ToDo-List App

Agenda

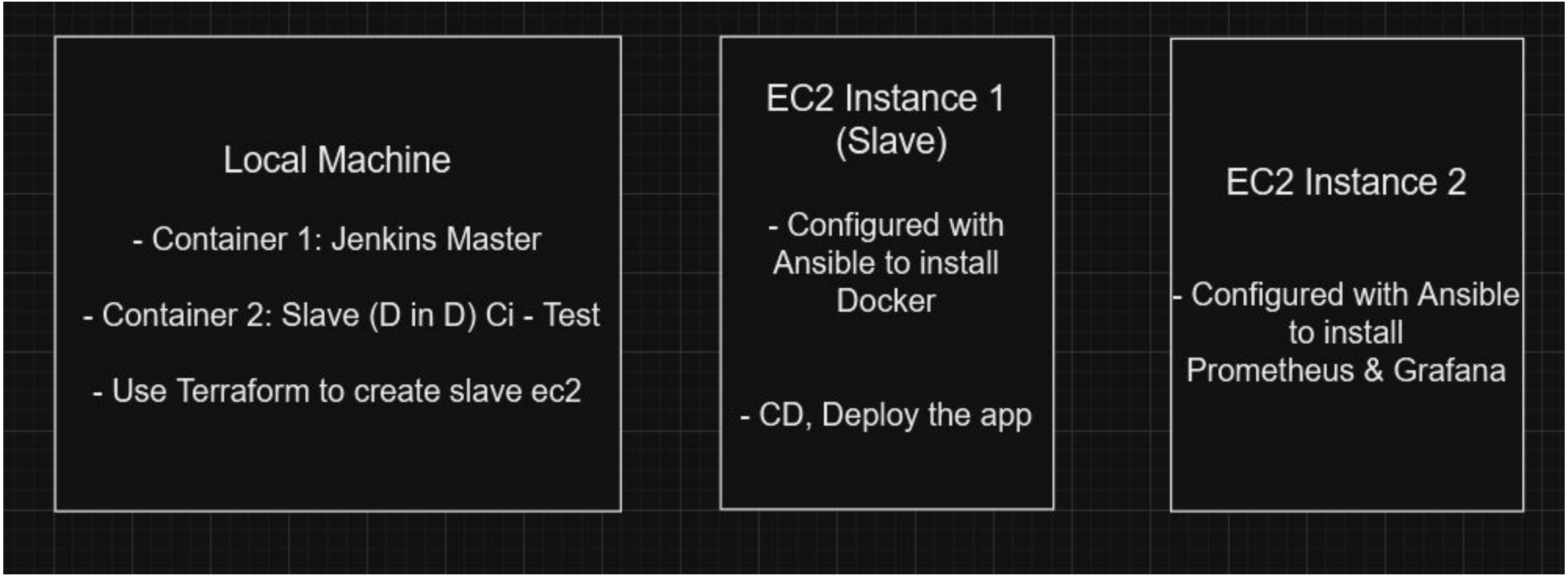
List of tools that are used in the app

- App Architecture
- Docker
- Terraform & AWS
- Ansible
- Jenkins & Slack

Git repo:

https://github.com/MohamedMSayed07/depi-project/

App Architecture



Local Machine

- Container 1: Jenkins Master

- Container 2: Slave (D in D) Ci - Test

- Use Terraform to create slave ec2

EC2 Instance 1 (Slave)

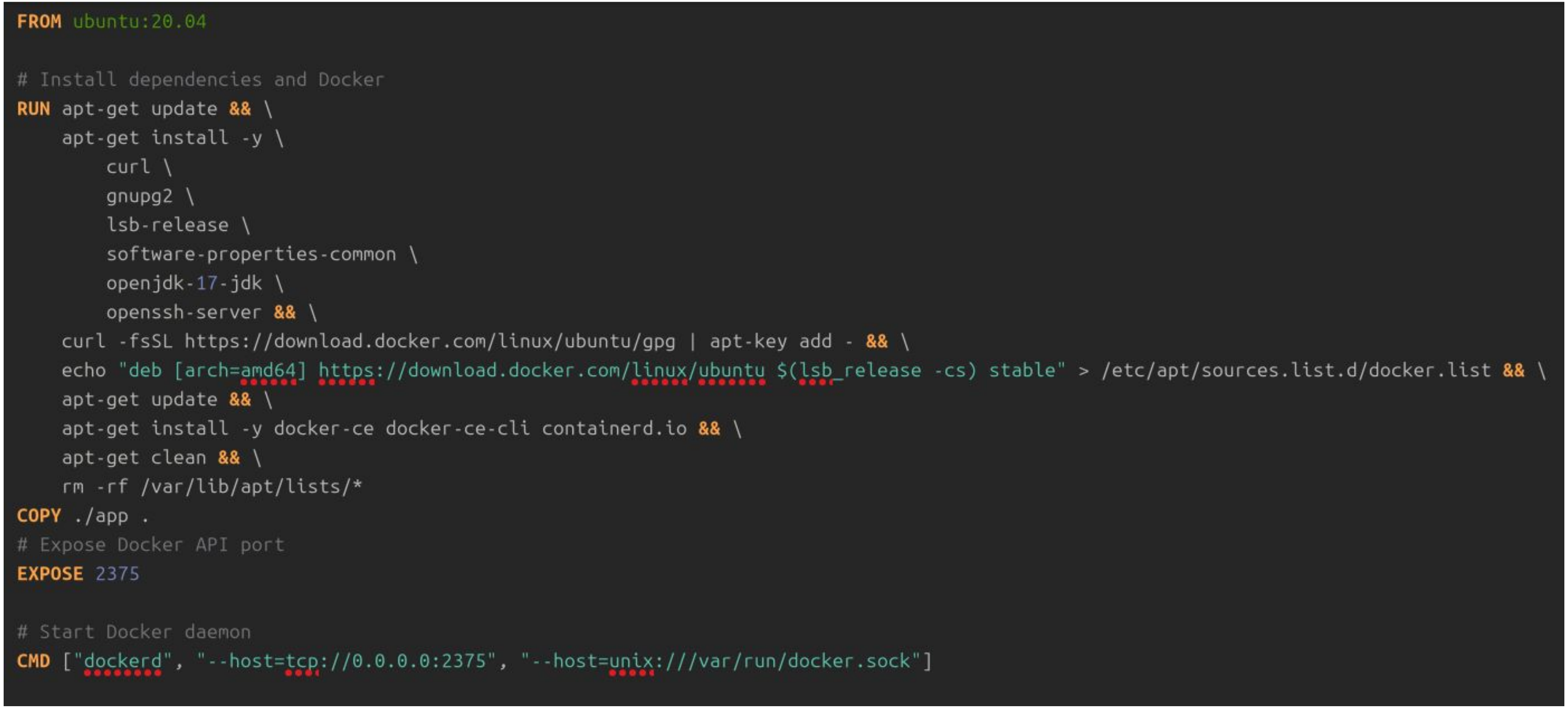
- Configured with Ansible to install Docker

- CD, Deploy the app

EC2 Instance 2

- Configured with Ansible to install Prometheus & Grafana

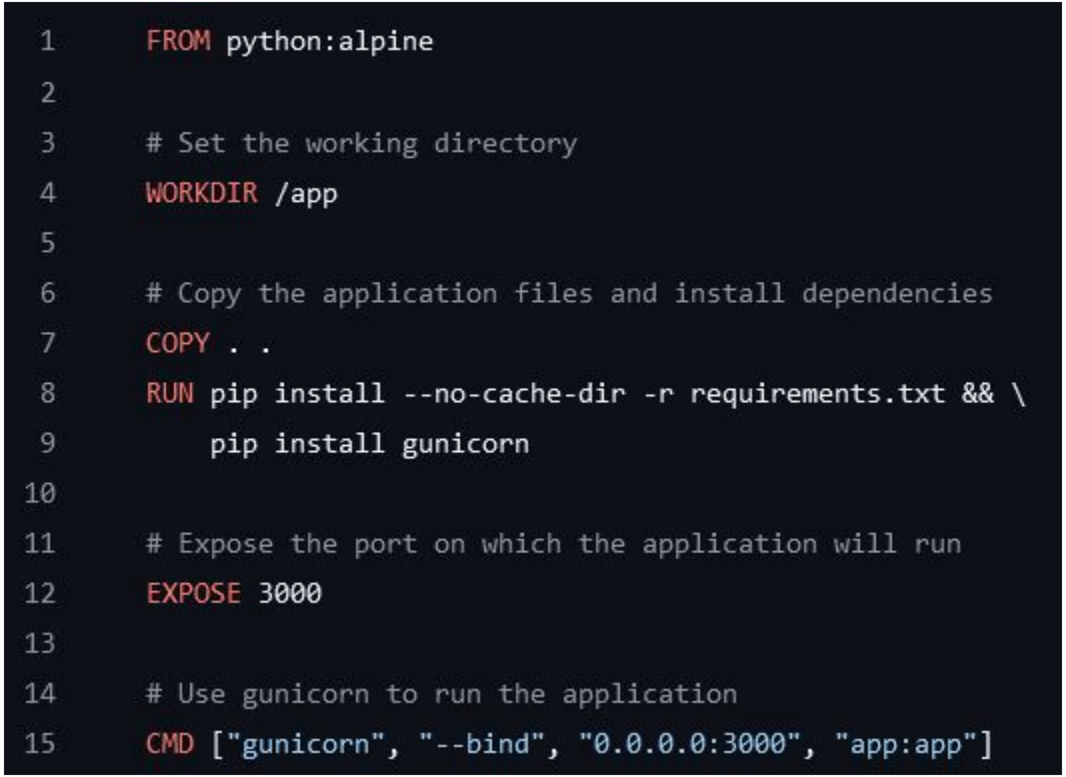
Customized DinD Dockerfile for the Jenkins-slave :



```
FROM ubuntu:20.04

# Install dependencies and Docker
RUN apt-get update && \
    apt-get install -y \
        curl \
        gnupg2 \
        lsb-release \
        software-properties-common \
        openjdk-17-jdk \
        openssh-server && \
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add - && \
    echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list && \
    apt-get update && \
    apt-get install -y docker-ce docker-ce-cli containerd.io && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
COPY ./app .
# Expose Docker API port
EXPOSE 2375

# Start Docker daemon
CMD ["dockerd", "--host=tcp://0.0.0.0:2375", "--host=unix:///var/run/docker.sock"]
```

