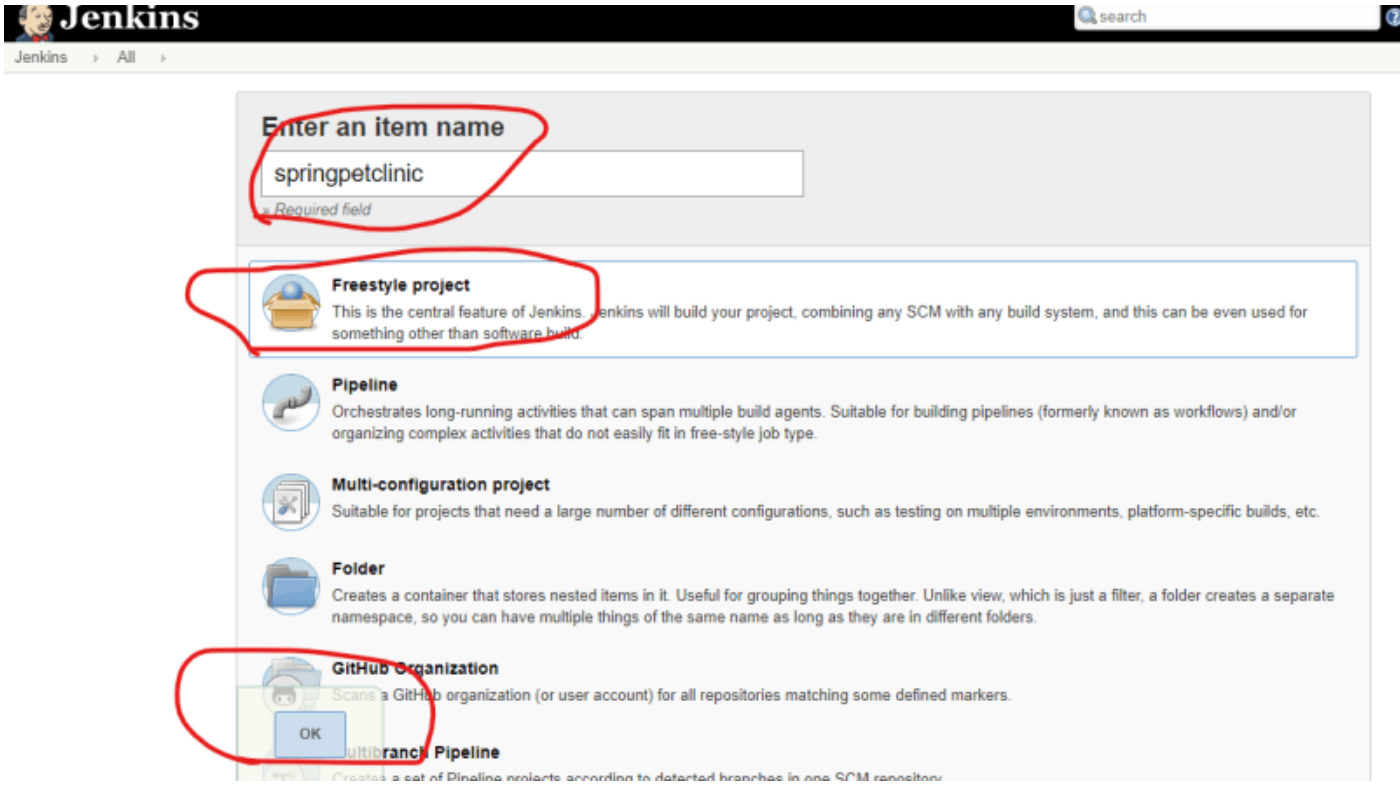
Jenkins Projects

- Two kinds of Popular Projects are available in Jenkins
  - FreeStyle:
    - Project to create schedules of any activities
    - Heavily relies on UI Components of Jenkins.
    - If UI Components are not found, they can be added using **plugins**
  - Pipeline

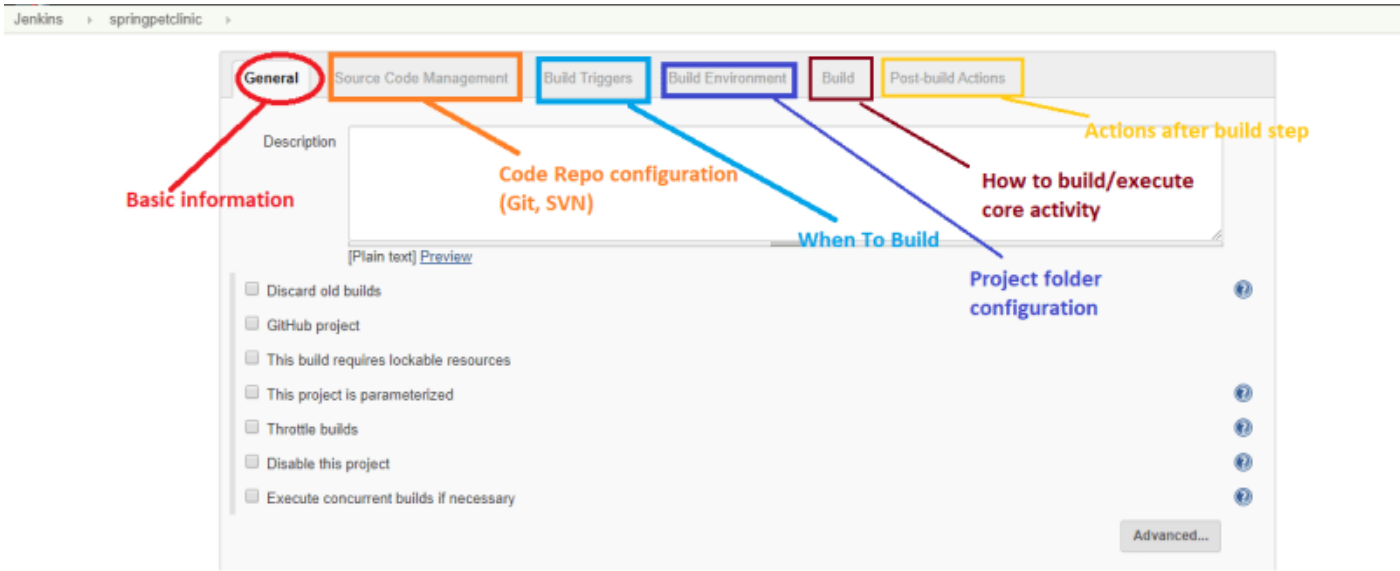
Create a free style project to clone the code from git (GitHub)

