

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 must 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

Procedure: -

- Ansible master= jenkins and java with the help of ansible
- java, docker on the slaves
- one = master, two and three= slaves. Total three instances
- install ansible on the master
- keygen to connect all the slave with the master
- sudo nano /etc/ansible/hosts and paste the private ips of both of the slaves

```
GNU nano 6.2 /etc/ansible/hosts
10.0.15.235
10.0.15.236
10.0.15.236

i-Offe665da173866eb (capstone-master)
PublicIPs: 35.172.229.153 PrivateIPs: 10.0.2.40
```

- save and exit from the file.
- Check whether we are successfully ping the test and prod “ansible -m ping all”

```
--ssh-extra-args SSH_EXTRA_ARGS
specify extra arguments to pass to ssh only (e.g. -R)
-T TIMEOUT, --timeout TIMEOUT
override the connection timeout in seconds (default depends on connection)
-c CONNECTION, --connection CONNECTION
connection type to use (default=ssh)
-u REMOTE_USER, --user REMOTE_USER
connect as this user (default=None)

Some actions do not make sense in Ad-Hoc (include, meta, etc)
ubuntu@ip-10-0-2-40:~$ ansible -m ping all
10.0.15.236 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
10.0.15.235 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ubuntu@ip-10-0-2-40:~$

i-Offe665da173866eb (capstone-master)
PublicIPs: 35.172.229.153 PrivateIPs: 10.0.2.40
```

- Now we will create playbook to install jenkins and java on the slave machine
- Sudo nano.play.yml

```
---
- name: tasks for the master
  hosts: localhost //master machine
  become: true
  tasks:
    - name: executing script on master
      script: master.sh

- name: tasks for slave
```

```

hosts: all //refers to all the slave machine
become: true
tasks:
  - name: executing script on slave
    script: slave.sh

```

➤ sudo nano master.sh

```

sudo apt update
sudo apt install openjdk-11-jdk -y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins
-----or-----

```

➤ slave.sh

```

sudo apt update
sudo apt install openjdk-11-jdk -y
sudo apt install docker.io -y

```

- ansible-playbook play.yaml --syntax-check
- ansible-playbook play.yaml --check //for the dry run
- ansible-playbook play.yaml

