POC- Terraform,Ansible,Docker,Jenkins

Step-1: Using Terraform Creating Infrastructure

- Lancing Instances
- Creating VPC
- Creating S3 Bucket

Now, Files That I have Used for Creating Infra is:

This files for Instance Creation and VPC:

backend.tf

```
terraform {
  backend "s3" {
    bucket         = "satwik-devops-terraform-state"
    key            = "platform/terraform.tfstate"
    region         = "us-east-2"

    dynamodb_table = "terraform-state-locks"
    encrypt        = true
  }
}
```

data.tf

```
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}
```

main.tf

```
module "network" {
  source              = "./modules/network"
  vpc_cidr            = var.vpc_cidr
  public_subnet_cidr = var.public_subnet_cidr
}

module "security" {
  source = "./modules/security"
  vpc_id = module.network.vpc_id
}

module "jenkins" {
  source            = "./modules/compute-jenkins"
  ami               = data.aws_ami.ubuntu.id
  instance_type     = "m7i-flex.large"
  subnet_id         = module.network.subnet_id
  security_group_id = module.security.jenkins_sg
  key_name          = var.key_name
  name              = "jenkins"
}

module "sonarqube" {
  source            = "./modules/compute-sonarqube"
  ami               = data.aws_ami.ubuntu.id
  instance_type     = "m7i-flex.large"
  subnet_id         = module.network.subnet_id
  security_group_id = module.security.sonarqube_sg
  key_name          = var.key_name
  name              = "sonarqube"
}

module "nexus" {
  source            = "./modules/compute-nexus"
  ami               = data.aws_ami.ubuntu.id
  instance_type     = "m7i-flex.large"
  subnet_id         = module.network.subnet_id
  security_group_id = module.security.nexus_sg
  key_name          = var.key_name
  name              = "nexus"
```

```
}

module "docker" {
  source             = "./modules/compute-docker"
  ami                = data.aws_ami.ubuntu.id
  instance_type      = "m7i-flex.large"
  subnet_id          = module.network.subnet_id
  security_group_id = module.security.docker_sg
  key_name           = var.key_name
  name               = "docker-host"
}
module "ansible" {
  source             = "./modules/compute-ansible"
  ami                = data.aws_ami.ubuntu.id
  instance_type      = "t3.micro"
  subnet_id          = module.network.subnet_id
  security_group_id = module.security.ansible_sg
  key_name           = var.key_name
  name               = "ansible-control"
}
```

output.tf

```
output "jenkins_ip" {
  description = "Public IP of Jenkins EC2 instance"
  value       = module.jenkins.public_ip
}

output "sonarqube_ip" {
  description = "Public IP of SonarQube EC2 instance"
  value       = module.sonarqube.public_ip
}

output "nexus_ip" {
  description = "Public IP of Nexus EC2 instance"
  value       = module.nexus.public_ip
}

output "docker_ip" {
  description = "Public IP of Docker host EC2 instance"
  value       = module.docker.public_ip
}
output "ansible_ip" {
```

```
  description = "Public IP of Ansible control node"
  value       = module.ansible.public_ip
}
```

provider.tf

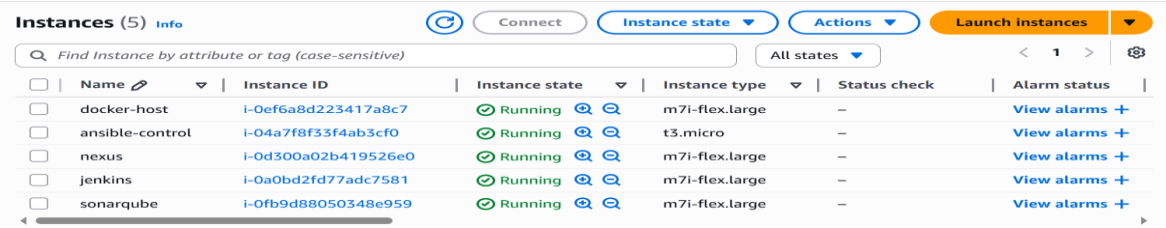
```
provider "aws" {
  region = var.aws_region
}
```

