RUJUTA MEDHI **D15A-28**

EXPERIMENT:01

Aim: Utilize AWS CodePipeline to deploy a Sample Application on an EC2 instance using AWS CodeDeploy.

Theory:

Amazon Elastic Compute Cloud (Amazon EC2) offers flexible and scalable computing power in the Amazon Web Services (AWS) Cloud. By using Amazon EC2, businesses can reduce the need for physical hardware, thereby lowering costs and speeding up the development and deployment of applications. EC2 provides the ability to launch as many virtual servers as necessary, enabling the configuration of security settings, networking, and storage management. This scalability allows you to easily increase capacity (scale up) to accommodate compute-intensive tasks, such as high-traffic periods, and reduce capacity (scale down) during periods of lower demand.

An EC2 instance represents a virtual server running in the AWS Cloud. When launching an EC2 instance, the specified instance type dictates the available hardware resources. Each instance type is designed to offer a unique balance of compute power, memory, network performance, and storage options. For more details, refer to the Amazon EC2 Instance Types Guide.

Key Features of Amazon EC2:

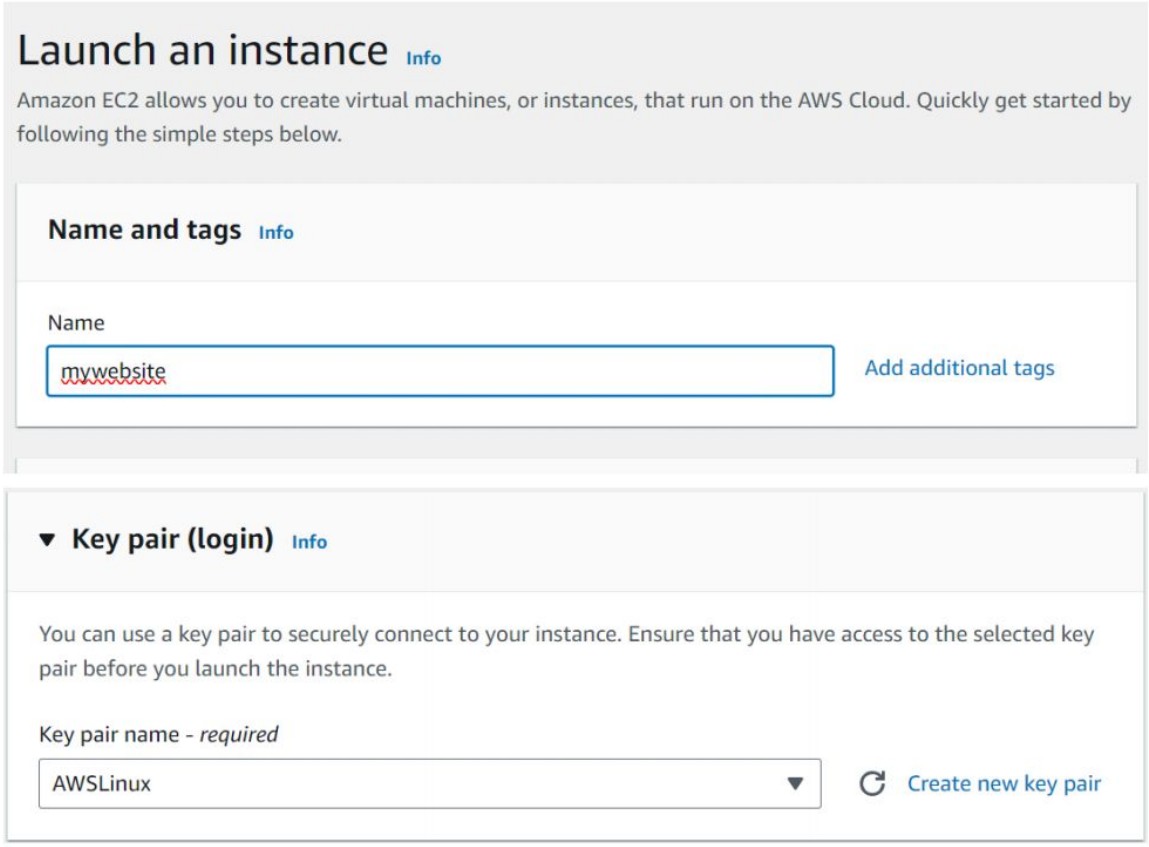
1. **Instances:**
   Virtual machines that run in the AWS Cloud.
2. **Amazon Machine Images (AMIs):**
   Prebuilt templates that define the necessary components for your instances, such as the operating system and additional software.
3. **Instance Types:**
   Different configurations of CPU, memory, storage, and networking capacity, allowing you to choose the optimal resources for your workload.
4. **Amazon EBS Volumes:**
   Persistent storage solutions using Amazon Elastic Block Store (Amazon EBS) that retain your data even when instances are stopped.
5. **Instance Store Volumes:**
   Temporary storage that is automatically deleted when an instance is stopped, hibernated, or terminated, suitable for ephemeral data.

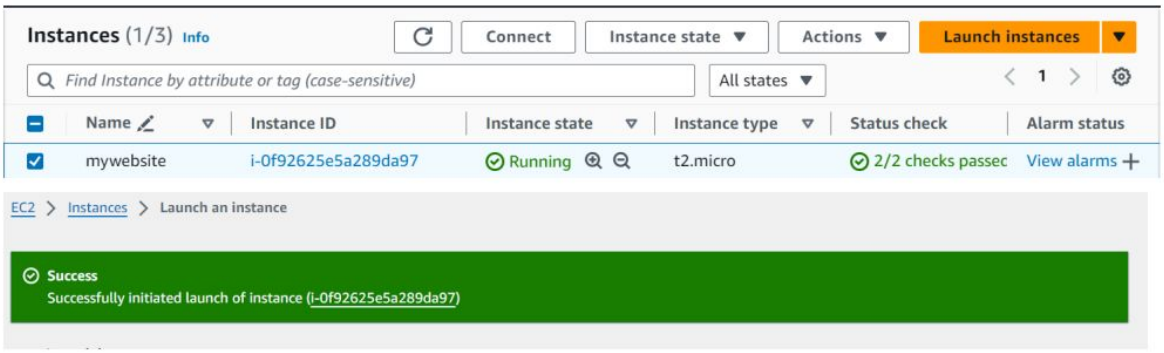
6. **Key Pairs:**
   A secure method for accessing your instances. AWS retains the public key, while you securely store the private key.
7. **Security Groups:**
   Virtual firewalls that enable you to control inbound and outbound traffic to your instances by specifying allowed protocols, ports, and IP address ranges.

Amazon EC2 also supports the secure processing, storage, and transmission of credit card information for merchants and service providers. It complies with the Payment Card Industry Data Security Standard (PCI DSS) and has been validated as PCI DSS compliant. For further information on PCI DSS and how to obtain the AWS PCI Compliance Package, refer to the PCI DSS Level 1 documentation.

Implementation:

Part A:
EC2 Instance creation:





Static website hosting using EC2:-

```
Complete!
[root@ip-172-31-62-13 ~]# systemctl status httpd
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-172-31-62-13 ~]# mkdir asw_assg3
[root@ip-172-31-62-13 ~]# cd aws_assg3
-bash: cd: aws_assg3: No such file or directory
[root@ip-172-31-62-13 ~]# cd asw_assg3
[root@ip-172-31-62-13 asw_assg3]# wget https://github.com/rujutamedhi22/infosys_html.git                https://github.com/rujutamedhi22/
infosys_html.git
--2024-08-17 15:29:04--  https://github.com/rujutamedhi22/infosys_html.git
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/rujutamedhi22/infosys_html [following]
--2024-08-17 15:29:04--  https://github.com/rujutamedhi22/infosys_html
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'infosys_html.git'

infosys_html.git                    [<=>                                                                ]       0  --.-KB/s    infosys_
html.git                      [ <=>                                                               ] 266.79K  --.-KB/s    in 0.02s
```

```
2024-08-17 15:29:04 (15.9 MB/s) - 'infosys_html.git' saved [273193]

[root@ip-172-31-62-13 asw_assg3]# ls -lrt
total 268
-rw-r--r--. 1 root root 273193 Aug 17 15:29 infosys_html.git
[root@ip-172-31-62-13 asw_assg3]# wget https://github.com/rujutamedhi22/infosys_html/archive/refs/headshttps://github.com/rujutamedhi22/
infosys_html/archive/refs/heads/master.zip
--2024-08-17 15:35:07--  https://github.com/rujutamedhi22/infosys_html/archive/refs/heads/master.zip
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/rujutamedhi22/infosys_html/zip/refs/heads/master [following]
--2024-08-17 15:35:07--  https://codeload.github.com/rujutamedhi22/infosys_html/zip/refs/heads/master
Resolving codeload.github.com (codeload.github.com)... 140.82.112.10
Connecting to codeload.github.com (codeload.github.com)|140.82.112.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'master.zip'

master.zip                          [<=>                                                                ]       0  --.-KB/s    master.z
ip                            [ <=>                                                               ] 233.35K  --.-KB/s    in 0.01s

2024-08-17 15:35:07 (18.4 MB/s) - 'master.zip' saved [238946]

[root@ip-172-31-62-13 asw_assg3]# ls -lrt
```

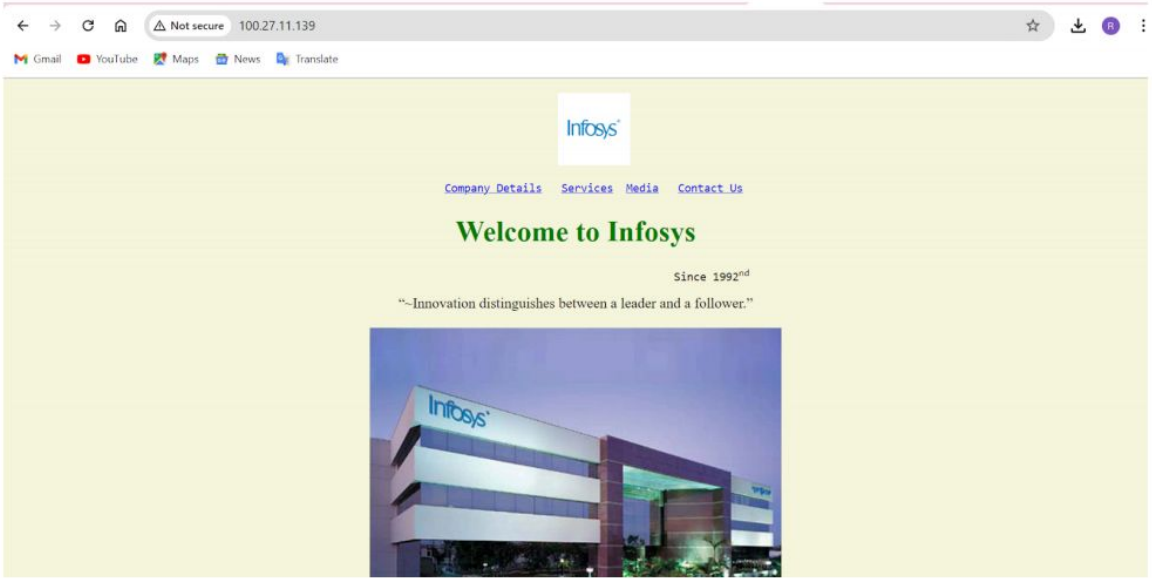
```
[root@ip-172-31-62-13 asw_assg3]# ls -lrt
total 504
-rw-r--r--. 1 root root 273193 Aug 17 15:29 infosys_html.git
-rw-r--r--. 1 root root 238946 Aug 17 15:35 master.zip
[root@ip-172-31-62-13 asw_assg3]# unzip master.zip
Archive:  master.zip
703660532cf867469a131d058181d873f4603d49
   creating: infosys_html-master/
  inflating: infosys_html-master/Infosys.jpg
  inflating: infosys_html-master/consultant.jpeg
  inflating: infosys_html-master/digital.jpeg
  inflating: infosys_html-master/facebook.png
  inflating: infosys_html-master/index.html
  inflating: infosys_html-master/infosys-logo.jpg
  inflating: infosys_html-master/twitter.png
  inflating: infosys_html-master/world-wide-signal.png
[root@ip-172-31-62-13 asw_assg3]# ls -lrt
total 504
drwxr-xr-x. 2 root root    178 Aug  4 05:08 infosys_html-master
-rw-r--r--. 1 root root 273193 Aug 17 15:29 infosys_html.git
-rw-r--r--. 1 root root 238946 Aug 17 15:35 master.zip
[root@ip-172-31-62-13 asw_assg3]# cd ^C
[root@ip-172-31-62-13 asw_assg3]# cd https://github.com/rujutamedhi22/infosys_html/archive/refs/headshttps://github.com/rujutame
wget https://github.com/rujutamedhi22/infosys_html/archive/refs/heads/master.zip                 ls -lrt^C
[root@ip-172-31-62-13 asw_assg3]# cd infosys_html-master
[root@ip-172-31-62-13 infosys_html-master]# ls -lrt
```

```
-rw-r--r--. 1 root root   4840 Aug  4 05:08 index.html
-rw-r--r--. 1 root root  15105 Aug  4 05:08 facebook.png
-rw-r--r--. 1 root root   8443 Aug  4 05:08 digital.jpeg
-rw-r--r--. 1 root root 103234 Aug  4 05:08 consultant.jpeg
-rw-r--r--. 1 root root  38350 Aug  4 05:08 Infosys.jpg
[root@ip-172-31-62-13 infosys_html-master]# mv * /var/www/html/
[root@ip-172-31-62-13 infosys_html-master]# cd /var/www/html
[root@ip-172-31-62-13 html]# ls -lrt
total 284
-rw-r--r--. 1 root root  10975 Aug  4 05:08 world-wide-signal.png
-rw-r--r--. 1 root root  23628 Aug  4 05:08 twitter.png
-rw-r--r--. 1 root root  67699 Aug  4 05:08 infosys-logo.jpg
-rw-r--r--. 1 root root   4840 Aug  4 05:08 index.html
-rw-r--r--. 1 root root  15105 Aug  4 05:08 facebook.png
-rw-r--r--. 1 root root   8443 Aug  4 05:08 digital.jpeg
-rw-r--r--. 1 root root 103234 Aug  4 05:08 consultant.jpeg
-rw-r--r--. 1 root root  38350 Aug  4 05:08 Infosys.jpg
[root@ip-172-31-62-13 html]# systemctl status httpd
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-172-31-62-13 html]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-62-13 html]# systemctl start httpd
[root@ip-172-31-62-13 html]#
```

i-0f92625e5a289da97 (mywebsite)

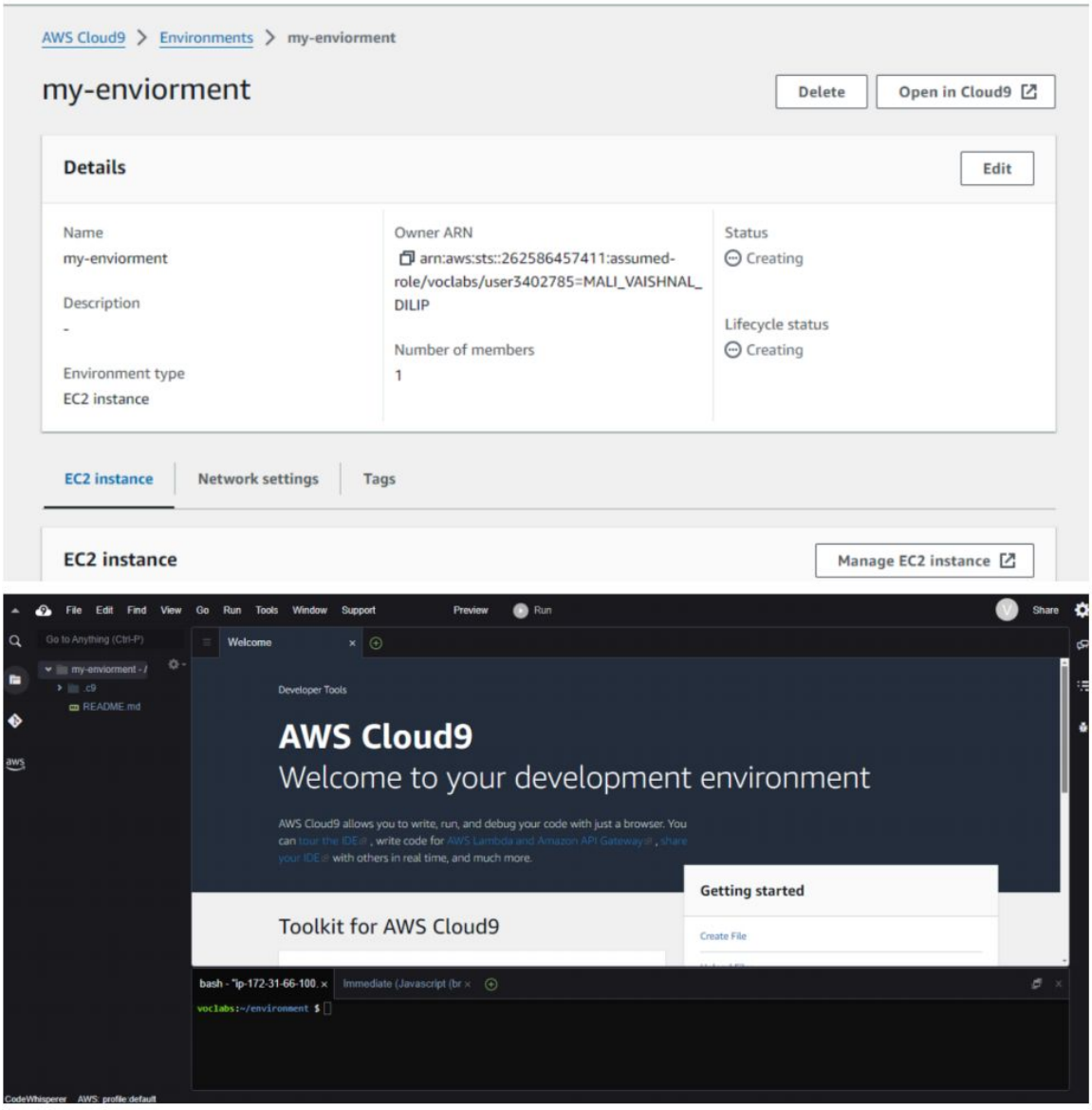
Auto-assigned IP address

100.27.11.139 [Public IP]



Part-B:

Create Environment on cloud 9



RUJUTA MEDHI **D15A-28**

EXPERIMENT:02

Aim: Deploy a Sample Application on Elastic Beanstalk using AWS CodePipeline and AWS CodeDeploy.

Theory:

AWS Elastic Beanstalk allows for the rapid deployment and management of applications in the AWS Cloud, simplifying the process by eliminating the need to understand the underlying infrastructure. AWS consists of over a hundred services, each offering specific functionalities. While the range of services provides extensive flexibility in managing your AWS infrastructure, determining which services to use and how to provision them can be complex. Elastic Beanstalk alleviates this complexity without sacrificing control or customization. By simply uploading your application, Elastic Beanstalk takes care of the details related to capacity provisioning, load balancing, scaling, and application health monitoring.

Elastic Beanstalk supports a variety of programming languages, including Go, Java, .NET, Node.js, PHP, Python, and Ruby. It also accommodates Docker containers, allowing you to define your programming language and application dependencies that might not be supported by other Elastic Beanstalk platforms. Upon deploying your application, Elastic Beanstalk automatically provisions the necessary AWS resources, such as Amazon EC2 instances, and configures the selected platform to run your application.

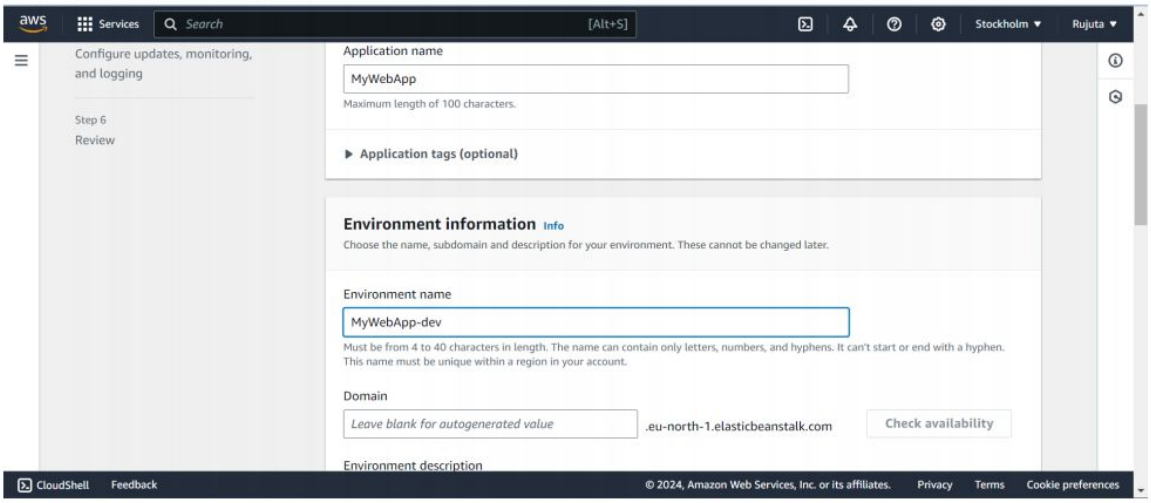
Interaction with Elastic Beanstalk can be done through multiple interfaces: the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or eb, a specialized high-level CLI tool designed specifically for Elastic Beanstalk.

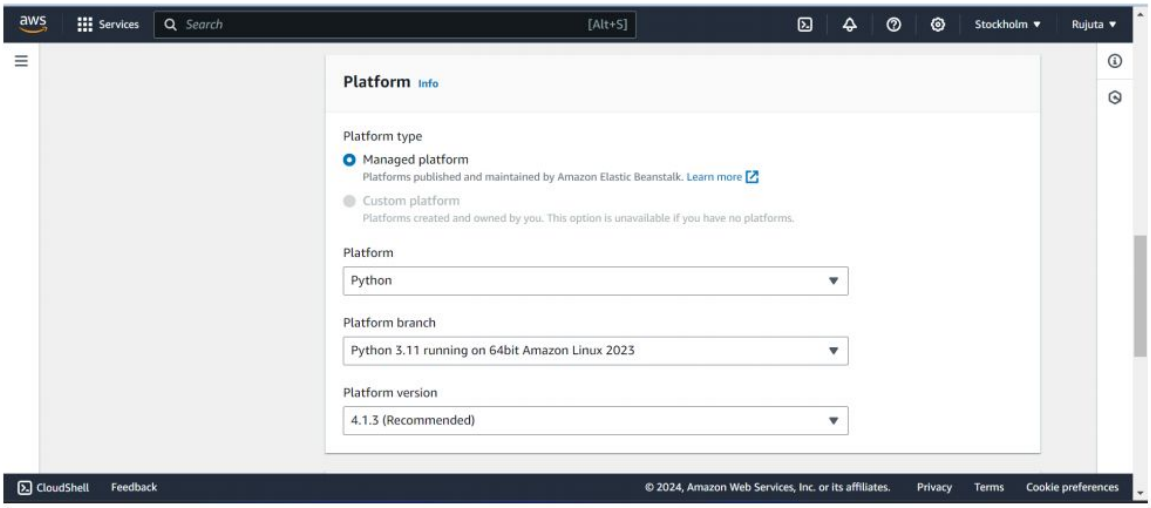
Most deployment tasks, like scaling your fleet of Amazon EC2 instances or monitoring application performance, can be easily managed via the Elastic Beanstalk web interface (console). To get started with Elastic Beanstalk, you create an application, upload an application version (e.g., a Java .war file) as a source bundle, and provide necessary configuration details. Elastic Beanstalk will then automatically set up the environment and configure all required AWS resources to run your application. Once the environment is active, you can manage it and deploy new application versions as needed. The following workflow diagram outlines the process of using Elastic Beanstalk.

