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.

Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

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

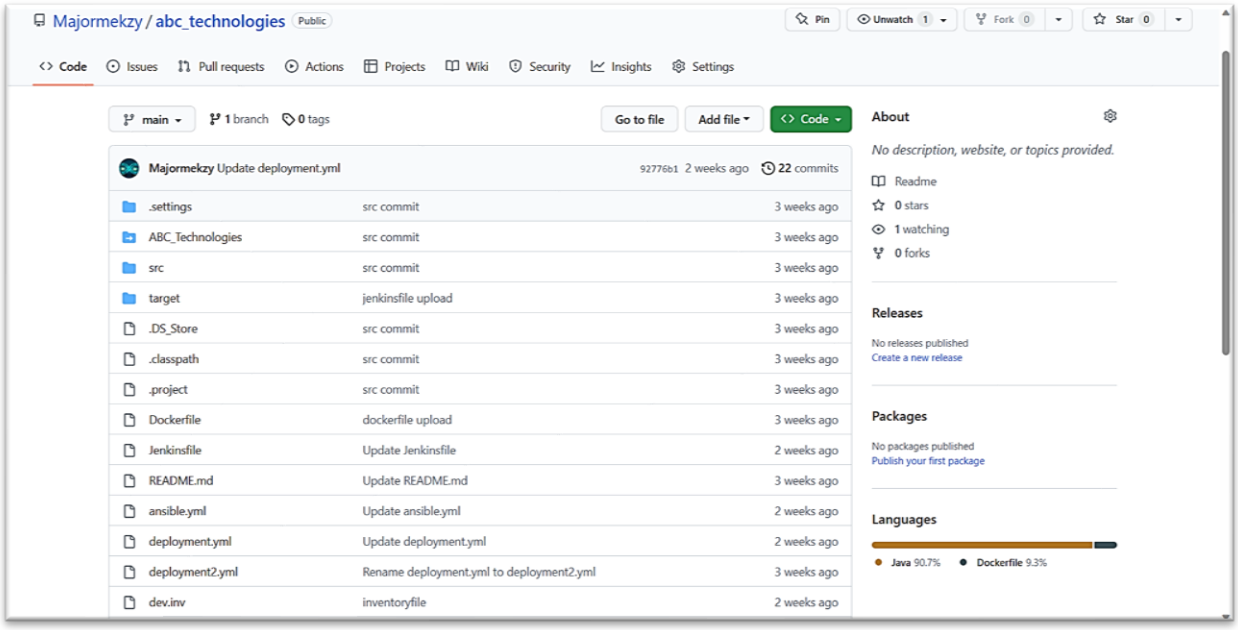
Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

