DevOps
Get more done

Jenkins Pipeline as Code

Learner Level

Cognizant

Agenda

Pipeline as Code with Jenkins Pipeline

- Pipeline
  - Pipeline types: Scripted and Declarative
  - Pipeline Concepts
  - Pipeline Syntax
  - Pipeline Examples
  - Pipeline Best Practices
- Demonstration
- Queries

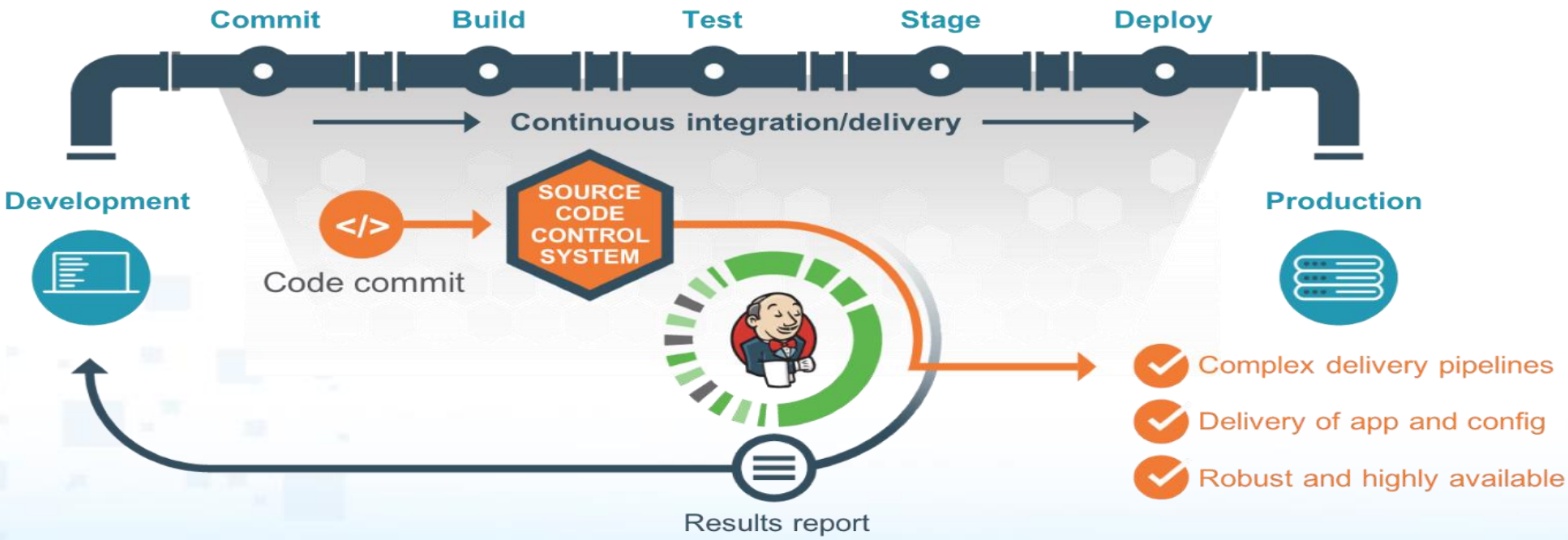
DevOps

Pipeline

https://jenkins.io/doc/book/pipeline/getting-started/

All the standard jobs in CI/CD process defined by Jenkins are manually written in one Script; stored and maintained in VCS

Features: Code, Durable, Pausable, Versatile, Extensible



DevOps

Prerequisites

- Jenkins 2.x or later
- Jenkins Pipeline Plugins
  - Jenkins Pipeline is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins. Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline DSL

  https://plugins.jenkins.io/workflow-aggregator
- Blue Ocean Suite

  https://plugins.jenkins.io/blueocean

DevOps

Pipeline as a code

The Text file (Jenkinsfile) stores the pipeline as code which contains the Series of jobs can be run in parallel manner or one after another with some extensible features such as,

Pipeline features:

- Can be maintained in VCS
- Allow Conditional Loops
- Incorporates User input
- Restart from saved Checkpoint
- Run Jobs in parallel
- Integrate with other plugins

DevOps

Pipeline Syntax

Scripted Pipeline

- Traditional way of writing code
- Stricter Groovy Syntax
- Code is written on Jenkins UI Instance
- Code defined within node block

Declarative Pipeline

- Recent Feature
- Simpler Groovy Syntax
- Code is written locally in a file & Maintained(VCS)
- Code defined within Pipeline block

https://jenkins.io/doc/book/pipeline/syntax/

Groovy DSL

- Written in Groovy DSL; short introduction about what is DSL and why pipeline is made up of Groovy DSL.
- What is DSL?
  - Language Targeted at particular type of problem/Domain
  - Language that domain expert understands
  - Syntax focused on particular domain/problem
  - Unlike GPL, you can’t use it for all kinds of stuff
  - Limited in scope and capability
  - Hence, they are small and simple
  - Easier to analyze, port, learn; it’s safer and provide meaningful errors
  - DSL is one way to increase level of abstraction.
  - DSL users can be either programmers or domain experts

DevOps

DSL and DOMAIN

DSL	Domain
SQL	Database Manipulation
Postscript	Publishing
Hibernate	Object Relational Mapping
Regex	Pattern Matching
BNL	Business Natural Language
Adhersion	Telecom

