A REPORT SUBMISSION ON

16 JUNE 2024

DEVOPS PROJECT
HOSTING APPLICATION

Submitted to 3RI Technologies

Submitted by

Ashutosh Gore

Under the Guidance of

Mr. Manish Badgujar

(Trainer)

DevOps CI/CD Project

Introduction

In this project, I integrated Terraform, AWS, GitHub, Jenkins, and Docker to create a robust CI/CD pipeline for hosting an application. The following documentation outlines the steps taken to achieve this.

Step 1: Creating AWS Infrastructure with Terraform

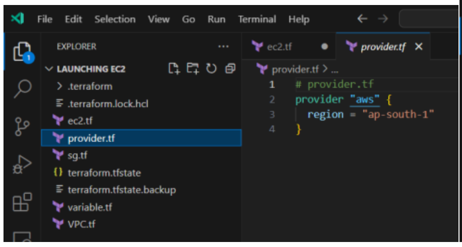
Tools Used

- **Terraform**: An open-source infrastructure as a code software tool.
- **Visual Studio Code**: A source-code editor used for writing and managing Terraform scripts.

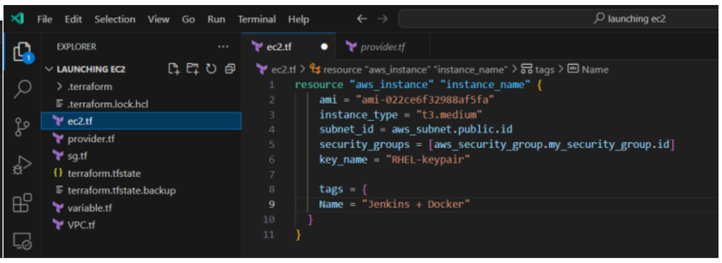
Objective

To provision and manage the necessary AWS infrastructure including EC2 instances, VPCs, and security groups using Terraform.

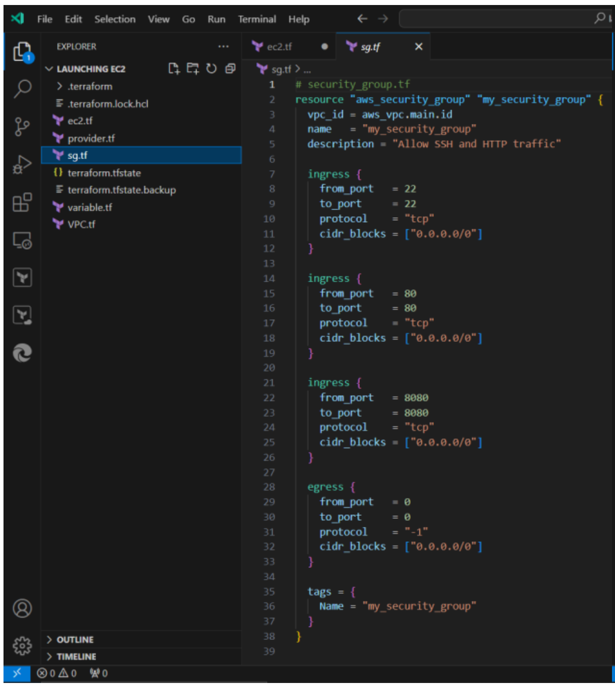
1. **Set Up Your Terraform Environment**
   - Ensure Terraform is installed on your machine.
   - Install Visual Studio Code.
2. **Write the Terraform Code**
   - Open Visual Studio Code and create a new directory for your Terraform scripts (‘LAUNCHING EC2’ directory name).



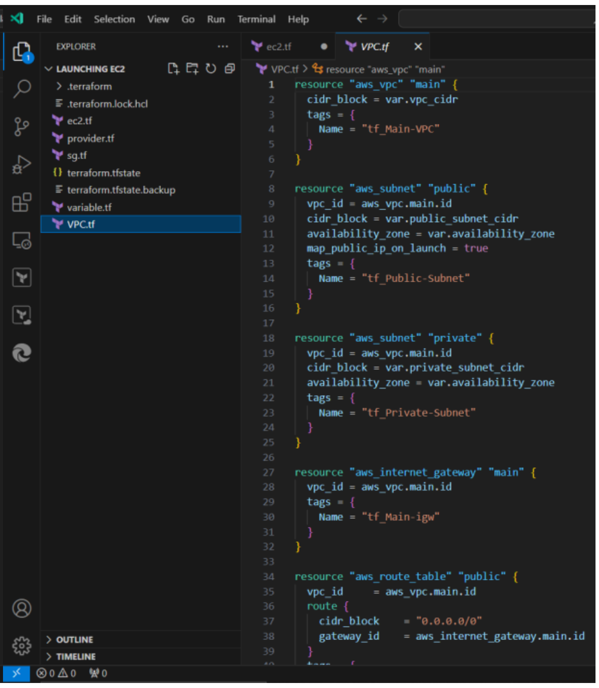
```
# provider.tf
provider "aws" {
  region = "ap-south-1"
}
```



