Advanced DevOps Lab
Experiment 1A

Name: Dev Gaonkar **Class/Roll No.** D15C/12

Aim: To develop a website and host it on your local machine and use a VM on AWS Cloud.

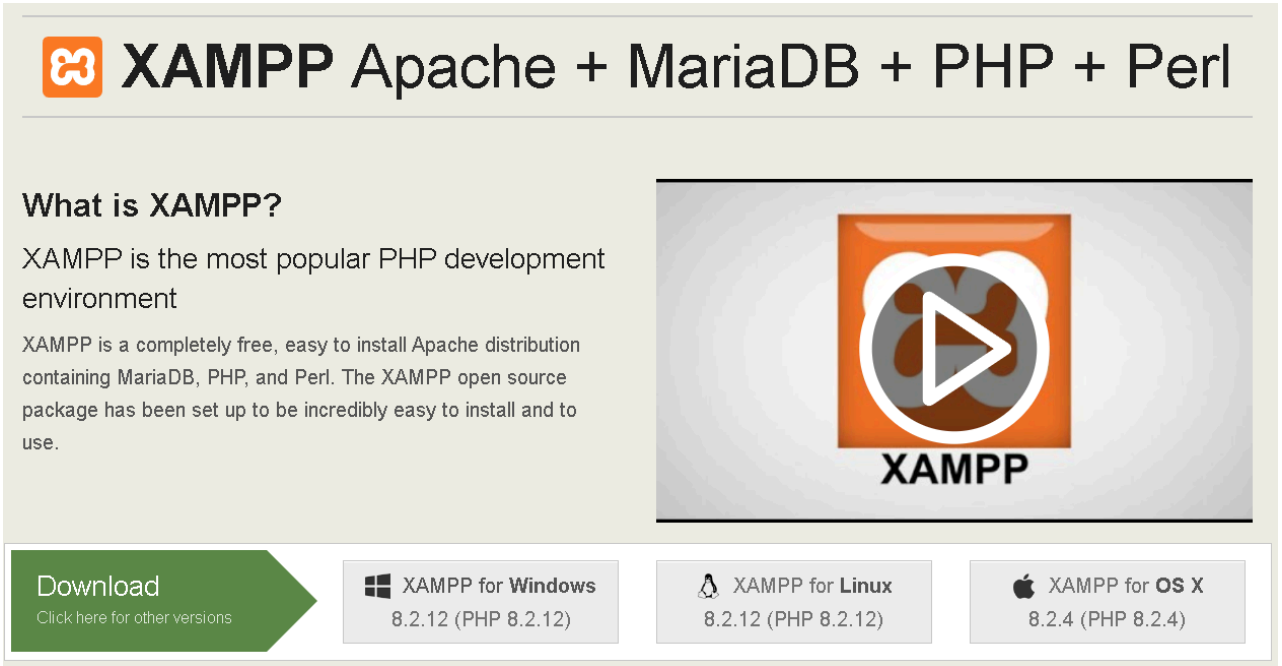
Theory and Prerequisites:

To host a website, we need its contents ready, preferably on a GitHub repository for easy fetching. In this example, we will host a portfolio website from this GitHub Repository. This repository has HTML and CSS, which means there are no build commands needed for hosting. If you are hosting a website using a Javascript framework, for example, the build command will be required for hosting it locally or on the Cloud.

Part 1: Hosting a website on localhost

Steps:

1. Downloading XAMPP.

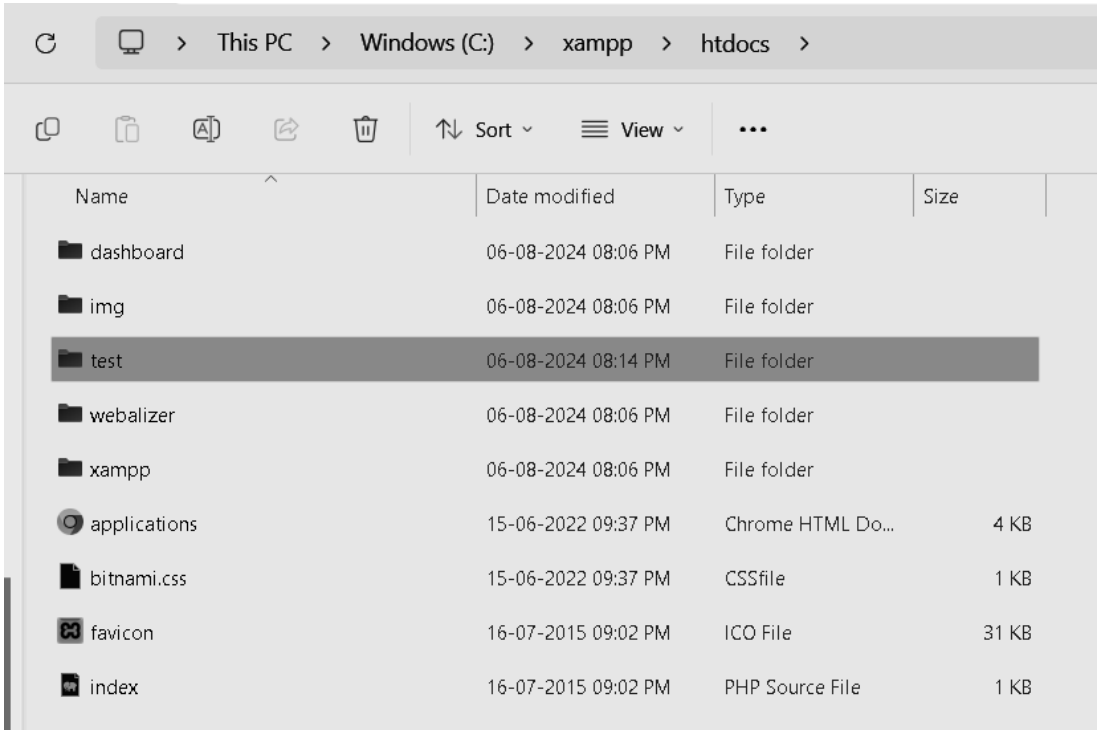


We need to install XAMPP, which creates an apache server for us on our local machine, on which we can host our website. You can also use applications like WAMP or MAM depending on your Operating System.

Please Note: Make sure you select Apache when you install XAMPP, so that you can use it with XAMPP later.

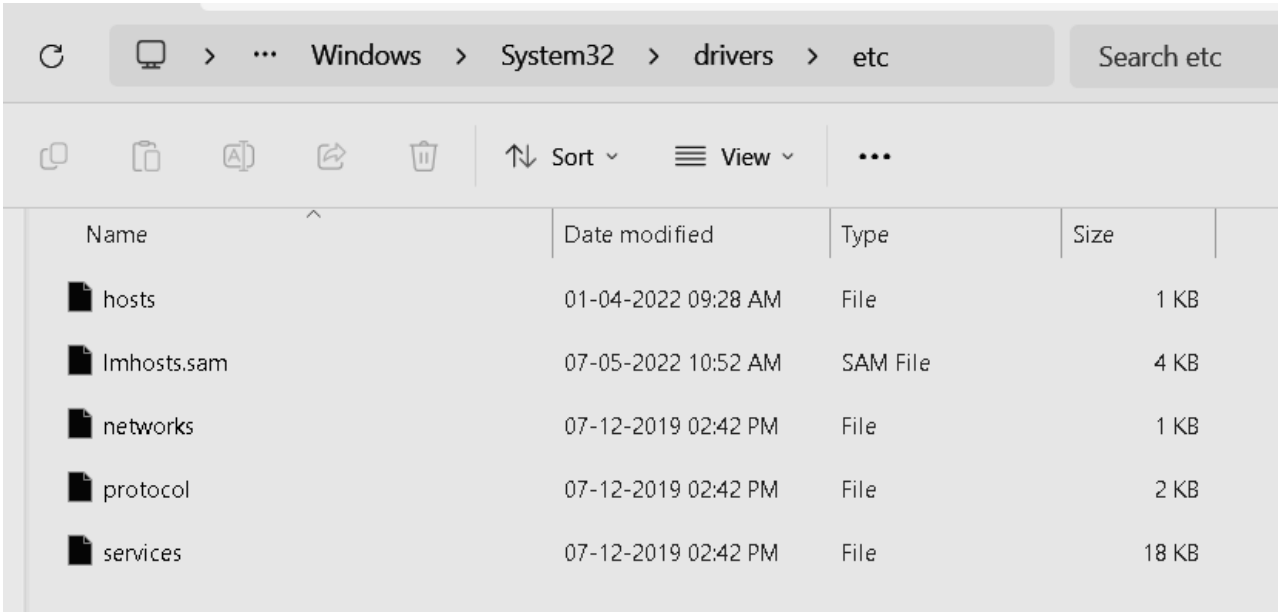
2. Dropping the Code folder in the **htdocs** folder

```
devpg@LAPTOP-7NM7ITJ2 MINGW64 /c/xampp/htdocs
$ git clone https://github.com/GaonkarDev/test.git
Cloning into 'test'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 13 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (13/13), 4.68 KiB | 958.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
```



Go to the xampp root folder, then locate the htdocs folder, such that you are now in *xampp/htdocs*. Now, open the Terminal here and use the ***git clone*** command to clone your code folder in this directory.

3. Update the hosts file to serve localhost as your domain name (will work only on your local machine)

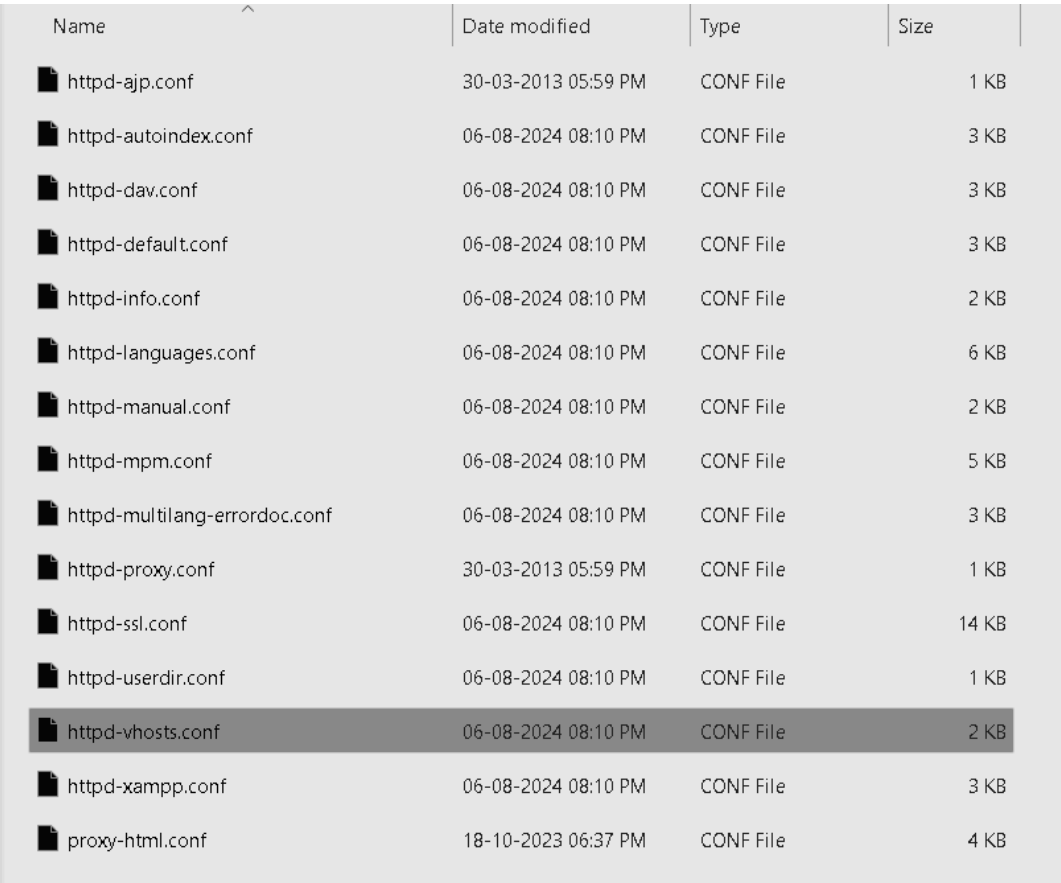


Open the notepad as administrator and open the **hosts** file in *Windows/System32/drivers/etc*. Change the filter to All Files to find the **hosts** file.

```
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost

127.0.0.1 localhost
127.0.0.1 devgaonkar.com
```

Then, on a new line, enter localhost and map it with your desired domain name, in my case, it is devgaonkar.com, as shown above.

4. Update the httpd-vhosts.conf file



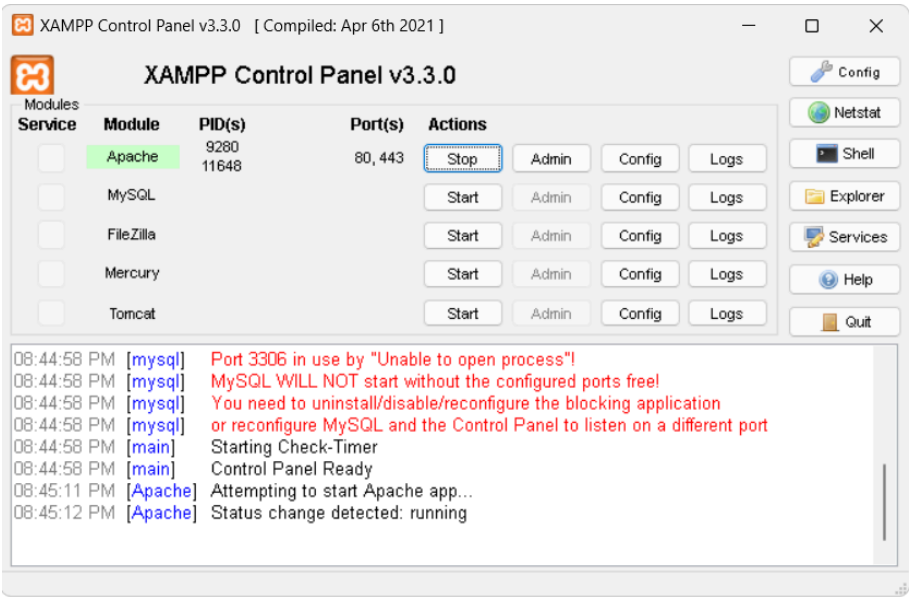
Name	Date modified	Type	Size
httpd-ajp.conf	30-03-2013 05:59 PM	CONF File	1 KB
httpd-autoindex.conf	06-08-2024 08:10 PM	CONF File	3 KB
httpd-dav.conf	06-08-2024 08:10 PM	CONF File	3 KB
httpd-default.conf	06-08-2024 08:10 PM	CONF File	3 KB
httpd-info.conf	06-08-2024 08:10 PM	CONF File	2 KB
httpd-languages.conf	06-08-2024 08:10 PM	CONF File	6 KB
httpd-manual.conf	06-08-2024 08:10 PM	CONF File	2 KB
httpd-mpm.conf	06-08-2024 08:10 PM	CONF File	5 KB
httpd-multilang-errordoc.conf	06-08-2024 08:10 PM	CONF File	3 KB
httpd-proxy.conf	30-03-2013 05:59 PM	CONF File	1 KB
httpd-ssl.conf	06-08-2024 08:10 PM	CONF File	14 KB
httpd-userdir.conf	06-08-2024 08:10 PM	CONF File	1 KB
httpd-vhosts.conf	06-08-2024 08:10 PM	CONF File	2 KB
httpd-xampp.conf	06-08-2024 08:10 PM	CONF File	3 KB
proxy-html.conf	18-10-2023 06:37 PM	CONF File	4 KB

Locate the httpd-vhosts.conf file in xampp/apache/conf/extra. Open this file with your desired text editor, in my case, VSCode.

```
##<VirtualHost *:80>
    ##ServerAdmin webmaster@dummy-host2.example.com
    ##DocumentRoot "C:/xampp/htdocs/dummy-host2.example.com"
    ##ServerName dummy-host2.example.com
    ##ErrorLog "logs/dummy-host2.example.com-error.log"
    ##CustomLog "logs/dummy-host2.example.com-access.log" common
##</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs/test"
    ServerName devgaonkar.com
</VirtualHost>
```

All the code in this file is commented out. Copy the last VirtualHost set which is commented out and paste it out, like so. Inside this tag, make the Document root mapped to the website folder, like so. Change the server name to the mapped domain name, as you did in the hosts folder.
Note: This could be the root folder for you. In my case, I have my HTML in the test folder.

5. Open the XAMPP control panel and start the apache server



6. Open up your domain name on your browser

Hello Dev!

This is test website for Advance DevOps.

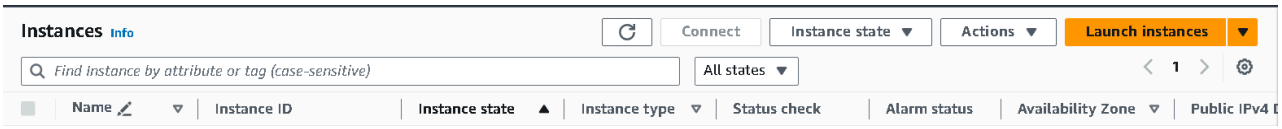
That's it, you have successfully hosted your website locally on your machine. If you have a static IP, you can do this publicly, except you need to buy a domain name for mapping it so that you can make the website public. You can still use services like ngrok to share this website temporarily with your friends, family or colleagues, etc.

Part 2: Hosting a website on a Cloud VM (AWS EC2 Instance)

To host our website on Cloud, we need to set up a Virtual Machine, in AWS EC2, in this case. You'd need an AWS free tier account to proceed.

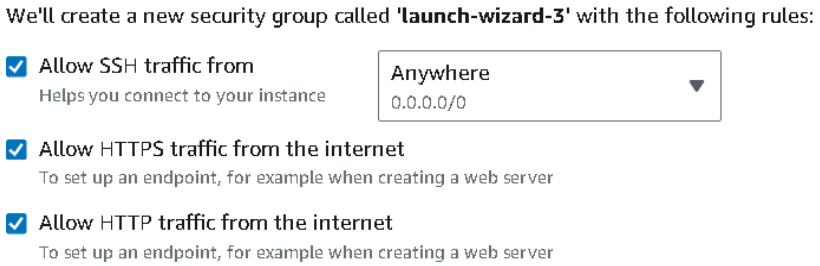
Steps:

1. Open up EC2 Console and Launch a new Instance



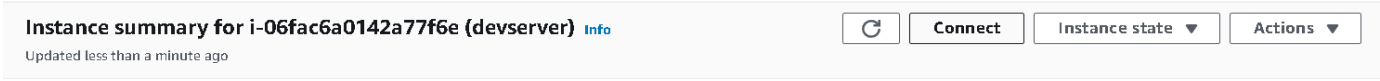
Proceed with everything default.

In Security groups, add these rules so that you can visit the website from anywhere.



Review the changes and launch. Create and download a Keypair file if asked for.

2. Connect to the instance to access the CLI



Click on the Connect button on the top right.

Use EC2 instance connect to directly access the machine on a CLI, opened on the browser.

3. Install Git and HTTPD

Use the following commands to install git and httpd.

```
sudo yum install git -y
sudo yum install httpd -y
```

4. Set up the GitHub Repository to host the website

   Go to var/www/html and clone the GitHub Repository.

```
cd var/www/html
git clone https://github.com/GaonkarDev/test
```

```
[ec2-user@ip-172-31-16-183 html]$ sudo git clone https://github.com/GaonkarDev/test.git
Cloning into 'test'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 13 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (13/13), 4.68 KiB | 4.68 MiB/s, done.
Resolving deltas: 100% (1/1), done.
[ec2-user@ip-172-31-16-183 html]$
```

5. Move all files inside the folder out to the html folder.

```
[ec2-user@ip-172-31-16-183 html]$ sudo mv test/* .
```

   Type the **ls** command just to confirm you have all your html files inside the html directory.

```
[ec2-user@ip-172-31-16-183 html]$ ls
README.md  index.html  test
```

6. Start the httpd server

```
[ec2-user@ip-172-31-16-183 html]$ sudo service httpd start
```

7. View your website

   Create a new tab on your browser and go to the public IP address of your EC2 instance to confirm your website is live. You can find the Public IP address inside the Networking tab of your EC2 console.

Hello Dev!

This is test website for Advance DevOps.

   Your website is live!

Conclusion:

In this experiment, we learned how to host our websites locally on our machines and on AWS EC2 Instances (Cloud).

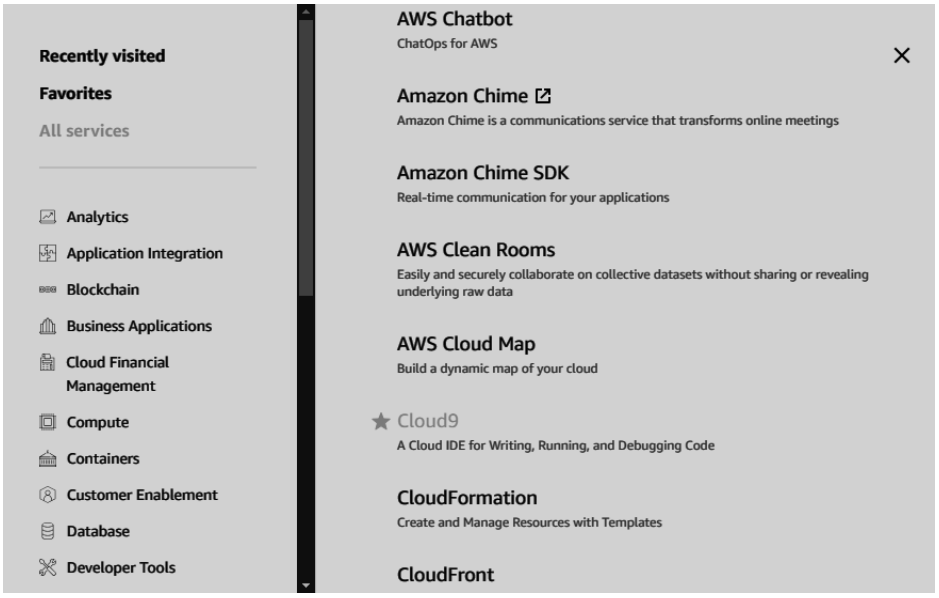
Experiment 1B

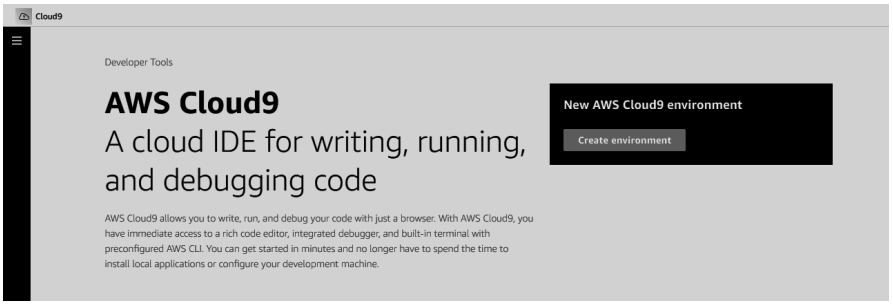
Name: **Dev Gaonkar** Div/Roll no: **D15C/ 12**

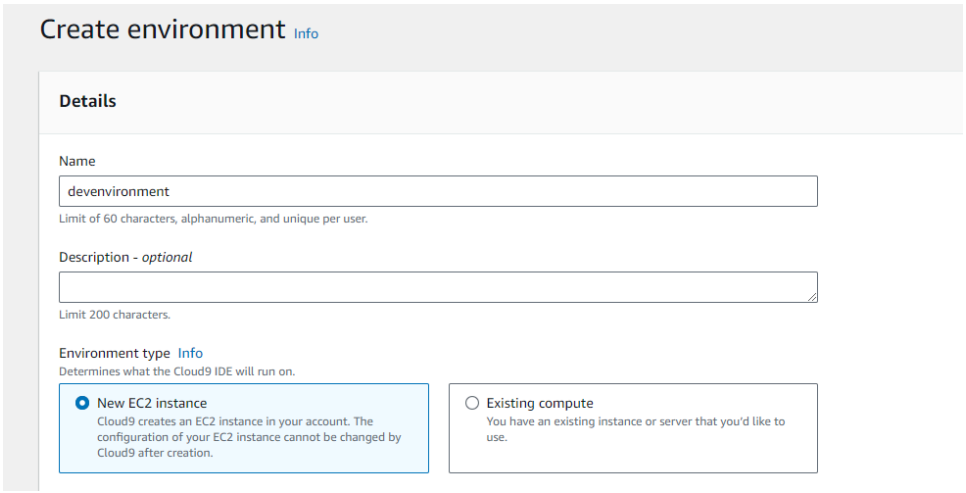
Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

Steps:

1. Open your AWS account and search for Cloud9 service inside Developer tools. Create a new Cloud9 environment by filling in the required details. Make sure you use an EC2 instance to create your environment.







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

Creating devenvironment. This can take several minutes. While you wait, see Best practices for using AWS Cloud9

For capabilities similar to AWS Cloud9, explore AWS Toolkits in your own IDE and AWS CloudShell in the AWS Management Console. Learn more

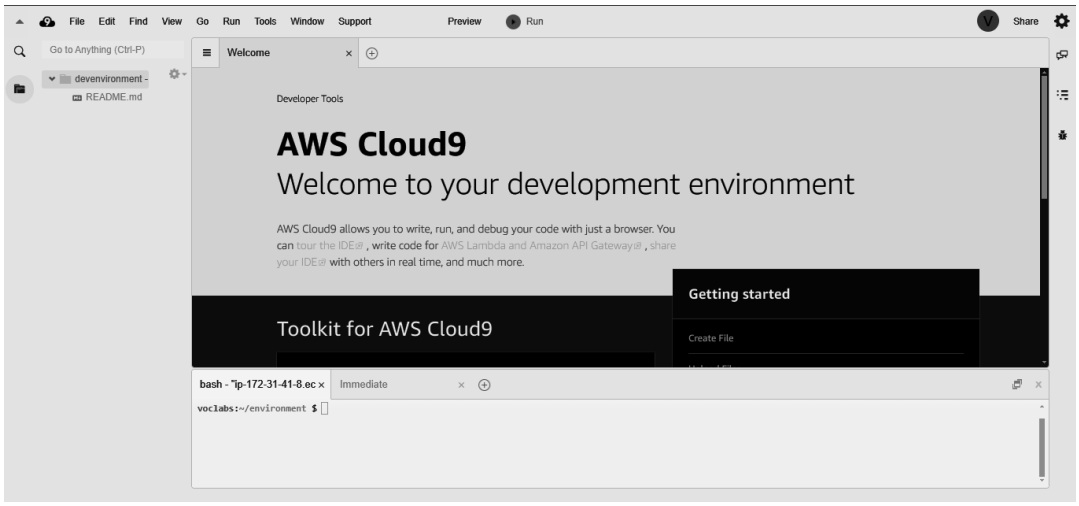
AWS Cloud9 > Environments

Environments (1) Delete | View details | Open in Cloud9 | Create environment

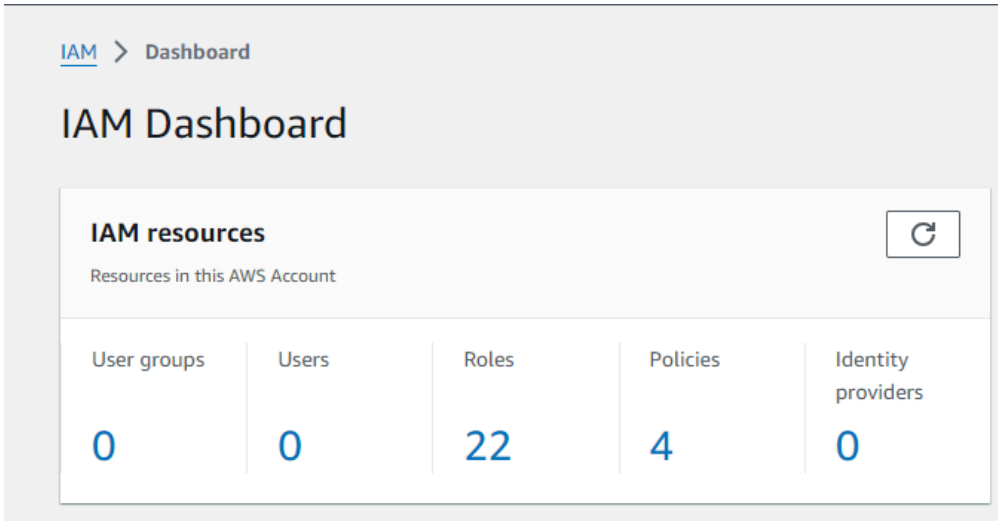
My environments — 1

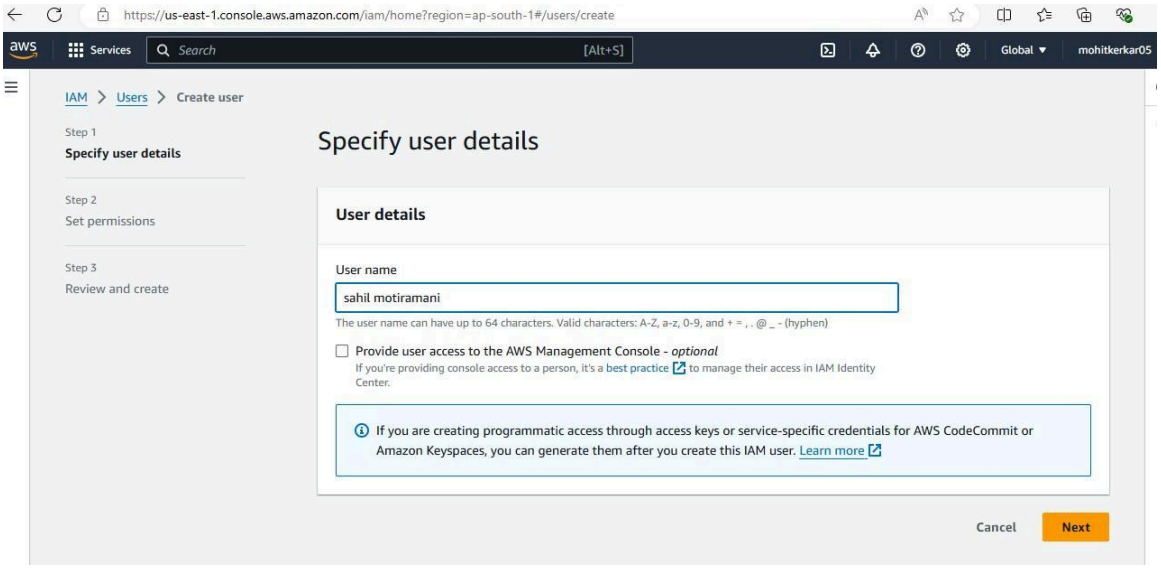
	Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
( )	devenvironment	Open	EC2 instance	Secure Shell (SSH)	Owner	arn:aws:sts::249883209473:assumed-role/voclabs/user3400210=GAONKAR_DEV_PRA

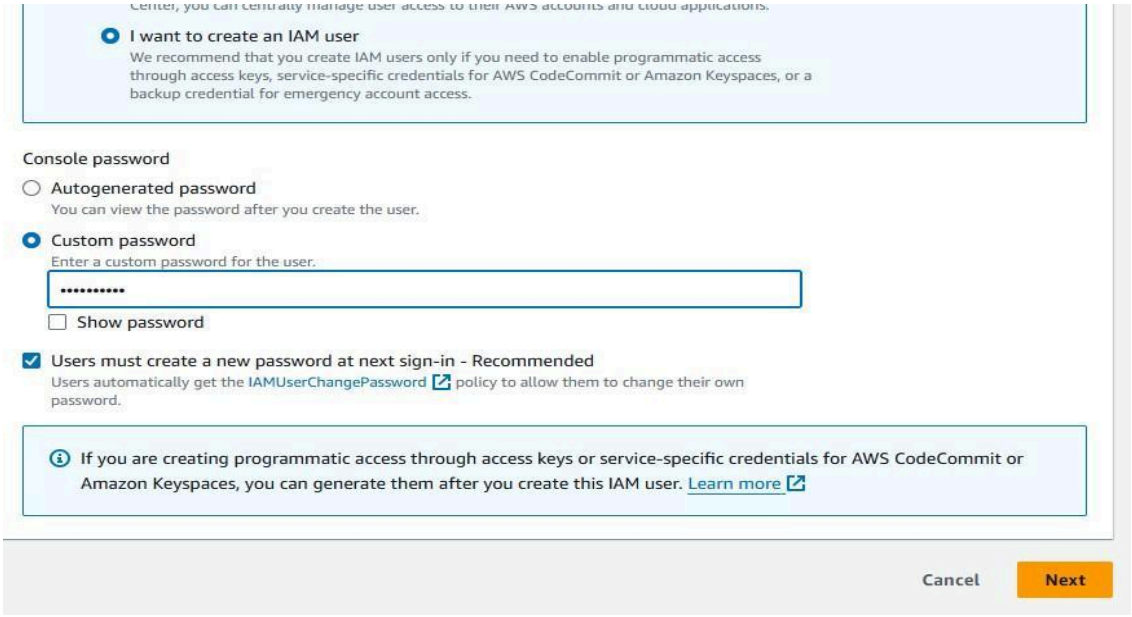
2. We have successfully set up and launched our Cloud9 environment. Over here, we can build and develop programs as per our desire. We are also allowed to collaborate with multiple other users and access shared resources.



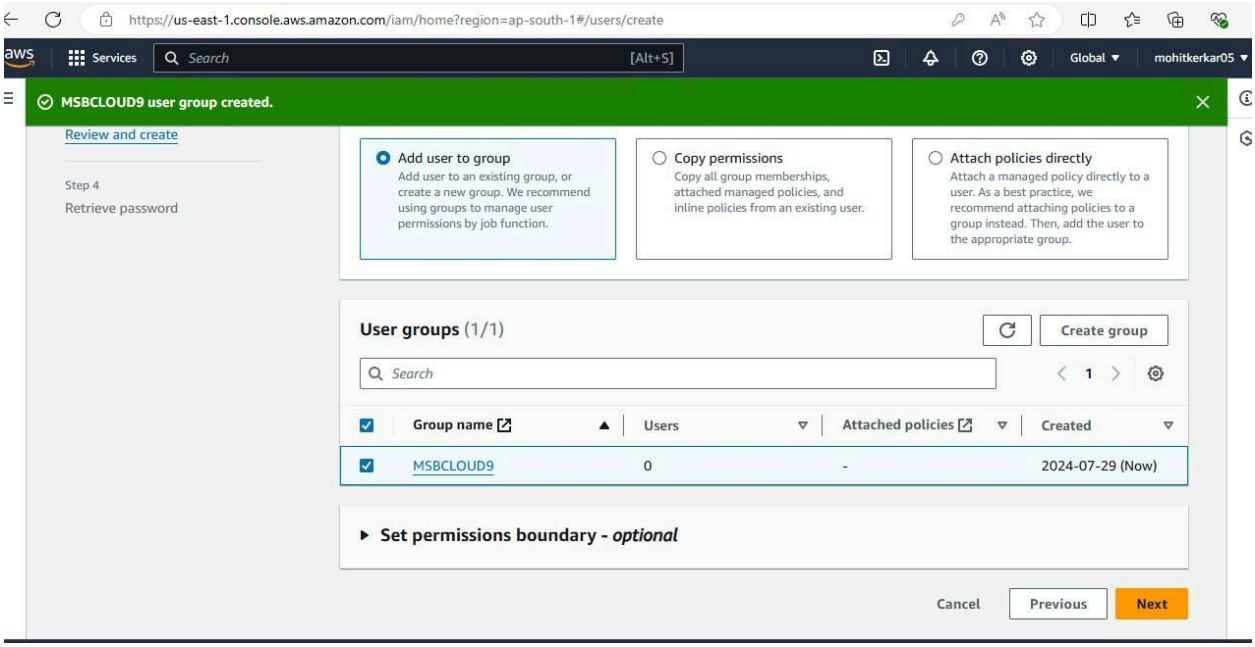
3. Moving on, we are supposed to create a new user. Give a suitable name to the user and decide the password for the same.



