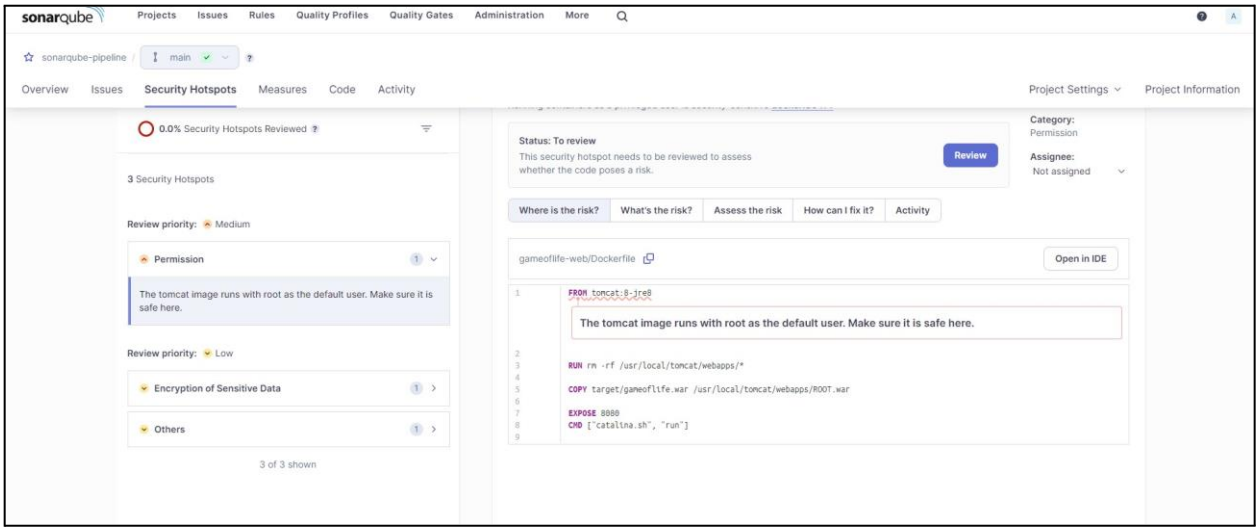
Anushka Shahane D15A 55



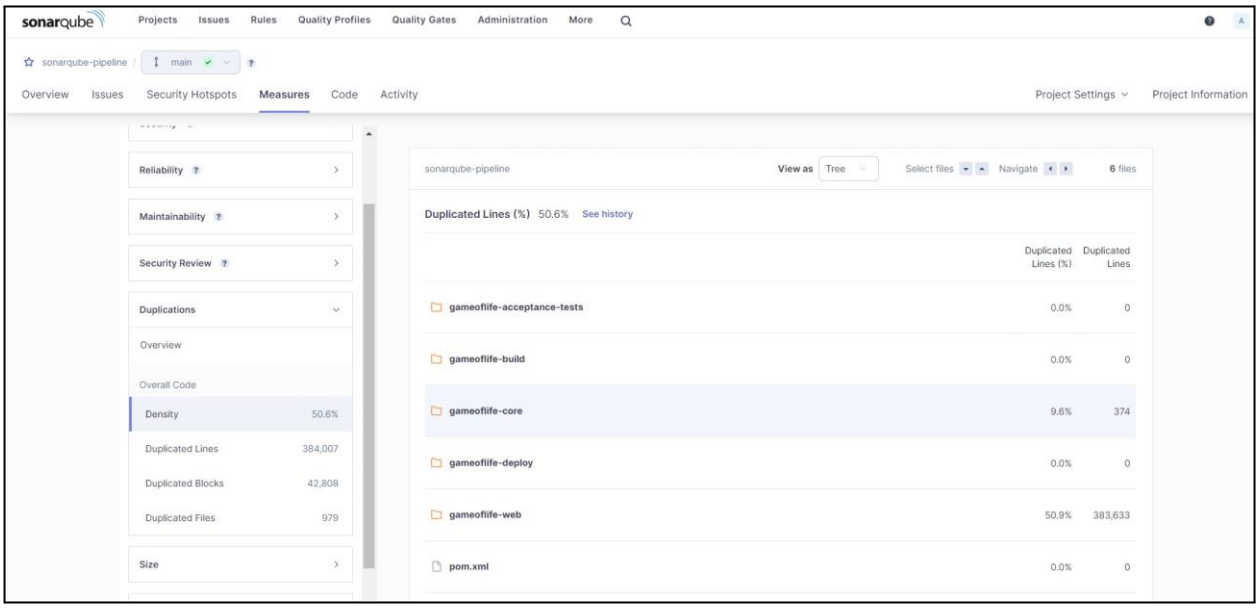


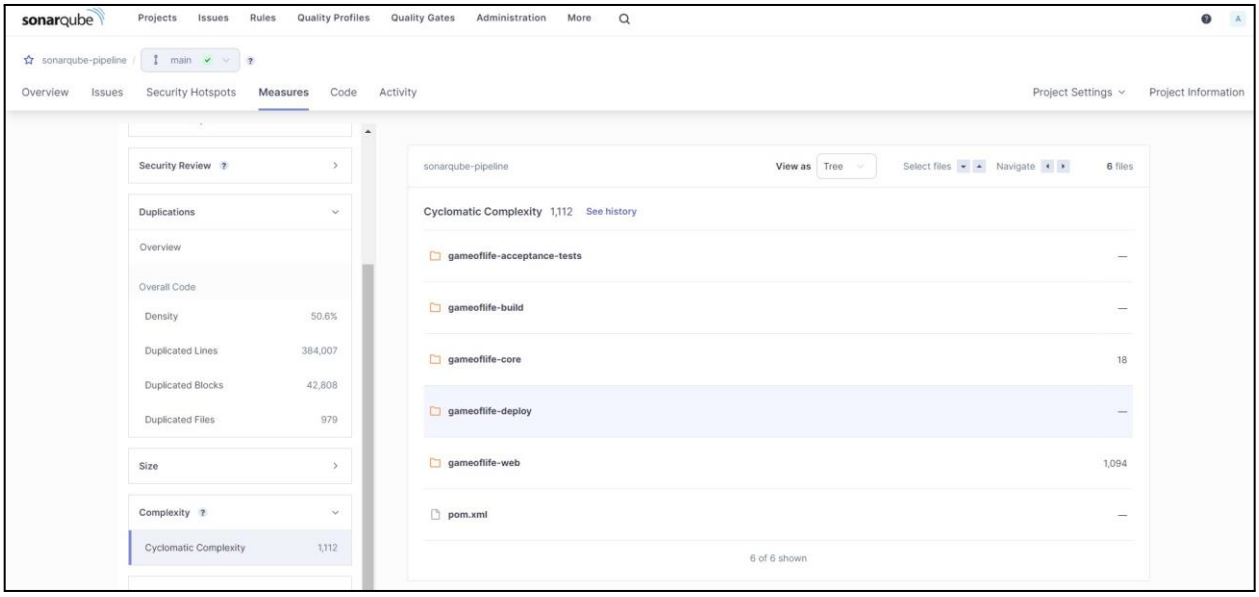
Anushka Shahane D15A 55





Anushka Shahane D15A 55





Anushka Shahane D15A 55

Advance devops Experiment No. 9

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

Theory:
What is Nagios?
Nagios is an open-source software for continuous monitoring of systems, networks, and infrastructures. It runs plugins stored on a server that is connected with a host or another server on your network or the Internet. In case of any failure, Nagios alerts about the issues so that the technical team can perform the recovery process immediately.
Nagios is used for continuous monitoring of systems, applications, service and business processes in a DevOps culture.

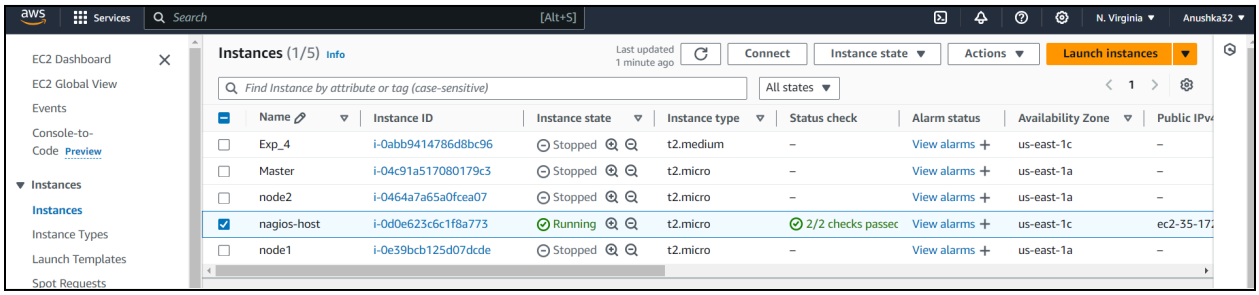
Features of Nagios

Following are the important features of Nagios monitoring tool:
● Relatively scalable, Manageable, and Secure
● Good log and database system
● Informative and attractive web interfaces
● Automatically send alerts if condition changes
● If the services are running fine, then there is no need to do check that host is an alive
● Helps you to detect network errors or server crashes
● You can troubleshoot the performance issues of the server.
● The issues, if any, can be fixed automatically as they are identified during the monitoring process

● You can monitor the entire business process and IT infrastructure with a single pass
● The product's architecture is easy to write new plugins in the language of your choice
● Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
● Utilizes topology to determine dependencies
● Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
● Helps you to define network host hierarchy using parent hosts
● Ability to define event handlers that runs during service or host events for proactive problem resolution
● Support for implementing redundant monitoring hosts

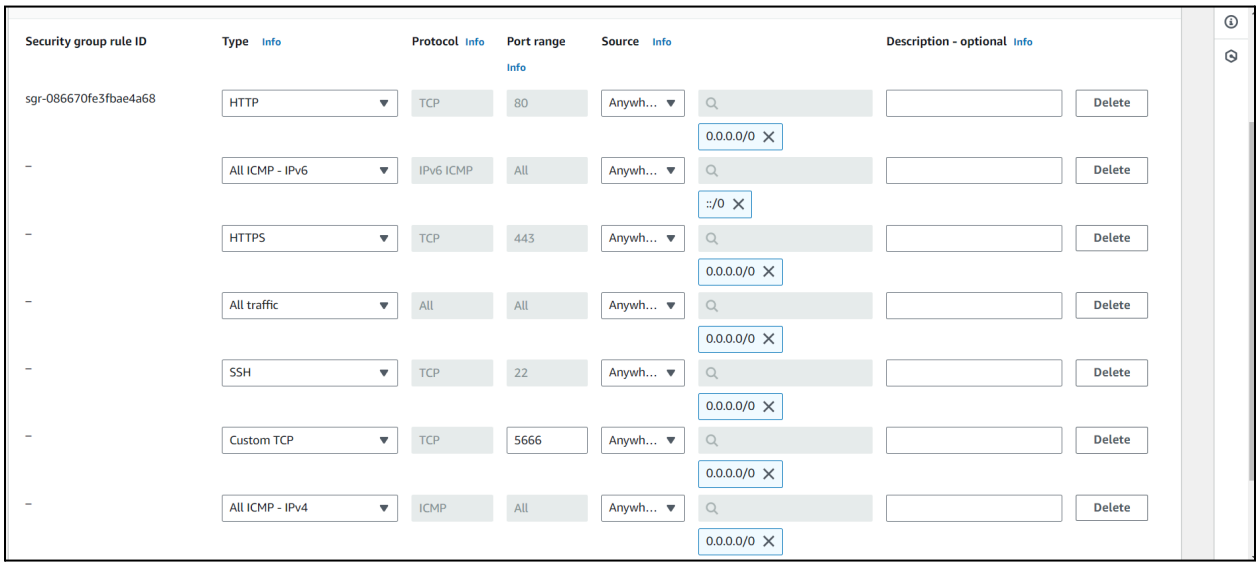
1. Create an Amazon Linux EC2 Instance

- Name it nagios-host.



