

## DEPI GRADUATION PROJECT

# AUTOMATED DEPLOYMENT PIPELINE WITH JENKINS AND DOCKER

**Track:** DevOps Engineer

**Instructor:** Ahmed Nasr

**Training Center:** YAT Learning Solutions

**Group Code:** CAI1\_SWD1\_S6d

**Team Number:** 3

**Team Members:**

- Noor Elhoda Mohamed Abdo Eisa
- Ola Youssef Mohammed
- Maram Hassan Mohamed
- Basant Ehab Sayed
- Saif Mohammed Dawood

**GitHub Repository of our Project:**

<https://github.com/DEPI-DevOps-tasks/DevOps-Challenge-fork>

## 1. Introduction

The goal of this project was to develop a fully automated CI/CD pipeline using Jenkins, Docker, Terraform, and Ansible. This pipeline supports the build, testing, and deployment of a sample Dockerized application, facilitating seamless integration and continuous deployment to cloud infrastructure.

## 2. Technologies used

**Jenkins:** For automating continuous integration and deployment.

**Docker:** For containerizing the application.

**Ansible:** For automating configuration management and deployment.

**Terraform:** For provisioning cloud infrastructure (AWS).

**AWS:** For hosting the application on cloud infrastructure (EC2).

**GitHub:** Version control for source code.

## 3. Project Phases

### Phase 1: Initial Setup & Dockerization

This phase focused on setting up the local environment, building the Dockerized application, and testing locally.

1. **Forking the Repository:** We forked the DevOps challenge repo and worked with the app to build a Dockerfile and docker-compose.yml.
2. **Creating the Docker Image:** A Dockerfile was written to containerize the application. Redis was added as a service using Docker Compose.
3. **Creating the Docker-compose file:** To orchestrate the containers (Python app and Redis), we created a docker-compose.yml file that sets up the necessary services.

### Dockerfile Code:

```
FROM python:3.8-slim

WORKDIR /app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "hello.py"]
```

### Docker-Compose.yml code:

```
version: '3'
services:
  app:
    build: .
    container_name: my_python_app
    ports:
      - "5000:5000"
    environment:
      - ENVIRONMENT=DEV
      - HOST=0.0.0.0
      - PORT=5000
      - REDIS_HOST=redis
      - REDIS_PORT=6379
      - REDIS_DB=0
    depends_on:
      - redis
    command: sh -c "sleep 5 && python hello.py" # Wait for Redis to start

  redis:
    image: "redis:5.0"
    container_name: redis
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data

volumes:
  redis_data:
```

4. **Testing the Application:** The application was tested using Docker Compose, confirming that both the Python app and Redis are up and running locally.

