

# Capstone-project-1

You have been hired as a Sr. DevOps Engineer in Abode Software. They want to implement DevOps Lifecycle in their company. You have been asked to implement this lifecycle as fast as possible. Abode Software is a product-based company and their product is available on this GitHub link.

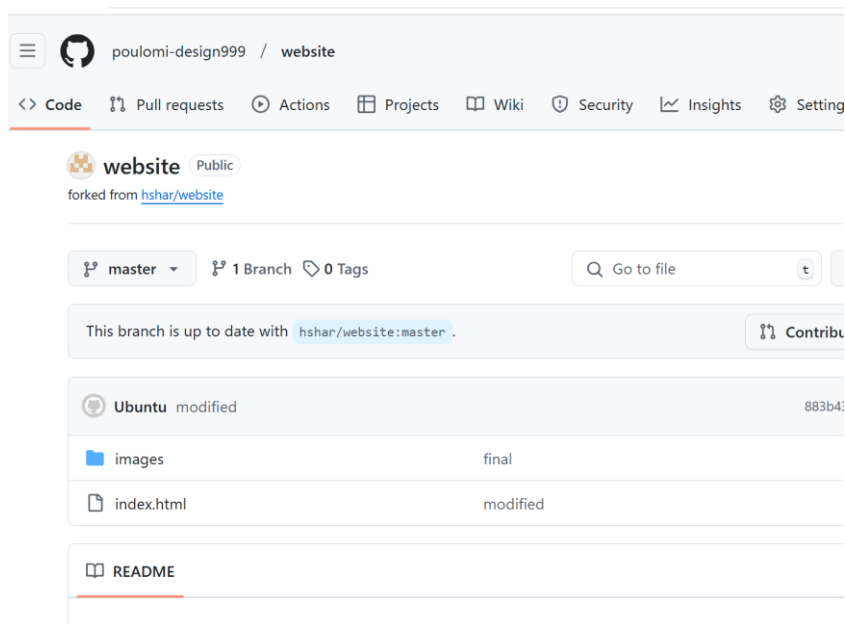
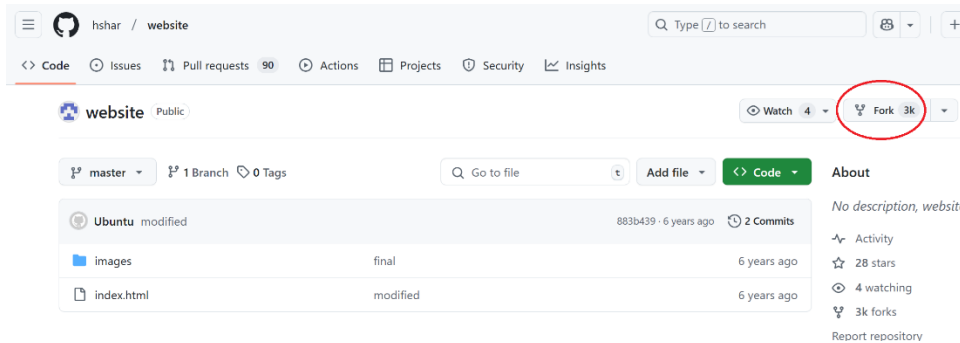
<https://github.com/hshar/website.git>

Following are the specifications of the lifecycle:

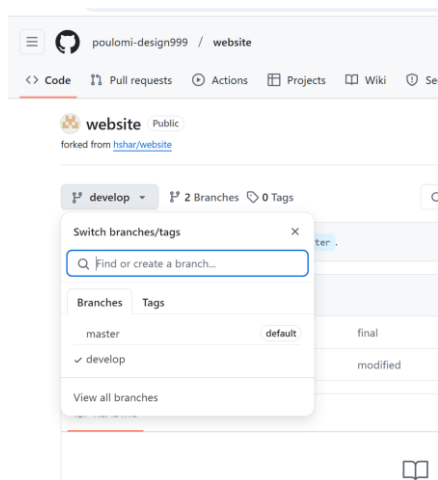
1. Install the necessary software on the machines using a configuration management tool
2. Git workflow has to be implemented
3. CodeBuild should automatically be triggered once a commit is made to master branch or develop branch.
  - a. If a commit is made to master branch, test and push to prod
  - b. If a commit is made to develop branch, just test the product, do not push to prod
4. The code should be containerized with the help of a Dockerfile. The Dockerfile should be built every time there is a push to GitHub. Use the following pre-built container for your application: hshar/webapp  
The code should reside in '/var/www/html'
5. The above tasks should be defined in a Jenkins Pipeline with the following jobs:
  - a. Job1 : build
  - b. Job2 : test
  - c. Job3 : prod

