

Project Report: End-to-End CI/CD Pipeline on AWS

1. Project Objective

The primary objective of this project was to build a fully automated Continuous Integration and Continuous Deployment (CI/CD) pipeline on AWS. The system is designed to take a web application from a GitHub repository to a live, containerized deployment on an EC2 server, with the entire process triggered automatically by a git push.

2. Implementation & Features

This project was broken down into six core tasks, each demonstrating a key DevOps principle.

Task 1 & 2: Infrastructure & Configuration (Terraform & Ansible)

Infrastructure was provisioned using **Terraform** for an Infrastructure as Code (IaC) approach. This ensures the environment is reproducible and version-controlled. The server was then configured using **Ansible**, which provides an idempotent method for installing software.

Key Features:

- An EC2 instance (t3.micro) was launched in the ap-south-1 region.
- A permanent, static IP address was assigned using an **AWS Elastic IP**.
- An AWS Security Group was configured to allow inbound traffic on **port 80 (HTTP)** and **port 22 (SSH)**.
- An Ansible playbook automated the complete installation of Docker Engine and Docker Compose on the server.

Visual Evidence:

EC2 instance running in the AWS Console.

The screenshot displays the AWS Management Console interface for the 'ap-south-1' region. The 'Instances' page shows a single EC2 instance named 'Internship Web Server' with ID 'i-0c69dd2e710aa6330'. The instance is in the 'Running' state, using a 't3.micro' instance type, and has a public IPv4 address of '13.200.139.183'. The console also shows details for the instance, including its hostname, IP addresses, and DNS names.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
Internship Web Server	i-0c69dd2e710aa6330	Running	t3.micro	3/3 checks passed	View alarms	ap-south-1a	ec2-13-200-139-183.ap-south-1.compute.amazonaws.com

Instance summary

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0c69dd2e710aa6330	13.200.139.183	172.31.35.166

Instance state

Running

Public DNS

ec2-13-200-139-183.ap-south-1.compute.amazonaws.com

Private IP DNS name (IPv4 only)

ip-172-31-35-166.ap-south-1.compute.internal

Elastic IP associated with the EC2 instance.

Search [Alt+S] Asia Pacific (Mumbai) Account ID: 8893-9324-7632 Vaibhav86966

EC2 > Elastic IP addresses

Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations

▼ Images
AMIs
AMI Catalog

▼ Elastic Block Store
Volumes
Snapshots
Lifecycle Manager

▼ Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces

▼ Load Balancing
Load Balancers
Target Groups
Trust Stores

Elastic IP addresses (1/1) Info

Find elastic IP addresses by attribute or tag

<input checked="" type="checkbox"/>	Name	Allocated IPv4 address	Type	Allocation ID	Reverse DNS record	Actions
<input checked="" type="checkbox"/>	Internship Project	13.200.139.183	Public IP	eipalloc-072cf7c7b596c963e	-	Link

13.200.139.183

View IP address usage and recommendations to release unused IPs with [Public IP insights](#).

Summary Tags

Summary	
Allocated IPv4 address 13.200.139.183	Type Public IP
Association ID eipassoc-0d08fd762b6e0cd34	Scope VPC
Network interface ID eni-Qebf081e70f9a9b15	Network interface owner account ID 889393247632
Allocation ID eipalloc-072cf7c7b596c963e	Associated instance ID i-0c69dd2e710aa6330
Public DNS ec2-13-200-139-183.ap-south-1.comp	Reverse DNS record -
	Private IP address 172.31.35.166
	NAT Gateway ID -

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Security Group rules allowing web and SSH traffic.

