DEVOPS

It's Not (Just) Tools

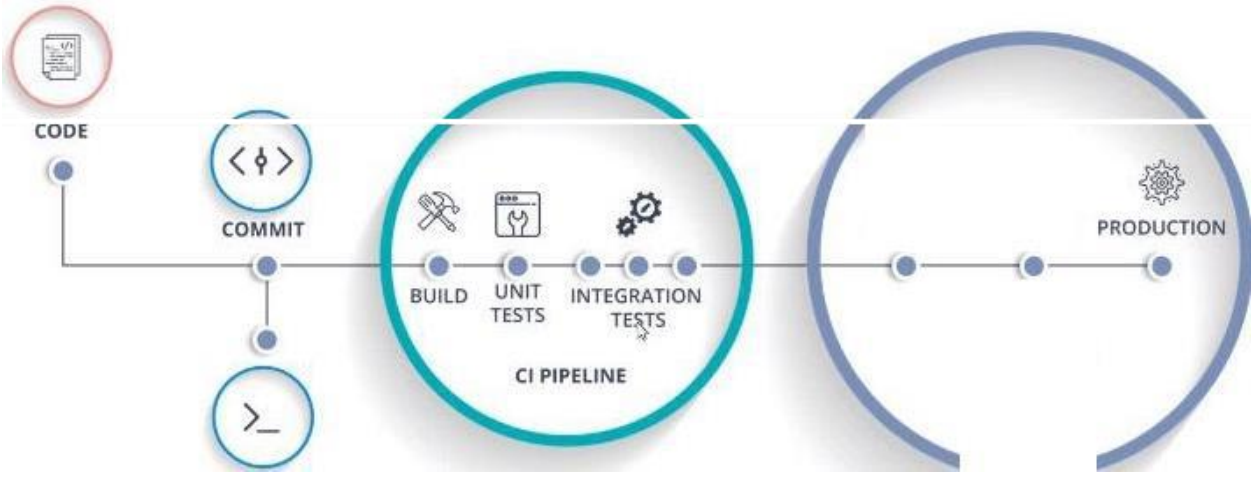
Never-ending process of continual improvement.

It is:

- Continuous Integration
- Continuous Development
- Continuous Testing
- Continuous Deployment
- Continuous Monitoring

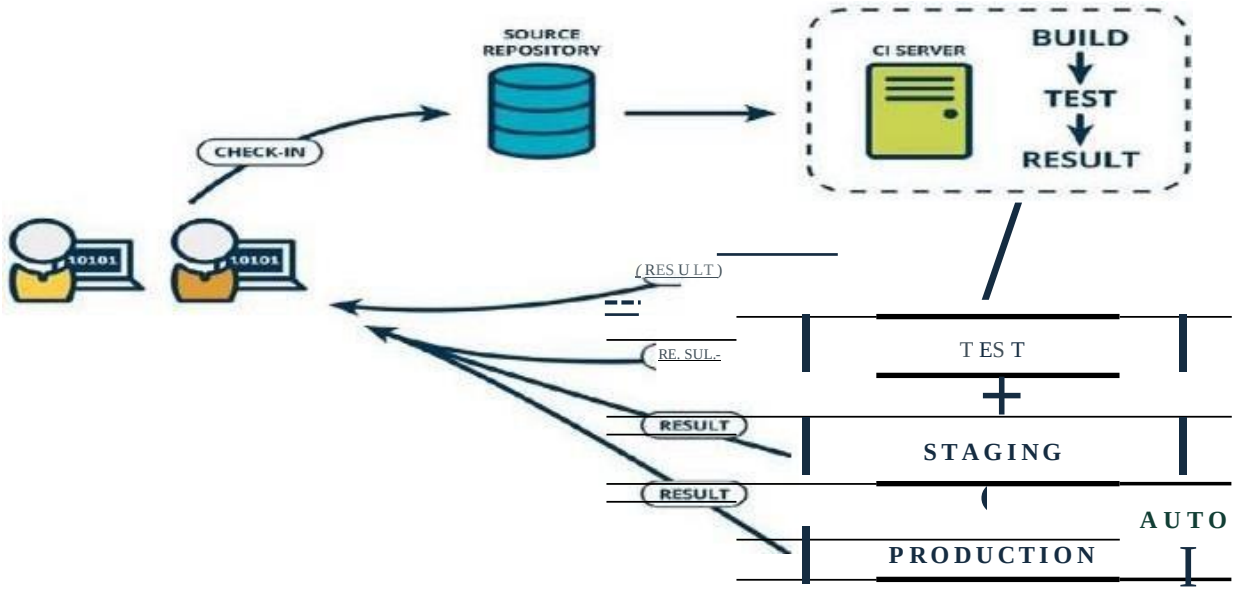
Why **Continuous Integration**?

- Early Detection of Bugs
  Helps in Determining Code-break
- Enables Continuous Deployment
- Enables Automated Testing
  Reduces the Risk of a Longer, Time-Consuming Project
- Increases the Quality of Software via continuous code quality



CONTINUOUS DELIVERY

- Various DevOps Stages and
- Continuous Delivery VS Deployment



ALL ABOUT GIT AND GITHUB

What is Version Control System?

- Version Control is the management of changes to documents, computer programs, large websites and other collection of information.
- Centralized version control
- Distributed version control

What is Git?