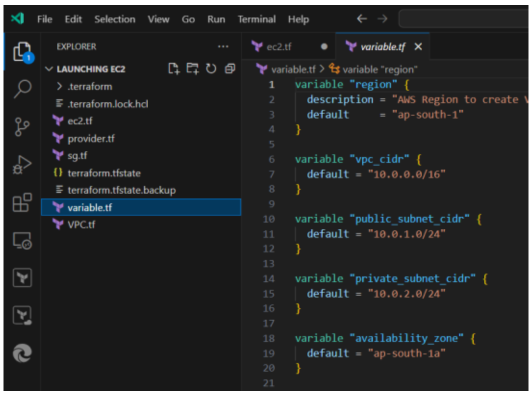
```
resource "aws_instance" "instance_name" {
    ami = "ami-022ce6f32988af5fa"
    instance_type = "t3.medium"
    subnet_id = aws_subnet.public.id
    security_groups = [aws_security_group.my_security_group.id]
    key_name = "RHEL-keypair"

    tags = {
    Name = "Jenkins + Docker"
    }
}
```



```hcl
# security_group.tf
resource "aws_security_group" "my_security_group" {
  vpc_id = aws_vpc.main.id
  name   = "my_security_group"
  description = "Allow SSH and HTTP traffic"

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

  ingress {
    from_port   = 80
    to_port     = 80
    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"]
  }

  tags = {
    Name = "my_security_group"
  }
}
```



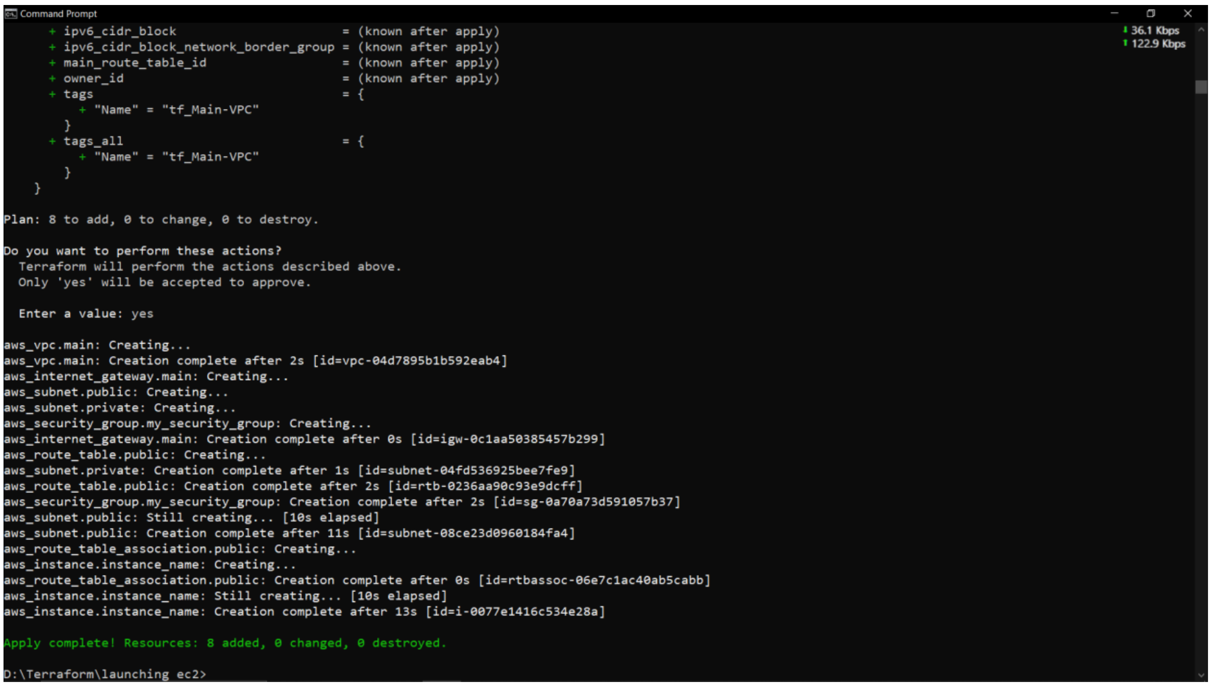
```hcl
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags = {
    Name = "tf_Main-VPC"
  }
}

resource "aws_subnet" "public" {
  vpc_id = aws_vpc.main.id
  cidr_block = var.public_subnet_cidr
  availability_zone = var.availability_zone
  map_public_ip_on_launch = true
  tags = {
    Name = "tf_Public-Subnet"
  }
}

resource "aws_subnet" "private" {
  vpc_id = aws_vpc.main.id
  cidr_block = var.private_subnet_cidr
  availability_zone = var.availability_zone
  tags = {
    Name = "tf_Private-Subnet"
  }
}

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "tf_Main-igw"
  }
}

resource "aws_route_table" "public" {
  vpc_id      = aws_vpc.main.id
  route {
    cidr_block     = "0.0.0.0/0"
    gateway_id     = aws_internet_gateway.main.id
  }
```