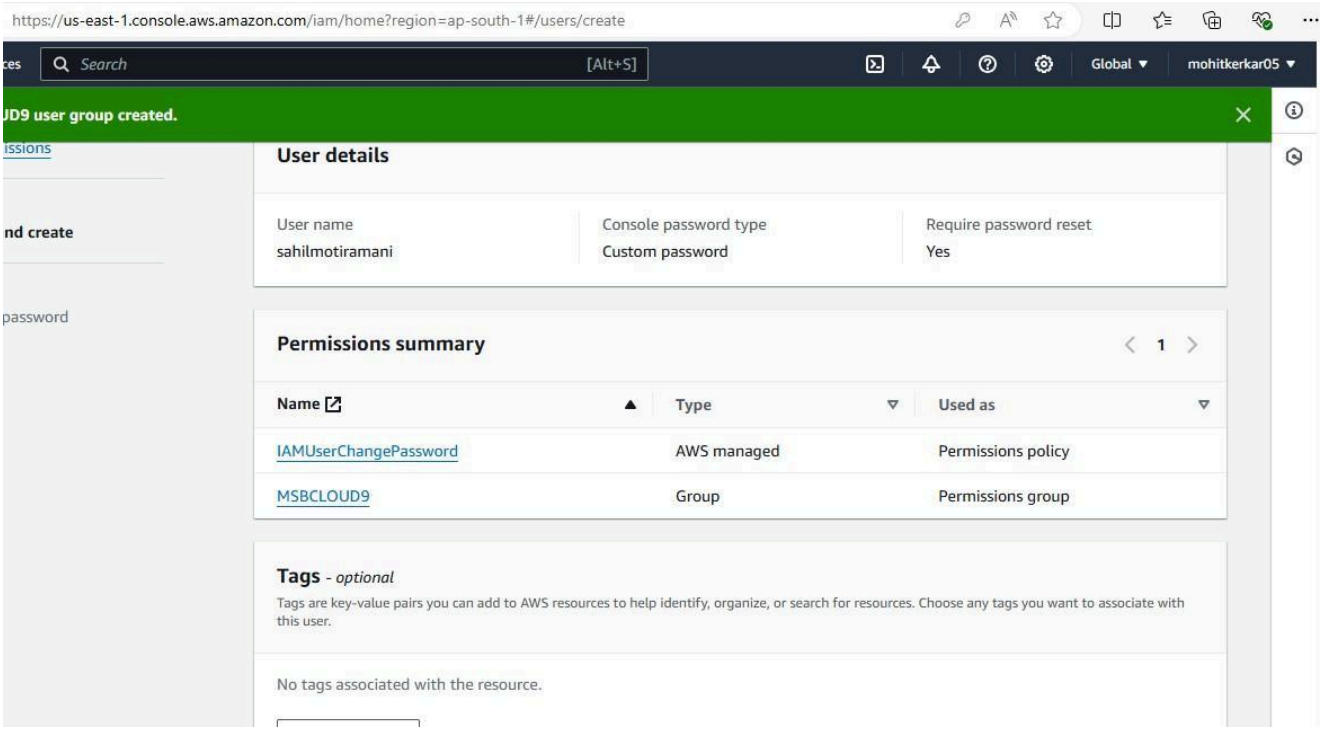




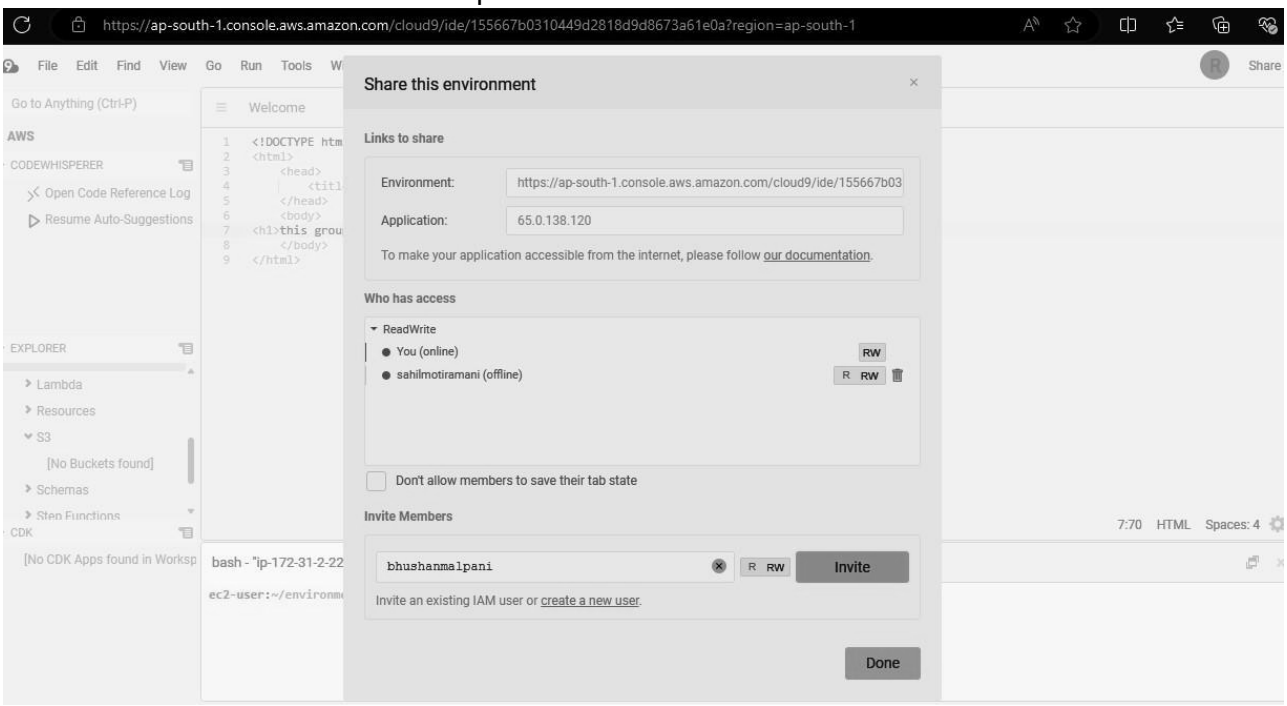
4. Similarly, create a new group and provide a suitable name for them. Include the IAM users in this group together for our convenience, that is, to provide similar kinds of permissions to the entire group rather than an individual user.

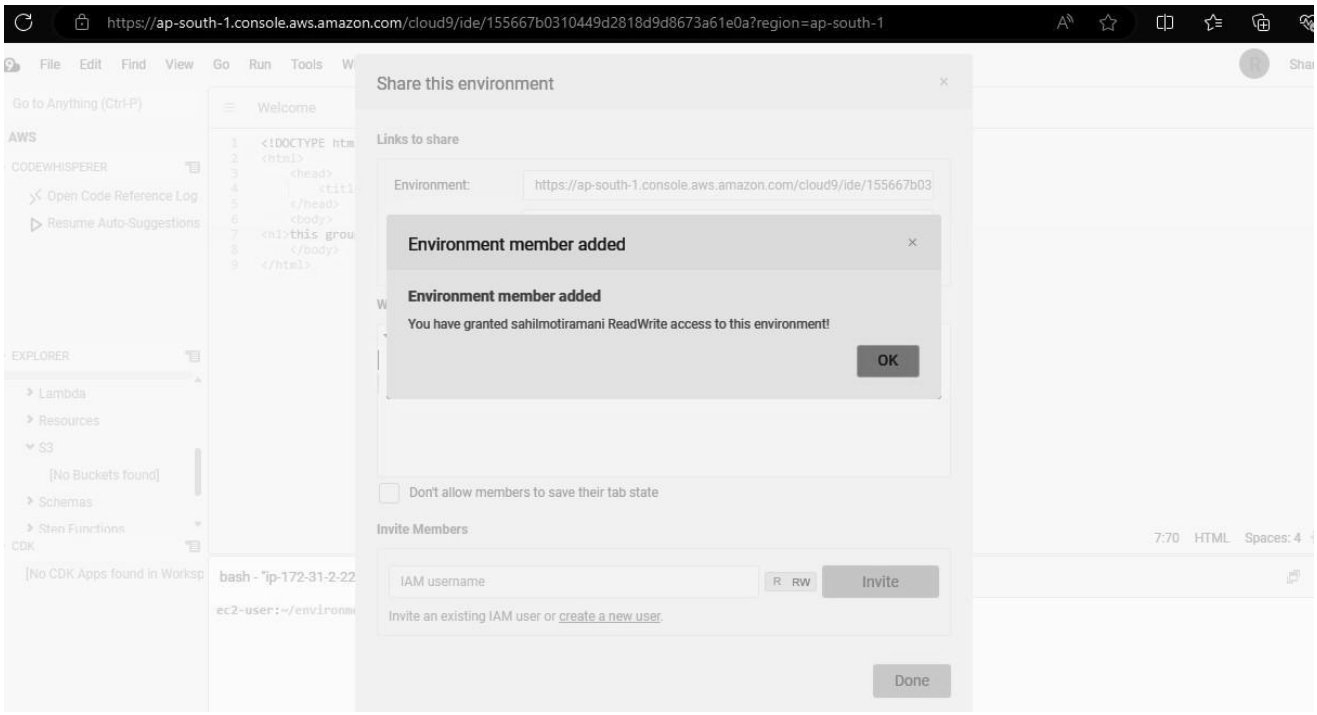


5. The user has successfully been created i.e. There is a custom-made username and a password for the IAM user.



6. Go back to the cloud9 environment. Click on the share this environment option so as to allow other collaborators to access you environment. Include your newly made IAM user in this environment and enable Read/Write permissions for it.





Further, we are supposed to login from another browser using the credentials of the IAM user, to access the shared cloud9 environment with us. These steps could not be completed because Cloud9 services have been disrupted and there is no access to the IAM user from the remote login.

Experiment 2

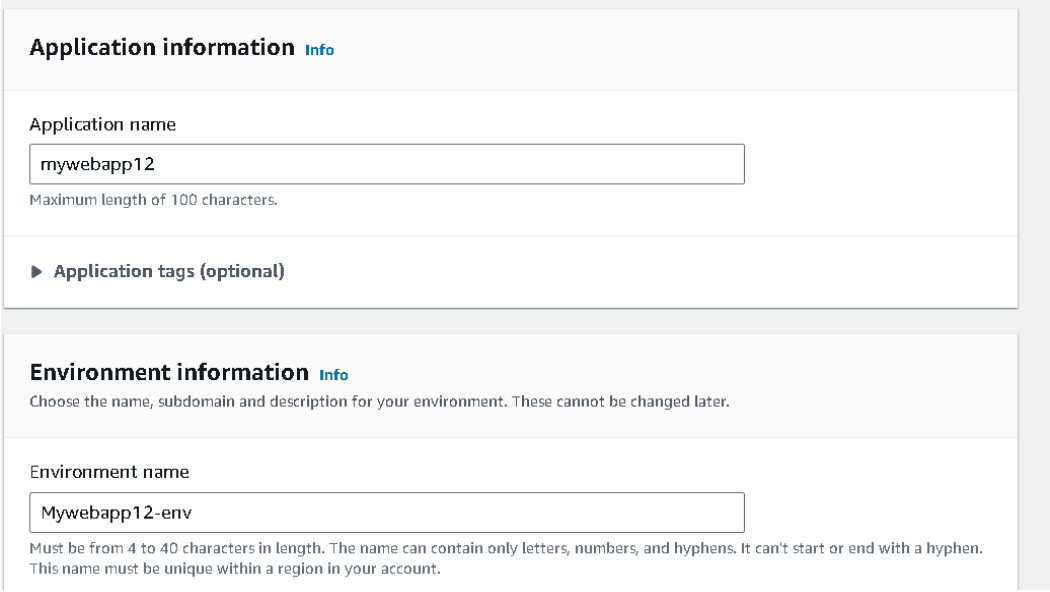
Name: **Dev Gaonkar** Div/Roll no: **D15C/ 12**

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

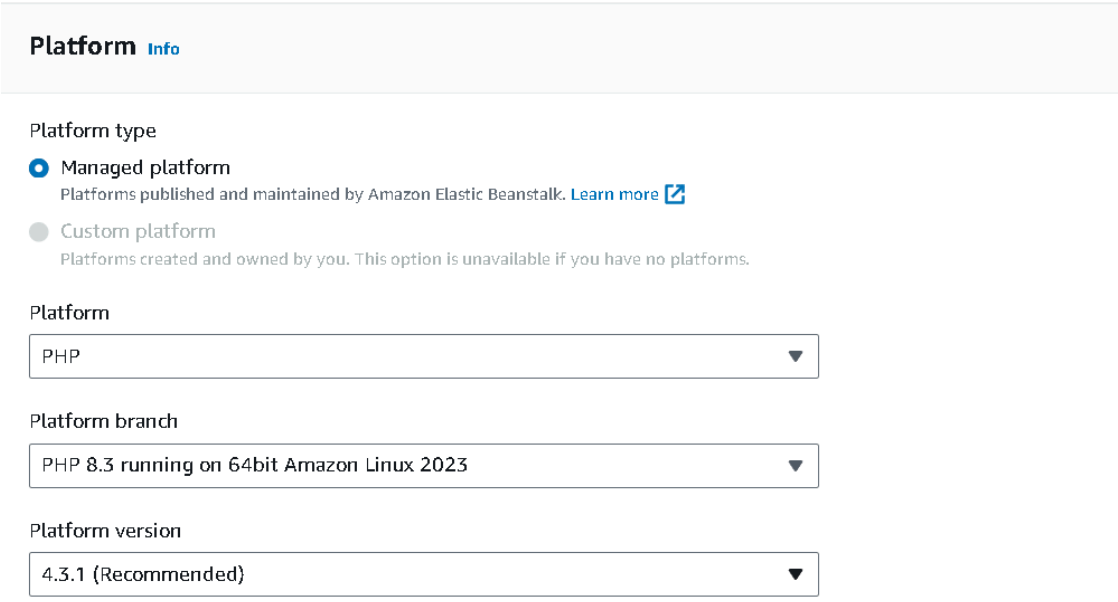
Step 1: Create a Deployment Environment

Your continuous deployment pipeline will need a target environment containing virtual servers, or Amazon EC2 instances, where it will deploy sample code. You will prepare this environment before creating the pipeline.

1. Open up Elastic Beanstalk and name your web app.



2. Choose PHP from the drop-down menu and then click Create Application.



3. Beanstalk creates a sample environment for you to deploy your application.
   By default, it creates an EC2 instance, a security group, an Auto Scaling group, an Amazon S3 Bucket, Amazon CloudWatch alarms and a domain name for your application.

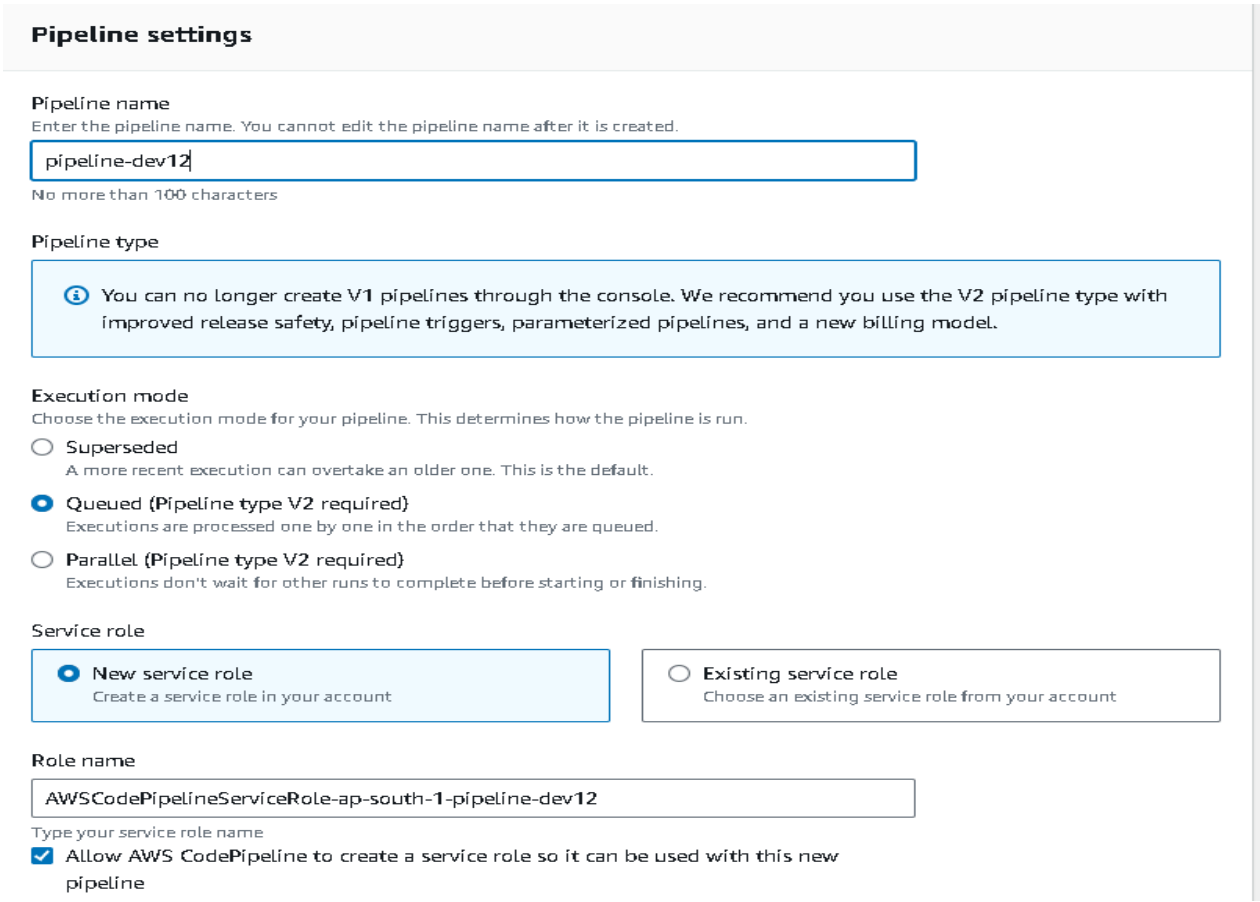
Step 2: Get a copy of your sample code

In this step, we will get the sample code from this GitHub Repository to later host it. The pipeline takes code from the source and then performs actions on it.
For this experiment, as a source, we will use this forked GitHub repository. We can alternatively also use Amazon S3 and AWS CodeCommit.
Go to the repository shared above and simply fork it.

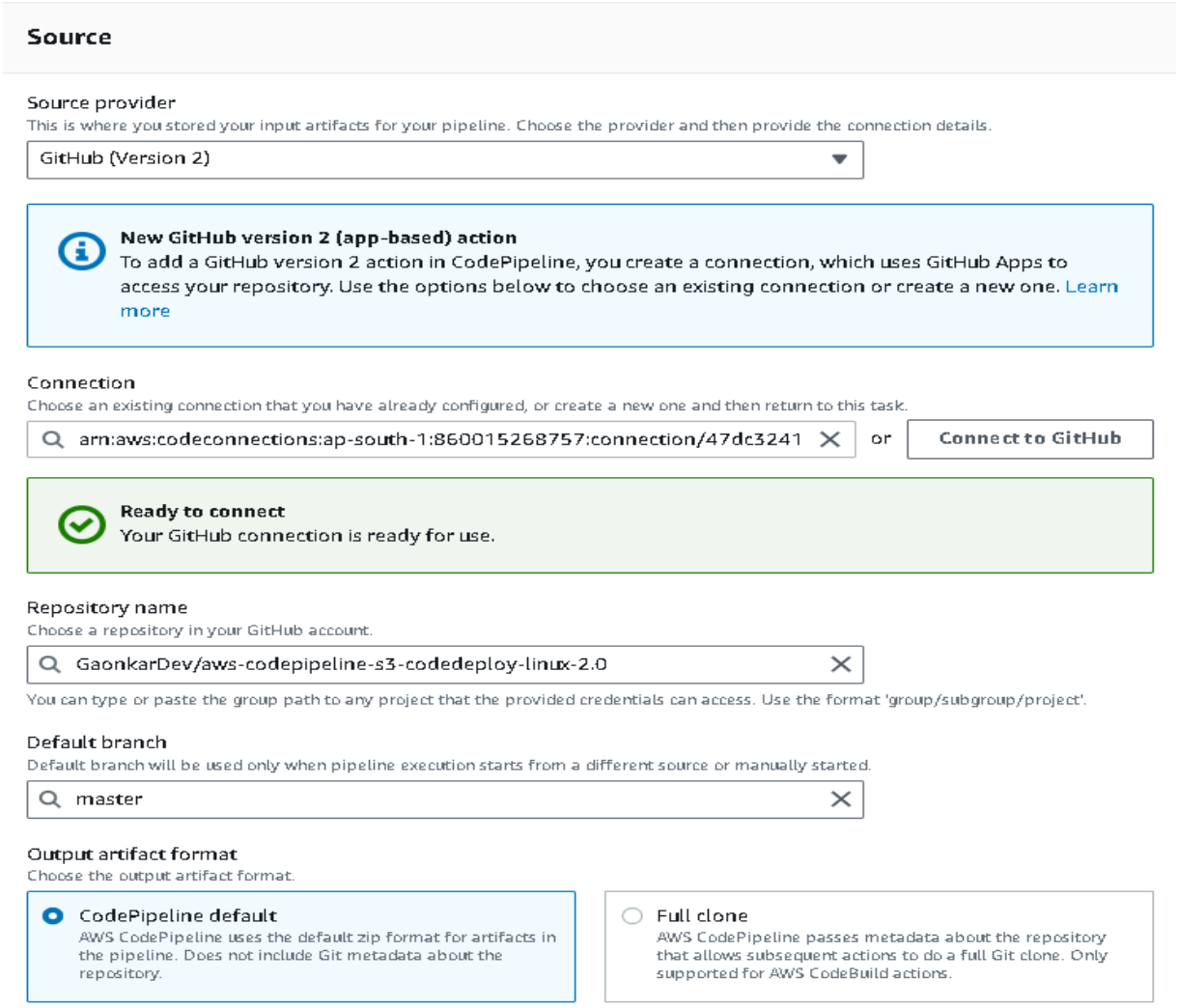
Step 3: Creating a CodePipeline

In this step, we'll create a simple pipeline that has its source and deployment information. In this case, however, we will skip the build stage where you get to plug in our preferred build provider.

1. Go to AWS Developer Tools -> CodePipeline and create a new Pipeline. Fill in the initial settings first.



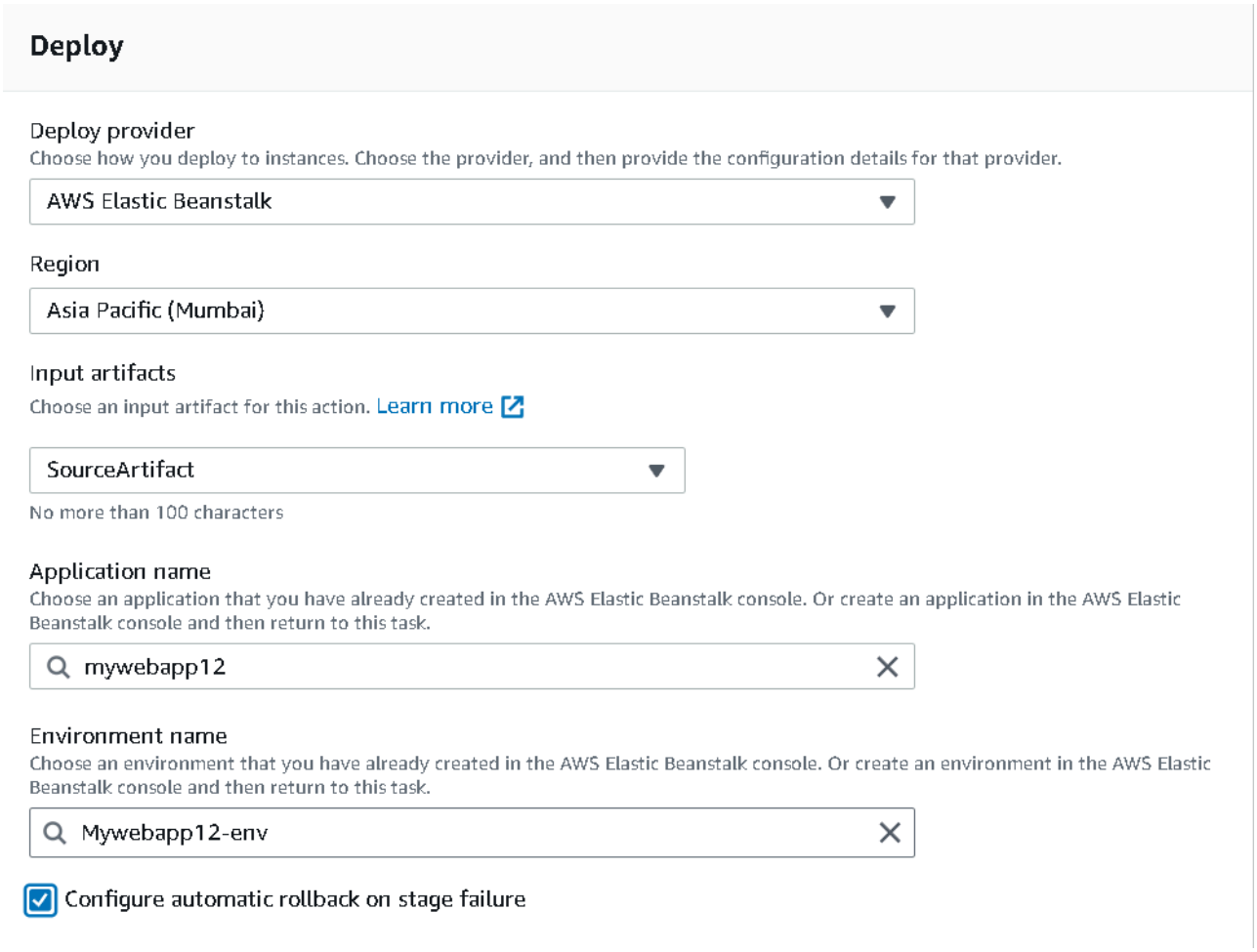
2. In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to AWS by creating a connection. You'd need your GitHub credentials and then you'd need to authorize and install AWS on the forked GitHub Repository.



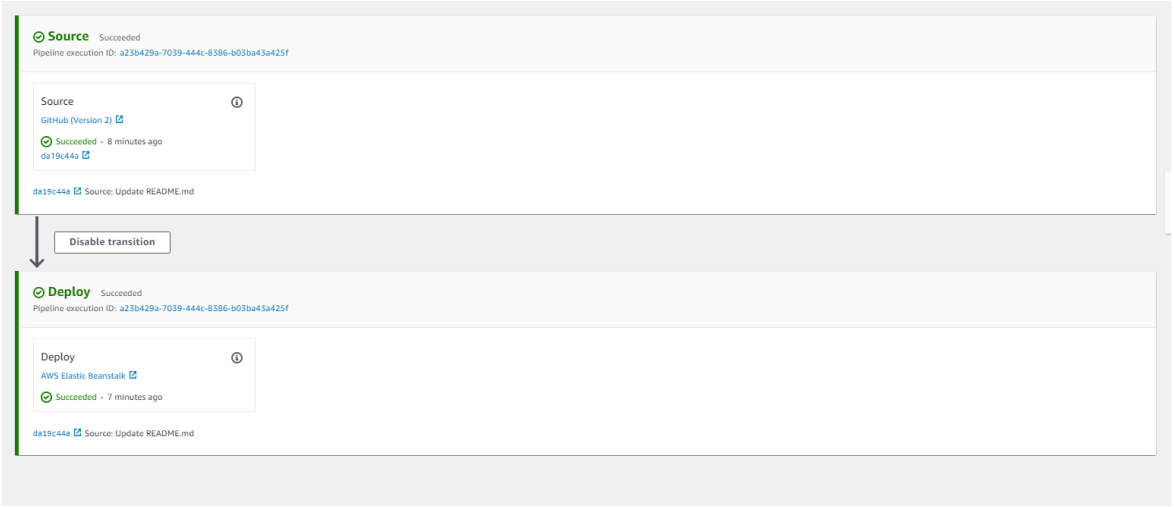
3. Then, simply choose this forked repository and the branch which you will be able to find in the search box. After that, click Continue and skip the build stage. Proceed to the Deployment stage.

Step 4: Deployment

1. Choose Beanstalk as the Deploy Provider, same region as the Bucket and Beanstalk, name and environment name. Click Next, Review and create the pipeline.



2. In a few minutes, we will have our pipeline created. Once we have the success message on the Deploy part, we can go ahead and check our URL provided in the EBS environment.



This is the sample website we just created.

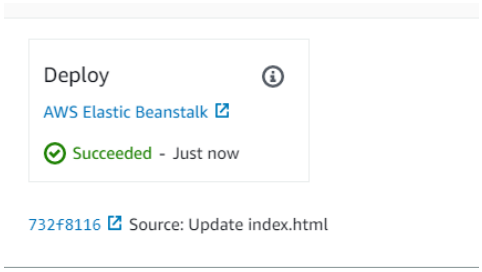
If you can see this, that means that you successfully created an automated software using CodePipeline.

Step 5: Committing changes to update app

1. In this step, we will update the code which we had and make a few changes to the HTML file (keep in mind, this is in our version of the forked repository).

2. In GitHub, open index.html. Then, make changes to either the heading tag or the paragraph tag. Commit these changes on the fly on GitHub.

3. You can view the changes on the website using the same URL, once the deployment section shows success.



4. Check the changes live on your website.

Conclusion:
In this experiment, we learned how to use AWS Elastic Beanstalk Environments to deploy our websites and create a CodePipeline under the AWS Development Console, source data to the Beanstalk using GitHub and finally, make real-time changes to the website just by pushing updates to GitHub.

Name: **Dev Gaonkar** Div/Roll no.: **12**

Advanced DevOps Lab
Experiment:3

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

Theory:

Container-based microservices architectures have profoundly changed the way development and operations teams test and deploy modern software. Containers help companies modernize by making it easier to scale and deploy applications, but containers have also introduced new challenges and more complexity by creating an entirely new infrastructure ecosystem.

Large and small software companies alike are now deploying thousands of container instances daily, and that's a complexity of scale they have to manage. So how do they do it?

Enter the age of Kubernetes.

Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the defacto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

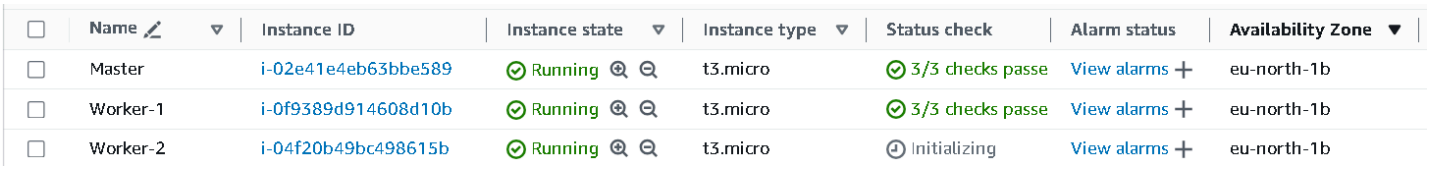
Kubernetes makes it easy to deploy and operate applications in a microservice architecture. It does so by creating an abstraction layer on top of a group of hosts so that development teams can deploy their applications and let Kubernetes manage the following activities:

- Controlling resource consumption by application or team
- Evenly spreading application load across a hosting infrastructure
- Automatically load balancing requests across the different instances of an application
- Monitoring resource consumption and resource limits to automatically stop applications from consuming too many resources and restarting the applications again
- Moving an application instance from one host to another if there is a shortage of resources in a host, or if the host dies
- Automatically leveraging additional resources made available when a new host is added to the cluster
- Easily performing canary deployments and rollbacks

Steps:

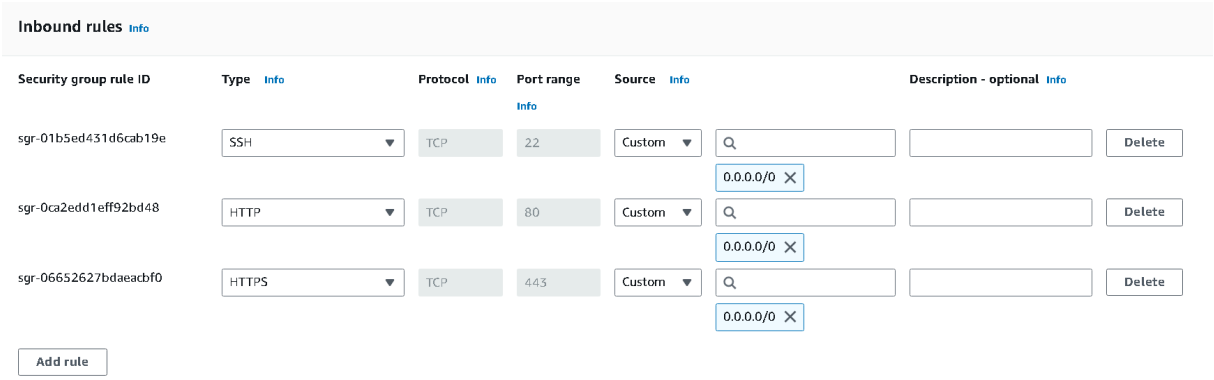
1. Create 3 EC2 Ubuntu Instances on AWS.

(Name 1 as Master, the other 2 as worker-1 and worker-2)



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Master	i-02e41e4eb63bbe589	Running	t3.micro	3/3 checks passe	View alarms	eu-north-1b
Worker-1	i-0f9389d914608d10b	Running	t3.micro	3/3 checks passe	View alarms	eu-north-1b
Worker-2	i-04f20b49bc498615b	Running	t3.micro	Initializing	View alarms	eu-north-1b

2. Edit the Security Group Inbound Rules to allow SSH



Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-01b5ed431d6cab19e	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-0ca2edd1eff92bd48	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-06652627bdaeacbf0	HTTPS	TCP	443	Custom	0.0.0.0/0		Delete

Add rule

3. SSH into all 3 machines

```
kagoran@LAPTOP-7NM7ITJ2:~$ chmod 400 devkeypair.pem
kagoran@LAPTOP-7NM7ITJ2:~$ ls -l devkeypair.pem
-r-------- 1 kagoran kagoran 1678 Sep 14 10:27 devkeypair.pem
```

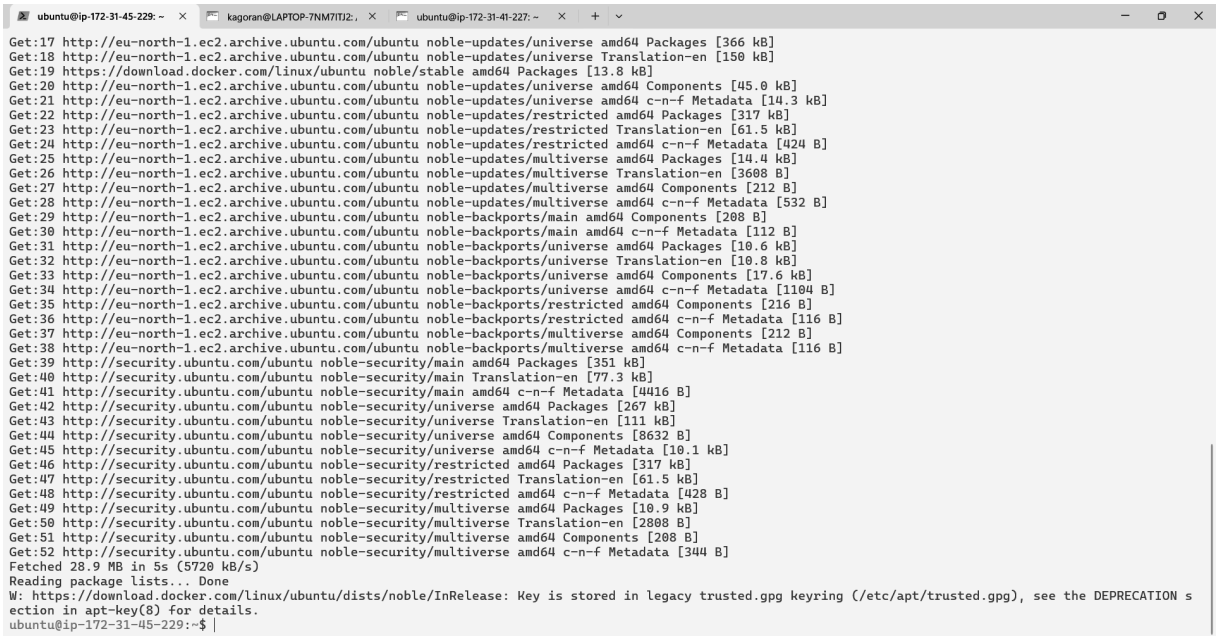
ssh -i <keyname>.pem ubuntu@<public_ip_address>

```
kagoran@LAPTOP-7NM7ITJ2:~$ ssh -i devkeypair.pem ubuntu@13.60.197.8
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Sep 14 04:58:19 UTC 2024

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   22.9% of 6.71GB   Processes:             108
  Memory usage: 21%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.45.229


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

Last login: Sat Sep 14 04:46:00 2024 from 13.48.4.203
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-45-229:~$
```

4. From now on, until mentioned, perform these steps on all 3 machines.

   Install Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce
```



Then, configure cgroup in a daemon.json file.