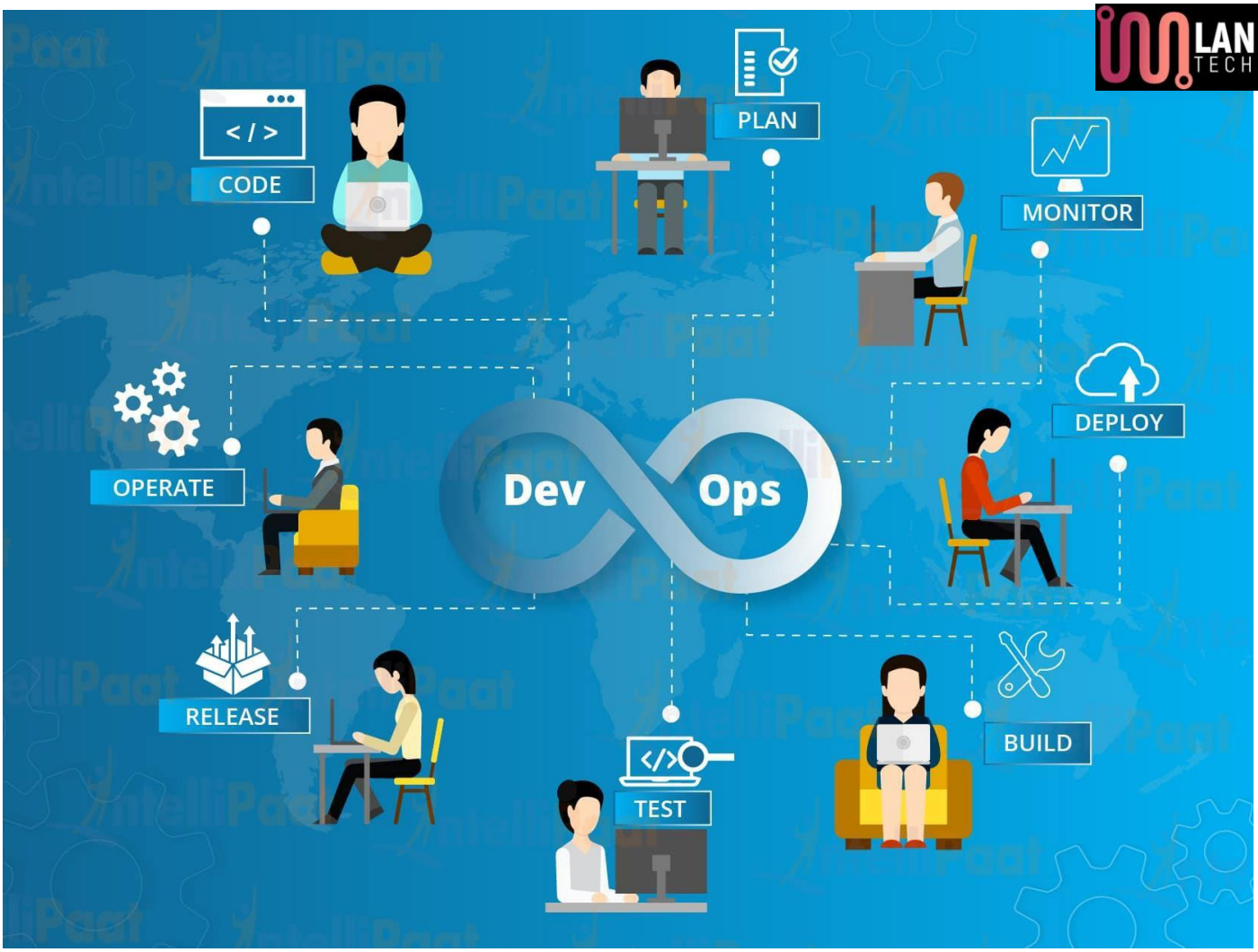
- Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software.

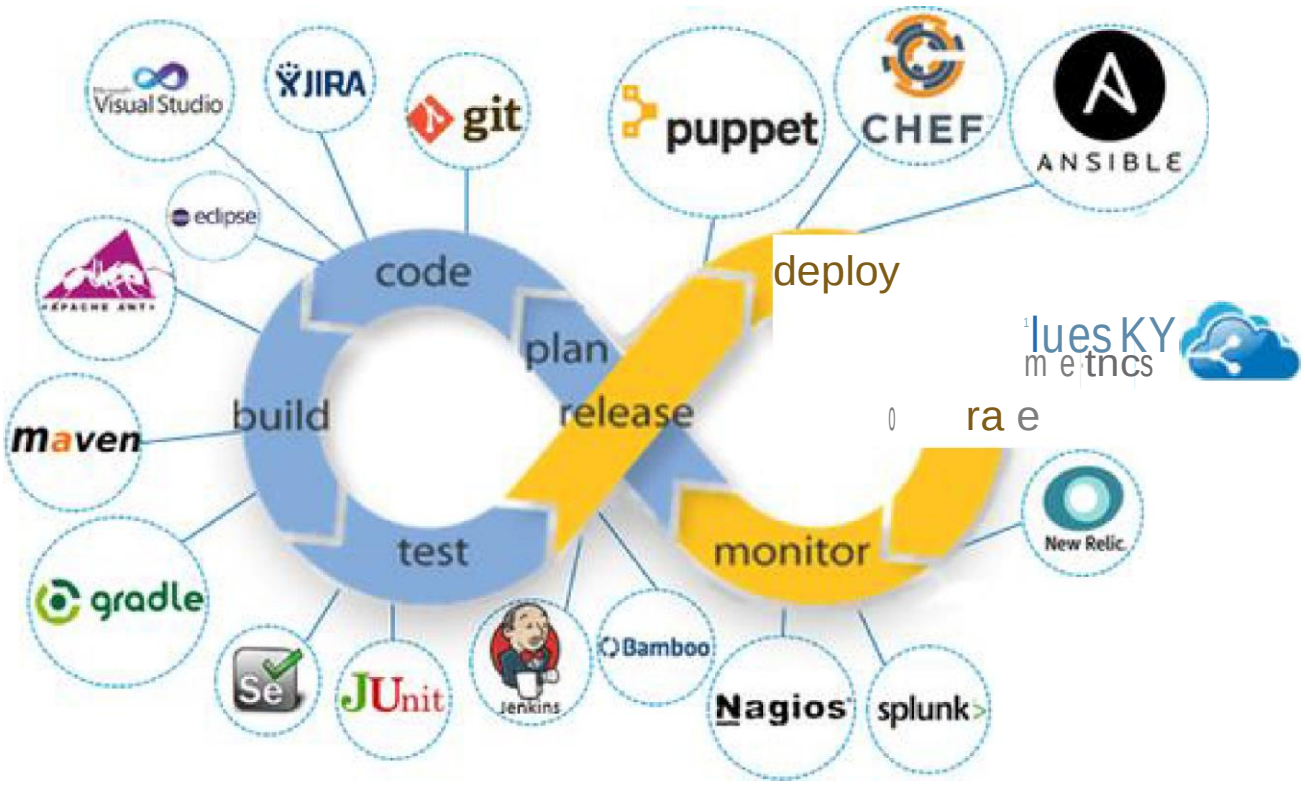
What is GitHub?

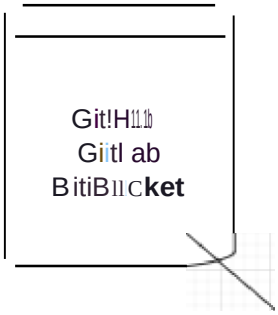
- It is a web based hosting service for version control using git

DevOps Engineer?