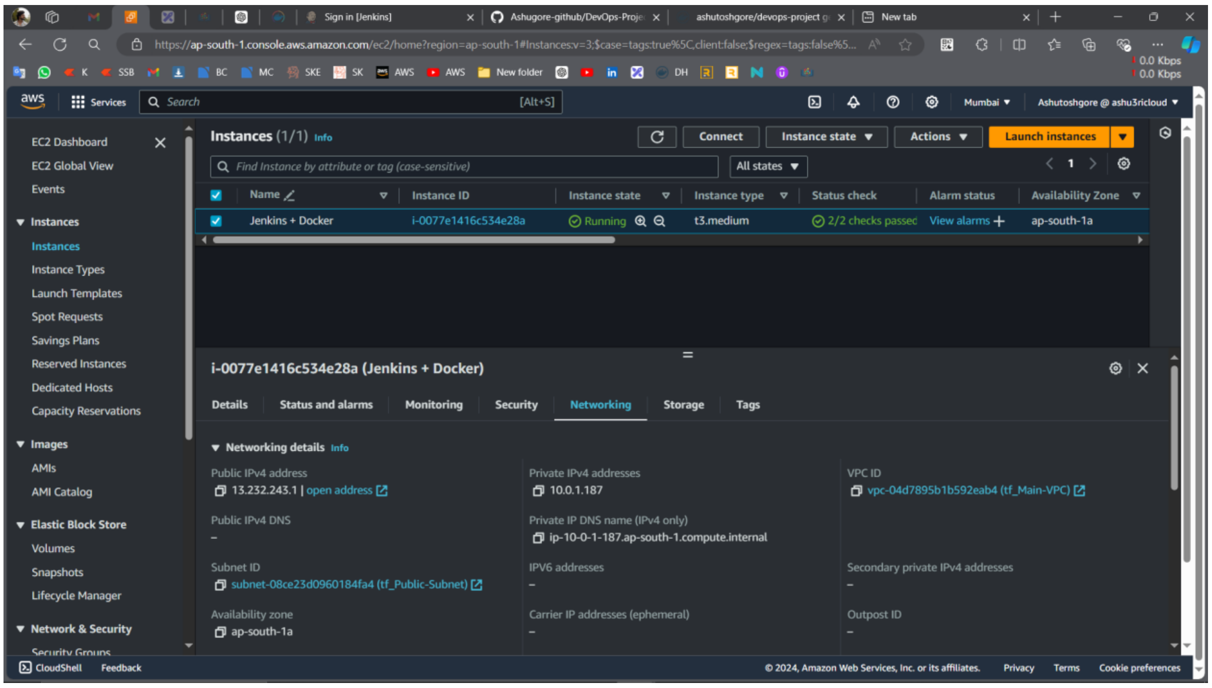
```hcl
variable "region" {
  description = "AWS Region to create V
  default     = "ap-south-1"
}

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

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

variable "private_subnet_cidr" {
  default = "10.0.2.0/24"
}

variable "availability_zone" {
  default = "ap-south-1a"
}
```

3. **Initialize and Apply Terraform**

- Open a terminal in CMD
- Run ‘terraform plan’
- Run ‘terraform apply’



```
Command Prompt
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "tf_Main-VPC"
        }
      + tags_all                             = {
          + "Name" = "tf_Main-VPC"
        }
    }

Plan: 8 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_vpc.main: Creating...
aws_vpc.main: Creation complete after 2s [id=vpc-04d7895b1b592eab4]
aws_internet_gateway.main: Creating...
aws_subnet.public: Creating...
aws_subnet.private: Creating...
aws_security_group.my_security_group: Creating...
aws_internet_gateway.main: Creation complete after 0s [id=igw-0c1aa50385457b299]
aws_route_table.public: Creating...
aws_subnet.private: Creation complete after 1s [id=subnet-04fd536925bee7fe9]
aws_route_table.public: Creation complete after 2s [id=rtb-0236aa90c93e9dcff]
aws_security_group.my_security_group: Creation complete after 2s [id=sg-0a70a73d591057b37]
aws_subnet.public: Still creating... [10s elapsed]
aws_subnet.public: Creation complete after 11s [id=subnet-08ce23d0960184fa4]
aws_route_table_association.public: Creating...
aws_instance.instance_name: Creating...
aws_route_table_association.public: Creation complete after 0s [id=rtbassoc-06e7c1ac40ab5cabb]
aws_instance.instance_name: Still creating... [10s elapsed]
aws_instance.instance_name: Creation complete after 13s [id=i-0077e1416c534e28a]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

D:\Terraform\launching ec2>
```



https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:v=3;$case=tags:true%5C,client:false;$regex=tags:false%5...

Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Jenkins + Docker	i-0077e1416c534e28a	Running	t3.medium	2/2 checks passed	View alarms +	ap-south-1a

i-0077e1416c534e28a (Jenkins + Docker)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

