

CI/CD Pipeline using Jenkins, Docker & AWS EC2

Project Overview

This project demonstrates the implementation of a CI/CD pipeline using Jenkins, Docker, and Docker Compose, deployed on an AWS EC2 instance. The pipeline automatically pulls source code from GitHub, builds a Docker image, and deploys the application in a containerized environment.

System Architecture

Developer pushes code to GitHub → Jenkins pulls the code → Jenkins builds Docker image (CI) → Jenkins deploys the application using Docker Compose (CD) → Application runs on AWS EC2 and is accessed via public IP.

Steps

This project is divided into two parts :

Part A - For CI

Part B - For CD

Part A Section

Step 1: Create a folder for CI files, then pushes the files to github. [Github repo: CI-CD]

Step 2: Create EC2 instance.

Instances (1/2) Info		Last updated about 4 hours ago		C	Connect	Instance state ▾	Actions ▾	Launch instances	▼
		<input type="text"/> Find Instance by attribute or tag (case-sensitive)		All states ▾					
	Name D	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4	
<input type="checkbox"/>	jhansi	i-07d28bba278b40f66	Shutting-down Q C	t3.micro	The instance ID 'i-07d28bba278b40f66' is invalid. View alarms +	us-east-1f	ec2-3-236-6		
<input checked="" type="checkbox"/>	first-jenkins-pr...	i-076c884e5e2c881e5	Running Q Q	t3.micro	3/3 checks passed View alarms +	us-east-1f	ec2-3-236-1		

Fig 1: Snapshot of the EC2 instance

Add inbound rules for jenkins. [port:8080]

Step 3: Run the ssh - i command in the terminal.

ssh - i <key name> aws@<instance ip address>

Step 4: Download the Jenkins and Docker inside this instance.

Step 5: Open the jenkins through url: <http://<ip address>:8080>

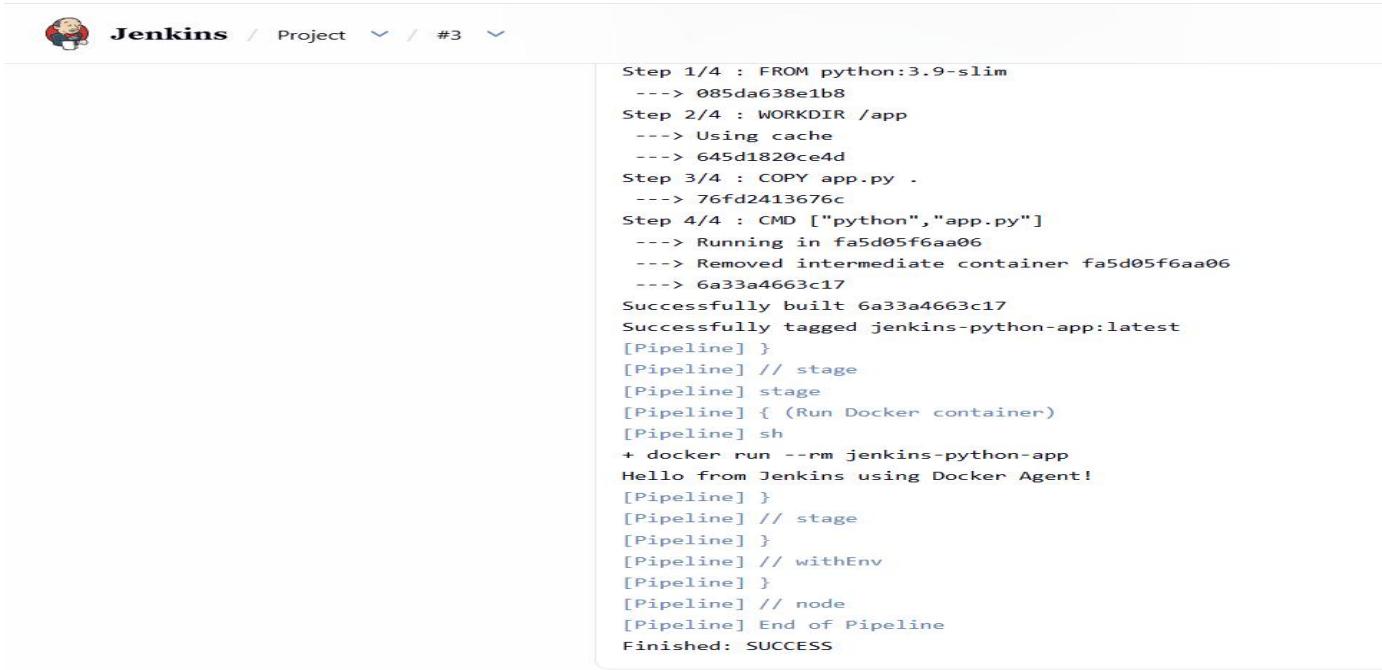
Step 6: Open and install inline pluggins.