Docker

- The app is containerized with **Docker**
- It depends on a **Docker Image** that is built and push on **Docker Hub**
- The used base image is **“python:alpine”**
- The app will be accessed on **port 3000**
- The entry point of the container for running the app is **“gunicorn”**

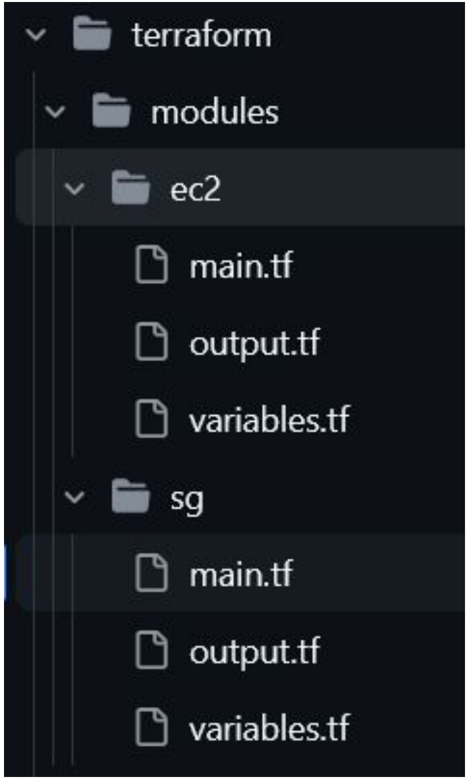


```
FROM python:alpine

# Set the working directory
WORKDIR /app

# Copy the application files and install dependencies
COPY . .
RUN pip install --no-cache-dir -r requirements.txt && \
    pip install gunicorn

# Expose the port on which the application will run
EXPOSE 3000

# Use gunicorn to run the application
CMD ["gunicorn", "--bind", "0.0.0.0:3000", "app:app"]
```

Terraform & AWS

Two modules were used in creating the infrastructure:

- EC2: for defining the EC2
- sg: for defining the security groups

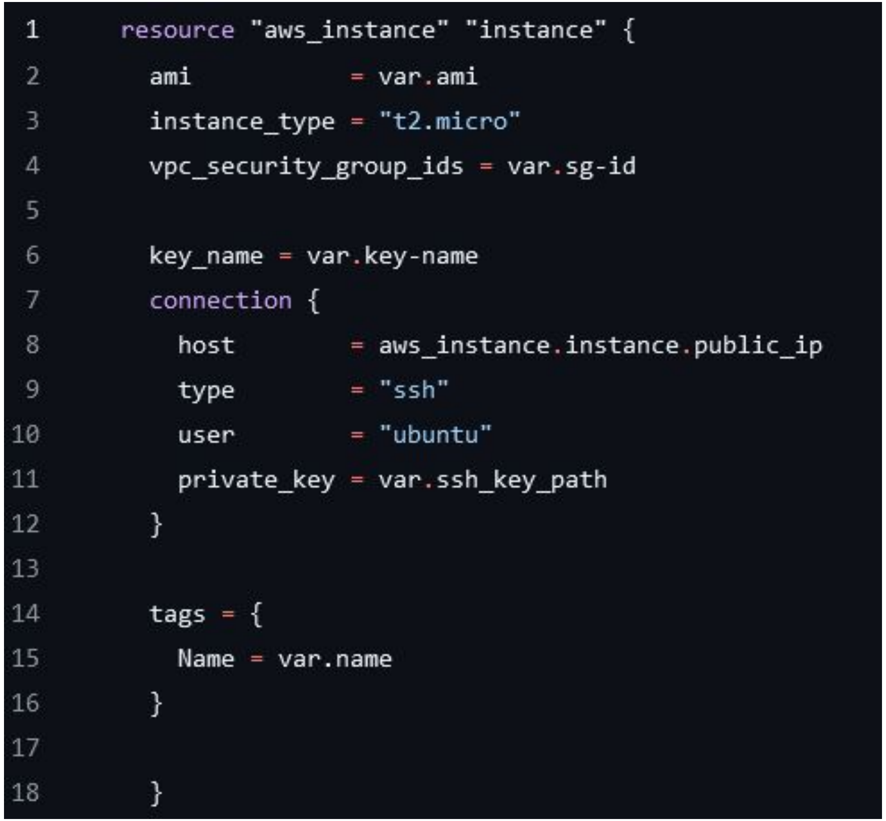


```
terraform
  modules
    ec2
      main.tf
      output.tf
      variables.tf
    sg
      main.tf
      output.tf
      variables.tf
```

Terraform & AWS: EC2

The “ec2” module is used to create 2 instances on AWS for the following:

- Instance 1: as a deployment environment for running the app
- Instance 2: as a monitoring environment for Prometheus and Grafana



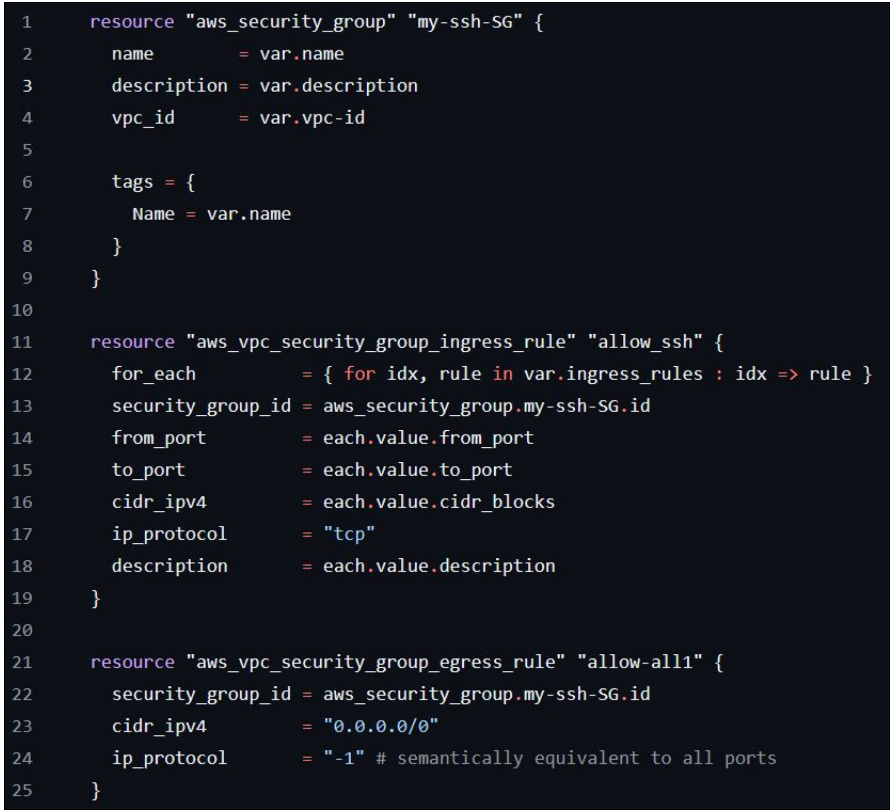
```
resource "aws_instance" "instance" {
  ami              = var.ami
  instance_type = "t2.micro"
  vpc_security_group_ids = var.sg-id

  key_name = var.key-name
  connection {
    host        = aws_instance.instance.public_ip
    type        = "ssh"
    user        = "ubuntu"
    private_key = var.ssh_key_path
  }

  tags = {
    Name = var.name
  }

  }
```

Terraform & AWS: sg

The “sg” module is used to create 2 security groups for the two EC2s on AWS for the following:

- Security Group 1 “todo-app-sg”

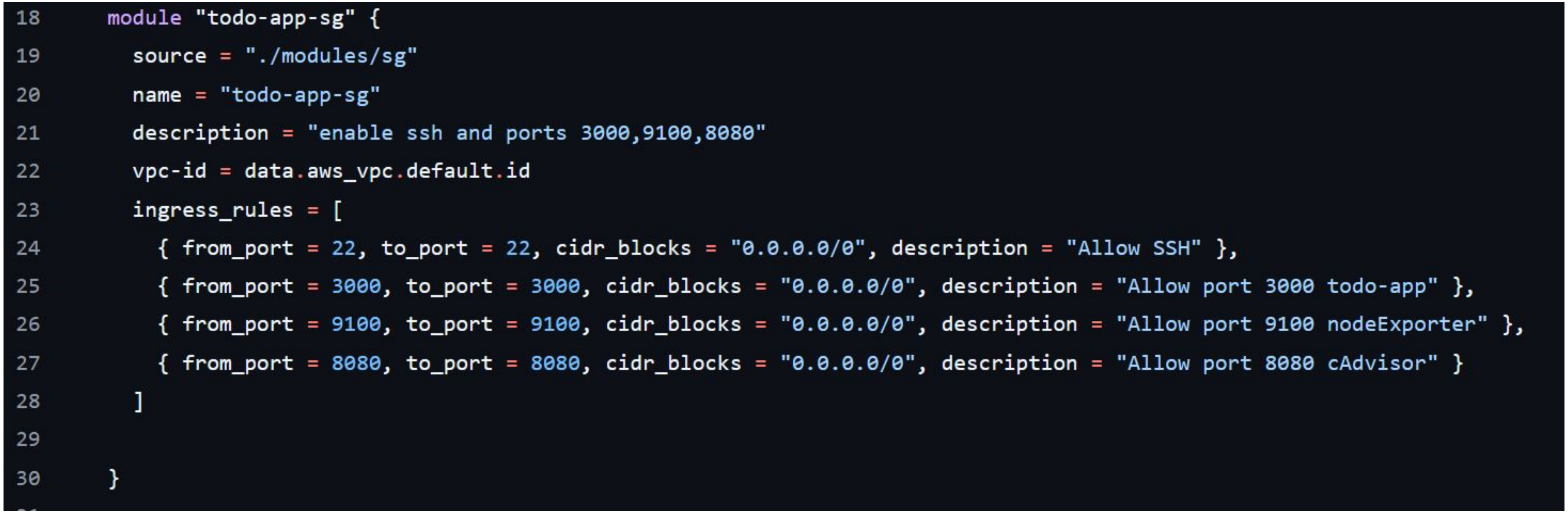
- Security Group 2 “prometheus-sg”



```
resource "aws_security_group" "my-ssh-SG" {
  name        = var.name
  description = var.description
  vpc_id      = var.vpc-id

  tags = {
    Name = var.name
  }
}

resource "aws_vpc_security_group_ingress_rule" "allow_ssh" {
  for_each          = { for idx, rule in var.ingress_rules : idx => rule }
  security_group_id = aws_security_group.my-ssh-SG.id
  from_port         = each.value.from_port
  to_port           = each.value.to_port
  cidr_ipv4         = each.value.cidr_blocks
  ip_protocol       = "tcp"
  description       = each.value.description
}

resource "aws_vpc_security_group_egress_rule" "allow-all1" {
  security_group_id = aws_security_group.my-ssh-SG.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1" # semantically equivalent to all ports
}
```