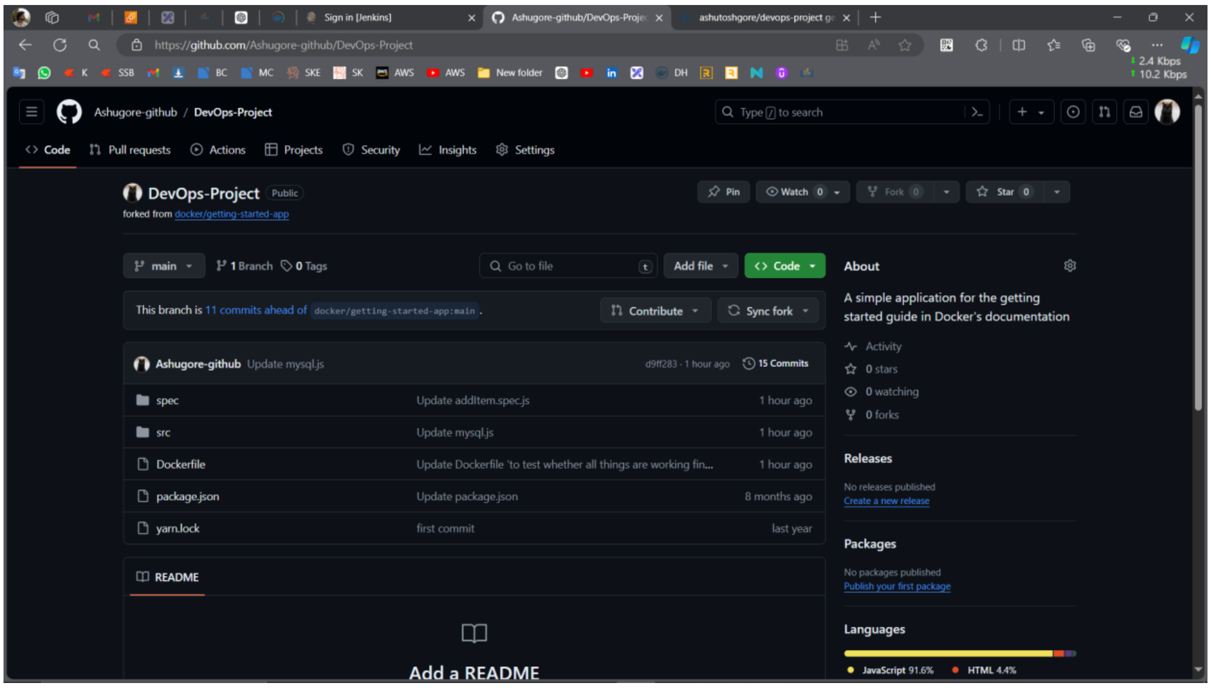
Networking details Info

Public IPv4 address: 13.232.243.1 | open address
Private IPv4 addresses: 10.0.1.187
VPC ID: vpc-04d7895b1b592eab4 (tf_Main-VPC)
Public IPv4 DNS: –
Private IP DNS name (IPv4 only): ip-10-0-1-187.ap-south-1.compute.internal
Subnet ID: subnet-08ce23d0960184fa4 (tf_Public-Subnet)
IPV6 addresses: –
Secondary private IPv4 addresses: –
Availability zone: ap-south-1a
Carrier IP addresses (ephemeral): –
Outpost ID: –

© 2024, Amazon Web Services, Inc. or its affiliates.

Step 2: Forking the Application Code from GitHub

My Github URL: - https://github.com/Ashugore-github/DevOps-Project


Ashugore-github / DevOps-Project

DevOps-Project Public
forked from docker/getting-started-app

main · 1 Branch · 0 Tags

This branch is 11 commits ahead of docker/getting-started-app:main.

Ashugore-github Update mysql.js · d9ff283 · 1 hour ago · 15 Commits

spec	Update addItem.spec.js	1 hour ago
src	Update mysql.js	1 hour ago
Dockerfile	Update Dockerfile 'to test whether all things are working fin...	1 hour ago
package.json	Update package.json	8 months ago
yarn.lock	first commit	last year

README

Add a README

About

A simple application for the getting started guide in Docker's documentation

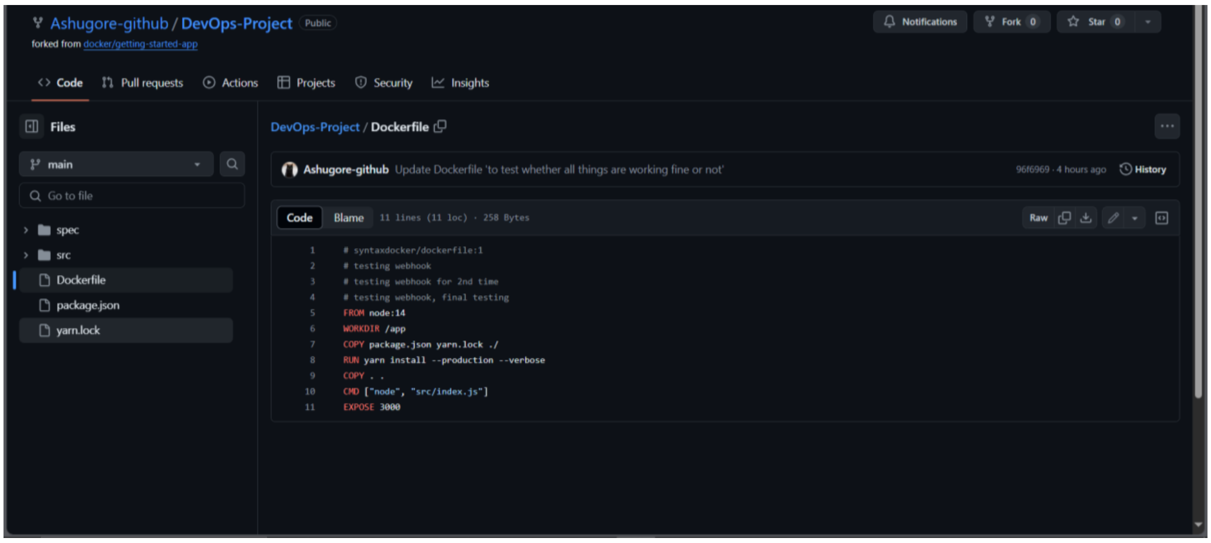
Activity
0 stars
0 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages
JavaScript 91.6% · HTML 4.4%