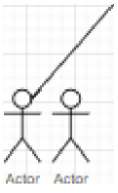
- "DevOps Engineer" is not a singular role or a type of developer or engineer, and doesn't even necessitate any particular technical skills
- Instead, a DevOps Engineer is a talented engineer or developer with a certain subset of business, technical, organizational and interpersonal skills

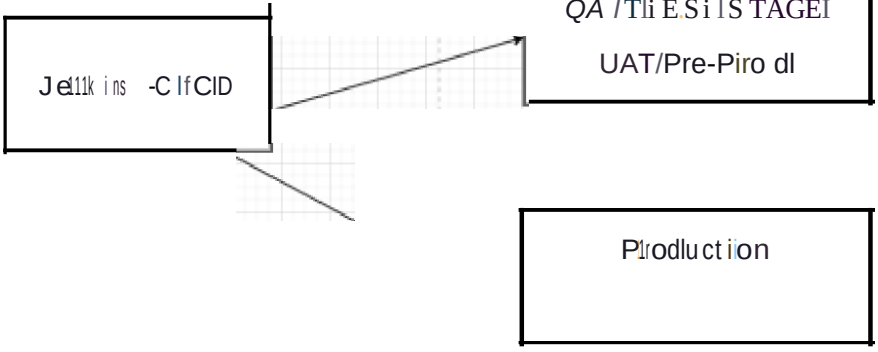



Visual Studio
JIRA
git
puppet
CHEF
ANSIBLE
eclipse
code
plan
build
release
deploy
maven
test
monitor
New Relic
gradle
Bamboo
JUnit
Jenkins
Nagios
splunk


Git!H111b
Gitl ab
BitiBuCket


Dev Environment


Actor
Actor


Jel11k ins -C lfCID
QA /Tli E.Si lS TAGEI
UAT/Pre-Piro dl
Prodluct ion

Jenkins

Introduction

Continuous Integration (CI)
Continuous Delivery (CD)
Continuous Deployment (CD)

Installation

In Linux Server

Create the Maven Project using Freestyle Project type

Integrate Maven software if not done.
Integrate Nexus with Jenkins Integrate
SonarQube with Jenkins Deploy the App
into Tomcat

1) Through “Deploy to container “plugin.
2) Through Script (If Tomcat and Jenkins are installed in the same Linux server)

Configure Email Functionality Poll
SCM
Build Periodically Git Web
Hooks Discard Old Build
Disable this project
Delete workspace before build starts Add
timestamps to the Console Output
JACOCO plugin

Jenkins Directory structure

Create the Maven Project using Maven Project type

Plugin Management

- Deploy to container
- Deploy WebLogic
- Maven Integration
- Safe Restart
- Next Build Number
- JACOCO
- SSH Agent
- Email Extension
- SonarQube Scanner
- Audit Trail Plugin
- Schedule Build
- Artifactory Plugin
- Cloud Foundry

- Blue Ocean
- Publish Over SSH
- ThinBackup
- Build Name Setter
- Convert To Pipeline

External Plugins Installation
Urban Code Deploy **Port**

Number Change Build with

parameters Create View

Jenkins Security

- Create Users (Default Admin)
- Provide the specific access Jenkins
- Provide access to specific projects/users

Create the Pipeline Project Jobs
http://localhost:8080/env-vars.html/

Create the Multibranch Pipeline Project Jobs

Create Master/Slave

Jenkins Backup

Jenkins Migration

Optional Topics

- Jenkins Home Directory Change in RHEL 7.5 Version
- Jenkins CLI
- Integrate the Urban Code Deploy server with Jenkins
- Deploy the App into IBM Cloud
- Slack integration

Introduction

Jenkins, is an open source Continuous Integration, cross-platform tool written in Java. Kohsuke Kawaguchi is the Creator of the Jenkins CI server in 2004. Initially, it was called Hudson, but in 2011 it was renamed to Jenkins because of disputes with Oracle.
The tool simplifies the process of integration of changes in to the project and delivery of fresh builds to users.

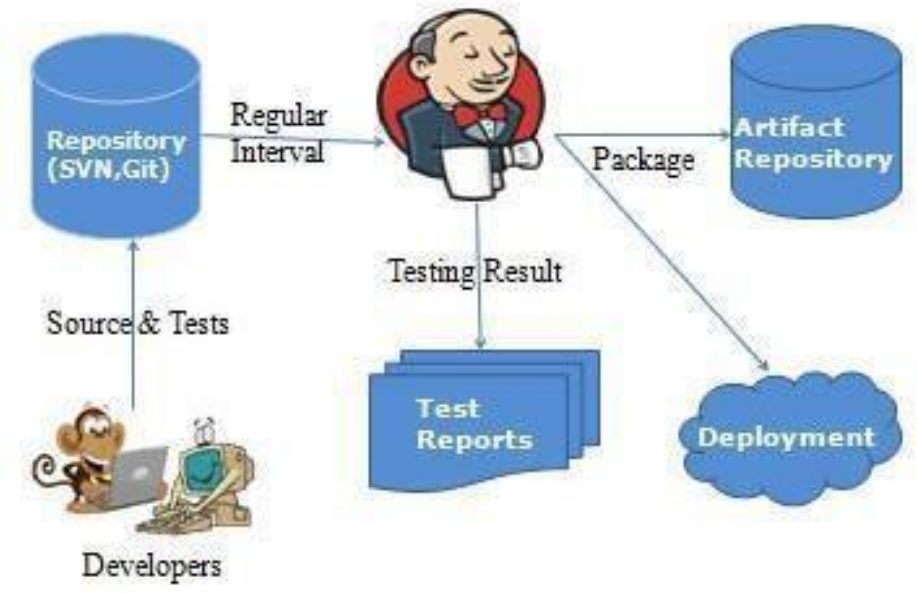
Continuous Integration: Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control.

(OR)

Continuous Integration is a development practice where developers integrate their code into a shared remote repository frequently, preferably several times a day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

CI Flow

Below diagram CI flow with Jenkins as Build tool.