- The project we will be cloning is spring pet clinic [from here]
-
- Create a new Item and select the options as shown below



- To understand different sections of Jenkins Freestyle project, Refer

below

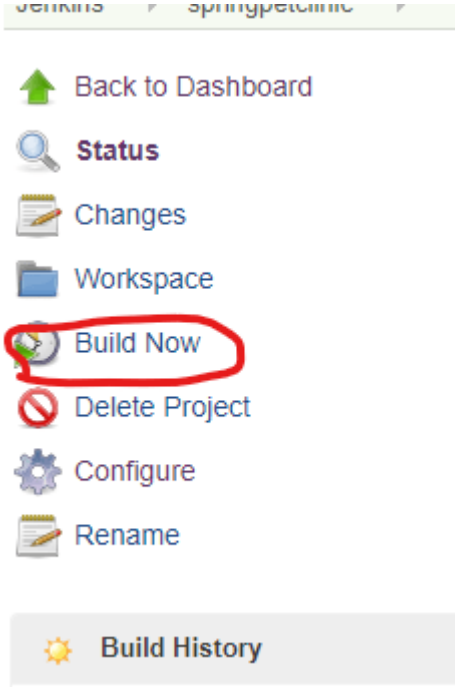


- Now configure Source Code Management with Git url https://github.com/spring-projects/spring-petclinic.git
- In the build section => Execute Shell

```
pwd
ls
```

- Lets examine what has happened in Jenkins Home Directory
- Whenever a project is created , In the Jobs folder of Jenkins HOme directory, a new folder with project name gets created. In this folder config.xml is project configuration, builds folder is available for symlinks of successful and pastbuilds

```
jenkins@ip-172-31-22-202:~/jobs$ tree springpetclinic/
springpetclinic/
├── builds
│   ├── legacyIds
│   └── permalinks
└── config.xml
```



- Click on Build Now as shown below
- When your project/job is executed a folder gets created in Jenkins Home Directory in workspace folder

Continuous Integration

- Purpose of CI is give feedback about the commit(s) from developer(s)

Continuous Delivery

- Purpose of CD is give feedback about the days work done by the dev team
- Importance is for System Test/Performance Testing

Build/Compile/test Java Using Maven.

- Building Java Code
  - Compiling Each and every .java file
  - archive (zip) all the generated .class files into war/jar/ear
- To do this build activity, there are many build tools
  - ANT
  - Maven
  - Gradle
- In this series we restrict ourselves to Maven
- Maven Installation on ubuntu

```
sudo apt-get update
sudo apt-get install maven -y
mvn -v
```

Maven

- Is a Project Management Tool.
- Can be used for
  - build
  - dependency Management
  - Releases
  - Documentation
  - Test Executions
- Maven prefers conventions over configurations
- Maven also works for Java Based Languages
  - groovy
  - Scala
- Maven uses a file called as **pom.xml** to define
  - dependencies
  - project information's
  - plugins

Installation

- Before installing maven ensure Java is installed
- Ubuntu:
-

```
sudo apt-get update
sudo apt-get install maven -y
```

Maven folder conventions

- Code: <projectdirectory>/src/main/java/
- Location of pom: <projectdirectory>/pom.xml
- Test: <projectdirectory>/src/test/java/
- Target Folder: <projectdirectory>/target

pom.xml

- POM (Project Object Model): is a xml which defines
  - Project info

- Dependencies
- Plugins
- Profiles

Goals

- compile : compile the code
- test: compile the code + test the code
- package: test the code + package the application
- install: pushes the pom file and jar/war to ~/.m2
- deploy: pushes the pom file and jar/war to Remote/Central Repo
- clean: remove the target folder

Executing goals

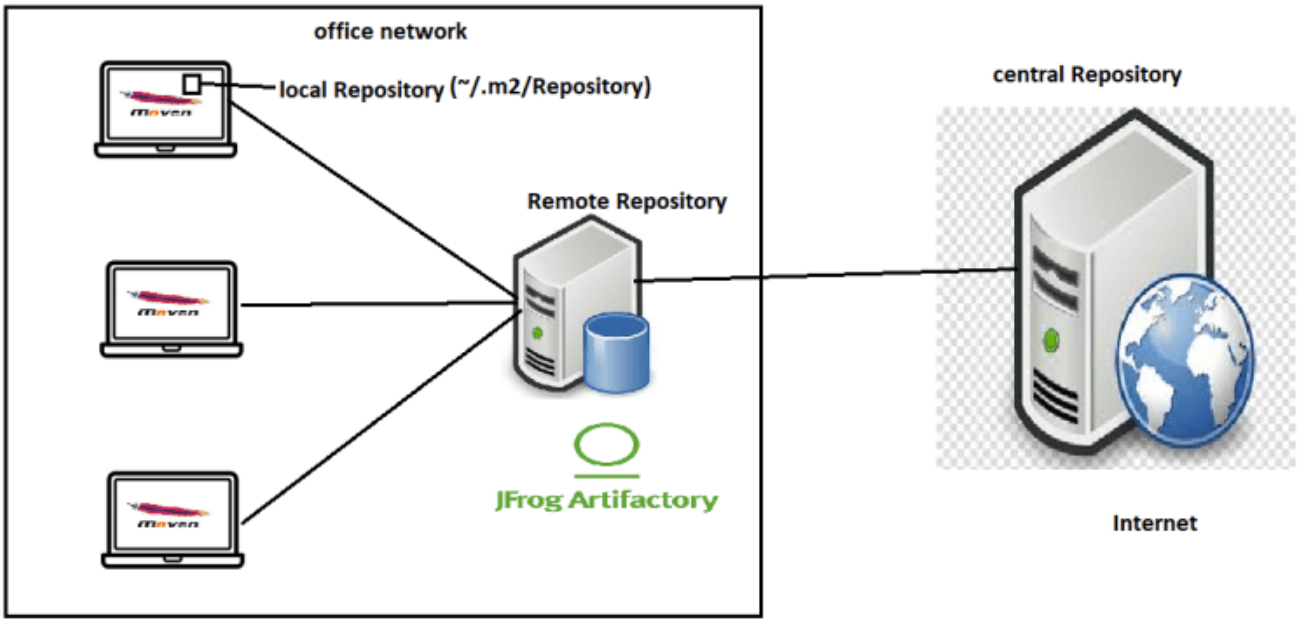
- single goal execution

```
mvn compile
mvn test
```

- multi goals

```
mvn clean package
```

Maven Repository Architecture



Using Maven with Jenkins

- Install maven and ensure Jenkins User has access to maven
- In the build section of Free style project
    - Execute Shell: directly execute the maven command
    - Invoke Top level Maven Targets: Specify goals from here

