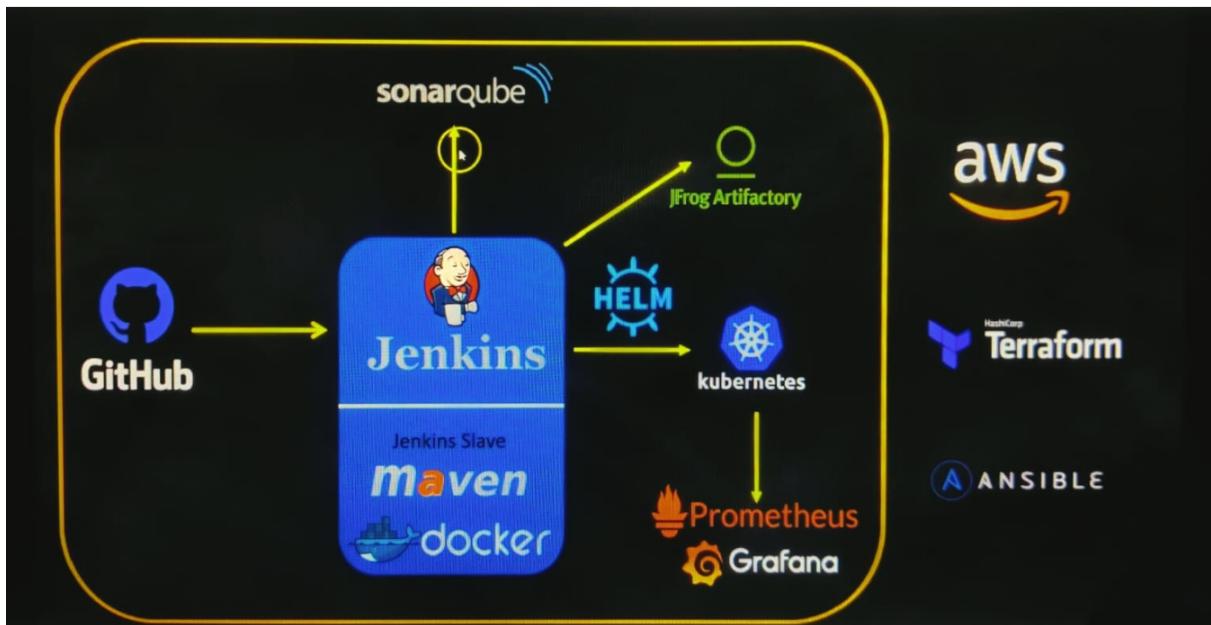


CICD Project Workshop



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

```

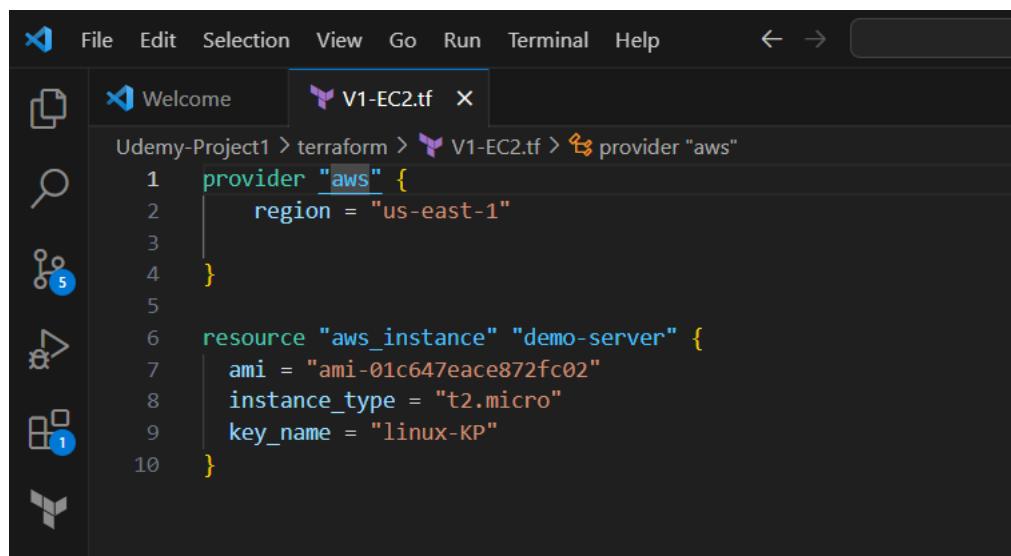
or you can run the below command in powershell or the command prompt

Terraform-code

1. Create IAM user
2. Login to aws cli

```
namratak@NAMRATAKFTVTC3 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



```
File Edit Selection View Go Run Terminal Help ← →
Welcome V1-EC2.tf X
Udemy-Project1 > terraform > V1-EC2.tf > provider "aws"
1 provider "aws" {
2   region = "us-east-1"
3 }
4
5
6 resource "aws_instance" "demo-server" {
7   ami = "ami-01c647eace872fc02"
8   instance_type = "t2.micro"
9   key_name = "linux-KP"
10 }
```

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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/Udemy-Project1/terraform
$ terraform validate
Success! The configuration is valid.

namratak@NAMRATAKFTVTC3 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

Instances (1/1) [Info](#)

| <input checked="" type="checkbox"/> Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Z. |
|---|---------------------|----------------------|---------------|---------------------------|--------------|-----------------|
| <input checked="" type="checkbox"/> demo-server | i-0f01ac0a964ab9b3c | Running | t2.micro | Initializing | No alarms | us-east-1d |

Instance: i-0f01ac0a964ab9b3c

[Details](#) [Security](#) [Networking](#) [Storage](#) [Status checks](#) [Monitoring](#) [Tags](#)

▼ Instance summary [Info](#)

| | | |
|--|---|---|
| Instance ID
i-0f01ac0a964ab9b3c | Public IPv4 address
35.170.50.106 open address | Private IPv4 addresses
172.31.34.27 |
| IPv6 address
- | Instance state
Running | Public IPv4 DNS
ec2-35-170-50-106.compute-1.amazonaws.com open address |

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-subnet-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-subnet-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-rt-a-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-rt-a-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@NAMRATAKFTVTC3 MINGW64 ~/Linux/workshop-devops2/Terraform (main)
$ ls
V4-EC2-With_VPC_for_each.tf  terraform.tfstate  terraform.tfstate.backup

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ terraform validate
Success! The configuration is valid.
```

```
terraform plan
```

```
namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC3 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

| Instances (3) Info | | | | | | | | |
|---|---------------------|----------------|---------------|--------------|--------------|-------------------|--|--|
| Connect Instance state ▾ Actions ▾ Launch instances ▾ | | | | | | | | |
| <input type="text" value="Find instance by attribute or tag (case-sensitive)"/> | | | | | | | | |
| Instance state = running X Clear filters | | | | | | | | |
| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | | |
| jenkins-master | i-065c318d6eb6be3ff | Running | t2.micro | Initializing | 0 in alarm | us-east-1a | | |
| jenkins-slave | i-03676f16009e1aa5 | Running | t2.micro | Initializing | 0 in alarm | us-east-1a | | |
| ansible | i-0426a6acf8bf7123 | Running | t2.micro | Initializing | 0 in alarm | us-east-1a | | |

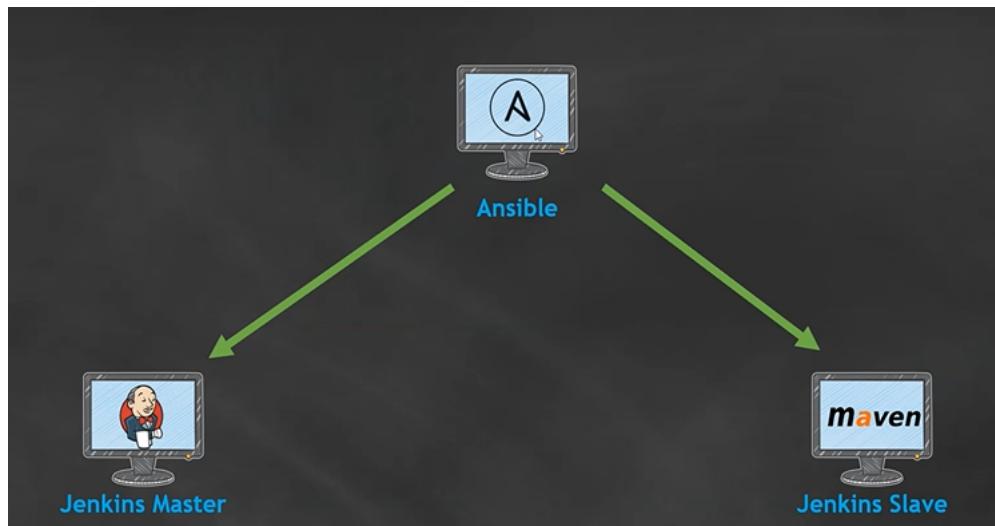
All code has committed into GitHub repo

The screenshot shows a GitHub repository named 'Workshop-devops2' with a single branch 'main'. The commit history is as follows:

- Commit 1: 'Created jenkins-master,slave and ansible servers through terraform-code' (by Namg04)
- Commit 2: 'adding V1-EC2.tf' (by Namg04)
- Commit 3: 'commiting V2-EC2.tf' (by Namg04)
- Commit 4: 'adding V3-EC2.tf' (by Namg04)
- Commit 5: 'Created jenkins-master,slave and ansible servers through terraform-code' (by Namg04)

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 ansible 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
```

A screenshot of a terminal window titled '2. Ansible'. The terminal shows the command-line process of installing Ansible on an Ubuntu 22.04 system. The commands entered are:

```
root@ip-10-1-1-177:/home/ubuntu#
root@ip-10-1-1-177:/home/ubuntu# sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [11
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packag
MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-
```

At the bottom of the terminal, the command 'ansible -version' is shown.

```
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

```
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-10-1-1-177:/home/ubuntu#
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/collections
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
root@ip-10-1-1-177:/home/ubuntu#
root@ip-10-1-1-177:/home/ubuntu#
root@ip-10-1-1-177:/home/ubuntu#
root@ip-10-1-1-177:/home/ubuntu#
root@ip-10-1-1-177:/home/ubuntu#
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# 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#
```

Remote monitoring

Follow terminal folder

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

[jenins-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 -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

The screenshot shows a terminal window with two tabs: '6. ANSIBLE' and '7. Jenkins-Master'. The '6. ANSIBLE' tab contains the Ansible playbook code:

```
- hosts: jenkins-master
  become: true
  tasks:
    - name: add jenkins key
      apt_key:
        url: https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
        state: present

    - name: add jenkins repo
      apt_repository:
        repo: 'deb https://pkg.jenkins.io/debian-stable binary/'
        state: present

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

    - name: install jenkins
      apt:
        name: jenkins
        state: present

    - name: start jenkins service
      service:
        name: jenkins
        state: started

    - name: enable jenkins to start at boot time
      service:
        name: jenkins
        enabled: yes
```

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

```
root@ip-10-1-1-177:/opt# ansible-playbook -i /opt/hosts jenkins-mster-setup.yml --check
[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] ****
fatal: [10.1.1.153]: FAILED! => {"changed": false, "msg": "No package matching 'jenkins'
is available"}
```

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]

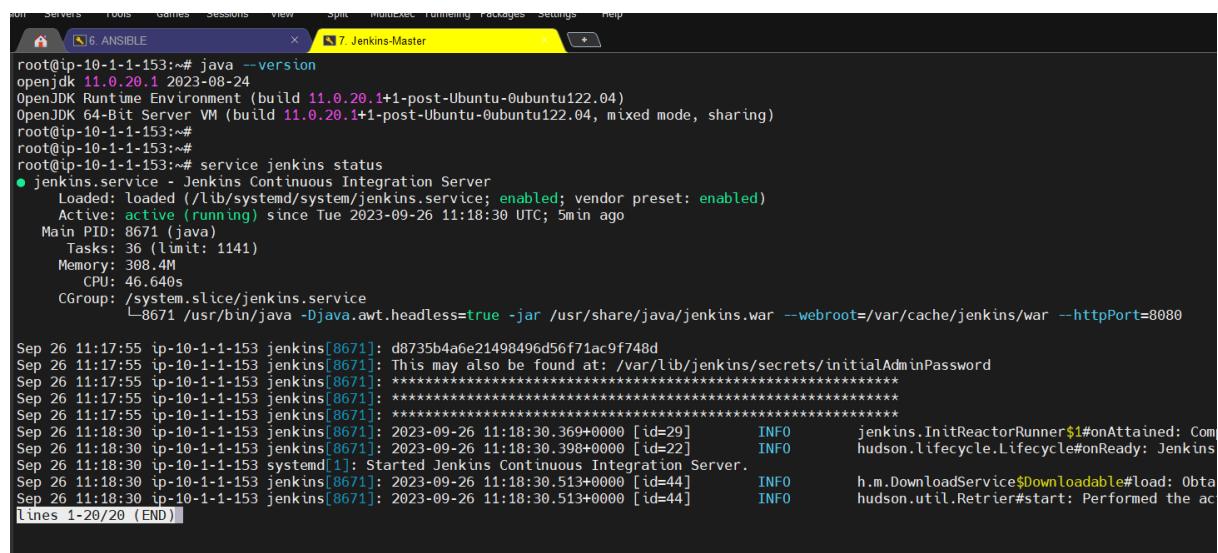
TASK [start jenkins service] ****
ok: [10.1.1.153]

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



The screenshot shows a terminal window with two tabs: '6. ANSIBLE' and '7. Jenkins-Master'. The '7. Jenkins-Master' tab is active and displays the following output:

```

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]: 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" "demo-sg"
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@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ ls
V4-EC2-With_VPC_for_each.tf  terraform.tfstate  terraform.tfstate.backup

namratak@NAMRATAKFTVTC3 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-024dfffc7c21f2a8c
3]
aws_subnet.Nam-public-subnet-02: Refreshing state... [id=subnet-0c0b0c9403ebbc5d
8]
aws_security_group.demo-sg: Refreshing state... [id=sg-031d62b69f2db6dd71]

```

```

}
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

The screenshot shows a table titled "Instance: i-098613d9fa83fffe7 (jenkins-master)". The table has columns: Name, Security group rule ID, Port range, Protocol, Source, and Security groups. There are two rows. The first row has a green highlighted "Port range" cell containing "8080". The second row has a "Port range" cell containing "22". The "Security groups" column for both rows lists "demo-sg". Below the table, there is a section titled "Outbound rules" with a table showing two entries, both associated with "demo-sg".

Access Jenkins-master

The screenshot shows the "Getting Started" screen of Jenkins. The title is "Unlock Jenkins". It instructs the user to copy the password from "/var/lib/jenkins/secrets/initialAdminPassword". A red box highlights this path. Below it, a placeholder text says "Please copy the password from either location and paste it below." An input field labeled "Administrator password" contains a redacted password. A blue "Continue" button is at the bottom right.

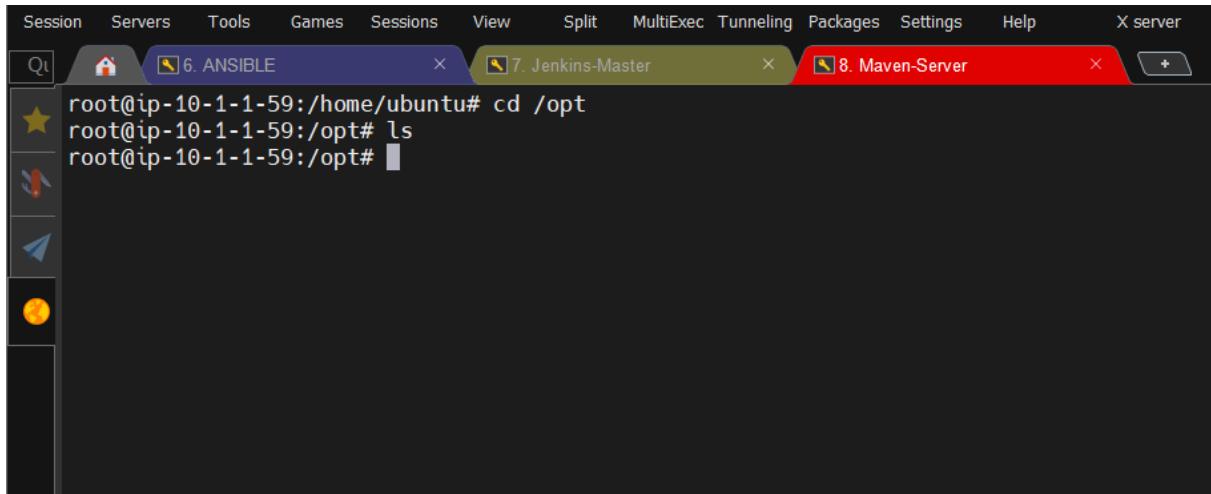
Ansible playbook to configure maven on Jenkins-slave server

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.

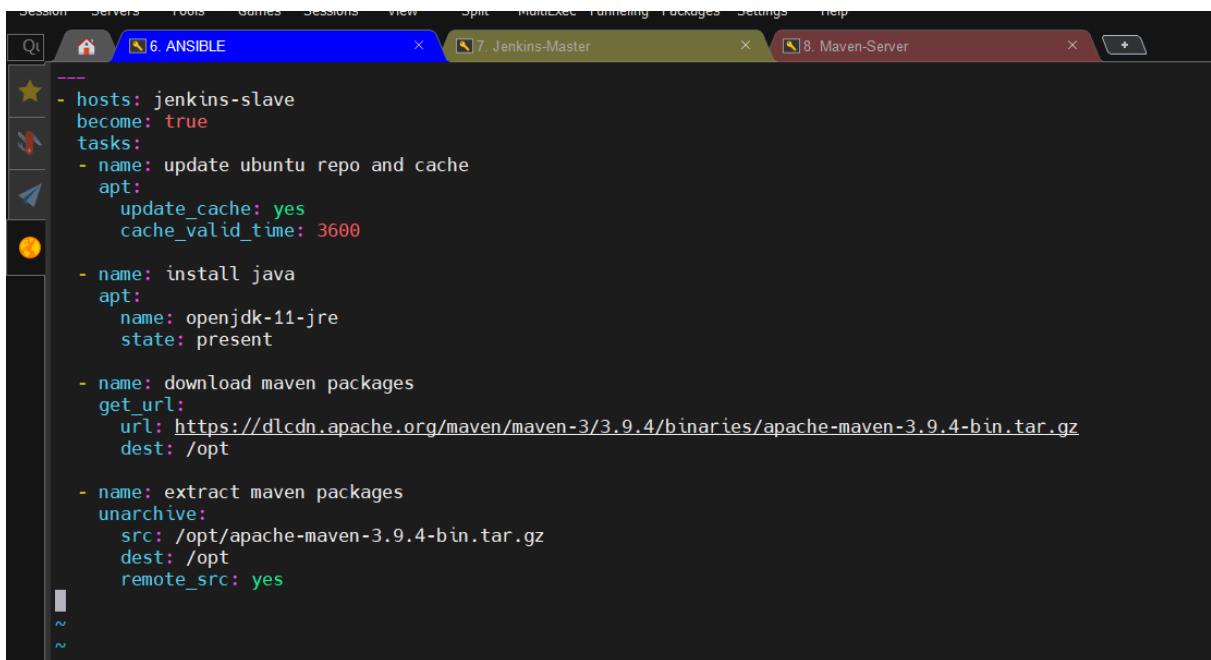
- **Update the System**
- **Install Java**
- **Download Maven Packages Extract it**
- **Add path to bash_profile**

Login to Jenkins-slave server



```
root@ip-10-1-1-59:/home/ubuntu# cd /opt
root@ip-10-1-1-59:/opt# ls
root@ip-10-1-1-59:/opt#
```

vi jenkins-slave-setup.yml



```
---
- 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
```

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

```

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Qu 6. ANSIBLE 7. Jenkins-Master 8. Maven-Server +
★ root@ip-10-1-1-59:/opt# ls
apache-maven-3.9.4 apache-maven-3.9.4-bin.tar.gz
★ root@ip-10-1-1-59:/opt# cd apache-maven-3.9.4
root@ip-10-1-1-59:/opt/apache-maven-3.9.4# ls
LICENSE NOTICE README.txt bin boot conf lib
root@ip-10-1-1-59:/opt/apache-maven-3.9.4# cd bin
root@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin# ls
m2.conf mvn mvn.cmd mvnDebug mvnDebug.cmd mvnjp
root@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin# ./mvn --version
Apache Maven 3.9.4 (dfbb324ad4a7c8fb0bf182e6d91b0ae20e3d2dd9)
Maven home: /opt/apache-maven-3.9.4
Java version: 11.0.20.1, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.19.0-1025-aws", arch: "amd64", family: "unix"
root@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin# pwd
/opt/apache-maven-3.9.4/bin
root@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin# 

```

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

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: maven-server-credentials

Description: maven server credentials

Description: maven server credentials

Username: ubuntu

Private Key:

Key:

```
tRVgnIjvt6bEHvvnwK0RufZe1XwJtvBF6Ek8PTp9XfkCoHeahW61B8KnbwUVDSx
t+RPyIdts9i1a1gT8dQrs8K7PT7g+OxTuaqPSTpN4G9Pd7ydovCHQ==
-----END RSA PRIVATE KEY-----
```

Passphrase:

Create

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

| ID | Name | Kind | Description |
|--------------------------|-----------------------------------|-------------------------------|--------------------------|
| maven-server-credentials | ubuntu (maven server credentials) | SSH Username with private key | maven server credentials |

Icon: S M L

Adding Node to Jenkins-master

Jenkins

Dashboard > Manage Jenkins > Nodes >

Nodes

Nodes Clouds Node Monitoring Build Queue Build Executor Status

| S | Name | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|--------------------------------|---------------|------------------|-----------------|-----------------|-----------------|---------------|
| | Built-In Node
Data obtained | Linux (amd64) | In sync | 4.64 GB | 1 0 B | 4.64 GB | 0ms |

No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle

Create New Node

Dashboard > Manage Jenkins > Nodes > New node

New node

Node name

Type Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

Add

Jenkins

Dashboard > Manage Jenkins > Nodes >

Name ?
maven-slave

Description ?
build java-based applications on this slave

Plain text [Preview](#)

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 ?
10.1.1.59

Credentials ?
ubuntu (maven server credentials)

Add ▾

Not secure | http://44.213.127.95:8080/manage/computer/createItem

Dashboard > Manage Jenkins > Nodes >

ubuntu (maven server credentials)

Add ▾

Host Key Verification Strategy ?
Non verifying Verification Strategy

Advanced ▾

Availability ?
Keep this agent online as much as possible

Node Properties

- Disable deferred wipeout on this node ?
- Environment variables
- Tool Locations

Save

DECT AND Jenkins

Node created

Nodes

| S | Name ↓ | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|---------------|---------------|------------------|-----------------|-----------------|-----------------|---------------|
| 💻 | Built-In Node | Linux (amd64) | In sync | 4.64 GB | 0 B | 4.64 GB | 0ms 🕒 |
| 💻 | maven-slave | | N/A | N/A | N/A | N/A | N/A 🕒 |
| | Data obtained | | 10 ms | 10 ms | 7 ms | 56 min | 56 min |

Build Queue ▾
No builds in the queue.

Build Executor Status ▾
Built-In Node
1 Idle

Logs : Agent successfully connected and online

Dashboard > Nodes > maven-slave > Log

```
[09/26/23 13:13:53] [SSH] Starting sftp client.
[09/26/23 13:13:53] [SSH] Remote file system root /home/ubuntu/jenkins does not exist. Will try to create it...
[09/26/23 13:13:53] [SSH] Copying latest remoting.jar...
[09/26/23 13:13:53] [SSH] Copied 1,371,113 bytes.
Expanded the channel window size to 4MB
[09/26/23 13:13:53] [SSH] Starting agent process: cd "/home/ubuntu/jenkins" && java -jar remoting.jar -workDir /home/ubuntu/jenkins -jar-cache /home/ubuntu/jenkins/remoting/jarCache
Sep 26, 2023 1:13:54 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Sep 26, 2023 1:13:54 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/jenkins/remoting
<===[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 3131.vf2b_b_798b_ce99
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by jenkins.slaves.StandardOutputSwapper$ChannelSwapper to constructor java.io.FileDescriptor(int)
WARNING: Please consider reporting this to the maintainers of jenkins.slaves.StandardOutputSwapper$ChannelSwapper
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Evacuated stdout
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

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 **Create Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Dashboard > test-job > Configuration

Configure

General

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.