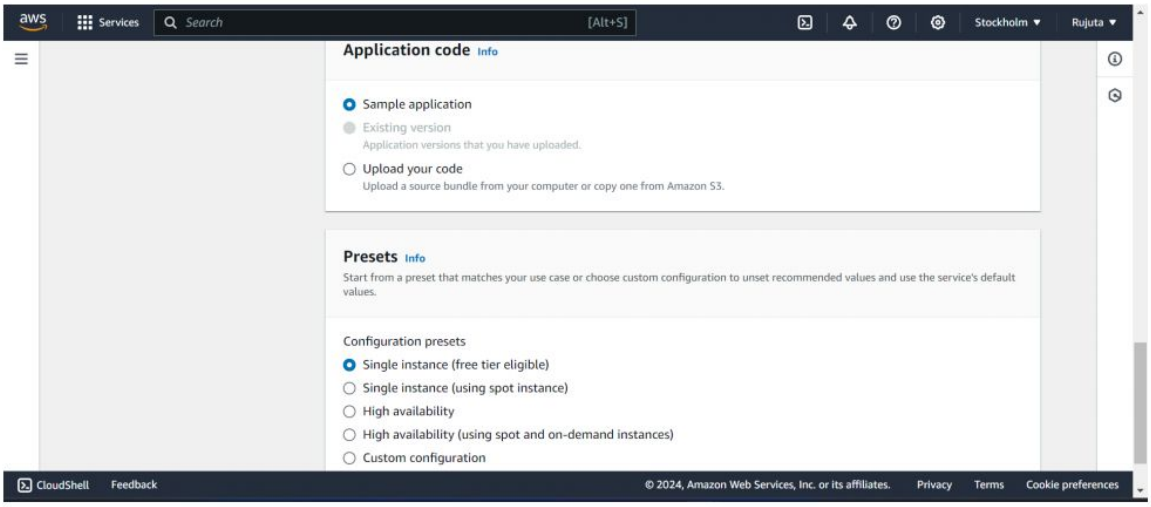
After your application is up and running, you can access information related to the application, such as metrics, events, and environment status, through the Elastic Beanstalk console, APIs, or the Command Line Interfaces, including the unified AWS CLI.

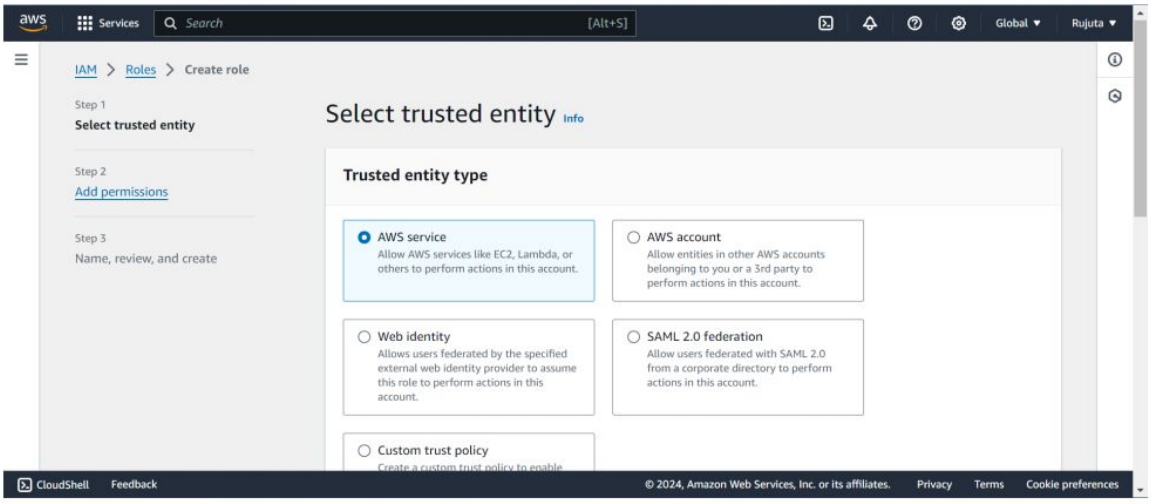
Implementation:

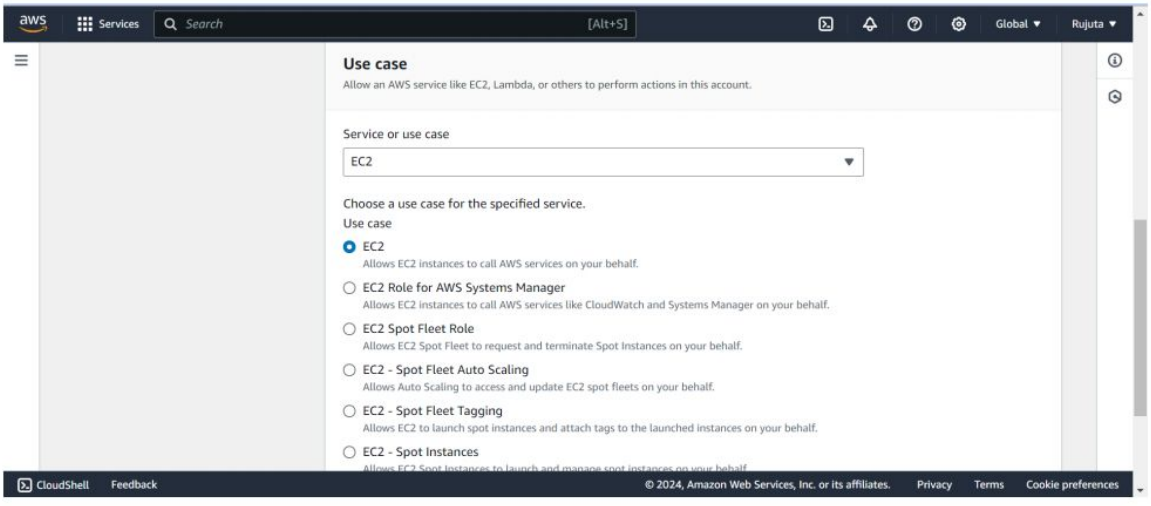
Deploying basic web page on Elastic Beanstalk

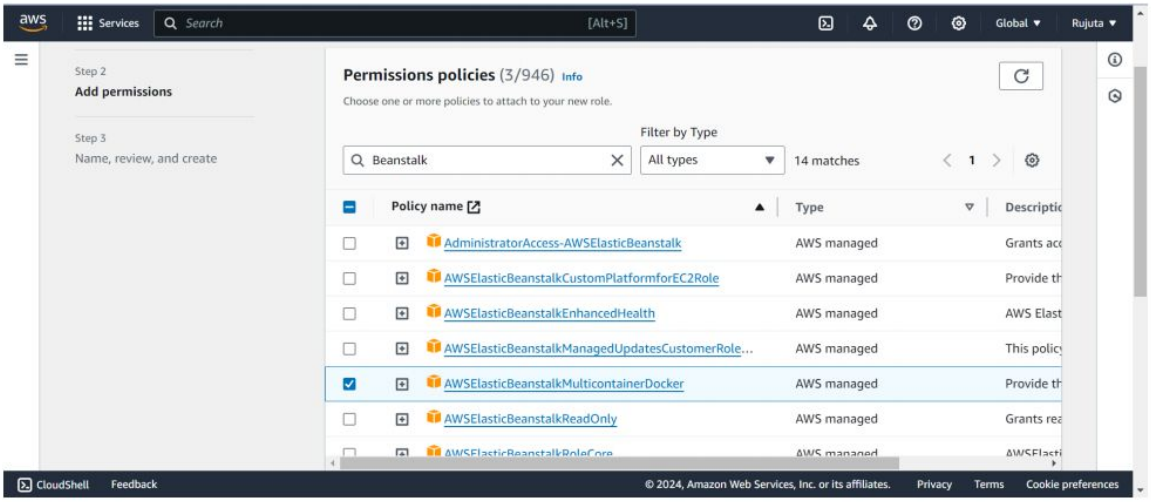


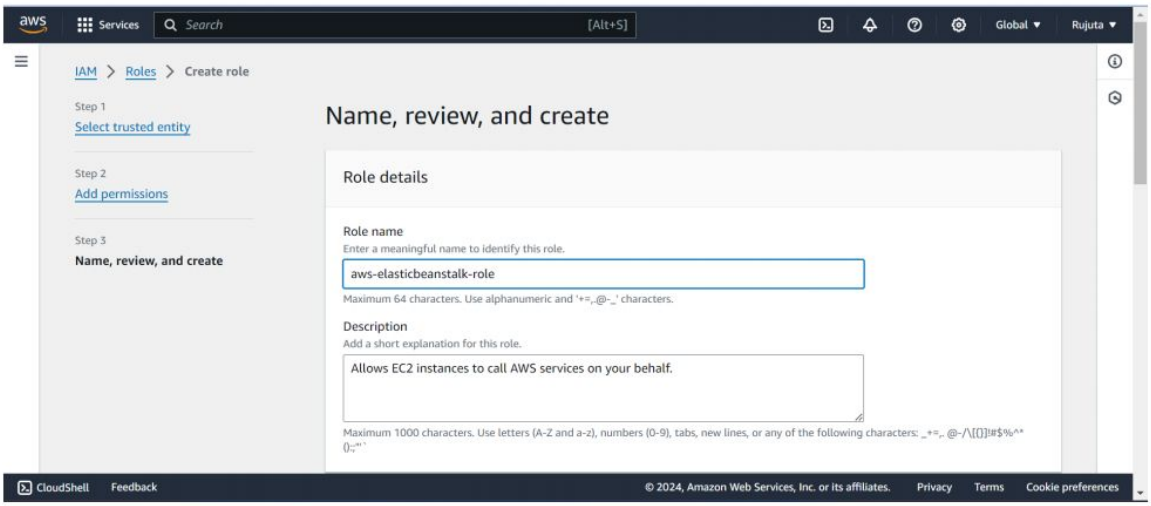



aws
Services
Search
[Alt+S]
Stockholm
Rujuta
Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.
Presets Info
Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.
Configuration presets
Single instance (free tier eligible)
Single instance (using spot instance)
High availability
High availability (using spot and on-demand instances)
Custom configuration
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

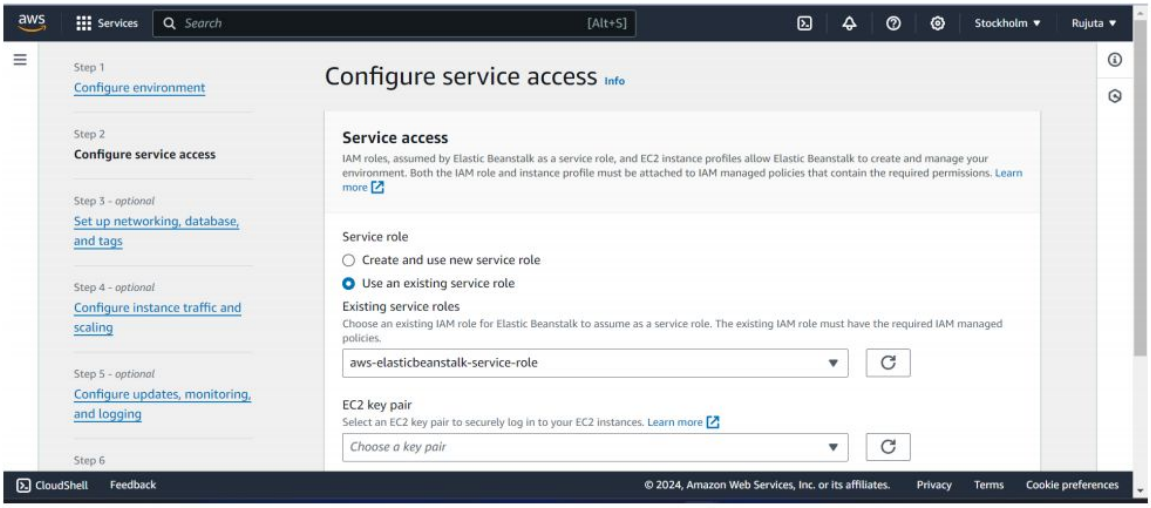

aws
Services
Search
[Alt+S]
Global
Rujuta
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Select trusted entity Info
Trusted entity type
AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
Custom trust policy
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

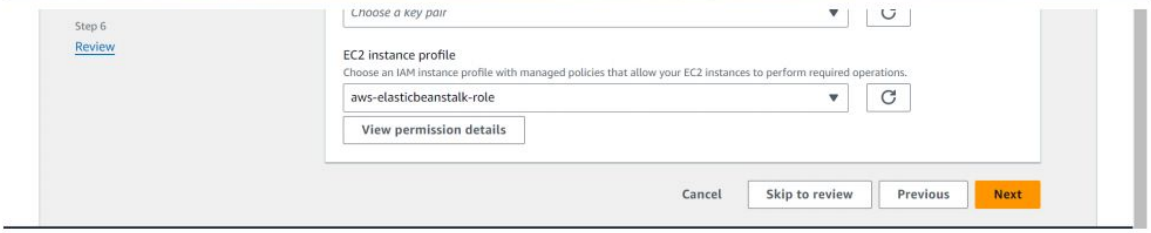

aws
Services
Search
[Alt+S]
Global
Rujuta
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.
Service or use case
EC2
Choose a use case for the specified service.
Use case
EC2
Allows EC2 instances to call AWS services on your behalf.
EC2 Role for AWS Systems Manager
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
EC2 Spot Fleet Role
Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
EC2 - Spot Fleet Auto Scaling
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
EC2 - Spot Fleet Tagging
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
EC2 - Spot Instances
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
Global
Rujuta
Step 2
Add permissions
Step 3
Name, review, and create
Permissions policies (3/946) Info
Choose one or more policies to attach to your new role.
Filter by Type
Beanstalk
All types
14 matches
1
Policy name
Type
AdministratorAccess-AWSElasticBeanstalk
AWS managed
AWSElasticBeanstalkCustomPlatformforEC2Role
AWS managed
AWSElasticBeanstalkEnhancedHealth
AWS managed
AWSElasticBeanstalkManagedUpdatesCustomerRole...
AWS managed
AWSElasticBeanstalkMulticontainerDocker
AWS managed
AWSElasticBeanstalkReadOnly
AWS managed
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

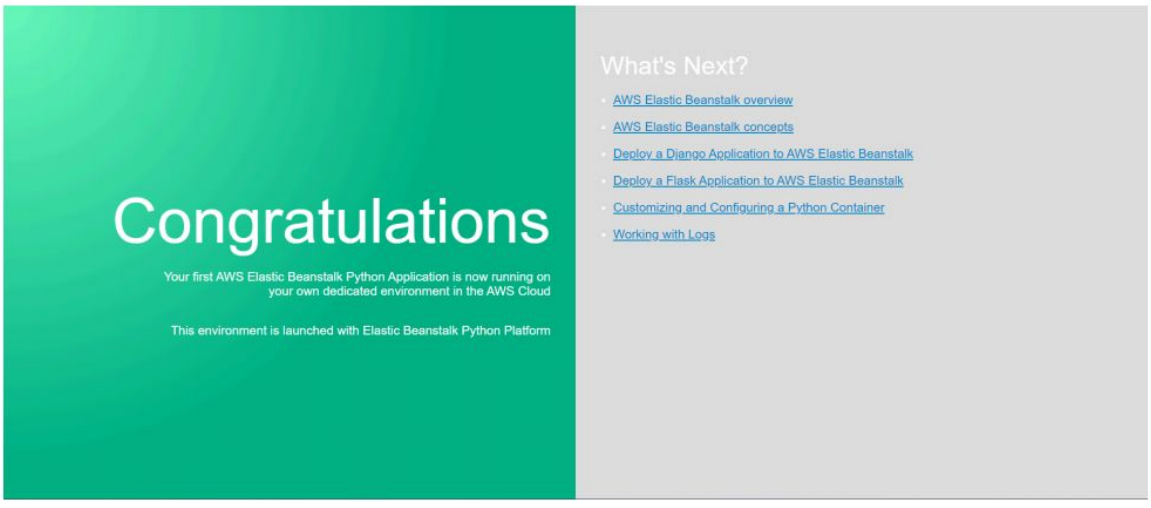

Services
Search
[Alt+S]
Global
Rujuta
IAM
Roles
Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
aws-elasticbeanstalk-role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^*
();"'
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
Global
Rujuta
Role aws-elasticbeanstalk-role created.
View role
Roles (4) Info
Delete
Create role
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

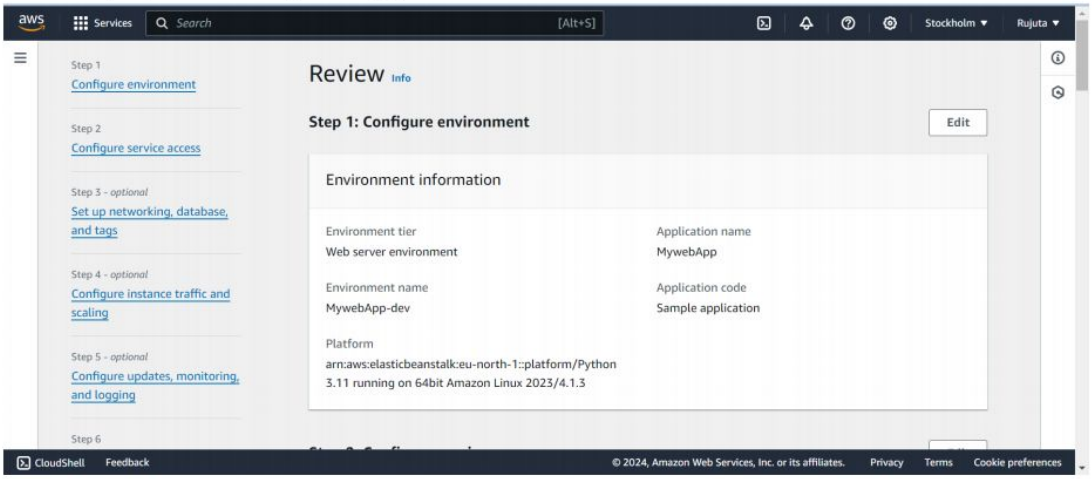

aws
Services
Search
[Alt+S]
Stockholm
Rujuta
Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Configure service access Info
Service access
IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. Learn more
Service role
Create and use new service role
Use an existing service role
Existing service roles
Choose an existing IAM role for Elastic Beanstalk to assume as a service role. The existing IAM role must have the required IAM managed policies.
aws-elasticbeanstalk-service-role
EC2 key pair
Select an EC2 key pair to securely log in to your EC2 instances. Learn more
Choose a key pair
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

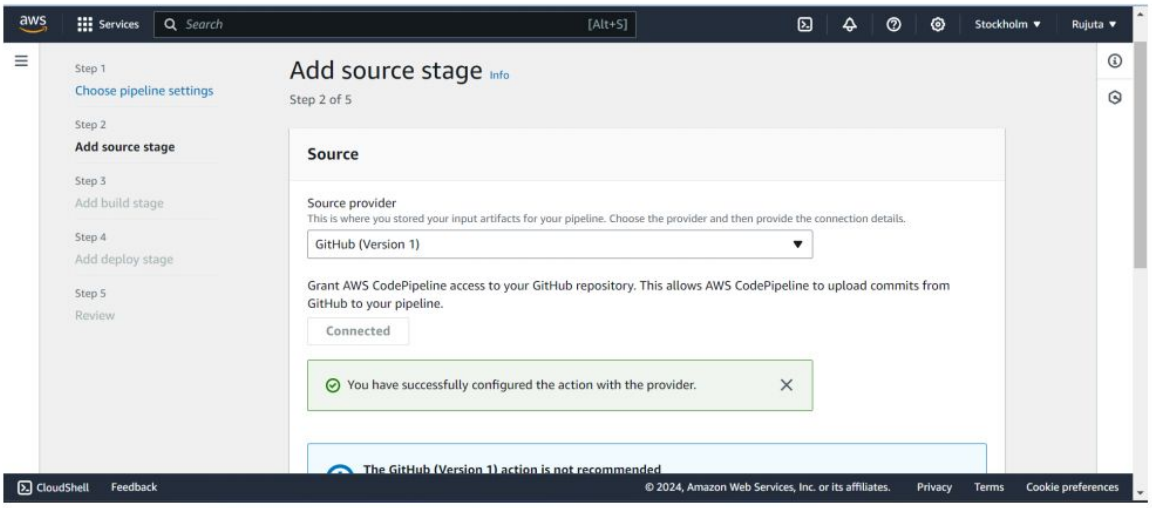

Step 6
Review
Choose a key pair
EC2 instance profile
Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.
aws-elasticbeanstalk-role
View permission details
Cancel
Skip to review
Previous
Next

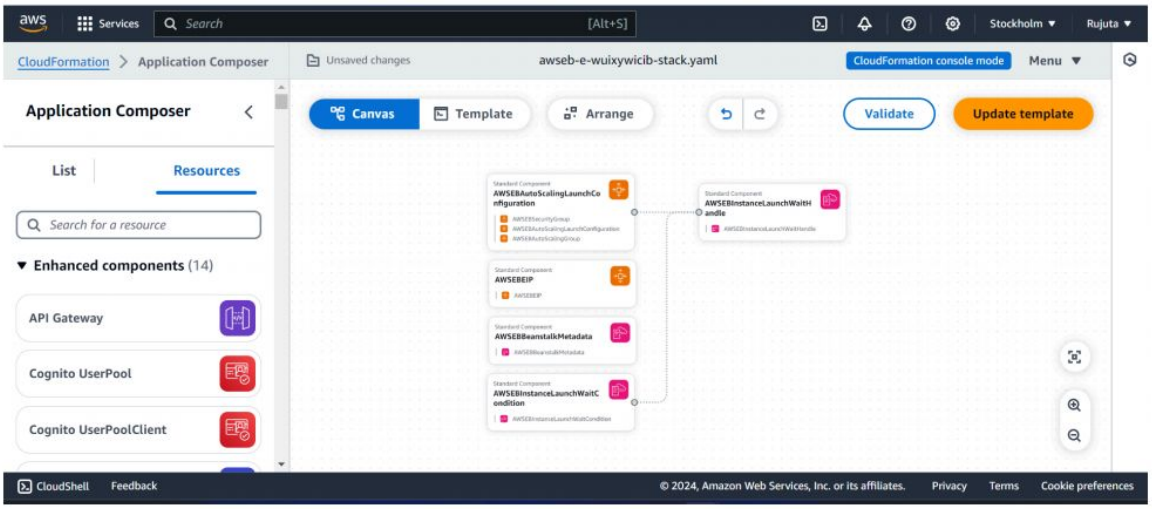

aws
Services
Search
[Alt+S]
Stockholm
Rujuta
Environment successfully launched.
Elastic Beanstalk
Environments
MywebApp-dev
Actions
Upload and deploy

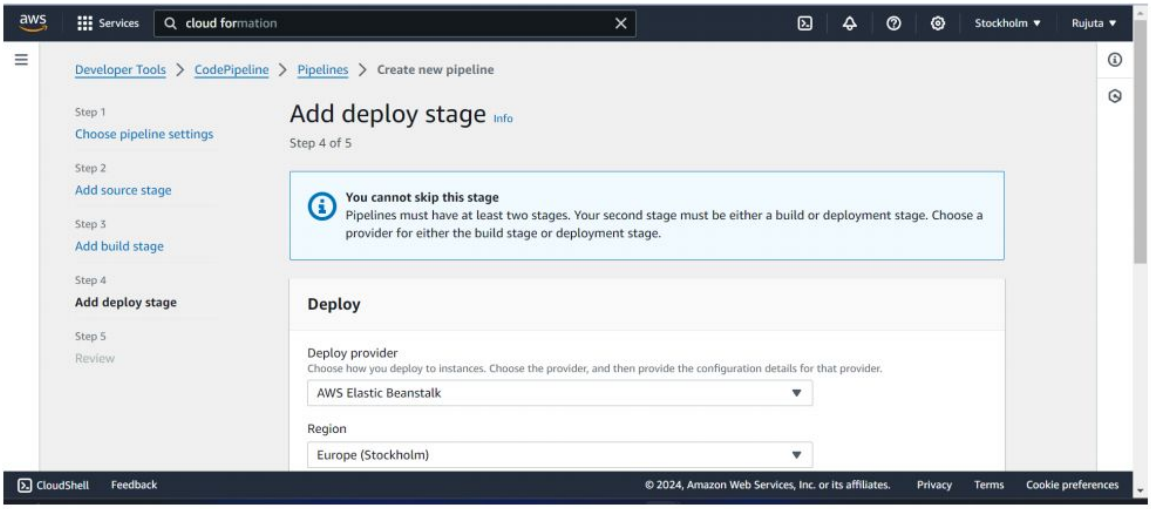

Congratulations
Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud
This environment is launched with Elastic Beanstalk Python Platform
What's Next?
AWS Elastic Beanstalk overview
AWS Elastic Beanstalk concepts
Deploy a Django Application to AWS Elastic Beanstalk
Deploy a Flask Application to AWS Elastic Beanstalk
Customizing and Configuring a Python Container
Working with Logs