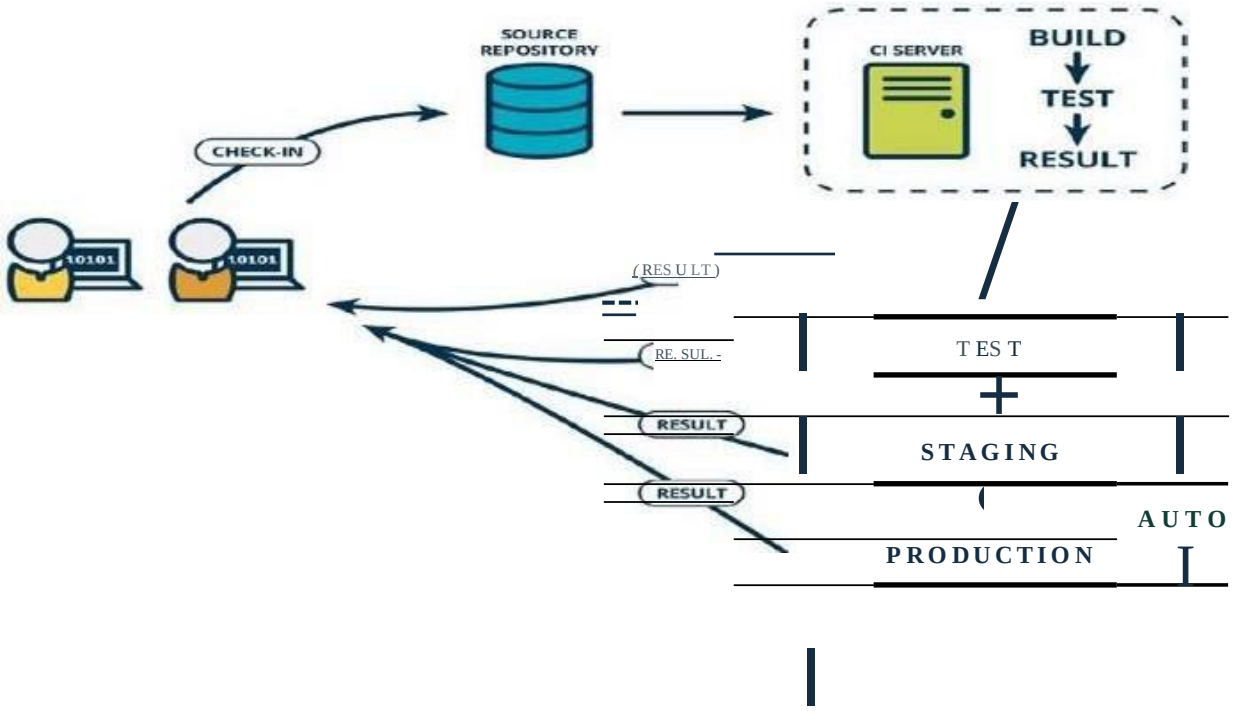


CI – Benefits

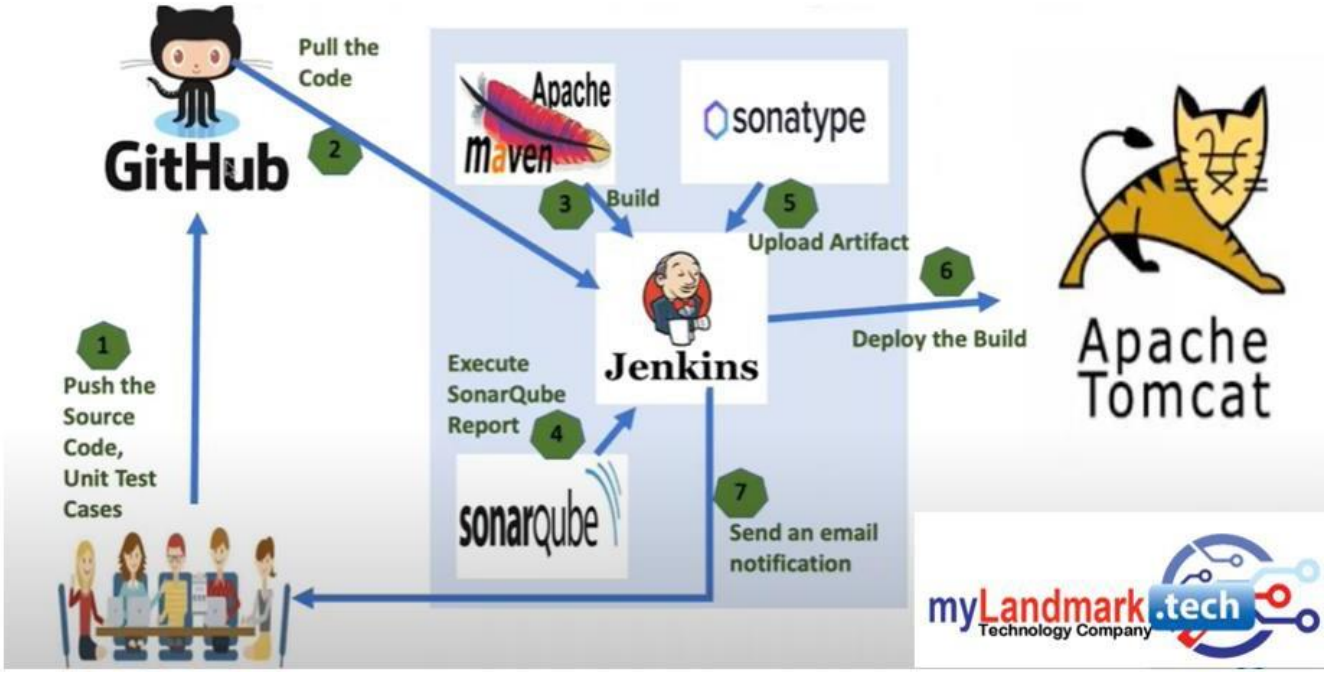
- Immediate bug detection
- No integration step in the Software Development lifecycle
- A deployable system at any given point
- Record of evolution of the project

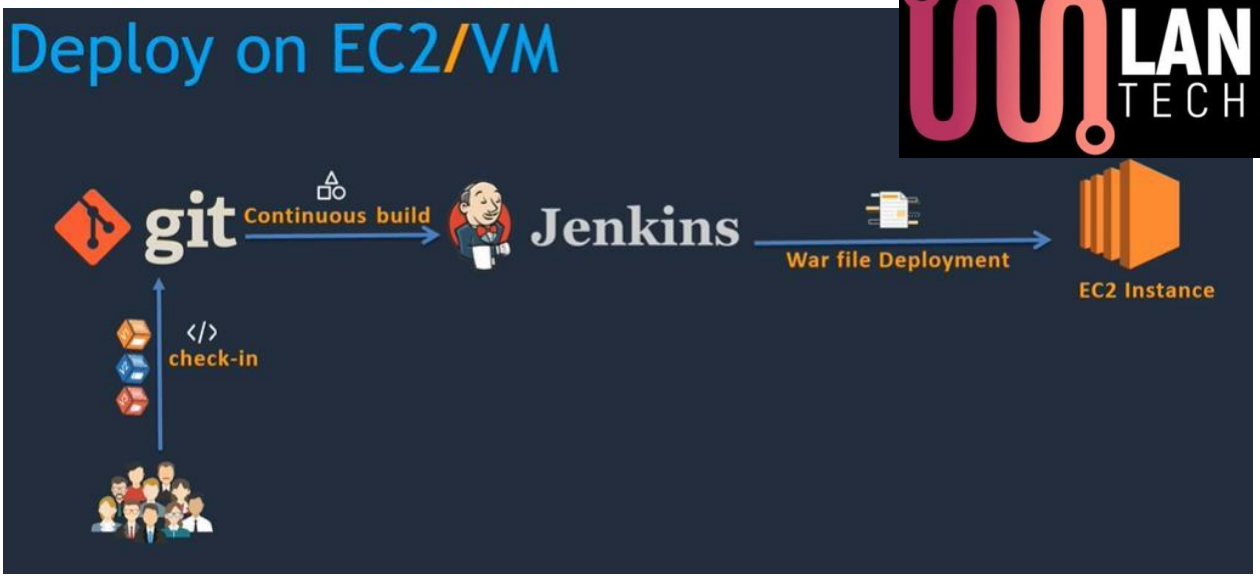
Continuous Delivery: Any and every successful build that has passed all the relevant automated tests and quality gates can potentially be deployed in to production via fully automated one click process.