Dashboard > test-job > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

Build Steps

Execute shell

Command

See [the list of available environment variables](#)

```
echo "Hello, I am slave system" >> /home/ubuntu/maven.txt
```

Advanced ▾

Add build step ▾

Build Now

Dashboard > test-job > #1 > Console Output

Console Output

- Status
- Changes
- Console Output**
- View as plain text
- Edit Build Information
- Delete build '#1'

```
Started by user admin
Running as SYSTEM
Building remotely on maven-slave (maven) in workspace /home/ubuntu/jenkins/workspace/test-job
[test-job] $ /bin/sh -xe /tmp/jenkins15012723124413513.sh
+ echo Hello, I am slave system
Finished: SUCCESS
```

maven.txt file created

```
ubuntu@ip-10-1-1-59:~$ ls
jenkins
ubuntu@ip-10-1-1-59:~$ cd jenkins/
ubuntu@ip-10-1-1-59:~/jenkins$ ls
remoting  remoting.jar
ubuntu@ip-10-1-1-59:~/jenkins$ cd ..
ubuntu@ip-10-1-1-59:~$ ls
jenkins  maven.txt
ubuntu@ip-10-1-1-59:~$ cat maven.txt
Hello, I am slave system
ubuntu@ip-10-1-1-59:~$
```

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

Dashboard > Nam-trend-job > Configuration

Configure

Pipeline

Definition

Script

```
1 * pipeline {
2     agent {label 'maven'}
3     stages {
4         stage('Clone') {
5             steps {
6                 git branch: 'main', url: 'https://github.com/Namg04/tweet-trend-new.git'
7             }
8         }
9     }
10 }
11
```

Use Groovy Sandbox

Save Apply

Build now

Dashboard > Nam-trend-job > #7

Console Output

- Status
- </> Changes
- Console Output**
- View as plain text
- Edit Build Information
- Delete build '#7'
- Git Build Data
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces
- ← Previous Build

```

Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on maven-slave in /home/ubuntu/jenkins/workspace/Nam-trend-job
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Namg04/tweet-trend-new.git
> git init /home/ubuntu/jenkins/workspace/Nam-trend-job # timeout=10
Fetching upstream changes from https://github.com/Namg04/tweet-trend-new.git
> git --version # timeout=10
> git -v version # 'git' version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Namg04/tweet-trend-new.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/Namg04/tweet-trend-new.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
Checking out Revision 2f1d8b3ecbcc2828eab1b1747df98caaf21407c9 (refs/remotes/origin/main)
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10

```

This is how to clone the job through pipeline job

```

ubuntu@ip-10-1-1-59:~$ ls
jenkins
ubuntu@ip-10-1-1-59:~$ cd jenkins/
ubuntu@ip-10-1-1-59:~/jenkins$ ls
remoting remoting.jar
ubuntu@ip-10-1-1-59:~/jenkins$ cd ..
ubuntu@ip-10-1-1-59:~$ ls
jenkins maven.txt
ubuntu@ip-10-1-1-59:~$ cat maven.txt
Hello, I am slave system
ubuntu@ip-10-1-1-59:~$ ls
jenkins maven.txt
ubuntu@ip-10-1-1-59:~$ cd jenkins/
ubuntu@ip-10-1-1-59:~/jenkins$ ls
remoting remoting.jar workspace
ubuntu@ip-10-1-1-59:~/jenkins$ cd workspace/
ubuntu@ip-10-1-1-59:~/jenkins/workspace$ ls
Nam-trend-job test-job
ubuntu@ip-10-1-1-59:~/jenkins/workspace$ cd Nam-trend-job/
ubuntu@ip-10-1-1-59:~/jenkins/workspace/Nam-trend-job$ ls
README.md pom.xml src
ubuntu@ip-10-1-1-59:~/jenkins/workspace/Nam-trend-job$ 

```

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

This branch is up to date with ravdy/tweet-trend-new:main.

ValaxyTech initial commit

2f1d8b3 on Jul 5 1 commit

| File | Description | Time |
|------------|----------------|--------------|
| src | initial commit | 2 months ago |
| .gitignore | initial commit | 2 months ago |
| README.md | initial commit | 2 months ago |
| pom.xml | initial commit | 2 months ago |

README.md

Cloned repo on local and created a file 'Jenkinsfile'

```
MINGW64:/c/Users/namratak/Linux/tweet-trend-new
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux
$ git clone https://github.com/Namg04/tweet-trend-new.git
Cloning into 'tweet-trend-new'...
remote: Enumerating objects: 24, done.
remote: Total 24 (delta 0), reused 0 (delta 0), pack-reused 24
Receiving objects: 100% (24/24), 4.61 KiB | 4.61 MiB/s, done.

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux
$ ls
Documents/      Udemy-Project1/      python_task/
'MEGA Downloads'/ Valaxy-DevOps/    terraform_accessKeys.csv
TWS/            Workshop-devops2/   tweet-trend-new/
Terraform/      pem-key/         

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux
$ cd tweet-trend-new/
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ ls
README.md  pom.xml  src/

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ touch Jenkinsfile
```

Write a Jenkins file with a build stage

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

```

Maven-Server
Terminal Sessions View Xserver Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
[6. ANSIBLE] [7. Jenkins-Master] [8. Maven-Server]

ubuntu@ip-10-1-1-59:~$ ls
★ jenkins maven.txt
ubuntu@ip-10-1-1-59:~$ cd /opt
ubuntu@ip-10-1-1-59:~$ ls
apache-maven-3.9.4 apache-maven-3.9.4-bin.tar.gz
ubuntu@ip-10-1-1-59:~$ cd apache-maven-3.9.4/
ubuntu@ip-10-1-1-59:/opt/apache-maven-3.9.4$ ls
LICENSE NOTICE README.txt bin boot conf lib
ubuntu@ip-10-1-1-59:/opt/apache-maven-3.9.4$ cd bin/
ubuntu@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin$ ls
m2.conf mvn mvn.cmd mvnDebug mvnDebug.cmd mvnjp
ubuntu@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin$ pwd
/opt/apache-maven-3.9.4/bin
ubuntu@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin$ /opt/apache-maven-3.9.4/bin/mvn clean
[INFO] Scanning for projects ...
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 0.187 s
[INFO] Finished at: 2023-09-27T02:23:12Z
[INFO]
[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (/opt/apache-maven-3.9.4/bin). Please verify you inv
ory. → [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MissingProjectException
ubuntu@ip-10-1-1-59:/opt/apache-maven-3.9.4/bin$ 

```

b)Give maven path in Jenkins file

```

File Edit Selection View Go Run Terminal Help ← → Workshop-devops2
EXPLORER Jenkinsfile
WORKSHOP-DEVOPS
Ansible hosts jenkins-master-setu...
jenkins-slave-s... U
Terraform .gitignore README.md
C: > Users > namratak > Linux > tweet-trend-new > Jenkinsfile
1 pipeline {
2     agent {label 'maven'}
3
4
5 environment [
6     PATH = "/opt/apache-maven-3.9.4/bin:$PATH"
7 ]
8
9 stages {
10     stage("build"){
11         steps {
12             sh 'mvn clean deploy'
13         }
14     }
15 }
16

```

c)commit the Jenkins file into Git repo

```

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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

tweet-trend-new Public

forked from [ravdy/tweet-trend-new](#)

main 1 branch 0 tags

This branch is 2 commits ahead of [ravdy:main](#).

[Go to file](#) [Add file](#) [Code](#)

| Commit History | | |
|---------------------------|---|--------------------------------------|
| | Namg04 updated jenkinsfile with build stage | c7c62ae 14 minutes ago |
| | src | initial commit |
| | .gitignore | initial commit |
| | Jenkinsfile | updated jenkinsfile with build stage |
| | README.md | initial commit |
| | pom.xml | initial commit |
| README.md | | |

Now build the job in Jenkins

Dashboard > Nam-trend-job >

Pipeline Nam-trend-job

- Status
- </> Changes
- ▷ Build Now
- ⚙️ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- ✍️ Rename
- ❓ Pipeline Syntax

Build History trend ▾

Filter builds...

| | | | |
|----|-----------------|----------|---|
| #9 | Sep 27
07:56 | 1 commit | Average stage times:
(Average full run time: ~36s) |
| | | | Declarative:
Checkout SCM |
| | 381ms | 35s | build |
| | 381ms | 35s | |

Stage View

Permalinks

- Last build (#9), 1 min 1 sec ago
- Last stable build (#9), 1 min 1 sec ago
- Last successful build (#9), 1 min 1 sec ago
- Last failed build (#5), 12 hr ago
- Last unsuccessful build (#5), 12 hr ago
- Last completed build (#9), 1 min 1 sec ago

How to add GitHub credentials to Jenkins.

Create Personal Token in Gitgub

Settings / Developer Settings

Type [] to search

- GitHub Apps
- OAuth Apps
- Personal access tokens
- Fine-grained tokens Beta
- Tokens (classic)

Personal access tokens (classic)

Generate new token ▾ Revoke all

Tokens you have generated that can be used to access the GitHub API.

| | | |
|--|------------|--------|
| workshop-devops2-token — admin:enterprise, admin:pgp_key, admin:org, admin:hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages | Never used | Delete |
| Expires on Fri, Oct 27 2023 | | |

Add GitHub credentials in Jenkins

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: Namg04

Treat username as secret

Password: *****

ID: Github_credentials

Description:

Create

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

| ID | Name | Kind | Description |
|--------------------------|-----------------------------------|-------------------------------|--------------------------|
| maven-server-credentials | ubuntu (maven server credentials) | SSH Username with private key | maven server credentials |
| github_credentials | Namg04/***** | Username with password | |

Add credentials in Nam-trend-job

Dashboard > Nam-trend-job > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Repositories

Repository URL: https://github.com/Namg04/tweet-trend-new.git

Credentials: Namg04/*****

Branches to build

Branch Specifier (blank for 'any'): */main

Save **Apply**

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**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK **Cancel** **Initialization Folder**

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 ▾

Dashboard > Nam-trend-multibranch > Configuration

Configuration

Build Configuration

Mode: by Jenkinsfile

Script Path: Jenkinsfile

Scan Multibranch Pipeline Triggers

Periodically if not otherwise run

Orphaned Item Strategy

Abort builds

Discard old items

Save **Apply**

Dashboard > Nam-trend-multibranch >

Nam-trend-multibranch

Status: **Green** | Configure | Scan Multibranch Pipeline Now | Scan Multibranch Pipeline Log | Multibranch Pipeline Events | Delete Multibranch Pipeline | People | Build History | Project Relationship | Disable Multibranch Pipeline

| S | W | Name | Last Success | Last Failure | Last Duration |
|---|---|------|----------------|--------------|---------------|
| | | main | 2 min 0 sec #1 | N/A | 28 sec |

Icon: S M L | Icon legend: Atom feed for all | Atom feed for failures | Atom feed for just latest builds

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.

Jenkins

Dashboard > Nam-trend-multibranch >

Nam-trend-multibranch

Status: **Green** | Configure | Scan Multibranch Pipeline Now | Scan Multibranch Pipeline Log | Multibranch Pipeline Events | Delete Multibranch Pipeline | People | Build History | Project Relationship | Check File Fingerprint | Disable Multibranch Pipeline

| S | W | Name | Last Success | Last Failure | Last Duration |
|---|---|------|----------------|--------------|---------------|
| | | dev | N/A | N/A | N/A |
| | | main | 4 min 0 sec #1 | N/A | 28 sec |

Icon: S M L | Icon legend: Atom feed for all | Atom feed for failures | Atom feed for just latest builds

Commit code into git

```
namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ git checkout -b stage
Switched to a new branch 'stage'

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (stage)
$ ls
Jenkinsfile README.md pom.xml src/

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (stage)
$ rm -rf Jenkinsfile

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git add .

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (stage)
$ git status
On branch stage
nothing to commit, working tree clean

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (stage)
```

Added Jenkinsfile to stage branch

```

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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
  a8fee5..f34091e stage -> stage

```

Branch Stage added in Jenkins multibranch pipeline

| S | W | Name ↓ | Last Success | Last Failure | Last Duration |
|---|---|--------|-----------------|--------------|---------------|
| | | dev | 29 min #1 | N/A | 27 sec |
| | | main | 33 min #1 | N/A | 28 sec |
| | | stage | 2 min 13 sec #3 | 13 min #2 | 28 sec |

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.

The screenshot shows the Jenkins Plugins page. On the left, there's a sidebar with links: Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress. The main area has a search bar at the top with the query 'multibranch scan webhook trigger'. Below the search bar, there's a table with one row. The row contains a checkbox, the plugin name 'Multibranch Scan Webhook Trigger 1.0.9', a description 'Trigger that can receive any HTTP request and trigger a multibranch job scan when token matched.', and a timestamp '2 yr 2 mo ago'. There are also 'Install' and 'Released' buttons.

Enable Scan by Webhook and set payload url accordingly

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

Content type: `application/json`

The screenshot shows the Jenkins Configuration page for a pipeline named 'Nam-trend-multibranch'. On the left, there's a sidebar with links: General, Branch Sources, Build Configuration, Scan Multibranch Pipeline Triggers (which is selected), Orphaned Item Strategy, Appearance, Health metrics, and a separator line. The main area is titled 'Scan Multibranch Pipeline Triggers'. It shows two options: 'Periodically if not otherwise run' (unchecked) and 'Scan by webhook' (checked). Below the checked option is a 'Trigger token' input field containing 'nam-token'. A tooltip explains that the token must match the webhook token to trigger a scan. At the bottom of the configuration section, there's a link 'Orphaned Item Strategy'.

add webhook

The screenshot shows the GitHub settings interface for a repository named 'tweet-trend-new'. The left sidebar is collapsed, and the main area is titled 'Webhooks / Add webhook'. The payload URL is set to 'http://44.213.127.95:8080/multibranch-webhook-trigger/invoke?token'. The content type is 'application/json'. The 'Active' checkbox is checked. Below the form, there's a note about triggering events: 'Just the push event.', 'Send me everything.', 'Let me select individual events.', and 'Active' (checked). A green 'Add webhook' button is at the bottom.

The screenshot shows the GitHub settings interface for the same repository. The left sidebar is collapsed, and the main area is titled 'Webhooks'. It displays a single webhook entry with the URL 'http://44.213.127.95:8080/multibranch-webhook-trigger/invoke?token' and a status indicator 'push'. There are 'Edit' and 'Delete' buttons next to the entry.

Made some changes in README.md file of main branch

The screenshot shows the GitHub repository page for 'tweet-trend-new'. The commit history shows a recent update to 'README.md' by user 'Namg04'. The commit message is 'Update README.md'. The commit details show the file was updated, with 7 lines (5 loc) and 236 Bytes. A note says 'Code 55% faster with GitHub Copilot'. The commit content is displayed below:

```
Nam-trend application

This is a small applicaiton which contains main and test folders.
Main contains application code.
Test contains test cases.
It also contains pom.xml which has all dependences and artifact name and version
```

Push has been triggered

Webhooks / Manage webhook

Recent Deliveries

- bfee9826-5cfa-11ee-8393-a67409b5900f push 2023-09-27 11:27:33 ...
- 25c0b639-5cfa-11ee-939e-9f940aa8257 ping 2023-09-27 11:23:14 ...

Build successfully triggered

Pipeline main

Stage View

| Declarative: Checkout SCM | build |
|---------------------------|-------|
| 416ms | 26s |

Average stage times:
(Average full run time: ~27s)

Build History

- #2 | Sep 27, 2023, 5:57 AM
- #1 | Sep 27, 2023, 5:13 AM

Permalinks

[Atom feed for all](#) [Atom feed for failures](#)

Nam-trend-multibranch

Branches (3)

| S | W | Name ↓ | Last Success | Last Failure | Last Duration |
|---|----|--------|-----------------|--------------|---------------|
| ✓ | ☀️ | dev | 43 min #1 | N/A | 27 sec |
| ✓ | ☀️ | main | 3 min 57 sec #2 | N/A | 27 sec |
| ✓ | ☁️ | stage | 16 min #3 | 28 min #2 | 28 sec |

Status

Configure

Scan Multibranch Pipeline Now

Scan Multibranch Pipeline Log

Multibranch Pipeline Events

Delete Multibranch Pipeline

People

Build History

Project Relationship

Check File Fingerprint

Rename

Pipeline Syntax

Icon: S M L

Icon legend: [Atom feed for all](#) [Atom feed for failures](#) [Atom feed for just latest builds](#)

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

Terms of Service'. The main content area shows logos for Microsoft, Apache, Wikimedia, brave, and TYPO3 under the heading 'TRUSTED BY'."/>

Go to Projects

Create an organization

An organization is a space where a team or a whole company can collaborate across many projects.



Manual setup is not recommended, and leads to missing features like appropriate setup of your project or analysis feedback in the Pull Request. We recommend to [import your organization](#).

1 Enter your organization details

Name

Up to 255 characters

Key *

Organization key must start with a lowercase letter or number, followed by lowercase letters, numbers or hyphens, and must end with a letter or number. Maximum length: 255 characters.

> Add additional info

Choose a free plan and click on Analyze new project

[Projects](#)[Quality Profiles](#)[Rules](#)[Quality Gates](#)[Members](#)[Billing & Upgrade](#)[Administration](#) ▾

No projects here yet

But your organization is all set, you can start analyzing code here!

[Analyze a new project](#)

Add Organization and project name

Analyze projects

Setup your project manually.



Manual setup is not recommended, and leads to missing features like appropriate setup of your project or analysis feedback in the Pull Request. We recommend to [import your projects](#)

Organization

 [Create another organization](#)

Display Name *

Up to 255 characters

Project Key *

Up to 400 characters. All letters, digits, dash, underscore, period or colon.

Project visibility

 Public

Anyone will be able to browse your source code and see the result of your analysis.

 Private [PAID PLAN](#)

[Generate Authentication token on sonar-qube](#)

sonarcloud 

[My Projects](#) [My Issues](#) [Explore](#) [!\[\]\(663236dede5b98d18a1036f15ae7e6e3_img.jpg\)](#)

 **Namg04**

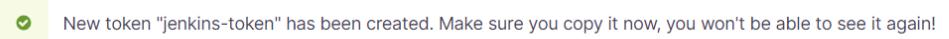
[Profile](#) [Security](#) [Notifications](#) [Organizations](#) [Appearance](#)

Security

If you want to enforce security by not providing credentials of a real SonarCloud user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

Enter Token Name [Generate Token](#)



| 250€ | 3cf30f |  |
|-----------------------------------|--------|---|
| Copy to clipboard | | |

Existing Tokens

| Name | Last use | Created | Action |
|---------------|----------|--------------------|------------------------|
| jenkins-token | Never | September 28, 2023 | Revoke |

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.

 Jenkins

[Dashboard](#) [Manage Jenkins](#) [Credentials](#) [System](#) [Global credentials \(unrestricted\)](#) [log out](#)

[Search \(CTRL+K\)](#) [?](#) [Bell icon](#) [Shield icon](#) [Admin icon](#) [Logout icon](#)

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

| ID | Name | Kind | Description | Action |
|--|-----------------------------------|-------------------------------|--------------------------|---|
|  maven-server-credentials | ubuntu (maven server credentials) | SSH Username with private key | maven server credentials |  |
|  Github_credentials | Namg04/***** | Username with password | |  |
|  sonar-credentials | sonar-credentials | Secret text | |  |

Icon: S M L

Download SonarQube plugings

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

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

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

| | |
|---|--|
| Name | <input type="text" value="namg-sonarqube-server"/> |
| Server URL | Default is http://localhost:9000
<input type="text" value="https://sonarcloud.io"/> |
| Server authentication token | SonarQube authentication token. Mandatory when anonymous access is disabled.
<input type="text" value="sonar-credentials"/>
<input type="button" value="Add ▾"/> |
| <input type="button" value="Advanced ▾"/> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <input type="button" value="Save"/> <input type="button" value="Apply"/> </div> | |

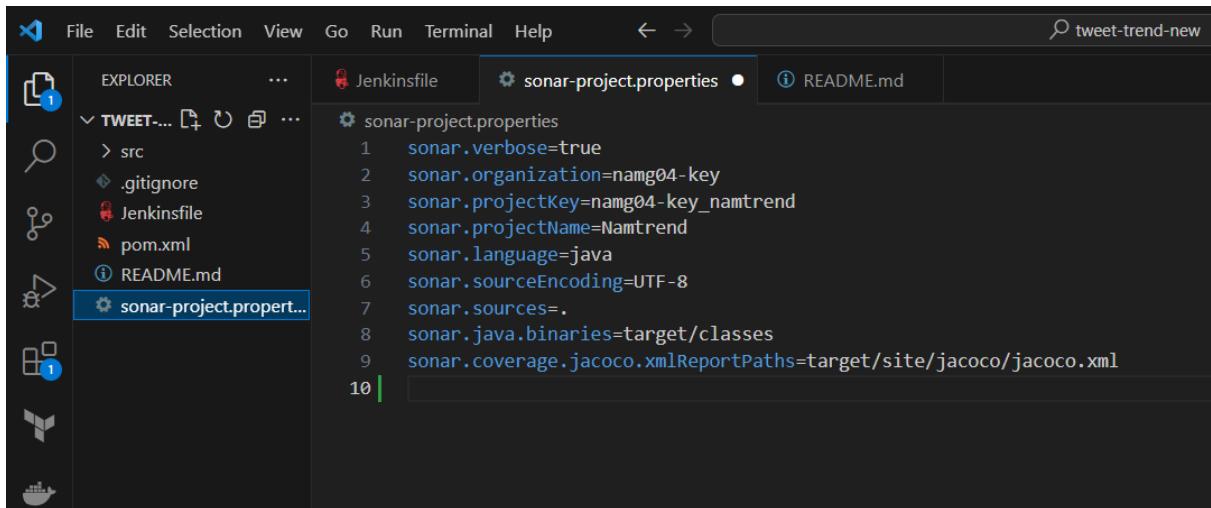
Add sonar qube scanner

Manage jenkins > Tools

SonarQube Scanner installations

| | |
|--|--|
| <input type="button" value="Add SonarQube Scanner"/> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <div> <p>SonarQube Scanner</p> <p>Name</p> <input type="text" value="namg-sonar-scanner"/> <p><input checked="" type="checkbox"/> Install automatically <small>?</small></p> <p><input type="button" value="Install from Maven Central"/></p> <p>Version</p> <input type="text" value="SonarQube Scanner 4.6.1.3023"/> <p><input type="button" value="Add Installer ▾"/></p> </div> <div> <p><input type="button" value="Save"/></p> <p><input type="button" value="Apply"/></p> </div> </div> | |
| <input type="button" value="Add SonarQube Scanner"/> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <input type="button" value="Save"/> <input type="button" value="Apply"/> </div> | |

write sonar-properties file