Testing Types

- Unit Testing:
    - Testing smaller unit of developments
    - Test Harness tools:
        - JUNIT
        - MSTest
        - pytest
- Automated System Testing:
    - web Based Applications:
        - Selenium
        - Karate
        - QTP
    - API Based Applications
        - Soap UI
        - Postman
    - Mobile Apps:
        - Appium
    - Performance Testing
        - JMeter

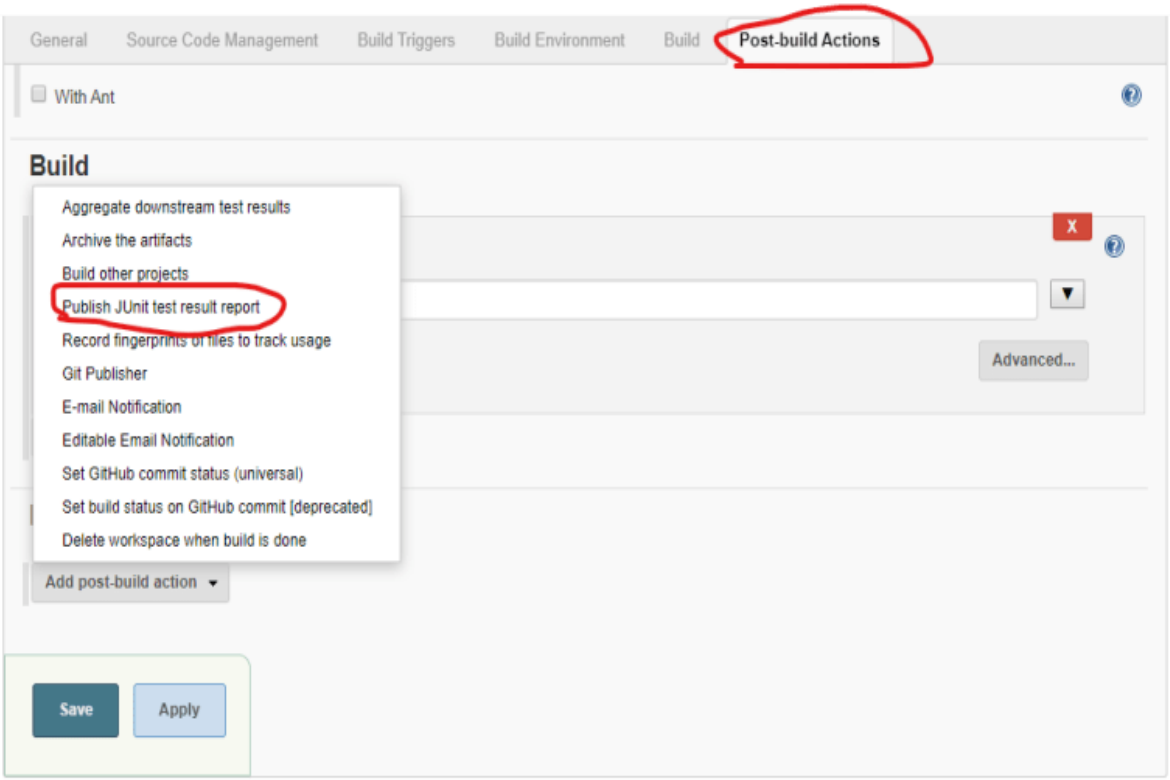
- Load Runner

What is Measured From Tests

- Pass Rate
- Coverage

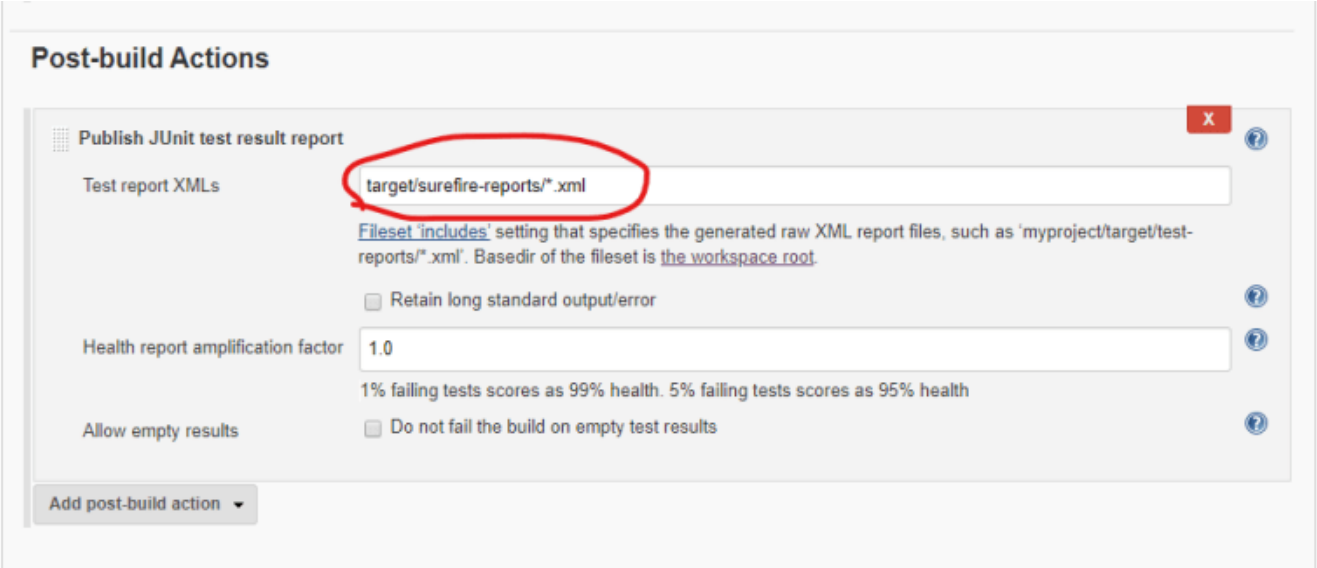
Integrating Maven tests with Jenkins

- To configure publishing of junit test results to jenkins
- ensure your maven goal has test execution
- Navigate to post-build Actions and Select as shown below



-

- Select xml files in surefire-reports folder in target



Archiving the Artifcat (Displaying the package built)

- In post build actions, select the section 'archive the artifact' and give the artifacts location (for eg: target/*.jar)