**Result:** Both services were successfully running on local ports (Python on 5000, Redis on 6379).

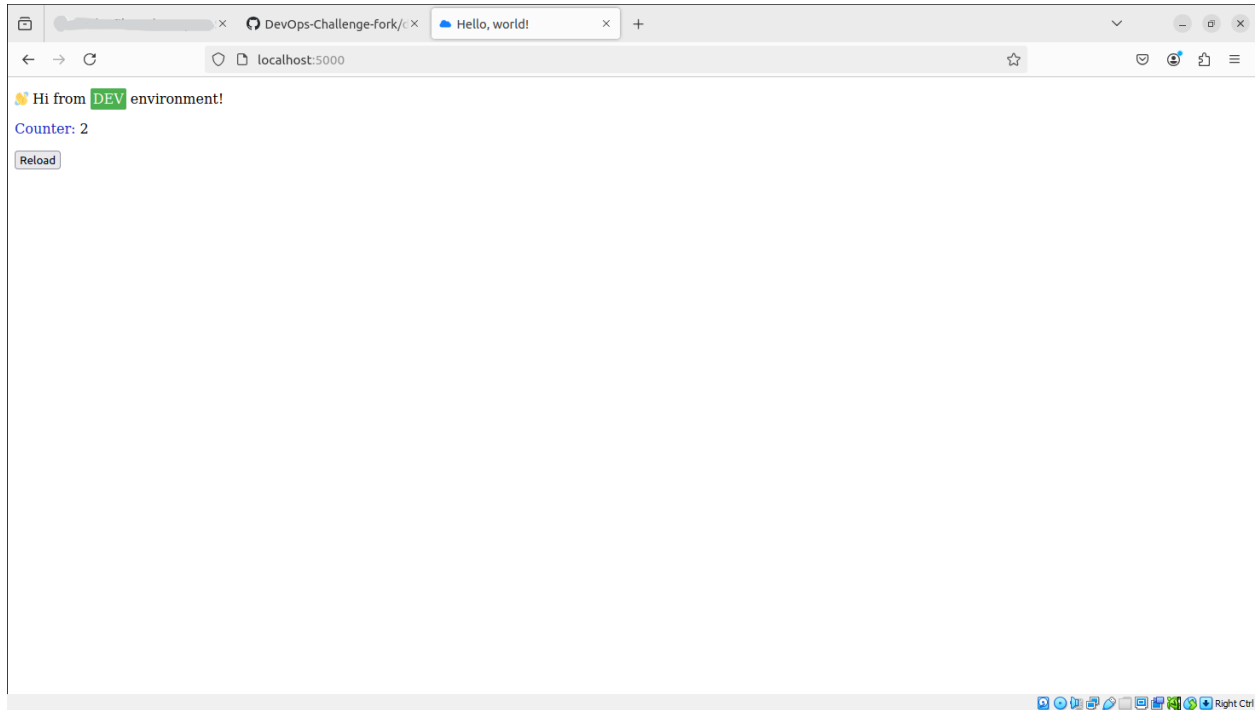
```
hp@Ubuntu-Khaleel:~$ ls
Desktop  Downloads  jenkins101  Pictures  snap  Videos
Documents  install-docker.sh  Music  Public  Templates

hp@Ubuntu-Khaleel:~$ mkdir devops-project
hp@Ubuntu-Khaleel:~$ cd devops-project
hp@Ubuntu-Khaleel:~/devops-project$ git clone https://github.com/your-repo/DevOps-Challenge-fork.git
Cloning into 'DevOps-Challenge-fork'...
Username for 'https://github.com': ^C
hp@Ubuntu-Khaleel:~/devops-project$ git clone https://github.com/DEPI-DevOps-tasks/DevOps-Challenge-fork
Cloning into 'DevOps-Challenge-fork'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 22 (delta 1), reused 0 (delta 0), pack-reused 16 (from 1)
Receiving objects: 100% (22/22), 6.89 KiB | 440.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.

hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$ ls
docker-compose.yml  hello.py  README.md  static  tests
Dockerfile          LICENSE  requirements.txt  templates

hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$ docker-compose build
redis uses an image, skipping
Building app
[*] Building 68.4s (10/10) FINISHED          docker:default
=> [internal] load build definition from Dockerfile          0.3s
=> == transferring dockerfile: 202B                          0.0s
=> [internal] load metadata for docker.io/library/python:3.8-slim 2.9s
=> [internal] load .dockerignore                          0.2s
=> == transferring context: 2B                                0.0s
=> [1/5] FROM docker.io/library/python:3.8-slimsha256:1d52838af602b4b5 44.5s
=> == resolve docker.io/library/python:3.8-slimsha256:1d52838af602b4b5a 0.3s
=> == sha256:1d52838af602b4b5a831beb13a0e4d073280665ea 10.41kB / 10.41kB 0.0s
=> == sha256:314bc2fb0714b7807bf5699c98f0c73817e579799f2 1.75kB / 1.75kB 0.0s
=> == sha256:b5f62925bd0f63f48ccbacd5e87d0c3a07e2f229cd2 5.25kB / 5.25kB 0.0s
=> == sha256:302e3ee498053a7b5332ac79e8efebec16e90028 29.13MB / 29.13MB 11.4s
=> == sha256:030d7bdc20a63e3d22192b292d006a09fa333949f5 3.51MB / 3.51MB 6.5s
=> == sha256:a3f1dfe736c5f959143f23d75ab522a0be2da90 14.53MB / 14.53MB 15.6s
=> == sha256:3971691a363796c39467aae4cdc6ef773273fe6bfc67154 248B / 248B 7.1s
=> == extracting sha256:302e3ee498053a7b5332ac79e8efebec16e900289f1cedc 21.1s
=> == extracting sha256:030d7bdc20a63e3d22192b292d006a09fa333949f536d62 1.8s
=> == extracting sha256:a3f1dfe736c5f959143f23d75ab522a0be2da902efac236 6.6s
=> == extracting sha256:3971691a363796c39467aae4cdc6ef773273fe6bfc67154 0.0s
=> [internal] load build context          0.5s
=> == transferring context: 46.97kB      0.2s
=> [2/5] WORKDIR /app                  2.0s
=> [3/5] COPY requirements.txt ./       0.5s
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt 15.3s
=> [5/5] COPY . .                      0.8s
=> == exporting to image                1.2s
=> == exporting layers                  1.0s
=> == writing image sha256:23e5f713e1fe4741cccc14349dc28cdf2b9f7f043c487c 0.0s
=> == naming to docker.io/library/devops-challenge-fork_app 0.1s

hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$ docker-compose up -d
Creating network "devops-challenge-fork_default" with the default driver
Creating volume "devops-challenge-fork_redis_data" with default driver
Pulling redis (redis:5.0)...
5.0: Pulling from library/redis
a603fa5e3b41: Pull complete
77631c3ef092: Pull complete
ed3847cf62b8: Pull complete
295236254fbb: Pull complete
1d25d6f70191: Pull complete
23acd6bf5eef: Pull complete
Digest: sha256:fc5ecd863862f89f04334b7cbb57e93c9790478ea8188a49f6e57b0967d38c75
Status: Downloaded newer image for redis:5.0
Creating redis ... done
Creating my_python_app ... done
hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
d79d6db7c28f   devops-challenge-fork_app           "python hello.py"       9 seconds ago Up 6 seconds  0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
b4ded7136856   redis:5.0                           "docker-entrypoint.s..." 13 seconds ago Up 10 seconds  0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
d3b4cc686bc8   myjenkins-blueocean:2.414.2       "/usr/bin/tini -- /u..." 3 weeks ago   Up 16 minutes  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000/tcp
hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$
```