```
cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart
docker
```

Install Kubernetes on all 3 machines

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-40-255:~$ # Add Kubernetes GPG key
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

# Add Kubernetes repository
sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

# Update package list
sudo apt-get update

# Install kubelet, kubeadm, and kubectl
sudo apt-get install -y kubelet kubeadm kubectl

# Hold the versions of Kubernetes components
sudo apt-mark hold kubelet kubeadm kubectl
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
deb https://apt.kubernetes.io/ kubernetes-xenial main
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-north-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
Ign:6 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release
```

After installing Kubernetes, we need to configure internet options to allow bridging.

```
sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

```
ubuntu@ip-172-31-45-229:~$ # Disable swap
sudo swapoff -a

# Allow bridging for iptables
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

# Apply sysctl changes
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
ubuntu@ip-172-31-45-229:~$
```

5. Perform this **ONLY on the Master machine**

Initialize the Kubecluster

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
--ignore-preflight-errors=all
```

```
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 172.31.45.229:6443 --token s9zq75.bsi7js5f62ridulc \
        --discovery-token-ca-cert-hash sha256:91eae090fdd49337bf70d5bf7478e60bc85820d0996651871129a082db6fa8f1
ubuntu@ip-172-31-45-229:~$
```

Copy the join command and keep it in a notepad, we'll need it later.

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
```

Then, add a common networking plugin called flammel file as mentioned in the code.

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/
k ube-flannel.yml
```

```
ubuntu@ip-172-31-45-229:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.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
```

Check the created pod using this command

Now, keep a watch on all nodes using the following command

```
watch kubectl get nodes
```

6. Perform this **ONLY on the worker machines**

```
sudo kubeadm join <ip> --token <token> \
        --discovery-token-ca-cert-hash <hash>
```

Now, notice the changes on the master terminal

```
Every 2.0s: kubectl get nodes                                                    ip-172-31-45-229: Sat Sep 14 12:19:42 20
24
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-45-229   Ready    control-plane   28m   v1.31.1
```

That's it, we now have a Kubernetes cluster running across 3 AWS EC2 Instances. This cluster can be used to further deploy applications and their loads being distributed across these machines.

Conclusion:

In this experiment, we set up a Kubernetes cluster across three AWS EC2 instances. Docker and Kubernetes components were successfully installed on each instance. The master node was initialized, and the Flannel network plugin was applied. While the master node is functioning correctly, worker nodes encountered issues joining the cluster, likely due to configuration or network problems. To complete the setup, further troubleshooting is needed on the worker nodes to resolve connectivity issues. Once resolved, the cluster will be fully operational, allowing for scalable management of containerized applications.

Name: DEV GAONKAR Div/Roll no.: **D15C/12**

Advanced DevOps Lab
Experiment 4

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

Theory:

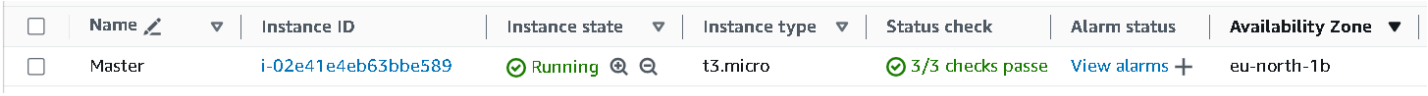
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes Deployment

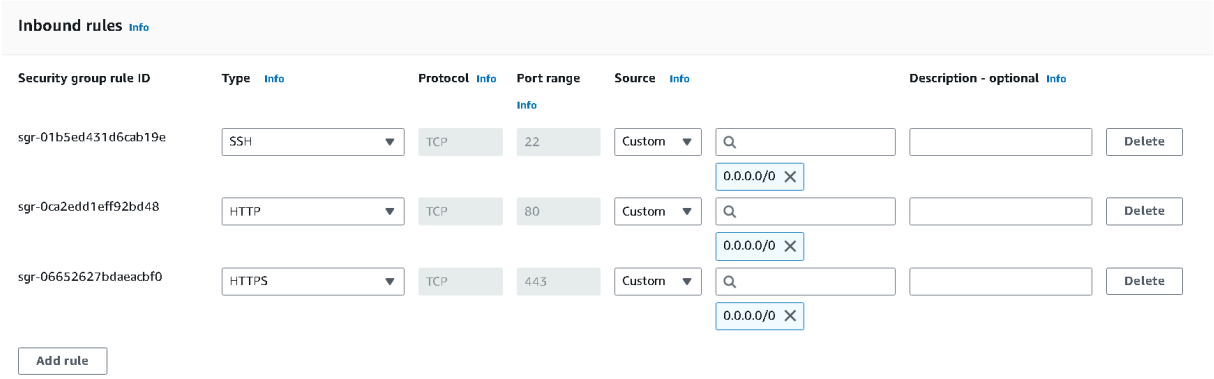
A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

Steps:

1. Create an EC2 Ubuntu Instance on AWS.



2. Edit the Security Group Inbound Rules to allow SSH



3. SSH into the machine

ssh -i <keyname>.pem ubuntu@<public_ip_address>

```
kagoran@LAPTOP-7NM7ITJ2:~$ chmod 400 devkeypair.pem
kagoran@LAPTOP-7NM7ITJ2:~$ ls -l devkeypair.pem
-r-------- 1 kagoran kagoran 1678 Sep 14 10:27 devkeypair.pem
```

```
kagoran@LAPTOP-7NM7ITJ2:~$ ssh -i devkeypair.pem ubuntu@13.60.197.8
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Sep 14 04:58:19 UTC 2024

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   22.9% of 6.71GB   Processes:             108
  Memory usage: 21%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.45.229


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

Last login: Sat Sep 14 04:46:00 2024 from 13.48.4.203
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-45-229:~$
```

4. Install Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce
```

```
Get:17 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [366 kB]
Get:18 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [150 kB]
Get:19 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [13.8 kB]
Get:20 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:21 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [14.3 kB]
Get:22 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [317 kB]
Get:23 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [61.5 kB]
Get:24 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 c-n-f Metadata [424 B]
Get:25 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.4 kB]
Get:26 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3608 B]
Get:27 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
Get:28 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
Get:29 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:30 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:31 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:32 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.8 kB]
Get:33 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:34 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:35 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:36 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:37 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:38 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:39 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [351 kB]
Get:40 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [77.3 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4416 B]
Get:42 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [267 kB]
Get:43 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [111 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:45 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.1 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [317 kB]
Get:47 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [61.5 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Fetched 28.9 MB in 5s (5720 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.
ubuntu@ip-172-31-45-229:~$
```

Then, configure cgroup in a daemon.json file.

```
cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

5. Install Kubernetes

```
sudo apt-get update
# apt-transport-https may be a dummy package; if so, you can skip that
package
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
# If the directory `/etc/apt/keyrings` does not exist, it should be created
before the curl command, read the note below.
# sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg
--dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
# This overwrites any existing configuration in
/etc/apt/sources.list.d/kubernetes.list
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
sudo systemctl enable --now kubelet
```

```
ubuntu@ip-172-31-40-255:~$ # Add Kubernetes GPG key
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

# Add Kubernetes repository
sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

# Update package list
sudo apt-get update

# Install kubelet, kubeadm, and kubectl
sudo apt-get install -y kubelet kubeadm kubectl

# Hold the versions of Kubernetes components
sudo apt-mark hold kubelet kubeadm kubectl
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
deb https://apt.kubernetes.io/ kubernetes-xenial main
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-north-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
Ign:6 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release
```

After installing Kubernetes, we need to configure internet options to allow bridging.

```
sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a
/etc/sysctl.conf
sudo sysctl -p
```

```
E: No packages found
ubuntu@ip-172-31-45-229:~$ # Disable swap
sudo swapoff -a

# Allow bridging for iptables
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

# Apply sysctl changes
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
ubuntu@ip-172-31-45-229:~$
```

6. Initialize the Kubecluster

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

```
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 172.31.45.229:6443 --token s9zq75.bsi7js5f62ridulc \
        --discovery-token-ca-cert-hash sha256:91eae090fdd49337bf70d5bf7478e60bc85820d0996651871129a082db6fa8f1
ubuntu@ip-172-31-45-229:~$
```

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
```

Then, add a common networking plugin called flannel as mentioned in the code.

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/
k ube-flannel.yml
```

```
ubuntu@ip-172-31-45-229:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.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
```

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
```

```
ubuntu@ip-172-31-45-229:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-45-229:~$
```

Use ‘kubectl get pods’ to verify if the deployment was properly created and the pod is working correctly.

```
ubuntu@ip-172-31-45-229:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-krhbv    0/1     Pending   0          2m29s
nginx-deployment-d556bf558-mhlm2    0/1     Pending   0          2m29s
ubuntu@ip-172-31-45-229:~$
```

Next up, create a name alias for this pod.

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

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

```
kubectl port-forward $POD_NAME 8080:80
```

9. Verify your deployment

Open up a new terminal and ssh to your EC2 instance.

Then, use this curl command to check if the Nginx server is running.

```
curl --head http://127.0.0.1:8080
```

```
ubuntu@ip-172-31-45-229:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Sat, 14 Sep 2024 7:20:53 GMT
Content-Type: text/html
Content-Length: 612
Connection: keep-alive
ETag: "5c0692e1-265"
Accept-Ranges: bytes
```

If the response is 200 OK and you can see the Nginx server name, your deployment was successful.

We have successfully deployed our Nginx server on our EC2 instance.

Conclusion:

In this experiment, we successfully installed Kubernetes and Docker on an AWS EC2 Ubuntu instance, configured essential settings, and initialized a Kubernetes cluster. We deployed an Nginx server using a Kubernetes Deployment and applied a Flannel networking plugin for pod communication. By verifying the pod status and forwarding ports, we accessed the Nginx server locally. The successful `200 OK` response from the curl command confirmed that the deployment was operational. This setup demonstrated fundamental Kubernetes operations, including cluster management, application deployment, and verification, showcasing Kubernetes' power in orchestrating containerized applications efficiently.

Experiment 5

Name: **Dev Gaonkar** Div/Roll no: **D15C/12**

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

Theory:

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. This includes low-level components such as compute instances, storage, and networking, as well as high level components such as DNS entries, SaaS features, etc.

Terraform can manage infrastructure on multiple cloud platforms. Terraform's state allows you to track resource changes throughout your deployments. You can commit your configurations to version control to safely collaborate on infrastructure. Terraform plugins called providers let Terraform interact with cloud platform and other services via their application programming interfaces (APIs).

Steps:

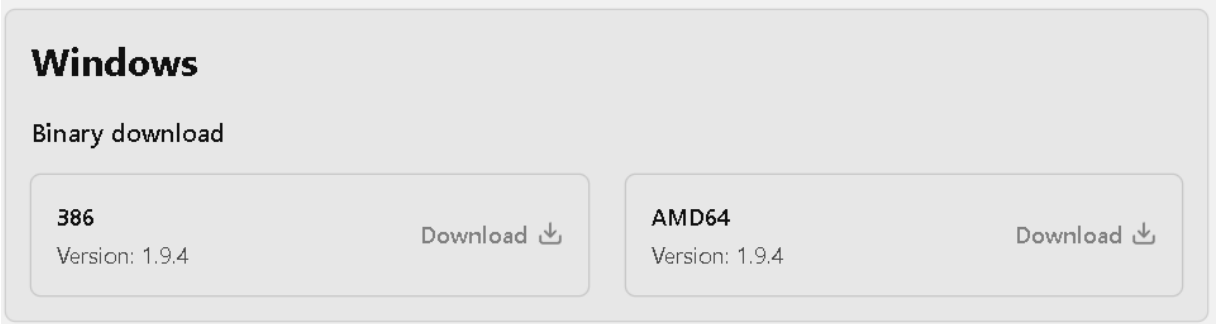
Installation and Configuration of Terraform in Window

Step 1: Download terraform

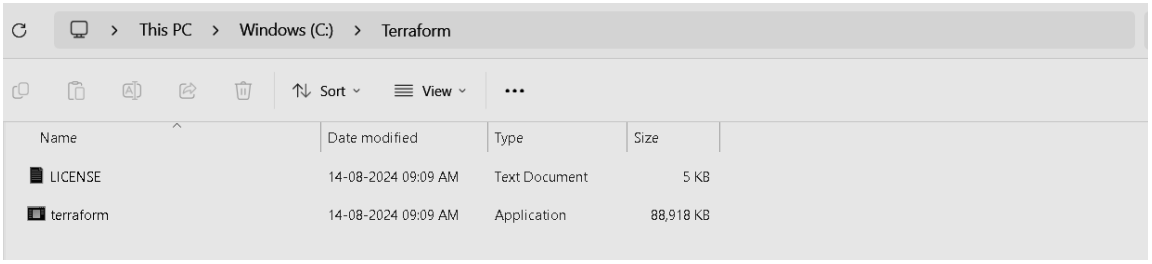
To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website

website:https://www.terraform.io/downloads.html

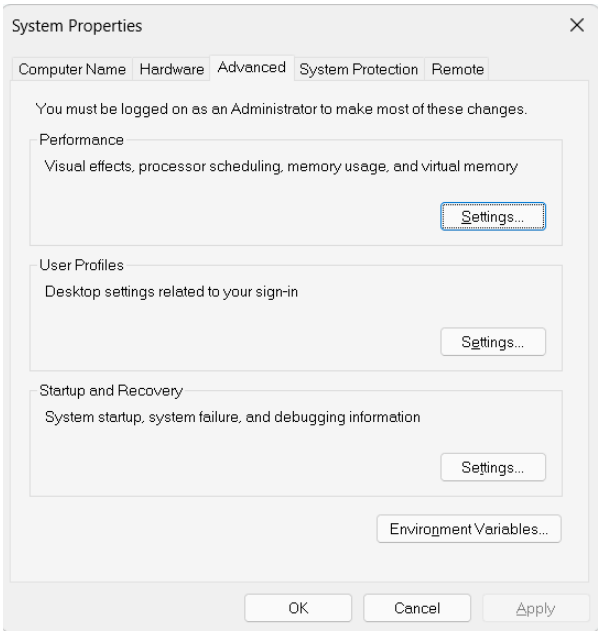
Select the Operating System Windows followed by either 32bit or 64 bit based on your OS type.

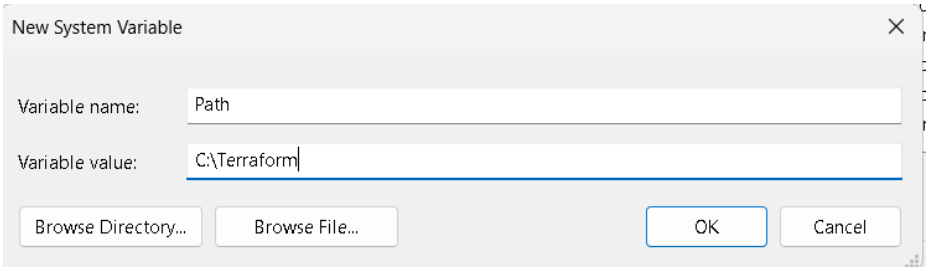


Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform directory



Step 3: Set the System path for Terraform in Environment Variables





Step 4: Open PowerShell with Admin Access

Step 5 : Open Terraform in PowerShell and check its functionality

```
PS C:\WINDOWS\system32> 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
  version       Show the current Terraform version
  workspace     Workspace management

Global options (use these before the subcommand, if any):
  -chdir=DIR    Switch to a different working directory before executing the
                given subcommand.
  -help         Show this help output, or the help for a specified subcommand.
  -version      An alias for the "version" subcommand.
PS C:\WINDOWS\system32>
```

Note: If any error comes, then please recheck or set the path of Terraform in the Environment variable again.

Experiment No.: 6

Name: Dev Gaonkar Div/Roll no.: D15C/12

Implementation:

A. Creating docker image using terraform

Prerequisite:

1) Download and Install Docker Desktop from https://www.docker.com/

Step 1: Check the docker functionality

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

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

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

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

Now, 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: Firstly create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then create a new docker.tf file using Atom editor and write the followingcontents into it to create a Ubuntu Linux container.

Script:

```
terraform
   { required_providers
   {docker = {
      source = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}
 provider "docker" {
  host = "npipe:////.//pipe//docker_engine" }
  # Pulls the image
resource "docker_image" "ubuntu"
   {name = "ubuntu:latest"
}
 # Create a container
resource "docker_container" "foo" {
   image =
  docker_image.ubuntu.image_idname =
  "foo"
  }
```

docker.tf

```
1  terraform
2  { required_providers
3  {docker = {
4  source = "kreuzwerker/docker" version = "2.21.0"
5  }
6  }
7  }
8  provider "docker" {
9  host = "npipe:////.//pipe//docker_engine" }
10 # Pulls the image
11 resource "docker_image" "ubuntu" {name = "ubuntu:latest"
12 }
13 # Create a container
14 resource "docker_container" "foo" { image =
15 docker_image.ubuntu.image_idname = "foo"
16 }
17
```

Step 3: Execute Terraform Init command to initialize the resources

```
PS C:\Terraform Scripts\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.
PS C:\Terraform Scripts\Docker>
```

Step 4: Execute Terraform plan to see the available resources

```
PS C:\Terraform Scripts\Docker> terraform plan

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + 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.

─────────────────────────────────────────────────────────────────────────────

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

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"**

```
  # 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: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Creation complete after 22s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
```

Docker images, After Executing Apply step:

```
PS C:\Terraform Scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   2 weeks ago   78.1MB
PS C:\Terraform Scripts\Docker>
```

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

```
PS C:\Terraform Scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]

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_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, 1 to destroy.

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

  Enter a value: yes

docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
PS C:\Terraform Scripts\Docker>
```

Docker images After Executing Destroy step

```
PS C:\Terraform Scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS C:\Terraform Scripts\Docker>
```

Name: **Dev Gaonkar** Div/Roll no.: **D15C/12**

Experiment 7

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

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 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.

Integrating Jenkins with SonarQube:

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

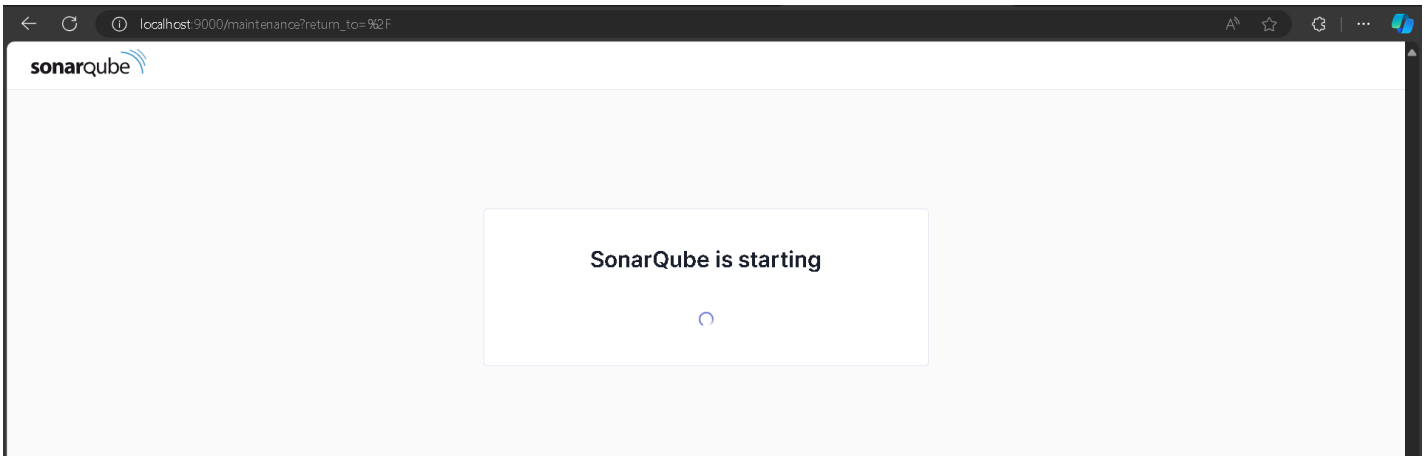
Steps to integrate Jenkins with SonarQube

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 -

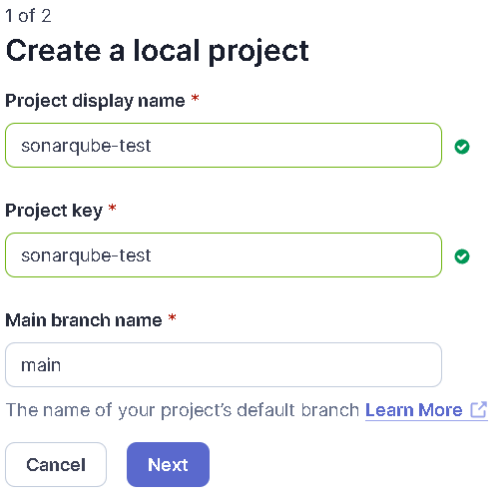
```
PS C:\Users\devpg> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
Status: Downloaded newer image for sonarqube:latest
60de6878d06142545 00f608f43d81a3430585dc282e74225fe2a8fa237ee9d76
PS C:\Users\devpg>
```

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



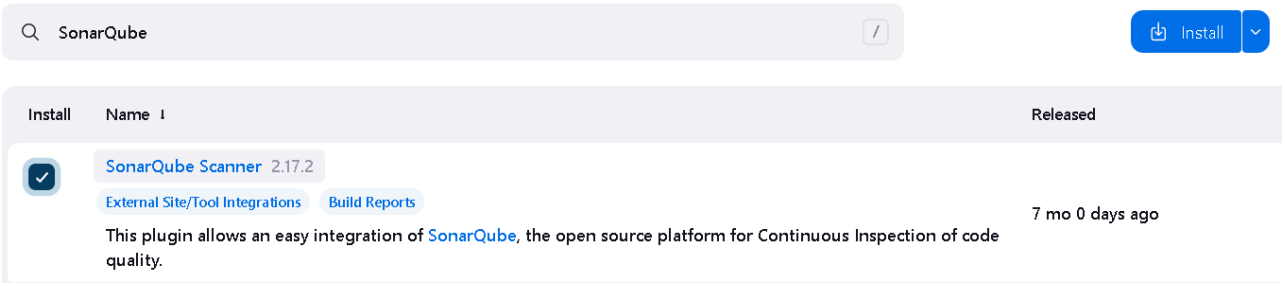
4. Login to SonarQube using username *admin* and password *admin*.

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



Setup the project and come back to Jenkins Dashboard.

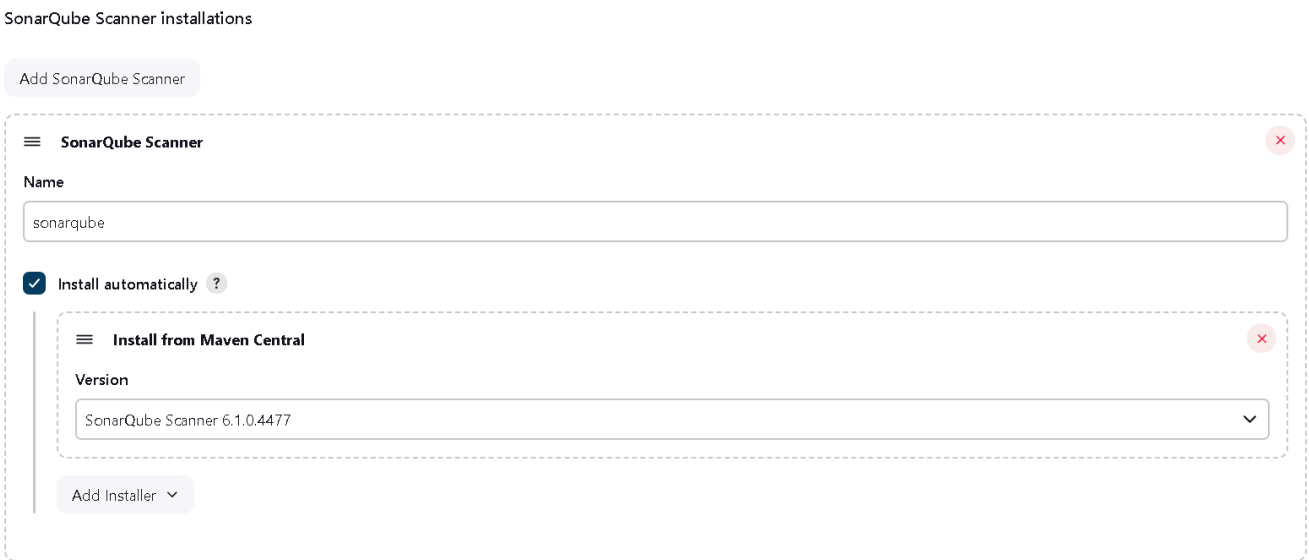
6. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



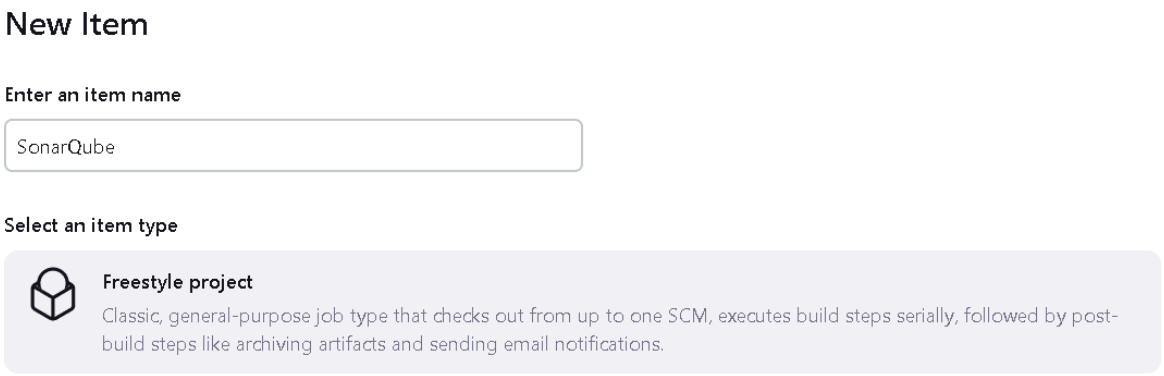
7. Under Jenkins ‘Configure System’, look for SonarQube Servers and enter the details.

   Enter the Server Authentication token if needed.

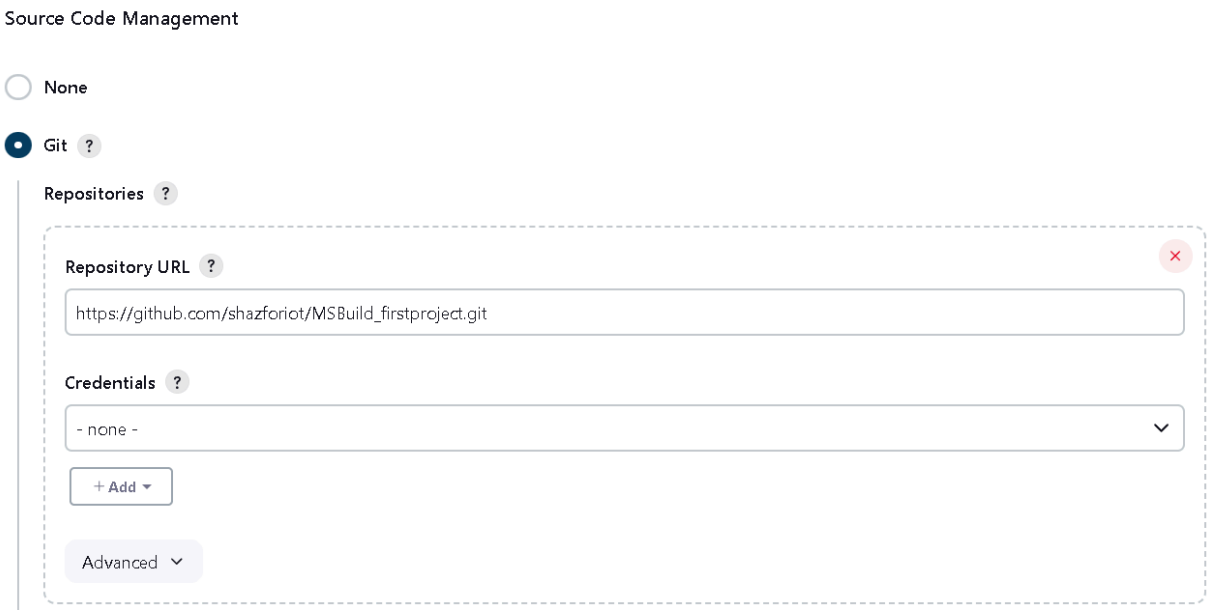
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.



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

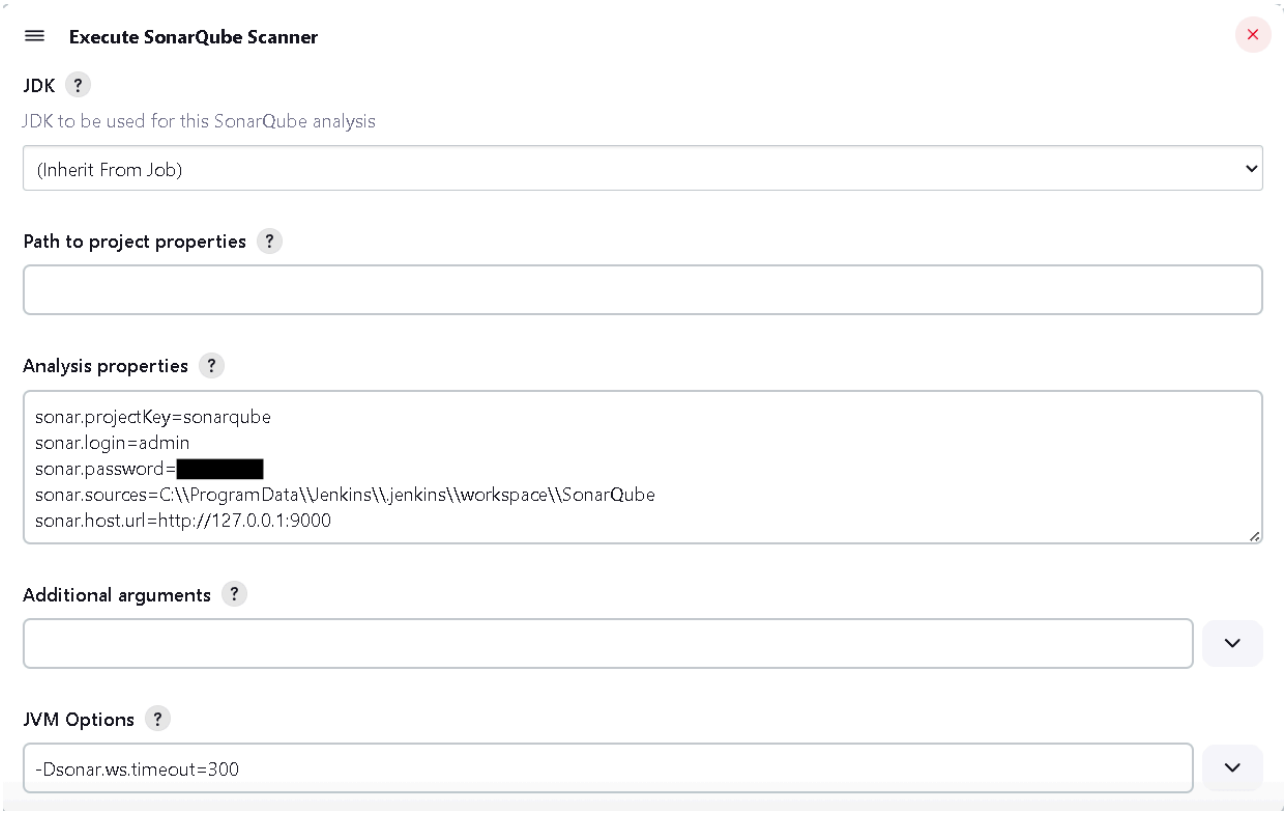


10. Choose this GitHub repository in Source Code Management.
https://github.com/shazforiot/MSBuild_firstproject.git

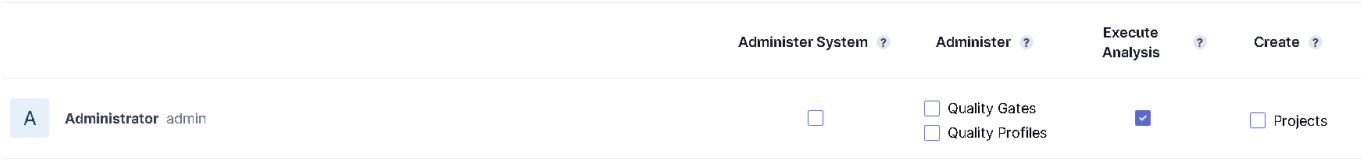


It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

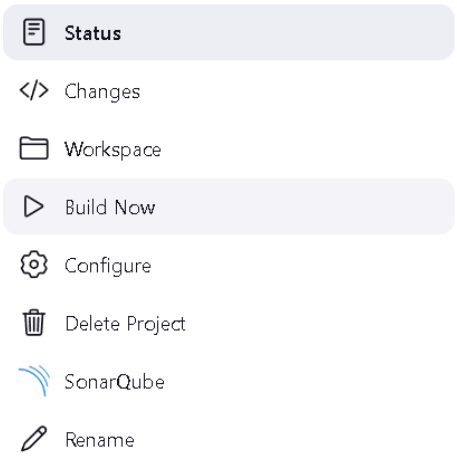
11. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



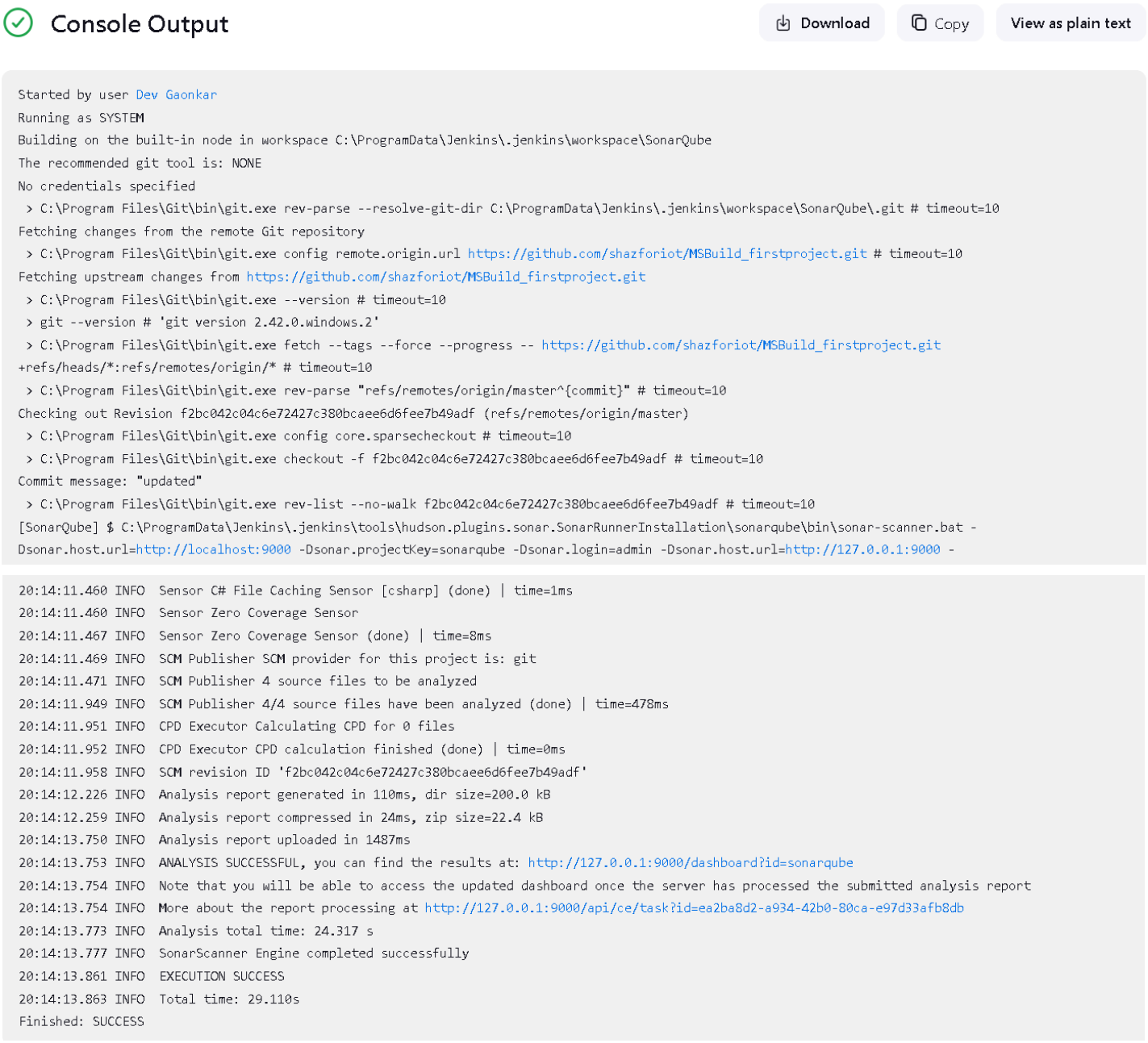
12. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.



