

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

Building a CI/CD Pipeline for a Retail Company - ABC Technologies.

Goals:

- Highly Available
- Highly scalable
- Highly performant
- Easily built and maintained
- Developed and deployed quickly
- Lower production bugs
- Frequent releases

Prerequisites:

- Java
- Maven
- Git
- Jenkins
- Docker
- Ansible
- Kubernetes
- Grafana
- Prometheus

My Approach:

Steps:

- **Cloning the project:** Cloned the project source code from GitHub and download it to the local machine.
- **Creating a new GitHub repository:** Created a new GitHub repository to host the project code and enable collaboration - https://github.com/Majormekzy/abc_technologies.git
- **Initializing Git:** Initialized Git on the local machine to start tracking changes made to the project code.
- **Pushing the source code to the new repository:** Pushed the cloned project code to the new GitHub repository to make it available to other collaborators.
- **Provisioning servers:** Provisioned three servers for the integration: Server A (Jenkins Master, Ansible Master, Docker), Server B (K8s Master, Ansible Node, Docker, Prometheus & Grafana), and Server C (K8s Node).
- **Configuring Server A:** Configured Jenkins on Server A, install the necessary plugins for pipeline integration, installed and configured Ansible for communication with Server B, install Docker and integrate it with Jenkins, and integrate Ansible with Docker and Kubernetes.

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

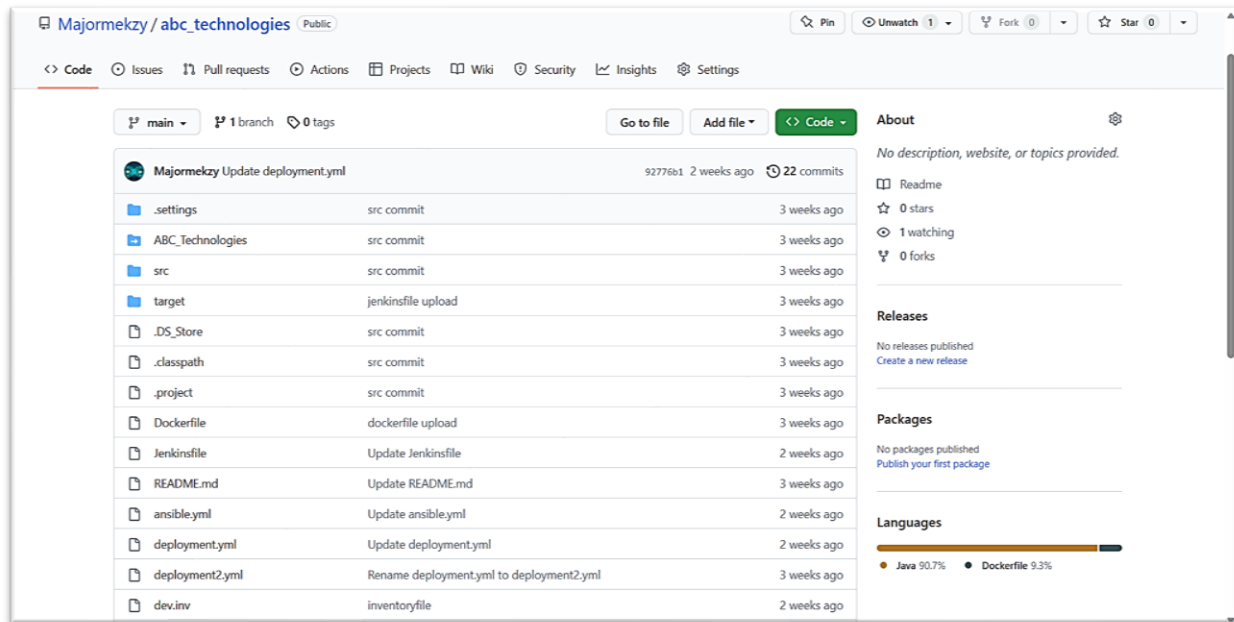
- **Configuring Server B:** Installed Kubernetes on Server B, join the node in Server C, create an Ansible user and configure SSH connection with Server A's master node, and install Docker.
- **Configuring Server C:** Configured K8s Node 1 on Server C.
- **Creating a CI/CD Pipeline in Jenkins:** Created a pipeline to automate the software delivery process and facilitate continuous integration and continuous delivery (CI/CD).
- **Configuring Jenkins with Maven tool:** Configured Jenkins to use Maven as a build tool for the pipeline.
- **Creating the Jenkinsfile:** Created a Jenkinsfile that defines the pipeline stages, including compilation, testing, and packaging.
- **Executing Maven commands:** Executed the Maven commands specified in the Jenkinsfile to compile, test, and package the project code.
- **Configuring Pipeline Build triggers and GitHub Webhook:** Configured the pipeline to trigger builds automatically on specific events, such as code changes pushed to the GitHub repository.
- **Configuring Docker and Ansible in Jenkins Global Tool Configuration:** Configured Docker and Ansible in the Jenkins Global Tool Configuration to make them available in the pipeline.
- **Creating a Docker repository:** Created a new Docker repository on Docker Hub to host the Docker images built in the pipeline.
- **Creating the Dockerfile:** Created a Dockerfile that defines the environment and dependencies required to run the project code.
- **Updating the Jenkinsfile for Docker build and push:** Updated the Jenkinsfile to include the Docker build and push stages to push the Docker images to the Docker repository.
- **Configuring K8S cloud on Jenkins:** Configured the Kubernetes cloud on Jenkins to enable deployment to Kubernetes.
- **Creating an Ansible playbook:** Created an Ansible playbook that deploys the project artifacts to Kubernetes, including the deployment and service.
- **Updating the Jenkinsfile to execute Ansible playbook:** Updated the Jenkinsfile to execute the Ansible playbook and deploy the containerized application to the Kubernetes server.
- **Monitoring K8s cluster using Prometheus & Grafana:** Installed Prometheus and Grafana on the K8s Master node to monitor the Kubernetes cluster