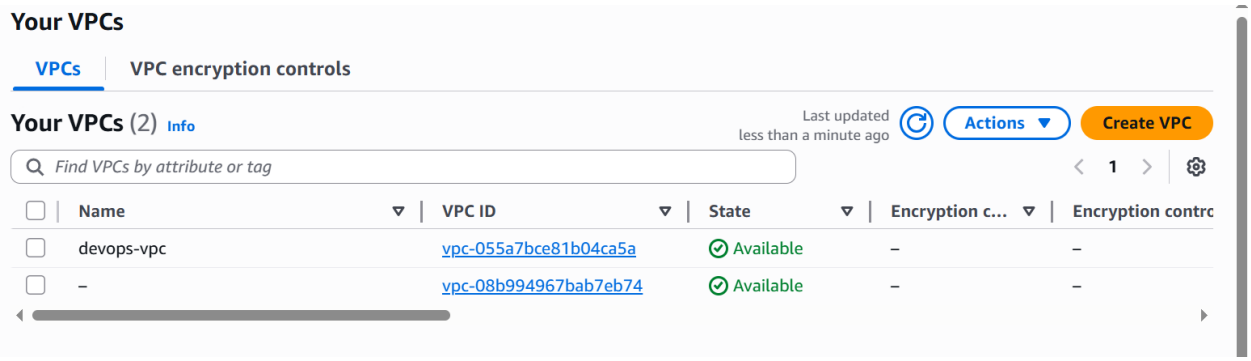
variable.tf

```
variable "aws_region" {
  default = "us-east-2"
}

variable "vpc_cidr" {
  default = "10.0.0.0/16"
}

variable "public_subnet_cidr" {
  default = "10.0.1.0/24"
}

variable "key_name" {
  default = "Lab"
}
```

version.tf

```
terraform {
  required_version = ">= 1.3.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "4.67.0"
    }
  }
}
```





This Files for s3 Creation:

main.tf

```
resource "aws_s3_bucket" "tf_state" {
  bucket = "satwik-devops-terraform-state"
  tags = {
    Name        = "terraform-state-bucket"
    Environment = "dev"
  }
}

resource "aws_s3_bucket_versioning" "tf_state_versioning" {
  bucket = aws_s3_bucket.tf_state.id
  versioning_configuration {
    status = "Enabled"
  }
}

resource "aws_s3_bucket_server_side_encryption_configuration"
"tf_state_encryption" {
  bucket = aws_s3_bucket.tf_state.id
  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm = "AES256"
    }
  }
}
```

```
resource "aws_s3_bucket_public_access_block" "tf_state_block" {
  bucket = aws_s3_bucket.tf_state.id
  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

resource "aws_dynamodb_table" "tf_lock" {
  name         = "terraform-state-locks"
  billing_mode = "PAY_PER_REQUEST"
  hash_key     = "LockID"
  attribute {
    name = "LockID"
    type = "S"
  }
  tags = {
    Name        = "terraform-lock-table"
    Environment = "dev"
  }
}
```

output.tf

```
output "s3_bucket_name" {
  description = "S3 bucket used for Terraform remote state"
  value       = aws_s3_bucket.tf_state.bucket
}

output "dynamodb_table_name" {
  description = "DynamoDB table used for Terraform state locking"
  value       = aws_dynamodb_table.tf_lock.name
}
```

provider.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
```

```
    }
  }
}

provider "aws" {
  region = "us-east-2"
}
```

modules

- compute-ansible
- compute-docker
- compute-sonar
- compute-nexus
- compute-jenkins

compute-ansible

main.tf

```
resource "aws_instance" "this" {
  ami                    = var.ami
  instance_type          = var.instance_type
  subnet_id              = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name               = var.key_name

  associate_public_ip_address = true

  tags = {
    Name        = var.name
    Role        = "ansible-control"
    Environment = "dev"
    ManagedBy   = "terraform"
  }

  root_block_device {
    volume_size = 20
    volume_type = "gp3"
    encrypted   = true
```

```
  }
}
```

output.tf

```
output "public_ip" {
  value = aws_instance.this.public_ip
}
```

variable.tf

```
variable "ami" {}
variable "instance_type" {}
variable "subnet_id" {}
variable "security_group_id" {}
variable "key_name" {}
variable "name" {}
```

compute-docker

main.tf

```
resource "aws_instance" "this" {
  ami                    = var.ami
  instance_type          = var.instance_type
  subnet_id              = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name               = var.key_name

  associate_public_ip_address = true

  tags = {
    Name        = var.name
    ManagedBy   = "terraform"
    Environment = "dev"
  }

  root_block_device {
    volume_size = 20
    volume_type = "gp3"
    encrypted   = true
```

```
  }
}
```

output.tf

```
output "public_ip" {
  value = aws_instance.this.public_ip
}
```

variable.tf

```
variable "ami" {}
variable "instance_type" {}
variable "subnet_id" {}
variable "security_group_id" {}
variable "key_name" {}
variable "name" {}
```

compute-sonar

main.tf

```
resource "aws_instance" "this" {
  ami                    = var.ami
  instance_type          = var.instance_type
  subnet_id              = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name               = var.key_name

  associate_public_ip_address = true

  tags = {
    Name        = var.name
    ManagedBy   = "terraform"
    Environment = "dev"
  }

  root_block_device {
    volume_size = 20
    volume_type = "gp3"
```

```
    encrypted   = true
  }
}
```

output.tf

```
output "public_ip" {
  value = aws_instance.this.public_ip
}
```

variable.tf