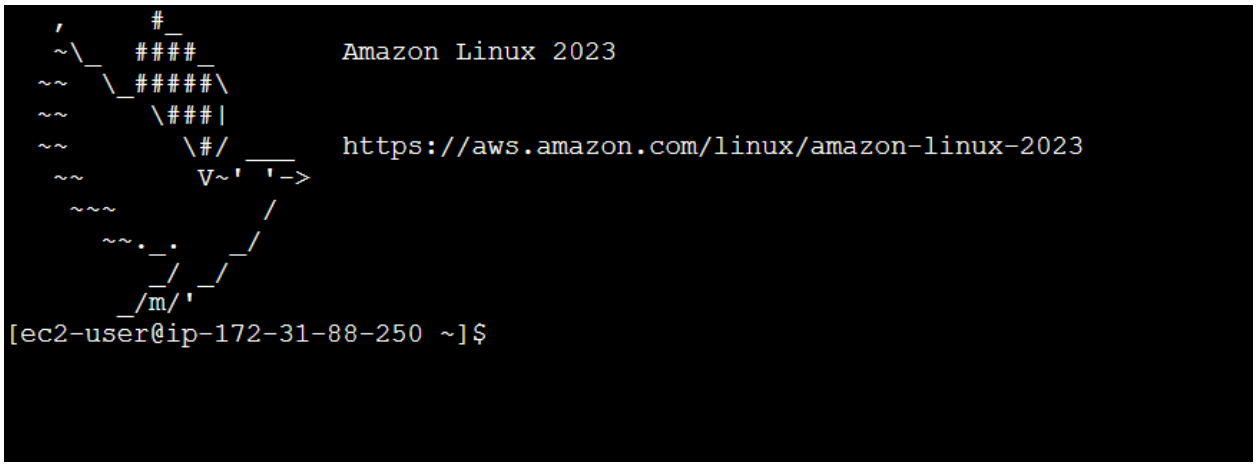
2. Configure Security Group

- 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 -

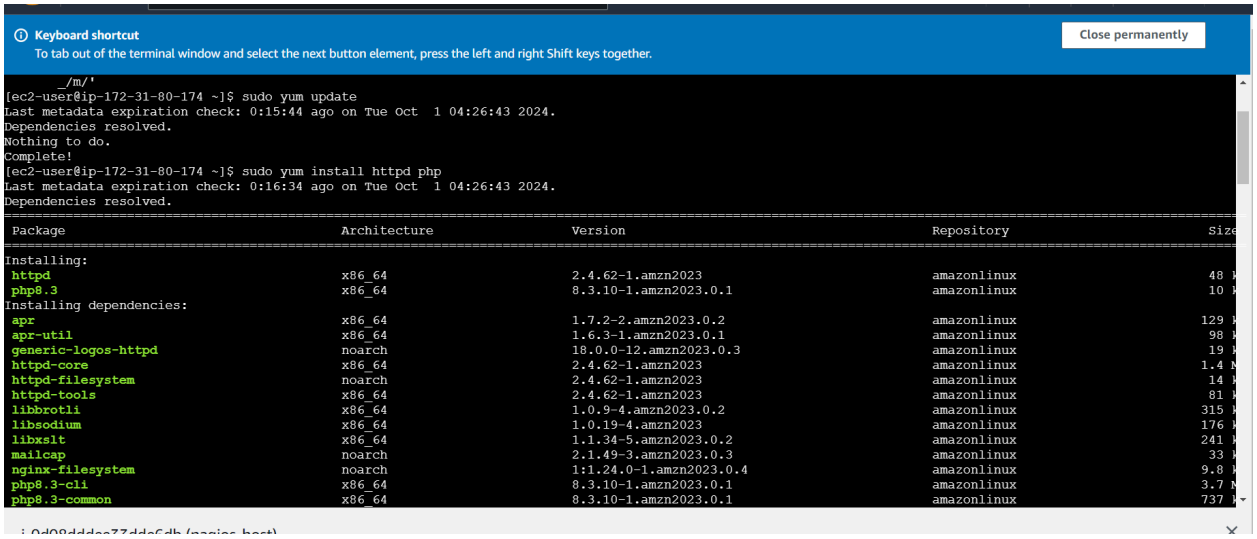
sudo yum update

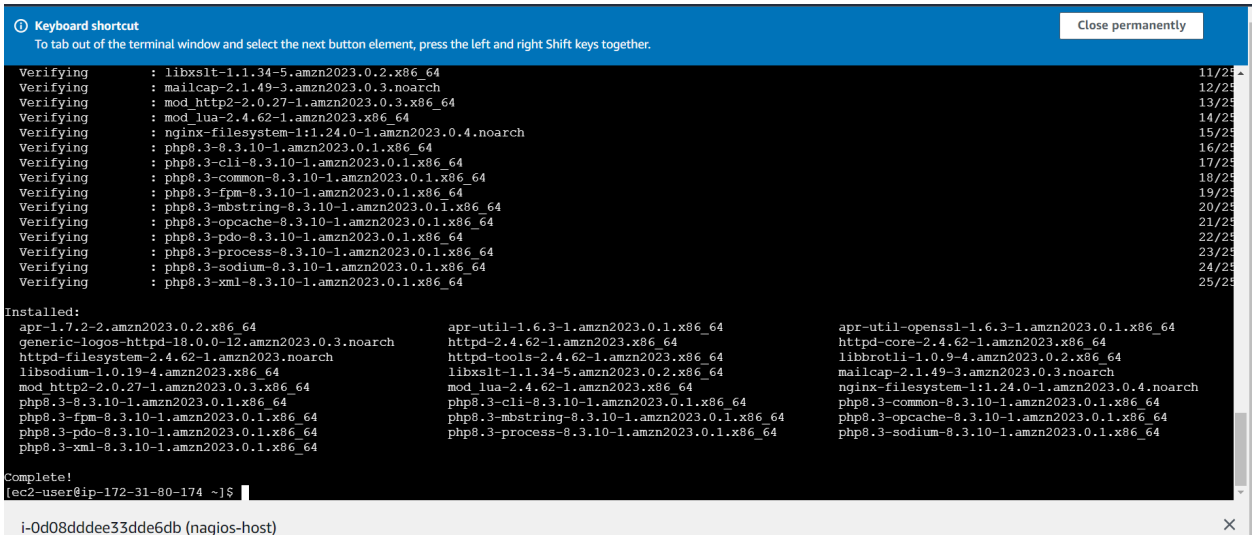
sudo yum install httpd php

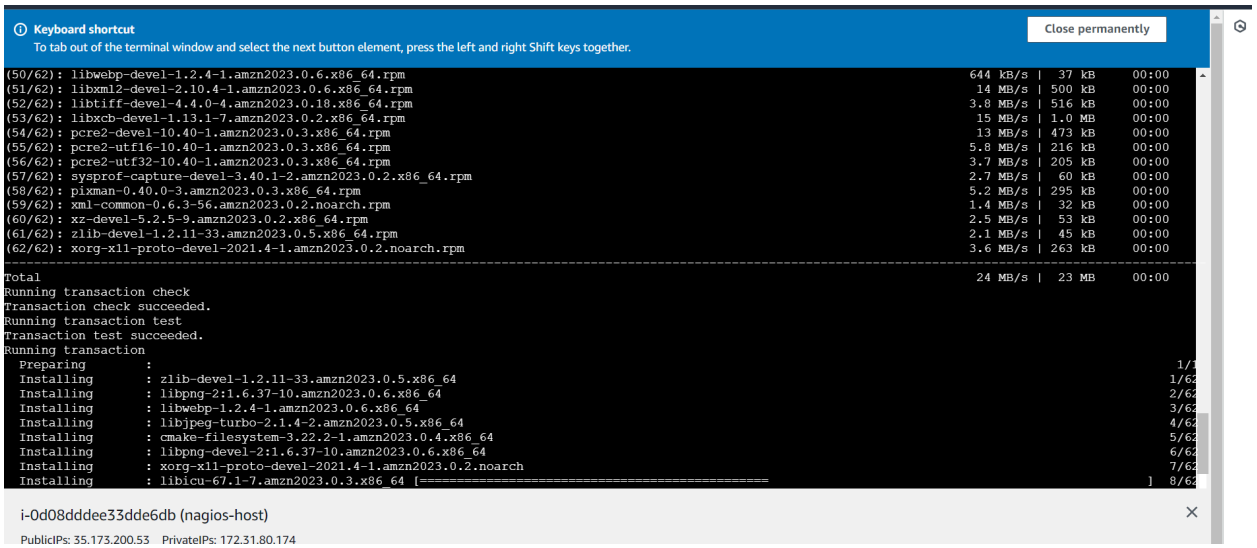
sudo yum install gcc glibc glibc-common

sudo yum install gd gd-devel

Anushka Shahane D15A 55







5. Create a New Nagios User

Commands -

sudo adduser -m nagios

sudo passwd nagios

```
Last login: Mon Sep 30 17:45:34 2024 from 18.206.107.27
[ec2-user@ip-172-31-88-250 ~]$ sudo adduser -m nagios
adduser: user 'nagios' already exists
[ec2-user@ip-172-31-88-250 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-88-250 ~]$
```

6. Create a New User Group

Commands -

sudo groupadd nagcmd

```
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-88-250 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-88-250 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-88-250 ~]$ sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-88-250 ~]$
```

7. Add Users to the Group

Commands -

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-80-22 ~]$ sudo usermod -aG nagcmd nagios
sudo usermod -aG nagcmd apache
```

8. Create a Directory for Nagios Downloads

Commands -

mkdir ~/downloads

cd ~/downloads

```
[ec2-user@ip-172-31-88-250 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-88-250 ~]$ cd ~/downloads
```

9. Download Nagios and Plugins Source Files

Commands -

wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz

wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz

```
[ec2-user@ip-172-31-88-250 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
--2024-09-30 18:02:53--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11333414 (11M) [application/x-gzip]
Saving to: 'nagios-4.4.6.tar.gz'

nagios-4.4.6.tar.gz                 100%[=============================================================================>]  10.81M  12.3MB/s    in 0.9s

2024-09-30 18:02:54 (12.3 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333414/11333414]

[ec2-user@ip-172-31-88-250 downloads]$
```

```
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2782610 (2.7M) [application/x-gzip]
Saving to: 'nagios-plugins-2.3.3.tar.gz.3'

nagios-plugins-2.3.3.tar.gz.3          100%[=====================================================================>]   2.65M  7.09MB/s    in 0.4s

2024-09-30 18:03:39 (7.09 MB/s) - 'nagios-plugins-2.3.3.tar.gz.3' saved [2782610/2782610]

[ec2-user@ip-172-31-88-250 downloads]$
```

```
Resolving prdownloads.sourceforge.net (prdownloads.sourceforge.net)... 204.68.111.105
Connecting to prdownloads.sourceforge.net (prdownloads.sourceforge.net)|204.68.111.105|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz [following]
--2024-10-01 04:54:16--  http://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz
Resolving downloads.sourceforge.net (downloads.sourceforge.net)... 204.68.111.105
Reusing existing connection to prdownloads.sourceforge.net:80.
HTTP request sent, awaiting response... 302 Found
Location: http://netactuate.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1 [following]
--2024-10-01 04:54:16--  http://netactuate.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1
Resolving netactuate.dl.sourceforge.net (netactuate.dl.sourceforge.net)... 104.225.3.66
Connecting to netactuate.dl.sourceforge.net (netactuate.dl.sourceforge.net)|104.225.3.66|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1805059 (1.7M) [application/x-gzip]
Saving to: 'nagios-4.0.8.tar.gz'

nagios-4.0.8.tar.gz                    100%[=====================================================================>]   1.72M  --.-KB/s    in 0.09s

2024-10-01 04:54:17 (18.6 MB/s) - 'nagios-4.0.8.tar.gz' saved [1805059/1805059]

[ec2-user@ip-172-31-80-174 downloads]$
```

i-0d08dddee33dde6db (nagios-host)

PublicIPs: 35.173.200.53 PrivateIPs: 172.31.80.174

```
nagios-4.0.8.tar.gz                    100%[=====================================================================>]   1.72M  --.-KB/s    in 0.09s

2024-10-01 04:54:17 (18.6 MB/s) - 'nagios-4.0.8.tar.gz' saved [1805059/1805059]

[ec2-user@ip-172-31-80-174 downloads]$ wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
--2024-10-01 04:55:13--  http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2659772 (2.5M) [application/x-gzip]
Saving to: 'nagios-plugins-2.0.3.tar.gz'

nagios-plugins-2.0.3.tar.gz            100%[=====================================================================>]   2.54M  7.82MB/s    in 0.3s

2024-10-01 04:55:13 (7.82 MB/s) - 'nagios-plugins-2.0.3.tar.gz' saved [2659772/2659772]

[ec2-user@ip-172-31-80-174 downloads]$
```

10. Extract the Nagios Source File

Commands -

tar zxvf nagios-4.4.6.tar.gz

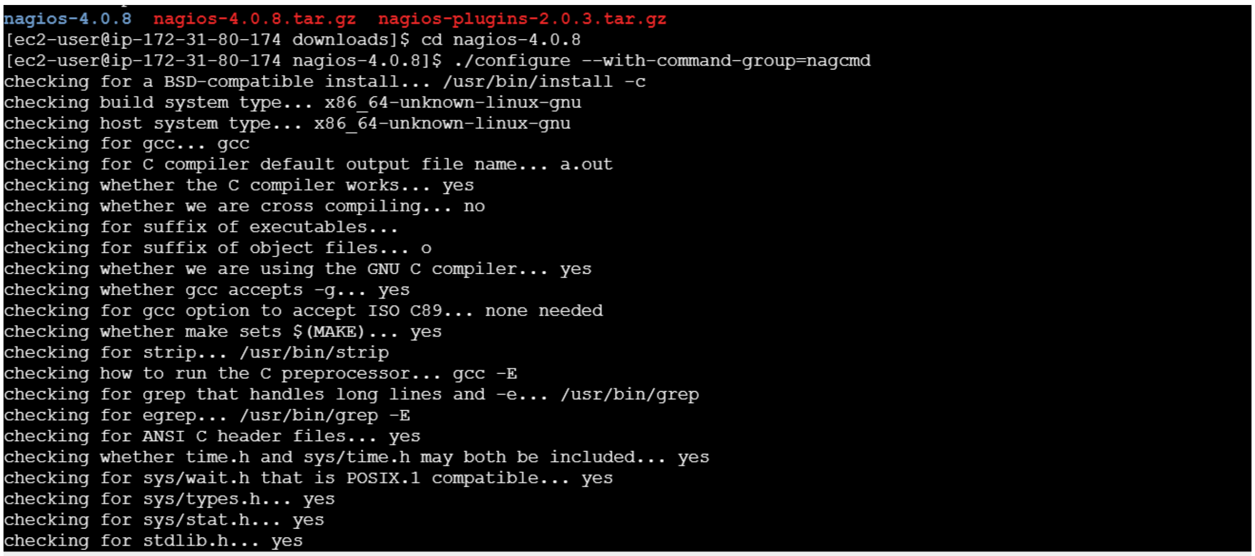
cd nagios-4.4.6

```
[ec2-user@ip-172-31-80-174 downloads]$ tar zxvf nagios-4.0.8.tar.gz
nagios-4.0.8/
nagios-4.0.8/.gitignore
nagios-4.0.8/Changelog
nagios-4.0.8/INSTALLING
nagios-4.0.8/LEGAL
nagios-4.0.8/LICENSE
nagios-4.0.8/Makefile.in
nagios-4.0.8/README
nagios-4.0.8/README.asciidoc
nagios-4.0.8/THANKS
nagios-4.0.8/UPGRADING
nagios-4.0.8/base/
nagios-4.0.8/base/.gitignore
nagios-4.0.8/base/Makefile.in
nagios-4.0.8/base/broker.c
nagios-4.0.8/base/checks.c
nagios-4.0.8/base/commands.c
nagios-4.0.8/base/config.c
nagios-4.0.8/base/events.c
nagios-4.0.8/base/flapping.c
nagios-4.0.8/base/logging.c
```

11. Run the Configuration Script

Commands -

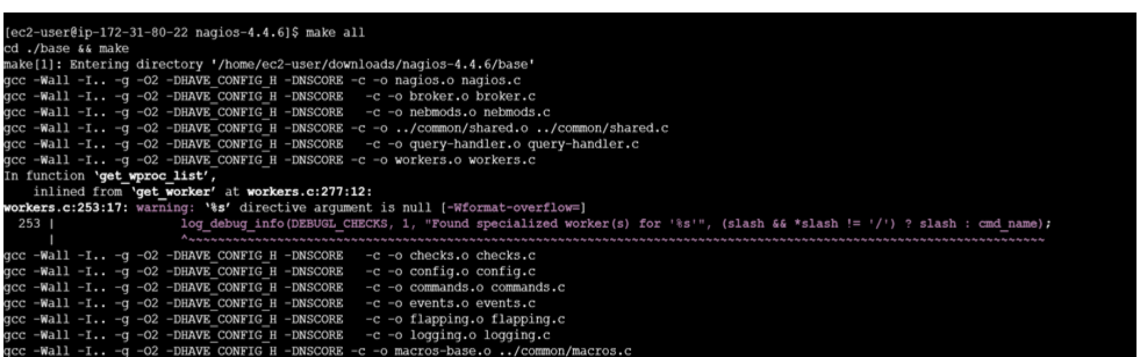
./configure --with-command-group=nagcmd



12. Compile the Source Code

Commands -

make all



```
done
/usr/bin/install: cannot stat '*.cgi': No such file or directory
make[2]: *** [Makefile:205: install-basic] Error 1
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.0.8/cgi'
make[1]: *** [Makefile:197: install] Error 2
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.0.8/cgi'
make: *** [Makefile:235: install] Error 2
[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /etc/rc.d/init.d
/usr/bin/install -c -m 755 -o root -g root daemon-init /etc/rc.d/init.d/nagios

*** Init script installed ***

[ec2-user@ip-172-31-80-174 nagios-4.0.8]$
```

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

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

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

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

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

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

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

Enjoy.

[ec2-user@ip-172-31-88-250 nagios-4.4.6]$
```

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

Commands -

./sudo make install

sudo make install-init

sudo make install-config

sudo make install-commandmode

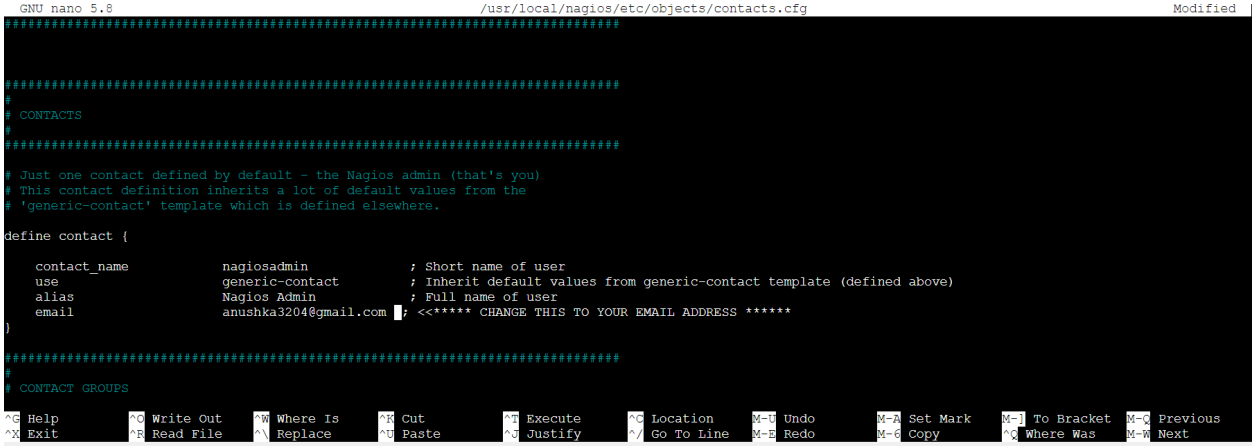
```
*** External command directory configured ***

[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf

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

[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ cd ~/downloads
[ec2-user@ip-172-31-80-174 downloads]$ tar zxvf nagios-plugins-2.0.3.tar.gz
nagios-plugins-2.0.3/
nagios-plugins-2.0.3/perlmods/
nagios-plugins-2.0.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.0.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.0.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.in
nagios-plugins-2.0.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.am
nagios-plugins-2.0.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.0.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.0.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.0.3/perlmods/Class-Accessor-0.34.tar.gz
```

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

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

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

[ec2-user@ip-172-31-88-250 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-88-250 nagios-4.4.6]$
```

```
nagios-plugins-2.3.3/pkg/fedora/requires
nagios-plugins-2.3.3/pkg/solaris/
nagios-plugins-2.3.3/pkg/solaris/preinstall
nagios-plugins-2.3.3/pkg/solaris/solpkg
nagios-plugins-2.3.3/pkg/solaris/pkginfo.in
nagios-plugins-2.3.3/pkg/solaris/pkginfo
nagios-plugins-2.3.3/pkg/redhat/
nagios-plugins-2.3.3/pkg/redhat/requires
[ec2-user@ip-172-31-88-250 downloads]$ cd nagios-plugins-2.3.3
[ec2-user@ip-172-31-88-250 nagios-plugins-2.3.3]$
```

```
make  install-exec-hook
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
cd /usr/local/nagios/libexec && \
for i in check_ftp check_imap check_nntp check_pop check_udp check_clamd  ; do rm -f $i; ln -s check_tcp $i ; done ;\
if [ -x check_ldap ] ; then rm -f check_ldaps ; ln -s check_ldap check_ldaps ; fi
make[3]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
Making install in plugins-scripts
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
test -z "/usr/local/nagios/libexec" || /usr/bin/mkdir -p "/usr/local/nagios/libexec"
 /usr/bin/install -c -o nagios -g nagios check_breeze check_disk_smb check_flexlm check_ircd check_log check_oracle check_rpc check_sensors check_wave check_ifs
tatus check_ifoperstatus check_mailq check_file_age check_ssl_validity utils.sh utils.pm '/usr/local/nagios/libexec'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
```

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
```

```
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.0.3/po'
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.0.3'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.0.3'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.0.3'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.0.3'
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo chkconfig --add nagios
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo chkconfig nagios on
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.0.8
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 08-12-2014
License: GPL

Website: http://www.nagios.org
Reading configuration data...
Error in configuration file '/usr/local/nagios/etc/nagios.cfg' - Line 452 (Check result path '/usr/local/nagios/var/spool/checkresults' is not a valid directory)
   Error processing main config file!

[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$
```

```
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
       Checked 8 services.
       Checked 1 hosts.
       Checked 1 host groups.
       Checked 0 service groups.
       Checked 1 contacts.
       Checked 1 contact groups.
       Checked 24 commands.
       Checked 5 time periods.
       Checked 0 host escalations.
       Checked 0 service escalations.
Checking for circular paths...
       Checked 1 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
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$
```

If Error
sudo mkdir -p /usr/local/nagios/var/spool/checkresults
sudo chown -R nagios:nagcmd /usr/local/nagios/var

