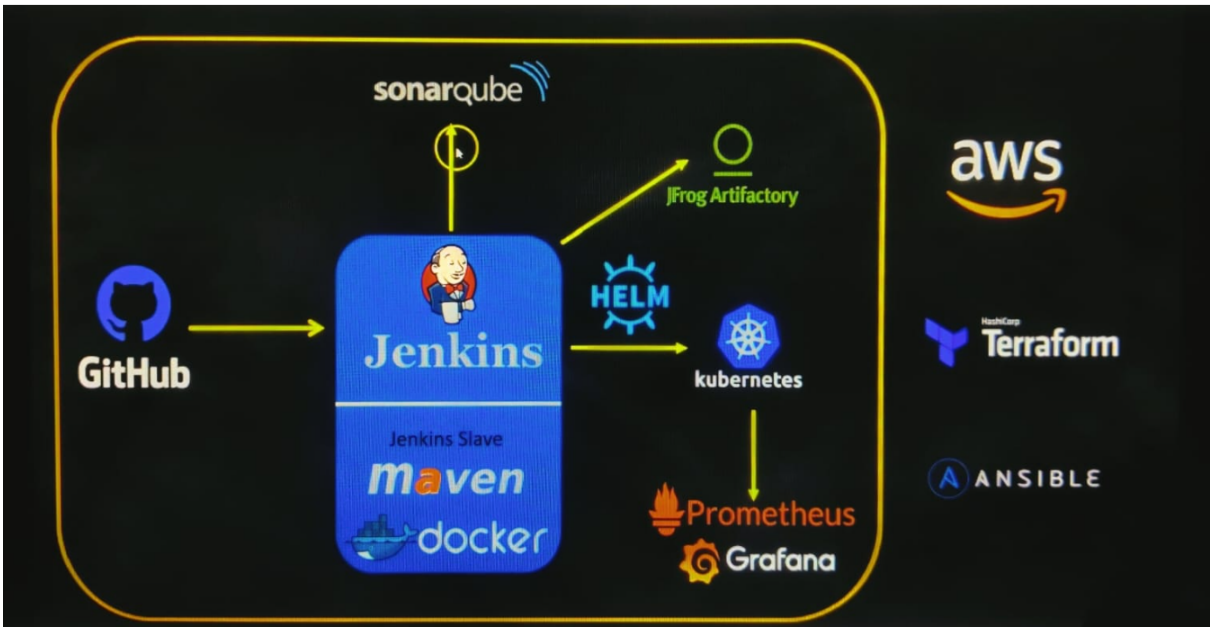
CICD Project Workshop



In this course we are going to create a CI CD pipeline by using various tools.

▼ **Tech stacks used in this project**

GitHub

Jenkins

Terraform

Ansible

Maven

SonarQube

Jfrog

Docker

Kubernetes

Helm Charts

Prometheus

Grafana

▼ **Steps to perform during this CICD pipeline project**

- Set up Terraform
- Provision Jenkins master, build Node and Ansible using Terraform.
- Set up Ansible server.
- Configure Jenkins master and build node using Ansible.
- Create a Jenkins pipeline job
- Create a Jenkins file from scratch.
- Create Multi-branch pipeline

- Enable webhook on GitHub.
- Configuring Sonar Cube and Sonar Scanner.
- Execute Sonar Cube analysis.
- Define rules and gates on Sonar Cube.
- Sonar callback rules.
- Jfrog Artifactory Setup.
- Create a Docker file
- Store Docker Images on Jfrog Artifactory.
- Provisioned Kubernetes cluster using Terraform.
- Create Kubernetes Objects.
- Deploying the Kubernetes objects using Helm.
- Set up Prometheus and Grafana using Helm Charts.
- Monitor Kubernetes Cluster using Prometheus.

▼ **Pre-requisites**

- Install below tools on local system
- Visual Studio
- Git
- Terraform
- AWS CLI
- Mobaxterm

Terraform

````
## Prepare Terraform Environment on Windows
As part of this, we should setup
1. Terraform
2. VS Code
3. AWSCLI

### Install Terraform
1. Download terraform the latest version from [here](https://developer.hashicorp.com/terraform/downloads)
2. Setup environment variable
click on start --> search "edit the environment variables" and click on it
Under the advanced tab, chose "Environment variables" --> under the system variables select path variable
and add terraform location in the path variable. system variables --> select path
add new --> terraform_Path
in my system, this path location is C:\Program Files\terraform_1.3.7

1. Run the below command to validate terraform version
   ```sh
   terraform -version
   ```
   the output should be something like below
   ```sh
   Terraform v1.3.7
   on windows_386
   ```

 ### Install Visual Studio code

  Download vs code latest version from [here](https://code.visualstudio.com/download) and install it.

### AWSCLI installation


Download AWSCLI latest version from [here](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html) and insta
````

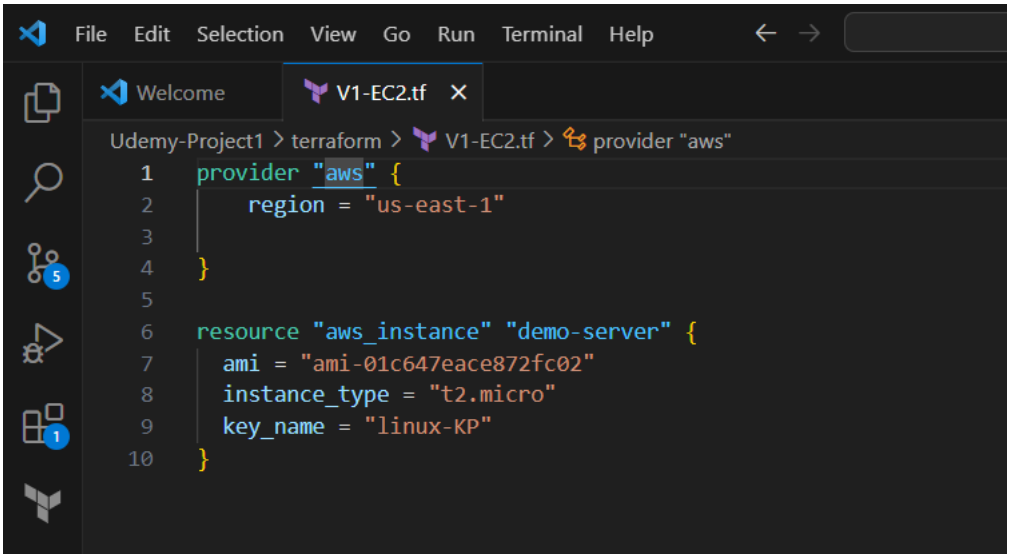
```
or you can run the below command in powershell or the command prompt
```

Terraform-code

1. Create IAM user
2. Login to aws cli

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Udemy-Project1/terraform
$ aws configure
AWS Access Key ID [****************FU46]: AKIAQXVMWBLQZR2E3OHG
AWS Secret Access Key [****************iq1x]: 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
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

3. Write the First Terraform code



4. **Run terraform command > terrafrom init > terraform validate > terraform plan**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Udemy-Project1/terraform
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.16.0

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.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Udemy-Project1/terraform
$ terraform validate
Success! The configuration is valid.

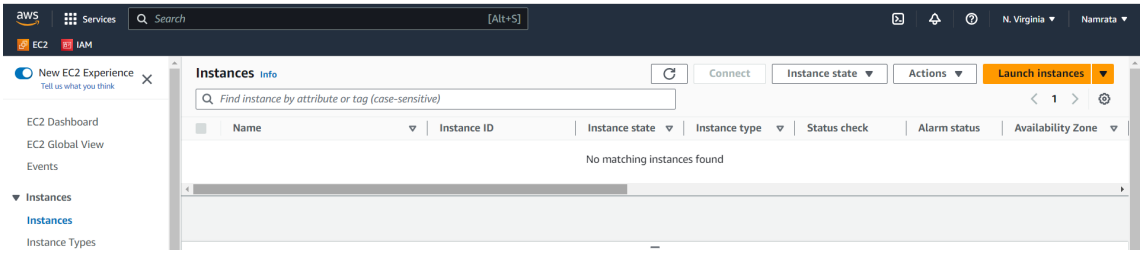

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Udemy-Project1/terraform
$ terraform plan

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

Terraform will perform the following actions:

  # aws_instance.demo-server will be created
  + resource "aws_instance" "demo-server" {
      + ami                                  = "ami-01c647eace872fc02"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
```

5. **Before running the 'terraform apply' command check AWS console**



6. **Run terraform apply**

```
        }
    }

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

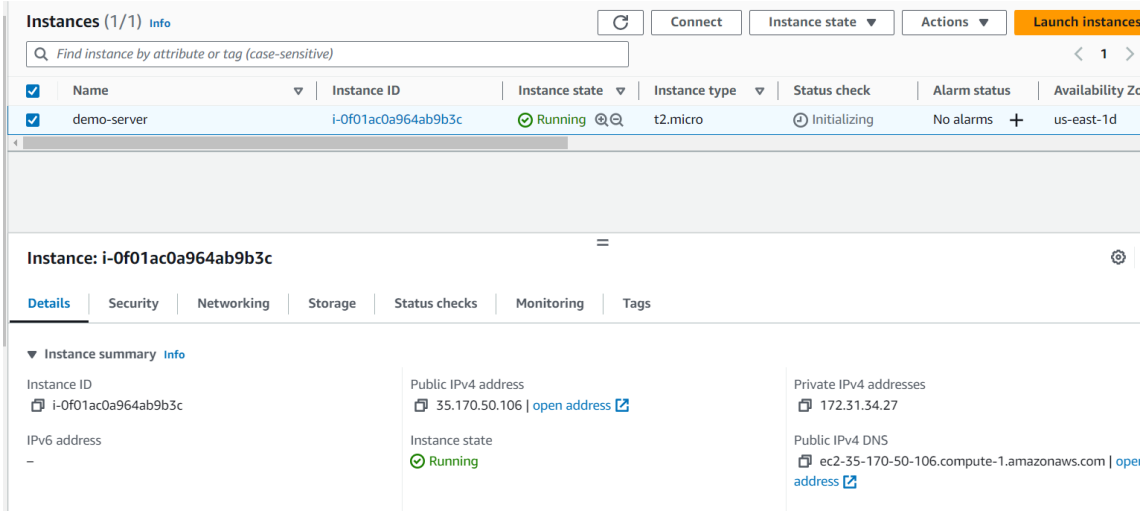
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.demo-server: Creating...
aws_instance.demo-server: Still creating... [10s elapsed]
aws_instance.demo-server: Still creating... [20s elapsed]
aws_instance.demo-server: Creation complete after 26s [id=i-0f01ac0a964ab9b3c]

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

7. **EC2 instance created**



8. **To destroy created infrastructure run 'terraform destroy'**

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

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

  Enter a value: yes

aws_instance.demo-server: Destroying... [id=i-0f01ac0a964ab9b3c]
aws_instance.demo-server: Still destroying... [id=i-0f01ac0a964ab9b3c, 10s elapsed]
aws_instance.demo-server: Still destroying... [id=i-0f01ac0a964ab9b3c, 20s elapsed]
aws_instance.demo-server: Still destroying... [id=i-0f01ac0a964ab9b3c, 30s elapsed]
aws_instance.demo-server: Still destroying... [id=i-0f01ac0a964ab9b3c, 40s elapsed]
aws_instance.demo-server: Destruction complete after 42s

Destroy complete! Resources: 1 destroyed.
```

==========

Terraform with Ansible

This document discusses using Terraform and Ansible to provision infrastructure and configure Jenkins master and slave servers.
The document includes Terraform code for setting up EC2 instances, a VPC, and other resources. It also mentions the process of converting one instance to an AWS instance and using Ansible playbooks to configure the Jenkins servers.
Ansible server is going to manage two different systems and through Ansible playbooks we are going to convert one server as a Jenkins master and another one as a Jenkins slave.
Screenshots of Terraform commands and the created instances are provided.

I have written a Terraform manifest file to create three EC2 instances, by using 'for each block'.

I need to convert one of these instances as a AWS instance. Then this

Features

- Setup 3 EC2 instances through Terraform
- Provision Jenkins-master, Jenkins-slave and Ansible
- Setup Ansible Server
- Configure Jenkins master using Ansible

We need to run the same script multiple times to create multiple instances. Rather than this, I am going to use one more parameter called 'for each'.

Write TF script to provision infrastructure V2-EC2-with-vpc-for-each.

▼ **Terraform code-with-VPC-for-each**

```
provider "aws" {
region = "us-east-1"

}

resource "aws_instance" "demo-server" {
ami = "ami-053b0d53c279acc90"
instance_type = "t2.micro"
key_name = "linux-KP"
//security_groups = ["demo-sg"]
vpc_security_group_ids = [aws_security_group.demo-sg.id]
subnet_id = aws_subnet.Nam-public-subnet-01.id

for_each = toset(["jenkins-master", "jenikns-slave", "ansible"])
tags = {
Name = "${each.key}"
}
}

resource "aws_security_group" "demo-sg" {
name        = "demo-sg"
description = "SSH Access"
vpc_id = aws_vpc.Nam-vpc.id

ingress {
description      = "Shh access"
from_port        = 22
to_port          = 22
protocol         = "tcp"
cidr_blocks      = ["0.0.0.0/0"]
}

egress {
from_port        = 0
to_port          = 0
protocol         = "-1"
cidr_blocks      = ["0.0.0.0/0"]
ipv6_cidr_blocks = ["::/0"]
}

tags = {
Name = "ssh-port"

}
}

resource "aws_vpc" "Nam-vpc" {
cidr_block = "10.1.0.0/16"
tags = {
Name = "Nam-vpc"
}

}

resource "aws_subnet" "Nam-public-subnet-01" {
vpc_id = aws_vpc.Nam-vpc.id
cidr_block = "10.1.1.0/24"
map_public_ip_on_launch = "true"
availability_zone = "us-east-1a"
tags = {
Name = "Nam-public-subent-01"
}
}
```

```
resource "aws_subnet" "Nam-public-subnet-02" {
vpc_id = aws_vpc.Nam-vpc.id
cidr_block = "10.1.2.0/24"
map_public_ip_on_launch = "true"
availability_zone = "us-east-1b"
tags = {
Name = "Nam-public-subent-02"
}
}

resource "aws_internet_gateway" "Nam-igw" {
vpc_id = aws_vpc.Nam-vpc.id
tags = {
Name = "Nam-igw"
}
}

resource "aws_route_table" "Nam-public-rt" {
vpc_id = aws_vpc.Nam-vpc.id
route {
cidr_block = "0.0.0.0/0"
gateway_id = aws_internet_gateway.Nam-igw.id
}
}

resource "aws_route_table_association" "Nam-rta-public-subnet-01" {
subnet_id = aws_subnet.Nam-public-subnet-01.id
route_table_id = aws_route_table.Nam-public-rt.id

}

resource "aws_route_table_association" "Nam-rta-public-subnet-02" {
subnet_id = aws_subnet.Nam-public-subnet-02.id
route_table_id = aws_route_table.Nam-public-rt.id

}
```

`terraform init`

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ ls
V4-EC2-With_VPC_for_each.tf  terraform.tfstate  terraform.tfstate.backup

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.17.0

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.
```

`terraform validate`

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ terraform validate
Success! The configuration is valid.
```

`terraform plan`

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ terraform plan

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

Terraform will perform the following actions:

  # aws_instance.demo-server["ansible"] will be created
  + resource "aws_instance" "demo-server" {
      + ami                                  = "ami-053b0d53c279acc90"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "linux-KP"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
```

`terraform apply`

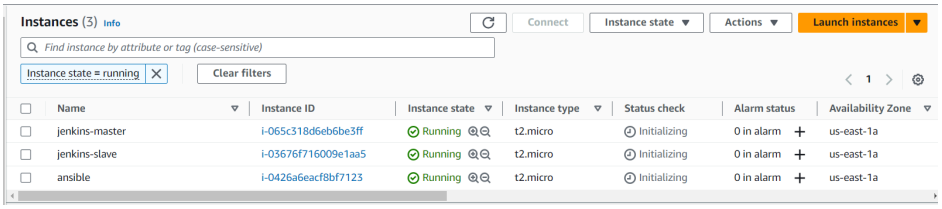
```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ terraform apply --auto-approve

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

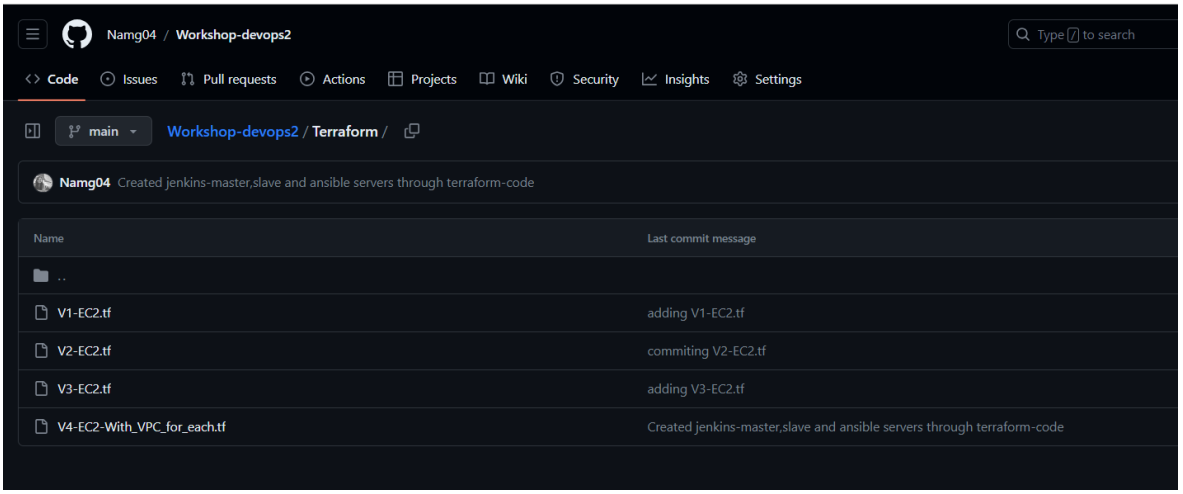
Terraform will perform the following actions:

  # aws_instance.demo-server["ansible"] will be created
  + resource "aws_instance" "demo-server" {
      + ami                                  = "ami-053b0d53c279acc90"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "linux-KP"
      + monitoring                           = (known after apply)
```

`Three instances created`

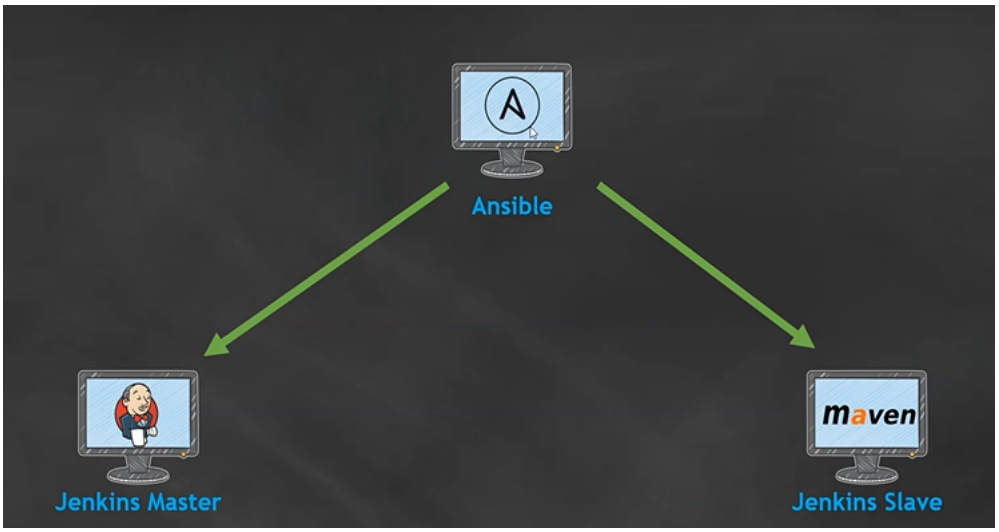


All code has committed into GitHub repo



We have seen how to set up three different instances by using Terraform. Now we need to convert one of these instances as a AWS instance and then this Ansible server is going to manage two different systems and through Ansible playbooks we

are going to convert one server as a Jenkins master and another one as a Jenkins slave. So, in the Jenkins slave we are going to install Maven.



==========

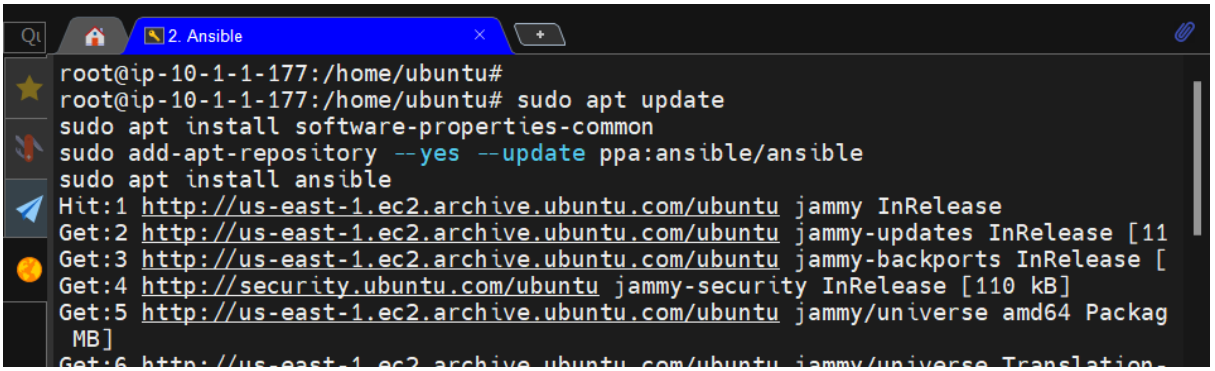
Ansible setup

This document provides instructions for setting up Ansible.
It covers installing Ansible on Ubuntu 22.04, adding Jenkins master and slave as hosts, copying .pem files to the Ansible server, testing the connection, and configuring Jenkins-master and Jenkins-slave in the hosts file.

1. **Install Ansible**

- Take the public IP of ansible server and login to it through Mobaxterm
- Install ansibe on Ubuntu 22.04

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```




```
ansible –version
```

```
root@ip-10-1-1-177:/home/ubuntu# ansible --version
ansible [core 2.15.4]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share
/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/co
llections
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] (/usr/bin/p
ython3)
  jinja version = 3.0.3
  libyaml = True
root@ip-10-1-1-177:/home/ubuntu#
```

- **Add Jenkins master and slave as hosts Add Jenkins master and slave private IPs in the inventory file in this case, we are using /opt is our working directory for Ansible.**

Jenkins-master

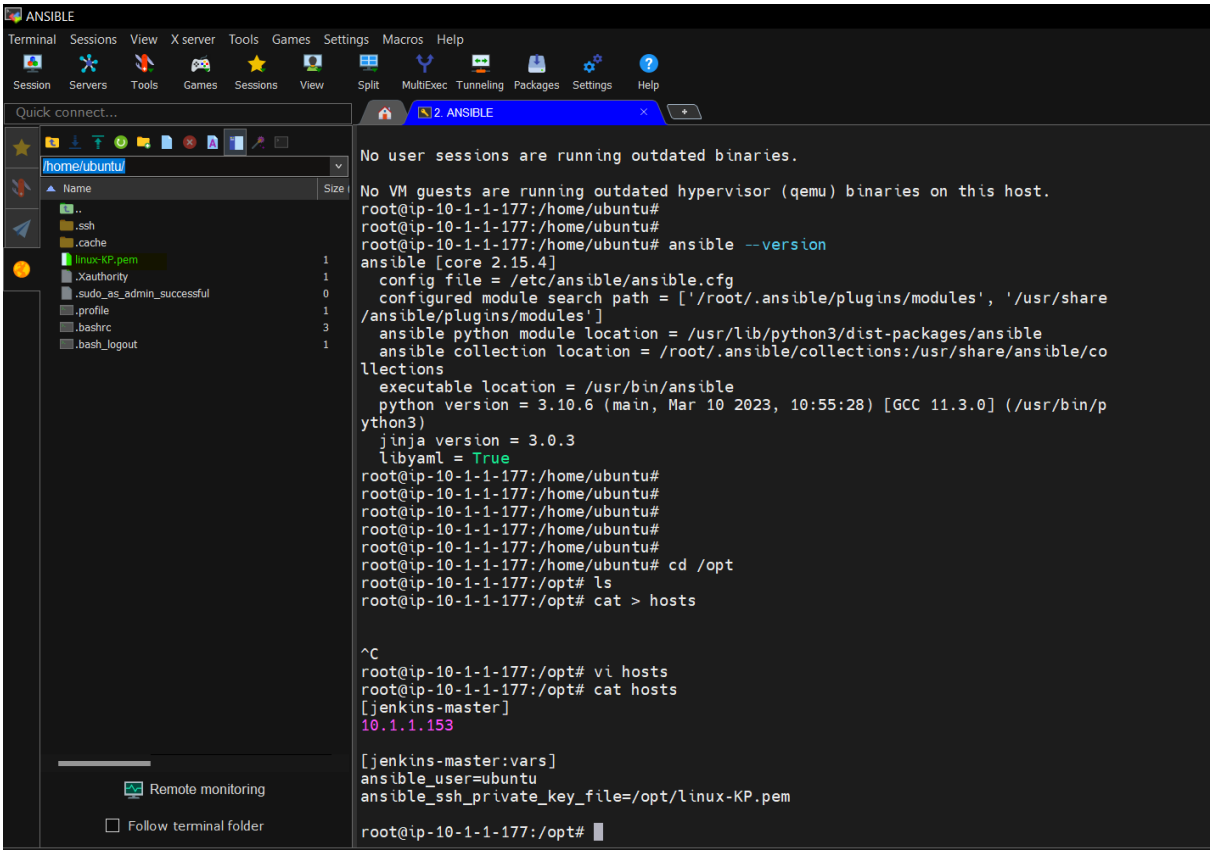
```
root@ip-10-1-1-177:/home/ubuntu# cd /opt
root@ip-10-1-1-177:/opt# ls
root@ip-10-1-1-177:/opt# cat > hosts


^C
root@ip-10-1-1-177:/opt# vi hosts
root@ip-10-1-1-177:/opt# cat hosts
[jenkins-master]
10.1.1.153

[jenkins-master:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=/opt/linux-KP.pem

root@ip-10-1-1-177:/opt#
```

2. **Copy .pem file into ANSIBLE server**



Move .pem file at /opt location

```
root@ip-10-1-1-177:/opt# vi hosts
root@ip-10-1-1-177:/opt# cat hosts
[jenkins-master]
10.1.1.153

[jenkins-master:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=/opt/linux-KP.pem

root@ip-10-1-1-177:/opt# cd /home/ubuntu
root@ip-10-1-1-177:/home/ubuntu# ls
linux-KP.pem
root@ip-10-1-1-177:/home/ubuntu# mv linux-KP.pem /opt
root@ip-10-1-1-177:/home/ubuntu# cd /opt
root@ip-10-1-1-177:/opt# ls
hosts  linux-KP.pem
root@ip-10-1-1-177:/opt#
```

Provide only read permission to .pem file

```
root@ip-10-1-1-177:/opt# chmod 400 linux-KP.pem
root@ip-10-1-1-177:/opt# ls -ltr
total 8
-rw-r--r-- 1 root   root    119 Sep 25 14:38 hosts
-r-------- 1 ubuntu ubuntu 1678 Sep 25 14:44 linux-KP.pem
root@ip-10-1-1-177:/opt#
```

3. **Test the connection**

`ansible -i hosts all -m ping`

```
root@ip-10-1-1-177:/opt# ansible all -i hosts -m ping
[WARNING]: Invalid characters were found in group names but not replaced, use
-vvvv to see details
The authenticity of host '10.1.1.153 (10.1.1.153)' can't be established.
ED25519 key fingerprint is SHA256:iuhS27R58XQNgEA9lFvonA+ivRcyv3C6Gl9ABr4yteA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
10.1.1.153 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Configure Jenkins-master and Jenkins-slave in hosts file

```
root@ip-10-1-1-177:/opt# cat hosts
[jenkins-master]
10.1.1.153

[jenkins-master:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=/opt/linux-KP.pem


[jenins-slave]
10.1.1.59

[jenkins-slave:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=/opt/linux-KP.pem

root@ip-10-1-1-177:/opt# vi hosts
root@ip-10-1-1-177:/opt# ansible all -i hosts -m ping
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
The authenticity of host '10.1.1.59 (10.1.1.59)' can't be established.
ED25519 key fingerprint is SHA256:smv40VAnFOVMY4VllWyNtcKMiLw6HbYNqe0uUcrY4zk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? 10.1.1.153 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
yes
10.1.1.59 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
root@ip-10-1-1-177:/opt#
```

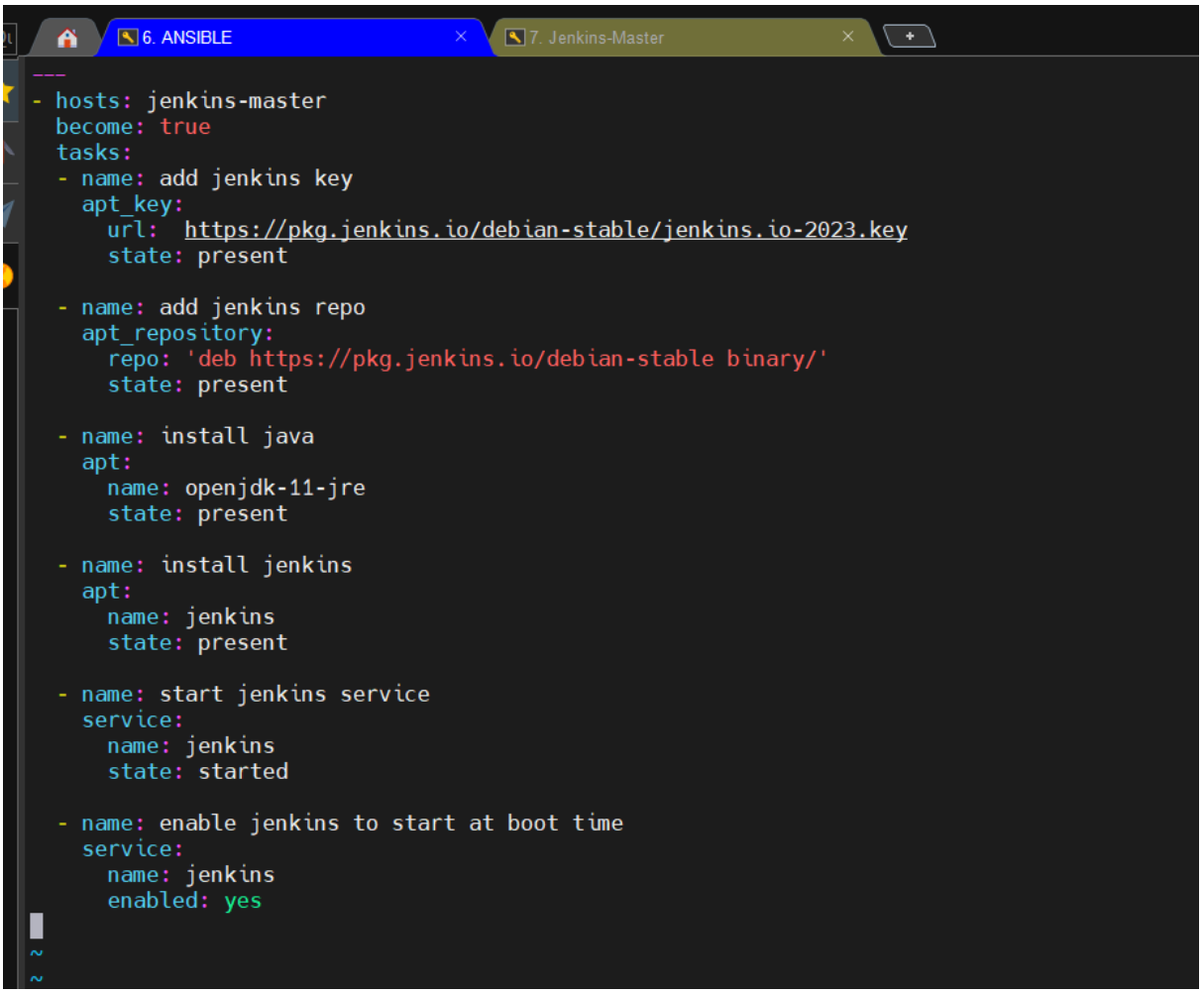
==========

Ansible playbook to install Jenkins on Jenkins-master server

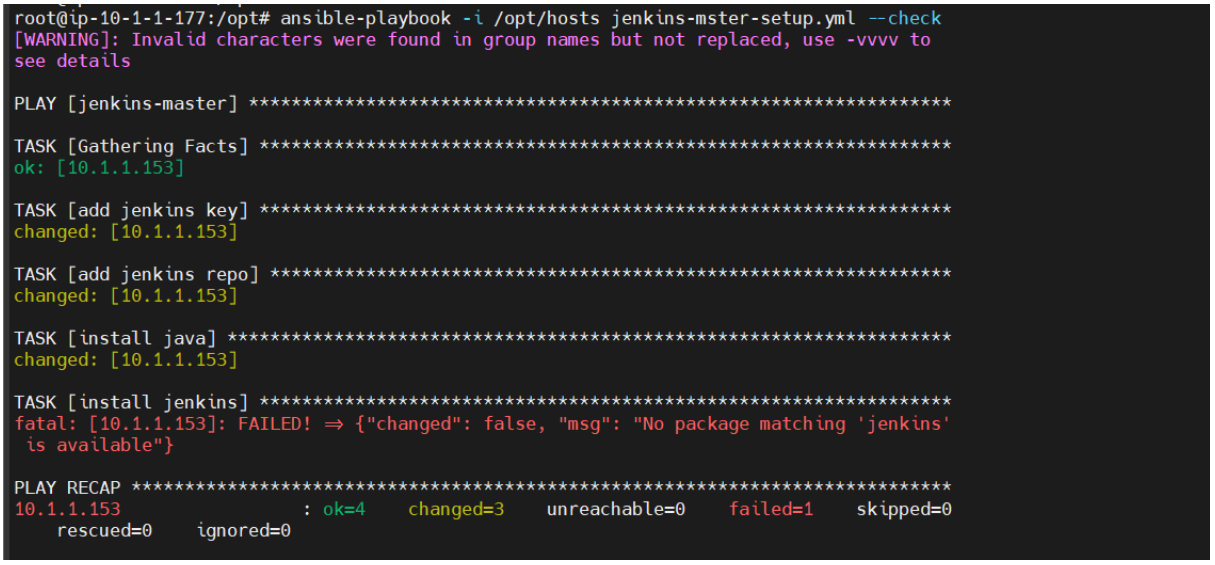
This document provides an Ansible playbook to install Jenkins on a Jenkins-master server. The playbook includes steps to add the Jenkins repo keys, add the repository, install dependencies, and install Jenkins.
It also includes instructions for a dry run and running the playbook, as well as checking the Java version and Jenkins status on the Jenkins-master server. The document concludes with instructions for opening port 8080 to access Jenkins and accessing Jenkins-master.

- Add the Jenkins repo keys to system
- Add repository to system
- Install dependencies
- Install Jenkins

Write ansible playbook to install Jenkins on Jenkins-master instance



dry run : ***ansible-playbook -i /opt/hosts jenkins-master-setup.yml --check***



Run playbook

```
root@ip-10-1-1-177:/opt# ansible-playbook -i /opt/hosts jenkins-mster-setup.yml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to
see details

PLAY [jenkins-master] ***********************************************************************

TASK [Gathering Facts] **********************************************************************
ok: [10.1.1.153]

TASK [add jenkins key] **********************************************************************
changed: [10.1.1.153]

TASK [add jenkins repo] *********************************************************************
changed: [10.1.1.153]

TASK [install java] *************************************************************************
changed: [10.1.1.153]

TASK [install jenkins] **********************************************************************
changed: [10.1.1.153]

TASK [start jenkins service] ****************************************************************
ok: [10.1.1.153]

TASK [enable jenkins to start at boot time] *************************************************
ok: [10.1.1.153]

PLAY RECAP **********************************************************************************
10.1.1.153                 : ok=7    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Checked java version and Jenkins status on Jenkins-master server

```
6. ANSIBLE   7. Jenkins-Master
root@ip-10-1-1-153:~# java --version
openjdk 11.0.20.1 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
root@ip-10-1-1-153:~#
root@ip-10-1-1-153:~#
root@ip-10-1-1-153:~# service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-09-26 11:18:30 UTC; 5min ago
   Main PID: 8671 (java)
      Tasks: 36 (limit: 1141)
     Memory: 308.4M
        CPU: 46.640s
     CGroup: /system.slice/jenkins.service
             └─8671 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Sep 26 11:17:55 ip-10-1-1-153 jenkins[8671]: d8735b4a6e21498496d56f71ac9f748d
Sep 26 11:17:55 ip-10-1-1-153 jenkins[8671]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 26 11:17:55 ip-10-1-1-153 jenkins[8671]: ************************************************************
Sep 26 11:17:55 ip-10-1-1-153 jenkins[8671]: ************************************************************
Sep 26 11:17:55 ip-10-1-1-153 jenkins[8671]: ************************************************************
Sep 26 11:18:30 ip-10-1-1-153 jenkins[8671]: 2023-09-26 11:18:30.369+0000 [id=29]        INFO        jenkins.InitReactorRunner$1#onAttained: Com
Sep 26 11:18:30 ip-10-1-1-153 jenkins[8671]: 2023-09-26 11:18:30.398+0000 [id=22]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins
Sep 26 11:18:30 ip-10-1-1-153 systemd[1]: Started Jenkins Continuous Integration Server.
Sep 26 11:18:30 ip-10-1-1-153 jenkins[8671]: 2023-09-26 11:18:30.513+0000 [id=44]        INFO        h.m.DownloadService$Downloadable#load: Obta
Sep 26 11:18:30 ip-10-1-1-153 jenkins[8671]: 2023-09-26 11:18:30.513+0000 [id=44]        INFO        hudson.util.Retrier#start: Performed the ac
lines 1-20/20 (END)
```

To access Jenkins port 8080 should be opened

-modified V4-EC2-With_VPC_for_each.tf

```
Terraform > V4-EC2-With_VPC_for_each.tf > resource "aws_security_group" "
18
19
20  resource "aws_security_group" "demo-sg" {
21    name        = "demo-sg"
22    description = "SSH Access"
23    vpc_id = aws_vpc.Nam-vpc.id
24
25    ingress {
26      description      = "SSH access"
27      from_port        = 22
28      to_port          = 22
29      protocol         = "tcp"
30      cidr_blocks      = ["0.0.0.0/0"]
31      }
32
33    ingress {
34      description      = "Jenkins-port"
35      from_port        = 8080
36      to_port          = 8080
37      protocol         = "tcp"
38      cidr_blocks      = ["0.0.0.0/0"]
39      }
40
41    egress {
42      from_port        = 0
43      to_port          = 0
44      protocol         = "-1"
45      cidr_blocks      = ["0.0.0.0/0"]
46      ipv6_cidr_blocks = ["::/0"]
47    }
48
49    tags = {
50      Name = "ssh-port"
51
```

Run terraform plan

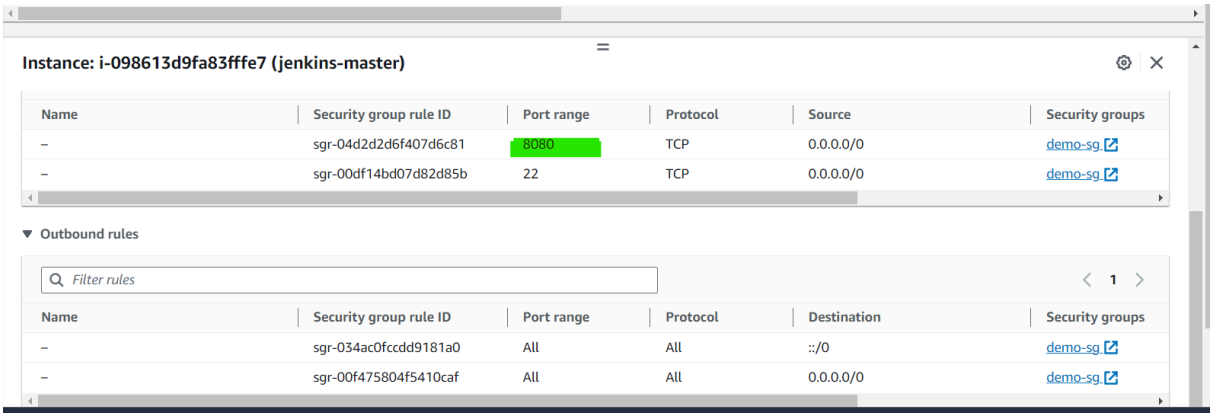
```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ ls
V4-EC2-With_VPC_for_each.tf  terraform.tfstate  terraform.tfstate.backup

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ terraform plan
aws_vpc.Nam-vpc: Refreshing state... [id=vpc-0c09026eb7725a827]
aws_internet_gateway.Nam-igw: Refreshing state... [id=igw-0540d289cd15e8b99]
aws_subnet.Nam-public-subnet-01: Refreshing state... [id=subnet-024dfff7c21f2a8c
3]
aws_subnet.Nam-public-subnet-02: Refreshing state... [id=subnet-0c0b0c9403ebbc5d
8]
aws_security_group.demo-sg: Refreshing state... [id=sg-031d62b69f2db6dd7]
    }

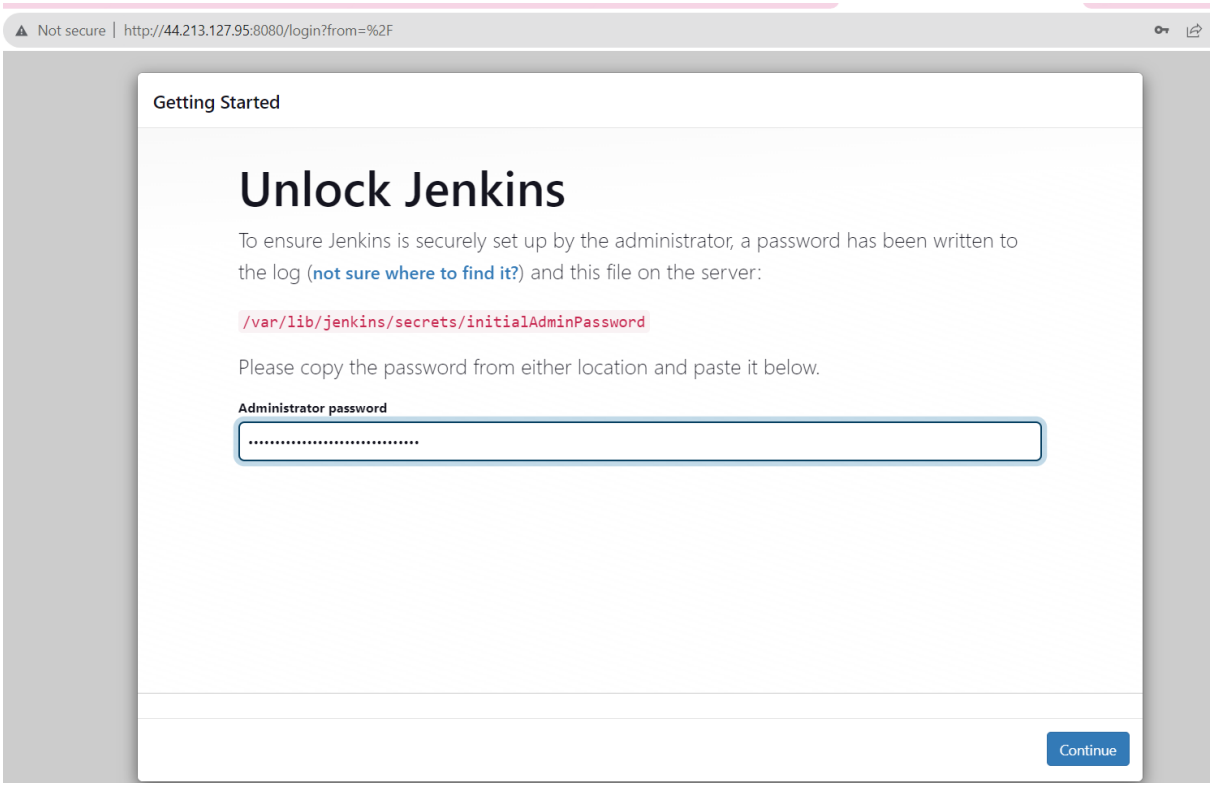
Plan: 0 to add, 1 to change, 0 to destroy.

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

Run terraform apply.
Port 8080 opened



Access Jenkins-master



==========

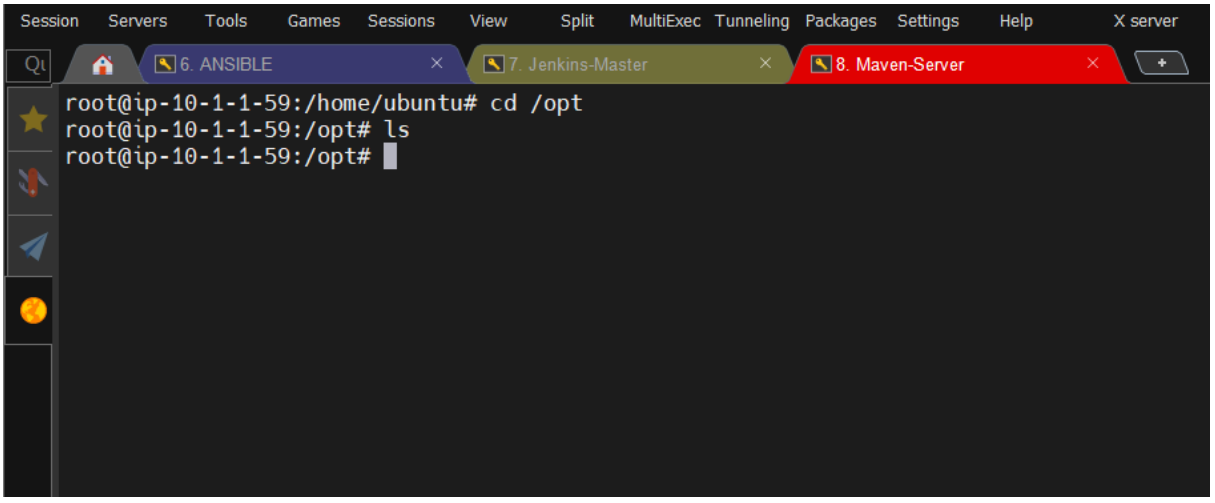
Ansible playbook to configure maven on Jenkins-slave server

This document provides an Ansible playbook to configure Maven on a Jenkins-slave server. The steps include updating the system, installing Java, downloading and extracting Maven packages, and adding the path to the bash_profile.

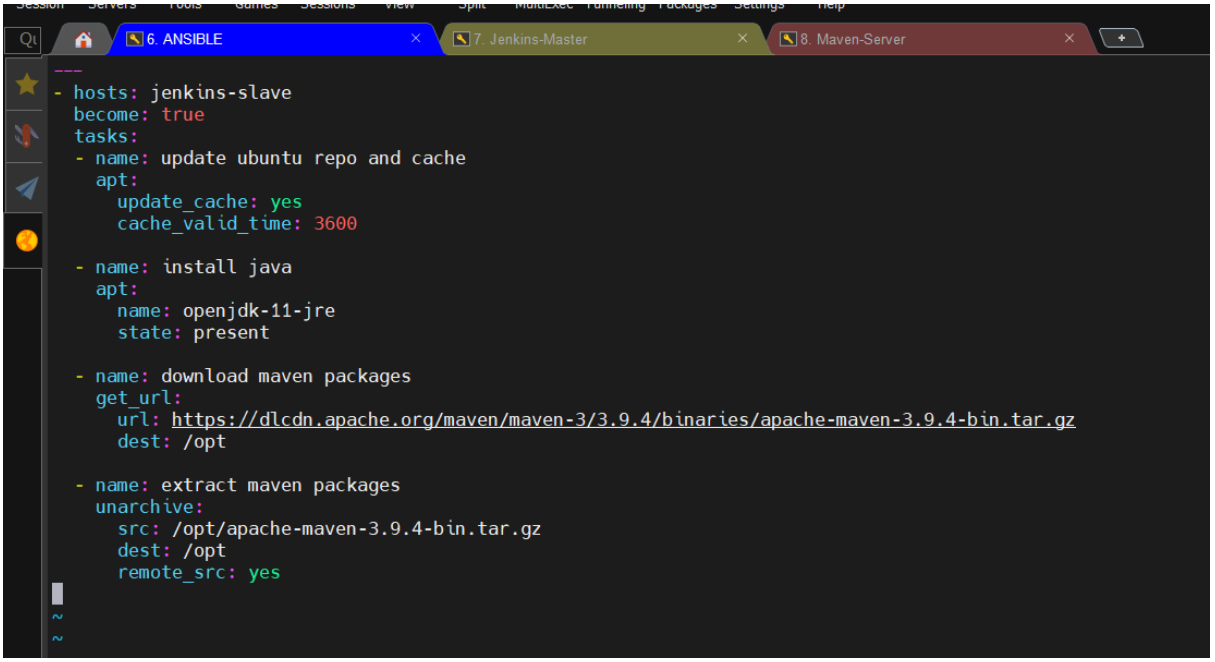
The document also includes screenshots of logging into the Jenkins-slave server, editing the playbook file, running the ansible-playbook command, and checking the Maven version on the Maven-server.

- **Update the System**
- **Install Java**
- **Download Maven Packages Extract it**
- **Add path to bash_profile**

Login to Jenkins-slave server



vi jenkins-slave-setup.yml



Run ansible-playbook

```
root@ip-10-1-1-177:/opt# vi jenkins-slave-setup.yml
root@ip-10-1-1-177:/opt# ansible-playbook -i /opt/hosts jenkins-slave-setup.yml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [jenkins-slave] ***********************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************
ok: [10.1.1.59]

TASK [update ubuntu repo and cache] ********************************************************************************************************
ok: [10.1.1.59]

TASK [install java] ************************************************************************************************************************
ok: [10.1.1.59]

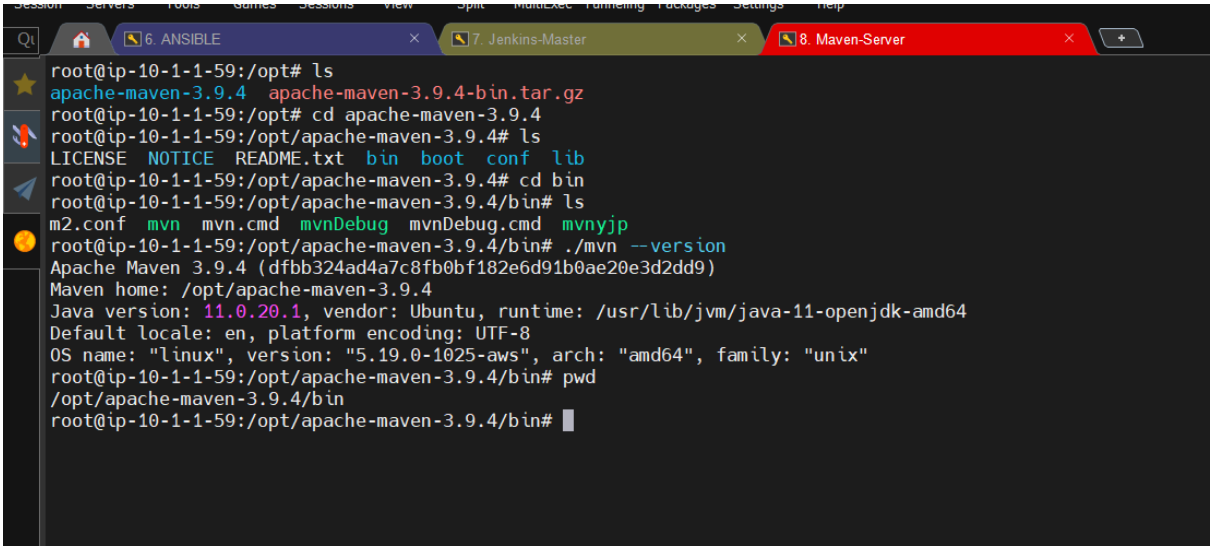
TASK [download maven packages] *************************************************************************************************************
changed: [10.1.1.59]

TASK [extract maven packages] **************************************************************************************************************
changed: [10.1.1.59]

PLAY RECAP *********************************************************************************************************************************
10.1.1.59                  : ok=5    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@ip-10-1-1-177:/opt#
```

Check mvn version on Maven-server



This is how you can configure your Jenkins master and slave systems by using Ansible. So this is how we can use Ansible in the real world for our project.

==========

Jenkins Pipeline

This document provides instructions for adding credentials and a slave node to Jenkins master. The steps include adding Jenkins-slave server credentials, creating a new node, and testing with a small freestyle project.

I need to follow two steps that is add credentials and slave node we need to add to Jenkins master. These credentials are used to log in to the slave system from the master node.