13. Run The Build.



Check the console output.

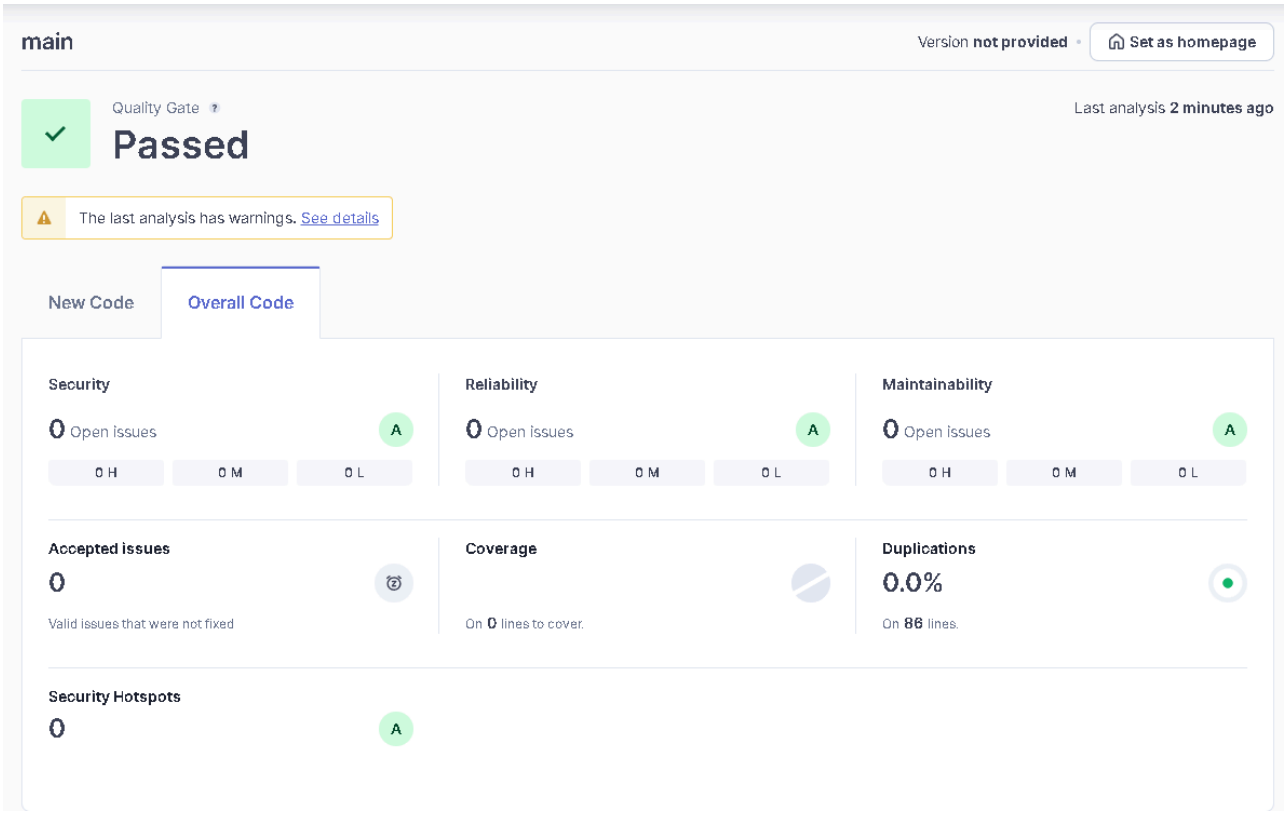


Console Output

Download Copy View as plain text

```
Started by user Dev Gaonkar
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
The recommended git tool is: NONE
No credentials specified
 > C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
 > C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.git
 > C:\Program Files\Git\bin\git.exe --version # timeout=10
 > git --version # 'git version 2.42.0.windows.2'
 > C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_firstproject.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
 > C:\Program Files\Git\bin\git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
 > C:\Program Files\Git\bin\git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
[SonarQube] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.login=admin -Dsonar.host.url=http://127.0.0.1:9000 -

20:14:11.460 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=1ms
20:14:11.460 INFO  Sensor Zero Coverage Sensor
20:14:11.467 INFO  Sensor Zero Coverage Sensor (done) | time=8ms
20:14:11.469 INFO  SCM Publisher SCM provider for this project is: git
20:14:11.471 INFO  SCM Publisher 4 source files to be analyzed
20:14:11.949 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=478ms
20:14:11.951 INFO  CPD Executor Calculating CPD for 0 files
20:14:11.952 INFO  CPD Executor CPD calculation finished (done) | time=0ms
20:14:11.958 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
20:14:12.226 INFO  Analysis report generated in 110ms, dir size=200.0 kB
20:14:12.259 INFO  Analysis report compressed in 24ms, zip size=22.4 kB
20:14:13.750 INFO  Analysis report uploaded in 1487ms
20:14:13.753 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube
20:14:13.754 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
20:14:13.754 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=ea2ba8d2-a934-42b0-80ca-e97d33afb8db
20:14:13.773 INFO  Analysis total time: 24.317 s
20:14:13.777 INFO  SonarScanner Engine completed successfully
20:14:13.861 INFO  EXECUTION SUCCESS
20:14:13.863 INFO  Total time: 29.110s
Finished: SUCCESS
```

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



In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion:

In this experiment, we explored the importance of Static Application Security Testing (SAST) and its role in identifying security vulnerabilities early in the software development lifecycle. By integrating Jenkins with SonarQube, we demonstrated an automated process for static code analysis. Using SonarQube's capabilities, we scanned a sample project for vulnerabilities, ensuring that code is secure and free of potential threats. This integration allows continuous code analysis during development, making it easier to address security issues before deployment. Overall, the experiment highlighted the efficiency and necessity of incorporating SAST tools into modern DevOps workflows for secure development.

Name: **Dev Gaonkar** Div/Roll no.: **D15C/12**

Adv.DevOps Experiment 8

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

Theory:

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 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.

What is a CI/CD Pipeline?

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. Before we dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. For example, there is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete.

Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

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.

Benefits of SonarQube

- **Sustainability -** Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.
- **Increase productivity -** Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
- **Quality code -** Code quality control is an inseparable part of the process of software development.
- **Detect Errors -** Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
- **Increase consistency -** Determines where the code criteria are breached and enhances the quality
- **Business scaling -** No restriction on the number of projects to be evaluated
- **Enhance developer skills -** Regular feedback on quality problems helps developers to improve their coding skills

Integrating Jenkins with SonarQube:

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

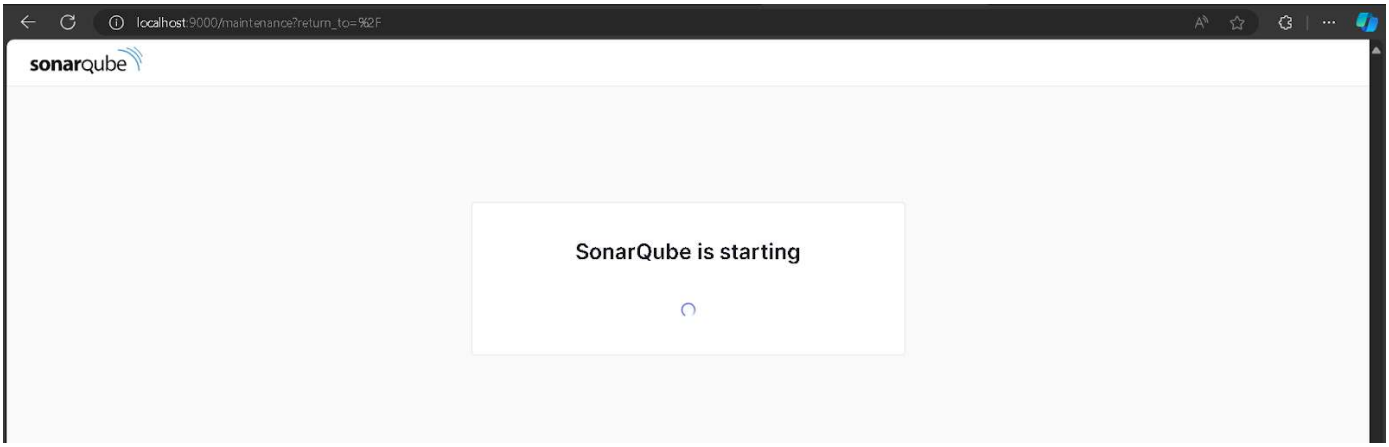
Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

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 -

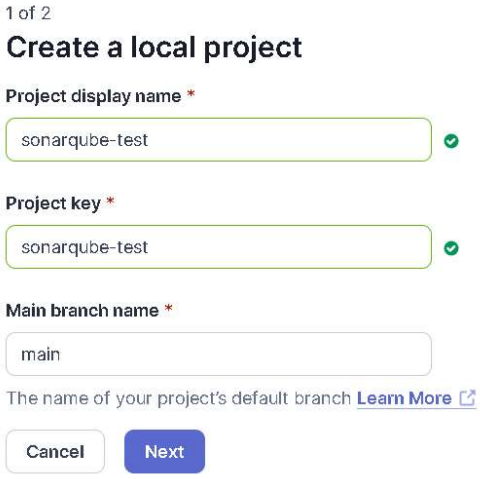
```
PS C:\Users\devpg> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
Status: Downloaded newer image for sonarqube:latest
60de6878d0614254500f608f43d81a3430585dc282e74225fe2a8fa237ee9d76
PS C:\Users\devpg>
```

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



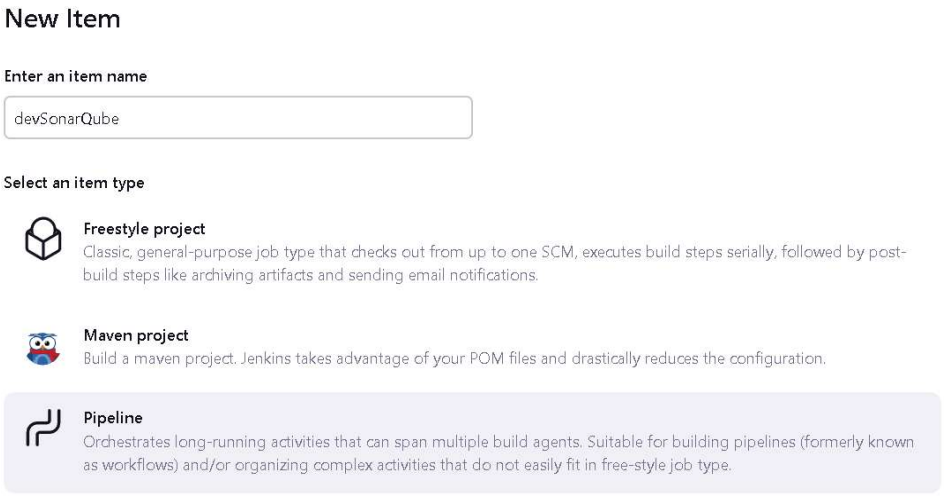
4. Login to SonarQube using username *admin* and password *admin*.

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



Setup the project and come back to Jenkins Dashboard.

6. Create a New Item in Jenkins, choose **Pipeline**.



7. Under Pipeline Script, enter the following -

```
node {
  stage('Cloning the GitHub Repo') {
    git 'https://github.com/shazforiot/GOL.git'
  }
  stage('SonarQube analysis') {
    withSonarQubeEnv('sonarqube') {
      sh "<PATH_TO_SONARQUBE_FOLDER>//bin//sonar-scanner \
      -D sonar.login=<SonarQube_USERNAME> \
      -D sonar.password=<SonarQube_PASSWORD> \
      -D sonar.projectKey=<Project_KEY> \
      -D sonar.exclusions=vendor/**,resources/**,**/*.java \
      -D sonar.host.url=http://127.0.0.1:9000/"
    }
  }
}
```

Pipeline

Definition

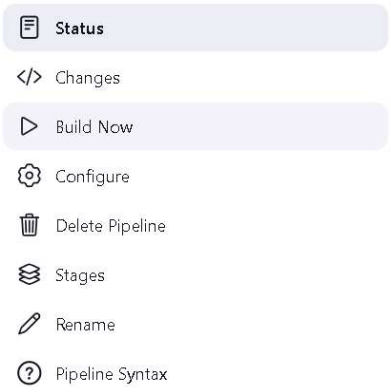
Pipeline script

Script ?

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }
    stage('SonarQube Analysis') {
        withSonarQubeEnv('sonarqube') {
            bat """
            C:/ProgramData/Jenkins/.jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarqube/bin/sonar-scanner.bat ^
            -Dsonar.login=admin ^
            -Dsonar.password=helloworld ^
            -Dsonar.projectKey=sonarqube-test ^
            -Dsonar.exclusions=vendor/**,resources/**,**/*.java ^
            -Dsonar.host.url=http://127.0.0.1:9000/
            """
        }
    }
}
```

It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

8. Run The Build.



9. Check the console output once the build is complete.

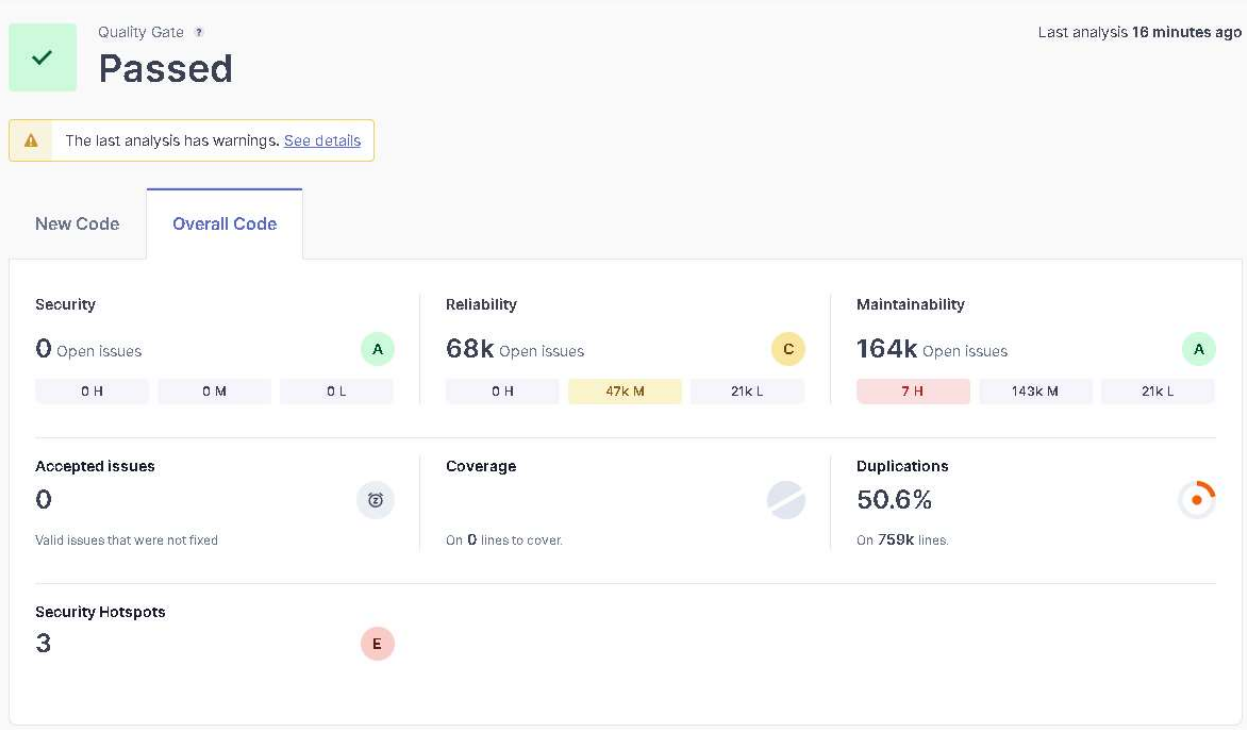
Console Output

Download Copy View as plain text

```
Started by user Dev Gaonkar
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\devSonarQube
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\devSonarQube\.git # timeout=10
Fetching changes from the remote Git repository
 > C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
 > C:\Program Files\Git\bin\git.exe --version # timeout=10
 > git --version # 'git version 2.42.0.windows.2'
 > C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
 > C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 (refs/remotes/origin/master)
 > C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
 > C:\Program Files\Git\bin\git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
 > C:\Program Files\Git\bin\git.exe branch -a -v --no-abbrev # timeout=10
 > C:\Program Files\Git\bin\git.exe branch -D master # timeout=10
 > C:\Program Files\Git\bin\git.exe checkout -b master ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
Commit message: "Update Jenkinsfile"
```

```
21:33:24.250 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html
for block at line 75. Keep only the first 100 references.
21:33:24.253 INFO  CPD Executor CPD calculation finished (done) | time=109702ms
21:33:24.381 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
21:34:52.072 INFO  Analysis report generated in 3326ms, dir size=127.2 MB
21:35:00.718 INFO  Analysis report compressed in 8633ms, zip size=29.6 MB
21:35:02.183 INFO  Analysis report uploaded in 1462ms
21:35:02.188 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-test
21:35:02.188 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
21:35:02.188 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=4fcac741-ad4f-4465-99b6-bd81a49a94cd
21:35:11.422 INFO  Analysis total time: 8:38.464 s
21:35:11.434 INFO  SonarScanner Engine completed successfully
21:35:12.040 INFO  EXECUTION SUCCESS
21:35:12.079 INFO  Total time: 8:45.529s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

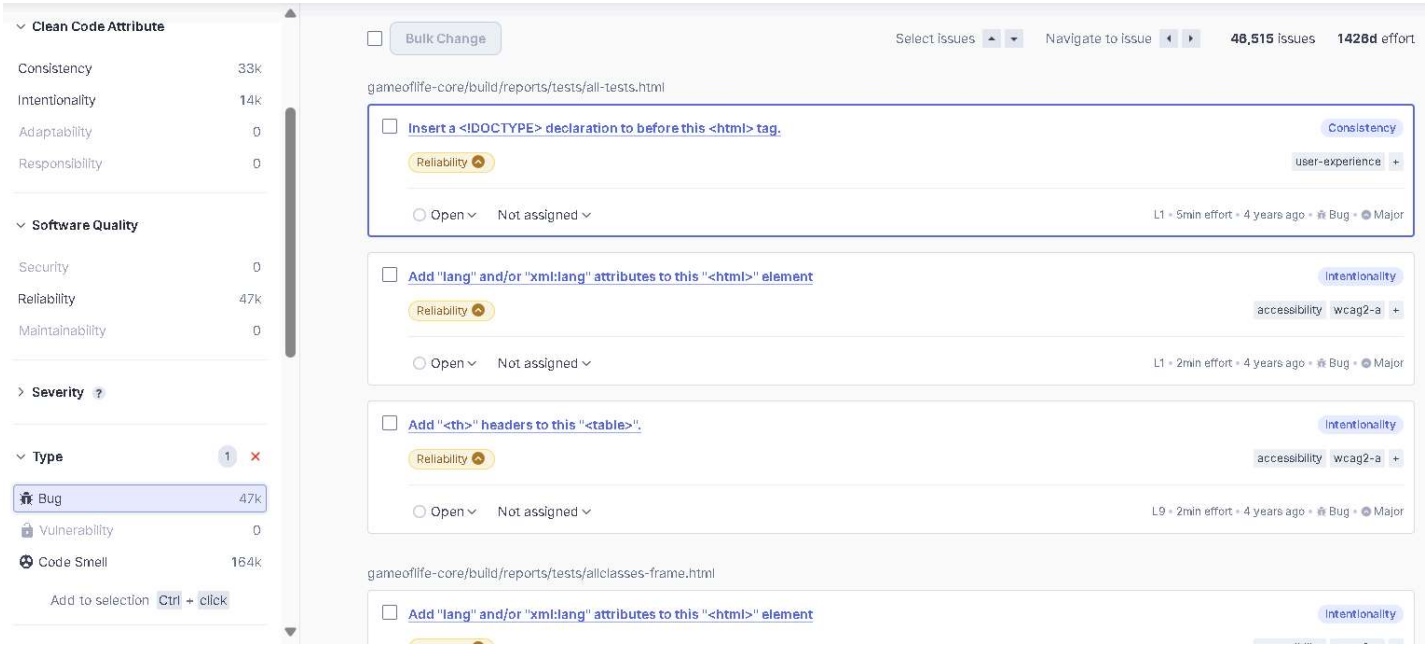
10. After that, check the project in SonarQube.



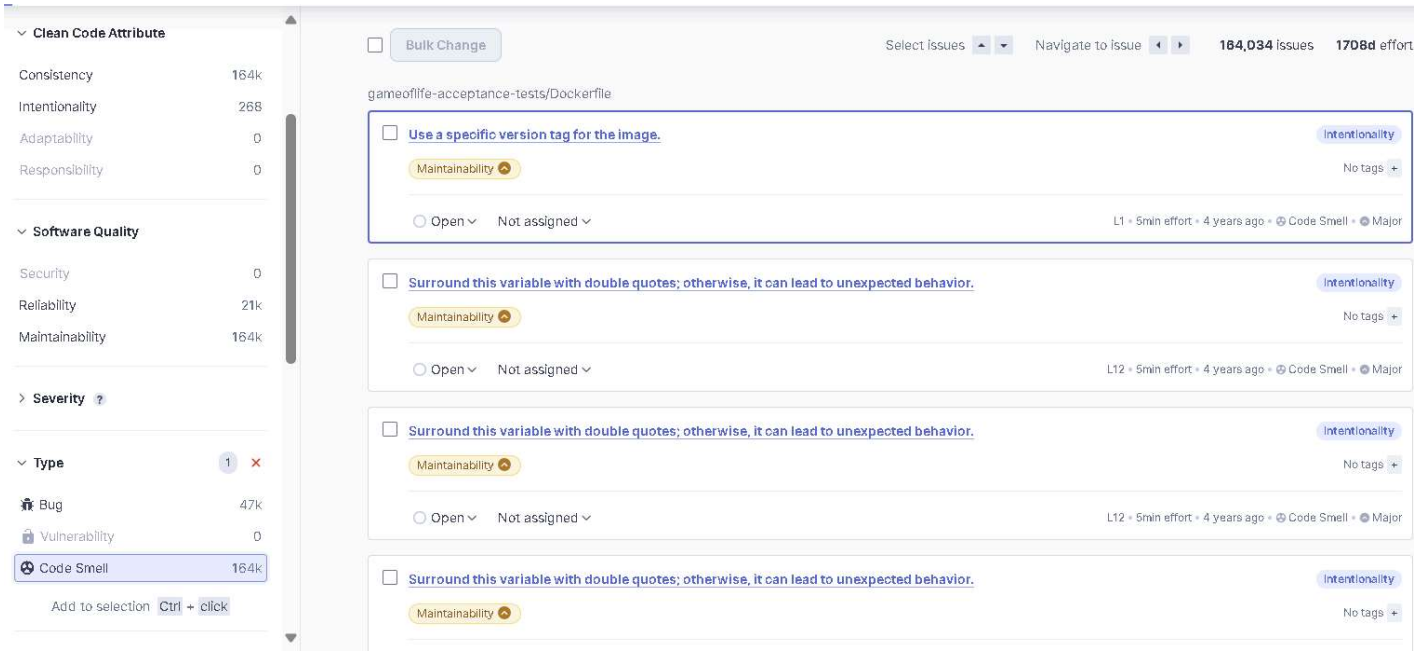
Under different tabs, check all different issues with the code.

11. Code Problems -

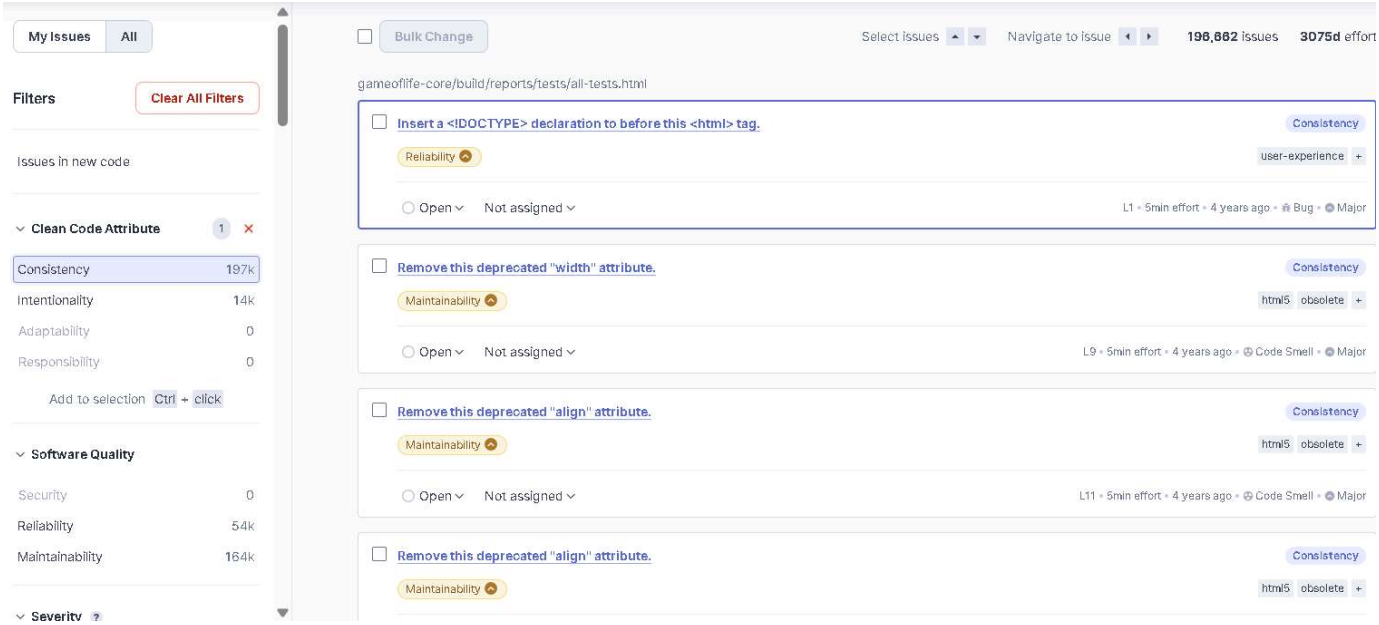
Bugs:



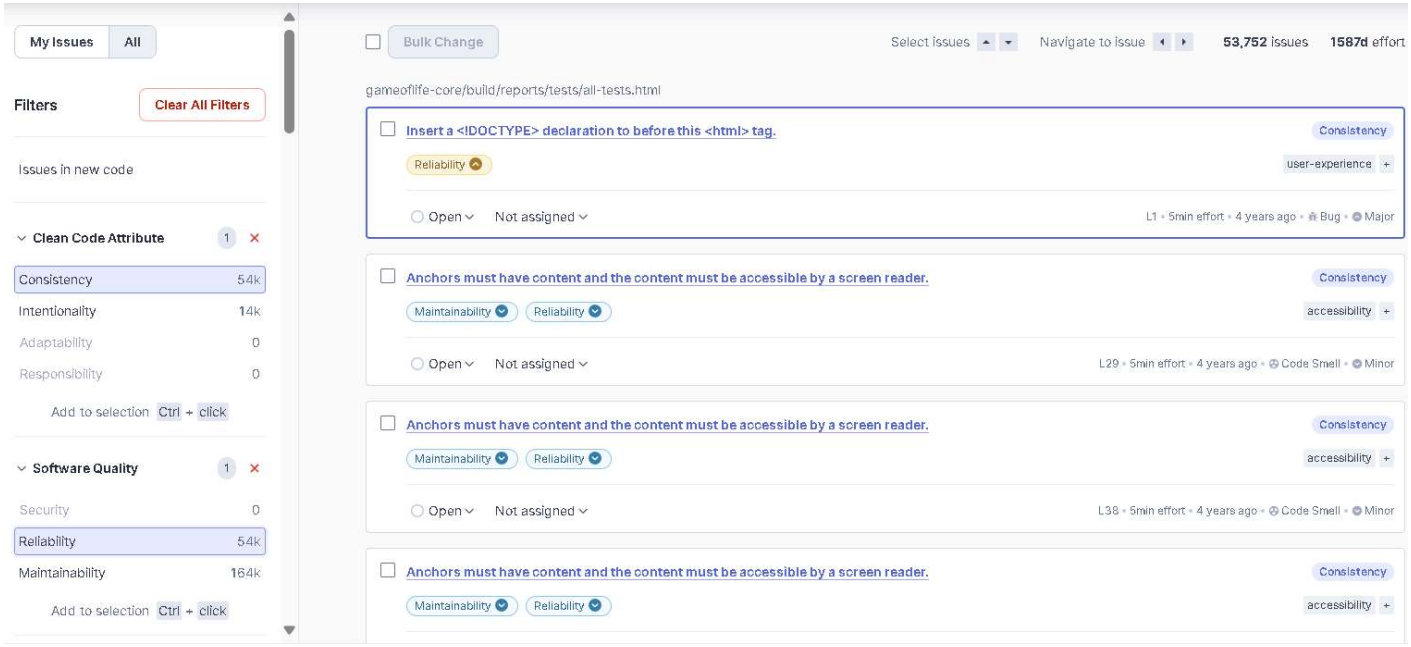
Code Smells:



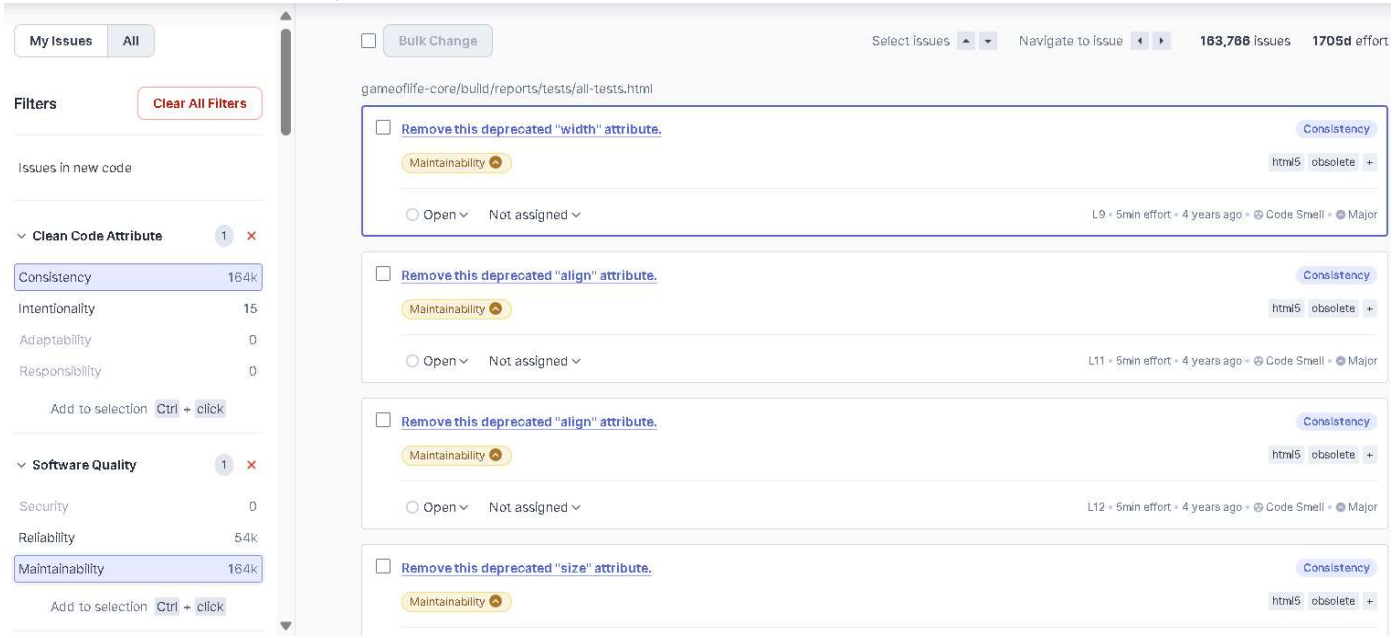
Consistency Issues:



Reliability Issues:



Maintainability Issues:



In this way, we have created a CI/CD Pipeline with Jenkins and integrated it with SonarQube to find issues in the code like bugs, code smells, duplicates, cyclomatic complexities, etc.

Conclusion:

In conclusion, integrating Jenkins with SonarQube through a CI/CD pipeline enhances code quality by automating static code analysis. By setting up SonarQube in a Docker container and configuring Jenkins to run SonarQube scans, we can efficiently detect and address bugs, code smells, and security vulnerabilities early in the development cycle. This approach ensures comprehensive code inspections and provides actionable feedback, leading to more secure and maintainable code. Ultimately, this integration streamlines the development process, improves software quality, and supports continuous improvement in coding practices.

Name: **Dev Gaonkar** Div/ Roll no: **D15C/ 12**

Experiment 9

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

Theory:

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.

Why Do We Need Nagios Tools?

Here are the important reasons to use Nagios monitoring tool:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitor and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation

Features of Nagios

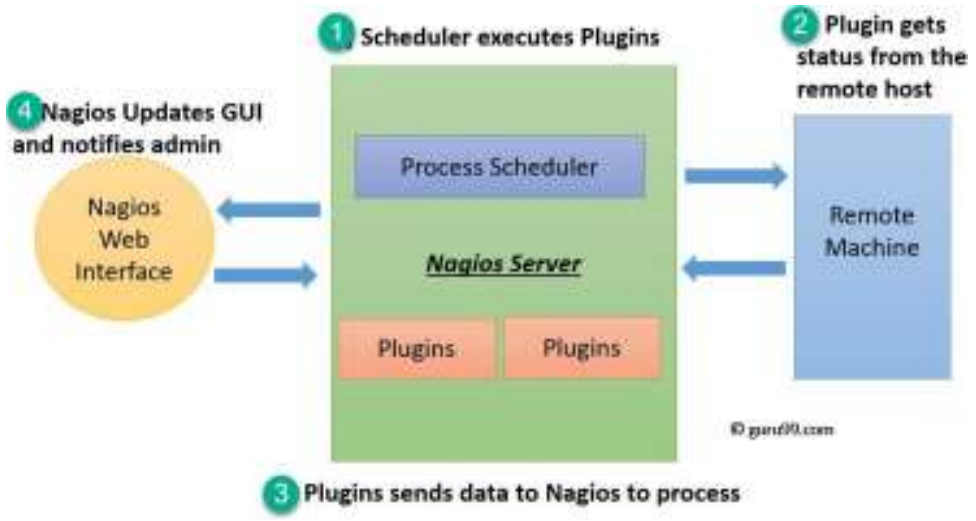
Following are the important features of Nagios monitoring tool:

- 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

Nagios Architecture

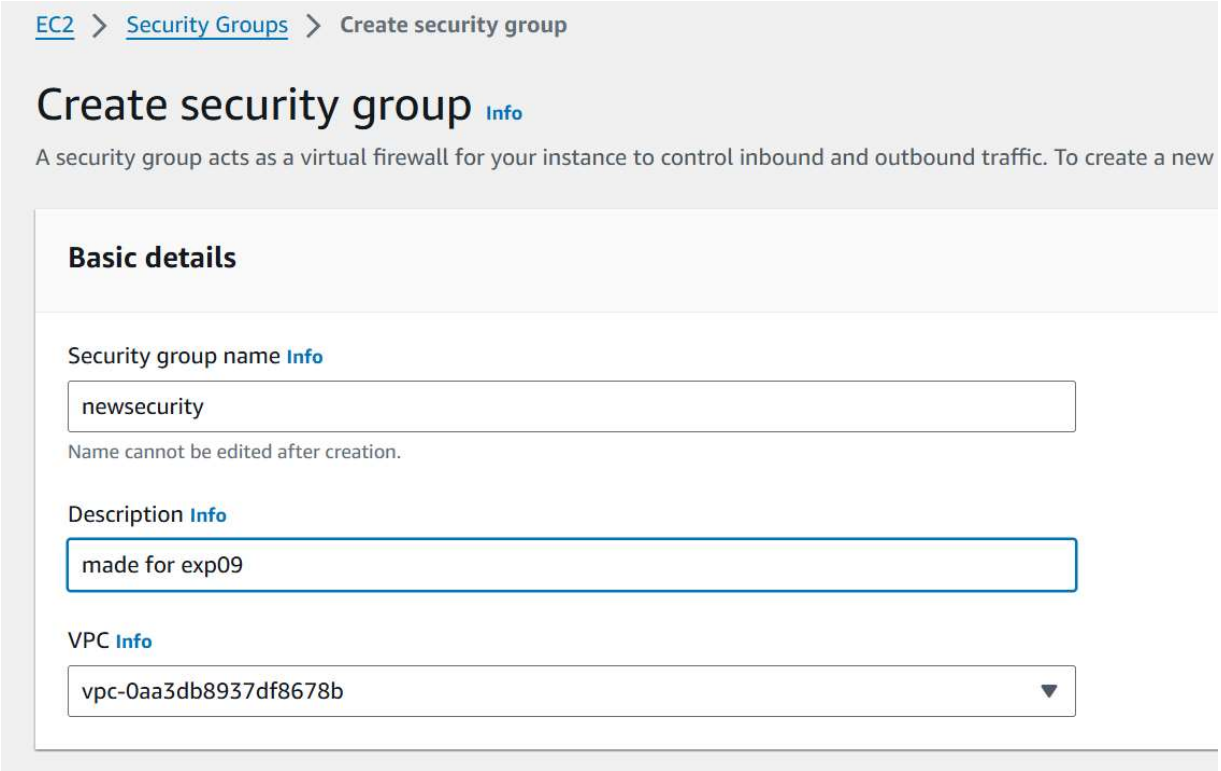
Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.