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo service nagios start
Reloading systemd:                                         [  OK  ]
Starting nagios (via systemctl):                           [  OK  ]
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$
```

19. Compile and Install Plugins

Commands -

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

make

sudo make install

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

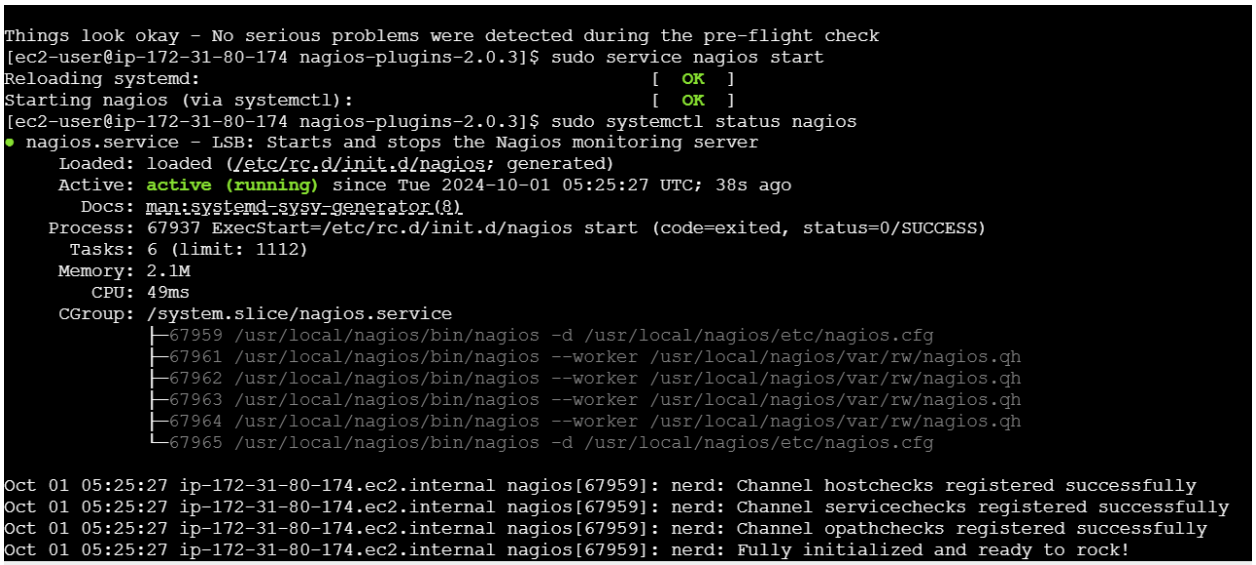
sudo systemctl start nagios

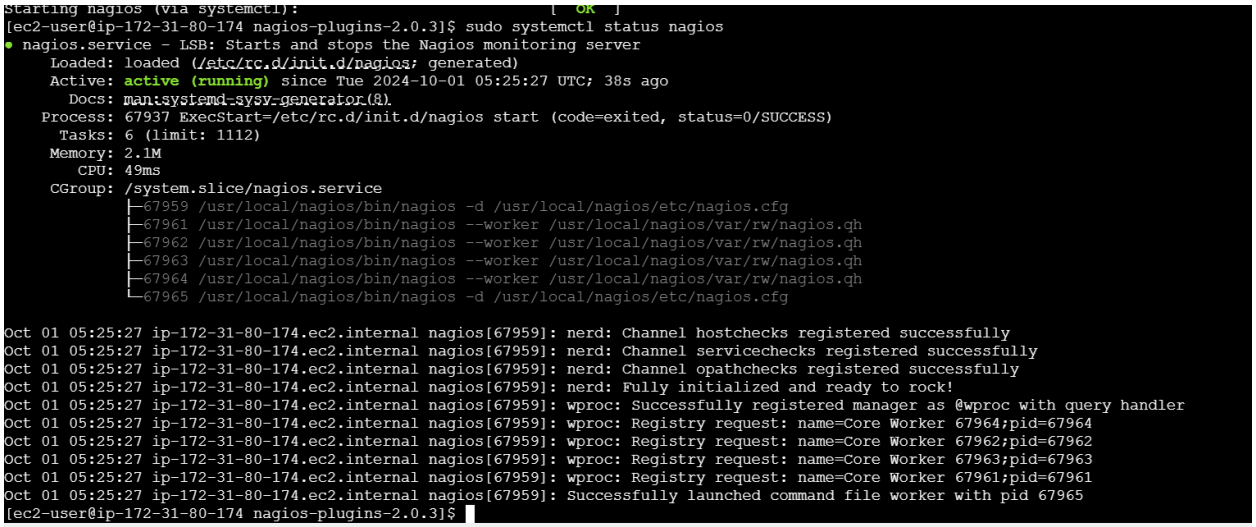
```
[ec2-user@ip-172-31-80-22 nagios-plugins-2.3.3]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo systemctl start nagios
Error reading information on service nagios: No such file or directory
Note: Forwarding request to '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 -

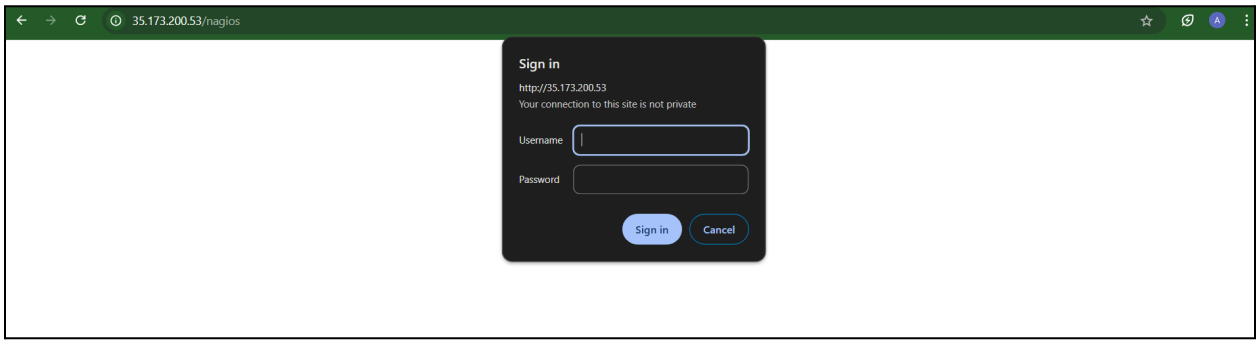
sudo systemctl status nagios

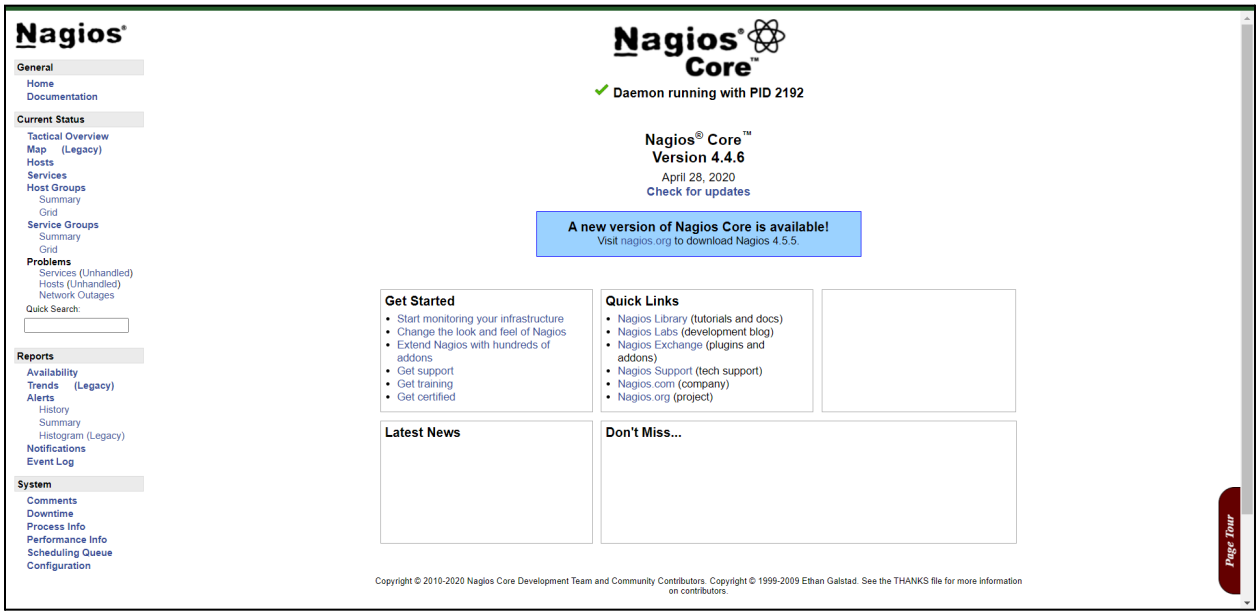




22. Access Nagios Web Interface

- 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.



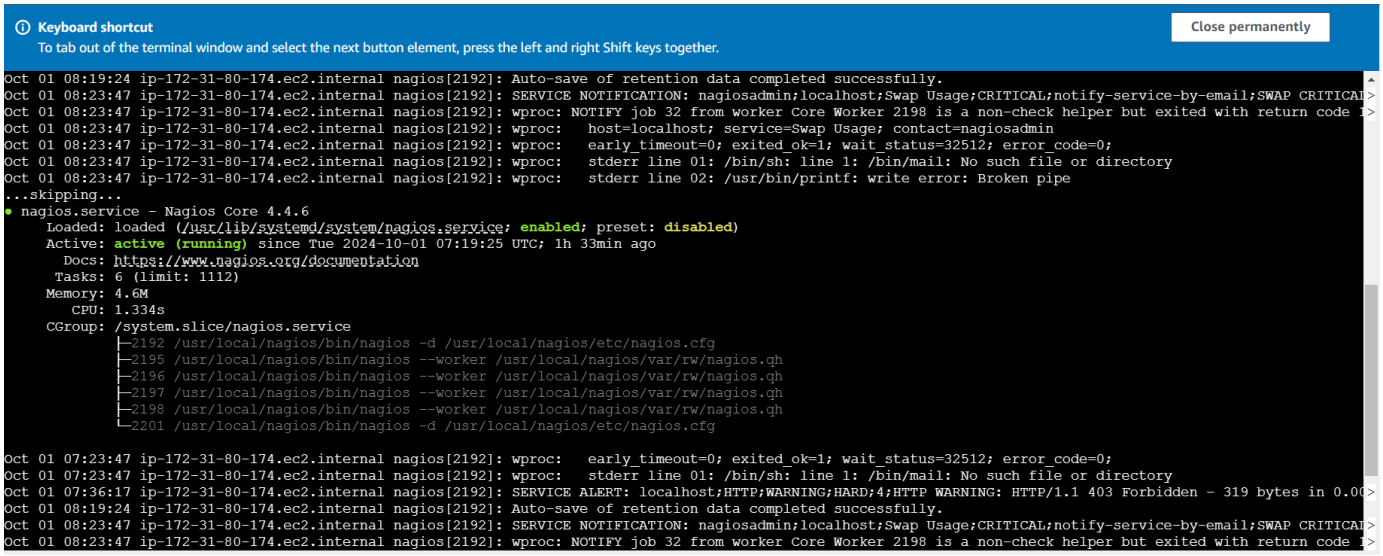


ADVANCE DEVOPS EXPERIMENT 10

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

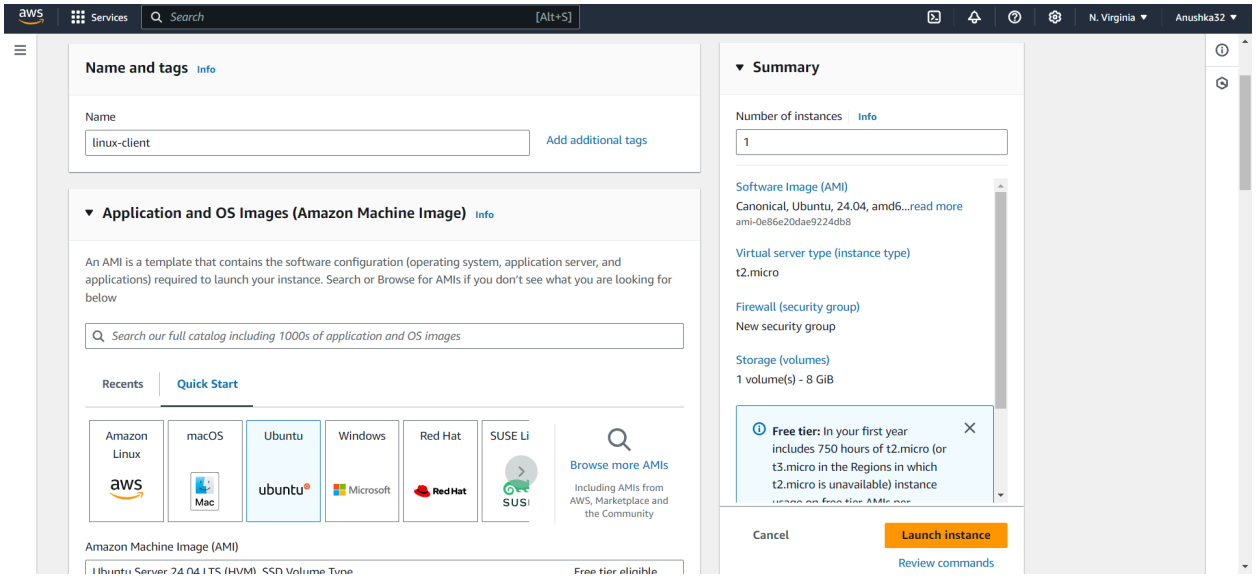
Steps:
Prerequisites: AWS Free Tier, Nagios Server running on Amazon Linux Machine.

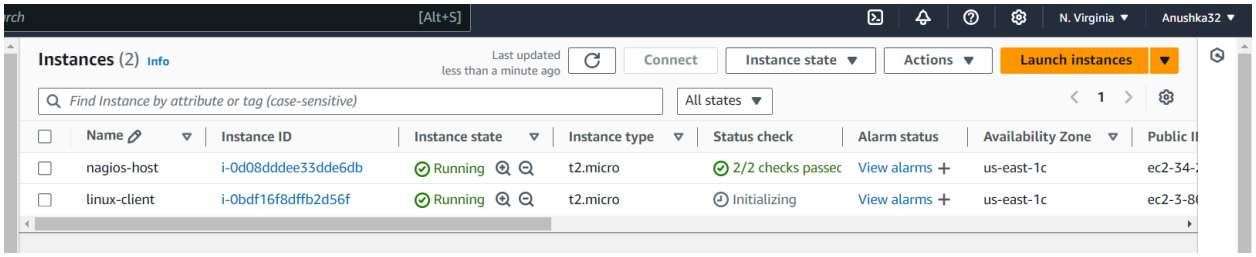
1. To Confirm that Nagios is running on the server side, run this sudo systemctl status nagios on the “NAGIOS HOST”

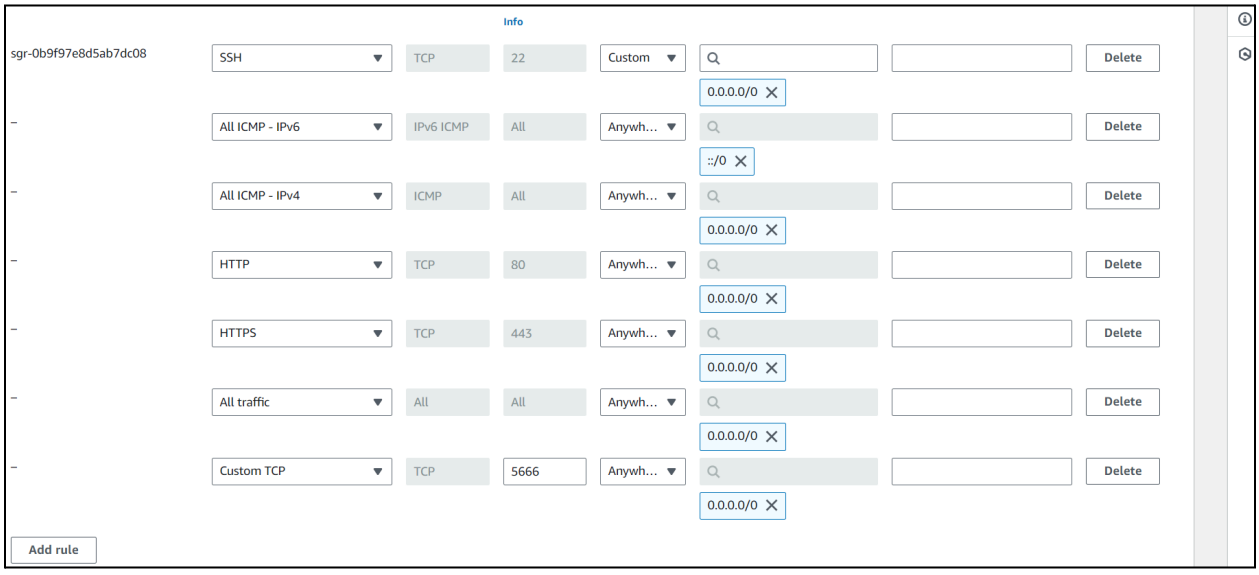


You can proceed if you get this message.

2. Before we begin,
To monitor a Linux machine, create an Ubuntu 20.04 server EC2 Instance in AWS. Provide it with the same security group as the Nagios Host and name it ‘linux-client’ alongside the host.







For now, leave this machine as is, and go back to your nagios HOST machine.