```
# Jenkins Master and Slave Setup

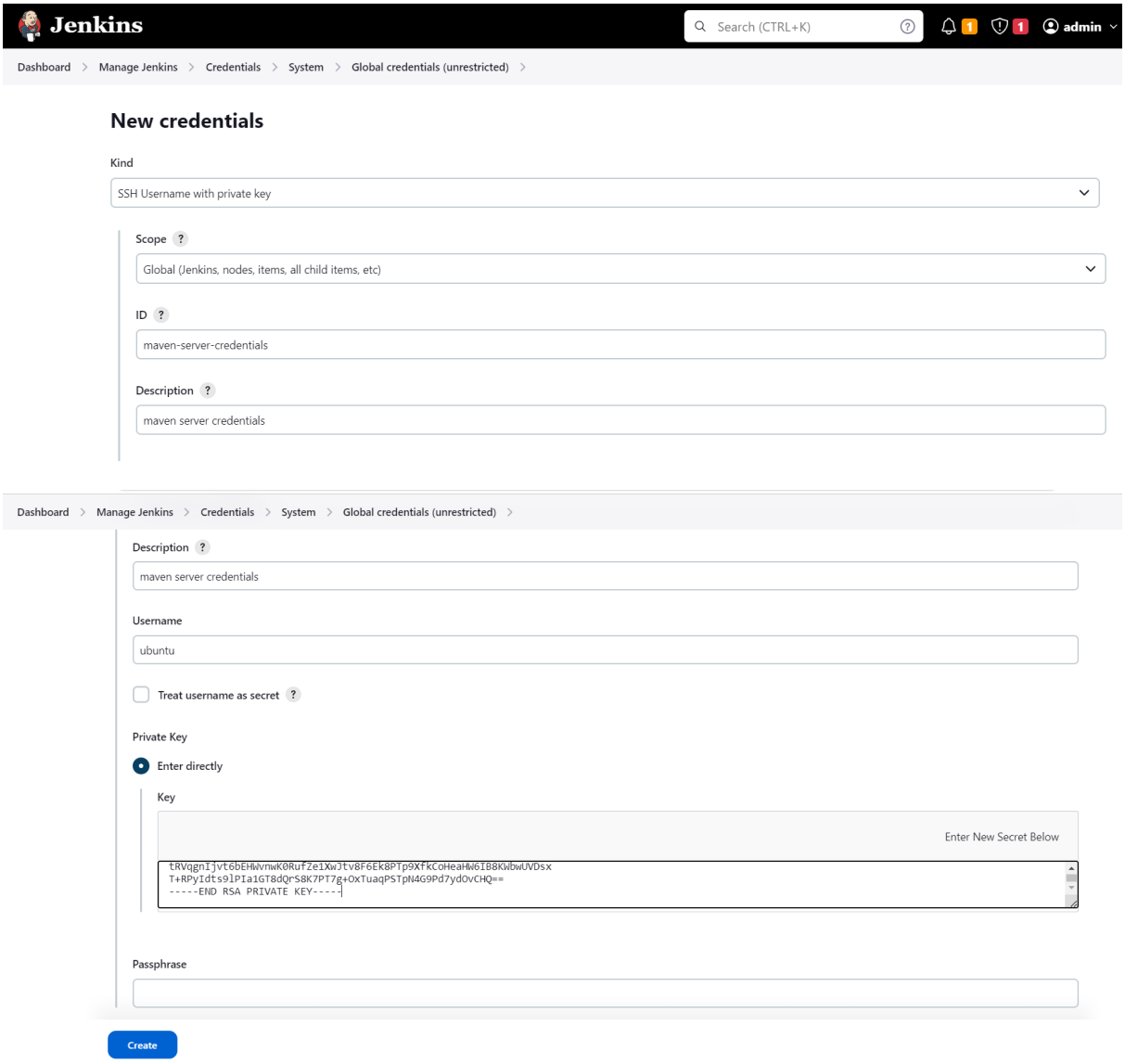
1. Add credentials
2. Add node

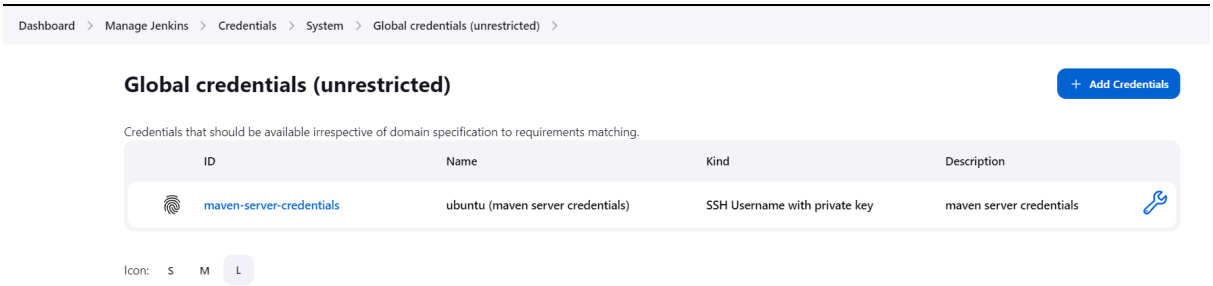
### Add Credentials
1. Manage Jenkins --> Manage Credentials --> System --> Global credentials --> Add credentials
2. Provide the below info to add credentials
   kind: `ssh username with private key`
```

```
   Scope: `Global`
   ID: `maven_slave`
   Username: `ubuntu`
   private key: `linux-KP.pem key content`

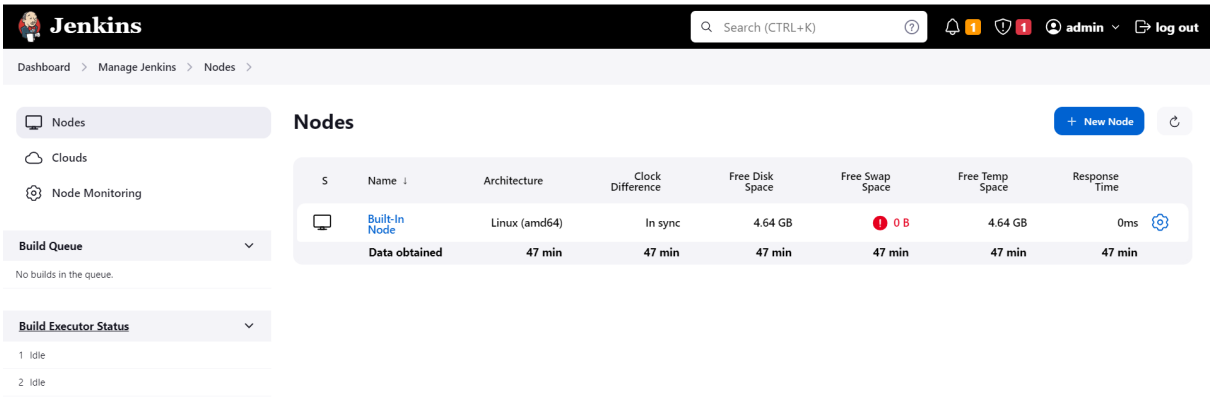
### Add node
   Follow the below setups to add a new slave node to the jenkins
1. Goto Manage Jenkins --> Manage nodes and clouds --> New node --> Permanent Agent
2. Provide the below info to add the node
   Number of executors: `3`
   Remote root directory: `/home/ubuntu/jenkins`
   Labels: `maven`
   Usage: `Use this node as much as possible`
   Launch method: `Launch agents via SSH`
        Host: `<Private_IP_of_Slave>`
        Credentials: `<Jenkins_Slave_Credentials>`
        Host Key Verification Strategy: `Non verifying Verification Strategy`
   Availability: `Keep this agent online as much as possible`
```

Adding Jenkins-slave server credentials to Jenkins-master

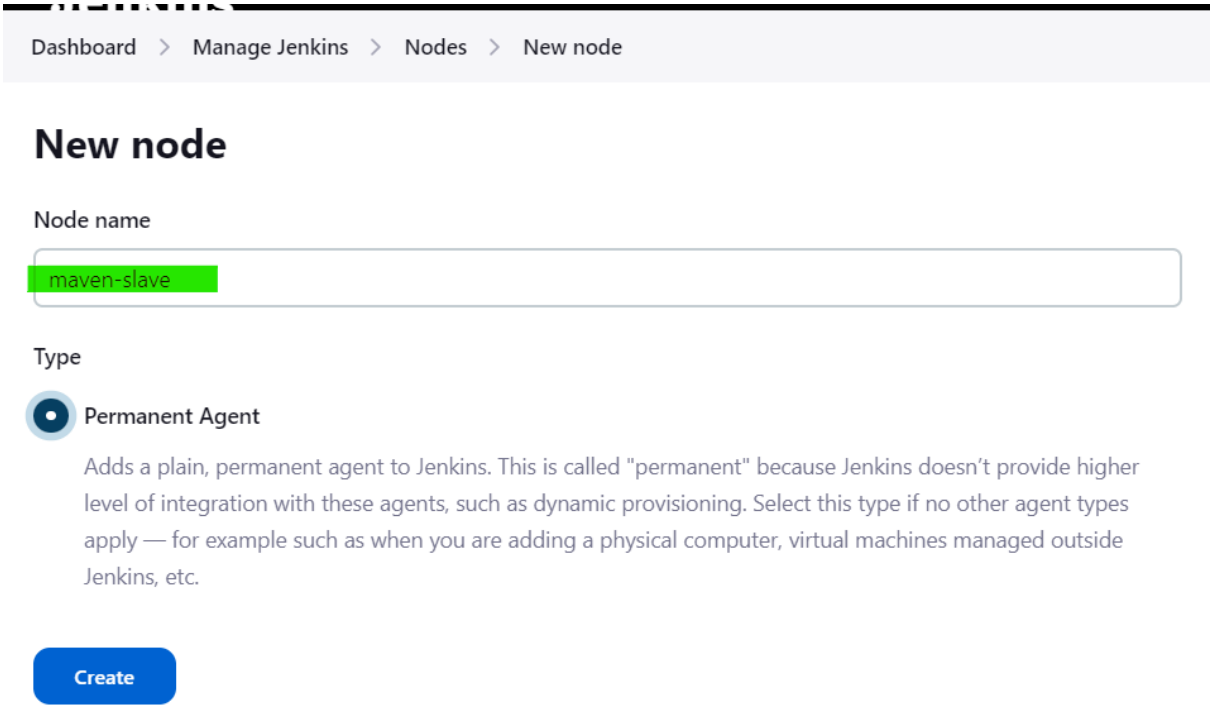




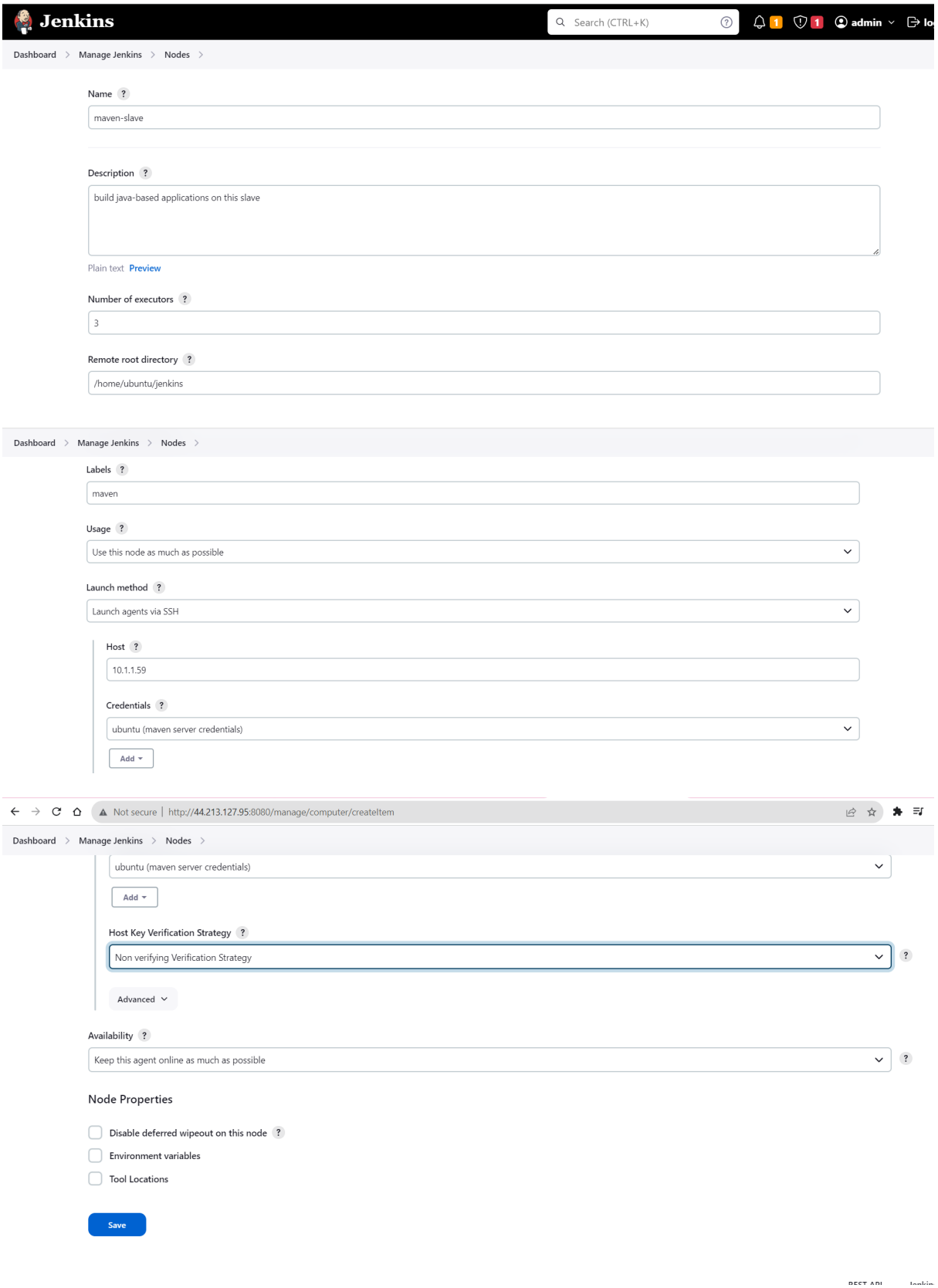
Adding Node to Jenkins-master



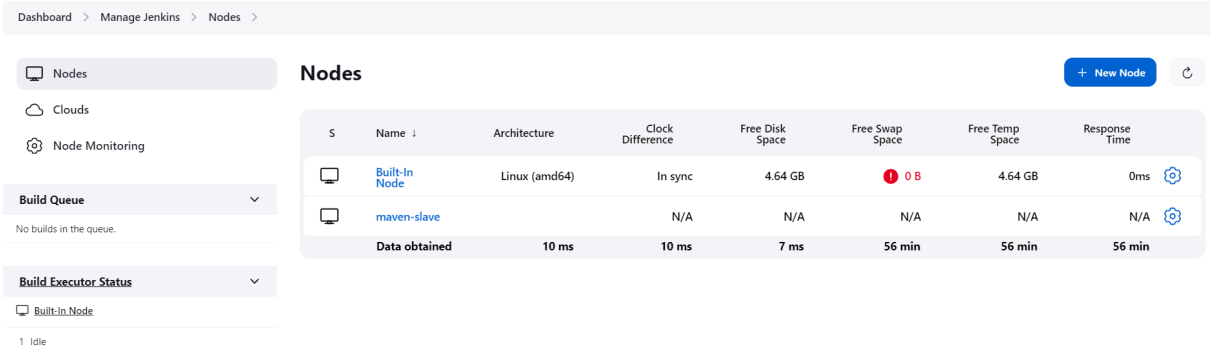
Create New Node



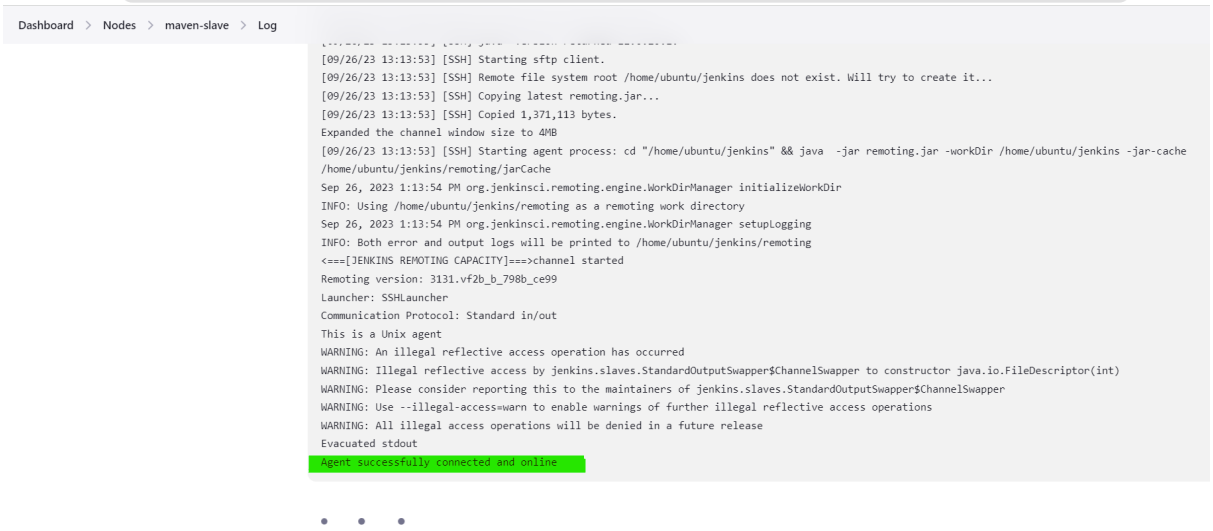
Add



Node created

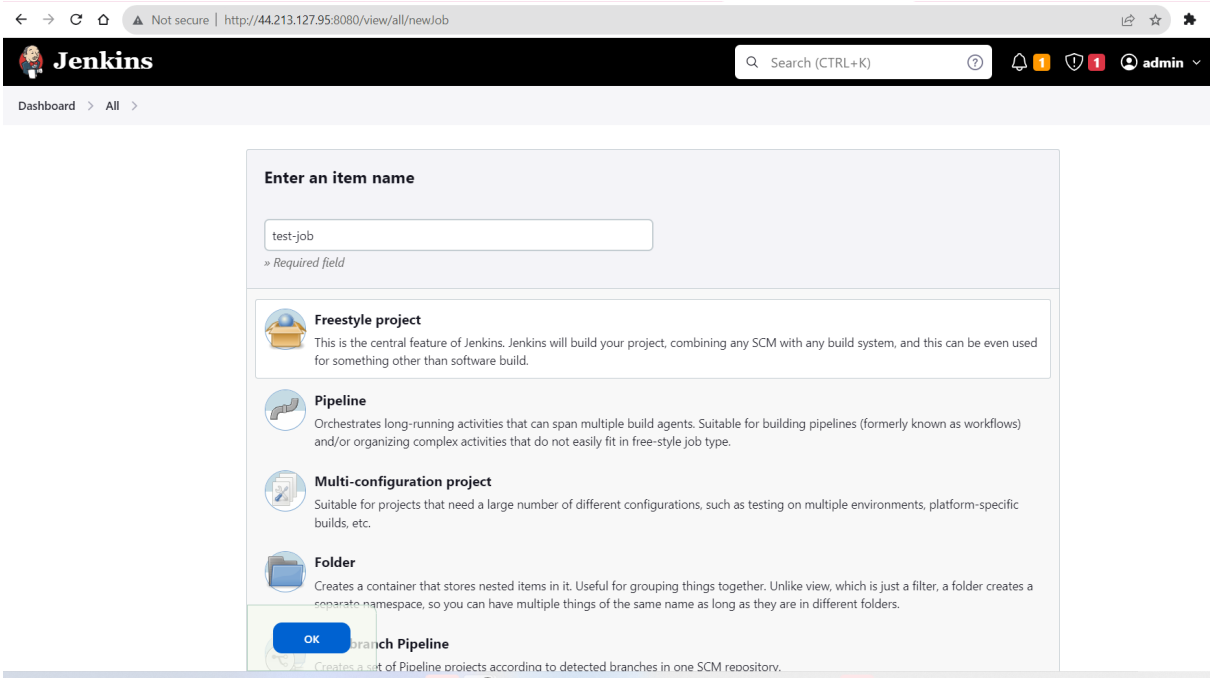


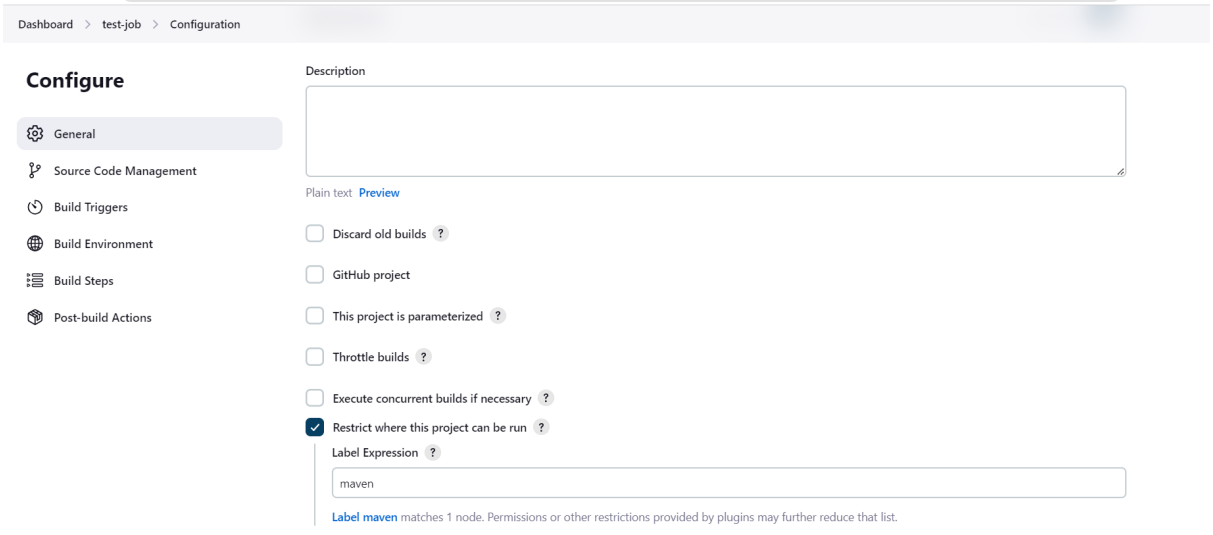
Logs : Agent successfully connected and online

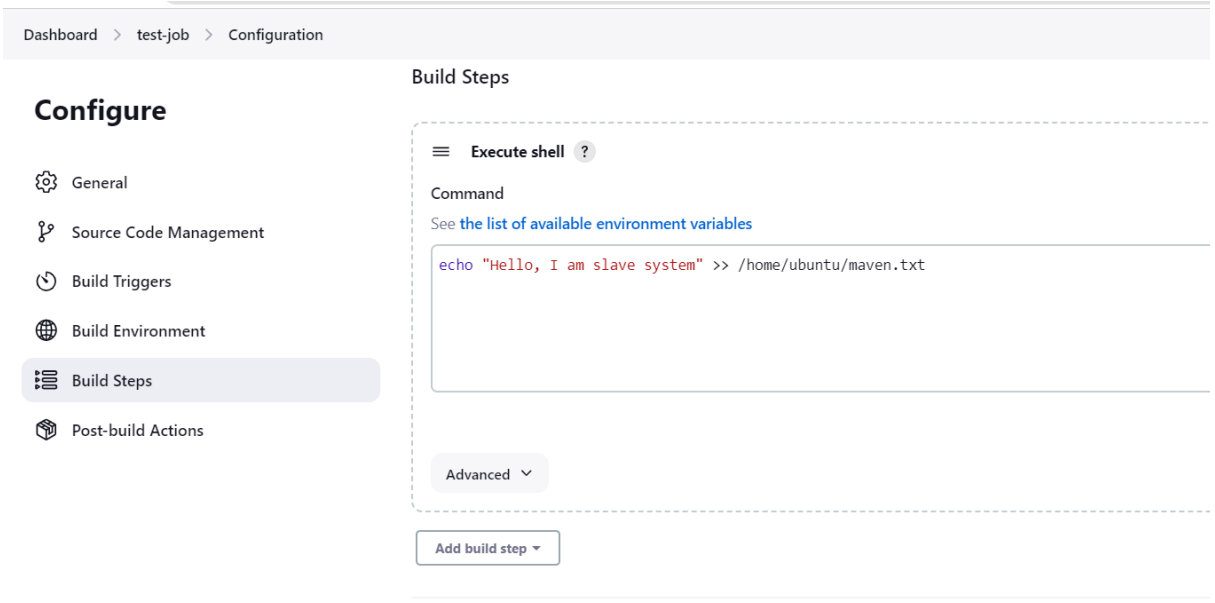


Testing with small freestyle project

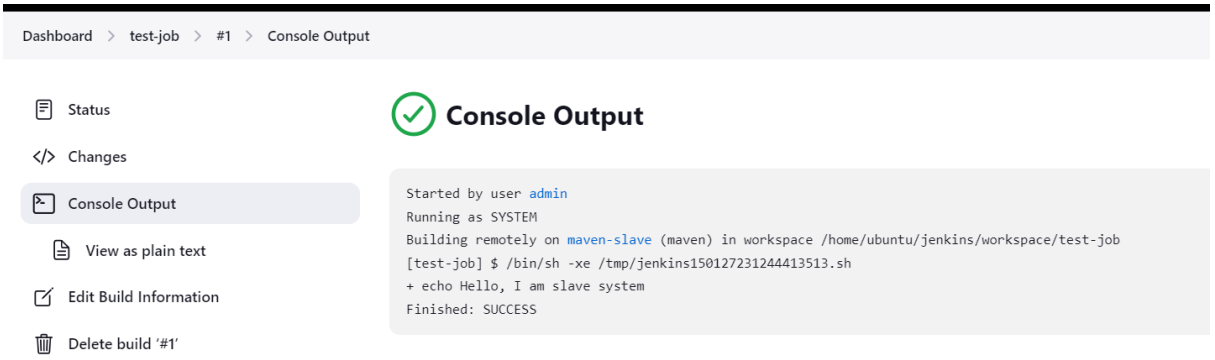
Created a test-job freestyle project


Not secure | http://44.213.127.95:8080/view/all/newJob
Jenkins
Search (CTRL+K)
admin
Dashboard > All >
Enter an item name
test-job
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
OK

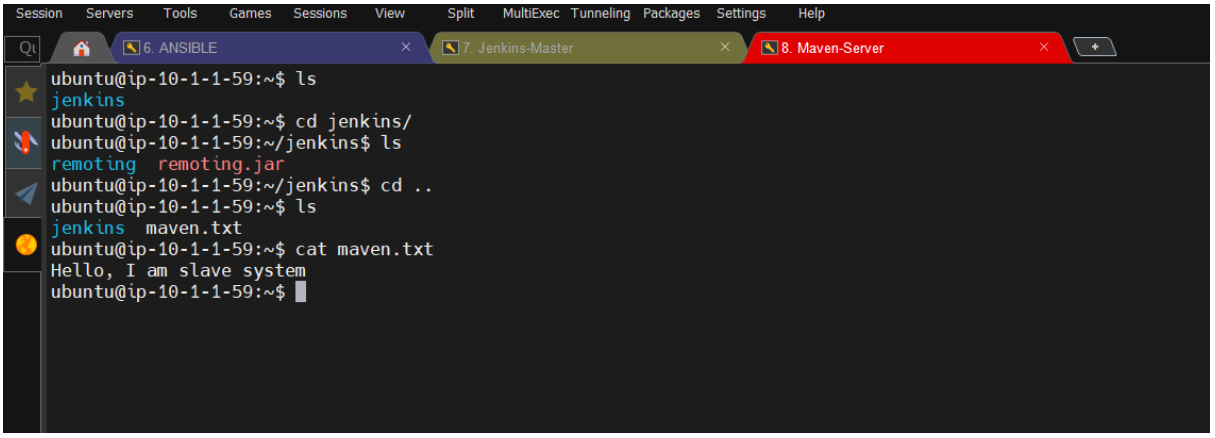

Dashboard > test-job > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Description
Plain text Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Restrict where this project can be run
Label Expression
maven
Label maven matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.



Build Now



maven.txt file created



==========

Multi-branch Pipeline and webhook

This document provides a step-by-step guide on setting up a pipeline and webhook in Jenkins. It covers topics such as creating a pipeline job, writing a Jenkinsfile, adding GitHub credentials, creating a multi-branch pipeline, and setting up a GitHub webhook. Screenshots are included to illustrate the process.

```
# Enable Webhook
1. Install "multibranch scan webhook trigger" plugin
    From dashboard --> manage jenkins --> manage plugins --> Available Plugins
    Search for "Multibranch Scan webhook Trigger" plugin and install it.

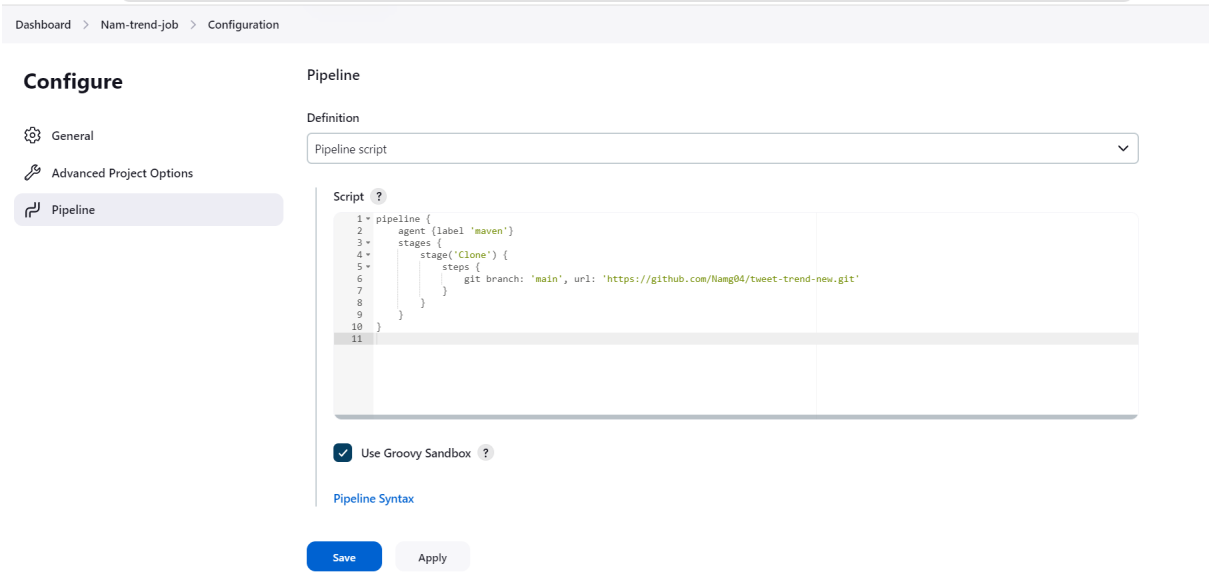
2. Go to multibranch pipeline job
     job --> configure --> Scan Multibranch Pipeline Triggers --> Scan Multibranch Pipeline Triggers  --> Scan by webhook
     Trigger token: `<token_name>`

3. Add webhook to GitHub repository
   Github repo --> settings --> webhooks --> Add webhook
   Payload URl: `<jenkins_IP>:8080/multibranch-webhook-trigger/invoke?token=<token_name>`
   Content type: `application/json`
   Which event would you like to trigger this webhook: `just the push event`

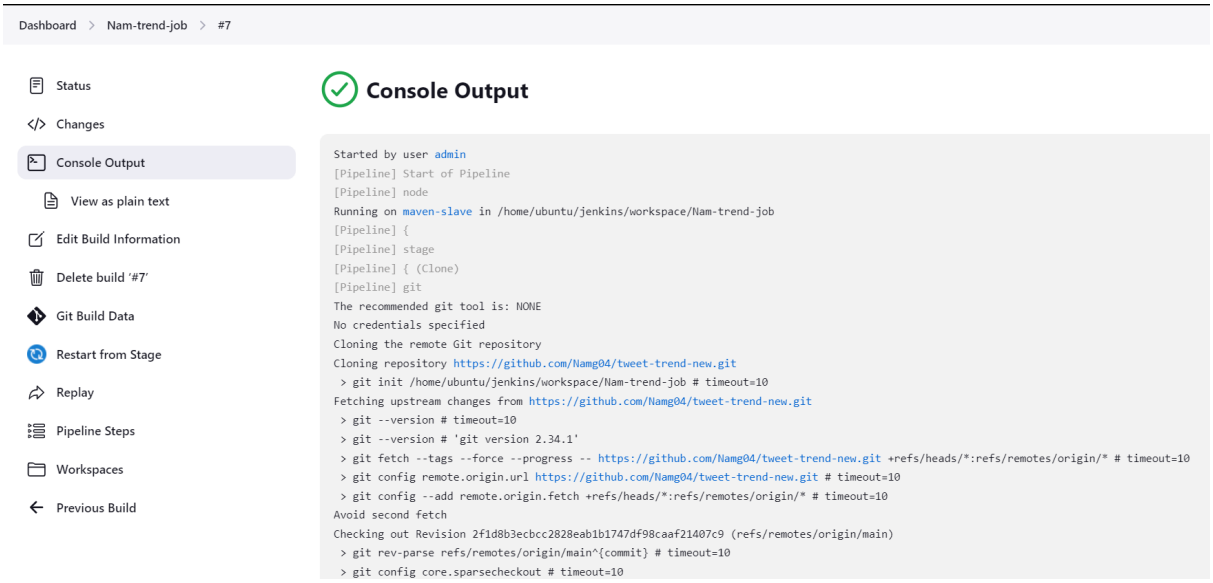

Once it is enabled make changes to source to trigger the build.
```

Creating Pipeline in Jenkins

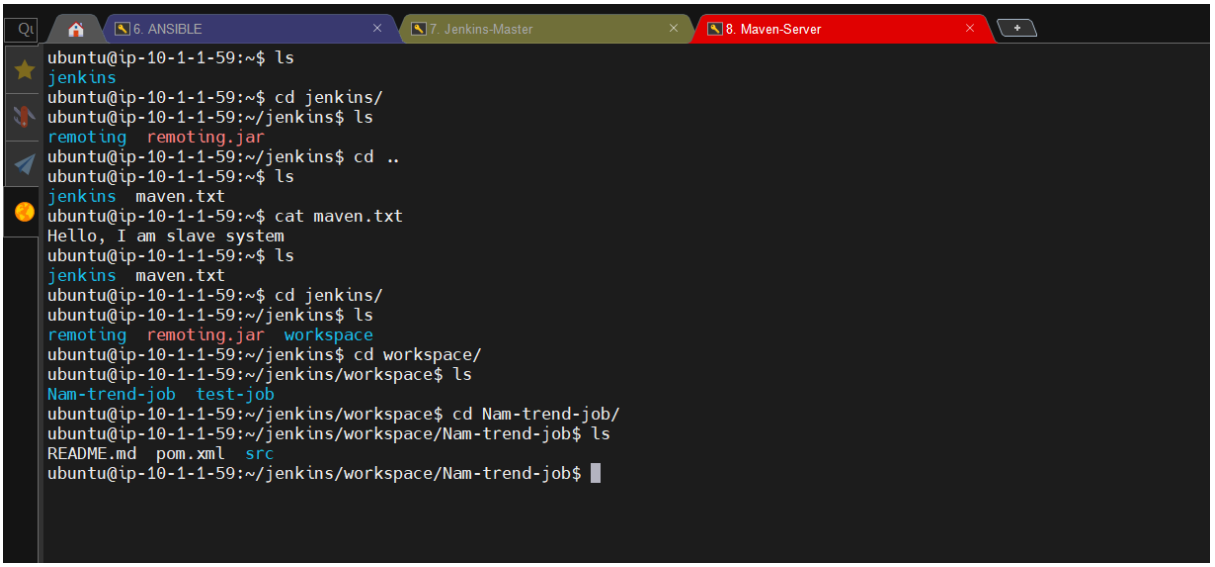
- New Project - Nam-trend-job as pipeline
- Pipeline declarative script



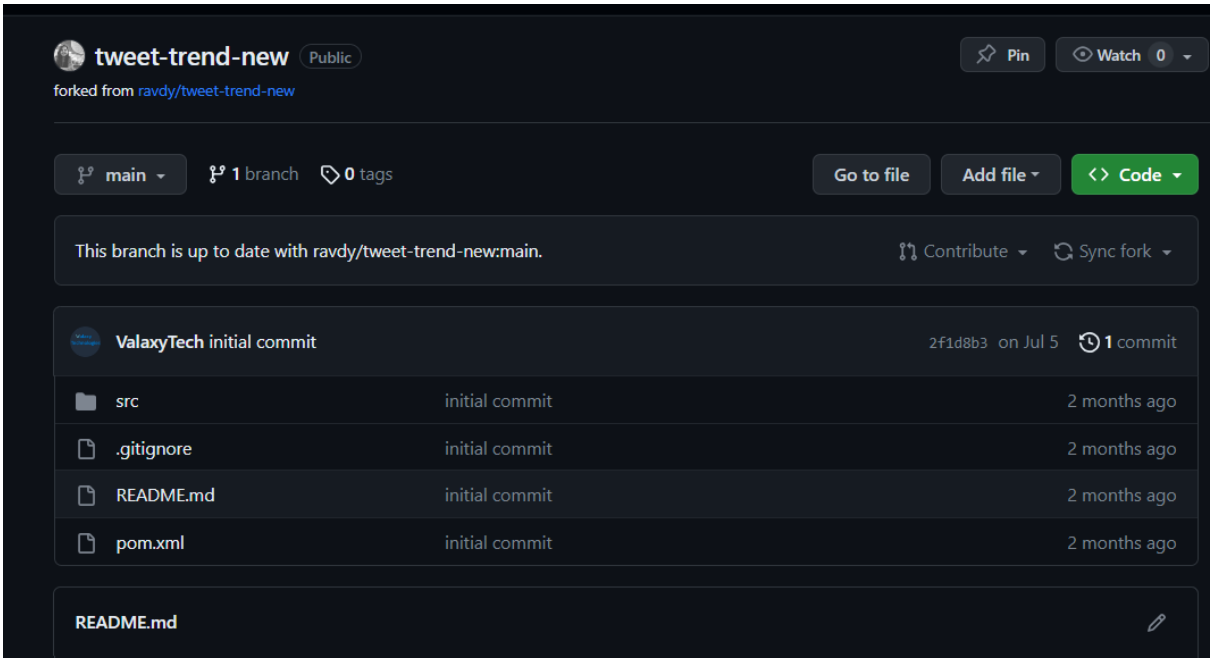
Build now



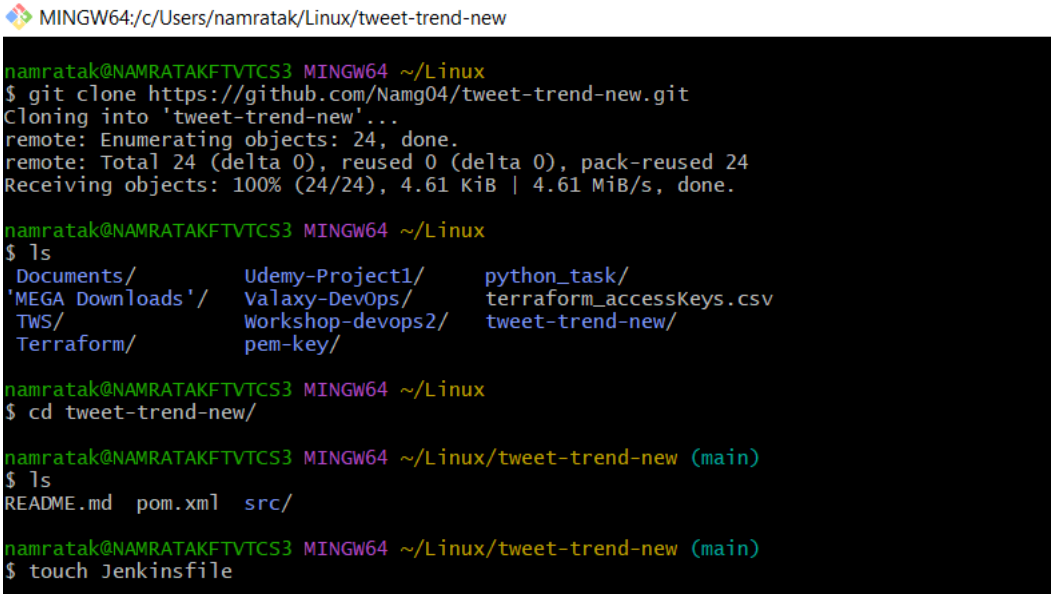
This is how to clone the job through pipeline job



Write Jenkins file with build stage. To update Jenkinsfile on Git repo first clone this repo on local

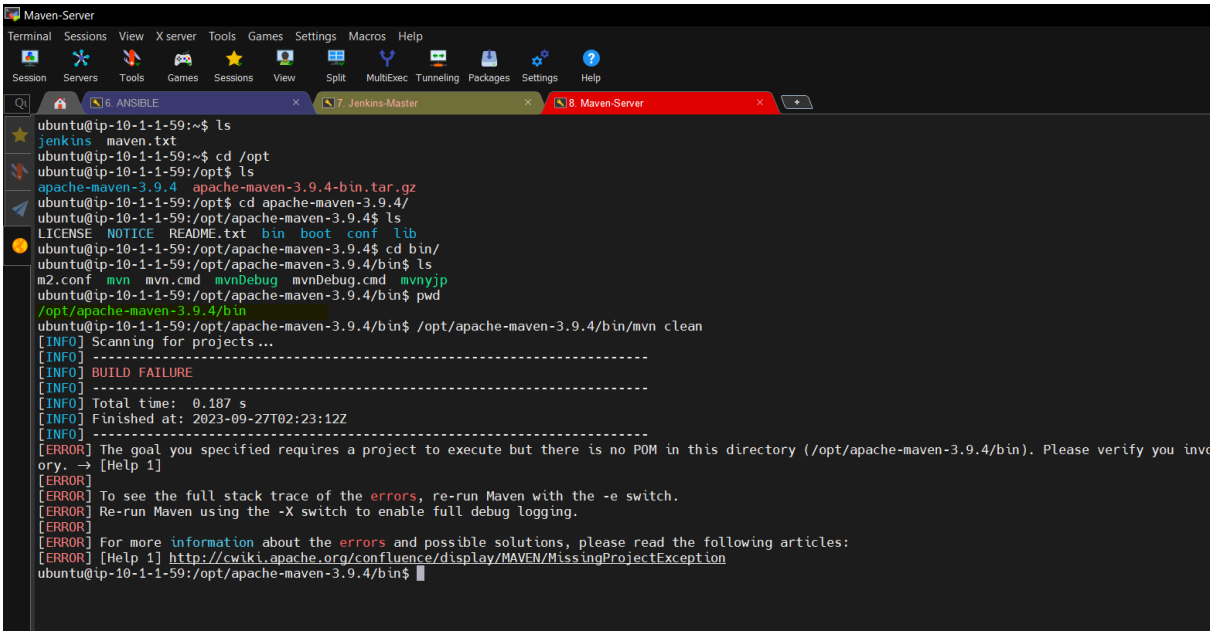


Cloned repo on local and created a file 'Jenkinsfile'



Write a Jenkins file with a build stage

a)You need to mention maven path where you have installed in your system



b)Give maven path in Jenkins file

c)commit the Jenkins file into Git repo

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Jenkinsfile

nothing added to commit but untracked files present (use "git add" to track)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

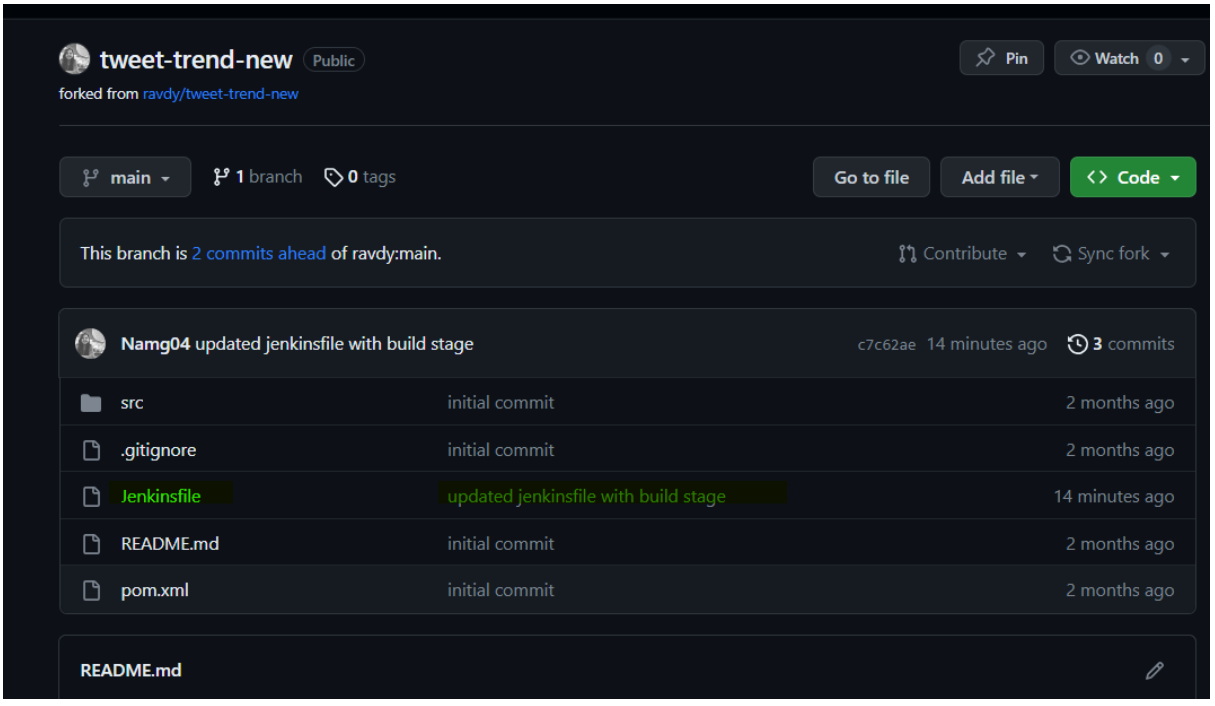
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Jenkinsfile

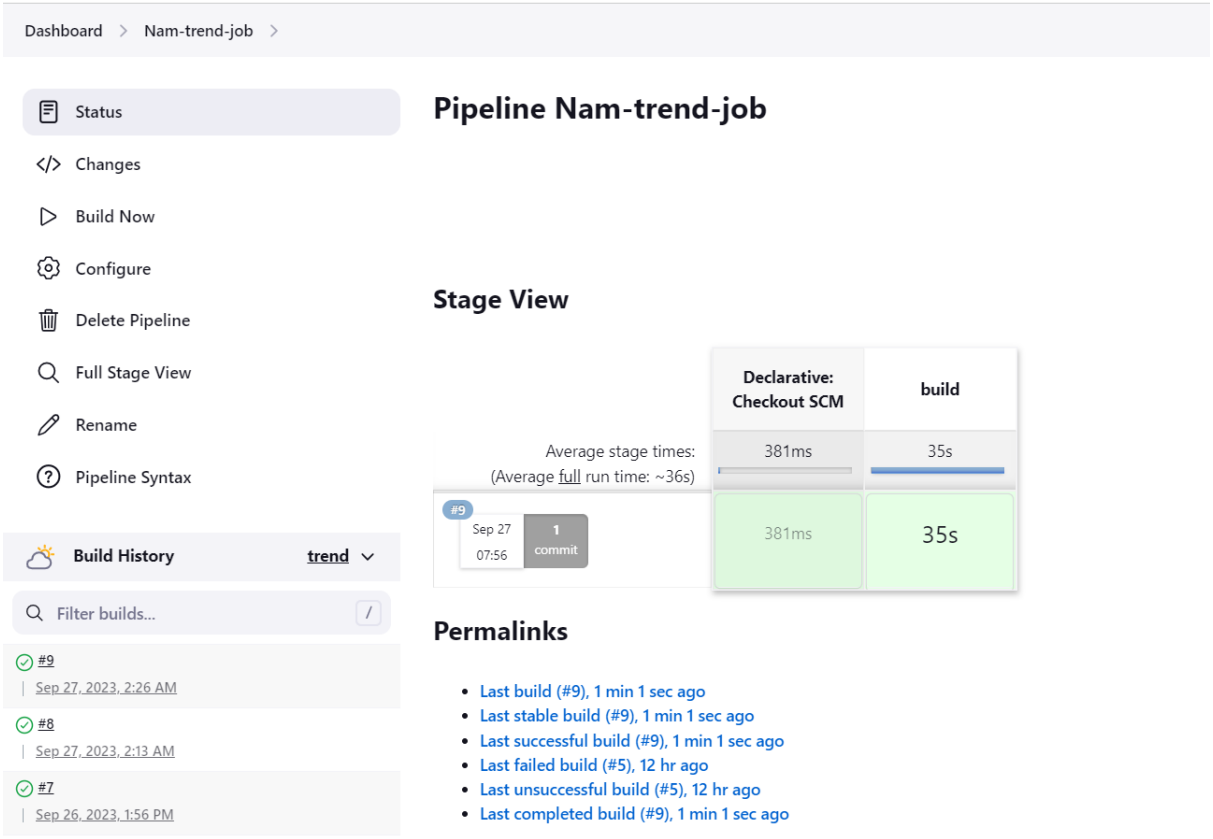

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "Added Jenkinsfile to repo"
[main adfc97d] Added Jenkinsfile to repo
 1 file changed, 10 insertions(+)
 create mode 100644 Jenkinsfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 413 bytes | 413.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Namg04/tweet-trend-new.git
   2f1d8b3..adfc97d  main -> main
```

d)code committed to git

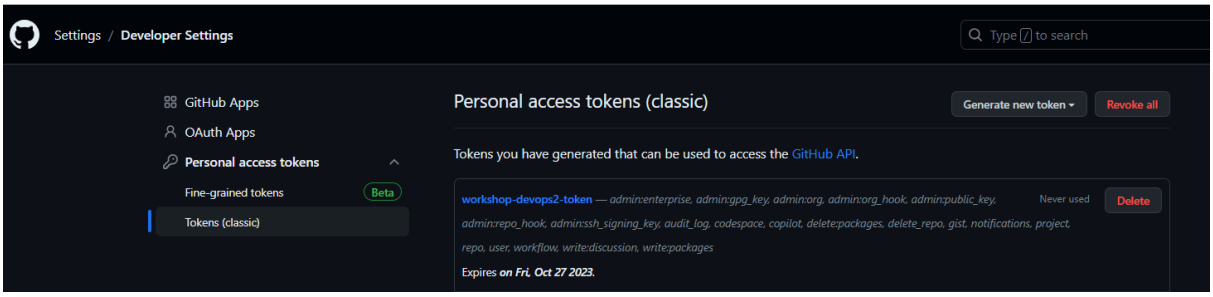


Now build the job in Jenkins

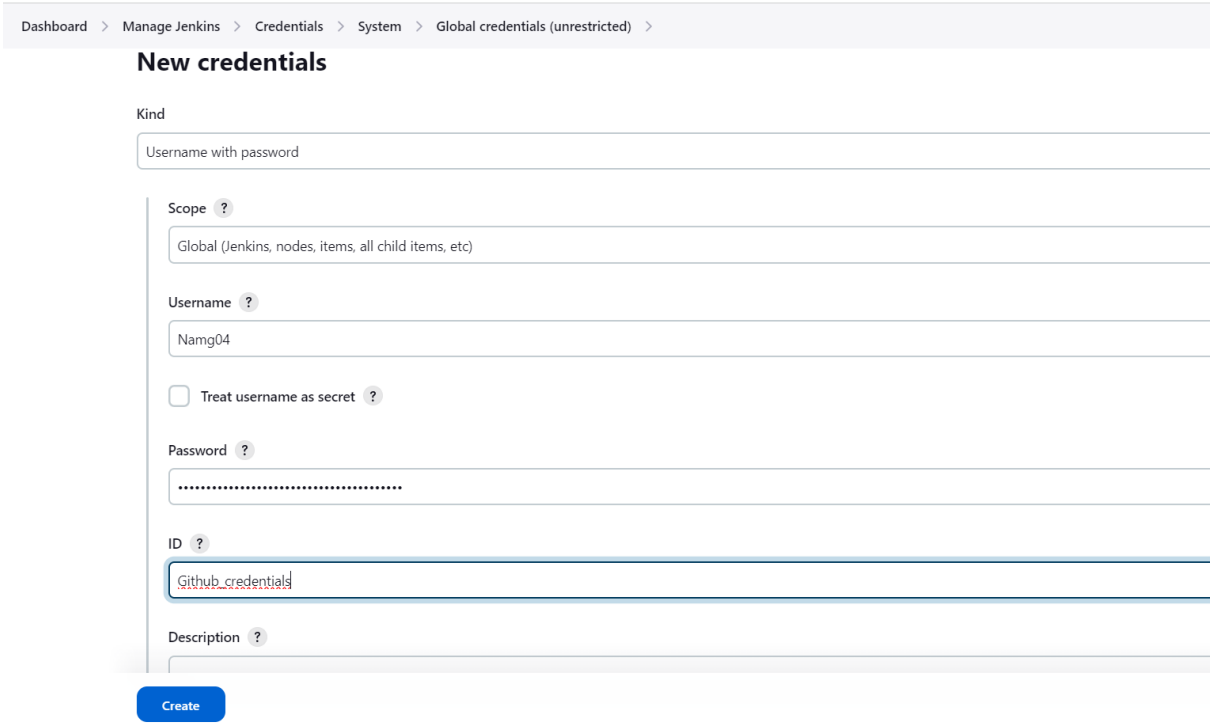


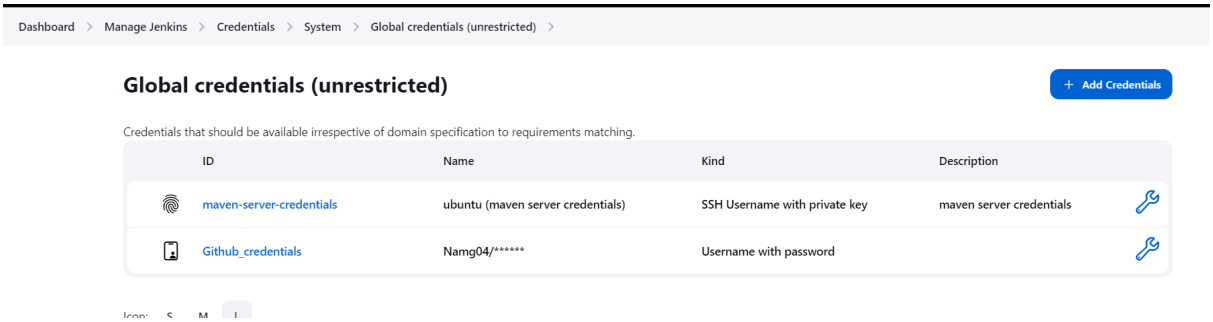
How to add GitHub credentials to Jenkins.