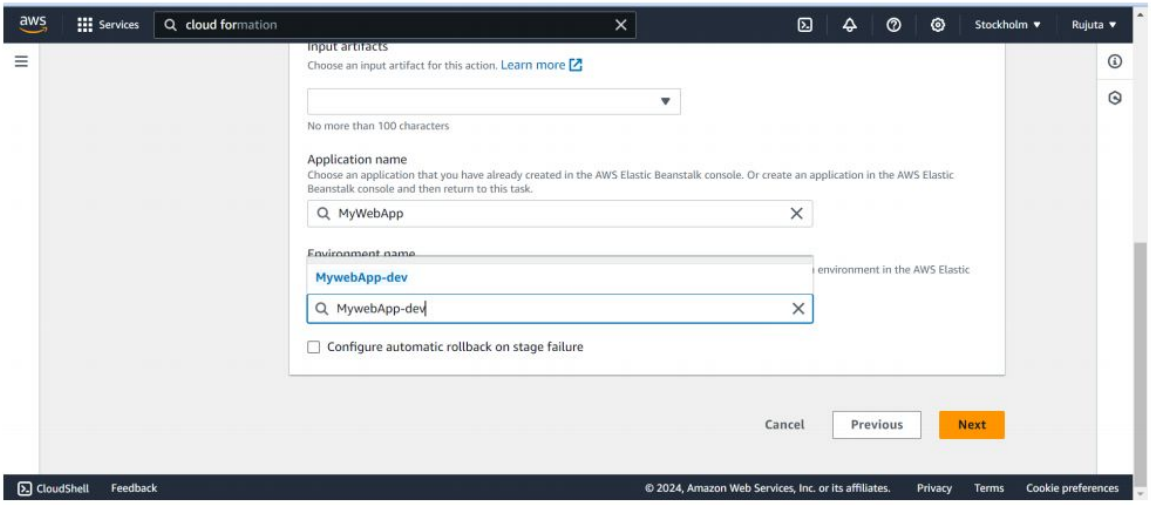
Code Deployment using Codepipeline:

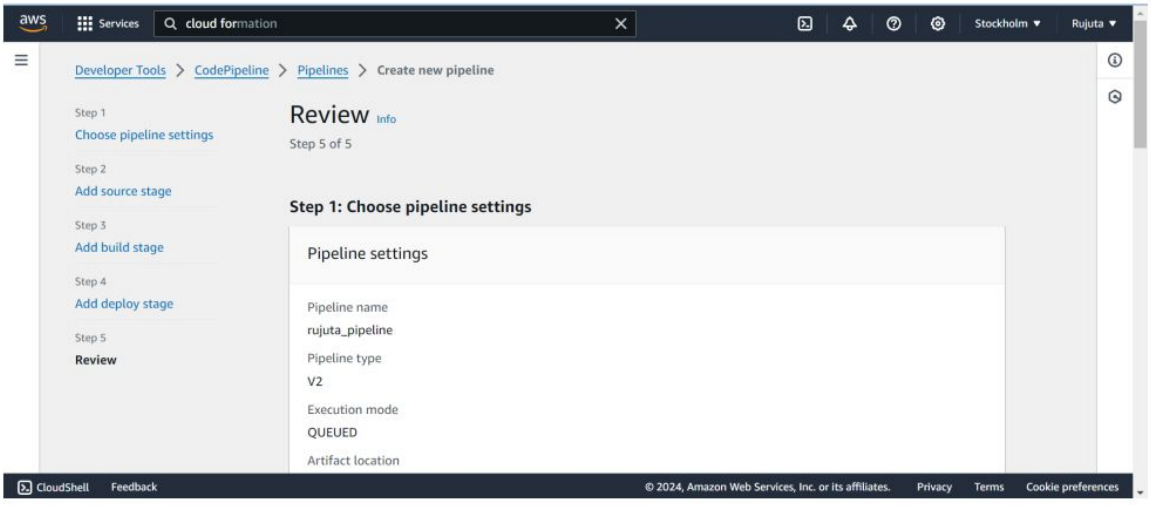


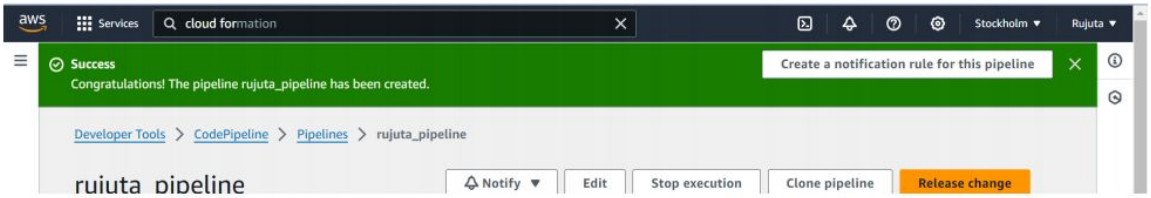


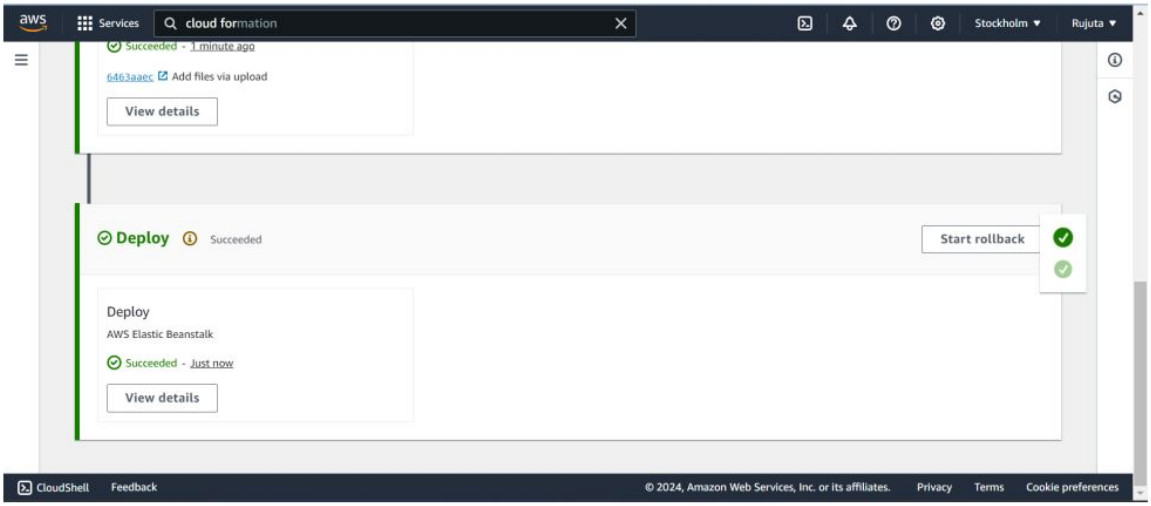


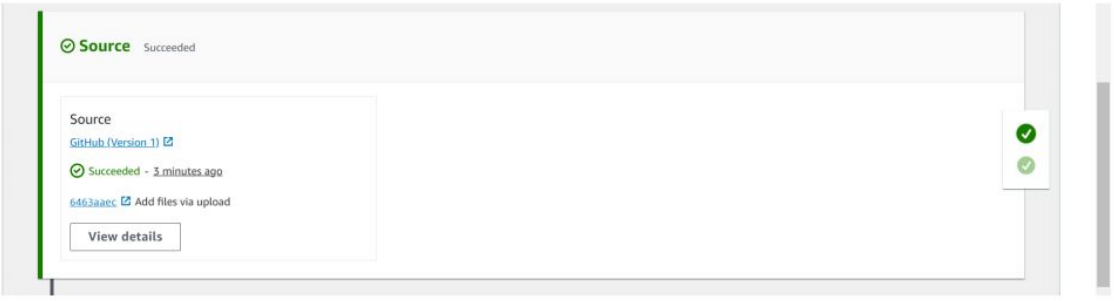

aws
Services
cloud formation
Stockholm
Rujuta
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Add deploy stage Info
Step 4 of 5
You cannot skip this stage
Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.
Deploy
Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.
AWS Elastic Beanstalk
Region
Europe (Stockholm)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
cloud formation
Stockholm
Rujuta
Input artifacts
Choose an input artifact for this action. Learn more
No more than 100 characters
Application name
Choose an application that you have already created in the AWS Elastic Beanstalk console. Or create an application in the AWS Elastic Beanstalk console and then return to this task.
MyWebApp
Environment name
MywebApp-dev
MywebApp-dev
environment in the AWS Elastic
Configure automatic rollback on stage failure
Cancel
Previous
Next
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
cloud formation
Stockholm
Rujuta
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Review Info
Step 5 of 5
Step 1: Choose pipeline settings
Pipeline settings
Pipeline name
rujuta_pipeline
Pipeline type
V2
Execution mode
QUEUED
Artifact location
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
cloud formation
Stockholm
Rujuta
Success
Congratulations! The pipeline rujuta_pipeline has been created.
Create a notification rule for this pipeline
Developer Tools > CodePipeline > Pipelines > rujuta_pipeline
rujuta_pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change


aws
Services
cloud formation
Stockholm
Rujuta
Succeeded - 1 minute ago
6463aaec Add files via upload
View details
Deploy
Succeeded
Start rollback
Deploy
AWS Elastic Beanstalk
Succeeded - Just now
View details
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Source
Succeeded
Source
GitHub (Version 1)
Succeeded - 3 minutes ago
6463aaec Add files via upload
View details


Infosys
Company Details
Services
Media
Contact Us
Welcome to Infosys
Since 1992nd
"~Innovation distinguishes between a leader and a follower."
Infosys

Rujuta Medhi **D15A-28**

EXPERIMENT:03

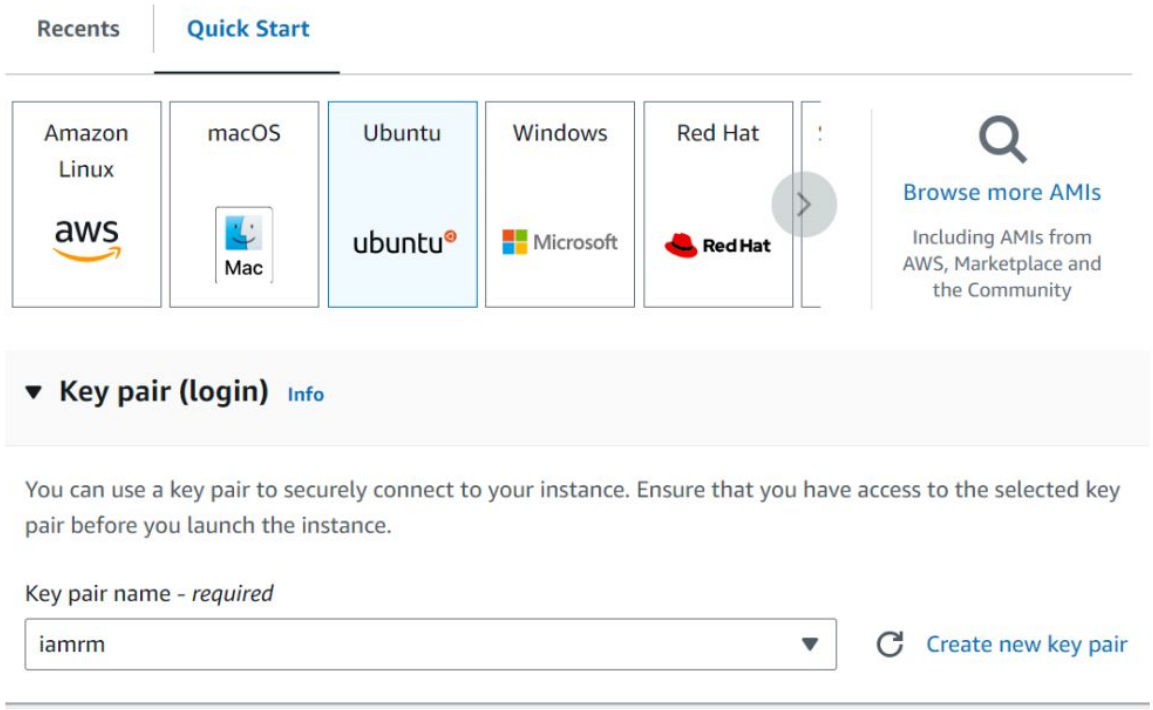
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

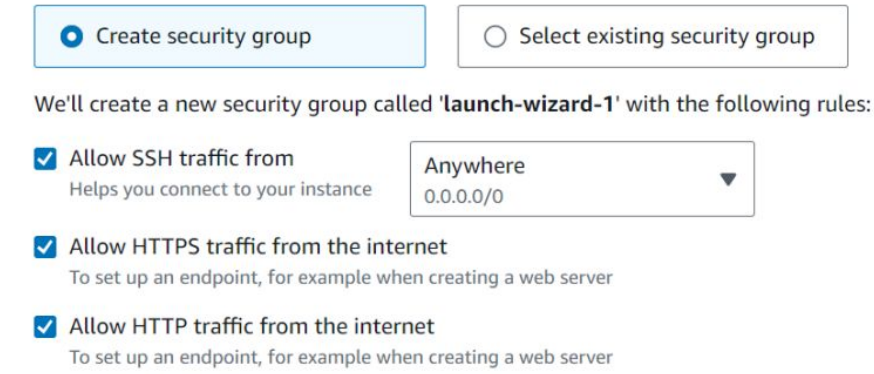
Theory: To understand Kubernetes Cluster Architecture and how to install and spin up a Kubernetes cluster on Linux machines or cloud platforms, it's essential to grasp the fundamental components and design principles of Kubernetes.

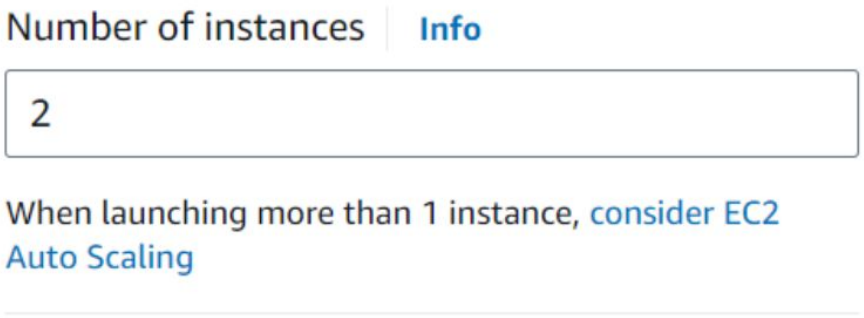
Overview of Kubernetes: Kubernetes is an open-source container orchestration platform developed by Google, designed to automate the deployment, scaling, and management of containerized applications. It provides a robust infrastructure that supports microservices architecture, offering features such as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

Implementation:

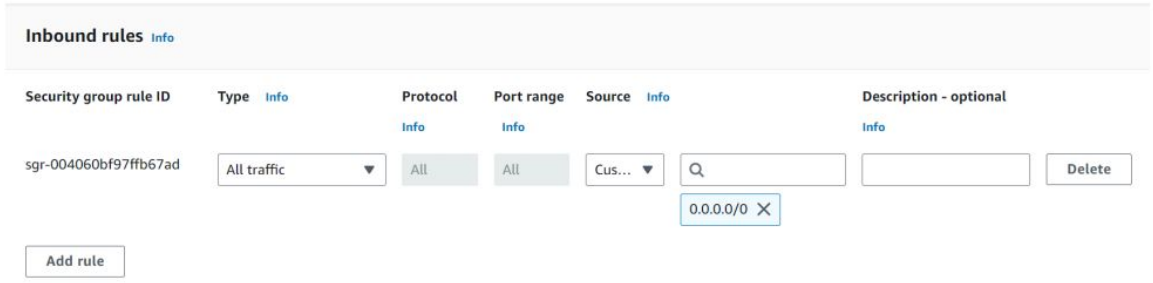
Creation of 2 EC2 instances



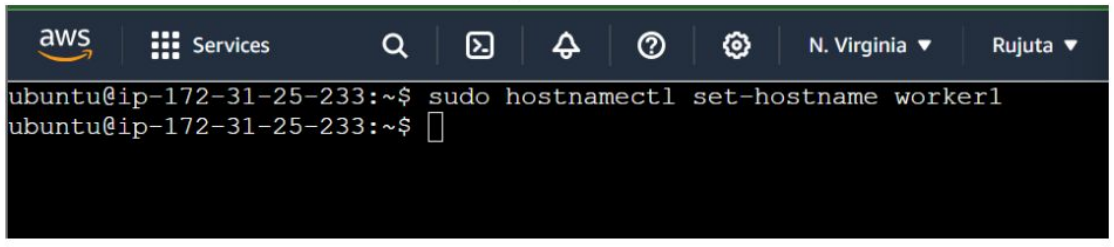




Edit its Inbound Rules set ‘All traffic’



Set master and worker as hostname on respective servers



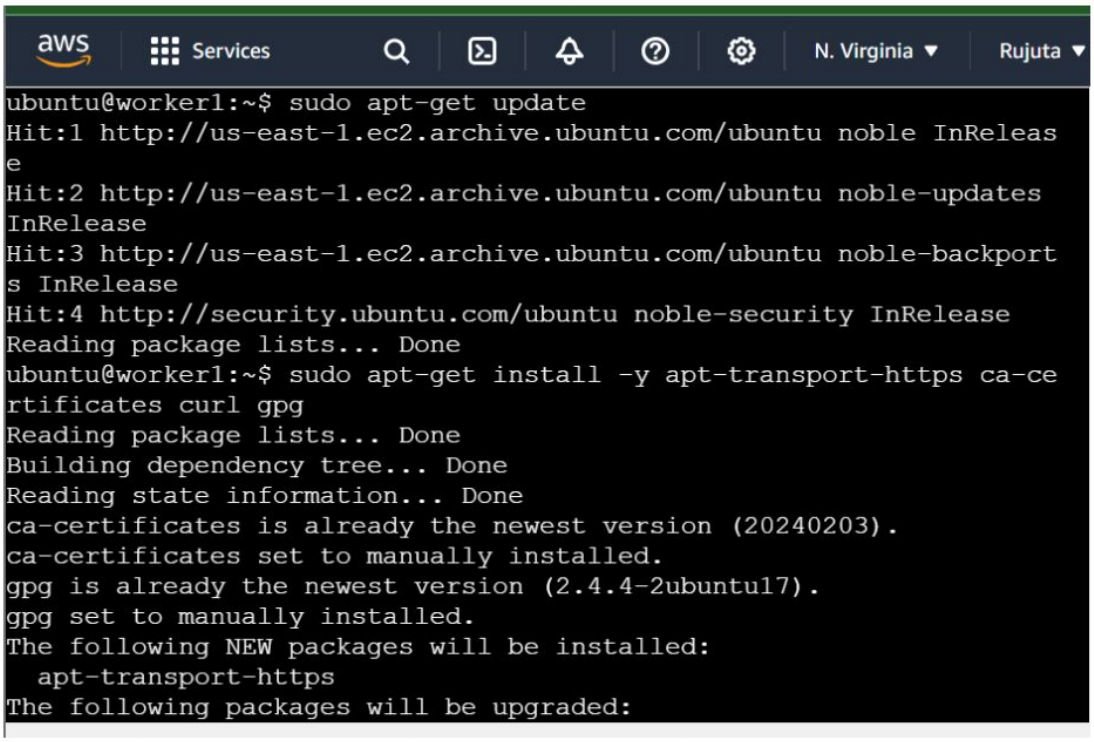
```
ubuntu@master:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InReleas
e
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates
InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backport
s InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [1
26 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe
 amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe
 Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 P
ackages [354 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Transla
```

Installation of Docker-

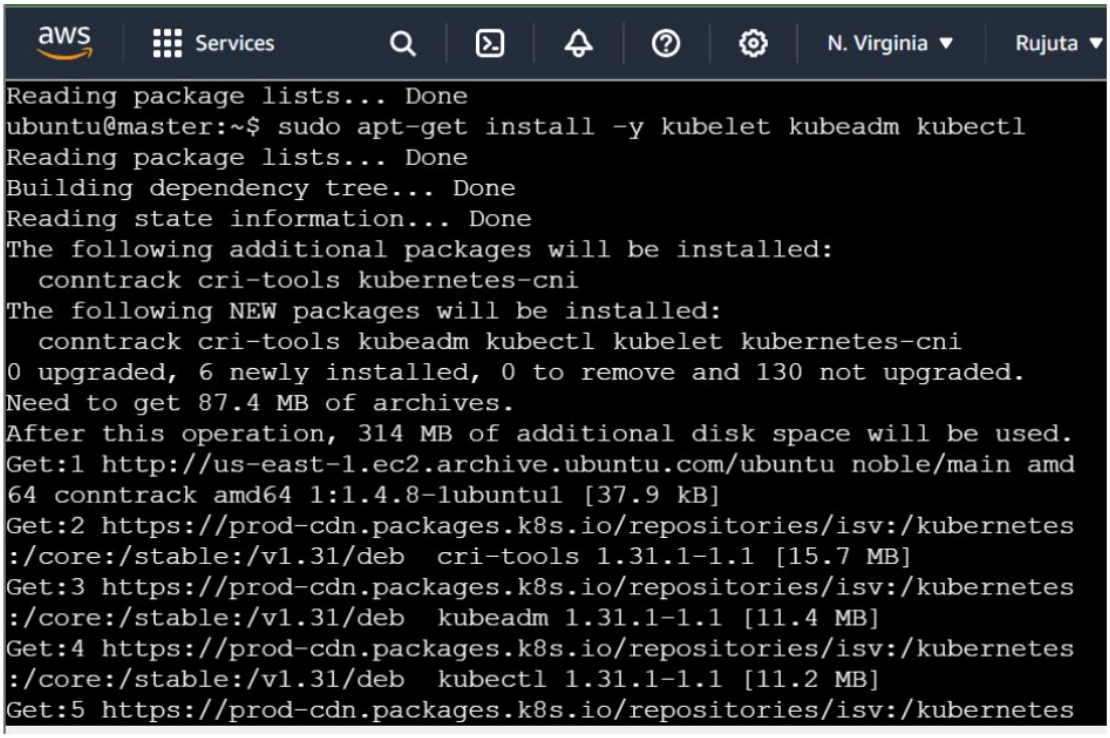
```
ubuntu@master:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc
  ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap
  docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse
  | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io
  pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 133 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Enabling the Docker

```
ubuntu@master:~$ sudo systemctl enable docker
ubuntu@master:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabl
     Active: active (running) since Mon 2024-09-16 16:28:44 UTC; 1
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 3117 (dockerd)
      Tasks: 8
     Memory: 32.6M (peak: 32.7M)
        CPU: 277ms
     CGroup: /system.slice/docker.service
             └─3117 /usr/bin/dockerd -H fd:// --containerd=/run/co

Sep 16 16:28:43 master systemd[1]: Starting docker.service - Docke
Sep 16 16:28:43 master dockerd[3117]: time="2024-09-16T16:28:43.62
Sep 16 16:28:43 master dockerd[3117]: time="2024-09-16T16:28:43.62
Sep 16 16:28:43 master dockerd[3117]: time="2024-09-16T16:28:43.72
Sep 16 16:28:43 master dockerd[3117]: time="2024-09-16T16:28:43.99
Sep 16 16:28:44 master dockerd[3117]: time="2024-09-16T16:28:44.09
Sep 16 16:28:44 master dockerd[3117]: time="2024-09-16T16:28:44.09
```



Installation of Kubernetes-



```
No VM guests are running outdated hypervisor (qemu) binaries on
 this host.
ubuntu@master:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@master:~$
```

```
ubuntu@master:~$ sudo swapoff -a
ubuntu@master:~$ ^C
```

Initialize Kubernetes on master-