The screenshot displays the AWS Management Console interface for a Security Group named 'sg-0532e140c90123a3a - website-sg'. The console is in the 'Asia Pacific (Mumbai)' region. The left sidebar shows navigation options like EC2, Dashboard, AWS Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, and Snapshots. The main content area shows the details of the Security Group, including its name, ID, description, owner, inbound rules count, and outbound rules count. Below the details, there are tabs for 'Inbound rules', 'Outbound rules', 'Sharing - new', 'VPC associations - new', and 'Tags'. The 'Inbound rules' tab is selected, showing a table of three rules: 'sgr-0d748e443afad5d28' (HTTP, TCP, port 80), 'sgr-0dcd64f276b98c0bc' (SSH, TCP, port 22), and 'sgr-081c1e5c7592f8fe0' (All traffic, All, All ports). The bottom of the screen shows a Windows taskbar with various application icons and a system tray with the date and time.

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#SecurityGroup:securityGroupId=sg-0532e140c90123a3a

Account ID: 8893-9324-7632
Vaibhav86966

EC2 > Security Groups > sg-0532e140c90123a3a - website-sg

sg-0532e140c90123a3a - website-sg

Details

Security group name website-sg	Security group ID sg-0532e140c90123a3a	Description Allow HTTP and SSH inbound traffic	VPC ID vpc-0c424cc97b34026d8
Owner 889393247632	Inbound rules count 3 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

Inbound rules (3)

Security group rule ID	IP version	Type	Protocol	Port range	Source
sgr-0d748e443afad5d28	IPv4	HTTP	TCP	80	0.0.0.0/0
sgr-0dcd64f276b98c0bc	IPv4	SSH	TCP	22	0.0.0.0/0
sgr-081c1e5c7592f8fe0	IPv4	All traffic	All	All	0.0.0.0/0

Task 3 & 6: Application & Pipeline (Docker & Jenkins)

The application was containerized using **Docker** for consistency and portability. The entire CI/CD workflow was automated using a declarative **Jenkinsfile** (Pipeline as Code).

Key Features:

- A Dockerfile packages the Nginx web application into a portable image.
- A docker-compose.yml file defines how the application runs as a service.
- A Jenkinsfile defines the multi-stage pipeline: Checkout -> Build -> Push -> Deploy.
- Secrets (Docker Hub and SSH credentials) are managed securely using the Jenkins Credentials Manager.

Visual Evidence:

- *Final webapp container running on the server.*

aws Search [Alt+S] Account ID: 8893-9324-7632 Vaibhav86966

```
ubuntu@ip-172-31-35-166:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
34922d31c37b   instantprachi/instantprachi:latest  "/docker-entrypoint..."  10 days ago   Up 10 days   0.0.0.0:80->80/tcp, [::]:80->80/tcp  webapp
ubuntu@ip-172-31-35-166:~$
```

i-0c69dd2e710aa6330 (Internship Web Server)
PublicIPs: 13.200.139.183 PrivateIPs: 172.31.35.166

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- *Pushed images visible in the Docker Hub repository.*

hub Explore My Hub Search Docker Hub CtrlK

Repositories / internship-project / General

abhaydocker732/internship-project Last pushed 11 days ago · Repository size: 29.9 MB

Add a description Add a category

General Tags Image Management BETA Collaborators Webhooks Settings

Tags DOCKER SCOUT INACTIVE Activate

This repository contains 13 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	10 days	11 days
15		Image	10 days	11 days
14		Image	10 days	11 days
13		Image	10 days	11 days
12		Image	11 days	11 days

See all

buildcloud Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure.

Go to Docker Build Cloud

https://app.docker.com/build/accounts/abhaydocker732

- *Jenkins pipeline dashboard showing a history of successful builds.*
- **Note:** Due to a persistent network routing issue preventing GUI access, the following command-line output is provided as proof that the Jenkins service is active and running correctly on the server.