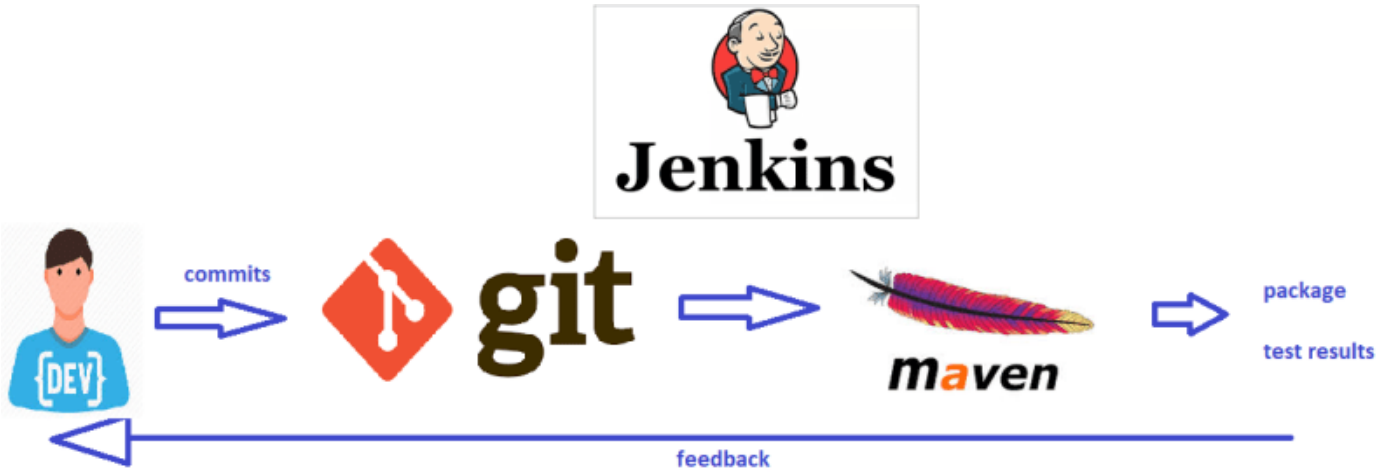
Post Build Actions:

- Activities that are performed after build is completed
- Most commonly post builds are
  - Show Test Results
  - Show the Package
  - Call other Jenkins Project to start building
  - Send Emails to the team

Jenkins Plugins

- Plugin is extra functionality into Jenkins
- Plugins can be installed into Jenkins. This installation can be from
  - Online
    - Plugins get downloaded from internet
  - Offline
    - Upload plugin to Jenkins
    - Plugin has two popular formats (hpi, jpi)

Day Builds



- Day builds basic intention is
  - to give feedback of code quality of the commit(s) done by one (or more) developers for shorter period of time during the day.
  - Have to finish quickly

Night Builds

- Night Builds basic intention is
  - to give feedback of the product quality for the collective work done by developers during past day
  - Execute extensive tests (unit, System, Performance)
  - Time is no bar

- Night Builds are used by system testers to execute the

Functional test Night Builds qualify for release

Jenkins Build Triggers

- Will help in triggering the Jenkins Build based on
  - Schedule:
    - Periodic: eg. Every 1 hour, Every 2 hours etc
    - On Schedule: eg Every weekday at 3 AM
  - Git Commits: Any new commits to Git Repo. Two ways of doing this
    - POll SCM: Jenkins will poll Git
    - Git/Web Hooks: Git will inform Jenkins whenever new commits happen

- After Other Jobs are Built

Build Periodically

- Here you configure the schedules by using syntax whichi is much like cron jobs.

```
MINUTE HOUR DOM MONTH DOW
H/15 * * * *
0 4,15 * * *
```

Poll SCM

- Triggers a build whenever changes appear in git.
- Using Jenkins, user should describe how frequently poll should happen. For this we use the same syntax mentioned above

Build Environment

- Some of the Prebuild Actions such as cleaning workspace
- Build Settings such when build is taking more time , abort/fail etc can be configured
- Additions to console logs