```
variable "ami" {}
variable "instance_type" {}
variable "subnet_id" {}
variable "security_group_id" {}
variable "key_name" {}
variable "name" {}
```

compute-nexus

main.tf

```
resource "aws_instance" "this" {
  ami                    = var.ami
  instance_type          = var.instance_type
  subnet_id              = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name               = var.key_name

  associate_public_ip_address = true

  tags = {
    Name        = var.name
    ManagedBy   = "terraform"
    Environment = "dev"
  }

  root_block_device {
    volume_size = 20
```

```
    volume_type = "gp3"
    encrypted   = true
  }
}
```

output.tf

```
output "public_ip" {
  value = aws_instance.this.public_ip
}
```

variable.tf

```
variable "ami" {}
variable "instance_type" {}
variable "subnet_id" {}
variable "security_group_id" {}
variable "key_name" {}
variable "name" {}
```

compute-jenkins

main.tf

```
resource "aws_instance" "this" {
  ami                    = var.ami
  instance_type          = var.instance_type
  subnet_id              = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name               = var.key_name

  associate_public_ip_address = true

  tags = {
    Name        = var.name
    ManagedBy   = "terraform"
    Environment = "dev"
  }

  root_block_device {
```

```
    volume_size = 20
    volume_type = "gp3"
    encrypted   = true
  }
}
```

output.tf

```
output "public_ip" {
  value = aws_instance.this.public_ip
}
```

variable.tf

```
variable "ami" {}
variable "instance_type" {}
variable "subnet_id" {}
variable "security_group_id" {}
variable "key_name" {}
variable "name" {}
```

network folder

main.tf

```
resource "aws_vpc" "this" {
  cidr_block           = var.vpc_cidr
  enable_dns_support   = true
  enable_dns_hostnames = true

  tags = {
    Name        = "devops-vpc"
    Environment = "dev"
    ManagedBy   = "terraform"
  }
}

resource "aws_subnet" "public" {
  vpc_id                  = aws_vpc.this.id
  cidr_block              = var.public_subnet_cidr
```

```
  map_public_ip_on_launch = true

  tags = {
    Name        = "devops-public-subnet"
    Environment = "dev"
    ManagedBy   = "terraform"
  }
}

resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id

  tags = {
    Name        = "devops-igw"
    Environment = "dev"
    ManagedBy   = "terraform"
  }
}

resource "aws_route_table" "public" {
  vpc_id = aws_vpc.this.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.this.id
  }

  tags = {
    Name        = "devops-public-rt"
    Environment = "dev"
    ManagedBy   = "terraform"
  }
}

resource "aws_route_table_association" "public" {
  subnet_id      = aws_subnet.public.id
  route_table_id = aws_route_table.public.id
}
```

output.tf

```
output "vpc_id" {
  value = aws_vpc.this.id
```

```
}

output "subnet_id" {
  value = aws_subnet.public.id
}
```

variable.tf

```
variable "vpc_cidr" {}
variable "public_subnet_cidr" {}
```

security folder

main.tf

```
resource "aws_security_group" "jenkins" {
  name   = "jenkins-sg"
  vpc_id = var.vpc_id

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

  ingress {
    from_port   = 8080
    to_port     = 8080
    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"]
  }
```

```
}

resource "aws_security_group" "sonarqube" {
  name   = "sonarqube-sg"
  vpc_id = var.vpc_id

  ingress {
    from_port       = 9000
    to_port         = 9000
    protocol        = "tcp"
    security_groups = [aws_security_group.jenkins.id]
  }

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

resource "aws_security_group" "nexus" {
  name   = "nexus-sg"
  vpc_id = var.vpc_id

  ingress {
    from_port       = 8081
    to_port         = 8081
    protocol        = "tcp"
    security_groups = [aws_security_group.jenkins.id]
  }

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

resource "aws_security_group" "docker" {
  name   = "docker-sg"
  vpc_id = var.vpc_id
```

```
  ingress {
    from_port       = 22
    to_port         = 22
    protocol        = "tcp"
    security_groups = [aws_security_group.jenkins.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
resource "aws_security_group" "ansible" {
  name   = "ansible-sg"
  vpc_id = var.vpc_id

  ingress {
    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"]
  }
}
```

output.tf