- Create Dockerfile in GitHub.
-


Ashugore-github / DevOps-Project Public
forked from docker/getting-started-app

DevOps-Project / Dockerfile

Ashugore-github Update Dockerfile 'to test whether all things are working fine or not' · 96f6969 · 4 hours ago

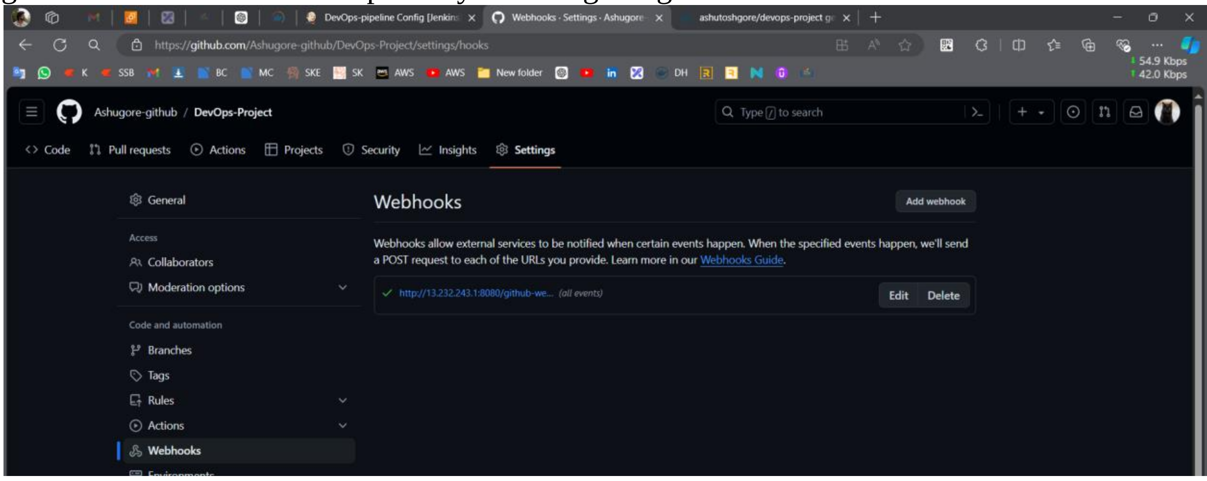
11 lines (11 loc) · 258 Bytes

```
# syntaxdocker/dockerfile:1
# testing webhook
# testing webhook for 2nd time
# testing webhook, final testing
FROM node:14
WORKDIR /app
COPY package.json yarn.lock ./
RUN yarn install --production --verbose
COPY . .
CMD ["node", "src/index.js"]
EXPOSE 3000
```

Step 3: Installation of Jenkins and Docker on RHEL 9 EC2 Instance

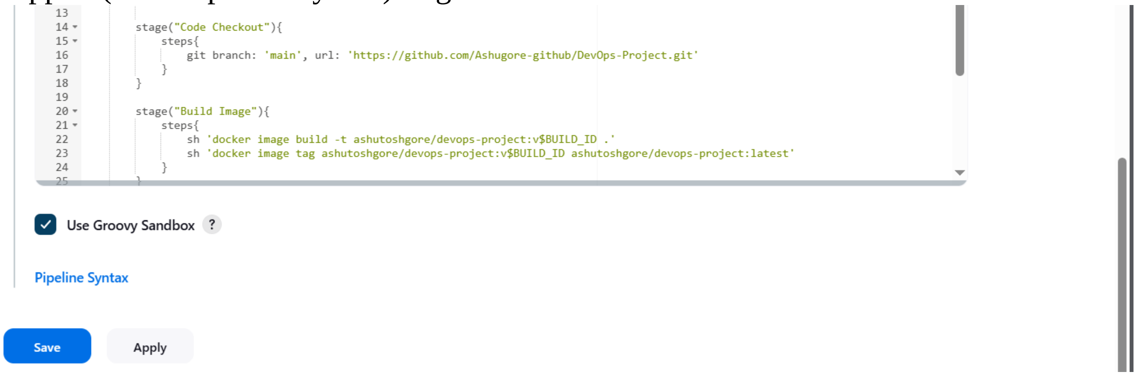
- Install Jenkins and create an account.
- Install docker.

Step 4: Integrating GitHub and Jenkins

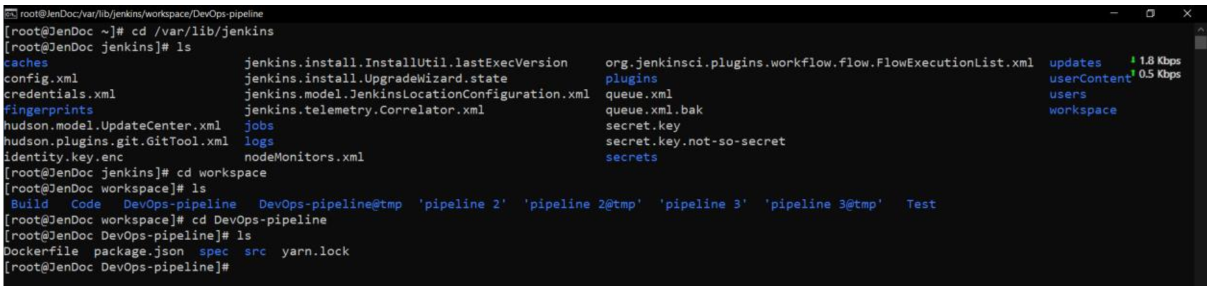
- Creating webhook on the GitHub repository and integrating it with Jenkins.
  - 