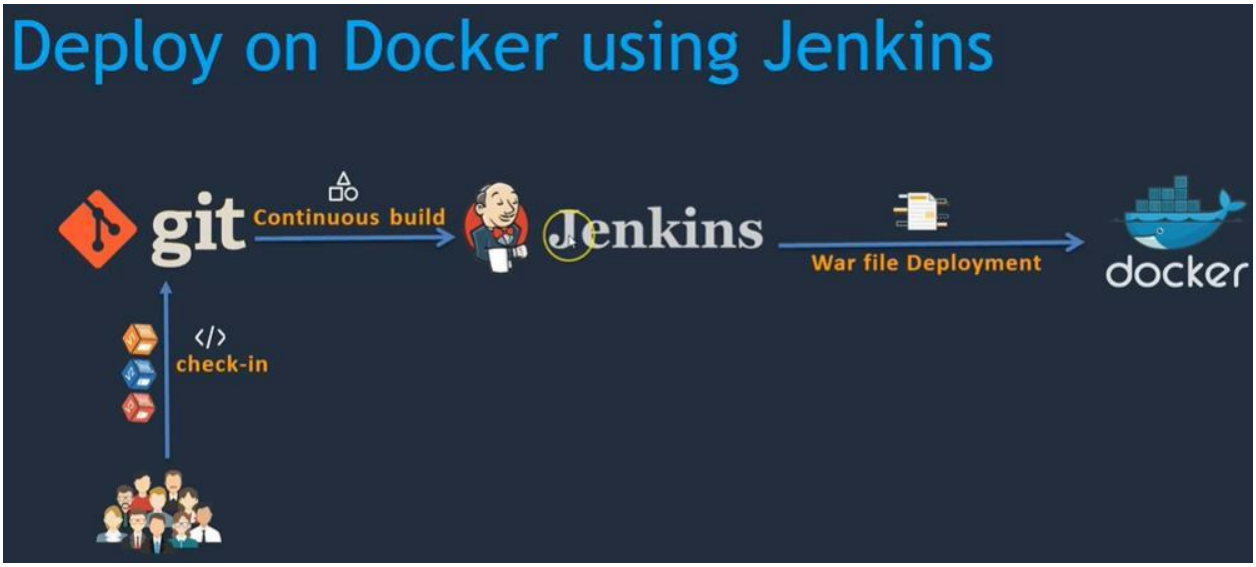
Continuous Deployment: The practicing of automatically deploying every successful build directly into production without any manual steps is known as Continuous deployment.
(OR)
It is closely related to Continuous Integration and refers to keeping your application deployable at any point or even automatically releasing to a test or production environment if the latest version passes all automated tests.