Post Graduate Certification Program in DevOps

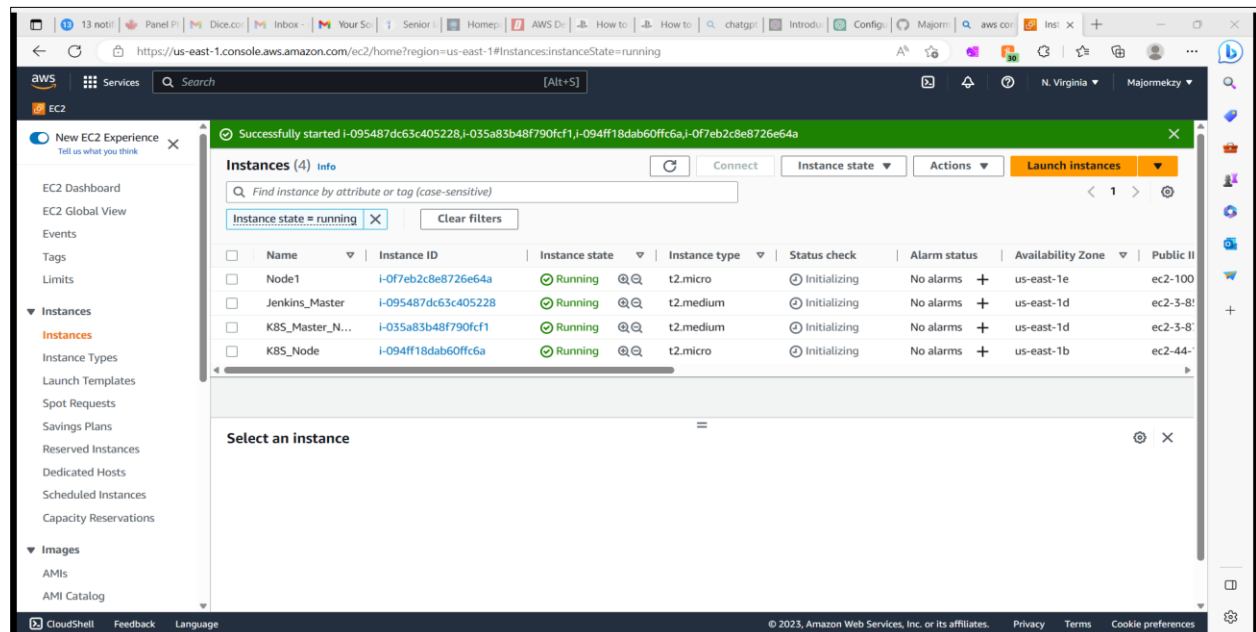
ABC-Technologies Edureka Project 1

Snapshots:

Github repository containing all the deployment file, ansible playbook and jenkinsfile