Terraform & AWS: sg

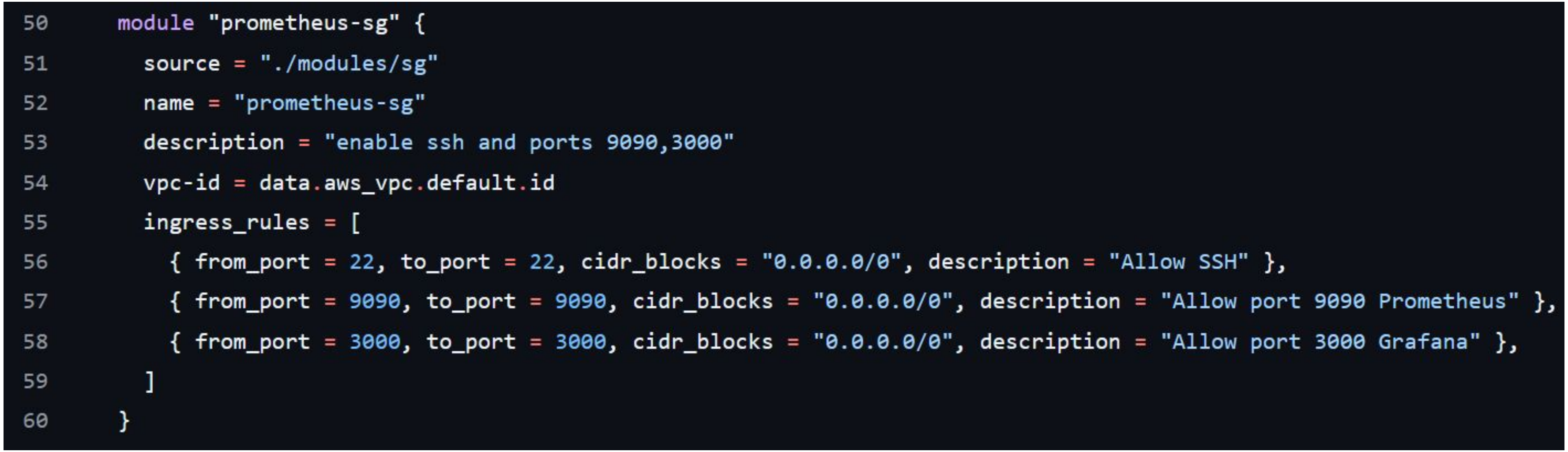
Security Group 1 “todo-app-sg” :



```
module "todo-app-sg" {
  source = "./modules/sg"
  name = "todo-app-sg"
  description = "enable ssh and ports 3000,9100,8080"
  vpc-id = data.aws_vpc.default.id
  ingress_rules = [
    { from_port = 22, to_port = 22, cidr_blocks = "0.0.0.0/0", description = "Allow SSH" },
    { from_port = 3000, to_port = 3000, cidr_blocks = "0.0.0.0/0", description = "Allow port 3000 todo-app" },
    { from_port = 9100, to_port = 9100, cidr_blocks = "0.0.0.0/0", description = "Allow port 9100 nodeExporter" },
    { from_port = 8080, to_port = 8080, cidr_blocks = "0.0.0.0/0", description = "Allow port 8080 cAdvisor" }
  ]

}
```

Terraform & AWS: sg

Security Group 2 “prometheus-sg” :



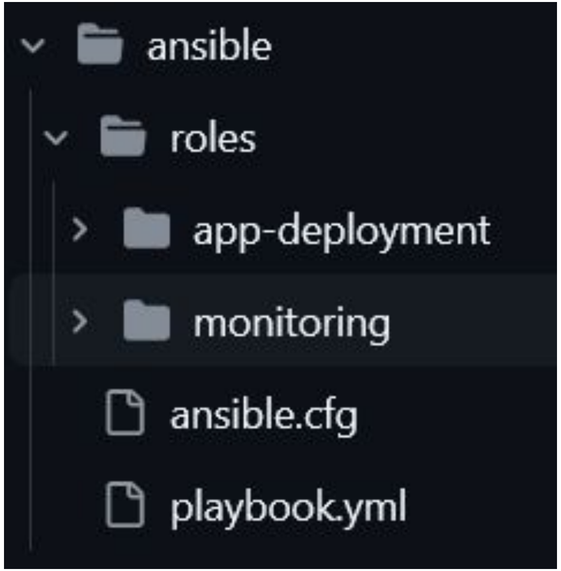
```
module "prometheus-sg" {
  source = "./modules/sg"
  name = "prometheus-sg"
  description = "enable ssh and ports 9090,3000"
  vpc-id = data.aws_vpc.default.id
  ingress_rules = [
    { from_port = 22, to_port = 22, cidr_blocks = "0.0.0.0/0", description = "Allow SSH" },
    { from_port = 9090, to_port = 9090, cidr_blocks = "0.0.0.0/0", description = "Allow port 9090 Prometheus" },
    { from_port = 3000, to_port = 3000, cidr_blocks = "0.0.0.0/0", description = "Allow port 3000 Grafana" },
  ]
}
```

Ansible

Two modules were used in configuring the EC2 instances:

- app-deployment:
  Includes the needed configurations for instance 1 (deploying the app)

- monitoring:
  Includes the needed configurations for instance 2 (monitoring the app)



```
ansible
  roles
    app-deployment
    monitoring
  ansible.cfg
  playbook.yml
```

Ansible: “app-deployment” module

Includes the needed tasks for configuring EC2 instance 1 on which the app will be deployed. The tasks are grouped in 3 categories to enable the following:

- Installing the needed tools for app-deployment
- Installing node-Exporter
- Installing cAdvisor

Ansible: “app-deployment” module, cont’d

Installing the needed tools for app-deployment will be done using the following tasks:

Task 1: - name: Update apt repository

Task 2: - name: Download Docker installation script

Task 3: - name: Run Docker installation script

Task 4: - name: Start and enable Docker service

Task 5: - name: Add the current user to the docker group

Ansible: “app-deployment” module, cont’d

Installing node-Exporter will be done using the following tasks:

Task 8: - name: Create a user for node exporter

Task 9: - name: Download Node Exporter using curl

Task 10: - name: Extract Node Exporter

Task 11: - name: Create Node Exporter systemd service file

Task 12: - name: Reload systemd daemon to register Node Exporter service

Task 13: - name: Enable and start Node Exporter service

Ansible: "app-deployment" module, cont'd

Installing cAdvisor will be done using the following tasks:

Task 6: - name: Install Python 3 and pip

Task 7: - name: Install Docker SDK for Python using apt

Task 14: - name: Pull cAdvisor image

Task 15: - name: Run cAdvisor container

Ansible: “monitoring” module

Includes the needed tasks for configuring the EC2 instance 2 on which the app will be monitored. The tasks are grouped in 2 categories to enable the following:

- **Installing Prometheus**

- **Installing Grafana**

Ansible: “monitoring” module, cont’d

Installing Prometheus will be done using the following tasks:

Task 1: - name: Update apt repository

Task 2: - name: Create Prometheus user with no shell access

Task 3: - name: Create Prometheus directories

Task 4: - name: Download Prometheus archive with curl

Task 5: - name: Extract Prometheus archive

Task 6: - name: Move Prometheus binaries

Task 7: - name: Set ownership for binaries

Ansible: “monitoring” module, cont’d

Installing Prometheus will be done using the following tasks:

Task 8: - name: Remove existing console_libraries directory if it exists

Task 9: - name: Remove existing consoles directory if it exists

Task 10: - name: Move Prometheus configuration and console libraries

Task 11: - name: Set ownership for Prometheus directories

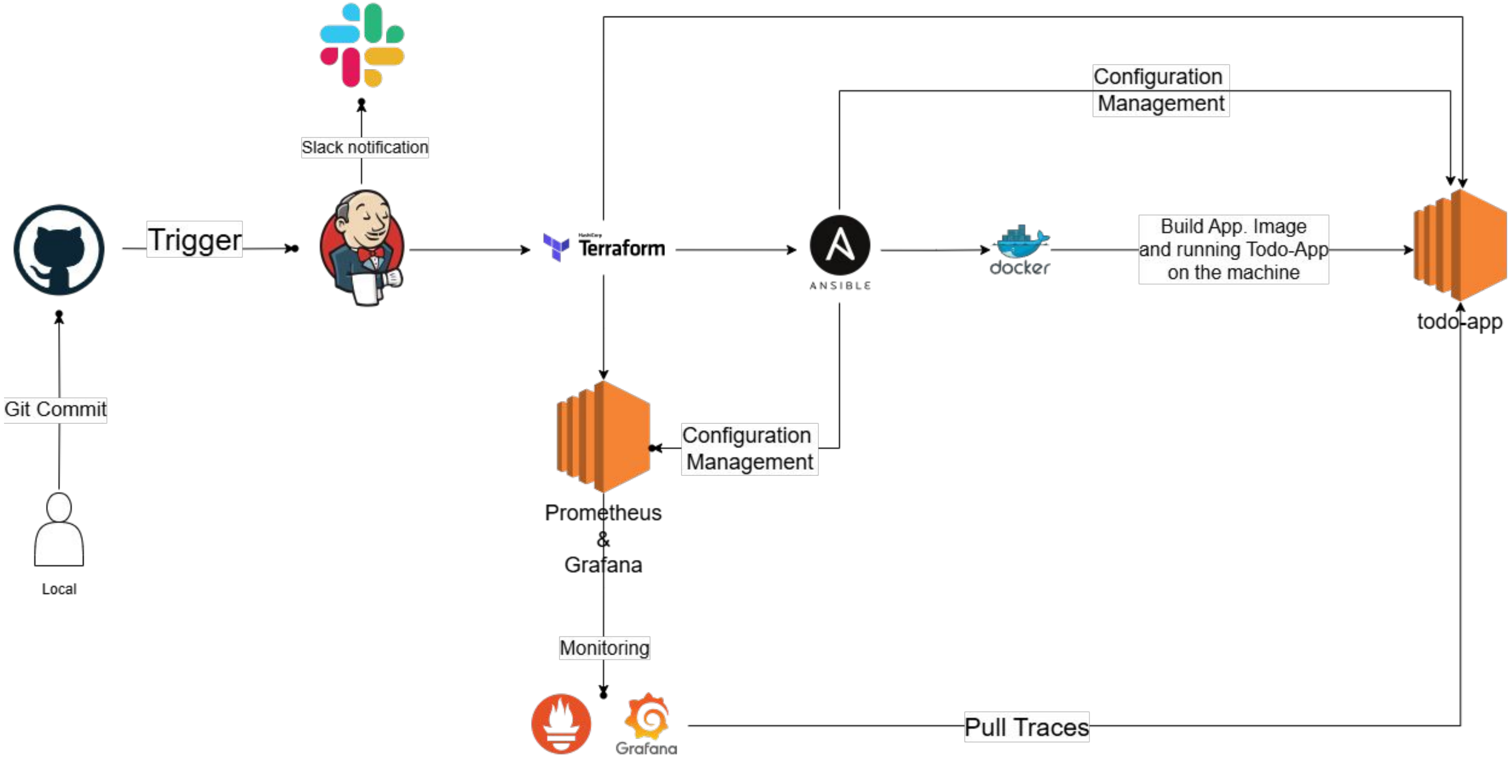
Task 12: - name: Create Prometheus systemd service

Task 13: - name: Reload systemd and start Prometheus service

Jenkins & Slack

- Jenkins is the used CI/CD tool for automating tasks and deploying the app.
- The pipeline is triggered using the **github webhook** according to the **changes on the github repository**
- The github repository includes the **Jenkins file** that includes the stages of the **CI/CD pipeline**.
- After the pipeline is finished, a notification message is sent to a **Slack Channel** to show whether the pipeline succeeded or failed.

Jenkins & Slack

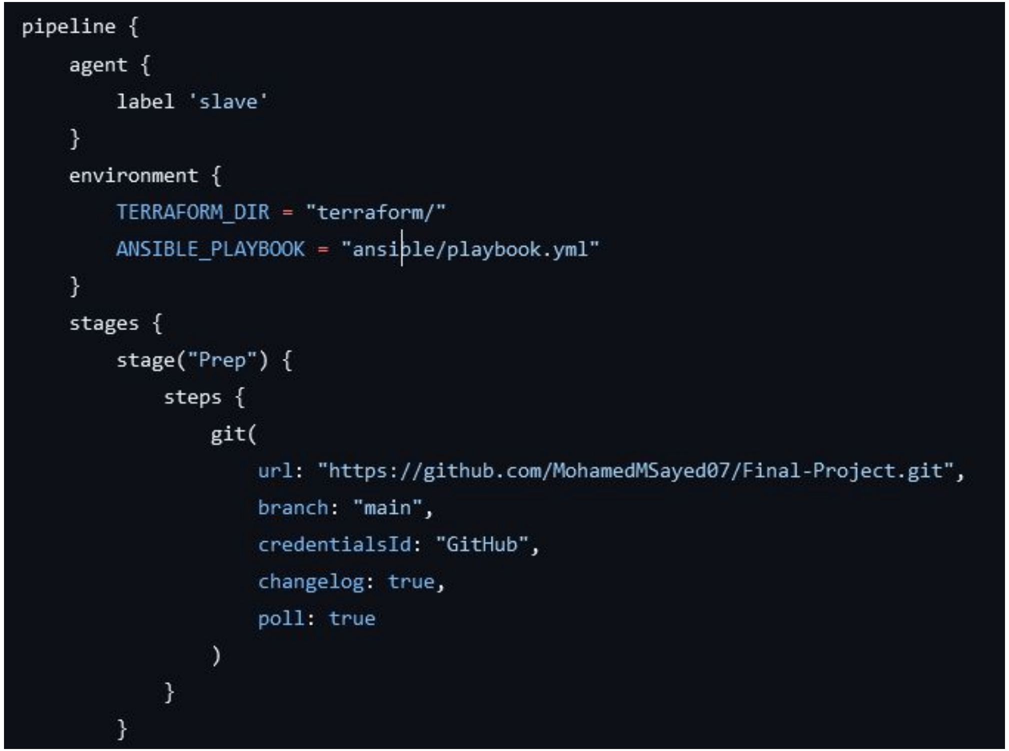


Jenkins & Slack, cont’d

Pipeline stages are as follows:

Stage 1 “Prep”:

Clones the github repo.



```
pipeline {
    agent {
        label 'slave'
    }
    environment {
        TERRAFORM_DIR = "terraform/"
        ANSIBLE_PLAYBOOK = "ansible/playbook.yml"
    }
    stages {
        stage("Prep") {
            steps {
                git(
                    url: "https://github.com/MohamedMSayed07/Final-Project.git",
                    branch: "main",
                    credentialsId: "GitHub",
                    changelog: true,
                    poll: true
                )
            }
        }
```

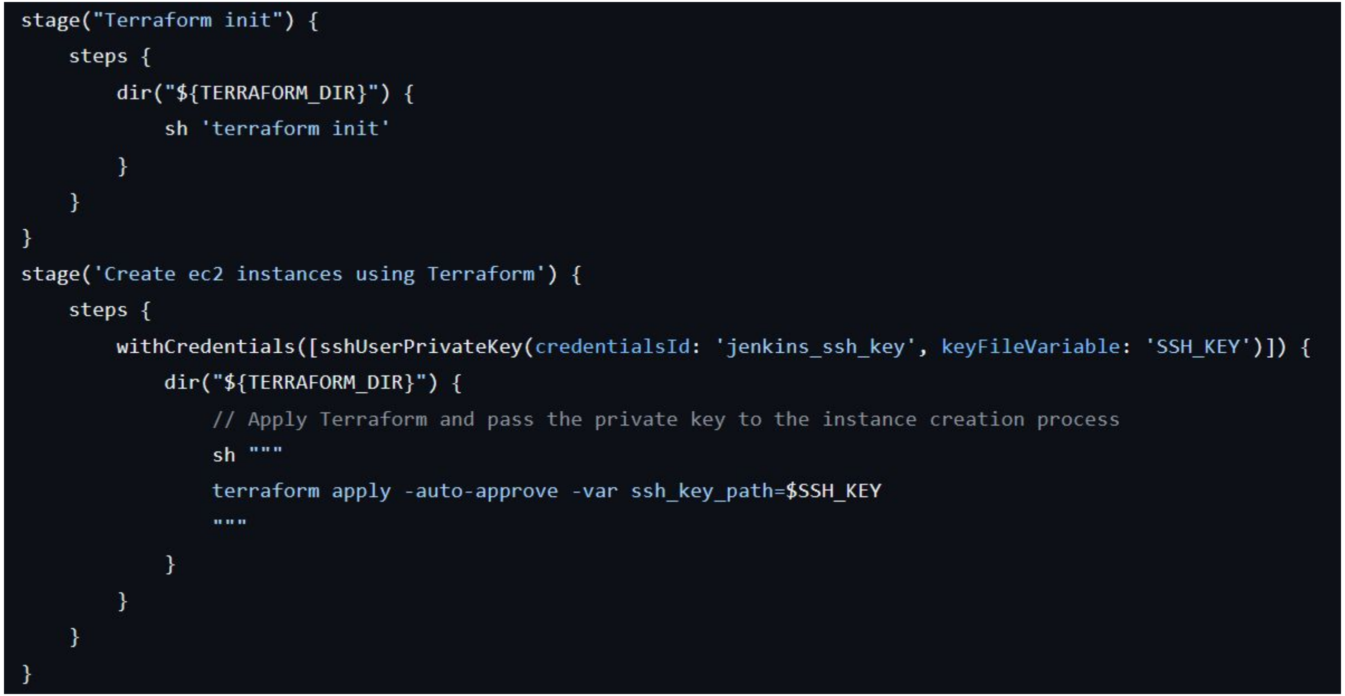
Jenkins & Slack, cont’d

Stage 2

“Terraform init”

Stage 3

“Create ec2 instances using Terraform”:

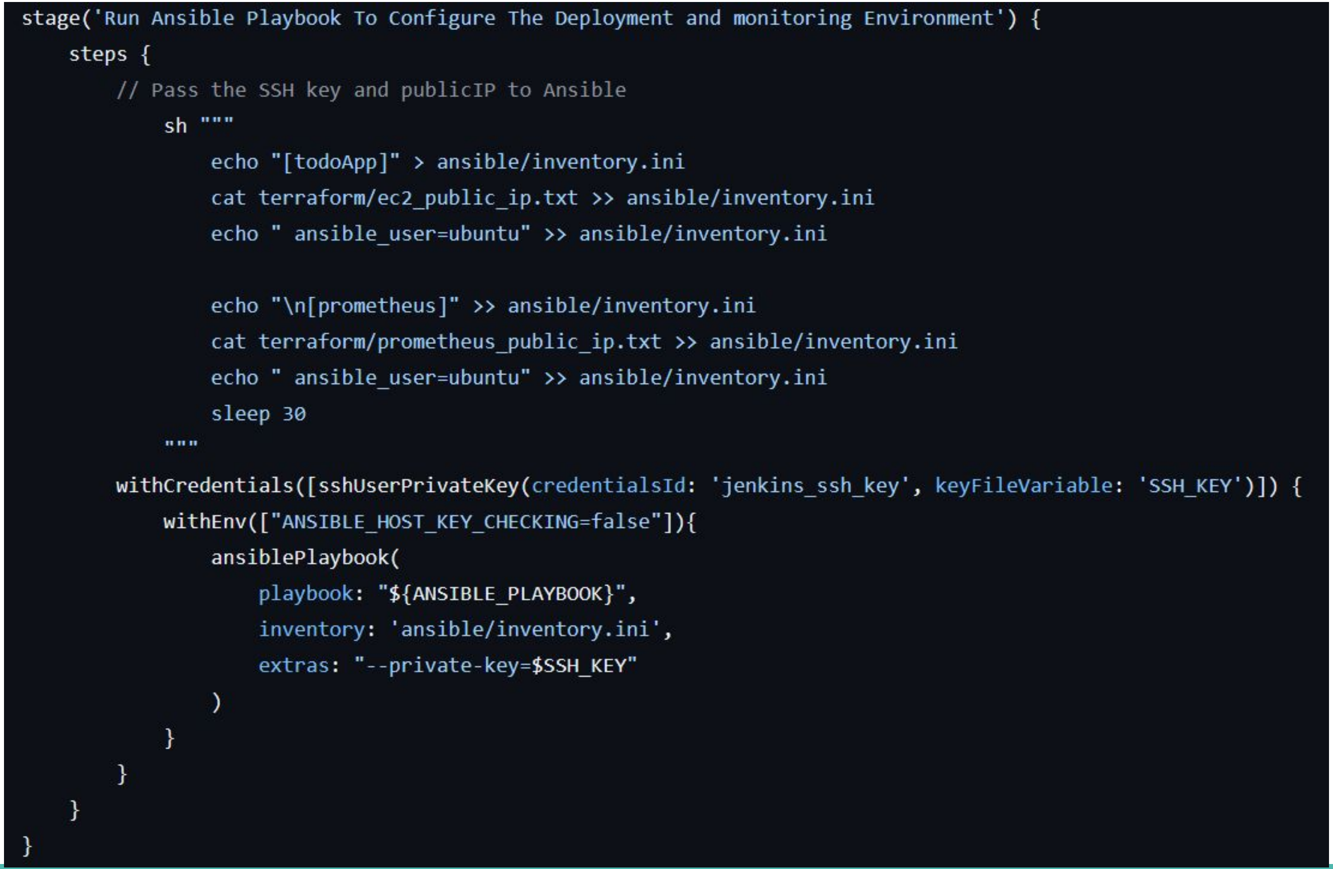


```
stage("Terraform init") {
    steps {
        dir("${TERRAFORM_DIR}") {
            sh 'terraform init'
        }
    }
}
stage('Create ec2 instances using Terraform') {
    steps {
        withCredentials([sshUserPrivateKey(credentialsId: 'jenkins_ssh_key', keyFileVariable: 'SSH_KEY')]) {
            dir("${TERRAFORM_DIR}") {
                // Apply Terraform and pass the private key to the instance creation process
                sh """
                terraform apply -auto-approve -var ssh_key_path=$SSH_KEY
                """
            }
        }
    }
}
```

Jenkins & Slack, cont’d

Stage 4

“Run Ansible Playbook To Configure The Deployment and monitoring Environment ”

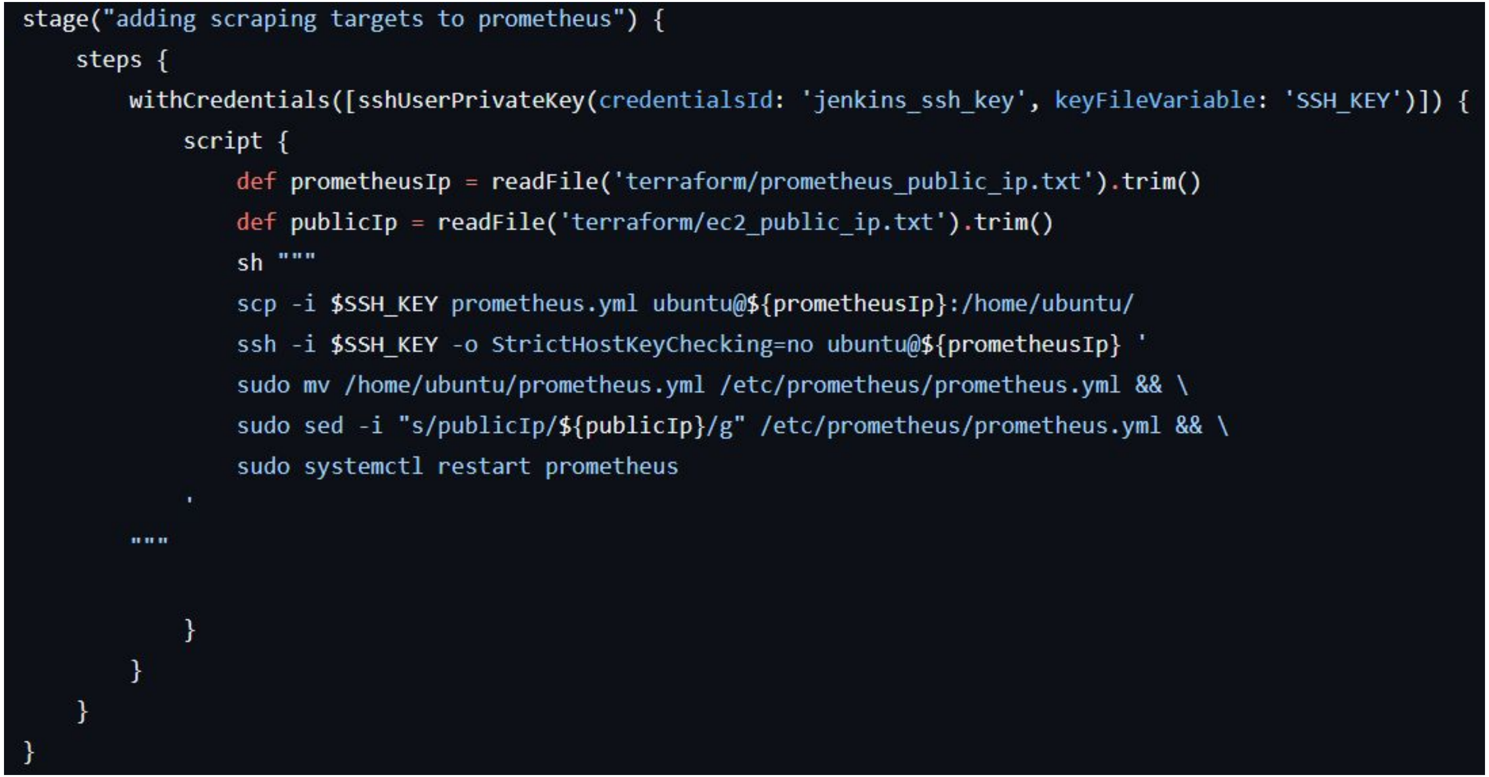


```
stage('Run Ansible Playbook To Configure The Deployment and monitoring Environment') {
    steps {
        // Pass the SSH key and publicIP to Ansible
            sh """
                echo "[todoApp]" > ansible/inventory.ini
                cat terraform/ec2_public_ip.txt >> ansible/inventory.ini
                echo " ansible_user=ubuntu" >> ansible/inventory.ini

                echo "\n[prometheus]" >> ansible/inventory.ini
                cat terraform/prometheus_public_ip.txt >> ansible/inventory.ini
                echo " ansible_user=ubuntu" >> ansible/inventory.ini
                sleep 30
            """
        withCredentials([sshUserPrivateKey(credentialsId: 'jenkins_ssh_key', keyFileVariable: 'SSH_KEY')]) {
            withEnv(["ANSIBLE_HOST_KEY_CHECKING=false"]){
                ansiblePlaybook(
                    playbook: "${ANSIBLE_PLAYBOOK}",
                    inventory: 'ansible/inventory.ini',
                    extras: "--private-key=$SSH_KEY"
                )
            }
        }
    }
}
```

