JENKINS – DEVOPS by Rohan Sen

PROJECT 1 : 24-08-2023 Declarative Pipeline

**Jenkins** is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. In simple it is tool for CICD, we can create CICD pipelines using it.

1. Jenkins basics.
2. How to create simple pipline.
3. How to create Declarative pipeline.

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. In simple it is tool for CICD, we can create CICD pipelines using it.

**CICD**

build -> test -> deployment

Continuous: No break and rapid development.

Continuous integration (CI): To integrate code and it should build and test.

Continuos delivery/deployment: automated deployment, delpoyment is when there is change. Delivery is manual when we want.

**Continuous integration Continuos deployment (CI/CD) pipeline:**

**This is how it works:**

* Get the code from Github
* build the code
* test the code
* Create image out of it
* maintain versioning
* push to dockerhub registry
* Deploy on AWS/server.

You can do this using jenkins, and it’s called jenkins pipeline.

**Installation:**

* Create aws instance
* Install dependencies

sudo apt update && sudo apt upgrade -y \\

&& sudo apt install docker.io docker-compose fontconfig openjdk-17-jre -y

Add the keys & repo

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

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

* Install

sudo apt-get update

sudo apt install jenkins -y

* Add jenkins & current user to docker group

sudo usermod -aG docker jenkins

sudo usermod -aG docker $USER

newgrp docker # reloads the group changes

* Get the initial password

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Now, installation is done visit your public/local IP and complete setup.

* Creating simple pipeline (jobs)

A Jenkins job is a sequential set of tasks that a user defines. It represents a specific unit of work, such as building a software project, running tests, deploying applications, or performing any other automated task.

**First, click on create job.**

Now, here we have multiple options we can create freestyle, pipeline or any other type of project. we will see freestyle first.

You can write description and select github project if you are using it.

Select git in source code management option, and enter the repo url also credentials if it is private repo. (select branch also)

Next, set build triggers, you can do remotely, periodically, with github wehooks or any other way.

If you want your jenkins to start build whenever there is a push/commit in repo, then you can check GitHub hook trigger for GITScm polling.

Now you need to setup github webhook, visit github repo.

Go to -> settings -> webhooks -> add webhook

Enter jenkins url and append github-webhook/, it should be accessible publicly.

http://89.304.53.2:8080/github-webhook/

Set the event, save and wait for github’s ping test.

Triggers: it is some action, when it is performed the pipeline will be started. build periodically: Schedule using cron.

Now configure build step. and save.

echo "Hello world", # your build code here.

docker build . -t myapp

docker run -d -t -p 8000:8000 myapp

And now you pipeline is ready, Now whenever you will commit or push to that repo jenkins will start the build process and all steps you want.

**Declarative pipeline**

Declarative pipeline can be explained like ‘pipeline as a code’, we can write script to build, test, push, and deploy the project; we can make the use of Environment variables and other features. Groovy language is used to write jenkins declarative pipeline.

**This is a skeleton or syntax of declarative pipeline’s groovy script:**

**GROOVY**

**Skeleton of Jenkins Declarative pipeline script**

pipeline {

agent any

stages {

stage("Code"){

steps{

}

}

stage("Build & test"){

steps{

}

}

stage("Push to repo"){

steps{

}

}

stage("Deploy"){

steps{

}

}

}

}

**Stage:** The stage/part where a particular task will be performed, like build or deployment, etc.

steps: The commands of task to perform for that stage, like git clone, docker build, etc

agent: Agent specifiy where the entire pipeline will be execute, or a particular stage, like in any , or in docker, etc.

**Steps**

Click on new item, select pipeline as project type and enter your project name.

Set the github url if it is on it, and configure other option accordingly

Set the trigger to GitHub hook trigger for GITScm polling If you want to use github webhook and automatic pipeline triggering.

Now, here the main part is pipeline. you can check some script sample by clicking on try scripts.

First write the skeleton.

Now, enter the commands of code cloning, build, push, deploy, etc.

GROOVY

Sample Declarative pipeline script

pipeline {

agent any

stages {

stage("Code"){

steps{

git url: "https://github.com/Dark-Kernel/node-todo-cicd.git", branch: "master"

echo "Cloned code successfully."

}

}

stage("Build & test"){

steps{

sh "docker build . -t node-todo-cicd"

echo "Docker build done"

}

}

stage("Push to repo"){

steps{

withCredentials([usernamePassword(credentialsId:"dockerhubid", passwordVariable:"dockerhubidPass", usernameVariable:"dockerhubidUser")]){

sh "docker login -u ${env.dockerhubidUser} -p ${env.dockerhubidPass}"

sh "docker tag node-todo-cicd ${env.dockerhubidUser}/node-todo-cicd:latest"

sh "docker push ${env.dockerhubidUser}/node-todo-cicd:latest"

}

echo "Pushed to docker hub registry"

}

}

stage("Deploy"){

steps{

sh "docker-compose up -d"

echo "Deployed to ec2"

}

}

}

}

It goes like:

* clone the code
* build & test using Dockerfile
* Login to registry -> tag image -> Push image to docker hub or any other registry

Finally, deploy (execute) it .

To login into registry we need username and password, so don’t enter your credentials directly in script, use Environment variables.

Go to -> Dashboard -> manage jenkins -> credentials -> system -> global credentials -> Add credentials.

Set type to accordingly, in this case it is username with password, set the scope.

Now, enter username and password, provide unique id, because we will use id to access our credentials.

credentials can used by id like for username: id+User password: id+Pass

Ex. id is myid then, username: myidUser, password: myidPass

To use it in file, enclose it with ${env.} Ex. ${env.myidUser}.

Now, save it and our pipeline is ready, you can setup github webhook by adding url as explained in above step.

Instead of writting script in jenkins, you can pass it with the source code it self.

You can create a Jenkinsfile and write your script inside it.

Select definition to Pipeline script from SCM, select SCM as Git

Enter the url of repo, select branch, give script path and save.

**Jenkins file:**

GROOVY

Jenkinsfile

pipeline {

agent any

stages {

stage("Code"){

steps{

git url: "https://github.com/Dark-Kernel/node-api.git", branch: "master"

}

}

stage("Build & test"){

steps{

sh "docker build . -t node-api"

}

}

stage("Push to repo"){

steps{

withCredentials([usernamePassword(credentialsId:"dockerhubid", passwordVariable:"dockerhubidPass", usernameVariable:"dockerhubidUser")]){

sh "docker login -u ${env.dockerhubidUser} -p ${env.dockerhubidPass}"

sh "docker tag node-api ${env.dockerhubidUser}/node-api:latest"

sh "docker push ${env.dockerhubidUser}/node-api:latest"

}

echo "Pushed to docker hub registry"

}

}

stage("Deploy"){

steps{

sh "docker-compose down && docker-compose up -d"

echo "Deployed to ec2"

}

}

}

}