National College of Ireland

Performance Analysis of CI/CD Pipeline for Terraform – Based Docker Deployments

Configuration Manual

MSc Research Project
Cloud Computing

Nirmal Mahale
Student ID: x23402741

School of Computing
National College of Ireland

Supervisor: Dr. Shivani Jaswal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nirmal Mahale
Student ID:	x23402741
Programme:	Cloud Computing **Year:** 2025
Module:	Msc Research Project
Lecturer: **Submission Due Date:**	11/12/2025
Project Title:	Performance and Integration Analysis of CI/CD pipelines for Terraform-based Docker Deployments in Cloud Environments
Word Count:	**925 Page Count: 17**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Nirmal Mahale

Date: 10/12/25

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	11/12/25
Penalty Applied (if applicable):	

Configuration Manual

Nirmal Mahale
Student ID: x23402741

Table of Content **Page Number**

1 Prerequisite

This is a guide that describes the configuration and run the entire CI/CD Benchmarking project. The steps should be followed in that order to complete the infrastructure, pipeline, metrics API, and dashboard configuration.

1.1 Local Machine Requirements

1.1.1 OS: Windows/ Linux / macOS
1.1.2 RAM : 8 GB minimum

Tools Needed
1.1.3 Python 3.11
1.1.4 Git
1.1.5 Docker and Docker Compose
1.1.6 Terraform
1.1.7 AWS CLI v2

2 AWS Requirements

One AWS account
Admin permission for:
2.1 EC2
2.2 VPC
2.3 Subnets
2.4 IAM
2.5 S3

3 AWS & Terraform Setup

3.1 Configure AWS CLI

aws configure

Enter : AWS Access Key – From the secrets Manager
AWS Secrets Key – From the secrets Manager
Default region (eg. eu-west-1)

Verify : *aws sts get-caller-identify*

3.2 Configure AWS CLI

Create a folder : ci-benchmark-infra/
Add : *provider.tf*

```
Provider "aws" {
   Region = "eu-west-1"
}
```

Main.tf (VPC, Subnets, Internet Gateway, EC2)

```
# Fetch the default VPC
data "aws_vpc" "default" {
  default = true
}

# Fetch all subnets inside that default VPC
data "aws_subnets" "default" {
  filter {
    name   = "vpc-id"
    values = [data.aws_vpc.default.id]
  }
}

# Security group for HTTP + optional SSH
resource "aws_security_group" "web_sg" {
  name        = "cicd-benchmark-web-sg-v2"
  description = "Allow HTTP from anywhere and SSH from my IP"
  vpc_id      = data.aws_vpc.default.id

  # HTTP 80 (for the container)
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # OPTIONAL: SSH 22 (open for now, you can restrict later)
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
```

Figure 3.2.1

3.3 Deploy AWS Infrastructure

terraform init
terraform plan
terraform apply -auto-approve

Save the EC2 Public IP

4 Configure the EC2 Server

SSH into EC2:

Ssh -i "key.pem" ubuntu@EC2_PUBLIC_IP

5 Install Docker on EC2

Sudo apt install
Sudo apt install docker.io -y
Sudo systemctl enable docker
Sudo usermod -aG docker ubuntu

6 Install Jenkins on EC2

Sudo apt install openjdk-17-jre -y
Wget -q -o – https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo apt install Jenkins -y
sudo systemctl start Jenkins

Open Jenkins UI: http://EC2_PUBLIC_IP:8080

Install: Pipeline plugin
Git plugin
Docker plugin

7 Setup the Metrics API(FastAPI/ Django)

Clone the project from Github:

git clone - https://github.com/nirmal7030/CI-CD-Benchmark-Dashboard.git
cd metrics-api

You can install project requirements using command: Pip install -r requiremnets.txt
Command to run the API : uvicorn main:app –host 0.0.0.0 –port 8000
Test : curl http://EC2_PUBLIC_IP:8000/api/metrics/ping