```
ubuntu@master:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
 --ignore-preflight-errors=all
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less th
an the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the min
imum 1700 MB
        [WARNING FileExisting-socat]: socat not found in system pat
h
[preflight] Pulling images required for setting up a Kubernetes clu
ster
[preflight] This might take a minute or two, depending on the speed
 of your internet connection
[preflight] You can also perform this action beforehand using 'kube
adm config images pull'
W0916 16:51:27.872611    4323 checks.go:846] detected that the sand
box image "registry.k8s.io/pause:3.8" of the container runtime is i
nconsistent with that used by kubeadm.It is recommended to use "reg
istry.k8s.io/pause:3.10" as the CRI sandbox image.
```

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

```
ubuntu@master:~$ kubectl apply -f https://github.com/flannel-io/fla
nnel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@master:~$
```

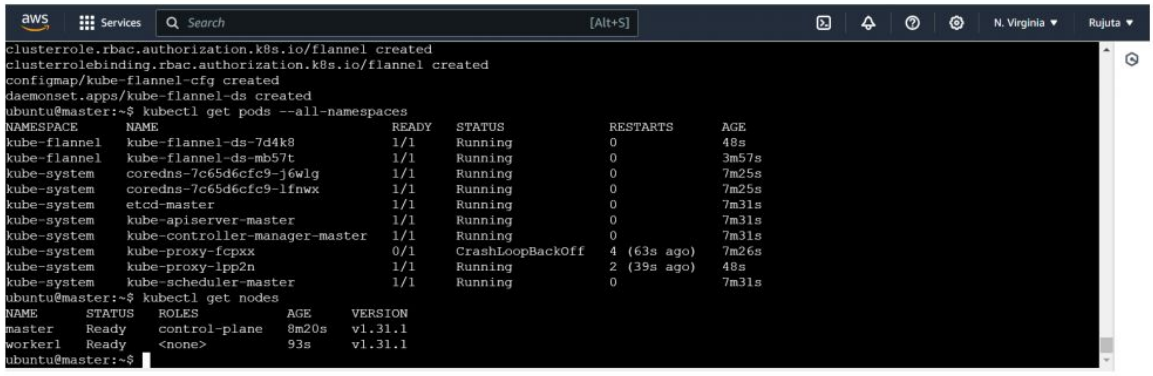
```
ubuntu@worker1:~$ sudo kubeadm join 172.31.29.228:6443 --token jvkm
es.9ghaoh4a0fr69k3c --discovery-token-ca-cert-hash sha256:3aeb516d8
5dd52182fa637385d5597989629856fa827dfd6688535d64b68a6b5 --ignore-pr
eflight-errors=all
[preflight] Running pre-flight checks
        [WARNING FileAvailable--etc-kubernetes-kubelet.conf]: /etc/
kubernetes/kubelet.conf already exists
        [WARNING FileExisting-socat]: socat not found in system pat
h
        [WARNING Port-10250]: Port 10250 is in use
        [WARNING FileAvailable--etc-kubernetes-pki-ca.crt]: /etc/ku
bernetes/pki/ca.crt already exists
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n
kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kub
elet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
 "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:1
```

```
ubuntu@master:~$ kubectl get pods --all-namespaces
NAMESPACE      NAME                              READY   STATUS
        RESTARTS       AGE
kube-flannel   kube-flannel-ds-f9cjg             1/1     Running
        0              11m
kube-system    coredns-7c65d6cfc9-hmrlq          1/1     Running
        0              21m
kube-system    coredns-7c65d6cfc9-r2c4m          1/1     Running
        0              21m
kube-system    etcd-master                       1/1     Running
        0              21m
kube-system    kube-apiserver-master             1/1     Running
        0              21m
kube-system    kube-controller-manager-master    1/1     Running
        0              21m
kube-system    kube-proxy-4csc6                  0/1     CrashLoopBa
ckOff   7 (118s ago)   21m
kube-system    kube-scheduler-master             1/1     Running
        0              21m
```

```
        0               21m
ubuntu@master:~$ kubectl get pods --all-namespaces
NAMESPACE      NAME                              READY   STATUS    R
ESTARTS        AGE
kube-flannel   kube-flannel-ds-f9cjg             1/1     Running   0
               14m
kube-system    coredns-7c65d6cfc9-hmrlq          1/1     Running   0
               24m
kube-system    coredns-7c65d6cfc9-r2c4m          1/1     Running   0
               24m
kube-system    etcd-master                       1/1     Running   0
               25m
kube-system    kube-apiserver-master             1/1     Running   0
               25m
kube-system    kube-controller-manager-master    1/1     Running   0
               25m
kube-system    kube-proxy-4csc6                  1/1     Running   8
 (5m27s ago)   24m
kube-system    kube-scheduler-master             1/1     Running   0
               25m
ubuntu@master:~$
```

```
ubuntu@worker1:~$ sudo kubeadm join 172.31.29.228:6443 --token jvkm
es.9ghaoh4a0fr69k3c --discovery-token-ca-cert-hash sha256:3aeb516d8
5dd52182fa637385d5597989629856fa827dfd6688535d64b68a6b5 --ignore-pr
eflight-errors=all
[preflight] Running pre-flight checks
        [WARNING FileAvailable--etc-kubernetes-kubelet.conf]: /etc/
kubernetes/kubelet.conf already exists
        [WARNING FileExisting-socat]: socat not found in system pat
h
        [WARNING Port-10250]: Port 10250 is in use
        [WARNING FileAvailable--etc-kubernetes-pki-ca.crt]: /etc/ku
bernetes/pki/ca.crt already exists
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n
kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kub
elet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
 "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:1
```

```
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:1
0248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 502.370836ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstra
p

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

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

ubuntu@worker1:~$
```



Rujuta Medhi **D15A-28**

EXPERIMENT NO:04

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

Theory:

What is Kubernetes?

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

Core Concepts of Kubernetes

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

Implementation:

Commands:-

Create yaml file

```
ubuntu@master:~$ sudo nano deploy.yaml
ubuntu@master:~$ kubectl create -f deploy.yaml
deployment.apps/nginx-deployment created
ubuntu@master:~$ kubectl get deploy
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           29s
```

Add below lines in the file

```
ubuntu@master:~$ cat deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
ubuntu@master:~$
```

```
ubuntu@master:~$ kubectl expose deployment.apps/nginx-deployment \
> --type="LoadBalancer"
service/nginx-deployment exposed
ubuntu@master:~$ kubectl get svc
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes         ClusterIP      10.96.0.1       <none>        443/TCP          18m
nginx-deployment   LoadBalancer   10.111.16.154   <pending>     80:30171/TCP     24s
ubuntu@master:~$
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

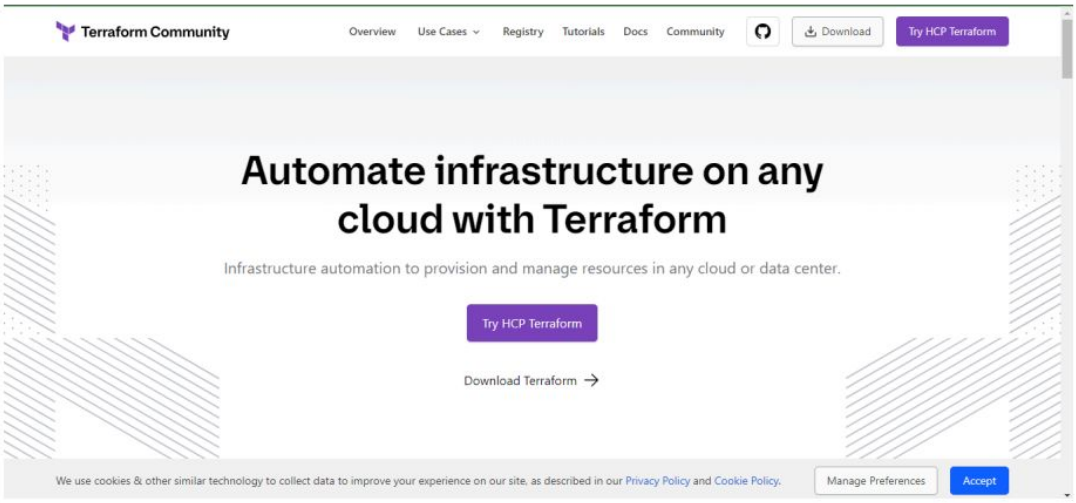
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

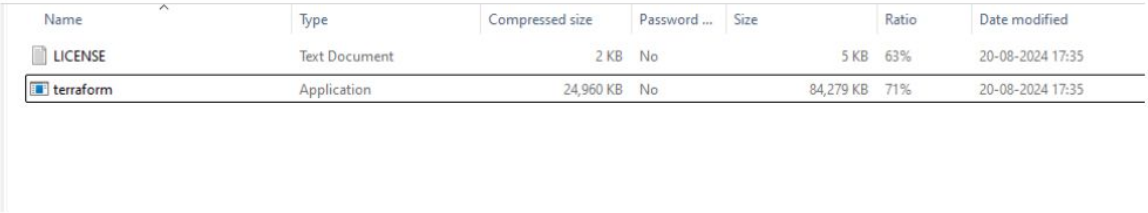
Thank you for using nginx.

Rujuta Medhi **D15A-28**

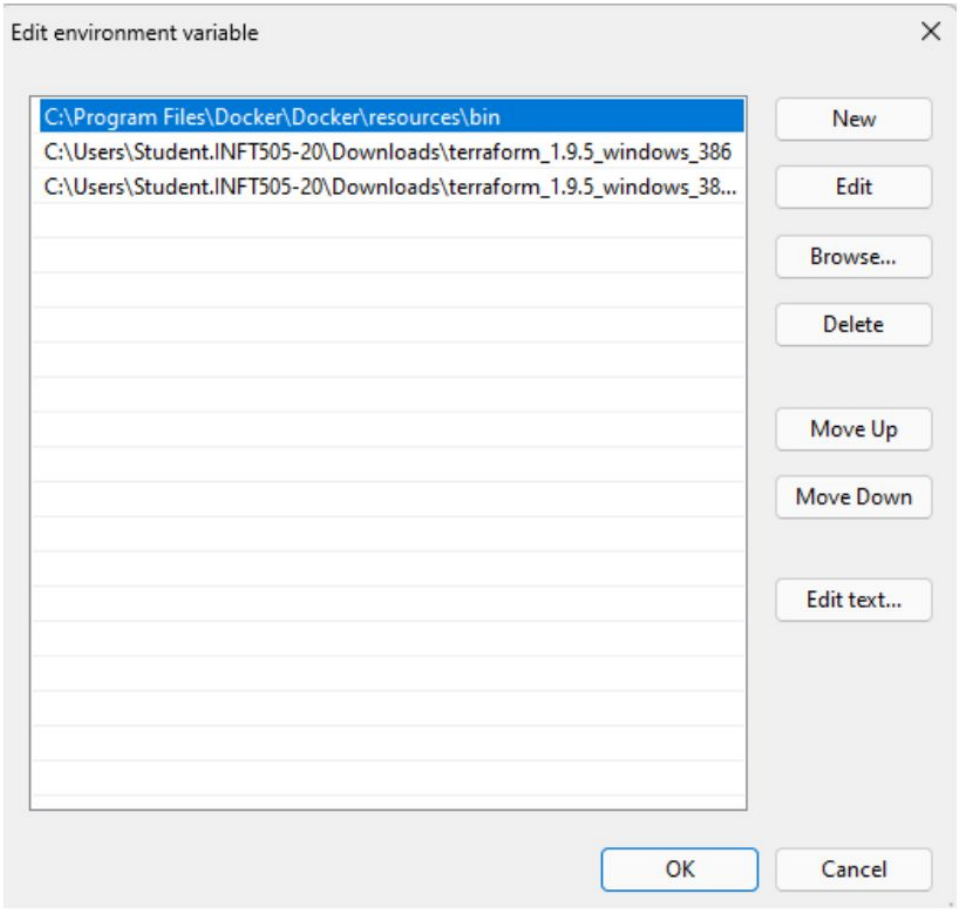
EXPERIMENT: 05

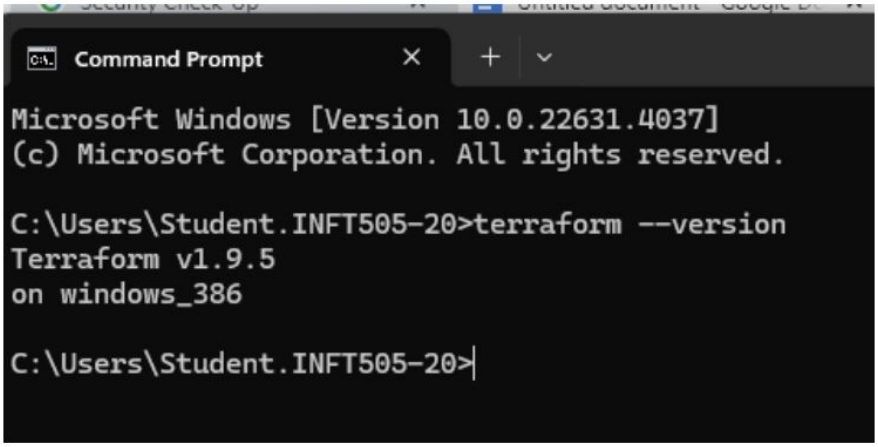
Aim: Installation and Configuration of Terraform in Windows.

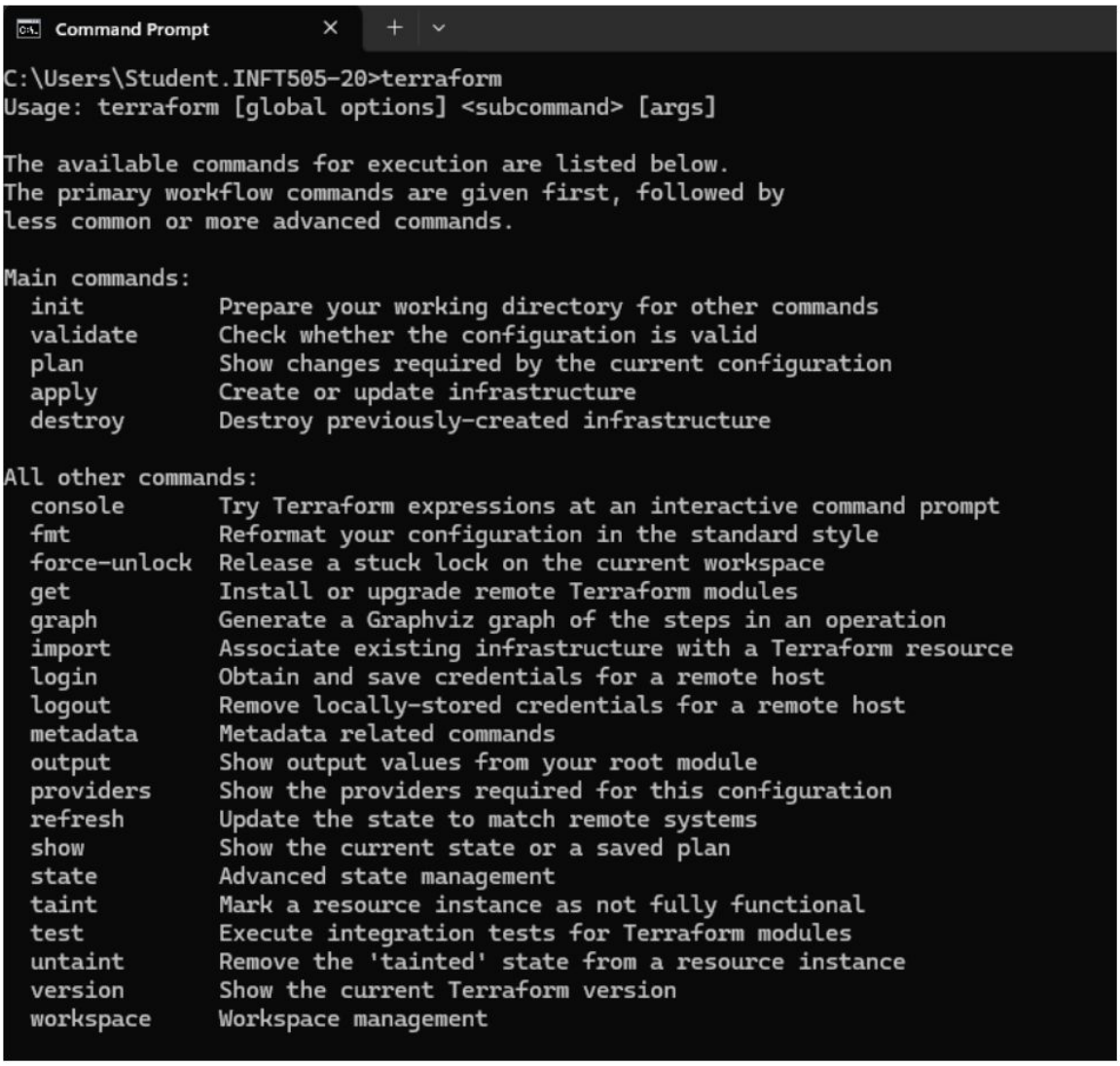




Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
LICENSE	Text Document	2 KB	No	5 KB	63%	20-08-2024 17:35
terraform	Application	24,960 KB	No	84,279 KB	71%	20-08-2024 17:35


Edit environment variable
C:\Program Files\Docker\Docker\resources\bin
C:\Users\Student.INFT505-20\Downloads\terraform_1.9.5_windows_386
C:\Users\Student.INFT505-20\Downloads\terraform_1.9.5_windows_38...
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel


Command Prompt
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Student.INFT505-20>terraform --version
Terraform v1.9.5
on windows_386
C:\Users\Student.INFT505-20>


Command Prompt
C:\Users\Student.INFT505-20>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

Rujuta Medhi **D15A-28**

EXPERIMENT NO. 6

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

Theory:

What is Terraform?

Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It allows you to define, provision, and manage infrastructure resources in a consistent and repeatable way. Terraform uses a high-level configuration language called HashiCorp Configuration Language (HCL) or JSON to describe the desired state of your infrastructure, and it can manage resources across various cloud providers, including AWS, Azure, Google Cloud, and others.

Benefits of Using Terraform with AWS

1.Multi-Cloud Support: Terraform can manage infrastructure across multiple cloud providers, including AWS. This allows you to create a consistent infrastructure environment across different clouds or migrate between them.

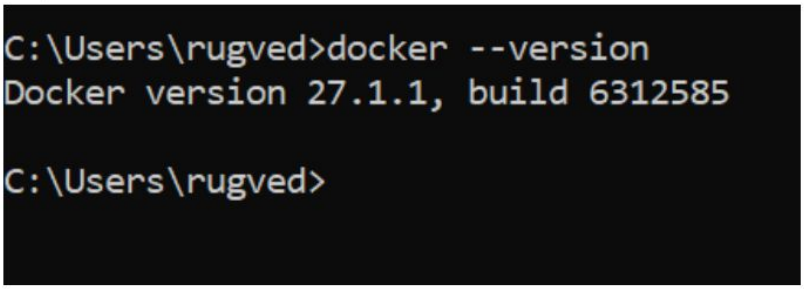
2.Infrastructure as Code: By using Terraform, you can define your AWS infrastructure as code. This makes it easier to version control your infrastructure, automate deployments, and collaborate with others.

3. Scalability and Flexibility: Terraform can manage infrastructure of any size, from small projects to large-scale, complex environments. It allows you to define reusable modules, which can be shared across different projects.

4. State Management: Terraform keeps track of the current state of your infrastructure in a state file. This allows Terraform to determine the actions required to achieve the desired state, ensuring that your infrastructure remains consistent.

Implementation :

Step 1 : check docker installation and version



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

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}
provider "docker" {
  host = "npipe:////.//pipe//docker_engine"
}
resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}
resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}
```

Step 3 : Type terraform init command to initialize terraform backend

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

Terraform has been successfully initialized!

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

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

Step 4(EXTRA) : type terraform fmt and validate commands . The two Terraform commands – terraform validate and terraform fmt – are used to maintain a clean, error-free, and well-structured Terraform codebase.

```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform fmt
docker.tf

C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform validate
Success! The configuration is valid.
```

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

```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform plan

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

Terraform will perform the following actions:

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

Step 6 : Type terraform apply to apply changes .

```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform apply

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

Terraform will perform the following actions:

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

```
      + healthcheck (known after apply)

      + labels (known after apply)

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

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

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

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

  Enter a value: yes

docker_image.nginx: Creating...
docker_image.nginx: Still creating... [10s elapsed]
docker_image.nginx: Still creating... [20s elapsed]
docker_image.nginx: Creation complete after 20s [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 5s [id=e7faae617f21e37b24b13261a2f62d93a1e436f6c1d51fe20f02413332e71c8f]
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Step 7 : Docker container before and after step 6 execution

```
C:\Users\rugved\OneDrive\Documents\terraform\docker>docker container list
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                  NAMES
e7faae617f21   5ef79149e0ec   "/docker-entrypoint.…"   30 seconds ago   Up 28 seconds   0.0.0.0:8000->80/tcp   tutorial
```

```
C:\Users\rugved\OneDrive\Documents\terraform\docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    5ef79149e0ec   13 days ago   188MB
```

Step 8 (EXTRA) : Execution of change .

```
1   terraform {
2     required_providers {
3       docker = {
4         source  = "kreuzwerker/docker"
5         version = "~> 3.0.1"
6       }
7     }
8   }
9   provider "docker" {
10    host = "npipe:////.//pipe//docker_engine"
11  }
12  resource "docker_image" "nginx" {
13    name         = "nginx:latest"
14    keep_locally = false
15  }
16  resource "docker_container" "nginx" {
17    image = docker_image.nginx.image_id
18    name  = "tutorial"
19    ports {
20      internal = 80
21      external = 8080
22    }
23  }
```

```
          + shm_size                                    = (known after apply)
          + start                                       = (known after apply)
          + stdin_open                                  = (known after apply)
          + stop_signal                                 = (known after apply)
          + stop_timeout                                = (known after apply)
          + storage_opts                                = (known after apply)
          + sysctls                                     = (known after apply)
          + tmpfs                                       = (known after apply)
          + tty                                         = (known after apply)
          + user                                        = (known after apply)
          + userns_mode                                 = (known after apply)
          + wait                                        = (known after apply)
          + wait_timeout                                = (known after apply)
          + working_dir                                 = (known after apply)
        } -> (known after apply)

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

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

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

  Enter a value: yes

docker_container.nginx: Destroying... [id=e7faae617f21e37b24b13261a2f62d93a1e436f6c1d51fe20f02413332e71c8f]
docker_container.nginx: Destruction complete after 1s
docker_container.nginx: Creating...
```

Step 9 : terraform destroy to destroy infrastructure.

```
C:\Users\rugved\OneDrive\Documents\terraform\docker>terraform destroy
docker_image.nginx: Refreshing state... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_container.nginx: Refreshing state... [id=5b92ebb92ea6c44d9873ef36755305b00ab85683f4b5becbc46e9f4c6b12888a]

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

Plan: 0 to add, 0 to change, 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.nginx: Destroying... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_image.nginx: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
```

Step 10 : Docker after destroy command.

```
C:\Users\rugved\OneDrive\Documents\terraform\docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\rugved\OneDrive\Documents\terraform\docker>
```

Rujuta Medhi **D15A-28**

EXPERIMENT:07

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

Theory:

Static Application Security Testing (SAST) is a method of debugging by examining source code before a program is run. It involves analyzing the application's source code, bytecode, or binary code to identify vulnerabilities and security flaws. SAST tools scan code for common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and buffer overflows, among others.

Problems SAST Solves:

1. **Early Detection of Vulnerabilities**: SAST enables developers to find security flaws early in the development lifecycle, reducing the cost and effort required to fix them later.
2. **Compliance with Security Standards**: It helps organizations comply with various security regulations and standards, such as PCI DSS, OWASP Top Ten, and ISO 27001, by identifying security weaknesses that need to be addressed.
3. **Integration into CI/CD Pipelines**: SAST tools can be integrated into Continuous Integration/Continuous Deployment (CI/CD) pipelines, allowing for automated security checks during the development process.
4. **Comprehensive Coverage**: It scans all code paths and identifies vulnerabilities that may not be detected during dynamic testing (which tests the application while it runs).
5. **Reduction of Technical Debt**: By catching vulnerabilities early, SAST helps prevent the accumulation of technical debt related to security issues, making the codebase more maintainable.
6. **Improved Code Quality**: Besides security, SAST tools often identify coding best practices and help improve overall code quality.
7. **Enhanced Collaboration**: By providing clear reports and insights, SAST tools foster better communication between development and security teams.
8. **Risk Mitigation**: It helps organizations manage risks associated with software vulnerabilities, thereby protecting against data breaches and cyberattacks.