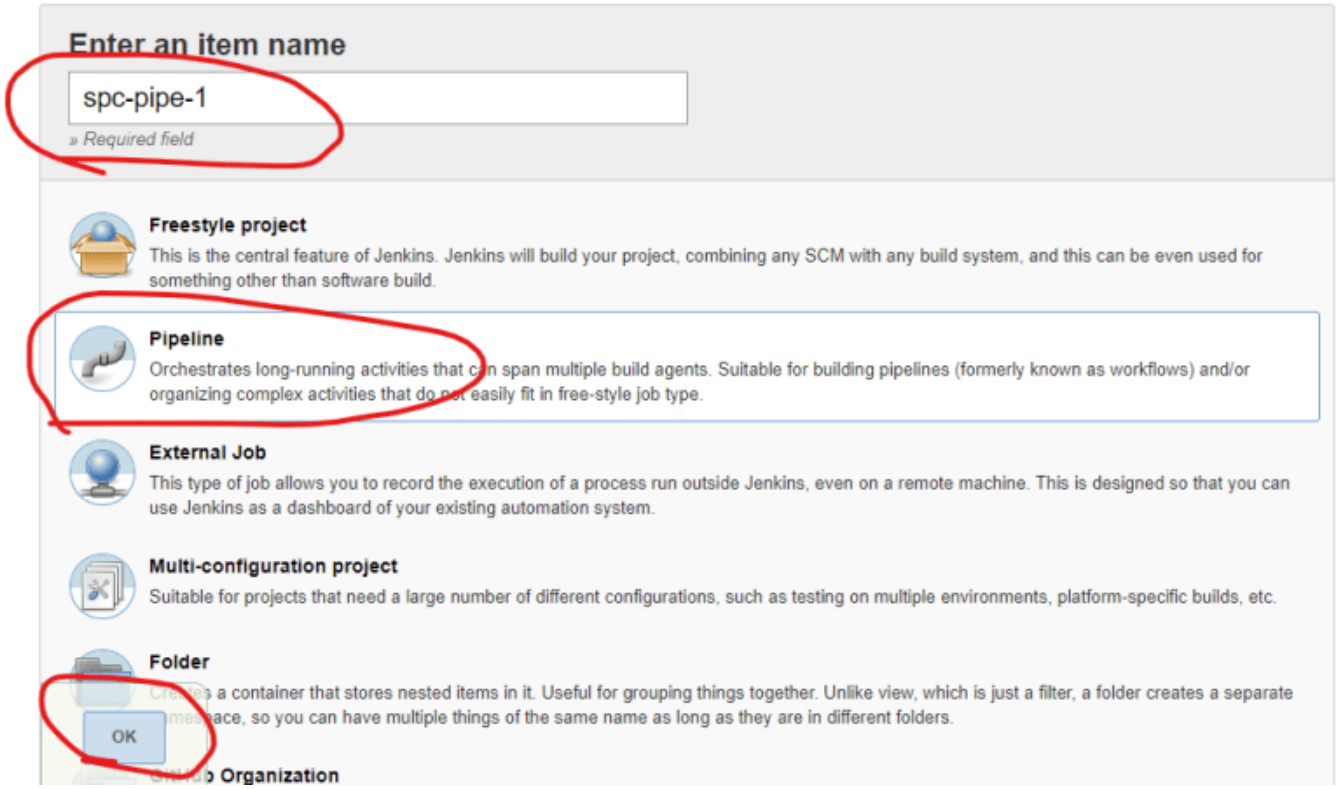
Jenkins Pipeline

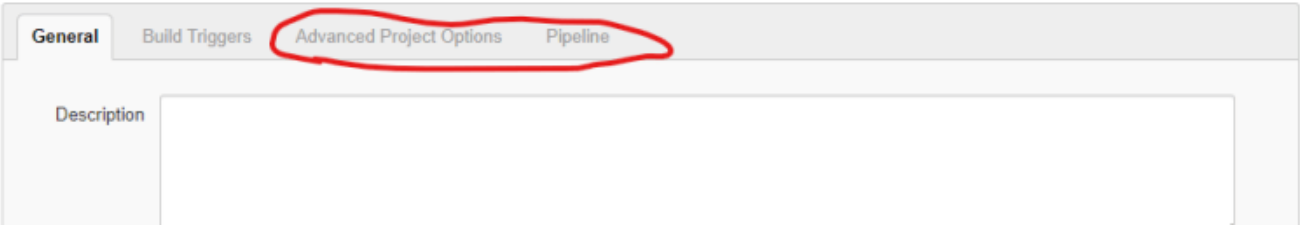
- Is a scripted way of defining jenkins job.
- Jenkins job can be defined in git repository in one file.
- Lets try this.
- Navigate to code repository and create a file **Jenkinsfile** with following content
-