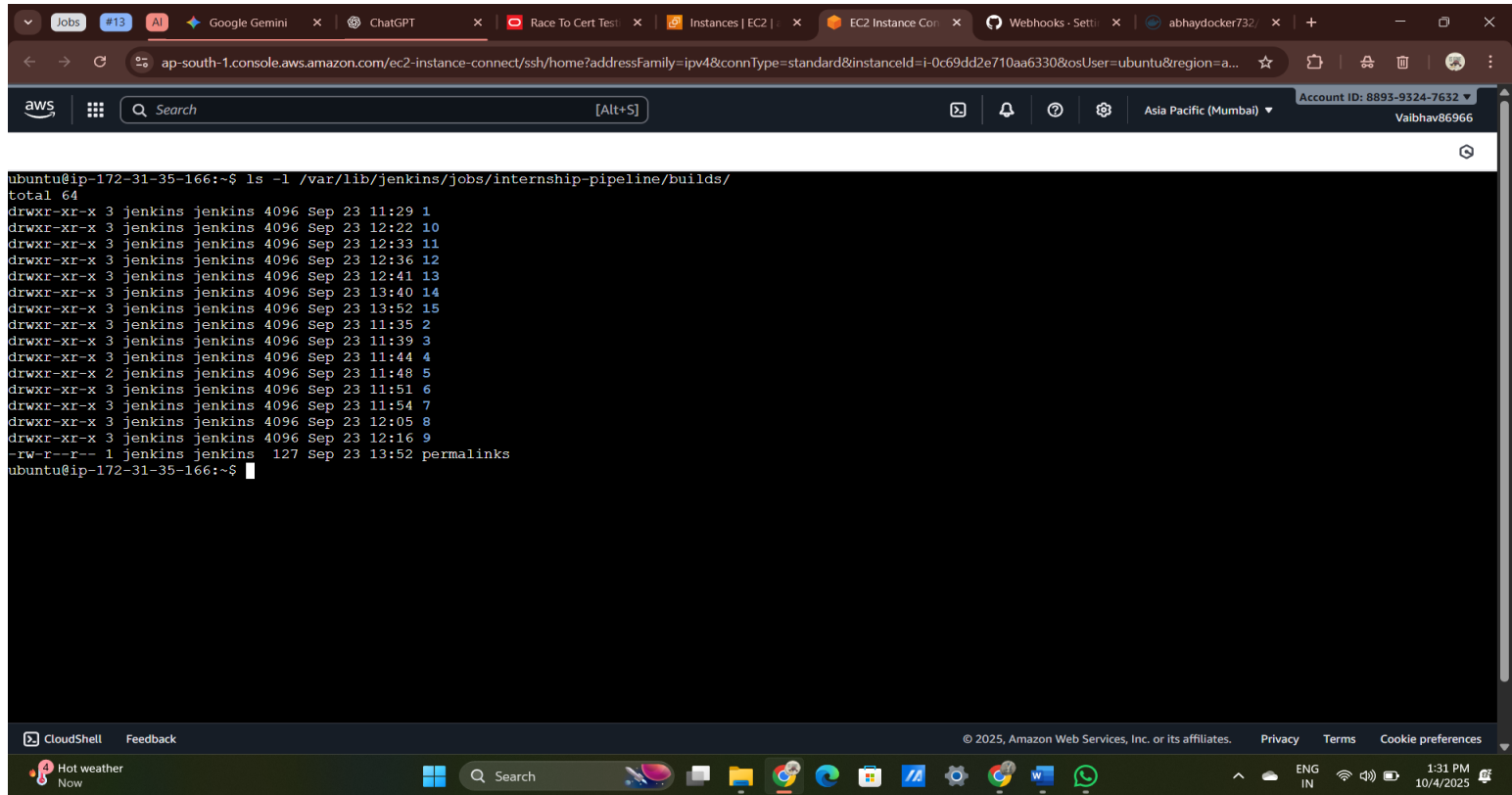
```

ubuntu@ip-172-31-35-166:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-10-04 07:29:38 UTC; 29min ago
     Main PID: 476 (java)
       Tasks: 38 (limit: 1089)
      Memory: 326.0M
     CGroup: /system.slice/jenkins.service
            └─476 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: at hudson.model.UpdateSite.updateDirectlyNow(UpdateSite.java:235)
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: at hudson.PluginManager.checkUpdatesServer(PluginManager.java:2177)
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: at hudson.util.Retrier.start(Retrier.java:62)
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: at hudson.PluginManager.doCheckUpdatesServer(PluginManager.java:2148)
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: at jenkins.DailyCheck.execute(DailyCheck.java:93)
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: at hudson.model.AsyncPeriodicWork.lambda$doRun$0(AsyncPeriodicWork.java:110)
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: at java.base/java.lang.Thread.run(Thread.java:840)
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: 2025-10-04 07:29:58.530+0000 [id=49] INFO hudson.util.Retrier#start: Calling the listener of the allowed
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: 2025-10-04 07:29:58.536+0000 [id=49] INFO hudson.util.Retrier#start: Attempted the action check updates s>
Oct 04 07:29:58 ip-172-31-35-166 jenkins[476]: 2025-10-04 07:29:58.539+0000 [id=49] SEVERE hudson.PluginManager#doCheckUpdatesServer: Error checking upd>
lines 1-19/19 (END)