## Solution:

### 1) Fork the github link <https://github.com/hshar/website.git>



### Create a develop branch :



### 2) Create 3 EC2 [ test, prod and master ] > connect and update all of them



```
ubuntu@ip-172-31-8-1:~$ cd .ssh
ubuntu@ip-172-31-8-1:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-8-1:~/.ssh$ vi authorized_keys
ubuntu@ip-172-31-8-1:~/.ssh$
```

- 5) Go to master server > cd /etc/ansible > ls > vi hosts

```
ubuntu@ip-172-31-4-35:~/.ssh$ cd
ubuntu@ip-172-31-4-35:~$ cd /etc/ansible
ubuntu@ip-172-31-4-35:/etc/ansible$ ls
ansible.cfg  hosts  roles
ubuntu@ip-172-31-4-35:/etc/ansible$ vi hosts
```

**i-0d50e93fdc45520e7 (master)**

- 6) Copy the private IP of slave servers [ test and prod ]

```
## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Ex4: Multiple hosts arranged into groups

## [Debian]
## alpha.example.org
## beta.example.org

## [openSUSE]
## green.example.com
## blue.example.com
[slave]
172.31.10.122
172.31.8.1
-- INSERT --
```

**i-0d50e93fdc45520e7 (master)**

- 7) Check the connection between master and slave servers  
ansible -m ping all

```

ubuntu@ip-172-31-4-35:/etc/ansible$ ansible -m ping all
[WARNING]: Platform linux on host 172.31.8.1 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another
Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
172.31.8.1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3.12"
  },
  "changed": false,
  "ping": "pong"
}
[WARNING]: Platform linux on host 172.31.10.122 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of an
other Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
172.31.10.122 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3.12"
  },
  "changed": false,
  "ping": "pong"
}
ubuntu@ip-172-31-4-35:/etc/ansible$

```

- 8) Now we create a playbook [vi play.yml](#) in master with the following code .  
according to it the script master.sh and slaves.sh will run when we play this playbook.

```

---
- name: task for master mc
  hosts: localhost
  become: true
  tasks:
    - name: master script
      script: master.sh
- name: task for slave mc
  hosts: slave
  become: true
  tasks:
    - name: slaves script
      script: slaves.sh

```

- 9) We create master.sh and slaves.sh [script file] which will run in playbook. Inside this script all the software dependencies will be downloaded as per the problem statement.

master.sh : install java & install jenkins

```

sudo apt update
sudo apt install openjdk-17-jre -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/" | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y

```

slaves.sh : install java & docker

```

sudo apt update
sudo apt install openjdk-17-jre -y
sudo apt install docker.io -y

```

## 10) Check the playbook syntax

```
ubuntu@ip-172-31-4-35:~$ ansible-playbook play.yml --syntax-check

playbook: play.yml
ubuntu@ip-172-31-4-35:~$
```

## 11) Check for errors , dry run playbook

```
ubuntu@ip-172-31-4-35:~$ ansible-playbook play.yml --check

PLAY RECAP *****
172.31.10.122      : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.31.8.1        : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
localhost         : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

## 12) Execute playbook