Create Personal Token in Gitgub

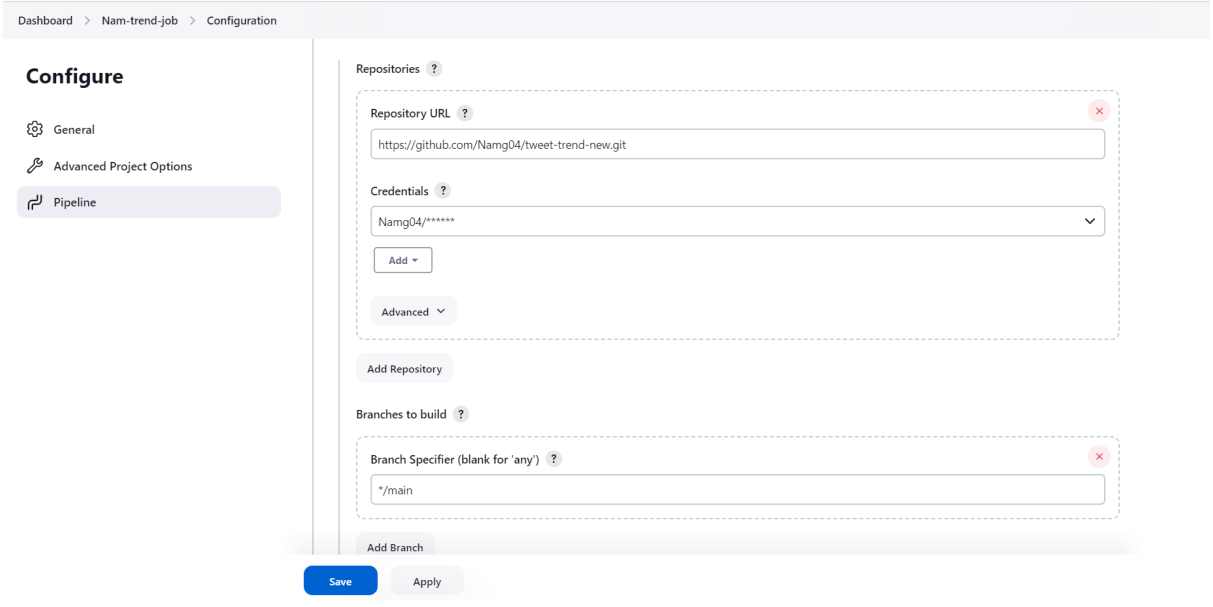


Add GitHub credentials in Jenkins

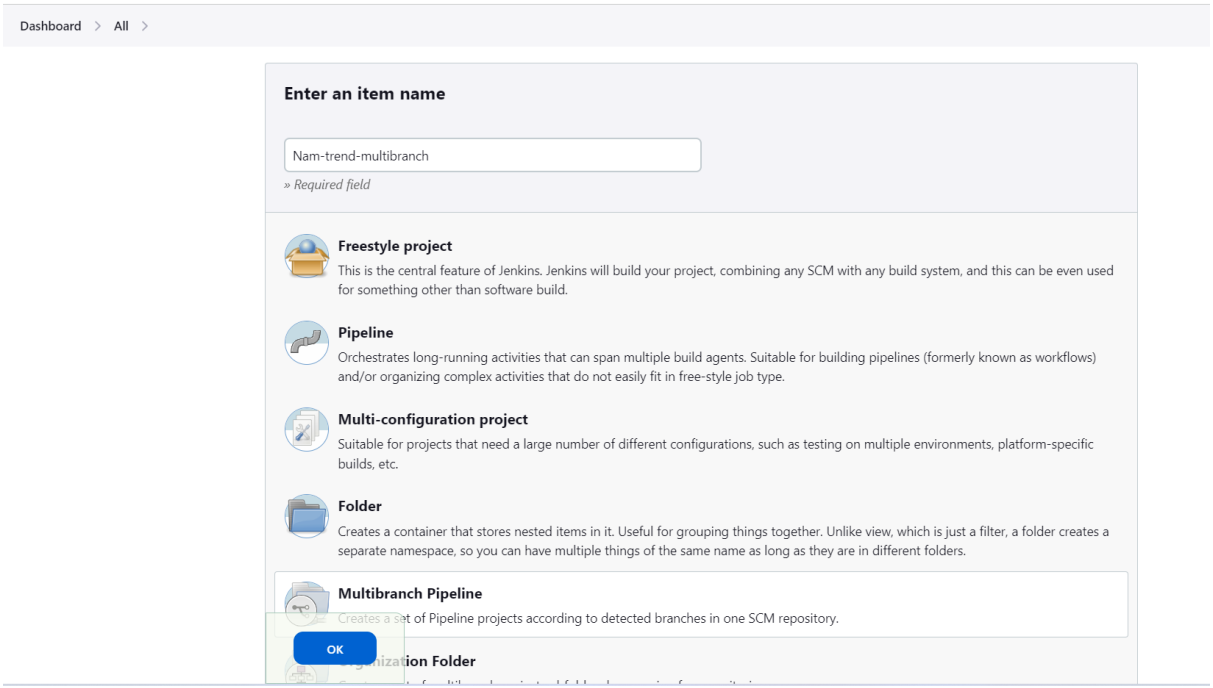


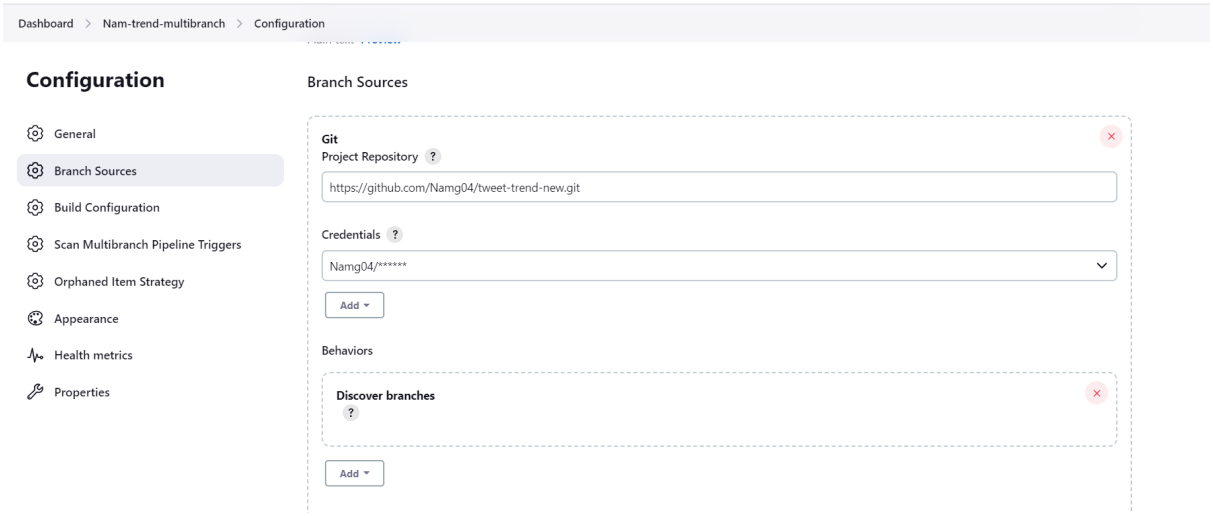


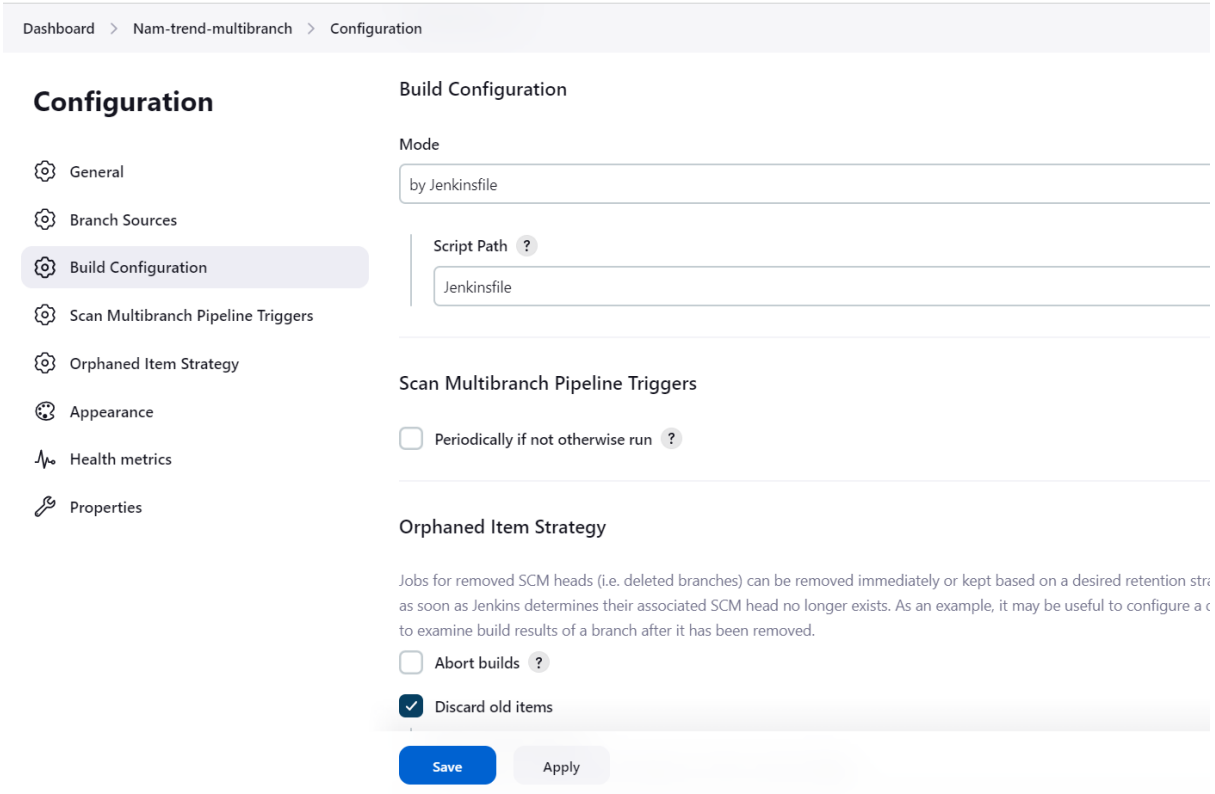
Add credentials in Nam-trend-job

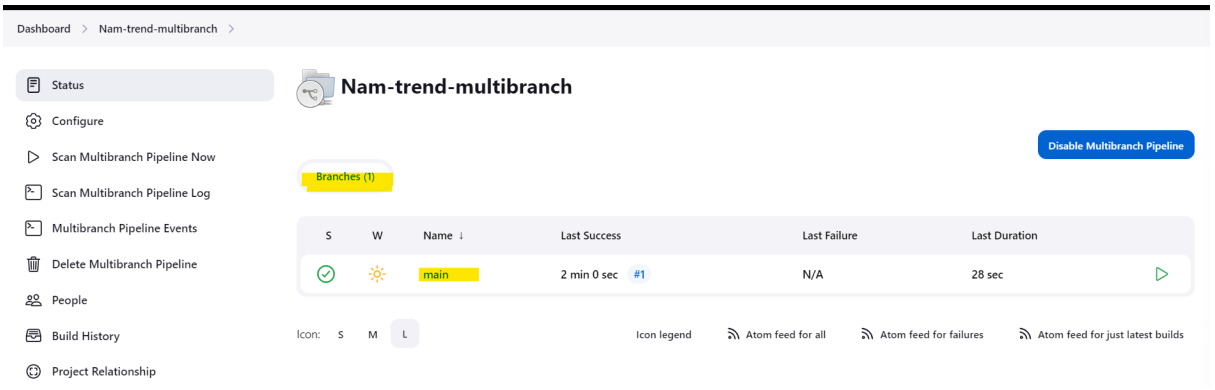


Create Multi branch pipeline

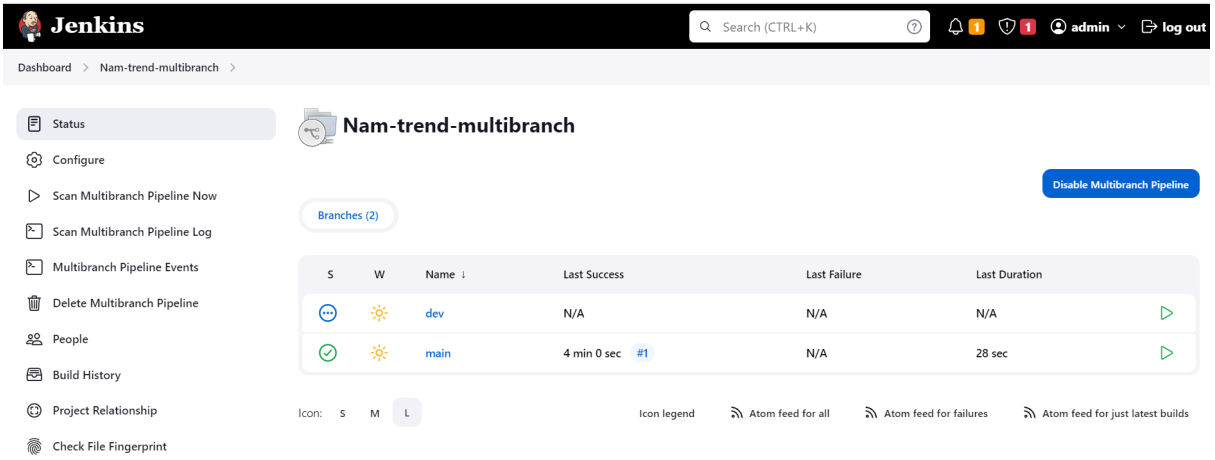

Dashboard
All
Enter an item name
Nam-trend-multibranch
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
Multibranch Pipeline
OK


Dashboard
Nam-trend-multibranch
Configuration
Configuration
General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Appearance
Health metrics
Properties
Branch Sources
Git
Project Repository
https://github.com/Namg04/tweet-trend-new.git
Credentials
Namg04/******
Add
Behaviors
Discover branches
Add





I have created a branch dev and clicked on scan multibranch pipeline now dev branch is automatically updated If you add any new branches, you need to scan multi branch pipeline now.



Commit code into git

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git checkout -b stage
Switched to a new branch 'stage'

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ ls
Jenkinsfile  README.md  pom.xml  src/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ rm -rf Jenkinsfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git status
On branch stage
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    Jenkinsfile

no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git add .

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git commit -m " adding stage branch"
[stage a8efee5]  adding stage branch
 1 file changed, 15 deletions(-)
 delete mode 100644 Jenkinsfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git status
On branch stage
nothing to commit, working tree clean

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git push origin stage
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 20 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 229 bytes | 229.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'stage' on GitHub by visiting:
remote:      https://github.com/Namg04/tweet-trend-new/pull/new/stage
remote:
To https://github.com/Namg04/tweet-trend-new.git
 * [new branch]      stage -> stage

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$
```

Added Jenkinsfile to stage branch

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ cat > Jenkinsfile
pipeline {
    agent {label 'maven'}


environment {
    PATH = "/opt/apache-maven-3.9.4/bin:$PATH"
}
    stages {
       stage("build"){
        steps {
            sh 'mvn clean deploy'
        }
       }
    }
}



namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git status
On branch stage
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Jenkinsfile

nothing added to commit but untracked files present (use "git add" to track)

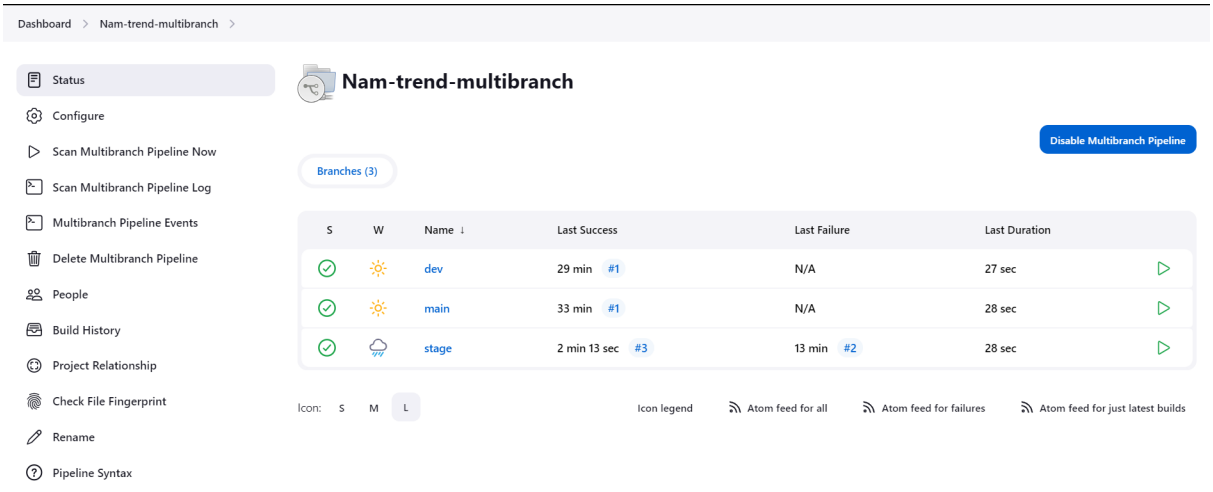
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git add .
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the next time Git touches it

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git status
On branch stage
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Jenkinsfile


namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git commit -m "added jenkinsfile to stage branch"
[stage f34091e] added jenkinsfile to stage branch
 1 file changed, 16 insertions(+)
 create mode 100644 Jenkinsfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git push origin stage
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 430 bytes | 430.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Namg04/tweet-trend-new.git
   a8efee5..f34091e  stage -> stage
```

Branch Stage added in Jenkins multibranch pipeline



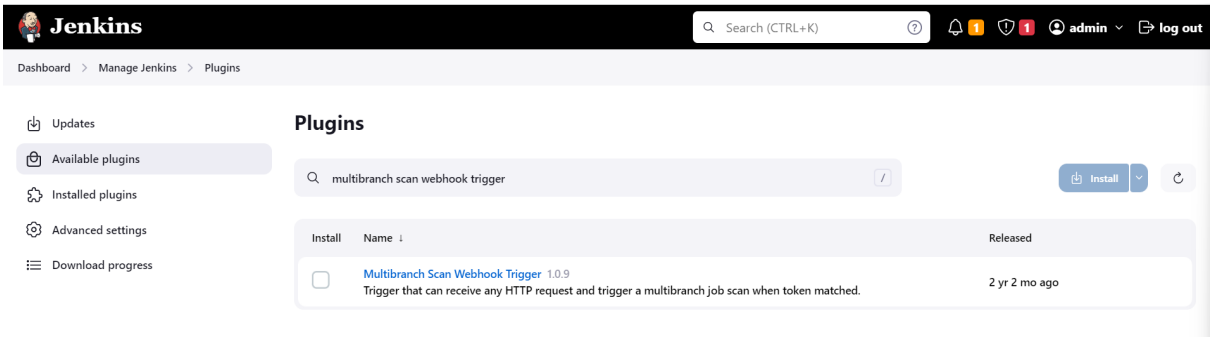
Setup GitHub Webhook

```
# Enable Webhook
1. Install "multibranch scan webhook trigger" plugin
    From dashboard --> manage jenkins --> manage plugins --> Available Plugins
    Search for "Multibranch Scan webhook Trigger" plugin and install it.

2. Go to multibranch pipeline job
     job --> configure --> Scan Multibranch Pipeline Triggers --> Scan Multibranch Pipeline Triggers  --> Scan by webhook
     Trigger token: `<token_name>`
```

```
3. Add webhook to GitHub repository
   Github repo --> settings --> webhooks --> Add webhook
   Payload URl: `<jenkins_IP>:8080/multibranch-webhook-trigger/invoke?token=<token_name>`
   Content type: `application/json`
   Which event would you like to trigger this webhook: `just the push event`

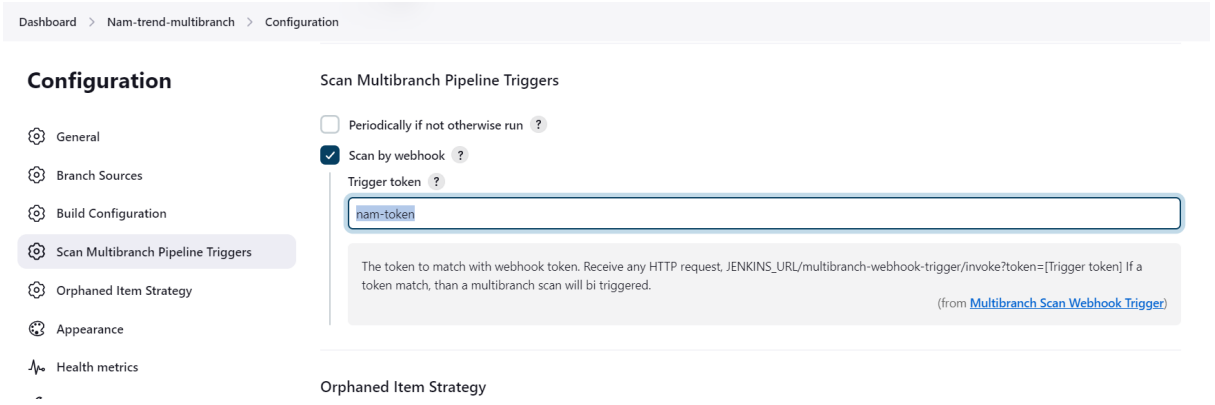

Once it is enabled make changes to source to trigger the build.
```



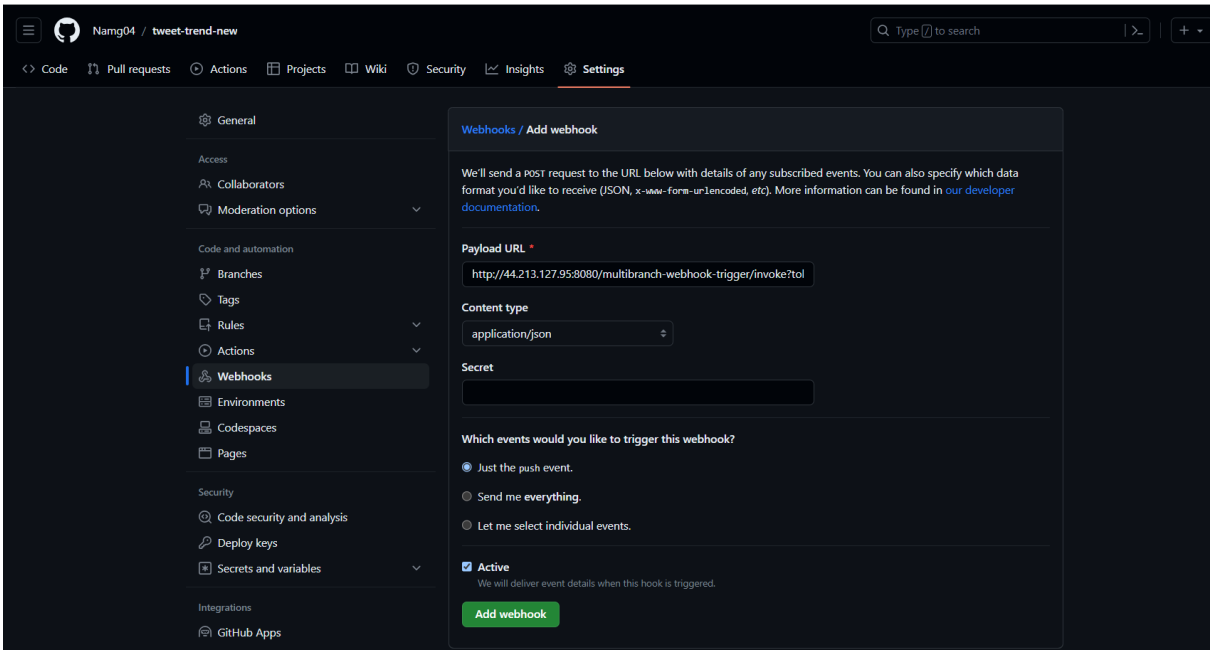
Enable Scan by Webhook and set payload url accordingly

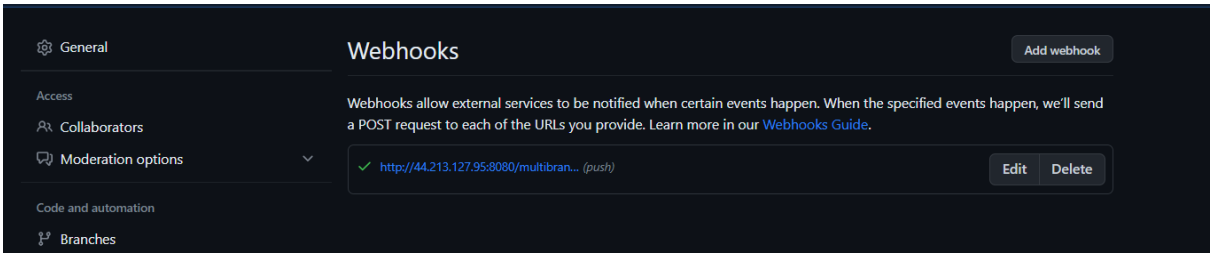
Payload URl: `<jenkins_IP>:8080/multibranch-webhook-trigger/invoke?token=<token_name>`

Content type: `application/json`

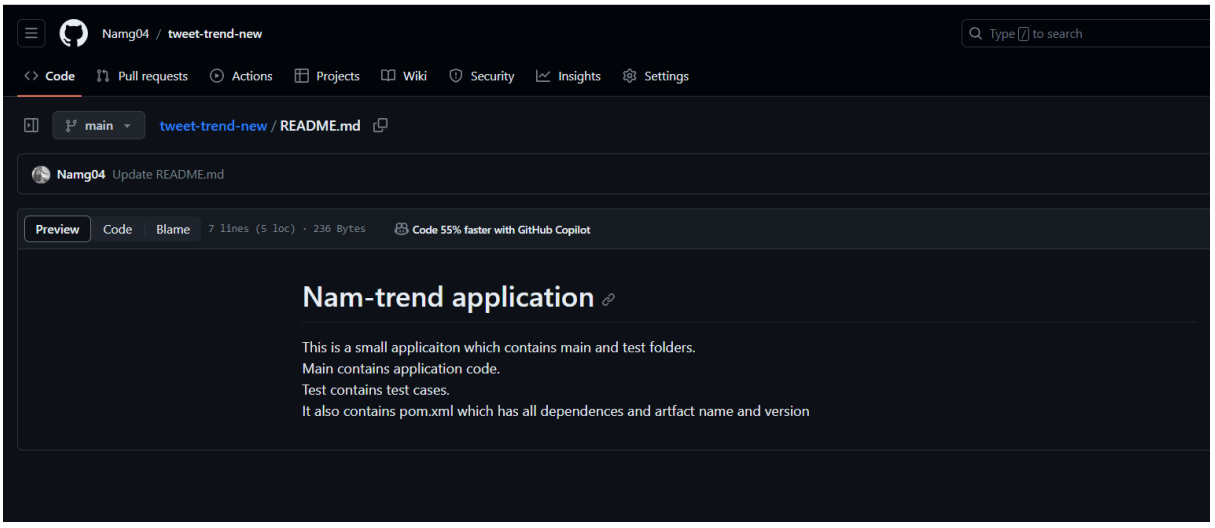


add webhook

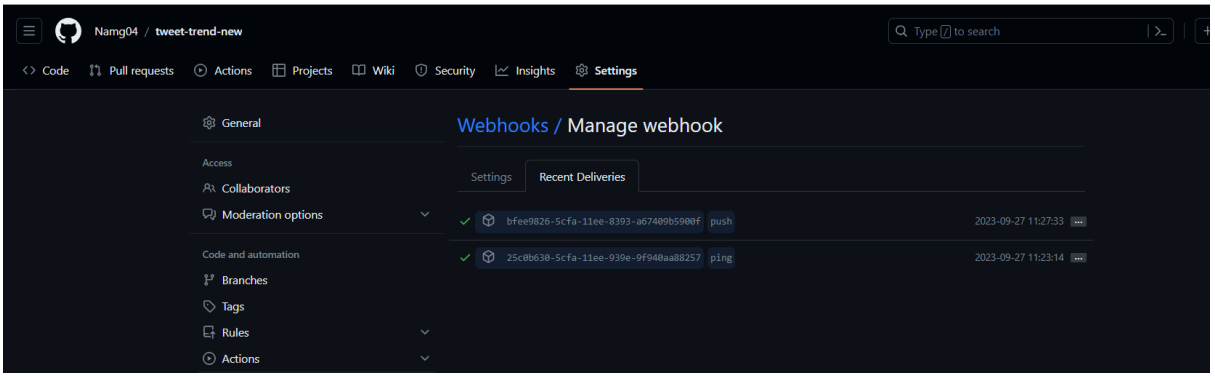




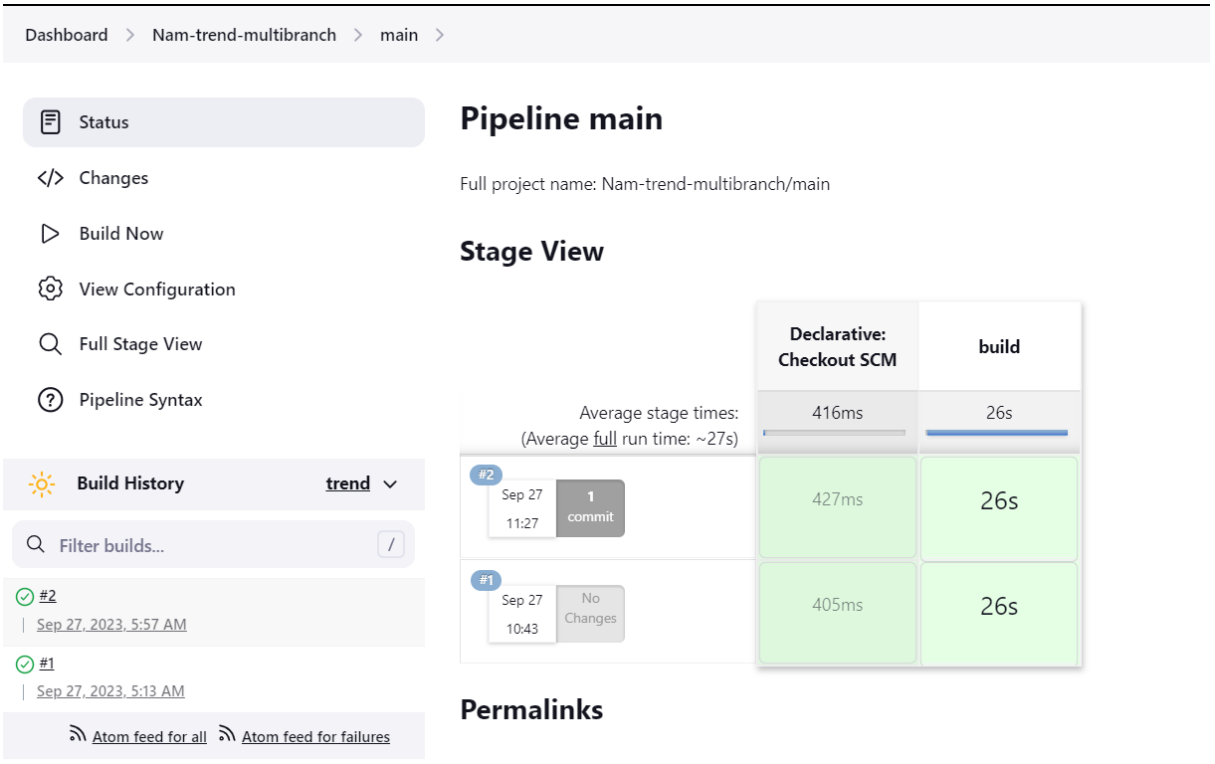
Made some changes in README.md file of main branch

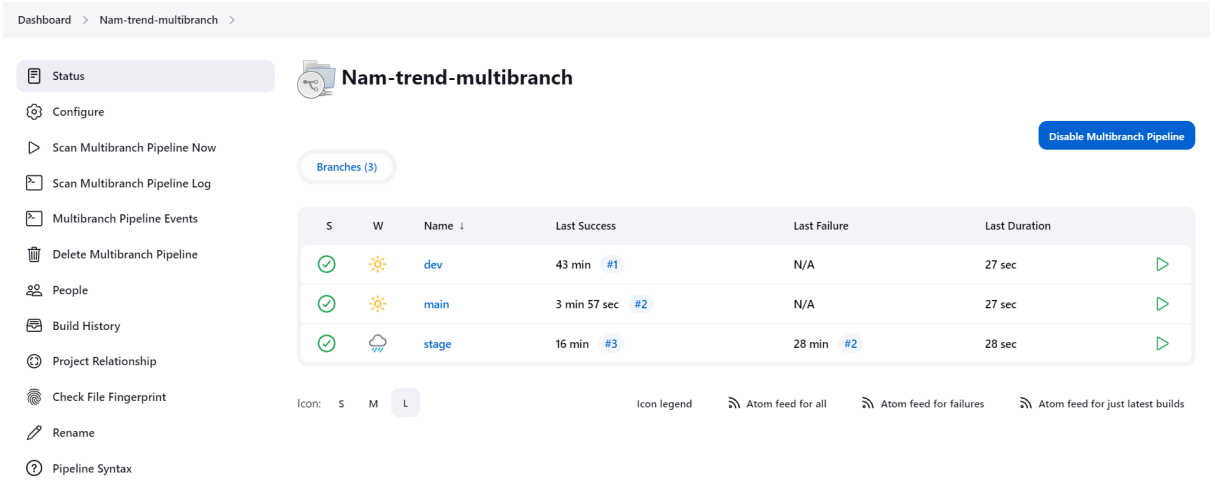


Push has been triggered

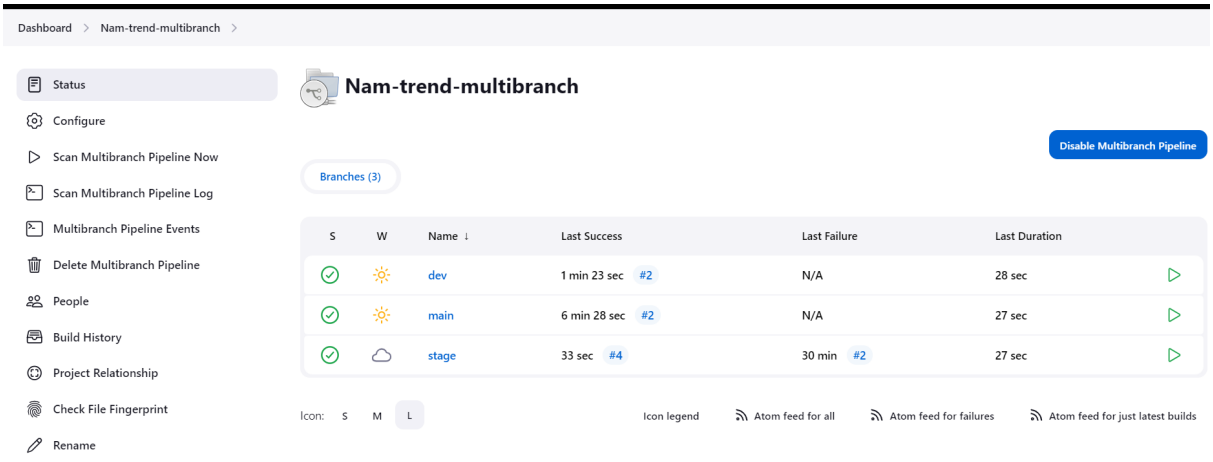


Build successfully triggered





I have made changes in all branched in Git and build have successfully done in all branches



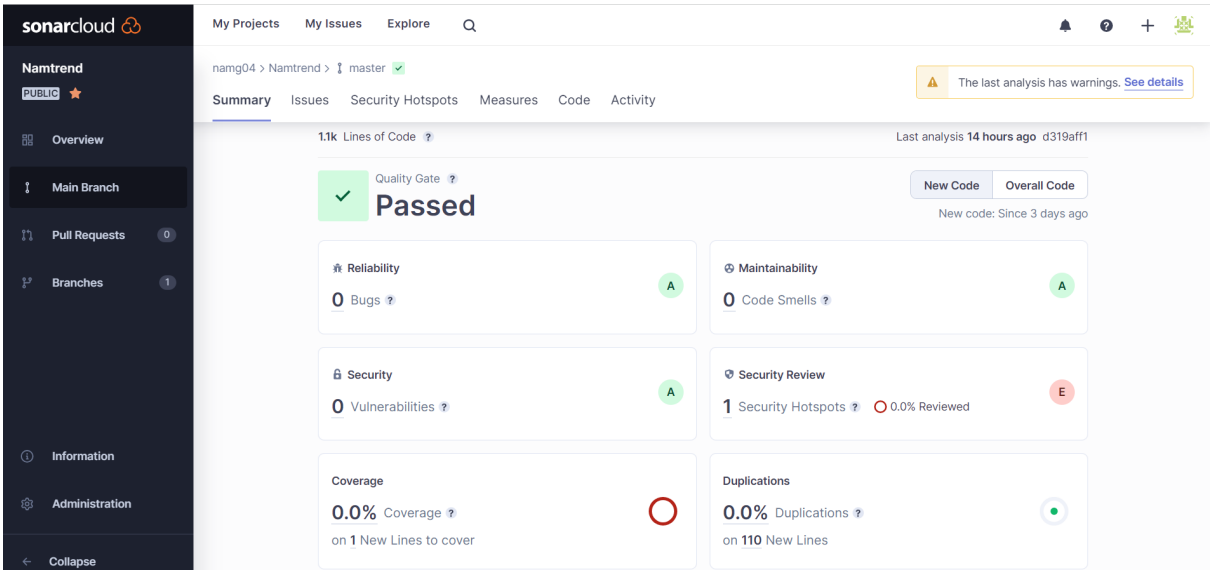
==========

Sonar Qube Integration Setup

This document provides instructions for setting up SonarQube integration with Jenkins.

It covers steps such as creating a SonarQube account, generating authentication tokens, installing SonarQube plugins, configuring SonarQube server and scanner, and adding SonarQube stages in the Jenkinsfile.

The document also includes screenshots of the setup process and demonstrates how to run builds and perform quality gate checks using SonarQube.



SonarQube account and add sonar credentials to Jenkins

```
## SonarQube Configuration

1. Create Sonar cloud account on https://sonarcloud.io
2. Generate an Authentication token on SonarQube
    Account --> my account --> Security --> Generate Tokens

3. On Jenkins create credentials
   Manage Jenkins --> manage credentials --> system --> Global credentials --> add credentials
 - Credentials type: `Secret text`
 - ID: `sonarqube-key`

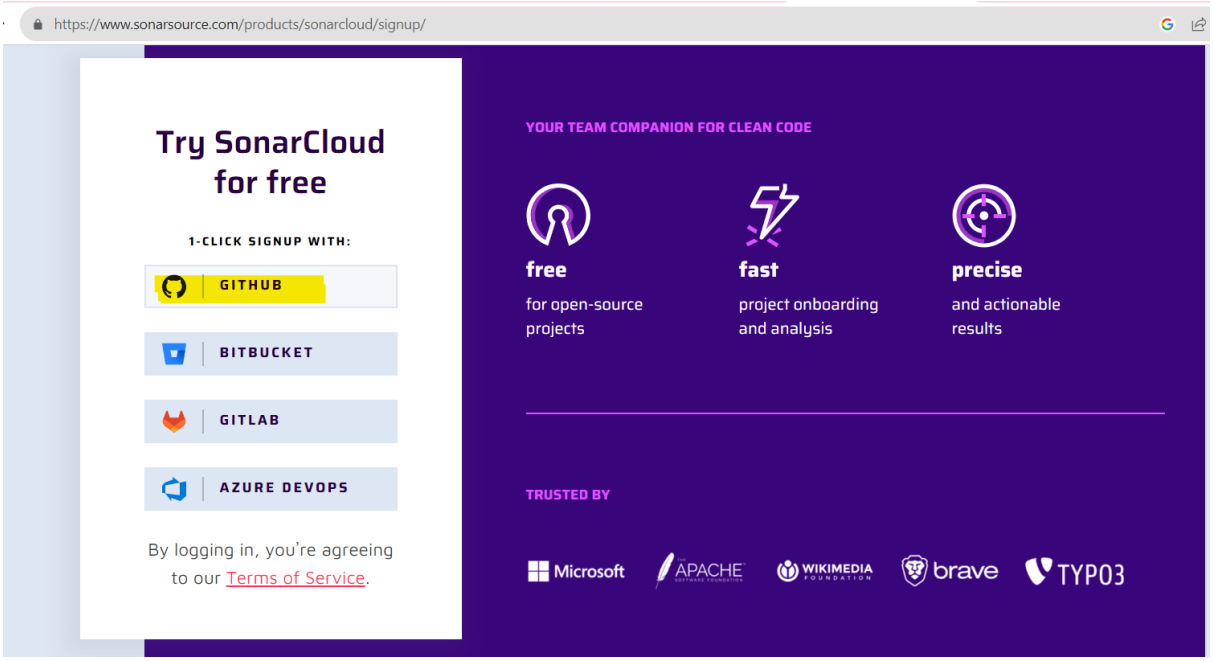
4. Install SonarQube plugin
    Manage Jenkins --> Available plugins
```

```
    Search for `sonarqube scanner`

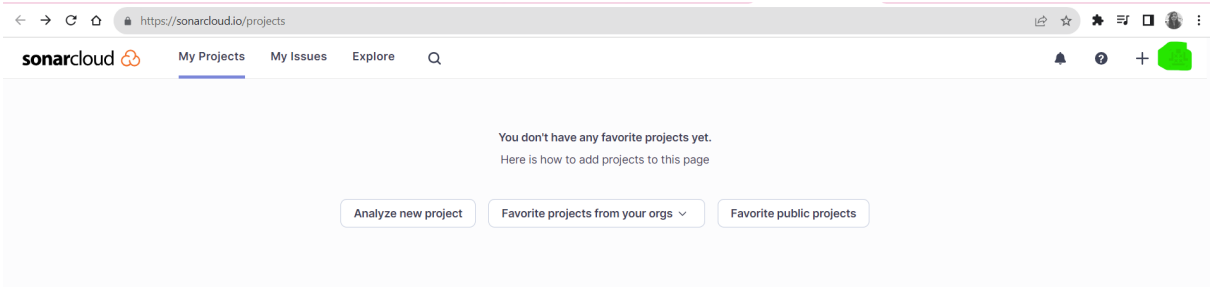
5. Configure sonarqube server
   Manage Jenkins --> Configure System --> sonarqube server
   Add Sonarqube server
   - Name: `sonar-server`
   - Server URL: `https://sonarcloud.io/`
   - Server authentication token: `sonarqube-key`

6. Configure sonarqube scanner
   Manage Jenkins --> Global Tool configuration --> Sonarqube scanner
   Add sonarqube scanner
   - Sonarqube scanner: `sonar-scanner`
```

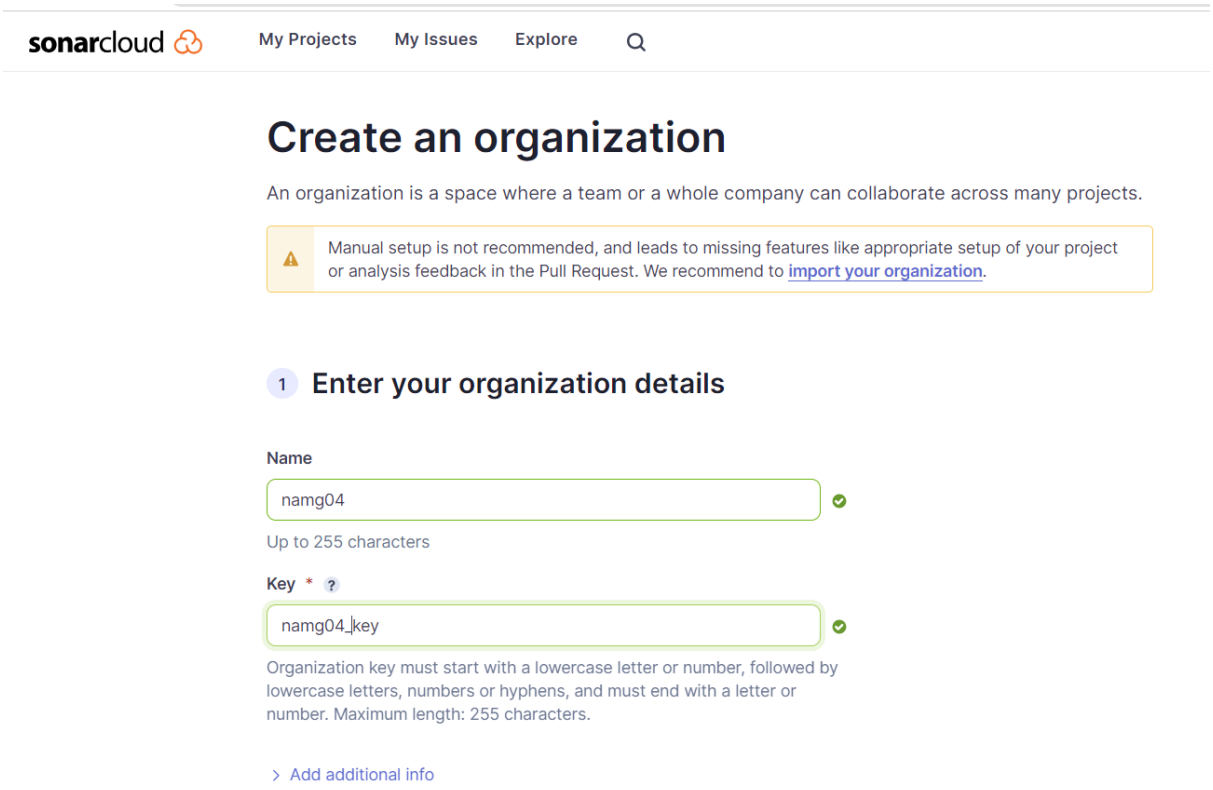
Login to Sonarcloud.io



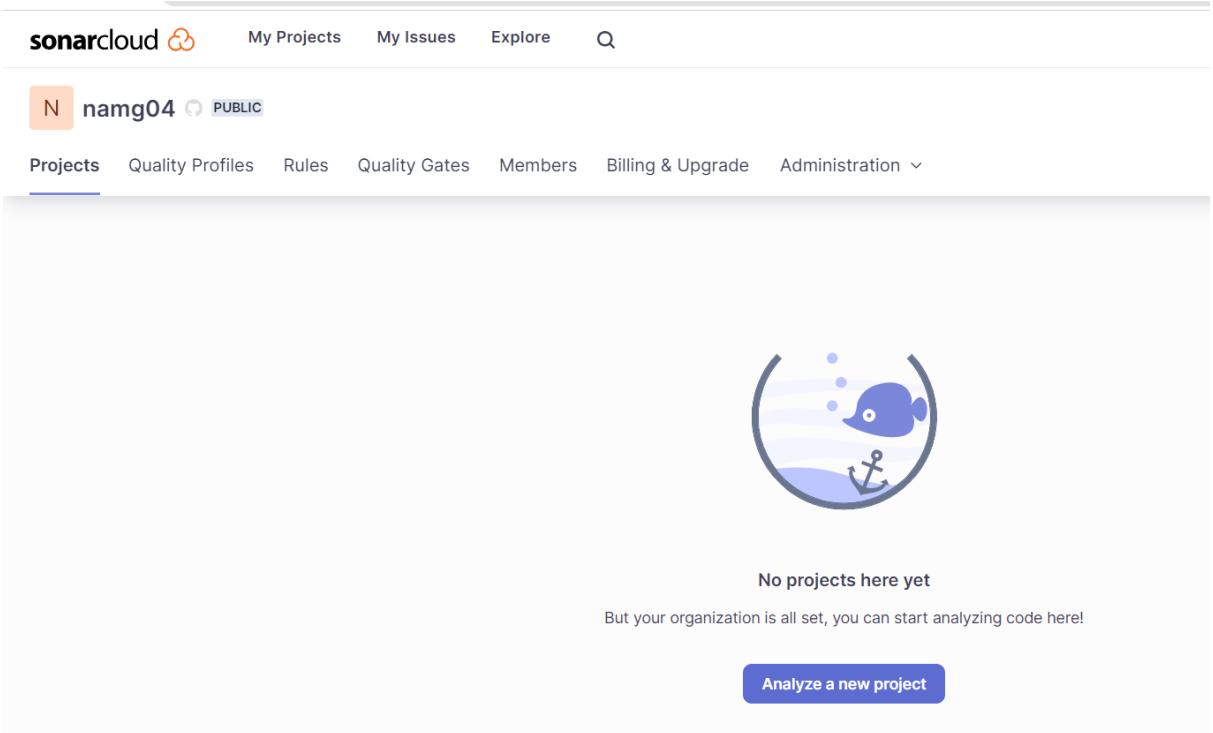
Go to Projects



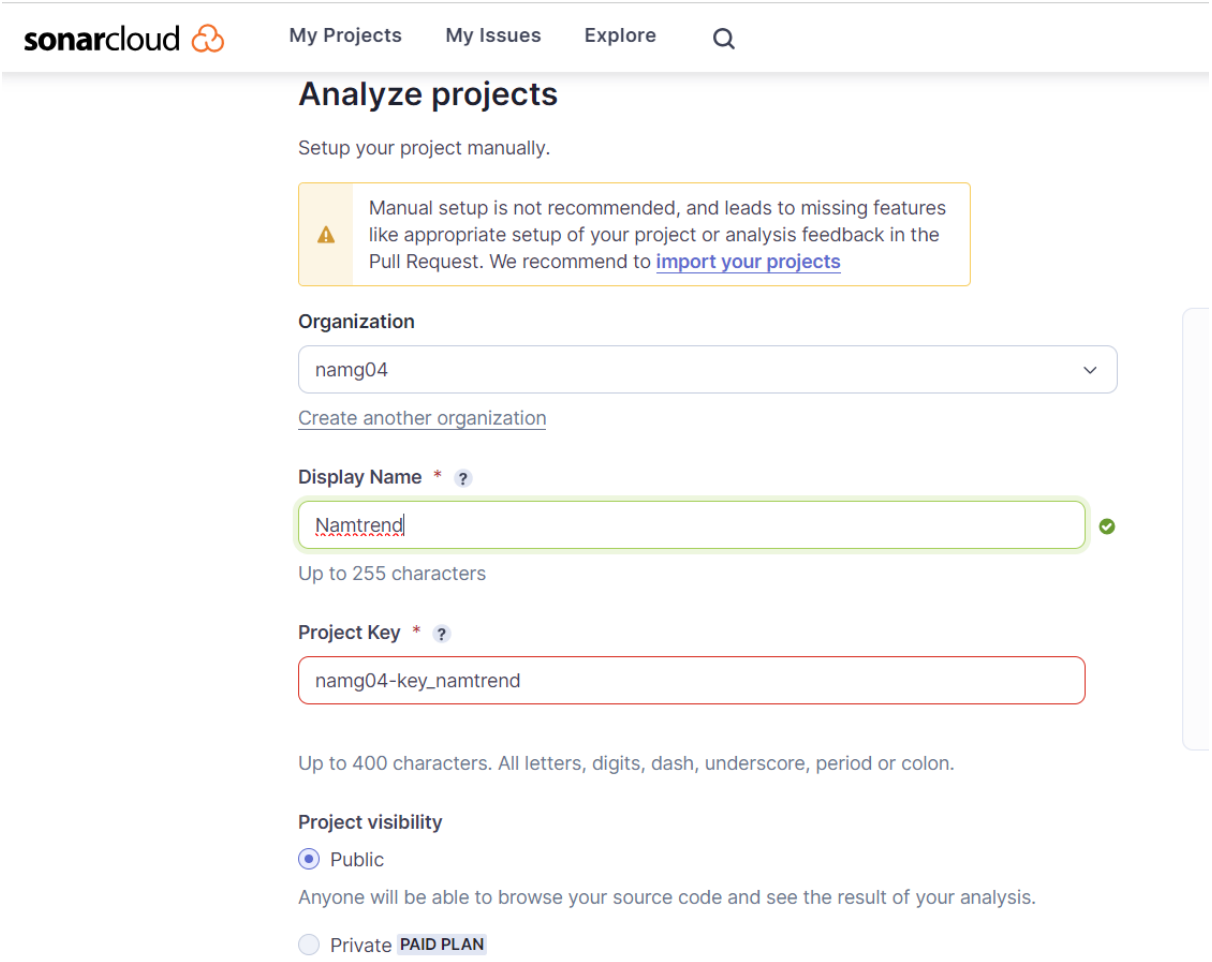
Create Organization



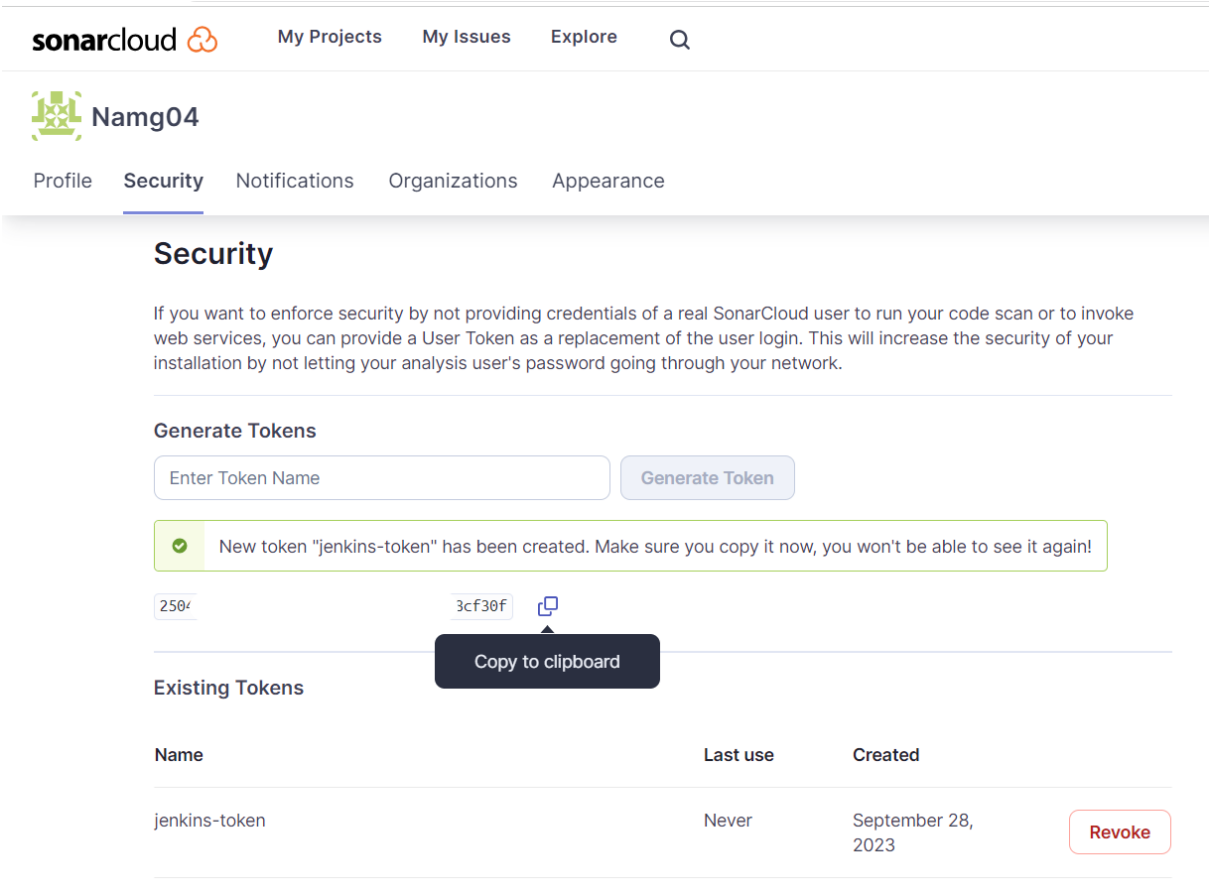
Choose a free plan and click on Analyze new project



Add Organization and project name

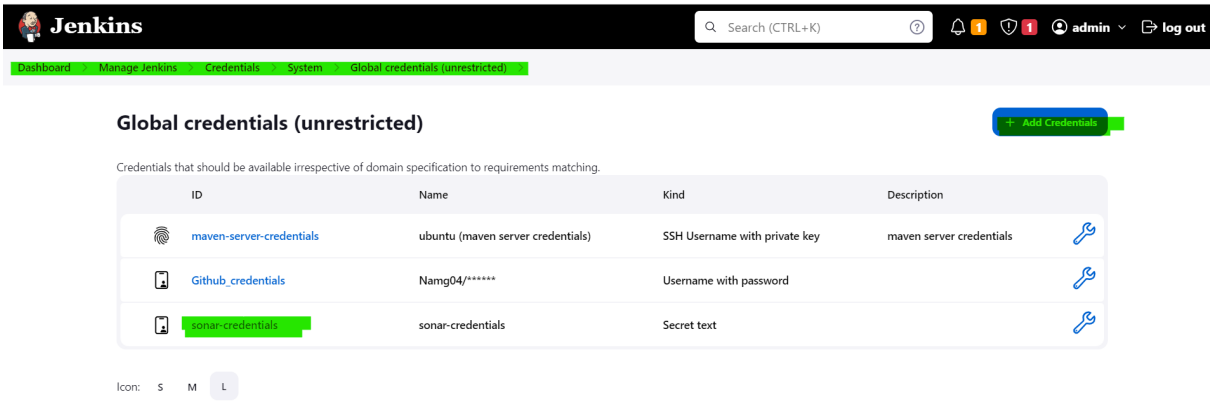


Generate Authentication token on sonar-qube



So this is the token which is generated.

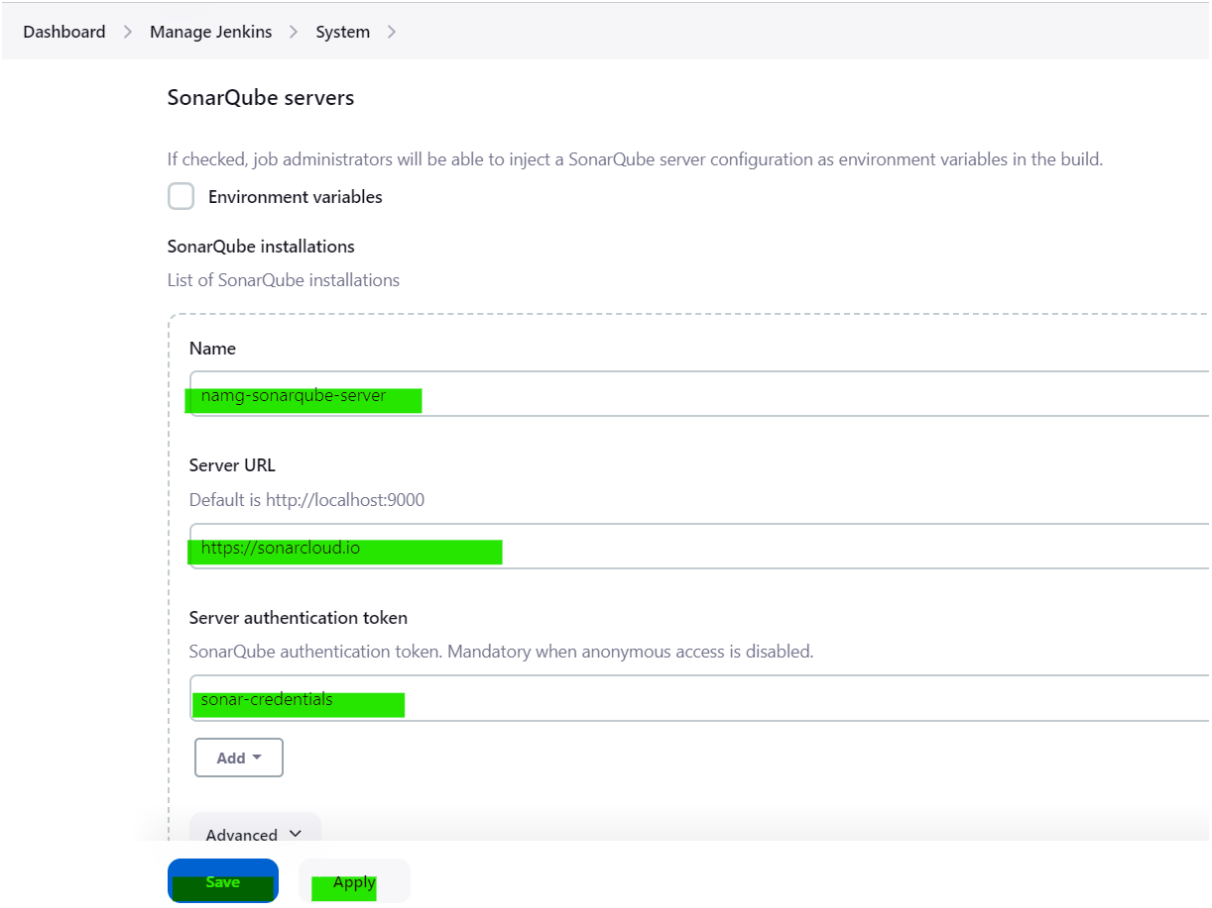
And if you see here new token Jenkins token has been created. Make sure that you copied it. You won't be able to see it again.