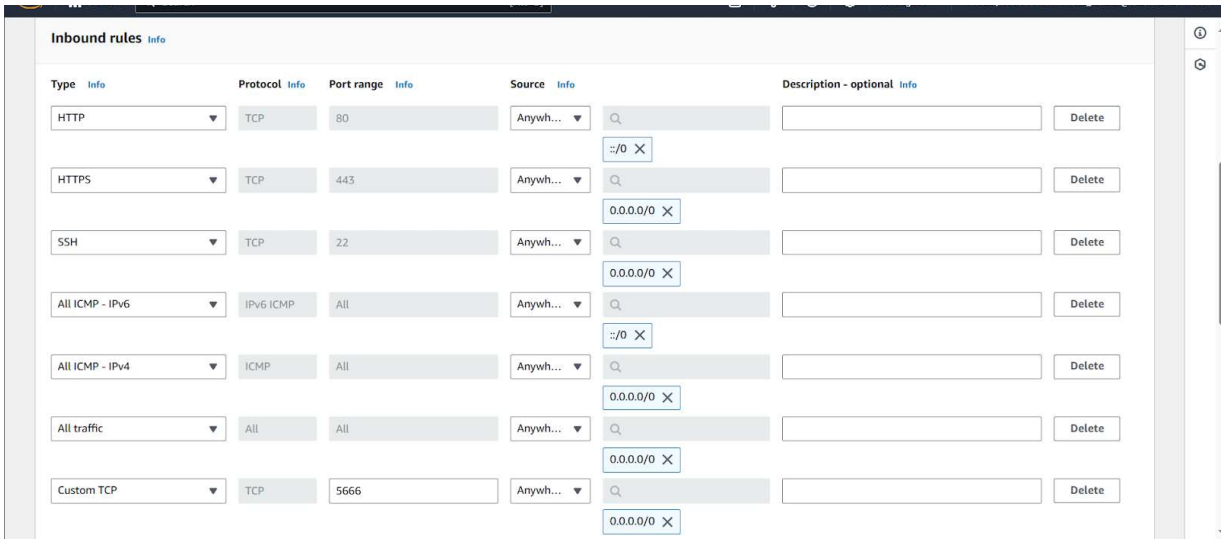
1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

Step 1: Create a security group with the required configurations

I have created a new security group with a name ‘newsecurity’

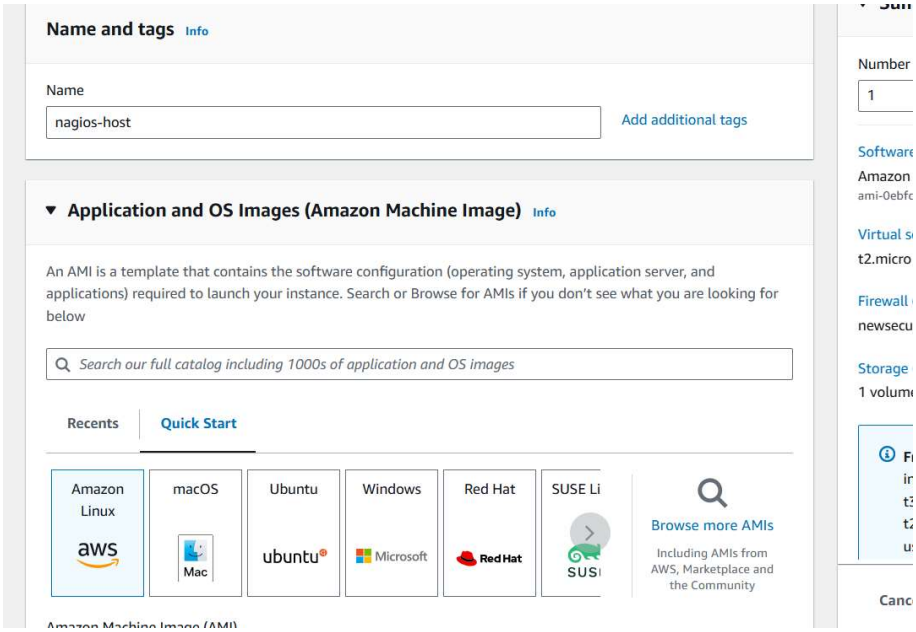


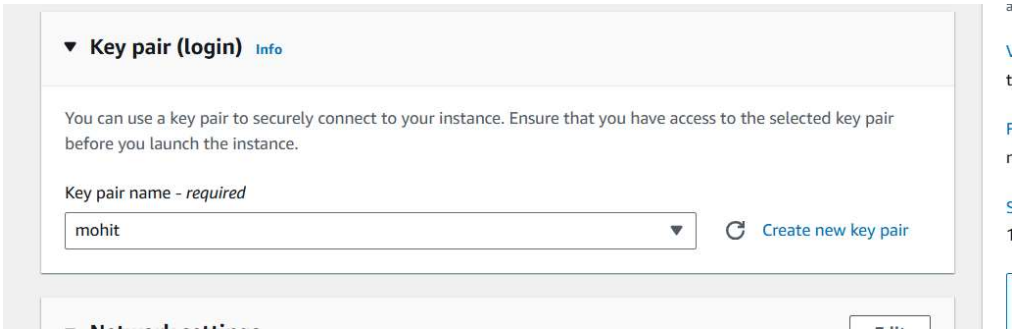
I have modified the INBOUND RULES as follows

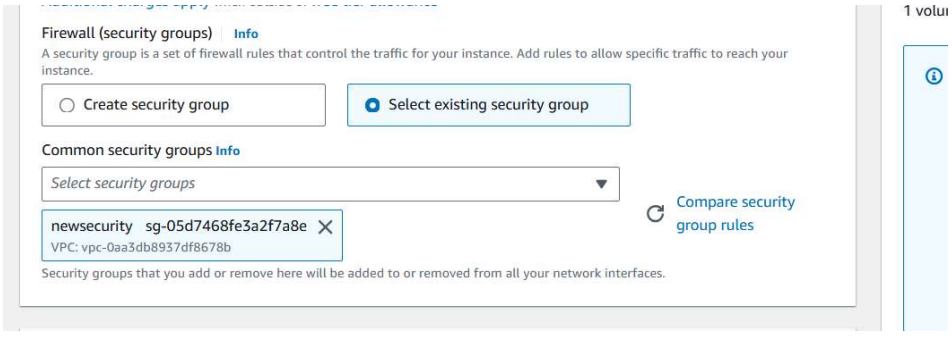


Step 2: Create ec2 instance

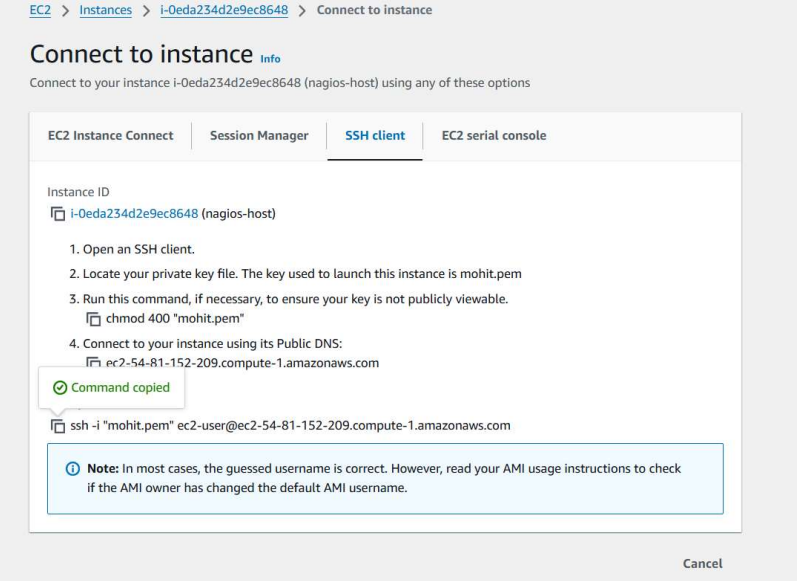
Name it as nagios-host. Select instance type as amazon-linux and choose the already created key pair and security group







Copy the given ssh command, as we will require it for logging into our nagios-host instance from our windows powershell



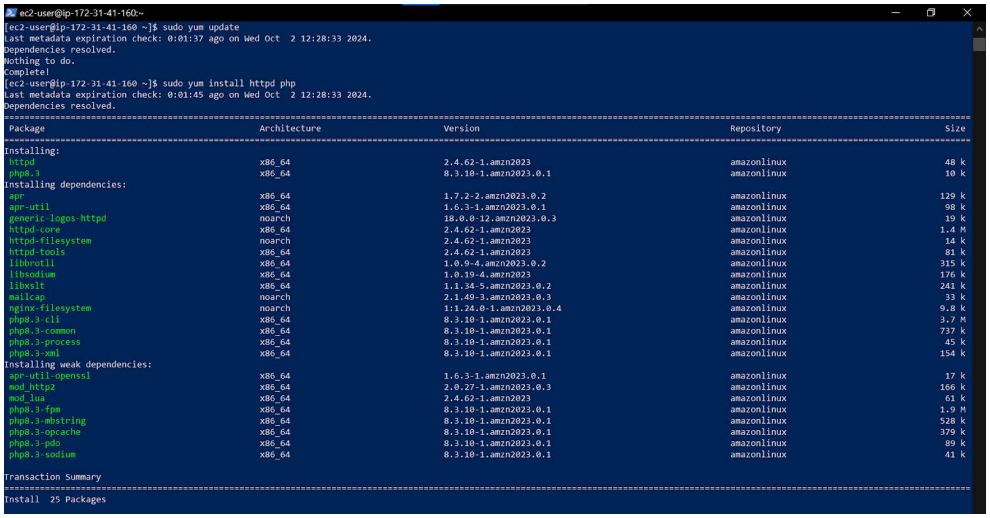
Step 3: Open an administrative powershell and remotely login using the above mentioned ssh command

```
ec2-user@ip-172-31-92-249:~
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> cd C:\Users\Dell\Downloads
PS C:\Users\Dell\Downloads> ssh -i "mohit.pem" ec2-user@ec2-54-81-152-209.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Mon Sep 30 09:25:13 2024 from 125.99.93.18
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Mon Sep 30 09:25:13 2024 from 125.99.93.18
[ec2-user@ip-172-31-92-249 ~]$ sudo yum update
Last metadata expiration check: 0:13:13 ago on Mon Sep 30 09:23:03 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-92-249 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:13:23 ago on Mon Sep 30 09:23:03 2024.
Package httpd-2.4.62-1.amzn2023.x86_64 is already installed.
Package php8.3-8.3.10-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

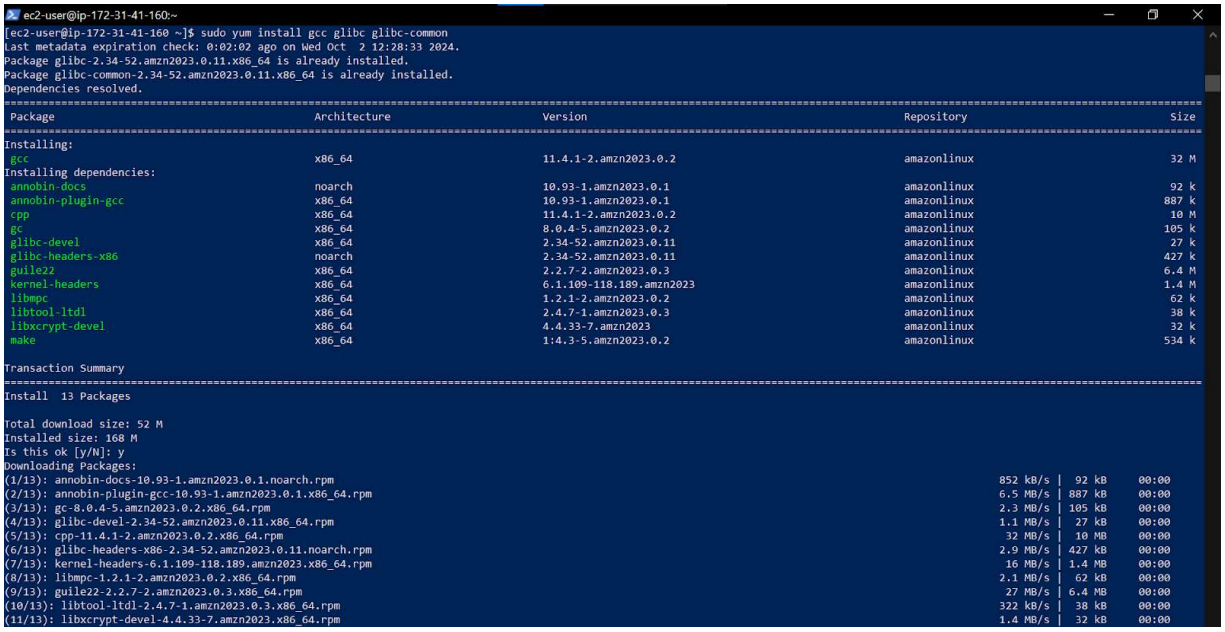
And then run these commands

sudo yum update

sudo yum install httpd php



sudo yum install gcc glibc glibc-common



sudo yum install gd gd-devel

```
ec2-user@ip-172-31-41-160:~
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:02:25 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                  Repository  Size
================================================================================
Installing:
 gd                      x86_64          2.3.3-5.amzn2023.0.3     amazonlinux 139 k
 gd-devel                x86_64          2.3.3-5.amzn2023.0.3     amazonlinux  38 k
Installing dependencies:
 brotli                  x86_64          1.0.9-4.amzn2023.0.2     amazonlinux 314 k
 brotli-devel            x86_64          1.0.9-4.amzn2023.0.2     amazonlinux  31 k
 bzip2-devel             x86_64          1.0.8-6.amzn2023.0.2     amazonlinux 214 k
 cairo                   x86_64          1.17.6-2.amzn2023.0.1    amazonlinux 684 k
 cmake-filesystem        x86_64          3.22.2-1.amzn2023.0.4    amazonlinux  16 k
 fontconfig              x86_64          2.13.94-2.amzn2023.0.2   amazonlinux 273 k
 fontconfig-devel        x86_64          2.13.94-2.amzn2023.0.2   amazonlinux 128 k
 fonts-filesystem        noarch          1:2.0.5-12.amzn2023.0.2  amazonlinux 9.5 k
 freetype                x86_64          2.13.2-5.amzn2023.0.1    amazonlinux 423 k
 freetype-devel          x86_64          2.13.2-5.amzn2023.0.1    amazonlinux 912 k
 glib2-devel             x86_64          2.74.7-689.amzn2023.0.2  amazonlinux 486 k
 google-noto-fonts-common noarch         20201206-2.amzn2023.0.2  amazonlinux  15 k
 google-noto-sans-vf-fonts noarch        20201206-2.amzn2023.0.2  amazonlinux 492 k
 graphite2               x86_64          1.3.14-7.amzn2023.0.2    amazonlinux  97 k
 graphite2-devel         x86_64          1.3.14-7.amzn2023.0.2    amazonlinux  21 k
 harfbuzz                x86_64          7.0.0-2.amzn2023.0.1     amazonlinux 868 k
 harfbuzz-devel          x86_64          7.0.0-2.amzn2023.0.1     amazonlinux 404 k
 harfbuzz-icu            x86_64          7.0.0-2.amzn2023.0.1     amazonlinux  18 k
 jbigkit-libs            x86_64          2.1-21.amzn2023.0.2      amazonlinux  54 k
 langpacks-core-font-en  noarch          3.0-21.amzn2023.0.4      amazonlinux  10 k
 libICE                  x86_64          1.0.10-6.amzn2023.0.2    amazonlinux  71 k
 libSM                   x86_64          1.2.3-8.amzn2023.0.2     amazonlinux  42 k
 libX11                  x86_64          1.7.2-3.amzn2023.0.4     amazonlinux 657 k
 libX11-common           noarch          1.7.2-3.amzn2023.0.4     amazonlinux 152 k
 libX11-devel            x86_64          1.7.2-3.amzn2023.0.4     amazonlinux 939 k
 libX11-xcb              x86_64          1.7.2-3.amzn2023.0.4     amazonlinux  12 k
 libXau                  x86_64          1.0.9-6.amzn2023.0.2     amazonlinux  31 k
 libXau-devel            x86_64          1.0.9-6.amzn2023.0.2     amazonlinux  14 k
 libXext                 x86_64          1.3.4-6.amzn2023.0.2     amazonlinux  41 k
 libXpm                  x86_64          3.5.15-2.amzn2023.0.3    amazonlinux  65 k
 libXpm-devel            x86_64          3.5.15-2.amzn2023.0.3    amazonlinux  59 k
 libXrender              x86_64          0.9.10-14.amzn2023.0.2   amazonlinux  28 k
 libXt                   x86_64          1.2.0-4.amzn2023.0.2     amazonlinux 181 k
 libblkid-devel          x86_64          2.37.4-1.amzn2023.0.4    amazonlinux  15 k
```

Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.

sudo adduser -m nagios

sudo passwd nagios

```
[ec2-user@ip-172-31-41-160 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-41-160 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-41-160 ~]$
```

Create a new user group & create a new directory for Nagios downloads using the following commands

sudo groupadd nagcmd

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

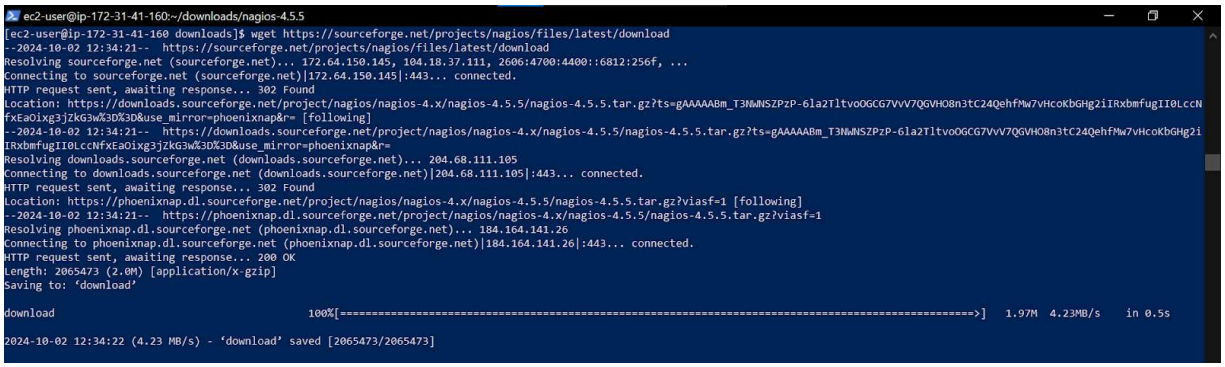
mkdir ~/downloads

cd ~/downloads

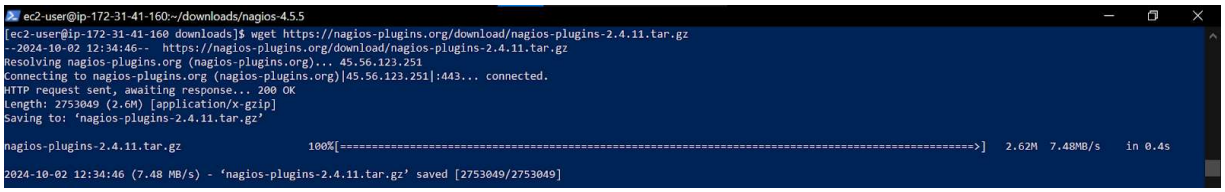
Use **wget** to download the source zip files.

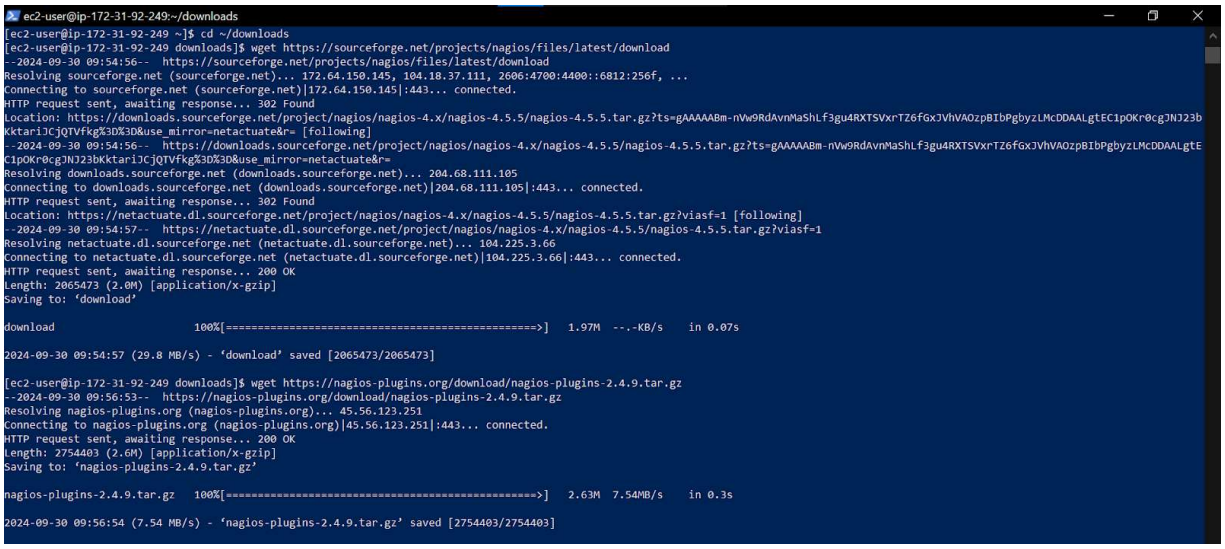
In this step, we are downloading, the latest version of nagios and the necessary plugins required to carry out the tasks of setting up a nagios server

wget https://sourceforge.net/projects/nagios/files/latest/download

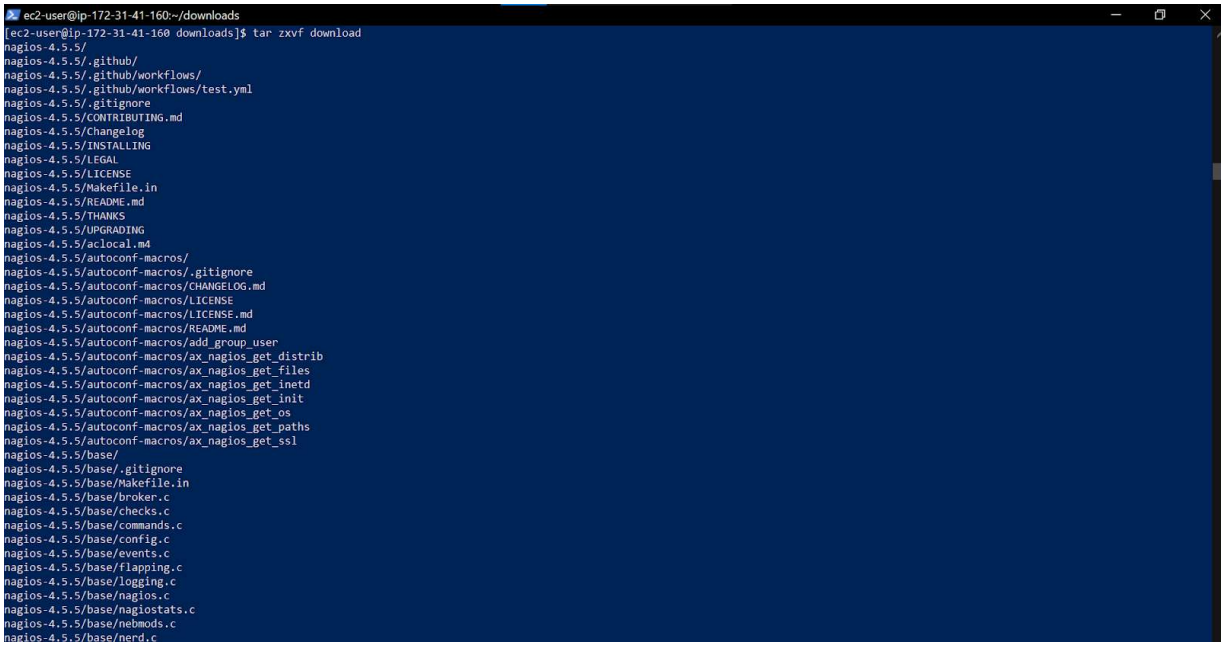


wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz



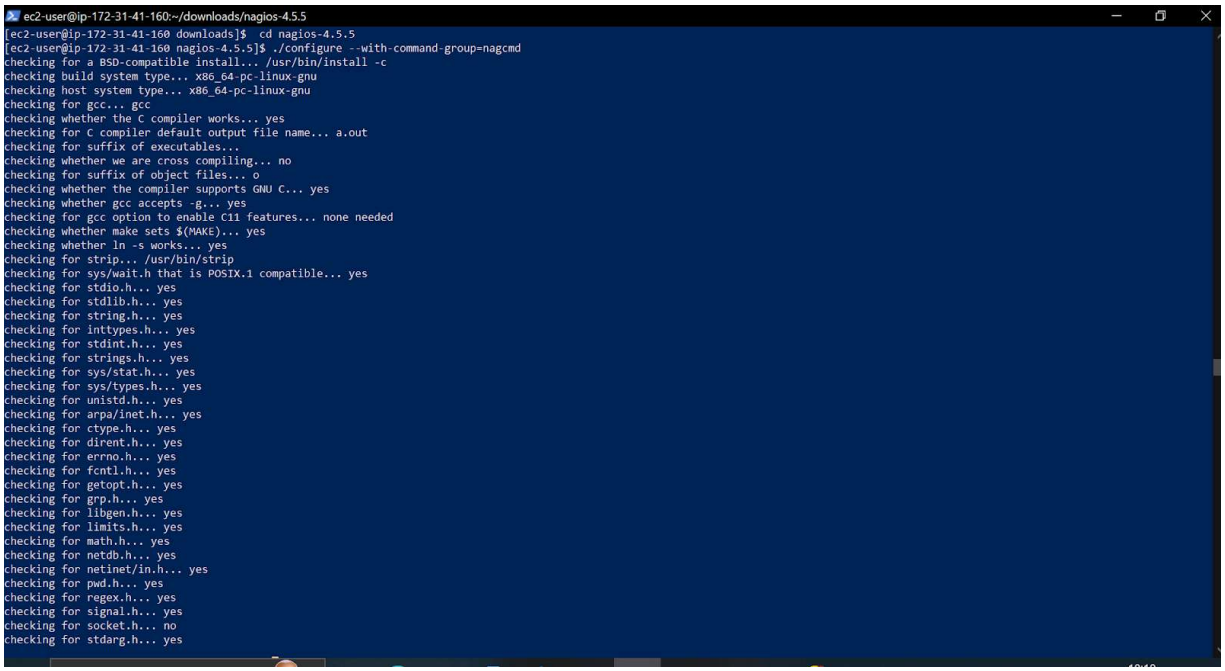


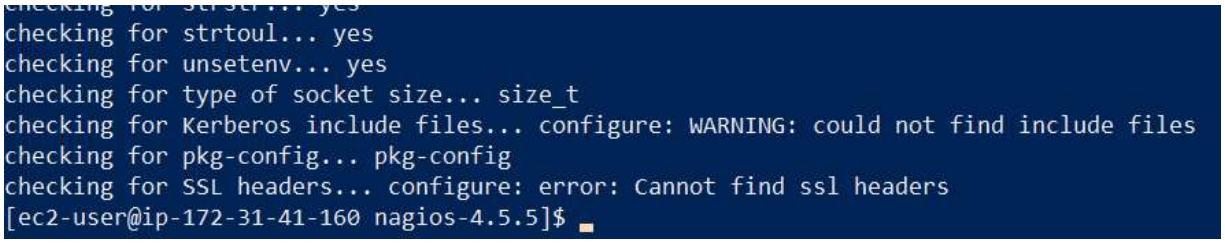
Now, we run the next command in the following manner
tar zxvf <nagios-4.5.5 version> (for me it has gotten saved as 'download')
So i wrote **tar zxvf download**



After which we are supposed to **change our directory** over there
For eg. **cd nagios-4.5.5**… depending on the version that we have downloaded

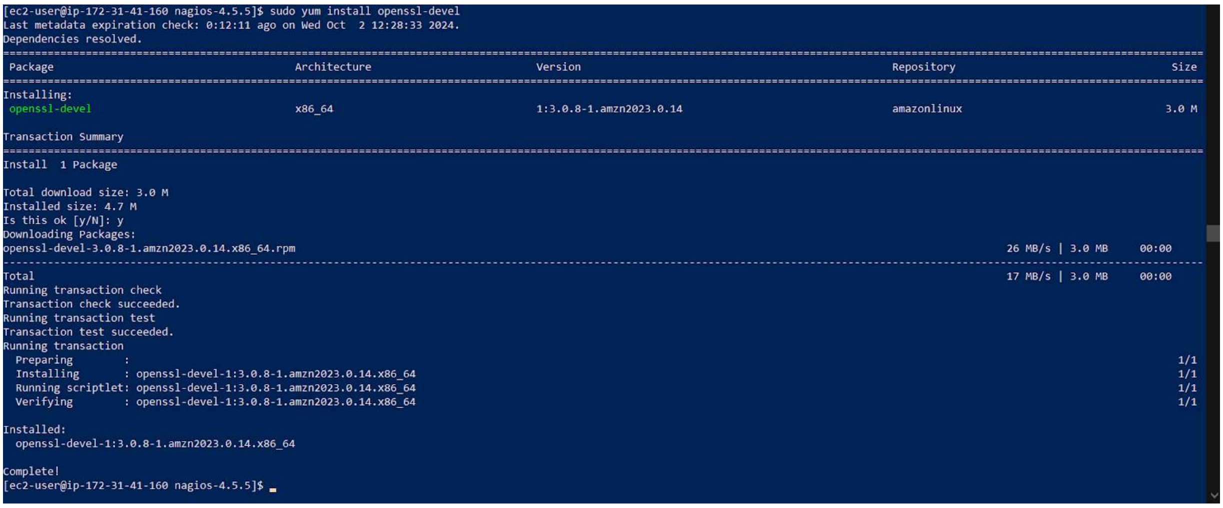
Next, Run this command (make sure that you are working inside nagios-4.x.x directory)
./configure --with-command-group=nagcmd





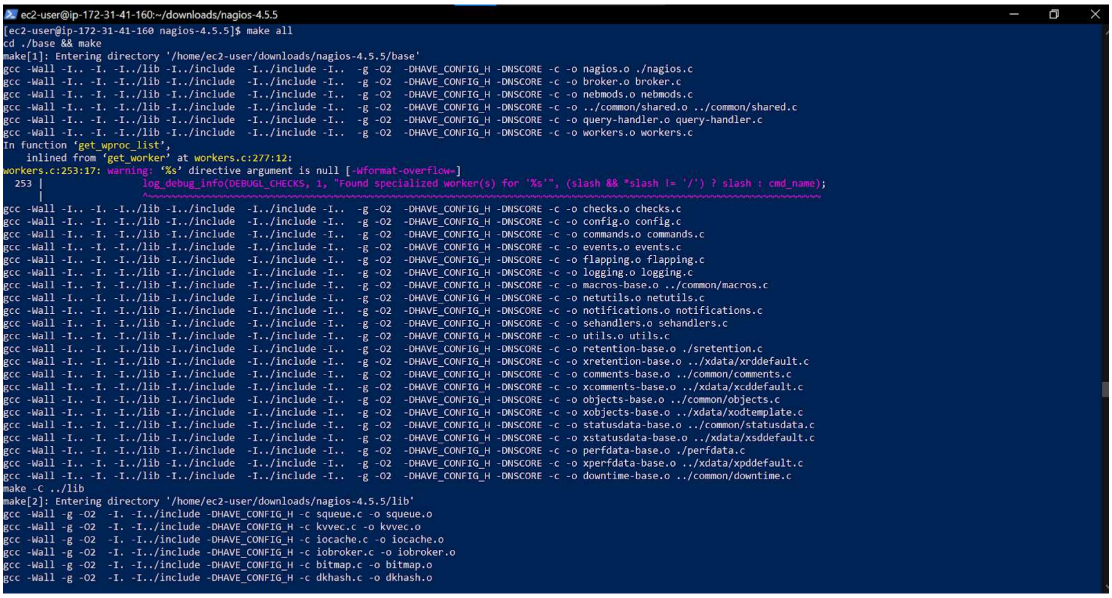
After running this command, we get an **error related to ssl header being absent**

For that purpose, we are to run the following command.
sudo yum install openssl-devel (for ssl header)

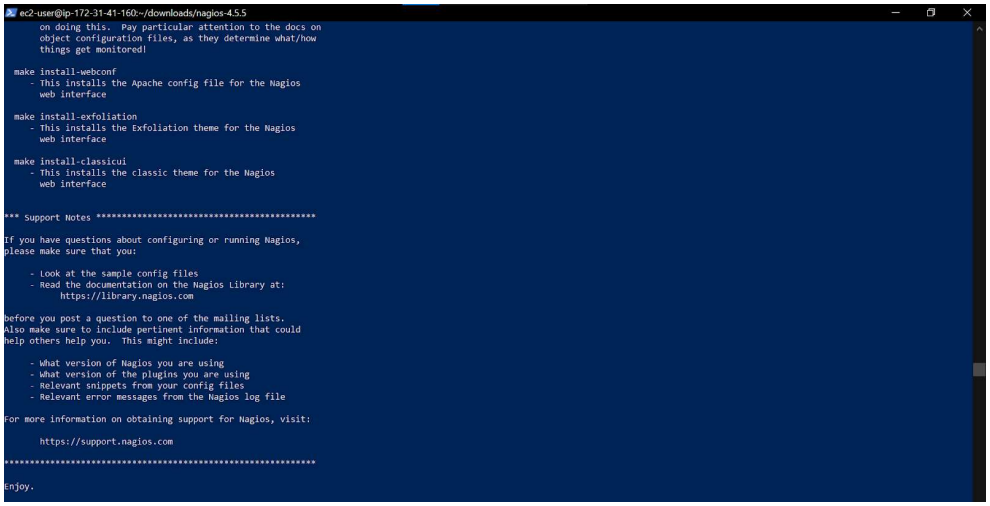


```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:12:11 ago on Wed Oct  2 12:28:33 2024.
Dependencies resolved.
================================================================================
 Package            Architecture      Version                    Repository        Size
================================================================================
Installing:
 openssl-devel      x86_64            1:3.0.8-1.amzn2023.0.14    amazonlinux       3.0 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm          26 MB/s | 3.0 MB     00:00
--------------------------------------------------------------------------------
Total                                                   17 MB/s | 3.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                          1/1
  Installing       : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64             1/1
  Running scriptlet: openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64             1/1
  Verifying        : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64             1/1

Installed:
  openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64

Complete!
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Now, Re-run **./configure --with-command-group=nagcmd**
After this, run **make all** command



```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function ‘get_wproc_list’,
    inlined from ‘get_worker’ at workers.c:277:12:
workers.c:253:17: warning: ‘%s’ directive argument is null [-Wformat-overflow=]
  253 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/') ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o checks.o checks.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o config.o config.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o commands.o commands.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o events.o events.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o flapping.o flapping.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o logging.o logging.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o macros-base.o ../common/macros.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o netutils.o netutils.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o notifications.o notifications.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o sehandlers.o sehandlers.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o utils.o utils.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o retention-base.o ./sretention.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xretention-base.o ../xdata/xrddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o comments-base.o ../common/comments.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xcomments-base.o ../xdata/xcddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o objects-base.o ../common/objects.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xobjects-base.o ../xdata/xodtemplate.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o statusdata-base.o ../common/statusdata.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xstatusdata-base.o ../xdata/xsddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o perfdata-base.o ./perfdata.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o xperfdata-base.o ../xdata/xpddefault.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2  -DHAVE_CONFIG_H -DNSCORE -c -o downtime-base.o ../common/downtime.c
make -C ../lib
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/lib'
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c squeue.c -o squeue.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c kvvec.c -o kvvec.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c iocache.c -o iocache.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c iobroker.c -o iobroker.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c bitmap.c -o bitmap.o
gcc -Wall -g -O2  -I. -I../include -DHAVE_CONFIG_H -c dkhash.c -o dkhash.o
```