Implementation:

Integrating Jenkins with SonarQube:

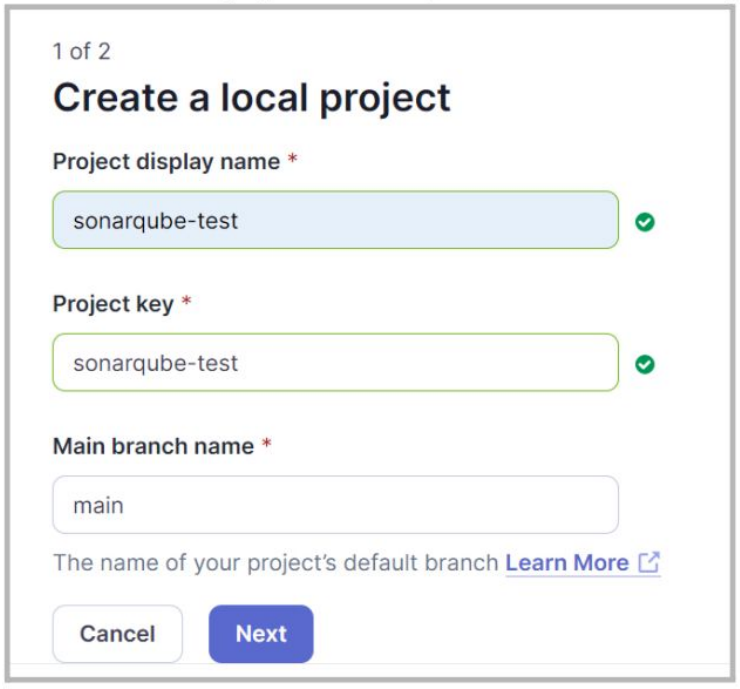
Step 1 Install JDK 1.8

Step 2 download and install jenkins

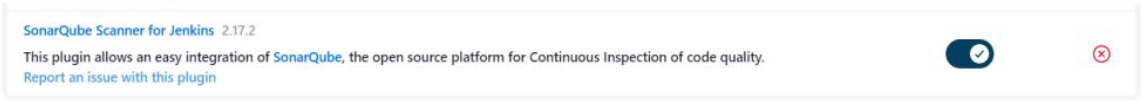
1.Run SonarQube in a Docker container using this command -

```
PS C:\Users\rugved> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:late
st
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
00e224f041dcc60a9cfe77f8e3b79cad3234c7a8ed17d109f58778f655a0d440
PS C:\Users\rugved>
```

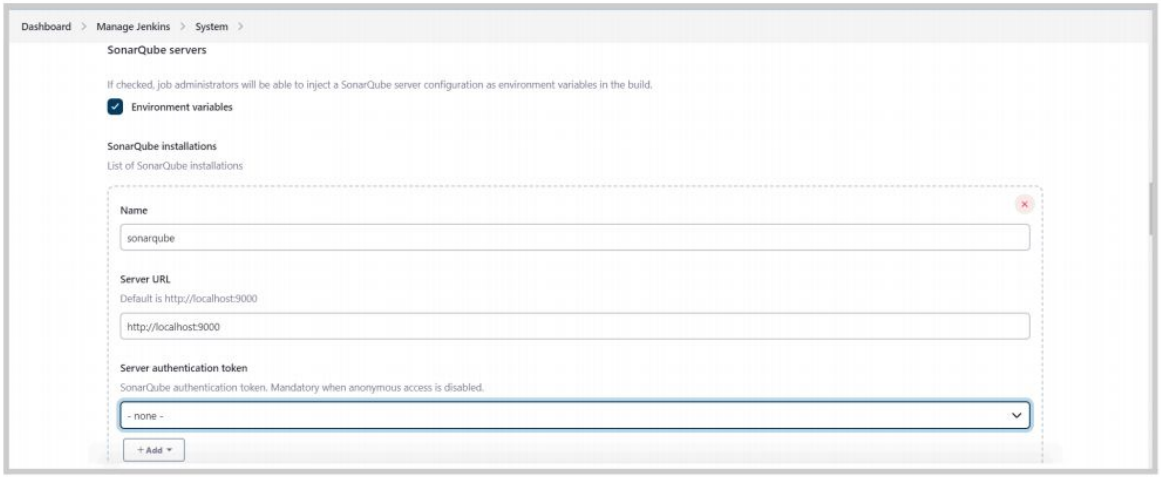
2. Create a manual project in SonarQube with the name sonarqube



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



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



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

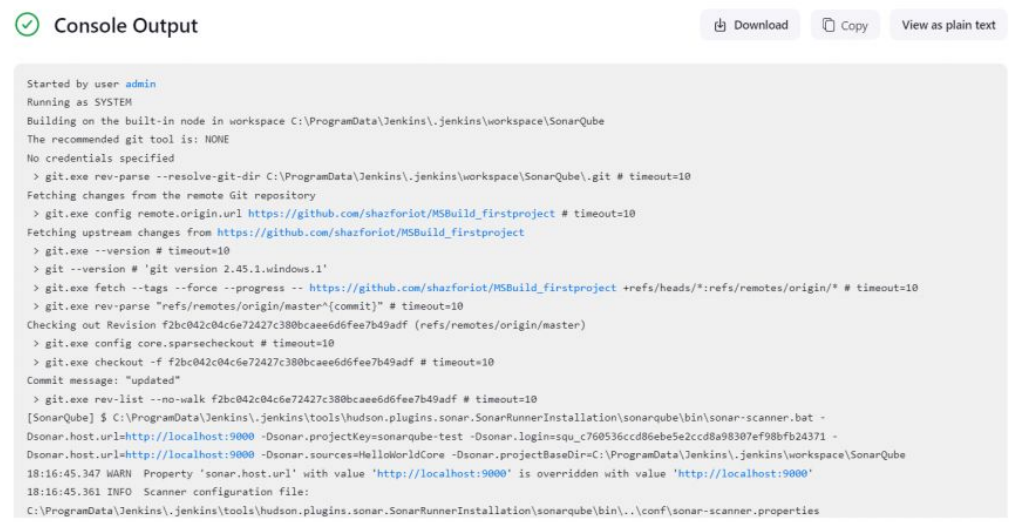
6. create a New Item in Jenkins, choose a freestyle project.

7. 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

8. allow Execute Permissions to the Admin user.

9. Under Build-> Execute SonarQube Scanner, enter these Analysis properties.
Generate the token from Sonarqube->My Account -> Security -> type-User->generate token.

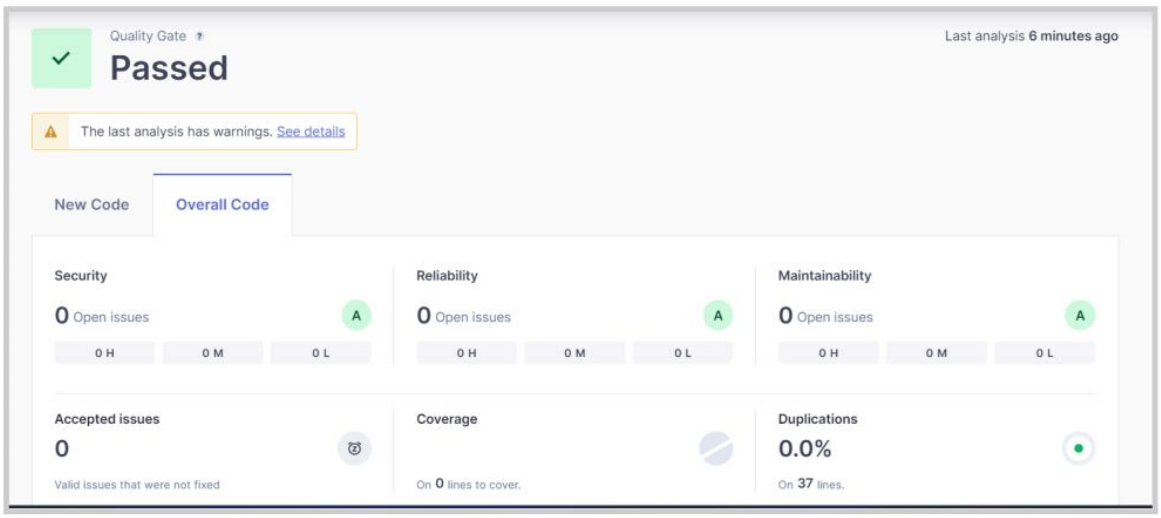
10. After build,check console output




```
18:16:45.396 INFO  SonarScanner CLI 6.2.0.4584
18:16:45.398 INFO  Java 17.0.12 Eclipse Adoptium (64-bit)
18:16:45.399 INFO  Windows 11 10.0 amd64
18:16:45.430 INFO  User cache: C:\WINDOWS\system32\config\systemprofile\.sonar\cache
18:16:47.767 INFO  JRE provisioning: os[windows], arch[amd64]
18:16:59.174 INFO  Communicating with SonarQube Server 10.6.0.92116
18:17:00.085 INFO  Starting SonarScanner Engine...
18:17:00.088 INFO  Java 17.0.11 Eclipse Adoptium (64-bit)
18:17:01.786 INFO  Load global settings
18:17:02.214 INFO  Load global settings (done) | time=428ms
18:17:02.220 INFO  Server id: 147B411E-AZIpEKOrG4EowtOFhNu0
18:17:02.245 INFO  Loading required plugins
18:17:02.246 INFO  Load plugins index
18:17:02.545 INFO  Load plugins index (done) | time=300ms
18:17:02.546 INFO  Load/download plugins
18:17:06.679 INFO  Load/download plugins (done) | time=4133ms
18:17:07.558 INFO  Process project properties
18:17:07.576 INFO  Process project properties (done) | time=18ms
18:17:07.599 INFO  Project key: sonarqube-test
18:17:07.600 INFO  Base dir: C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube
18:17:07.601 INFO  Working dir: C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube\.scannerwork
18:17:07.631 INFO  Load project settings for component key: 'sonarqube-test'
18:17:08.083 INFO  Load project settings for component key: 'sonarqube-test' (done) | time=452ms
18:17:08.134 INFO  Load quality profiles
18:17:08.656 INFO  Load quality profiles (done) | time=522ms
18:17:08.670 INFO  Auto-configuring with CI 'Jenkins'
18:17:08.747 INFO  Load active rules
18:17:36.634 INFO  Load active rules (done) | time=27819ms
```

```
18:17:44.043 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the
SonarScanner for .NET 5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
18:17:44.051 INFO  Sensor C# [csharp] (done) | time=0ms
18:17:44.053 INFO  Sensor Analysis Warnings import [csharp]
18:17:44.056 INFO  Sensor Analysis Warnings import [csharp] (done) | time=0ms
18:17:44.057 INFO  Sensor C# File Caching Sensor [csharp]
18:17:44.062 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
18:17:44.063 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=0ms
18:17:44.064 INFO  Sensor Zero Coverage Sensor
18:17:44.064 INFO  Sensor Zero Coverage Sensor (done) | time=8ms
18:17:44.066 INFO  SCM Publisher SCM provider for this project is: git
18:17:44.066 INFO  SCM Publisher 2 source files to be analyzed
18:17:44.805 INFO  SCM Publisher 2/2 source files have been analyzed (done) | time=742ms
18:17:44.816 INFO  CPD Executor Calculating CPD for 0 files
18:17:44.816 INFO  CPD Executor CPD calculation finished (done) | time=0ms
18:17:44.825 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
18:17:45.202 INFO  Analysis report generated in 165ms, dir size=199.0 kB
18:17:45.279 INFO  Analysis report compressed in 58ms, zip size=20.6 kB
18:17:45.587 INFO  Analysis report uploaded in 305ms
18:17:45.589 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test
18:17:45.589 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
18:17:45.590 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=f1f1bf2b-6ac8-498b-afc3-de9bf4555929
18:17:45.608 INFO  Analysis total time: 38.695 s
18:17:45.611 INFO  SonarScanner Engine completed successfully
18:17:45.722 INFO  EXECUTION SUCCESS
18:17:45.815 INFO  Total time: 1:00.370s
Finished: SUCCESS
```

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



Rujuta Medhi D15A-28

EXPERIMENT:08

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

THEORY:

Static Application Security Testing (SAST) :

SAST is a methodology for testing an application's source code to identify security vulnerabilities before the code is compiled. This type of testing, also referred to as white-box testing, helps improve application security by finding weaknesses early in development.

Problems SAST Solves

● Early Detection: SAST finds vulnerabilities early in the Software Development Life Cycle (SDLC), allowing developers to fix issues without affecting builds or passing vulnerabilities to the final release.

● Real-Time Feedback: Developers receive immediate feedback during coding, helping them address security issues before moving to the next stage of development.

● Graphical Representations: SAST tools often provide visual aids to help developers navigate the code and identify the exact location of vulnerabilities, offering suggestions for fixes.

● Regular Scanning: SAST tools can be configured to scan code regularly, such as during daily builds, code check-ins, or before releases.

Importance of SAST

● Resource Efficiency: With a larger number of developers than security experts, SAST allows full codebase analysis quickly and efficiently, without relying on manual code reviews.

● Speed: SAST tools can analyze millions of lines of code within minutes, detecting critical vulnerabilities such as buffer overflows, SQL injection, and cross-site scripting (XSS) with high accuracy.

CI/CD Pipeline

A Continuous Integration/Continuous Delivery (CI/CD) pipeline is a sequence of automated tasks designed to build, test, and deploy new software versions rapidly and consistently. It plays a crucial role in DevOps practices, ensuring fast and reliable software releases.

SonarQube

SonarQube is an open-source platform from SonarSource that performs continuous code quality inspections through static code analysis. It identifies bugs, code smells, security vulnerabilities,

and code duplications in a wide range of programming languages. SonarQube is extendable with plugins and integrates seamlessly into CI/CD pipelines.

Benefits of SonarQube

Sustainability: By reducing complexity and vulnerabilities, SonarQube extends the lifespan of applications and helps maintain cleaner code. Increased

Productivity: SonarQube minimizes maintenance costs and risks, resulting in fewer code changes and a more stable codebase.

Quality Code: Ensures code quality checks are integrated into the development process. Error

Detection: Automatically identifies coding errors and alerts developers to resolve them before moving to production.

Consistency: Helps maintain consistent code quality by detecting and reporting violations of coding standards. Business

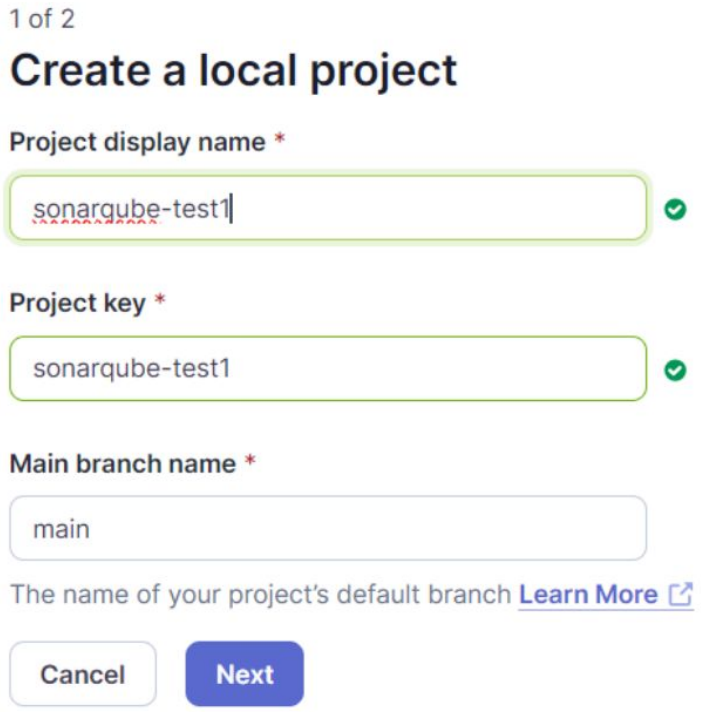
Scaling: SonarQube supports scaling as the business grows without any restrictions.

Implementation:

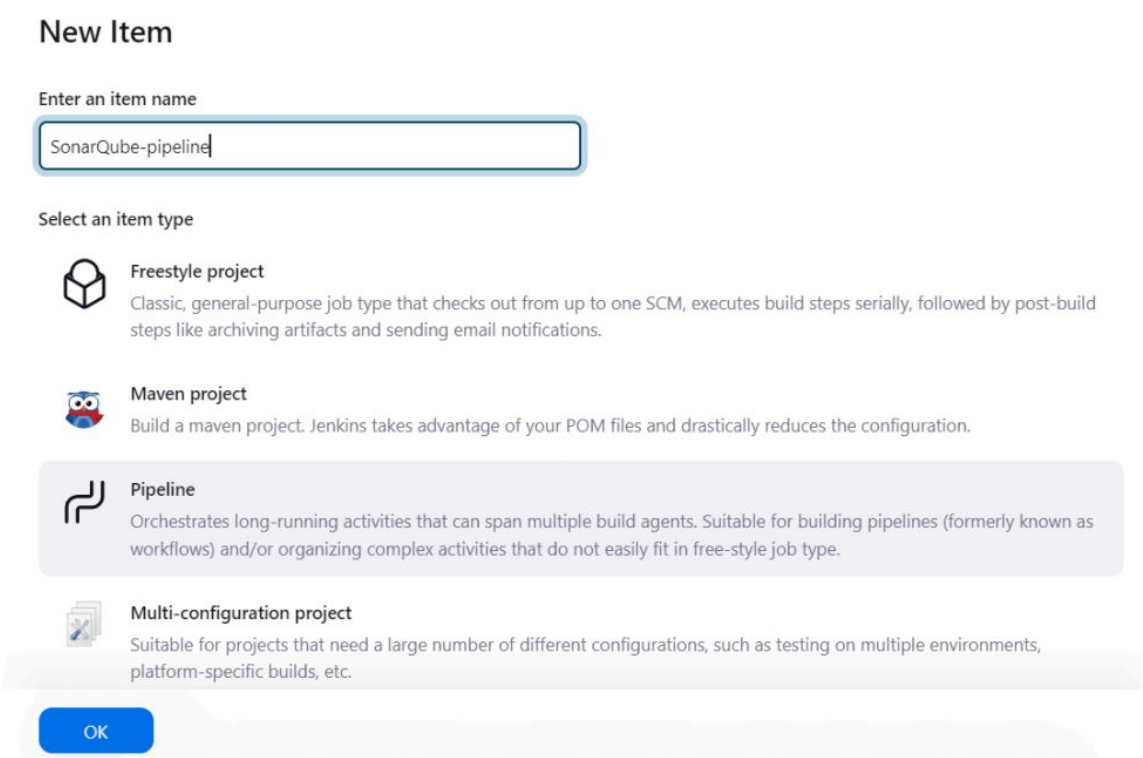
Prerequisites:

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

1. Login to SonarQube create a manual project in SonarQube with the name sonarqube-test



2. Create a New Item in Jenkins, choose Pipeline.



3. 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/"
}
}
}
```

Definition

Pipeline script

Script

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }
    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            bat """
            C:\\Users\\rugved\\Downloads\\sonar-scanner-cli-6.2.0.4584-windows-x64\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat ^
            -Dsonar.login=admin ^
            -Dsonar.password=rujutamedhi@04 ^
            -Dsonar.projectKey=sonarqube-test1 ^
            -Dsonar.exclusions=vendor/*,resources/,/.java ^
            -Dsonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

4. Build the project and check the console output

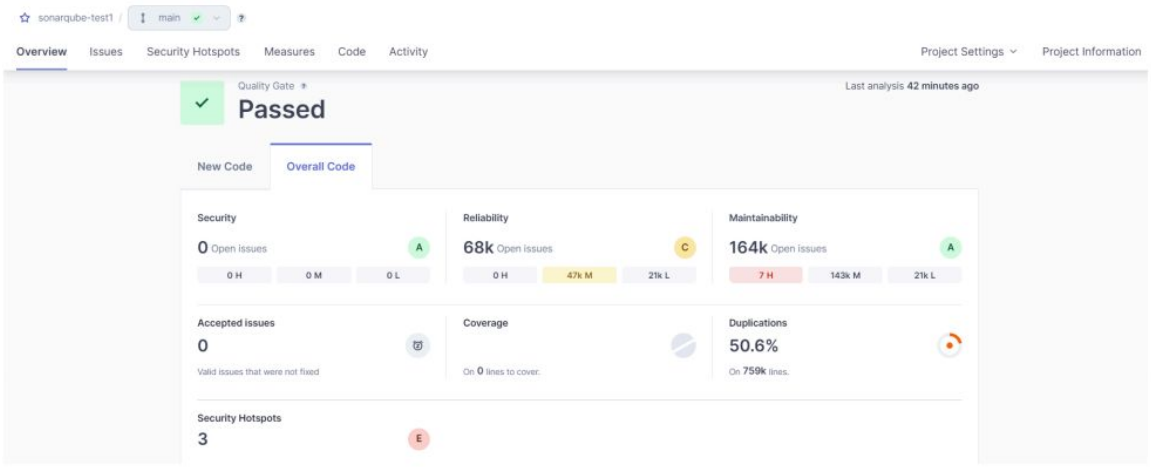
Console Output

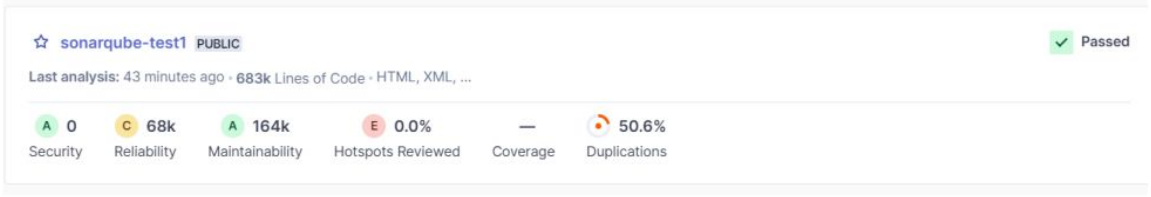
Download Copy View

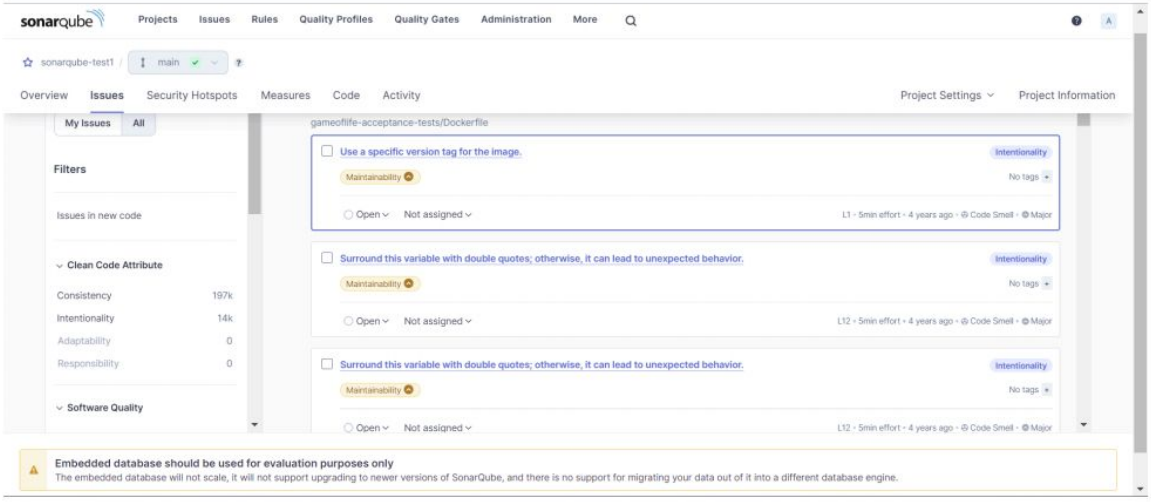
```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\SonarQube-pipeline\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.45.1.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
line 551. Keep only the first 100 references.
22:19:32.482 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/control/gui/WorkBenchGui.html fo
line 158. Keep only the first 100 references.
22:19:32.483 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/control/gui/WorkBenchGui.html fo
line 551. Keep only the first 100 references.
22:19:32.489 INFO  CPD Executor CPD calculation finished (done) | time=205503ms
22:19:32.926 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
22:22:24.354 INFO  Analysis report generated in 9704ms, dir size=127.2 MB
22:22:56.383 INFO  Analysis report compressed in 32019ms, zip size=29.6 MB
22:23:01.253 INFO  Analysis report uploaded in 4863ms
22:23:01.261 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test1
22:23:01.261 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
22:23:01.261 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=ef72a135-bbcf-4922-a543-4b8bf5ac62b6
22:23:31.270 INFO  Analysis total time: 19:19.694 s
22:23:31.375 INFO  SonarScanner Engine completed successfully
22:23:31.967 INFO  EXECUTION SUCCESS
22:23:32.113 INFO  Total time: 19:24.977s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