    Webhooks

    Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

    ✓ http://13.232.243.1:8080/github-we... (all events)
- Enable the checkbox of pipeline→Build Triggers→GitHub hook trigger for GITScm polling
- Create a snippet (from Pipeline Syntax) to generate url.
  - 

```
        stage("Code Checkout"){
            steps{
                git branch: 'main', url: 'https://github.com/Ashugore-github/DevOps-Project.git'
            }
        }

        stage("Build Image"){
            steps{
                sh 'docker image build -t ashutoshgore/devops-project:v$BUILD_ID .'
                sh 'docker image tag ashutoshgore/devops-project:v$BUILD_ID ashutoshgore/devops-project:latest'
            }
```

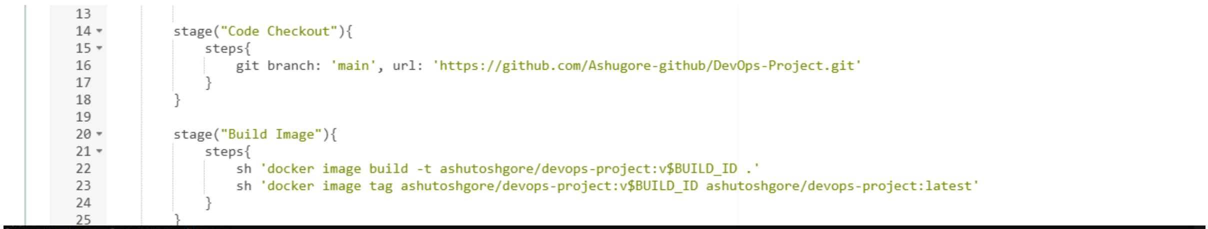
- Now we can see the GitHub file in an instance.
  - 

```
[root@JenDoc ~]# cd /var/lib/jenkins
[root@JenDoc jenkins]# ls
caches                       jenkins.install.InstallUtil.lastExecVersion     org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml  updates
config.xml                   jenkins.install.UpgradeWizard.state             plugins                                                    userContent
credentials.xml              jenkins.model.JenkinsLocationConfiguration.xml  queue.xml                                                  users
fingerprints                 jenkins.telemetry.Correlator.xml                queue.xml.bak                                              workspace
hudson.model.UpdateCenter.xml  jobs                                          secret.key
hudson.plugins.git.GitTool.xml  logs                                         secret.key.not-so-secret
identity.key.enc             nodeMonitors.xml                                secrets
[root@JenDoc jenkins]# cd workspace
[root@JenDoc workspace]# ls
 Build   Code   DevOps-pipeline   DevOps-pipeline@tmp  'pipeline 2'  'pipeline 2@tmp'  'pipeline 3'  'pipeline 3@tmp'   Test
[root@JenDoc workspace]# cd DevOps-pipeline
[root@JenDoc DevOps-pipeline]# ls
Dockerfile  package.json  spec  src  yarn.lock
[root@JenDoc DevOps-pipeline]#
```

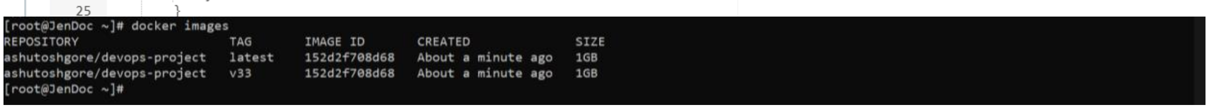
Step 5: Build an Image and Push it in Docker Hub.

1. Installing images

- These commands will 2 images one is with the latest tag and another one is with the build no.
- The $BUILD_ID is a placeholder for a build identifier, which you would replace with an actual value during the build process.

- 

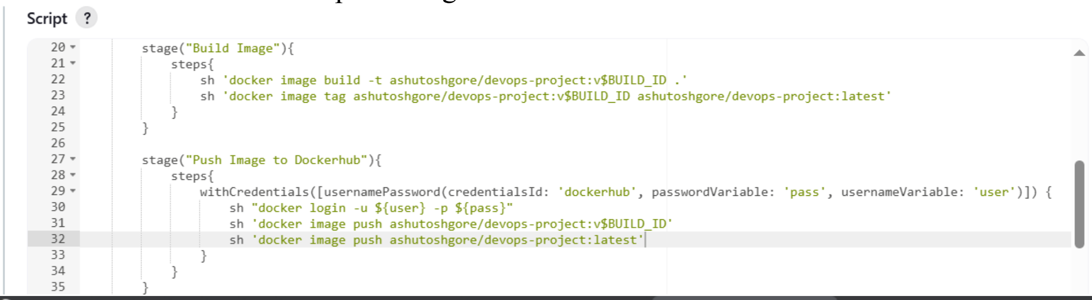
```
    stage("Code Checkout"){
        steps{
            git branch: 'main', url: 'https://github.com/Ashugore-github/DevOps-Project.git'
        }
    }

    stage("Build Image"){
        steps{
            sh 'docker image build -t ashutoshgore/devops-project:v$BUILD_ID .'
            sh 'docker image tag ashutoshgore/devops-project:v$BUILD_ID ashutoshgore/devops-project:latest'
        }
    }
```

- 

```
[root@JenDoc ~]# docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
ashutoshgore/devops-project latest    152d2f708d68   About a minute ago   1GB
ashutoshgore/devops-project v33       152d2f708d68   About a minute ago   1GB
[root@JenDoc ~]#
```