```
sonar-project.properties
1 sonar.verbose=true
2 sonar.organization=namg04-key
3 sonar.projectKey=namg04-key_namtrend
4 sonar.projectName=Namtrend
5 sonar.language=java
6 sonar.sourceEncoding=UTF-8
7 sonar.sources=.
8 sonar.java.binaries=target/classes
9 sonar.coverage.jacoco.xmlReportPaths=target/site/jacoco/jacoco.xml
```

commit to git

```
namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add .

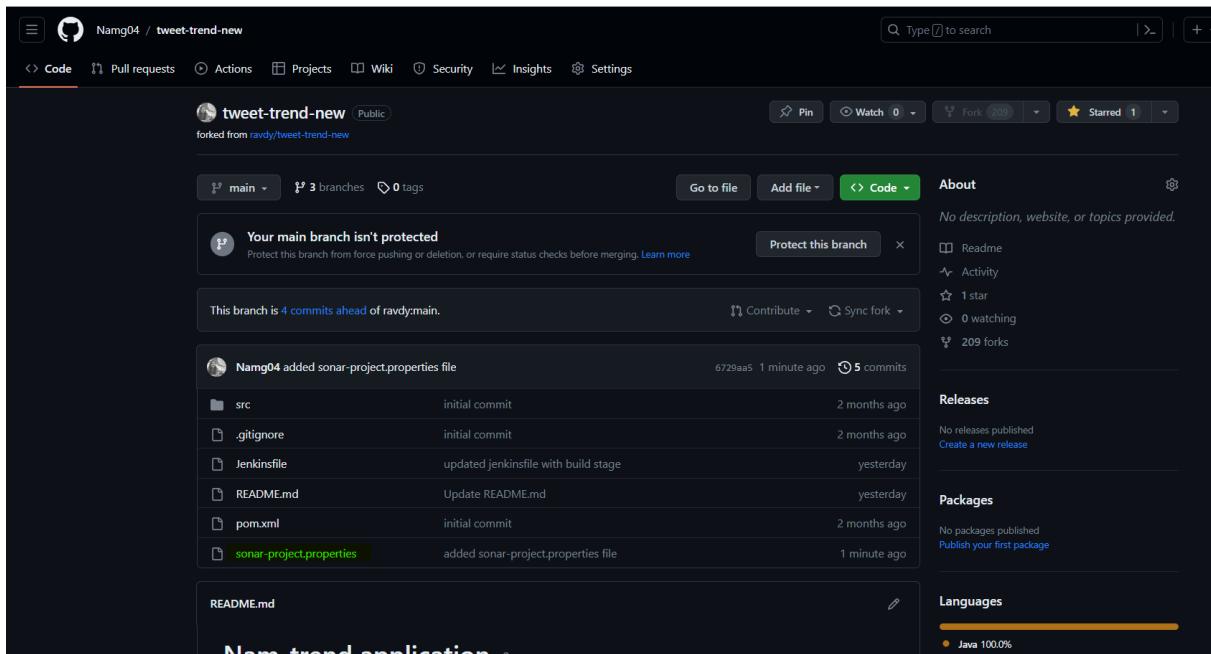
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "added sonar-project.properties file"
[main 6729aa5] added sonar-project.properties file
 1 file changed, 9 insertions(+)
 create mode 100644 sonar-project.properties

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ |
```



Add SonarQube stage in the Jenkinsfile.

```

Jenkinsfile | sonar-project.properties | README.md
1 pipeline {
2     agent {label 'maven'}
3
4
5     environment {
6         PATH = "/opt/apache-maven-3.9.4/bin:$PATH"
7     }
8     stages {
9         stage("build"){
10            steps {
11                echo "----- build started -----"
12                sh 'mvn clean deploy -Dmaven.test.skip=true'
13                echo "-----build completed -----"
14            }
15        }
16
17        stage("test"){
18            steps{
19                echo "-----unit test started -----"
20                sh 'mvn surefire-report:report'
21                echo "-----unit test completed -----"
22            }
23        }
24
25        stage('SonarQube analysis') {
26            environment{
27                scannerHome = tool 'namg-sonar-scanner' //sonar scanner name should be same as what we have defined in the tools
28            }
29
30            steps {
31                withSonarQubeEnv('namg-sonarqube-server') {
32
33                    sh "${scannerHome}/bin/sonar-scanner" // This is going to communicate with our sonar cube server and send the analysis report.
34                }
35            }
36        }
37    }
}

```

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(-)

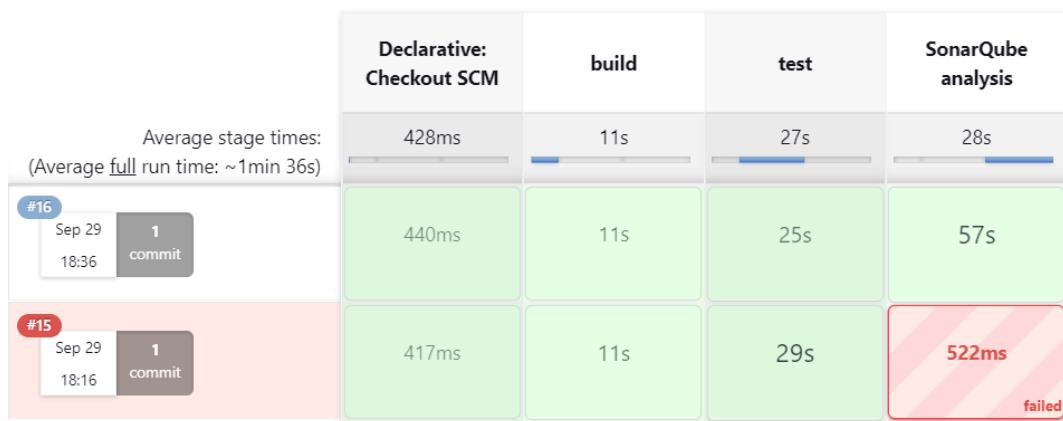
namratak@NAMRATAKFTVTCS3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 724 bytes | 724.00 KiB/s, done.
Total 3 (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

Namtrend N/A

server-side processing: Success

Permalinks

SonarQube Dashboard

The dashboard shows the following information:

- Project:** namg04 (PUBLIC)
- Quality Gate:** Not computed (Passed: 0, Failed: 0)
- Reliability:** A (0), B (0), C (1), D (0), E (0)
- Security:** 0
- Project Overview:**
 - Search: Search for projects
 - Perspective: Overall Status
 - Sort by: Name
 - 1 projects
 - Namtrend (NEW PUBLIC) - Last analysis: 9/29/2023, 6:37 PM - 1k Lines of Code - CSS, XML, ...
 - Bugs: 25, Vulnerabilities: 0, Hotspots Reviewed: 100%, Code Smells: 42, Coverage: 0.0%, Duplications: 0.0%

Add Quality gates

sonarcloud

My Projects My Issues Explore

N namg04 PUBLIC

Projects **Quality Profiles** Rules Quality Gates Members Billing & Upgrade Administration

namg04 > Quality Profiles

Quality Profiles

Quality Profiles are collections of rules to apply during an analysis. For each language there is a default profile. All projects not explicitly assigned to some other profile will be analyzed with the default. Ideally, all projects will use the same profile for a language. [Learn More](#)

Filter by

| Java, 1 profile | Projects | Rules | Updated | Used |
|-----------------|----------|-------|-------------|----------------|
| Sonar way | DEFAULT | 509 | 1 month ago | 42 minutes ago |

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.

N namg04 PUBLIC Key: namg04-key

Projects Quality Profiles Rules Quality Gates Members Billing & Upgrade Administration

Quality Gates

| Sonar way | DEFAULT | BUILT-IN |
|---------------------|---------|----------|
| namg-java-QG | | |

namg-java-QG

Conditions

This Quality Gate does not set any conditions on New Code. To get the most effective analysis from SonarCloud, you should set conditions on new code. [Learn More](#)

Conditions on Overall Code
Conditions on Overall Code apply to long-lived branches only.

| Metric | Operator | Value |
|-------------|-----------------|-------|
| Bugs | is greater than | 50 |
| Code Smells | is greater than | 50 |

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

sonarcloud

My Projects My Issues Explore Q

Namtrend
PUBLIC ★

Overview Main Branch Pull Requests 0 Branches 1

Information

Administration

Quality Gate has been successfully updated.

Choose which Quality Gate is associated with this project. Or create a new Quality Gate in your organization.

namg-Java-QG

This Quality Gate sets conditions on overall code but not on new code. It will not appear on branches. To enable it for these cases, add conditions on new code to [the Quality Gate](#)

© 2023, SonarCloud by SonarSource SA. All rights reserved. Terms Pricing Privacy

Run Build now

Status Changes Build Now View Configuration Full Stage View SonarQube Pipeline Syntax

Build History trend ▾ Filter builds... #17 Sep 29, 2023, 2:01 #16 Sep 29, 18:36 #15 Sep 29, 18:16

Build History trend ▾

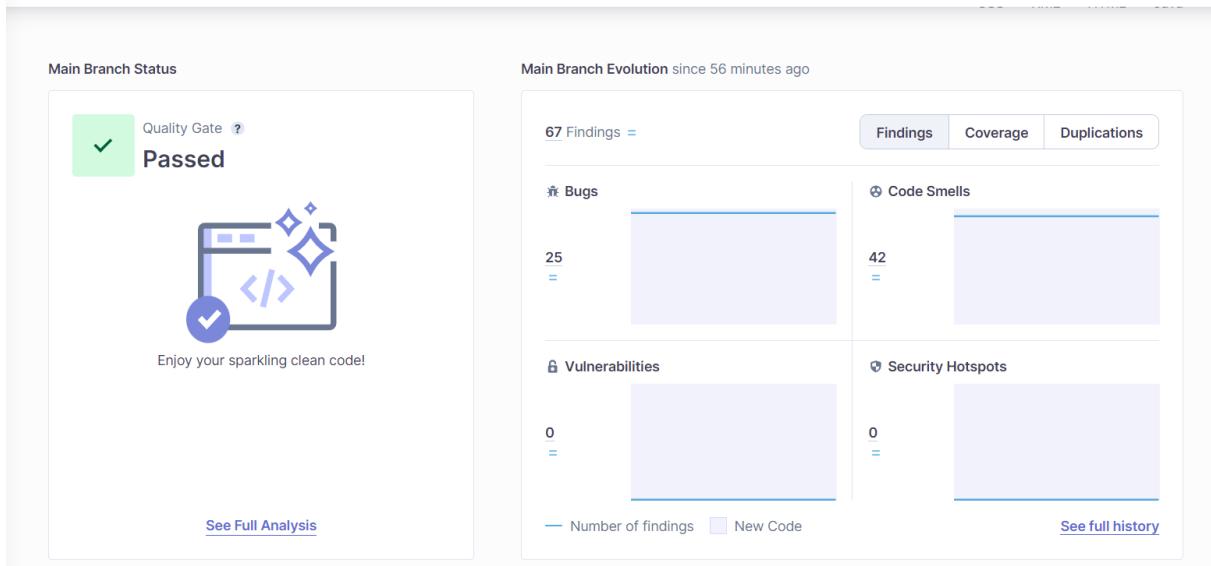
Full project name: Nam-trend-multibranch/main

Average stage times: (Average full run time: ~1min 34s)

Declarative: Checkout SCM build test SonarQube analysis

| #17 | Sep 29 19:31 | No Changes | 480ms | 11s | 26s | 36s |
|-----|--------------|------------|-------|-----|-----|--------------|
| | | | 585ms | 12s | 26s | 50s |
| #16 | Sep 29 18:36 | 1 commit | 440ms | 11s | 25s | 57s |
| #15 | Sep 29 18:16 | 1 commit | 417ms | 11s | 29s | 522ms failed |

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



Added sonar gate in Jenkinsfile

```

File Edit Selection View Go Run Terminal Help ↵ → tweet-trend-new
Jenkinsfile • sonar-project.properties • README.md
1 pipeline {
2   agent {label 'maven'}
3
4
5   environment {
6     PATH = "/opt/apache-maven-3.9.4/bin:$PATH"
7   }
8   stages {
9     stage("build"){
10       steps {
11         echo "----- build started -----"
12         sh 'mvn clean deploy -Dmaven.test.skip=true'
13         echo "-----build completed -----"
14       }
15     }
16   }
17   stage("test"){
18     steps{
19       echo "-----unit test started -----"
20       sh 'mvn surefire-report:report'
21       echo "-----unit test completed -----"
22     }
23   }
24   stage('SonarQube analysis'){
25     environment {
26       scannerHome = tool 'namg-sonar-scanner' //sonar scanner name should be same as what we have defined in the tools
27     }
28     steps {
29       withSonarQubeEnv('namg-sonarqube-server') {
30         sh "${scannerHome}/bin/sonar-scanner" // This is going to communicate with our sonar cube server and send the analysis report.
31       }
32     }
33   }
34   stage("Quality Gate") {
35     steps {
36       script {
37         timeout(time: 1, unit: 'HOURS') {
38           def qg = waitForQualityGate()
39           if (qg.status != 'OK') {
40             error "Pipeline aborted due to quality gate failure: ${qg.status}"
41           }
42         }
43       }
44     }
45   }
}

```

Commit the code in Git

```

$ git pull
hint: Waiting for your editor to close the file...      1 [sig] bash 2019! sigpacket::process
s: Suppressing signal 18 to win32 process (pid 21072) 228812 [sig] bash 2019! sigpacket::process
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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 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/Nam04/tweet-trend-new.git
  33af32f..f8b3c92  main -> main