3. On the server, run this command
ps -ef | grep nagios

```
          | | ├─6677 /usr/sbin/httpd -DFOREGROUND
lines 2-29
-bash: nagios: command not found
[ec2-user@ip-172-31-80-174 ~]$ ps -ef | grep nagios
nagios      2192       1  0 07:19 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
nagios      2195    2192  0 07:19 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2196    2192  0 07:19 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2197    2192  0 07:19 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2198    2192  0 07:19 ?        00:00:00 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
nagios      2201    2192  0 07:19 ?        00:00:00 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
ec2-user   33385   33343  0 09:08 pts/1    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-80-174 ~]$
```

4. Become a root user and create 2 folders
sudo su
mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

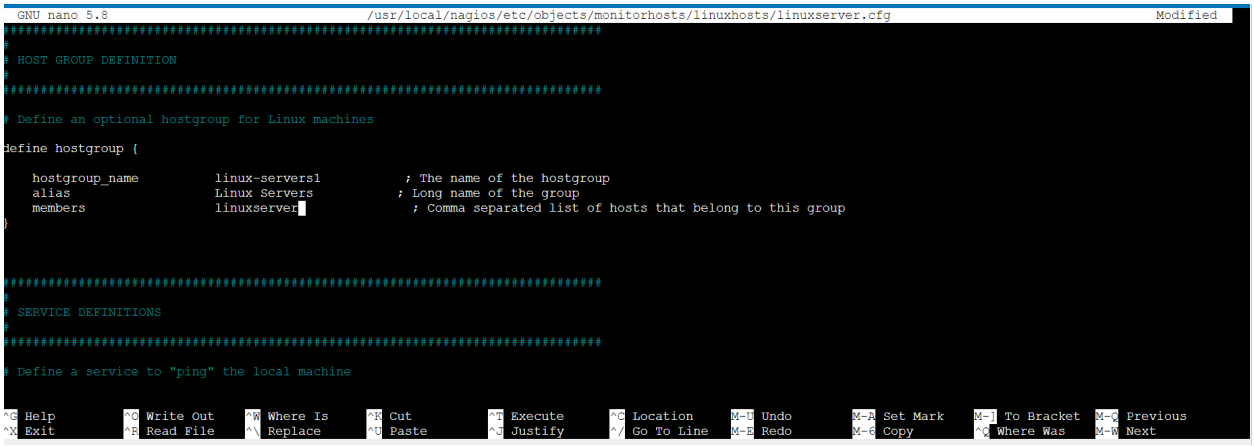
5. Copy the sample localhost.cfg file to linuxhost folder
cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

```
ec2-user   33385   33343  0 09:08 pts/1    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-80-174 ~]$ sudo su
[root@ip-172-31-80-174 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-80-174 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-80-174 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg
cp: missing destination file operand after '/usr/local/nagios/etc/objects/localhost.cfg'
Try 'cp --help' for more information.
```

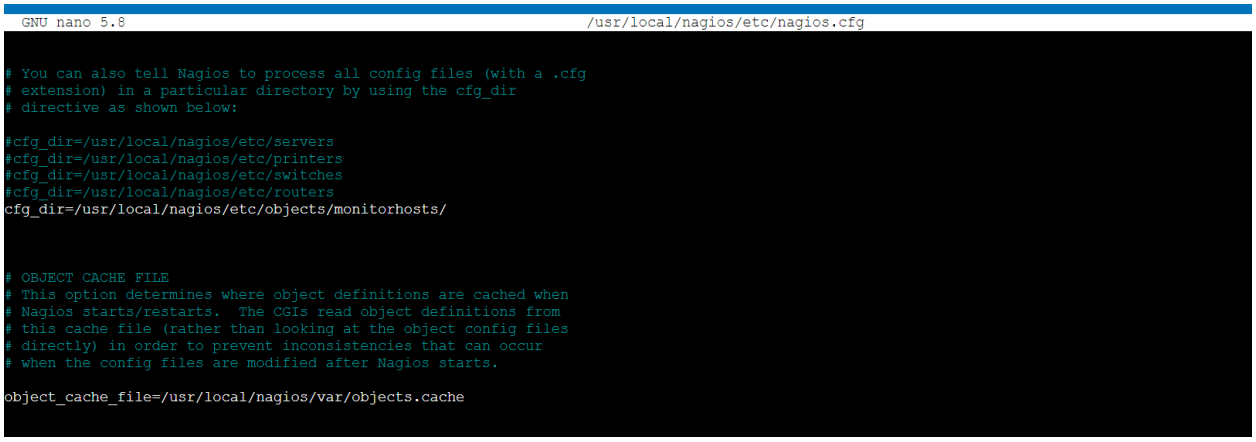
```
bash: /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg: No such file or directory
[root@ip-172-31-80-174 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
cp: missing destination file operand after '/usr/local/nagios/etc/objects/localhost.cfg/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg'
Try 'cp --help' for more information.
[root@ip-172-31-80-174 ec2-user]# cp/usr/local/nagios/etc/objects/localhost.cfg/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
bash: cp/usr/local/nagios/etc/objects/localhost.cfg/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg: No such file or directory
[root@ip-172-31-80-174 ec2-user]#
```

6. Open linuxserver.cfg using nano and make the following changes
nano
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
Change the hostname to linuxserver (EVERYWHERE ON THE FILE)
Change address to the public IP address of your LINUX CLIENT.

Change hostgroup_name under hostgroup to linux-servers1

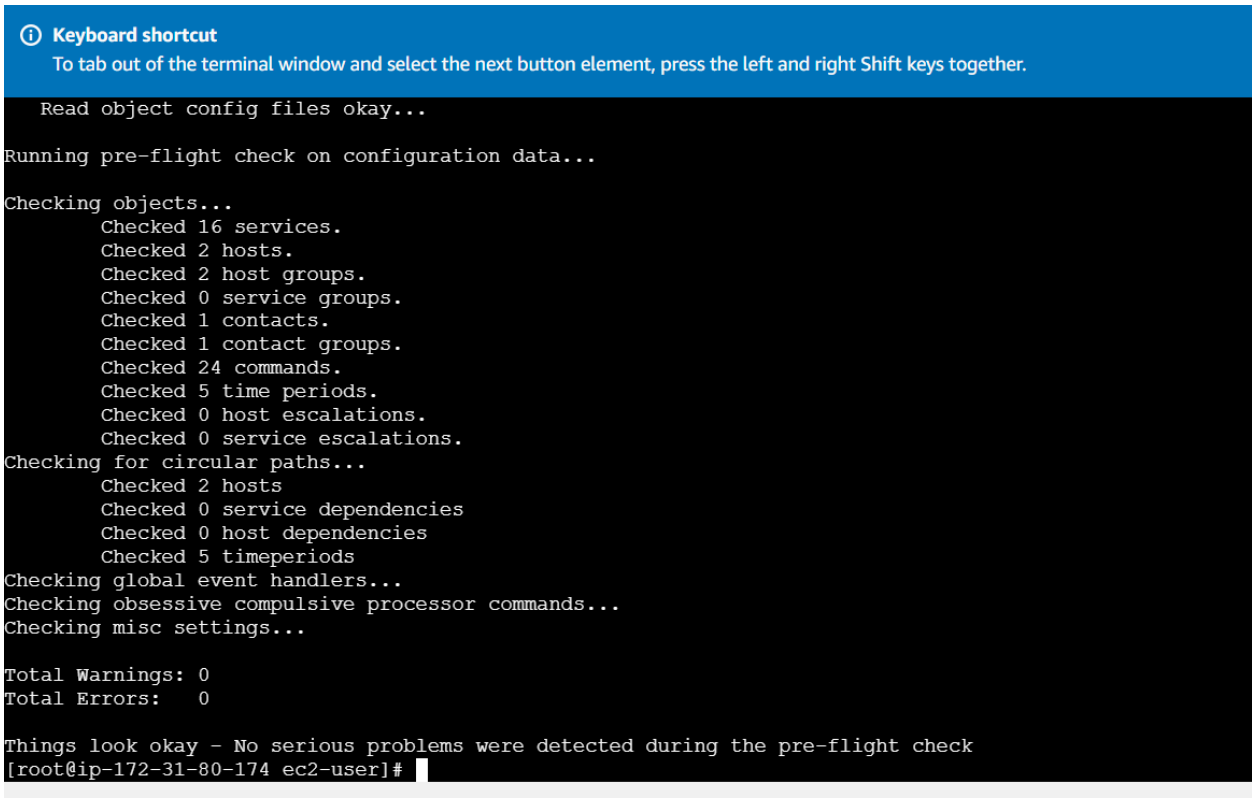


Everywhere else on the file, change the hostname to linuxserver instead of localhost.
7. Open the Nagios Config file and add the following line
nano /usr/local/nagios/etc/nagios.cfg
##Add this line
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/



8. Verify the configuration files

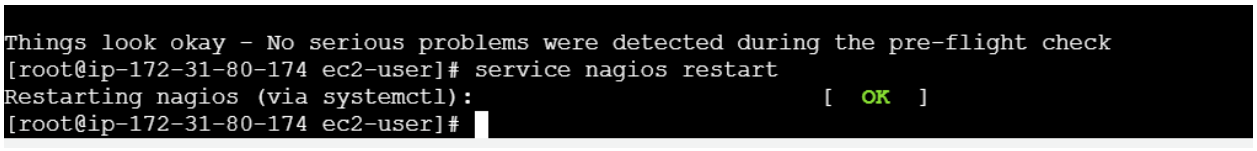
You are good to go if there are no errors.



9. Restart the nagios service
service nagios restart

Now it is time to switch to the client machine.

10. SSH into the machine or simply use the EC2 Instance Connect feature.



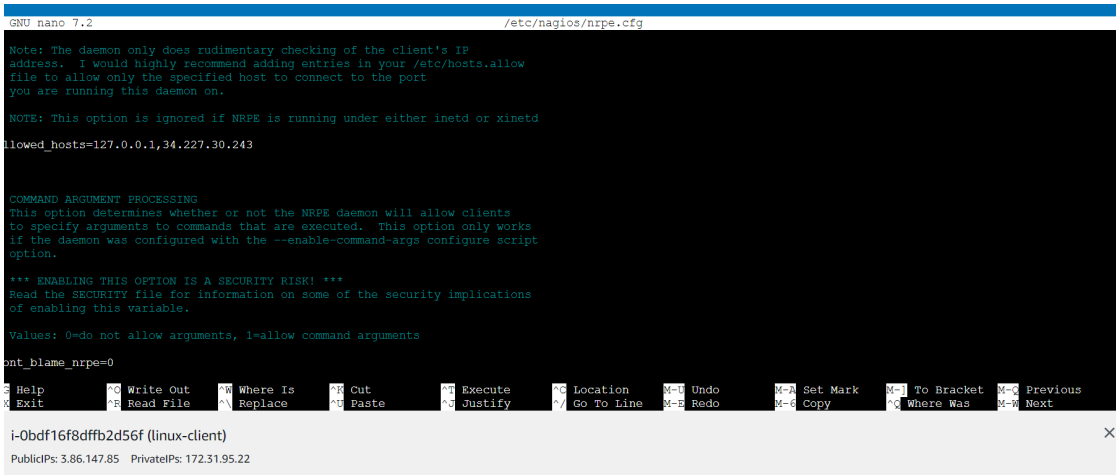
```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-95-22:~$ sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins
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://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.1 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4576 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [274 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [116 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.4 kB]
```

i-0bdf16f8dffb2d56f (linux-client)

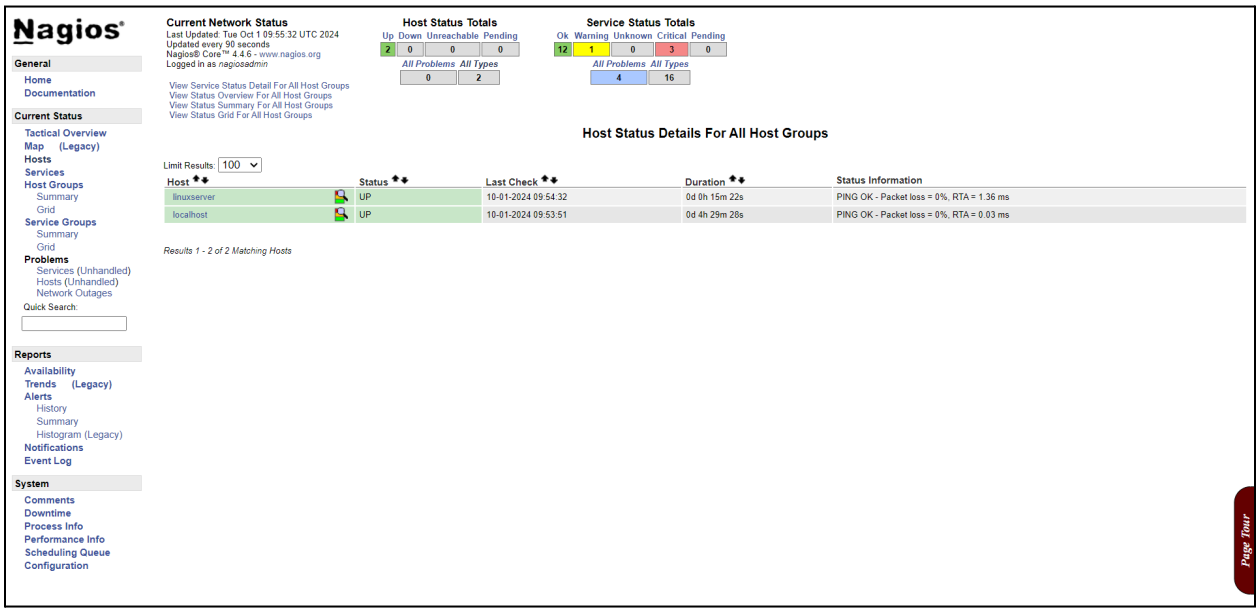
11. Make a package index update and install gcc, nagios-nrpe-server and the plugins.
sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins

12. Open nrpe.cfg file to make changes.
sudo nano /etc/nagios/nrpe.cfg
Under allowed_hosts, add your nagios host IP address like so

13. Restart the NRPE server
sudo systemctl restart nagios-nrpe-server



14. Now, check your nagios dashboard and you'll see a new host being added.



Advance Devops-11

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

Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services.

The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

● AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.

● It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.

● You don't need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.

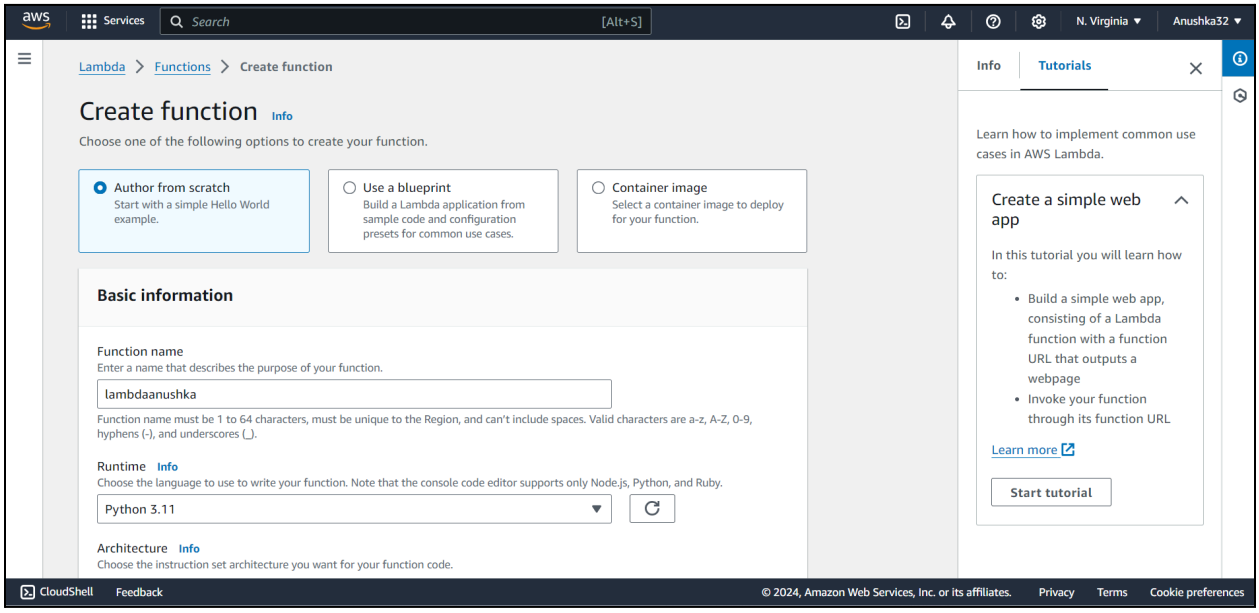
● AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.

● It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down.

Anushka Shahane D15A 55

Steps to create an AWS Lambda function

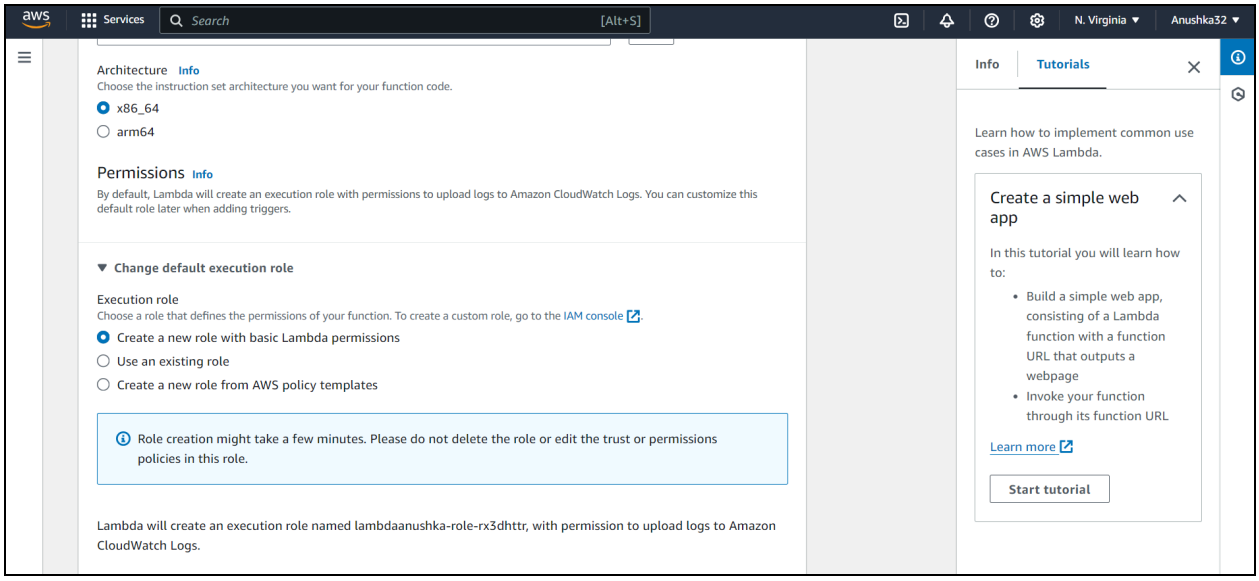
1. Open up the Lambda Console and click on the Create button. Be mindful of where you create your functions since Lambda is region-dependent.