## Phase 2: Jenkins & CI Integration

This phase established the CI process, integrated Git, and enabled automated testing.

1. **Jenkins Pipeline:** A Jenkinsfile was created to clone the repo, build the Docker image, and run the application using Docker Compose. Unit tests were added to verify the build.
2. **Automated Notifications:** Configured Jenkins to send emails for build successes or failures using a webhook.

**Part of the Pipeline Code:**

```
pipeline {
  agent any

  environment {
    ENVIRONMENT = 'DEV'
    HOST = '0.0.0.0'
    PORT = '5000'
    REDIS_HOST = 'redis'
    REDIS_PORT = '6379'
    REDIS_DB = '0'
    DOCKER_IMAGE_NAME = 'olayoussef/my_python_app'
  }

  triggers {
    githubPush()
  }

  stages {
    stage('Clone github repo') {
      steps {
        git 'https://github.com/DEPI-DevOps-tasks/DevOps-Challenge-fork'
      }
    }
  }
}
```

```
stage('Build and Push Docker Image') {
  steps {
    script {
      sh 'docker build -t my_python_app .'

      withCredentials([usernamePassword(credentialsId: 'dockerhub-credentials', passwordVariable:
'pass', usernameVariable: 'dockerhubuser')]) {
        sh 'docker login -u $dockerhubuser -p $pass'
        sh 'docker tag my_python_app:latest $DOCKER_IMAGE_NAME:latest'
        sh 'docker push $DOCKER_IMAGE_NAME:latest'
      }
    }
  }
}
```