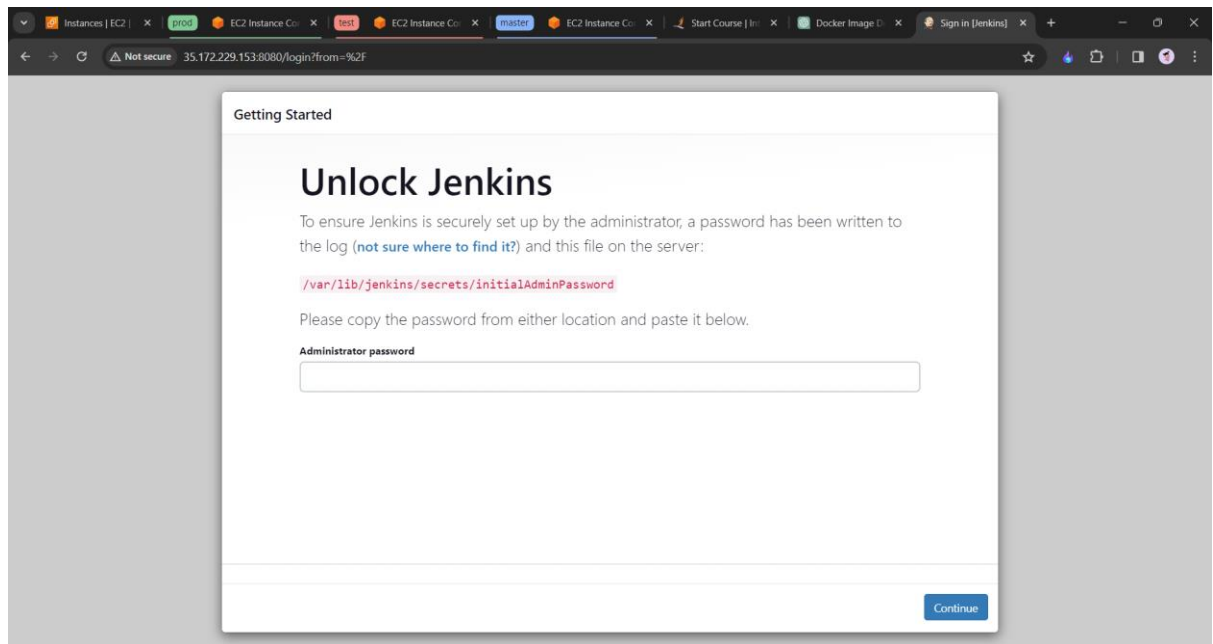
```

**
ok: [localhost]
TASK [executing script on master] *****
**
changed: [localhost]
PLAY [tasks for slave] *****
**
TASK [Gathering Facts] *****
**
ok: [10.0.15.235]
ok: [10.0.15.236]
TASK [executing script on slave] *****
**
changed: [10.0.15.235]
changed: [10.0.15.236]
PLAY RECAP *****
10.0.15.235      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
10.0.15.236      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost       : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ubuntu@ip-10-0-2-40:~$
i-Offe665da173866eb (capstone-master)
PublicIP: 35.172.229.153  PrivateIP: 10.0.2.40

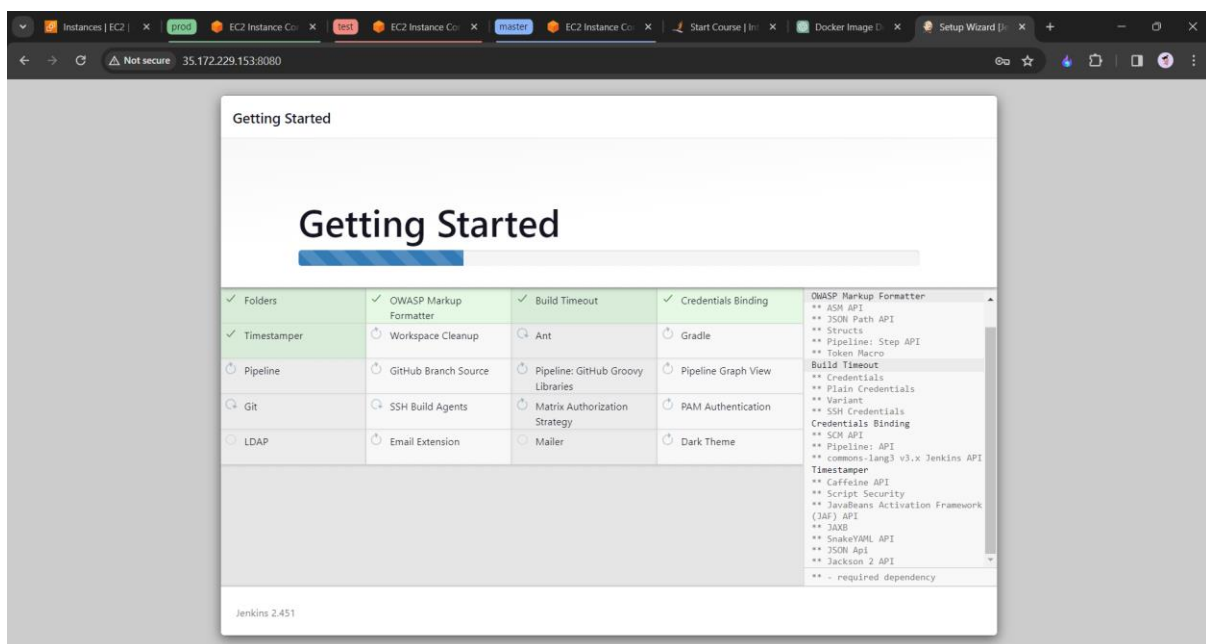
```

- We need to set the jenkins dashborad and to add the nodes over jenkins

Dashboard setting of jenkins



- Unlock jenkins with the password
- Install the suggested pluggins



- We need to create the admin user(username,password and emailID)
- Please install the plugin called “SSH Agent” to connect the nodes.
- We are ready to see the dashboard.
- We will add the nodes now.
- Goto to manage jenkins>add nodes>newnode>
- Give “name” permanent agent and create
- Add description “remote root directory as :home/ubuntu/jenkins

