# **Plugins:**

# Trigger builds remotely (e.g., from scripts) => Build Authorization Token Root Plugin

Build Pipeline Plugin

Deploy to container Plugin

Safe Restart

1. Building a simple pipeline
2. Difference between Continuous Delivery and Continuous Deployment
3. How to run parallel jobs

=======================

Jenkins Workshop Agenda

=======================

What is Build & Deployment Process

Application Environments In Real-Time

Challenges in Manual Build & Deployments

What is CI CD & Why do we need ?

Jenkins Introduction

Jenkins Setup

Jenkins Job Creation

Job Scheduling

Conclusion

========================

Build & Deployment Process

========================

Take latest source code from GitHub

Compile project source code

Execute Unit Test cases

Perform Code Review using SonarQube

Package the application ( jar / war )

Upload Build Artifact in Nexus

Deploy the Application in Server

=======================

Application Environments:

======================

Environment : A platform which is used to run our application

DEV env : Developers will use to perform code integration testing

QA env : Testers will use to perform functional testing

UAT env : User acceptance testing ( client side team will perform application testing )

Pilot env : It is also called as Pre-Prod environment ( performance testing )

Prod env : Live environment. End users will access applications running in the PROD environment.

www.gmail.com ----> Production env url of the project

www.facebook.com ---> Production env url of the project

================================

Challenges in Manual Build Process

================================

Every day we need to deploy latest code

Deploy code in multiple environments

Takes lot of time

Repeated Work

Error Prone

================

Project Teams

================

Development Team : Responsible for project development ( Coding )

Testing Team : Responsible for project functionality testing ( verification & validation )

Operations Team : Responsible for Build & Deployment process

Dev + Ops ====> DevOps

Development + Operations =============> DevOps

DevOps is a process which is used to collaborate development team work & operations team work

Using DevOps process we can simply application Build & Deployment process

===========

Jenkins

===========

-> Jenkins is a free software

-> Jenkins developed using java language

-> Jenkins is used to automate build & deployment process

-> We can implement CI CD using jenkins

==============

What is CI CD ?

===============

CI : Continuous Integration

CD : Continuous Delivery / Continuous Deployment

-> CI CD is one of the trending approach in software development life cycle

-> CI CD is used to simplify and automate project deployment & delivery process

Continuous Integration : When code changes happen it should be ready to test

Continuous Delivery : Keep it ready in repository for release

Continuous Deployment : Release / deploy the project to Production

Note: For Production deployment we need to take Client Approval.

===============

Part-1 : Summary

===============

1) What is Build & Deployment Process

2) Application Environments

3) Why we need several environments for our application

4) Challenges in Manual Build & Deployment process

5) What is Jenkins

6) What is CI CD

====================

Part-2

====================

================

Manual Deployment

================

GitHub Repo URL : https://github.com/ashokitschool/maven-web-app.git

1) Downloaded code from github

2) Executed Maven Goal ( clean package ) ==> war file created

3) Uploaded war file into tomcat server (deployment)

4) Access application from tomcat dashboard

===================

Infrastructure Setup

====================

-> Create Linux VM using EC2 in AWS cloud and install tomcat server

-> Create Linux VM using EC2 in AWS cloud and install Jenkins server

-> Clone Github Repository

===========================

Build & Deployment Process

===========================

Download project from github

package the project using maven

Maven will create war file

Deploy war file into tomcat (post build action)

Note: Above build & deployment process can be automated using Jenkins

Access application using URL in browser

============================

Jenkins Job Creation Process

=============================

1) Login into Jenkins

2) Configure Maven in Global Tool Configuration (Jenkins will download maven)

-> Manage Jenkins

-> Global tools Configurations

-> Add Maven

3) Install 'Deploy To Container' Plugin (To deploy war to tomcat server)

-> Manage Jenkins

-> Manage Plugins

-> Go to available tab

-> search for 'Deploy To Container' plugin

-> Click on install without restart

Note: Git s/w will be available by default in Jenkins Global Tools Configuration

4) Create Free Style Project

5) Enter Git Repo URL

6) Build Trigger (configure Maven which is added in Global Tools and Provide Maven Goals as clean package)

7) Add Tomcat Server in Post Build Action For deployment

8) Save the Job configuration

9) Run the Job

10) See Tomcat Dashboard (Application should display) and access the application

=================

Poll SCM

==================

-> Click on Job name

-> Click On Configure

-> Configure Poll SCM with cron expression as ( \* \* \* \* \*)

-> Every minute it will check for code changes, if code changes available then jenkins job will run

============================

How to deploy web applications ?

============================

-> To deploy a web application we need a server

-> We are using Apache Tomcat as web server to run web applications

-> Web Application will be packaged as war file

-> WAR file we will keep in tomcat server webapps folder.

-> webapps folder is called as Deployment folder

===================================

How to deploy Spring Boot Application ?