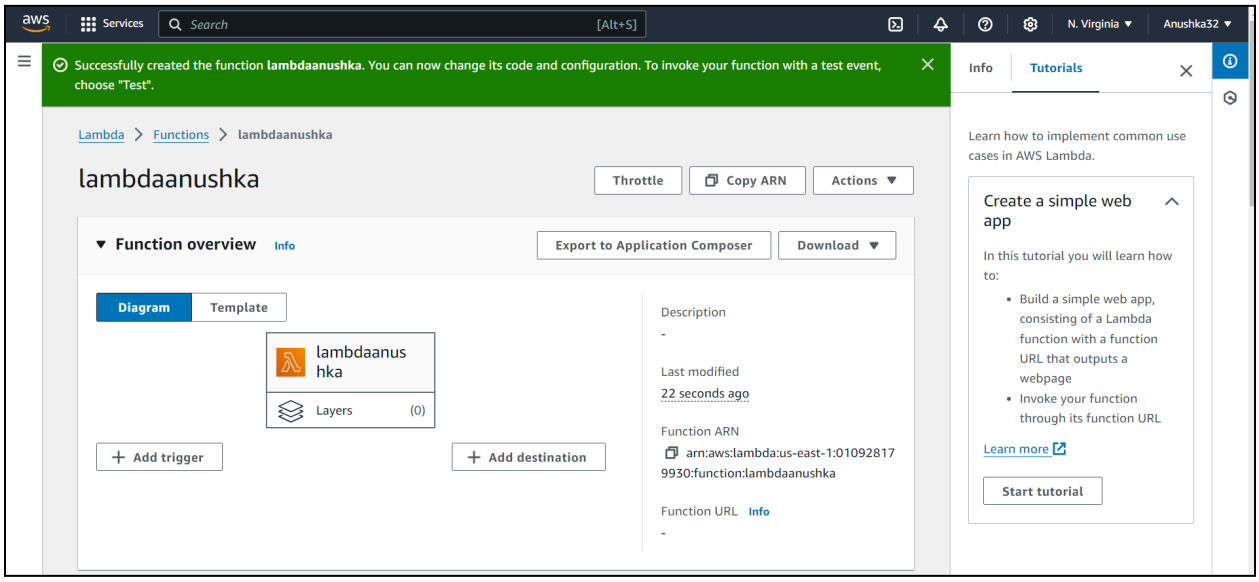


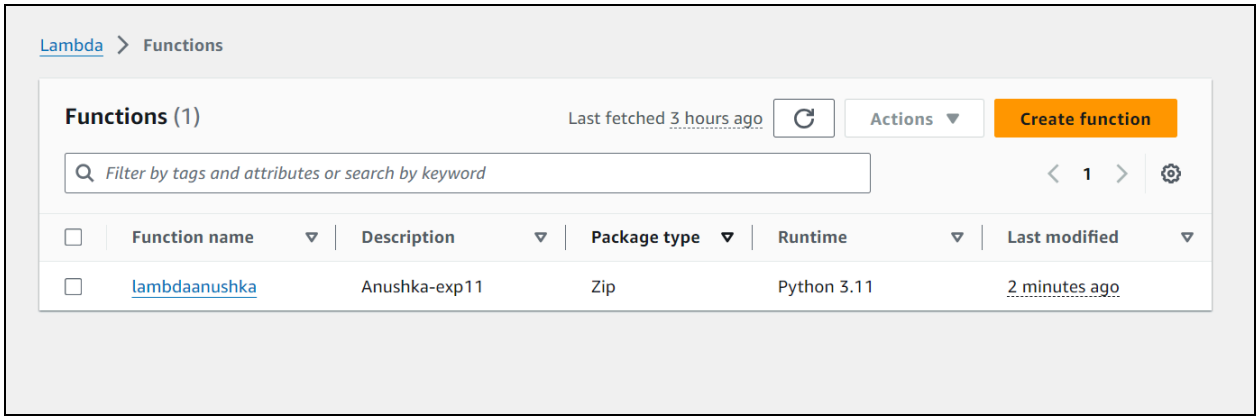
2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.

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.

Anushka Shahane D15A 55

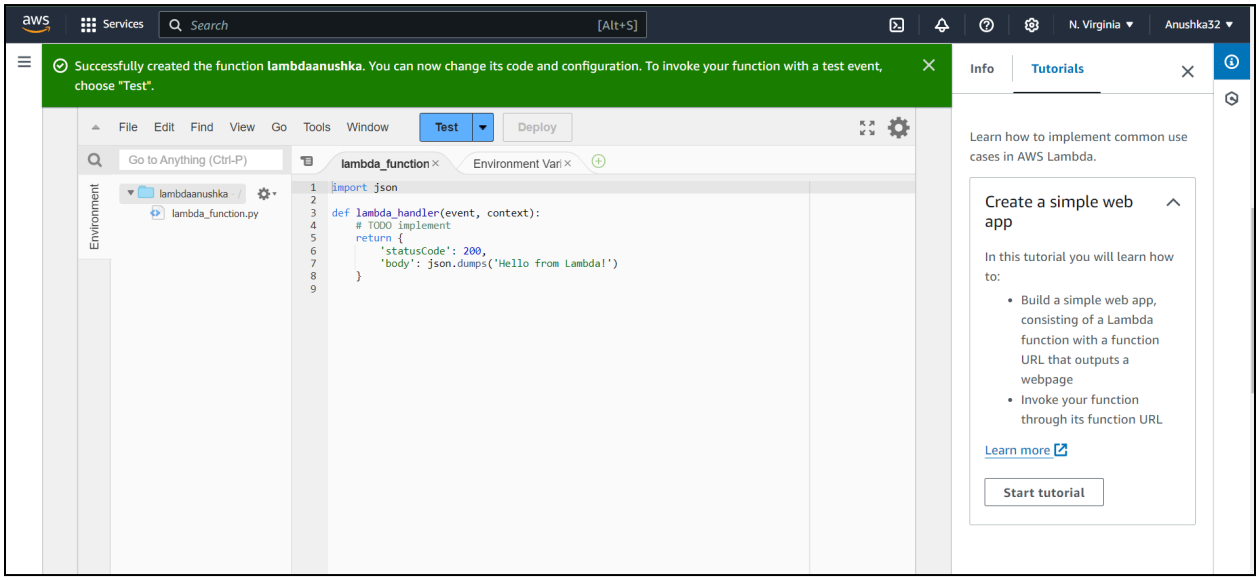


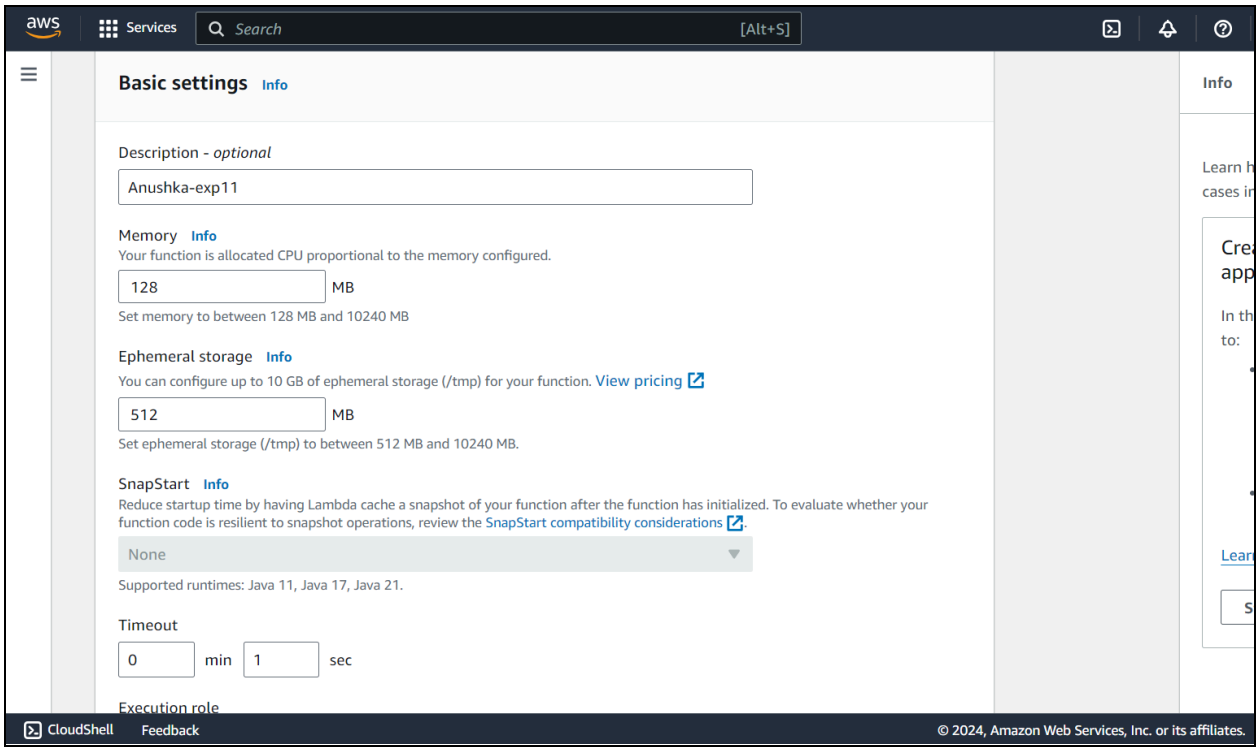




Click on the Create button.

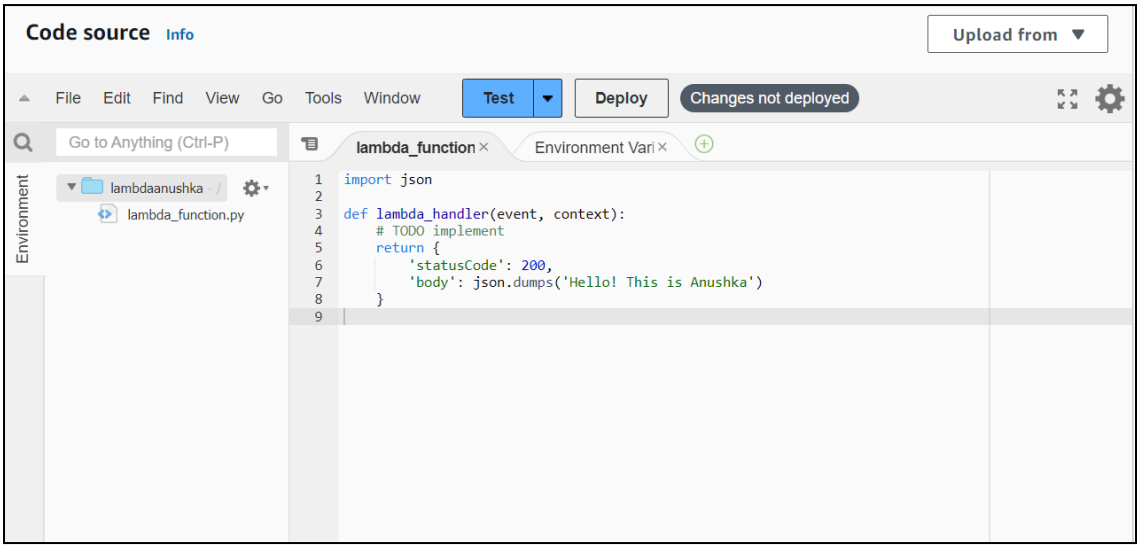
3. This process will take a while to finish and after that, you'll get a message that your function was successfully created.



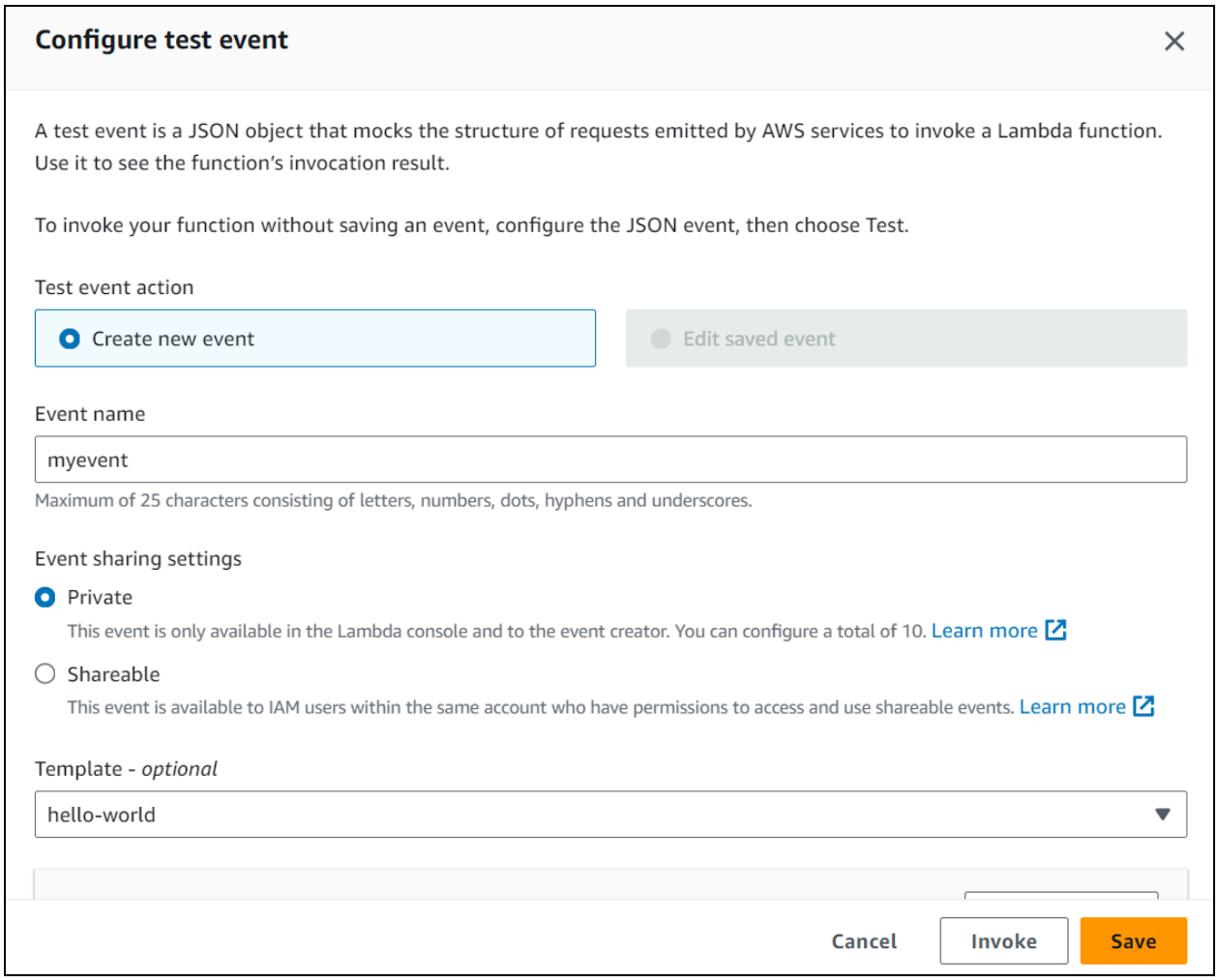


4. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.

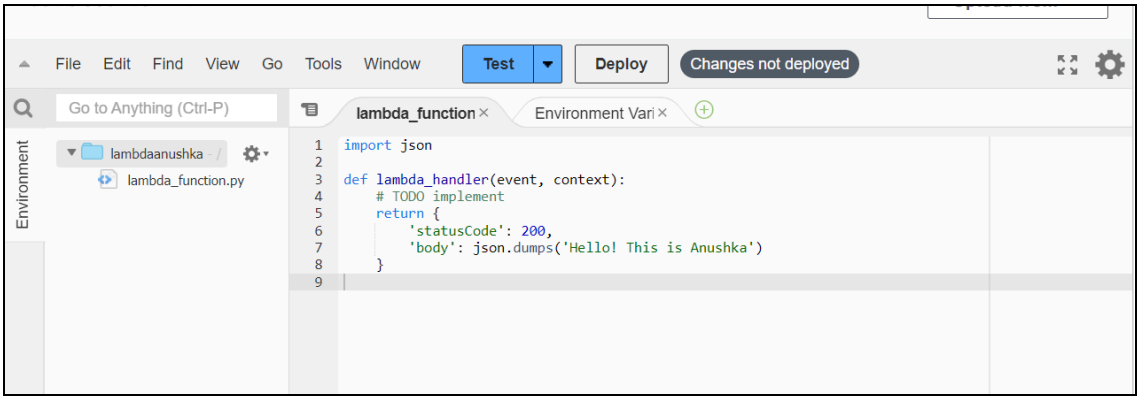
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.



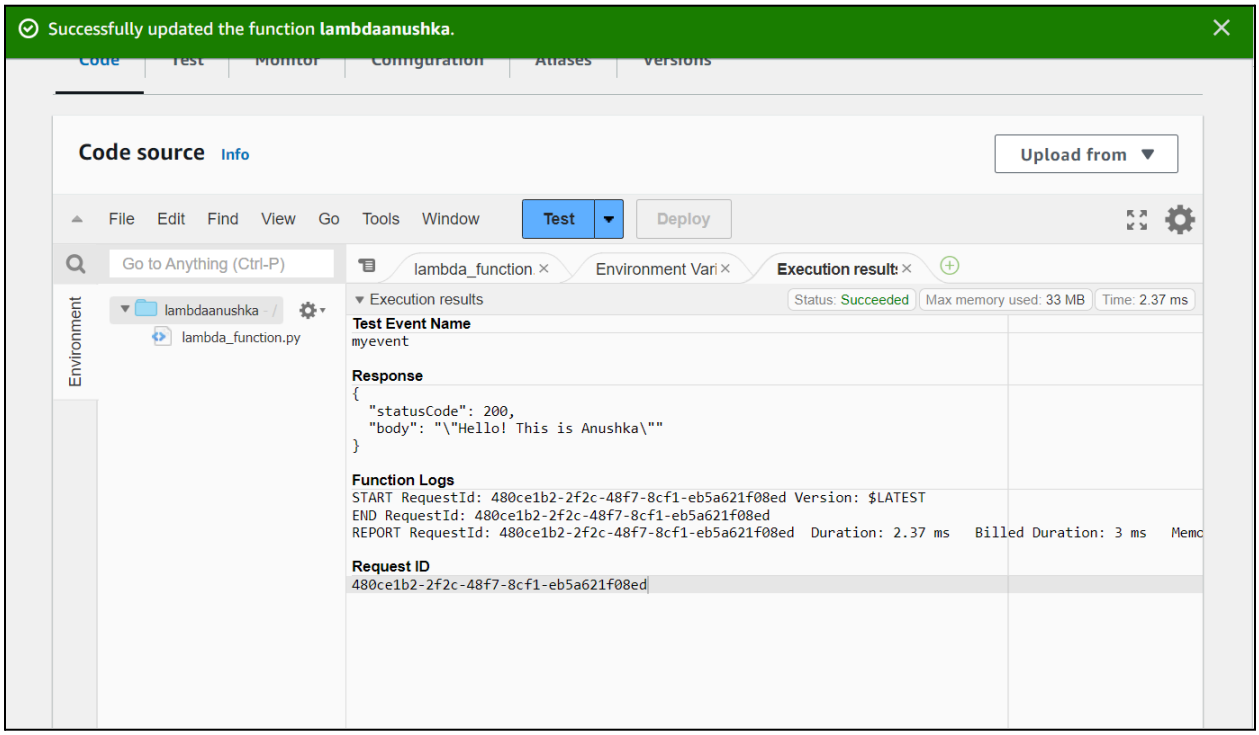
5. You can make changes to your function inside the code editor. You can also upload a zip
file of your function or upload one from an S3 bucket if needed.
Press Ctrl + S to save the file and click Deploy to deploy the changes.