Jenkins & Slack, cont’d

Stage 5

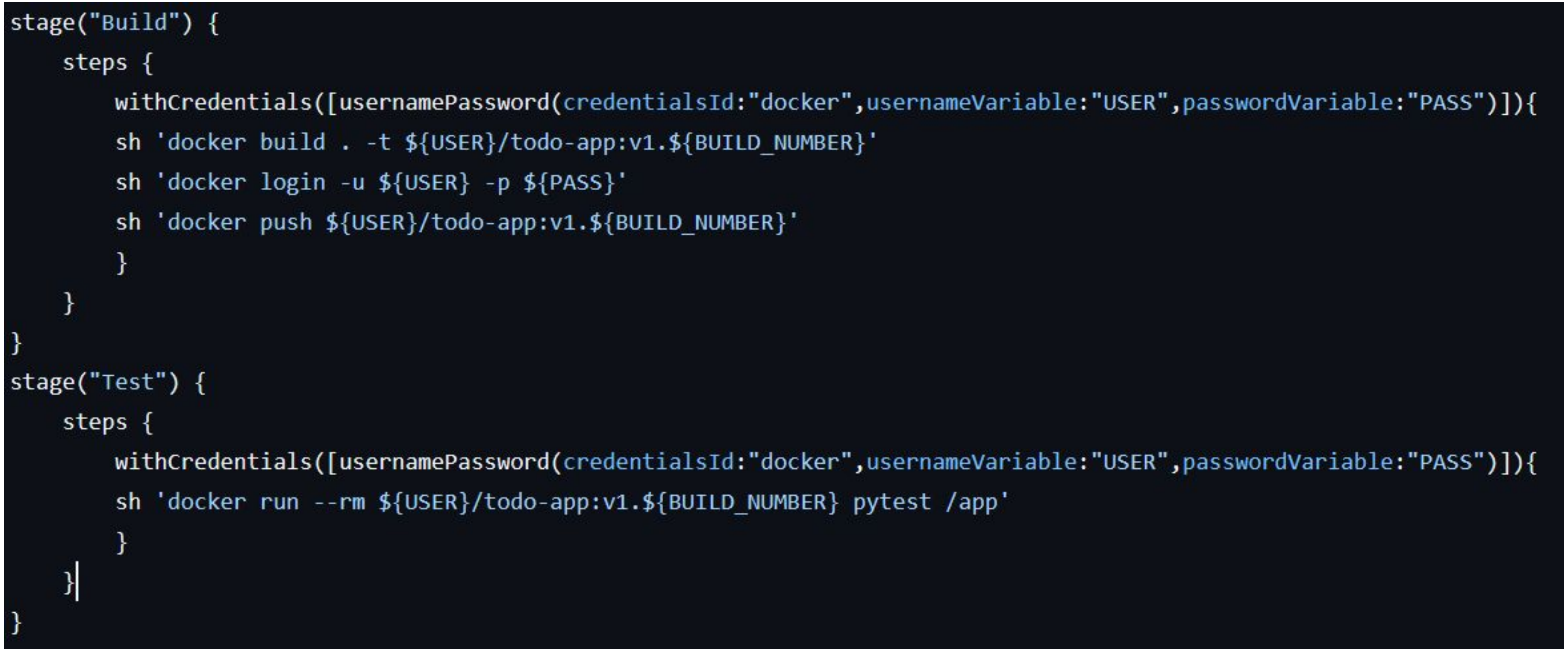
“ Adding scraping targets to prometheus “



```
stage("adding scraping targets to prometheus") {
    steps {
        withCredentials([sshUserPrivateKey(credentialsId: 'jenkins_ssh_key', keyFileVariable: 'SSH_KEY')]) {
            script {
                def prometheusIp = readFile('terraform/prometheus_public_ip.txt').trim()
                def publicIp = readFile('terraform/ec2_public_ip.txt').trim()
                sh """
                scp -i $SSH_KEY prometheus.yml ubuntu@${prometheusIp}:/home/ubuntu/
                ssh -i $SSH_KEY -o StrictHostKeyChecking=no ubuntu@${prometheusIp} '
                sudo mv /home/ubuntu/prometheus.yml /etc/prometheus/prometheus.yml && \
                sudo sed -i "s/publicIp/${publicIp}/g" /etc/prometheus/prometheus.yml && \
                sudo systemctl restart prometheus
            '
        """

            }
        }
    }
}
```

Jenkins & Slack, cont’d

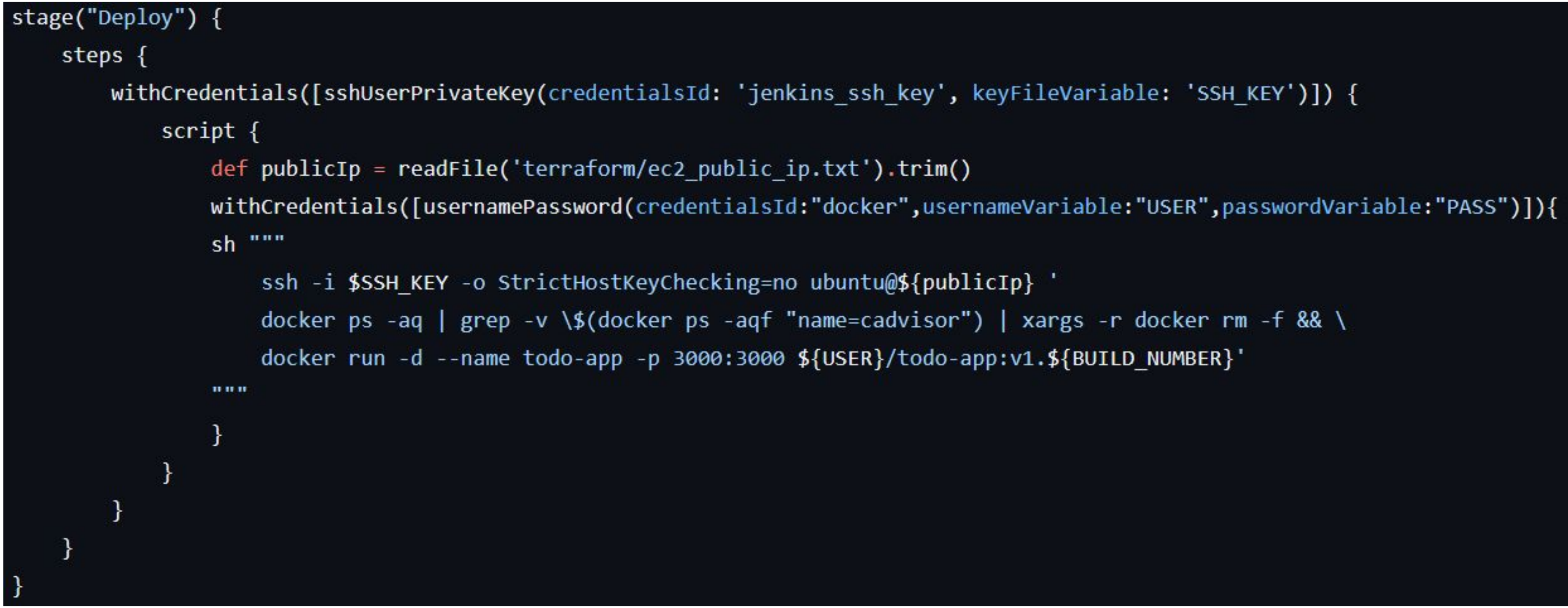
Stage 6 “Build” & Stage 7 “Test”



```
stage("Build") {
    steps {
        withCredentials([usernamePassword(credentialsId:"docker",usernameVariable:"USER",passwordVariable:"PASS")]){
        sh 'docker build . -t ${USER}/todo-app:v1.${BUILD_NUMBER}'
        sh 'docker login -u ${USER} -p ${PASS}'
        sh 'docker push ${USER}/todo-app:v1.${BUILD_NUMBER}'
        }
    }
}
stage("Test") {
    steps {
        withCredentials([usernamePassword(credentialsId:"docker",usernameVariable:"USER",passwordVariable:"PASS")]){
        sh 'docker run --rm ${USER}/todo-app:v1.${BUILD_NUMBER} pytest /app'
        }
    }
}
```

Jenkins & Slack, cont’d

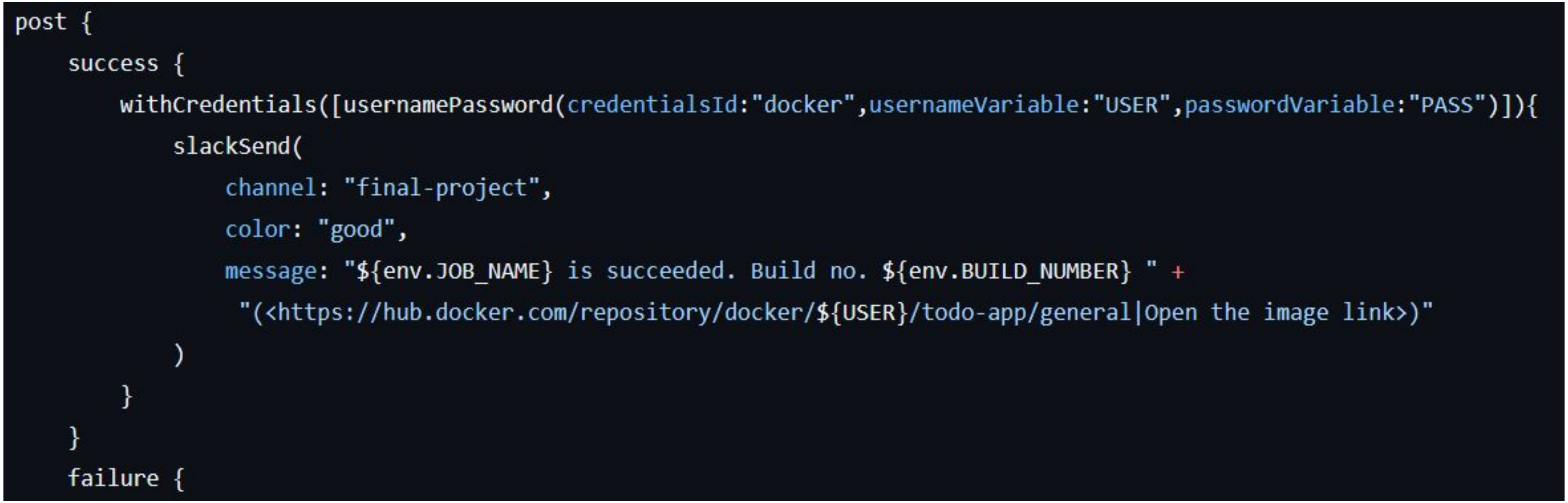
Stage 8 “Deploy”



```
stage("Deploy") {
    steps {
        withCredentials([sshUserPrivateKey(credentialsId: 'jenkins_ssh_key', keyFileVariable: 'SSH_KEY')]) {
            script {
                def publicIp = readFile('terraform/ec2_public_ip.txt').trim()
                withCredentials([usernamePassword(credentialsId:"docker",usernameVariable:"USER",passwordVariable:"PASS")]){
                sh """
                    ssh -i $SSH_KEY -o StrictHostKeyChecking=no ubuntu@${publicIp} '
                    docker ps -aq | grep -v \$(docker ps -aqf "name=cadvisor") | xargs -r docker rm -f && \
                    docker run -d --name todo-app -p 3000:3000 ${USER}/todo-app:v1.${BUILD_NUMBER}'
                """
                }
            }
        }
    }
}
```