```
ubuntu@ip-172-31-4-35:~$ ansible-playbook play.yml
```

## Access Jenkins

### 1) Copy and paste the public IP:8080 (master) on the browser and we can see unlock Jenkins page

### Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

### 2) Unlock Jenkins > download plugins > create a user > we can access Jenkins dashboard

## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

### Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

## Jenkins

Dashboard

- New Item
- Build History
- Manage Jenkins
- My Views

Build Queue

No builds in the queue.

Build Executor Status

0/2

### Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

+

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

### 3) Manage > security > agents > random > save

Dashboard > Manage Jenkins > Security [? \(not saved\)](#)

---

#### Markup Formatter

Markup Formatter [?](#)

Plain text

Shows descriptions mostly as written. HTML unsafe characters like

---

#### Agents

TCP port for inbound agents [?](#)

☐ Fixed

☒ Random

☐ Disable

### 4) Create slave 1 : manage > nodes > give name > create

Dashboard > Manage Jenkins > Nodes > New node

---

#### New node

Node name

slave 1

Type

☒ Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

### 5) Give root dir : /home/ubuntu/Jenkins > launch method: ssh > host : private IP of slave

Dashboard > Manage Jenkins > Nodes >

---

Name [?](#)

slave 1

---

Description [?](#)

Plain text [Preview](#)

Number of executors [?](#)

1

Remote root directory [?](#)

/home/ubuntu/jenkins

Labels [?](#)

Usage [?](#)

Use this node as much as possible

---

Usage [?](#)

Use this node as much as possible

---

Launch method [?](#)

Launch agents via SSH

---

Host [?](#)

172.31.10.122

Credentials [?](#)

ubuntu

+ Add

Host Key Verification Strategy [?](#)

Non verifying Verification Strategy

Advanced [v](#)

---

Availability [?](#)

Keep this agent online as much as possible

6) Under credentials : choose ssh , username : ubuntu , key : paste the key pair of EC2

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

SSH Username with private key

Scope

Global (Jenkins, nodes, items, all child items, etc)

Description

Username

ubuntu

☒ Treat username as secret

Private Key

☒ Enter directly

Key

-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----

Passphrase

Slave 1 created.

7) Create slave 2 .

Nodes

[+ New Node](#) [Configure M](#)

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response T
	Built-In Node	Linux (amd64)	In sync	3.33 GiB	0 B	3.33 GiB	
	slave 1	Linux (amd64)	In sync	3.98 GiB	0 B	3.98 GiB	
	slave 2	Linux (amd64)	In sync	3.98 GiB	0 B	3.98 GiB	
Data obtained		12 sec	12 sec	12 sec	12 sec	12 sec	

Create jobs

## New Item

Enter an item name

job 1

Select an item type



### Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, e steps like archiving artifacts and sending email notifications.

1)

Pipeline