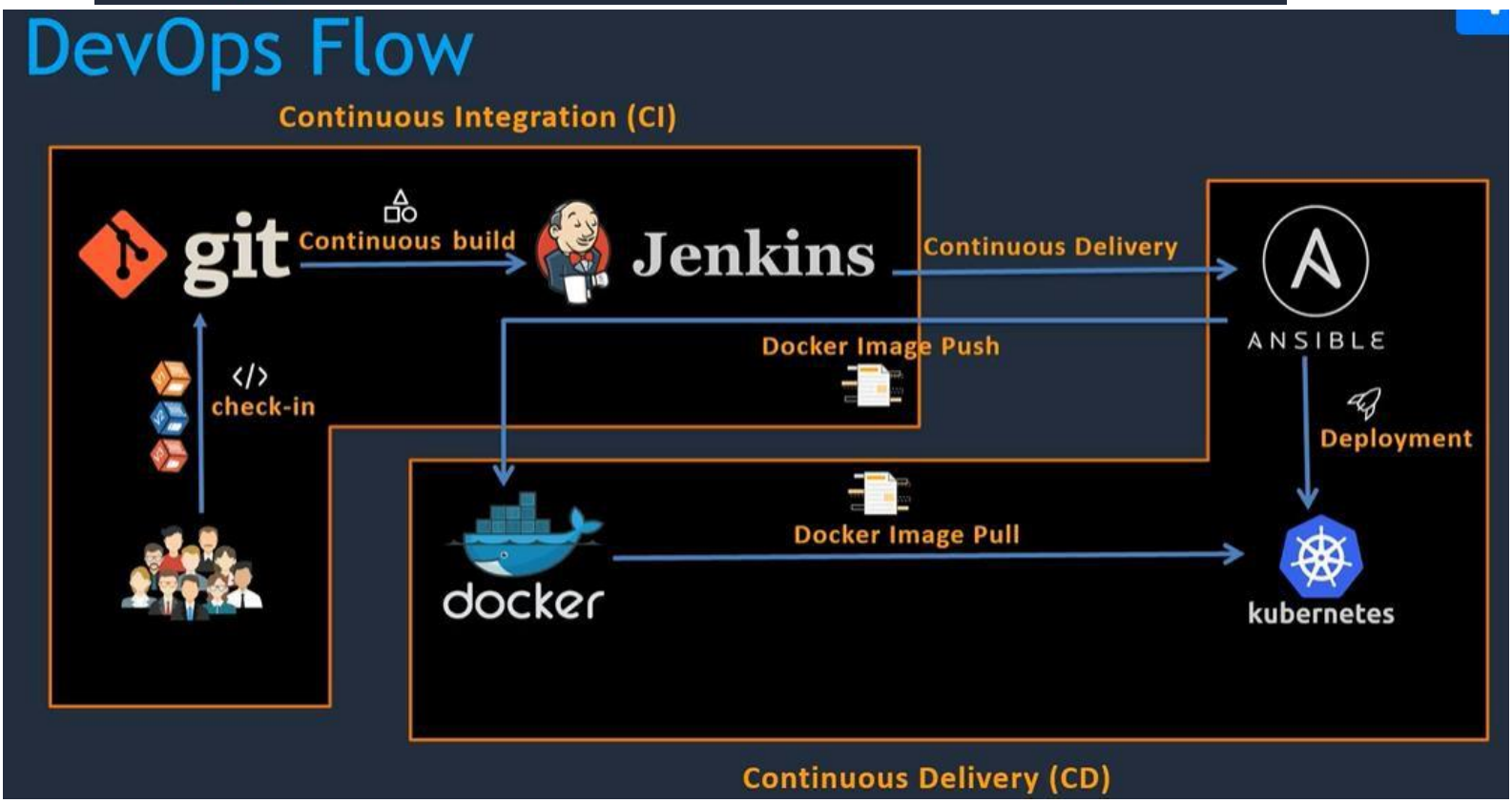


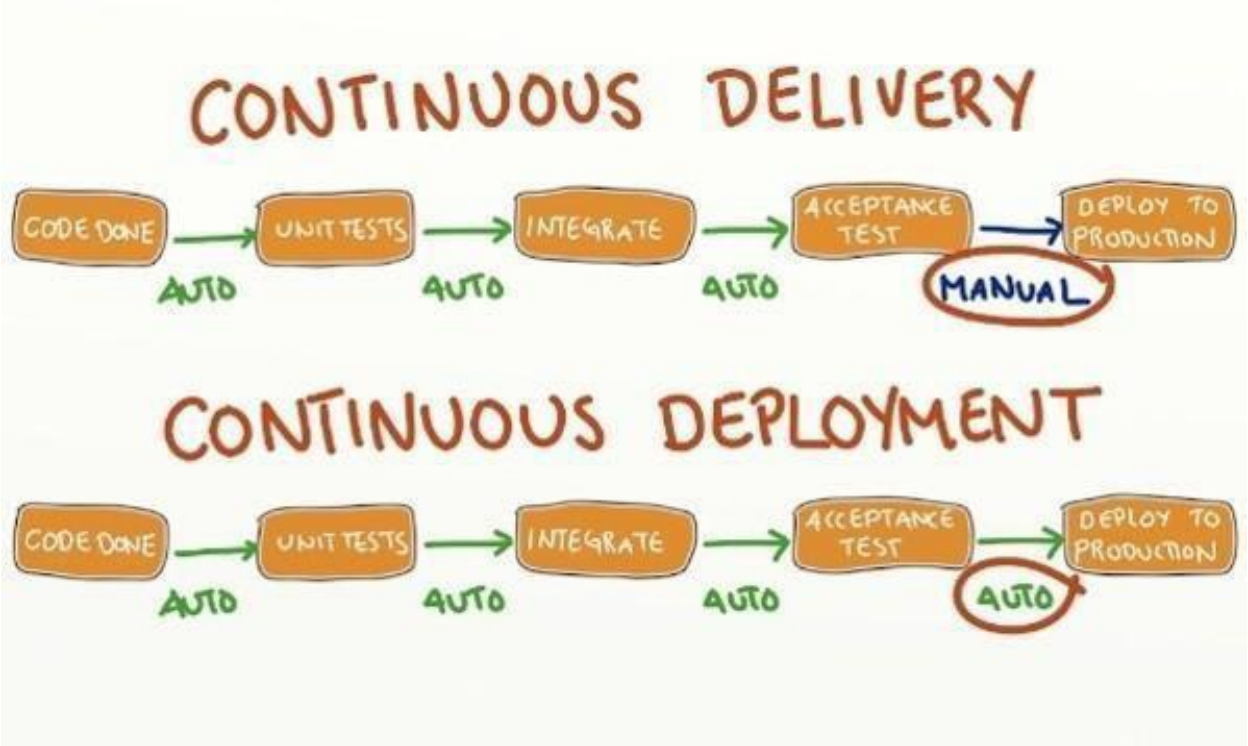
DevOps Project 1: www.mylandmark.tech **Tel: + 1 437 215 2483**











What Jenkins can do?

- Integrate with many different Version Control Systems (GitHub, CVS, SVN, TFS…)
- Generate test reports（JUnit), using JaCoCo plugins (Java Code Coverage).
- It can stop the deployment if code coverage is LESS than 80% or defined threshold.
- Push the builds to various artifact repositories. JFrog, Nexus.
- Deploys directly to target environments; production, stage, or test environments.
- Notify stakeholders of build status (Through Email)
- Bug tracking with Jira integration.

Benefits of Jenkins

- ✓ It's an open-source tool with great community support. If you facing issues with Jenkins, you can use Jira create a ticket and the community will help fixed it.
- ✓ Easy to install and It has a simple configuration through a web-based GUI, which speeds up the Job
- ✓ It has around 1000+ plugins to ease your work. If a plugin does not exist, just code it up and share with the community (https://plugins.jenkins.io/). LSS can develop a plugin and share with the community.
- ✓ Its built with Java and hence, it is portable on all major platforms.
- ✓ Good documentation and enriched support articles/information available on internet which will help beginners to start easy.
- ✓ Specifically, for a test only project, it is used to schedule jobs for regression testing

without manual intervention and hence monitor infrastructural and functional health of a application. It can be used like a scheduler for integration testing and also can be used to validate new deployments/environments on a single click on a Build now button.

The diagram below depicts that Jenkins is integrating various DevOps stages:

TBD (ToBeDocument)

List of popular Continuous Integration tools

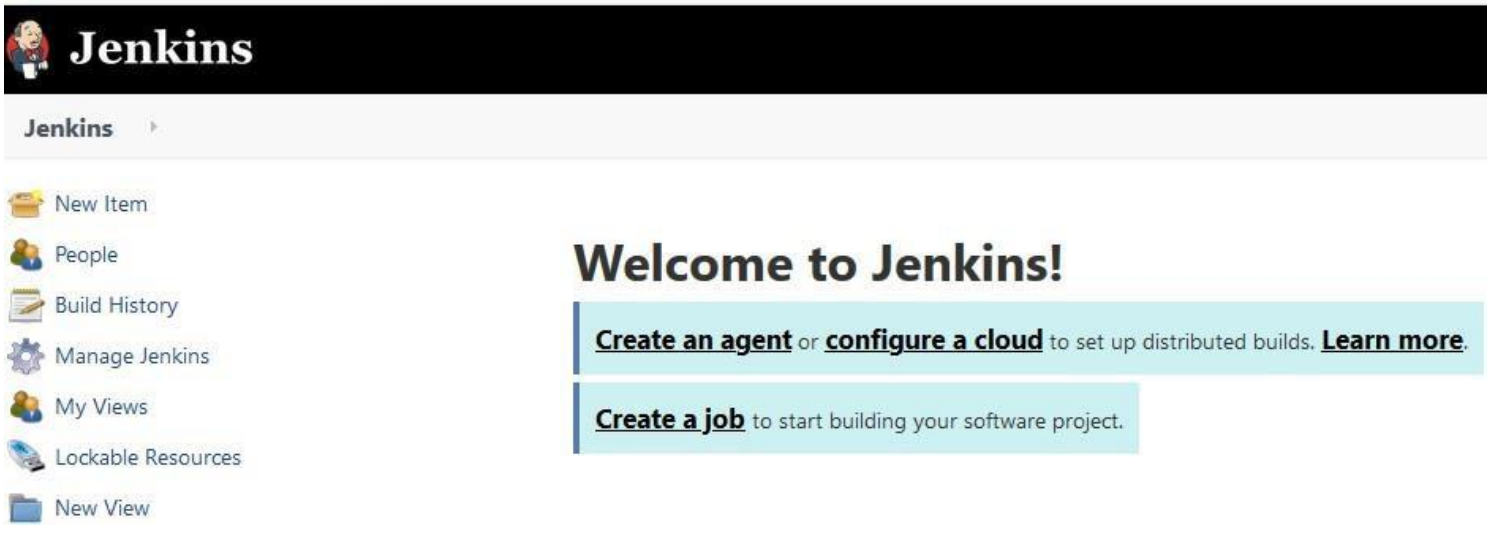
SNo	Product	Is Open Source?
1	Jenkins	Yes
2	Cloudbees Jenkins	No
2	Bamboo	No
3	Cruise Control	Yes
4	Travis CI	Yes and Paid also
5	Circle CI	Yes and Paid also
6	GitLab CI	Yes and Paid
7	TeamCity	Yes and Paid