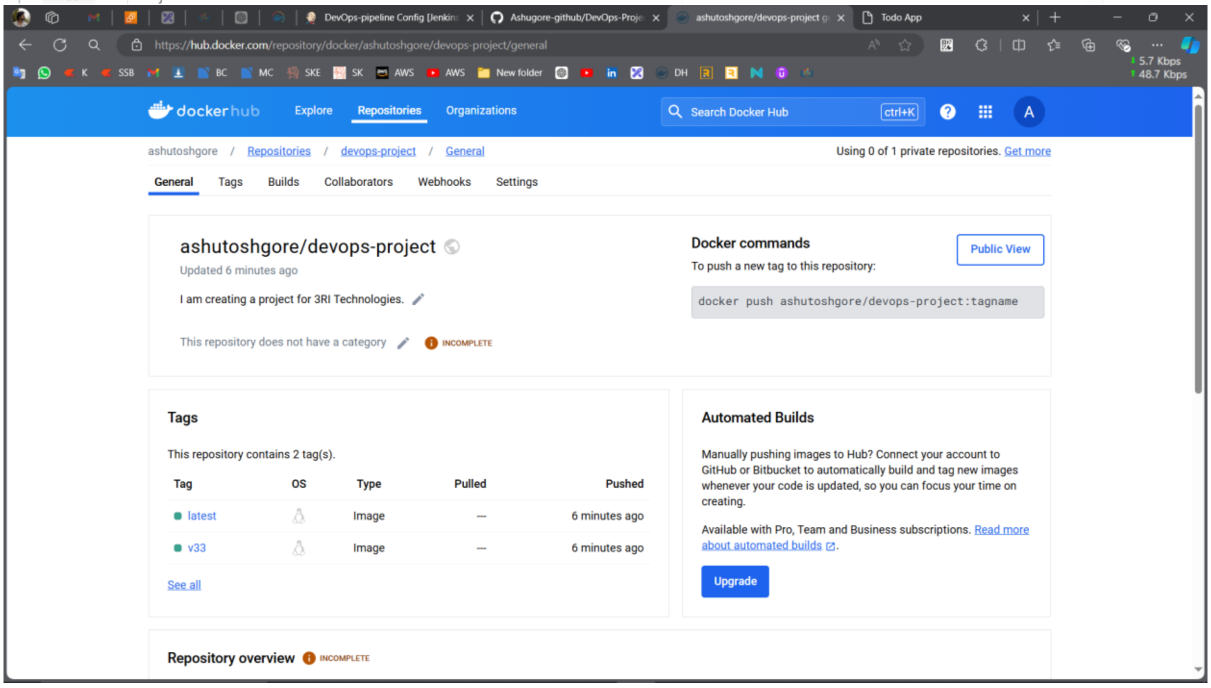
2. **Pushing image to Docker Hub**.

- First create credentials
- Secondly create a snippet to generate url.
- And now write commands to push images.

- 

Script ?

```
    stage("Build Image"){
        steps{
            sh 'docker image build -t ashutoshgore/devops-project:v$BUILD_ID .'
            sh 'docker image tag ashutoshgore/devops-project:v$BUILD_ID ashutoshgore/devops-project:latest'
        }
    }

    stage("Push Image to Dockerhub"){
        steps{
            withCredentials([usernamePassword(credentialsId: 'dockerhub', passwordVariable: 'pass', usernameVariable: 'user')]) {
                sh "docker login -u ${user} -p ${pass}"
                sh 'docker image push ashutoshgore/devops-project:v$BUILD_ID'
                sh 'docker image push ashutoshgore/devops-project:latest'
            }
        }
    }
```

- 
DevOps-pipeline Config [Jenkins]
Ashugore-github/DevOps-Project
ashutoshgore/devops-project
Todo App
https://hub.docker.com/repository/docker/ashutoshgore/devops-project/general
docker hub
Explore
Repositories
Organizations
Search Docker Hub
ctrl+K

ashutoshgore / Repositories / devops-project / General

Using 0 of 1 private repositories. Get more

General Tags Builds Collaborators Webhooks Settings

ashutoshgore/devops-project

Updated 6 minutes ago

I am creating a project for 3RI Technologies.

This repository does not have a category INCOMPLETE

Docker commands

To push a new tag to this repository:

Public View

```
docker push ashutoshgore/devops-project:tagname
```

Tags

This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	---	6 minutes ago
v33		Image	---	6 minutes ago

See all

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

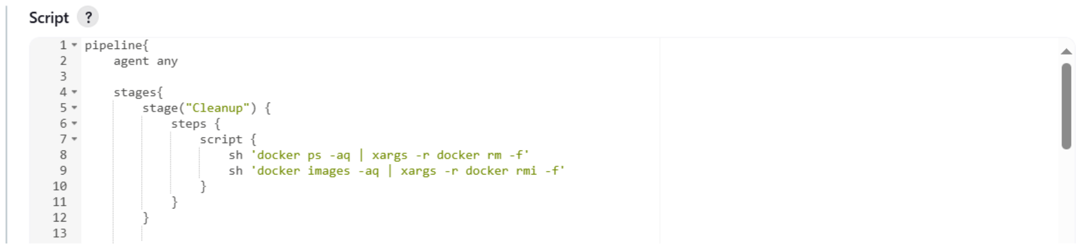
Available with Pro, Team and Business subscriptions. Read more about automated builds.