## Source Code Management

Connect and manage your code repository to automatically pull the latest code for

☐ None

☒ Git

Repositories

Repository URL

https://github.com/poulomi-design999/website.git

Credentials

ubuntu

[+ Add](#)

Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

\*/develop

2)

☐ Discard old builds

☒ GitHub project

Project url

https://github.com/poulomi-design999/website.git

Advanced

☐ This project is parameterized

☐ Throttle builds

☐ Execute concurrent builds if necessary

☒ Restrict where this project can be run

Label Expression

slave 1



## Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

3) Create job 2 : branch = \*/master [ all same as job 1]

4) Create job 3 : branch = \*/master , label :slave 2 [ all same as job 1]

S	W	Name	Last Success	Last Failure	Last Duration	
...	☀	job 1	N/A	N/A	N/A	▶
...	☀	job 2	N/A	N/A	N/A	▶
...	☀	job 3	N/A	N/A	N/A	▶

The screenshot shows the Jenkins Dashboard with a table of jobs. The table has columns for Status (S), Warnings (W), Name, Last Success, Last Failure, Last Duration, and a link icon. The jobs listed are job1, job2, and job3. job1 has a last success of 1 min 30 sec, job2 has 1 min 9 sec, and job3 has 39 sec. The dashboard also shows a sidebar with navigation links and a 'Build Queue' section.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	job1	1 min 30 sec #1	N/A	6.7 sec
✓	☀	job2	1 min 9 sec #1	N/A	0.86 sec
✓	☀	job3	39 sec #1	N/A	6.1 sec

## Github

1) Create a webhook in the repo

Go to the repo > settings > webhook >

under URL : mention Jenkins URL/github-webhook/

The screenshot shows the GitHub repository settings page for 'poulomi-design999 / website'. The 'Webhooks' section is selected in the left sidebar. The 'Settings' tab is active, showing the 'Payload URL' field with the value 'http://43.204.98.83:8080/github-webhook/'. The 'Content type' is set to 'application/json'.

Webhooks / Manage webhook

Settings

We'll send a POST request to the URL below with details of any subscribed events. You can a format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be fo [documentation](#).

Payload URL \*

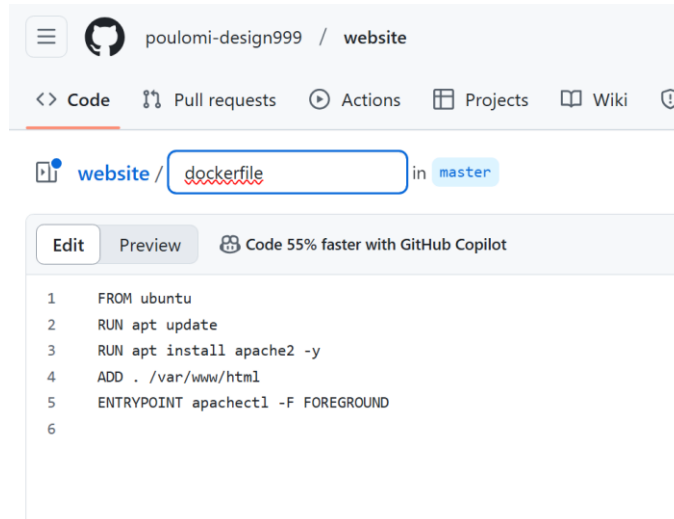
http://43.204.98.83:8080/github-webhook/

Content type \*

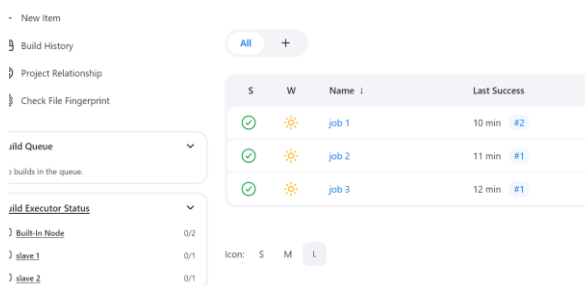
application/json

Secret

## 2) Create a docker file in your repo

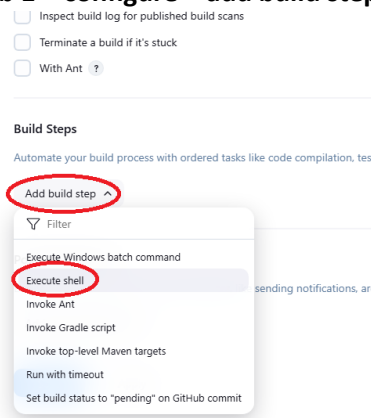


## 3) We can see that after commit the jobs ran successfully automatically.



## 4) Dockerize the docker file

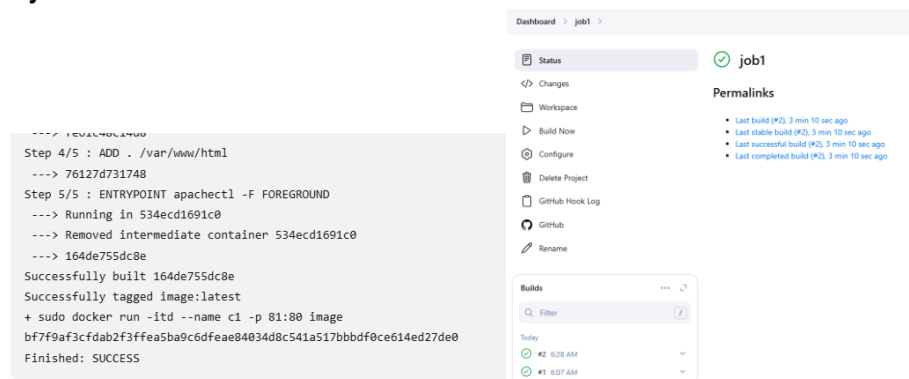
### Job 1 > configure > add build step > execute shell



- 5) Write the code to build image from dockerfile and containerize and run on a specific port.



- 6) Build job1 and we can see it was successful



- 7) For job 2 & 3 follow the same steps to build image from dockerfile and containerize and run on a specific port.
- 8) Paste the public IP of test machine and see the default apache2 page