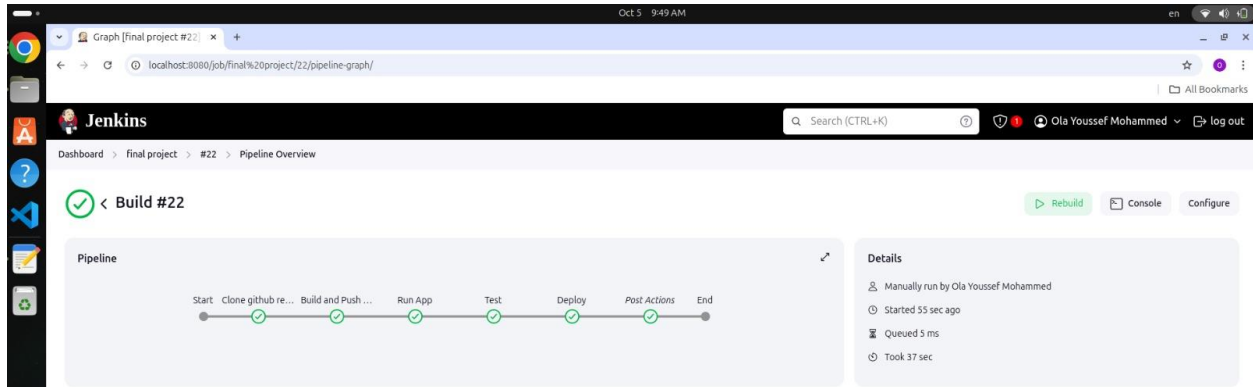


```
stage('Run App') {
  steps {
    script {
      sh '''
        # Clean up any existing containers
        docker rm -f my_python_app || true
        docker rm -f redis || true
        # Run the application
        docker-compose up -d
      '''
    }
  }
}

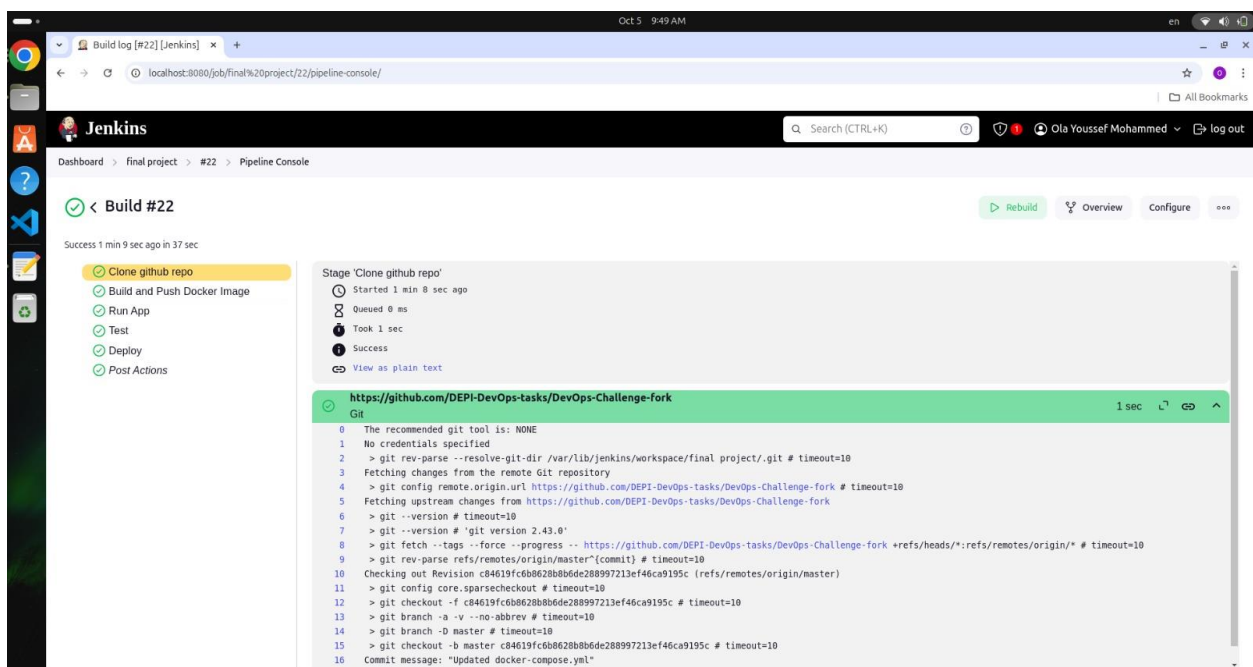
stage('Test') {
  steps {
    sh 'docker exec my_python_app python tests/test.py' // Replace with your actual test command
  }
}

stage('Deploy') {
  when {
    expression {
      return currentBuild.result == null || currentBuild.result == 'SUCCESS'
    }
  }
  steps {
    sh 'docker-compose up -d'
  }
}
```

```
post {
  always {
    sh 'docker-compose down --volumes'
  }
  success {
    mail to: 'olayoubadr@gmail.com, maram.hassan95@gmail.com, basantehab83@gmail.com, saifdawoodcs@gmail.com, noor.mohamed.eisa@gmail.com',
        subject: "Build Succeeded: ${env.BUILD_TAG}",
        body: "The build was successful. Check the logs for details."
  }
  failure {
    mail to: 'olayoubadr@gmail.com, maram.hassan95@gmail.com, basantehab83@gmail.com, saifdawoodcs@gmail.com, noor.mohamed.eisa@gmail.com',
        subject: "Build Failed: ${env.BUILD_TAG}",
        body: "The build failed. Please check the logs."
  }
}
```



The screenshot shows the Jenkins Pipeline Overview for Build #22. The pipeline consists of the following stages: Start, Clone github re..., Build and Push..., Run App, Test, Deploy, Post Actions, and End. All stages are marked as successful with green checkmarks. The details section indicates that the build was manually run by Ola Youssef Mohammed, started 55 seconds ago, and took 37 seconds to complete.