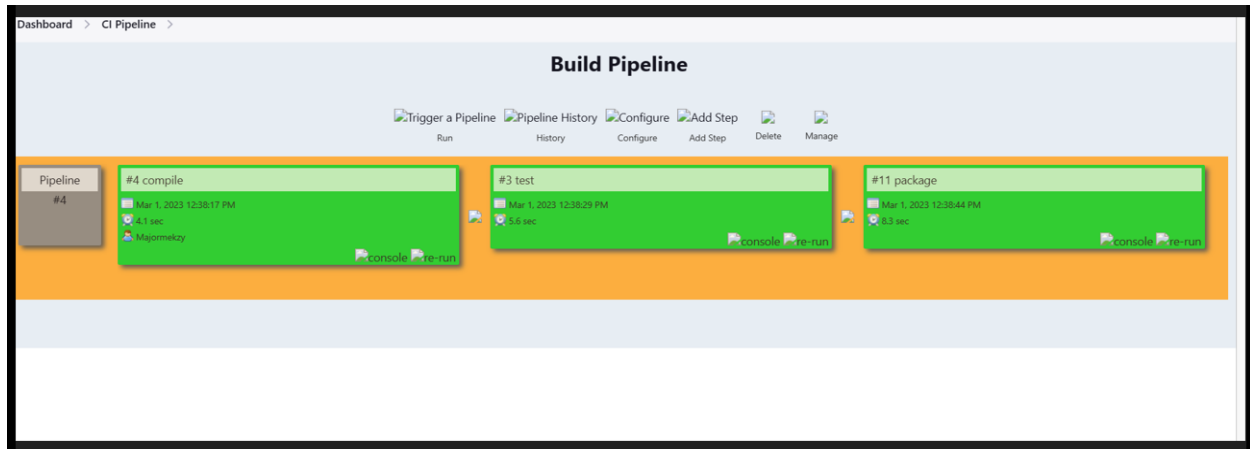
Provisioned four virtual servers on AWS namely: Jenkins master, slave node, K8s master and node



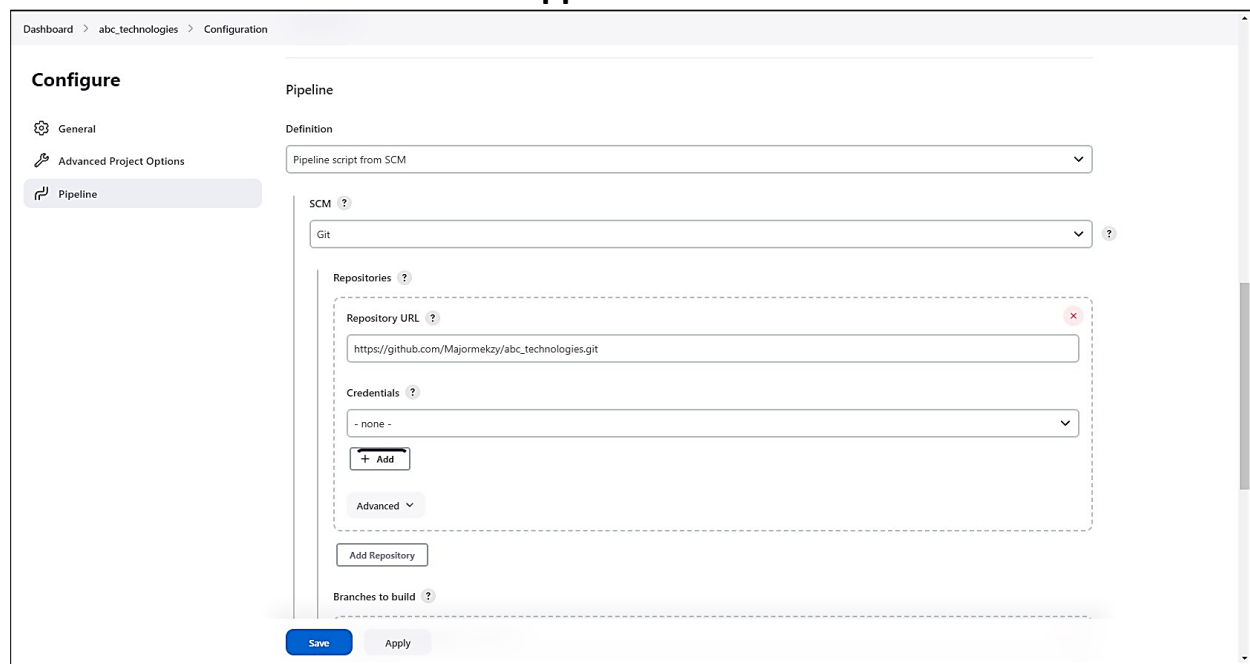
Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

Using the Jenkins slave node to perform the maven compile, test and packaging of the src



Setting up the main Jenkins pipeline to perform the build, test, docker image build and containerization of the application.



Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

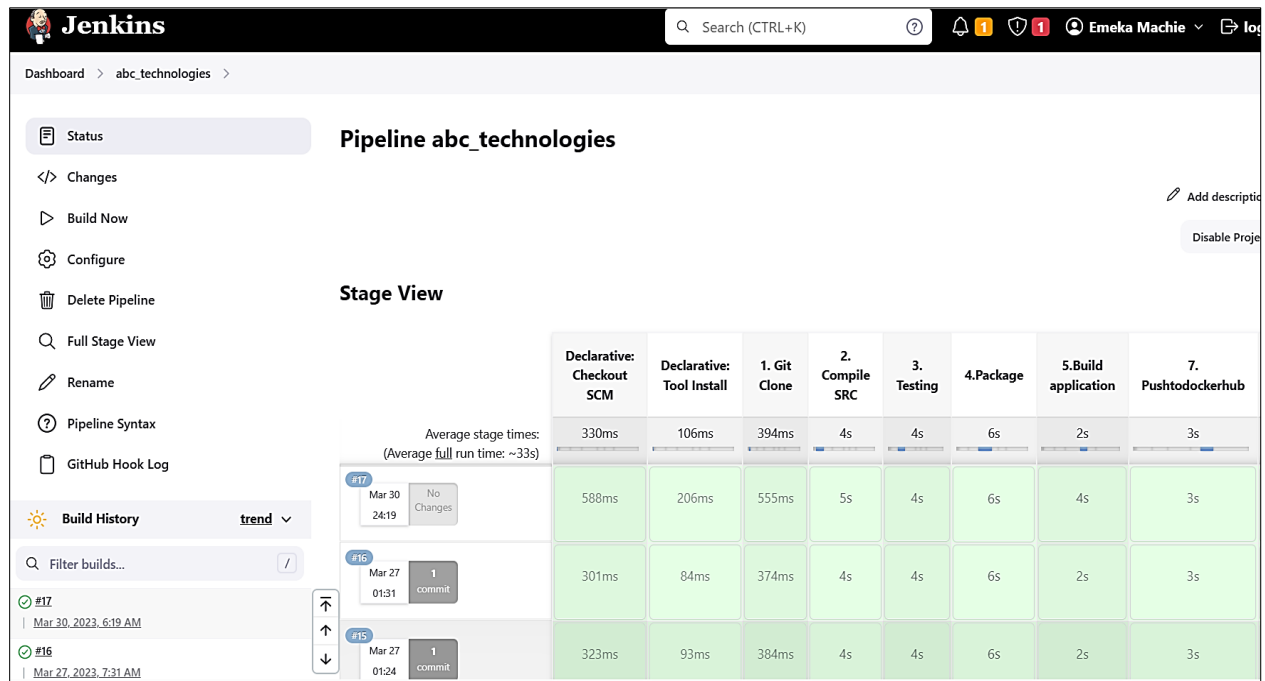
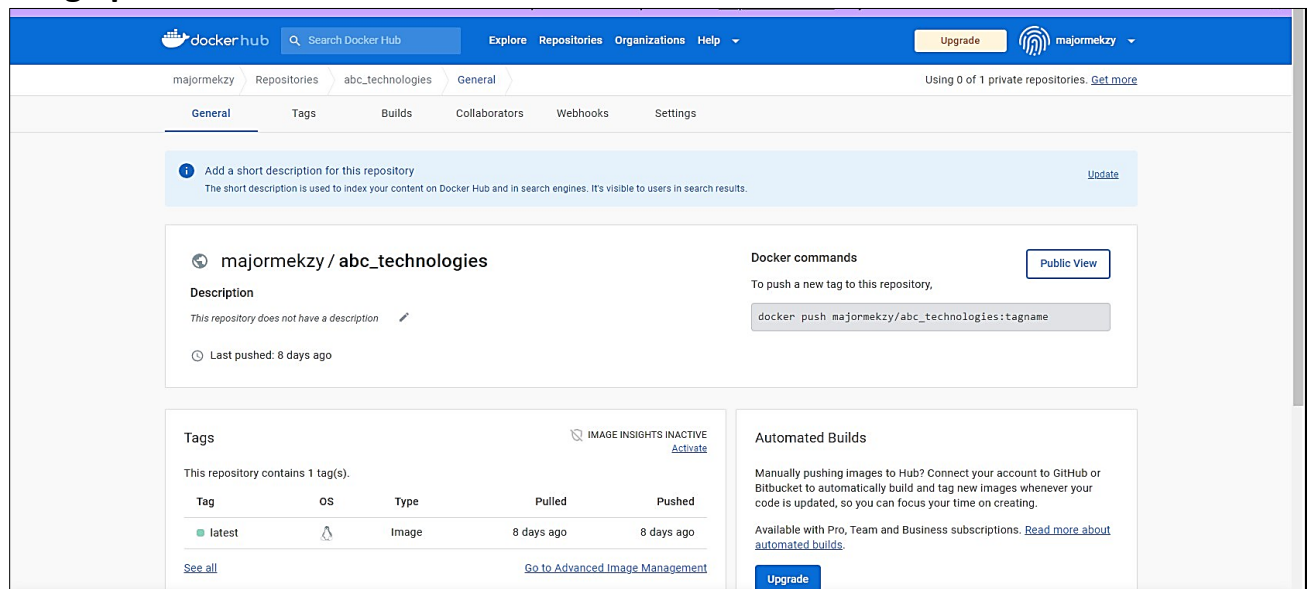