6. Click on Test and you can change the configuration, like so. If you do not have anything
in the request body, it is important to specify two curly braces as valid JSON, so make sure they are there.



7. Now click on Test and you should be able to see the results.



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.

Advance Devops-12

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

Theory:
AWS Lambda and S3 Integration: AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:
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.

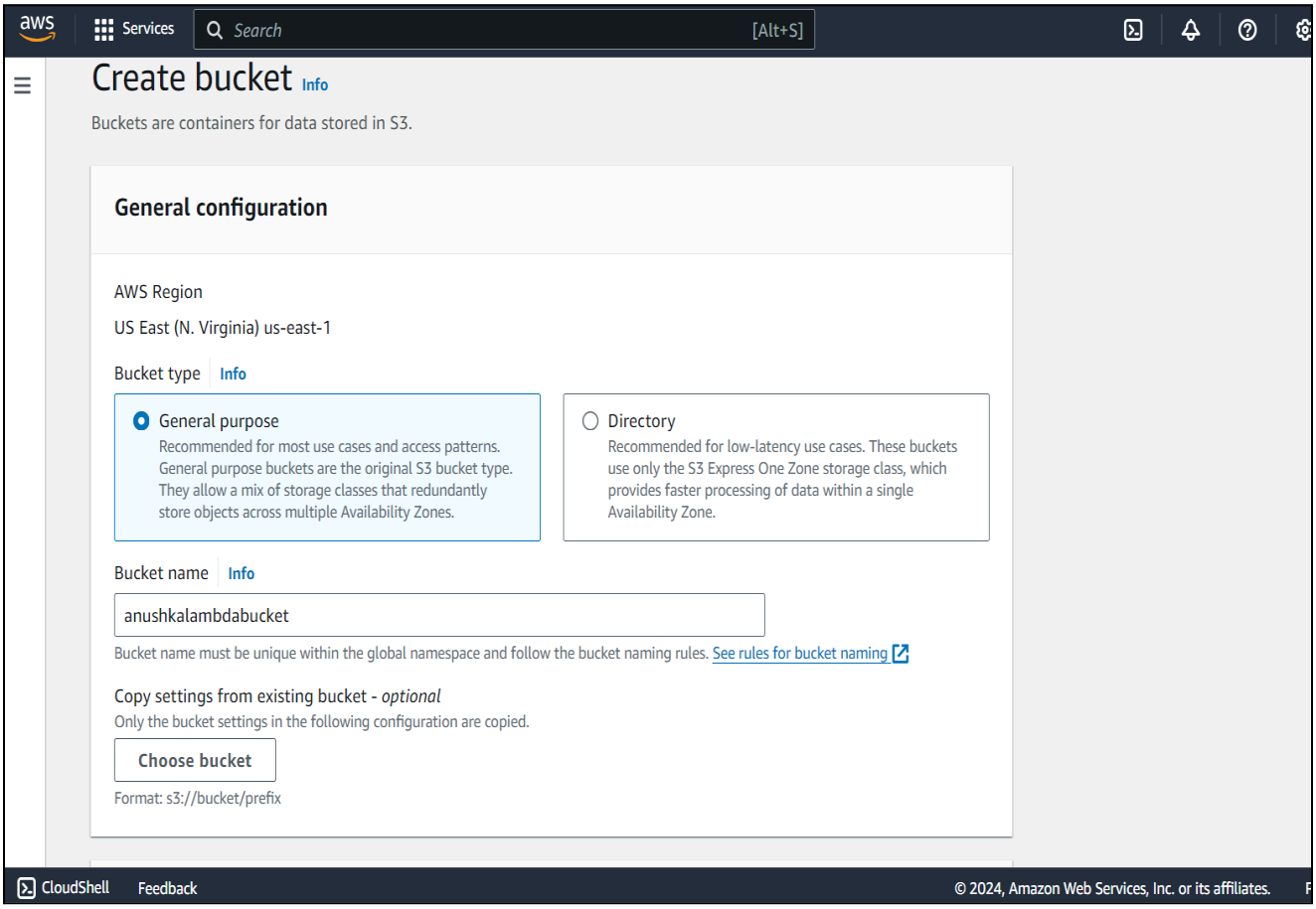
2. Create the Lambda Function:
Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.
Write code that logs a message like “An Image has been added” when triggered.

3. Set Up Permissions:
Ensure that the Lambda function has the necessary permissions to access S3.
You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. 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).

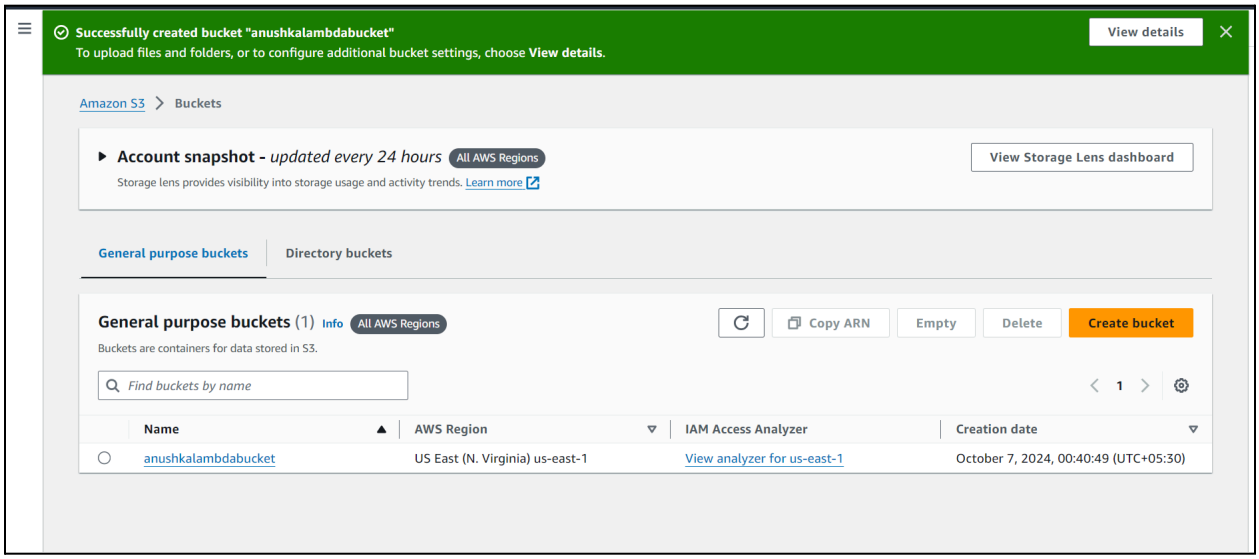
5. Test the Setup:
Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

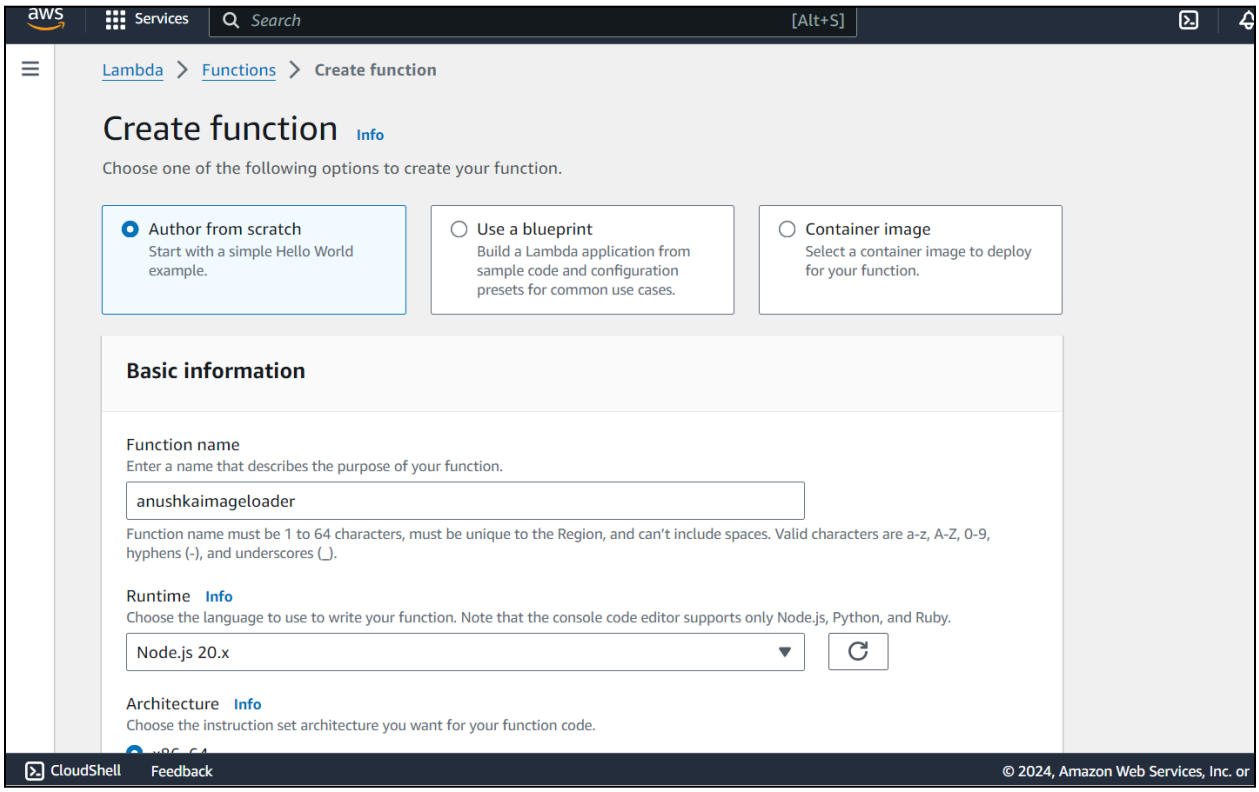
1. Create an S3 Bucket:
First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.



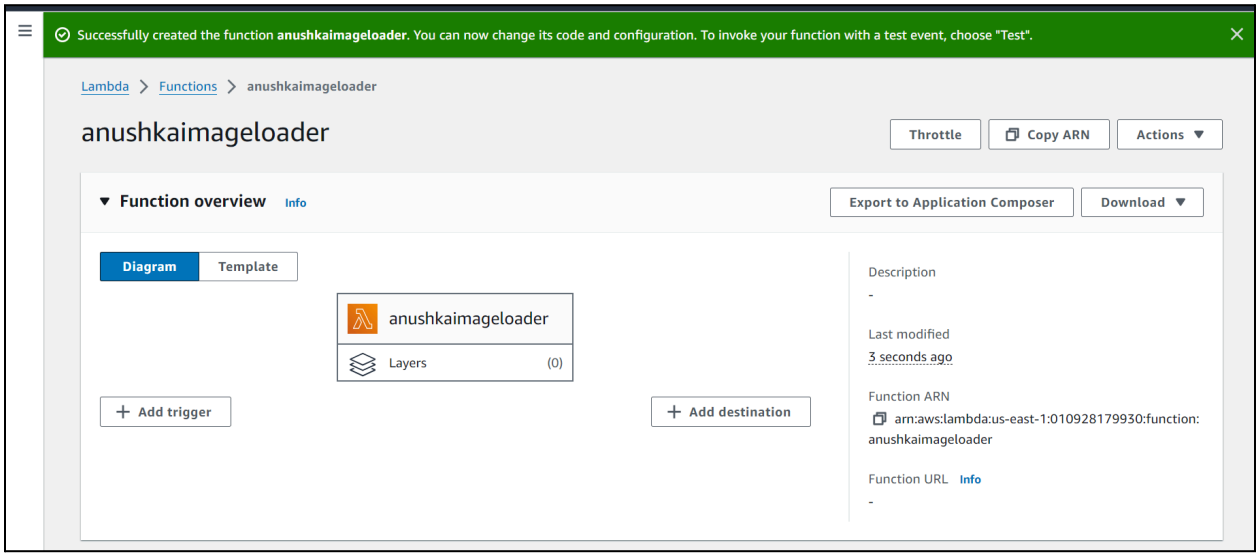
2. Create the Lambda Function:
Set up a new Lambda function using AWS Lambda's console. You can choose a runtime environment like Python, Node.js, or Java.
Write code that logs a message like "An Image has been added" when triggered.

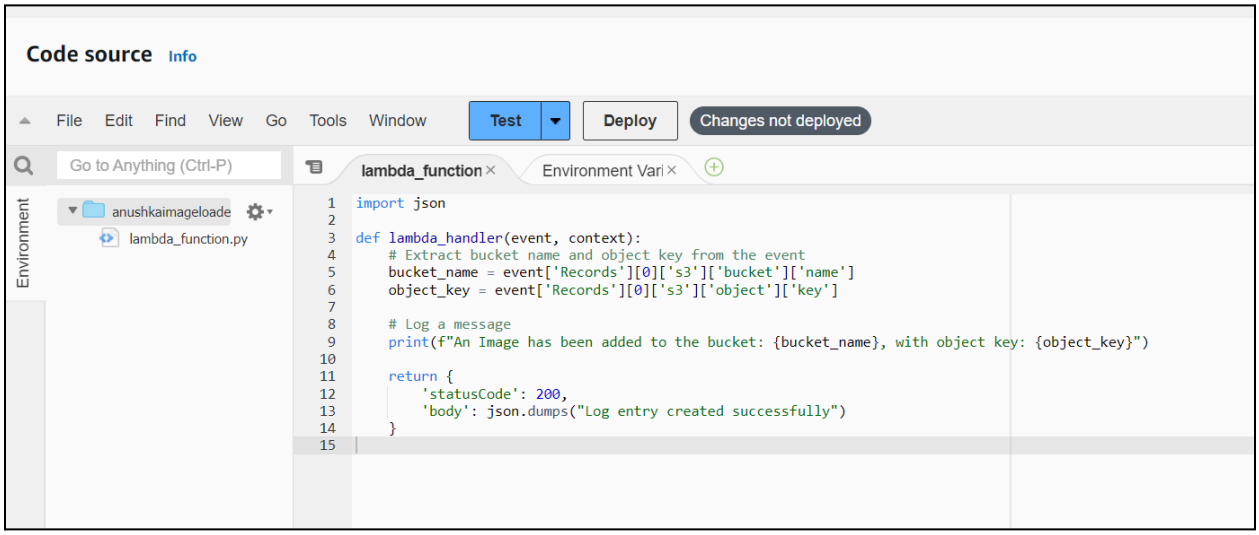
Anushka Shahane D15A 55



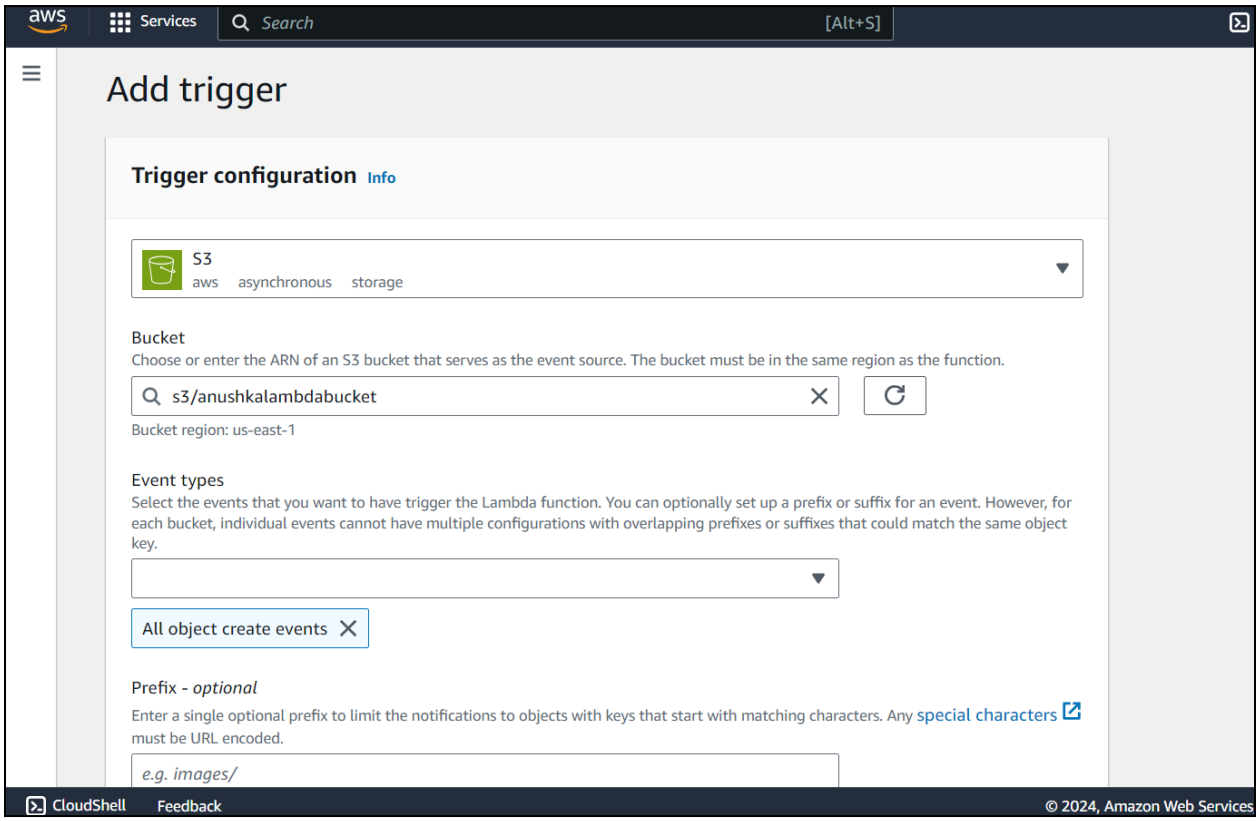


3. Set Up Permissions:
Ensure that the Lambda function has the necessary permissions to access S3.
You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