Jenkins Installation

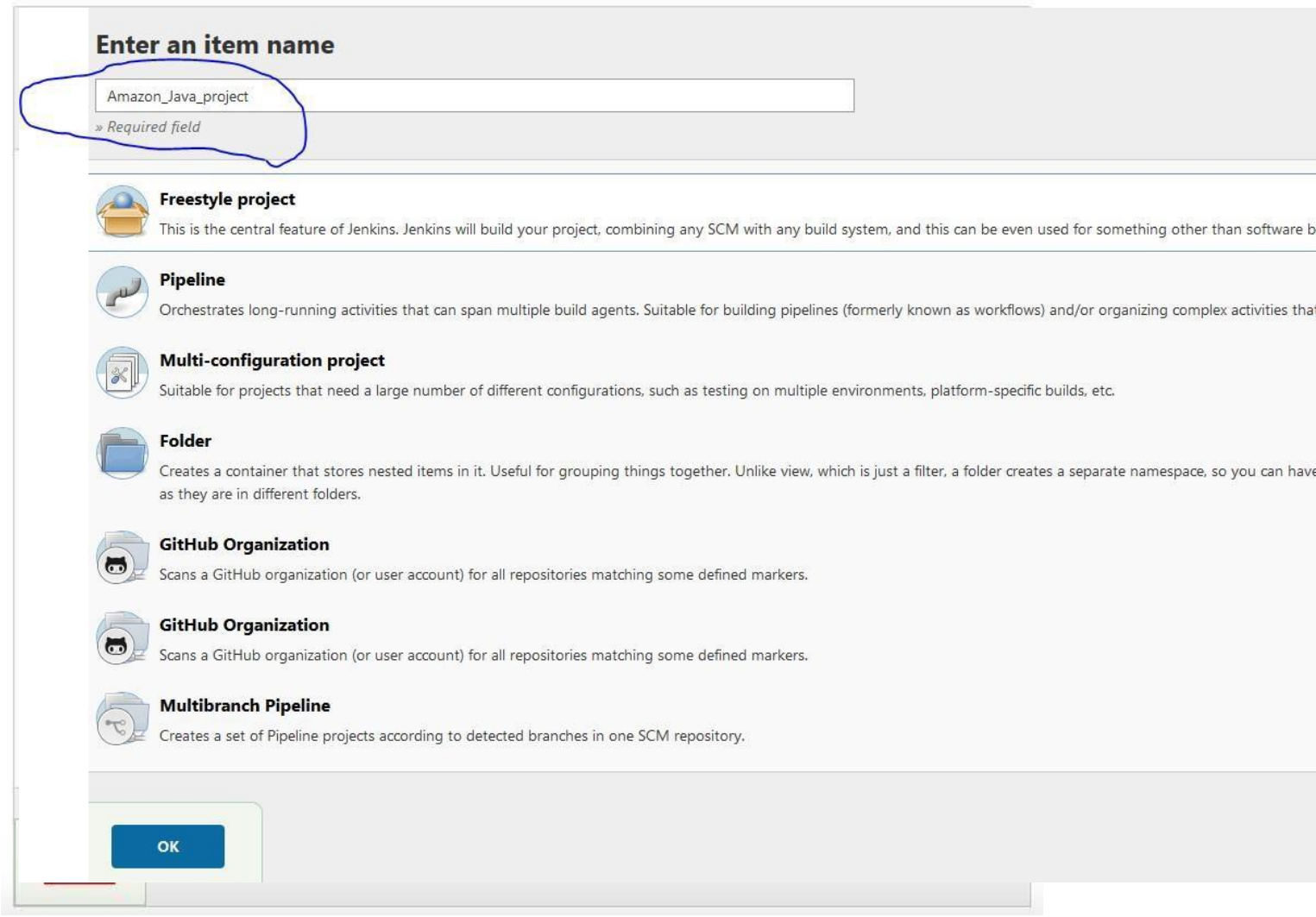
- Jenkins is java based CI tool, so we need to install jdk/jre before installing Jenkins.
- **Pre-Requisite Software:** Java (Check weather java is installed or not with java -version command)

Create the project/job in Jenkins

Step 1: Login into the Jenkins, go to the Jenkins dashboard left side top corner, click on **New Item**.



Step 2: Enter the project name in **Enter an item name** input box and select the **Freestyle project**
and click on **OK** Button.