Jenkins & Slack, cont’d

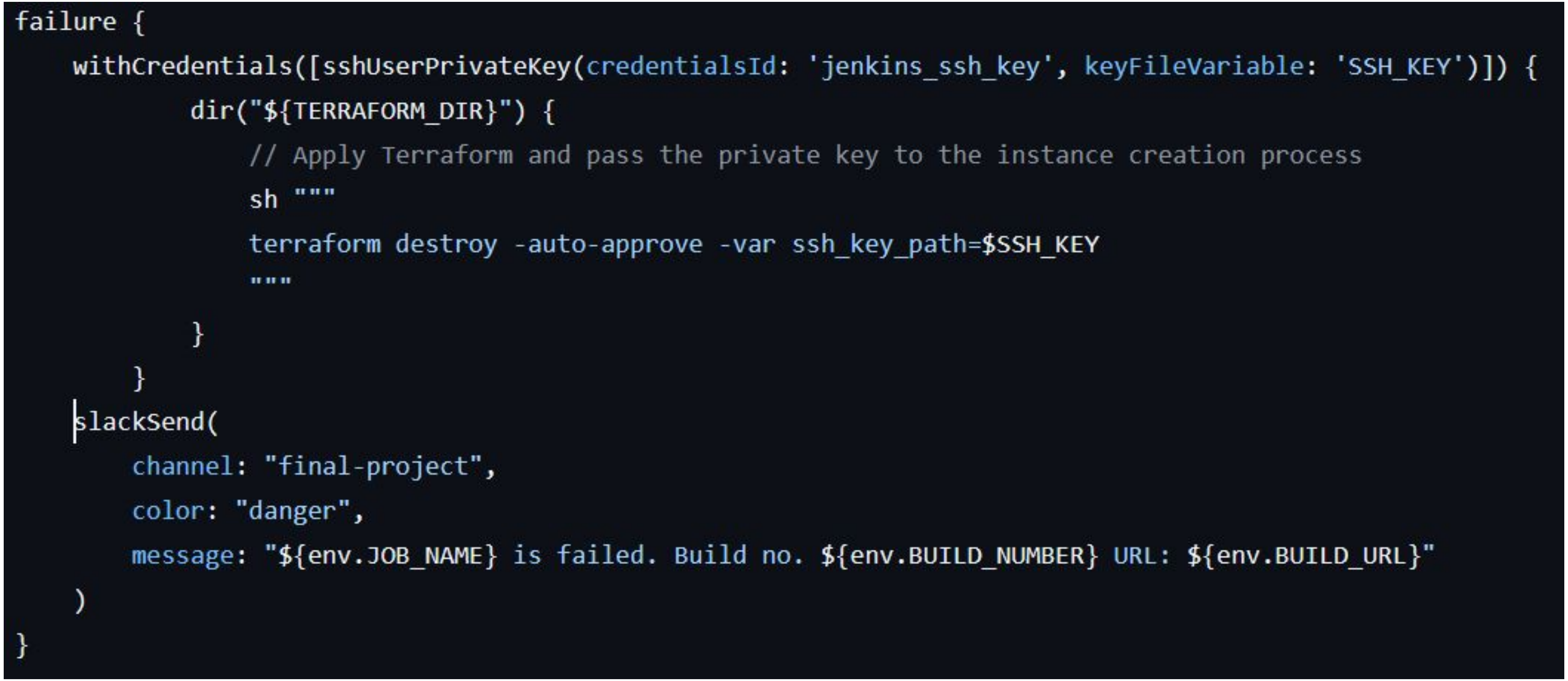
Slack Integration



```
post {
    success {
        withCredentials([usernamePassword(credentialsId:"docker",usernameVariable:"USER",passwordVariable:"PASS")]){
            slackSend(
                channel: "final-project",
                color: "good",
                message: "${env.JOB_NAME} is succeeded. Build no. ${env.BUILD_NUMBER} " +
                 "(<https://hub.docker.com/repository/docker/${USER}/todo-app/general|Open the image link>)"
            )
        }
    }
    failure {
```

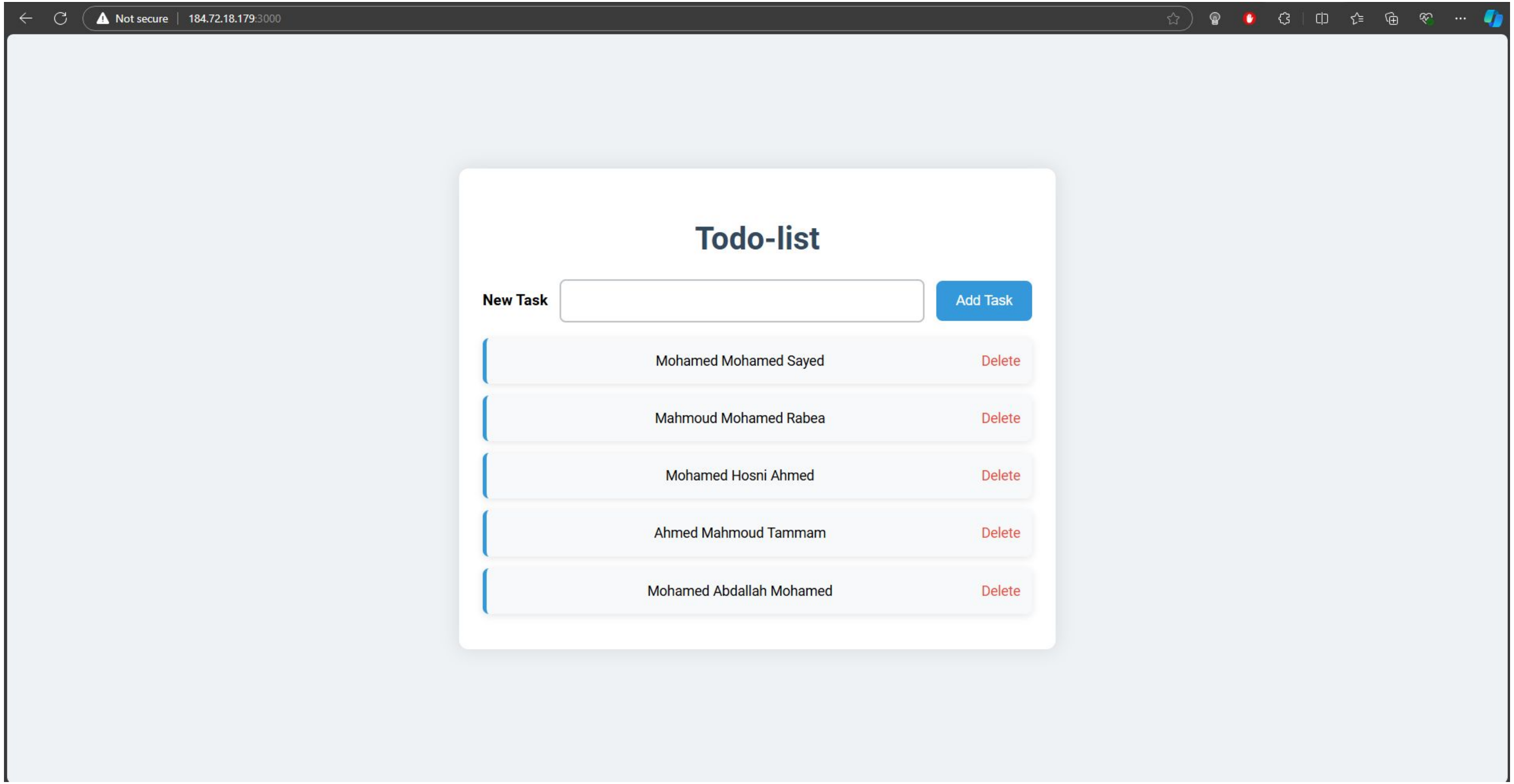
Jenkins & Slack, cont'd

Slack Integration, cont'd



```
failure {
    withCredentials([sshUserPrivateKey(credentialsId: 'jenkins_ssh_key', keyFileVariable: 'SSH_KEY')]) {
            dir("${TERRAFORM_DIR}") {
                // Apply Terraform and pass the private key to the instance creation process
                sh """
                terraform destroy -auto-approve -var ssh_key_path=$SSH_KEY
                """

            }
        }
    slackSend(
        channel: "final-project",
        color: "danger",
        message: "${env.JOB_NAME} is failed. Build no. ${env.BUILD_NUMBER} URL: ${env.BUILD_URL}"
    )
}
```

Results

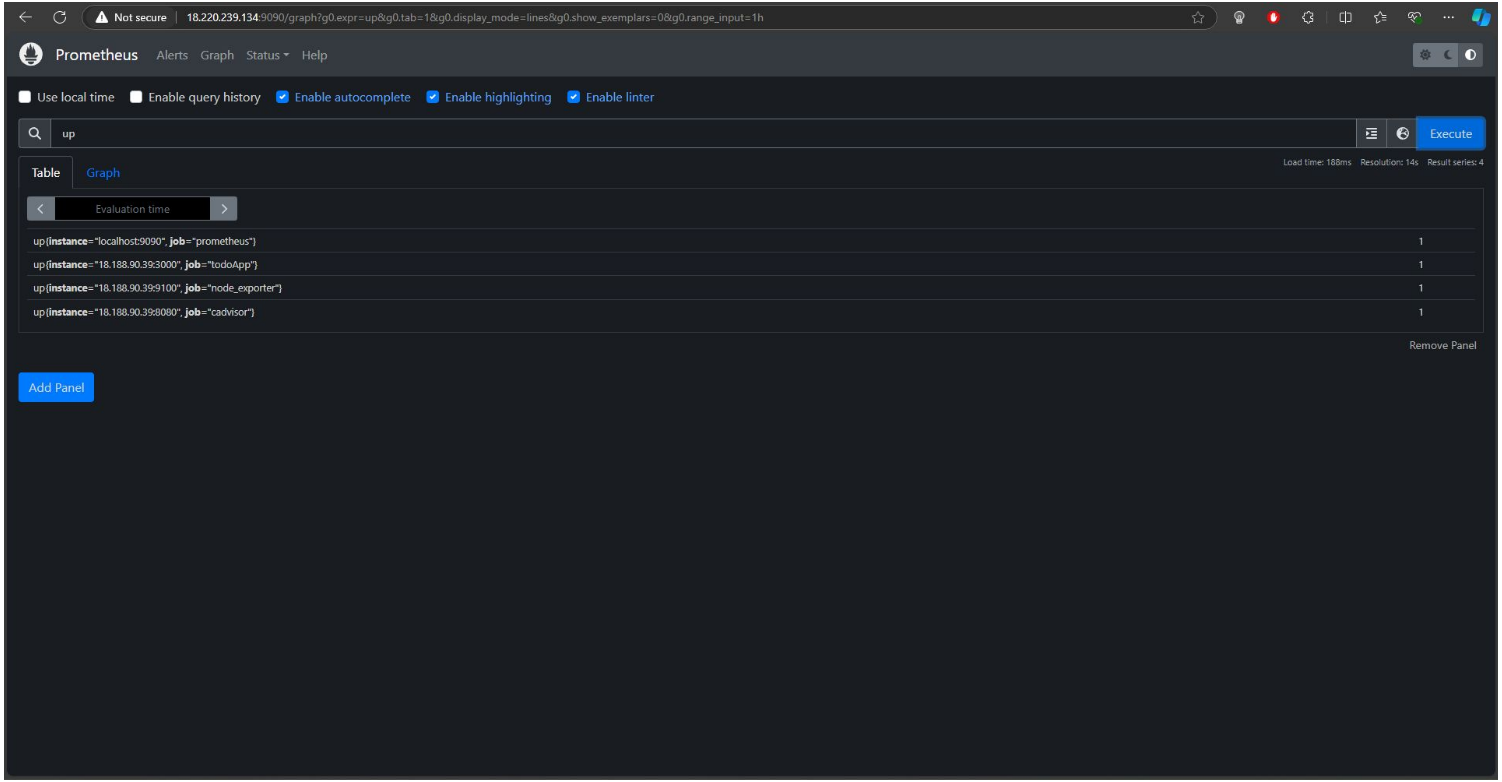


Todo-list

New Task □ Add Task

- Mohamed Mohamed Sayed Delete
- Mahmoud Mohamed Rabea Delete
- Mohamed Hosni Ahmed Delete
- Ahmed Mahmoud Tammam Delete
- Mohamed Abdallah Mohamed Delete