Post Graduate Certification Program in DevOps

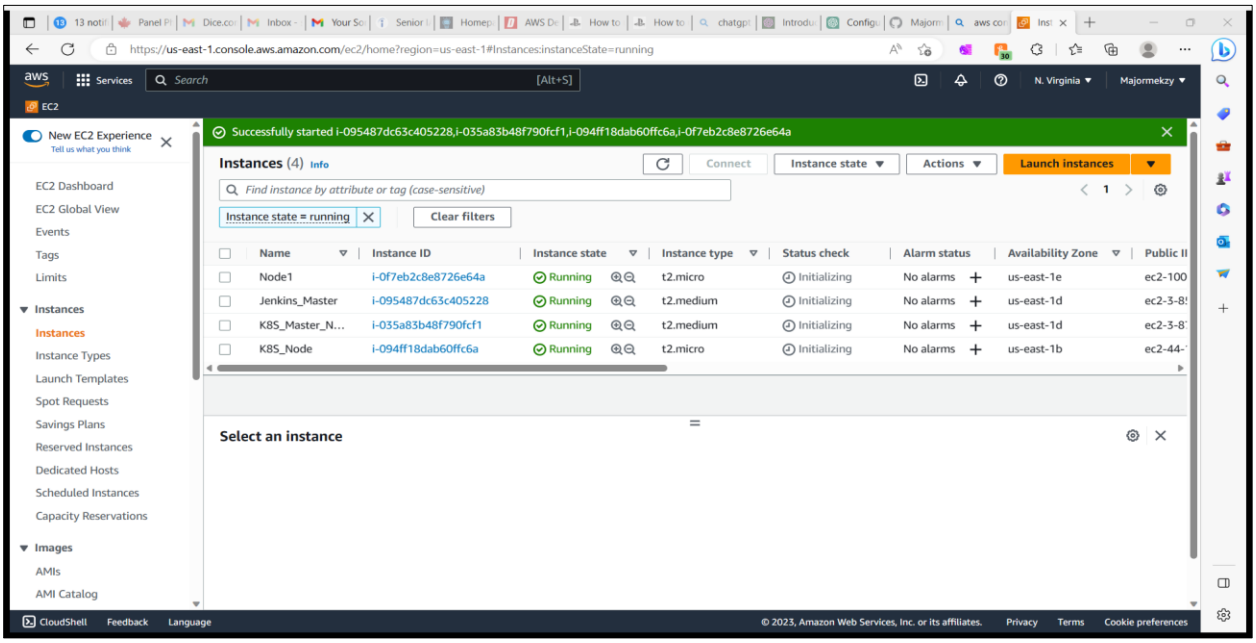
ABC-Technologies Edureka Project 1

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

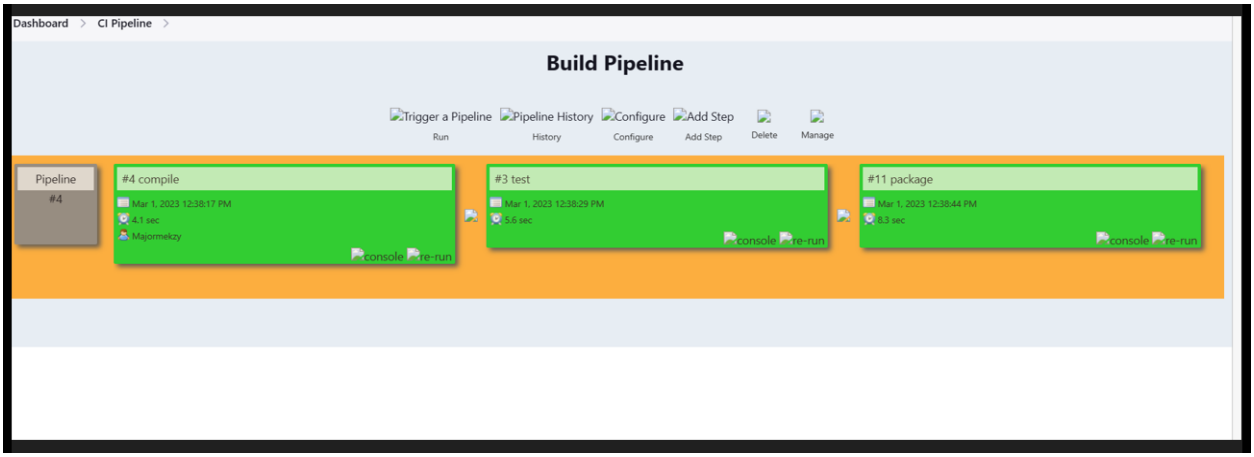


Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

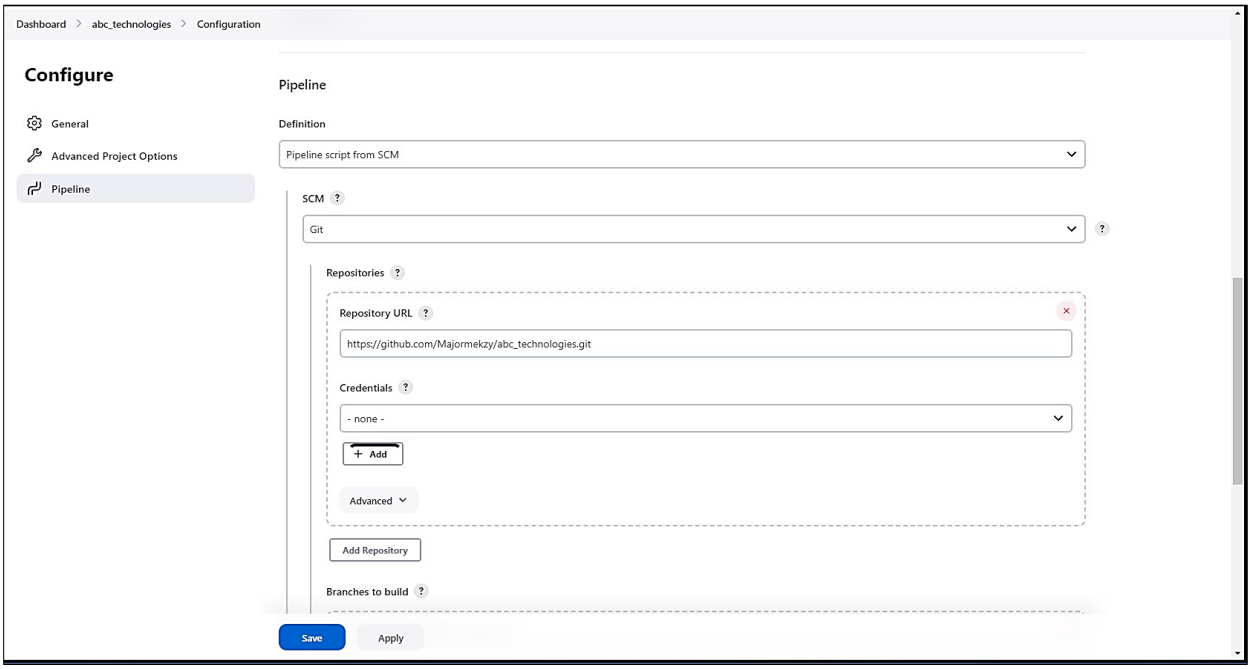
Post Graduate Certification Program in DevOps

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.



Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

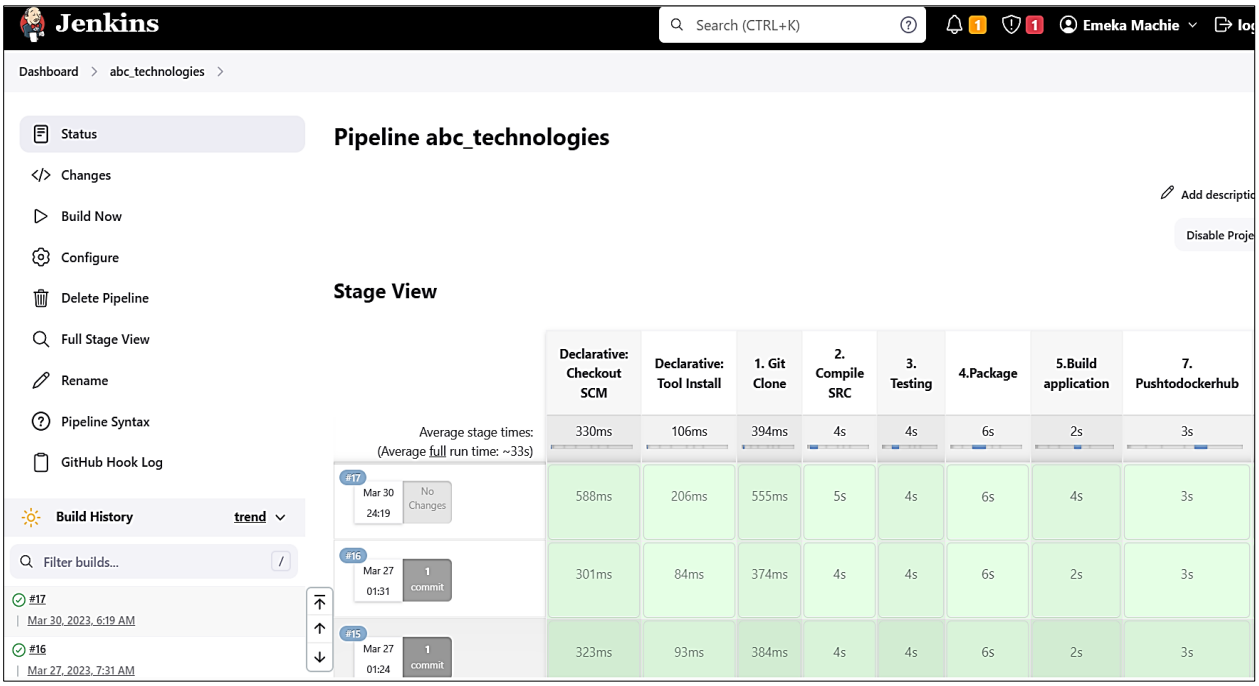
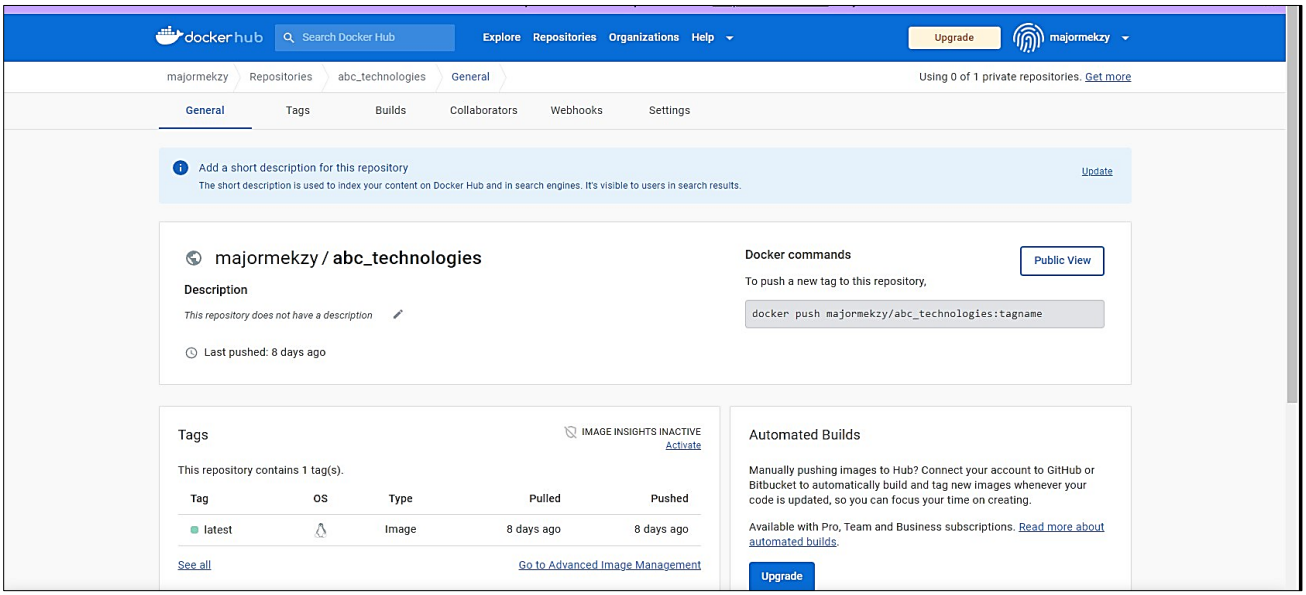