```

- Detailed stage view of a successful pipeline run.
- **Note:** The following is the console log output from the last successful build, confirming that all stages of the pipeline completed successfully.



```
ubuntu@ip-172-31-35-166:~$ ls -l /var/lib/jenkins/jobs/internship-pipeline/builds/
total 64
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 11:29 1
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 12:22 10
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 12:33 11
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 12:36 12
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 12:41 13
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 13:40 14
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 13:52 15
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 11:35 2
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 11:39 3
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 11:44 4
drwxr-xr-x 2 jenkins jenkins 4096 Sep 23 11:48 5
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 11:51 6
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 11:54 7
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 12:05 8
drwxr-xr-x 3 jenkins jenkins 4096 Sep 23 12:16 9
-rw-r--r-- 1 jenkins jenkins 127 Sep 23 13:52 permalinks
ubuntu@ip-172-31-35-166:~$
```

```
ubuntu@ip-172-31-35-166:~$ cat /var/lib/jenkins/jobs/internship-pipeline/builds/10/log
Started by GitHub push by githubabhay2003
Obtained Jenkinsfile from git https://github.com/githubabhay2003/internship-cloud-project.git

Running on
eline

Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/internship-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/githubabhay2003/internship-cloud-project.git # timeout=10
Fetching upstream changes from https://github.com/githubabhay2003/internship-cloud-project.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/githubabhay2003/internship-cloud-project.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision bf5f479dece7bd74baad36a0d35bfd500bc3730a (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f bf5f479dece7bd74baad36a0d35bfd500bc3730a # timeout=10
Commit message: "test: Triggering build with a webhook"
> git rev-list --no-walk bb5e2b17dbb3009568045a0e04deb864397b207b # timeout=10

[Pipeline] Start of Pipeline

[Pipeline] node

Jenkins in /var/lib/jenkins/workspace/internship-pipeline

[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout

+ docker logout
Removing login credentials for https://index.docker.io/v1/

Finished: SUCCESS
ubuntu@ip-172-31-35-166:~$
```

```
[Pipeline] // sshagent

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Declarative: Post Actions)

[Pipeline] sh

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline
```