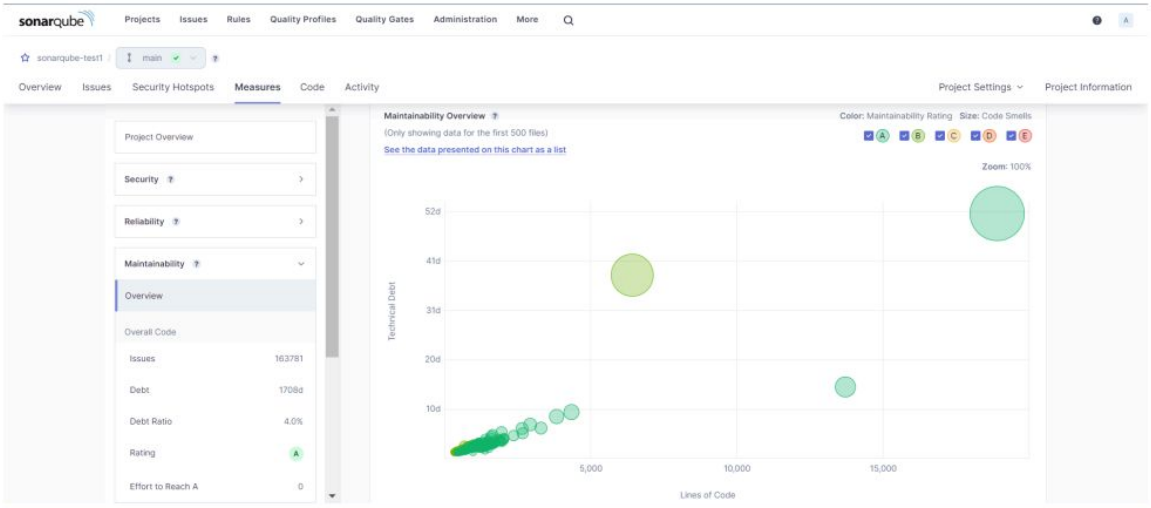
5. After that, check the project in SonarQube

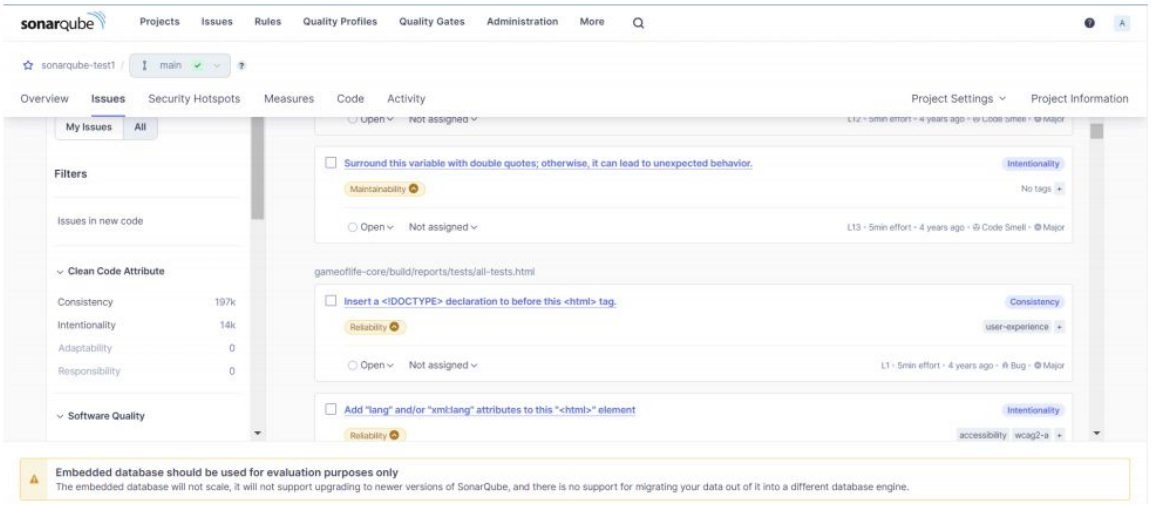
6. After that, check the project in SonarQube











Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

Rujuta Medhi D15A-28

EXPERIMENT:09

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

Theory:

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

Features of Nagios

Following are the important features of Nagios monitoring tool:
- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring
  process
- You can monitor the entire business process and IT infrastructure with a single pass
- The product's architecture is easy to write new plugins in the language of your choice
- Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
- Utilizes topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers that runs during service or host events for proactive

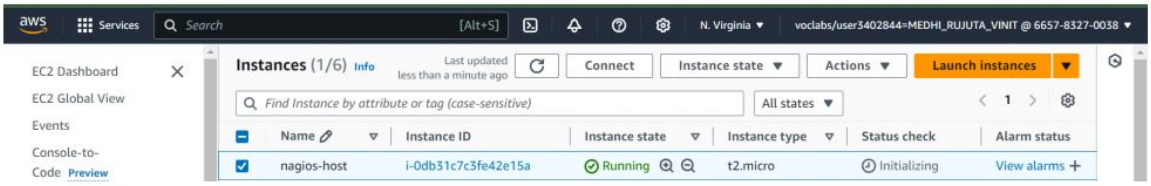
problem resolution

- Support for implementing redundant monitoring hosts

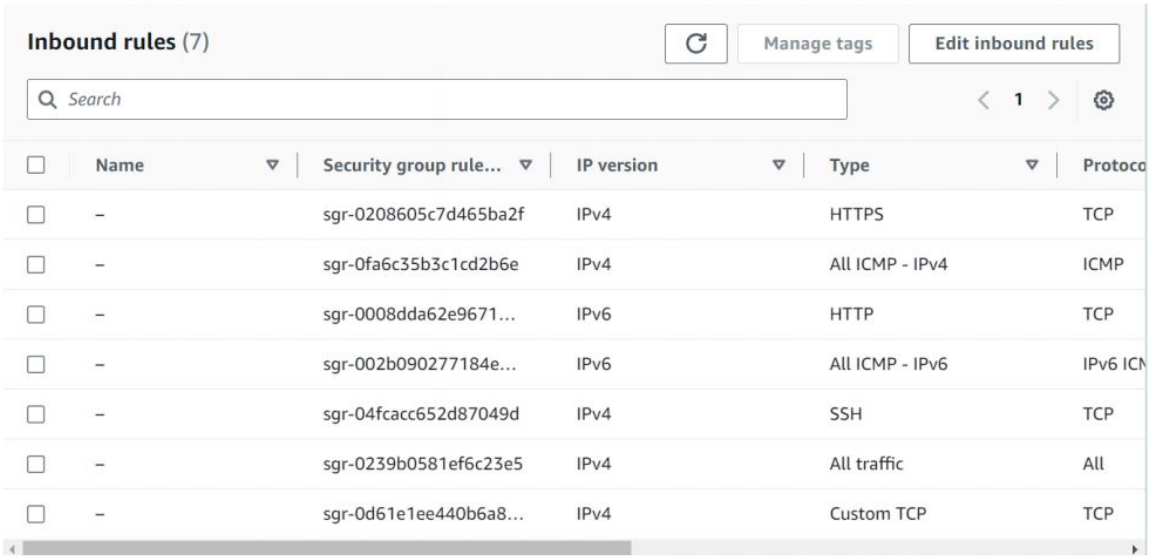
Implementation:

Installation of Nagios:

1. Create an Amazon Linux EC2 Instance in AWS and name it - nagios-host



2. Under Security Group, make sure HTTP, HTTPS, SSH, ICMP are open from everywhere.



3. Update the package indices and install the following packages using yum

```
[ec2-user@ip-172-31-80-189 ~]$ sudo yum update
Last metadata expiration check: 0:13:07 ago on Thu Sep 26 08:28:45 2024.
Dependencies resolved.
Nothing to do.
Complete!
```

```
[ec2-user@ip-172-31-80-189 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:13:29 ago on Thu Sep 26 08:28:45 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
httpd	x86_64	2.4.62-1.amzn2023	amazonlinux	48 k
php8.3	x86_64	8.3.10-1.amzn2023.0.1	amazonlinux	10 k
Installing dependencies:				
apr	x86_64	1.7.2-2.amzn2023.0.2	amazonlinux	129 k
apr-util	x86_64	1.6.3-1.amzn2023.0.1	amazonlinux	98 k
generic-logos-httpd	noarch	18.0.0-12.amzn2023.0.3	amazonlinux	19 k
httpd-core	x86_64	2.4.62-1.amzn2023	amazonlinux	1.4 M
httpd-filesystem	noarch	2.4.62-1.amzn2023	amazonlinux	14 k
httpd-tools	x86_64	2.4.62-1.amzn2023	amazonlinux	81 k
libbrotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	315 k
libsodium	x86_64	1.0.19-4.amzn2023	amazonlinux	176 k
libxslt	x86_64	1.1.34-5.amzn2023.0.2	amazonlinux	241 k
mailcap	noarch	2.1.49-3.amzn2023.0.3	amazonlinux	33 k

```
[ec2-user@ip-172-31-80-189 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:14:04 ago on Thu Sep 26 08:28:45 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
gcc	x86_64	11.4.1-2.amzn2023.0.2	amazonlinux	32 M
Installing dependencies:				
annobin-docs	noarch	10.93-1.amzn2023.0.1	amazonlinux	92 k
annobin-plugin-gcc	x86_64	10.93-1.amzn2023.0.1	amazonlinux	887 k
cpp	x86_64	11.4.1-2.amzn2023.0.2	amazonlinux	10 M
gc	x86_64	8.0.4-5.amzn2023.0.2	amazonlinux	105 k
glibc-devel	x86_64	2.34-52.amzn2023.0.11	amazonlinux	27 k
glibc-headers-x86	noarch	2.34-52.amzn2023.0.11	amazonlinux	427 k
guile22	x86_64	2.2.7-2.amzn2023.0.3	amazonlinux	6.4 M
kernel-headers	x86_64	6.1.109-118.189.amzn2023	amazonlinux	1.4 M
libmpc	x86_64	1.2.1-2.amzn2023.0.2	amazonlinux	62 k
libtool-ltdl	x86_64	2.4.7-1.amzn2023.0.3	amazonlinux	38 k

```
[ec2-user@ip-172-31-80-189 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:20:01 ago on Thu Sep 26 08:28:45 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

4. 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

```
Last login: Thu Sep 26 08:41:37 2024 from 18.206.107.29
[ec2-user@ip-172-31-80-189 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-80-189 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-80-189 ~]$
```

5. Create a new user group
sudo groupadd nagcmd

6. Use these commands so that you don't have to use sudo for Apache and Nagios
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-80-189 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-80-189 ~]$ sudo usermod -a -G nagcmd apache
```

7. Create a new directory for Nagios downloads
mkdir ~/downloads
cd ~/downloads

8. Use wget to download the source zip files.
wget
http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz
wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz

```
[ec2-user@ip-172-31-80-189 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-80-189 ~]$ cd ~/downloads
[ec2-user@ip-172-31-80-189 downloads]$ wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
--2024-09-26 08:58:48--  http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2659772 (2.5M) [application/x-gzip]
Saving to: 'nagios-plugins-2.0.3.tar.gz'

nagios-plugins-2.0.3.tar.gz       100%[==========================================================>]   2.54M  10.7MB/s    in 0.2s

2024-09-26 08:58:48 (10.7 MB/s) - 'nagios-plugins-2.0.3.tar.gz' saved [2659772/2659772]
```

```
[ec2-user@ip-172-31-80-189 nagios-plugins-2.0.3]$ wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz
--2024-09-26 09:45:07--  http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz
Resolving prdownloads.sourceforge.net (prdownloads.sourceforge.net)... 204.68.111.105
Connecting to prdownloads.sourceforge.net (prdownloads.sourceforge.net)|204.68.111.105|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz [following]
--2024-09-26 09:45:07--  http://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz
Resolving downloads.sourceforge.net (downloads.sourceforge.net)... 204.68.111.105
Reusing existing connection to prdownloads.sourceforge.net:80.
HTTP request sent, awaiting response... 302 Found
Location: http://psychz.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1 [following]
--2024-09-26 09:45:07--  http://psychz.dl.sourceforge.net/project/nagios/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz?viasf=1
Resolving psychz.dl.sourceforge.net (psychz.dl.sourceforge.net)... 208.87.241.191
Connecting to psychz.dl.sourceforge.net (psychz.dl.sourceforge.net)|208.87.241.191|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1805059 (1.7M) [application/x-gzip]
Saving to: 'nagios-4.0.8.tar.gz'

nagios-4.0.8.tar.gz              100%[=====================================================>]   1.72M  2.63MB/s    in 0.7s

2024-09-26 09:45:08 (2.63 MB/s) - 'nagios-4.0.8.tar.gz' saved [1805059/1805059]
```

9. Use tar to unzip and change to that directory.
tar zxvf nagios-4.0.8.tar.gz

```
[ec2-user@ip-172-31-80-189 downloads]$ tar zxvf nagios-plugins-2.0.3.tar.gz
nagios-plugins-2.0.3/
nagios-plugins-2.0.3/perlmods/
nagios-plugins-2.0.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.0.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.0.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.in
nagios-plugins-2.0.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.am
nagios-plugins-2.0.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.0.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.0.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.0.3/perlmods/Class-Accessor-0.34.tar.gz
nagios-plugins-2.0.3/perlmods/Try-Tiny-0.18.tar.gz
nagios-plugins-2.0.3/perlmods/Module-Implementation-0.07.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile
nagios-plugins-2.0.3/perlmods/Perl-OSType-1.003.tar.gz
nagios-plugins-2.0.3/perlmods/install_order
nagios-plugins-2.0.3/perlmods/Nagios-Plugin-0.36.tar.gz
nagios-plugins-2.0.3/perlmods/Math-Calc-Units-1.07.tar.gz
nagios-plugins-2.0.3/perlmods/Module-Build-0.4007.tar.gz
```

10. Run the configuration script with the same group name you previously created.
./configure --with-command-group=nagcmd

```
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
```

```
[ec2-user@ip-172-31-80-189 nagios-4.0.8]$ sudo yum install -y httpd php gcc glibc glibc-common make net-snmp
Last metadata expiration check: 1:24:24 ago on Thu Sep 26 08:28:45 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.
Package gcc-11.4.1-2.amzn2023.0.2.x86_64 is already installed.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package make-1:4.3-5.amzn2023.0.2.x86_64 is already installed.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
net-snmp	x86_64	1:5.9.3-2.amzn2023.0.2	amazonlinux	296 k
Installing dependencies:				
mariadb-connector-c	x86_64	3.1.13-1.amzn2023.0.3	amazonlinux	196 k
mariadb-connector-c-config	noarch	3.1.13-1.amzn2023.0.3	amazonlinux	9.2 k
net-snmp-agent-libs	x86_64	1:5.9.3-2.amzn2023.0.2	amazonlinux	696 k
net-snmp-libs	x86_64	1:5.9.3-2.amzn2023.0.2	amazonlinux	745 k
perl-B	x86_64	1.80-477.amzn2023.0.6	amazonlinux	179 k
perl-Data-Dumper	x86_64	2.174-460.amzn2023.0.2	amazonlinux	55 k

11. Compile the source code.
make all

12. Install binaries, init script and sample config files. Lastly, set permissions on the external
command directory.
sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode

```
[ec2-user@ip-172-31-80-189 nagios-4.0.8]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/base'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.0.8/base'
make strip-post-install
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/base'
/usr/bin/strip /usr/local/nagios/bin/nagios
/usr/bin/strip /usr/local/nagios/bin/nagiostats
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.0.8/base'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.0.8/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.0.8/cgi'
make install-basic
```

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

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

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

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

13. Edit the config file and change the email address.
sudo nano /usr/local/nagios/etc/objects/contacts.cfg

```
# template which is defined elsewhere.

define contact{
        contact_name                    nagiosadmin             ; Short name of user
        use                             generic-contact         ; Inherit default values from generic-contact template (
        alias                           Nagios Admin            ; Full name of user

        email                           rujutamedhi@gmail.com   ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
        }
```

14. Configure the web interface.
sudo make install-webconf

```
[ec2-user@ip-172-31-80-189 nagios-4.0.8]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf

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

[ec2-user@ip-172-31-80-189 nagios-4.0.8]$
```

15. Create a nagiosadmin account for nagios login along with password. You'll have to specify the password twice.
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-80-189 nagios-4.0.8]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-80-189 nagios-4.0.8]$
```

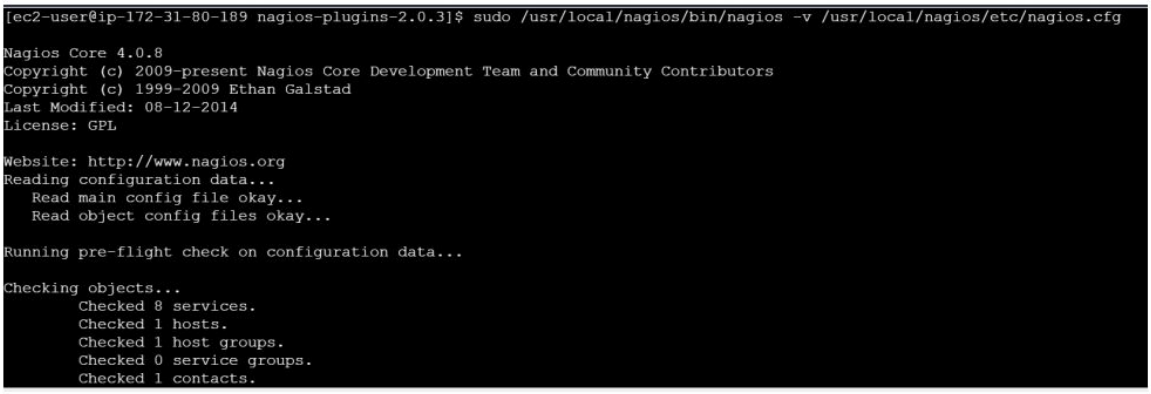
16. Restart Apache
sudo service httpd restart

```
Adding password for user nagiosadmin
[ec2-user@ip-172-31-80-189 nagios-4.0.8]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-80-189 nagios-4.0.8]$
```

17. Go back to the downloads folder and unzip the plugins zip file.
cd ~/downloads
tar zxvf nagios-plugins-2.0.3.tar.gz

18. Compile and install plugins
cd nagios-plugins-2.0.3
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install

19. Start Nagios
Add Nagios to the list of system services
sudo chkconfig --add nagios
sudo chkconfig nagios on
Verify the sample configuration files
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg



```
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 1 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0
```

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-80-189 nagios-plugins-2.0.3]$ sudo service nagios start
Starting nagios (via systemctl):                           [  OK  ]
```

20. Check the status of Nagios
sudo systemctl status nagios

```
[ec2-user@ip-172-31-80-189 nagios-plugins-2.0.3]$ sudo systemctl status nagios
● nagios.service - LSB: Starts and stops the Nagios monitoring server
     Loaded: loaded (/etc/rc.d/init.d/nagios; generated)
     Active: active (running) since Thu 2024-09-26 10:10:23 UTC; 3min 24s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 128105 ExecStart=/etc/rc.d/init.d/nagios start (code=exited, status=0/SUCCESS)
      Tasks: 6 (limit: 1112)
     Memory: 2.6M
        CPU: 103ms
     CGroup: /system.slice/nagios.service
             ├─128127 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─128129 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─128130 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─128131 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─128132 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─128133 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 26 10:10:23 ip-172-31-80-189.ec2.internal nagios[128127]: nerd: Channel opathchecks registered successfully
Sep 26 10:10:23 ip-172-31-80-189.ec2.internal nagios[128127]: nerd: Fully initialized and ready to rock!
Sep 26 10:10:23 ip-172-31-80-189.ec2.internal nagios[128127]: wproc: Successfully registered manager as @wproc with query hand
Sep 26 10:10:23 ip-172-31-80-189.ec2.internal nagios[128127]: wproc: Registry request: name=Core Worker 128132;pid=128132
```

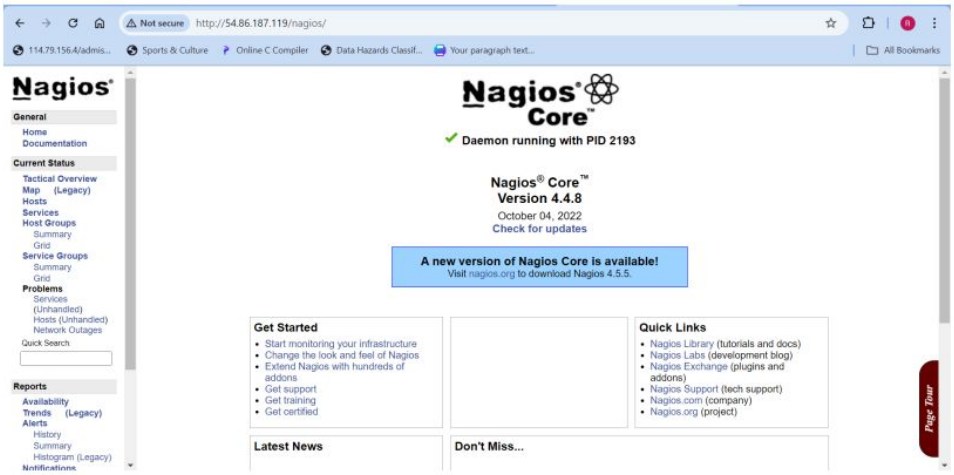
Forbidden

You don't have permission to access this resource.

Error:
[ec2-user@ip-172-31-80-189 nagios-plugins-2.0.3]$ sudo make install-init make: *** No rule to make target 'install-init'. Stop.

21. Received error forbidden:
To solve these errors run:
Install the OpenSSL development libraries and any other required dependencies:
sudo yum install openssl-devel -y
sudo yum install httpd gcc glibc glibc-common perl php gcc-c++ make wget -y
sudo yum install php-mysqlnd -y

22. After entering the correct credentials, you will see this page.



Rujuta Medhi **D15A-28**

Experiment 10

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

Theory:

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

Key Components of Nagios

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

How Nagios Works

1. **Configuration**: Administrators define what to monitor and how to monitor it using configuration files.
2. **Plugins:** Nagios uses plugins to gather information about the status of various services and hosts. These plugins can be custom scripts or pre-built ones.
3. **Scheduling:** Nagios schedules regular checks of the defined services and hosts using the configured plugins.
4. **Alerting:** If a check indicates a problem, Nagios triggers an alert. Alerts can be configured to escalate if not acknowledged within a certain timeframe.

5. **Log Management**: Centralizing and analyzing logs from various sources to detect issues and ensure compliance.

Implementation :

Prerequisites

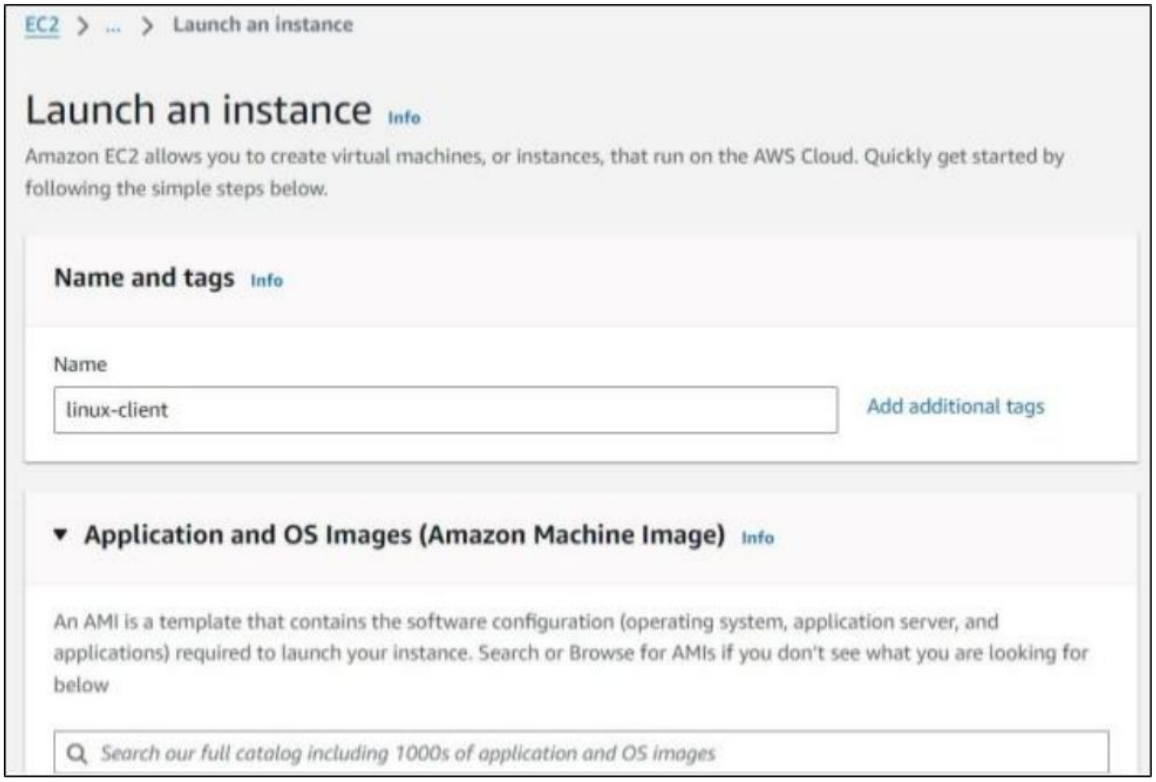
- AWS Free Tier
- Nagios Server running on an Amazon Linux Machine

1. **Confirm Nagios is Running on the Server**
   - sudo systemctl status nagios
   - Proceed if you see that Nagios is active and running.

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-10-07 16:28:45 UTC; 38s ago
       Docs: https://www.nagios.org/documentation
    Process: 69362 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
    Process: 69363 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (c
   Main PID: 69364 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.1M
        CPU: 22ms
     CGroup: /system.slice/nagios.service
             ├─69364 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─69365 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69366 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69367 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69368 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─69369 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

2. Create an Ubuntu 20.04 Server EC2 Instance

- Name it linux-client.
- Use the same security group as the Nagios Host



3. Verify Nagios Process on the Server

Commands

- - ps -ef | grep nagios