```
output "jenkins_sg" {
  value = aws_security_group.jenkins.id
}

output "sonarqube_sg" {
  value = aws_security_group.sonarqube.id
}
```

```
output "nexus_sg" {
  value = aws_security_group.nexus.id
}

output "docker_sg" {
  value = aws_security_group.docker.id
}

output "ansible_sg" {
  value = aws_security_group.ansible.id
}
```

variable.tf

```
variable "vpc_id" {}
```

Step –2: Using Ansible

- Install and Expose Docker
- Install and Expose SonarQube
- Install and Expose Nexus
- Install and Expose Jenkins

Now, Install Ansible and Clone the Project

```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ ls
ansible.cfg     docker.yml     jenkins-docker.yml  nexus.yml  roles
deploy-app.yml  inventory.ini  jenkins.yml         readme.md  sonarqube.yml
```

Now, Adding all IP's into Inventory.ini

```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ cat inventory.ini
[jenkins]
10.0.1.11

[sonarqube]
10.0.1.179

[nexus]
10.0.1.224

[docker]
10.0.1.209

[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

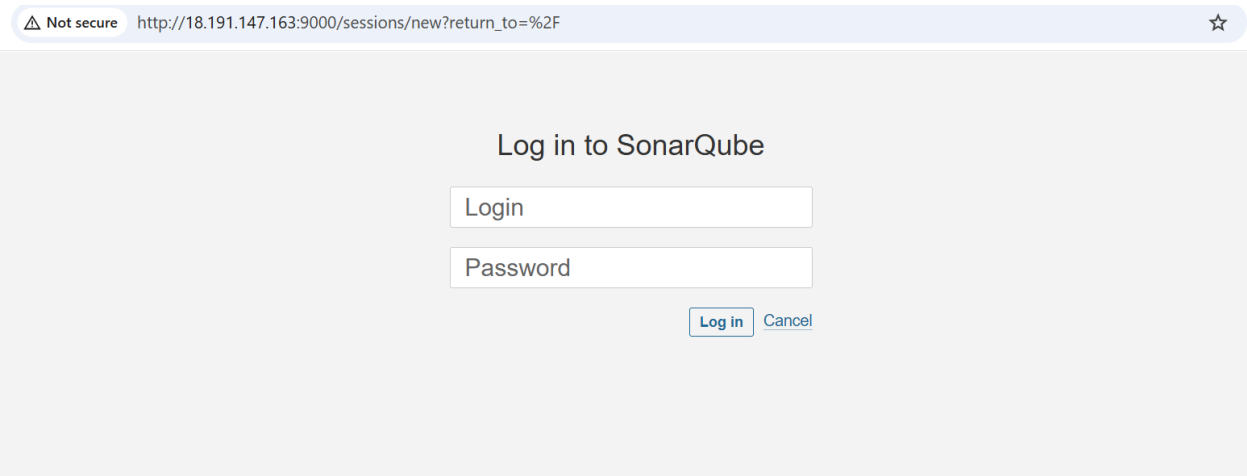
Now, add the required .yml files to expose

Now, Installing Docker

```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ cat docker.yml
---
- name: Install Docker on required servers
  hosts: docker,sonarqube,nexus
  become: true
  roles:
    - common
    - docker
```

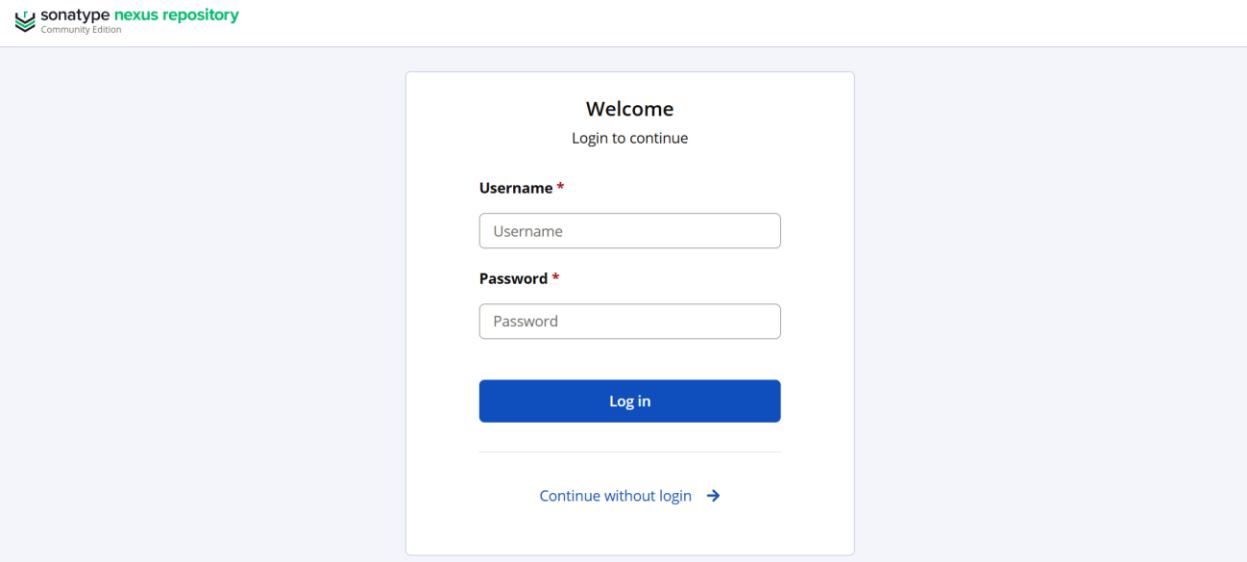
Now, Installing SonarQube

```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ cat sonarqube.yml
---
- name: Setup SonarQube server
  hosts: sonarqube
  become: true
  roles:
    - sonarqube
```