Download SonarQube plugings

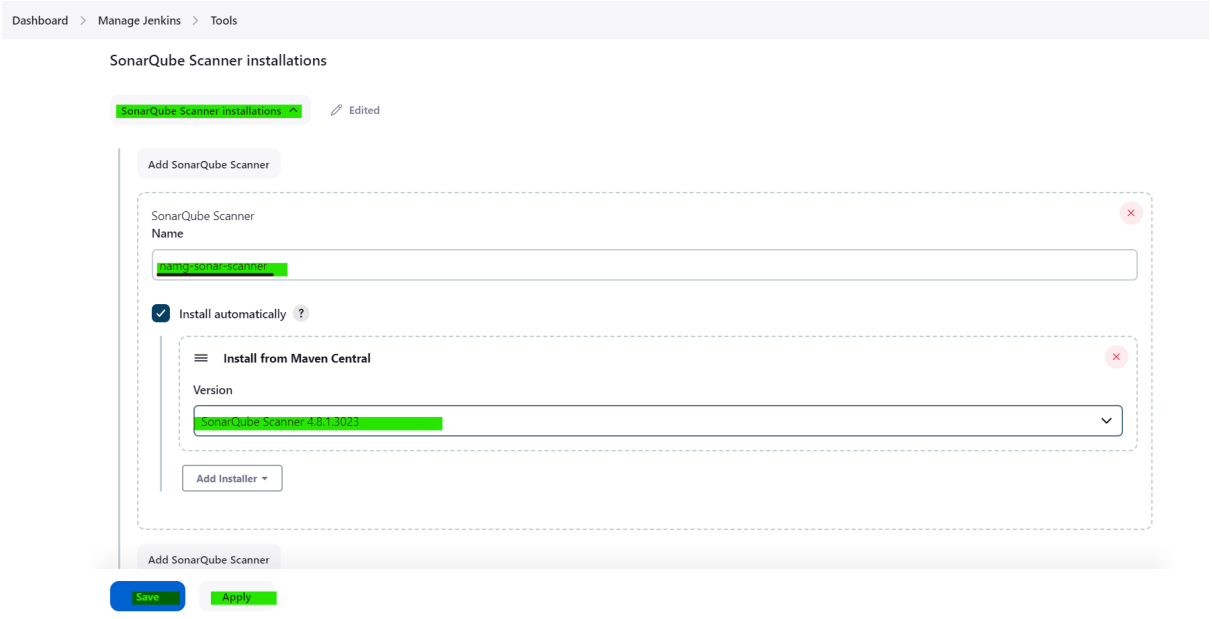
Jenkins>manage jenkins > plugins> Available plugins > sonarqube scanner

Add sonar qube server: manage jenkins > systems> SonarQube server

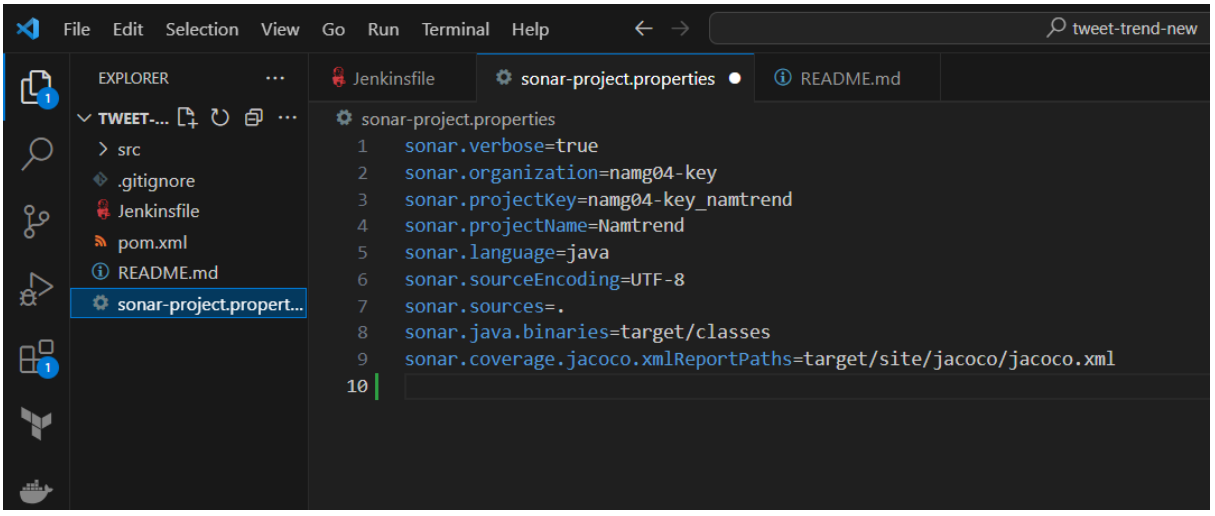


Add sonar qube scanner

Manage jenkins > Tools



write sonar-properties file



commit to git

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        sonar-project.properties

nothing added to commit but untracked files present (use "git add" to track)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add .

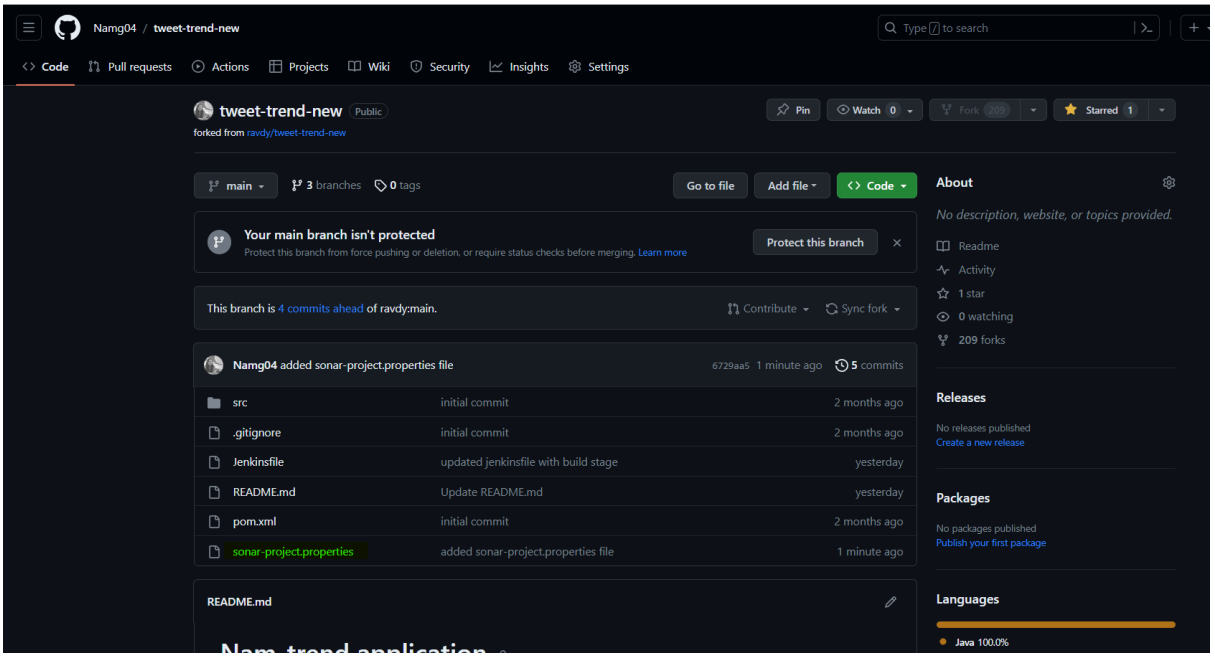
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "added sonar-project.properties file"
[main 6729aa5] added sonar-project.properties file
 1 file changed, 9 insertions(+)
 create mode 100644 sonar-project.properties

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 491 bytes | 491.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Namg04/tweet-trend-new.git
   9a97be8..6729aa5  main -> main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```



Add SonarQube stage in the Jenkinsfile.

Jenkinsfile | sonar-project.properties | README.md

```
Jenkinsfile
pipeline {
    agent {label 'maven'}


environment {
    PATH = "/opt/apache-maven-3.9.4/bin:$PATH"
}
    stages {
       stage("build"){
        steps {
            echo "------- build started --------"
            sh 'mvn clean deploy -Dmaven.test.skip=true'
            echo "---------build completed ----------"
        }

    }
    stage("test"){
        steps{
            echo "--------unit test started ---------"
            sh 'mvn surefire-report:report'
            echo "---------unit test completed -------"
        }
    }

    stage('SonarQube analysis') {
    environment{
      scannerHome = tool 'namg-sonar-scanner' //sonar scanner name should be same as what we have defined in the tools
    }

    steps {                                 // in the steps we are adding our sonar cube server that is with Sonar Cube environment.
    withSonarQubeEnv('namg-sonarqube-server') {

       sh "${scannerHome}/bin/sonar-scanner" // This is going to communicate with our sonar cube server and send the analysis report.
    }
    }
  }
}
```

commit the code into git

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git fetch

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Jenkinsfile

no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Jenkinsfile

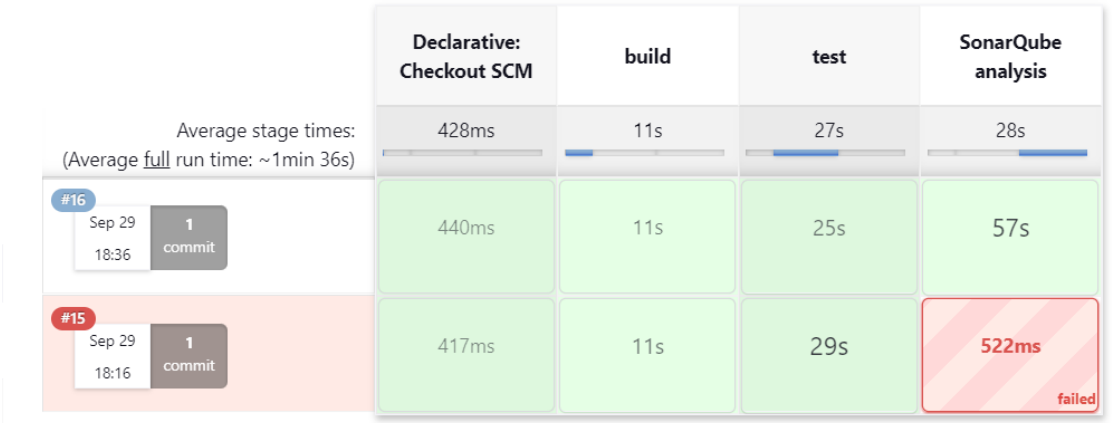

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "added test stage into jenkins file"
[main b3b38a0] added test stage into jenkins file
 1 file changed, 14 insertions(+), 4 deletions(-)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 724 bytes | 724.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Namg04/tweet-trend-new.git
   e28fc32..b3b38a0  main -> main

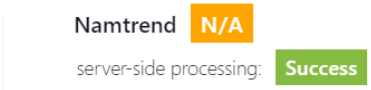
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

Build successfully done

Stage View

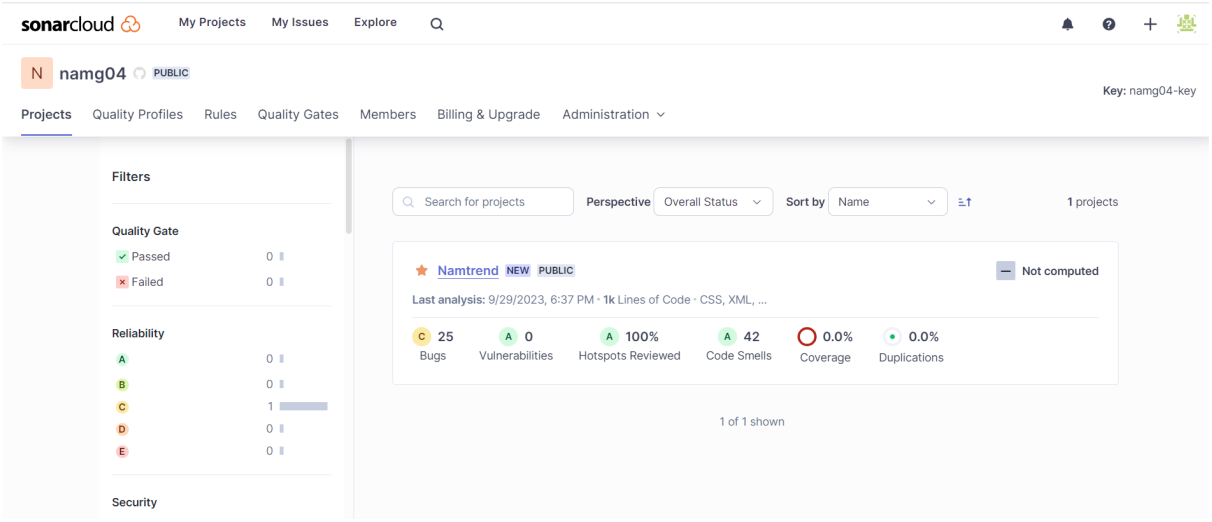


SonarQube Quality Gate

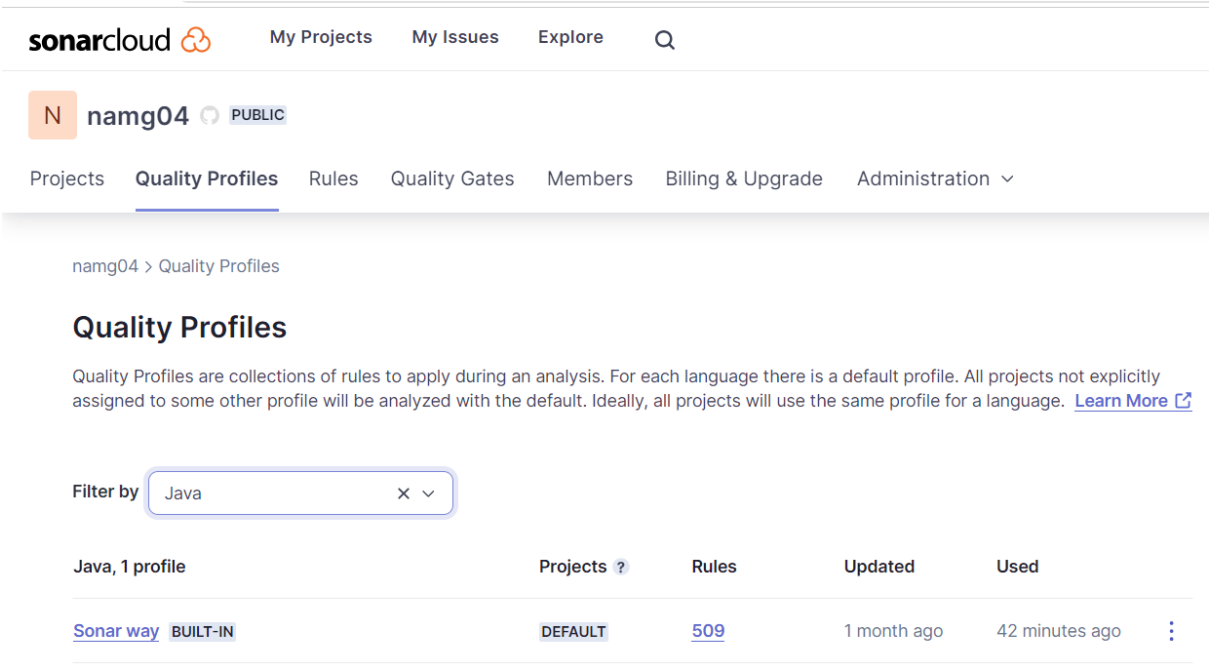


Permalinks

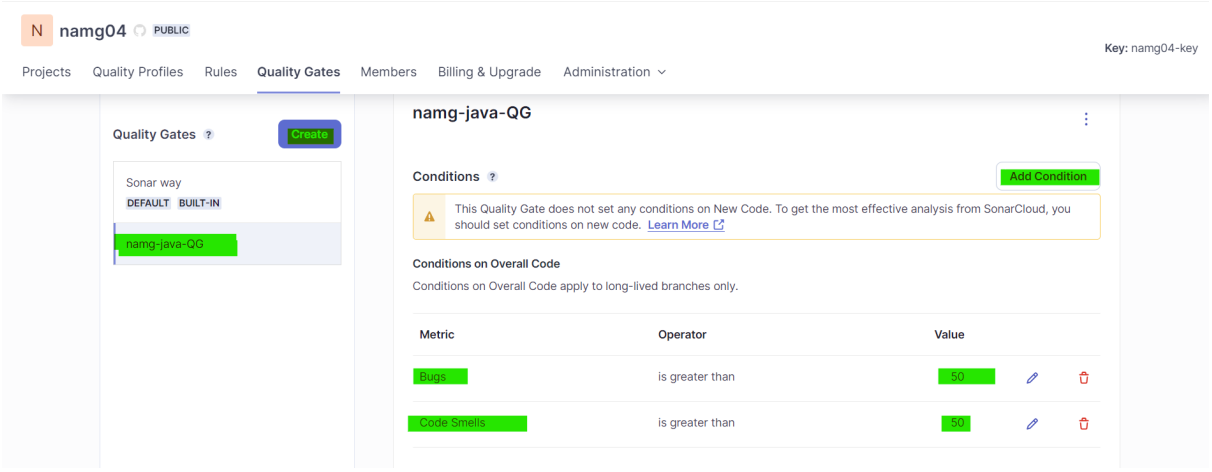
SonarQube Dashboard



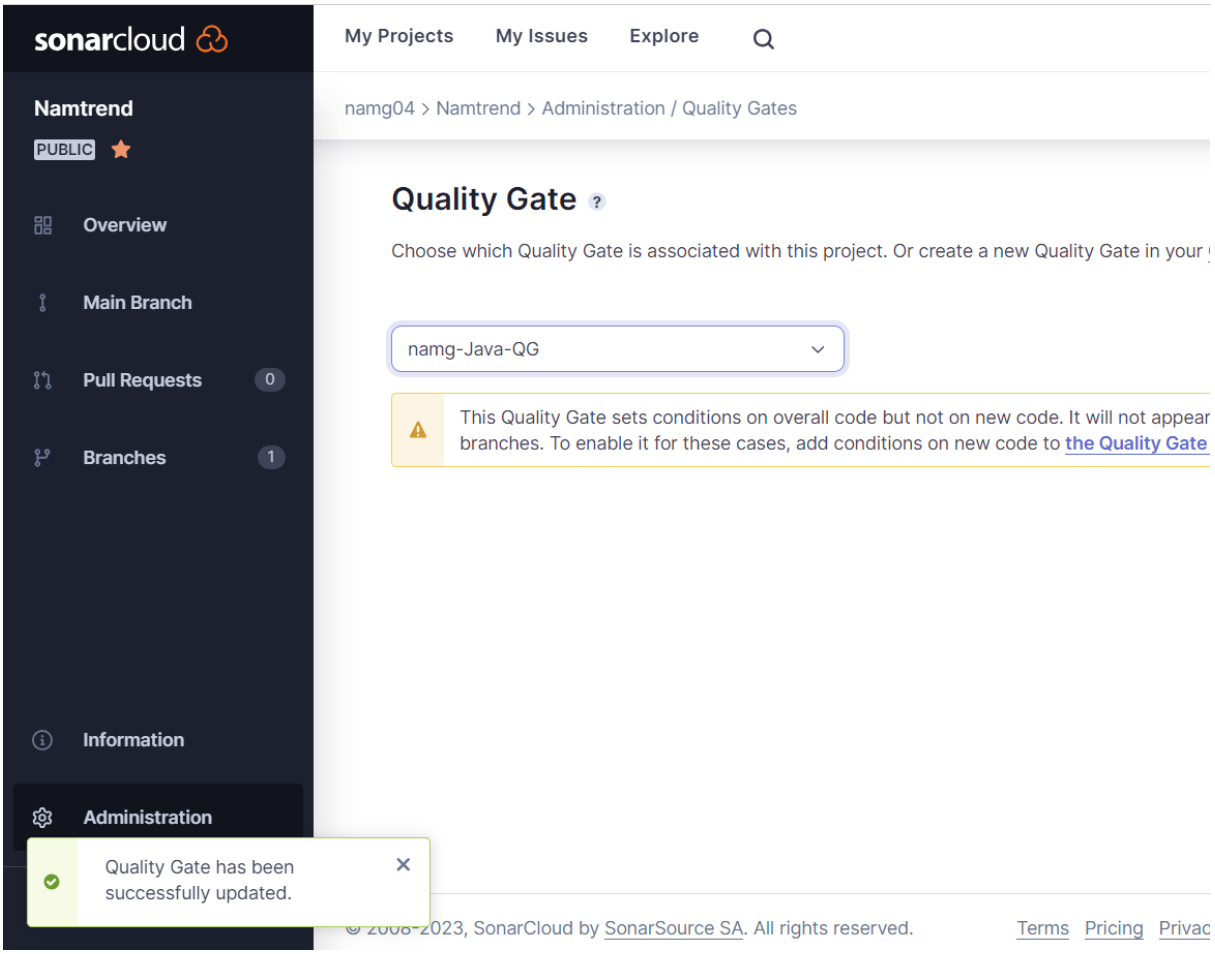
Add Quality gates



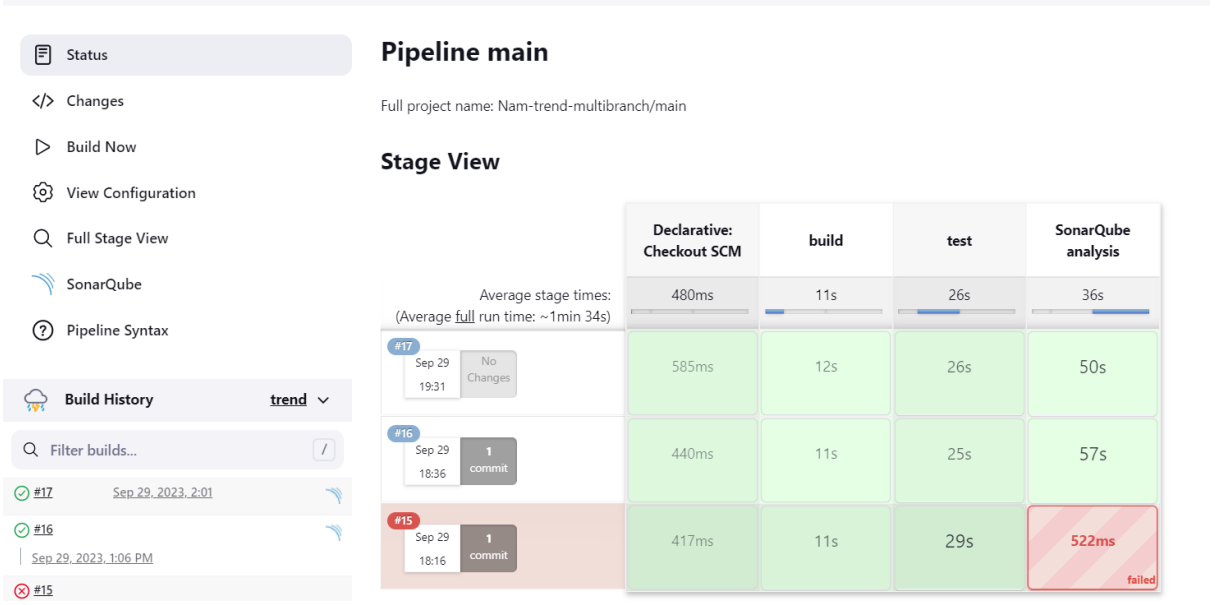
Add conditions I have added quality gates for bugs and code smell and set the values as well. If both cross the values my build will get fail.



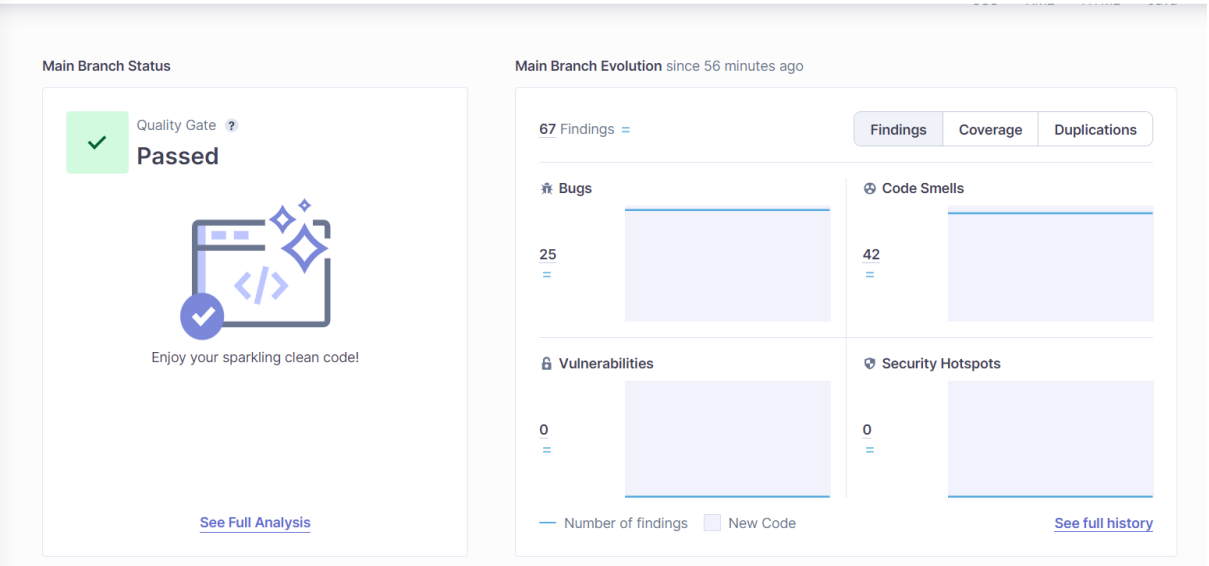
Go to Projects > Administration> Quality Gates and add the created QG.



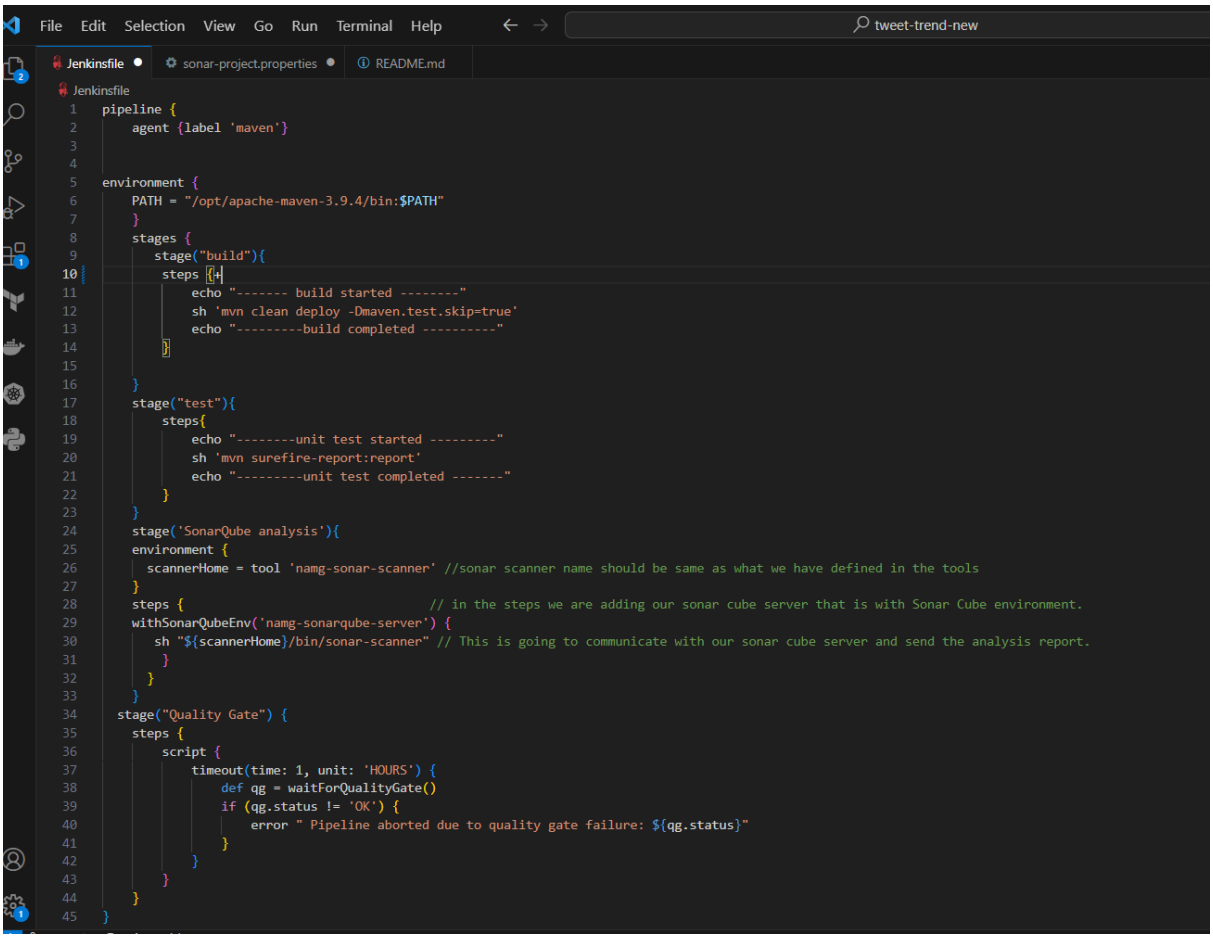
Run Build now



Quality Gate has been passed since the bug is less than 50.



Added sonar gate in Jenkinsfile



Commit the code in Git

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git fetch
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 935 bytes | 44.00 KiB/s, done.
From https://github.com/Namg04/tweet-trend-new
   b3b38a0..33af32f  main       -> origin/main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git pull
hint: Waiting for your editor to close the file...       1 [sig] bash 2019! sigpacket::proces
s: Suppressing signal 18 to win32 process (pid 21072)
                                                    228812 [sig] bash 2019! sigpacket::proc
ess: Suppressing signal 18 to win32 process (pid 21072)
                                                   1647017 [sig] bash 2019! sigpacket::pr
ocess: Suppressing signal 18 to win32 process (pid 21072)
                                                   2775067 [sig] bash 2019! sigpacket::
Merge made by the 'ort' strategy.
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

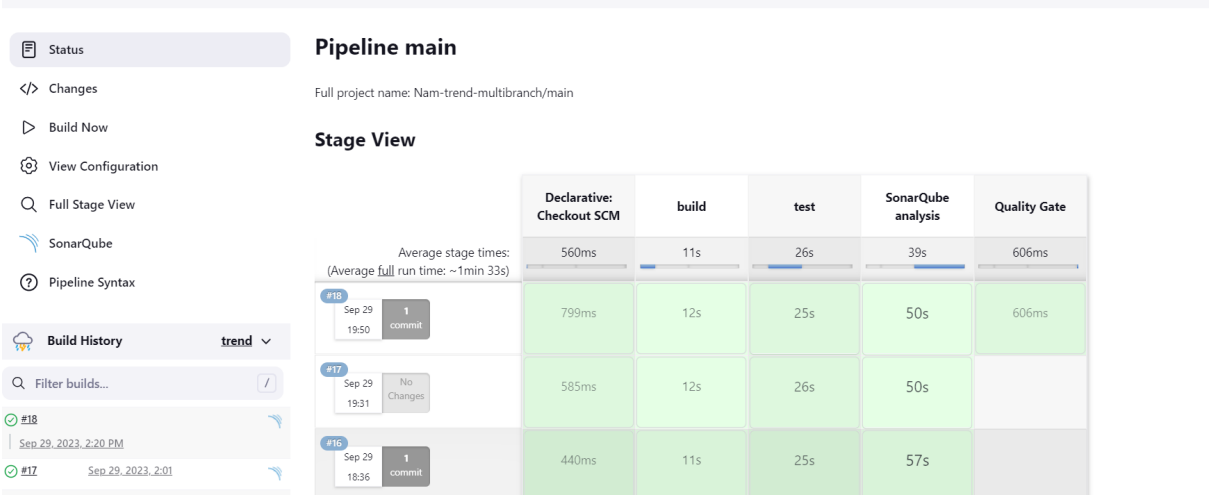
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "added sonar-gate stage to Jenkinsfile"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

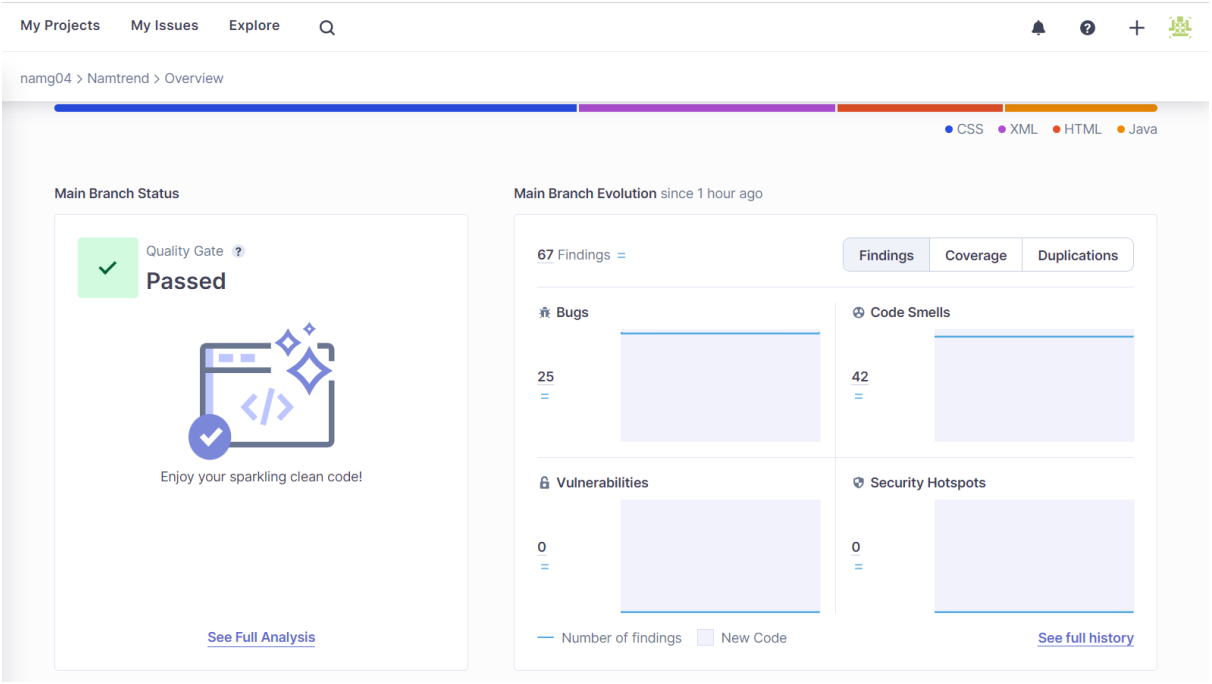
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 809 bytes | 809.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/Namg04/tweet-trend-new.git
   33af32f..f8b3c92  main -> main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

Build triggered automatically



Sonar Gate test passes successfully

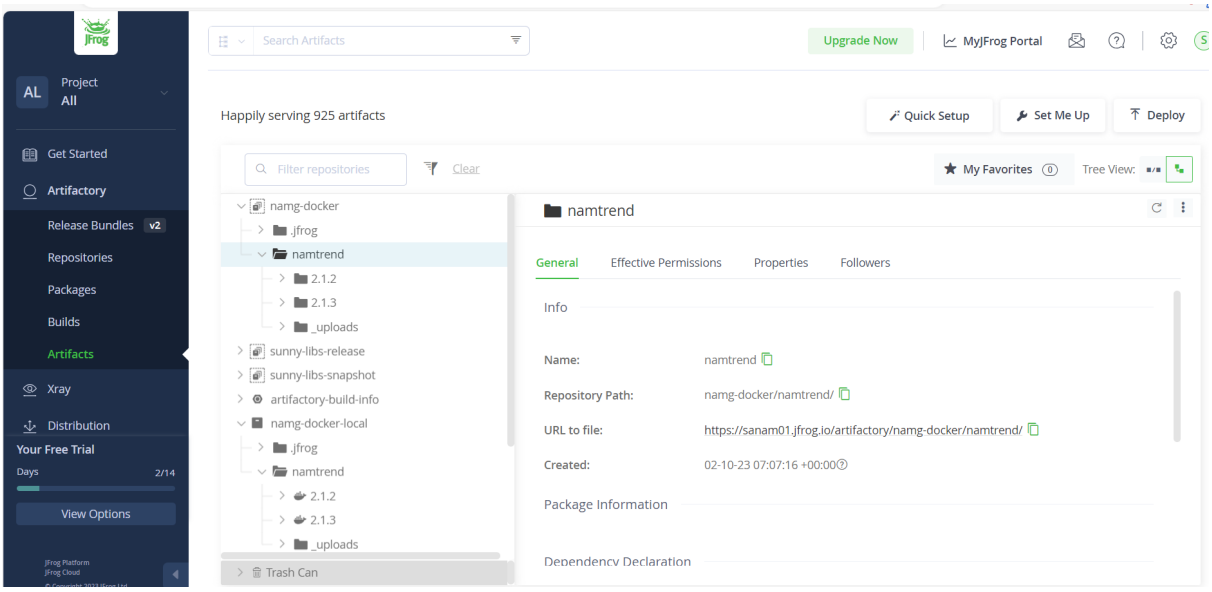


==========

Jfrog Artifactory Integration with Jenkins

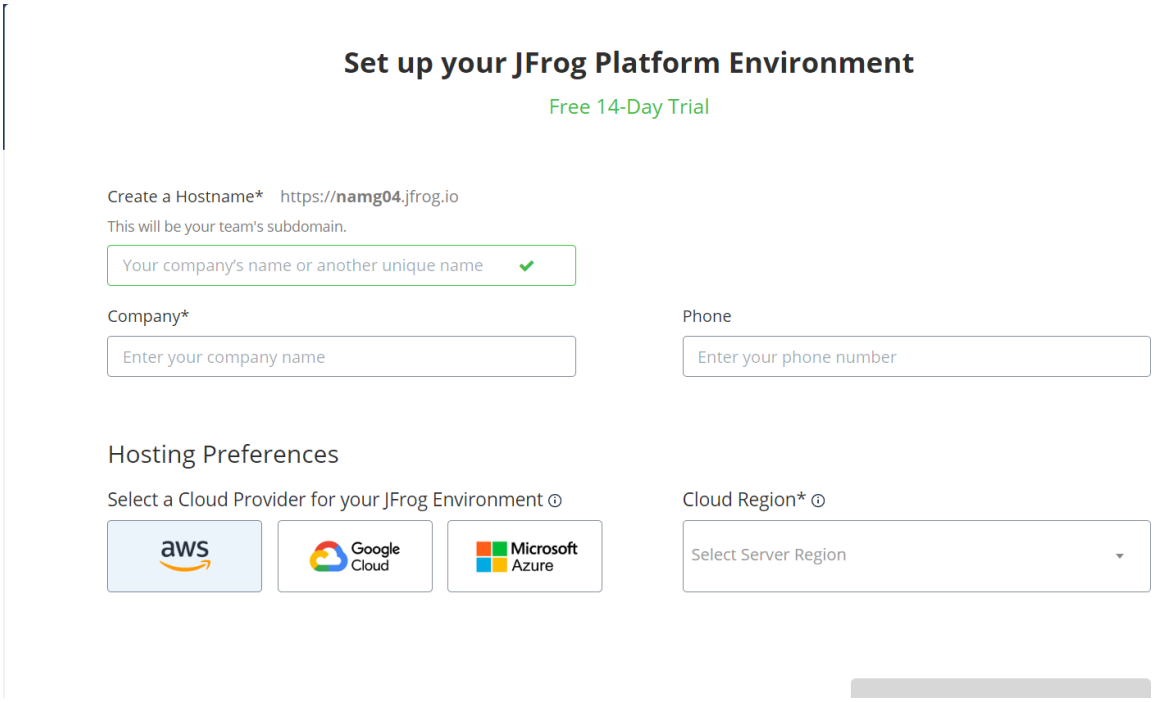
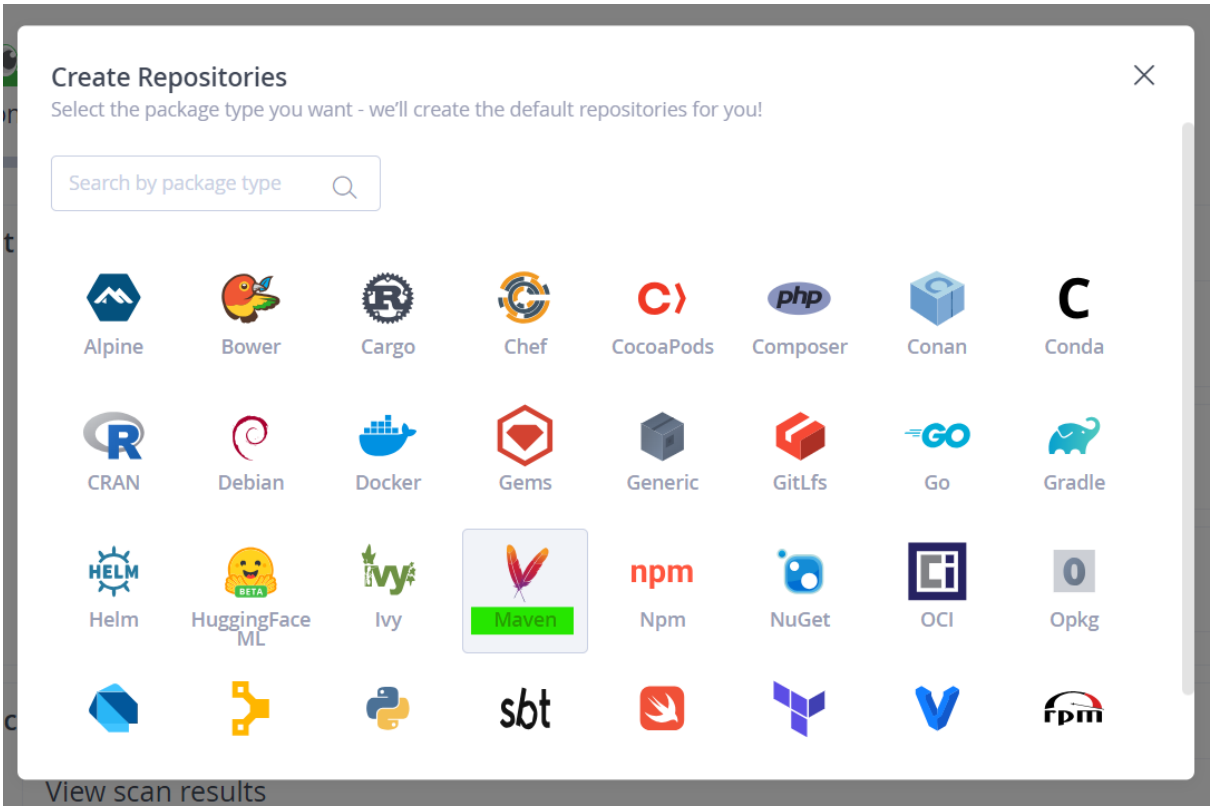
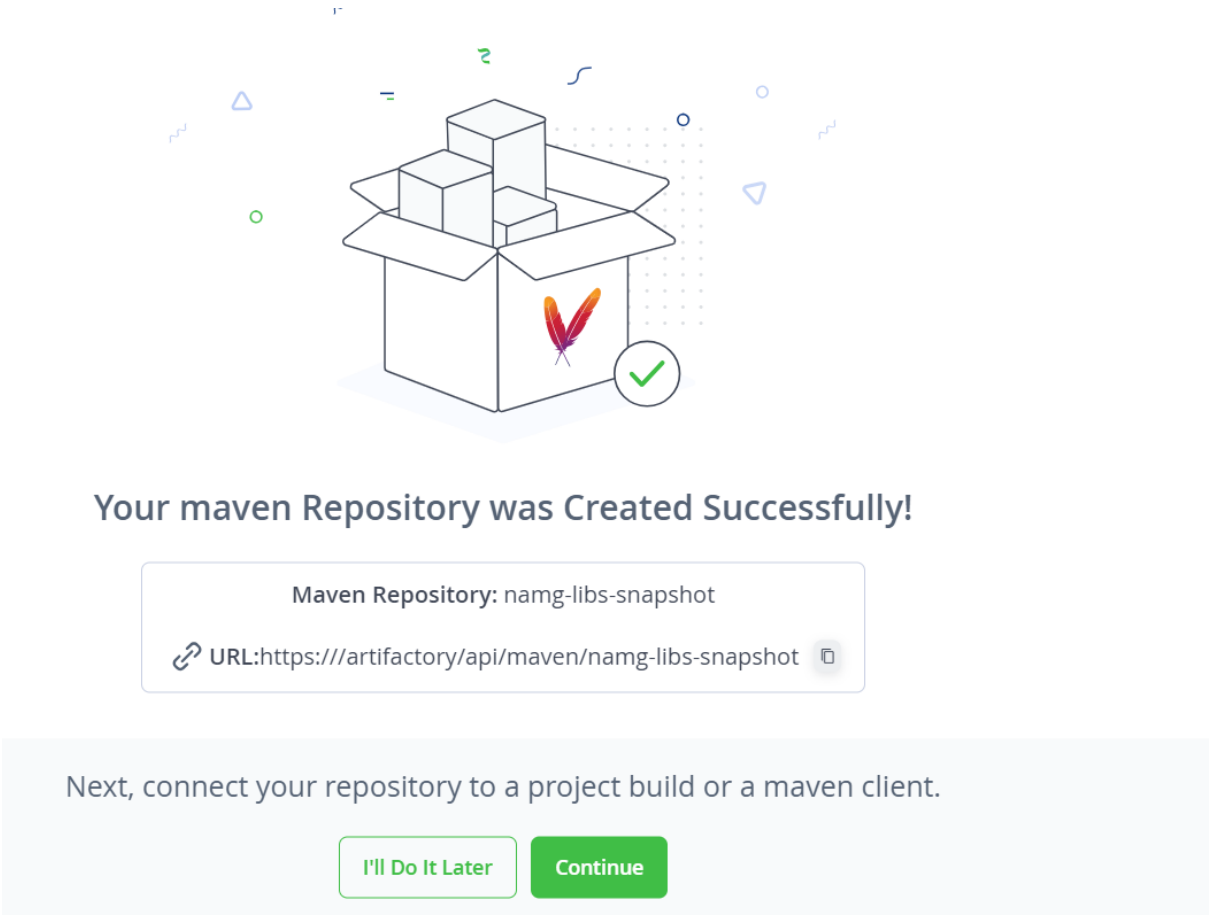
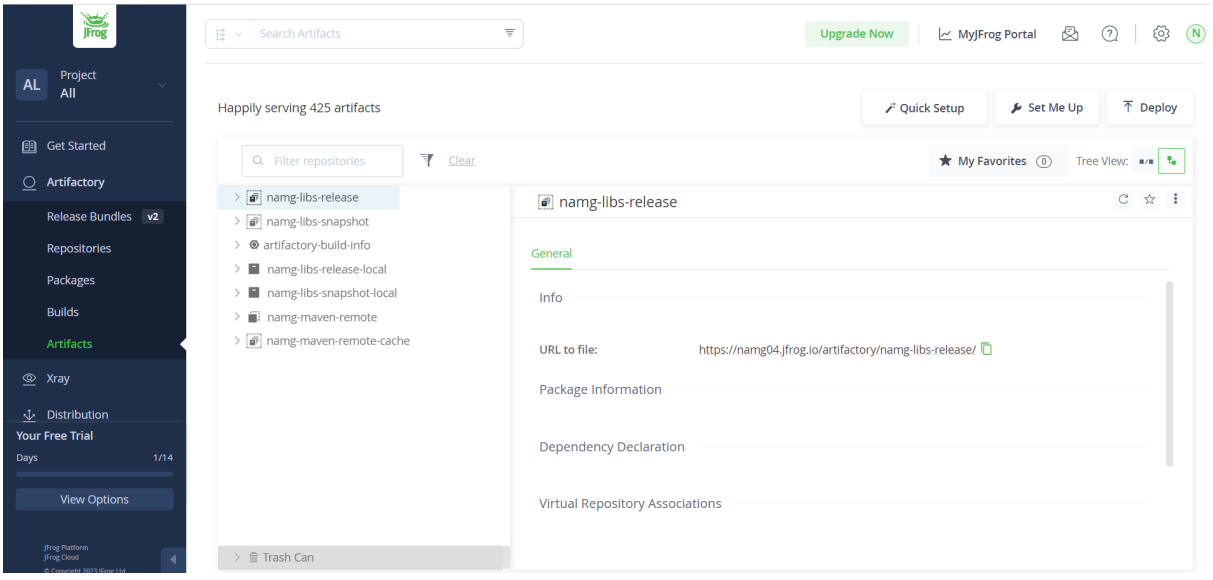
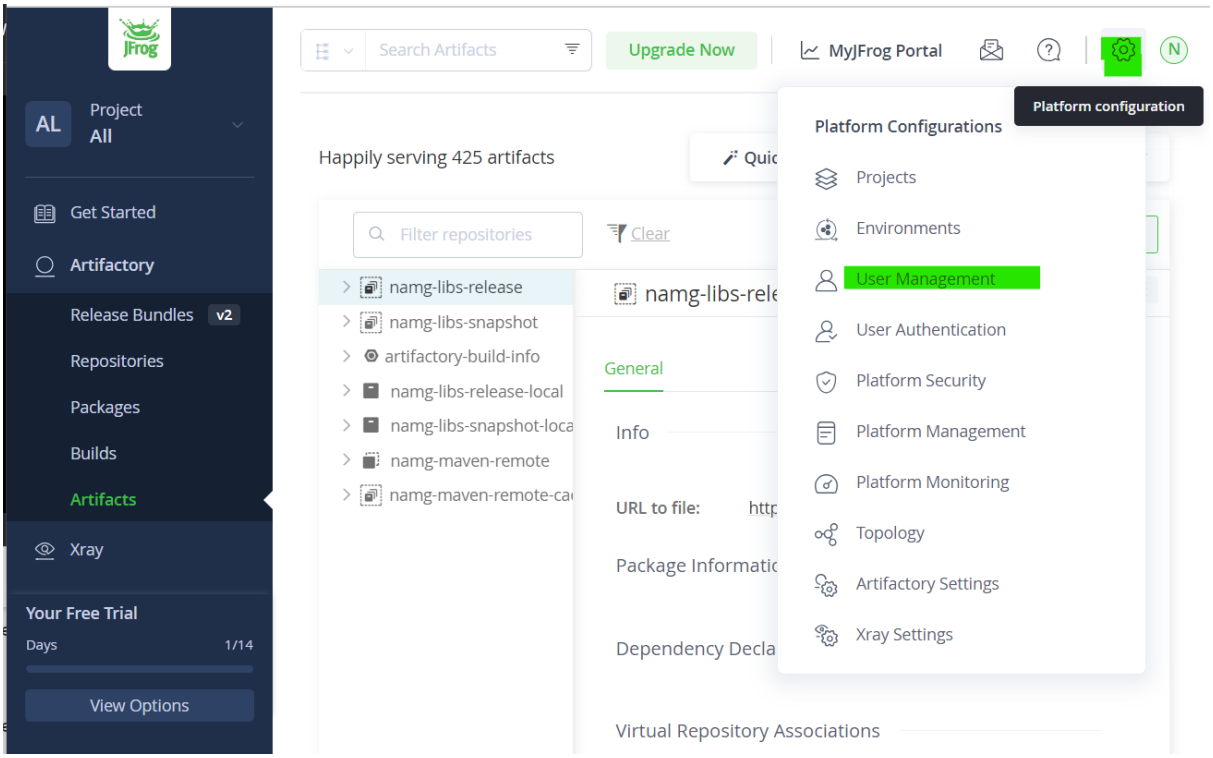
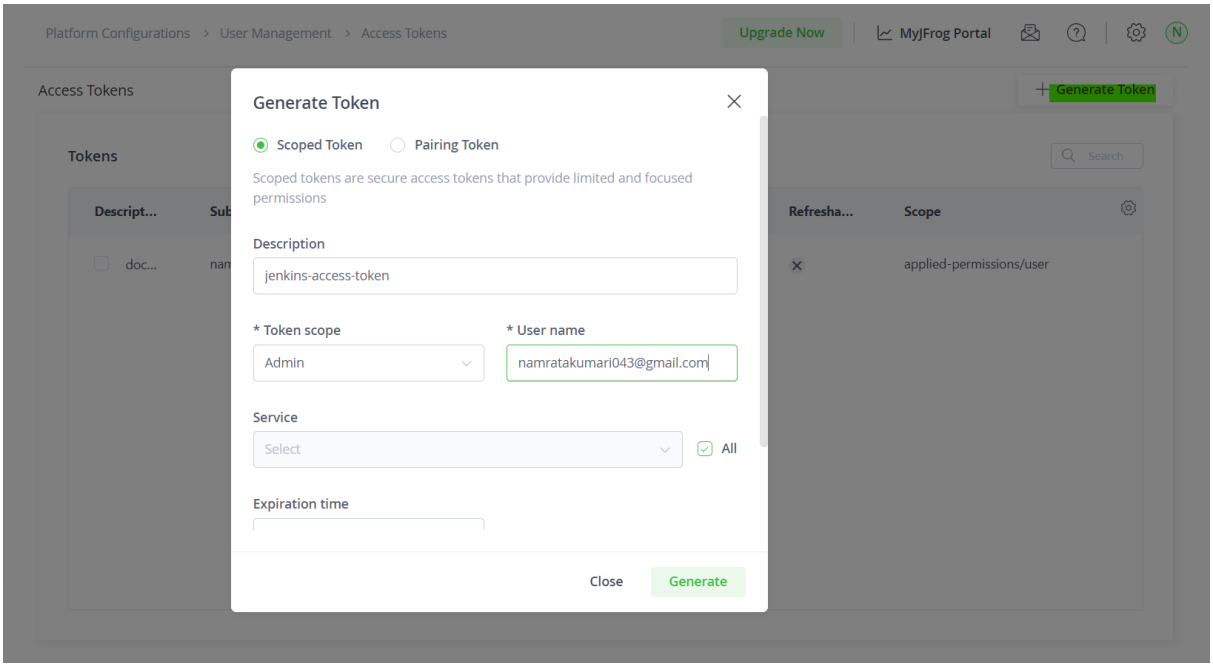
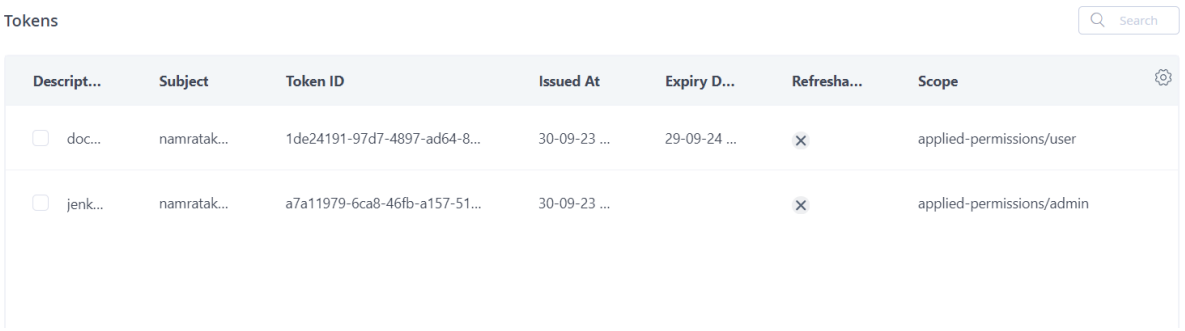
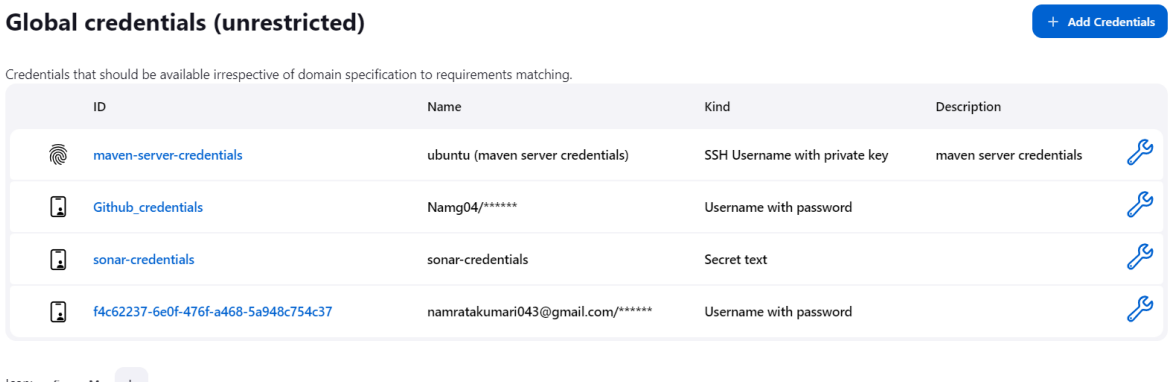
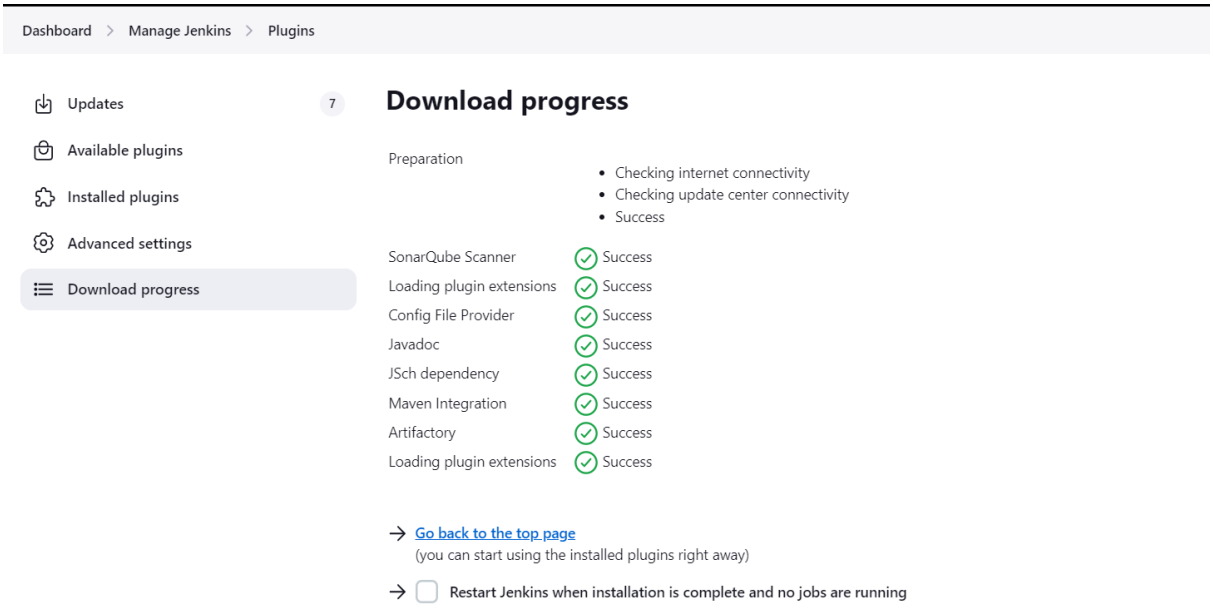
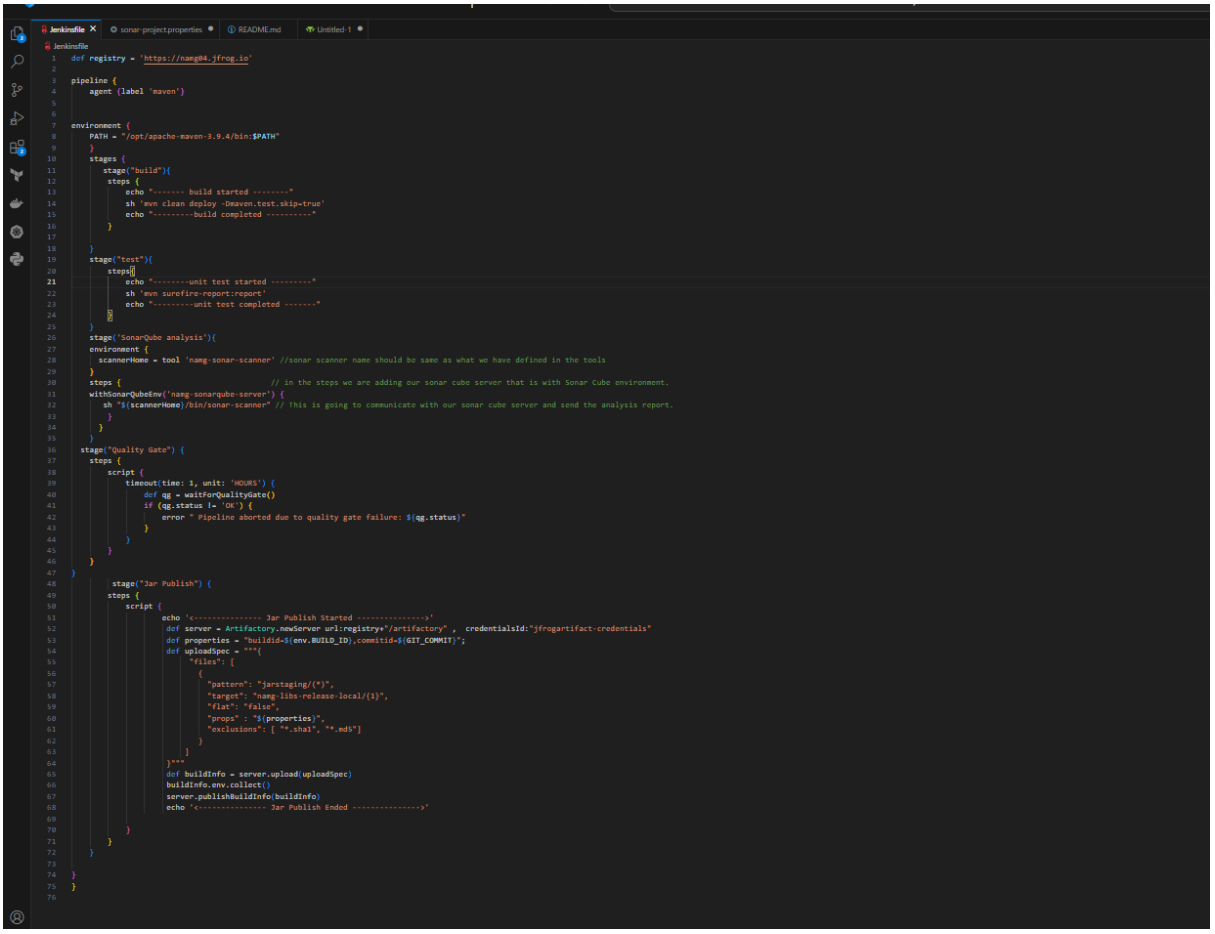
So far we have seen how to integrate GitHub with Jenkins, Maven Server with Jenkins, SonarQube with Jenkins. In this section we are going to see how to integrate Jfrog Artifactory with Jenkins.