Results



Not secure
18.220.239.134:9090/graph?g0.expr=up&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.range_input=1h
Prometheus
Alerts
Graph
Status
Help

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

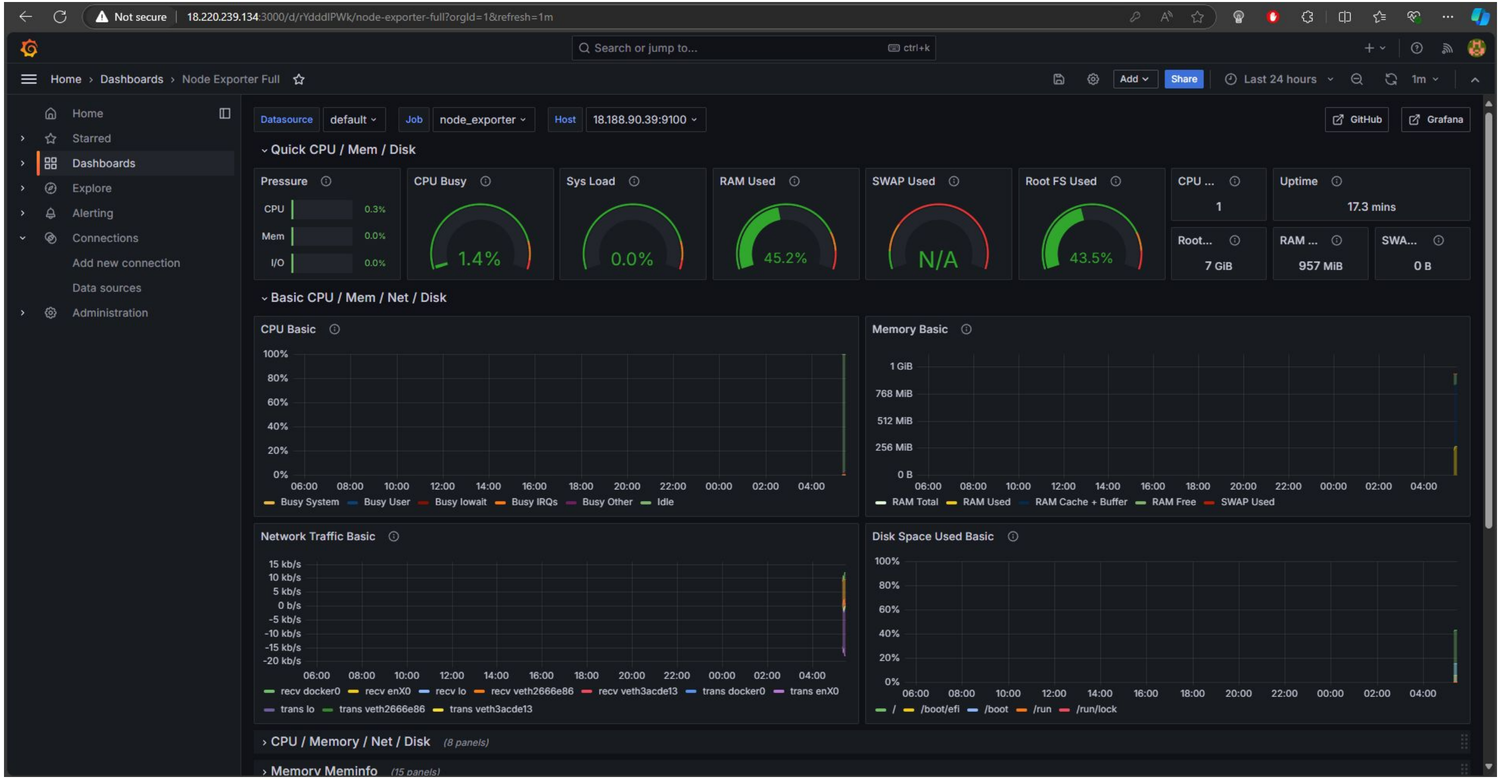
```
up
```

Execute
Table
Graph
Evaluation time

Load time: 188ms Resolution: 14s Result series: 4

```
up{instance="localhost:9090", job="prometheus"}        1
up{instance="18.188.90.39:3000", job="todoApp"}        1
up{instance="18.188.90.39:9100", job="node_exporter"}  1
up{instance="18.188.90.39:8080", job="cadvisor"}       1
```

Results



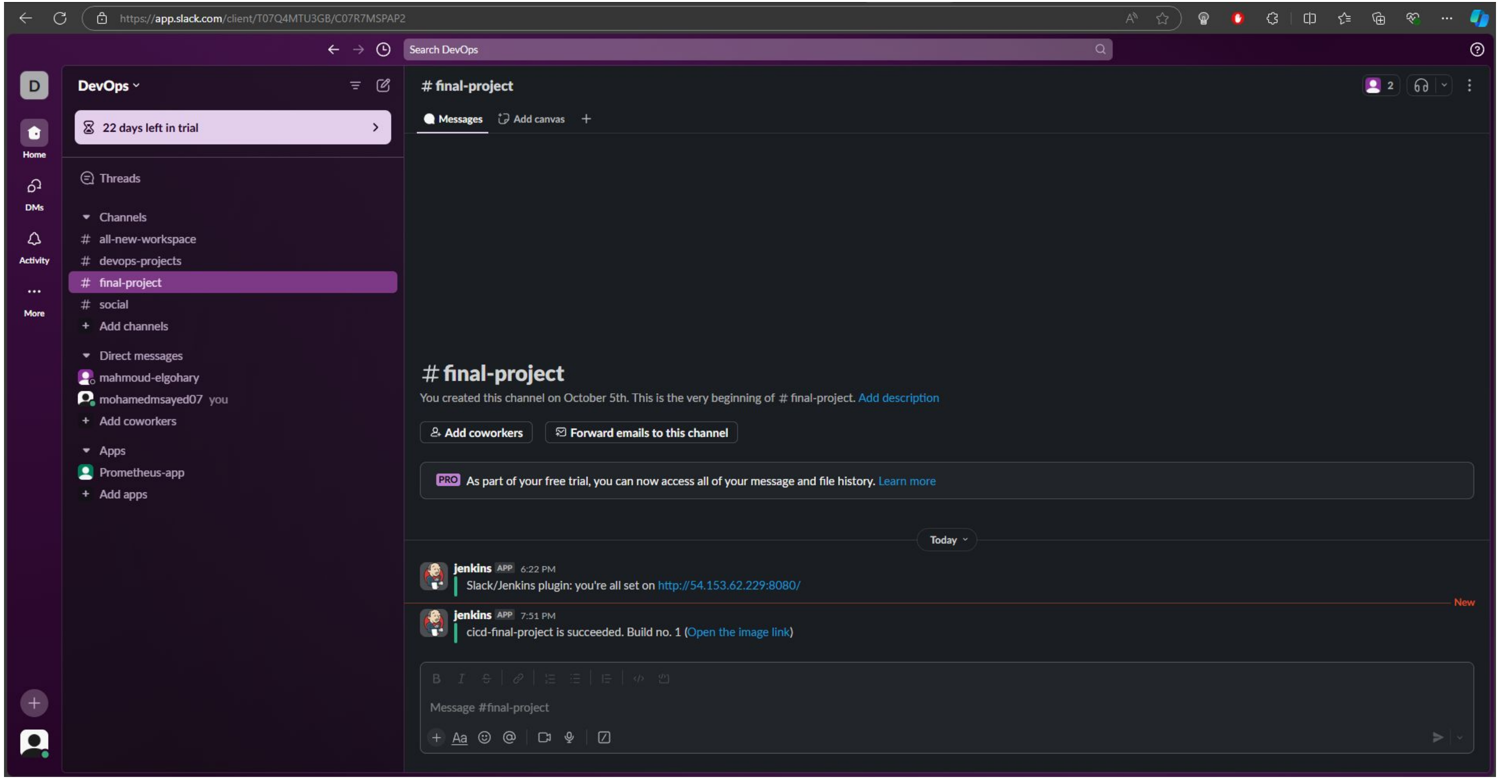
Not secure | 18.220.239.134:3000/d/rYdddlPWk/node-exporter-full?orgId=1&refresh=1m

Home › Dashboards › Node Exporter Full

Datasource: default
Job: node_exporter
Host: 18.188.90.39:9100

Last 24 hours
1m

Results



final-project

You created this channel on October 5th. This is the very beginning of # final-project. Add description

PRO As part of your free trial, you can now access all of your message and file history. Learn more

jenkins APP 6:22 PM
Slack/Jenkins plugin: you're all set on http://54.153.62.229:8080/

jenkins APP 7:51 PM
cicd-final-project is succeeded. Build no. 1 (Open the image link)

Thank You