Now, Installing Nexus

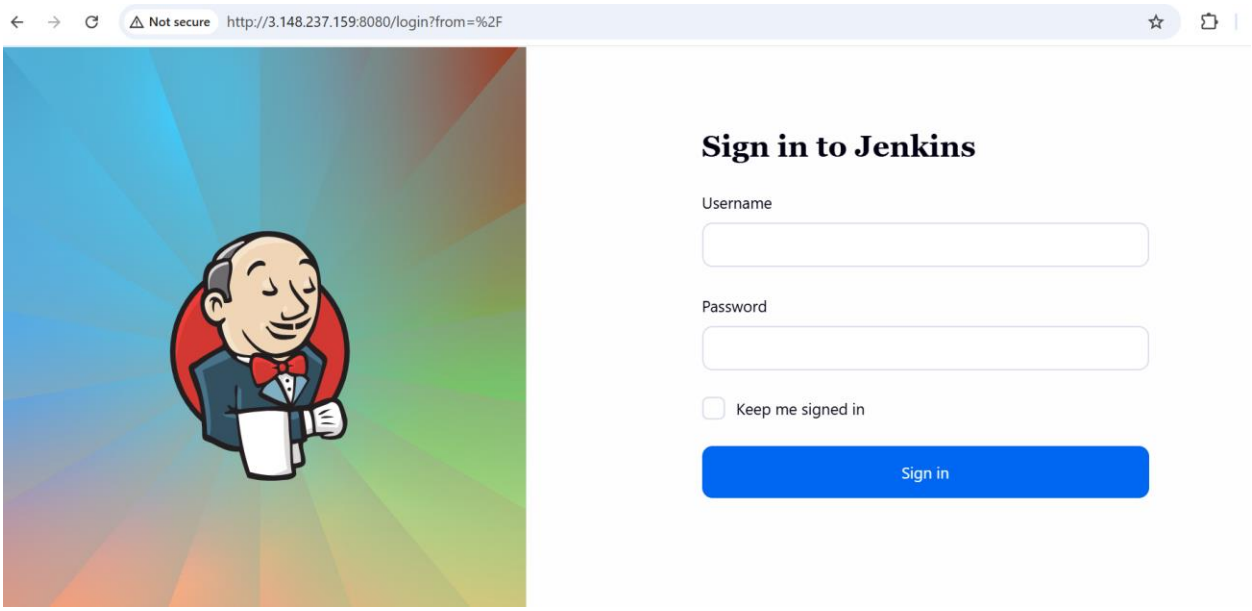
```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ cat nexus.yml
---
- name: Setup Nexus server
  hosts: nexus
  become: true
  roles:
    - nexusubuntu@ansible-host:~/Devops-Capestone-project/ansible$
```



Now, Installing Jenkins

```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ cat jenkins.yml
---
- name: Setup Jenkins server
  hosts: jenkins
  become: true
  roles:
    - common
    - jenkinsubuntu@ansible-host:~/Devops-Capestone-project/ansible$
```

```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ cat jenkins-docker.yml
---
- name: Install Docker on Jenkins server
  hosts: jenkins
  become: yes

  roles:
    - docker

  tasks:
    - name: Restart Jenkins service
      service:
        name: jenkins
        state: restarted
```