This document provides a step-by-step guide on integrating Jfrog Artifactory with Jenkins. The process includes creating an Artifactory account, generating an access token, adding credentials in Jenkins, installing the Artifactory plugin, updating the Jenkinsfile, creating a Dockerfile, and publishing a Docker image on Artifactory. The document also includes screenshots illustrating the process.




```
## Publish jar file onto Jfrog Artifactory
```

```
1. Create Artifactory account
2. Generate an access token  with username  (username must be your email id)
3. Add username and password under jenkins credentials
4. Install Artifactory plugin
5. Update Jenkinsfile with jar publish stage
    ```sh
         def registry = 'https://valaxy01.jfrog.io'
             stage("Jar Publish") {
            steps {
                script {
                        echo '<-------------- Jar Publish Started -------------->'
                         def server = Artifactory.newServer url:registry+"/artifactory" ,  credentialsId:"artifactory_token"
                         def properties = "buildid=${env.BUILD_ID},commitid=${GIT_COMMIT}";
                         def uploadSpec = """{
                              "files": [
                                {
                                  "pattern": "jarstaging/(*)",
                                  "target": "libs-release-local/{1}",
                                  "flat": "false",
                                  "props" : "${properties}",
                                  "exclusions": [ "*.sha1", "*.md5"]
                                }
                             ]
                         }"""
                         def buildInfo = server.upload(uploadSpec)
                         buildInfo.env.collect()
                         server.publishBuildInfo(buildInfo)
                         echo '<-------------- Jar Publish Ended -------------->'

                }
            }
        }
    ```

Check-point:
Ensure below are update
1. your jfrog account details in place of `https://valaxy01.jfrog.io` in the defination of registry `def registry = 'https://valaxy01.j
2. Credentials id in the place of `jfrogforjenkins` in the `def server = Artifactory.newServer url:registry+"/artifactory" , credenti
3. Maven repository name in the place of `libs-release-local` in the `"target": "ttrend-libs-release-local/{1}",`
```

- **Create an Artifactory Account.**
- **Generate an access token with a username.**
- **Add username and Password under Jenkins Credentials.**
- **Install the Artifactory plugin.**
- **Update Jenkinsfile with jar publish stage.**
- **Create a Dockerfile.**
- **Create and publish a docker image on Artifactory.**
- **Login to jfrog.com create an account over there and set up your platform environment on the cloud.**



Create Maven repository







Create access token



Generate token



Token generated



Add credentials in Jenkins



Now install 'Artifactory' plugins



Update Jenkinsfile with jar publish stage.

source code path:

jfrog Artifactory url: https://sanam01.jfrog.io
Artifact location: /home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop/2.1.2

Credentials: jfrogartifact-credentials

```
Maven-Server
Terminal  Sessions  View  X server  Tools  Games  Settings  Macros  Help
Session  Servers  Tools  Games  Sessions  View  Split  MultiExec  Tunneling  Packages  Settings  Help  X server  Exit
6. ANSIBLE  8. Maven-Server  10. Jenkins-Master
ubuntu@ip-10-1-1-59:~$ sudo su
root@ip-10-1-1-59:/home/ubuntu# ls
jenkins  maven.txt
root@ip-10-1-1-59:/home/ubuntu# cd jenkins/
root@ip-10-1-1-59:/home/ubuntu/jenkins# ls
caches  remoting  remoting.jar  tools  workspace
root@ip-10-1-1-59:/home/ubuntu/jenkins# cd workspace/
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace# ls
Nam-trend-job      Nam-trend-multibranch_dev      Nam-trend-multibranch_main      Nam-trend-multibranch_stage      test-job
Nam-trend-job@tmp  Nam-trend-multibranch_dev@tmp  Nam-trend-multibranch_main@tmp  Nam-trend-multibranch_stage@tmp  workspaces.txt
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace# cd Nam-trend-multibranch_main
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main# ls
Jenkinsfile  README.md  jarstaging  pom.xml  sonar-project.properties  src  target
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main# cd jarstaging
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging# ls
com
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging# cd com
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com# ls
valaxy
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com# cd valaxy/
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy# ls
demo-workshop
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy# cd demo-workshop/
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop# ls
2.1.2  maven-metadata.xml  maven-metadata.xml.md5  maven-metadata.xml.sha1
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop# cd 2.1.2/
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop/2.1.2# ls
demo-workshop-2.1.2.jar  demo-workshop-2.1.2.jar.md5  demo-workshop-2.1.2.jar.sha1  demo-workshop-2.1.2.pom  demo-workshop-2.1.2.pom.md5  demo-workshop-2.
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop/2.1.2#
```

Jenkinsfile | sonar-project.properties | README.md | Untitled-1

```
1
2   //collect all information to mentioned in the jenkinsfile
3   //artifact URL
4   //Artifact location from Jenkins-slave server(Maven-server)
5
6   jfrog Artfactory URL: https://namg04.jfrog.io
7   Artifact location: /home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop/2.1.2
8   Credentials: jfrogartifact-credentials
9
10  ************************************************************************************************************************
11      def registry = 'https://namg04.jfrog.io'
12          stage("Jar Publish") {
13          steps {
14              script {
15                      echo '<-------------- Jar Publish Started -------------->'
16                      def server = Artifactory.newServer url:registry+"/artifactory" ,  credentialsId:"jfrogartifact-credentials"
17                      def properties = "buildid=${env.BUILD_ID},commitid=${GIT_COMMIT}";
18                      def uploadSpec = """{
19                           "files": [
20                             {
21                               "pattern": "jarstaging/(*)",
22                               "target": "libs-release-local/{1}",
23                               "flat": "false",
24                               "props" : "${properties}",
25                               "exclusions": [ "*.sha1", "*.md5"]
26                             }
27                           ]
28                      }"""
29                      def buildInfo = server.upload(uploadSpec)
30                      buildInfo.env.collect()
31                      server.publishBuildInfo(buildInfo)
32                      echo '<-------------- Jar Publish Ended -------------->'
33
34              }
35          }
36      }
```

Updated Jenkinsfile (update target path carefully)



```
Jenkinsfile
def registry = 'https://namg04.jfrog.io'

pipeline {
    agent {label 'maven'}


environment {
    PATH = "/opt/apache-maven-3.9.4/bin:$PATH"
    }
    stages {
      stage("build"){
       steps {
          echo "------- build started --------"
          sh 'mvn clean deploy -Dmaven.test.skip=true'
          echo "---------build completed ---------"
       }

    }
    stage("test"){
        steps{
          echo "--------unit test started ---------"
          sh 'mvn surefire-report:report'
          echo "---------unit test completed -------"
        }
    }
    stage('SonarQube analysis'){
    environment {
     scannerHome = tool 'namg-sonar-scanner' //sonar scanner name should be same as what we have defined in the tools
    }
    steps {                                 // in the steps we are adding our sonar cube server that is with Sonar Cube environment.
    withSonarQubeEnv('namg-sonarqube-server') {
       sh "${scannerHome}/bin/sonar-scanner" // This is going to communicate with our sonar cube server and send the analysis report.
       }
     }
    }
  stage("Quality Gate") {
    steps {
        script {
            timeout(time: 1, unit: 'HOURS') {
                def qg = waitForQualityGate()
                if (qg.status != 'OK') {
                    error " Pipeline aborted due to quality gate failure: ${qg.status}"
                }
            }
        }
    }
}
        stage("Jar Publish") {
        steps {
            script {
                    echo '<-------------- Jar Publish Started -------------->'
                    def server = Artifactory.newServer url:registry+"/artifactory" ,  credentialsId:"jfrogartifact-credentials"
                    def properties = "buildid=${env.BUILD_ID},commitid=${GIT_COMMIT}";
                    def uploadSpec = """{
                         "files": [
                            {
                              "pattern": "jarstaging/(*)",
                              "target": "namg-libs-release-local/{1}",
                              "flat": "false",
                              "props" : "${properties}",
                              "exclusions": [ "*.sha1", "*.md5"]
                            }
                         ]
                    }"""
                    def buildInfo = server.upload(uploadSpec)
                    buildInfo.env.collect()
                    server.publishBuildInfo(buildInfo)
                    echo '<-------------- Jar Publish Ended -------------->'

            }
        }
    }

}
}
```

Committed code into Git

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git fetch

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Jenkinsfile

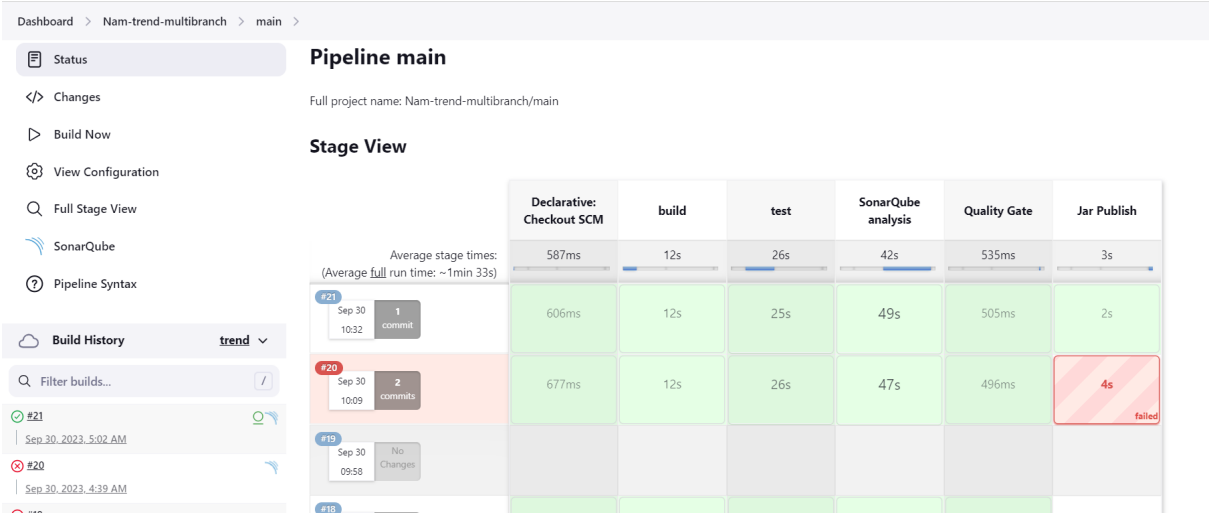
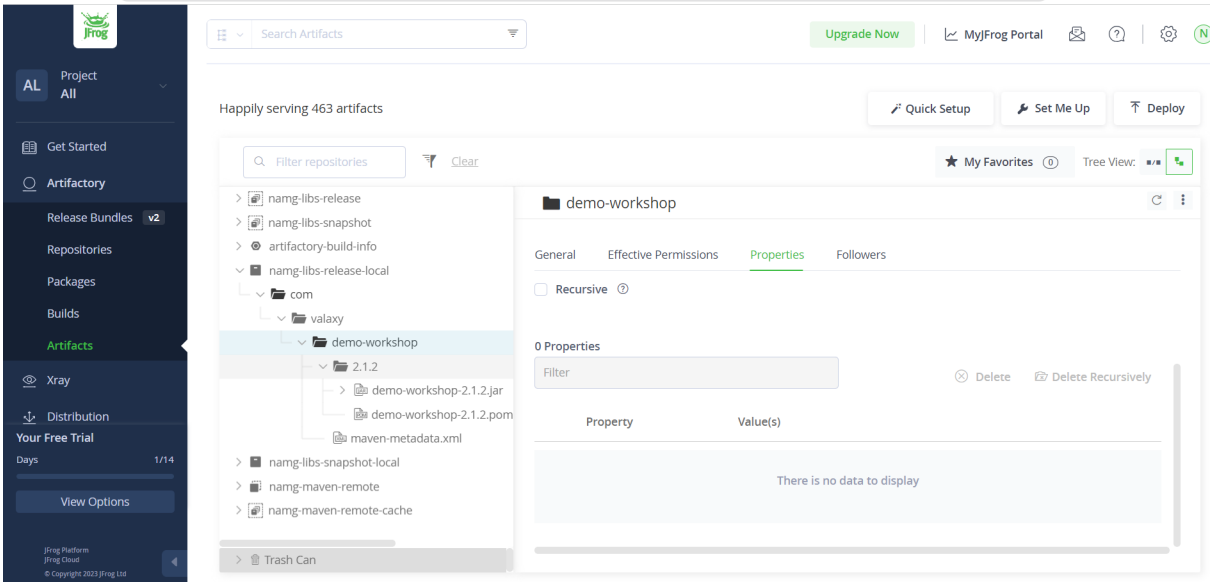
no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

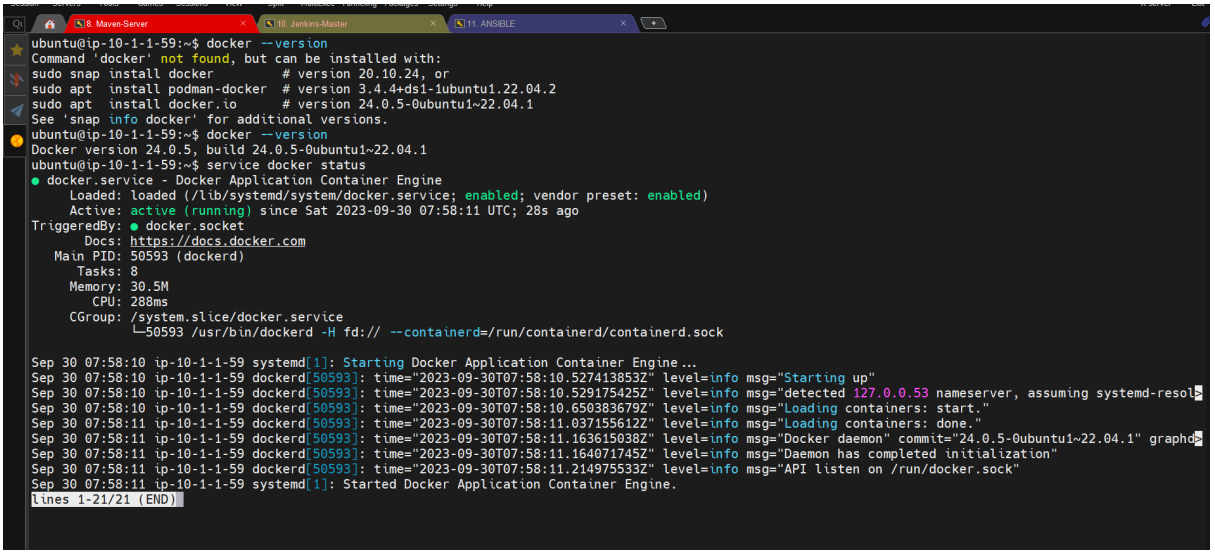
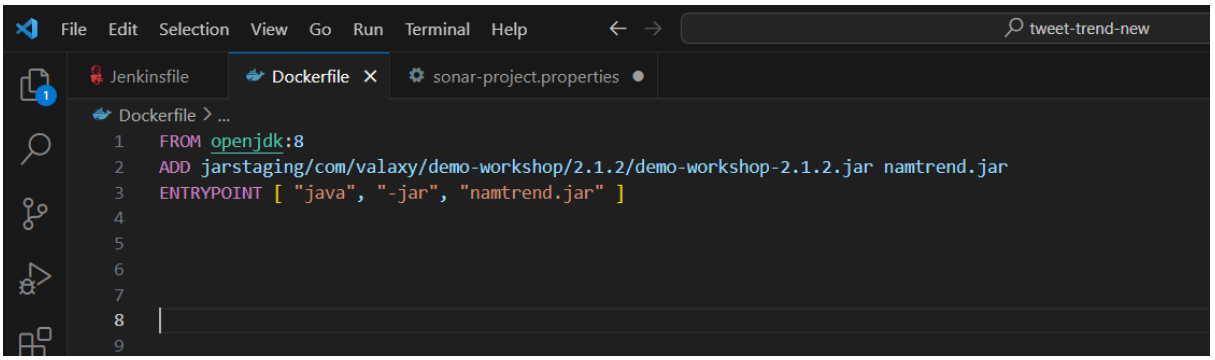
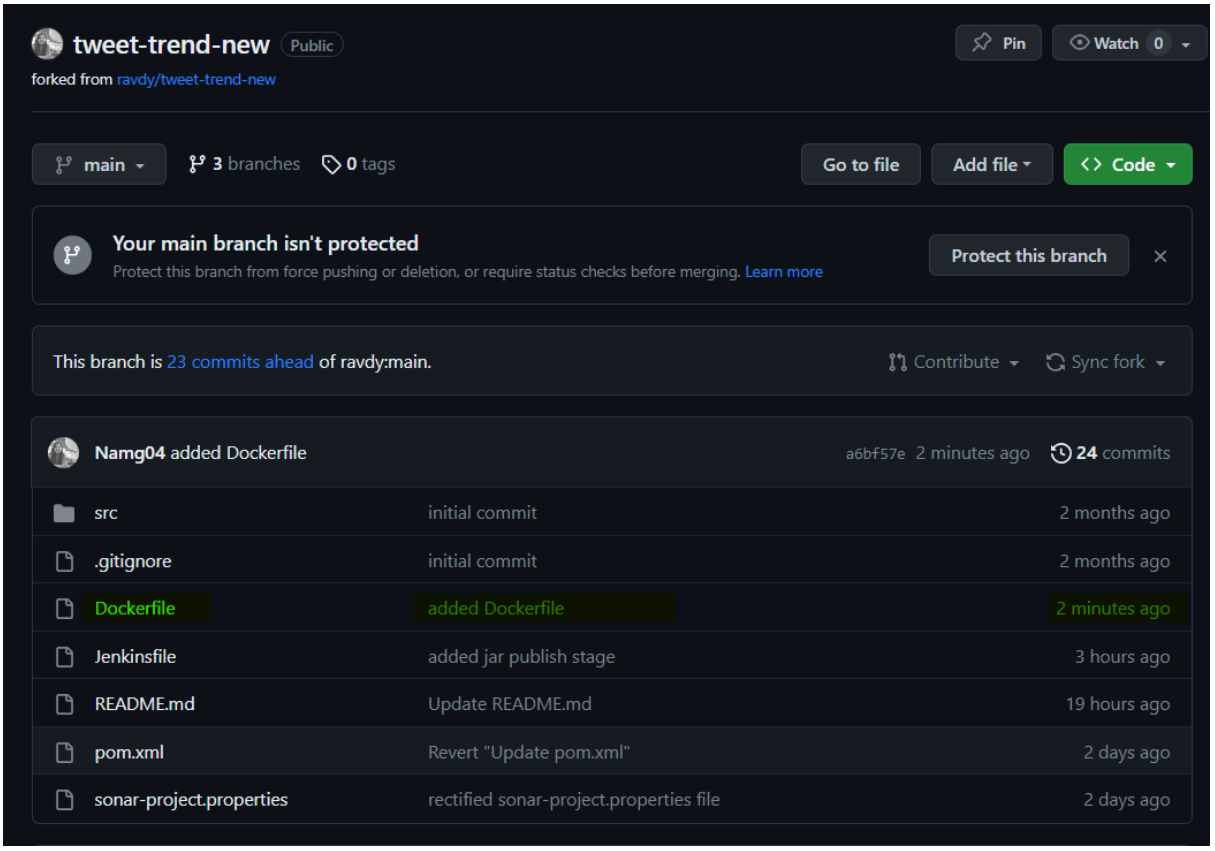
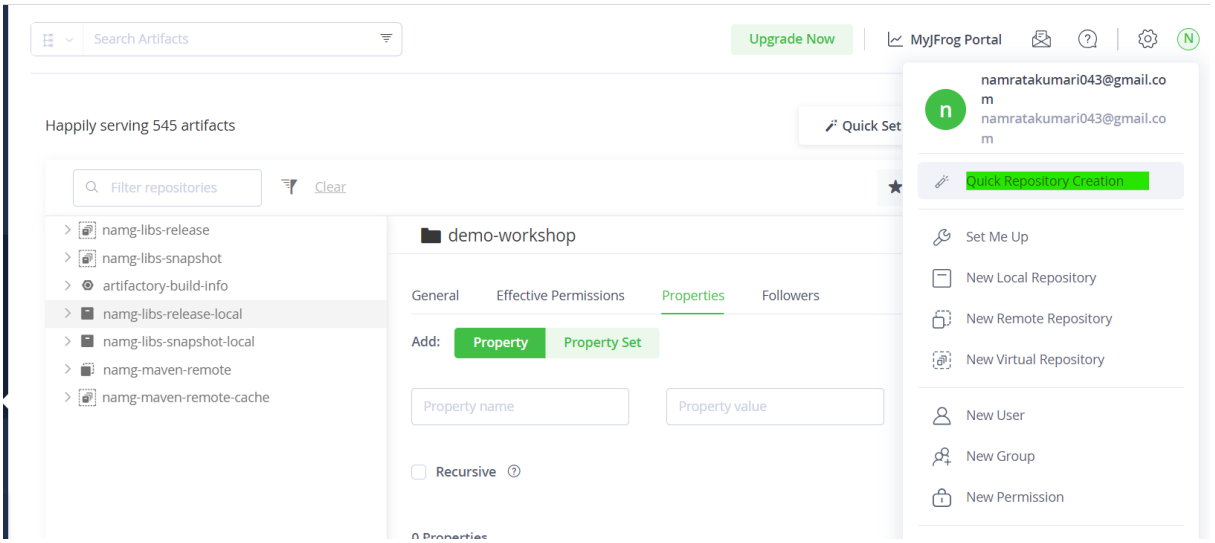
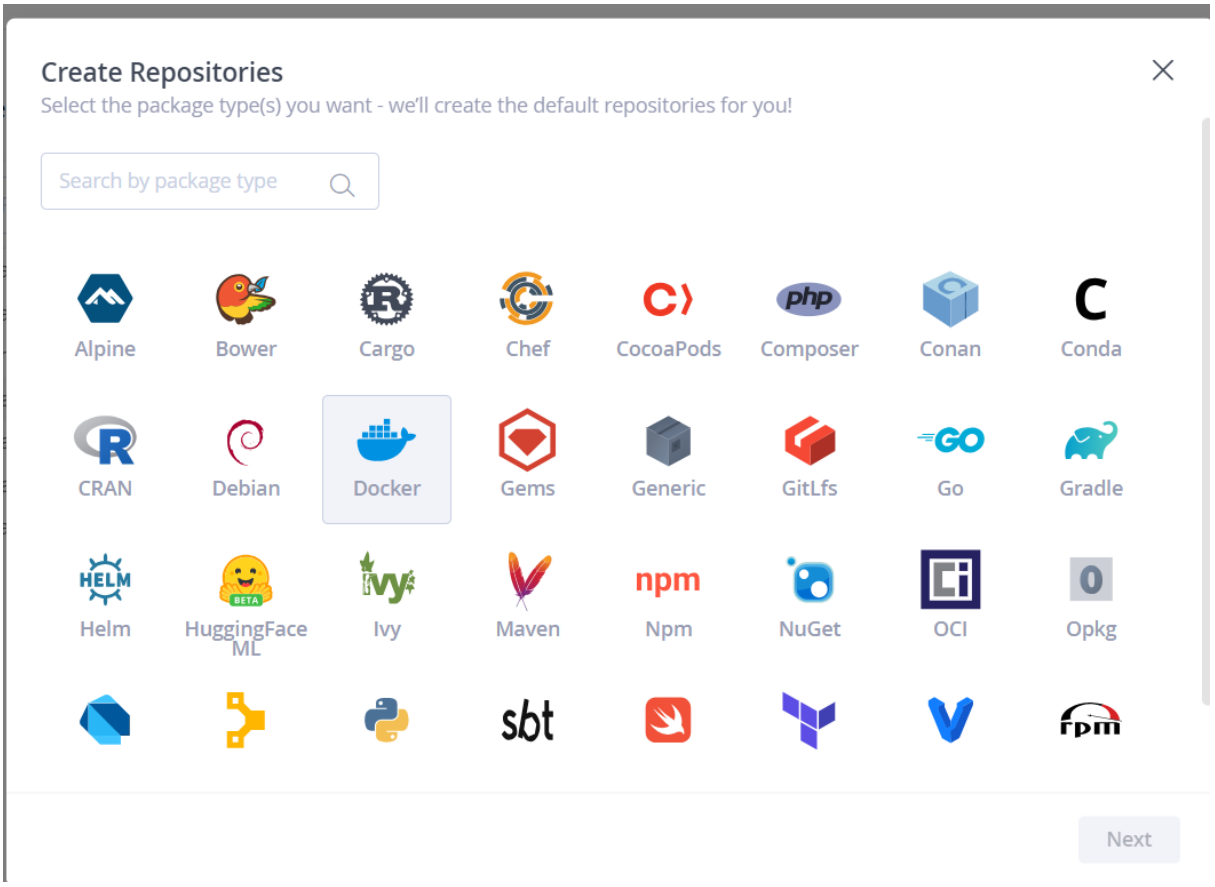
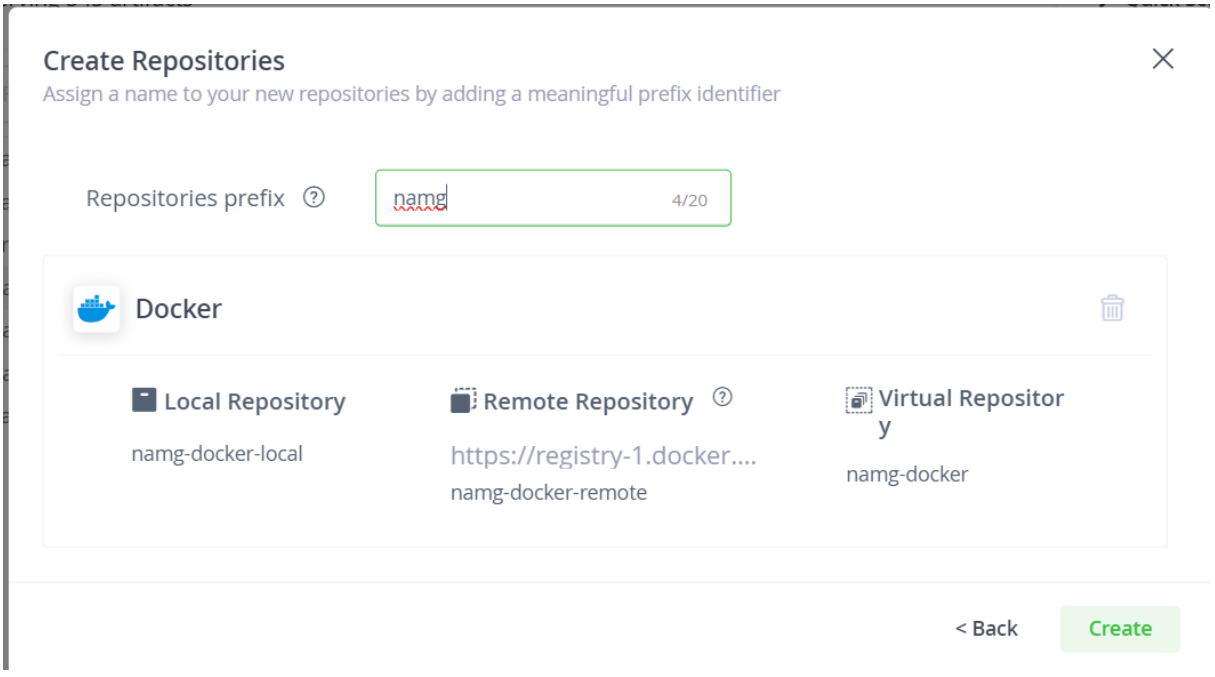
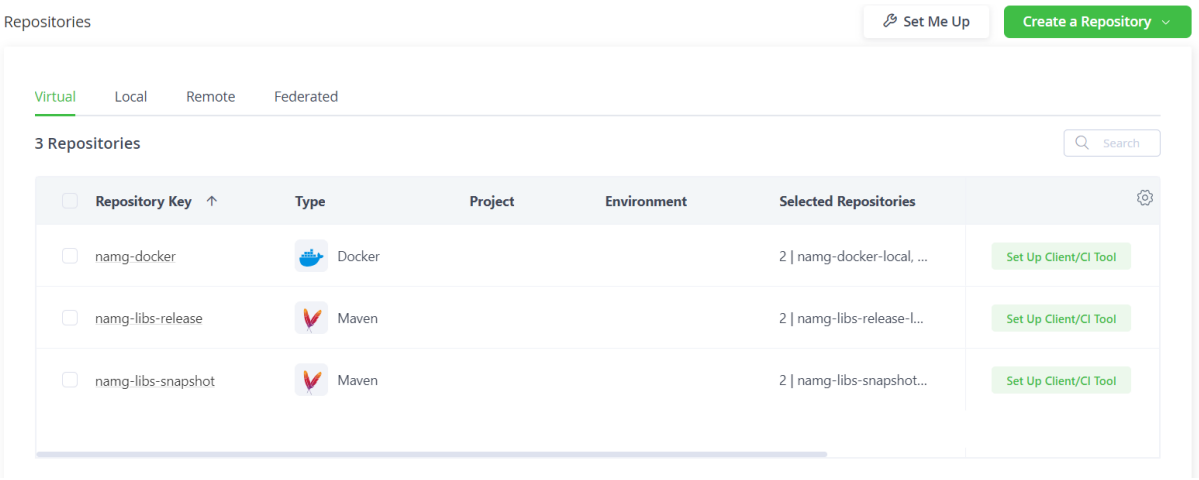
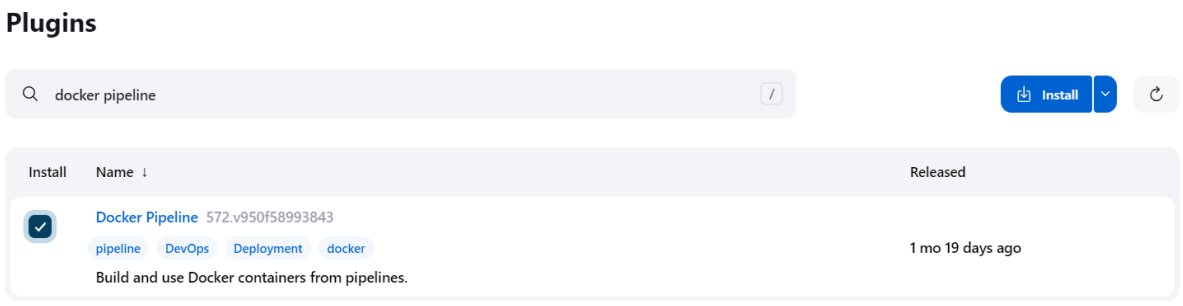
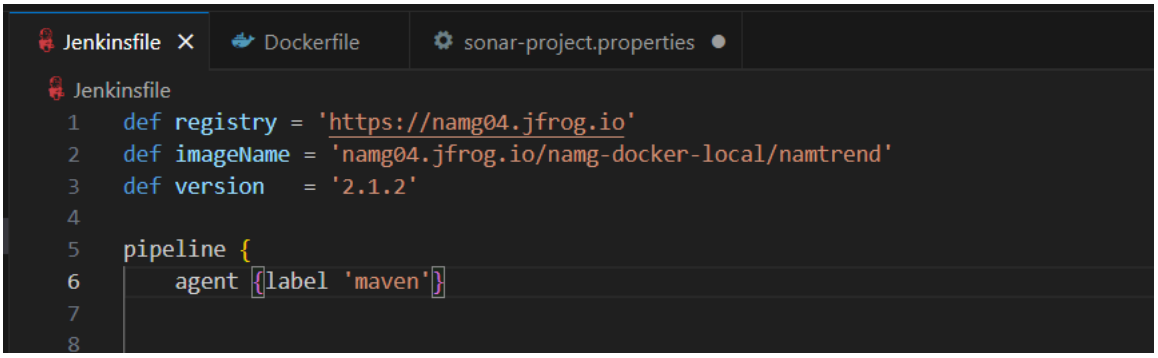
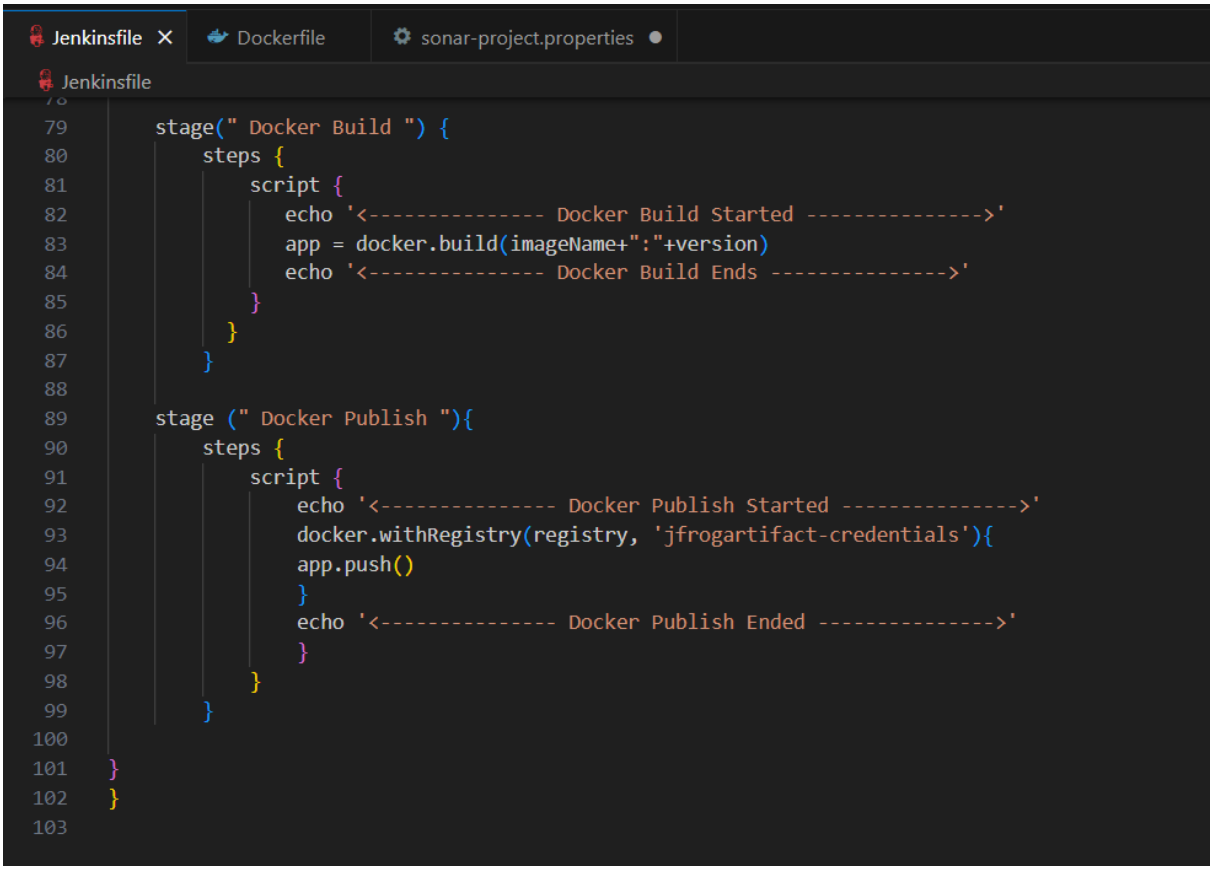
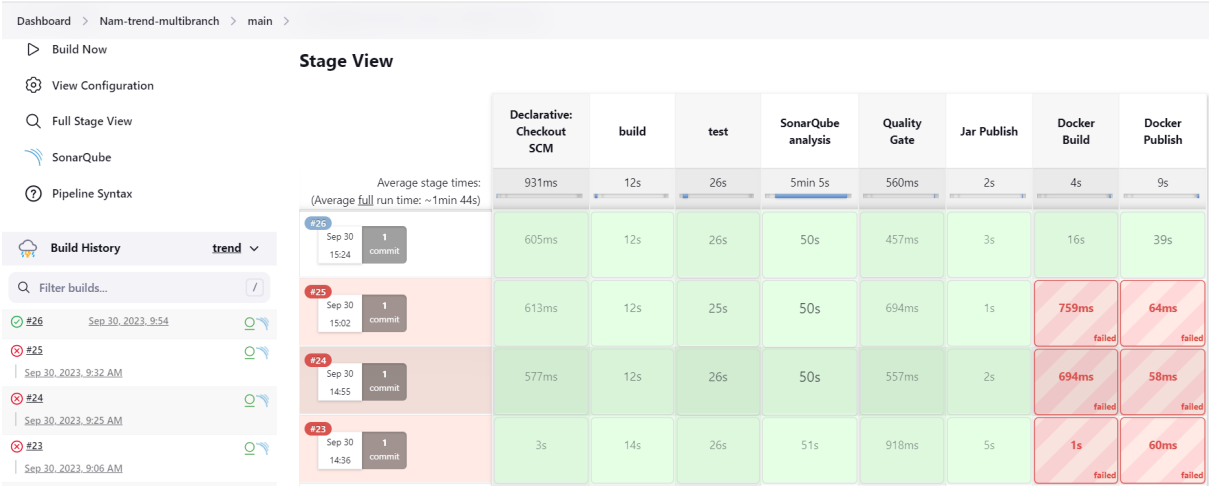
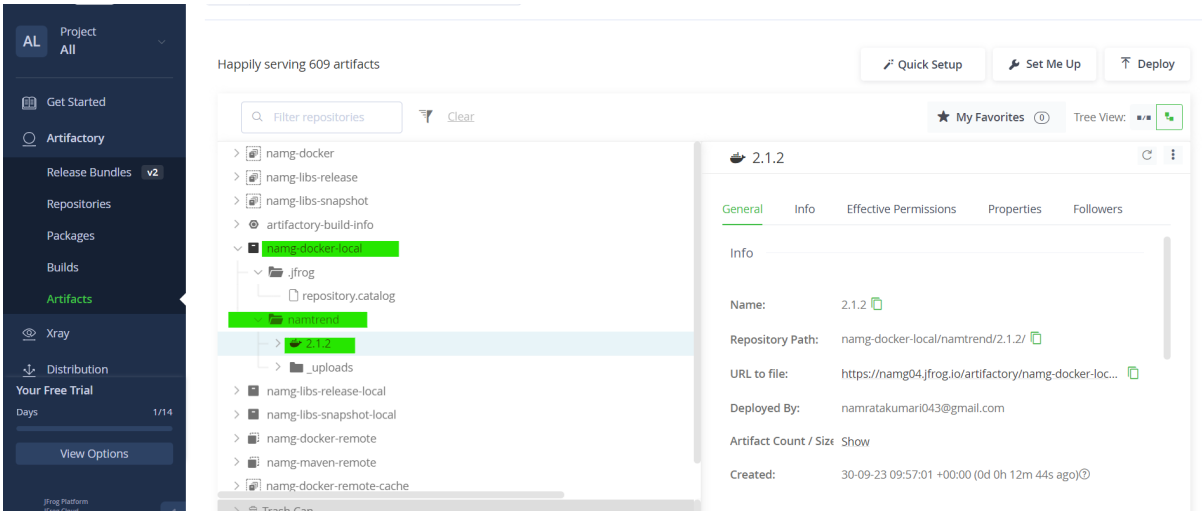
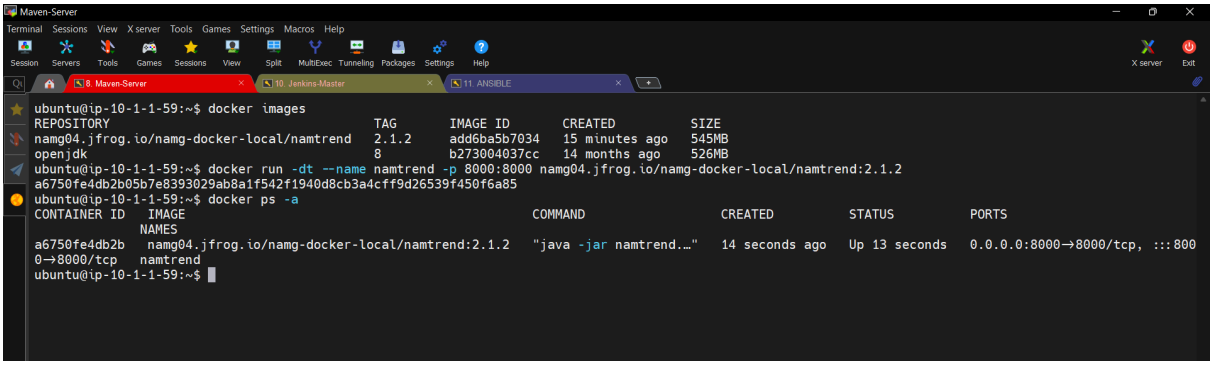
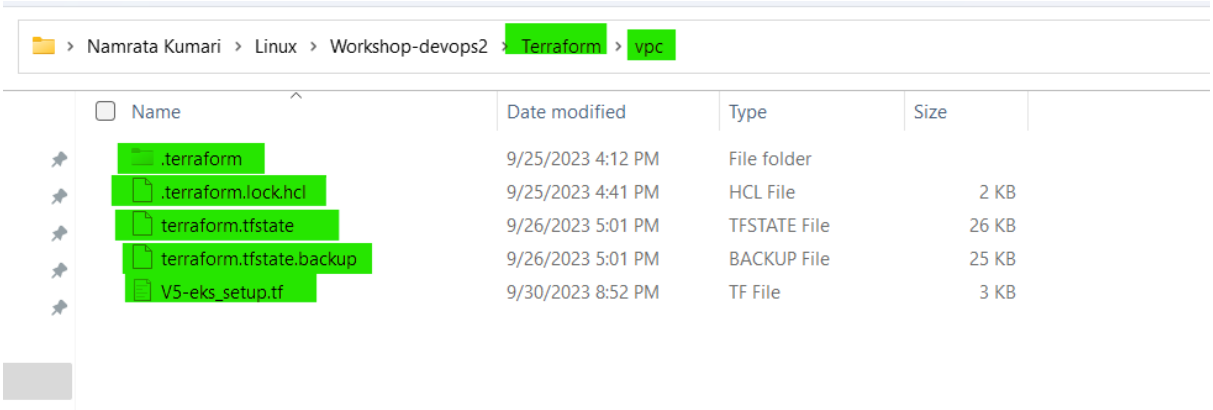
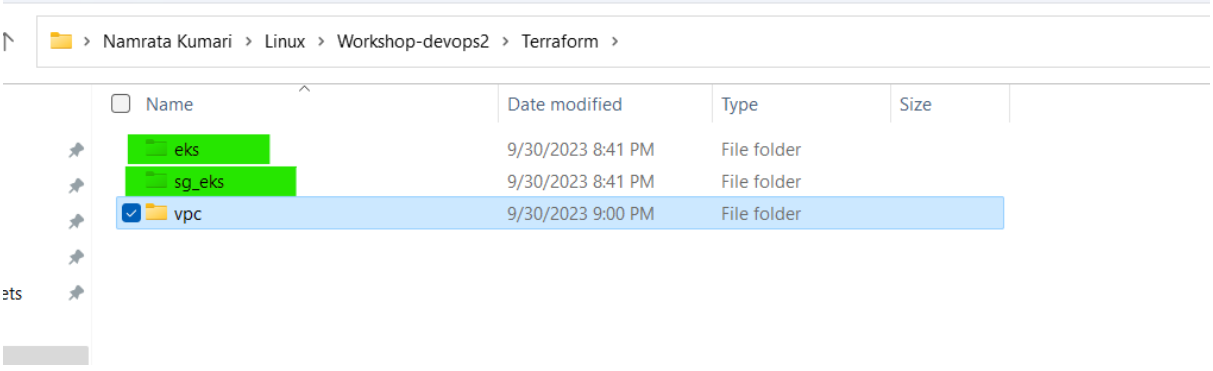
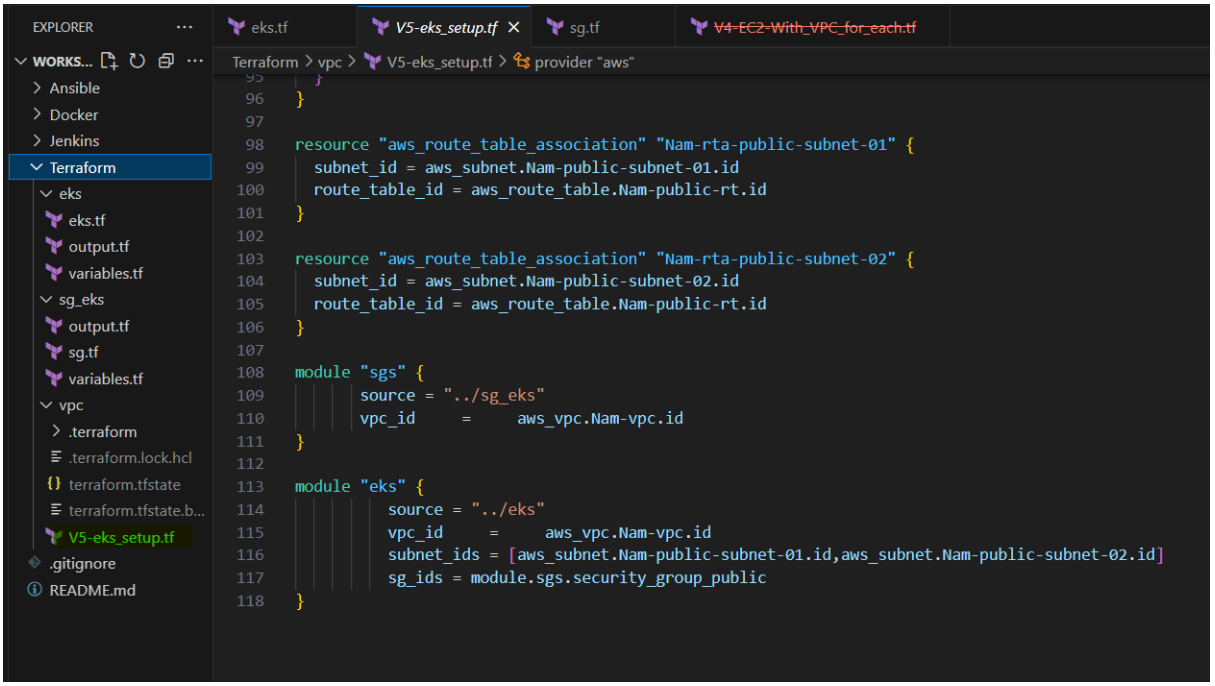
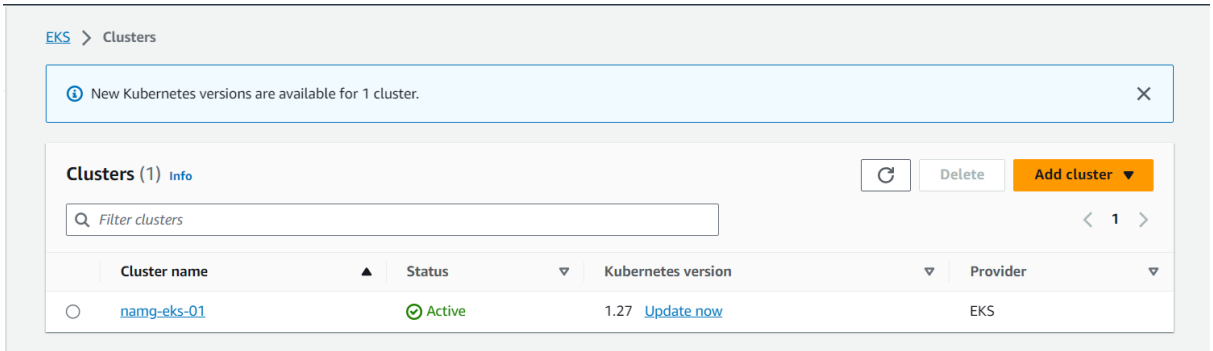
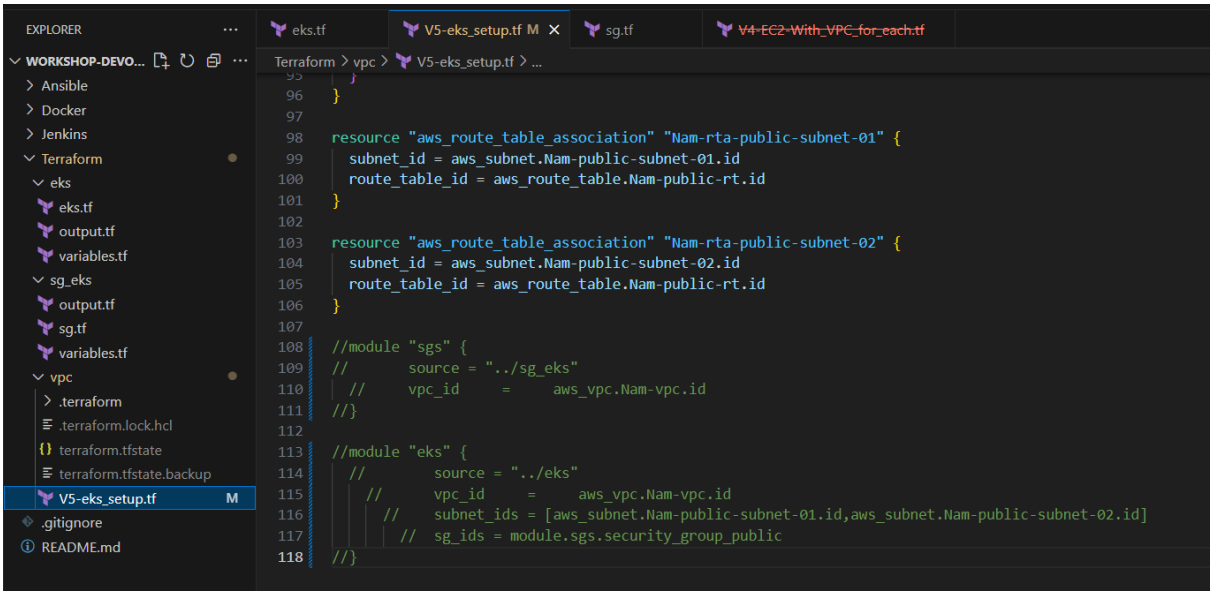
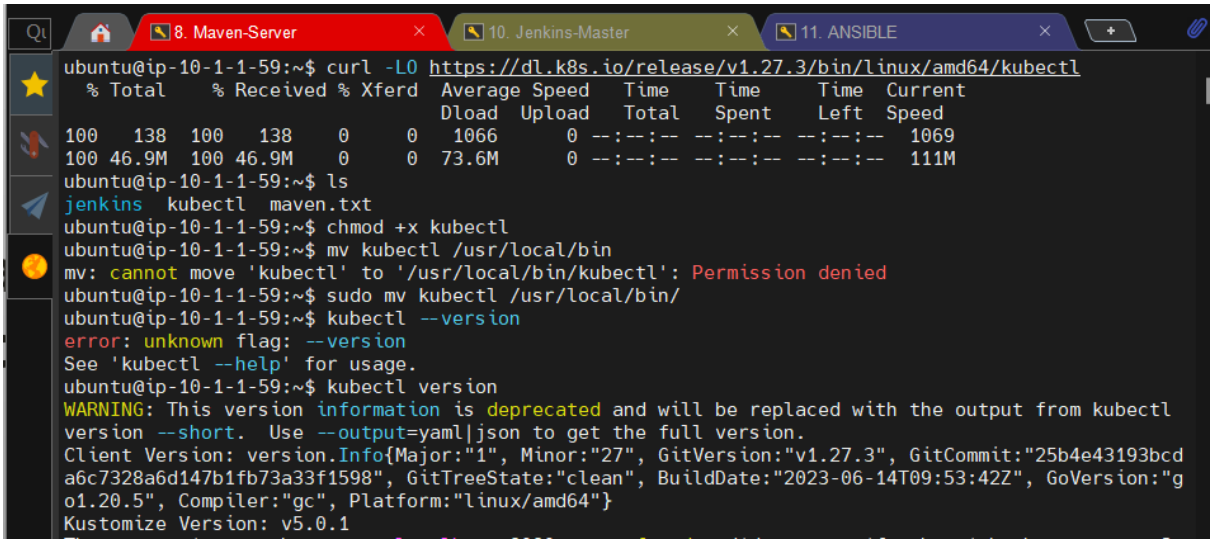
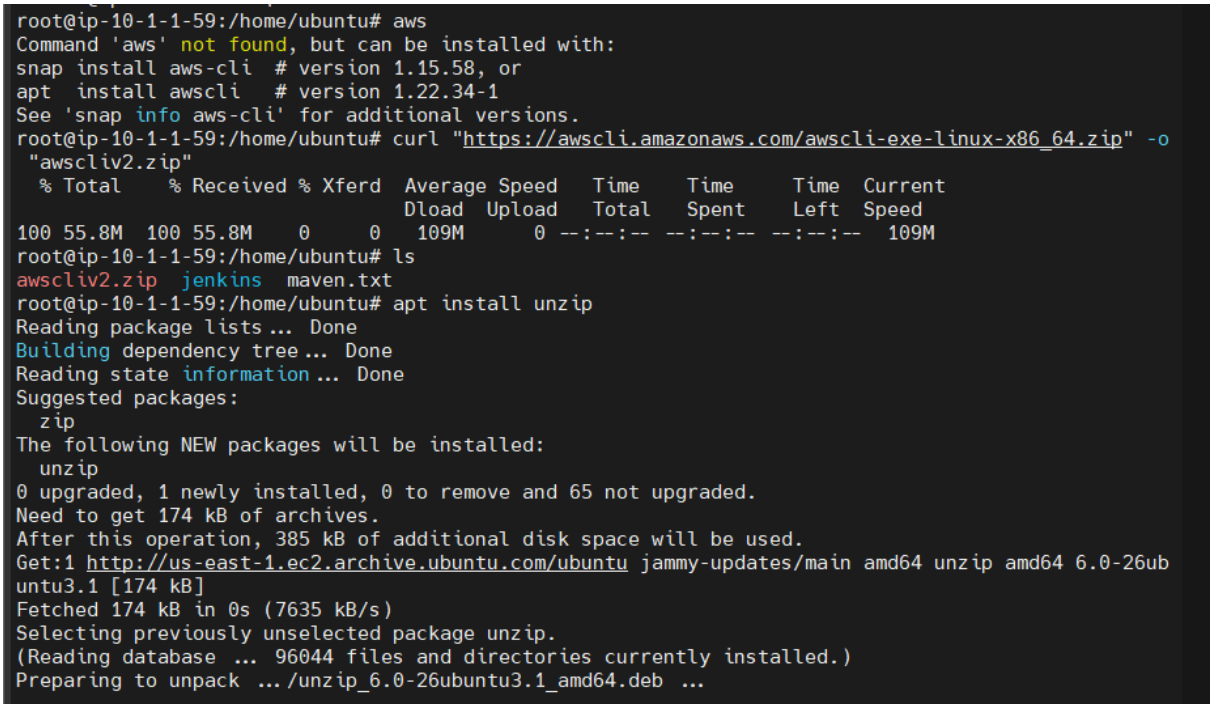
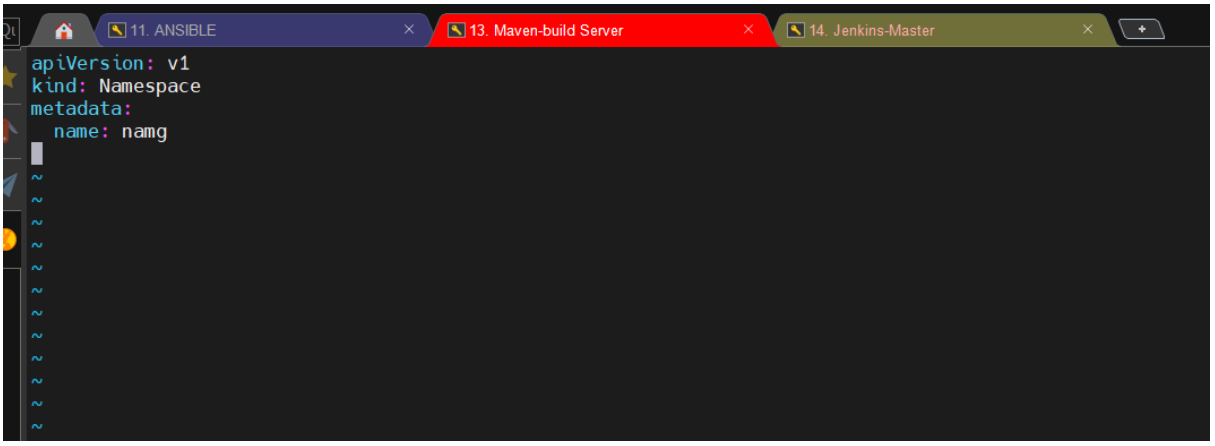
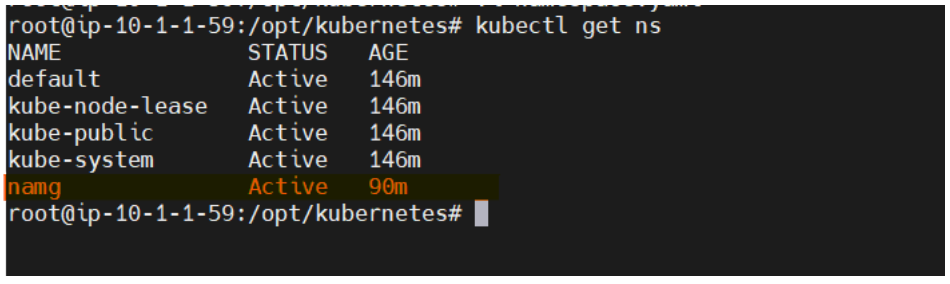
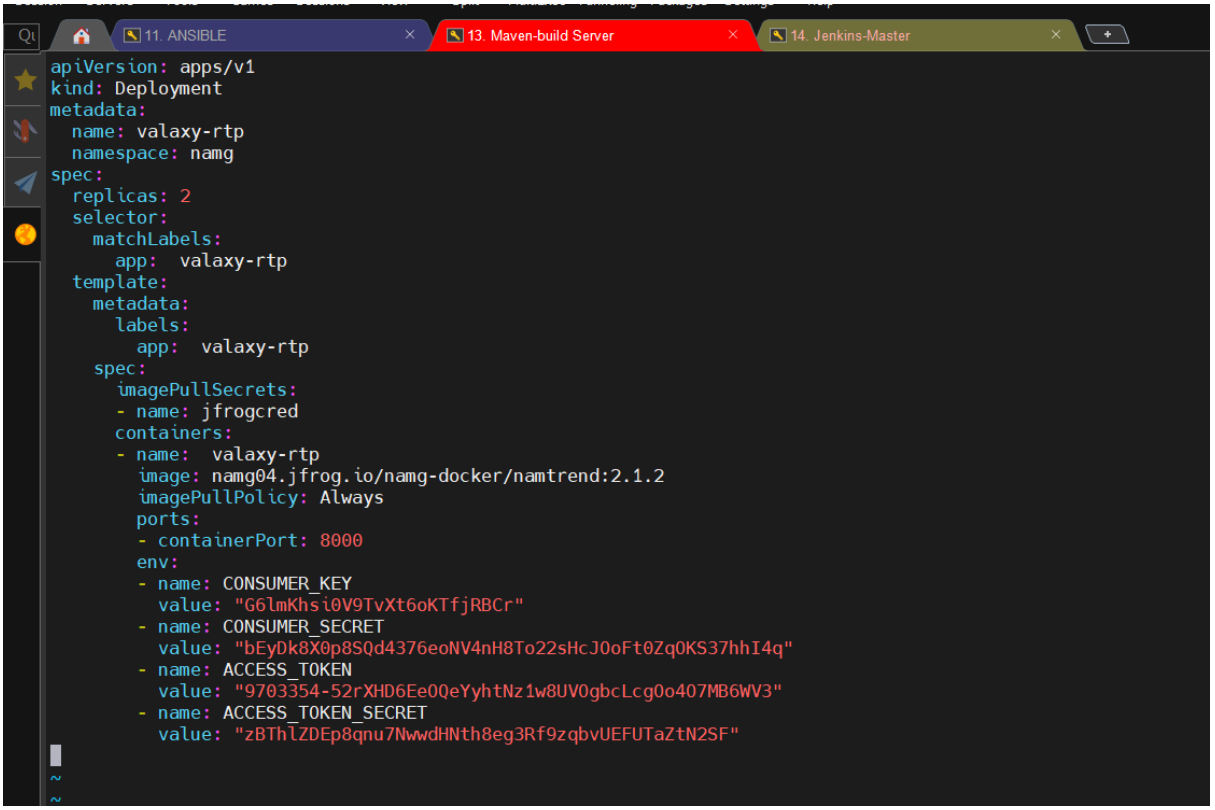
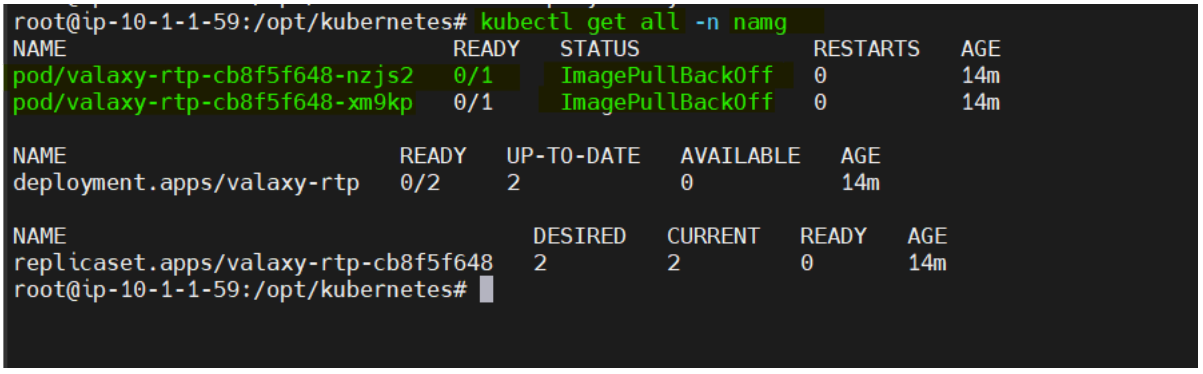
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ ls
Jenkinsfile  README.md  pom.xml  sonar-project.properties  src/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m " added jar publish stage"
[main adbe941]  added jar publish stage
 1 file changed, 29 insertions(+), 1 deletion(-)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 785 bytes | 785.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Namg04/tweet-trend-new.git
   f8b3c92..adbe941  main -> main