The screenshot shows the Jenkins Pipeline Console for Build #22, specifically the 'Clone github repo' stage. The console output shows the following commands and their results:

```

0 The recommended git tool is: NONE
1 No credentials specified
2 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/final project/.git # timeout=10
3 Fetching changes from the remote Git repository
4 > git config remote.origin.url https://github.com/DEPI-DevOps-tasks/DevOps-Challenge-fork # timeout=10
5 Fetching upstream changes from https://github.com/DEPI-DevOps-tasks/DevOps-Challenge-fork
6 > git --version # timeout=10
7 > git --version # 'git version 2.43.0'
8 > git fetch --tags --force --progress -- https://github.com/DEPI-DevOps-tasks/DevOps-Challenge-fork +refs/heads/*:refs/remotes/origin/* # timeout=10
9 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
10 Checking out Revision c84619fc6b8628b8b6de288997213ef46ca9195c (refs/remotes/origin/master)
11 > git config core.sparsecheckout # timeout=10
12 > git checkout -f c84619fc6b8628b8b6de288997213ef46ca9195c # timeout=10
13 > git branch -a -v --no-abbrev # timeout=10
14 > git branch -D master # timeout=10
15 > git checkout -b master c84619fc6b8628b8b6de288997213ef46ca9195c # timeout=10
16 Commit message: "Updated docker-compose.yml"
  
```



Build log [#22] [Jenkins] x +

localhost:8080/job/final%20project/22/pipeline-console/

Jenkins Search (CTRL+K) Ola Youssef Mohammed log out

Dashboard > final project > #22 > Pipeline Console

Build #22

Success 1 min 9 sec ago in 37 sec

- Clone github repo
- Build and Push Docker Image
- Run App
- Test
- Deploy
- Post Actions

Stage 'Build and Push Docker Image'

Started 1 min 7 sec ago

Queued 0 ms

Took 18 sec

Success

View as plain text

docker build -t my\_python\_app . 7 sec

Shell Script

docker login -u \$dockerhubuser -p \$pass 1.6 sec

Shell Script

docker tag my\_python\_app:latest \$DOCKER\_IMAGE\_NAME:latest 0.32 sec

Shell Script

docker push \$DOCKER\_IMAGE\_NAME:latest 8.5 sec

Shell Script

```
0 + docker push ****/my_python_app:latest
1 The push refers to repository [docker.io/****/my_python_app]
2 92502dfb44b4: Preparing
3 248a43c483ec: Preparing
4 30ca1f50bb75: Preparing
5 9bfbae837d37: Preparing
6 d2a2207b52a4: Preparing
7 5d2d143f3d7f: Preparing
8 c3772b569c3a: Preparing
9 8d853e8add5d: Preparing
```

Build log [#22] [Jenkins] x +

localhost:8080/job/final%20project/22/pipeline-console/

Jenkins Search (CTRL+K) Ola Youssef Mohammed log out

Dashboard > final project > #22 > Pipeline Console

Build #22

Success 1 min 9 sec ago in 37 sec

- Clone github repo
- Build and Push Docker Image
- Run App
- Test
- Deploy
- Post Actions

Stage 'Run App'

Started 49 sec ago

Queued 0 ms

Took 1.4 sec

Success

View as plain text

docker-compose up -d 1.3 sec

Shell Script

```
1 time="2024-10-05T09:48:23+03:00" level=warning msg="/var/lib/jenkins/workspace/final project/docker-compose.yml: the attribute 'version' is obsolete, it will be
  ignored, please remove it to avoid potential confusion"
2 Network finalproject_default Creating
3 Network finalproject_default Created
4 Volume "finalproject_redis_data" Creating
5 Volume "finalproject_redis_data" Created
6 Container redis Creating
7 Container redis Created
8 Container my_python_app Creating
9 Container my_python_app Created
10 Container redis Starting
11 Container redis Started
12 Container my_python_app Starting
```

Build log [#22] [Jenkins] x +

localhost:8080/job/final%20project/22/pipeline-console/

Jenkins Search (CTRL+K) Ola Youssef Mohammed log out

Dashboard > final project > #22 > Pipeline Console

Build #22

Success 1 min 9 sec ago in 37 sec

- Clone github repo
- Build and Push Docker Image
- Run App
- Test
- Deploy
- Post Actions

Stage 'Test'

Started 47 sec ago

Queued 0 ms

Took 0.36 sec

Success

View as plain text

docker exec my\_python\_app python tests/test.py

Shell Script

```
0 + docker exec my_python_app python tests/test.py
1 ...
2 .....
3 Ran 3 tests in 0.000s
4
5 OK
```

0.3 sec

Build log [#22] [Jenkins] x +

localhost:8080/job/final%20project/22/pipeline-console/

Jenkins Search (CTRL+K) Ola Youssef Mohammed log out

Dashboard > final project > #22 > Pipeline Console

Build #22

Success 1 min 9 sec ago in 37 sec

- Clone github repo
- Build and Push Docker Image
- Run App
- Test
- Deploy
- Post Actions