Now Install Suggested Plugins:

commons-lang3 v3.x Jenkins API

Ionicons API

Folders

OWASP Markup Formatter

ASM API

JSON Path API

Structs

Pipeline: Step API

commons-text API

Token Macro

Build Timeout

bouncycastle API

Credentials

Plain Credentials

Variant

SSH Credentials

Credentials Binding

SCM API

Pipeline: API

Timestamper

Caffeine API

Script Security

JavaBeans Activation Framework (JAF) API

JAXB

SnakeYAML API

Jakarta Activation API

Jakarta XML Binding API

JSON Api

Jackson 2 API

Pipeline: Supporting APIs

Plugin Utilities API

Font Awesome API

Bootstrap 5 API

JQuery3 API

ECharts API

Prism API

Display URL API

Checks API

JUnit

Matrix Project

Resource Disposer

Workspace Cleanup

Ant

OkHttp

Apache HttpComponents Client 4.x API

Pipeline: Job

Gradle

Pipeline: Milestone Step

Durable Task

Pipeline: Nodes and Processes

Pipeline: Build Step

Pipeline: SCM Step

Pipeline: Groovy

Pipeline: Groovy Libraries

Joda Time API

Pipeline: Model API

Pipeline: Stage Step

Pipeline: Declarative Extension Points API

Jakarta Mail API

Instance Identity

Mailer

Branch API

Pipeline: Multibranch

Pipeline: Stage Tags Metadata

Pipeline: Input Step

Pipeline: Basic Steps

Pipeline: Declarative

Pipeline

Java JSON Web Token (JJWT)

GitHub API

Mina SSHD API :: Common

Mina SSHD API :: Core

Gson API

Git client

Git

GitHub

GitHub Branch Source

Pipeline: GitHub Groovy Libraries

Metrics

Pipeline Graph View

Git

EDDSA API

Trilead API

SSH Build Agents

Matrix Authorization Strategy

LDAP

jsoup API

Email Extension

Mailer

Theme Manager

Dark Theme

Loading plugin extensions

SSH server

Git server

Pipeline Graph Analysis

Common API for Blue Ocean

REST API for Blue Ocean

Pub-Sub "light" Bus

Pipeline SCM API for Blue Ocean

HTML Publisher

Design Language

Blue Ocean Core JS

Web for Blue Ocean

JWT for Blue Ocean

Favorite

REST Implementation for Blue Ocean

Pipeline implementation for Blue Ocean

Git Pipeline for Blue Ocean

GitHub Pipeline for Blue Ocean

Git Parameter

Jersey 3 API

GitLab

Generic Webhook Trigger

GitLab API

GitLab Authentication

GitHub Integration

GitHub Authentication

Pipeline: GitHub

JavaMail API

Pipeline GitHub Notify Step

SonarQube Scanner

Apache HttpComponents Client 5.x API

Sonar Quality Gates

Oracle Java SE Development Kit Installer

Command Agent Launcher

Quality Gates

Sonargraph Integration

DataTables.net API

Forensics API

Coverage

Code Coverage

SonarQube Generic Coverage

Nexus Artifact Uploader

artifact-promotion

Cloud Statistics

Authentication Tokens API

Docker Commons

Commons Compress API

Docker API

Docker

Docker Pipeline

Docker API

Dev Tools Symbols API

Javadoc

JSch dependency

Maven Integration

docker-build-step

Docker Compose Build Step

Ansible Pending

Ansible Tower

Maven Integration

Config File Provider

Jersey 2 API

Artifactory

Pipeline Graph Analysis

Pipeline: REST API

Pipeline: Stage View

Pipeline: Declarative Agent API

Pipeline Maven Plugin API

Pipeline Maven Integration

SSH Pipeline Steps

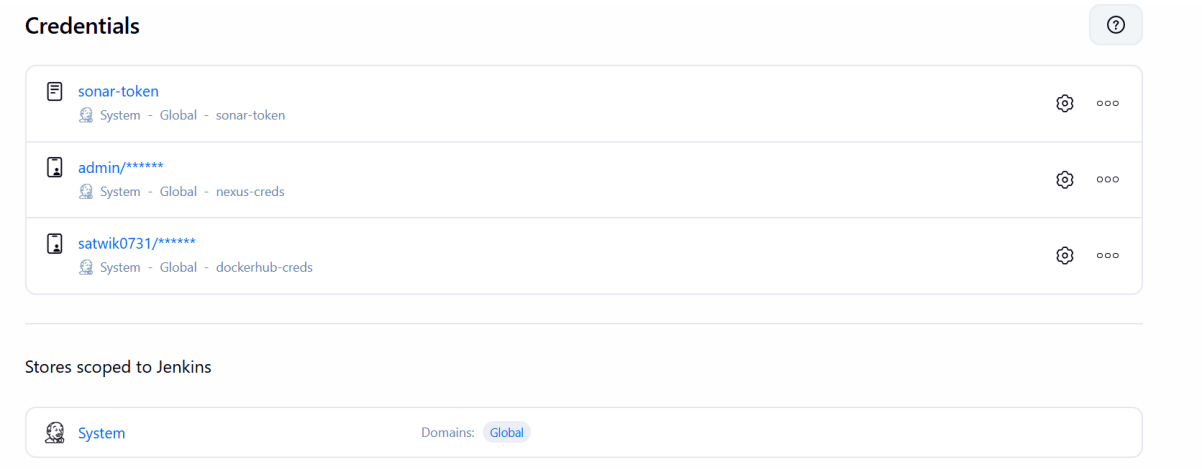
Parameterized Trigger

jQuery

Build Pipeline

Loading plugin extensions

After Installation add Credentials and Tokens of sonar, nexus and docker hub



Now Create One Pipeline Job and add pipeline to it

Pipeline:

pipeline {

agent any

```
environment {
    SONARQUBE_URL = ' http://18.191.147.163:9000'
    APP_NAME      = ' registrationform'
    DOCKER_IMAGE  = ' satwik0731/registrationform'
    VERSION       = "${env.BUILD_NUMBER}"
}

tools {
    maven 'maven3'
}

stages {

    stage('Checkout') {
        steps {
            checkout scm
        }
```

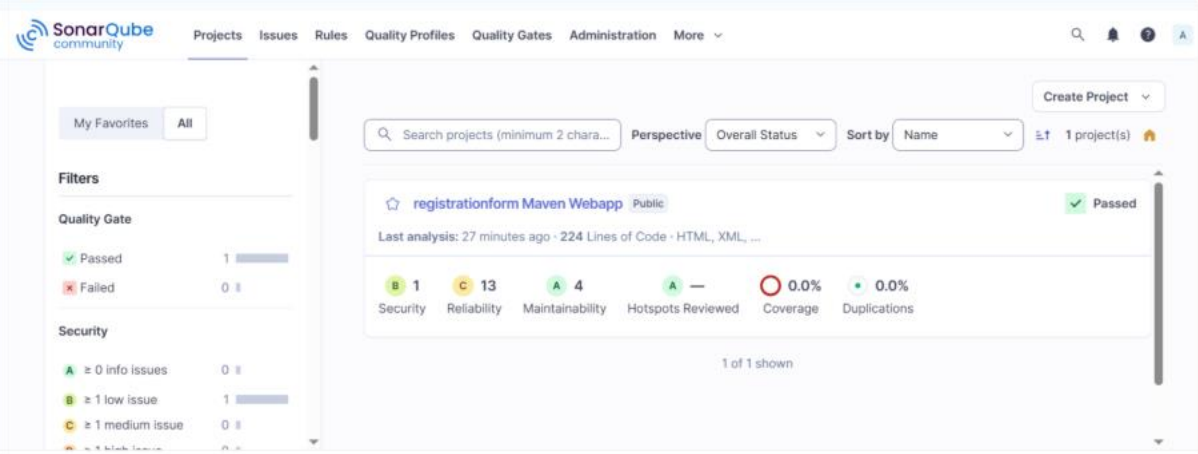
```
}

stage('Build & Test') {
    steps {
        sh 'mvn clean package'
    }
    post {
        always {
            junit testResults: '**/target/surefire-reports/*.xml',
                  allowEmptyResults: true
        }
    }
}

stage('SonarQube Analysis') {
    steps {
        withCredentials([string(
            credentialsId: 'sonarqube-token',
            variable: 'SONAR_TOKEN'
        )]) {
            sh '''
                mvn sonar:sonar \
                -Dsonar.projectKey= registrationform \
                -Dsonar.host.url= http://18.191.147.163:9000 \
                -Dsonar.login=$SONAR_TOKEN
            '''
        }
    }
}

stage('Upload to Nexus (Snapshots)') {
    steps {
        withCredentials([usernamePassword(
            credentialsId: 'nexus-creds',
            usernameVariable: 'NEXUS_USER',
            passwordVariable: 'NEXUS_PASS'
        )]) {
            sh '''
```