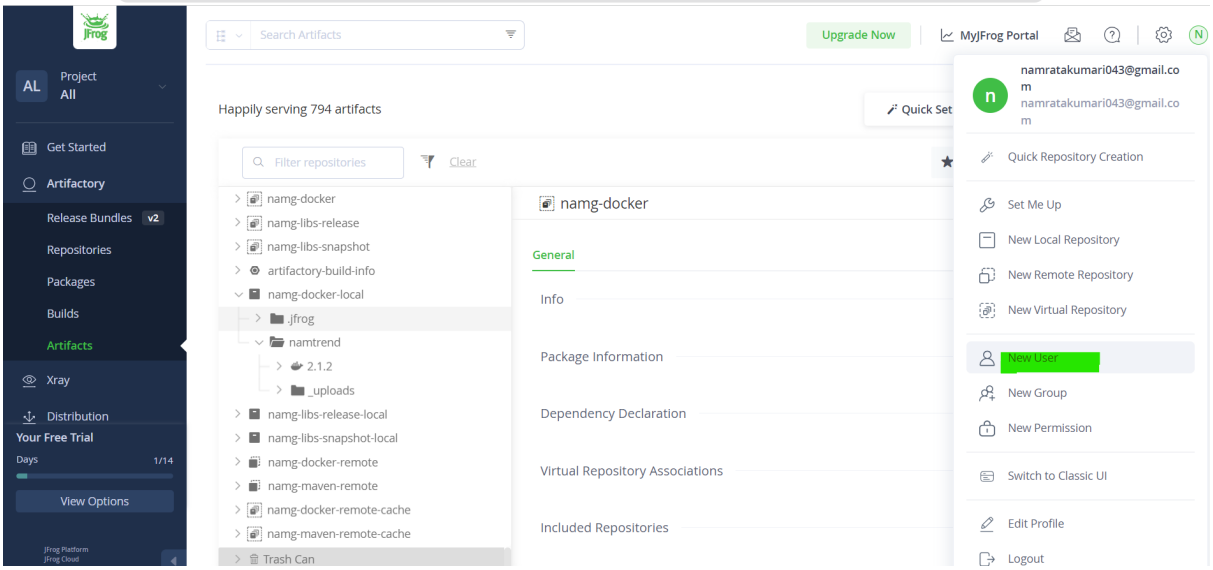
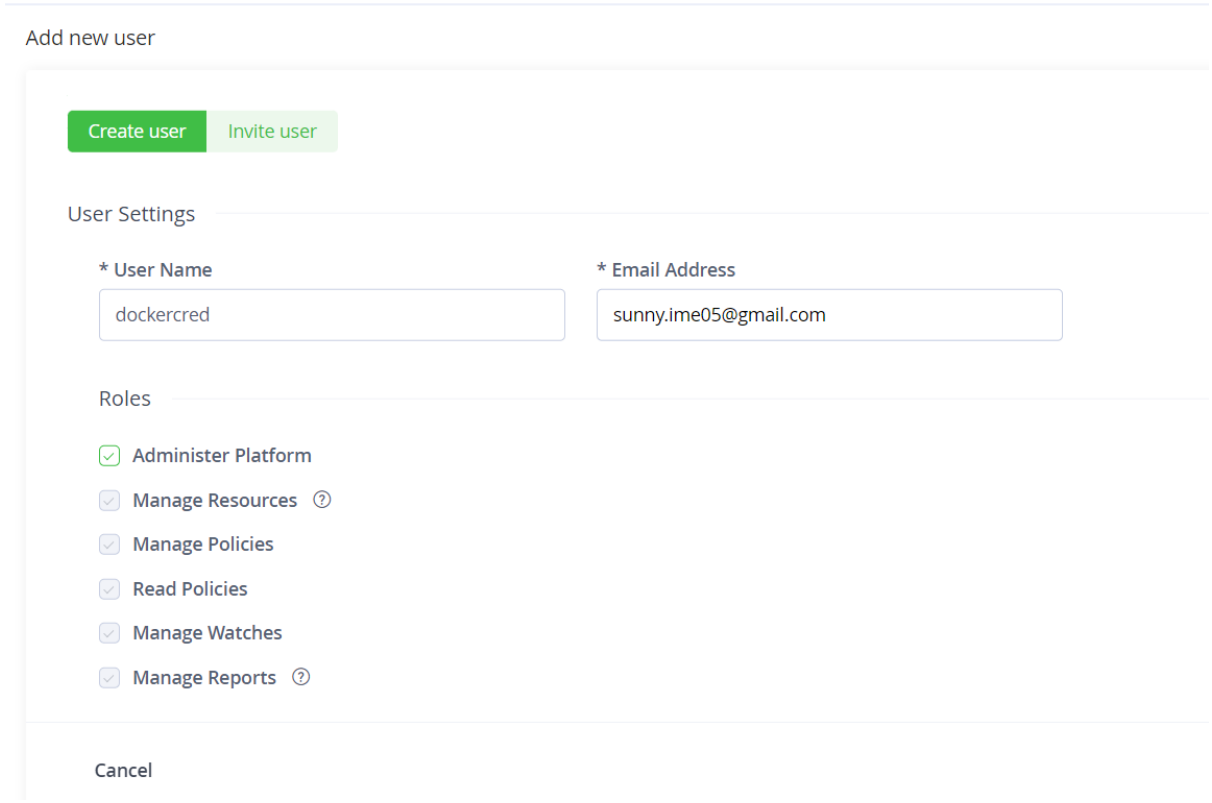
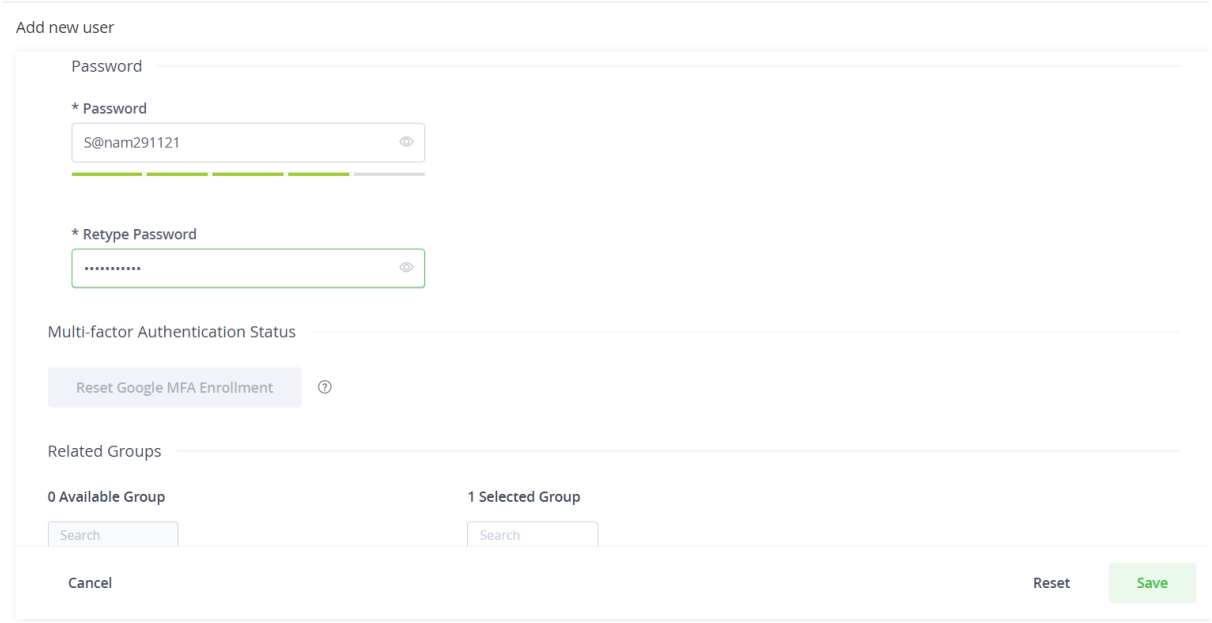
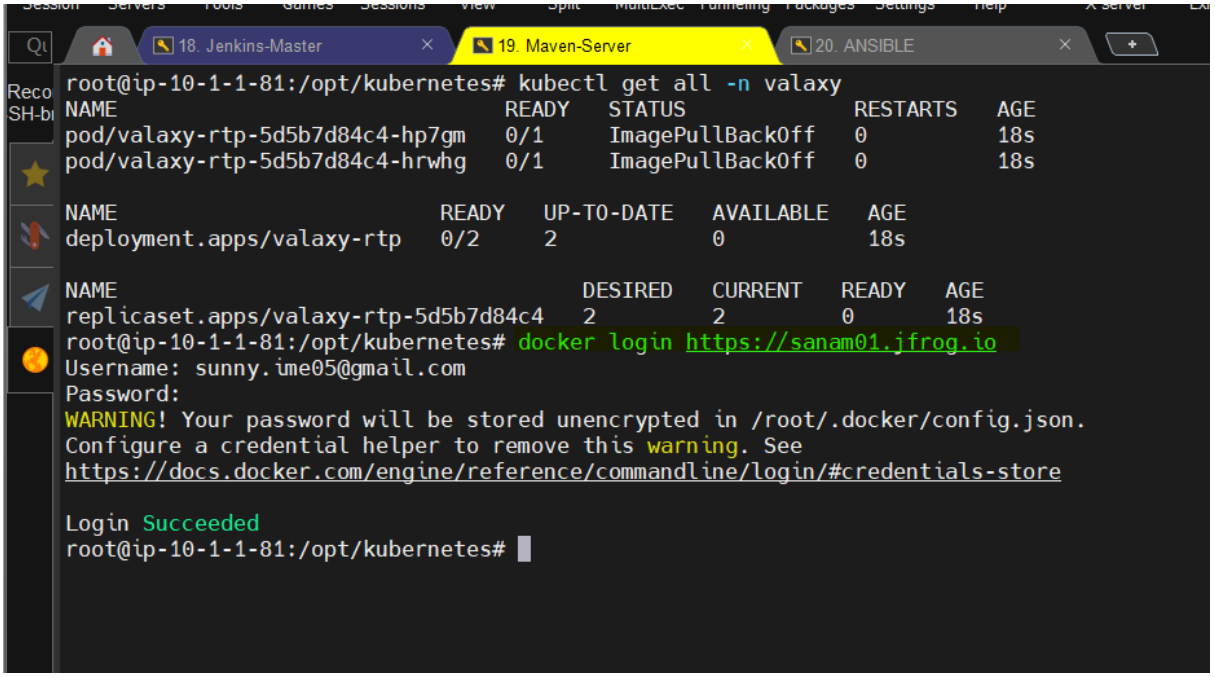
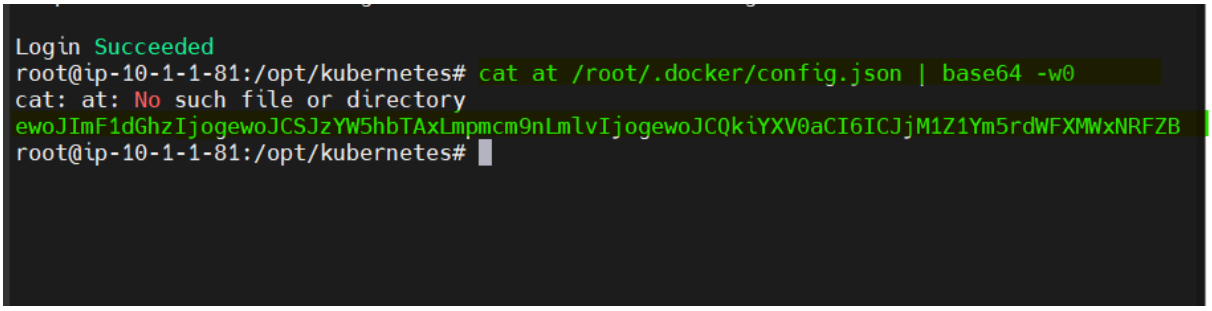
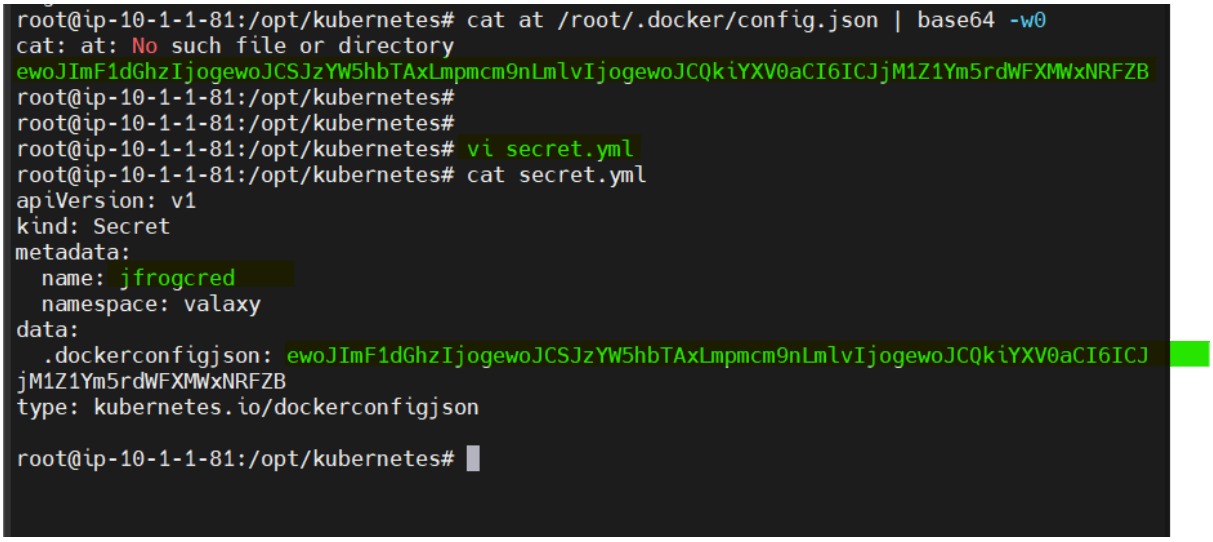
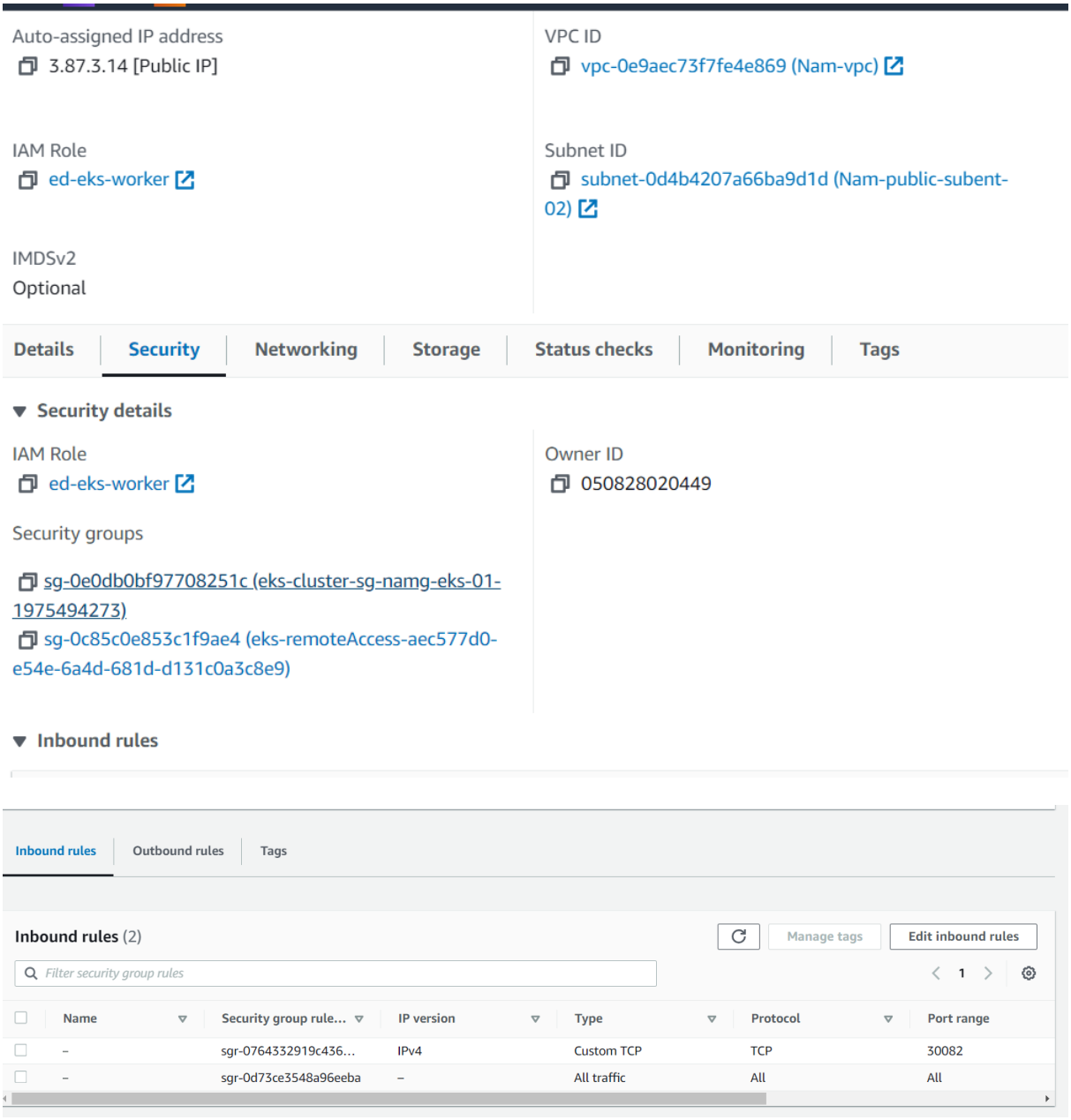
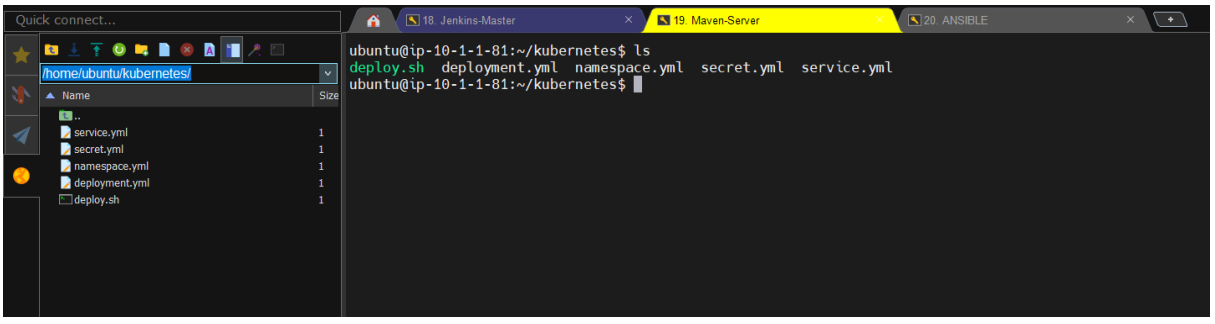
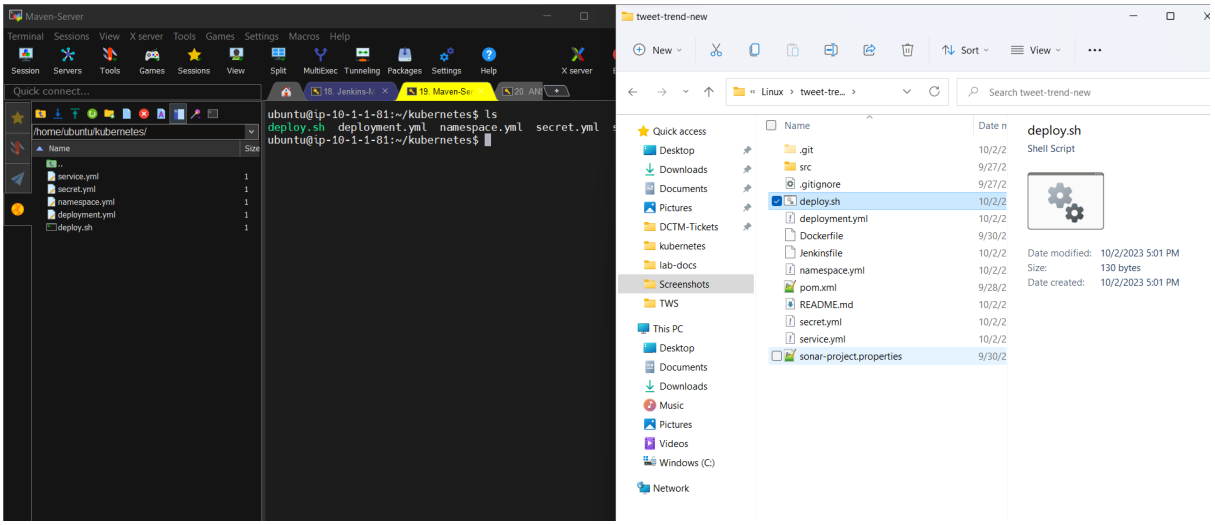
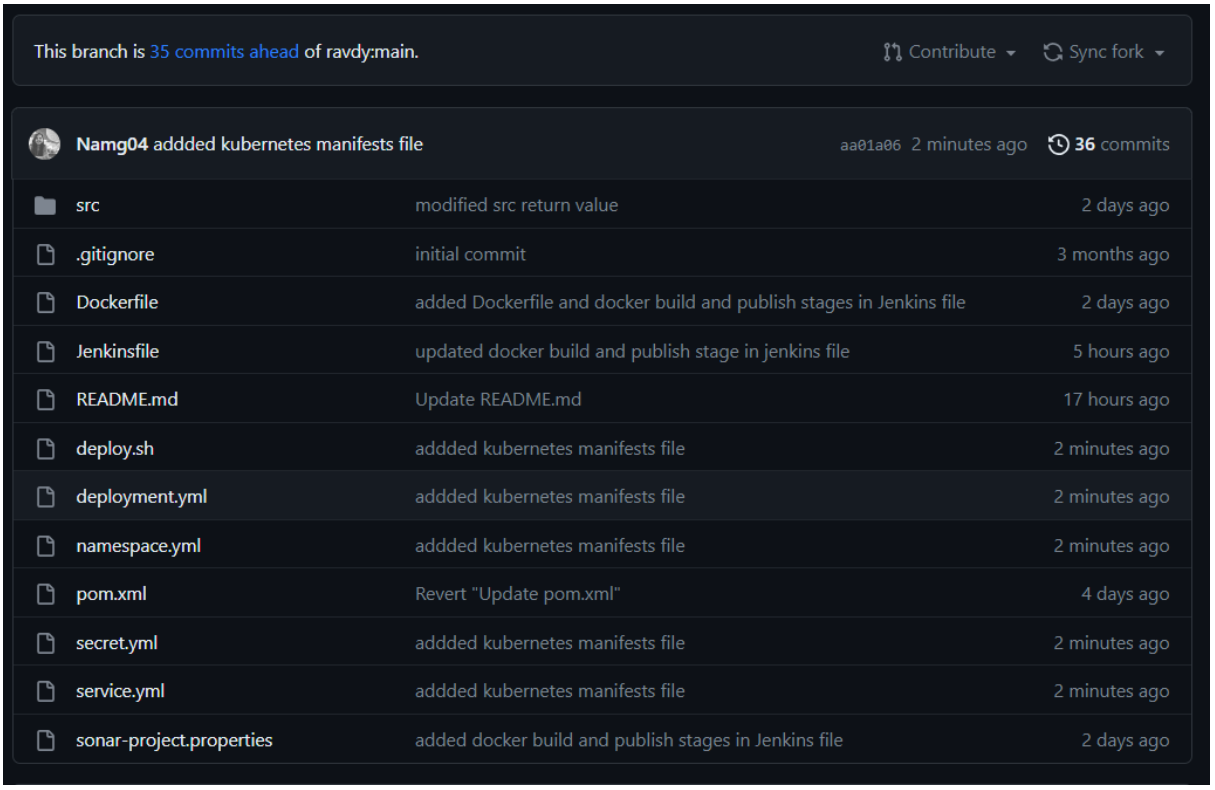
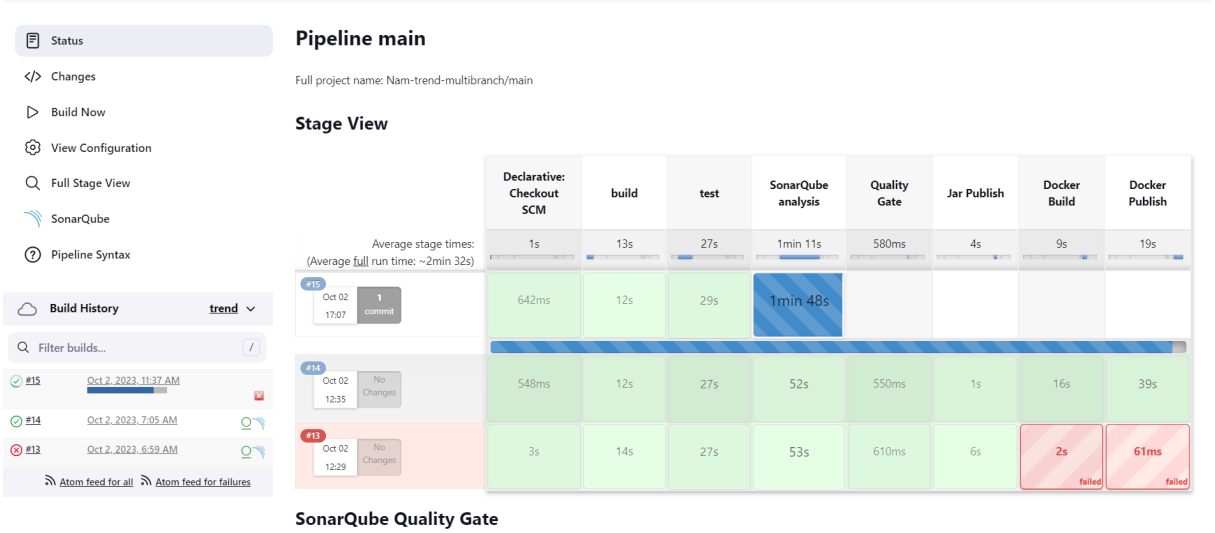
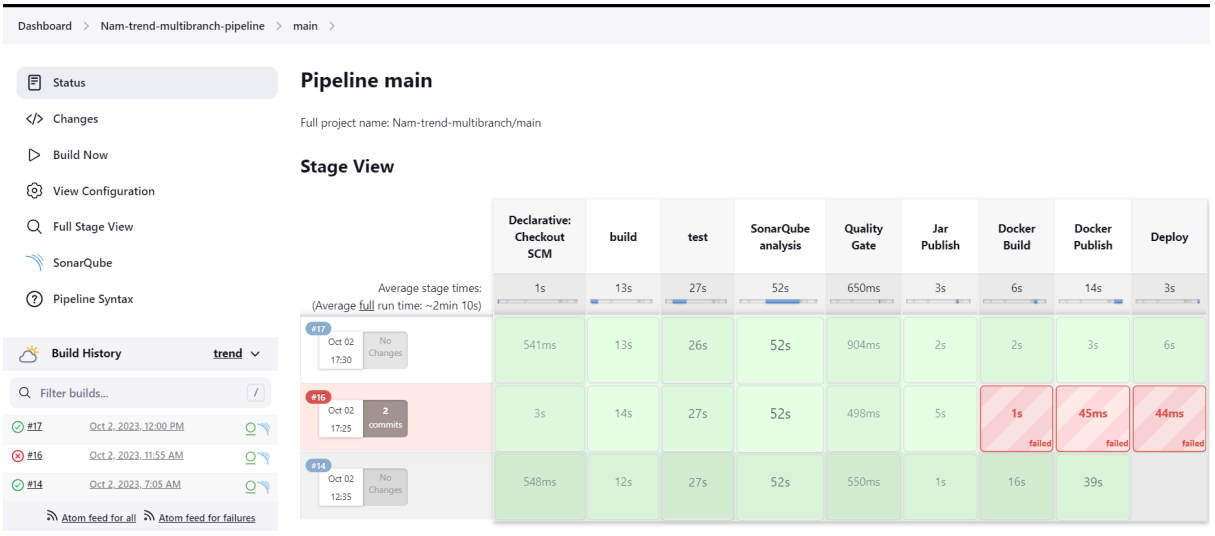
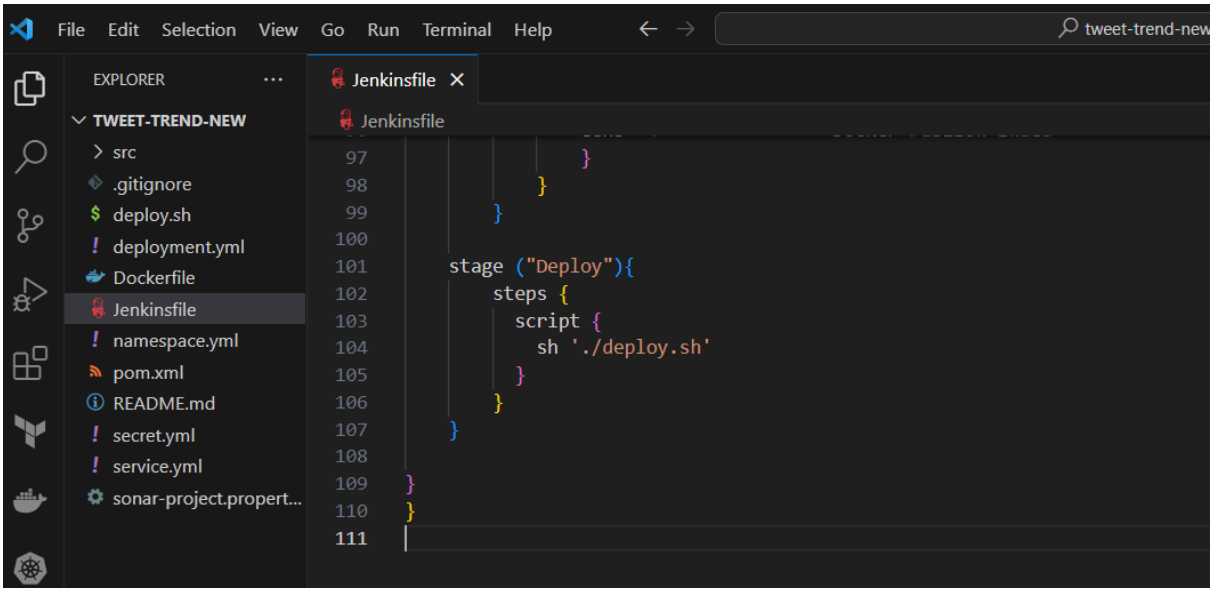
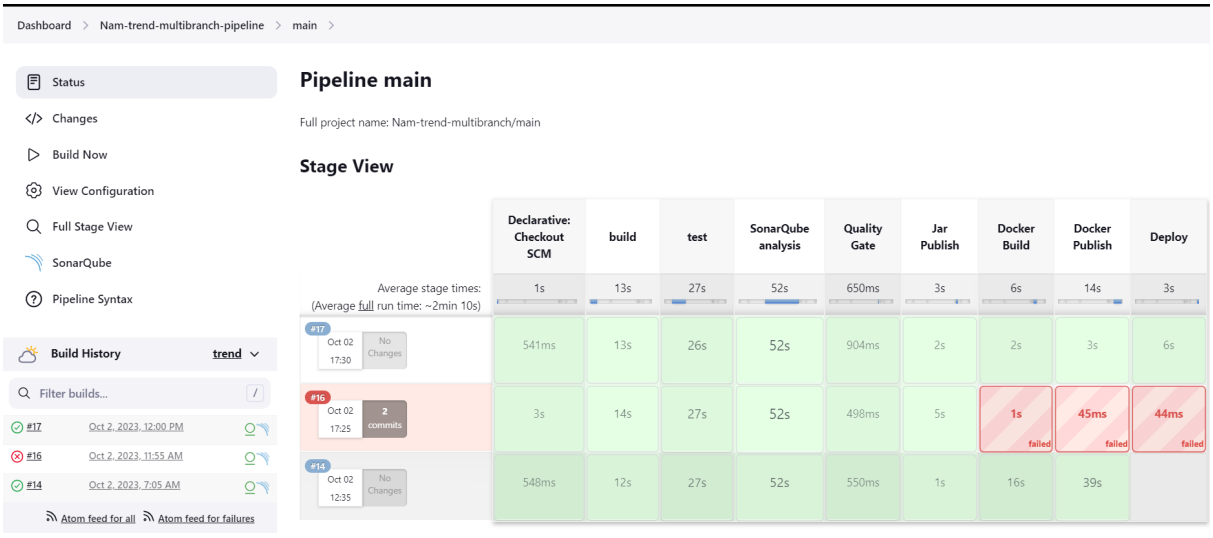
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

Build successful



Published artifact in Jfrog Artifactory



==========

Docker Integration with Jenkins

We have successfully built our code using Maven and performed SonarQube analysis. We have also published the artifact to JFrog Artifactory as a JAR file.

However, since we are deploying this as a microservice, we need to convert it into a Docker image. To do this, we need to generate Docker images as artifacts.

````
## Build and Publish a Docker image

1. Write and add dockerfile in the source code
  ```sh
    FROM openjdk:8
    ADD jarstaging/com/valaxy/demo-workshop/2.0.2/demo-workshop-2.0.2.jar demo-workshop.jar
    ENTRYPOINT ["java", "-jar", "demo-workshop.jar"]
  ```
   `Check-point:`  version number in pom.xml and dockerfile should match
1. Create a docker repository in the Jfrog
    repository name: valaxy-docker
1. Install `docker pipeline` plugin

1. Update Jenkins file with the below stages
    ```sh
     def imageName = 'valaxy01.jfrog.io/valaxy-docker/ttrend'
     def version   = '2.0.2'
        stage(" Docker Build ") {
          steps {
            script {
               echo '<--------------- Docker Build Started --------------->'
               app = docker.build(imageName+":"+version)
               echo '<--------------- Docker Build Ends --------------->'
            }
          }
        }

               stage (" Docker Publish "){
            steps {
                script {
                   echo '<--------------- Docker Publish Started --------------->'
                    docker.withRegistry(registry, 'artifactory_token'){
                        app.push()
                    }
                   echo '<--------------- Docker Publish Ended --------------->'
                }
            }
        }
    ```
````

```
Check-point:
1. Provide jfrog repo URL in the place of `valaxy01.jfrog.io/valaxy-docker` in `def imageName = 'valaxy01.jfrog.io/valaxy-docker/ttrend
2. Match version number in `def version   = '2.0.2'` with pom.xml version number
3. Ensure you have updated credentials in the field of `artifactory_token` in `docker.withRegistry(registry, 'artifactory_token'){`

Note: make sure docker service is running on the slave system, and docker should have permissions to /var/run/docker.sock
```

Integrate docker with Jenkins : Build and Publish a Docker image

- **Install docker on Jenkins slave system (Maven-server) .**
- **Create a Dockerfile.**
- **Create a docker repository in jfrog.**
- **Install “docker pipeline” plugin.**
- **Update Jenkins file with docker build and publish stage.**

1. **Using Ansible playbook to install Docker on Jenkins-slave/maven-server.**
2. **Update jenkins-slave-setup.yml file**

```yaml
---
- hosts: jenkins-slave
  become: true
  tasks:
  - name: update ubuntu repo and cache
    apt:
      update_cache: yes
      cache_valid_time: 3600

  - name: install java
    apt:
      name: openjdk-11-jre
      state: present

  - name: download maven packages
    get_url:
      url: https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
      dest: /opt

  - name: extract maven packages
    unarchive:
      src: /opt/apache-maven-3.9.4-bin.tar.gz
      dest: /opt
      remote_src: yes

  - name: install docker
    apt:
      name: docker.io
      state: present

  - name: start docker services
    service:
      name: docker
      state: started

  - name: give 777 permissions on /var/run/docker.sock
    file:
      path: /var/run/docker.sock
      state: file
      mode: 0777
  - name: start docker on boot time
    service:
      name: docker
      enabled: yes
```

3. **Docker installed on jenkins-slave/maven-server**



4. **Committed code into Git**

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git fetch

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    ../Terraform/V1-EC2.tf
        deleted:    ../Terraform/V2-EC2.tf
        deleted:    ../Terraform/V3-EC2.tf

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        V1-jenkins-slave-setup.yml
        V2-jenkins-slave-setup.yml

no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git add V1-jenkins-slave-setup.yml

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git add V2-jenkins-slave-setup.yml

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git commit -m "added docker installation in Ansible playbook"
[main b4516da] added docker installation in Ansible playbook
 2 files changed, 60 insertions(+)
 create mode 100644 Ansible/V1-jenkins-slave-setup.yml
 create mode 100644 Ansible/V2-jenkins-slave-setup.yml

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 791 bytes | 791.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Namg04/Workshop-devops2.git
   cb6c47c..b4516da  main -> main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$
```

Write and add Dockerfile in the source code



Committed to Git

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ ls
Dockerfile  Jenkinsfile  README.md  pom.xml  sonar-project.properties  src/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   sonar-project.properties

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Dockerfile

no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Dockerfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "added Dockerfile"
[main a6bf57e] added Dockerfile
 1 file changed, 8 insertions(+)
 create mode 100644 Dockerfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 385 bytes | 385.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Namg04/tweet-trend-new.git
   3741d49..a6bf57e  main -> main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

Pushed into Git Repo



Create a docker repository in the Jfrog : repository name: namg-docker







Repo created



Install `docker pipeline` plugin in Jenkins



Update Jenkins file with the below stages

Check-point:

1. **Provide jfrog repo URL in the place of `valaxy01.jfrog.io/valaxy-docker` in `def imageName = 'namg04.jfrog.io/namg-docker/namtrend'`**
2. **Match version number in `def version = '2.1.2'` with pom.xml version number**
3. **Ensure you have updated credentials in the field of `artifactory_token` in `docker.withRegistry(registry, 'jfrogartifact-credentials'){`**

Note: make sure docker service is running on the slave system, and docker should have permissions to /var/run/docker.sock





Commit updated Jenkins file into Git

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Jenkinsfile

no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add .

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "added docker build and publish stages in Jenkins file"
[main ee5b7dc] added docker build and publish stages in Jenkins file
 1 file changed, 18 insertions(+), 14 deletions(-)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 398 bytes | 398.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Namg04/tweet-trend-new.git
   afd9068..ee5b7dc  main -> main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

Build triggered



Verify Docker repo in Jfrog artifactory



Testing Docker image by creating container out of it



Open 8000 port in jenkins-slave/maven-server security group and access image

==========

Kubernetes Setup

To set up a Kubernetes cluster using Terraform, follow these steps:
1) Access the EKS module code from the provided GitHub repository. This module helps create IAM roles, policies, the EKS cluster, and a node group.
2) Copy the eks and sg_eks modules into the Terraform folder.
3) Create a VPC folder and move existing files into it.
4) Add the sg_eks and eks modules in the vpc.tf file, specifying the necessary parameters.
Once these steps are completed, the Terraform modules are ready to create the cluster.

Created vpc folder and moved existing files inside to this



Copy eks and sg_eks modules onto terraform folder



Add sg_eks module and eks modules in the vpc.tf file



Added EKS module to Terraform

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ ls
Ansible/  Docker/  Jenkins/  README.md  Terraform/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    Terraform/V1-EC2.tf
        deleted:    Terraform/V2-EC2.tf
        deleted:    Terraform/V3-EC2.tf
        deleted:    Terraform/V4-EC2-With_VPC_for_each.tf

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Terraform/eks/
        Terraform/sg_eks/
        Terraform/vpc/

no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git add ^C

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git add Terraform/eks/ Terraform/sg_eks/  Terraform/vpc/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git commit -m "added EKS module to terraform code
> ^C

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git commit -m "added EKS module to terraform code"
[main 86224b2] added EKS module to terraform code
 7 files changed, 335 insertions(+)
 create mode 100644 Terraform/eks/eks.tf
 create mode 100644 Terraform/eks/output.tf
 create mode 100644 Terraform/eks/variables.tf
 create mode 100644 Terraform/sg_eks/output.tf
 create mode 100644 Terraform/sg_eks/sg.tf
 create mode 100644 Terraform/sg_eks/variables.tf
 create mode 100644 Terraform/vpc/V4-EC2-With_VPC_for_each.tf

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git push origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 20 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 2.91 KiB | 2.91 MiB/s, done.
Total 13 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Namg04/Workshop-devops2.git
```

Renamed vpc file name

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ ls
Ansible/  Docker/  Jenkins/  README.md  Terraform/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ cd Terraform/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ ls
eks/  sg_eks/  vpc/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ cd vpc/

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ ls
V4-EC2-With_VPC_for_each.tf  terraform.tfstate  terraform.tfstate.backup

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ mv V4-EC2-With_VPC_for_each.tf V5-eks_setup.tf

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ git add .