Stage 'Deploy'

Started 47 sec ago

Queued 0 ms

Took 0.39 sec

Success

View as plain text

docker-compose up -d

Shell Script

```
1 time="2024-10-05T09:48:25+03:00" level=warning msg="/var/lib/jenkins/workspace/final project/docker-compose.yml: the attribute `version` is obsolete, it will be
  ignored, please remove it to avoid potential confusion"
2 Container redis Running
```

0.3 sec

Scroll to Bottom

Build log [#22] [Jenkins] x +

localhost:8080/job/final%20project/22/pipeline-console/

Jenkins Search (CTRL+K) Ola Youssef Mohammed log out

Dashboard > final project > #22 > Pipeline Console

Build #22

Success 1 min 9 sec ago in 37 sec

- Clone github repo
- Build and Push Docker Image
- Run App
- Test
- Deploy
- Post Actions

Stage 'Post Actions'

Started 46 sec ago

Queued 0 ms

Took 14 sec

Success

View as plain text

docker-compose down --volumes

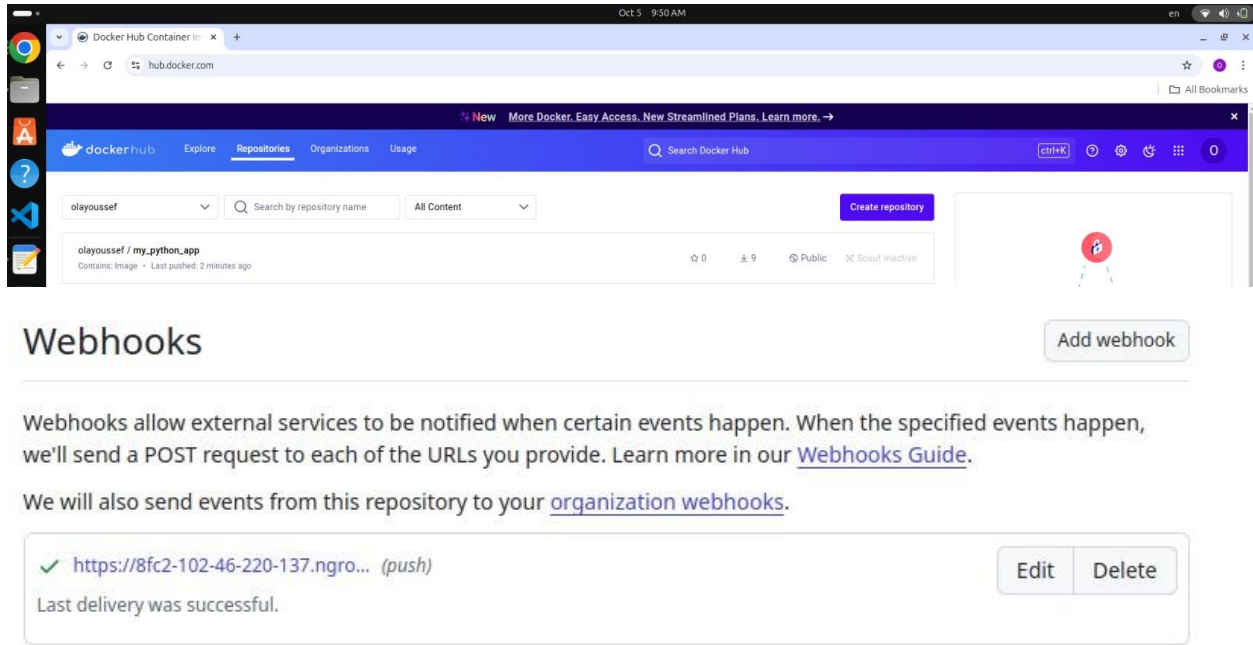
Shell Script

12 sec

Build Succeeded: jenkins-final project-22

Mail

2.3 sec



## Phase 3: Infrastructure Provisioning with Terraform

This phase used Terraform to provision AWS resources for the application's deployment.