Step 7: Then restart the jenkins.

Step 8: In settings inside pluggins, search for “Docker Pipeline” select it and install it. Then once again restart the jenkins.

Step 9: Jenkins pull the files from the github, then runs it.

Step 10:



The screenshot shows the Jenkins interface with a pipeline job named "Project #3". The console output window displays the following log:

```
Step 1/4 : FROM python:3.9-slim
--> 085da638e1b8
Step 2/4 : WORKDIR /app
--> Using cache
--> 645d1820ce4d
Step 3/4 : COPY app.py .
--> 76fd2413676c
Step 4/4 : CMD ["python", "app.py"]
--> Running in fa5d05f6aa06
--> Removed intermediate container fa5d05f6aa06
--> 6a33a4663c17
Successfully built 6a33a4663c17
Successfully tagged jenkins-python-app:latest
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run Docker container)
[Pipeline] sh
+ docker run --rm jenkins-python-app
Hello from Jenkins using Docker Agent!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Fig 2 : Final console output

Part B Section

Step 1: Create a folder for CD files, then pushes the files to github. [Github repo: Jenkins-continue]

Step 2: Create EC2 instance.

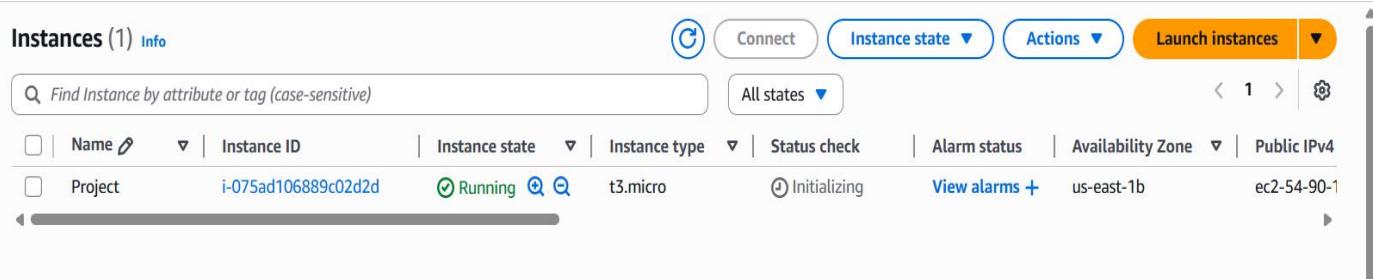


Fig 3: Snapshot of the EC2 instance

Add 2 inbound rules for jenkins. [port:8080 - jenkins and port:8081 to avoid the conflict]

Step 3: Run the ssh - i command in the terminal.

ssh - I <key name> aws@<instance ip address>

Step 4: Download the Jenkins, Docker and Docker compose inside this instance.

Step 5: Open the jenkins through url: <http://<ip address>:8080>

Step 6: Open and install inline pluggins.

Step 7: Then restart the jenkins.

Step 8: In settings inside pluggins, search for “Docker Pipeline” select it and install it. Then once again restart the jenkins.

Step 9: Jenkins pull the files from the github, then runs it.

Step 10:

A screenshot of the Jenkins Pipeline console output. The pipeline stage is named 'Pipeline-cicd'. The log shows the Jenkinsfile code being executed. It includes commands like 'sh + docker-compose up -d', 'Creating network "pipeline-cicd_default" with the default driver', 'Creating prod-python-app ... done', and 'Creating prod-python-app ... done'. It also shows the declaration of a post-action stage and an echo command with the message 'Application deployed successfully!'. The pipeline concludes with 'Finished: SUCCESS'.

Fig 4 : Final console output

Step 11: To access the application. Open the url : http://<EC2-PUBLIC-IP>:8081

Step 12:

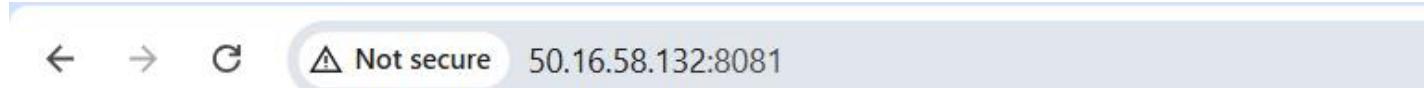


Fig 5 : Application Successfully Deployed using Jenkins