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    V4-EC2-With_VPC_for_each.tf -> V5-eks_setup.tf

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    ../V1-EC2.tf
        deleted:    ../V2-EC2.tf
        deleted:    ../V3-EC2.tf
        deleted:    ../V4-EC2-With_VPC_for_each.tf


namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ git commit -m "renamed vpc file name"
[main fa1c2ba] renamed vpc file name
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename Terraform/vpc/{V4-EC2-With_VPC_for_each.tf => V5-eks_setup.tf} (100%)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 362 bytes | 362.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Namg04/Workshop-devops2.git
```

Execute terraform manifest file to setup EKS cluster

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ terraform init
Initializing modules...
- eks in ..\eks
- sgs in ..\sg_eks

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.17.0

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.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ terraform validate
Success! The configuration is valid.


namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ terraform plan
aws_vpc.Nam-vpc: Refreshing state... [id=vpc-0c09026eb7725a827]
aws_internet_gateway.Nam-igw: Refreshing state... [id=igw-0540d289cd15e8b99]
aws_subnet.Nam-public-subnet-02: Refreshing state... [id=subnet-0c0b0c9403ebbc5d8]
aws_subnet.Nam-public-subnet-01: Refreshing state... [id=subnet-024dfff7c21f2a8c3]
aws_security_group.demo-sg: Refreshing state... [id=sg-031d62b69f2db6dd7]
aws_route_table.Nam-public-rt: Refreshing state... [id=rtb-014960311b0d80c0b]
aws_instance.demo-server["jenkins-master"]: Refreshing state... [id=i-098613d9fa83fffe7]
aws_instance.demo-server["jenikns-slave"]: Refreshing state... [id=i-0d39a6671353d4b43]
aws_instance.demo-server["ansible"]: Refreshing state... [id=i-07b852789631b3ed9]
aws_route_table_association.Nam-rta-public-subnet-02: Refreshing state... [id=rtbassoc-0ca8631046a95217
c]
aws_route_table_association.Nam-rta-public-subnet-01: Refreshing state... [id=rtbassoc-0b471f813fa3213e
f]

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

Terraform will perform the following actions:

  # aws_security_group.demo-sg will be updated in-place
  ~ resource "aws_security_group" "demo-sg" {
        id                     = "sg-031d62b69f2db6dd7"
      ~ ingress                = [
          - {
              - cidr_blocks      = [
                    "0.0.0.0/0",
```

```
    }

Plan: 17 to add, 1 to change, 0 to destroy.

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

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

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ terraform apply --auto-approve
aws_vpc.Nam-vpc: Refreshing state... [id=vpc-0c09026eb7725a827]
aws_subnet.Nam-public-subnet-01: Refreshing state... [id=subnet-024dfff7c21f2a8c3]
aws_internet_gateway.Nam-igw: Refreshing state... [id=igw-0540d289cd15e8b99]
aws_subnet.Nam-public-subnet-02: Refreshing state... [id=subnet-0c0b0c9403ebbc5d8]
aws_security_group.demo-sg: Refreshing state... [id=sg-031d62b69f2db6dd7]
aws_route_table.Nam-public-rt: Refreshing state... [id=rtb-014960311b0d80c0b]
aws_instance.demo-server["jenikns-slave"]: Refreshing state... [id=i-0d39a6671353d4b43]
aws_instance.demo-server["ansible"]: Refreshing state... [id=i-07b852789631b3ed9]
aws_instance.demo-server["jenkins-master"]: Refreshing state... [id=i-098613d9fa83fffe7]
aws_route_table_association.Nam-rta-public-subnet-02: Refreshing state... [id=rtbassoc-0ca8631046a9521
c]
aws_route_table_association.Nam-rta-public-subnet-01: Refreshing state... [id=rtbassoc-0b471f813fa3213
f]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create
  ~ update in-place
```

```
module.eks.aws_eks_node_group.backend: Still creating... [1m40s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [1m50s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [2m0s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [2m10s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [2m20s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [2m30s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [2m40s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [2m50s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [3m0s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [3m10s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [3m20s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [3m30s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [3m40s elapsed]
module.eks.aws_eks_node_group.backend: Still creating... [3m50s elapsed]
module.eks.aws_eks_node_group.backend: Creation complete after 3m55s [id=namg-eks-01:dev]

Apply complete! Resources: 17 added, 1 changed, 0 destroyed.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$
```

Cluster created in AWS



To destroy EKS infrastructure, just comment modules in vpc project




```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ terraform plan
module.eks.aws_iam_role_policy_attachment.AmazonEC2ContainerRegistryReadOnly: Re
freshing state... [id=ed-eks-worker-20230930170607517600000004]
module.eks.aws_iam_role_policy_attachment.autoscaler: Refreshing state... [id=ed
-eks-worker-20230930170607755600000006]
module.eks.aws_iam_role_policy_attachment.s3: Refreshing state... [id=ed-eks-wor
ker-20230930170608179500000000a]
```

```
    }

Plan: 0 to add, 0 to change, 17 to destroy.

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

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

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ terraform apply --auto-approve
module.eks.aws_iam_role_policy_attachment.AmazonEKS_CNI_Policy: Refreshing state... [id=ed-eks-worker-2
023093017060780510000008]
module.eks.aws_iam_role_policy_attachment.AmazonSSMManagedInstanceCore: Refreshing state... [id=ed-eks-
worker-20230930170608158600000009]
```

```
module.eks.aws_eks_cluster.eks: Destruction complete after 52s
module.eks.aws_iam_role_policy_attachment.AmazonEKSClusterPolicy: Destroying... [id=ed-eks-master-20230
930170607128900000001]
module.eks.aws_iam_role_policy_attachment.AmazonEKSServicePolicy: Destroying... [id=ed-eks-master-20230
930170607204200000002]
module.eks.aws_iam_role_policy_attachment.AmazonEKSVPCResourceController: Destroying... [id=ed-eks-mast
er-20230930170607212500000003]
module.eks.aws_iam_role_policy_attachment.AmazonEKSVPCResourceController: Destruction complete after 0s
module.eks.aws_iam_role_policy_attachment.AmazonEKSClusterPolicy: Destruction complete after 0s
module.eks.aws_iam_role_policy_attachment.AmazonEKSServicePolicy: Destruction complete after 1s
module.eks.aws_iam_role.master: Destroying... [id=ed-eks-master]
module.eks.aws_iam_role.master: Destruction complete after 0s

Apply complete! Resources: 0 added, 0 changed, 17 destroyed.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$
```

==========

Integrate maven server with kubernetes cluster

Setup kubectl

```
curl -LO https://dl.k8s.io/release/v1.27.3/bin/linux/amd64/kubectl
chmod +x ./kubectl
mv ./kubectl /usr/local/bin
kubectl version
```




```
Make sure you have installed awscli latest version. If it has awscli version 1.X then remove it and install awscli 2.X

yum remove awscli
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install --update
```




```
Configure awscli to connect with aws account
aws configure
Provide access_key, secret_key
```

```
  inflating: aws/dist/docutils/writers/pep_html/template.txt
root@ip-10-1-1-59:/home/ubuntu# ls
aws  awscliv2.zip  jenkins  maven.txt
root@ip-10-1-1-59:/home/ubuntu# ./aws/install
You can now run: /usr/local/bin/aws --version
root@ip-10-1-1-59:/home/ubuntu# aws --version
aws-cli/2.13.22 Python/3.11.5 Linux/6.2.0-1012-aws exe/x86_64.ubuntu.22 prompt/off
root@ip-10-1-1-59:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIAQXVMWBLQZR2E3OHG
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]: ^C
root@ip-10-1-1-59:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIAQXVMWBLQZR2E3OHG
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
root@ip-10-1-1-59:/home/ubuntu# aws s3 ls
```

```
Download Kubernetes credentials and cluster configuration (.kube/config file) from the cluster
aws eks update-kubeconfig --region us-east-1 --name namg-eks-01
```

```
root@ip-10-1-1-59:/home/ubuntu# aws eks update-kubeconfig --region us-east-1 --name namg-eks-01
Added new context arn:aws:eks:us-east-1:050828020449:cluster/namg-eks-01 to /root/.kube/config
root@ip-10-1-1-59:/home/ubuntu# kubectl get nodes
NAME                          STATUS   ROLES    AGE   VERSION
ip-10-1-1-34.ec2.internal     Ready    <none>   31m   v1.27.5-eks-43840fb
ip-10-1-2-183.ec2.internal    Ready    <none>   31m   v1.27.5-eks-43840fb
root@ip-10-1-1-59:/home/ubuntu#
```

let's execute our command so that it can able to authenticate with our cluster. You can see here now that Kubeconfig has been copied into the slash home ubuntu. Earlier it could be in the slash root cube and config.

```
ubuntu@ip-10-1-1-81:~$ aws configure
AWS Access Key ID [None]: AKIAQXVMWBLQZR2E3OHG
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:
ubuntu@ip-10-1-1-81:~$
ubuntu@ip-10-1-1-81:~$ aws eks update-kubeconfig --region us-east-1 --name namg-eks-01
Added new context arn:aws:eks:us-east-1:050828020449:cluster/namg-eks-01 to /home/ubuntu
/.kube/config
ubuntu@ip-10-1-1-81:~$
```

```
ubuntu@ip-10-1-1-81:~$ ls
aws  awscliv2.zip  jenkins  kubernetes
ubuntu@ip-10-1-1-81:~$ cd kubernetes/
ubuntu@ip-10-1-1-81:~/kubernetes$ ls
deployment.yml  namespace.yml  secret.yml  service.yml
ubuntu@ip-10-1-1-81:~/kubernetes$ cat > deploy.sh
#!/bin/sh
kubectl apply -f namespace.yaml
kubectl apply -f secret.yaml
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
^C
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl delete -f service.yml
service "valaxy-rtp-service" deleted
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl delete -f deployment.yml
deployment.apps "valaxy-rtp" deleted
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl delete -f secrect.yml
error: the path "secrect.yml" does not exist
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl delete -f service.yml
Error from server (NotFound): error when deleting "service.yml": services "valaxy-rtp-se
rvice" not found
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl delete -f secret.yml
secret "jfrogcred" deleted
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl delete -f namespace.yml
namespace "valaxy" deleted
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl get ns
NAME              STATUS   AGE
default           Active   137m
kube-node-lease   Active   137m
kube-public       Active   137m
kube-system       Active   137m
ubuntu@ip-10-1-1-81:~/kubernetes$ ls -l deploy.sh
-rw-rw-r-- 1 ubuntu ubuntu 134 Oct  2 11:16 deploy.sh
ubuntu@ip-10-1-1-81:~/kubernetes$ chmod +x deploy.sh
ubuntu@ip-10-1-1-81:~/kubernetes$ ls -l deploy.sh
-rwxrwxr-x 1 ubuntu ubuntu 134 Oct  2 11:16 deploy.sh
ubuntu@ip-10-1-1-81:~/kubernetes$ ./deploy.sh
```

```
#!/bin/sh
kubectl apply -f namespace.yml
kubectl apply -f secret.yml
kubectl apply -f deployment.yml
kubectl apply -f service.yml
~
~
~
~
~
~
~
~
~
~
~
~
```

```
ubuntu@ip-10-1-1-81:~/kubernetes$ ls
deploy.sh  deployment.yml  namespace.yml  secret.yml  service.yml
ubuntu@ip-10-1-1-81:~/kubernetes$ vi deploy.sh
ubuntu@ip-10-1-1-81:~/kubernetes$ ./deploy.sh
namespace/valaxy created
secret/jfrogcred created
deployment.apps/valaxy-rtp created
service/valaxy-rtp-service created
ubuntu@ip-10-1-1-81:~/kubernetes$
ubuntu@ip-10-1-1-81:~/kubernetes$
ubuntu@ip-10-1-1-81:~/kubernetes$ kubectl get all -n valaxy
NAME                              READY   STATUS    RESTARTS   AGE
pod/valaxy-rtp-5d5b7d84c4-4dkgv   1/1     Running   0          28s
pod/valaxy-rtp-5d5b7d84c4-h85qw   1/1     Running   0          28s

NAME                         TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/valaxy-rtp-service   NodePort   172.20.151.236   <none>        8000:30082/TCP   27s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/valaxy-rtp   2/2     2            2           28s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/valaxy-rtp-5d5b7d84c4   2         2         2       28s
ubuntu@ip-10-1-1-81:~/kubernetes$
```

==========

Deploying application on kubernetes

Create directories called kubernetes

Write manifest files under kubernetes: have written manifest file to create Namespace



Run command to create namespace
'kubectl apply -f namespace.yml'
'kubectl get ns' will give all namespace available in kubernetes



Write deployment.yaml manifest file

NOTE:

why did configure the environment variables in the manifest file?
Consumer_key, consuemer_token, Access_token, Access_token_secret

These are application specific token which has been created by application team.

Just like if you use mysql you have to pass some variable like username and password. If you dont use mysql will not start. This is how mysql has been design by application team.

Similarly this is how this project is designed by application team. you don't need to bother about it. They will share you documentation how to use any particular image.



Run command 'kubectl apply -f deployment.yaml' to deploy deployment.yaml



Describe your pod to see the error

```
root@ip-10-1-1-59:/opt/kubernetes#  kubectl describe pod/valaxy-rtp-cb8f5f648-nzjs2 -n namg
Name:             valaxy-rtp-cb8f5f648-nzjs2
Namespace:        namg
Priority:         0
Service Account:  default
Node:             ip-10-1-2-86.ec2.internal/10.1.2.86
Start Time:       Sun, 01 Oct 2023 06:07:07 +0000
Labels:           app=valaxy-rtp
                  pod-template-hash=cb8f5f648
Annotations:      <none>
Status:           Pending
IP:               10.1.2.160
IPs:
  IP:           10.1.2.160
Controlled By:  ReplicaSet/valaxy-rtp-cb8f5f648
Containers:
  valaxy-rtp:
    Container ID:
    Image:          namg04.jfrog.io/namg-docker/namtrend:2.1.2
    Image ID:
    Port:           8000/TCP
    Host Port:      0/TCP
    State:          Waiting
      Reason:       ImagePullBackOff
    Ready:          False
    Restart Count:  0
    Environment:
      CONSUMER_KEY:         G6lmKhsi0V9TvXt6oKTfjRBCr
      CONSUMER_SECRET:      bEyDk8X0p8SQd4376eoNV4nH8To22sHcJOoFt0Zq0KS37hhI4q
      ACCESS_TOKEN:         9703354-52rXHD6EeOQeYyhtNz1w8UVOgbcLcgOo4O7MB6WV3
      ACCESS_TOKEN_SECRET:  zBThlZDEp8qnu7NwwdHNth8eg3Rf9zqbvUEFUTaZtN2SF
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-lw96v (ro)
Conditions:
```

Warning Failed 17m (x4 over 18m) kubelet Failed to pull image "namg04.jfrog.io/namg-docker/namtrend:2.1.2": rpc error: code = Unknown desc = failed to pull and unpack image "namg04.jfrog.io/namg-docker/namtrend:2.1.2": failed to resolve reference "namg04.jfrog.io/namg-docker/namtrend:2.1.2": **failed to authorize: failed to fetch anonymous token: unexpected status: 401**
Warning Failed 17m (x4 over 18m) kubelet Error: ErrImagePull
Warning Failed 16m (x6 over 18m) kubelet Error: ImagePullBackOff
Normal BackOff 3m15s (x66 over 18m) kubelet Back-off pulling image "namg04.jfrog.io/namg-docker/namtrend:2.1.2"

```
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-lw96v (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  kube-api-access-lw96v:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  18m                  default-scheduler  Successfully assigned namg/valaxy-rtp-cb8f5f648-nzjs2 to ip-10-1-2-86.ec2.internal
  Normal   Pulling    17m (x4 over 18m)    kubelet            Pulling image "namg04.jfrog.io/namg-docker/namtrend:2.1.2"
  Warning  Failed     17m (x4 over 18m)    kubelet            Failed to pull image "namg04.jfrog.io/namg-docker/namtrend:2.1.2": rpc error: code = Unknown desc = failed to pull and unp
ack image "namg04.jfrog.io/namg-docker/namtrend:2.1.2": failed to resolve reference "namg04.jfrog.io/namg-docker/namtrend:2.1.2": failed to authorize: failed to fetch anonymous token: u
nexpected status: 401
  Warning  Failed     17m (x4 over 18m)    kubelet            Error: ErrImagePull
  Warning  Failed     16m (x6 over 18m)    kubelet            Error: ImagePullBackOff
  Normal   BackOff    3m15s (x66 over 18m)  kubelet            Back-off pulling image "namg04.jfrog.io/namg-docker/namtrend:2.1.2"
root@ip-10-1-1-59:/opt/kubernetes#
```

To resolve this error we need to use the secrets so that it can able to communicate with our jfrog artifactory to pull the image.

```
## Integrate Jfrog with Kubernetes cluster

Create a dedicated user to use for a docker login
     user menu --> new user
     `user name`: jfrogcred
     `email address`: logintoaws@gmail.com
     `password`: <passwrod>
```

````
To pull an image from jfrog at the docker level, we should log into jfrog using username and password
```sh
 docker login https://namg04.jfrog.io
```

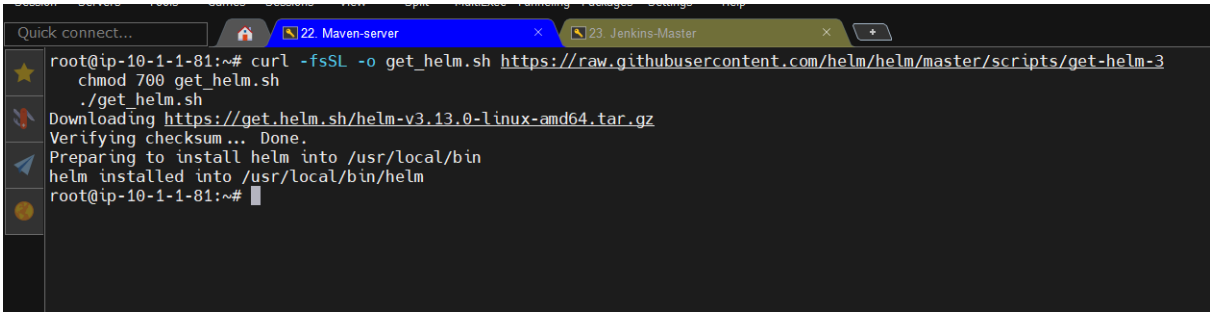
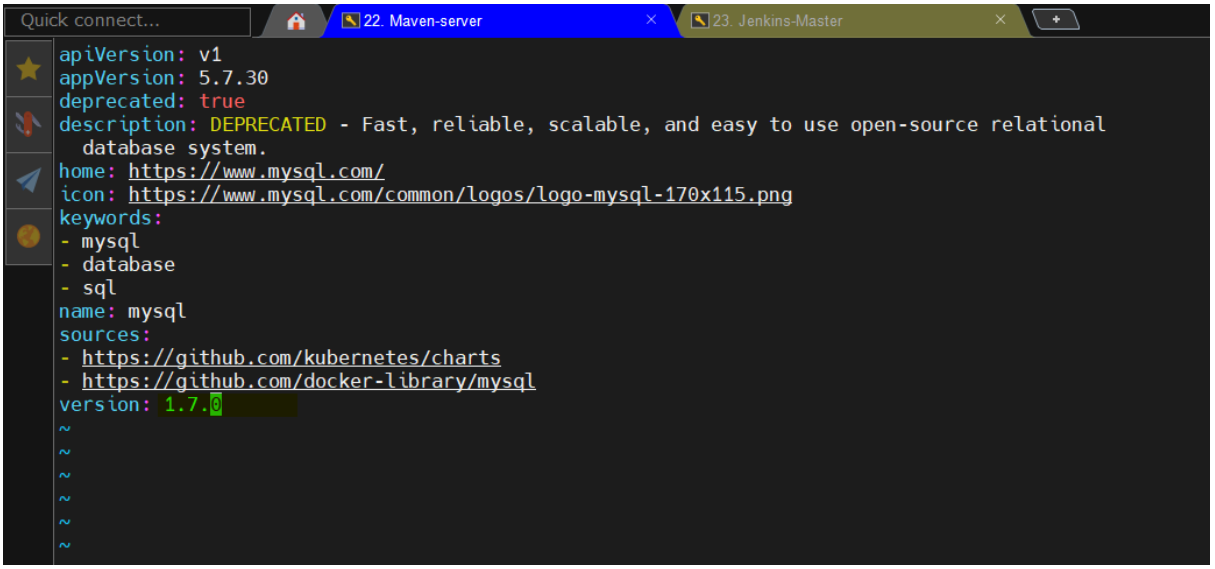
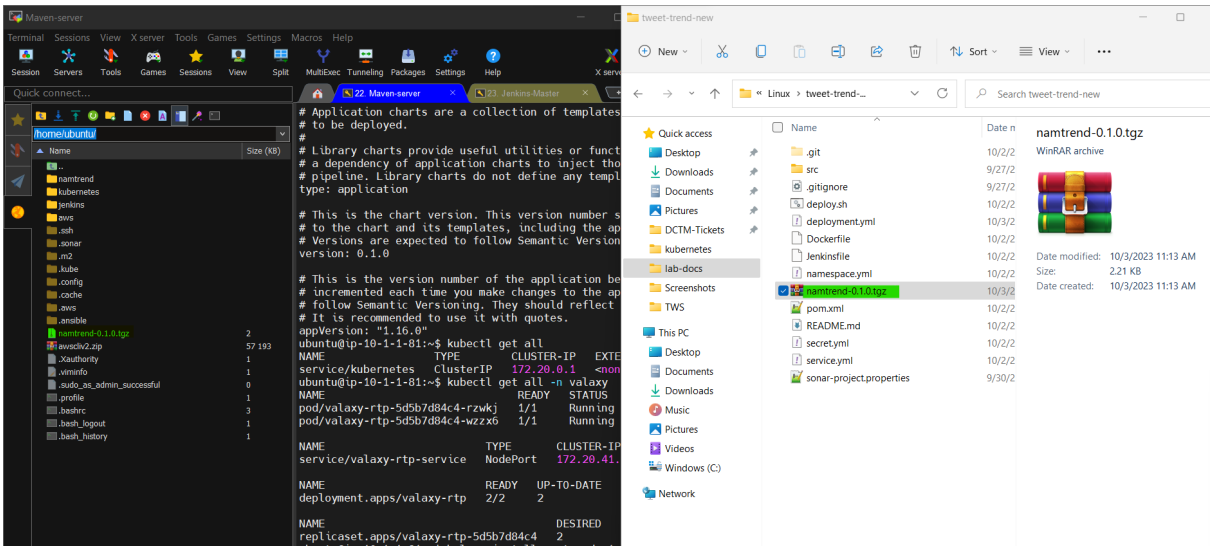
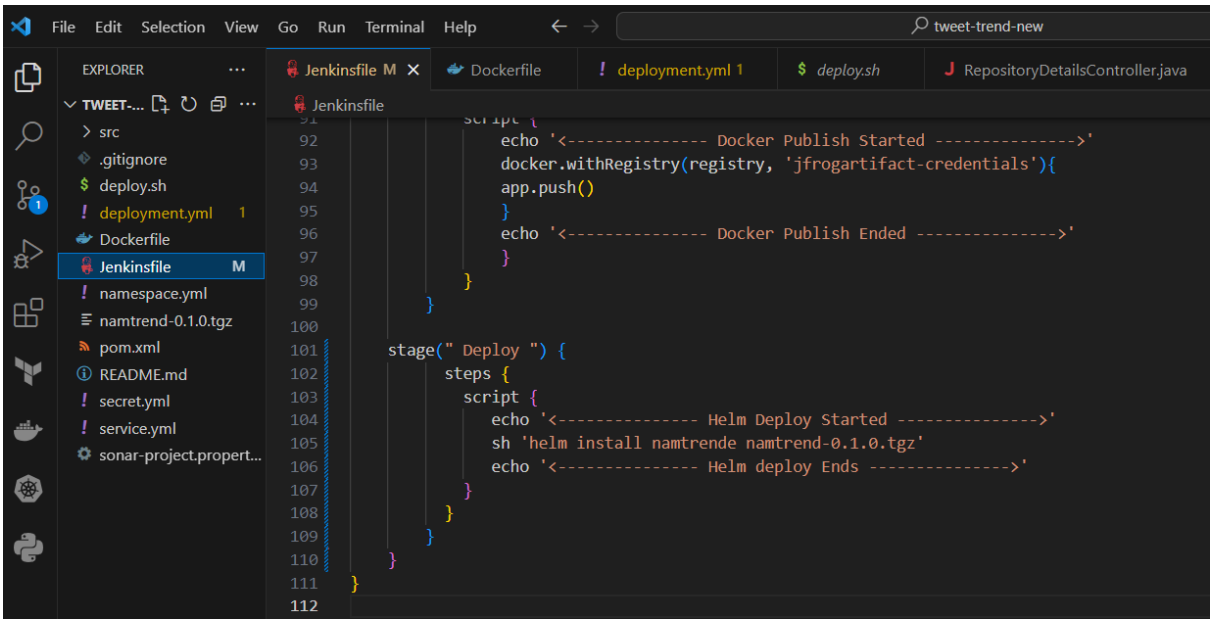
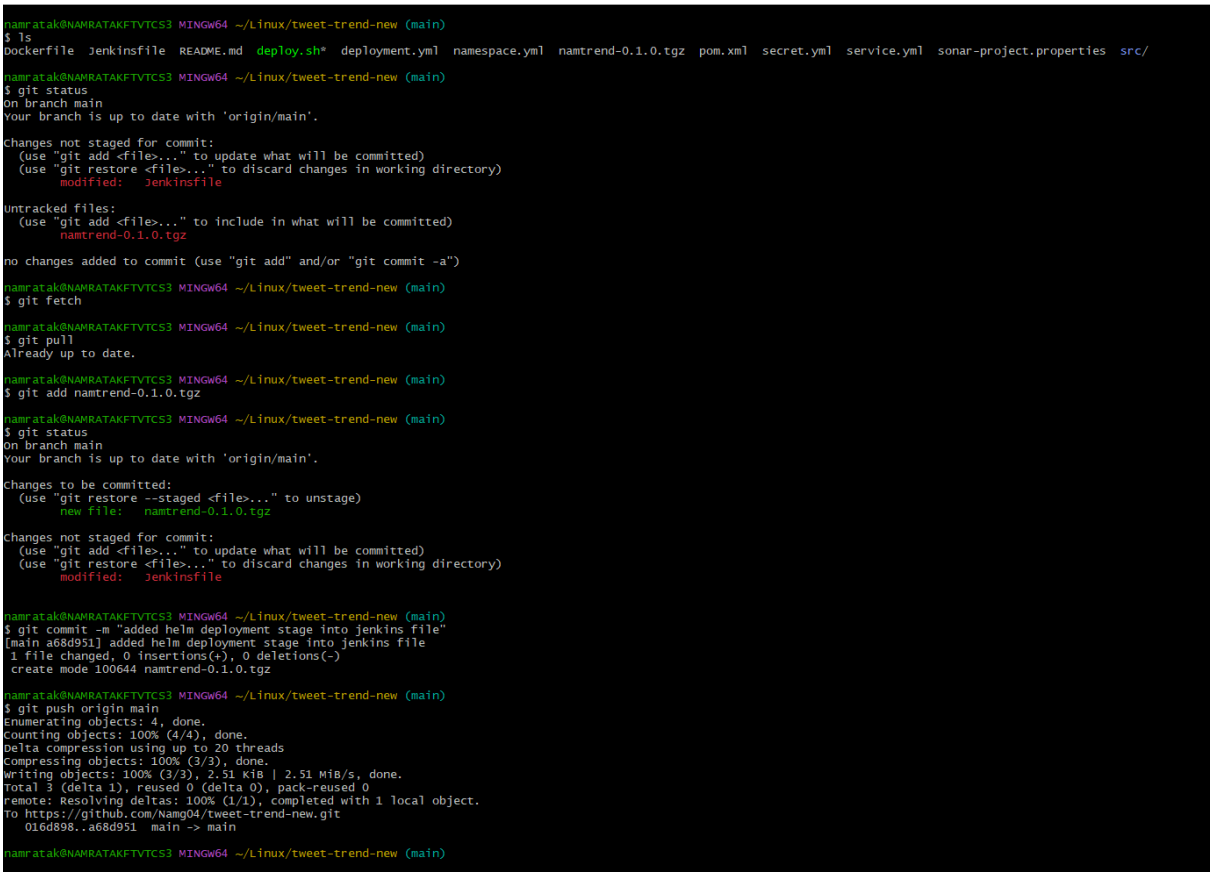
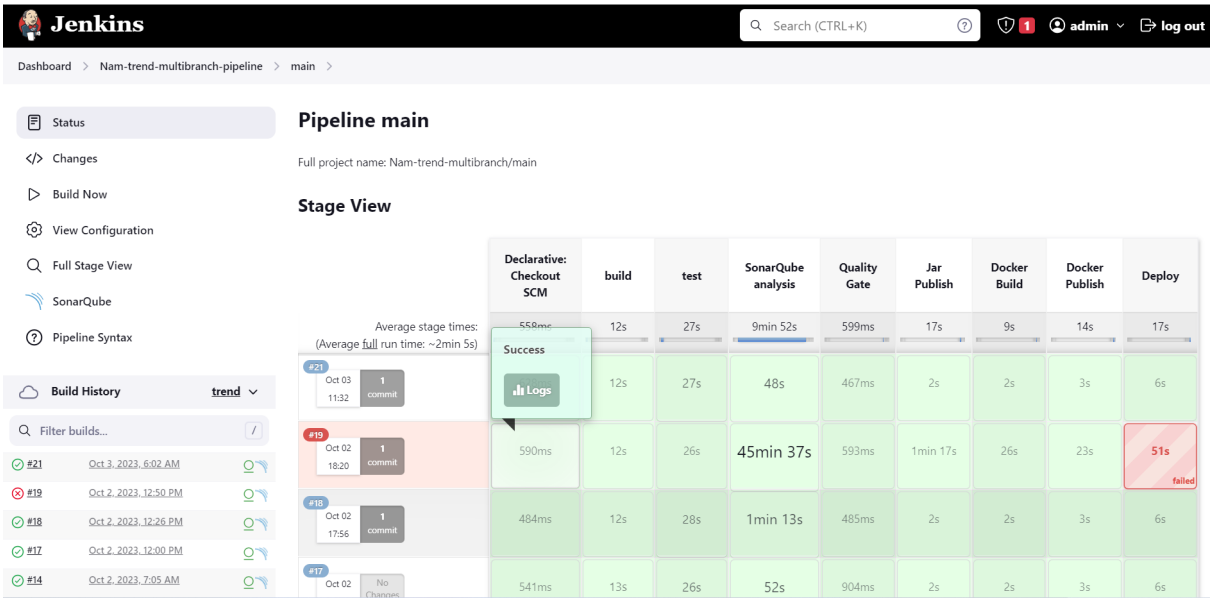
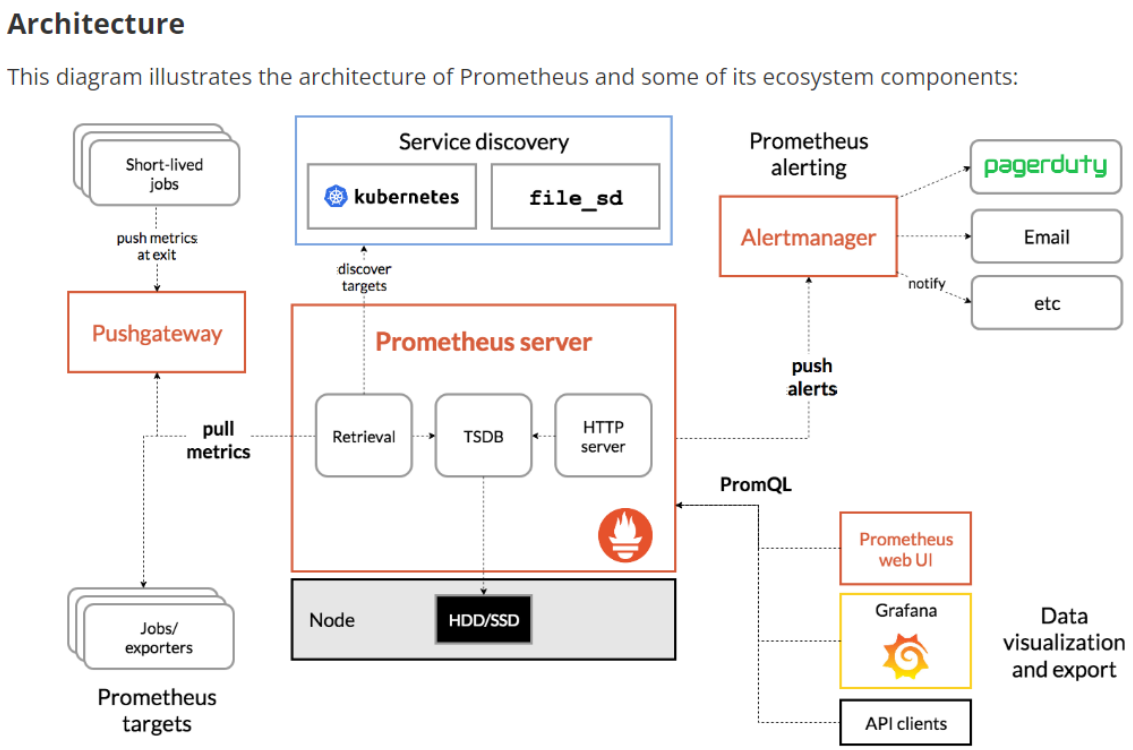
genarate encode value for ~/.docker/config.json file
  ```sh
   cat ~/.docker/config.json | base64 -w0
   ```

`Note:` For more refer to [Kuberentes documentation](https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-regist
Make sure secret value name `regcred` is updated in the deployment file.

  `copy auth value to encode`
  cat ~/.docker/config.json | base64 -w0
````

Create docker login user




Add new user
Create user
Invite user
User Settings
* User Name
dockercred
* Email Address
sunny.ime05@gmail.com
Roles
Administer Platform
Manage Resources
Manage Policies
Read Policies
Manage Watches
Manage Reports
Cancel


Add new user
Password
* Password
S@nam291121
* Retype Password
Multi-factor Authentication Status
Reset Google MFA Enrollment
Related Groups
0 Available Group
1 Selected Group
Search
Search
Cancel
Reset
Save



save password in notepad



write secret.yml file



```
root@ip-10-1-1-81:/opt/kubernetes# kubectl apply -f secret.yml
secret/jfrogcred created
root@ip-10-1-1-81:/opt/kubernetes# kubectl get secret -n valaxy
NAME        TYPE                             DATA   AGE
jfrogcred   kubernetes.io/dockerconfigjson   1      3m6s
root@ip-10-1-1-81:/opt/kubernetes#
```

create service.yml

```
root@ip-10-1-1-81:/opt/kubernetes# vi service.yml
root@ip-10-1-1-81:/opt/kubernetes# cat service.yml
apiVersion: v1
kind: Service
metadata:
  name:  valaxy-rtp-service
  namespace: valaxy
spec:
  type: NodePort
  selector:
    app: valaxy-rtp
  ports:
  - nodePort: 30082
    port: 8000
    targetPort: 8000
root@ip-10-1-1-81:/opt/kubernetes#  kubectl apply -f service.yml
service/valaxy-rtp-service created
root@ip-10-1-1-81:/opt/kubernetes# kubectl get all -n valaxy
NAME                              READY   STATUS    RESTARTS   AGE
pod/valaxy-rtp-5d5b7d84c4-hp7gm   1/1     Running   0          75m
pod/valaxy-rtp-5d5b7d84c4-hrwhg   1/1     Running   0          75m

NAME                         TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/valaxy-rtp-service   NodePort   172.20.159.234   <none>        8000:30082/TCP
20s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/valaxy-rtp   2/2     2            2           75m

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/valaxy-rtp-5d5b7d84c4   2         2         2       75m
root@ip-10-1-1-81:/opt/kubernetes#
```

open port 30082 in security group



Greetings from NamG DevTalks

Using deploy.sh file

So far, we have seen how to deploy our application as a microservice and how we can access this outside of Kubernetes cluster.

```
root@ip-10-1-1-81:/opt/kubernetes# cd ..
root@ip-10-1-1-81:/opt# ls
apache-maven-3.9.4  apache-maven-3.9.4-bin.tar.gz  containerd  kubernetes
root@ip-10-1-1-81:/opt# mv kubernetes/ /home/ubuntu/
root@ip-10-1-1-81:/opt# chown -R ubuntu:ubuntu /home/ubuntu/
root@ip-10-1-1-81:/opt# ls -l /home/ubuntu/
total 57212
drwxr-xr-x 3 ubuntu ubuntu     4096 Sep 27 16:44 aws
-rw-r--r-- 1 ubuntu ubuntu 58565974 Oct  2 09:13 awscliv2.zip
drwx------ 6 ubuntu ubuntu     4096 Oct  2 04:00 jenkins
drwxr-xr-x 2 ubuntu ubuntu     4096 Oct  2 10:47 kubernetes
root@ip-10-1-1-81:/opt# ls -l /home/ubuntu/kubernetes/
total 16
-rw-r--r-- 1 ubuntu ubuntu 850 Oct  2 10:18 deployment.yml
-rw-r--r-- 1 ubuntu ubuntu  55 Oct  2 09:20 namespace.yml
-rw-r--r-- 1 ubuntu ubuntu 286 Oct  2 10:34 secret.yml
-rw-r--r-- 1 ubuntu ubuntu 207 Oct  2 10:47 service.yml
root@ip-10-1-1-81:/opt#
```

switch to ubuntu user

```
root@ip-10-1-1-81:/opt# sudo su - ubuntu
ubuntu@ip-10-1-1-81:~$ kubectl get all
E1002 11:05:55.975783    3918 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refu
sed
E1002 11:05:55.976842    3918 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refu
sed
E1002 11:05:55.977821    3918 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refu
sed
E1002 11:05:55.979371    3918 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refu
sed
E1002 11:05:55.980884    3918 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refu
sed
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

so this command doesn't work because we don't have the cluster credentials.

==========

Commit manifests in GitHub



Drag manifests files from Mobaxterm to Project source code folder



Commit the code into Github

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ ls
Dockerfile   README.md   deployment.yml  pom.xml      service.yml              src/
Jenkinsfile  deploy.sh*  namespace.yml   secret.yml   sonar-project.properties

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ ls -ltr
total 19
drwxr-xr-x 1 namratak 1049089    0 Sep 27 07:29 src/
-rw-r--r-- 1 namratak 1049089 3560 Sep 28 20:56 pom.xml
-rw-r--r-- 1 namratak 1049089  157 Sep 30 15:00 Dockerfile
-rw-r--r-- 1 namratak 1049089  287 Sep 30 20:43 sonar-project.properties
-rw-r--r-- 1 namratak 1049089  253 Oct  2 10:15 README.md
-rw-r--r-- 1 namratak 1049089 3508 Oct  2 11:23 Jenkinsfile
-rw-r--r-- 1 namratak 1049089  207 Oct  2 17:00 service.yml
-rw-r--r-- 1 namratak 1049089  286 Oct  2 17:01 secret.yml
-rw-r--r-- 1 namratak 1049089   55 Oct  2 17:01 namespace.yml
-rw-r--r-- 1 namratak 1049089  850 Oct  2 17:01 deployment.yml
-rwxr-xr-x 1 namratak 1049089  130 Oct  2 17:01 deploy.sh*

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        deploy.sh
        deployment.yml
        namespace.yml
        secret.yml
        service.yml

nothing added to commit but untracked files present (use "git add" to track)
```

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git fetch

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        deploy.sh
        deployment.yml
        namespace.yml
        secret.yml
        service.yml

nothing added to commit but untracked files present (use "git add" to track)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add deploy.sh deployment.yml namespace.yml service.yml secret.yml
warning: in the working copy of 'deploy.sh', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'deployment.yml', LF will be replaced by CRLF the next time Git touches
 it
warning: in the working copy of 'namespace.yml', LF will be replaced by CRLF the next time Git touches
it
warning: in the working copy of 'secret.yml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'service.yml', LF will be replaced by CRLF the next time Git touches it

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   deploy.sh
        new file:   deployment.yml
        new file:   namespace.yml
        new file:   secret.yml
        new file:   service.yml
```

Give executable permission to deploy.sh file

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add --chmod=+x deployploy.sh
fatal: pathspec 'deployploy.sh' did not match any files

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add --chmod=+x deploy.sh

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "addded kubernetes manifests file"
[main aa01a06] addded kubernetes manifests file
 5 files changed, 68 insertions(+)
 create mode 100755 deploy.sh
 create mode 100644 deployment.yml
 create mode 100644 namespace.yml
 create mode 100644 secret.yml
 create mode 100644 service.yml

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 20 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.55 KiB | 1.55 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Namg04/tweet-trend-new.git
   1fe3b37..aa01a06  main -> main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

Committed in to Github



Build has triggered



SonarQube Quality Gate

Deploying app using Jenkins through deploy.sh



Write deploy stage in Jenkinsfile



Commit the file in Github

```
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Jenkinsfile

no changes added to commit (use "git add" and/or "git commit -a")

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "updated jenkinsfile with deployment stage"
[main 43d70a4] updated jenkinsfile with deployment stage
 1 file changed, 8 insertions(+)

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 365 bytes | 365.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Namg04/tweet-trend-new.git
   aa01a06..43d70a4  main -> main

namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

Build completed automatically



==========

Helm Charts

Helm is a package manager for Kubernetes.

💡 **Package manager nothing but it is a utility like, you know, about the APT or Yum. So whenever we want to install any software or packages we are going to use yum install git, yum install httpd. Even apt install git, apt install apache2. So similar way we are going to use the helm. Similar way we are going to use the helm. It is a package manager which helps to install specific packages. For example, we want to deploy Jenkins server on Kubernetes, then we can use helm install Jenkins, this is how we can use Helm.**

A chart is a package for a Pre-configured Kubernetes resources, so chart is nothing but a manifest file.

💡 **So here, if we want to create any resources on Kubernetes, we need to use the charts.**
These charts are pre-configured nothing, but somebody has already written. We just need to use it. As I said, if we want to install Jenkins, then we can use the predefined charts to install Jenkins on our Kubernetes.

A repository is a group of published charts which can be made available to others.

💡 **Where these charts are located that place we call it as a repository. All predefined charts are available in the repository so we can pull the charts from the repository and we can use it for our purpose.**

Helm is used for the repetitive tasks and applications.

💡 **Helm will be help us to perform the repetitive tasks and applications.**

Helm should be installed on our Jenkins slave (Build server).

💡 **In our case, we need to install Helman Jenkins live.**

````
# Helm setup

1. Install helm
   ```sh
   curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
   chmod 700 get_helm.sh
   ./get_helm.sh
   ```
1. Validate helm installation
   ```sh
   helm version
   helm list
   ```
1. [optional] Download .kube/config file to build the node
   ```sh
   aws eks update-kubeconfig --region us-eats-1 --name namg-eks-01
   ```

1. Setup helm repo
   ```sh
   helm repo list
   helm repo add stable https://charts.helm.sh/stable
   helm repo update
   helm search repo <helm-chart>
   helm search repo stable
   ```

1. Install mysql charts on Kubernetes
   ```sh
   helm install demo-mysql stable/mysql
   ```
1. To pull the package from repo to local
   ```sh
````
````

```
   helm pull stable/mysql
   ```

  Once you have downloaded the helm chart, it comes as a zip file. You should extract it.

  In this directory, you can find
  - templates
  - values.yaml
  - README.md
  - Chart.yaml

  If you'd like to change the chart, please update your templates directory and modify the version (1.6.9 to 1.7.0) in the chart.yaml

then you can run the command to pack it after your update
```sh
 helm package mysql
```

To deploy helm chat
```sh
 helm install mysqldb mysql-1.6.9.tgz
```

Above command deploy MySQL
To check deployment
```sh
 helm list
```
To uninstall
```sh
 helm uninstall mysqldb
```

To install nginx
```sh
 helm repo search nginx
 helm install demo-nginx stable/nginx-ingress
```

Install helm on Jenkins-slave server



Add stable repository

```
root@ip-10-1-1-81:~# helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
root@ip-10-1-1-81:~# helm repo list
NAME    URL
stable  https://charts.helm.sh/stable
root@ip-10-1-1-81:~# helm search repo jenkins
NAME            CHART VERSION   APP VERSION     DESCRIPTION
stable/jenkins  2.5.4           lts             DEPRECATED - Open source continuous integration ...
root@ip-10-1-1-81:~# helm search repo mysql
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
stable/mysql                            1.6.9           5.7.30          DEPRECATED - Fast, reliable, scalable, and easy ...
stable/mysqldump                        2.6.2           2.4.1           DEPRECATED! - A Helm chart to help backup MySQL ...
stable/prometheus-mysql-exporter        0.7.1           v0.11.0         DEPRECATED A Helm chart for prometheus mysql ex ...
stable/percona                          1.2.3           5.7.26          DEPRECATED - free, fully compatible, enhanced,  ...
stable/percona-xtradb-cluster           1.0.8           5.7.19          DEPRECATED - free, fully compatible, enhanced,  ...
stable/phpmyadmin                       4.3.5           5.0.1           DEPRECATED phpMyAdmin is an mysql administratio ...
stable/gcloud-sqlproxy                  0.6.1           1.11            DEPRECATED Google Cloud SQL Proxy
stable/mariadb                          7.3.14          10.3.22         DEPRECATED Fast, reliable, scalable, and easy t ...
root@ip-10-1-1-81:~#
```

To install mysql repository

```
root@ip-10-1-1-81:~# helm search repo jenkins
NAME            CHART VERSION   APP VERSION     DESCRIPTION
stable/jenkins  2.5.4           lts             DEPRECATED - Open source continuous integration ...
root@ip-10-1-1-81:~# helm search repo mysql
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
stable/mysql                            1.6.9           5.7.30          DEPRECATED - Fast, reliable, scalable, and easy ...
stable/mysqldump                        2.6.2           2.4.1           DEPRECATED! - A Helm chart to help backup MySQL ...
stable/prometheus-mysql-exporter        0.7.1           v0.11.0         DEPRECATED A Helm chart for prometheus mysql ex ...
stable/percona                          1.2.3           5.7.26          DEPRECATED - free, fully compatible, enhanced,  ...
stable/percona-xtradb-cluster           1.0.8           5.7.19          DEPRECATED - free, fully compatible, enhanced,  ...
stable/phpmyadmin                       4.3.5           5.0.1           DEPRECATED phpMyAdmin is an mysql administratio ...
stable/gcloud-sqlproxy                  0.6.1           1.11            DEPRECATED Google Cloud SQL Proxy
stable/mariadb                          7.3.14          10.3.22         DEPRECATED Fast, reliable, scalable, and easy t ...
root@ip-10-1-1-81:~#
root@ip-10-1-1-81:~#
root@ip-10-1-1-81:~# helm list
NAME    NAMESPACE       REVISION        UPDATED STATUS  CHART   APP VERSION
root@ip-10-1-1-81:~# helm search repo mysql
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
stable/mysql                            1.6.9           5.7.30          DEPRECATED - Fast, reliable, scalable, and easy ...
stable/mysqldump                        2.6.2           2.4.1           DEPRECATED! - A Helm chart to help backup MySQL ...
stable/prometheus-mysql-exporter        0.7.1           v0.11.0         DEPRECATED A Helm chart for prometheus mysql ex ...
stable/percona                          1.2.3           5.7.26          DEPRECATED - free, fully compatible, enhanced,  ...
stable/percona-xtradb-cluster           1.0.8           5.7.19          DEPRECATED - free, fully compatible, enhanced,  ...
stable/phpmyadmin                       4.3.5           5.0.1           DEPRECATED phpMyAdmin is an mysql administratio ...
stable/gcloud-sqlproxy                  0.6.1           1.11            DEPRECATED Google Cloud SQL Proxy
stable/mariadb                          7.3.14          10.3.22         DEPRECATED Fast, reliable, scalable, and easy t ...
```

helm install demo-mysql stable/mysql

```
root@ip-10-1-1-81:~# helm install demo-mysql stable/mysql
WARNING: This chart is deprecated
NAME: demo-mysql
LAST DEPLOYED: Tue Oct  3 04:38:49 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
MySQL can be accessed via port 3306 on the following DNS name from within your cluster:
demo-mysql.default.svc.cluster.local

To get your root password run:

    MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace default demo-mysql -o jsonpath=
"{.data.mysql-root-password}" | base64 --decode; echo)

To connect to your database:

1. Run an Ubuntu pod that you can use as a client:

    kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash -il

2. Install the mysql client:

    $ apt-get update && apt-get install mysql-client -y

3. Connect using the mysql cli, then provide your password:
    $ mysql -h demo-mysql -p

To connect to your database directly from outside the K8s cluster:
    MYSQL_HOST=127.0.0.1
    MYSQL_PORT=3306

    # Execute the following command to route the connection:
    kubectl port-forward svc/demo-mysql 3306

    mysql -h ${MYSQL_HOST} -P${MYSQL_PORT} -u root -p${MYSQL_ROOT_PASSWORD}
root@ip-10-1-1-81:~#
```

```
root@ip-10-1-1-81:~#
root@ip-10-1-1-81:~#
root@ip-10-1-1-81:~# helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
demo-mysql      default         1               2023-10-03 04:38:49.133036003 +0000 UTC deployed        mysql-1.6.9     5.7.30
root@ip-10-1-1-81:~#
```

```
root@ip-10-1-1-81:~# kubectl get all
NAME                              READY   STATUS    RESTARTS   AGE
pod/demo-mysql-65db77785f-vzqxw   0/1     Pending   0          3m13s

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/demo-mysql   ClusterIP   172.20.148.35   <none>        3306/TCP   3m13s
service/kubernetes   ClusterIP   172.20.0.1      <none>        443/TCP    91m

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/demo-mysql   0/1     1            0           3m13s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/demo-mysql-65db77785f   1         1         0       3m13s
root@ip-10-1-1-81:~#
```

Pull helm charts

```
root@ip-10-1-1-81:~# helm pull stable/mysql
root@ip-10-1-1-81:~# ls
get_helm.sh  mysql-1.6.9.tgz  snap
root@ip-10-1-1-81:~# tar -xvzf mysql-1.6.9.tgz
mysql/Chart.yaml
mysql/values.yaml
mysql/templates/NOTES.txt
mysql/templates/_helpers.tpl
mysql/templates/configurationFiles-configmap.yaml
mysql/templates/deployment.yaml
mysql/templates/initializationFiles-configmap.yaml
mysql/templates/pvc.yaml
mysql/templates/secrets.yaml
mysql/templates/serviceaccount.yaml
mysql/templates/servicemonitor.yaml
mysql/templates/svc.yaml
mysql/templates/tests/test-configmap.yaml
mysql/templates/tests/test.yaml
mysql/.helmignore
mysql/README.md
root@ip-10-1-1-81:~# ls
get_helm.sh  mysql  mysql-1.6.9.tgz  snap
root@ip-10-1-1-81:~# cd mysql
root@ip-10-1-1-81:~/mysql# ls
Chart.yaml  README.md  templates  values.yaml
root@ip-10-1-1-81:~/mysql#
```

Modified Chart.yml



Modified values.yml

```
  resources: {}
  annotations: {}
    # prometheus.io/scrape: "true"
    # prometheus.io/port: "9104"
  livenessProbe:
    initialDelaySeconds: 15
    timeoutSeconds: 5
  readinessProbe:
    initialDelaySeconds: 5
    timeoutSeconds: 1
  flags: []
  serviceMonitor:
    enabled: false
    additionalLabels: {}

## Configure the service
## ref: http://kubernetes.io/docs/user-guide/services/
service:
  annotations: {}
  ## Specify a service type
  ## ref: https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services---service-types
  type: NodePort
  port: 3306
  nodePort: 32000
  # loadBalancerIP:

## Pods Service Account
## ref: https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/
serviceAccount:
  ## Specifies whether a ServiceAccount should be created
  ##
  create: false
  ## The name of the ServiceAccount to use.
  ## If not set and create is true, a name is generated using the mariadb.fullname template
  # name:

ssl:
  enabled: false
  secret: mysql-ssl-certs
  certificates:
#  - name: mysql-ssl-certs
-- INSERT --
```

deploy it- helm install demo-1-mysql mysql

```
root@ip-10-1-1-81:~# helm install demo-2-mysql mysql
WARNING: This chart is deprecated
NAME: demo-2-mysql
LAST DEPLOYED: Tue Oct  3 04:57:36 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
MySQL can be accessed via port 3306 on the following DNS name from within your cluster:
demo-2-mysql.default.svc.cluster.local

To get your root password run:

    MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace default demo-2-mysql -o jsonpath="{.data.mysql-root-password}" | base64 --decode; echo)

To connect to your database:

1. Run an Ubuntu pod that you can use as a client:

    kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash -il

2. Install the mysql client:

    $ apt-get update && apt-get install mysql-client -y

3. Connect using the mysql cli, then provide your password:
    $ mysql -h demo-2-mysql -p

To connect to your database directly from outside the K8s cluster:
    MYSQL_HOST=$(kubectl get nodes --namespace default -o jsonpath='{.items[0].status.addresses[0].address}')
    MYSQL_PORT=$(kubectl get svc --namespace default demo-2-mysql -o jsonpath='{.spec.ports[0].nodePort}')

    mysql -h ${MYSQL_HOST} -P${MYSQL_PORT} -u root -p${MYSQL_ROOT_PASSWORD}
root@ip-10-1-1-81:~# helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
demo-2-mysql    default         1               2023-10-03 04:57:36.679173065 +0000 UTC deployed        mysql-1.7.0     5.7.30
demo-mysql      default         1               2023-10-03 04:38:49.133036003 +0000 UTC deployed        mysql-1.6.9     5.7.30
root@ip-10-1-1-81:~#
```

This is how you can able to do changes to your predefined helm charts according to your requirement and deploy it.

```
root@ip-10-1-1-81:~# helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
demo-2-mysql    default         1               2023-10-03 04:57:36.679173065 +0000 UTC deployed        mysql-1.7.0     5.7.30
demo-mysql      default         1               2023-10-03 04:38:49.133036003 +0000 UTC deployed        mysql-1.6.9     5.7.30
root@ip-10-1-1-81:~# kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/demo-2-mysql-7884c687df-fbgln   0/1     Pending   0          2m19s
pod/demo-mysql-65db77785f-vzqxw     0/1     Pending   0          21m

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/demo-2-mysql   NodePort    172.20.211.65   <none>        3306:32000/TCP   2m19s
service/demo-mysql     ClusterIP   172.20.148.35   <none>        3306/TCP         21m
service/kubernetes     ClusterIP   172.20.0.1      <none>        443/TCP          109m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/demo-2-mysql   0/1     1            0           2m19s
deployment.apps/demo-mysql     0/1     1            0           21m

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/demo-2-mysql-7884c687df   1         1         0       2m19s
replicaset.apps/demo-mysql-65db77785f     1         1         0       21m
root@ip-10-1-1-81:~#
```

To delete it

```
root@ip-10-1-1-81:~# helm uninstall demo-2-mysql
release "demo-2-mysql" uninstalled
root@ip-10-1-1-81:~# kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/demo-mysql-65db77785f-vzqxw     0/1     Pending   0          24m

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/demo-mysql   ClusterIP   172.20.148.35   <none>        3306/TCP   24m
service/kubernetes   ClusterIP   172.20.0.1      <none>        443/TCP    113m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/demo-mysql     0/1     1            0           24m

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/demo-mysql-65db77785f     1         1         0       24m
root@ip-10-1-1-81:~# helm uninstall demo-mysql
release "demo-mysql" uninstalled
root@ip-10-1-1-81:~# kubectl get all
NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   172.20.0.1      <none>        443/TCP   113m
root@ip-10-1-1-81:~#
```

How to deploy Namtrend app using helm chart?

So over here we are going to see how to create a helm chart and how we can deploy our own namtrend application for this.

