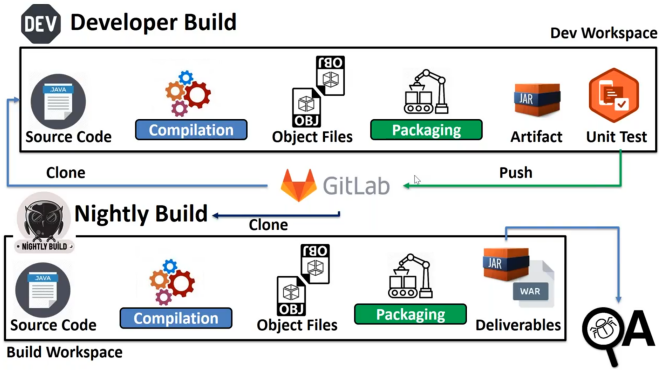
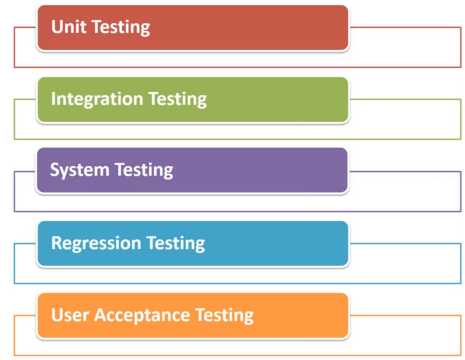
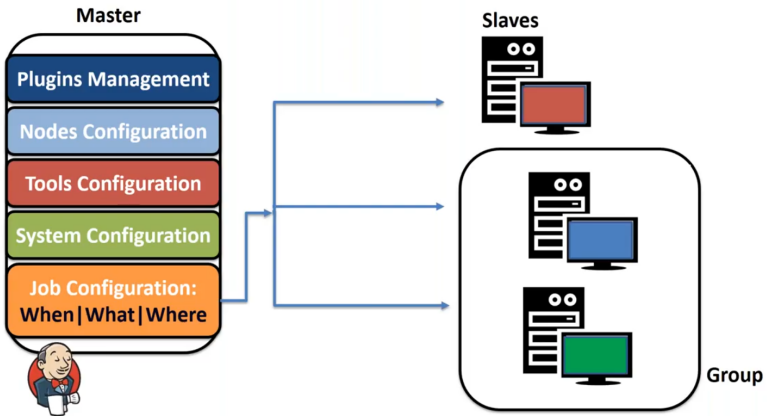




Jenkins Architecture



- **Setup SSH Keys on Master**

$ ssh-keygen -t rsa

* Take the content of ~/.ssh/id_rsa.pub (as ubuntu user) from master & put it in ~/.ssh/authorized_keys on slave as ubuntu user

- **Install JDK 8**

$ apt update
$ apt install -y openjdk-8-jdk

Setup Slave on Ubuntu 18

- **Setup Passwordless SSH between Jenkins Master & Slave**

✓ Take the content of **~/.ssh/id_rsa.pub** (as ubuntu user) from **master**
✓ Put it in **~/.ssh/authorized_keys** on **slave** as ubuntu user

- **Setup Slave on Jenkins**

❑ Add new node & make sure the **Remote Root Directory** has **read/write** permission for Ubuntu user
❑ Choose Launch method as "**SSH**"
  ✓ Give the private IP of the EC2 slave for "Host"
  ✓ Add Credentials, choose Kind as "SSH Username with private key"
  ✓ Give user name as "ubuntu" & put the content of "**~/.ssh/id_rsa**" from master as ubuntu & put it as private key content
  ✓ Choose 'Host Key Verification Strategy' as "**Manually trusted key verification strategy**"

Maven Environment Setup

JAVA_HOME
M2_HOME
M2
MAVEN_OPTS = *-Xms256m -Xmx512m*
PATH=%PATH%;%M2%

```
% yum install java-1.8.0-openjdk-devel (install JDK, check the binary link)
% cd /usr/local
% wget https://archive.apache.org/dist/maven/maven-3/3.5.3/binaries/apache-maven-3.5.3-bin.tar.gz
% tar xzf apache-maven-3.5.3-bin.tar.gz
% ln -s apache-maven-3.5.3  maven
export M2_HOME=/usr/local/maven
export PATH=${M2_HOME}/bin:${PATH}
```

```
mvn archetype:generate
```

What is Jenkins Pipeline?