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$
```

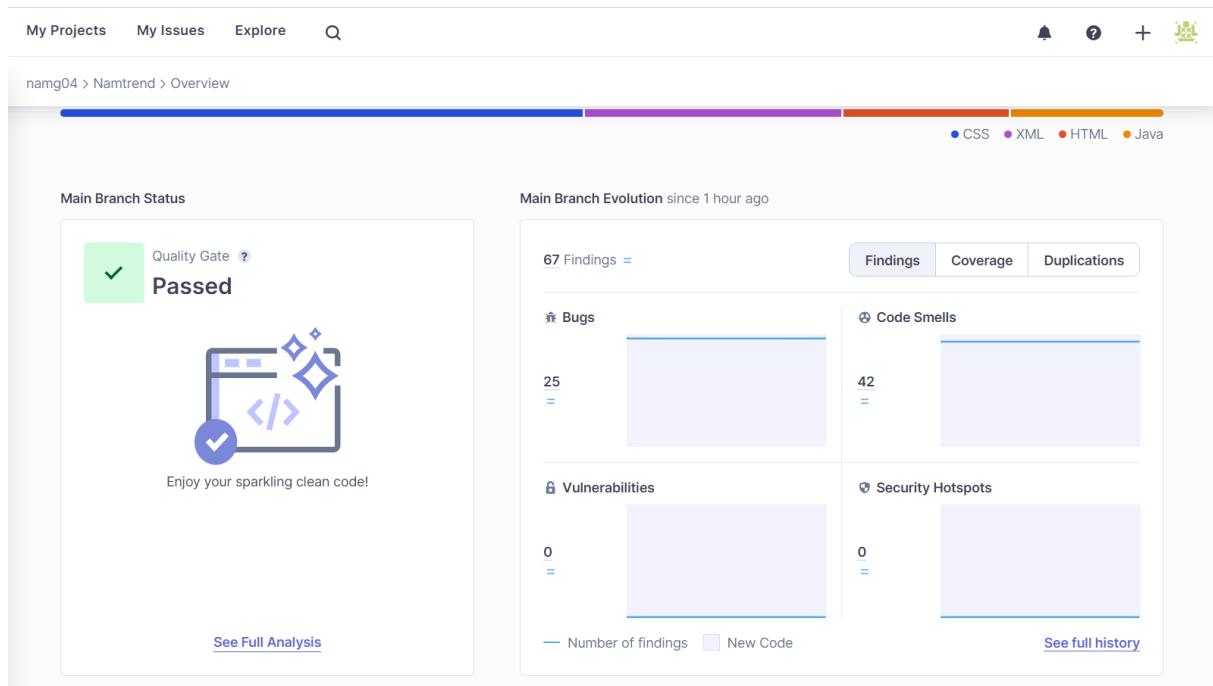
Build triggered automatically

The screenshot shows the Jenkins Pipeline main view. On the left, there's a sidebar with links for Status, Changes, Build Now, View Configuration, Full Stage View, SonarQube, and Pipeline Syntax. Below that is a Build History panel with a dropdown set to 'trend'. It lists builds #18, #17, and #16, each with a timestamp and commit count.

The main area is titled 'Pipeline main' and shows a 'Stage View' table. The table has columns for Declarative: Checkout SCM, build, test, SonarQube analysis, and Quality Gate. The first three columns show average times: 560ms, 11s, and 26s. The SonarQube analysis column shows 39s. The Quality Gate column shows 606ms. The table also includes a header row with these column names.

| Declarative: Checkout SCM | build | test | SonarQube analysis | Quality Gate |
|--|-------|------|--------------------|--------------|
| Average stage times:
(Average full run time: ~1min 33s) | 560ms | 11s | 26s | 39s |
| #18
Sep 29 19:50
1 commit | 799ms | 12s | 25s | 50s |
| #17
Sep 29 19:31
No Changes | 585ms | 12s | 26s | 50s |
| #16
Sep 29 18:36
1 commit | 440ms | 11s | 25s | 57s |

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 definition of registry `def registry = 'https://valaxy01.jfrog.io'`
2. Credentials id in the place of `jfrogforjenkins` in the `def server = Artifactory.newServer url:registry+"/artifactory" , credentialsId:"jfrogforjenkins"`
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.**

Set up your JFrog Platform Environment

Free 14-Day Trial

Create a Hostname* <https://namg04.jfrog.io>

This will be your team's subdomain.

Your company's name or another unique name 

Company*

Enter your company name

Phone

Enter your phone number

Hosting Preferences

Select a Cloud Provider for your JFrog Environment ⓘ



Cloud Region* ⓘ

Select Server Region

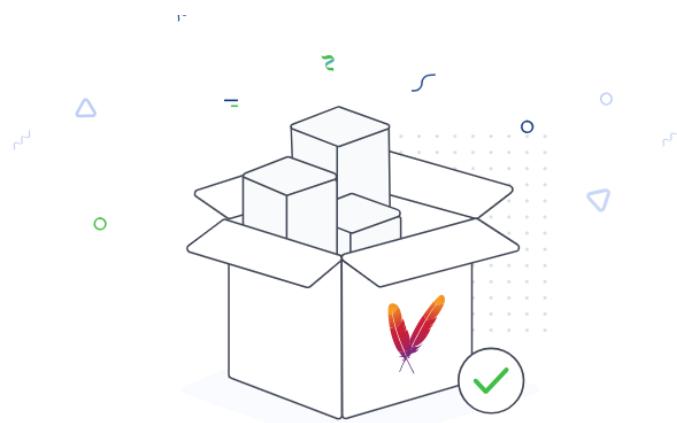
Create Maven repository

Create Repositories
Select the package type you want - we'll create the default repositories for you!

Search by package type 

| | | | | | | | |
|---|---|---|--|--|--|--|---|
| 
Alpine | 
Bower | 
Cargo | 
Chef | 
CocoaPods | 
Composer | 
Conan | 
Conda |
| 
CRAN | 
Debian | 
Docker | 
Gems | 
Generic | 
GitLfs | 
Go | 
Gradle |
| 
Helm | 
HuggingFace ML | 
Ivy | 
Maven | 
npm | 
Npm | 
NuGet | 
OCI |
| 
Npm | 
Python | 
sbt | 
Swift | 
Yarn | 
Veeam | 
RPM | |

[View scan results](#)



Your maven Repository was Created Successfully!

Maven Repository: namg-libs-snapshot

🔗 URL:<https://artifactory/api/maven/namg-libs-snapshot> 🌐

Next, connect your repository to a project build or a maven client.

[I'll Do It Later](#) [Continue](#)

The screenshot shows the JFrog Artifactory web interface. On the left, there's a dark sidebar with navigation links like 'Get Started', 'Artifactory', 'Release Bundles v2', 'Repositories', 'Packages', 'Builds', and 'Artifacts'. The 'Artifacts' link is highlighted. The main area has a search bar at the top with 'Search Artifacts'. Below it, a message says 'Happily serving 425 artifacts'. A tree view on the left lists repositories: 'namg-libs-release', 'namg-libs-snapshot', 'artifactory-build-info', 'namg-libs-release-local', 'namg-libs-snapshot-local', 'namg-maven-remote', and 'namg-maven-remote-cache'. The 'namg-libs-release' repository is selected and shown in detail on the right. Its 'General' tab is active, displaying the 'URL to file:' as <https://namg04.jfrog.io/artifactory/namg-libs-release/>. Other tabs include 'Info', 'Package Information', 'Dependency Declaration', and 'Virtual Repository Associations'. Top navigation includes 'Upgrade Now', 'MyJFrog Portal', 'Set Me Up', and 'Deploy'.

[Create access token](#)

The screenshot shows the JFrog Artifactory web interface. On the left, there's a dark sidebar with navigation links like 'Get Started', 'Artifactory' (selected), 'Builds', 'Artifacts' (selected), and 'Xray'. The main area displays 'Happily serving 425 artifacts' and a list of repositories under 'Filter repositories'. A modal window titled 'Platform Configurations' is open, showing a tree view with 'User Management' selected. Other options include 'Projects', 'Environments', 'User Authentication', 'Platform Security', 'Platform Management', 'Platform Monitoring', 'Topology', 'Artifactory Settings', and 'Xray Settings'. At the bottom of the modal, it says 'Virtual Repository Associations'.

Generate token

The screenshot shows the 'Access Tokens' page under 'User Management'. A modal window titled 'Generate Token' is open, showing fields for 'Description' (set to 'jenkins-access-token'), 'Token scope' (set to 'Admin'), 'User name' (set to 'namratakumari043@gmail.com'), 'Service' (set to 'Select'), and 'Expiration time' (set to 'Never'). There are 'Close' and 'Generate' buttons at the bottom. In the background, the 'Tokens' table lists several tokens, and a search bar is visible.

Token generated

| Tokens | | | | | | |
|----------------------------------|-------------|-------------------------------|--------------|--------------|----------------------------------|---------------------------|
| Descript... | Subject | Token ID | Issued At | Expiry D... | Refresha... | Scope |
| <input type="checkbox"/> doc... | namratak... | 1de24191-97d7-4897-ad64-8... | 30-09-23 ... | 29-09-24 ... | <input type="button" value="X"/> | applied-permissions/user |
| <input type="checkbox"/> jenk... | namratak... | a7a11979-6ca8-46fb-a157-51... | 30-09-23 ... | | <input type="button" value="X"/> | applied-permissions/admin |

Add credentials in Jenkins

| Global credentials (unrestricted) | | | | |
|---|-----------------------------------|-------------------------------|--------------------------|--|
| Credentials that should be available irrespective of domain specification to requirements matching. | | | | |
| ID | Name | Kind | Description | |
|  maven-server-credentials | ubuntu (maven server credentials) | SSH Username with private key | maven server credentials |  |
|  Github_credentials | Namg04/***** | Username with password | |  |
|  sonar-credentials | sonar-credentials | Secret text | |  |
|  f4c62237-6e0f-476f-a468-5a948c754c37 | namratakumari043@gmail.com/***** | Username with password | |  |

Now install 'Artifactory' plugins

Dashboard > Manage Jenkins > Plugins

| Updates | Download progress |
|---|--|
|  Available plugins | Preparation |
|  Installed plugins | <ul style="list-style-type: none"> • Checking internet connectivity • Checking update center connectivity • Success |
|  Advanced settings | SonarQube Scanner  Success |
|  Download progress | Loading plugin extensions  Success |
| | Config File Provider  Success |
| | Javadoc  Success |
| | JSch dependency  Success |
| | Maven Integration  Success |
| | Artifactory  Success |
| | Loading plugin extensions  Success |

→ [Go back to the top page](#)
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

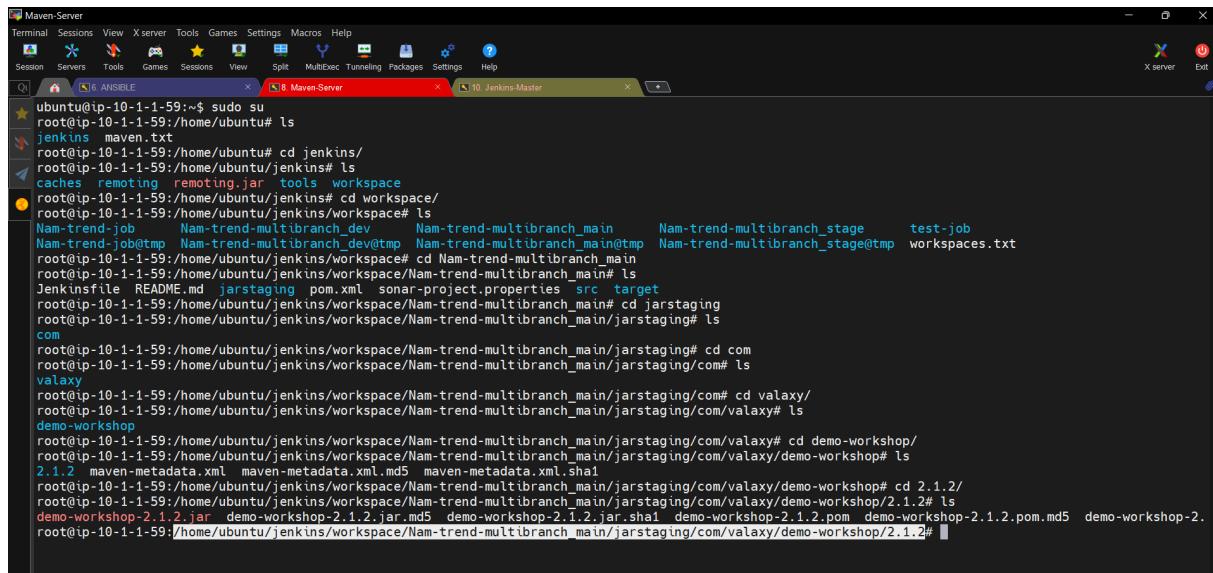
Update Jenkinsfile with jar publish stage.

source code path:

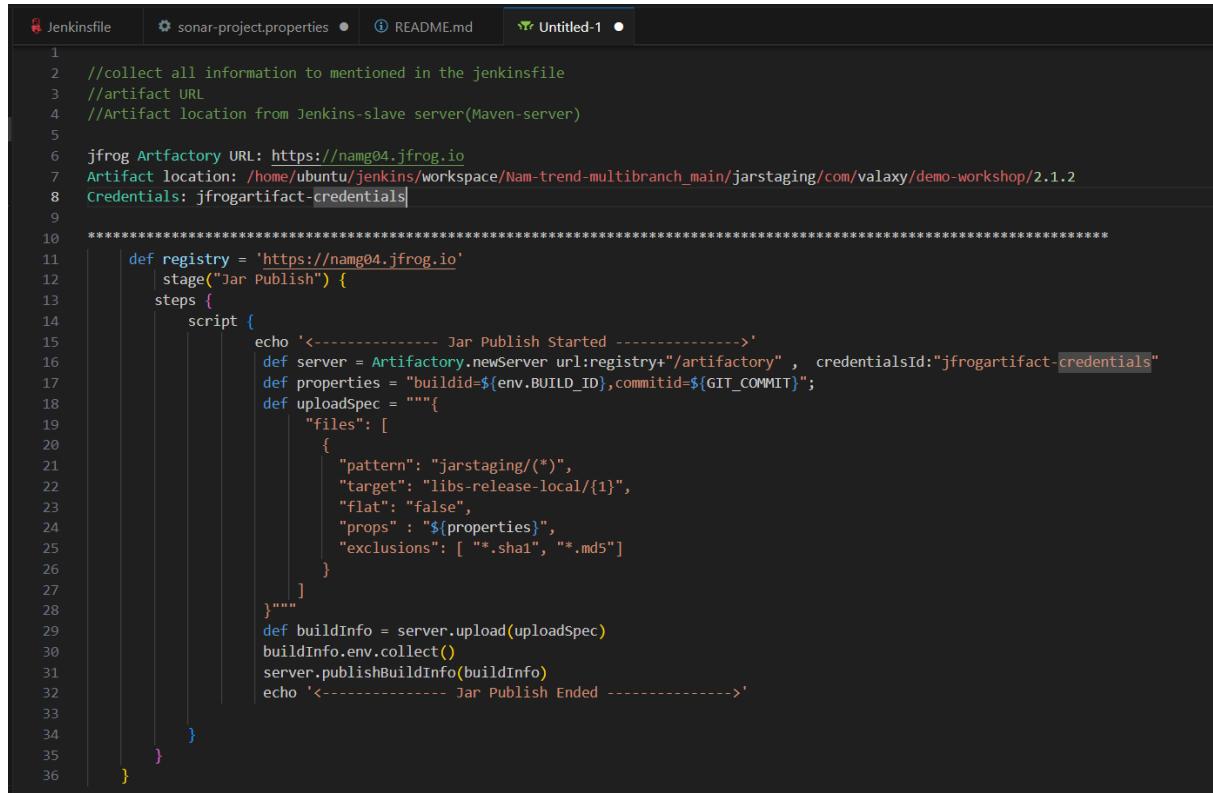
jfrog Artifactory url: <https://sanam01.jfrog.io>

Artifact location: /home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop/2.1.2

Credentials: jfrogartifact-credentials

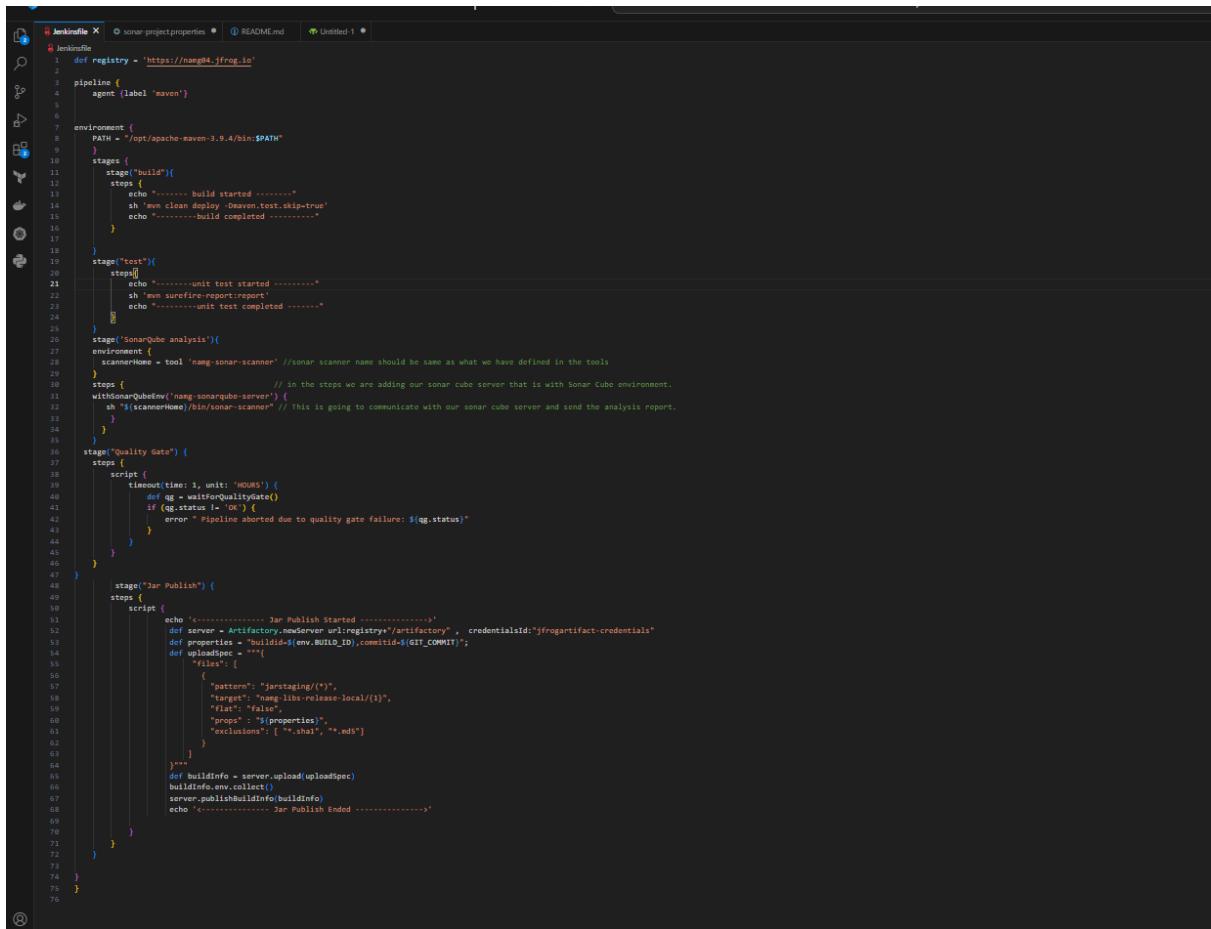


```
Ubuntu@ip-10-1-1-59:~$ sudo su
root@ip-10-1-1-59:/home/ubuntu# ls
jenkins maven.txt
root@ip-10-1-1-59:/home/ubuntu# cd jenkins/
root@ip-10-1-1-59:/home/ubuntu/jenkins# ls
caches remoting remotng.jar tools workspace
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace#
Nam-trend-job Nam-trend-multibranch dev Nam-trend-multibranch_main Nam-trend-multibranch_stage test-job
Nam-trend-job@tmp Nam-trend-multibranch_devtmp 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-2.1.2.pom.sha1
root@ip-10-1-1-59:/home/ubuntu/jenkins/workspace/Nam-trend-multibranch_main/jarstaging/com/valaxy/demo-workshop/2.1.2#
```



```
1 //collect all information to mentioned in the jenkinsfile
2
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 = Artfactory.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)



```

1 def registry = 'https://nang84.jfrog.io'
2
3 pipeline {
4     agent {label 'maven'}
5
6     environment {
7         PATH = "/opt/apache-maven-3.9.4/bin:$PATH"
8     }
9     stages {
10         stage("build") {
11             steps {
12                 echo "..... build started ....."
13                 sh 'mvn clean deploy -Dmaven.test.skip=true'
14                 echo ".....build completed ....."
15             }
16         }
17         stage("test") {
18             steps {
19                 echo ".....unit test started ....."
20                 sh 'mvn surefire-report:report'
21                 echo ".....unit test completed ....."
22             }
23         }
24     }
25     stage("SonarQube analysis") {
26         environment {
27             scannerHome = tool 'nang-sonar-scanner' //sonar scanner name should be same as what we have defined in the tools
28         }
29         steps {
30             withSonarQubeEnv('nang-sonarqube-server') {
31                 sh "${scannerHome}/bin/sonar-scanner" // This is going to communicate with our sonar cube server and send the analysis report.
32             }
33         }
34     }
35     stage("Quality Gate") {
36         steps {
37             script {
38                 timeout(time: 1, unit: 'HOURS') {
39                     def gg = waitForQualitygate()
40                     if (gg.status != 'OK') {
41                         error "Pipeline aborted due to quality gate failure: ${gg.status}"
42                     }
43                 }
44             }
45         }
46     }
47     stage("Jar Publish") {
48         steps {
49             script {
50                 echo '..... Jar Publish Started .....'
51                 def server = Artifactory.newServer(url:'registry/artifactory', credentialsId:'jfrogartifact-credentials'
52                 def properties = "buildId=${env.BUILD_ID},commitId=${GIT_COMMIT}";
53                 def uploadSpec = """
54                     #files: [
55                         {
56                             "pattern": "*/target/*",
57                             "target": "nang-libs-release-local/{1}",
58                             "flat": "false",
59                             "props": "${properties}",
60                             "exclusions": [ "*.shai", "*.md5" ]
61                         }
62                     ]"""
63                 def buildInfo = server.upload.uploadSpec(uploadSpec)
64                 buildInfo.env.collect()
65                 server.publishBuildInfo(buildInfo)
66                 echo '..... Jar Publish Ended .....'
67             }
68         }
69     }
70 }
71
72
73
74 }
75
76

```

Committed code into Git

```

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git fetch

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ ls
Jenkinsfile README.md pom.xml sonar-project.properties src/
namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ 

```

Build successful

Dashboard > Nam-trend-multibranch > main >

Pipeline main

- Status
- Changes
- Build Now
- View Configuration
- Full Stage View
- SonarQube
- Pipeline Syntax

Stage View

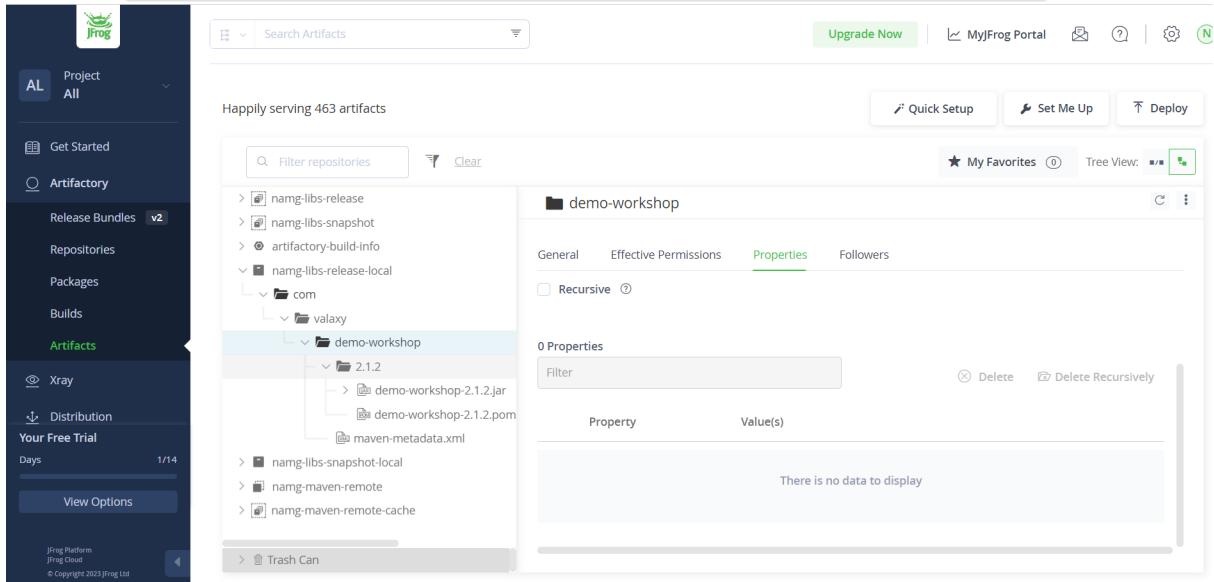
Average stage times: (Average full run time: ~1min 33s)

| #21 | Declarative: Checkout SCM | build | test | SonarQube analysis | Quality Gate | Jar Publish |
|-----------------|---------------------------|-------|------|--------------------|--------------|-------------|
| Sep 30
10:32 | 587ms | 12s | 26s | 42s | 535ms | 3s |
| #20 | 606ms | 12s | 25s | 49s | 505ms | 2s |
| Sep 30
10:09 | 677ms | 12s | 26s | 47s | 496ms | 4s failed |
| #19 | No Changes | | | | | |
| #18 | | | | | | |

Build History trend ▾

- Filter builds...
- #21 Sep 30, 2023, 5:02 AM
- #20 Sep 30, 2023, 4:39 AM
- ...