Image pushed to dockerhub



Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

Console output for the CI/CD pipeline

Dashboard > abc_technologies > #1 > Git > Console Output

Up

Status

Console Output

✓ Console Output

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/abc_technologies/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Majormekzy/abc_technologies.git # timeout=10
Fetching upstream changes from https://github.com/Majormekzy/abc_technologies.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Majormekzy/abc_technologies.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 1f860999d7a7ae529703b4b23f198a0a733ba958 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 1f860999d7a7ae529703b4b23f198a0a733ba958 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git checkout -b main 1f860999d7a7ae529703b4b23f198a0a733ba958 # timeout=10
Commit message: "Update Jenkinsfile"

Dashboard > abc_technologies > #1 > Shell Script > Console Output

Up

Status

Console Output

✓ Console Output

+ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.abc:ABCtechnologies >-----
[INFO] Building RetailModule 1.0
[INFO] -----[war]-----
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.6:prepare-agent (jacoco-initialize) @ ABCtechnologies ---
[INFO] argLine set to -javaagent:/var/lib/jenkins/.m2/repository/org/jacoco/org.jacoco.agent/0.8.6/org.jacoco.agent-0.8.6-runtime.jar-destfile=/var/lib/jenkins/workspace/abc_technologies/target/jacoco.exec
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ ABCtechnologies ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/abc_technologies/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ ABCtechnologies ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to /var/lib/jenkins/workspace/abc_technologies/target/classes
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.287 s
[INFO] Finished at: 2023-03-19T09:56:32Z
[INFO] -----

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

```
Dashboard > abc_technologies > #1 > Shell Script > Console Output

Up
Status
Console Output

+ docker build -t majormekzy/abc_technologies .
#1 [internal] load .dockerignore
#1 transferring context: 2B done
#1 DONE 0.0s

#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 437B done
#2 DONE 0.0s

#3 [auth] library/tomcat:pull token for registry-1.docker.io
#3 DONE 0.0s

#4 [internal] load metadata for docker.io/library/tomcat:9.0.72-jdk11-corretto-al2
#4 DONE 0.2s

#5 [1/2] FROM docker.io/library/tomcat:9.0.72-jdk11-corretto-al2@sha256:5279b08089e349d6bc63c1e9fbf3de4c6edf71f3c2a67b030aa79c0b3834451b
#5 DONE 0.0s

#6 [internal] load build context
#6 transferring context: 7.13MB 0.1s done
#6 DONE 0.1s

#5 [1/2] FROM docker.io/library/tomcat:9.0.72-jdk11-corretto-al2@sha256:5279b08089e349d6bc63c1e9fbf3de4c6edf71f3c2a67b030aa79c0b3834451b
#5 CACHED

#7 [2/2] COPY **/*.war /usr/local/tomcat/webapps/
#7 DONE 0.2s

#8 exporting to image
#8 exporting layers 0.1s done
#8 writing image sha256:2bcd0eeea103d2fb9aeb8002454df747068e7fe882afe4433a8b0ee5245d23f done
#8 naming to docker.io/majormekzy/abc_technologies done
#8 DONE 0.1s
```

```
Dashboard > abc_technologies > #5 > Shell Script > Console Output

Up
Status
Console Output

+ docker push majormekzy/abc_technologies:latest
The push refers to repository [docker.io/majormekzy/abc_technologies]
b8eca83860e5: Preparing
89cd10063f49: Preparing
61b10eeeb0a6: Preparing
7d8ea14a9eb4: Preparing
d67c000345c6: Preparing
941bb0a5987: Preparing
941bb0a5987: Waiting
89cd10063f49: Layer already exists
7d8ea14a9eb4: Layer already exists
61b10eeeb0a6: Layer already exists
d67c000345c6: Layer already exists
941bb0a5987: Layer already exists
b8eca83860e5: Pushed
latest: digest: sha256:84eeab327e4d50f25674bc73ce8b5fd26001f3d8fa9f4892cf5e21ccadb94546 size: 1579
```

```
Dashboard > abc_technologies > #1 > Shell Script > Console Output

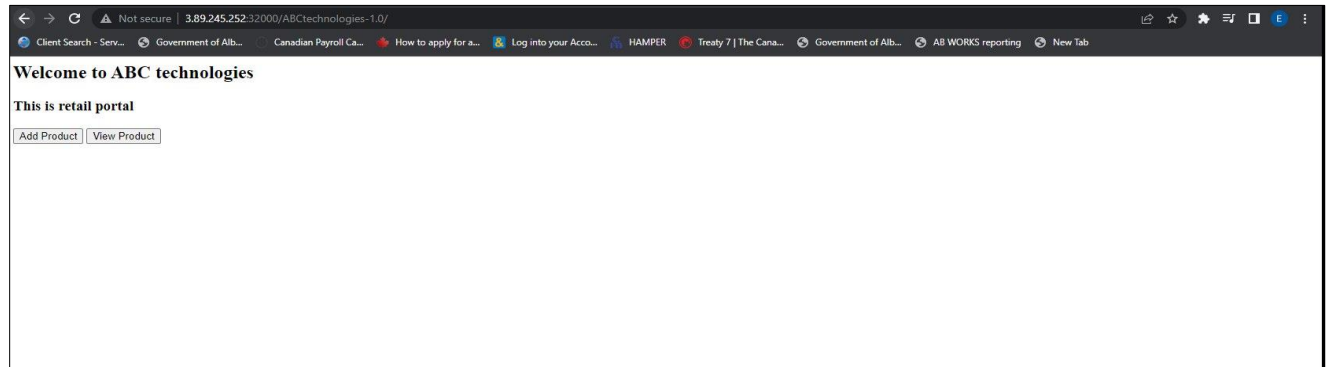
Up
Status
Console Output

+ docker run --name abc-tech-app -d -p 2000:8080 majormekzy/abc_technologies
9fecc707b6c28f99e873080b061982571ba54df34981d26f33e02da727ed8ce
```