===================================

-> Spring Boot application will execute from main ( ) method

-> Spring Boot Application can be deployed as jar file

-> We no need to configure external server to run / deploy spring boot application

-> Spring Boot having embedded server to run Spring Boot Application

========================================

Running Spring Boot Application In AWS Cloud

========================================

1) Create Linux Virtual Machine (Amazon Linux)

2) Connect Linux Virtual Machine using MobaXterm

3) Install Java software

$ sudo yum install java

4) Upload Spring Boot Jar file into Linux VM

5) Run Spring Boot Jar file using below command

$ java -jar <jar-file-name>

6) Enable 8080 port number in security group of our Linux Virtual Machine

7) Access our application in browser

URL : http://public-ip:8080/

DSR:10/06/2024

Hello,

i. What I did today:

1. Started with Jenkins.

2. Explored the Jenkins Dashboard.

3. Checked various options while creating a Jenkins jobs:

i. Source Code Management

ii. Build Triggers

iii. Build Steps

iv. Post-build actions.

4. Created simple jobs to understand the workflow of Jenkins.

5. Installed different plugins while learning.

ii. What will I do tomorrow-

Will study Jenkins and CI further.

iii. Any roadblocks -

No roadblocks.

Deploying Spring Boot Application

1. Clone the project

git clone <https://github.com/coolgourav147/spring-boot-war-example.git>

1. Install Maven

sudo apt update

sudo apt install maven -y

Cd project

1. mvn test
2. mvn install
3. apt-get install tomcat10
4. cp -rvf target/maven-web-app.war /var/lib/tomcat10/webapps/app.war

**Types of jobs:**

1. Freestyle Project
2. Maven
3. External Job
4. Multi Configuration
5. Pipeline

**Jenkins job anatomy:**

* Name/description
* Advanced Option
* Source Code Management
* Build Triggers
* Pre Build
* Build
* Post Build

### **Types of Build Triggers**

1. Trigger builds remotely - Jobs can be triggered remotely outside of jenkins.
2. Build after other projects are built
3. Build periodically
4. Poll SCM

**Jenkins Tutorial:**

* [Chapter 2: Setting up Learning Environment](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/020_install_jenkins.md)
* [Chapter 3: Jenkins Systems Configurations](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/030_configure_jenkins.md)
* [Chapter 4: Creating First Project with Jenkins](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/040_creating_first_job.md)
* [Chapter 5: Adding Build Triggers](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/050_add_build_triggers.md)
* [Chapter 6: Building a Pipeline](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/060_building_jobs_pipeline.md)
* [Chapter 7: Preparing to build Java Projects](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/070_preparing_for_java_builds.md)
* [Chapter 8: Creating Java Build Project](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/080_creating_java_build_job.md)
* [Chapter 9: Integrating with Artifactory](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/090_resolving_libs_from_artifactory.md)
* [Chapter 10: Creating Test Job](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/100_creating_test_job.md)
* [Chapter 11: Static Code Analysis with SonarQube](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/110_static_code_analysis_with_sonarqube.md)
* [Chapter 12: Creating Package Job](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/120_create_package_job.md)
* [Chapter 13: Deploy application to Tomcat](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/130_deploy_to_tomcat.md)
* [Chapter 14: Create Docker Image](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/140_create_docker_image.md)
* [Chapter 15: Deploy application Using Docker Compose](https://github.com/schoolofdevops/learn-jenkins/blob/master/continuous-delivery/chapters/150_Deploy_with_Docker_compose.md)

**Declarative Pipeline Syntax:**

pipeline {

agent any

stages {

stage('Build') {

steps {

// Your build steps here

echo 'Building...'

sh 'make'

}

}

stage('Test') {

steps {

// Your test steps here

echo 'Testing...'

sh 'make test'

}

}

stage('Deploy') {

steps {

// Your deployment steps here

echo 'Deploying...'

sh 'make deploy'

}

}

}

post { // execute some logic after all stages

always{

echo 'Executed always!'

}

success {

echo 'Pipeline succeeded!'

// Additional actions on success

}

failure {

echo 'Pipeline failed!'

// Additional actions on failure

}

}

}

Multi-configuration Project

Multi-branch Project

Email Notification

**Jenkins Installation:**

#!/bin/bash

sudo yum update –y

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

sudo yum upgrade

sudo dnf install java-17-amazon-corretto -y

sudo yum install jenkins -y

sudo systemctl enable jenkins

sudo systemctl start jenkins

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

DSR:19/06/2024

Hello,

i. What I did today:

1. Hands on - Master Slave Architecture - Launch agent by connecting it to controller - Connected Linux Slave

2. Hands on - Master Slave Architecture - Launch agent by connecting it to controller - Connected Windows Slave

3. Tried a simple Multi-configuration project.

- Added two axis, Platform and Browser

4. Understood Multi-configuration project.

5. Revised Docker

ii. What will I do tomorrow-

Will study Jenkins and CI further.

iii. Any roadblocks -

No roadblocks.

**Jenkins Topics:**

1. Jenkins Job Anatomy

* Name/description
* Advanced Option
* Source Code Management
* Build Triggers
* Pre Build
* Build
* Post Build

1. Types of Jobs - Freestyle, Multi-configuration, Multi-branch, Pipeline, Maven
2. Adding Build Triggers
3. Adding Slave using two methods - using shh, by connecting to the controller (JNLP)-ubuntu, windows
4. Pipeline as a code
5. Email Notification in Jenkins
6. Git Integration in Jenkins
7. Parallel Jobs
8. Parameterized Jobs
9. Jenkins Workspace

Set up webhooks in GitHub

Added checkout script in jenkinsfile

Added Access and Secret Key in System Configuration