```
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ ps -ef | grep nagios
nagios     69364       1  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios -d
nagios     69365   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69366   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69367   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69368   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69369   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios -d
ec2-user   70969    2909  0 16:55 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$
```

4. Become Root User and Create Directories

sudo su

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

```
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-42-50 nagios-plugins-2.3.3]#
```

5. Copy Sample Configuration File

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

```
[root@ip-172-31-42-50 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/
[root@ip-172-31-42-50 ec2-user]# nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-42-50 ec2-user]#
```

6. Edit the Configuration File

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

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

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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   linuxserver
    address                 127.0.0.1
}

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

7. Update Nagios Configuration

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

- Add the following line: cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
- Change hostgroup_name under hostgroup to linux-servers1

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

# Define an optional hostgroup for Linux machines

define hostgroup {

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



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

8. Verify Configuration Files

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

```
[root@ip-172-31-42-50 ec2-user]# sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

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

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

Running pre-flight check on configuration data...

Checking objects...
        Checked 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 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-172-31-42-50 ec2-user]#
```

9. **Restart Nagios Service**
   - sudo systemctl restart nagios

10. SSH into the Client Machine

● Use SSH or EC2 Instance Connect to access the linux-client.

11. Update Package Index and Install Required Packages

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

12. Edit NRPE Configuration File Commands -

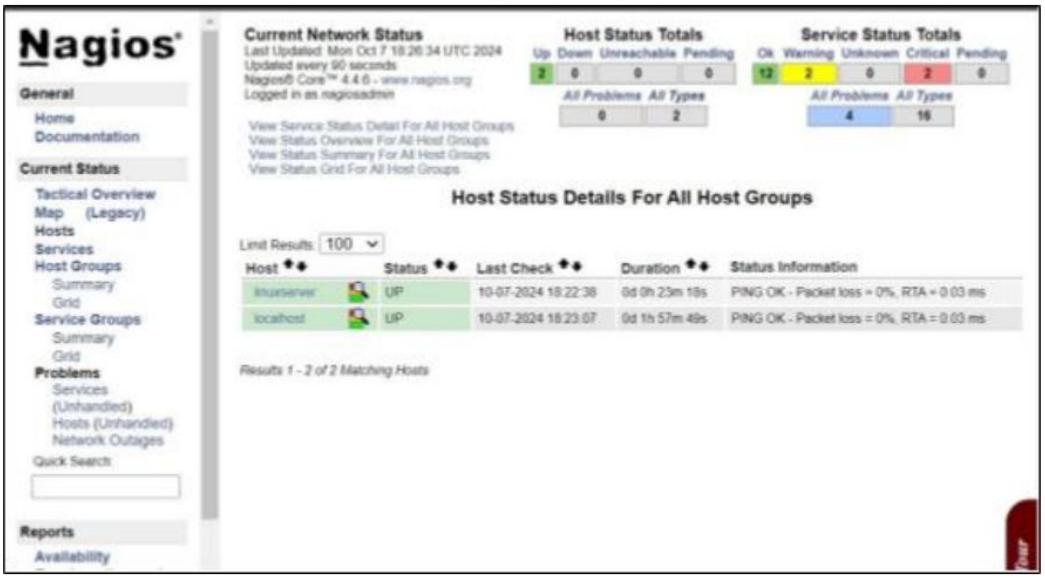
sudo nano /etc/nagios/nrpe.cfg

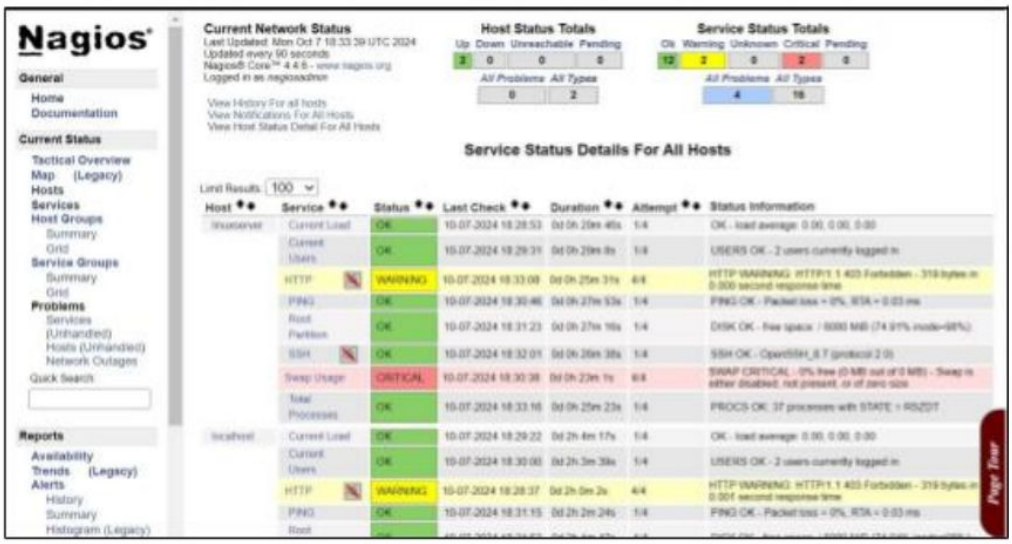
● Add your Nagios host IP address under allowed_hosts: allowed_hosts=