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**

```

# ...
- 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**

```

ubuntu@ip-10-1-1-59:~$ docker --version
Command 'docker' not found, but can be installed with:
  sudo snap install docker      # version 20.10.24, or
  sudo apt install podman-docker # version 3.4.4+ds1-ubuntu1.22.04.2
  sudo apt install docker.io     # version 24.0.5-0ubuntu1~22.04.1
See 'snap info docker' for additional versions.
ubuntu@ip-10-1-1-59:~$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
ubuntu@ip-10-1-1-59:~$ service docker status
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2023-09-30 07:58:11 UTC; 28s ago
    TriggeredBy: ● docker.socket
      Docs: https://docs.docker.com
    Main PID: 50593 (dockerd)
      Tasks: 8
        Memory: 30.5M
          CPU: 288ms
        CGroup: /system.slice/docker.service
                └─50593 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Sep 30 07:58:10 ip-10-1-1-59 systemd[1]: Starting Docker Application Container Engine...
Sep 30 07:58:10 ip-10-1-1-59 dockerd[50593]: time="2023-09-30T07:58:10.527413853Z" level=info msg="Starting up"
Sep 30 07:58:10 ip-10-1-1-59 dockerd[50593]: time="2023-09-30T07:58:10.529175425Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolv.conf"
Sep 30 07:58:10 ip-10-1-1-59 dockerd[50593]: time="2023-09-30T07:58:10.650383679Z" level=info msg="Loading containers: start."
Sep 30 07:58:11 ip-10-1-1-59 dockerd[50593]: time="2023-09-30T07:58:11.037155612Z" level=info msg="Loading containers: done."
Sep 30 07:58:11 ip-10-1-1-59 dockerd[50593]: time="2023-09-30T07:58:11.163615038Z" level=info msg="Docker daemon" commit="24.0.5-0ubuntu1~22.04.1" graphdriver="overlay2"
Sep 30 07:58:11 ip-10-1-1-59 dockerd[50593]: time="2023-09-30T07:58:11.164071745Z" level=info msg="Daemon has completed initialization"
Sep 30 07:58:11 ip-10-1-1-59 dockerd[50593]: time="2023-09-30T07:58:11.214975533Z" level=info msg="API listen on /run/docker.sock"
Sep 30 07:58:11 ip-10-1-1-59 systemd[1]: Started Docker Application Container Engine.
lines 1-21/21 (END)

```

4. Committed code into Git

```

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git fetch

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTC53 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<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@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git add V1-jenkins-slave-setup.yml

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git add V2-jenkins-slave-setup.yml

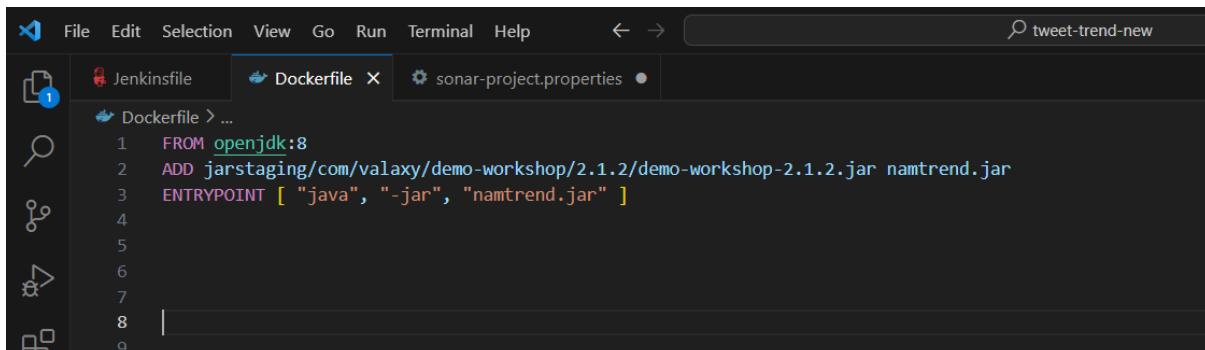
namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ git commit -m "added docker installation in Ansible playbook"
[main b4516da] added docker installation in Ansible playbook
 2 files changed, 60 insertions(+)
  create mode 100644 Ansible/V1-jenkins-slave-setup.yml
  create mode 100644 Ansible/V2-jenkins-slave-setup.yml

namratak@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Ansible (main)
$ |

```

Write and add Dockerfile in the source code



A screenshot of a code editor window titled "tweet-trend-new". The tabs at the top are "Jenkinsfile", "Dockerfile", and "sonar-project.properties". The "Dockerfile" tab is active, showing the following content:

```
FROM openjdk:8
ADD jarstaging/com/valaxy/demo-workshop/2.1.2/demo-workshop-2.1.2.jar namtrend.jar
ENTRYPOINT [ "java", "-jar", "namtrend.jar" ]
```

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

Your main branch isn't protected
Protect this branch

This branch is 23 commits ahead of ravyd:main.

| File | Commit Message | Time |
|--------------------------|---|---------------|
| src | initial commit | 2 months ago |
| .gitignore | initial commit | 2 months ago |
| Dockerfile | added Dockerfile | 2 minutes ago |
| Jenkinsfile | added jar publish stage | 3 hours ago |
| README.md | Update README.md | 19 hours ago |
| pom.xml | Revert "Update pom.xml" | 2 days ago |
| sonar-project.properties | rectified sonar-project.properties file | 2 days ago |

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

Happily serving 545 artifacts

demo-workshop

Properties

Add: Property Property Set

Property name: Property value:

Recursive

Quick Repository Creation

- Set Me Up
- New Local Repository
- New Remote Repository
- New Virtual Repository
- New User
- New Group
- New Permission

Create Repositories

Select the package type(s) you want - we'll create the default repositories for you!

Search by package type 🔍

| | | | | | | | |
|--------|----------------|--------|-------|-----------|----------|-------|--------|
| Alpine | Bower | Cargo | Chef | CocoaPods | Composer | Conan | Conda |
| CRAN | Debian | Docker | Gems | Generic | GitLfs | Go | Gradle |
| Helm | HuggingFace ML | Ivy | Maven | Npm | NuGet | OCI | Opkg |
| | | | sbt | | | V | fpm |

Next

Create Repositories

Assign a name to your new repositories by adding a meaningful prefix identifier

Repositories prefix ? 4/20

| | | |
|--|---|---|
|  Docker >Delete | | |
| <input checked="" type="checkbox"/> Local Repository
namg-docker-local | <input checked="" type="checkbox"/> Remote Repository ?
https://registry-1.docker...
namg-docker-remote | <input checked="" type="checkbox"/> Virtual Repository
namg-docker |

< Back Create

Repo created

Repositories

Virtual Local Remote Federated

3 Repositories

| Repository Key | Type | Project | Environment | Selected Repositories | |
|--------------------|--------|---------|-------------|----------------------------|--|
| namg-docker | Docker | | | 2 namg-docker-local, ... | <button>Set Up Client/CI Tool</button> |
| namg-libs-release | Maven | | | 2 namg-libs-release-l... | <button>Set Up Client/CI Tool</button> |
| namg-libs-snapshot | Maven | | | 2 namg-libs-snapshot... | <button>Set Up Client/CI Tool</button> |

Install `docker pipeline` plugin in Jenkins

Plugins

Search: docker pipeline

Install Name Released

Docker Pipeline 572.v950f58993843
✓ pipeline DevOps Deployment docker 1 mo 19 days ago
 Build and use Docker containers from pipelines.

Update Jenkins file with the below stages

Check-point:

- Provide jfrog repo URL in the place of `valaxy01.jfrog.io/valaxy-docker` in `def imageName = 'namg04.jfrog.io/namg-docker/namtrend'`
- Match version number in `def version = '2.1.2'` with pom.xml version number
- 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`

```

Jenkinsfile X Dockerfile sonar-project.properties

Jenkinsfile
1 def registry = 'https://namg04.jfrog.io'
2 def imageName = 'namg04.jfrog.io/namg-docker-local/namtrend'
3 def version = '2.1.2'
4
5 pipeline {
6   agent {label 'maven'}
7 }
8

```

```
79      stage(" Docker Build ") {
80          steps {
81              script {
82                  echo '<----- Docker Build Started ----->'
83                  app = docker.build(imageName+"."+version)
84                  echo '<----- Docker Build Ends ----->'
85              }
86          }
87      }
88
89      stage (" Docker Publish "){
90          steps {
91              script {
92                  echo '<----- Docker Publish Started ----->'
93                  docker.withRegistry(registry, 'jfrogartifact-credentials'){
94                      app.push()
95                  }
96                  echo '<----- Docker Publish Ended ----->'
97              }
98          }
99      }
100
101 }
102 }
103 }
```

Commit updated Jenkins file into Git

```

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add .

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ |

```

Build triggered

Dashboard > Nam-trend-multibranch > main >

[Build Now](#)

[View Configuration](#)

[Full Stage View](#)

[SonarQube](#)

[Pipeline Syntax](#)

Stage View

| | Declarative: Checkout SCM | build | test | SonarQube analysis | Quality Gate | Jar Publish | Docker Build | Docker Publish |
|--|---------------------------|-------|------|--------------------|--------------|-------------|--------------|----------------|
| Average stage times:
(Average full run time: ~1min 44s) | 931ms | 12s | 26s | 5min 5s | 560ms | 2s | 4s | 9s |
| #26 Sep 30 15:24 1 commit | 605ms | 12s | 26s | 50s | 457ms | 3s | 16s | 39s |
| #25 Sep 30 15:02 1 commit | 613ms | 12s | 25s | 50s | 694ms | 1s | 759ms failed | 64ms failed |
| #24 Sep 30 14:55 1 commit | 577ms | 12s | 26s | 50s | 557ms | 2s | 694ms failed | 58ms failed |
| #23 Sep 30 14:36 1 commit | 3s | 14s | 26s | 51s | 918ms | 5s | 1s failed | 60ms failed |

Build History trend ▾

- #26 Sep 30, 2023, 9:54
- #25 Sep 30, 2023, 9:32 AM
- #24 Sep 30, 2023, 9:25 AM
- #23 Sep 30, 2023, 9:06 AM

[Filter builds...](#)

Verify Docker repo in Jfrog artifactory

The screenshot shows the JFrog Artifactory web interface. On the left, there's a sidebar with navigation links like 'Get Started', 'Artifactory', 'Release Bundles v2', 'Repositories', 'Packages', 'Builds', 'Artifacts' (which is selected), 'Xray', and 'Distribution'. A 'Your Free Trial' section indicates 1/14 days remaining. The main area displays a tree view of repositories under 'namg-docker-local'. One node, 'namtrend' under '2.1.2', is highlighted in green. To the right, a detailed view of the '2.1.2' version is shown with tabs for 'General', 'Info', 'Effective Permissions', 'Properties', and 'Followers'. The 'Info' tab displays information such as Name: 2.1.2, Repository Path: namg-docker-local/namtrend/2.1.2, URL to file: https://namg04.jfrog.io/artifactory/namg-docker-loc..., Deployed By: namratakumar043@gmail.com, and Created: 30-09-23 09:57:01 +00:00 (0d 0h 12m 44s ago).

Testing Docker image by creating container out of it

```

Maven-Server
Terminal Sessions View Xserver Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Session Servers Tools Games Sessions View 10 Jenkins-Master 11 ANSIBLE
ubuntu@ip-10-1-1-59:~$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
namg04.jfrog.io/namg-docker-local/namtrend  2.1.2    add6ba5b7034  15 minutes ago  545MB
openjdk              8        b273004037cc  14 months ago  526MB
ubuntu@ip-10-1-1-59:~$ docker run -dt --name namtrend -p 8000:8000 namg04.jfrog.io/namg-docker-local/namtrend:2.1.2
a6750fed4db2b05b7e8393029ab8a1f542f1940d8cb3a4cff9d26539f450f6a85
ubuntu@ip-10-1-1-59:~$ docker ps -a
CONTAINER ID   IMAGE           COMMAND            CREATED          STATUS          PORTS
 NAMES
a6750fed4db2b  namg04.jfrog.io/namg-docker-local/namtrend:2.1.2   "java -jar namtrend..."   14 seconds ago   Up 13 seconds   0.0.0.0:8000->8000/tcp,  :::8000
  namtrend
ubuntu@ip-10-1-1-59:~$ 

```

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

Not secure | http://3.237.79.159:8000

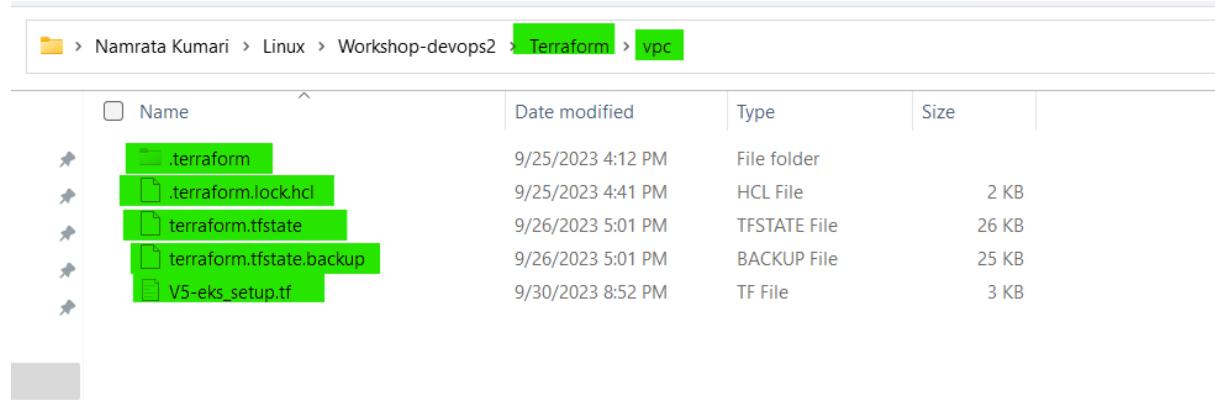
Greetings from NamG DevTalks

Kubernetes Setup

To set up a Kubernetes cluster using Terraform, follow these steps:

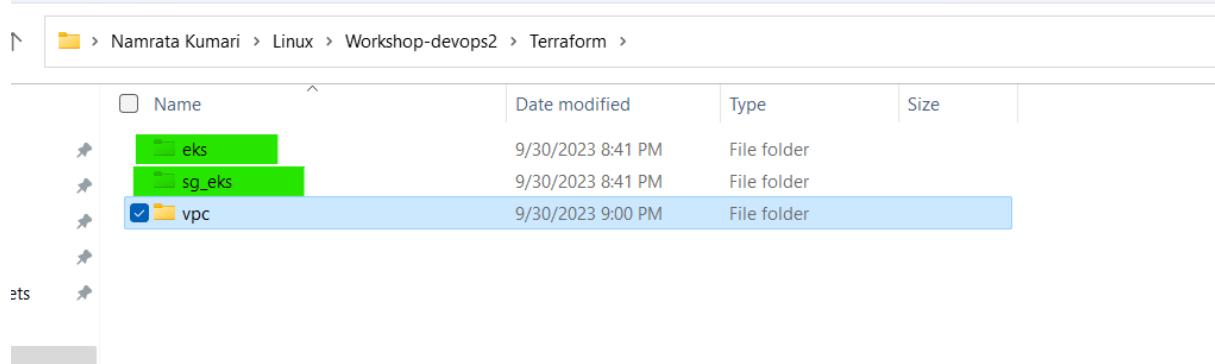
- 1) Access the EKS module code from the provided GitHub repository. This module helps create IAM roles, policies, the EKS cluster, and a node group.
 - 2) Copy the eks and sg_eks modules into the Terraform folder.
 - 3) Create a VPC folder and move existing files into it.
 - 4) Add the sg_eks and eks 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



| Name | Date modified | Type | Size |
|--------------------------|-------------------|--------------|-------|
| .terraform | 9/25/2023 4:12 PM | File folder | |
| .terraform.lock.hcl | 9/25/2023 4:41 PM | HCL File | 2 KB |
| terraform.tfstate | 9/26/2023 5:01 PM | TFSTATE File | 26 KB |
| terraform.tfstate.backup | 9/26/2023 5:01 PM | BACKUP File | 25 KB |
| V5-eks_setup.tf | 9/30/2023 8:52 PM | TF File | 3 KB |

Copy eks and sg_eks modules onto terraform folder



| Name | Date modified | Type | Size |
|--------|-------------------|-------------|------|
| eks | 9/30/2023 8:41 PM | File folder | |
| sg_eks | 9/30/2023 8:41 PM | File folder | |
| vpc | 9/30/2023 9:00 PM | File folder | |

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

The screenshot shows a code editor interface with several tabs at the top: eks.tf, V5-eks_setup.tf (which is the active tab), sg.tf, and V4-EC2-With_VPC_for_each.tf. The left sidebar displays a file tree under the 'Terraform' category, including sub-directories like eks, sg_eks, vpc, and .vpc, along with files such as eks.tf, output.tf, variables.tf, sg.tf, sg_eks.tf, V5-eks_setup.tf, .terraform.lock.hcl, .terraform, .terraform.state, .gitignore, and README.md.

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

module "sgs" {
    source = "../sg_eks"
    vpc_id      = aws_vpc.Nam-vpc.id
}

module "eks" {
    source = "../eks"
    vpc_id      = aws_vpc.Nam-vpc.id
    subnet_ids = [aws_subnet.Nam-public-subnet-01.id, aws_subnet.Nam-public-subnet-02.id]
    sg_ids = module.sgs.security_group_public
}
```

Added EKS module to Terraform

```

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ ls
Ansible/ Docker/ Jenkins/ README.md Terraform/

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git add ^

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git add Terraform/eks/ Terraform/sg_eks/ Terraform/vpc/
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ git commit -m "added EKS module to terraform code"
> ^C

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ ls
Ansible/ Docker/ Jenkins/ README.md Terraform/
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2 (main)
$ cd Terraform/
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ ls
eks/ sg_eks/ vpc/
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2/Terraform (main)
$ cd vpc/
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ ls
V4-EC2-With_VPC_for_each.tf terraform.tfstate terraform.tfstate.backup
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ mv V4-EC2-With_VPC_for_each.tf V5-eks_setup.tf
namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ git add .
namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC53 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@NAMRATAKFTVTC53 MINGW64 ~/Linux/Workshop-devops2/Terraform/vpc (main)
$ terraform validate
Success! The configuration is valid.

namratak@NAMRATAKFTVTC53 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-024dfffc21f2a8c3]
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-098613d9fa83ffffe7]
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@NAMRATAKFTVTC53 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-024dfffc21f2a8c3]
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-098613d9fa83ffffe7]
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-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

```

```

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@NAMRATAKFTVTC3 MINGW64 ~/Linux/workshop-devops2/Terraform/vpc (main)
$
```

Cluster created in AWS

The screenshot shows the AWS EKS console with the URL [https://console.aws.amazon.com/eks/clusters?region=us-east-1](#). The page displays a table titled 'Clusters (1) Info' with one item. The table has columns: Cluster name, Status, Kubernetes version, and Provider. The single entry is 'namg-eks-01' (Status: Active, Kubernetes version: 1.27, Provider: EKS). Above the table, a message box says 'New Kubernetes versions are available for 1 cluster.'

| Clusters (1) Info | | | | | |
|---|--------------------------------------|---|--------------------------------------|---------------------------------|-----|
| Filter clusters C Delete Add cluster ▾ | | | | | |
| Cluster name | ▲ | Status | ▼ | Kubernetes version | ▼ |
| namg-eks-01 | ● | Active | ↻ | 1.27 Update now | EKS |

To destroy EKS infrastructure, just comment modules in vpc project

The screenshot shows the VS Code code editor with the file 'V5-eks_setup.tf' open. The file contains Terraform code for setting up an EKS cluster. There are several sections of code that are currently commented out with double slashes (//). These commented-out sections include configurations for route table associations, security groups, and the EKS provider module.

```

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
}

//module "sgs" {
//    source = ".../sg_eks"
//    vpc_id      = aws_vpc.Nam-vpc.id
//}

//module "eks" {
//    source = ".../eks"
//    vpc_id      = aws_vpc.Nam-vpc.id
//    subnet_ids = [aws_subnet.Nam-public-subnet-01.id,aws_subnet.Nam-public-subnet-02.id]
//    sg_ids     = module.sgs.security_group_public
//}

```

```

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/workshop-devops2/Terraform/vpc (main)
$ terraform plan
module.eks.aws_iam_role_policy_attachment.AmazonEC2ContainerRegistryReadOnly: Refreshing state... [id=ed-eks-worker-20230930170607517600000004]
module.eks.aws_iam_role_policy_attachment.autoscaler: Refreshing state... [id=ed-eks-worker-20230930170607556000000006]
module.eks.aws_iam_role_policy_attachment.s3: Refreshing state... [id=ed-eks-worker-2023093017060817950000000a]
module.eks.aws_iam_instance_profile_worker: Refreshing state... [id=ed-eks-worker-2023093017060817950000000b]