Run the following set of commands to ensure that

sudo make install

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/js
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images/logos
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/includes
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/ssi
/usr/bin/install -c -m 664 -o nagios -g nagios ./robots.txt /usr/local/nagios/share
/usr/bin/install -c -m 664 -o nagios -g nagios ./jsonquery.html /usr/local/nagios/share
rm -f /usr/local/nagios/share/index.html
rm -f /usr/local/nagios/share/main.html
rm -f /usr/local/nagios/share/side.html
rm -f /usr/local/nagios/share/map.html
rm -f /usr/local/nagios/share/rss-*
rm -f /usr/local/nagios/share/graph-header.html
rm -f /usr/local/nagios/share/histogram.html
rm -f /usr/local/nagios/share/histogram-form.html
rm -f /usr/local/nagios/share/histogram-graph.html
rm -f /usr/local/nagios/share/histogram-links.html
rm -f /usr/local/nagios/share/infobox.html
rm -f /usr/local/nagios/share/map.php
```

sudo make install-init

```
ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
```

sudo make install-config

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

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

sudo make install-webconf

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

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

[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Next, we are supposed to create a nagiosadmin account for nagios login along with password.
Specify the password twice.

sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Restart Apache

sudo service httpd restart

Go back to the downloads folder and unzip the plugins zip file.

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz

```
ec2-user@ip-172-31-41-160:~/downloads
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-41-160 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
```

Compile and install plugins
cd nagios-plugins-2.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios

Run the following command:
sudo chkconfig --add nagios
On running the above command

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
```

If this is the output that one is getting, then it means that the init script is missing…
We can check this by running ls /etc/init.d/

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$ ls /etc/init.d/
README  functions
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$
```

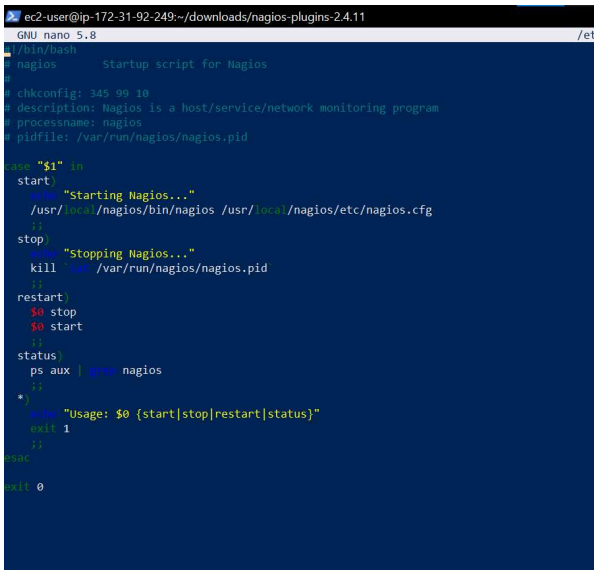
With ls command, we must see a file named nagios, which i was not able to see
If the Init Script is Missing i.e If you don't see the nagios script in /etc/init.d/, you can create it manually. Here's how:

Run the following command:
sudo nano /etc/init.d/nagios

Within this file, paste the following script

#!/bin/bash

```
# nagios       Startup script for Nagios
#
# chkconfig: 345 99 10
# description: Nagios is a host/service/network monitoring program
# processname: nagios
# pidfile: /var/run/nagios/nagios.pid
case "$1" in
  start)
    echo "Starting Nagios..."
    /usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg
    ;;
  stop)
    echo "Stopping Nagios..."
    kill `cat /var/run/nagios/nagios.pid`
    ;;
  restart)
    $0 stop
    $0 start
    ;;
  status)
    ps aux | grep nagios
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
    ;;
esac
exit 0
```



Make the Script Executable: After saving the file, run the following command to make it executable:

sudo chmod +x /etc/init.d/nagios

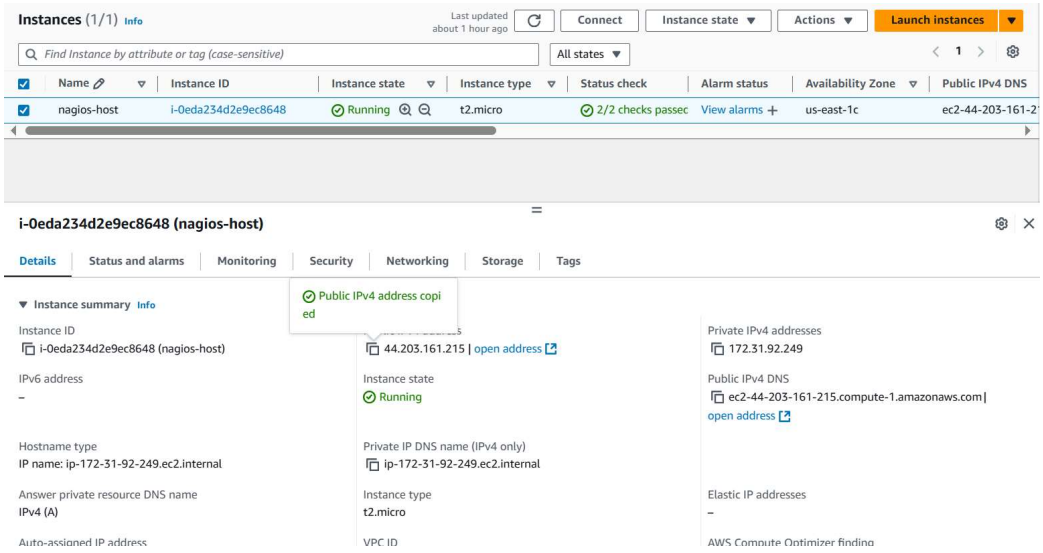
Run **sudo chkconfig --add nagios** again
And then run **sudo chkconfig nagios on**

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo nano /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chmod +x /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig nagios on
Note: Forwarding request to 'systemctl enable nagios.service'.
Synchronizing state of nagios.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

sudo service nagios start

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.11]$ sudo service nagios start
Starting Nagios...

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Nagios 4.5.5 starting... (PID=72261)
Local time is Tue Oct 01 20:59:58 UTC 2024
wproc: Successfully registered manager as @wproc with query handler
wproc: Registry request: name=Core Worker 72265;pid=72265
wproc: Registry request: name=Core Worker 72264;pid=72264
wproc: Registry request: name=Core Worker 72263;pid=72263
wproc: Registry request: name=Core Worker 72262;pid=72262
Successfully launched command file worker with pid 72266
wproc: NOTIFY job 4 from worker Core Worker 72262 is a non-check helper but exited with return code 127
wproc:   host=localhost; service=Swap Usage; contact=nagiosadmin
wproc:   early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
wproc:   stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
wproc:   stderr line 02: /usr/bin/printf: write error: Broken pipe
```

Get your public IPv4 address from your instance. We will require it for connecting to our nginx server



Browse for this url: http://<your_public_ip_address>/nagios
The browser may ask you for your nagios credentials which set in the earlier steps
The username is nagiosadmin and enter the password that you set earlier

Conclusion:

In this experiment, we successfully installed and configured Nagios Core on an Amazon Linux EC2 instance, showcasing its role in continuous monitoring within a DevOps environment. We learned about user management and service configuration, emphasizing Nagios's ability to monitor systems and networks effectively. This experience laid the groundwork for enhancing infrastructure reliability and integrating advanced monitoring strategies in future projects.

Name: **Dev Gaonkar** Div/ Roll no: **D15C/ 12**

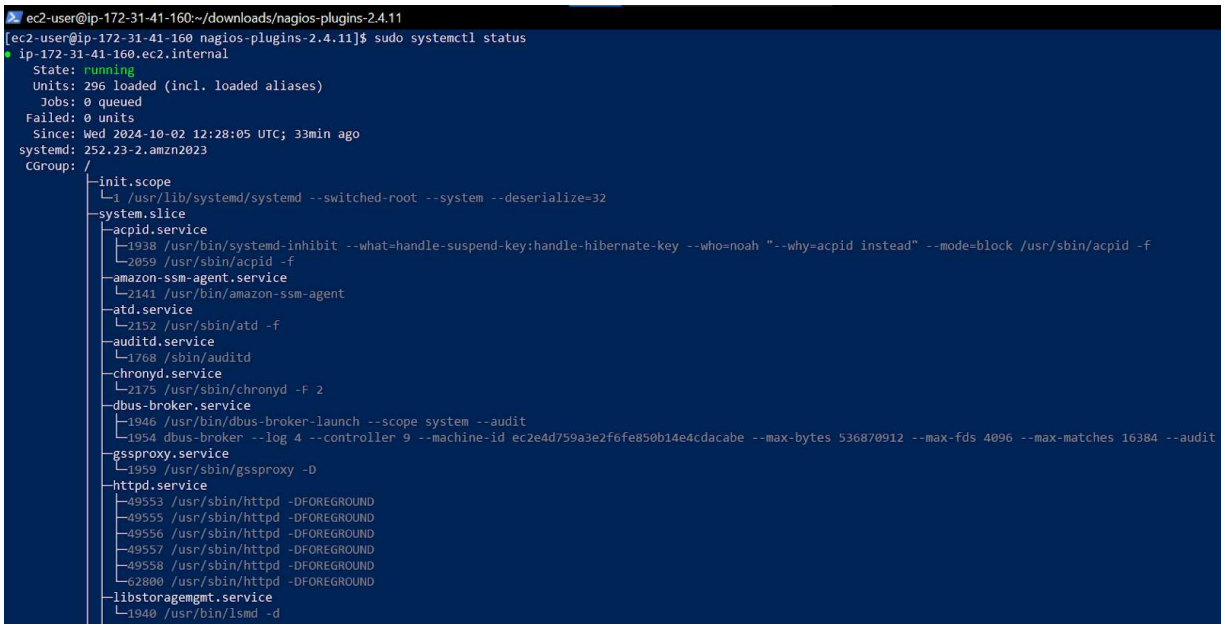
Experiment 10

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

Monitoring Using Nagios:

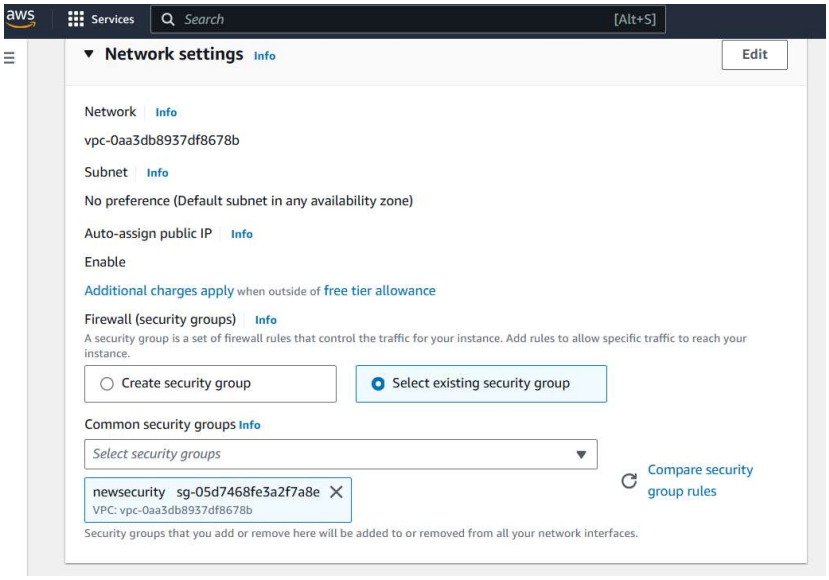
Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).

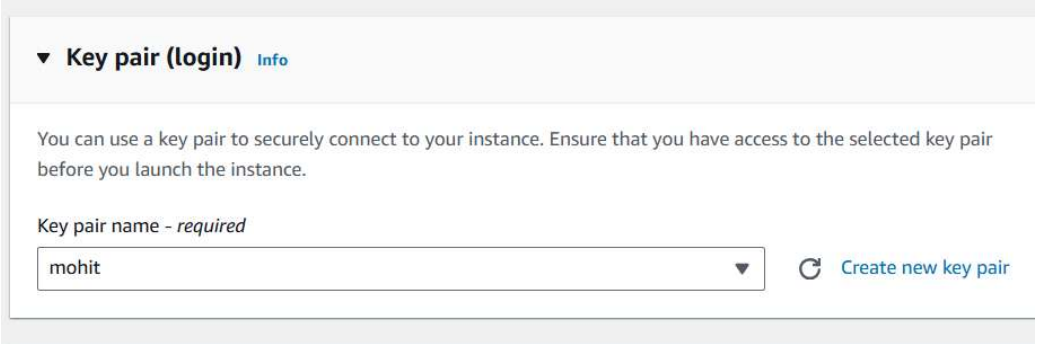
Run this command **sudo systemctl status**

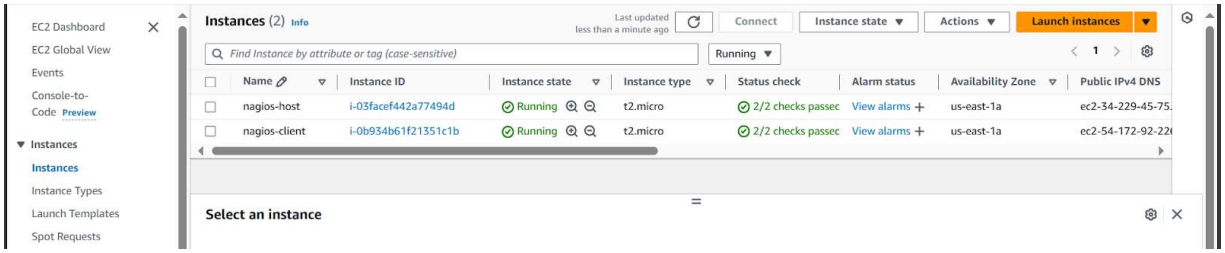


Step 2: Before we begin,

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 ‘nagios-client’ alongside the host.







Step 3: TO BE DONE IN THE Nagios-host TERMINAL

In the nagios-host terminal, run this command

ps -ef | grep nagios

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ ps -ef | grep nagios
ec2-user   63115    2315  0 13:03 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

To become a root user, run **'sudo su'** and make two directories using the following commands.

If one is running these commands in windows powershell, make sure that he/she copies it line by line as powershell might make an error while interpreting multiple lines

mkdir /usr/local/nagios/etc/objects/monitorhosts
mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-92-249 ~]$ sudo su
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-92-249 ec2-user]#
```

Copy the sample localhost.cfg file to linuxhost folder. Use the following mentioned command to achieve it
cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Open linuxserver.cfg using nano and make the following changes. This is a conf type file in which we will have to modify the configurations in way which will help us specify the hosts and clients to be monitored
nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Changes to be made:

1. Change the hostname to linux-server (EVERYWHERE ON THE FILE)
2. Change address to the public IP address of your LINUX CLIENT.
3. Change hostgroup_name under hostgroup to linux-servers1

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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linux-server
    alias                   localhost
    address                 54.172.92.226
}



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

```
# Define an optional hostgroup for Linux machines

define hostgroup {

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

IMP: Everywhere else on the file, change the hostname to linux-server instead of localhost.

Open the Nagios Config file and add the following line
nano /usr/local/nagios/etc/nagios.cfg

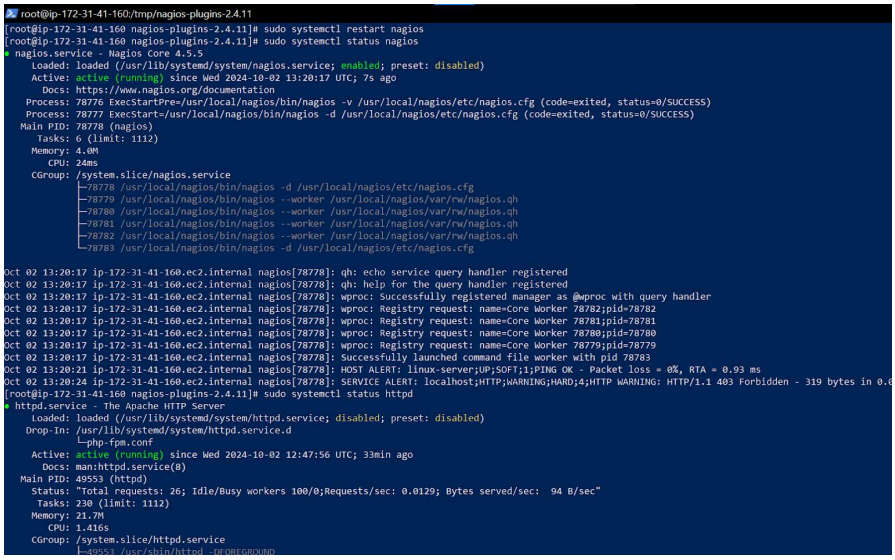
Add the following line in the file and save
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