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

Checking application



Steps in the configuration of server A and server B to enable ansible deploy the application into the Kubernetes cluster:

Step 1: I Integrated Ansible with Docker by installing Docker SDK on Server A. This will enable interaction with Docker API using Python. I executed the following commands:

- `sudo apt-get install python3-pip`
- `pip install docker`

To verify the installation:

- `python -c "import docker; print(docker.from_env().version())"`

Step 2: I integrated Ansible with Kubernetes (Server B) so that Ansible can deploy the application into the K8S cluster.

From the Ansible controller, I configured Ansible to use the Kubernetes API by setting the following environment variables:

- `export K8S_AUTH_API_KEY=<API key>`
- `export K8S_AUTH_HOST=<Kubernetes API server hostname>`
- `export K8S_AUTH_VERIFY_SSL=false`

To obtain the API key and hostname from the K8S control master node, use the following commands:

API key:

- `kubectl get secrets -n kube-system`

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

```

root@ip-172-31-22-199 ec2-user]# kubectl get secrets -n kube-system
NAME                                TYPE                                DATA  AGE
attachdetach-controller-token-5mct2  kubernetes.io/service-account-token 3      31d
bootstrap-signer-token-vt2ll         kubernetes.io/service-account-token 3      31d
calico-kube-controllers-token-npzfm   kubernetes.io/service-account-token 3      31d
calico-node-token-t8c5w              kubernetes.io/service-account-token 3      31d
certificate-controller-token-xm2ft    kubernetes.io/service-account-token 3      31d
clusterrole-aggregation-controller-token-lb7q9 kubernetes.io/service-account-token 3      31d
coredns-token-szgnz                  kubernetes.io/service-account-token 3      31d
cronjob-controller-token-gdw6m        kubernetes.io/service-account-token 3      31d
daemon-set-controller-token-dvzbq     kubernetes.io/service-account-token 3      31d
default-token-n92cp                  kubernetes.io/service-account-token 3      31d
deployment-controller-token-zgzbl     kubernetes.io/service-account-token 3      31d
disruption-controller-token-pmrlz     kubernetes.io/service-account-token 3      31d
endpoint-controller-token-p2m84       kubernetes.io/service-account-token 3      31d
endpointlice-controller-token-db2w1    kubernetes.io/service-account-token 3      31d
endpointlicesimirrorng-controller-token-jpm6n kubernetes.io/service-account-token 3      31d
ephemeral-volume-controller-token-q29wb kubernetes.io/service-account-token 3      31d
expand-controller-token-7xt8m         kubernetes.io/service-account-token 3      31d