- Launch method. Launch agent via ssh
- Hosts : add ip address of the slave private ip
- Credentials: click on add and click on jenkins “select kind ssh username with private key”
- Below scroll username as ubuntu private key copy its content and paste it
- Host key verification strategy to non verifying strategy
- Click on save

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.46 GiB	0 B	4.46 GiB	0ms
	slave1	Linux (amd64)	In sync	5.09 GiB	0 B	5.09 GiB	31ms
	slave2	Linux (amd64)	In sync	5.09 GiB	0 B	5.09 GiB	46ms
Data obtained			0.41 sec	0.42 sec	0.39 sec	0.38 sec	0.39 sec

Icon: S M L Legend

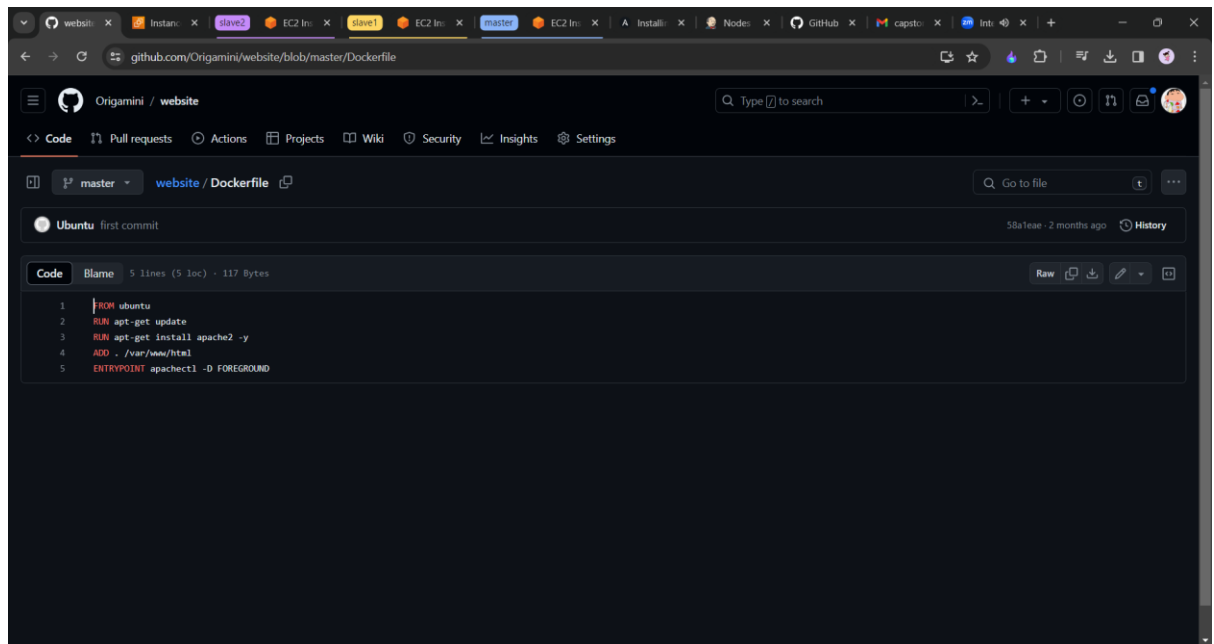
REST API Jenkins 2.452

We will create the repository in github that we are going to use for the project.

- Fork the repository that is give by the project.
- Then we will create the docker file to containerize
- On the github create a new file with the name “Dockerfile”