8 Build the Docker Image

```
# ---- Base image ----
FROM python:3.11-slim

# Prevent Python from writing .pyc files and using buffered stdout/stderr
ENV PYTHONDONTWRITEBYTECODE=1 \
    PYTHONUNBUFFERED=1

# Install OS deps (for building some wheels etc.)
RUN apt-get update && apt-get install -y \
    build-essential \
  && rm -rf /var/lib/apt/lists/*

# Create working directory
WORKDIR /app

# Install Python dependencies first (layer cache friendly)
COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt

# Copy project code
COPY . /app/

# Ensure entrypoint is executable
RUN chmod +x /app/entrypoint.sh

# Environment variables (can be overridden at runtime)
ENV DJANGO_SETTINGS_MODULE=webapp.settings \
    DEBUG=0 \
    ALLOWED_HOSTS="*"

# Expose the port gunicorn will run on
EXPOSE 8000

# Run the entrypoint script
ENTRYPOINT ["/app/entrypoint.sh"]
```

Figure 8

Navigate to the Django folder :

Docker build -t metrics-api .
Docker run -d -p 8000:8000 metrics-api

9 Configure The Metrics Dashboard

Clone repository: git clone https://github.com/nirmal7030/CI-CD-Benchmark-Dashboard.git

Cd dashboard

Install Node package: npm install
From the configuration load API endpoints export const: export const API_URL = http://EC2_PUBLIC_IP:8000/api/metrics;
Run : npm start
Open : http://EC2_PUBLIC_IP:3000

10 Configure GitHub Actions

Create: .github/workflows/ci.yml

```yaml
version: 0.2

env:
  variables:
    METRICS_API: "http://3.238.28.240/api/metrics/ingest/"
    BENCH_KEY: "bench-22b043e37d23fec188816928ce0f545e"

phases:
  install:
    runtime-versions:
      python: 3.12
    commands:
      - echo "=== [INSTALL] Creating virtualenv and installing dependencies ==="
      - python -m venv venv
      - . venv/bin/activate
      - pip install --upgrade pip
      - pip install -r requirements.txt

  pre_build:
    commands:
      - echo "=== [PRE_BUILD] Starting build timer ==="
      - BUILD_START=$(date +%s)

  build:
    commands:
      - echo "=== [BUILD] Running tests ==="
      - . venv/bin/activate
      - TEST_START=$(date +%s)

      # Run pytest if available, otherwise run Django tests
      - |
        if command -v pytest >/dev/null 2>&1; then
          echo "Running tests with pytest..."
          pytest
          TEST_EXIT=$?
        else
          echo "pytest not found, running Django tests..."
```

Figure 10.1

```
phases:
  build:
    commands:
      - |
        else
          echo "pytest not found, running Django tests..."
          python manage.py test
          TEST_EXIT=$?
        fi

      # Treat pytest exit code 5 ("no tests collected") as success
      - |
        if [ "${TEST_EXIT}" -eq 5 ]; then
          echo "pytest found no tests, treating as success."
          TEST_EXIT=0
        fi

      # If tests really failed (any other non-zero), stop the build
      - |
        if [ "${TEST_EXIT}" -ne 0 ]; then
          echo "Tests failed with exit code ${TEST_EXIT}"
          exit ${TEST_EXIT}
        fi

      - TEST_END=$(date +%s)
      - BUILD_END=$(date +%s)

      - BUILD_DURATION=$((BUILD_END - BUILD_START))
      - TEST_DURATION=$((TEST_END - TEST_START))

      - echo "Build duration: ${BUILD_DURATION}s"
      - echo "Test  duration: ${TEST_DURATION}s"

      # Save durations for post_build
      - echo "BUILD_DURATION=${BUILD_DURATION}" > metrics.env
      - echo "TEST_DURATION=${TEST_DURATION}" >> metrics.env
```