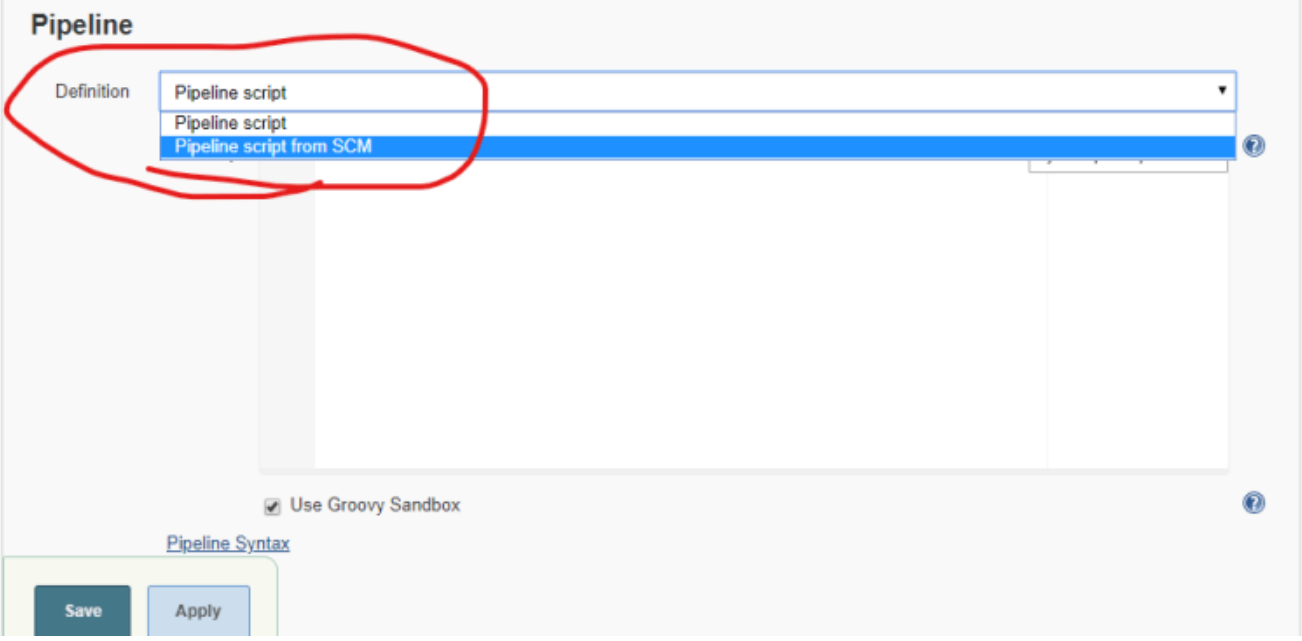
```
node {
```

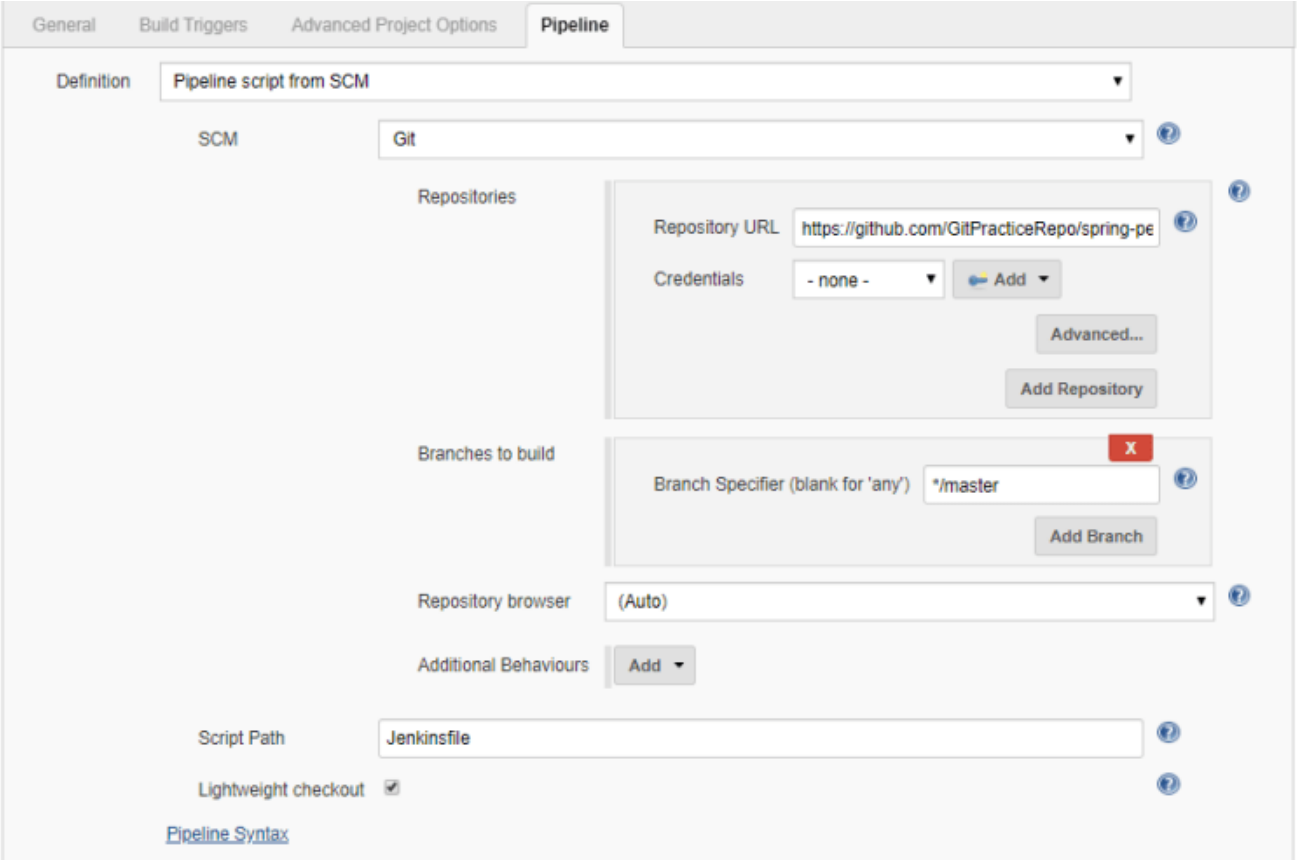
```
   stage('SCM') {
      // git clone
           git 'https://github.com/GitPracticeRepo/spring-petclinic.git'
   }

   stage ('build the packages') {
      // mvn package
           sh 'mvn package'
   }



   stage ('archival') {
     // archiving artifacts
          archive 'target/*.jar'
   }

}
```

- Now open Jenkins and create a pipeline project as shown below



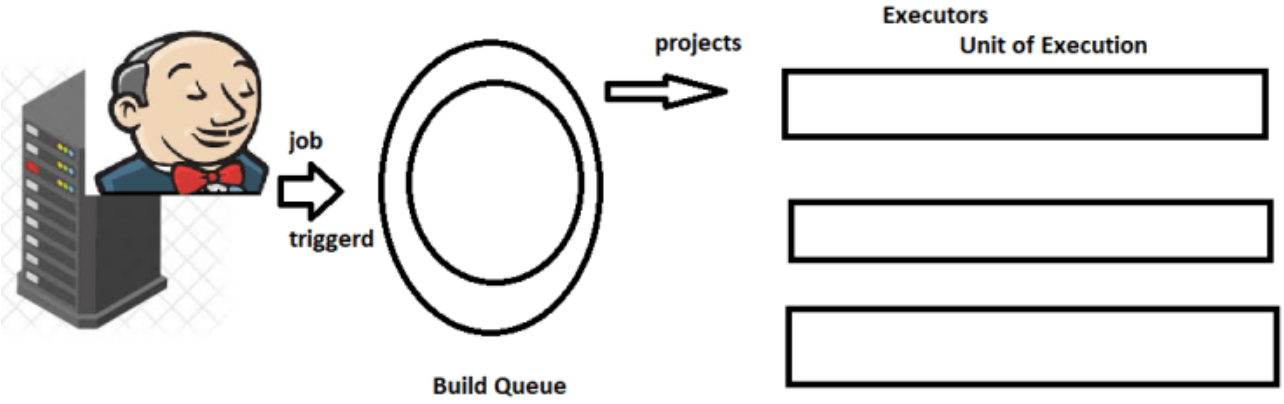






- In Jenkins Pipeline the script is written by using groovy language (Java Based Language) with this we get the following benefits
    - Customization of builds become simpler
    - Creating reusable build library is possible
    - Even in the case of plugins what we generally call from Script is functions.

Jenkins Logical Architecture



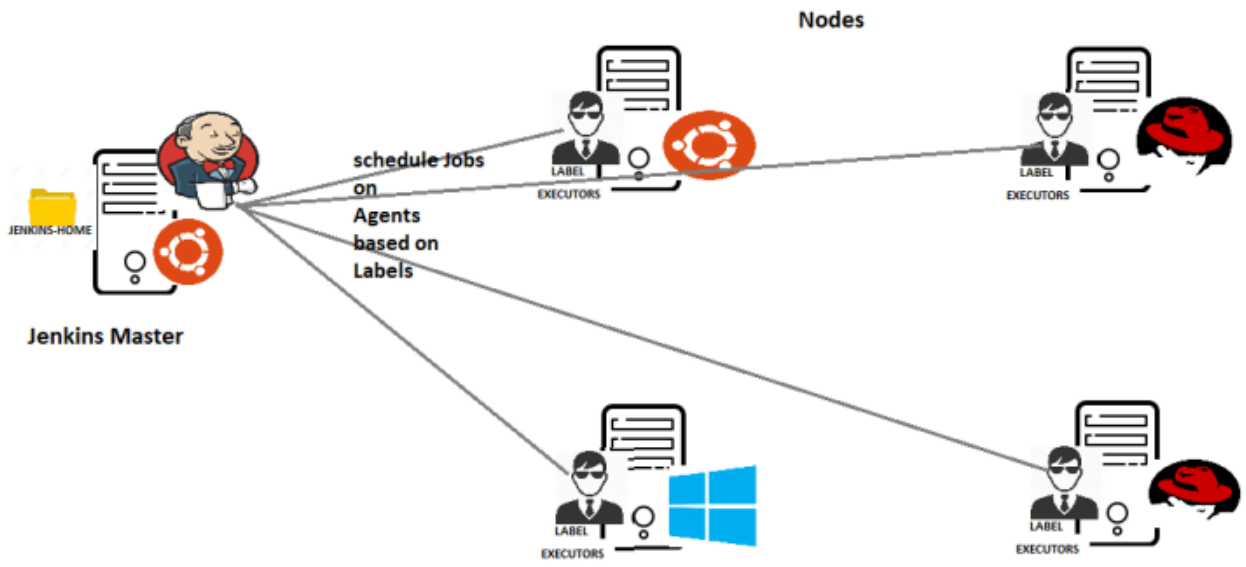
Queue

- Any build from Jenkins will be sent to Queue.
- It stays in the Queue till it finds free executor

Executor

- Is Execution unit of Jenkins (This is where build happens)
- Number of Executors signify number of parallel builds that can happen
- By Default, A project if it is under execution , will not be executed by other executors (Clicking Build now twice on same project will not lead to two executors)

Jenkins Distributed Builds



- Two kinds of Server Components are present in Jenkins
  - Master:
    - Jenkins Service will be running
    - Jenkins Home directory will be present
    - Communicates with Jenkins Agents using SSH/TCP (jnlp)
  - Node
    - Jenkins Agent will be running
    - Only the builds workspaces scheduled will be available
    - Every Agent should have some labels.