```

```

}

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@NAMRATAKFTVTC3 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
0230930170607805100000008]
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
93017060720420000002]
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@NAMRATAKFTVTC3 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
```

```

ubuntu@ip-10-1-1-59:~$ curl -LO https://dl.k8s.io/release/v1.27.3/bin/linux/amd64/kubectl
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload   Total Spent  Left Speed
100  138  100  138    0      0  1066      0 --:--:-- --:--:-- --:--:-- 1069
100 46.9M  100 46.9M   0      0  73.6M      0 --:--:-- --:--:-- --:--:-- 111M
ubuntu@ip-10-1-1-59:~$ ls
jenkins  kubectl  maven.txt
ubuntu@ip-10-1-1-59:~$ chmod +x kubectl
ubuntu@ip-10-1-1-59:~$ mv kubectl /usr/local/bin
mv: cannot move 'kubectl' to '/usr/local/bin/kubectl': Permission denied
ubuntu@ip-10-1-1-59:~$ sudo mv kubectl /usr/local/bin/
ubuntu@ip-10-1-1-59:~$ kubectl --version
error: unknown flag: --version
See 'kubectl --help' for usage.
ubuntu@ip-10-1-1-59:~$ kubectl version
WARNING: This version information is deprecated and will be replaced with the output from kubectl version --short. Use --output=yaml|json to get the full version.
Client Version: version.Info{Major:"1", Minor:"27", GitVersion:"v1.27.3", GitCommit:"25b4e43193bcd
a6c7328a6d147b1fb73a33f1598", GitTreeState:"clean", BuildDate:"2023-06-14T09:53:42Z", GoVersion:"g
o1.20.5", Compiler:"gc", Platform:"linux/amd64"}
Kustomize Version: v5.0.1

```

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

```

```

root@ip-10-1-1-59:/home/ubuntu# aws
Command 'aws' not found, but can be installed with:
snap install aws-cli # version 1.15.58, or
apt install awscli # version 1.22.34-1
See 'snap info aws-cli' for additional versions.
root@ip-10-1-1-59:/home/ubuntu# curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
% Total    % Received % Xferd  Average Speed   Time   Time     Time  Current
          Dload  Upload   Total Spent  Left Speed
100 55.8M  100 55.8M    0      0  109M      0 --:--:-- --:--:-- --:--:-- 109M
root@ip-10-1-1-59:/home/ubuntu# ls
awscliv2.zip  jenkins  maven.txt
root@ip-10-1-1-59:/home/ubuntu# apt install unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 65 not upgraded.
Need to get 174 kB of archives.
After this operation, 385 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 unzip amd64 6.0-26ub
untu3.1 [174 kB]
Fetched 174 kB in 0s (7635 kB/s)
Selecting previously unselected package unzip.
(Reading database ... 96044 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-26ubuntu3.1_amd64.deb ...

```

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 awscli2.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]: AKIAQXVMWBLQZR2E30HG
AWS Secret Access Key [None]: PvqW8e45E8yM0yiQx+h1kU/LDMTfkdvHM014nMxs
Default region name [None]:
Default output format [None]: ^C
root@ip-10-1-1-59:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIAQXVMWBLQZR2E30HG
AWS Secret Access Key [None]: PvqW8e45E8yM0yiQx+h1kU/LDMTfkdvHM014nMxs
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]: AKIAQXVMWBLQZR2E30HG
AWS Secret Access Key [None]: PvqW8e45E8yM0yiQx+h1kU/LDMTfkdvHM014nMxs
Default region name [None]: us-east-1
Default output format [None]:
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 awscli2.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 secret.yml
error: the path "secret.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-service" 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$ kubectl get all -n valaxy
NAME                      READY   STATUS    RESTARTS   AGE
pod/valaxy-rtp-5d5b7d84c4-4dkvgv  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

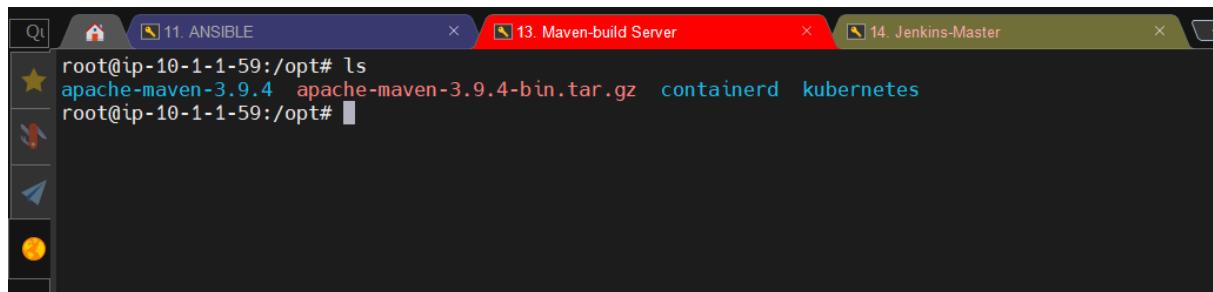
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/valaxy-rtp  2/2     2           2          28s

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

```
=====
```

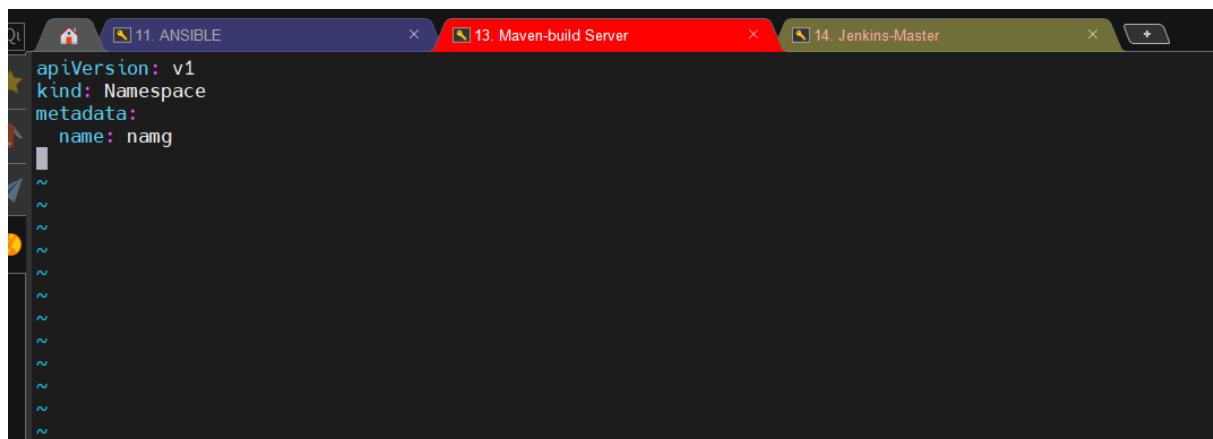
Deploying application on kubernetes

Create directories called kubernetes



```
root@ip-10-1-1-59:/opt# ls
apache-maven-3.9.4  apache-maven-3.9.4-bin.tar.gz  containerd  kubernetes
root@ip-10-1-1-59:/opt#
```

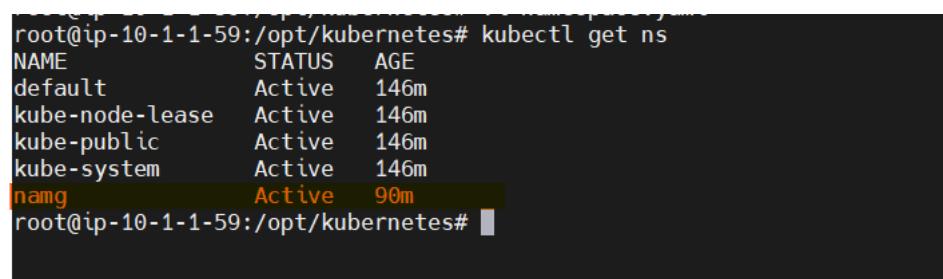
Write manifest files under kubernetes: have written manifest file to create Namespace



```
apiVersion: v1
kind: Namespace
metadata:
  name: namg
```

Run command to create namespace

```
'kubectl apply -f namespace.yml'  
'kubectl get ns' will give all namespaces available in kubernetes
```



```
root@ip-10-1-1-59:/opt/kubernetes# kubectl get ns
NAME      STATUS  AGE
default   Active  146m
kube-node-lease  Active  146m
kube-public  Active  146m
kube-system  Active  146m
namg      Active  90m
root@ip-10-1-1-59:/opt/kubernetes#
```

Write deployment.yaml manifest file

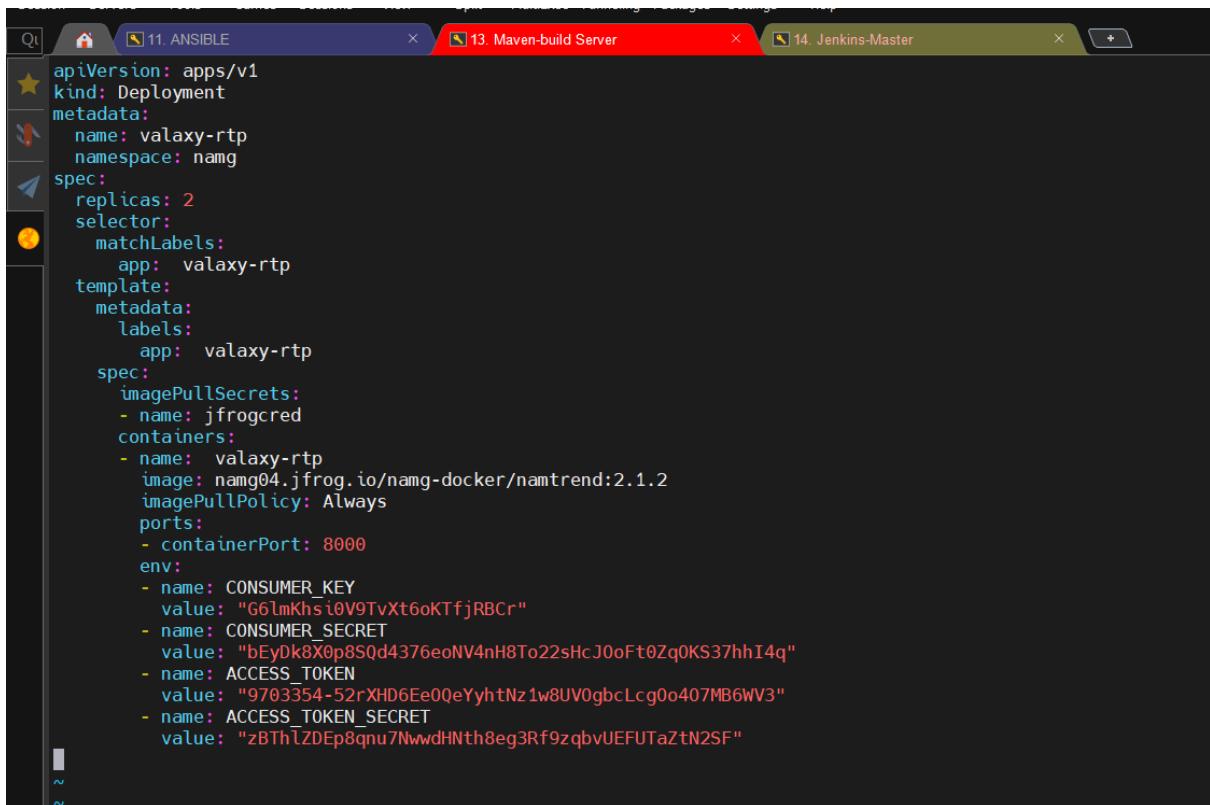
NOTE:

why did configure the environment variables in the manifest file?
Consumer_key, consumer_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.



The screenshot shows a browser window with three tabs open: '11. ANSIBLE', '13. Maven-build Server', and '14. Jenkins-Master'. The active tab, '14. Jenkins-Master', displays a YAML configuration for a Kubernetes Deployment. The deployment is named 'valaxy-rtp' and is defined in the 'namg' namespace. It specifies two replicas, uses a selector matching 'app: valaxy-rtp', and includes a template with metadata and labels. The 'spec' section contains an 'imagePullSecrets' block with a single secret named 'jfrogcred'. It also defines two containers, both named 'valaxy-rtp', using an image from 'namg04.jfrog.io/namg-docker/namtrend:2.1.2'. The 'imagePullPolicy' is set to 'Always'. Each container is exposed on port 8000. The deployment's environment variables include 'CONSUMER_KEY', 'CONSUMER_SECRET', 'ACCESS_TOKEN', and 'ACCESS_TOKEN_SECRET', each with a long, randomly generated value.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: valaxy-rtp
  namespace: namg
spec:
  replicas: 2
  selector:
    matchLabels:
      app: valaxy-rtp
  template:
    metadata:
      labels:
        app: valaxy-rtp
    spec:
      imagePullSecrets:
      - name: jfrogcred
      containers:
      - name: valaxy-rtp
        image: namg04.jfrog.io/namg-docker/namtrend:2.1.2
        imagePullPolicy: Always
        ports:
        - containerPort: 8000
      env:
      - name: CONSUMER_KEY
        value: "G61mKhs10V9TvXt6oKTfjRBCr"
      - name: CONSUMER_SECRET
        value: "bEydk8X0p8Sq0d4376eoNV4nH8To22sHcJ0oFt0Zq0KS37hhI4q"
      - name: ACCESS_TOKEN
        value: "9703354-52rXHD6Ee00eYyhtNz1w8UV0gbcLcg0o407MB6WV3"
      - name: ACCESS_TOKEN_SECRET
        value: "zBThlZDEp8qnu7NwdHNth8eg3Rf9zqbvUEFUTaZtN2SF"
```

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

```
root@ip-10-1-1-59:/opt/kubernetes# kubectl get all -n namg
NAME                               READY   STATUS            RESTARTS   AGE
pod/valaxy-rtp-cb8f5f648-nzjs2   0/1    ImagePullBackOff   0          14m
pod/valaxy-rtp-cb8f5f648-xm9kp   0/1    ImagePullBackOff   0          14m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/valaxy-rtp     0/2     2           0           14m

NAME                           DESIRED  CURRENT  READY   AGE
replicaset.apps/valaxy-rtp-cb8f5f648  2        2         0       14m
root@ip-10-1-1-59:/opt/kubernetes#
```

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:   namg04.jfrog.io/namg-docker/namtrend:2.1.2
    Image:          namg04.jfrog.io/namg-docker/namtrend:2.1.2
    Image ID:       <none>
    Port:           8000/TCP
    Host Port:     0/TCP
    State:          Waiting
      Reason:       ImagePullBackOff
    Ready:          False
    Restart Count:  0
    Environment:
      CONSUMER_KEY:      G6lmKhsioV9TvxT6oKTfjRBCr
      CONSUMER_SECRET:   bEyDk8X0p8S0d4376eoVN4nH8To22sHcJ0oFt0Zq0KS37hhI4q
      ACCESS_TOKEN:      9703354-52rXHD6Ee0QeYhntNz1w8UV0gbcLcg0o407MB6WV3
      ACCESS_TOKEN_SECRET: zBThlZDEp8gnu7NwvdHNth8eg3Rf9zqbvUEFUtaZtN2SF
    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
  PodsScheduled  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 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   30m5s (x6 over 18m)  kubelet         Back-off pulling image "namg04.jfrog.io/namg-docker/namtrend:2.1.2"
  normal 0/1 [0/1]  10m  kubelet        Normal  PodScheduled  
```

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': jfrogred
    'email address': logintoaws@gmail.com
    'password': <password>
```

```
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
```

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

`Note:` For more refer to [Kubernetes documentation](https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry). 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

The screenshot shows the JFrog Artifactory web interface. On the left, there's a sidebar with navigation links like 'Get Started', 'Artifactory', 'Builds', and 'Artifacts'. The 'Artifacts' link is highlighted. The main area displays a list of repositories under 'namg-docker'. A modal window is open on the right, titled 'New User'. It contains fields for 'Name' (namratakumari043@gmail.com), 'Email' (namratakumari043@gmail.com), and a 'Password' field. Below these fields are several buttons: 'Quick Repository Creation', 'Set Me Up', 'New Local Repository', 'New Remote Repository', 'New Virtual Repository', 'New Group', 'New Permission', 'Switch to Classic UI', 'Edit Profile', and 'Logout'. The 'New User' button is highlighted with a green background.

Add new user

[Create user](#) [Invite user](#)

User Settings

* User Name

* Email Address

Roles

Administer Platform

Manage Resources ?

Manage Policies

Read Policies

Manage Watches

Manage Reports ?

[Cancel](#)

Add new user

Password

* Password

* Retype Password

Multi-factor Authentication Status

[Reset Google MFA Enrollment](#) ?

Related Groups

0 Available Group

1 Selected Group

[Cancel](#) [Reset](#) [Save](#)

```

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X Server Exit
QL 18. Jenkins-Master 19. Maven-Server 20. ANSIBLE +
Reco SH-b
NAME READY STATUS RESTARTS AGE
pod/valaxy-rtp-5d5b7d84c4-hp7gm 0/1 ImagePullBackOff 0 18s
pod/valaxy-rtp-5d5b7d84c4-hrwhg 0/1 ImagePullBackOff 0 18s
NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/valaxy-rtp 0/2 2 0 18s
NAME DESIRED CURRENT READY AGE
replicaset.apps/valaxy-rtp-5d5b7d84c4 2 2 0 18s
root@ip-10-1-1-81:/opt/kubernetes# docker login https://sanam01.jfrog.io
Username: sunny.ime05@gmail.com
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-10-1-1-81:/opt/kubernetes#

```

save password in notepad

```

Login Succeeded
root@ip-10-1-1-81:/opt/kubernetes# cat at /root/.docker/config.json | base64 -w0
cat: at: No such file or directory
ewoJImF1dGhzIjogewoJCSJzYW5hbTAxLmpmcn9nLmlvIjogewoJCQkiYXV0aCI6ICJjM1Z1Ym5rdWFXMWxNRFZB
root@ip-10-1-1-81:/opt/kubernetes#

```

write secret.yml file

```

root@ip-10-1-1-81:/opt/kubernetes# cat at /root/.docker/config.json | base64 -w0
cat: at: No such file or directory
ewoJImF1dGhzIjogewoJCSJzYW5hbTAxLmpmcn9nLmlvIjogewoJCQkiYXV0aCI6ICJjM1Z1Ym5rdWFXMWxNRFZB
root@ip-10-1-1-81:/opt/kubernetes#
root@ip-10-1-1-81:/opt/kubernetes# vi secret.yml
root@ip-10-1-1-81:/opt/kubernetes# cat secret.yml
apiVersion: v1
kind: Secret
metadata:
  name: jfrogcred
  namespace: valaxy
data:
  .dockerconfigjson: ewoJImF1dGhzIjogewoJCSJzYW5hbTAxLmpmcn9nLmlvIjogewoJCQkiYXV0aCI6ICJjM1Z1Ym5rdWFXMWxNRFZB
type: kubernetes.io/dockerconfigjson

root@ip-10-1-1-81:/opt/kubernetes#

```

```

root@ip-10-1-1-81:/opt/kubernetes#
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

Auto-assigned IP address
🔗 3.87.3.14 [Public IP]

VPC ID
🔗 vpc-0e9aec73f7fe4e869 (Nam-vpc) 🔗

IAM Role
🔗 ed-eks-worker 🔗

Subnet ID
🔗 subnet-0d4b4207a66ba9d1d (Nam-public-subent-02) 🔗

IMDSv2
Optional

Details **Security** **Networking** **Storage** **Status checks** **Monitoring** **Tags**

▼ Security details

IAM Role
🔗 ed-eks-worker 🔗

Owner ID
🔗 050828020449

Security groups

🔗 sg-0e0db0bf97708251c (eks-cluster-sg-namg-eks-01-1975494273)
🔗 sg-0c85c0e853c1f9ae4 (eks-remoteAccess-aec577d0-e54e-6a4d-681d-d131c0a3c8e9)

▼ Inbound rules

| Inbound rules | | | | | | Outbound rules | Tags | |
|--|------|------------------------|------------|-------------|----------|---|--|---|
| Inbound rules (2) | | | | | | 🔗 | Manage tags | 🔗 Edit inbound rules |
| <input type="text"/> Filter security group rules | | | | | | 🔗 < 1 > 🔗 | | |
| | Name | Security group rule... | IP version | Type | Protocol | Port range | | |
| <input type="checkbox"/> | - | Sgr-0764332919c436... | IPv4 | Custom TCP | TCP | 30082 | 🔗 | |
| <input type="checkbox"/> | - | sgr-0d73ce3548a96eeba | - | All traffic | All | All | 🔗 | |

🔗 ← → 🔗 ⌂ 🔗 Not secure | http://3.87.3.14:30082

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#
```