garbage-collector-token-srdjd         kubernetes.io/service-account-token 3      31d
horizontal-pod-autoscaler-token-sl8ql  kubernetes.io/service-account-token 3      31d
job-controller-token-brgw5            kubernetes.io/service-account-token 3      31d
kube-proxy-token-r9zhz                kubernetes.io/service-account-token 3      31d
kubernetes-dashboard-token-574pd       kubernetes.io/service-account-token 3      31d
node-controller-token-bmmgz           kubernetes.io/service-account-token 3      31d
persistent-volume-binder-token-gqzlv  kubernetes.io/service-account-token 3      31d
pod-garbage-collector-token-27hb4     kubernetes.io/service-account-token 3      31d
pod-protection-controller-token-qmh85  kubernetes.io/service-account-token 3      31d
pvc-protection-controller-token-5h9xc  kubernetes.io/service-account-token 3      31d
replicaset-controller-token-t5t2d     kubernetes.io/service-account-token 3      31d
replicaset-controller-token-mmft9     kubernetes.io/service-account-token 3      31d
resourcequota-controller-token-trr7l   kubernetes.io/service-account-token 3      31d
root-ca-cert-publisher-token-zb9kr     kubernetes.io/service-account-token 3      31d
service-account-controller-token-zb8qh  kubernetes.io/service-account-token 3      31d
service-controller-token-tvm56         kubernetes.io/service-account-token 3      31d
statefulset-controller-token-rc52p     kubernetes.io/service-account-token 3      31d
token-cleaner-token-cygqcs            kubernetes.io/service-account-token 3      31d
ttl-after-finished-controller-token-khwkz kubernetes.io/service-account-token 3      31d
ttl-controller-token-xd4fb             kubernetes.io/service-account-token 3      31d

```

- `kubectl get secret service-account-controller-token-zb8qh -n kube-system -o jsonpath='{.data.token}' | base64 -d`

[illegible]

Hostname:

- `kubectrl config view --minify --flatten -o jsonpath='{.clusters[].cluster.server}'`

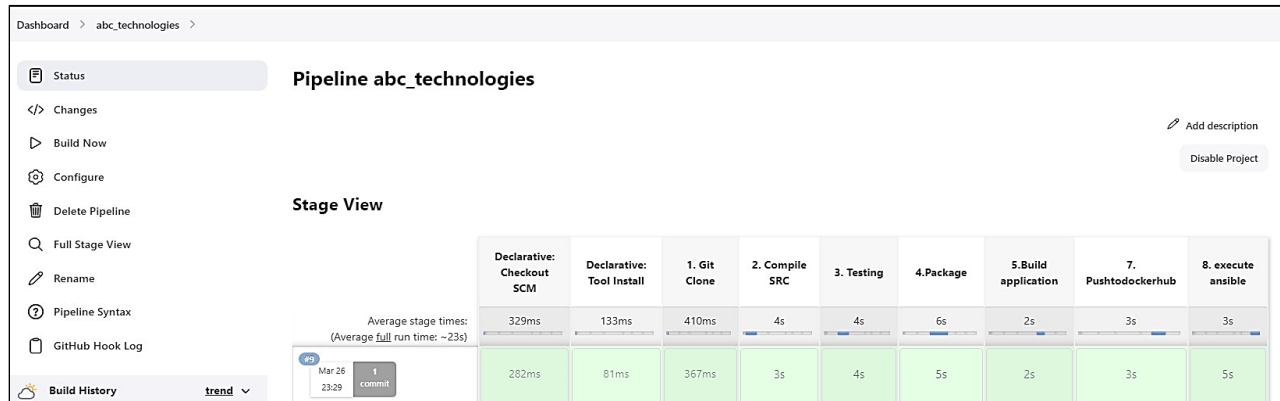
```
[root@ip-172-31-22-199 ec2-user]# kubectl config view --minify --output 'jsonpath={.clusters[0].cluster.server}'
https://172.31.22.199:6443/
[root@ip-172-31-22-199 ec2-user]#
```

Step 3: I copied the deployment file to the Kubernetes node by adding a task to the Ansible playbook to copy the deployment file to the Kubernetes node using the "copy" module and then deployed the deployment file in Kubernetes cluster by adding a task to the Ansible playbook to deploy the deployment file in Kubernetes using the "k8s" module. (See deployment file in github)

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

After updating my github repository with my ansible playbook and updated my Jenkinsfile, my Jenkins pipeline was triggered to run the process.



