Harsh Awasthi

harsh123awasthi@gmail.com

Abstract

The document outlines the steps for a Capstone Project focused on Application Deployment using AWS, Docker, Git and Jenkins.

CAPSTONE PROJECT

A REACT APPLICATION DEPLOYEMENT

The process begins with creating a t2.micro EC2 instance on AWS and installing nginx web server. The next step involves cloning a Github repository provided in document.

Repo URL: <https://github.com/sriram-R-krishnan/devops-build>

Deploying the application on port 80

***Steps:***

* sudo apt-get update
* sudo apt install nginx -y
* sudo systemctl start nginx
* sudo systemctl enable nginx
* sudo systemctl status nginx
* Create a new configuration file in the /etc/nginx/sites-available/ directory. This file will contain the server block settings
* Create and Insert the following configuration in configuration file

server {

listen 80;

server\_name <your public\_ip>;

root /var/www/html;

index index.html;

location / {

try\_files $uri $uri/ =404;

}

}

* Activate the server block by linking the file you just created to the sites-enabled directory
* sudo ln -s /etc/nginx/sites-available/mysites /etc/nginx/sites-enabled/
* Test NGINX Configuration – sudo nginx -t
* Restart NGINX – sudo systemctl restart nginx
* Access Your Website

**CONCLUSION**

Successfully set up a virtual host in NGINX. This configuration allows to host multiple websites on a single server, optimizing resources and management. By following these steps, we can expand your server's capabilities and host various websites efficiently and effectively.

We will install following tools necessary for the deployment of the app using documentations provided in links

1. Docker:- <https://docs.docker.com/engine/install/ubuntu/>
2. Docker Compose:- <https://docs.docker.com/compose/install/linux/>
3. Java:- sudo apt install openjdk-17-jdk -y
4. Jenkins:- <https://www.jenkins.io/doc/book/installing/linux/>

We are making a public git Repository with master branch and add one more branch ‘dev’ (using CLI). We will use this ‘dev’ branch to push the codes.

Similarly, we are going to create two Repositories in docker hub – ‘dev’ (public) and ‘prod’ (private) as per instructions.

We will create 6 files using vi editor:

1. **Dockerfile** - A Dockerfile is a text file containing instructions for building your source code
2. **Docker-compose.yml** – A Docker-compose is a yaml file that helps define services and run multi-container applications
3. **build.sh** – to build docker images
4. **deploy.sh** – to deploy docker images to your server
5. **.gitignore** –we are going to use this file to avoid committing artifacts to source control.
6. **.dockerignore** – Docker ignore contains files which you want Docker build to ignore

**Jenkins:**

Since we have already installed Jenkins we will configure build steps as per our need to build and deploy the application.

Let us start by creating password to access the Jenkins

Open your web browser and navigate to http://<your-ec2-public-ip>:8080

for this you should enable port 8080 in your instance security group.

Retrieve the initial Jenkins admin password:

* sudo cat /var/lib/jenkins/secrets/initialAdminPassword
* Install the neccesary plugins and signup inside Jenkins

For our requirement we will create a Multi-Branch Pipeline project

First we shall add our Dockerhub and to our Jenkins for that we are going to follow below steps:

* We will use personal Access Token from Dockerhub -> Security -> Access tokens -> New Access token -> Copy token
* Next in manage Jenkins -> Security -> Credentials -> Global credentials -> Add credentials -> username(docker username) & password (Personal Access token from Dockerhub) -> ID (copy this Id)
* Open manage Jenkins -> System configuration -> System -> Environment variables -> add -> use Copied ID from above

**Multi-Branch Pipelines**

A multi-branch pipeline is a concept of automatically creating Jenkins pipelines based on Git branches.

It can automatically discover new branches in the source control (Github) and **automatically create a pipeline for that branch**. When the pipeline build starts, Jenkins uses the **Jenkinsfile** in that branch for build.

Multi-branch pipeline supports **PR based branch discovery**. Meaning, branches get discovered automatically in the pipeline if someone raises a PR (pull request) from a branch or a forked repository. So if you are looking for a Pull Request based Jenkins build workflow, this is a great option.

**Steps:**

1. From the Jenkins home page create a “new item”.
2. Select the “Multibranch pipeline” from the option and click ok.
3. Click “Add a Source” and select Github.
4. Under the credentials field, select Jenkins, and create a credential with your Github username and password.
5. Select the created credentials and provide your Github repo to validate the credential.
6. Under “**Behaviours**” select the required option matches your requirement.
7. If you choose to have a different name for Jenkinsfile, you can specify it in the build configuration
8. In the “Script Path” option, you can provide the required name. Ensure the Jenkinsfile is present in the repo with the same name you provide in the pipeline configuration.
9. Save all the job configurations and Jenkins scans the configured Github repo for all the branches and PR requests based on our configurations.

## Configure Webhook for Multibranch Pipeline

## Head over to the Github repo and click on the settings.

## Select the webhook option at the left and click “Add Webhook” button.

## Add your Jenkins URL followed by “multibranch-webhook-trigger/invoke?token=mytoken” under payload URL. Select the content type as “application/json” and click “Add Webhook”

## Once our Jenkins is configured and ready we will push our codes to Github ‘dev’ branch.

**Version Control Push Code to Github**:

**Steps:**

* sudo apt-get update
* sudo apt-get install git
* git config --global user.name <Your username>
* git config --global user.email <your.email@example.com>
* Copy the repository URL cd to project path and use git init
* Create Branch :- git checkout -b dev
* git add .
* git status
* git commit -m "Your commit message"
* git branch –M dev
* git push -u origin dev

Once our code get pushed to Github automatic build will trigger as we have set webhook. Jenkins will build image and push the image to required Dockerhub repo.

For multibranch pipeline we require Jenkinsfile in each branch we wish to build, therefore I pushed complete code to dev branch and only Jenkinsfile to master branch.

To monitor the application we will use Prometheus.

**Prometheus**

We will install and set up Prometheus as per documentation provided by below link

<https://www.cherryservers.com/blog/install-prometheus-ubuntu>

To access Prometheus web interface we require to open port 9090 in our machine

With Prometheus running successfully, we can access it via your web browser using localhost:9090 or <ip\_address>:9090

**Deploying the application**

We can deploy the application by various methods

* Using Docker build and Run command
  + “docker build –t <image\_name> .”
  + “docker run –d -–name <container\_name> -p ports:ports <image\_name>”
* Using Docker compose command
  + “docker-compose up -d”
* Using ssh agent in Jenkins file
* Using Shell scripts build and deploy
* Accessing the nginx web server web page as we have already configured the files.

Links:

website url - <http://13.201.15.39:300/>

Github Link - https://github.com/HARSAWASTHI/Finalcapstoneproject.git