Jenkins Pipeline Syntax

- Pipelines are written in Groovy language.
- Jenkins has defined certain Functions and blocks
- Jenkins has Developed DSL (Domain Specific Language)

Jenkins Pipeline – Important Definitions

- Node: Machine where Jobs get executed. Node is idenified by labels

```
node('LABEL') {
    // your build config
}
```

- Stage: Complete build activity can be broken into multiple stages.

```
stage('<Stage name>') {


}

# Generally stages are part of nodes

node('MAVEN') {
    stage('GIT') {

    }
    stage('BUILD'){

    }
}
```

- shell: Executing shell in jenkins

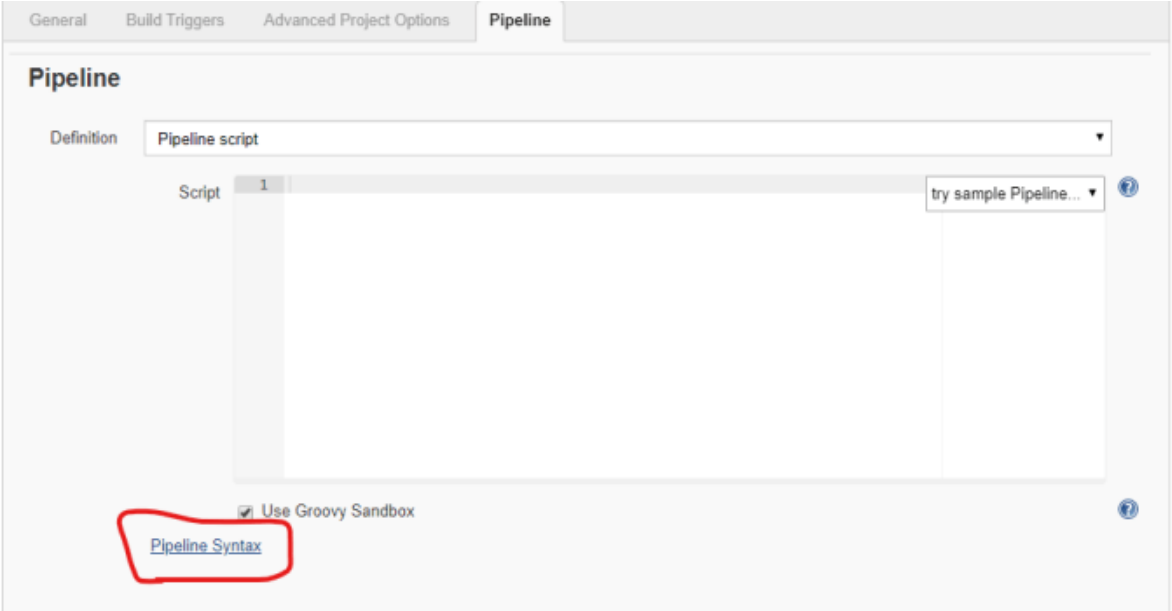
```
sh <command>

sh 'mvn clean'
```

- git: Executes the git clone and pull operation. Refer here
-

```
git '<url>'
```

Quick Wins for generating pipelines.

- Create a jenkins pipeline project and navigate to pipeline section perform the below
  steps



Jenkins › testing › Pipeline Syntax

- Back
- Snippet Generator
- Declarative Directive Generator
- Declarative Online Documentation
- Steps Reference
- Global Variables Reference
- Online Documentation
- Examples Reference
- IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step: archiveArtifacts: Archive the artifacts

Files to archive

Advanced...

Generate Pipeline Script

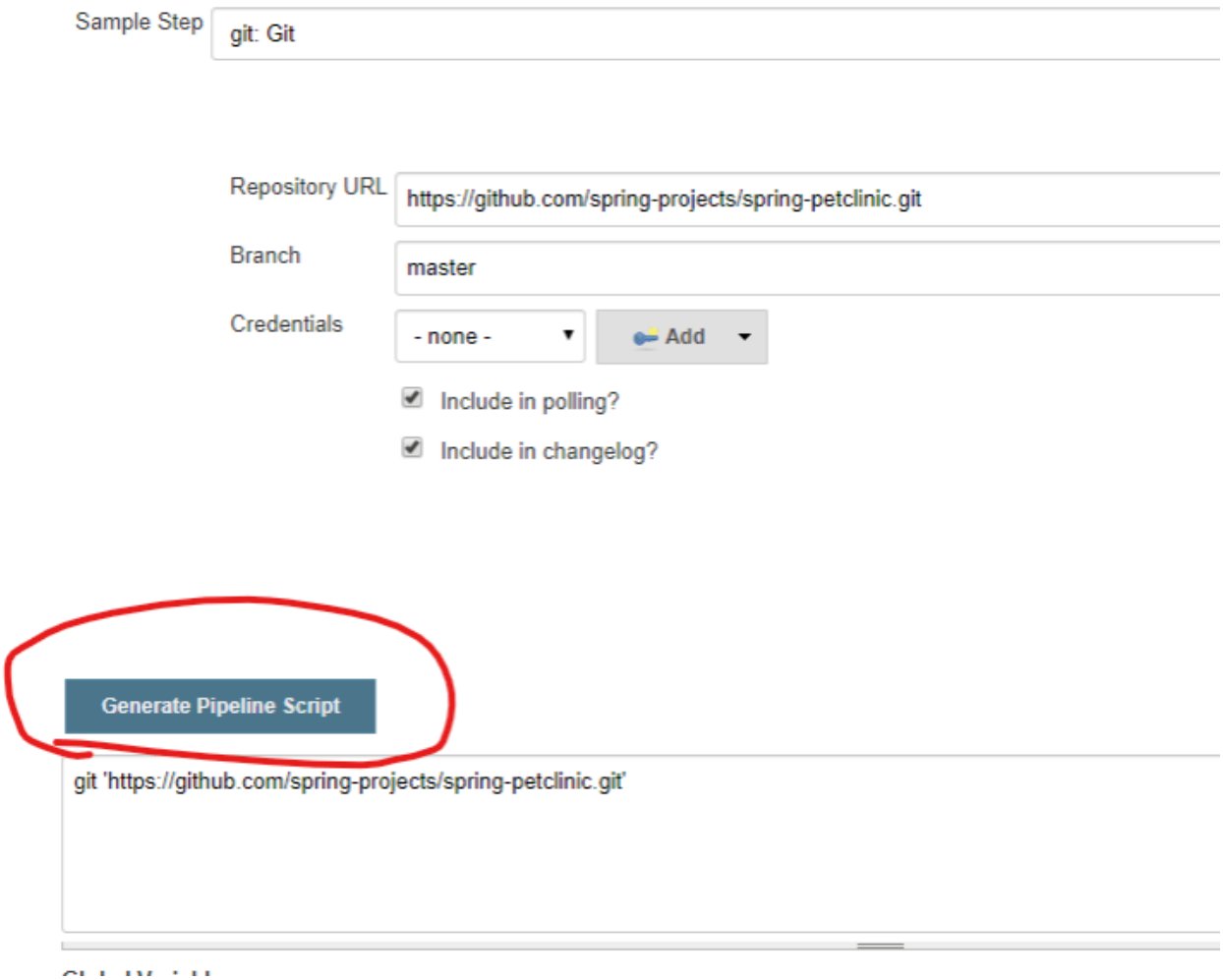
- Declarative Online Documentation
- Steps Reference
- Global Variables Reference
- Online Documentation
- Examples Reference
- IntelliJ IDEA GDSL