First, we need to create a helm chart to create a helm chart.

```
root@ip-10-1-1-81:~# ls
get_helm.sh  mysql  mysql-1.6.9.tgz  snap
root@ip-10-1-1-81:~# exit
exit
ubuntu@ip-10-1-1-81:~/kubernetes$ ls
deploy.sh  deployment.yml  namespace.yml  secret.yml  service.yml
ubuntu@ip-10-1-1-81:~/kubernetes$ cd ..
ubuntu@ip-10-1-1-81:~$ ls
aws  awscliv2.zip  jenkins  kubernetes
ubuntu@ip-10-1-1-81:~$ helm create namtrend
Creating namtrend
ubuntu@ip-10-1-1-81:~$ ls
aws  awscliv2.zip  jenkins  kubernetes  namtrend
ubuntu@ip-10-1-1-81:~$ cd namtrend/
ubuntu@ip-10-1-1-81:~/namtrend$ ls
Chart.yaml  charts  templates  values.yaml
ubuntu@ip-10-1-1-81:~/namtrend$
```

Deleted existing manifest files and copied own manifest files created for namtrend application

```
ubuntu@ip-10-1-1-81:~$ cd namtrend/
ubuntu@ip-10-1-1-81:~/namtrend$ ls
Chart.yaml  charts  templates  values.yaml
ubuntu@ip-10-1-1-81:~/namtrend$ cd templates/
ubuntu@ip-10-1-1-81:~/namtrend/templates$ ls
NOTES.txt  _helpers.tpl  deployment.yaml  hpa.yaml  ingress.yaml  service.yaml  serviceaccount.yaml  tests
ubuntu@ip-10-1-1-81:~/namtrend/templates$ rm -rf *
ubuntu@ip-10-1-1-81:~/namtrend/templates$ ls
ubuntu@ip-10-1-1-81:~/namtrend/templates$ cd ..
ubuntu@ip-10-1-1-81:~/namtrend$ ls
Chart.yaml  charts  templates  values.yaml
ubuntu@ip-10-1-1-81:~/namtrend$  cd..
cd..: command not found
ubuntu@ip-10-1-1-81:~/namtrend$ cd ..
ubuntu@ip-10-1-1-81:~$ ls
aws  awscliv2.zip  jenkins  kubernetes  namtrend
ubuntu@ip-10-1-1-81:~$ cd kubernetes/
ubuntu@ip-10-1-1-81:~/kubernetes$ ls
deploy.sh  deployment.yml  namespace.yml  secret.yml  service.yml
ubuntu@ip-10-1-1-81:~/kubernetes$ mv deployment.yml namespace.yml service.yml secret.yml ../namtrend/templates
ubuntu@ip-10-1-1-81:~/kubernetes$ ls
deploy.sh
ubuntu@ip-10-1-1-81:~/kubernetes$ cd ..
ubuntu@ip-10-1-1-81:~$ ls
aws  awscliv2.zip  jenkins  kubernetes  namtrend
ubuntu@ip-10-1-1-81:~$ cd namtrend/
ubuntu@ip-10-1-1-81:~/namtrend$ ls
Chart.yaml  charts  templates  values.yaml
ubuntu@ip-10-1-1-81:~/namtrend$ cd templates/
ubuntu@ip-10-1-1-81:~/namtrend/templates$ ls
deployment.yml  namespace.yml  secret.yml  service.yml
ubuntu@ip-10-1-1-81:~/namtrend/templates$
```

Deploy it

```
ubuntu@ip-10-1-1-81:~/namtrend/templates$ kubectl get all -n vaxaly
No resources found in vaxaly namespace.
ubuntu@ip-10-1-1-81:~/namtrend/templates$ kubectl get all -n valaxy
No resources found in valaxy namespace.
ubuntu@ip-10-1-1-81:~/namtrend/templates$ kubectl delete ns valaxy
Error from server (NotFound): namespaces "valaxy" not found
ubuntu@ip-10-1-1-81:~/namtrend/templates$ kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   172.20.0.1   <none>        443/TCP   128m
ubuntu@ip-10-1-1-81:~/namtrend/templates$ kubectl get ns
NAME              STATUS   AGE
default           Active   129m
kube-node-lease   Active   129m
kube-public       Active   129m
kube-system       Active   129m
```

```
ubuntu@ip-10-1-1-81:~/namtrend/templates$ ls
deployment.yml  namespace.yml  secret.yml  service.yml
ubuntu@ip-10-1-1-81:~/namtrend/templates$ cd ..
ubuntu@ip-10-1-1-81:~/namtrend$ ls
Chart.yaml  charts  templates  values.yaml
ubuntu@ip-10-1-1-81:~/namtrend$ cd ..
ubuntu@ip-10-1-1-81:~$ ls
aws  awscliv2.zip  jenkins  kubernetes  namtrend
ubuntu@ip-10-1-1-81:~$ helm install namtrend-v1 namtrend
NAME: namtrend-v1
LAST DEPLOYED: Tue Oct  3 05:21:12 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
ubuntu@ip-10-1-1-81:~$
```

```
ubuntu@ip-10-1-1-81:~$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 S
TATUS   CHART           APP VERSION
namtrend-v1     default         1               2023-10-03 05:21:12.766719077 +0000 UTCd
eployed namtrend-0.1.0  1.16.0
ubuntu@ip-10-1-1-81:~$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
namtrend-v1     default         1               2023-10-03 05:21:12.766719077 +0000 UTC deployed        namtrend-0.1.0  1.16.0
ubuntu@ip-10-1-1-81:~$ ls
aws  awscliv2.zip  jenkins  kubernetes  namtrend
ubuntu@ip-10-1-1-81:~$ cat namtrend/Chart.yaml
apiVersion: v2
name: namtrend
description: A Helm chart for Kubernetes

# A chart can be either an 'application' or a 'library' chart.
#
# Application charts are a collection of templates that can be packaged into versioned archives
# to be deployed.
#
# Library charts provide useful utilities or functions for the chart developer. They're included as
# a dependency of application charts to inject those utilities and functions into the rendering
# pipeline. Library charts do not define any templates and therefore cannot be deployed.
type: application

# This is the chart version. This version number should be incremented each time you make changes
# to the chart and its templates, including the app version.
# Versions are expected to follow Semantic Versioning (https://semver.org/)
version: 0.1.0

# This is the version number of the application being deployed. This version number should be
# incremented each time you make changes to the application. Versions are not expected to
# follow Semantic Versioning. They should reflect the version the application is using.
# It is recommended to use it with quotes.
appVersion: "1.16.0"
ubuntu@ip-10-1-1-81:~$
```

This is how we can deploy our microservice by using the helm charts

```
ubuntu@ip-10-1-1-81:~$ kubectl get all
NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   172.20.0.1    <none>        443/TCP   135m
ubuntu@ip-10-1-1-81:~$ kubectl get all -n valaxy
NAME                                 READY   STATUS    RESTARTS   AGE
pod/valaxy-rtp-5d5b7d84c4-rzwkj   1/1     Running   0          4m34s
pod/valaxy-rtp-5d5b7d84c4-wzzx6   1/1     Running   0          4m34s

NAME                          TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
service/valaxy-rtp-service   NodePort   172.20.41.10   <none>        8000:30082/TCP   4m34s

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/valaxy-rtp   2/2     2            2           4m34s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/valaxy-rtp-5d5b7d84c4   2         2         2       4m34s
ubuntu@ip-10-1-1-81:~$
```

==========

Helm chart ‘namtrend’ app deployment using Jenkins

```
# Create a custom Helm chart

1. To create a helm chart template
   ```sh
   helm create ttrend
   ```

    by default, it contains
    - values.yaml
    - templates
    - Charts.yaml
    - charts

2. Replace the template directory with the manifest files and package it
   ```sh
   helm package ttrend
   ```
3. Change the version number in the
   ```sh
   helm install ttrend ttrend-0.1.0.tgz
```
```

```
4. Create a jenkins job for the deployment
   ```sh
   stage(" Deploy ") {
          steps {
            script {
               echo '<--------------- Helm Deploy Started --------------->'
               sh 'helm install ttrend ttrend-0.1.0.tgz'
               echo '<--------------- Helm deploy Ends --------------->'
            }
          }
        }
   ```

5. To list installed helm deployments
   ```sh
   helm list -a
   ```

Other useful commands
1. to change the default namespace to valaxy
   ```sh
   kubectl config set-context --current --namespace=valaxy
   ```
```

Delete deployments

```
ubuntu@ip-10-1-1-81:~$ kubectl get all -n valaxy
NAME                                READY   STATUS    RESTARTS   AGE
pod/valaxy-rtp-5d5b7d84c4-rzwkj     1/1     Running   0          4m34s
pod/valaxy-rtp-5d5b7d84c4-wzzx6     1/1     Running   0          4m34s

NAME                            TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/valaxy-rtp-service      NodePort   172.20.41.10    <none>        8000:30082/TCP   4m34s

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/valaxy-rtp      2/2     2            2           4m34s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/valaxy-rtp-5d5b7d84c4      2         2         2       4m34s
ubuntu@ip-10-1-1-81:~$ helm uninstall namtrend-v1
release "namtrend-v1" uninstalled
ubuntu@ip-10-1-1-81:~$ kubectl get all -n valaxy
No resources found in valaxy namespace.
ubuntu@ip-10-1-1-81:~$ helm package namtrend
Successfully packaged chart and saved it to: /home/ubuntu/namtrend-0.1.0.tgz
ubuntu@ip-10-1-1-81:~$
```

helm charts copied into source code repository (tweet-trend-new)



Added deploy stage into Jenkinsfile



Commit the code into Github



Pipelines success



==========

Prometheus and Grafana

So far we have implemented pulling the code from the GitHub and build it with help of Jenkins slave. We have done SonarQube analysis on the source code. Next we have Build a Jar file and published in the Jfrog Artifactory then we converted that into a Docker image and published into the Jfrog Artifactory. After that, we have deployed it into the Kubernetes cluster by using Helm.
Now we would like to monitor our Kubernetes cluster for that we can use Prometheus and Grafana.

- **Prometheus is an open source system monitoring and alerting toolkit which helps us to monitor systems.**
- **And in case if we find any abnormal behavior, it is going to send the alerts.**
- **Prometheus collects and stores The Matrix as a Time series data. It goes and collects the data from the sources and it keeps in the server, in the server.**
  It stores as a time series data.
  For example, let's take a there is a web server.
  That web server will have requests.
  So the requests will be changing time to time now it could be 100 requests.
  After five minutes, it could be 120.
  After ten minutes it could be 150.
  So like that, based on time, the request numbers are going to change.
  So that time series data, it is going to help us to store.
- **Prometheus scraps targets.**
  Nothing but Prometheus itself goes and collects the data from the targets.
  Targets doesn't sends the data to the Prometheus.
- **Next Promql is the language to query time series in Prometheus.**
  In the Prometheus we store matrix If we want to retrieve, we need to use the promql language.
- **We can just give the request.**
  That request will convert it into the promql language and it queries the data.

- **Service Discovery helps find out services and monitor them. So by default Prometheus can able to monitor some of the services that will be done through the help of service discovery.**
  Even we don't need to install anything on Kubernetes. Kubernetes will be monitored by Prometheus by default because it is part of service discovery.
- **Exporters helps to monitor third party components. Prometheus does not able to identify those targets by using the service discovery. In such cases, we need to install the exporters. Exporters are nothing but an agent for the Prometheus.**
  Let's take you are running a nginx server.
  We want to get the information of nginx server.
  How many requests are there?
  How it is performing?
  Then we need to install exporters on the nginx because by default service discovery is not available for Nginx.
  This is just an example. In such cases we need to install the exporters.
- **Exporters are nothing but just like an agent next to thing, Prometheus can send alerts to the alert manager.**
- **By default, Prometheus comes up with an alert manager. Alert manager is like a GUI, which helps us to find out what are the alerts are there.**
- **Instead of alert manager, we are going to use the Grafana. Grafana and alert manager both do the same tasks, but alert manager is the inbuilt feature in the Prometheus.**
- **Prometheus runs on port number 9090 and alert manager runs on 9093.**
  So these two will be in the same server.
  Prometheus by default runs on port number 9090 and alert manager 9093.

Prometheus architecture.



First let's understand about the retrieval retrieval nothing but it pulls the matrix from various sources. Various sources nothing but from the service.

Discovery Service nothing but Prometheus can able to identify some of the services by default among them, Kubernetes is one of the service. You don't need to install any agent on the Kubernetes cluster. By default, it can able to monitor the Kubernetes cluster. For that, it uses the service discovery, discover targets and retrieve the matrix from the Kubernetes cluster.

Similar way we can monitor by using the exporters. As said exporters are agents. Some services are not possible to monitor by Prometheus by default in such cases we need to use the exporters. If you install exporters, exporters will help us to pull the matrix onto the target server. I mean to say Prometheus server.

Next thing is short lived jobs nothing but some jobs will be just instantly run and it disappears. Those kind of jobs we call it as a short lived jobs. Those jobs collects the data and it push to the pushgateway. So PushGateway will have all the matrix. That data will be pulled by the retrieval. So the retrieval major task is collect all the matrix from the various sources.

Once it is collected the data, it stores it in the Tsdb. Tsdb stands for Time series Database. TSDB will have all the matrix with the different intervals. I want to know how my web server is performing. Our database server is performing in such cases.

If I see the data from past couple of days or from past couple of weeks, how many requests are coming in each time then only I will come to know that how it is performing now and if there is any degradation in the performance, I can try to increase the capacity that will be possible only in case if you collect the matrix in the different time intervals. Let's take that every five minutes. I want to monitor Http requests are every five minutes. I want to see how many database connections are there so that data will be stored in the time series database.

Next http server http server can help us to pull the data to the alert manager and alert manager is going to see whether the alerts are normal or abnormal in case if it is abnormal, it is going to send the alert to the different sources. It can be a pagerduty or email or slack or MS teams.

Let's check the CPU utilization is more or more requests are there are more connections are there in the database?

All these kind of things can be monitored by the alert manager and it will send the notification.

Same data you can convert it as a data visualization with the help of Prometheus, Web UI or Grafana or any other clients.

So same data you can see over here as a graphical user interface with the nice graphs so that you can take appropriate decision and also these alerts can be monitored over here.

That's the reason Prometheus and Grafana works together to give the better solution.

==========

Prometheus Setup

````
# Prometheus setup
### pre-requisites
1. Kubernetes cluster
2. helm

## Setup Prometheus

1. Create a dedicated namespace for prometheus
   ```sh
   kubectl create namespace monitoring
   ```

2. Add Prometheus helm chart repository
   ```sh
   helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
   ```

3. Update the helm chart repository
   ```sh
   helm repo update
   helm repo list
   ```

4. Install the prometheus

   ```sh
    helm install prometheus prometheus-community/kube-prometheus-stack --namespace monitoring
   ```

5. Above helm create all services as ClusterIP. To access Prometheus out side of the cluster, we should change the service type load ba
   ```sh
   kubectl edit svc prometheus-kube-prometheus-prometheus -n monitoring

   ```
````

6. Loginto Prometheus dashboard to monitor application
   https://ELB:9090

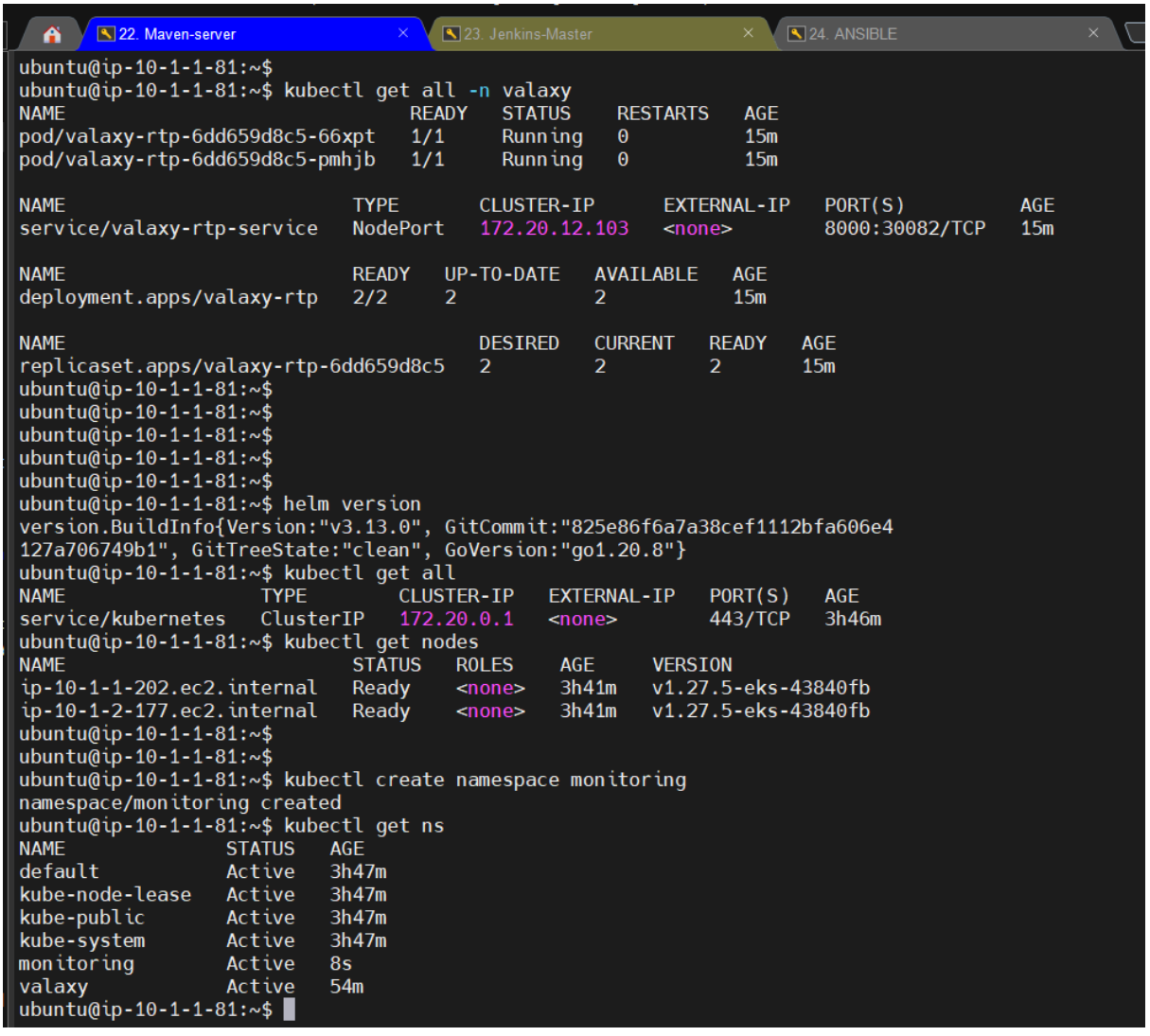
7. Check for node_load15 executor to check cluster monitoring

8. We check similar graphs in the Grafana dashboard itself. for that, we should change the service type of Grafana to LoadBalancer
   ```sh
   kubectl edit svc prometheus-grafana
   ```

9. To login to Grafana account, use the below username and password
    ```sh
    username: admin
    password: prom-operator
    ```
10. Here we should check for "Node Exporter/USE method/Node" and "Node Exporter/USE method/Cluster"
    USE - Utilization, Saturation, Errors

11. Even we can check the behavior of each pod, node, and cluster

Create namespace monitoring



Add Prometheus helm chart repository and update helm chart repository

```
ubuntu@ip-10-1-1-81:~$ helm repo list
Error: no repositories to show
ubuntu@ip-10-1-1-81:~$ sudo su -
root@ip-10-1-1-81:~# helm repo list
NAME    URL
stable  https://charts.helm.sh/stable
root@ip-10-1-1-81:~# exit
logout
ubuntu@ip-10-1-1-81:~$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
ubuntu@ip-10-1-1-81:~$ helm repo list
NAME    URL
stable  https://charts.helm.sh/stable
ubuntu@ip-10-1-1-81:~$ helm repo add prometheus-grafana-community https://pr
ometheus-community.github.io/helm-charts
"prometheus-grafana-community" has been added to your repositories
ubuntu@ip-10-1-1-81:~$ helm repo list
NAME                            URL

stable                          https://charts.helm.sh/stable

prometheus-grafana-community    https://prometheus-community.github.io/helm-
charts
ubuntu@ip-10-1-1-81:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-grafana-community" chart
repository
...Successfully got an update from the "stable" chart repository
Update Complete. *Happy Helming!*
ubuntu@ip-10-1-1-81:~$
```

Install the prometheus

```
ubuntu@ip-10-1-1-81:~$ ls
aws           kube-prometheus-stack            namtrend
awscliv2.zip  kube-prometheus-stack-51.2.0.tgz  namtrend-0.1.0.tgz
jenkins       kubernetes
ubuntu@ip-10-1-1-81:~$ cd kube-prometheus-stack/
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack$ ls
CONTRIBUTING.md  Chart.lock  Chart.yaml  README.md  charts  templates  values.yaml
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack$ cd charts/
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack/charts$ ls
crds     kube-state-metrics        prometheus-windows-exporter
grafana  prometheus-node-exporter
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack/charts$ cd ..
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack$ cd templates/
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack/templates$ ls
NOTES.txt     alertmanager  extra-objects.yaml  prometheus           thanos-ruler
_helpers.tpl  exporters     grafana             prometheus-operator
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack/templates$ helm install prometheus prometheu
s-grafana-community/kube-prometheus-stack --namespace monitoring

NAME: prometheus
LAST DEPLOYED: Tue Oct  3 07:10:29 2023
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=prometheus"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances usin
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack/templates$
ubuntu@ip-10-1-1-81:~/kube-prometheus-stack/templates$
```

kubectl get all

```
ubuntu@ip-10-1-1-81:~$ kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   172.20.0.1   <none>        443/TCP   4h4m
ubuntu@ip-10-1-1-81:~$ kubectl get all -n monitoring
NAME                                                     READY   STATUS    RESTART
S   AGE
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0
    9m
pod/prometheus-grafana-865656ffbb-cfqnh                  3/3     Running   0
    9m8s
pod/prometheus-kube-prometheus-operator-7c4994c9f9-c65nx     1/1     Running   0
    9m8s
pod/prometheus-kube-state-metrics-84f546f65-l98rc            1/1     Running   0
    9m8s
pod/prometheus-prometheus-kube-prometheus-prometheus-0       2/2     Running   0
    8m59s
pod/prometheus-prometheus-node-exporter-jw8x9                1/1     Running   0
    9m9s
pod/prometheus-prometheus-node-exporter-prhbk                1/1     Running   0
    9m8s

NAME                                              TYPE        CLUSTER-IP       EXTERNA
L-IP   PORT(S)                      AGE
service/alertmanager-operated                     ClusterIP   None             <none>
       9093/TCP,9094/TCP,9094/UDP   9m
service/prometheus-grafana                        ClusterIP   172.20.188.1     <none>
       80/TCP                       9m9s
service/prometheus-kube-prometheus-alertmanager   ClusterIP   172.20.189.230   <none>
       9093/TCP,8080/TCP            9m9s
service/prometheus-kube-prometheus-operator       ClusterIP   172.20.87.131    <none>
       443/TCP                      9m9s
service/prometheus-kube-prometheus-prometheus     ClusterIP   172.20.77.144    <none>
       9090/TCP,8080/TCP            9m9s
service/prometheus-kube-state-metrics             ClusterIP   172.20.197.8     <none>
       8080/TCP                     9m9s
service/prometheus-operated                       ClusterIP   None             <none>
       9090/TCP                     9m
service/prometheus-prometheus-node-exporter       ClusterIP   172.20.92.89     <none>
       9100/TCP                     9m9s

NAME                                                DESIRED   CURRENT   READY   UP-TO
-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter   2         2         2       2
        2           kubernetes.io/os=linux   9m9s

NAME                                                READY   UP-TO-DATE   AVAILABLE
 AGE
deployment.apps/prometheus-grafana                  1/1     1            1
 9m9s
deployment.apps/prometheus-kube-prometheus-operator   1/1     1            1
 9m9s
deployment.apps/prometheus-kube-state-metrics       1/1     1            1
 9m9s

NAME                                                           DESIRED   CURRENT   R
EADY   AGE
replicaset.apps/prometheus-grafana-865656ffbb                  1         1         1
       9m9s
replicaset.apps/prometheus-kube-prometheus-operator-7c4994c9f9   1         1         1
       9m9s
replicaset.apps/prometheus-kube-state-metrics-84f546f65         1         1         1
       9m9s

NAME                                                                READY   AGE
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager   1/1     9m
statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus       1/1     9m
ubuntu@ip-10-1-1-81:~$
```

Above helm create all services as ClusterIP. To access Prometheus out side of the cluster, we should change the service type load balancer

```
kubectl edit svc prometheus-kube-prometheus-prometheus -n monitoring
```

```
    app.kubernetes.io/part-of: kube-prometheus-stack
    app.kubernetes.io/version: 51.2.0
    chart: kube-prometheus-stack-51.2.0
    heritage: Helm
    release: prometheus
    self-monitor: "true"
  name: prometheus-kube-prometheus-prometheus
  namespace: monitoring
  resourceVersion: "40756"
  uid: 117bfeb3-f949-46ac-bd64-9cdc65798d08
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: 172.20.77.144
  clusterIPs:
  - 172.20.77.144
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http-web
    nodePort: 30607
    port: 9090
    protocol: TCP
    targetPort: 9090
  - appProtocol: http
    name: reloader-web
    nodePort: 30491
    port: 8080
    protocol: TCP
    targetPort: reloader-web
  selector:
    app.kubernetes.io/name: prometheus
    operator.prometheus.io/name: prometheus-kube-prometheus-prometheus
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - hostname: a117bfeb3f94946acbd649cdc65798d0-1263591558.us-east-1.elb.amazonaws.com
```

service changed

```
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0          25m
pod/prometheus-grafana-865656ffbb-cfqnh                  3/3     Running   0          25m
pod/prometheus-kube-prometheus-operator-7c4994c9f9-c65nx 1/1     Running   0          25m
pod/prometheus-kube-state-metrics-84f546f65-l98rc        1/1     Running   0          25m
pod/prometheus-prometheus-kube-prometheus-prometheus-0   2/2     Running   0          25m
pod/prometheus-prometheus-node-exporter-jw8x9            1/1     Running   0          25m
pod/prometheus-prometheus-node-exporter-prhbk            1/1     Running   0          25m

NAME                                              TYPE           CLUSTER-IP       EXTERNAL-IP
RT(S)                      AGE
service/alertmanager-operated                     ClusterIP      None             <none>
93/TCP,9094/TCP,9094/UDP   25m
service/prometheus-grafana                        ClusterIP      172.20.188.1     <none>
/TCP                       25m
service/prometheus-kube-prometheus-alertmanager   ClusterIP      172.20.189.230   <none>
93/TCP,8080/TCP            25m
service/prometheus-kube-prometheus-operator       ClusterIP      172.20.87.131    <none>
3/TCP                      25m
service/prometheus-kube-prometheus-prometheus     LoadBalancer   172.20.77.144    a117bfeb3f94946acbd649cdc65798d0-1263591558.us-east-1.elb.amazonaws.com
90:30607/TCP,8080:30491/TCP   25m
service/prometheus-kube-state-metrics             ClusterIP      172.20.197.8     <none>
80/TCP                     25m
service/prometheus-operated                       ClusterIP      None             <none>
90/TCP                     25m
service/prometheus-prometheus-node-exporter       ClusterIP      172.20.92.89     <none>
00/TCP                     25m

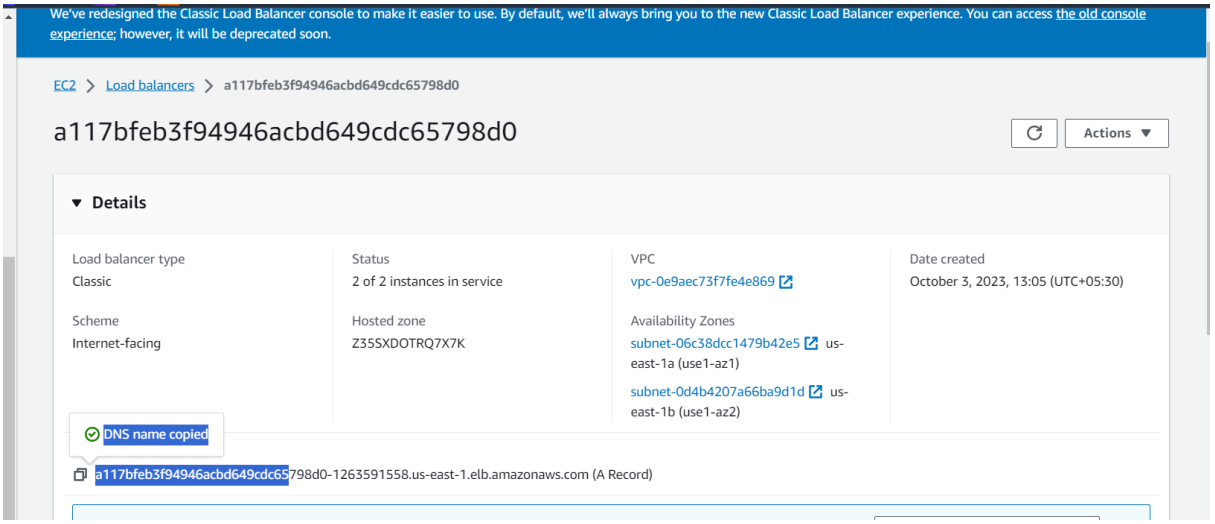
NAME                                              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter   2         2         2       2            2           kubernetes.io/os=linux   25m

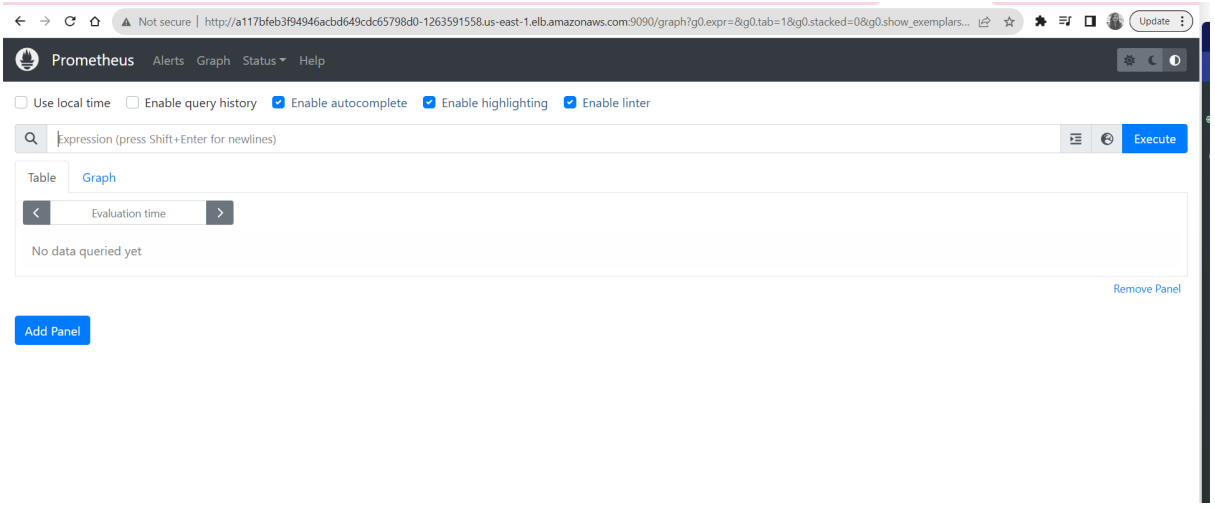
NAME                                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-grafana               1/1     1            1           25m
deployment.apps/prometheus-kube-prometheus-operator   1/1     1            1           25m
deployment.apps/prometheus-kube-state-metrics    1/1     1            1           25m

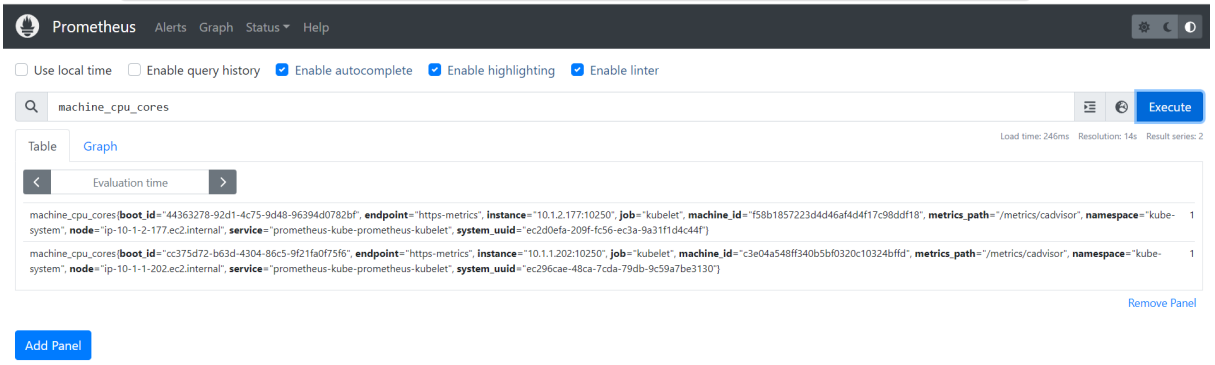
NAME                                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-grafana-865656ffbb               1         1         1       25m
replicaset.apps/prometheus-kube-prometheus-operator-7c4994c9f9   1         1         1       25m
replicaset.apps/prometheus-kube-state-metrics-84f546f65     1         1         1       25m

NAME                                                    READY   AGE
```

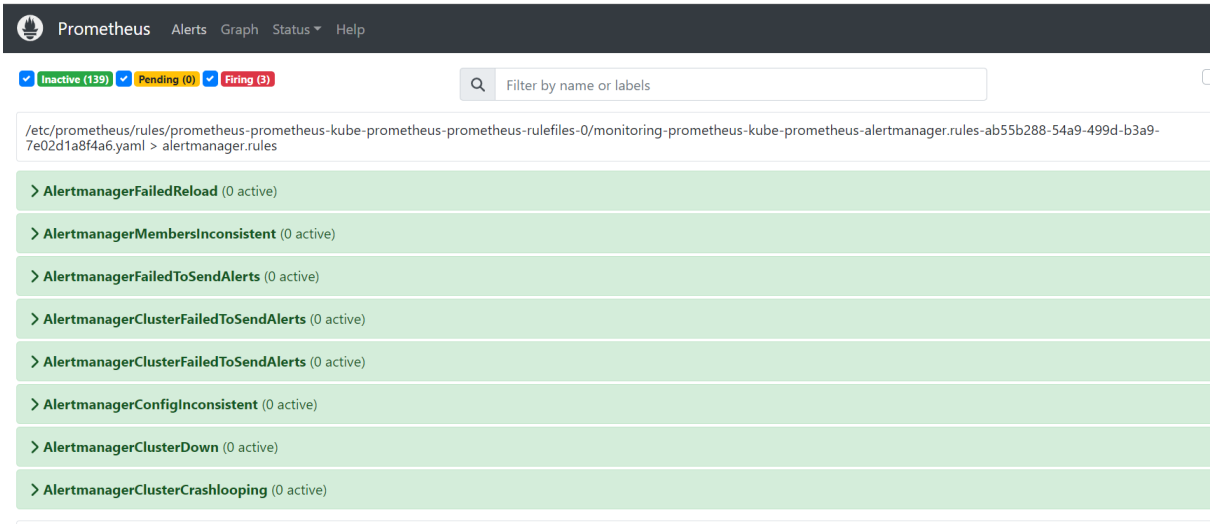
Loginto Prometheus dashboard to monitor application- https://ELB:9090





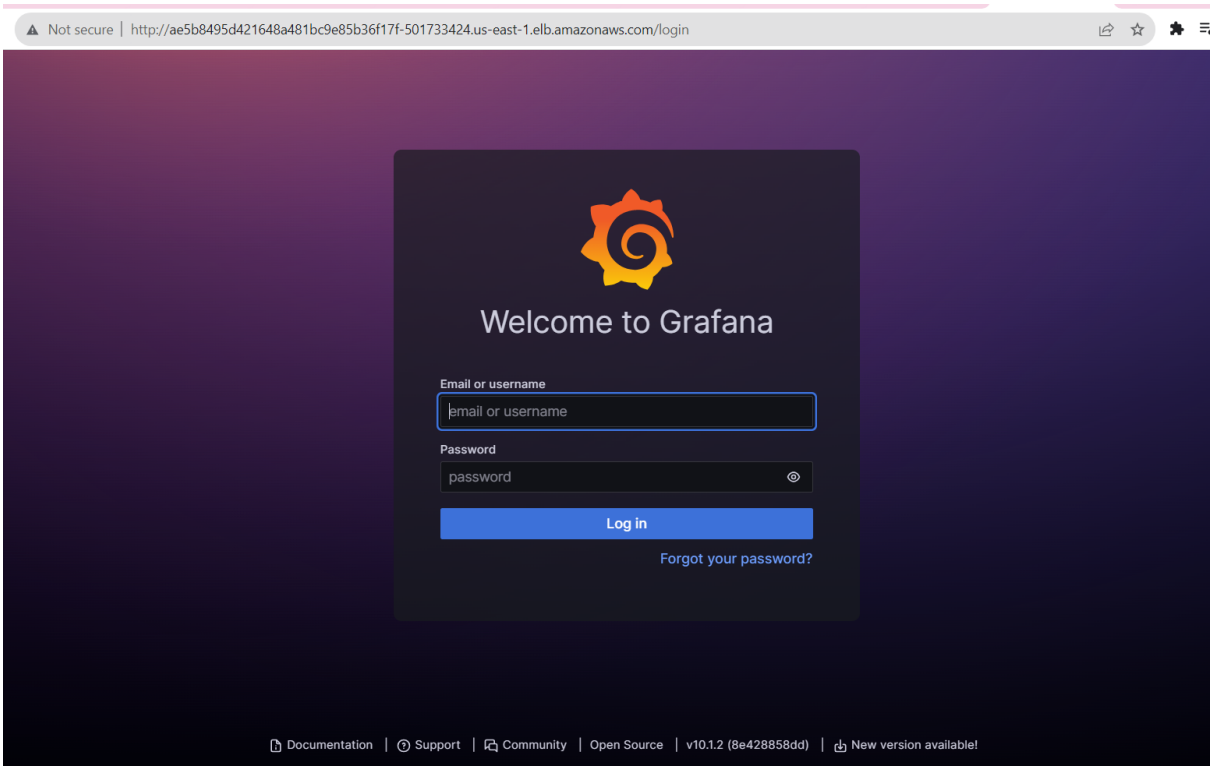


Alerts



==========

Grafana



💡 **Grafana is a multi-platform, open source analytics and interactive visualization web application. So Grafana is a open source tool which helps us to analyze and generate the interactive visualizations.**

💡 **It provides charts, graphs and alerts for the web when connected to the supported data services. It requires the data services, data services, nothing but which can help us to provide the data. Whenever it connects to the data services it generates the charts, graphs and alerts based on the data, what it gets from the other data services.**

💡 **Grafana allows us to query, visualize, alert and understand our metrics no matter where they are stored. Some supported data sources, in addition to the Prometheus are Cloudwatch, Azure Monitor, PostgreSQL, Elasticsearch and many more. Grafana requires the data sources, data sources.**
nothing but we have seen the Prometheus. I mean to say metrics so similar way Cloud, Azure Monitor, PostgreSQL, Elasticsearch. These are the data sources from these services. It takes the data and it generates the queries, visualize the data and generate the alerts as well.

💡 **We can create our own dashboards or use the existing ones provided by the Grafana. We can personalize the dashboards as per our requirement.**
So Grafana provides some of the existing dashboards. Apart from existing dashboards, we can create our own custom dashboards.

How to enable Grafana

````
We check similar graphs in the Grafana dashboard itself. for that, we should change the service type of Grafana to LoadBalancer

   kubectl edit svc prometheus-grafana
   ```
To login to Grafana account, use the below username and password
    ```sh
    username: admin
    password: prom-operator
    ```
Here we should check for "Node Exporter/USE method/Node" and "Node Exporter/USE method/Cluster"
    USE - Utilization, Saturation, Errors

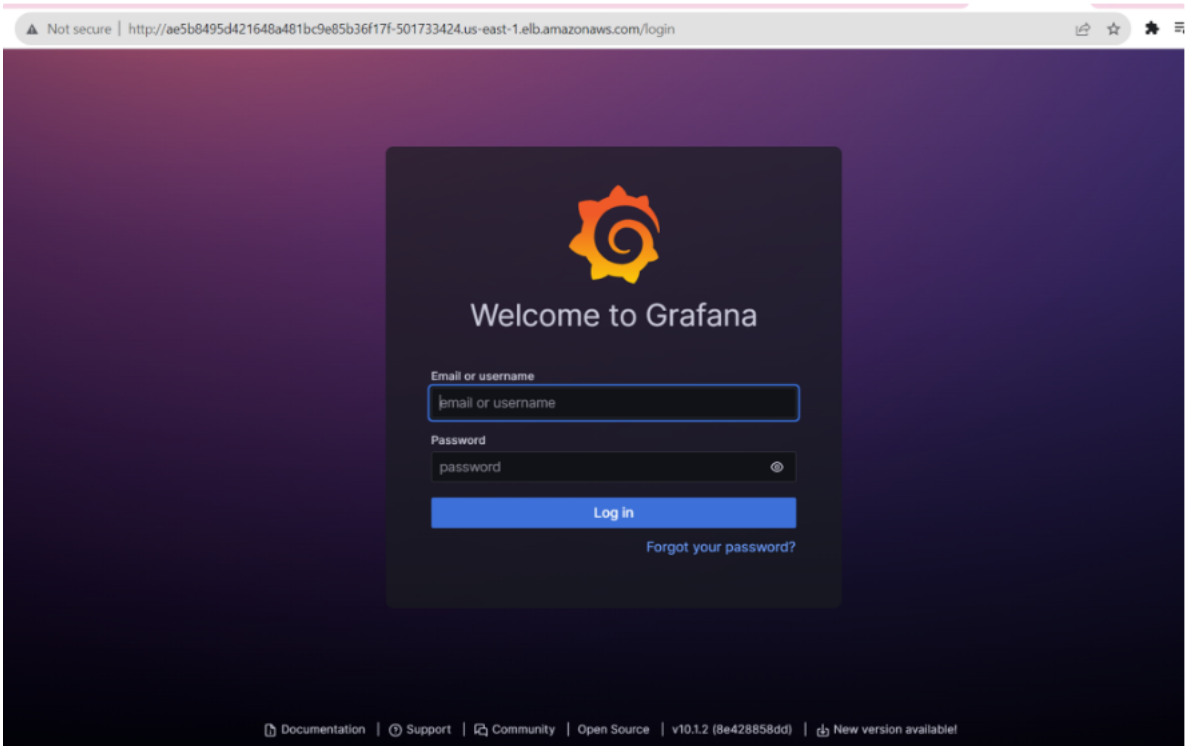
Even we can check the behavior of each pod, node, and cluster
````

kubectl edit svc prometheus-grafana
````

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  annotations:
    meta.helm.sh/release-name: prometheus
    meta.helm.sh/release-namespace: monitoring
  creationTimestamp: "2023-10-03T07:10:45Z"
  labels:
    app.kubernetes.io/instance: prometheus
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: grafana
    app.kubernetes.io/version: 10.1.2
    helm.sh/chart: grafana-6.59.5
  name: prometheus-grafana
  namespace: monitoring
  resourceVersion: "36789"
  uid: e5b8495d-4216-48a4-81bc-9e85b36f17f1
spec:
  clusterIP: 172.20.188.1
  clusterIPs:
  - 172.20.188.1
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http-web
    port: 80
    protocol: TCP
    targetPort: 3000
  selector:
    app.kubernetes.io/instance: prometheus
    app.kubernetes.io/name: grafana
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer: {}
-- INSERT --
```

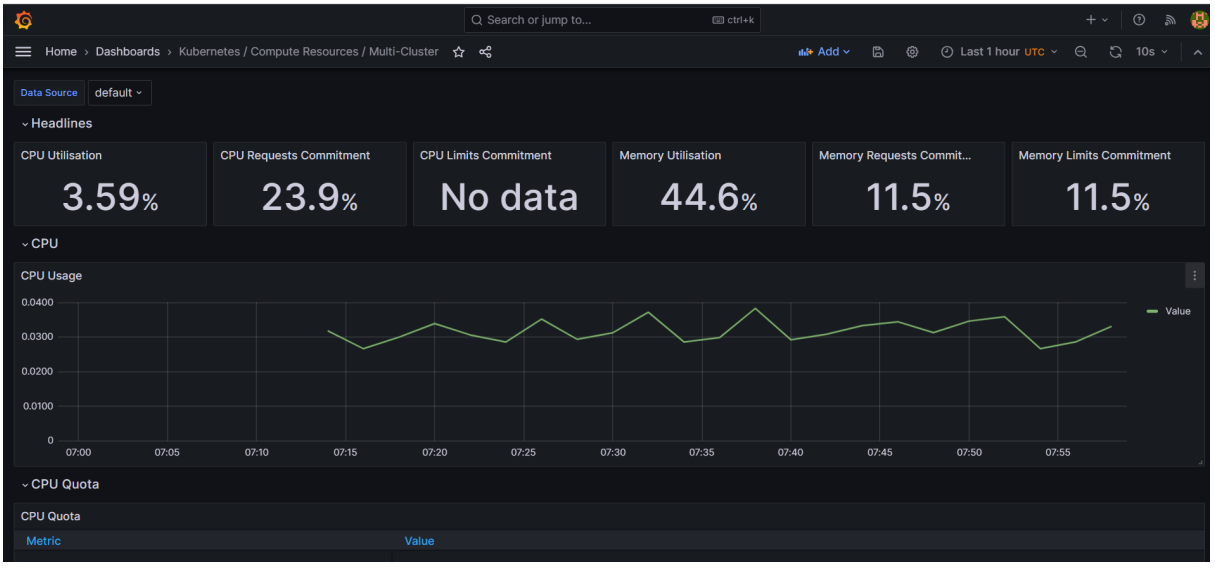
Change type to Loadbalancer

```
NAME                                      TYPE           CLUSTER-IP       EXTERNAL-IP
RT(S)                        AGE
service/alertmanager-operated             ClusterIP      None             <none>
93/TCP,9094/TCP,9094/UDP     43m
service/prometheus-grafana                LoadBalancer   172.20.188.1     ae5b8495d421648a481bc9e85b36f17f-501733424.us-east-1.elb.amazonaws.com
:30508/TCP                   43m
service/prometheus-kube-prometheus-alertmanager   ClusterIP      172.20.189.230   <none>
93/TCP,8080/TCP              43m
service/prometheus-kube-prometheus-operator       ClusterIP      172.20.87.131    <none>
3/TCP                        43m
service/prometheus-kube-prometheus-prometheus     LoadBalancer   172.20.77.144    a117bfeb3f94946acbd649cdc65798d0-1263591558.us-east-1.elb.amazonaws.com
90:30607/TCP,8080:30491/TCP  43m
service/prometheus-kube-state-metrics             ClusterIP      172.20.197.8     <none>
80/TCP                       43m
service/prometheus-operated               ClusterIP      None             <none>
90/TCP                       43m
service/prometheus-prometheus-node-exporter       ClusterIP      172.20.92.89     <none>
00/TCP                       43m
```

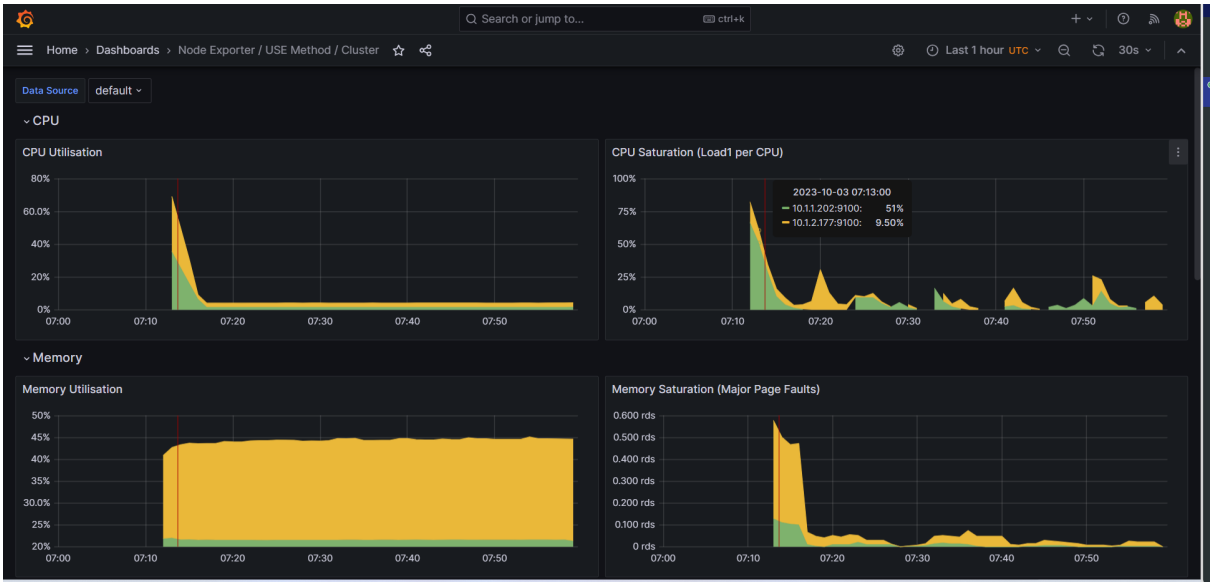
username: admin
password: prom-operator
Access on port :80



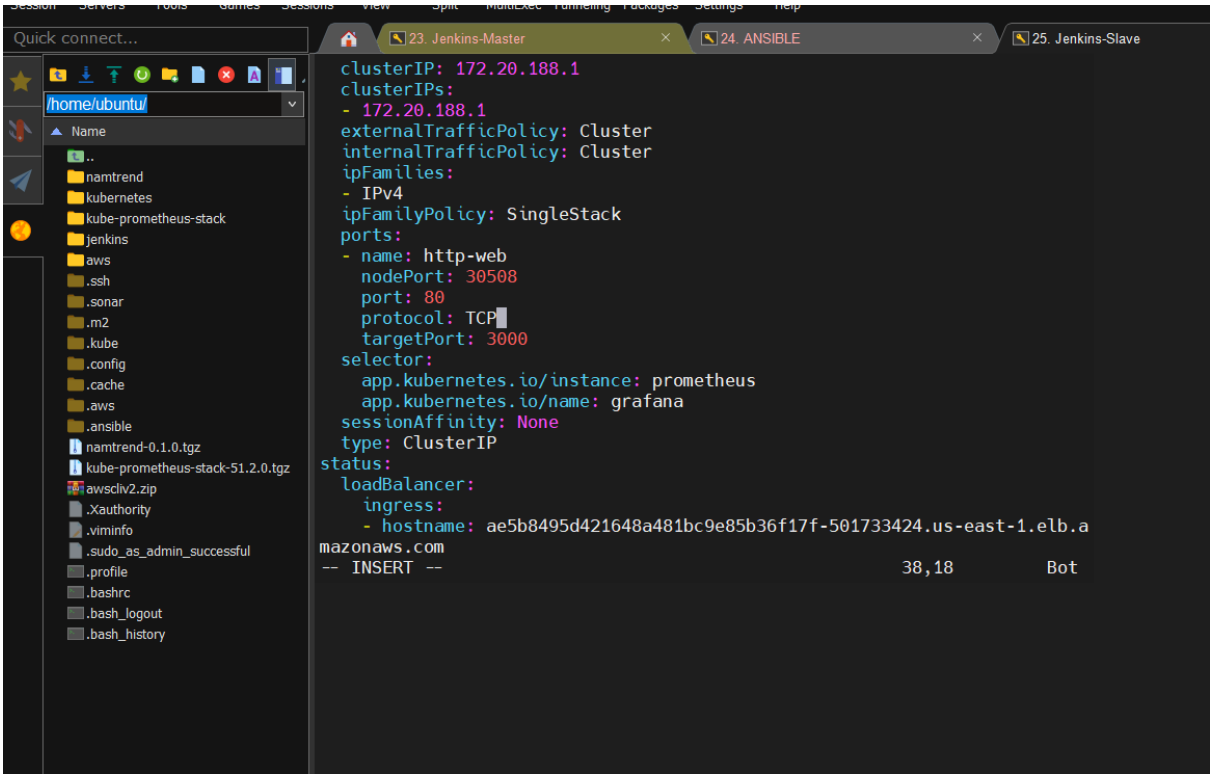
we should check for "Node Exporter/USE method/Node" and "Node Exporter/USE method/Cluster"
USE - Utilization, Saturation, Errors



Check for node_load15 executor to check cluster monitoring



To delete the Prometheus and Grafana setup. Grafana: change the LoadBalancer to ClusterIP



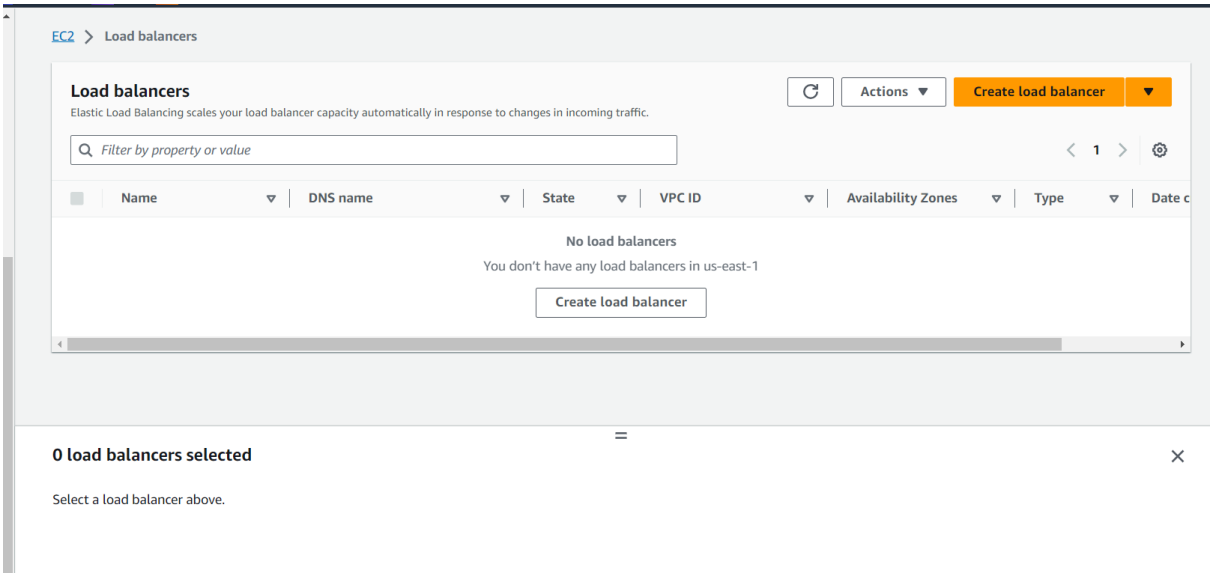
On Prometheus: Changed IP from LoadBalancer to ClusterIP

```
    app.kubernetes.io/part-of: kube-prometheus-stack
    app.kubernetes.io/version: 51.2.0
    chart: kube-prometheus-stack-51.2.0
    heritage: Helm
    release: prometheus
    self-monitor: "true"
  name: prometheus-kube-prometheus-prometheus
  namespace: monitoring
  resourceVersion: "40756"
  uid: 117bfeb3-f949-46ac-bd64-9cdc65798d08
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: 172.20.77.144
  clusterIPs:
  - 172.20.77.144
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http-web
    nodePort: 30607
    port: 9090
    protocol: TCP
    targetPort: 9090
  - appProtocol: http
    name: reloader-web
    nodePort: 30491
    port: 8080
    protocol: TCP
    targetPort: reloader-web
  selector:
    app.kubernetes.io/name: prometheus
    operator.prometheus.io/name: prometheus-kube-prometheus-prometheus
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer:
    ingress:
    - hostname: a117bfeb3f94946acbd649cdc65798d0-1263591558.us-east-1.elb.amazonaws.com
-- INSERT --
```

Type has changed to ClusterIP

```
ubuntu@ip-10-1-1-81:~$ kubectl get all -n monitoring
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0          52m
pod/prometheus-grafana-865656ffbb-cfqnh                  3/3     Running   0          52m
pod/prometheus-kube-prometheus-operator-7c4994c9f9-c65nx     1/1     Running   0          52m
pod/prometheus-kube-state-metrics-84f546f65-l98rc            1/1     Running   0          52m
pod/prometheus-prometheus-kube-prometheus-prometheus-0       2/2     Running   0          52m
pod/prometheus-prometheus-node-exporter-jw8x9                1/1     Running   0          52m
pod/prometheus-prometheus-node-exporter-prhbk                1/1     Running   0          52m

NAME                                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                     ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP   52m
service/prometheus-grafana                        ClusterIP   172.20.188.1     <none>        80/TCP                       52m
service/prometheus-kube-prometheus-alertmanager   ClusterIP   172.20.189.230   <none>        9093/TCP,8080/TCP            52m
service/prometheus-kube-prometheus-operator       ClusterIP   172.20.87.131    <none>        443/TCP                      52m
service/prometheus-kube-prometheus-prometheus     ClusterIP   172.20.77.144    <none>        9090/TCP,8080/TCP            52m
service/prometheus-kube-state-metrics             ClusterIP   172.20.197.8     <none>        8080/TCP                     52m
service/prometheus-operated                       ClusterIP   None             <none>        9090/TCP                     52m
service/prometheus-prometheus-node-exporter       ClusterIP   172.20.92.89     <none>        9100/TCP                     52m

NAME                                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter   2         2         2       2            2           kubernetes.io/os=linux   52m

NAME                                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-grafana                 1/1     1            1           52m
deployment.apps/prometheus-kube-prometheus-operator   1/1     1            1           52m
deployment.apps/prometheus-kube-state-metrics      1/1     1            1           52m

NAME                                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-grafana-865656ffbb                 1         1         1       52m
replicaset.apps/prometheus-kube-prometheus-operator-7c4994c9f9   1         1         1       52m
replicaset.apps/prometheus-kube-state-metrics-84f546f65       1         1         1       52m

NAME                                                                READY   AGE
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager   1/1     52m
statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus       1/1     52m
ubuntu@ip-10-1-1-81:~$
```

Load Balancers have deleted from AWS



😇

🙌 Congrats on finishing the course!

Find more courses