```
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1,3.81.151.142



# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
# to specify arguments to commands that are executed.  This option only works
# if the daemon was configured with the --enable-command-args configure script
# option.
#
```

14. Check Nagios Dashboard

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





Conclusion:

To perform port, service, and Windows/Linux server monitoring using Nagios, configure the necessary plugins and agents, define the monitoring parameters in the configuration files, and set up alerting mechanisms to ensure timely notifications of any issues. This comprehensive approach ensures robust monitoring and quick response to potential problems, maintaining the health and performance of your IT infrastructure.

Rujuta Medhi D15A-28

EXPERIMENT:11

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

Theory:

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

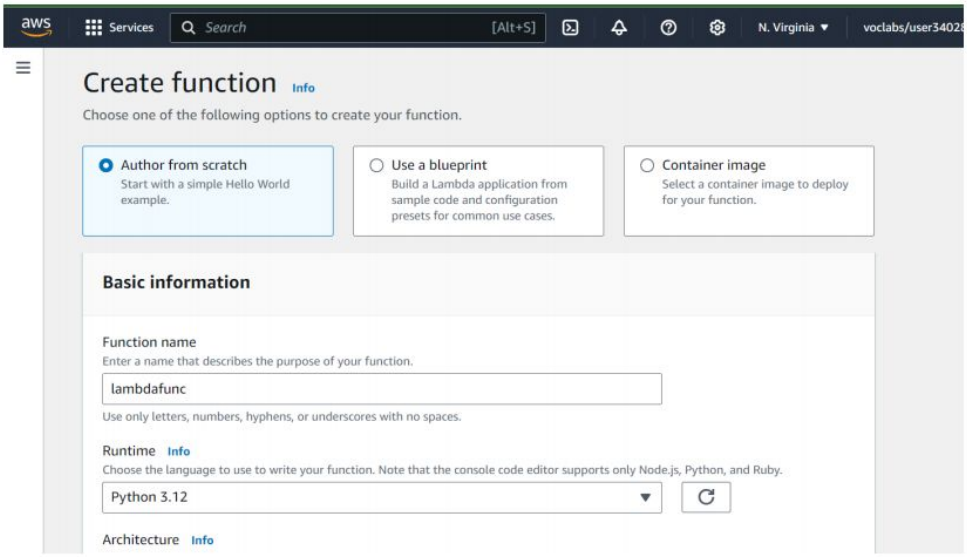
The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers,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

Features of AWS Lambda

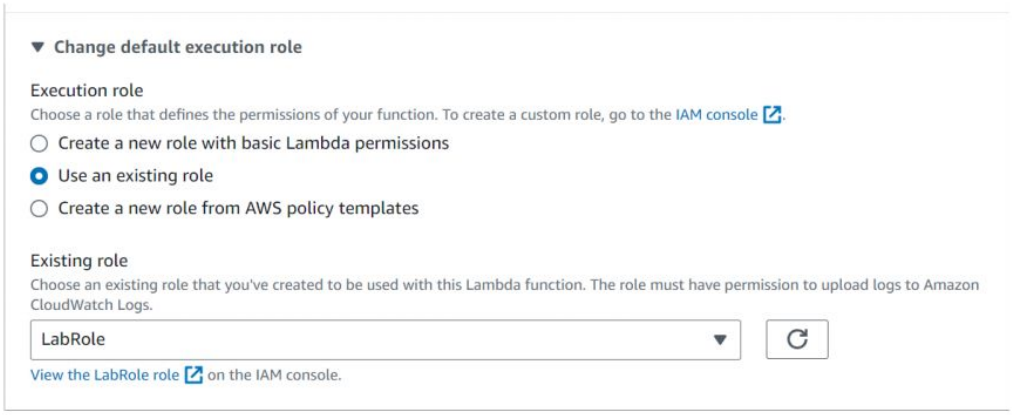
● AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
● It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
● You don't need to invest upfront. You pay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

Implementation:
Steps to create an AWS Lambda function

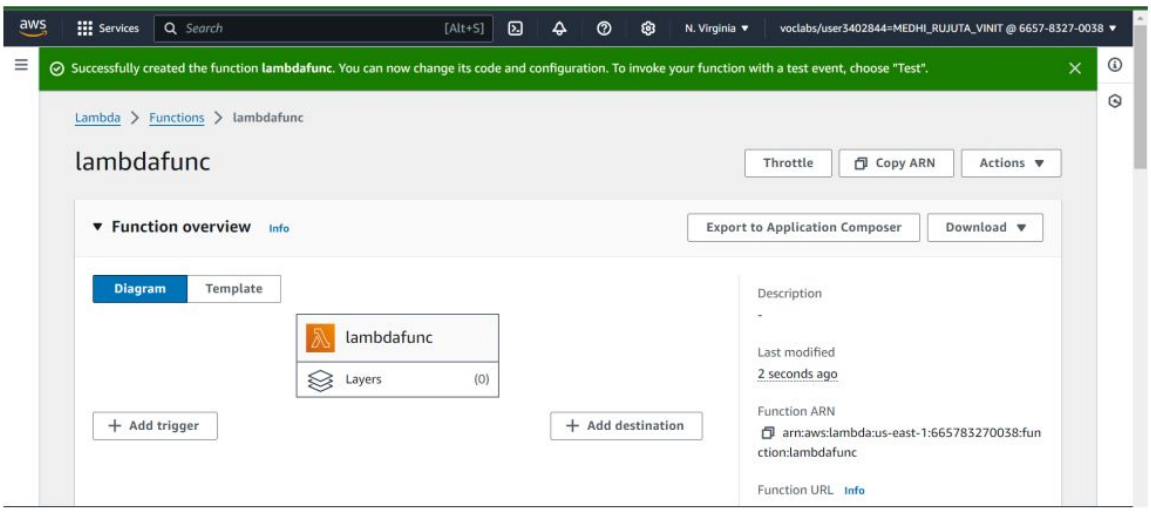
1. Open up the Lambda Console and click on the Create button.



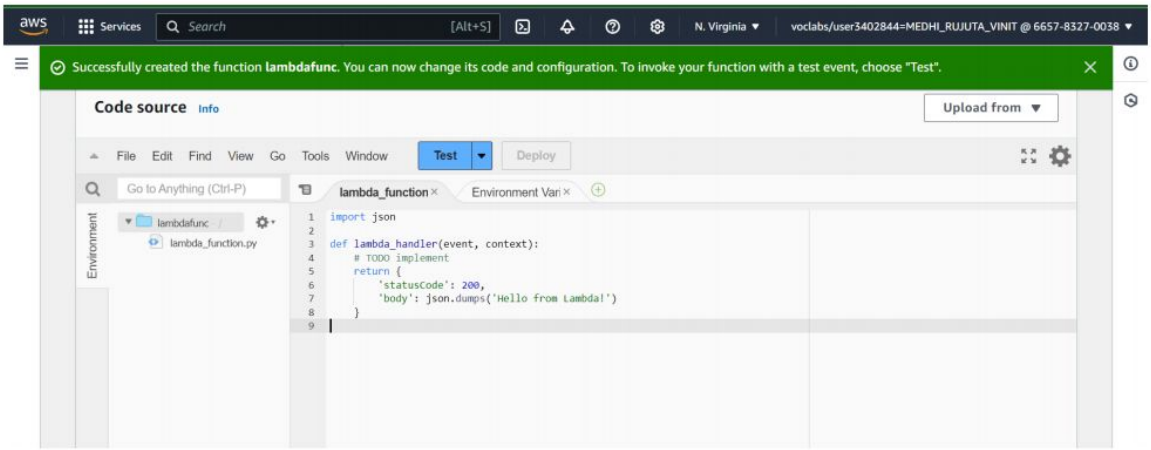
2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.
Then, choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones.



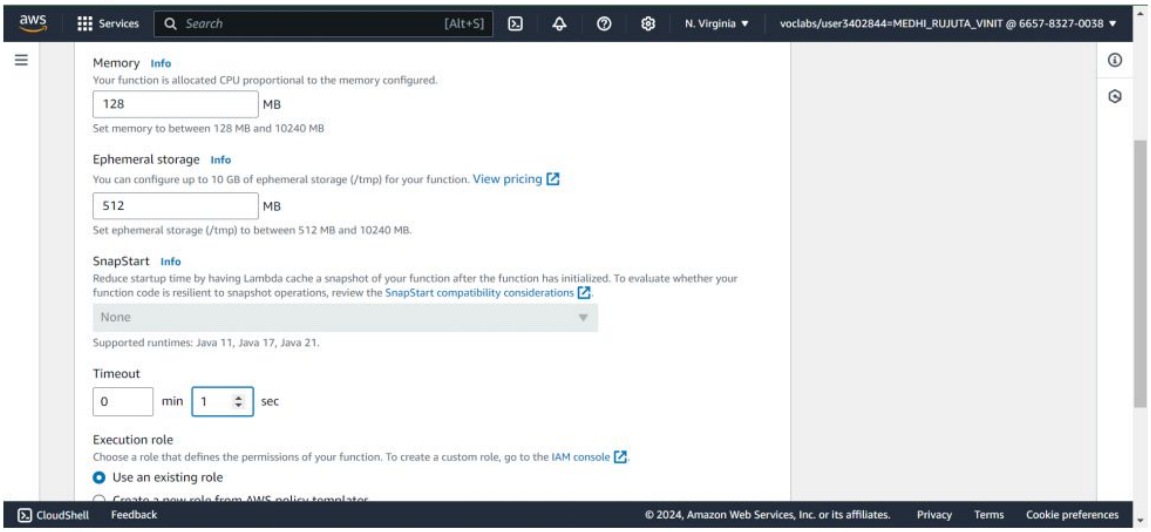
3. Lambda function is created.



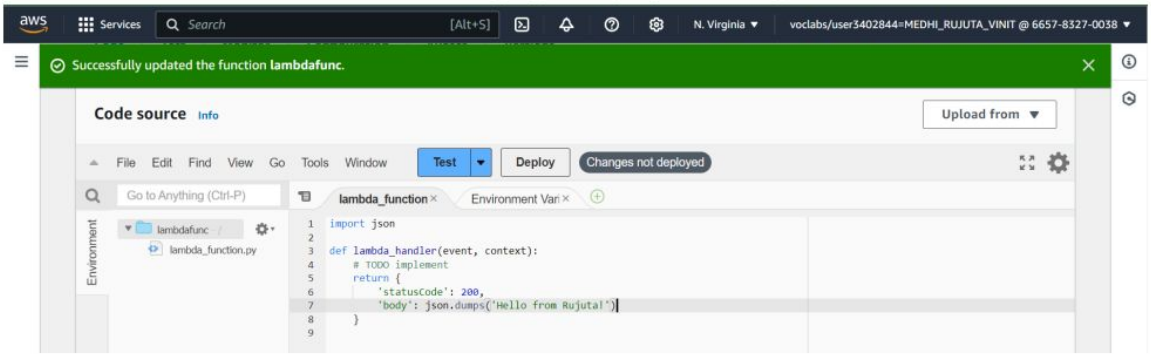
4. This is the source code



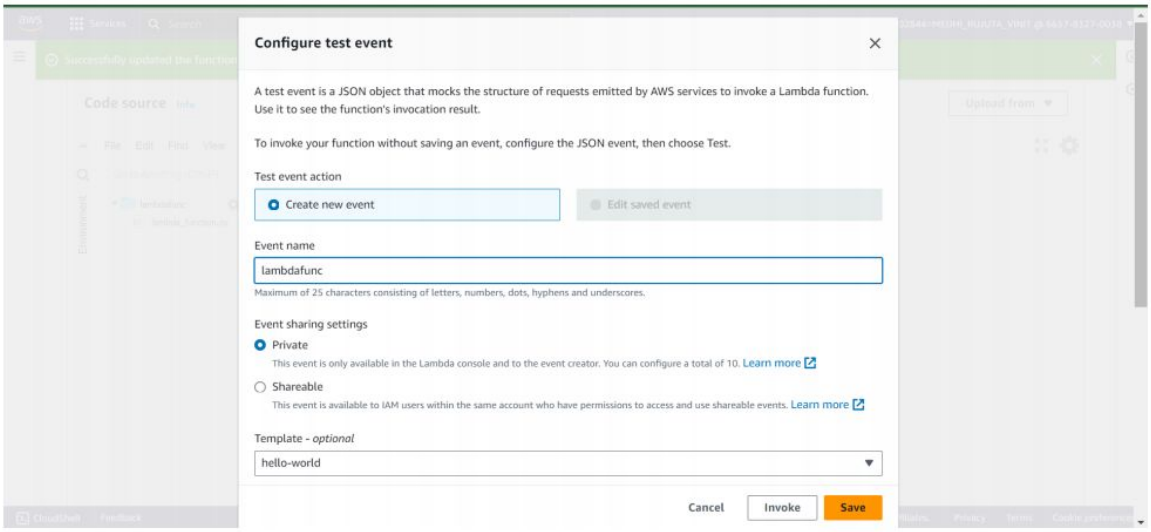
5. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.



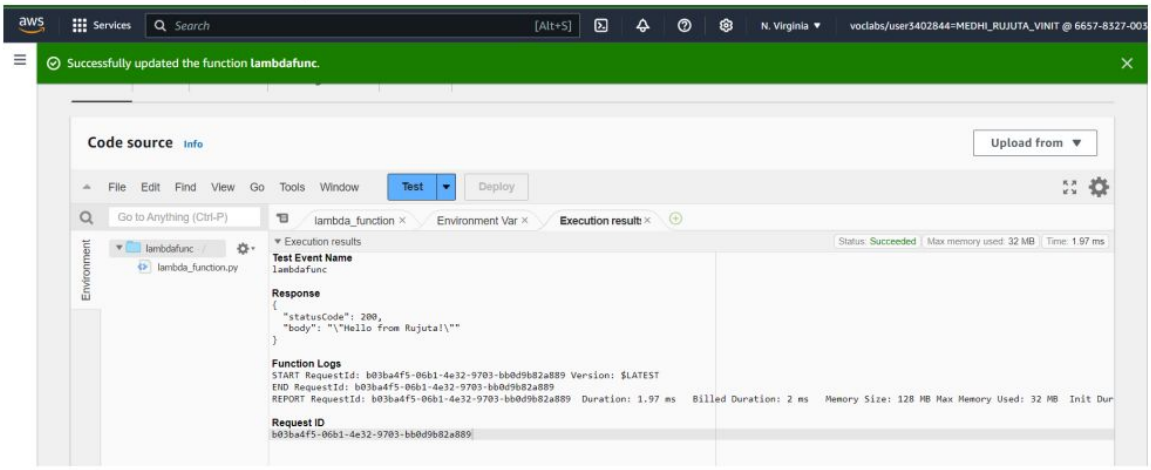
6. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed.
Press Ctrl + S to save the file and click Deploy to deploy the changes..



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



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



Rujuta Medhi D15A-28

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.
AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. It automatically scales your application by running code in response to events. One of the most common use cases for Lambda is integrating it with Amazon S3, a scalable object storage service.
When an object is added to or modified in an S3 bucket, you can configure an S3 event notification to trigger a Lambda function. This event-driven architecture simplifies processing data in real time and reduces the need for manual intervention.

Key Benefits

1. Serverless Architecture:
○ No need to manage servers or infrastructure. You focus solely on writing code.
○ AWS handles scaling automatically based on the number of events.

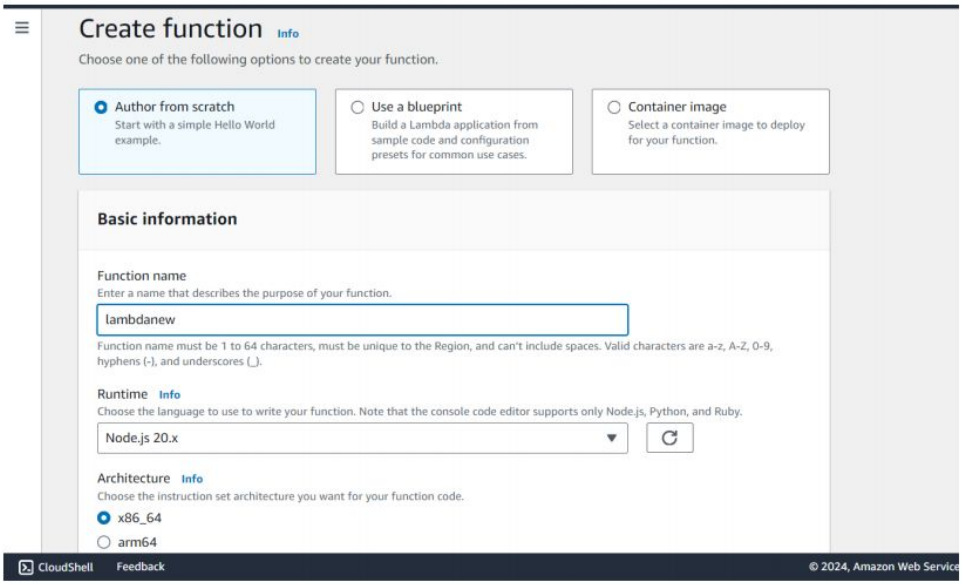
2. Cost Efficiency:
○ Pay only for the compute time you consume. You are charged based on the number of requests and the duration of execution.

3. Real-time Processing:
○ Automatically process files as they are uploaded, allowing for instant reactions to events (e.g., generating thumbnails, analyzing data).

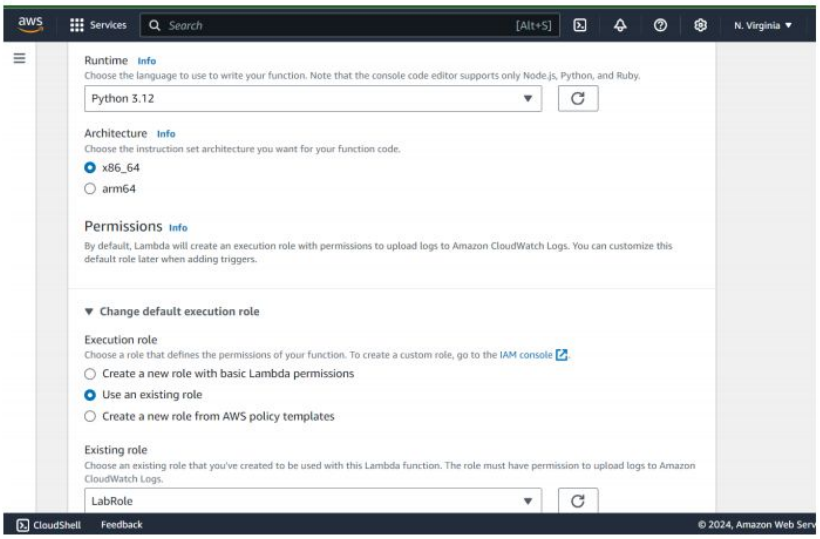
4. Ease of Integration:
○ Lambda integrates easily with other AWS services, enabling seamless workflows and data processing pipelines.

Implementation:

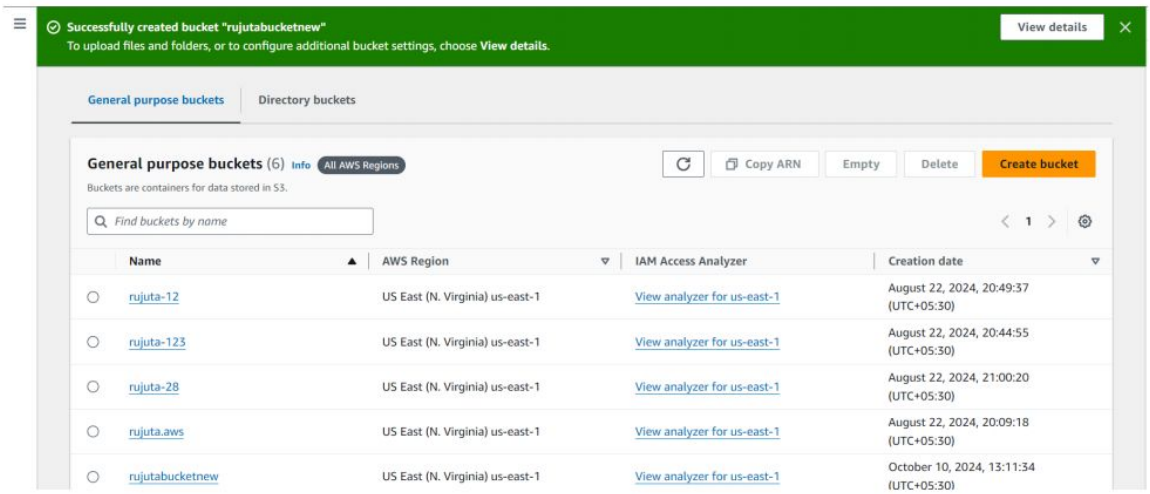
1. Created a lambda function by selecting runtime as python



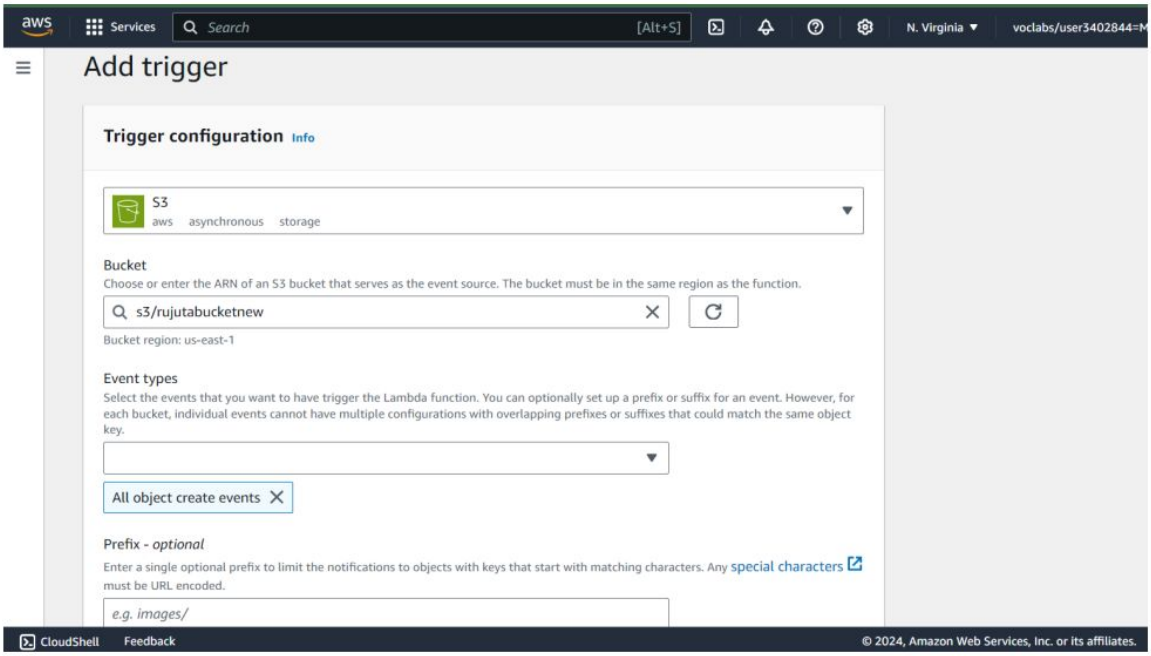
2. Select execution role as lab role

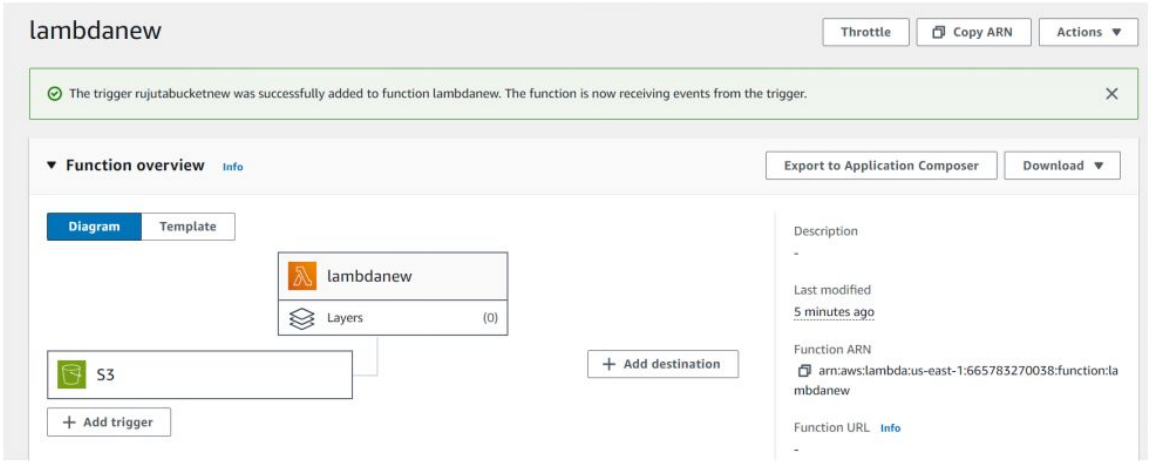


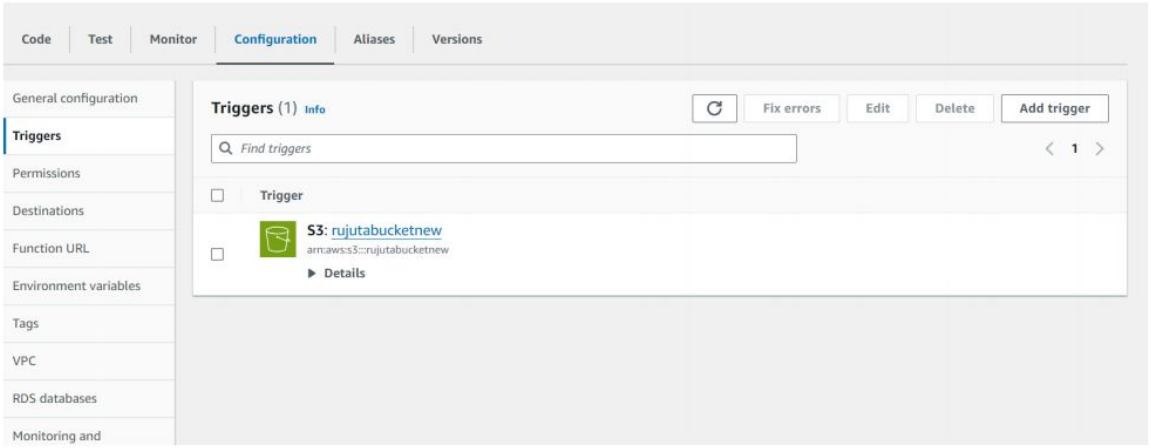
3. Create S3 bucket



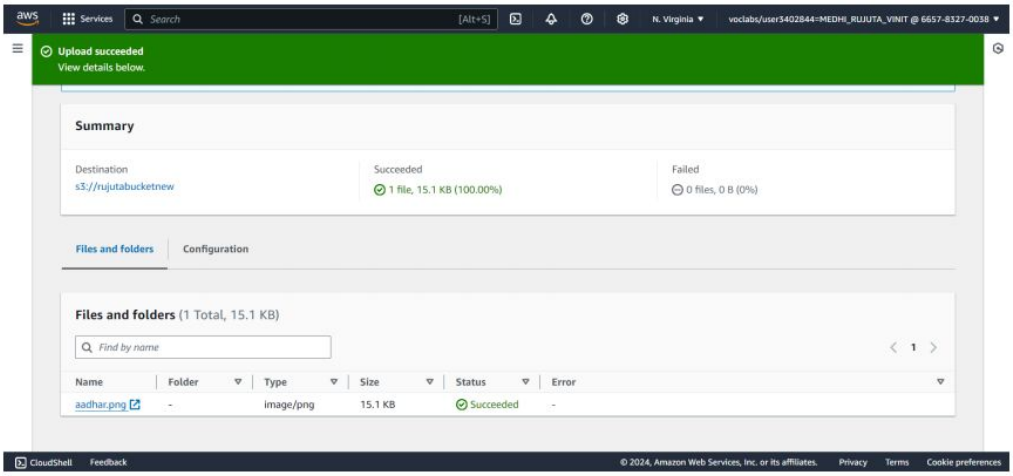
4. Select S3 in triggered configuration



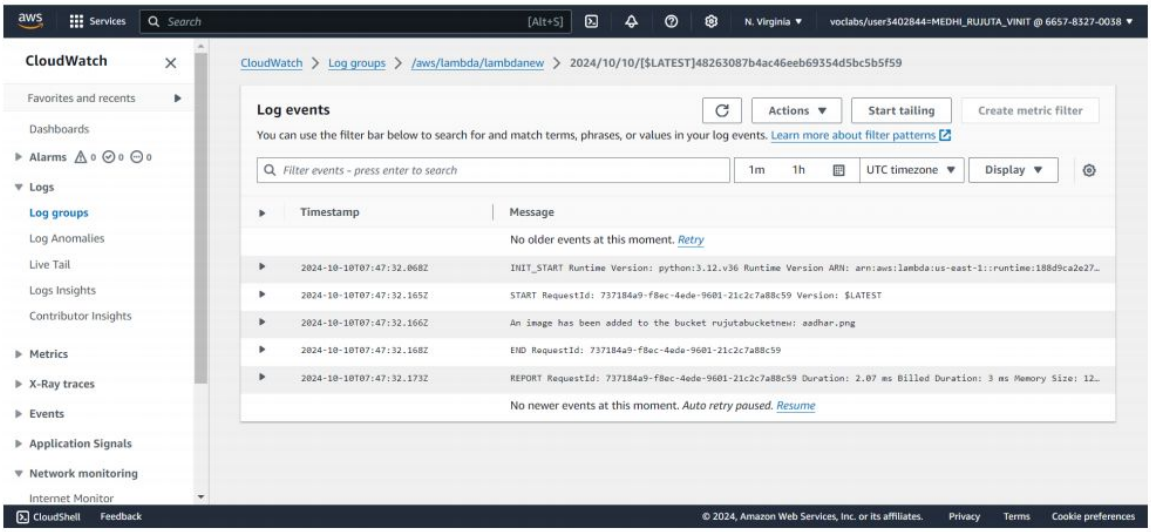




5. Upload image in S3 bucket



6. Check log events in cloud watch



Rujuta Medhi
D15A - 28

04/03

DATE

Adv DevOps.

Assignment - 1

Q1 Use S3 bucket and host video streaming

Steps to host video streaming on S3 bucket

Prerequisites : Register and configure a custom domain with route 53

Before we start, it is recommended to register and configure a custom domain (for example : example.com) with Route 53 so that you can configure your cloud front distribution to use a custom domain name later.

Without custom domain name, S3 video is publicly accessible and hosted through cloud-front at a URL that looks similar to following:

https://cloudfrontdistributions domain name / Path to an S3 video.

Step 1 : Create an S3 bucket

Sign In to AWS console and open Amazon S3

Choose Buckets > create Bucket

Enter Bucket name (For example : tutorial-bucket)

Choose Region

For Block Public Access setting for this bucket Keep default settings

For remaining settings, keep default settings

Click on create Bucket.

Step 2 : Upload a video to the S3

In the Bucket list, choose name of bucket that we created in step 1 to upload your file to

On the object tab for your Bucket, choose Upload
On the upload page, under Files and Folders choose add files
choose file to upload and then choose open
choose Upload.

Step 3: Create a CloudFront origin access identity

From AWS console, now open CloudFront service
Under security section, choose origin access. Under identities tab, choose create origin access identity and enter a name. click on create

Step 4 :- Create a CloudFront distribution

a) create a CloudFront distribution
choose Distributions > create Distribution
In origin section, origin domain slect domain name which starts with name of S3 bucket created in step 1.
For origin access identity choose origin access identity created in step 3.
Under Bucket Policy, choose Yes. Update Bucket policy. In Default cache behaviour > Viewer Protocol policy.
choose Redirect HTTP or HTTPS.
Keep the remaining setting set to default.
click on create distribution.

b) Review the bucket Policy

Step 5: Access the video through the cloud-Front distribution.

go to Distribution
Find Distribution by matching S3 origin name and copy name, open new tab & paste copied domain name. Return previous tab choose video object created in step 2 copy key from object.
Step 6: Configure your cloud Front distribution to use your custom Domain name
a) Request SSL certificate
b) Create CNS record to route traffic from alternate domain name to your cloud Front distribution.
Step 7: Access the S3 video through cloud Front distribution with custom domain name
Distribution > Find Distribution > copy alternate domain name.
Step 8: View data about request received
Hence, video streaming is hosted successfully.

Q2. Discuss BMW and Hotstar case study using AWS
→ BMW group, headquatered in Munich, Germany is a global manufacturer of premium automobils and motorcycle
The company needed to more easily scale its data to support the growing demands of internal & external stakeholders. As data wasnt easily accessible and spread across, BMW's innovation was slowed down by its own IT infrastructure. Hence, BMW needed to develop solution agile enough to support data need & allow company to move quickly to address customer demand.

The BMW group rearchitectured its on-premises data to AWS cloud - creating cloud Data Hub that integrates anonymised data from vehicle sensor and other sources. Using AWS services like S3, Athena, Kinesis Firehose, Glue, BMW streamlined data management & enabled scalable agile operations for data engineers. This setup allowed teams to maintain their own DevOps process.

The company uses this data to monitor vehicle health indications such as check controls errors to identify potential issues across vehicle lines. This enables BMW groups to leverage fleet data ingested, collected and refined from CDH to better resolve issue, even before they impact customers

Hotstar is an Indian subscription video on-demand streaming service owned and operated by star India, a subsidiary of the Walt Disney Company India. In 2019, during ICC World Cup Semi-Final between India & New Zealand, Hotstar set a new record of 25.3 million viewers. On the game day, the first spike witnessed was from 1.5M to 15M, as India started batting. Then, when Dhoni got out there was viewer drop to < 1M viewers

Hotstar does not use traditional auto scaling from AWS because of insufficient capacity error & step size autoscaling group. They build their own scaling strategy.

At the backend side, Hotstar uses Amazon Route 53 and Amazon Cloud Front services for video streaming.

Q3 Why Kubernetes and advantage and disadvantage of Kubernetes

→ Kubernetes is an open-source platform for automating the deployment, scaling and management of containerized applications.

Advantages :-

1. Scalability - Automatically adjust resources based on demand
2. Portability - consistent performance across cloud and on-premises environments.
3. High Availability - Self-heals and ensure uptime.
4. Load Balancing - Distributes traffic effectively.

Disadvantages :-

1. complexity - Steep learning curve and setup time
2. Resource Intensive - Requires significant computing resources.
3. Operational Overhead - Needs continuous management and monitoring.
4. Networking challenges :- complicated configurations can be tricky to troubleshoot.
5. Ecosystem Fragmentation - Numerous tools can complicate stack choices.

How Adidas Uses Kubernetes :-

1. Microservices - Manages independent services to speed up development.
2. Scalability - Automatically scales during high-traffic events.
3. Continuous deployment - Implements CI/CD for rapid feature releases.
4. Resource Optimization - Efficiently manages cloud resources to reduce costs.
5. Cloud-Native Strategy - Enhances agility and flexibility in operations.

Q4. What are Nagios and explain how Nagios are used in E-Services?

→ Nagios is an open-source monitoring system designed to keep track of network services, host resources and server performance. It provides alerts on system failures and issues.

How Nagios is Used in E-Services -

1. Infrastructure Monitoring - Nagios monitors servers, network devices and applications ensuring that critical components of e-services are running smoothly.

(2) Service Availability - Nagios checks availability of essential services and alerts administrators if any services go down.

(3) Performance Metrics :- By gathering performance data, Nagios helps in identifying trends and potential bottlenecks in e-services.

(4) Alerting and Notifications - Nagios sends alerts via email, SMS or other methods when issues are detected.

(5) Customizable checks - Users can define custom checks based on their specific e-service needs, allowing for tailored monitoring.

(6) Reporting and Analytics - Nagios provides reporting features that help in analyzing performance over time which can be critical for auditing and improving service quality.

(7) Integration with Other Tools - Nagios can be integrated with various third-party applications and plugins, enhancing its functionality for managing e-services.

Rujita Medhi
D15A - 28

Assignment - 2.

05/05

Q1 Create a REST API with serverless framework

Creating REST API with serverless framework is an efficient way to deploy serverless application that can scale automatically without managing servers.

i) Serverless framework: A powerful tool that simplifies deployment of serverless applications across various cloud providers such as AWS, Azure and Google cloud.

ii) Serverless Architecture: This design model allows developers to build applications without carrying about underlying infrastructure, enabling focus on code and business logic.

iii) Rest API: Representational state transfer is architectural style for designing network applications.

Steps for creating REST API for server-less framework:-

1] Install serverless framework:
You can start by installing serverless framework CLI globally using npm.

2] Create a Node.js serverless project:
A directory is created for your project, where you initialise serverless service. This service will have all lambda functions, configurations and cloud resources.

3] Project structure :
The project scaffold creates essential files like handler.js

4] Create Rest API resource :
In the serverless.yml file you define function. to handle HTTP.

5] Deploy the service :
deploy using 'S/S deploy' command

6] Testing the API :
Once deployed, Test API using curl or Postman.

7] Add more functionalities :
Add functionalities like 'List all candidates'.

8] Monitoring and maintenance
After deployment serverless framework provide service information like deployed endpoints.

Q2] Case Study for Sonarqube
- Create your own profile in sonarqube for testing project quality.
- Use sonarcloud to analyze your GitHub code
- Install sonarlint in your Java Intelliy IDE. or
- Analyze python project
- Analyze node js project with sonarqube

→. Create the sonarqube profile for testing project quality.
- Open intelliJ setting, find Tools > Sonarlint.
entry and select + to open connection wizard
- Enter name for this connection, select sonarcloud or sonarqube
- choose authentication method:
a) Generate token on sonarqube or sonarcloud
b) Username + Password. This can be used on sonarqube connection only
- For sonarcloud only select organization that you want to connect
- Sonarqube and sonarcloud can push notifications to developers.
- Validate the connection testing by selecting finishing at the end of the wizard.
- save the connection in global setting by clicking OK.

Bind Python Project to Sonarqube
- Seled Sonarlint > Bind Projed to Sonarqube.
- choose the correct project from sonarqube
Analyze the project (Python Project)
- Trigger an analysis by going to code > Analyze code

Analyze Node.js project
Make sure your Node.js project is properly configured with sonar-qube properties file or equivalent for the analysis to run.

Q3 At large organization your centralized operations team may get many repetitive infrastructure requests. You can use Terraform modules to build "self-serve" infrastructure model that lets product teams manage their own infrastructure independently. Terraform cloud can also integrate with ticketing system like service now to automatically generate new infrastructure requests.

Self-serve infrastructure model with Terraform modules:
At a large organization, implementation a self-serve infrastructure model using Terraform can significantly streamline process of managing infrastructure across different teams. This approach allows users to manage their own infrastructure independently while adhering to organizational standards.
Key aspects are :-

a) Standardization through Terraform Modules by creating and utilizing Terraform modules, organizations can codify their infrastructure deployment and management standards.

b) Efficient Deployment: Product teams can leverage these standardized modules to quickly deploy services without needing to reinvent the wheel or wait for the centralized operation team to handle every request.

c) Compliance: By using predefined modules, teams ensure that their deployment comply with organizations

established practices and security guidelines.

d] Automation :
The use of Terraform modules promotes automation, reducing manual interventions and potential human errors in infrastructure management.

e] Version control :
With modules stored in version control system like Git, teams can track changes, collaborate on improvements and maintain a history of infrastructure configuration.

Integration with Ticketing Systems :
Terraform cloud offer integration capabilities that further enhance the self-serve model:

a) Automate infrastructure requests :
Terraform cloud can integrate with ticketing systems like servicenow to automatically generate new infrastructure requests.

b) Centralised Management :
By centralizing infrastructure management through Terraform cloud organisation can maintain better control over who can request and approve infrastructure changes.

c) Governance :
The integration with ticketing systems allows for better governance of infrastructure requests.

Collaborative Infrastructure Management

a) Team Based performance permissions
b) State and run history.
c) Sensitive Information protection.
d) Module Registry.

By implementing these features and promises large organisations can effectively leverage Terraform to build a robust, scalable and compliant infrastructure management system.