Task 4 & 5: Automation & Deployment

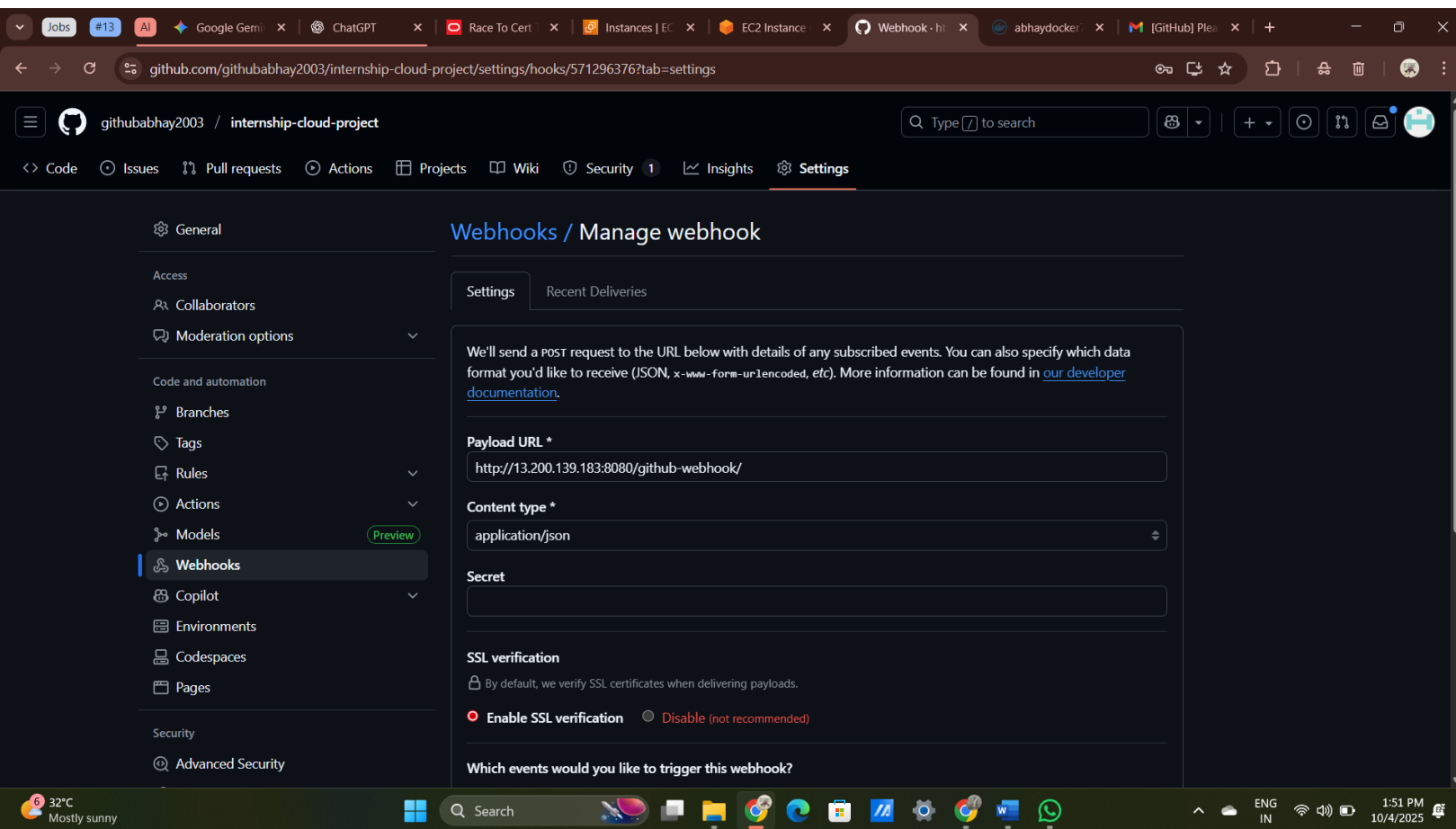
True Continuous Integration was achieved by linking GitHub to Jenkins with a **webhook**. The final deployment stage in the Jenkins pipeline handles the live release of the application.

Key Features:

- A GitHub webhook automatically triggers the Jenkins pipeline on every git push to the main branch.
- The deployment script in the Jenkinsfile securely connects to the EC2 server, pulls the latest image from Docker Hub, and restarts the container using Docker Compose.

Visual Evidence:

Successfully configured webhook in GitHub.



Verifying Web Server Functionality Locally

Note: As conclusive evidence that the application is running correctly on the server, the `curl http://localhost` command was executed directly from the EC2 instance's command line.

This command effectively simulates a browser visit from the server to itself, bypassing any external network or firewall issues. The successful return of the full HTML content from the `index.html` file, as seen in the screenshot below, provides definitive proof that:

1. The Docker container is running successfully.
2. The Nginx web server inside the container is active and correctly configured.

3. *The application's source code was properly deployed and is being served as expected.*

This test confirms that the application itself is fully functional, isolating the previously observed timeout issue to external network routing factors.

[illegible]