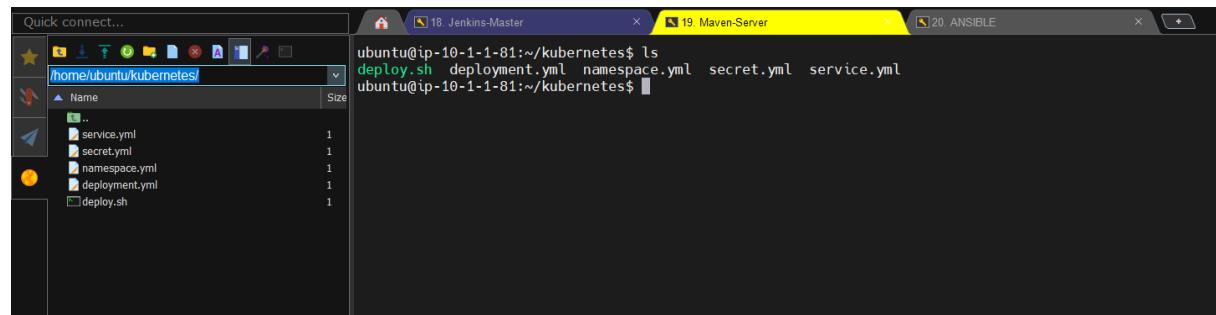
switch to ubuntu user

```
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 refused
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 refused
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 refused
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 refused
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 refused
sed
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

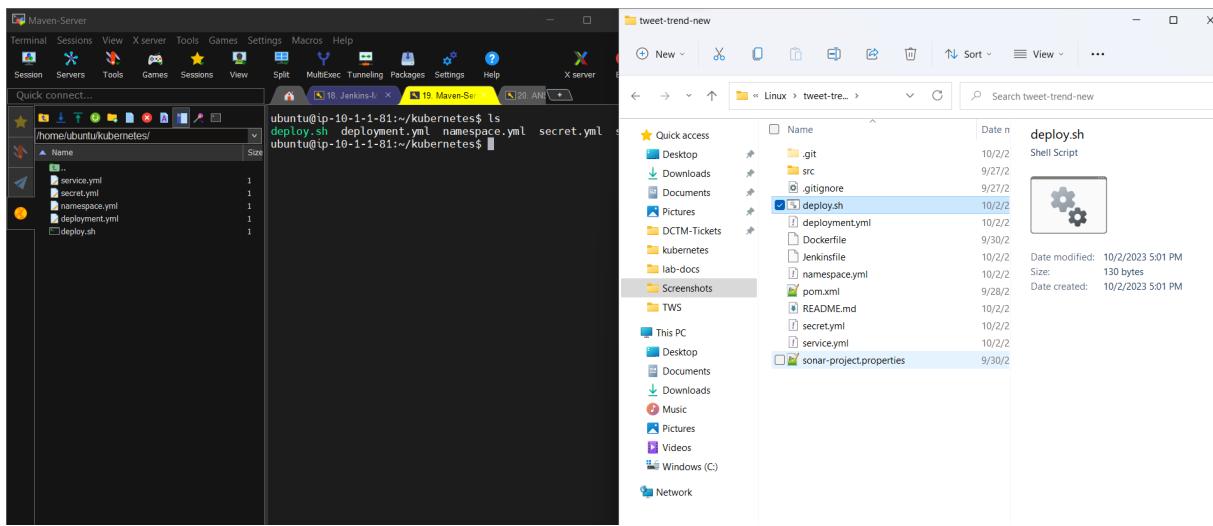
so this command doesn't work because we don't have the cluster credentials.

=====

Commit manifests in GitHub



Drag manifests files from Mobaxterm to Project source code folder



Commit the code into Github

```

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git fetch

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add --chmod=+x deploy.sh
fatal: pathspec 'deploy.sh' did not match any files

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add --chmod=+x deploy.sh

namratak@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "addedd 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@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ |

```

Committed in to Github

This branch is 35 commits ahead of ravdy:main.

[Contribute](#) [Sync fork](#)

| | | |
|--|-----------------------|--------------|
|  Namg04 addedd kubernetes manifests file | aa01a06 2 minutes ago | ⌚ 36 commits |
| 📁 src modified src return value | 2 days ago | |
| 📄 .gitignore initial commit | 3 months ago | |
| 📄 Dockerfile added Dockerfile and docker build and publish stages in Jenkins file | 2 days ago | |
| 📄 Jenkinsfile updated docker build and publish stage in jenkins file | 5 hours ago | |
| 📄 README.md Update README.md | 17 hours ago | |
| 📄 deploy.sh addded kubernetes manifests file | 2 minutes ago | |
| 📄 deployment.yml addded kubernetes manifests file | 2 minutes ago | |
| 📄 namespace.yml addded kubernetes manifests file | 2 minutes ago | |
| 📄 pom.xml Revert "Update pom.xml" | 4 days ago | |
| 📄 secret.yml addded kubernetes manifests file | 2 minutes ago | |
| 📄 service.yml addded kubernetes manifests file | 2 minutes ago | |
| 📄 sonar-project.properties added docker build and publish stages in Jenkins file | 2 days ago | |

Build has triggered

[Status](#)

[Changes](#)

[Build Now](#)

[View Configuration](#)

[Full Stage View](#)

[SonarQube](#)

[Pipeline Syntax](#)

[Build History](#) trend ▾

Filter builds... /

#15 Oct 2, 2023, 11:37 AM ✓ 1 commit

#14 Oct 2, 2023, 7:05 AM ✓ No Changes

#13 Oct 2, 2023, 6:59 AM ✘ No Changes

Atom feed for all Atom feed for failures

Pipeline main

Full project name: Nam-trend-multibranch/main

Stage View

Average stage times: (Average full run time: ~2min 32s)

| Declarative: Checkout SCM | build | test | SonarQube analysis | Quality Gate | Jar Publish | Docker Build | Docker Publish |
|---------------------------|-------|------|--------------------|--------------|-------------|--------------|----------------|
| 1s | 13s | 27s | 1min 11s | 580ms | 4s | 9s | 19s |
| 642ms | 12s | 29s | 1min 48s | | | | |
| 548ms | 12s | 27s | 52s | 550ms | 1s | 16s | 39s |
| 3s | 14s | 27s | 53s | 610ms | 6s | 2s failed | 61ms failed |

SonarQube Quality Gate

Deploying app using Jenkins through deploy.sh

Dashboard > Nam-trend-multibranch-pipeline > main >

Pipeline main

Full project name: Nam-trend-multibranch/main

Changes

Build Now

View Configuration

Full Stage View

SonarQube

Pipeline Syntax

Build History trend ▾

Filter builds...

- #17 Oct 2, 2023, 12:00 PM
- ✗ #16 Oct 2, 2023, 11:55 AM
- #14 Oct 2, 2023, 7:05 AM

Average stage times: (Average full run time: ~2min 10s)

| Declarative: Checkout SCM | build | test | SonarQube analysis | Quality Gate | Jar Publish | Docker Build | Docker Publish | Deploy |
|-----------------------------|-------|------|--------------------|--------------|-------------|--------------|----------------|-------------|
| 1s | 13s | 27s | 52s | 650ms | 3s | 6s | 14s | 3s |
| Oct 02 17:30 No Changes | 541ms | 13s | 26s | 52s | 904ms | 2s | 2s | 6s |
| #16 Oct 02 17:25 2 commits | 3s | 14s | 27s | 52s | 498ms | 5s | 1s failed | 45ms failed |
| #14 Oct 02 12:35 No Changes | 548ms | 12s | 27s | 52s | 550ms | 1s | 16s | 39s |

Atom feed for all Atom feed for failures

Write deploy stage in Jenkinsfile

```

File Edit Selection View Go Run Terminal Help ← → 🔍 tweet-trend-new

EXPLORER
  ✓ TWEET-TREND-NEW
    > src
    ⚡ .gitignore
    $ deploy.sh
    ! deployment.yml
    Dockerfile
    Jenkinsfile
    ! namespace.yml
    pom.xml
    README.md
    ! secret.yml
    ! service.yml
    sonar-project.propert...
    sonar-project.properties

Jenkinsfile X
  Jenkinsfile
  97
  98
  99 }
  100
  101 stage ("Deploy"){
  102   steps {
  103     script {
  104       sh './deploy.sh'
  105     }
  106   }
  107 }
  108
  109 }
  110 }
  111

```

Commit the file in Github

```

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add Jenkinsfile

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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)

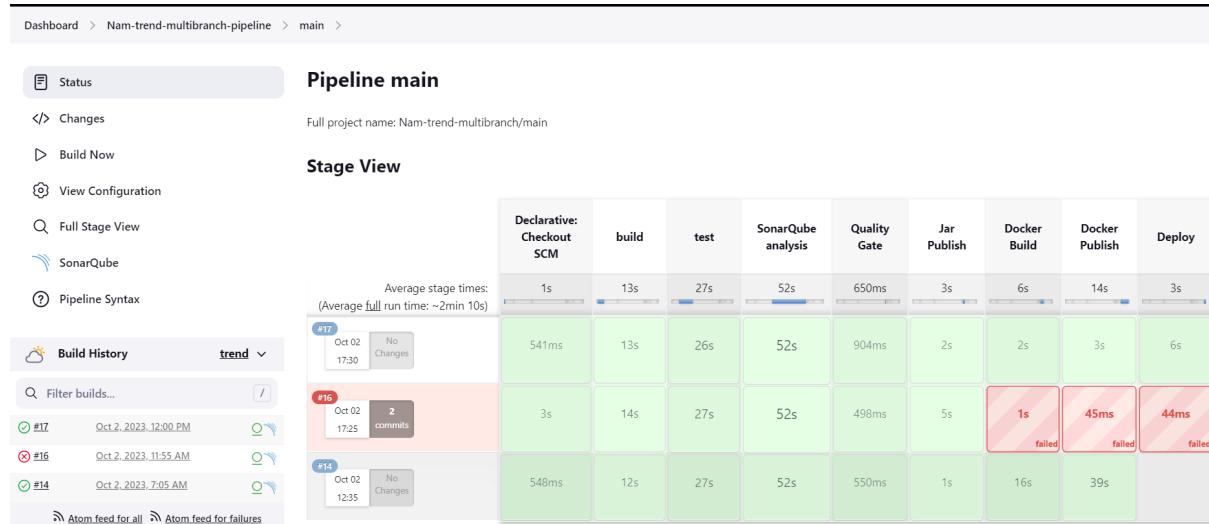
nothing to commit, working tree clean

namratak@NAMRATAKFTVTC3 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@NAMRATAKFTVTC3 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-east-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 chart
```sh
helm install mysqlDb mysql-1.6.9.tgz
```

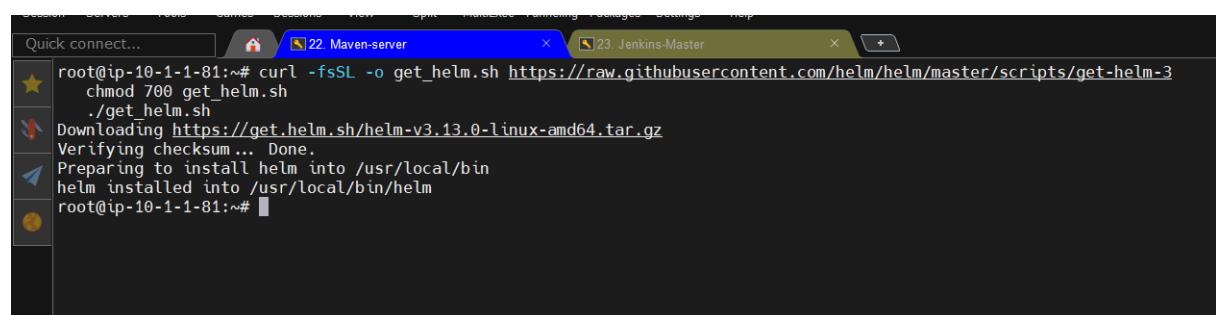
Above command deploy MySQL
To check deployment
```sh
helm list
```

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



```

root@ip-10-1-1-81:~# curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
root@ip-10-1-1-81:~# chmod 700 get_helm.sh
root@ip-10-1-1-81:~# ./get_helm.sh
root@ip-10-1-1-81:~# helm version
version.BuildInfo{Version:"v3.13.0", GitCommit:"825e86f6a7a38cef1112bfa66e4127a706749b1", GitTreeState:"clean", GoVersion:"go1.20.8"}
root@ip-10-1-1-81:~# helm repo list
Error: no repositories to show
root@ip-10-1-1-81:~# helm list
Error: Kubernetes cluster unreachable: Get "https://ED9144A42E31439276F6570C1E660C0E.gr7.us-east-1.eks.amazonaws.com/version": dial tcp: lookup ED9144A42E31439276F6570C1E660C0E.gr7.us-east-1.eks.amazonaws.com on 127.0.0.53:53: no such host
root@ip-10-1-1-81:~# helm list
Error: Kubernetes cluster unreachable: Get "https://ED9144A42E31439276F6570C1E660C0E.gr7.us-east-1.eks.amazonaws.com on 127.0.0.53:53: no such host
root@ip-10-1-1-81:~# aws eks update-kubeconfig --region us-east-1 --name namg-eks-01
Could not connect to the endpoint URL: "https://eks.us-east-1.amazonaws.com/clusters/namg-eks-01"
root@ip-10-1-1-81:~# C
root@ip-10-1-1-81:~# aws eks update-kubeconfig --region us-east-1 --name namg-eks-01
Updated context arn:aws:eks:us-east-1:050828020449:cluster/namg-eks-01 in /root/.kube/config
root@ip-10-1-1-81:~# helm list
NAME      NAMESPACE      REVISION      UPDATED STATUS      CHART      APP VERSION
root@ip-10-1-1-81:~#

```

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:~#
```

| 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:~# 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 | | |

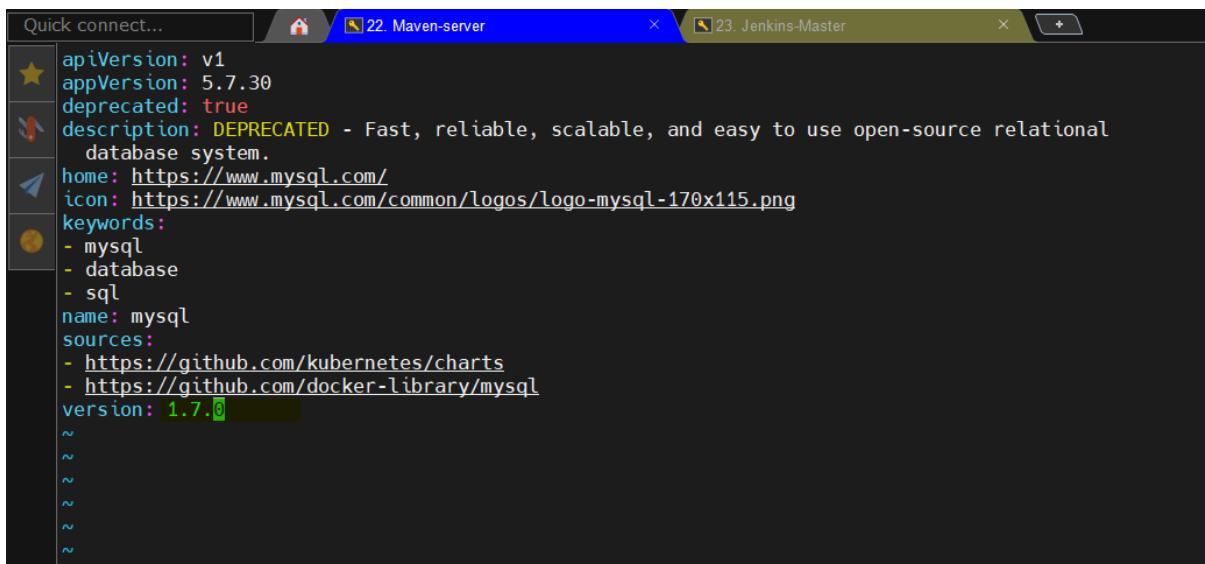
Pull helm charts

```

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.yaml



```

apiVersion: v1
appVersion: 5.7.30
deprecated: true
description: DEPRECATED - Fast, reliable, scalable, and easy to use open-source relational
  database system.
home: https://www.mysql.com/
icon: https://www.mysql.com/common/logos/logo-mysql-170x115.png
keywords:
  - mysql
  - database
  - sql
name: mysql
sources:
  - https://github.com/kubernetes/charts
  - https://github.com/docker-library/mysql
version: 1.7.0
~
~
~
~
~
~
```

Modified values.yaml

```

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-fbqln  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
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.yaml  namespace.yaml  secret.yaml  service.yaml
ubuntu@ip-10-1-1-81:~/kubernetes$ mv deployment.yaml namespace.yaml service.yaml secret.yaml .. /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.yaml  namespace.yaml  secret.yaml  service.yaml
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.yaml  namespace.yaml  secret.yaml  service.yaml
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      STATUS      CHART          APP VERSION
TATUS    CHART        APP VERSION
namtrend-v1  default     1           2023-10-03 05:21:12.766719077 +0000 UTC deployed
deployed 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-v1  default     1           2023-10-03 05:21:12.766719077 +0000 UTC deployed
ubuntu@ip-10-1-1-81:~$ ls
aws awscli2.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

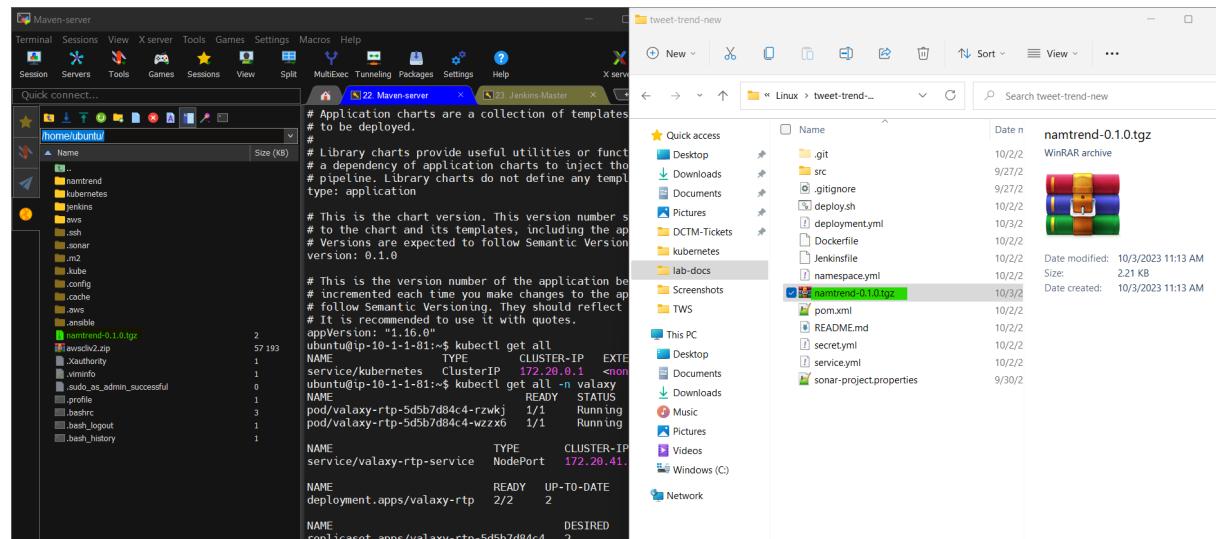
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/valaxy-rtp-service   NodePort   172.20.41.10   <none>        8000:30082/TCP   4m34s

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

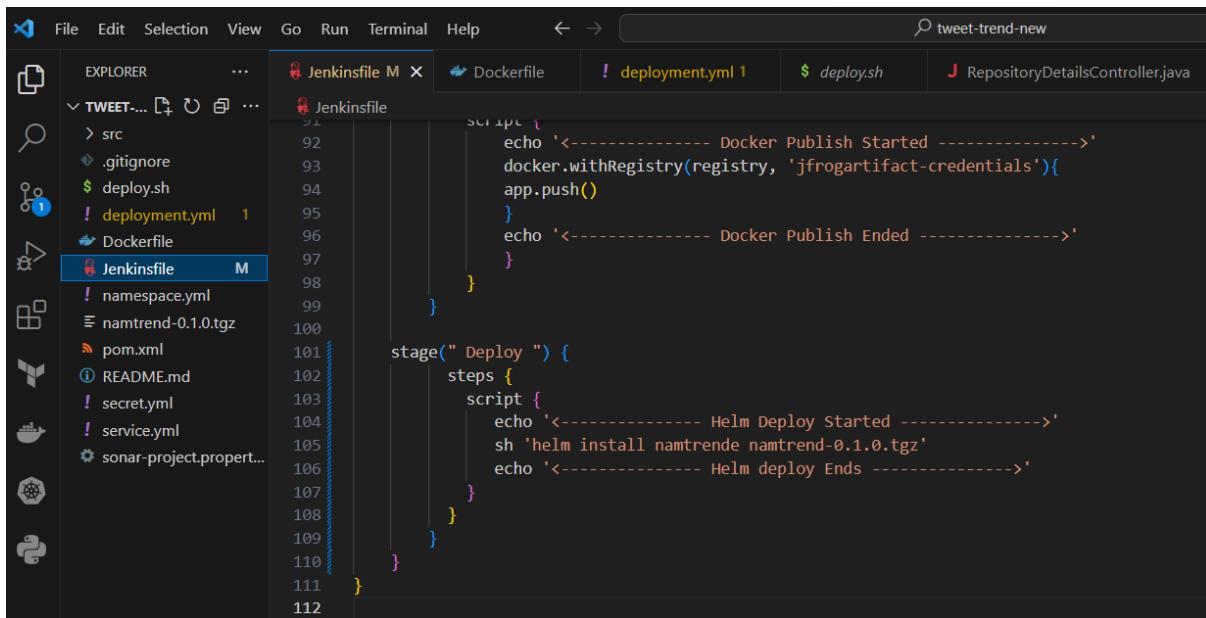
NAME                           DESIRED  CURRENT   READY   AGE
replicaset.apps/valaxy-rtp-5d5b7d84c4  2        2         2       4m34s
ubuntu@ip-10-1-1-81:~$ helm uninstall namtrend-v1
release "namtrend-v1" uninstalled
ubuntu@ip-10-1-1-81:~$ kubectl get all -n valaxy
No resources found in valaxy namespace.
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



```
EXPLORER      ...  Jenkinsfile M X  Dockerfile  deployment.yml 1  deploy.sh  RepositoryDetailsController.java
TWEET...  src .gitignore $ deploy.sh ! deployment.yml 1 Dockerfile Jenkinsfile M namespace.yml namtrend-0.1.0.tgz pom.xml README.md secret.yml service.yml sonar-project.properties

Jenkinsfile
script {
    echo '<----- Docker Publish Started ----->'
    docker.withRegistry(registry, 'jfrogartifact-credentials'){
        app.push()
    }
    echo '<----- Docker Publish Ended ----->'
}

stage(" Deploy ") {
    steps {
        script {
            echo '<----- Helm Deploy Started ----->'
            sh 'helm install namtrende namtrend-0.1.0.tgz'
            echo '<----- Helm deploy Ends ----->'
        }
    }
}
```

Commit the code into Github

```
namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ ls
Dockerfile Jenkinsfile README.md deploy.sh deployment.yml namespace.yml namtrend-0.1.0.tgz pom.xml secret.yml service.yml sonar-project.properties src/
namratak@NAMRATAKFTVTC53 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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    namtrend-0.1.0.tgz

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

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ git fetch
namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ git pull
Already up to date.