Figure 10.2

```
phases:
  post_build:
    commands:
      - |
        dept = 5.0
        clbc = 0.9

        payload = {
            "source": "codepipeline",
            "build_duration": build,
            "test_duration": test,
            "lce": round(lce, 3),
            "prt": round(prt, 3),
            "smo": smo,
            "dept": dept,
            "clbc": clbc,
        }

        url = os.getenv("METRICS_API")
        bench_key = os.getenv("BENCH_KEY")

        headers = {
            "Content-Type": "application/json",
            "X-Bench-Key": bench_key,
        }

        print("Sending metrics to", url)
        print("Payload:", payload)

        resp = requests.post(url, headers=headers, data=json.dumps(payload))
        print("Response:", resp.status_code, resp.text)
        resp.raise_for_status()
        PY

artifacts:
  files:
    - '**/*'
```

Figure 10.3

Add secrets: METRICS_API = http://EC2_PUBLIC_IP:8000/api/metrics/push
GitHub is triggered through committing.

11 Jenkins Pipeline Configuration

Create a Jenkins Pipeline

```
pipeline {
    agent any

    environment {
        VENV_DIR     = "venv"
        METRICS_API  = "http://3.238.28.240/api/metrics/ingest/"
        BENCH_HEADER = "X-Bench-Key: bench-22b043e37d23fec188816928ce0f545e"
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main',
                    url: 'https://github.com/nirmal7030/CI-CD-Benchmark-Dashboard.git',
                    credentialsId: 'github-token'
            }
        }

        stage('Setup Python & Install deps') {
            steps {
                script {
                    def start = System.currentTimeMillis() / 1000
                    sh '''
                        set -e
                        python3 -m venv "$VENV_DIR"
                        . "$VENV_DIR/bin/activate"
                        pip install --upgrade pip
                        pip install --upgrade pip
                        pip install -r requirements.txt
                    '''
                    def end = System.currentTimeMillis() / 1000
                    env.BUILD_DURATION = (end - start).toString()
                    echo "Build/Install duration (s): ${env.BUILD_DURATION}"
                }
            }
        }

        stage('Run tests') {
            steps {
                script {
                    def start = System.currentTimeMillis() / 1000
```

```
                def start = System.currentTimeMillis() / 1000

                // 🔹 Run tests but capture exit code instead of failing immediately
                int status = sh(
                    script: '''
                        . "$VENV_DIR/bin/activate"

                        if command -v pytest >/dev/null 2>&1; then
                            echo "Running tests with pytest..."
                            pytest
                        else
                            echo "pytest not found, running Django tests..."
                            python manage.py test
                        fi
                    ''',
                    returnStatus: true   // <-- key part
                )

                // 🔹 pytest exit code 5 = "no tests collected" → treat as success
                if (status == 5) {
                    echo "pytest found no tests, treating as success."
                    status = 0
                }

                // 🔹 any other non-zero = real failure
                if (status != 0) {
                    error "Test stage failed with exit code ${status}"
                }

                def end = System.currentTimeMillis() / 1000
                env.TEST_DURATION = (end - start).toString()
                echo "Test duration (s): ${env.TEST_DURATION}"
            }
        }
    }

    stage('Compute novel metrics (demo)') {
        steps {
            script {
                double buildDur = (env.BUILD_DURATION ?: "0") as double
                double testDur  = (env.TEST_DURATION  ?: "0") as double
```

```
                    double testDur  = (env.TEST_DURATION  ?: "0") as double

                    double lce  = 120.0 / (buildDur + 60.0)   // higher when faster
                    double prt  = buildDur + testDur          // total pipeline time
                    double smo  = 1.0                         // pretend 1s secrets overhead
                    double dept = 5.0                         // pretend 5s env provisioning
                    double clbc = 0.9                         // pretend high consistency

                    if (lce < 0.0) { lce = 0.0 }
                    if (lce > 1.0) { lce = 1.0 }

                    env.LCE  = String.format("%.3f", lce)
                    env.PRT  = String.format("%.3f", prt)
                    env.SMO  = String.format("%.3f", smo)
                env.DEPT = String.format("%.3f", dept)
                env.CLBC = String.format("%.3f", clbc)

                echo "Computed metrics → LCE=${env.LCE}, PRT=${env.PRT}, SMO=${env.SMO}, DEPT=${env.DEPT}, CLBC=$
            }
        }
    }

    stage('Send metrics to dashboard') {
        steps {
            script {
                echo "📊 Sending metrics to dashboard..."
                sh """
                    curl -X POST \\
                echo "📊 Sending metrics to dashboard..."
                sh """
                    curl -X POST \\
                        -H 'Content-Type: application/json' \\
                        -H '${BENCH_HEADER}' \\
                        -d "{\\"source\\":\\"jenkins\\",\\"build_duration\\":${BUILD_DURATION},\\"test_duration\
                        ${METRICS_API}
                """
            }
        }
    }
}
```