Verify the configuration files by running the following command
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
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 16 services.
        Checked 2 hosts.
        Checked 2 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 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-41-160 nagios-plugins-2.4.11]#
```

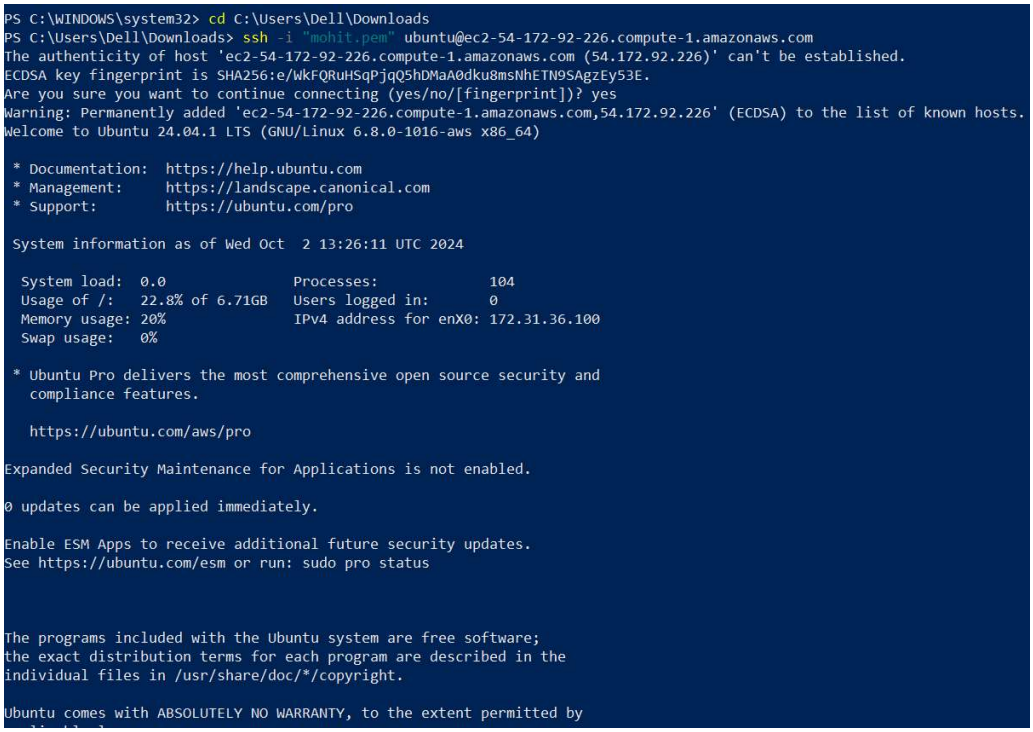
You are good to go if there are no errors.
Restart the nagios service
service nagios restart
And by running sudo systemctl status nagios, we can again check whether our server is running or not



Step 4: TO BE DONE IN THE Nagios-client TERMINAL

Now it is time to switch to the client machine.

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

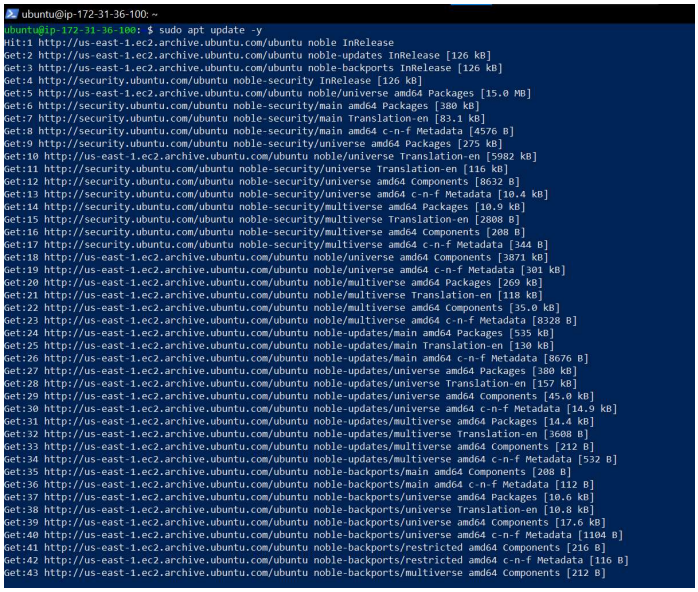


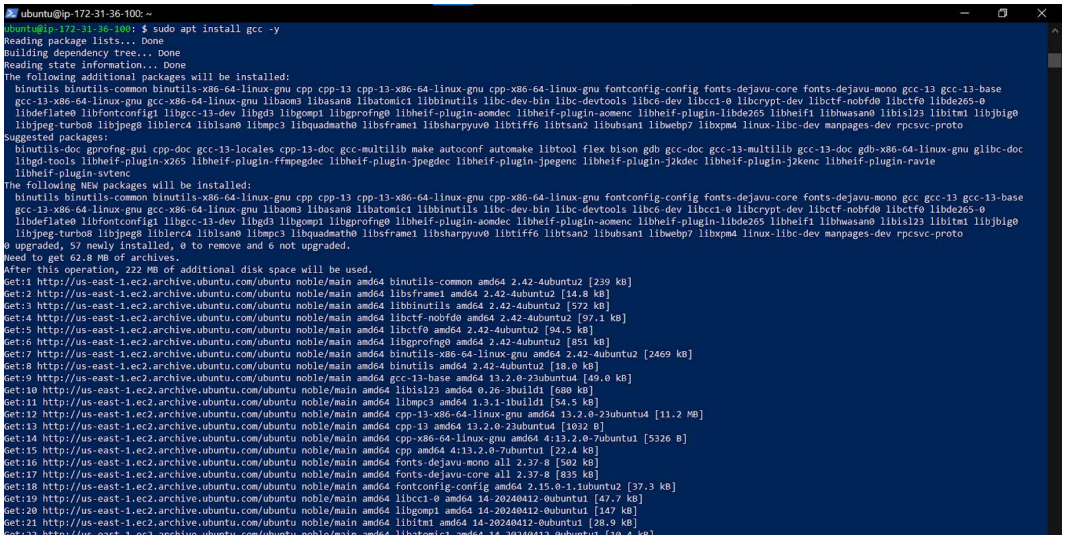
Make a package index update and install gcc, nagios-nrpe-server and the plugins. Run the following commands to achieve the same.

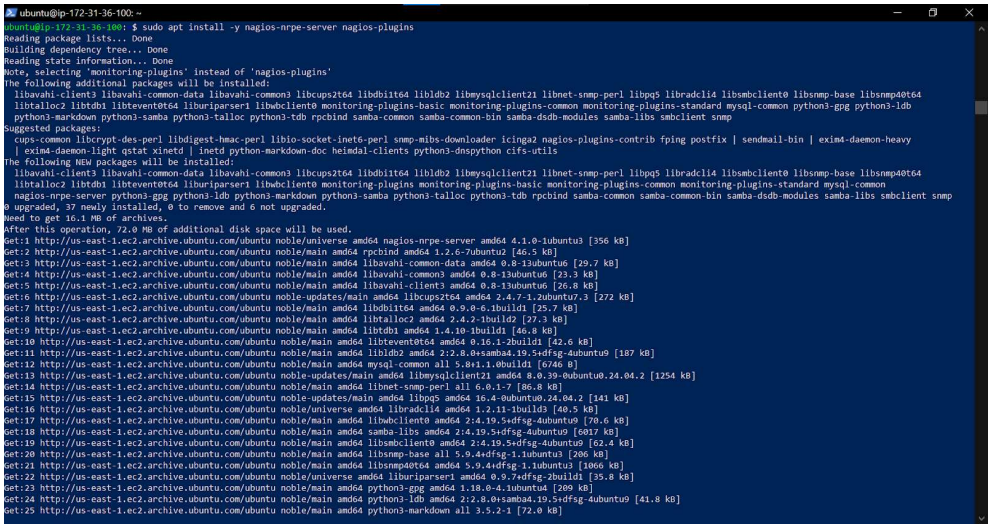
sudo apt update -y

sudo apt install gcc -y

sudo apt install -y nagios-nrpe-server nagios-plugins



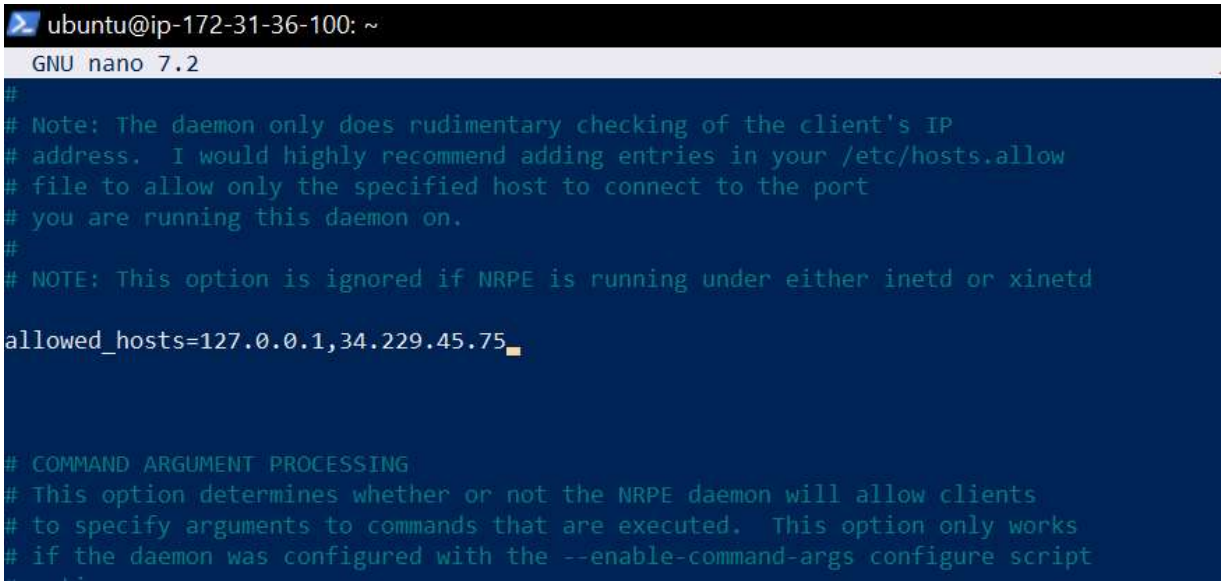




Open nrpe.cfg file to make changes.

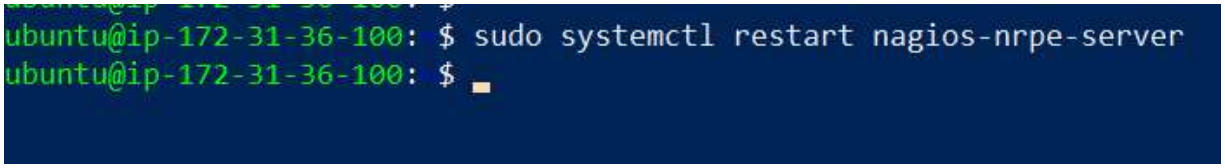
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so



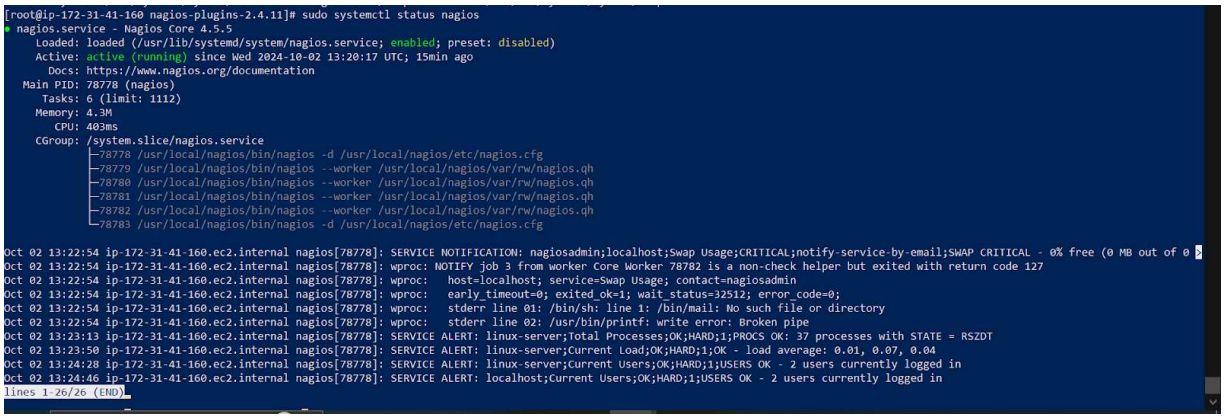
Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server



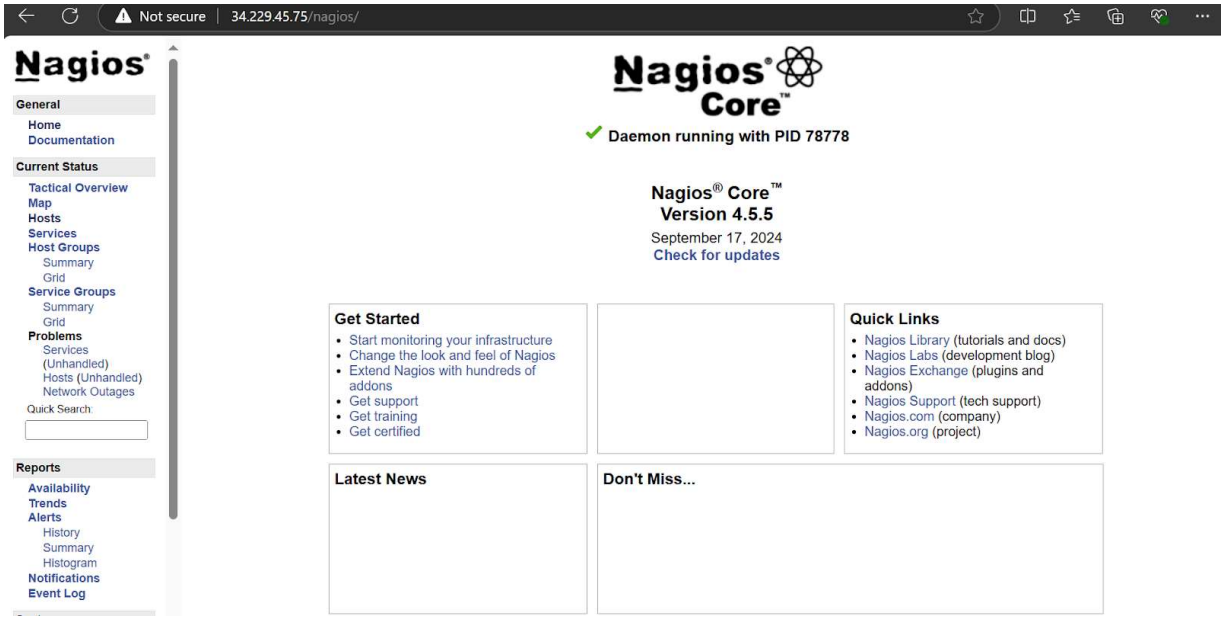
Run the following command in the Nagios-host terminal

sudo systemctl status nagios

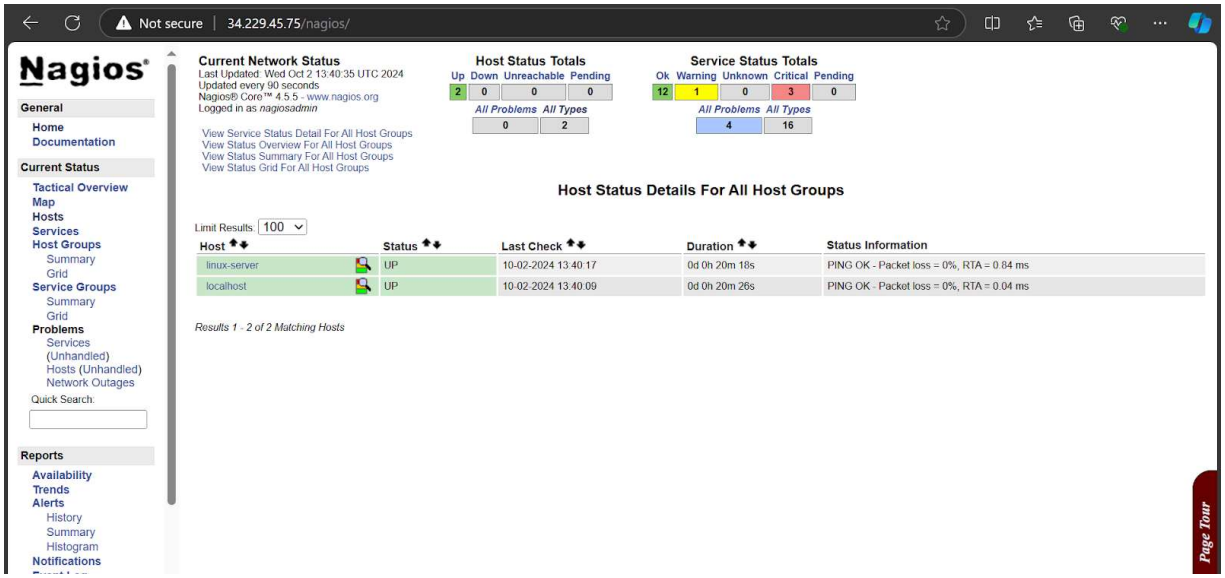


Step 5: Visiting your nagios server using your nagios-host ip address

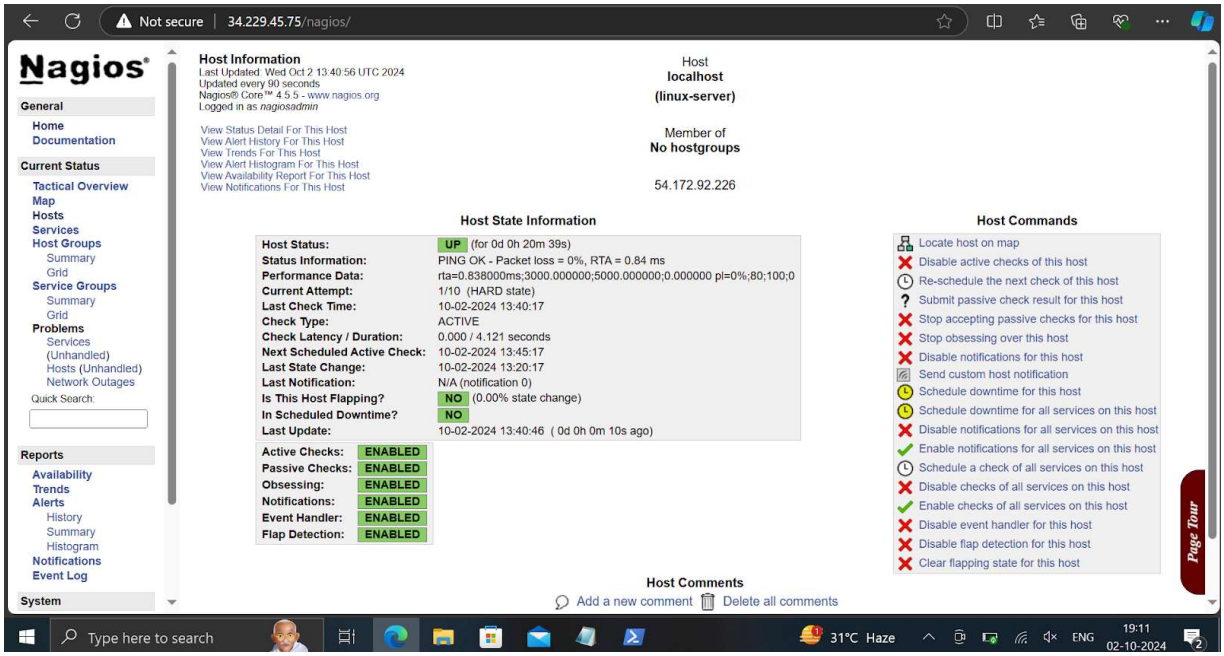
Open up your browser and look for http://<public_ip_address_of_nagios-host>/nagios



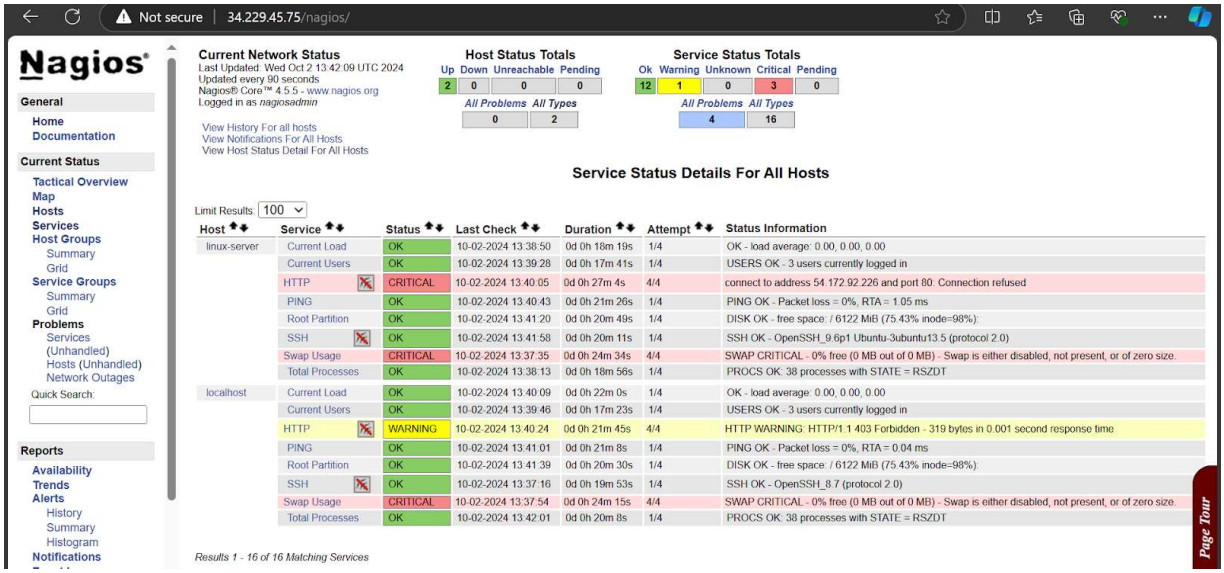
Click on Hosts.



Click on linux-server to view host information



We can even navigate to the services section, which explicitly mentions the status, duration, checks, information about the numerous services present on our hosts



Conclusion: In conclusion, the experiment focused on monitoring ports, services, and a Linux server using Nagios. Through the step-by-step process, we successfully configured Nagios to monitor essential network services on the Linux server. By setting up both the Nagios host and client, we were able to track system performance, ensure service availability, and monitor key metrics like CPU and memory usage

Name: **Dev Gaonkar** Div/ Roll no: **D15C/ 12**

Experiment 11

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

Theory:

AWS Lambda

A fully managed, serverless computing service where you run code without provisioning or managing servers. Lambda automatically scales your application based on the number of incoming requests or events, ensuring efficient resource utilization. You are only charged for the time your code is running, with no upfront cost, making it cost-effective for on-demand workloads.

Lambda Workflow

- **Create a Function**: Write the function code and define its handler (entry point). You can use the AWS Console, CLI, or upload a deployment package.
- **Set Event Sources**: Define how the function is triggered (e.g., when an object is uploaded to S3 or a DynamoDB table is updated).
- **Execution**: When triggered, Lambda runs your function, executes the logic, and automatically scales to handle the incoming event volume.
- **Scaling and Concurrency**: Lambda scales automatically by launching more instances of the function to handle simultaneous invocations. There are also options for configuring **reserved concurrency** to manage traffic.
- **Monitoring and Logging**: Lambda integrates with Amazon CloudWatch for logging and monitoring. Logs for each invocation are sent to CloudWatch, allowing you to track performance and troubleshoot errors.

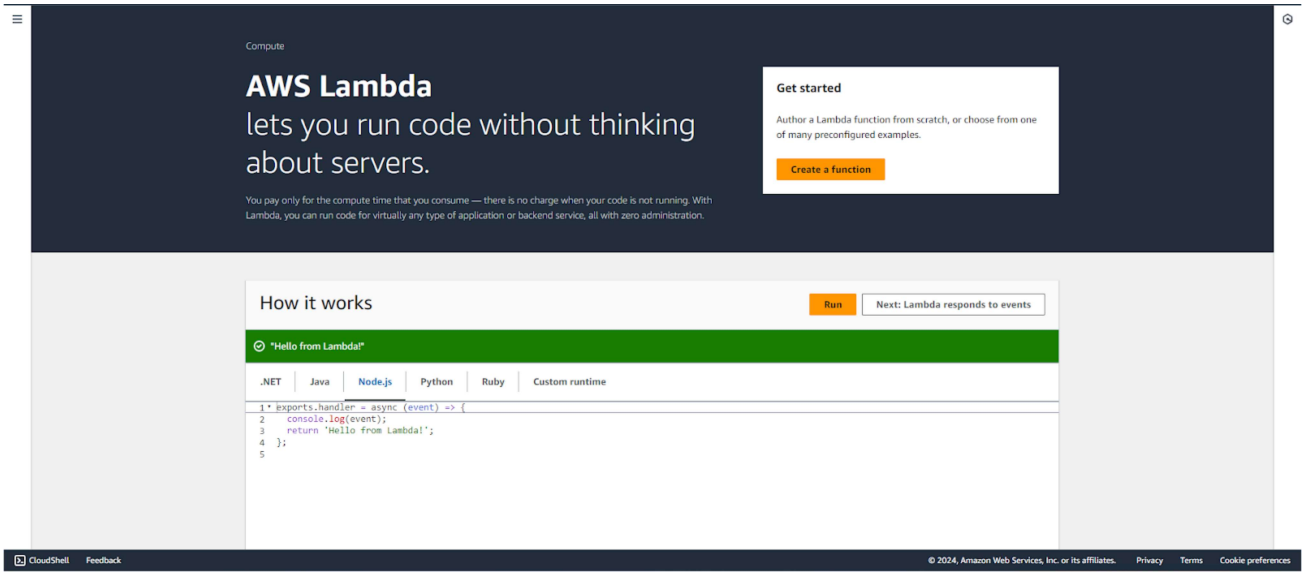
AWS Lambda Functions

- **Python**: Great for quick development with its rich standard library and support for lightweight tasks.
- **Java**: Typically used for more complex, compute-intensive tasks. While it's robust, cold start times can be higher.
- **Node.js**: Excellent for I/O-bound tasks like handling APIs or streaming data, with fast startup times and efficient memory usage.

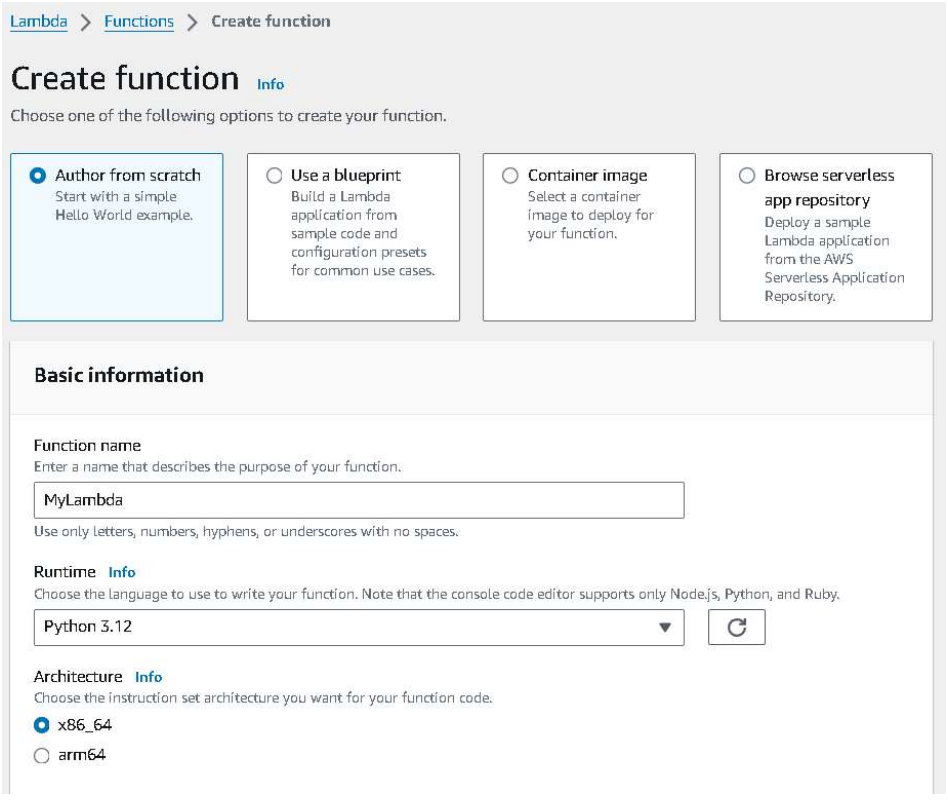
Prerequisites: AWS Personal/Academy Account

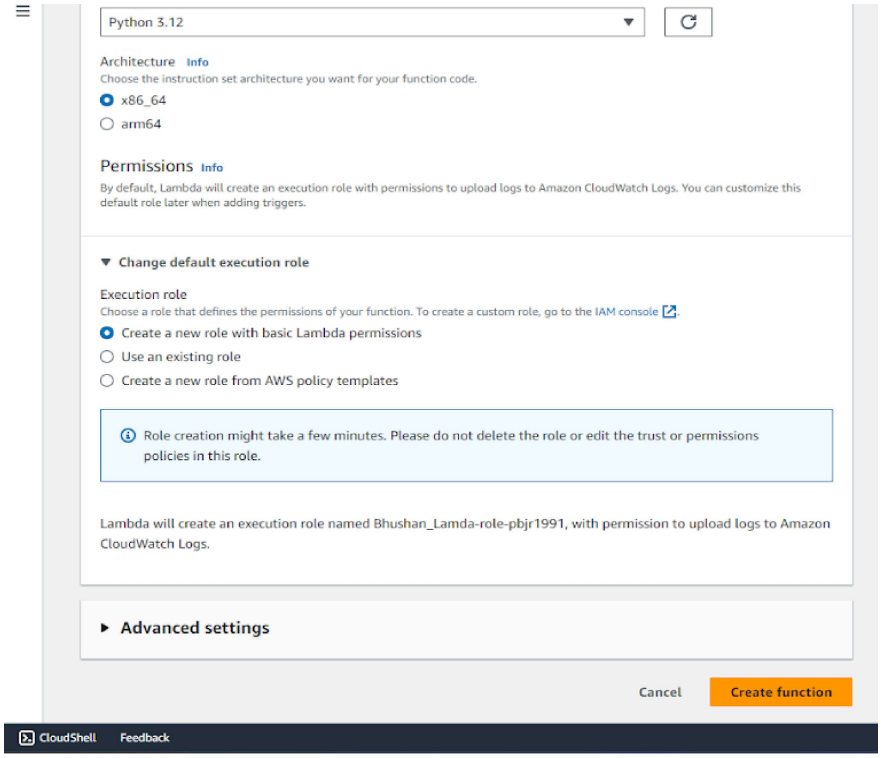
Steps To create the lambda function:

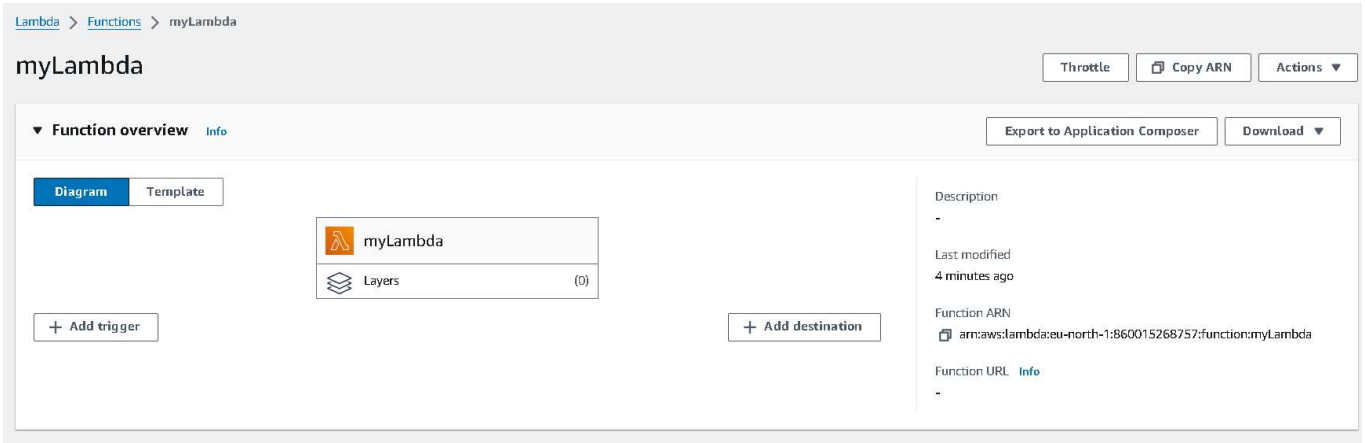
Step 1: Login to your AWS Personal/Academy Accout.Open lambda and click on create function button.



Step 2: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.

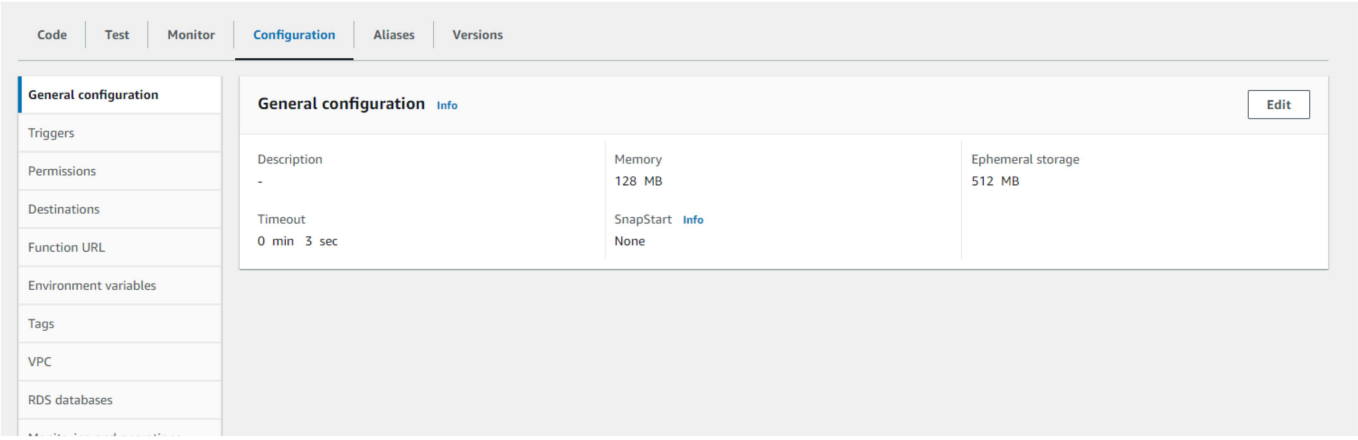



Python 3.12
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
Change default execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.
Lambda will create an execution role named Bhushan_Lamda-role-pbjr1991, with permission to upload logs to Amazon CloudWatch Logs.
Advanced settings
Cancel
Create function
CloudShell
Feedback

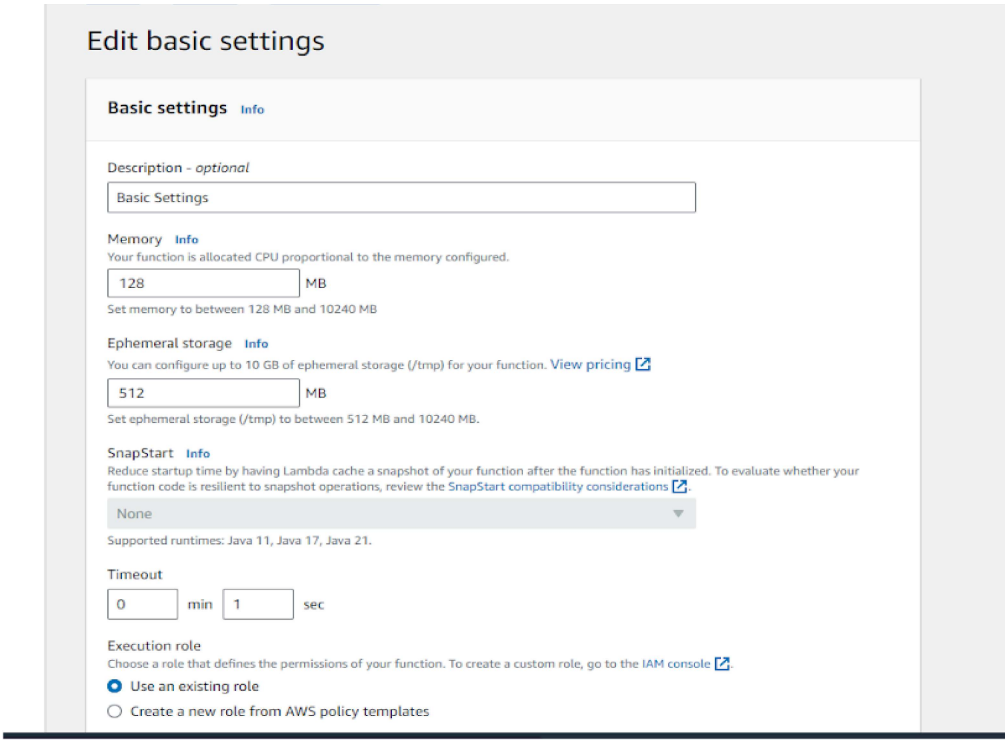

Lambda > Functions > myLambda
myLambda
Throttle
Copy ARN
Actions
Function overview Info
Export to Application Composer
Download
Diagram
Template
myLambda
Layers (0)
Add trigger
Add destination
Description
-
Last modified
4 minutes ago
Function ARN
arn:aws:lambda:eu-north-1:860015268757:function:myLambda
Function URL Info
-


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Vari
Environment
myLambda
lambda_function.py
import json
def lambda_handler(event, context):
# TODO implement
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}

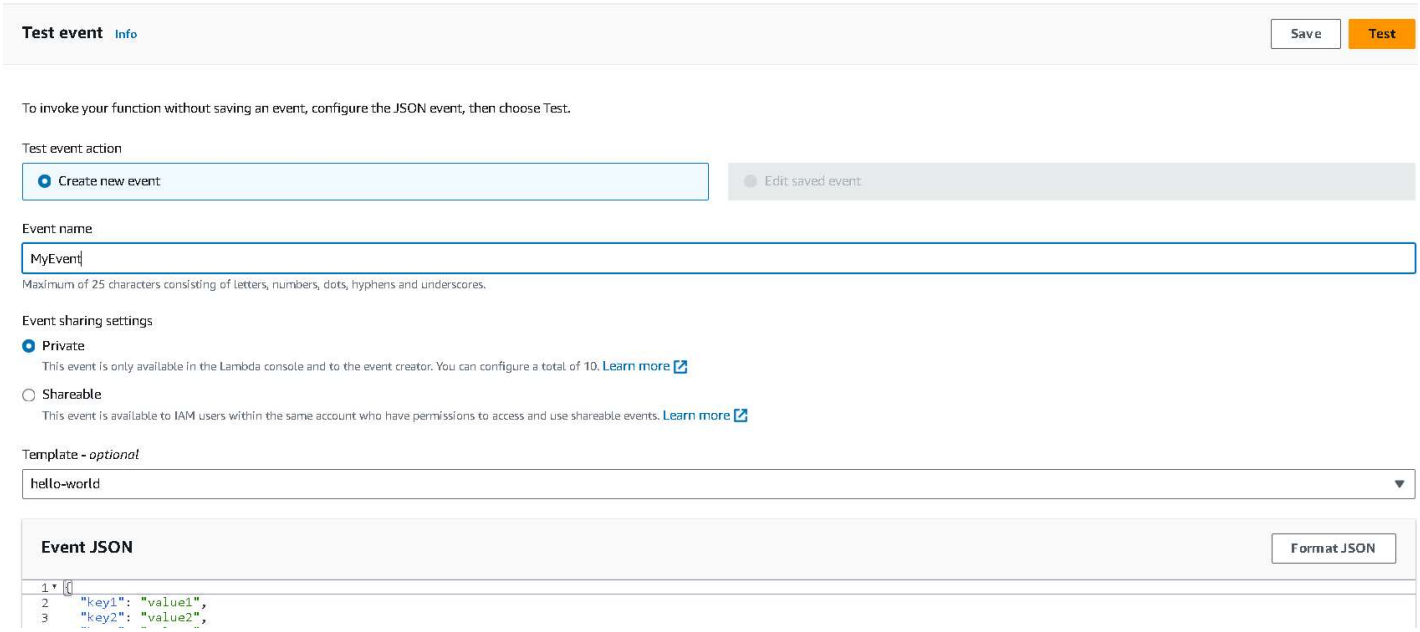
So See or Edit the basic settings go to configuration then click on edit general setting.



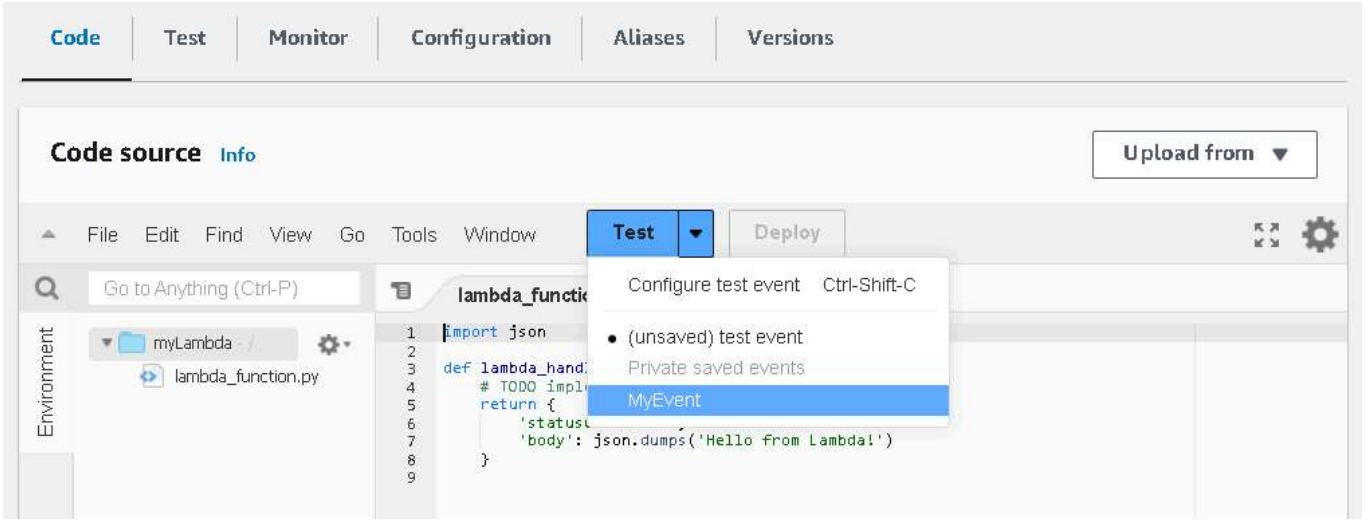
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

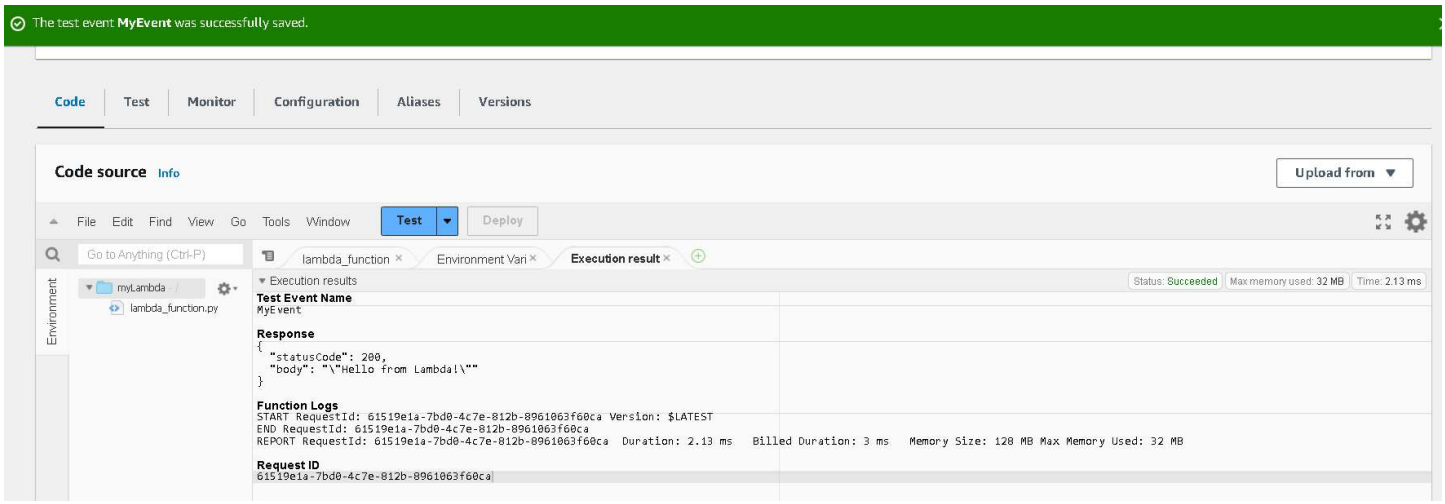


Step 3: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.



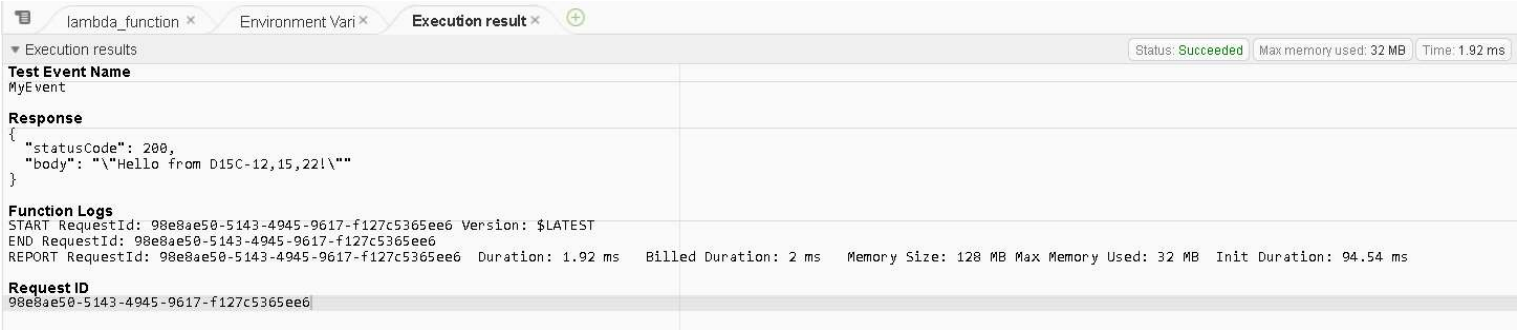