Upgrade

Repository overview INCOMPLETE

3. **Clean up**

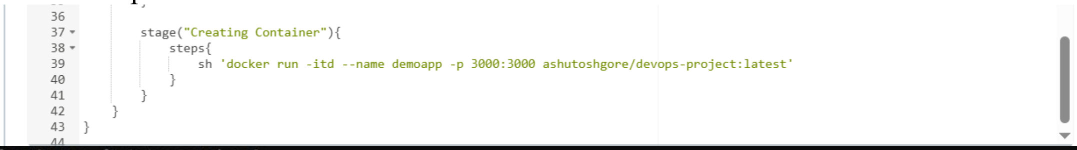
- To delete preinstall images and docker containers.

- 
Script ?

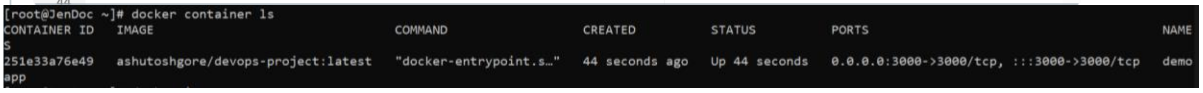
```
pipeline{
    agent any

    stages{
        stage("Cleanup") {
            steps {
                script {
                    sh 'docker ps -aq | xargs -r docker rm -f'
                    sh 'docker images -aq | xargs -r docker rmi -f'
                }
            }
        }
        |
```

Step 6: Create a Container and Access the application

1. Create a Container:

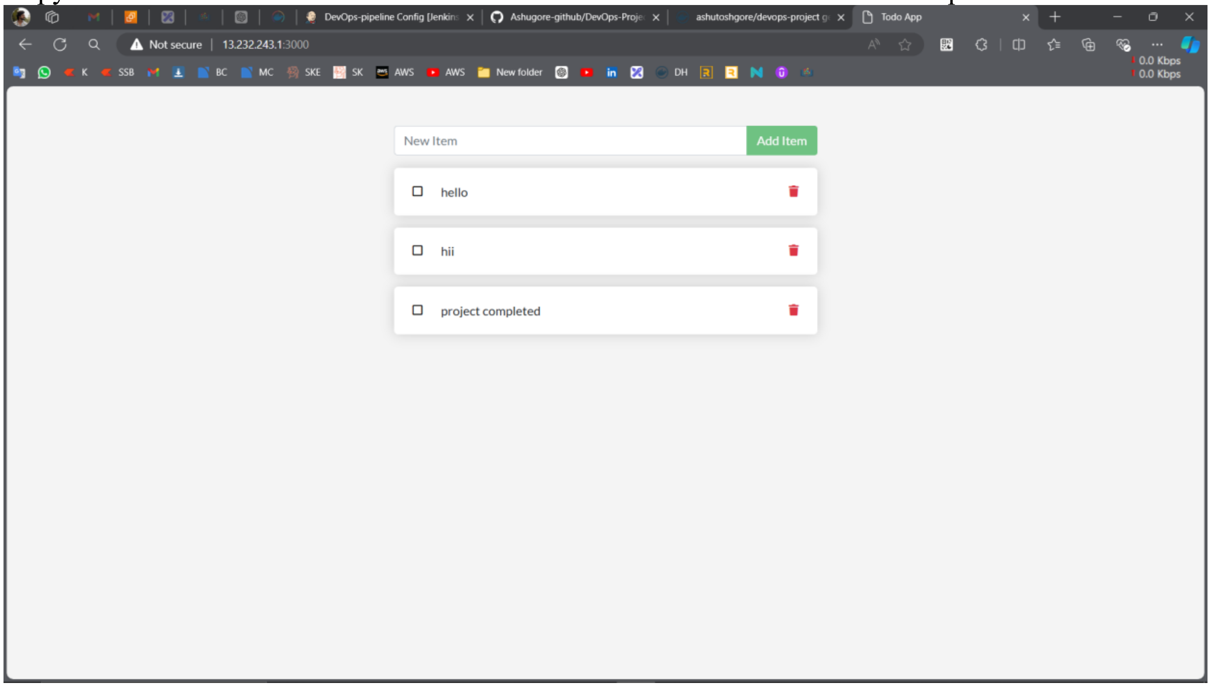
- I give the 'demoapp' name to the container and the image will be 'latest' and exposed on 3000:3000 port.
- 

```

        stage("Creating Container"){
            steps{
                sh 'docker run -itd --name demoapp -p 3000:3000 ashutoshgore/devops-project:latest'
            }
        }
    }
}
```

- 

```
[root@JenDoc ~]# docker container ls
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS                                       NAME
S
251e33a76e49   ashutoshgore/devops-project:latest   "docker-entrypoint.s…"   44 seconds ago   Up 44 seconds   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   demo
app
```

2. Access the application:

- Copy the instance IP and with 3000 and run it on browser. <instanceIP:containerport>.
- 
DevOps-pipeline Config [Jenkins]
Ashugore-github/DevOps-Proje
ashutoshgore/devops-project
Todo App
Not secure
13.232.243.1:3000
New Item
Add Item
hello
hii
project completed