Image pushed to dockerhub

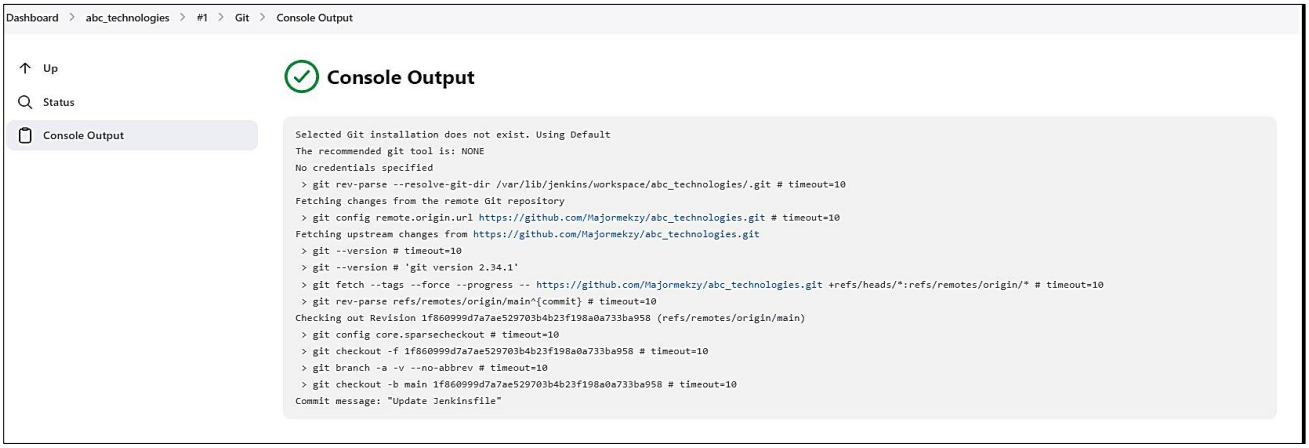


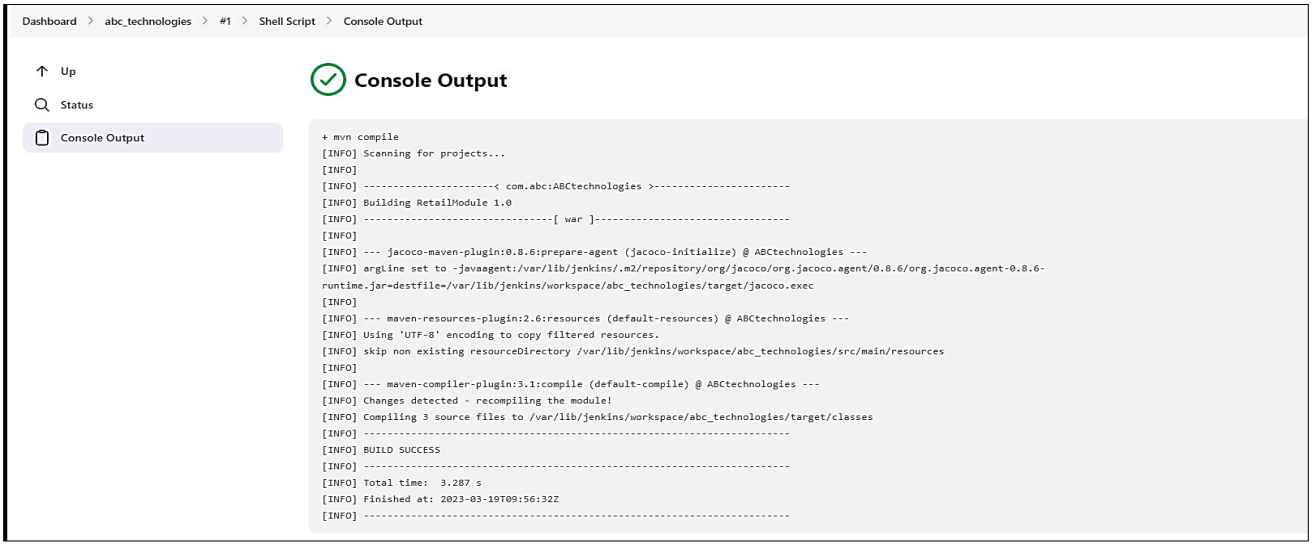
Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