values.)

Steps

Sample Step: archiveArtifacts: Archive the artifacts

archiveArtifacts: Archive the artifacts
bat: Windows Batch Script
build: Build a job
catchError: Catch error and set build result to failure
checkout: Check out from version control
cleanWs: Delete workspace when build is done
deleteDir: Recursively delete the current directory from the workspace
dir: Change current directory
echo: Print Message
emailext: Extended Email
emailextrecipients: Extended Email Recipients
error: Error signal
fileExists: Verify if file exists in workspace
findBuildScans: Find published build scans
fingerprint: Record fingerprints of files to track usage
git: Git
input: Wait for interactive input
isUnix: Checks if running on a Unix-like node
junit: Archive JUnit-formatted test results
library: Load a shared library on the fly



Scripted Pipeline

- Directly allows users to write Groovy
- Learning curve towards Groovy
- Main blocks/steps are
    - Node
    - Stage
    - git
    - sh
    - bat
- Example

```
node('mvn') {
    stage('scm') {
        git 'https://github.com/springpetclinic.git'
    }
    stage('build') {
        if env.branch == 'master'
            sh 'mvn package'
        else
            sh 'mvn clean package'
    }
}
```

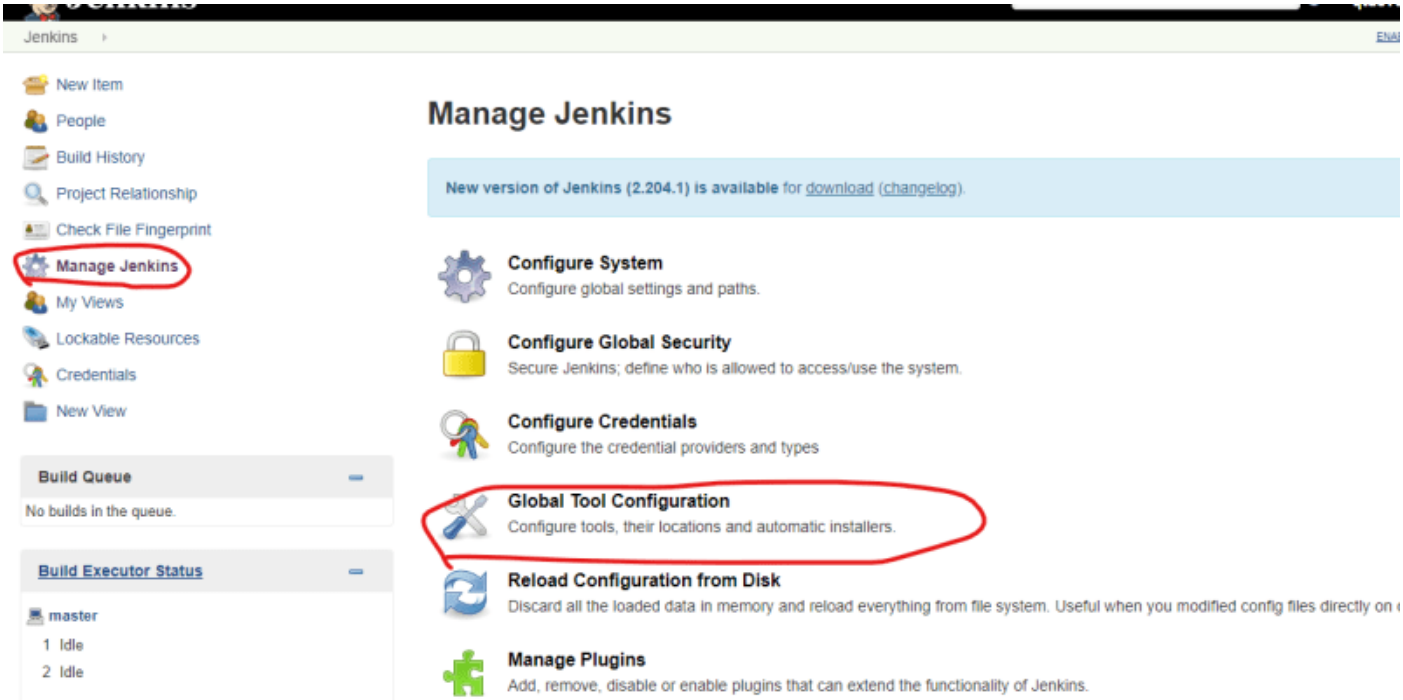
Declarative Pipeline

- Created Declarative Syntax for Jenkins (DSL)
- Simpler learning Curve
- Declarative Steps are
    - pipeline
    - stages
    - stage
    - label
    - when
    - agent
    - sh
    - git
-
- Example

```
pipeline {
    agent {
        label 'MVN'
    }
    stages {
        stage ('git') {
            ....
        }
        stage ('build') {
            ......
            when branch == 'master'
               ....
        }
    }
}
```

- [Refer Here](#) for specifics

Configuring Tools to Jenkins



- Configure Tools like
  - git
  - Maven
  - Docker
  - Ant
  - Gradle
  - Artifactory
  - SonarQube