1. **Terraform Script:** A Terraform script was written to create an EC2 instance on AWS. It also used a local provisioner to initiate the Ansible playbook for configuration management.
2. **Jenkins Pipeline for Terraform:** A Jenkins pipeline was created to execute the Terraform script, setting up the infrastructure automatically.
3. **Code:** (See full code in our GitHub Repo)

```

provider "aws" {
  region = "us-west-2"
}
resource "aws_instance" "my_ec2" { ... }

```

https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#vpcs:

aws Services Search [Alt+S]

EC2 S3 Billing and Cost Management VPC DynamoDB

### VPC dashboard

EC2 Global View

Filter by VPC

Virtual private cloud

Your VPCs

Subnets

Route tables

Internet gateways

Egress-only Internet gateways

Carrier gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

### Your VPCs (1/3)

Last updated less than a minute ago

Actions Create VPC

Search

Name	VPC ID	State	IPv4 CIDR
-	vpc-079c3dc720e60a342	Available	172.31.0.0/16
MyVPC	vpc-0c287d3cdb937c935	Available	10.0.0.0/16

### vpc-0c287d3cdb937c935 / MyVPC

Details Resource map CIDRs Flow logs Tags Integrations

#### Details

VPC ID vpc-0c287d3cdb937c935	State Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-05beb5e2be7b05d28	Main route table rtb-0f5d90041de764329	Main network ACL acl-011926800d41823ac

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=:

aws Services Search [Alt+S]

EC2 S3 Billing and Cost Management VPC DynamoDB

### EC2 Dashboard

EC2 Global View

Events

Console-to-Code Preview

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

### Instances (1/2)

Last updated less than a minute ago

Connect Instance state Actions Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

Instance state = running Clear filters

Name	Instance ID	Instance state	Instance type	Status check
TerraformInstance	i-0a921c63c93791b37	Running	t2.medium	Initializing

### i-0a921c63c93791b37 (TerraformInstance)

Details Status and alarms Monitoring Security Networking Storage Tags

#### Instance summary Info

Instance ID i-0a921c63c93791b37 (TerraformInstance)	Public IPv4 address 54.226.209.196   open address	Private IPv4 addresses 10.0.1.5
IPv6 address	Instance state	Public IPv4 DNS

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

← → ↻ <https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#subnets:> ☆

aws Services Search [Alt+S] N. Virginia maram

EC2 S3 Billing and Cost Management VPC DynamoDB

**VPC dashboard** ×

EC2 Global View [↗](#)

Filter by VPC ▾

▼ Virtual private cloud

- Your VPCs
- Subnets**
- Route tables
- Internet gateways
- Egress-only Internet gateways
- Carrier gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists
- Endpoints

**Subnets (1/3) Info** Last updated 5 minutes ago [↻](#) **Actions** ▾ [Create subnet](#)

Find resources by attribute or tag

Name	Subnet ID	State	VPC
terraform-subnet	subnet-0e8742933fde33292	Available	vpc-0b146d925c00ed772

**subnet-0e8742933fde33292 / terraform-subnet**

**Details** Flow logs Route table Network ACL CIDR reservations Sharing Tags

Subnet ID subnet-0e8742933fde33292	Subnet ARN arn:aws:ec2:us-east-1:851725249314:subnet/subnet-0e8742933fde33292	State Available	IPv4 CIDR 10.0.1.0/24
Available IPv4 addresses 251	IPv6 CIDR -	IPv6 CIDR association ID -	Availability Zone us-east-1e
		VPC	Route table

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

← → ↻ <https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#RouteTables:> ☆

aws Services Search [Alt+S] N. Virginia maram

EC2 S3 Billing and Cost Management VPC DynamoDB

**VPC dashboard** ×

EC2 Global View [↗](#)

Filter by VPC ▾

▼ Virtual private cloud

- Your VPCs
- Subnets
- Route tables**
- Internet gateways
- Egress-only Internet gateways
- Carrier gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists
- Endpoints

**Route tables (1/5) Info** Last updated less than a minute ago [↻](#) **Actions** ▾ [Create route table](#)

Find resources by attribute or tag

Name	Route table ID	Explicit subnet associ...	Edge associations
terraform-routeTable	rtb-040f2f0897f559a4c	subnet-0b201459a9d31c...	-

**rtb-040f2f0897f559a4c / terraform-routeTable**

**Details** Routes Subnet associations Edge associations Route propagation Tags

Route table ID rtb-040f2f0897f559a4c	Main No	Explicit subnet associations subnet-0b201459a9d31c298 / terraform-subnet	Edge associations -
VPC vpc-0c287d3cdb937c935   MyVPC	Owner ID 851725249314		

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



← → ↻ <https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#igws:> ☆

aws Services Search [Alt+S] N. Virginia maram

EC2 S3 Billing and Cost Management VPC DynamoDB

**VPC dashboard** ×

EC2 Global View [Filter by VPC](#)