Console output for the CI/CD pipeline

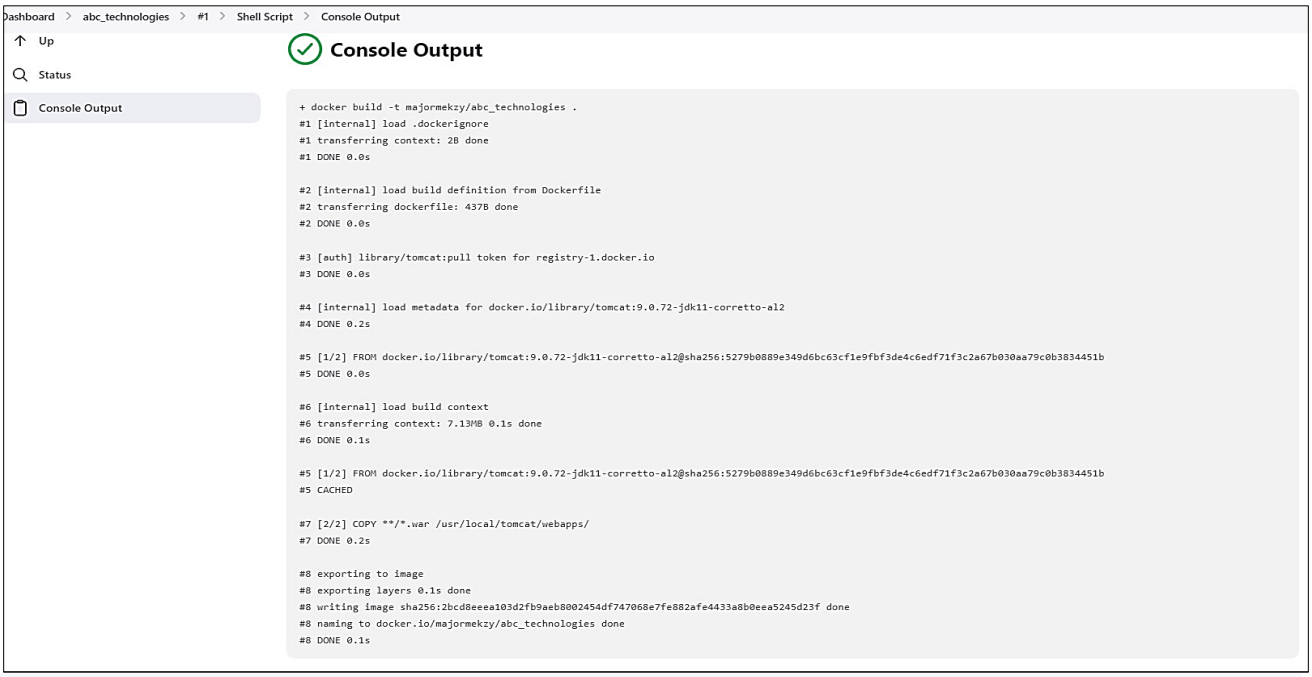




Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

Post Graduate Certification Program in DevOps

ABC-Technologies Edureka Project 1

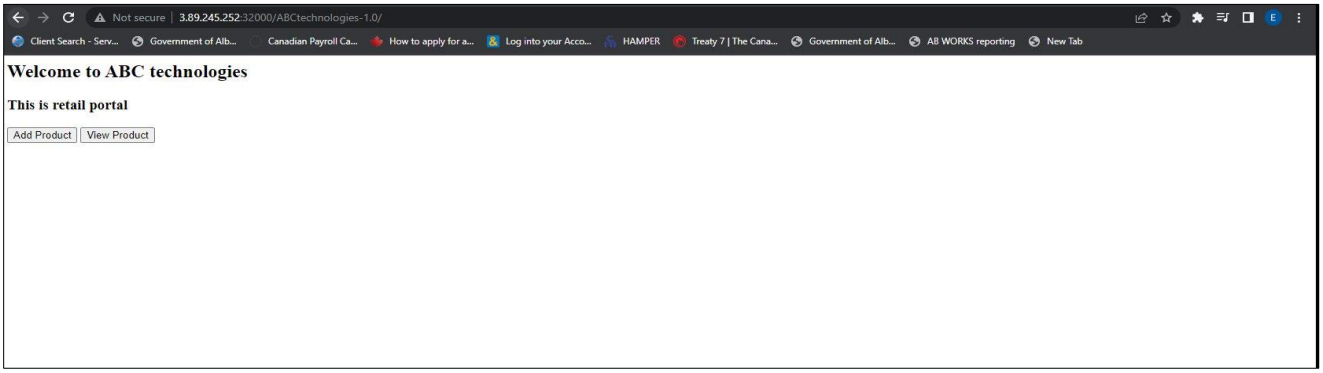


Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

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:

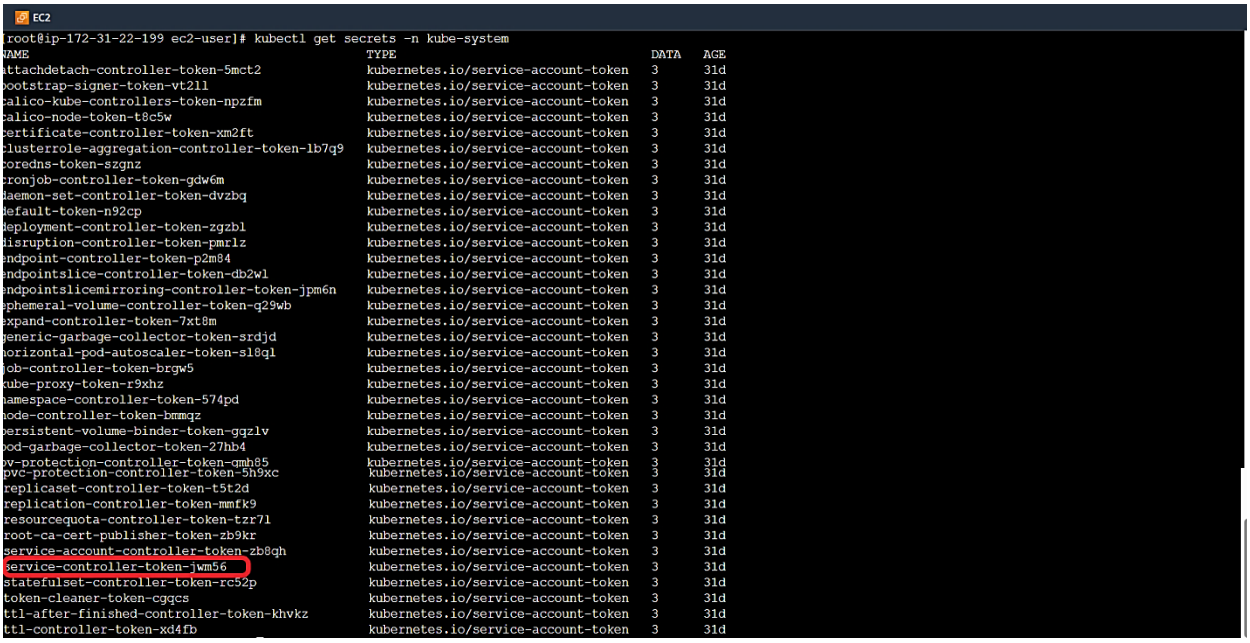
API key:

- kubectl get secrets -n kube-system

Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271

Post Graduate Certification Program in DevOps

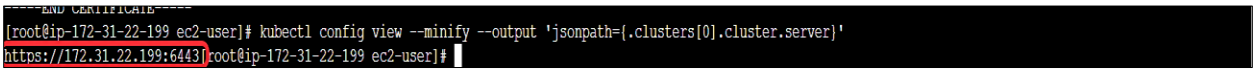
ABC-Technologies Edureka Project 1



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

Hostname:

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

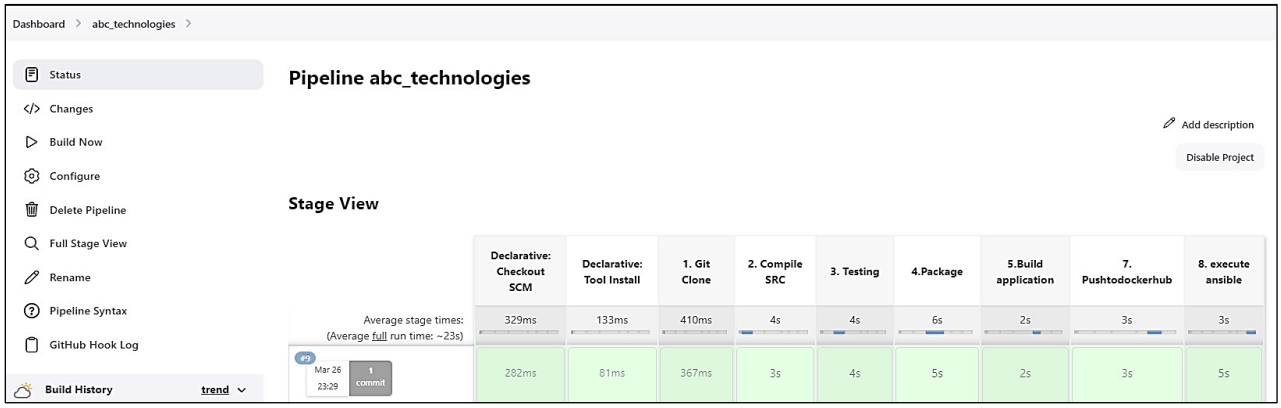


Step 3: I copied the deployment file to the Kubernetes node by adding a task to the Ansible playbook to copy the deployment file to the Kubernetes node using the "copy" module and then deployed the deployment file in Kubernetes cluster by adding a task to the Ansible playbook to deploy the deployment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