Step 4: Now In the Code section select the created event from the dropdown of test then click on test . You will see the below output.





Step 5: You can edit your lambda function code. I have changed the code to display the new String.

Step 6: Now click on the test and observe the output. We can see the status code 200 and your string output and function logs. On successful deployment.



Conclusion:

In this experiment, we successfully implemented an AWS Lambda function, covering all the key steps involved. Starting with the function's setup in Python, we configured essential settings such as adjusting the timeout to 1 second. A test event was then created, followed by deploying the function and verifying its output. We also made code modifications to the Lambda function, redeployed it, and observed the real-time effects of these changes. This hands-on experience highlighted the ease and adaptability of AWS Lambda for building serverless applications, enabling developers to concentrate on writing code while AWS handles infrastructure and scalability.

Name: **Dev Gaonkar** Div/ Roll no: **D15C/ 12**

Experiment 12

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

Theory:

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.

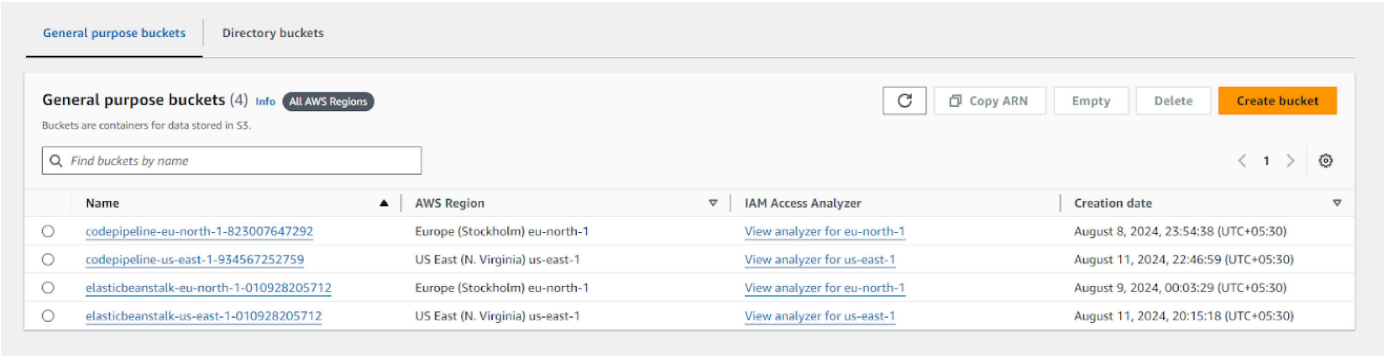
4. **Configure S3 Trigger:**
   - Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

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.

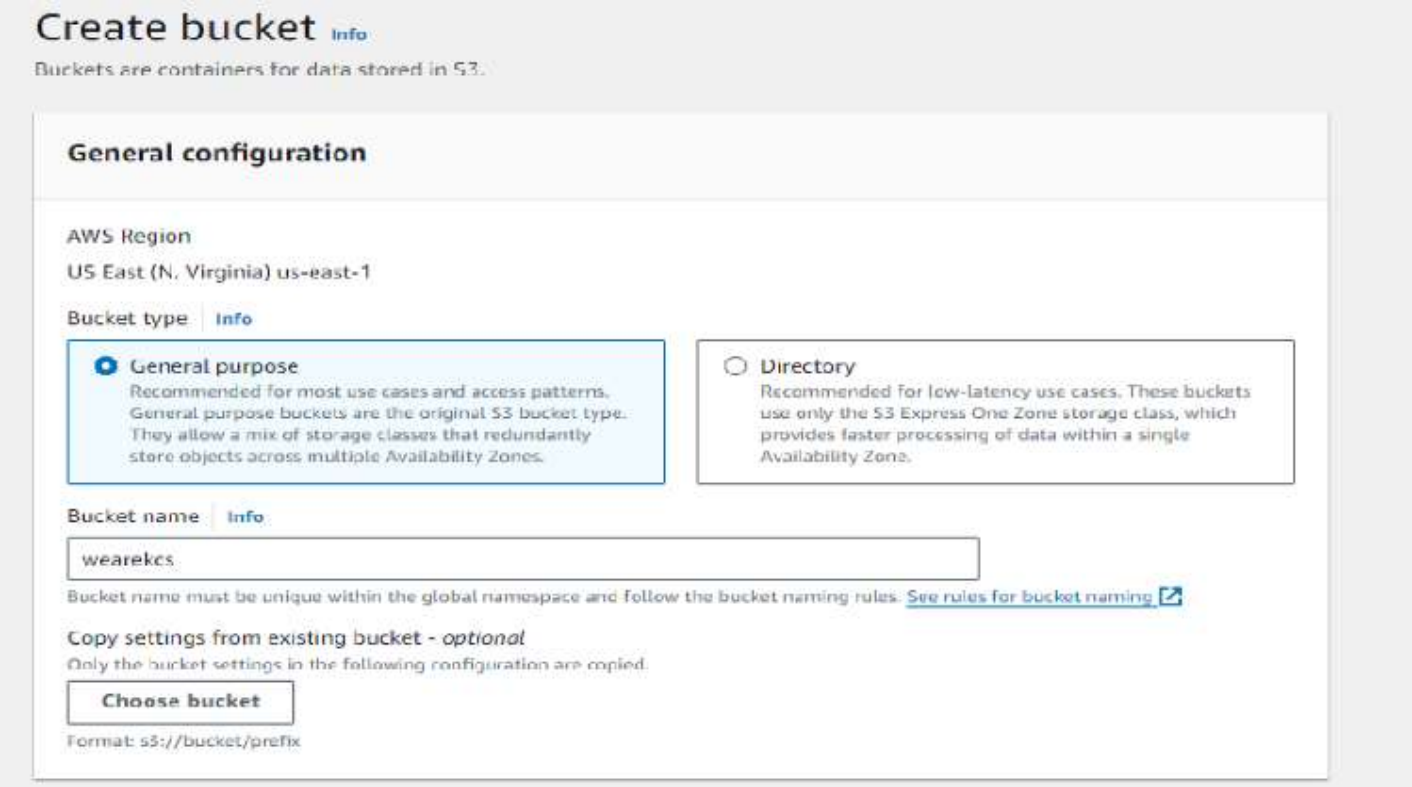
Prerequisites: AWS Personal Account

Steps To create the lambda function:

Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.



Step 2: Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other this to default.



Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- (●) ACLs disabled (recommended)
  All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ( ) ACLs enabled
  Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

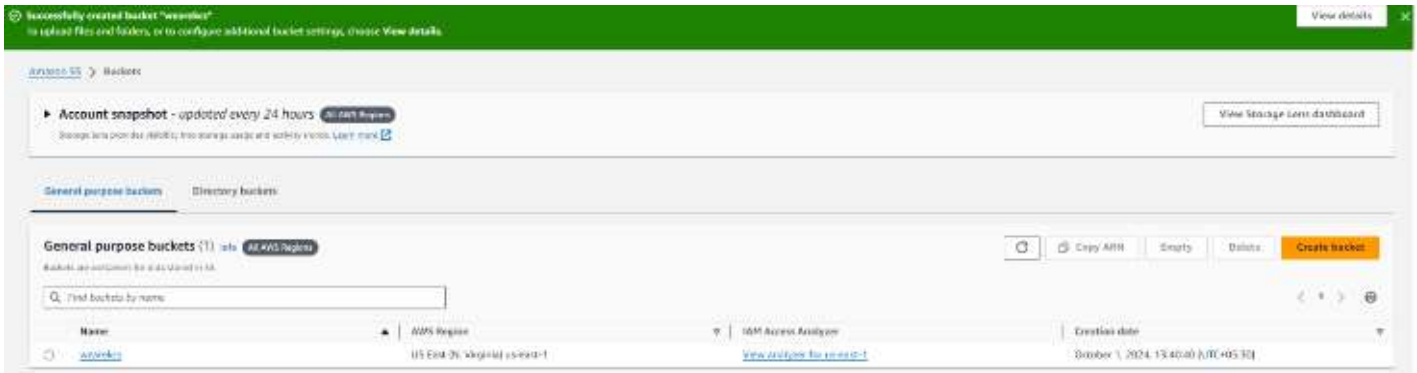
Object Ownership

Bucket owner enforced

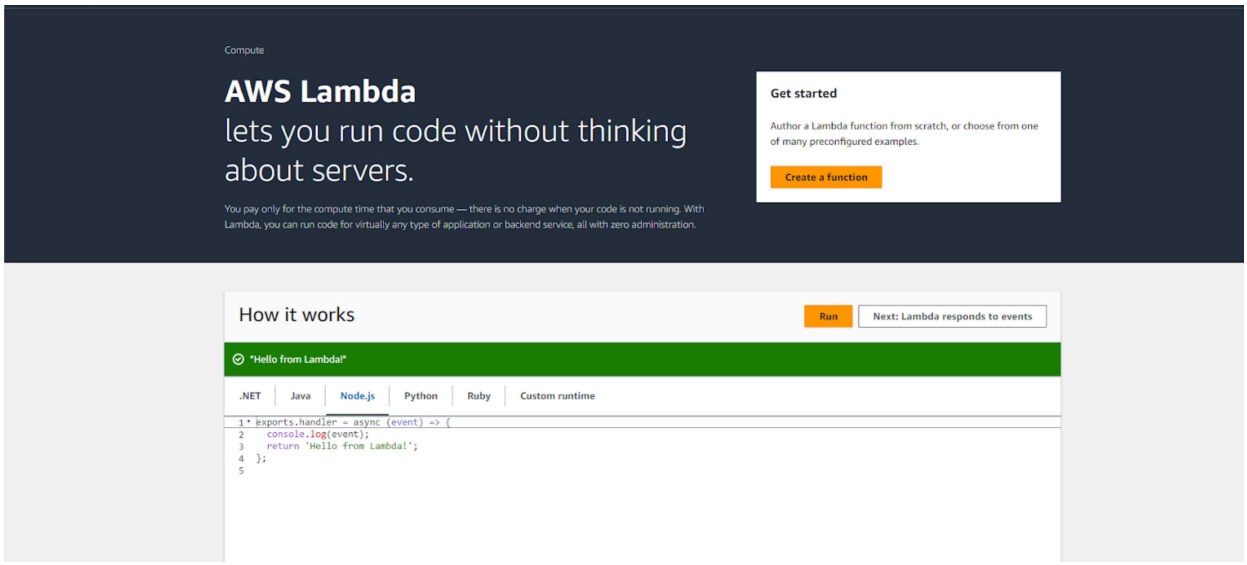
Block Public Access settings for this bucket

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 this bucket and its 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 this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

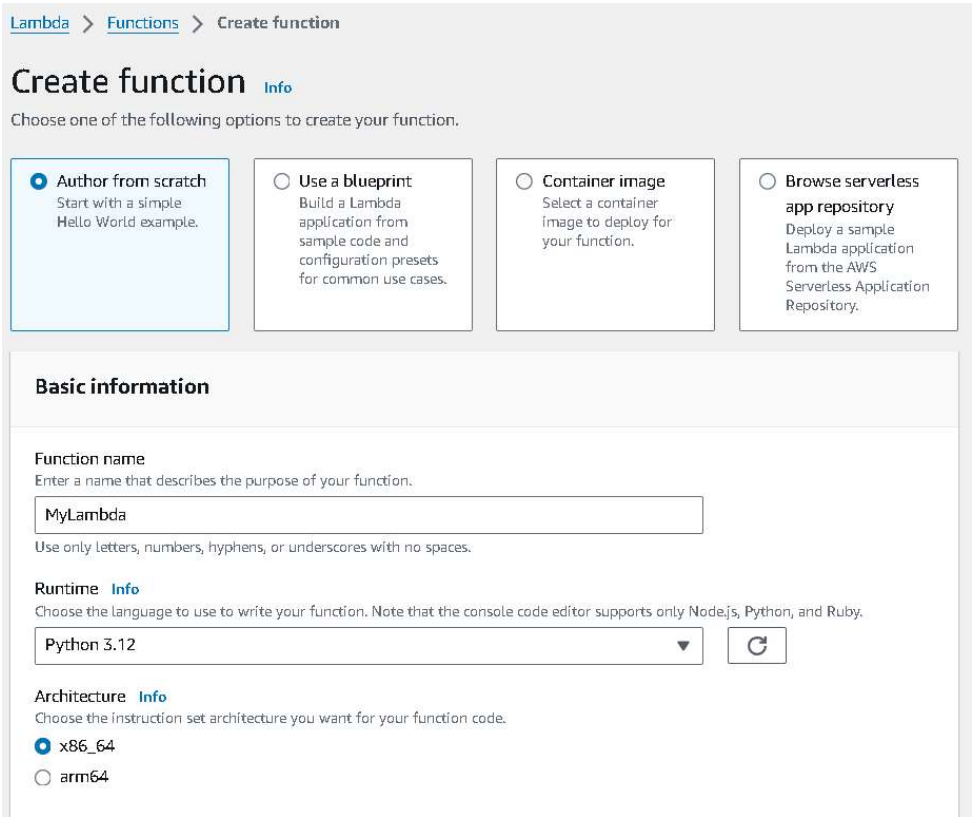
- [ ] **Block *all* public access**
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - [ ] **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
    S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - [ ] **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
    S3 will ignore all ACLs that grant public access to buckets and objects.
  - [ ] **Block public access to buckets and objects granted through *new* public bucket or access point policies**
    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - [ ] **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

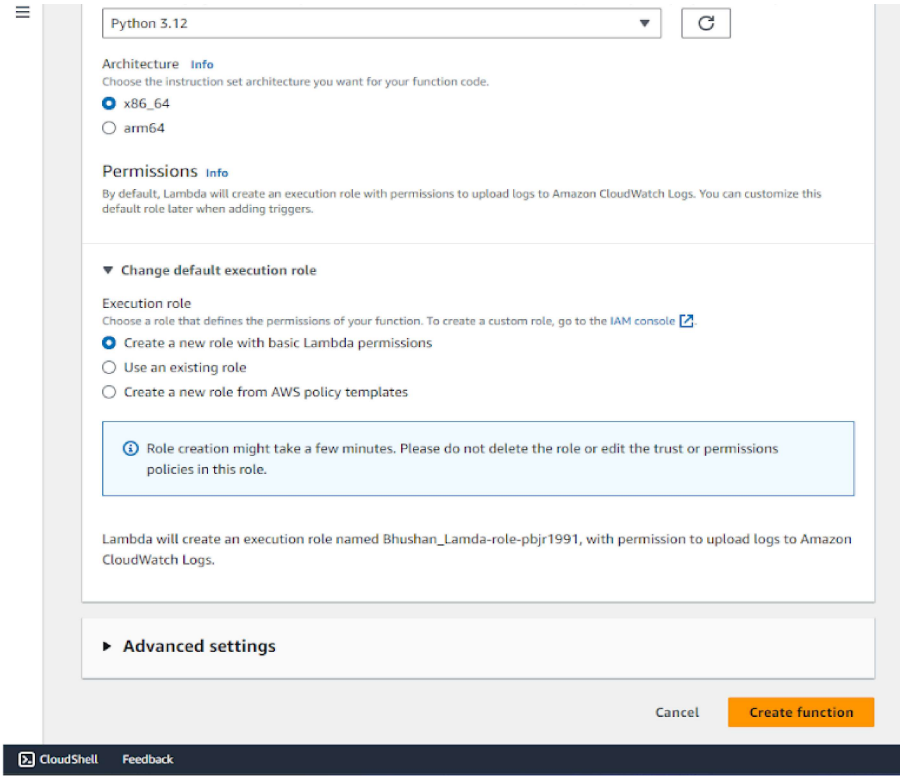


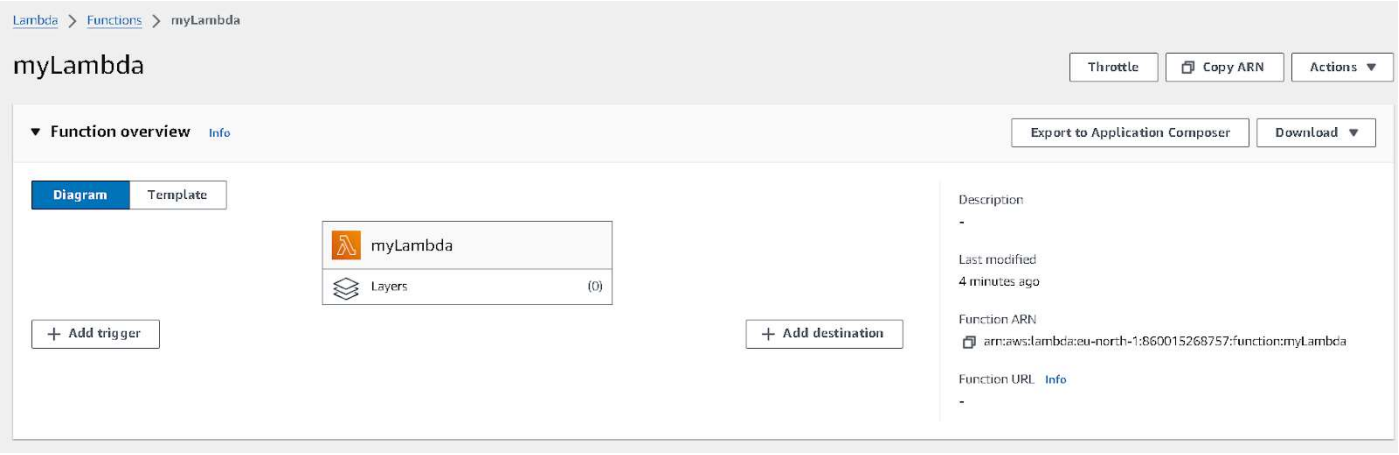
Step 3: Open lambda console and click on create function button.

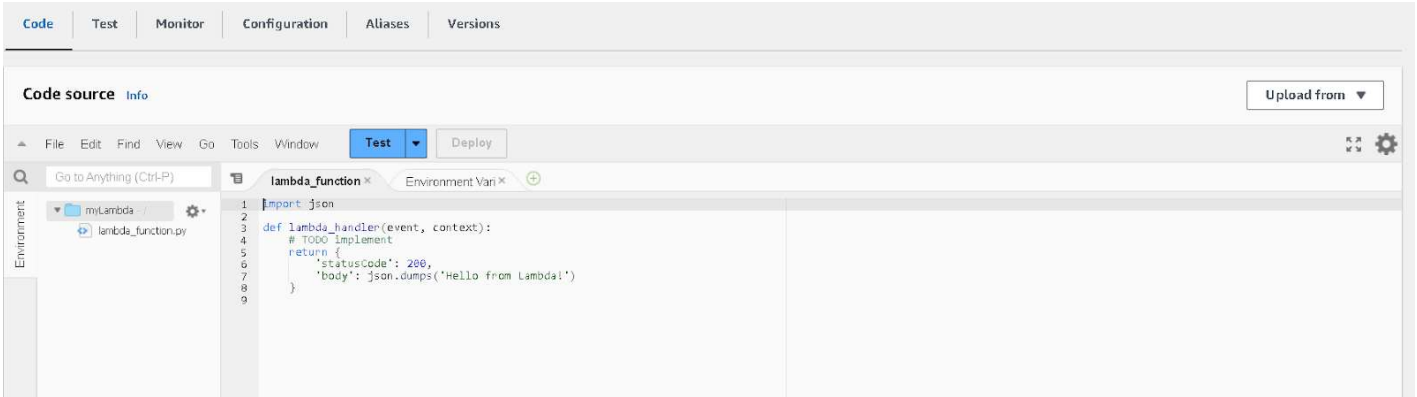


Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12
, Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.

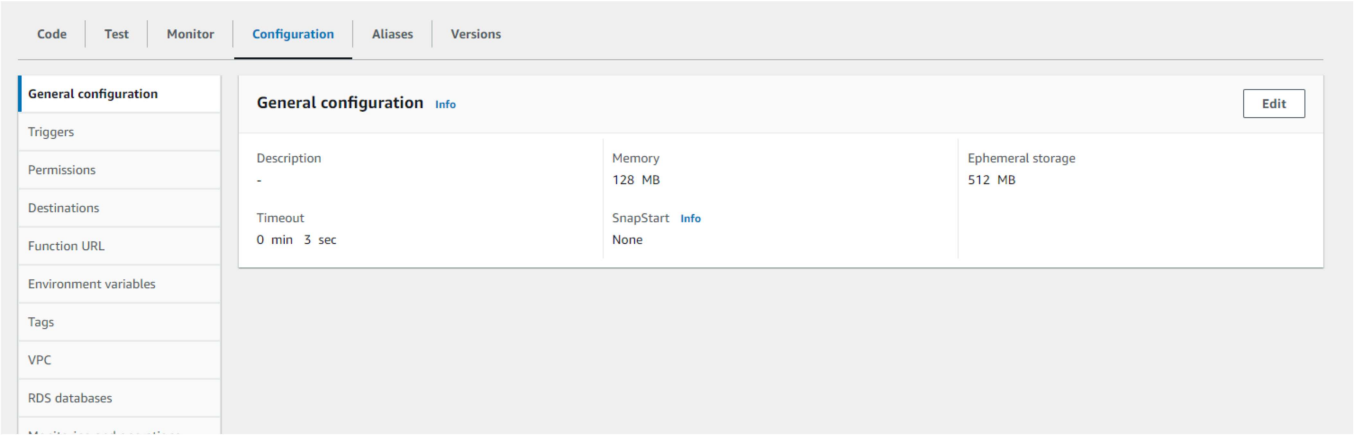



Python 3.12
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
Change default execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.
Lambda will create an execution role named Bhushan_Lamda-role-pbjr1991, with permission to upload logs to Amazon CloudWatch Logs.
Advanced settings
Cancel
Create function
CloudShell
Feedback

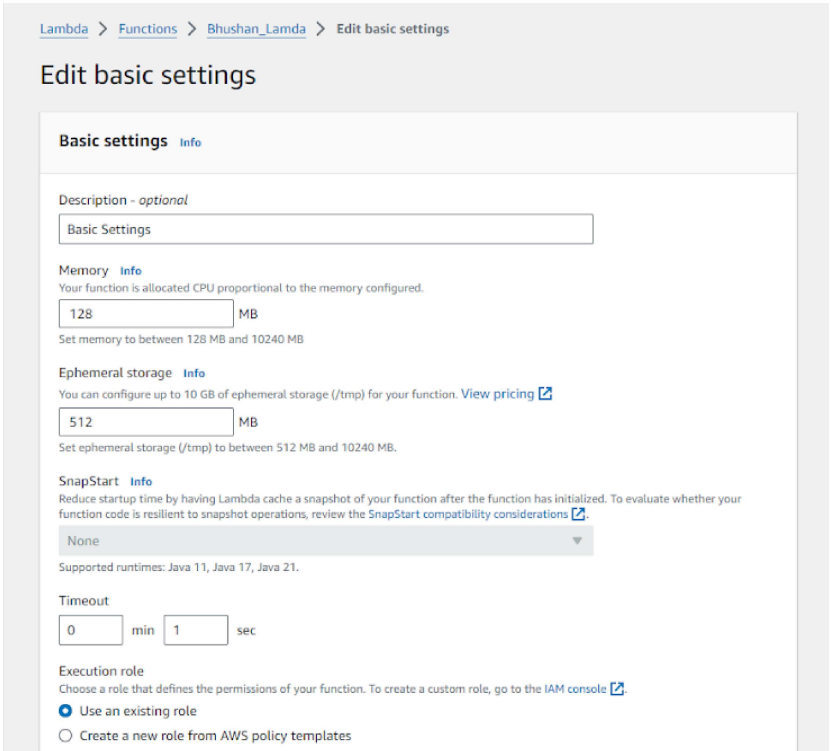

Lambda > Functions > myLambda
myLambda
Throttle
Copy ARN
Actions
Function overview Info
Export to Application Composer
Download
Diagram
Template
myLambda
Layers
(0)
Add trigger
Add destination
Description
-
Last modified
4 minutes ago
Function ARN
arn:aws:lambda:eu-north-1:860015268757:function:myLambda
Function URL Info
-


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Vari
Environment
myLambda
lambda_function.py
import json
def lambda_handler(event, context):
# TODO implement
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}

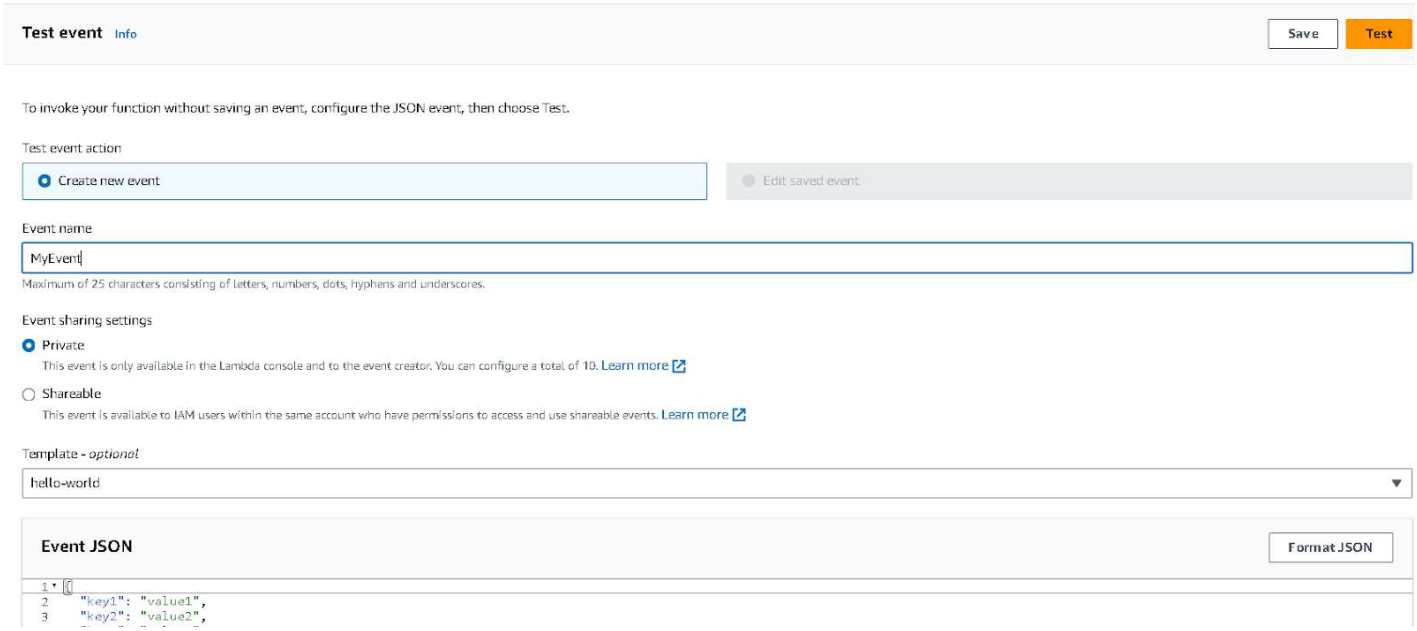
So See or Edit the basic settings go to configuration then click on edit general setting.

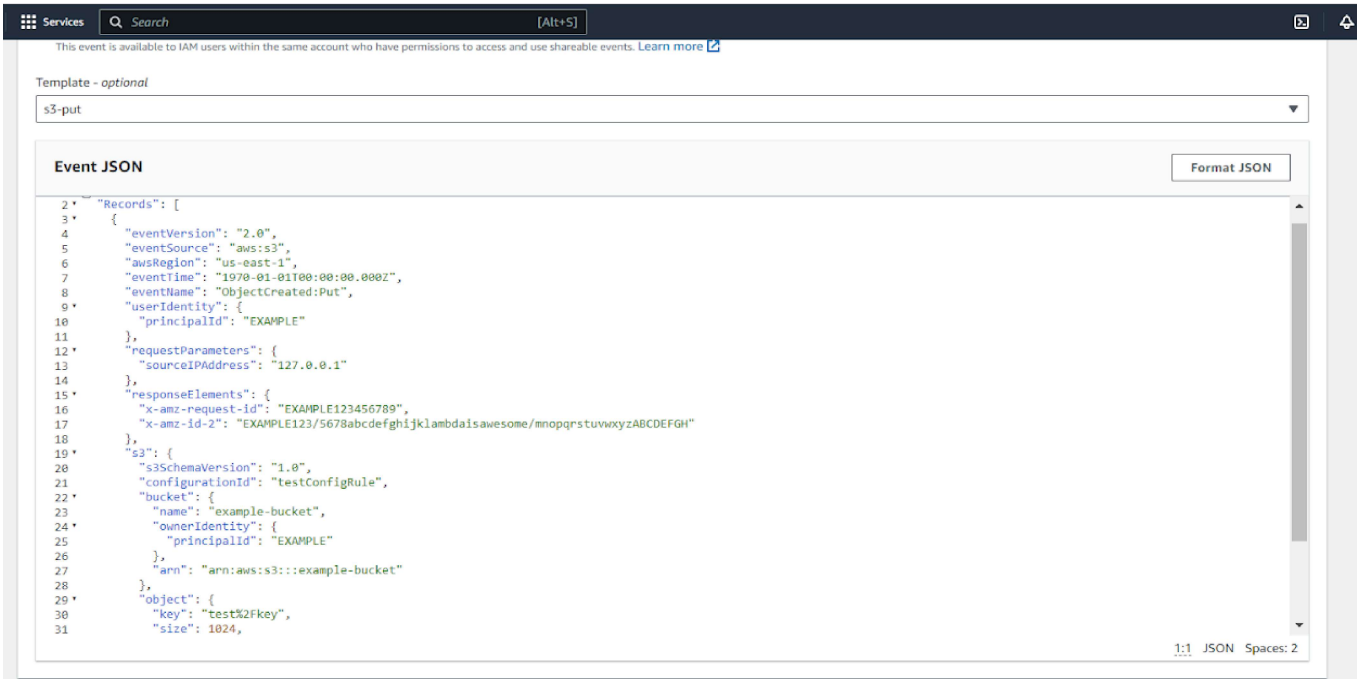


Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

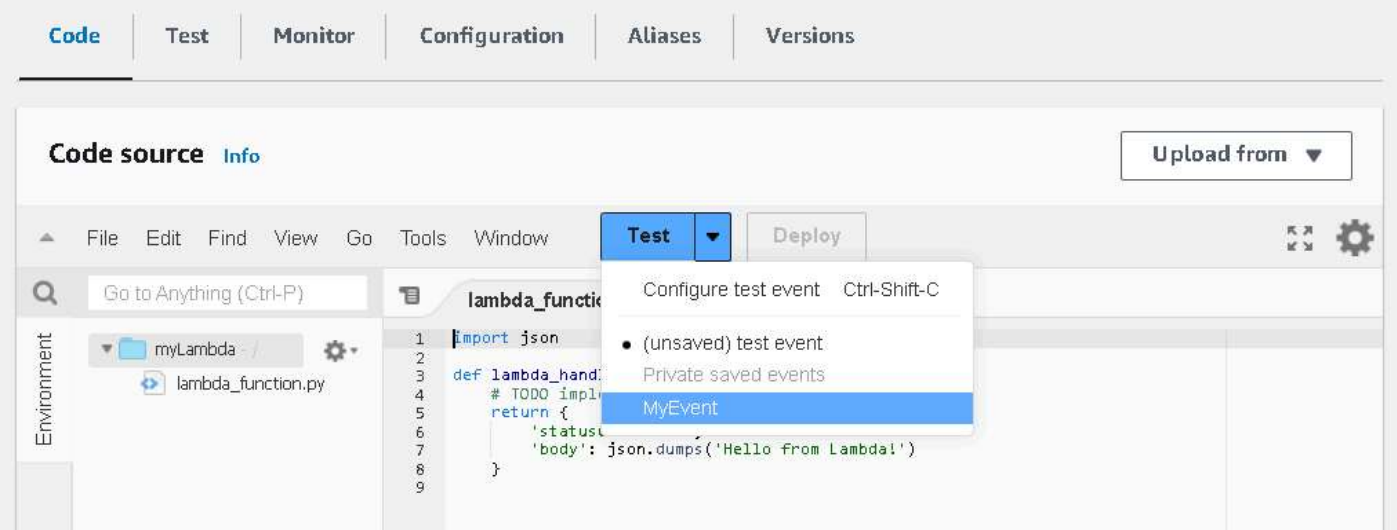


Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

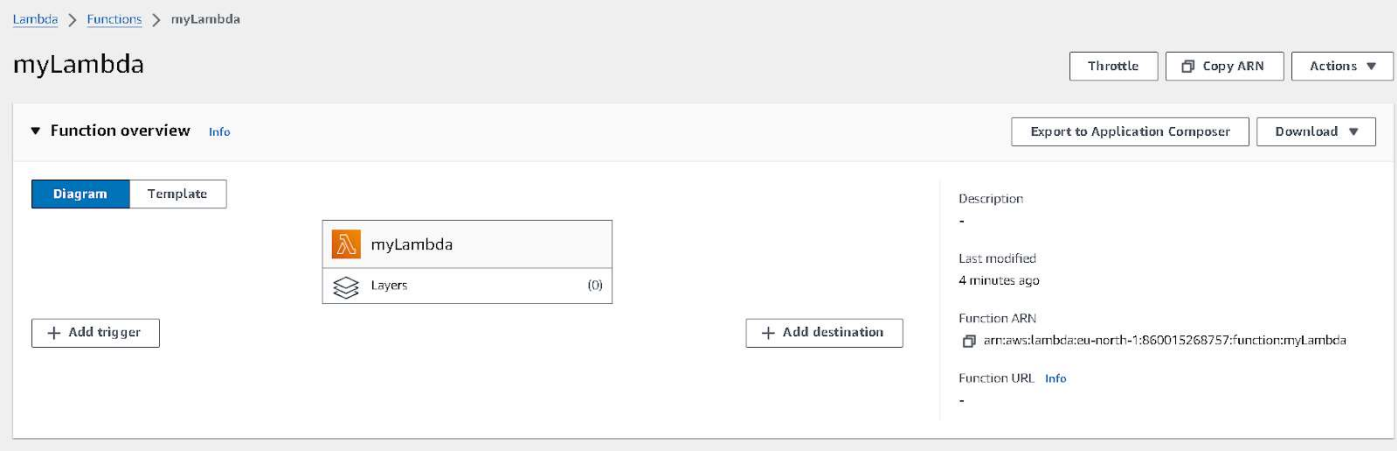




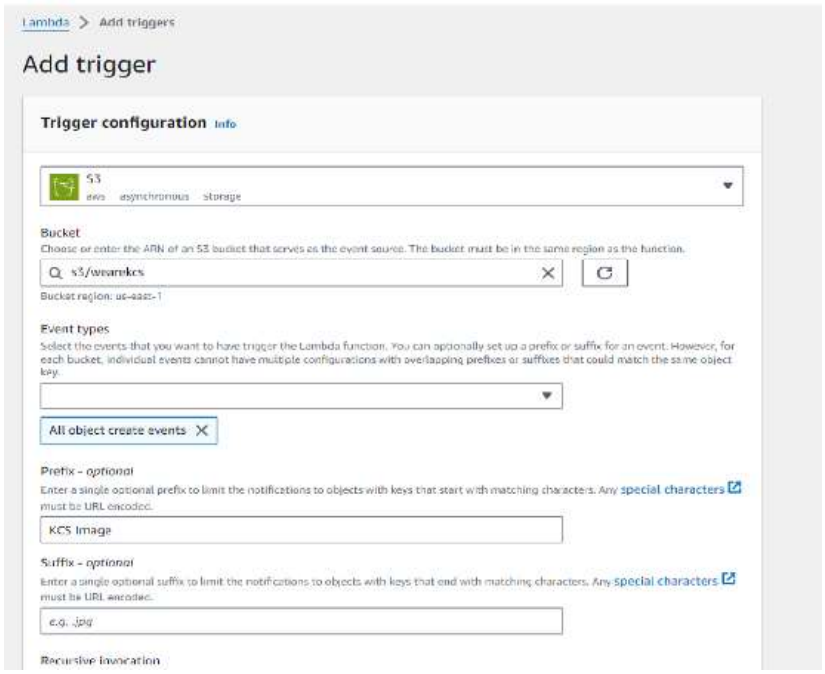
Step 6: Now In Code section select the created event from the dropdown .



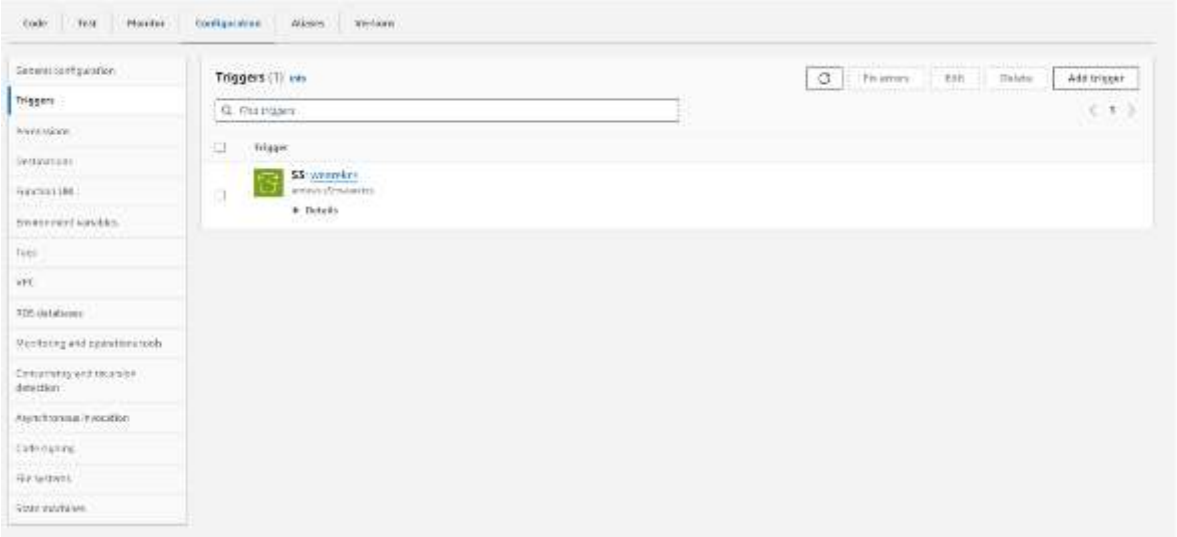
Step 7: Now In the Lambda function click on add tigger.



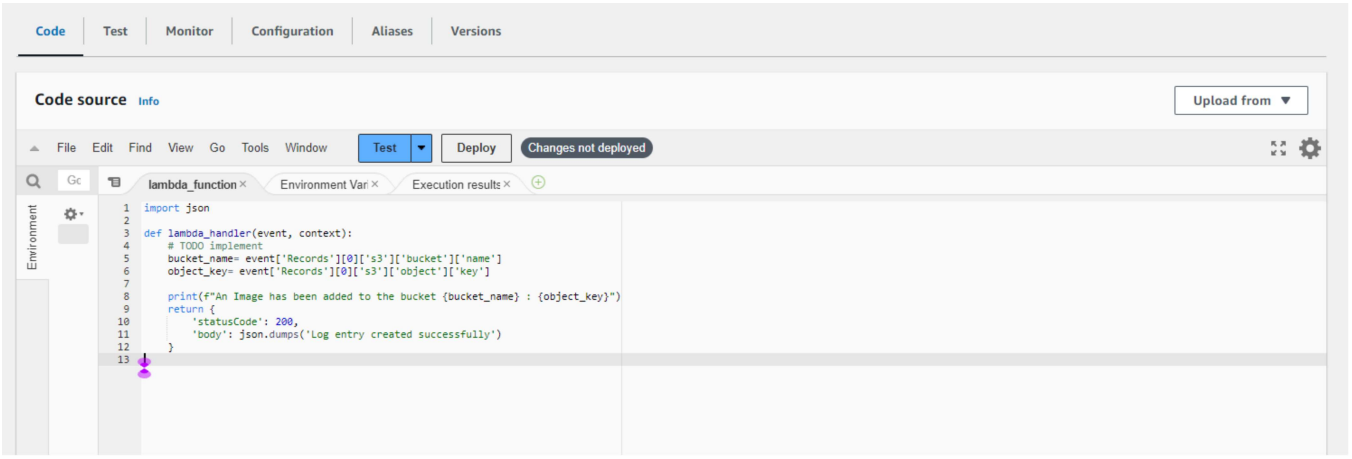
Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to image.


Lambda > Add triggers
Add trigger
Trigger configuration Info
S3
aws asynchronous storage
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/weareks
Bucket region: us-east-1
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
KCS Image
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.
e.g. .jpg
Recursive invocation

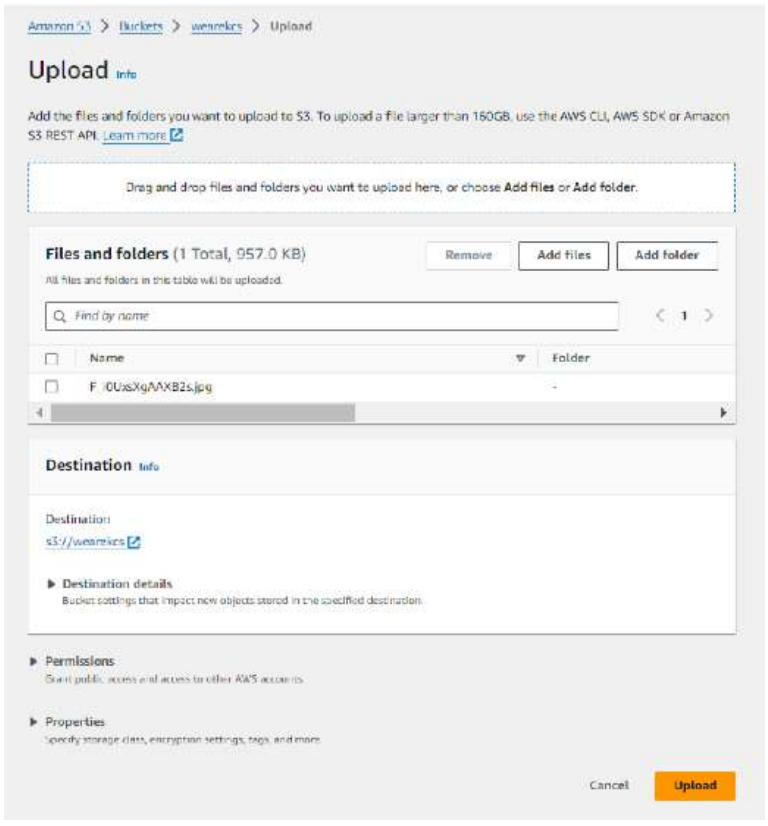

Function overview Info
Export to Application Composer
Download
Diagram
Template
myLambda
Layers (0)
S3
Add trigger
Add destination
Description
Basic Settings
Last modified
1 hour ago
Function ARN
arn:aws:lambda:us-east-1:010928205712:function:Bhushan_Lamda
Function URL Info
-


Triggers (1) Info
Add trigger
S3: weareks
Details

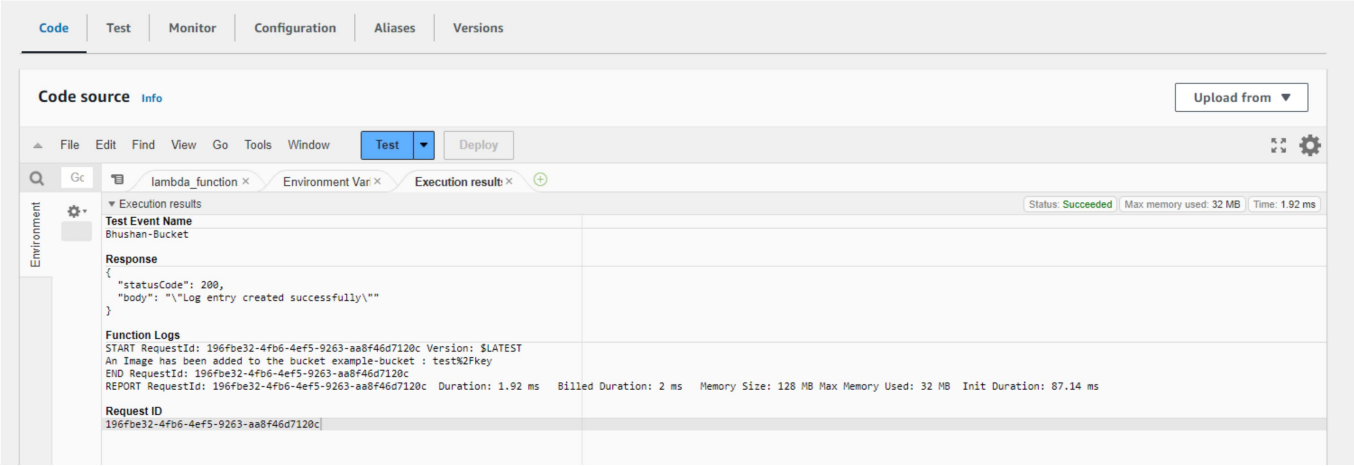
Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.



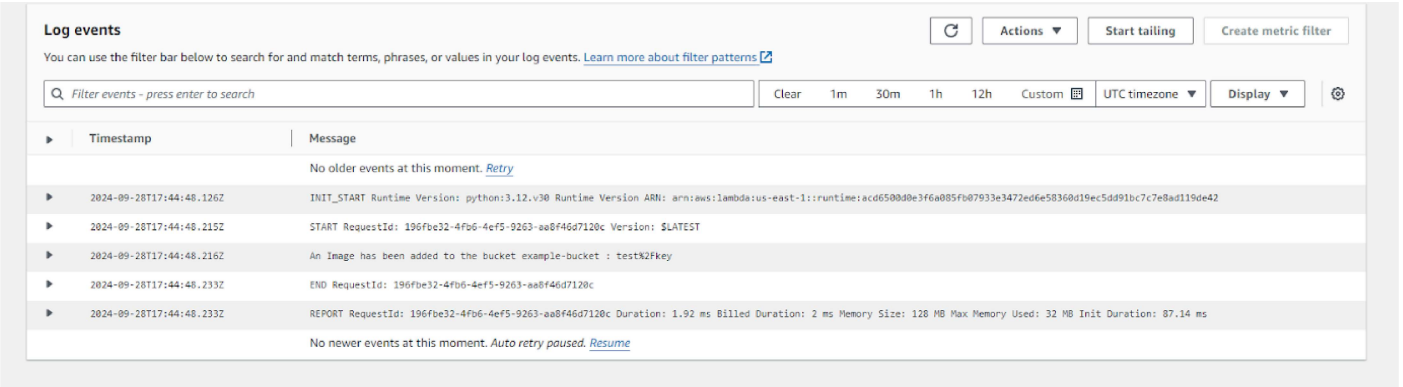
Step 9: Now upload any image to the bucket.



Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3.



Step 11: Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.



Conclusion:

In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It is important to note that we have to select S3-put template in the event otherwise code will give an error.The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.