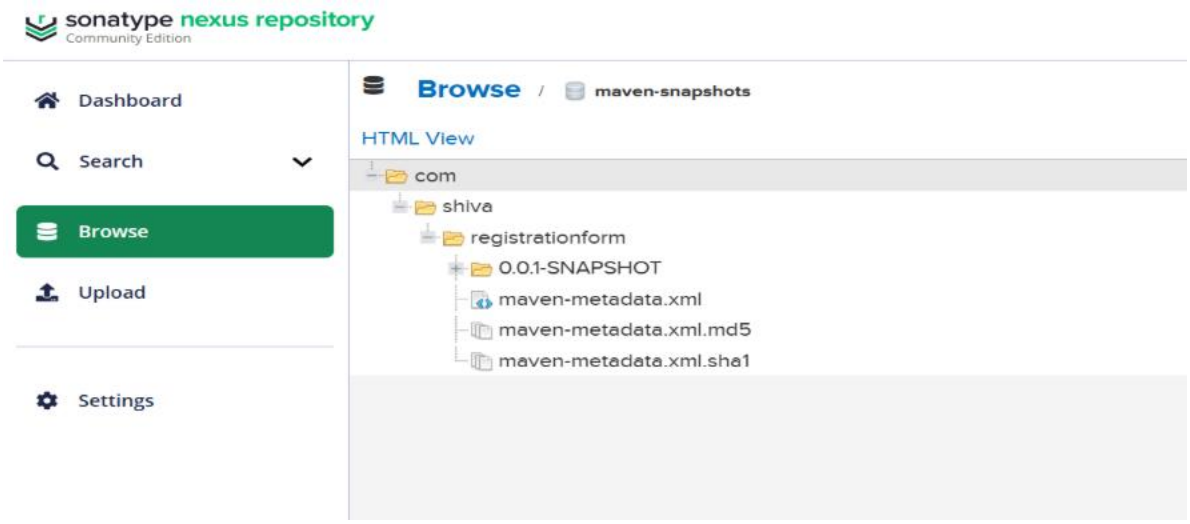
cat < settings.xml nexus ${NEXUS_USER} ${NEXUS_PASS} EOF '" sh 'mvn deploy -DskipTests -s settings.xml' } } }

```
   stage('Build Docker Image') {
        steps {
            sh """
                docker build -t ${DOCKER_IMAGE}:${VERSION} .
            """
        }
    }

    stage('Push Docker Image to DockerHub') {
        steps {
            withCredentials([usernamePassword(
                credentialsId: 'dockerhub-creds',
                usernameVariable: 'DH_USER',
                passwordVariable: 'DH_PASS'
            )]) {
                sh """
                    echo "$DH_PASS" | docker login -u "$DH_USER" --
password-stdin
                    docker push ${DOCKER_IMAGE}:${VERSION}
                    docker tag ${DOCKER_IMAGE}:${VERSION}
${DOCKER_IMAGE}:latest
                    docker push ${DOCKER_IMAGE}:latest
                """
            }
        }
    }

    stage('Deploy to Docker Host (via Ansible)') {
        steps {
            sh """
                ssh -o StrictHostKeyChecking=no ubuntu@3.148.241.187\
                'cd ~/ansible && bash -lc "ansible-playbook -i
inventory.ini deploy-app.yml"'
            """
        }
    }
}
```

```
post {
    success {
        echo "✅ CI/CD Pipeline completed successfully!"
    }
    failure {
        echo "❌ Pipeline failed. Check logs."
    }
    always {
        cleanWs()
    }
}


}
```

Now, Check Sonar, Nexus Once to check the application





Now, run deploy-app.yml file in ansible to expose

```
ubuntu@ansible-host:~/Devops-Capestone-project/ansible$ cat deploy-app.yml
---
- name: Deploy registrationform on Docker Host
  hosts: docker
  become: yes

  tasks:
    - name: Pull latest image
      docker_container:
        name: registrationform
        image: satwik0731/registrationform:latest
        state: started
        restart_policy: always
        ports:
          - "8866:8080"
        recreate: yes
```

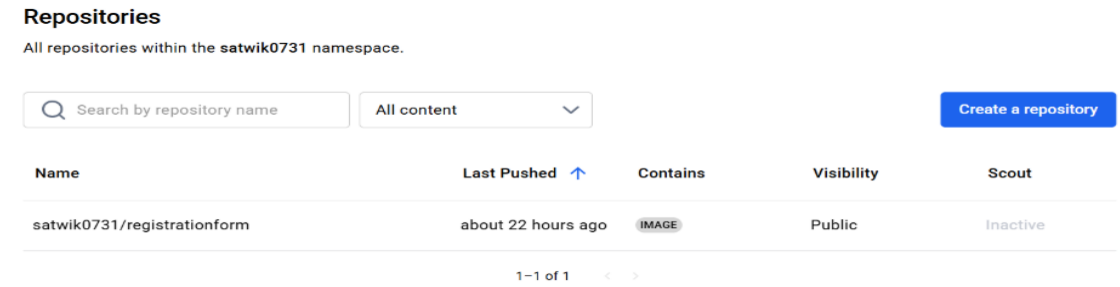
Now, you can Observe your Project Image and Container is Up and Running

- docker images
- docker ps

```
ubuntu@Java-Application:~$ docker images
REPOSITORY                   TAG       IMAGE ID       CREATED         SIZE
satwik0731/registrationform   v11       25b9d87d14a1   22 hours ago   505MB
```

```
ubuntu@Java-Application:~$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED        STATUS                  PORTS
                  NAMES
e49db8c3d0ac   25b9d87d14a1       "catalina.sh run"        22 hours ago   Up 3 hours (healthy)    0.0.0.0:8866->8080/tcp, [::]:8866->808
0/tcp             satwik11
```

Now The Image has Stored in Docker Hub:



Now Our Project is Expose on port: 8866