Figure 11.1

Add Jenkins : Metrics API = http://ec2_public_key:8000/api/metrics/push
The Metrics API has to update in the link

Run job = metrics displays on the dashboard.

12 AWS CodePipeline

Create buildspec.yml:

```yaml
version: 0.2

env:
  variables:
    METRICS_API: "http://44.220.53.248:8000/api/generic-metrics/push/"

phases:
  install:
    commands:
      - echo Installing Python dependencies...
      - pip install --upgrade pip
      - pip install -r requirements.txt
  pre_build:
    commands:
      - echo Running Django checks...
      - python manage.py check
  build:
    commands:
      - echo "Starting timed build and tests..."
      - START_TS=$(date +%s)
      # run migrations just to ensure DB schema OK
      - python manage.py migrate --noinput
      # run tests (even if 0 tests, still OK)
      - python manage.py test || true
      - END_TS=$(date +%s)
      - BUILD_TIME=$((END_TS-START_TS))
      - echo "Build+Test duration (sec): $BUILD_TIME"



      - |
        cat <<EOF > metrics_payload.json
        {
```

Figure 12.1

```
    - |
      cat <<EOF > metrics_payload.json
      {
        "tool": "$PIPELINE_TOOL",
        "pipeline_name": "$PIPELINE_NAME",
        "build_time_sec": $BUILD_TIME,
        "tests_total": $TOTAL_TESTS,
        "failed_tests": $FAILED_TESTS,
        "coverage_percent": $COVERAGE
      }
      EOF

    - echo "Sending metrics to dashboard..."
    - cat metrics_payload.json
    - curl -X POST "$METRICS_API" \
        -H "Content-Type: application/json" \
        -d @metrics_payload.json || echo "Metrics push failed (but build will still finish)."

artifacts:
  files:
    - '**/*'
```

Figure 12.2

13 Starting the Entire Project(Run Sequence)

Once everything is configured:

13.1 Start EC2 services
Sudo systemctl start Jenkins
Docker start metrics-api

13.2 Launch Metrics Dashboard
npm start

13.3 Trigger Github Actions
Commit a small file – workflow runs.

13.4 Run Jenkins Job
Click Build Now
View Metrics – http://EC2_PUBLIC_IP:8000

Figure 13.1

References

HashiCorp (2024) *Terraform documentation – AWS provider*. Available at: https://developer.hashicorp.com/terraform/docs/providers/aws

HashiCorp (2023) *Getting started with Terraform*. HashiCorp Learning Portal. Available at: https://developer.hashicorp.com/terraform/tutorials

Amazon Web Services (2024) *Amazon EC2 user guide for Linux instances*. Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/

Amazon Web Services (2024) *Amazon VPC user guide*. Available at: https://docs.aws.amazon.com/vpc/latest/userguide/

Amazon Web Services (2023) *AWS CLI command reference*. Available at: https://docs.aws.amazon.com/cli/latest/

Docker (2024) *Dockerfile reference and best practices*. Available at: https://docs.docker.com/engine/reference/builder/

Jenkins Project (2024) *Installing Jenkins*. Available at: https://www.jenkins.io/doc/book/installing/

GitHub (2024) *GitHub Actions documentation – workflow syntax*. Available at: https://docs.github.com/en/actions

FastAPI (2024) *FastAPI documentation – path operations*. Available at: https://fastapi.tiangolo.com/

Django Software Foundation (2024) *Django deployment checklist*. Available at: https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/