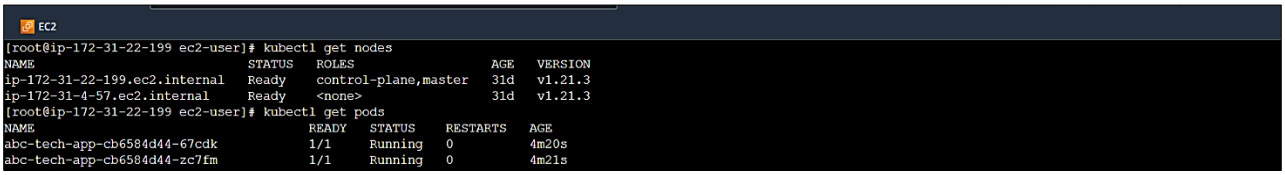
Post Graduate Certification Program in DevOps

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:



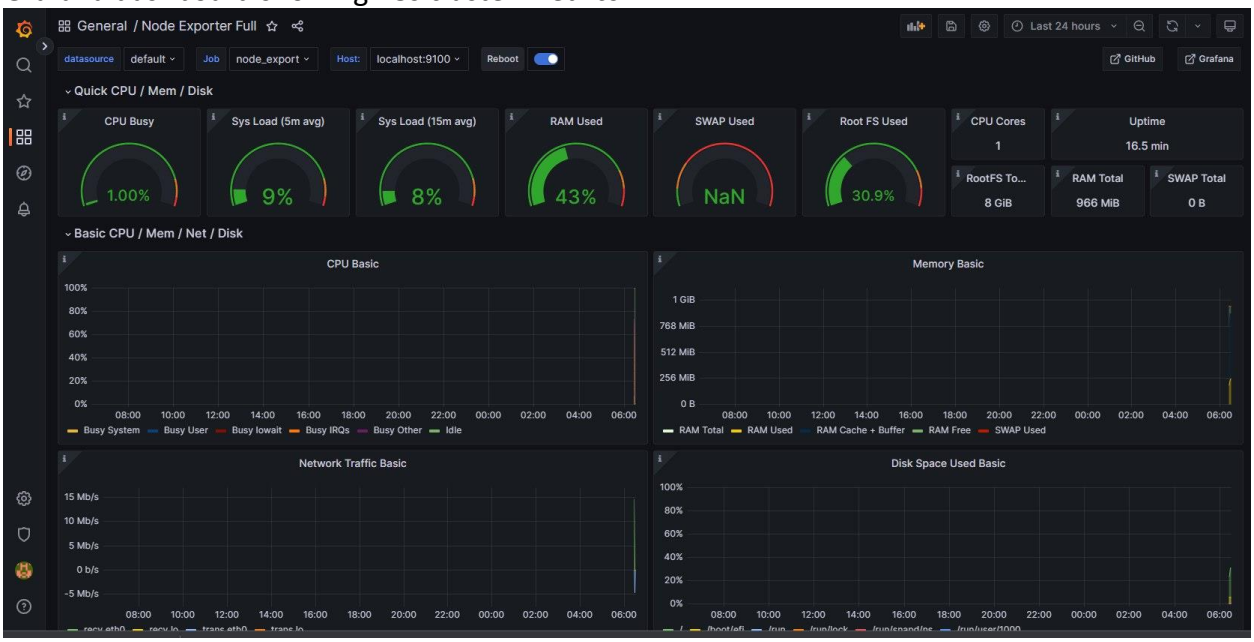
Step 4: I used the following steps below to install Prometheus and Grafana on my K8s Master node to monitor my Kubernetes cluster (see manifest files on github)

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

Name: Emeka Machie Email:emmyjones4u@gmail.com Contact: +1 (587) 839 8271