K8S cluster:

```
EC2
[root@ip-172-31-22-199 ec2-user]# kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
ip-172-31-22-199.ec2.internal       Ready    control-plane,master   31d   v1.21.3
ip-172-31-4-57.ec2.internal         Ready    <none>    31d   v1.21.3
[root@ip-172-31-22-199 ec2-user]# kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
abc-tech-app-cb6584d44-67cdk        1/1     Running   0           4m20s
abc-tech-app-cb6584d44-zc7fm        1/1     Running   0           4m21s
```

Step 4: I used the following steps below to install Prometheus and Grafana on my K8s Master node to monitor my Kubernetes cluster (see manifest files on github)

1. Update the package manager on your Amazon Linux instance using the command `sudo yum update`.
2. Install the Prometheus package by running the command `sudo yum install prometheus`.
3. After the installation completes, you can start the Prometheus service using the command `sudo systemctl start prometheus`.
4. To enable Prometheus to start automatically when the server boots up, run the command `sudo systemctl enable prometheus`.
5. Verify that Prometheus is running by accessing its web interface at `http://<your-server-ip>:9090` using a web browser.
6. Now, you can install Grafana by adding the Grafana repository to the package manager using the command `sudo rpm -Uvh https://dl.grafana.com/oss/release/grafana-7.5.10-1.x86_64.rpm`.
7. After adding the repository, install Grafana using the command `sudo yum install grafana`.
8. Start the Grafana service by running the command `sudo systemctl start grafana-server`.
9. To enable Grafana to start automatically when the server boots up, run the command `sudo systemctl enable grafana-server`.
10. Verify that Grafana is running by accessing its web interface at `http://<your-server-ip>:3000` using a web browser.

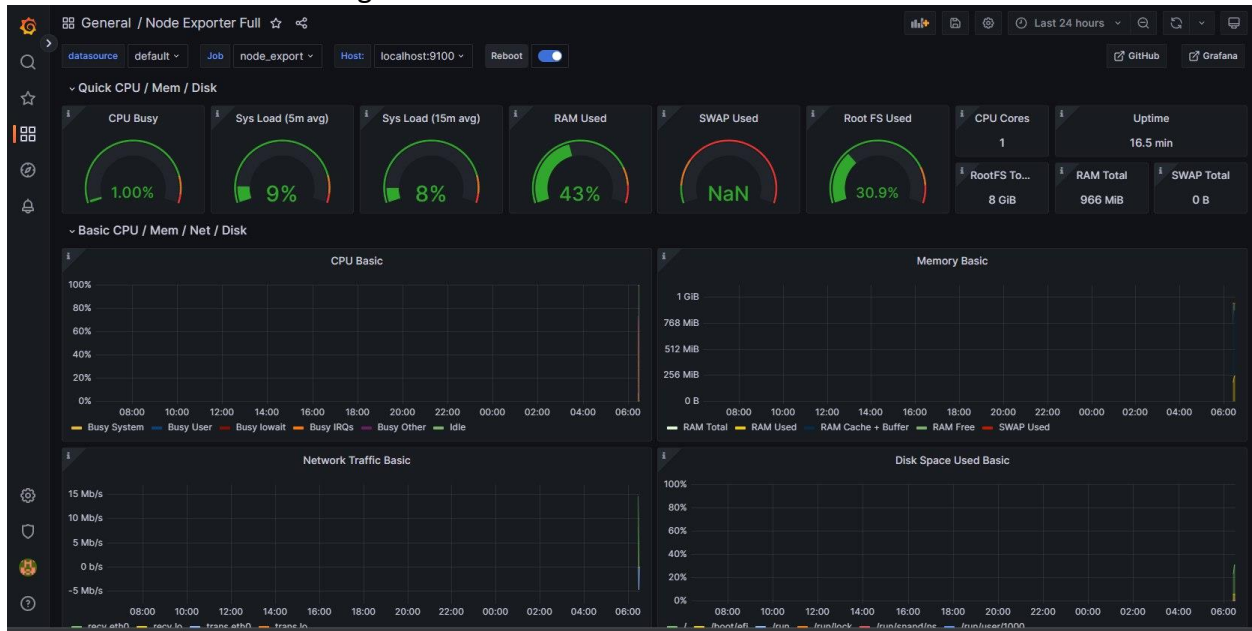
Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

11. Configure Grafana to use Prometheus as a data source by adding a new data source in the Grafana web interface and specifying the URL for Prometheus, which should be <http://localhost:9090>.

12. Once the data source is configured, you can create dashboards in Grafana to visualize the data collected by Prometheus.

Grafana dashboard showing K8s cluster metrics



THE END