- Pipeline is a group of events or jobs which are interlinked with one another in a sequence.
- **Jenkins Pipeline** is a combination of plugins that support the integration and implementation of **continuous delivery** pipelines using Jenkins
- Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline domain-specific language (DSL) syntax
- Instead of building several jobs for each phase of the pipeline, you can now code the entire workflow as one script i.e 'pipeline as code'

Why Jenkins Pipeline?

- It models simple to complex pipelines as code by using **Groovy DSL** (Domain Specific Language)
- The code is stored in a text file called the **Jenkinsfile** which can be checked into a **SCM** (Source Code Management)
- Improves user interface by incorporating **user input** within the pipeline
- It is **durable** in terms of unplanned restart of the Jenkins master
- It can restart from saved **checkpoints**
- It supports complex pipelines by incorporating conditional loops, fork or join operations and allowing tasks to be performed in parallel
- It can integrate with several other plugins

- A Jenkinsfile can be written using two types of syntax - **Declarative** and **Scripted**
- Scripted pipeline is a traditional way of writing the code.
  - uses stricter groovy based syntaxes
  - Jenkinsfile is written on the **Jenkins UI instance**

- Declarative Pipeline is a more recent feature of Jenkins Pipeline which:
  - provides richer syntactical features over Scripted Pipeline syntax, and
  - is designed to make writing and reading Pipeline code easier

What is a Jenkinsfile?

- Jenkinsfile is a text file that stores the entire workflow as code and it can be checked into a SCM on your local system
- This enables the developers to **access, edit and check the code at all times**
- The Jenkinsfile is written using the **Groovy DSL**
- Jenkinsfile can be created through a **text/groovy editor** or through the **configuration page** on the Jenkins instance
- While the syntax for defining a Pipeline, either in the web UI or with a Jenkinsfile is the same, it is considered best practice to define the Pipeline in a Jenkinsfile and check that in to source control:
  - ✓ Automatically creates a Pipeline build process for all branches and pull requests.
  - ✓ Code review/iteration on the Pipeline
  - ✓ Single source of truth [3] for the Pipeline, which can be viewed and edited by multiple members of the project.

Steps:

- A step is nothing but a single task that executes a specific process at a defined time
- These steps are carried out in sequence to execute a stage.
- There must be at least one step within a steps

Syntax:

```
pipeline {
    agent {    //DSL    }
    stages {
        stage ('STAGENAME') {
            steps {
                //DSL
            }
        }
    }
```

Pipeline Concepts

Pipeline:

- This is a user defined block which contains all the processes such as build, test, deploy, etc.
- It is a collection of all the stages in a Jenkinsfile. All the stages and steps are defined within this block.
- It is the key block for a declarative pipeline syntax.

Syntax:

```
pipeline {
    // DSL
}
```

Node/Agent:

- A node is a machine that executes an entire workflow. It is a key part of the scripted pipeline syntax.
- A agent is used on declarative pipeline syntax

Syntax:

```
pipeline {
    agent {    // DSL    }
}
```

Stage:

- The work is specified in the form of stages.
- There must be at least one stage & can be more than one stage within this directive.
- Each stage performs a specific task

Syntax:

```
pipeline {
stages {agent {    //DSL    }
        stage ('STAGENAME') {
            //DSL
        }
    }
}
```

```
pipeline{
  stages{
    stage("stage1"){
        agent {   }
        steps { }
    }
    stage("stage2"){
        agent {   }
        steps { }
    }
  }
}
```