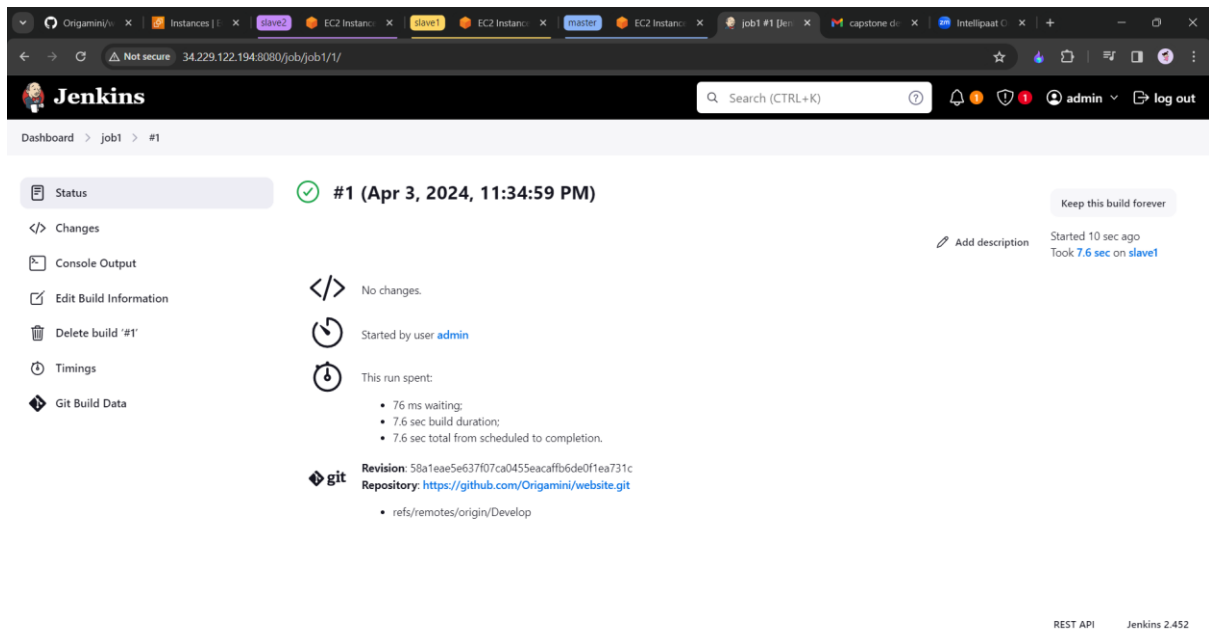
```
FROM ubuntu
RUN apt update
RUN apt install apache2 -y
ADD . /var/www/html
ENTRYPOINT apachectl -D FOREGROUND
```

- Commit the changes
- We need to also create a new branch with the name develop.



Now we will create the jobs on the jenkins dashboard.

- Job1 = slave1-develop, job2=slave1-main, job3=slave2-main
- Click on new item name it as job1 and select freesyle project click on next.
- Restrict where this project can be run checkbox
- Slave1 click on that
- Source code management select "git"
- Repo url :https://
- Branches to build : name of the branch "*/develop"
- Goto buildstep and select execute shell



```
ubuntu@ip-10-0-2-189:~$ cd home
ubuntu@ip-10-0-2-189:~/home$ ls
ubuntu
ubuntu@ip-10-0-2-189:~/home$ cd ubuntu
ubuntu@ip-10-0-2-189:~/home/ubuntu$ ls
jenkins
ubuntu@ip-10-0-2-189:~/home/ubuntu$ cd jenkins
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins$ ls
home  remoting.jar
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins$ cd /home/ubuntu/jenkins
-bash: cd: /home/ubuntu/jenkins: No such file or directory
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins$ cd home
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home$ ls
ubuntu
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home$ cd ubuntu
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu$ ls
jenkins
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu$ cd jenkins
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu/jenkins$ ls
remoting workspace
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu/jenkins$ cd workspace
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu/jenkins/workspace$ ls
job1
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu/jenkins/workspace$ cd job1
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu/jenkins/workspace/job1$ ls
Dockerfile  images  index.html
ubuntu@ip-10-0-2-189:~/home/ubuntu/jenkins/home/ubuntu/jenkins/workspace/job1$
```

i-0a2ef9d7c3fc4f837 (pr-slave1)
PublicIPs: 52.6.212.233 PrivateIPs: 10.0.2.189

After this again configure the file with the following command.

```
sudo docker rm -f c1
sudo docker build /home/ubuntu/home/ubuntu/jenkins/home/ubuntu/jenkins/workspace/job1/ -t job1
//name of the image
sudo docker run -itd -p 83:80 --name=c1 job1
```



Goto to build trigger and click on github hook trigger for GitSCM polling

Now we will create webhook

- Goto to settings and webhooks click on add webhook put the url of the jenkins
- <https://address/github-webhook/>
- Refresh it
- Now if we make any changes to the repository the job1 will be automatically executed.
- Now we will not be able to add the sudo docker rm -f c1 command,

Now we will create the job2(slave1 machine, select the master branch) and job3(select the slave2 machine , select the master branch)