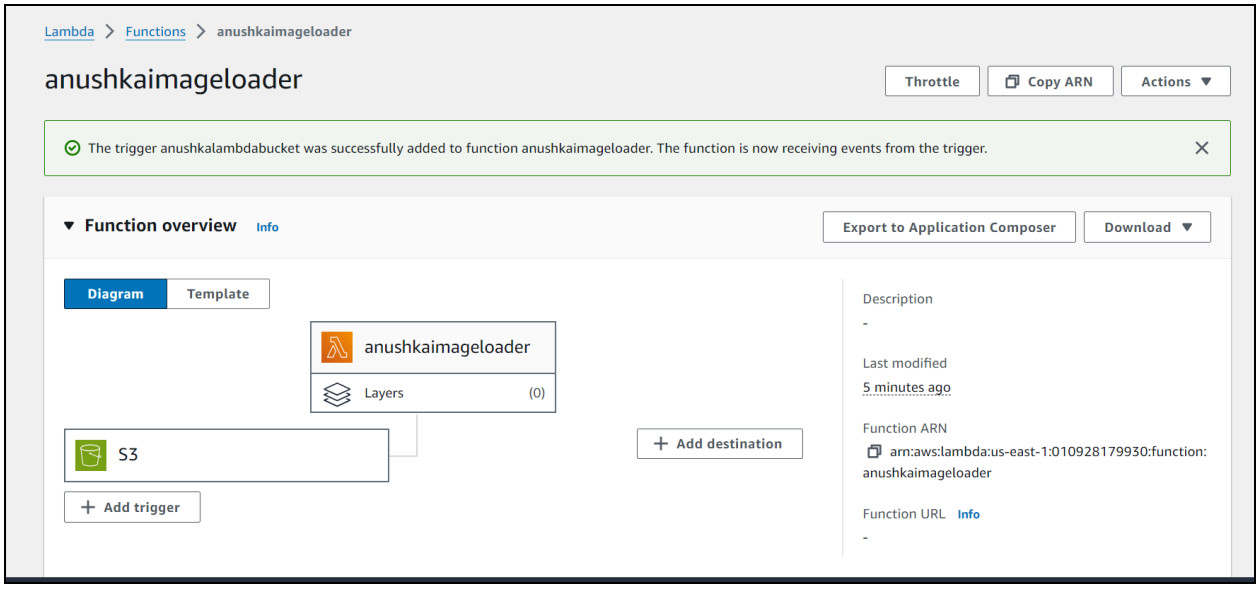


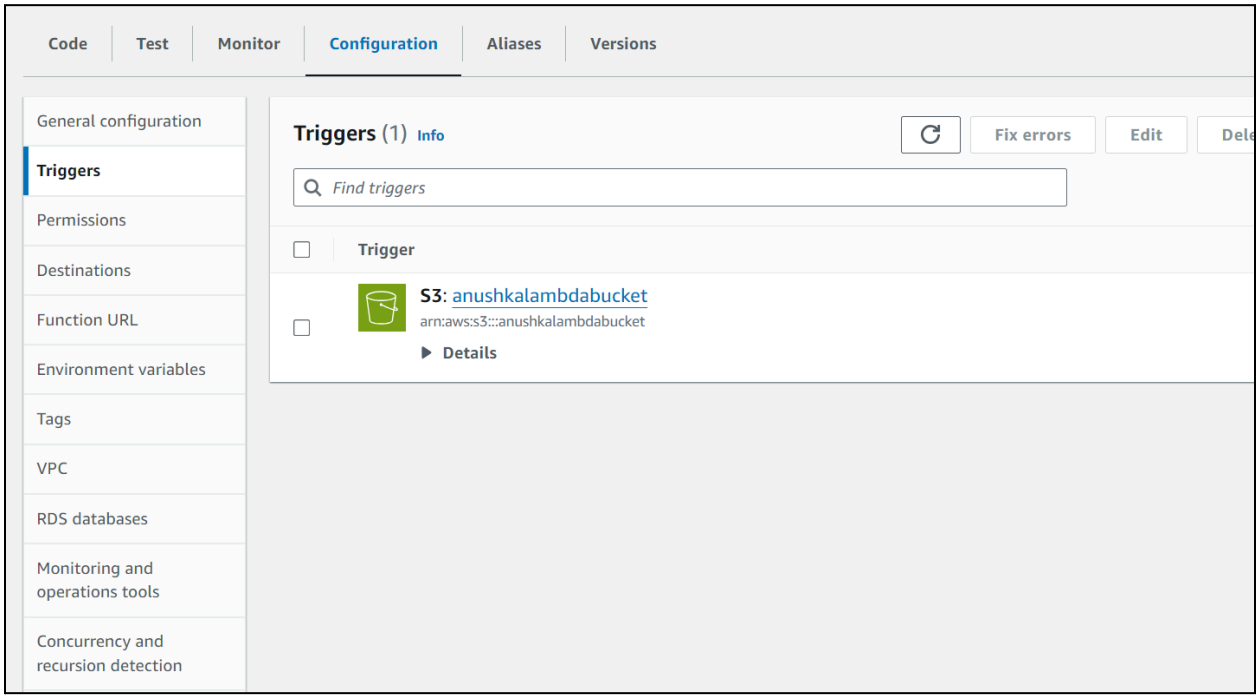


4. Configure S3 Trigger:
Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

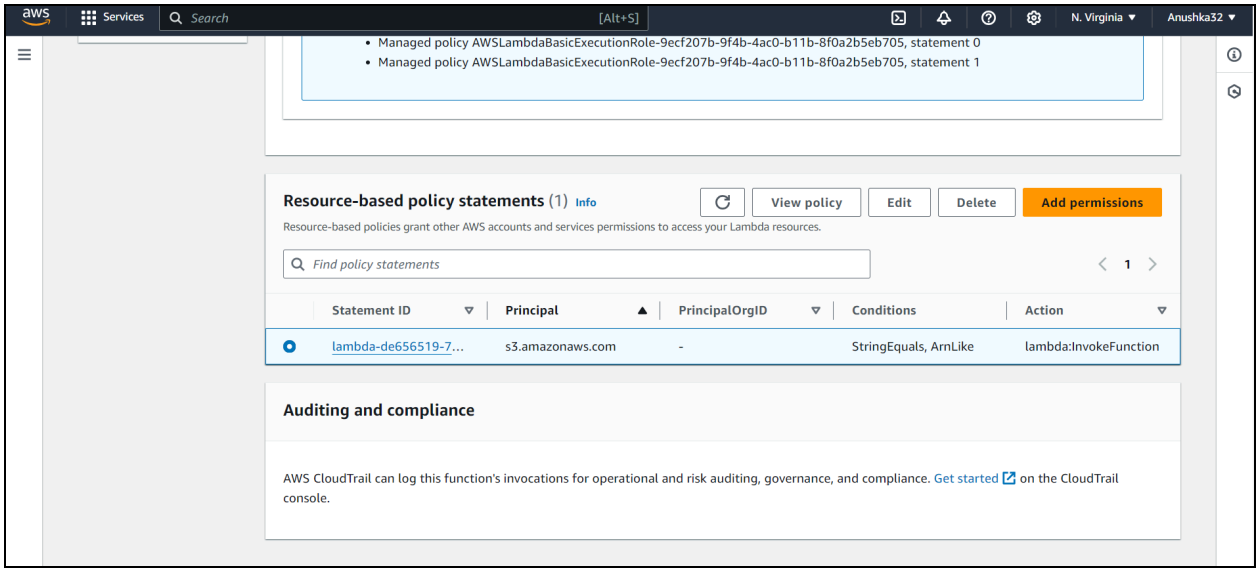


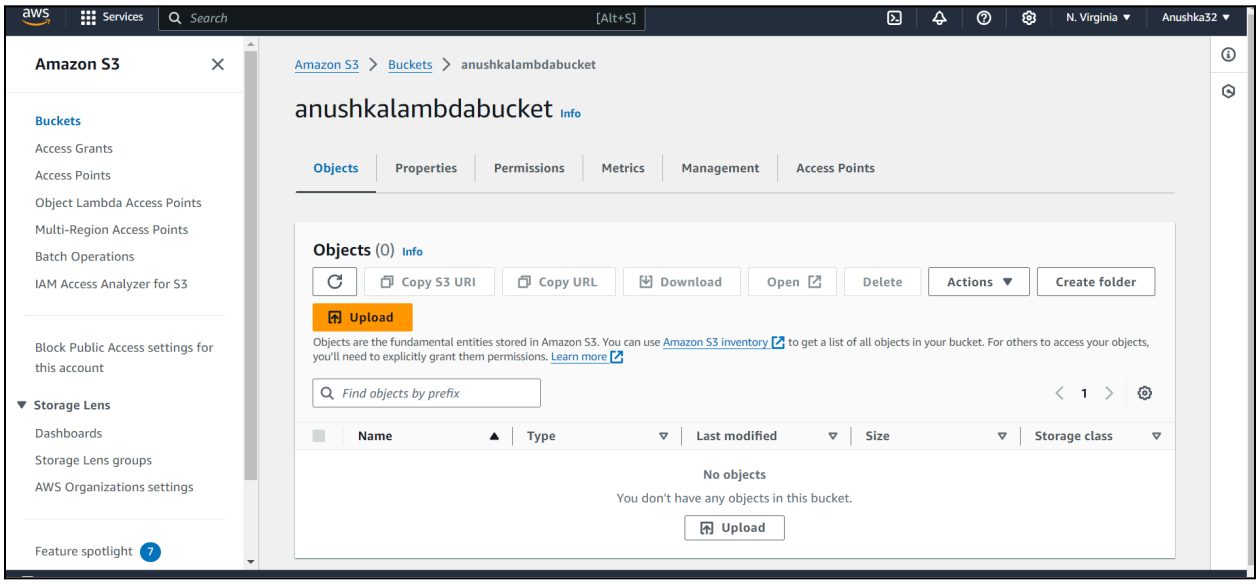
Anushka Shahane D15A 55



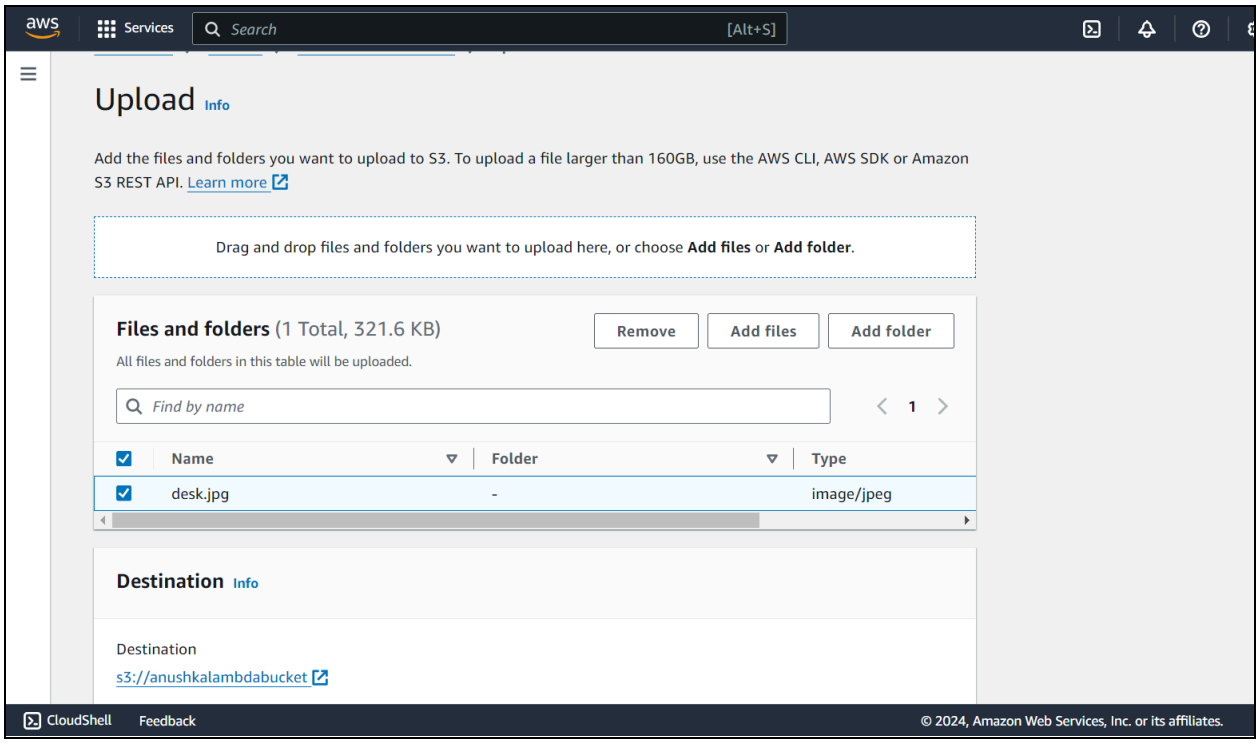


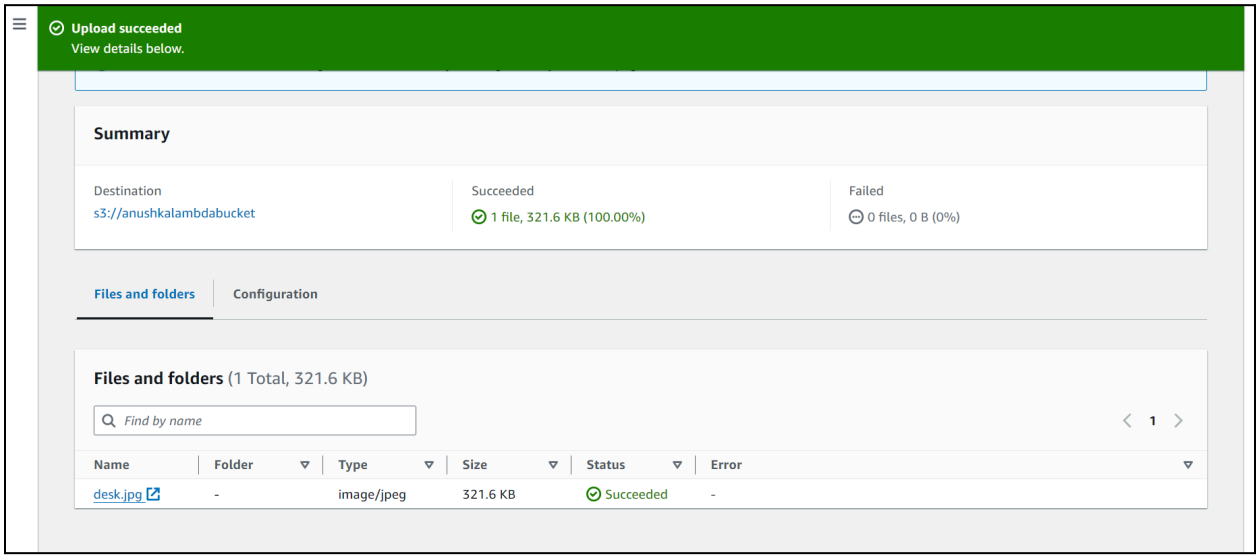
Anushka Shahane D15A 55

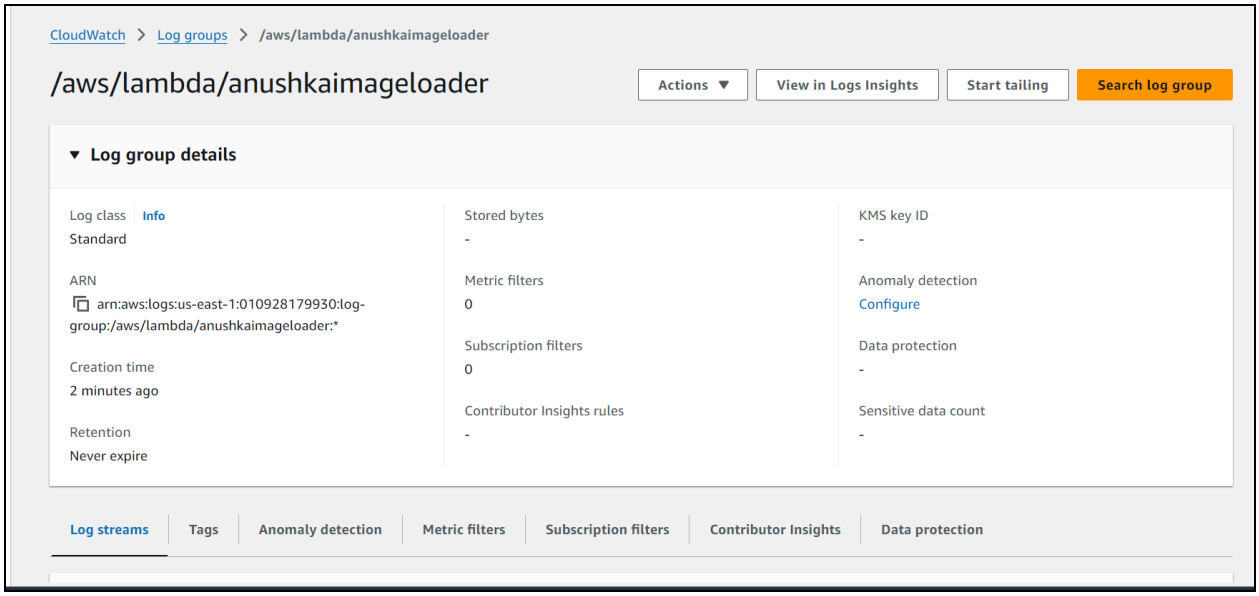




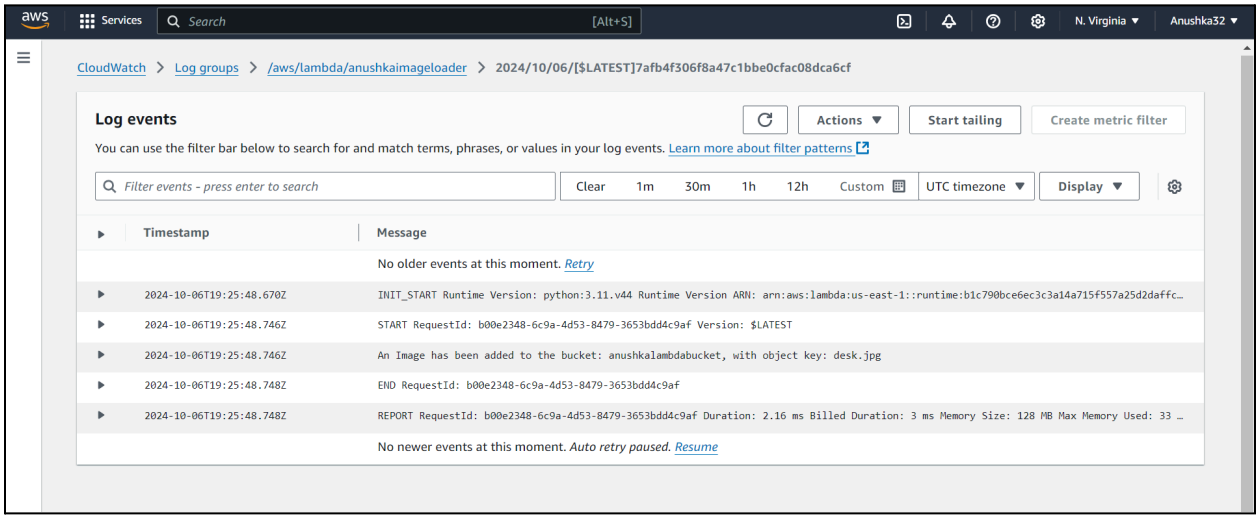
Anushka Shahane D15A 55







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



Conclusion:

Integrating AWS Lambda with S3 allows for real-time, automated processing of events such as file uploads. In this example, a Lambda function is configured to log a message whenever an image is added to a specific S3 bucket. This setup demonstrates the power and flexibility of serverless computing by automating tasks without requiring manual intervention or server management. By leveraging AWS Lambda, developers can efficiently handle event-driven workflows, reduce operational overhead, and quickly deploy scalable solutions that respond to specific actions within cloud environments.

Anushka Shahane
D15A (55)

05/05

ASSIGNMENT NO. 1
Advance Devops

Q1. Use S3 bucket and host Video Streaming

Ans: To host video streaming using Amazon S3 bucket you can create a scalable and cost-effective solution by utilizing S3 for video storage and AWS CloudFront as a content Delivery Network (CDN) to efficiently stream video to users.

Steps to Host Video Streaming with S3 bucket:

(1) Create an S3 bucket
Store video files (MP4, WebM etc) in an S3 bucket. set appropriate permissions to allow access to videos via CloudFront.

(2) Upload Videos:
upload your videos to the S3 bucket. Ensure the files are publicly accessible if necessary, or restrict access based on user roles.

(3) Enable S3 Static Website Hosting (optional):
If you want users to access a website along with streaming content, enable static website hosting for your S3 bucket.

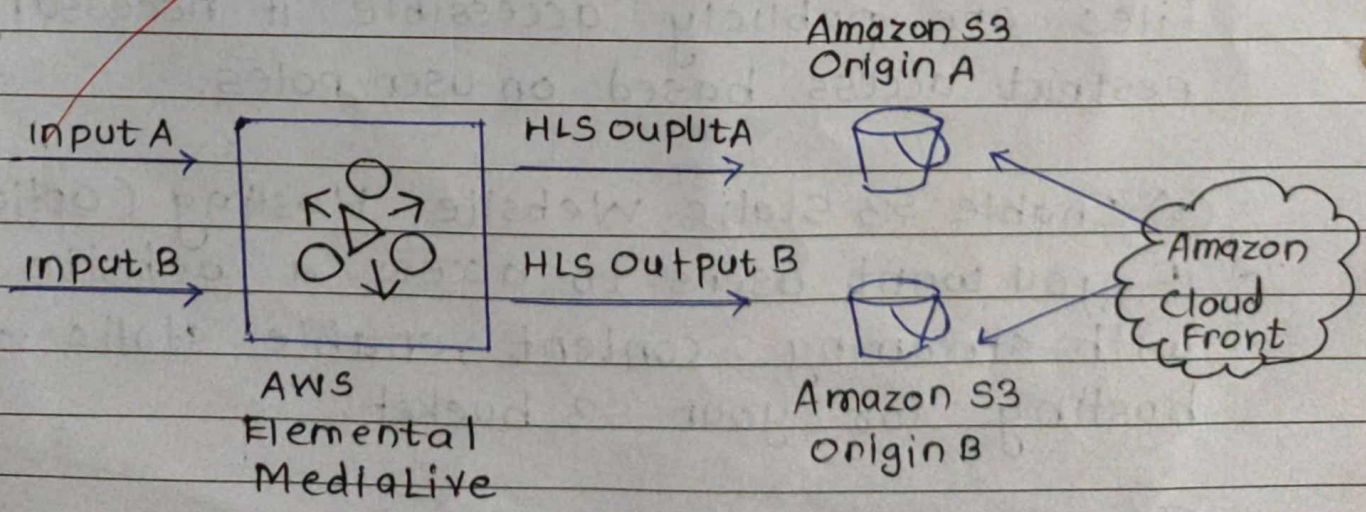
(4) Configure AWS CloudFront:
Create a CloudFront distribution with the S3 bucket as the origin. Cloudfront caches your video content at edge locations worldwide,

reducing latency for users.

(5) Setup permissions:
Set bucket policies or IAM roles to secure your S3 bucket from unauthorized access, ensuring only CloudFront can fetch content.

(6) Streaming optimization:
You can transcode videos using AWS Elastic Transcoder to support multiple resolutions & formats..
Use chunk-based streaming to provide smooth streaming to users with varying bandwidth.

(7) Custom Domain and SSL (Optional)
Integrate a custom domain for your cloudfront distribution and use an SSL certificate for secure streaming.



Q(2) Discuss BMW and Hot Star case Studies using AWS.

Soln BMW Case study using AWS

Ans: Background: BMW is a leading automotive manufacturer known for its luxury vehicles, while Hotstar (now known as Disney+Hotstar) is a popular streaming platform in India, offering a variety of content including movies, TV shows and live sports. Both companies have utilized AWS to drive innovation improve customer experiences & optimize their operations

BMW's use of AWS

(1) Connected vehicles and Data Analytics

BMW has been at the fonefront of integrating technology into their vehides. By leveraging AWS, they can collect and analyze vast amount of data from their connected cars. This includes: ① vehical performance data ② Driver Behaviour

Benefits:

① Predictive maintenance: Aws enables BMW to use machine learning algorithms to predict when a vehicle needs servicing, reducing downtime and improving customer satisfaction.

(2) Scalability and Cost Management

BMW utilizes AWS scalable infrastructure to

handle varying workloads, especially during product launches on events.

Benefits:

① Cost efficiency: BMW can scale resources up or down based on demand, ensuring they only pay for what they use.

② Global Reach: Aws's global infrastructure allows BMW to deploy applications closer to their customers, reducing latency and improving service delivery.

Hotstar's Use of AWS

① Content Delivery and streaming Services
Hotstar relies heavily on AWS to manage its massive content library and delivery of high quality streaming experiences to millions of users.

⊙ Aws Cloudfront: Hotstar uses Aws's Content Delivery Network (CDN) to deliver content quickly and efficiently ensuring minimal buffering and downtime during peak times such as major sports events.

Benefits:

· Scalability: During events like the IPL user traffic can spike dramatically. Aws allows Hotstar to scale resources dynamically

to handle these spikes without comparomising performance.

· Global Content Reach: Aws enables Hotstar to distribule content across multiple regions, ensuring that users worldwide can access their services seamlessly.

(2) Data Aanalytics for User Engagement: Hotstar leverages AWS data analytics tools to gather insights about user behaviour, content performances and viewing patterns.

· AWS Redshift and Athena: These services help Hotstar analyze large datasets to improve content recommendations and user engagement strategies.

Benefits:
⊙ Personalized Content Recommendations
⊙ Targeted Advertising

challenges and Solutions:
while both BMW and Hotstar have seen significant benefits from using AWS they also face challenges:

① Data Security and Compliance: Protecting user data
② Cost management: As usage scales, manage

costs can become challenging

Conclusion:
The integration of Aws into BMW and Hotstar operations demonstrates how cloud computing can drive innovation, improve customer experiences and enhance operational efficiency.

Q(3) why Kubernetes and advantages and disadvantages of Kubernetes. Explain how adidas Uses Kubernetes.

Soln: Kubernetes is an open-source container orchestration platform designed to automate deploying, scaling, and managing containerized applications. It abstracts the underlying infrastructure, allowing developers to focus on writing code rather than managing servers.

Advantages of Kubernetes:
(1) Scalability: Kubernetes can automatically scale applications up or down based on demand, ensuring optimal resource use.

(2) High Availability: It provides mechanisms for self-handling, meaning if a container fails, Kubernetes can automatically restart or replace it.

(3) Load Balancing : Kubernetes can distribute network traffic evenly across containers, improving application performance and reliability.

(4) Declarative Configuration: Users can define the desired state of applications using YAML files making deployments reproducible and version-controlled.

(5) Portability : Kubernetes can run on any cloud or on-premises infrastructure making it easier to move applications between enviroments.

Disadvantages of Kubernetes:

(1) complexity : The learning curve can be steep due to its many components and abstractions, requiring more expertise to manage effectively.

(2) Overhead : Running Kubernetes introduces additional resource overhead, which might not be justified for smaller applications

(3) Configuration Management: Manage configuration can become complicated, in larger environments.

(4) Debugging Challenges : Debugging issues in a distributed system can be more difficult

than in traditional, monolithic architecture.

How Adidas use Kubernetes:

Adidas has adopted Kubernetes to enhance its development and operational efficiencies. Here's how they utilize it:

(a) Microservices Architecture: Adidas leverages Kubernetes to manage its microservice, enabling them to develop, deploy and scale services independently.

(b) Devops Practices: Kubernetes supports Adidas Devops practices allowing for continuous Integration and continuous delivery (CI/CD) pipelines. The facilitates frequents updates and rapid iteration on their applications.

(c) Resource Optimization: By using Kubernetes Adidas can efficiently manage cloud resources, reducing costs while ensuring that applications are responsive to user demand, especially during peak shopping seasons.

(d) Enhanced Collaboration: Kubernetes fosters collaboration between development and operations teams by standardizing deployment processes and improving visibility into application performances.

(e) Global scaling: with a worldwide presence Adidas uses Kubernetes to deploy applications across multiple regions seamlessly ensuring consistent performances for customers everywhere

Q(4) what are Nagios and explain how Nagios are used in E-services

Soln: Nagios is an open-source monitoring system that enables organisations to monitor their IT infrastructure, including servers, networks and applications. It provides real-time monitoring and alerting capabilities to ensure that critical systems are running smoothly and any issues are promptly addressed.

Key features of Nagios:

① monitoring: Tracks network services (HTTP, SMTP, POP3, FTP etc) server resources (CPU, memory, disk usage) and application performance

② Alerts: Sends notifications (via email, SMS, or other methods) when thresholds are breached or services are down.

③ Reporting: Provides historical reports for performance analysis and capacity planning.

④ Customization: Offers plugins to expand functionality and supports integration with third party

systems for automation.
⑤ Scalability: Capable of monitoring large-scale environments.

Use of Nagios in E services:
In E services such as web-based systems or applications, Nagios is used to ensure the availability and performance of critical services. It can monitor the health of web servers, databases, API's and other infrastructure components detecting issues like high response times, system downtimes, or overloads. By doing this, Nagios helps in:
① Minimizing Downtime: Real-time monitoring helps detect and rectify service failures promptly ensuring continuous service availability.

② Proactive Issue Resolution: It alerts administrators before performance degradation affects the user experience enabling them to act preemptively.

③ Service-Level Agreement (SLA) Compliance: By tracking uptime and service quality, Nagios assists businesses in meeting their service-level commitments.

④ Security Monitoring: Detects abnormal behaviour or unauthorized access attempts by monitoring logs and user activities.
Overall, Nagios is widely used in E services to ensure seamless operation and a consistent user experience.

Anushka Shahane
D15A (55)

05/10

ADVANCE DEVOPS
ASSIGNMENT 2

Q1) Create a REST API with the serverless Framework

Soln: The serverless Framework helps you deploy applications to cloud providers like AWS using a simplified configuration.

(1) Install serverless framework: Ensure you have Node.js installed, then install the serverless Framework using npm

```
npm install -g serverless
```

(2) Set up AWS credentials: Serverless uses AWS Lambda and API Gateway so configure your AWS Console

```
aws configure
```

Add your AWS Access Key, Secret key Region etc.

(3) Create a New Service: Create a new project using a Node.js template.

```
serverless create --template aws-nodejs
                  --path rest-api
```

This creates a basic serverless service with a structure including serverless.yml and handler.js file for code.

(4) Define the REST API in serverless.yml:
Edit the serverless.yml to define the REST API endpoints.
For eg: service: rest-api-service

```
provider:
  name: aws
  runtime: nodejs14.x
functions:
  hello:
    handler: handler.hello
    events:
      - http:
          path: hello
          method: get
```

The handler.js file would contain the logic

(5) Deploy the Service: Deploy the API to AWS Lambda and API gateway using: serverless deploy

This deploys infrastructure and you'll get a URL to access API

(6) Testing: Use the URL provided to access your REST API. You can test it with tools like Postman or simply via your browser.

(2) Case Study for Sonarqube.

Soln:

SonarQube helps to automatically review your code for bugs, vulnerabilities, and code smells. The steps below cover Java, Python and Node.js analysis.

(1) Create a SonarQube Profile:
Go to SonarQube Cloud and create an account You can link this account to your Github profile to analyze code directly from repositories.
Create a project in SonarCloud and connect it with your Github repository.

(2) Analyze Code on SonarCloud:
For your Github repository, configure it with SonarCloud. This can be done using CI pipelines (eg. Github Actions) or manually.

Example of Github Actions YAML for SonarCloud

```
name: SonarCloud
on:
  push:
    branches:
      - main
```

```
jobs:
  sonarcloud:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: SonarCloud Scan
        uses: sonarsource/sonarcloud-github-
                                action@v1.4
        with:
          args: >
            -Dsonar.projectKey=my-project-key
            -Dsonar.organization=my-org
            -Dsonar.host.url=https://sonarcloud.io
```

(3) SonarLint setup in Java IDE:

Install SonarLint in IntelliJ IDEA or Eclipse from the plugin marketplace.

Configure it to link with your Sonar cloud account for continuous quality checks.

Once installed SonarLint will analyze your Java code locally for issues.

(4) Python Project Analysis:

① Create a sonar-project.properties file in the root of your python project.

```
sonar.projectKey = my_python_project
sonar.organization = my_org
sonar.sources = .
```

Run the analysis using SonarQube scanner:
sonar-scanner

(5) Nodejs Project Analysis:

For a Node.js project the steps are similar to Python. Configure a sonar-project.properties file and scan the project using sonar-scanner.

Example sonar-project.properties

```
sonar.projectKey = my-nodejs-project
sonar.sources = .
sonar.exclusions = node_modules/**,**/*.test.js
```

Run sonar-scanner to analyze your Node.js project.

Analyze Results:
Just like with Python, the results of the analysis will be uploaded to Sonar Cloud. The report can be accessed from the dashboard which will highlight issues such as missing semicolons, unused variables and more.

Key features:

- Sonancloud allows you to integrate with Github easily and provides a cloud based dashboard for viewing the quality of your projects.

- SonarLint helps developers fix issues in real time as they write code, making it easier to catch issues before they are committed.

- For Python and Node.js, the sonar-project.properties file is essential for configuring the analysis and the sonar-scanner tool helps run the analysis locally.

3) At a large organization, your centralized operations team get many repetitive infrastructure request. You can use Terraform to build a "self-serve" infrastructure model that lets product teams manage their own infrastructure independently. You can create and use Terraform modules that codify the standards for deploying and managing services in your organization, allowing teams to efficiently deploy services in compliance with your organization's practices. Terraform Cloud can also integrate with ticketing systems like ServiceNow to automatically generate new infrastructure requests.

Soln: The goal of this task is to use Terraform to create a reusable infrastructure model enabling teams to deploy and manage their own resources without involving central operations.

① Understand the self serve infrastructure Model:
In large organizations product teams often requests infrastructure resources from a central ops team. This can be repetitive and time-consuming

By using Terraform you can create reusable modules that product teams can use independently to deploy their own infrastructure based on organization standards.

(2) Creating Terraform Modules:

Modules allow you to reuse Terraform code. Create a module that defines a common infrastructure component, such as an EC2 instance.

Teams can use this module in Terraform configurations:

```
module "ec2" {
  source = "./ec2-instance"
  ami-id = "ami-0abcd1234"
  instance_name = "team-app-instance"
}
```

(3) Automating with Terraform Cloud and Ticketing Systems:

Terraform Cloud allows you to collaborate on infrastructure deployments. It can be integrated with a ticketing system like Service Now to automate infrastructure requests.

This integration ensures that infrastructure is deployed in compliance with the organization security and governance standards.

Key Tools to Use:

① Serverless Framework: for deploying REST API's

② SonarQube/SonarCloud: for code quality analysis.

③ SonarLint in IntelliJ/Eclipse for on-the-fly Java Analysis.

④ Terraform: for infrastructure automation and self-service infrastructure.