▼ Virtual private cloud

- Your VPCs
- Subnets
- Route tables
- Internet gateways**
- Egress-only internet gateways
- Carrier gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists
- Endpoints

**Internet gateways (1/3) Info** [Create internet gateway](#)

Search

Name	Internet gateway ID	State	VPC ID
-	igw-050bcd071b483f112	Attached	vpc-079c3dc720e
MyInternetGateway	igw-0d51e3f75b91d4767	Attached	vpc-0dbc21e23fd

**igw-0d51e3f75b91d4767 / MyInternetGateway**

[Details](#) [Tags](#)

**Details**

Internet gateway ID	State	VPC ID	Owner
igw-0d51e3f75b91d4767	Attached	vpc-0dbc21e23fd0592ef   MyVPC	851725249314

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

← → ↻ <localhost:8080/blue/organizations/jenkins/new/detail/new/5/pipeline> ☆

✓ new < 5 Pipeline Changes Tests Artifacts [Restart Terraform Apply](#) [Logout](#) ×

Branch: — 1m 59s No changes

Commit: — 3 minutes ago Started by user maram hassan

Start Checkout Terraform... Install Terraform Terraform Init Terraform Plan Terraform Apply End

**Terraform Apply - 35s** [Restart Terraform Apply](#) [Download](#)

✓	> terraform apply -auto-approve — Shell Script	33s
✓	> Delete workspace when build is done	<1s

## Phase 4: Deployment with Ansible

Ansible was used to automate the deployment of the application to the AWS EC2 instance.

1. **Ansible Playbook:** The playbook installs Docker on the EC2 instance, pulls the application's Docker image, and runs the app and Redis containers.
2. **Testing Deployment:** Verified that the application was running on the EC2 instance.

Code:

```
- name: Deploy Docker Application on AWS EC2
  hosts: all
  become: true

  tasks:
    - name: Gather facts
      setup:

    - name: Update apt and install required packages
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - docker.io
        - python3-pip
        - python3-venv
        - python3-apt
        - curl # Ensure curl is installed
        - git # Optional: Install git if you need version control
```



```

- name: Start Docker service
  service:
    name: docker
    state: started
    enabled: true

- name: Create Python virtual environment
  command: python3 -m venv /home/ubuntu/venv
  args:
    creates: /home/ubuntu/venv

- name: Install Docker Python module in virtual environment
  command: /home/ubuntu/venv/bin/python -m pip install docker

- name: Create a directory for the application
  file:
    path: /home/ubuntu/app
    state: directory

```

```

- name: Copy docker-compose.yml to EC2 instance
  copy:
    src: ./docker-compose.yml
    dest: /home/ubuntu/app/docker-compose.yml

- name: Install Docker Compose
  shell: >
    curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)"
  -o /usr/local/bin/docker-compose
  args:
    creates: /usr/local/bin/docker-compose

- name: Set permissions for Docker Compose
  command: chmod +x /usr/local/bin/docker-compose

- name: Verify Docker Compose installation
  command: docker-compose --version
  register: docker_compose_version

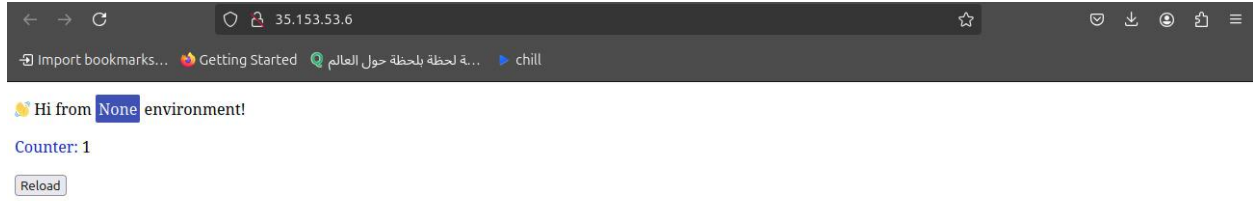
- debug:
  var: docker_compose_version.stdout

- name: Pull Docker images
  command: docker-compose -f /home/ubuntu/app/docker-compose.yml pull
  args:
    chdir: /home/ubuntu/app

- name: Run Docker containers
  command: docker-compose -f /home/ubuntu/app/docker-compose.yml up -d
  args:
    chdir: /home/ubuntu

```





## CONCLUSION

This project implemented an automated CI/CD pipeline that allows for continuous integration, testing, and deployment of a Dockerized application to a cloud environment using Jenkins, Docker, Terraform, and Ansible. The pipeline is flexible, scalable, and enables the rapid deployment of applications to production environments.