Pipeline Concepts

- Pipeline
  - A Pipeline is a user-defined model of a CD pipeline. A Pipeline's code defines your entire build process, which typically includes stages for building an application, testing it and then delivering it. Also, a pipeline block is a key part of Declarative Pipeline syntax.
- Node
  - A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline. Also, a node block is a key part of Scripted Pipeline syntax.
- Stage
  - A stage block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualize or present Jenkins Pipeline status/progress.
- Step
  - A single task. Fundamentally, a step tells Jenkins what to do at a particular point in time (or "step" in the process). For example, to execute the shell command make use the sh step: sh 'make'. When a plugin extends the Pipeline DSL, [1] that typically means the plugin has implemented a new step

DevOps

Pipeline Concepts breakdown

Jenkinsfile (Declarative Pipeline)

```
Pipeline
 {
        agent any
        stages {
                stage('Build')
                        {
                        steps { // }
                        }
                stage('Test')
                        {
                        steps { // }
                        }
                stage('Deploy')
                        {
                        steps { // }
                        }
                }

 }
```


DevOps

Pipeline Best Practices

Do

- *Use the real Jenkins Pipeline*
- *Develop your pipeline as code*
- *All work within a stage*
- *All material work within a node*
- *Work you can within a parallel step*
- *Acquire nodes within parallel steps*
- *Wrap your inputs in a timeout*

Don't

- *Use input within a node block*
- *Set environment variables with env global variable*
- *Prefer stashing files to archiving*

https://www.cloudbees.com/blog/top-10-best-practices-jenkins-pipeline-plugin

DevOps

Pipeline Examples

- **Hello World (Declarative and Scripted)**
- **Jobs In Parallel**
- **Load From File**
- **Office 365 Connector (General)**

- Ref: https://jenkins.io/doc/pipeline/examples/

DevOps

Basics

PIPELINE SCRIPT	EXAMPLE(S)
stage *Stage*	`stage 'build'` `stage concurrency: 3, name: 'test'`

node
Allocate a node
	`node('ubuntu') {`
	`  // some block`
	`}`

ws
Allocate a workspace
	`ws('sub-workspace') {`
	`  // some block`
	`}`

echo *Print a message*	`echo 'Hello Bees'`
batch *Windows batch script*	`bat 'dir'`
sh *Shell script*	`sh 'mvn -B verify'`
checkout *General SCM*	`checkout([$class: 'GitSCM', branches: [[name: '*/master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[url: 'http://github.com/cloudbees/todo-api.git']]])`
git *Git SCM*	`git 'http://github.com/cloudbees/todo-api.git'`
svn *Subversion SCM*	`svn 'svn://svn.cloudbees.com/repo/trunk/todo-api'`
step *General build step*	`step([$class: 'JUnitResultArchiver', testResults: 'target/test-reports/*.xml'])` `step([$class: 'Mailer', notifyEveryUnstableBuild: true, recipients: 'info@cloudbees.com', sendToIndividuals: false])`

wrap	`wrap([$class:'Xvnc', useXauthority: true]){`
	`  sh 'make selenium-tests'`
	`}`

tool
Install a tool
	`def mvnHome = tool name: 'M3'`
	`sh "${mvnHome}/bin/mvn -B verify"`
	`tool name: 'jgit', type: 'hudson.plugins.git.GitTool'`

mail *Send an e-mail*	`mail body: 'Uh oh.', charset: '', from: '', mimeType: '', replyTo: '', subject: 'Build Failed!', to: 'dev@cloudbees.com'`

DevOps

Flow Control

PIPELINE SCRIPT	EXAMPLE(S)
sleep *Sleep*	`sleep 60` `sleep time: 1000, unit: 'NANOSECONDS'`

waitUntil
Wait for condition
	`waitUntil {`
	`  // some block`
	`}`

retry
Retry body up to N times
	`retry(5) {`
	`  // some block`
	`}`

input *Pause for manual or automated intervention*	`input 'Are you sure?'` `input message: 'Are you sure?', ok: 'Deploy', submitter: 'qa-team'`

parallel
Execute sub-flows in parallel
	`parallel "quality scan": {`
	`  // do something`
	`}, "integration test": {`
	`  // do something else`
	`},`
	`failFast: true`

timeout
Execute body with a timeout
	`timeout(time: 30, unit: 'SECONDS') {`
	`  // some block`
	`}`

error *Stop build with an error*	`error 'No sources'`

DevOps

References

CB Pipeline URL

https://www.cloudbees.com/blog/using-pipeline-plugin-accelerate-continuous-delivery-part-1

https://www.cloudbees.com/blog/using-pipeline-plugin-accelerate-continuous-delivery-part-2

https://www.cloudbees.com/blog/using-pipeline-plugin-accelerate-continuous-delivery-part-3

Pipeline Plugin

Wiki: https://wiki.jenkins.io/display/JENKINS/Pipeline+Plugin

Pipeline Best Practices:

https://www.cloudbees.com/blog/top-10-best-practices-jenkins-pipeline-plugin

Pipeline Examples:

https://jenkins.io/doc/pipeline/examples/

Demonstration

Queries

Thank You !!!

DevOps