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ git add namtrend-0.1.0.tgz
namratak@NAMRATAKFTVTC53 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:   namtrend-0.1.0.tgz

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

namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
$ git commit -m "added helm deployment stage into Jenkins file"
[main a68d951] added helm deployment stage into Jenkins file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 namtrend-0.1.0.tgz
namratak@NAMRATAKFTVTC53 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), 2.51 KiB | 2.51 MiB/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
   016d898..a68d951  main -> main
namratak@NAMRATAKFTVTC53 MINGW64 ~/Linux/tweet-trend-new (main)
```

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 scrapes 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.

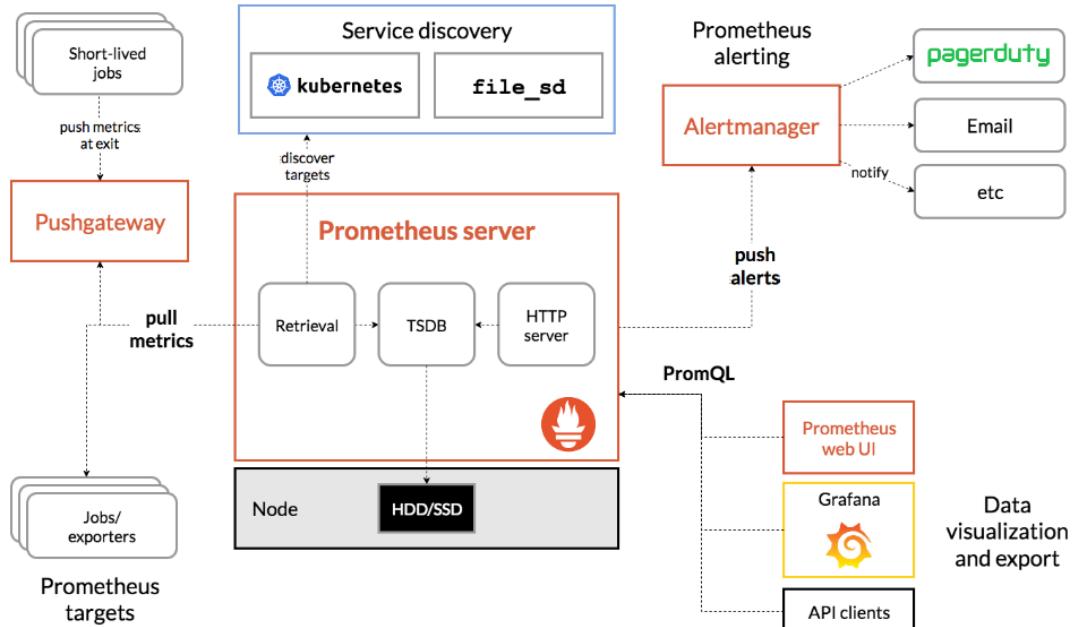
So these two will be in the same server.

Prometheus by default runs on port number 9090 and alert manager 9093.

Prometheus architecture.

Architecture

This diagram illustrates the architecture of Prometheus and some of its ecosystem components:



First let's understand about the 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 balancer
```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

The screenshot shows a terminal window with three tabs at the top: 22. Maven-server, 23. Jenkins-Master, and 24. ANSIBLE. The main area displays the following command outputs:

```

ubuntu@ip-10-1-1-81:~$ kubectl get all -n valaxy
NAME                           READY   STATUS    RESTARTS   AGE
pod/valaxy-rtp-6dd659d8c5-66xpt   1/1     Running   0          15m
pod/valaxy-rtp-6dd659d8c5-pmhjb   1/1     Running   0          15m

NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)       AGE
service/valaxy-rtp-service   NodePort   172.20.12.103   <none>        8000:30082/TCP   15m

NAME              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/valaxy-rtp   2/2     2           2           15m

NAME              DESIRED   CURRENT   READY   AGE
replicaset.apps/valaxy-rtp-6dd659d8c5   2         2         2         15m

ubuntu@ip-10-1-1-81:~$ helm version
version.BuildInfo{Version:"v3.13.0", GitCommit:"825e86f6a7a38cef1112bfa606e4
127a706749b1", GitTreeState:"clean", GoVersion:"go1.20.8"}
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      3h46m
ubuntu@ip-10-1-1-81:~$ kubectl get nodes
NAME                  STATUS   ROLES   AGE   VERSION
ip-10-1-1-202.ec2.internal   Ready   <none>   3h41m   v1.27.5-eks-43840fb
ip-10-1-2-177.ec2.internal   Ready   <none>   3h41m   v1.27.5-eks-43840fb
ubuntu@ip-10-1-1-81:~$ 
ubuntu@ip-10-1-1-81:~$ kubectl create namespace monitoring
namespace/monitoring created
ubuntu@ip-10-1-1-81:~$ kubectl get ns
NAME      STATUS   AGE
default   Active   3h47m
kube-node-lease   Active   3h47m
kube-public   Active   3h47m
kube-system   Active   3h47m
monitoring   Active   8s
valaxy     Active   54m
ubuntu@ip-10-1-1-81:~$ 

```

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://prometheus-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 prometheus-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 using
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                           AGE        READY   STATUS    RESTART
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0   9m       2/2     Running   0
pod/prometheus-grafana-865656ffbb-cfqnh   9m8s    3/3     Running   0
pod/prometheus-kube-prometheus-operator-7c4994c9f9-c65nx   9m8s    1/1     Running   0
pod/prometheus-kube-state-metrics-84f546f65-l98rc   9m8s    1/1     Running   0
pod/prometheus-prometheus-kube-prometheus-prometheus-0   8m59s   2/2     Running   0
pod/prometheus-prometheus-node-exporter-jw8x9   9m9s    1/1     Running   0
pod/prometheus-prometheus-node-exporter-prhbk   9m8s    1/1     Running   0

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>
5090/TCP                      9m
service/prometheus-prometheus-node-exporter   ClusterIP  172.20.92.89 <none>
9160/TCP                      9m9s

NAME          DESIRED   CURRENT   READY   UP-TO
-DATE AVAILABLE NODE SELECTOR AGE
daemonset.apps/prometheus-prometheus-node-exporter   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
READY   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
RT(S) | AGE | TYPE | CLUSTER-IP | EXTERNAL-IP |
|--|--------------|---------------|---|-------------|
| service/alertmanager-operated | 25m | ClusterIP | <none> | |
| 93/TCP, 9094/TCP, 9094/UDP | 25m | | | |
| service/prometheus-grafana | 25m | ClusterIP | 172.20.188.1 | <none> |
| /TCP | 25m | | | |
| service/prometheus-kube-prometheus-alertmanager | 25m | ClusterIP | 172.20.189.230 | <none> |
| 93/TCP, 8080/TCP | 25m | | | |
| service/prometheus-kube-prometheus-operator | 25m | 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 | 25m | ClusterIP | 172.20.197.8 | <none> |
| 80/TCP | 25m | | | |
| service/prometheus-operated | 25m | ClusterIP | None | <none> |
| 90/TCP | 25m | | | |
| service/prometheus-prometheus-node-exporter | 25m | ClusterIP | 172.20.92.89 | <none> |
| 00/TCP | 25m | | | |
| NAME | DESIRED | CURRENT | READY | UP-TO-DATE |
| daemonset.apps/prometheus-prometheus-node-exporter | 2 | 2 | 2 | 2 |
| 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 | | |

Log into Prometheus dashboard to monitor application- https://ELB:9090

We've redesigned the Classic Load Balancer console to make it easier to use. By default, we'll always bring you to the new Classic Load Balancer experience. You can access the old console experience; however, it will be deprecated soon.

EC2 > Load balancers > a117bfeb3f94946acbd649cdc65798d0

a117bfeb3f94946acbd649cdc65798d0

Details

| | | | |
|-------------------------------|---------------------------------------|--|--|
| Load balancer type
Classic | Status
2 of 2 instances in service | VPC
vpc-0e9aec73f7fe4e869 | Date created
October 3, 2023, 13:05 (UTC+05:30) |
| Scheme
Internet-facing | Hosted zone
Z355XDOTRQ7X7K | Availability Zones
subnet-06c38dcc1479b42e5 us-east-1a (use1-az1)
subnet-0d4b4207a66ba9d1d us-east-1b (use1-az2) | |

DNS name copied

a117bfeb3f94946acbd649cdc65798d0-1263591558.us-east-1.elb.amazonaws.com (A Record)

Not secure | http://a117bfeb3f94946acbd649cdc65798d0-1263591558.us-east-1.elb.amazonaws.com:9090/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show_exemplars... Update

Prometheus Alerts Graph Status Help

Use local time Enable query history Enable autocomplete Enable highlighting Enable linter

Expression (press Shift+Enter for newlines)

Graph

Evaluation time

No data queried yet

Add Panel Remove Panel

Prometheus Alerts Graph Status Help

Use local time Enable query history Enable autocomplete Enable highlighting Enable linter

machine_cpu_cores

Table Graph

Evaluation time

```
machine_cpu_cores{boot_id="44363278-92d1-4c75-9d48-96394d0782bf", endpoint="https-metrics", instance="10.1.2.177:10250", job="kubellet", machine_id="f58b1857223d4d46af4d4f17c98ddf18", metrics_path="/metrics/cadvisor", namespace="kubernetes", node="ip-10-1-2-177.ec2.internal", service="prometheus-kube-prometheus-kubellet", system_uuid="ec2d0defa-209fc56-e3a-9a31f1d4c44f"}  
machine_cpu_cores{boot_id="cc375d72-b63d-4304-86c5-9f21fa0f7516", endpoint="https-metrics", instance="10.1.1.202:10250", job="kubellet", machine_id="c3e04a548ff340b5bf0320c10324bffd", metrics_path="/metrics/cadvisor", namespace="kubernetes", node="ip-10-1-1-202.ec2.internal", service="prometheus-kube-prometheus-kubellet", system_uuid="ec296cae-48ca-7cda-79db-9c59a7be3130")
```

Load time: 246ms Resolution: 14s Result series: 2

Add Panel Remove Panel

Alerts

The screenshot shows the Prometheus Alertmanager interface. At the top, there are filter buttons for 'Inactive (139)', 'Pending (0)', and 'Firing (3)'. A search bar with the placeholder 'Filter by name or labels' is also present. Below the filters, a file path is shown: '/etc/prometheus/rules/prometheus-kube-prometheus-prometheus-rulefiles-0/monitoring-prometheus-kube-prometheus-alertmanager.rules-ab55b288-54a9-499d-b3a9-7e02d1a8f4a6.yaml > alertmanager.rules'. The main area displays a list of active alerts under the heading 'AlertmanagerFailedReload (0 active)'. Other categories listed include 'AlertmanagerMembersInconsistent', 'AlertmanagerFailedToSendAlerts', 'AlertmanagerClusterFailedToSendAlerts', 'AlertmanagerConfigInconsistent', 'AlertmanagerClusterDown', and 'AlertmanagerClusterCrashlooping', all with 0 active alerts.

=====

Grafana

The screenshot shows the Grafana login page. The URL in the browser is 'http://ae5b8495d421648a481bc9e85b36f17f-501733424.us-east-1.elb.amazonaws.com/login'. The page features a dark background with a central light-colored login form. It includes a yellow gear logo, the text 'Welcome to Grafana', and input fields for 'Email or username' and 'Password'. A blue 'Log in' button is at the bottom of the form. Below the form, a link 'Forgot your password?' is visible. At the bottom of the page, there is footer text: 'Documentation | Support | Community | Open Source | v10.1.2 (8e428858dd) | New version available!'



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**

```

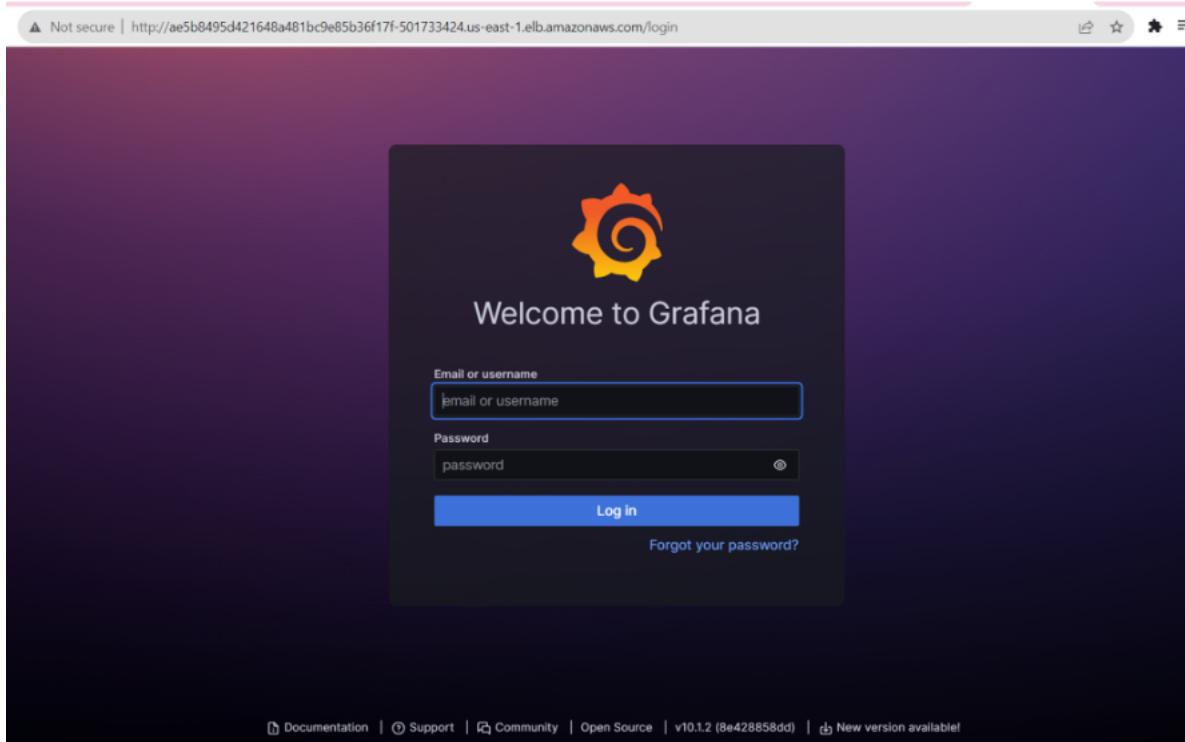
Please edit the object below. Lines beginning with a '#' will be ignored,
and an empty file will abort the edit. If an error occurs while saving this file will be
reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
 annotations:
 meta.helm.sh/release-name: prometheus
 meta.helm.sh/release-namespace: 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 --

```

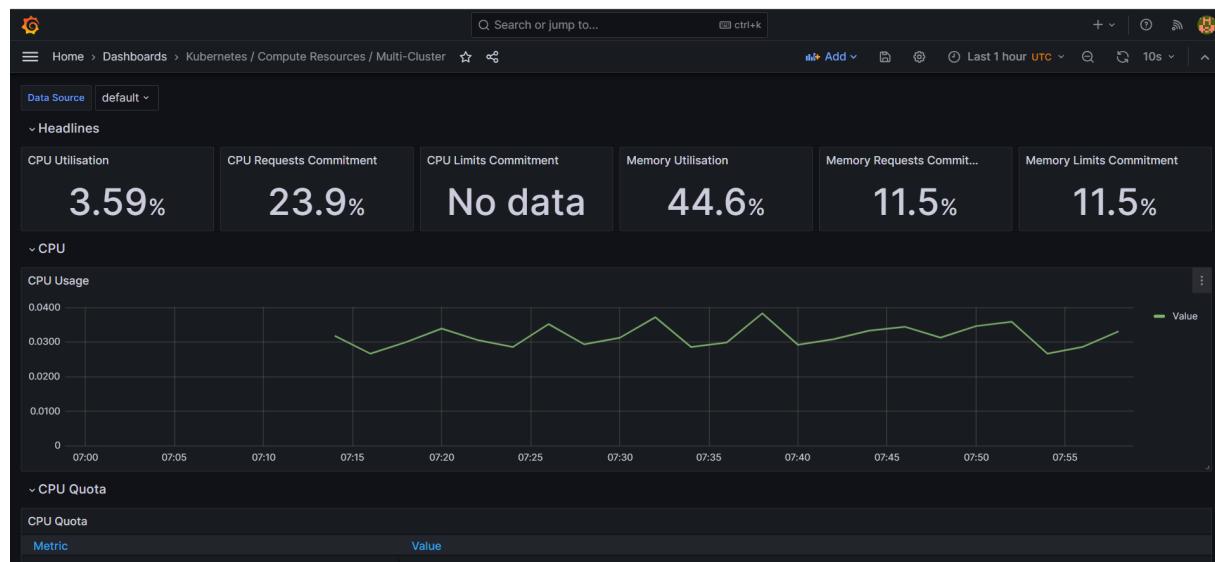
### Change type to Loadbalancer

| NAME<br>RT(S)                                   | AGE | TYPE         | CLUSTER-IP     | EXTERNAL-IP                                                             |
|-------------------------------------------------|-----|--------------|----------------|-------------------------------------------------------------------------|
| service/alertmanager-operated                   |     | ClusterIP    | None           | <none>                                                                  |
| 93/TCP,9694/TCP,9694/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,8880/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  | a117bfef3f94946acbd649cdc65798d0-1263591558.us-east-1.elb.amazonaws.com |
| 90:30607/TCP,8880:30491/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 |              |                |                                                                         |

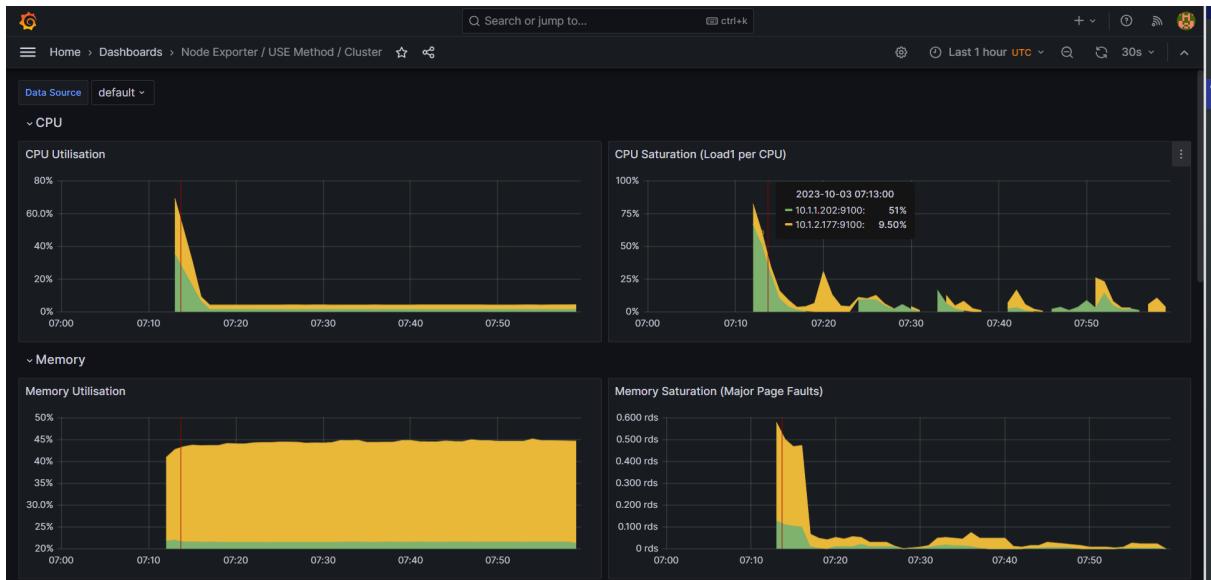
username: admin  
 password: prom-operator  
 Access on port :80



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

```

clusterIP: 172.20.188.1
clusterIPs:
- 172.20.188.1
externalTrafficPolicy: Cluster
internalTrafficPolicy: Cluster
ipFamilies:
- IPv4
ipFamilyPolicy: SingleStack
ports:
- name: http-web
nodePort: 30508
port: 80
protocol: TCP
targetPort: 3000
selector:
app.kubernetes.io/instance: prometheus
app.kubernetes.io/name: grafana
sessionAffinity: None
type: ClusterIP
status:
loadBalancer:
ingress:
- hostname: ae5b8495d421648a481bc9e85b36f17f-501733424.us-east-1.elb.amazonaws.com
-- INSERT --
38,18 Bot

```

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

```

See https://get.k8s.io for more examples
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-865656ffb-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> 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> 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-865656ffb 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-kube-prometheus-prometheus 1/1 52m
ubuntu@ip-10-1-1-81:~$

```

### Load Balancers have deleted from AWS

The screenshot shows the AWS EC2 Load balancers console. At the top, there's a breadcrumb navigation: EC2 > Load balancers. Below the header, a section titled "Load balancers" is displayed with the sub-instruction: "Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic." A search bar labeled "Filter by property or value" is present. To the right of the search bar are buttons for "Actions" and "Create load balancer". Below the search bar is a table header with columns: Name, DNS name, State, VPC ID, Availability Zones, Type, and Date c. Underneath the table, a message states: "No load balancers" and "You don't have any load balancers in us-east-1". A prominent orange "Create load balancer" button is centered below this message. At the bottom of the main content area, it says "0 load balancers selected" and "Select a load balancer above.".



🎉 Congrats on finishing the course!

[Find more courses](#)