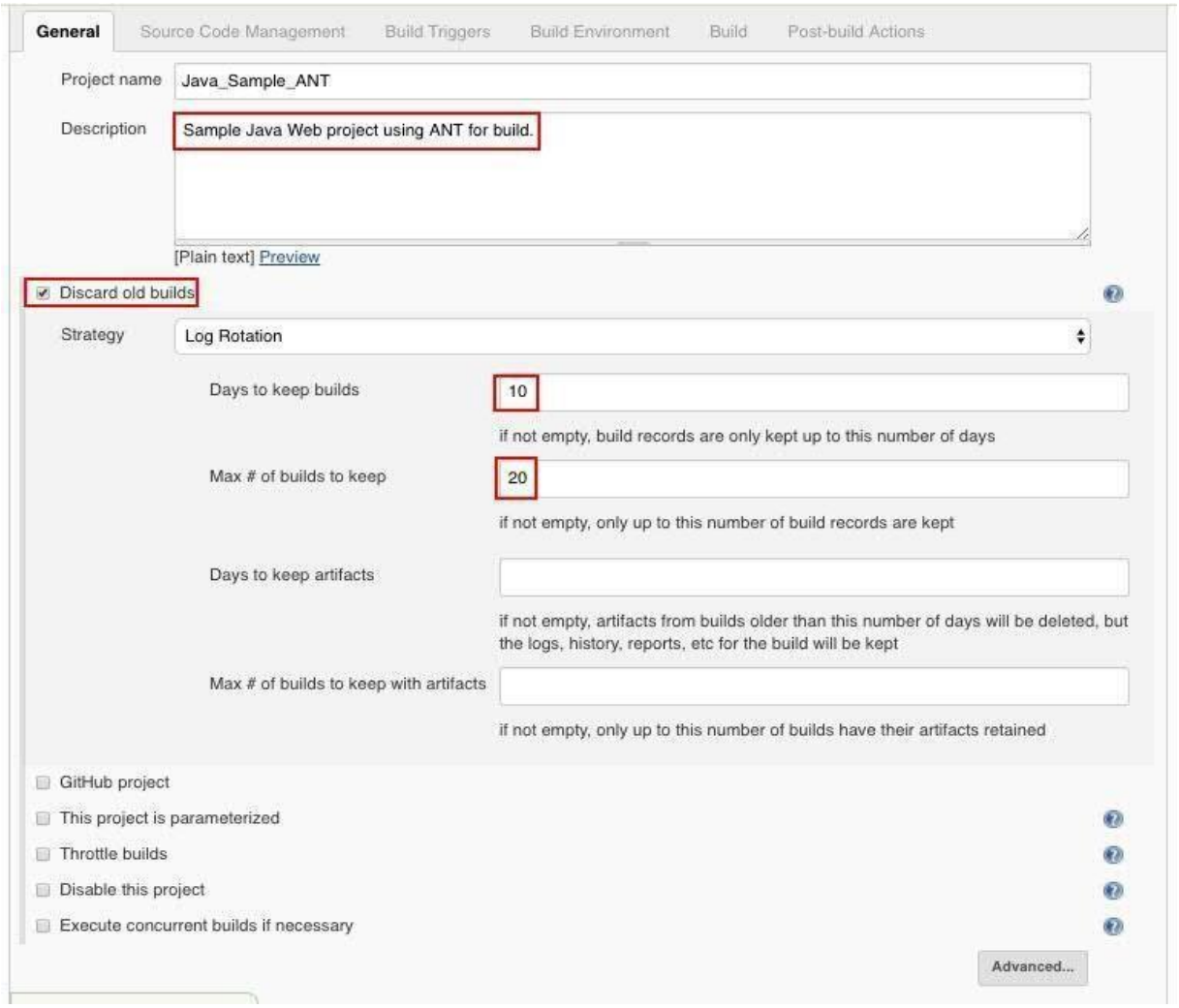
Freestyle project: This is the central feature of Jenkins. Jenkins will build your project combining any SCM and any build system.

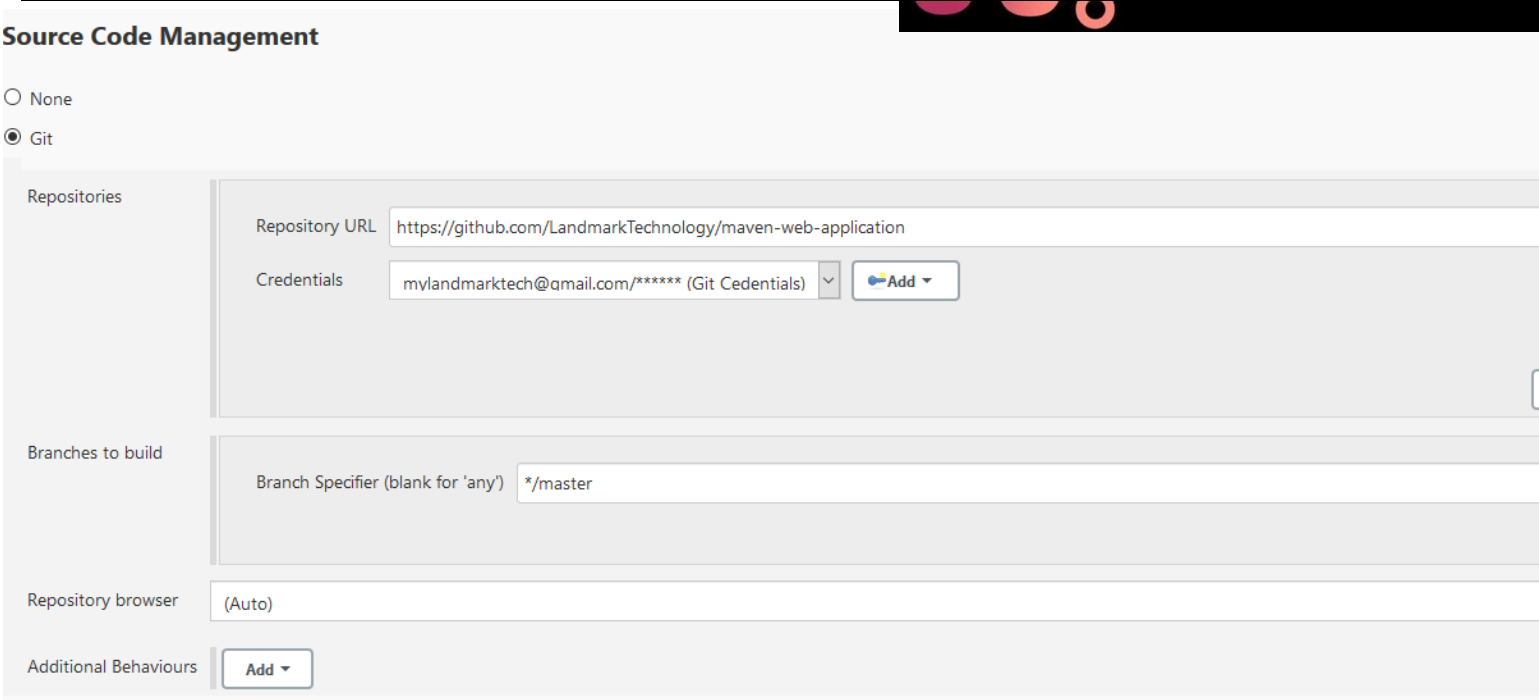
A Free-Style project is a project that can incorporate almost any type of build. The Free-Style project is the more "generic" form of a project. You can execute shell/dos scripts, invoke ant/maven, and a lot more. Majority of the plugins are written to use the free-style project.

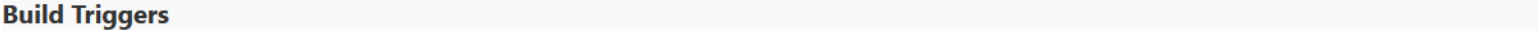
Maven project: A maven project is a project that will analyze the pom.xml file in greater detail and produce a project that's geared towards the targets that are invoked. The maven project is smart enough to incorporate build targets like the javadoc or test targets and automatically setup the reports for those targets.

Multi-configuration project: The “multiconfiguration project” (also referred to as a “matrix project”) permit you to run the same build job in many different configurations. This powerful feature can be useful for testing an application in many different environments, with different databases, or even on different build machines. We will be looking at how to configure multiconfiguration build jobs later on in the book.

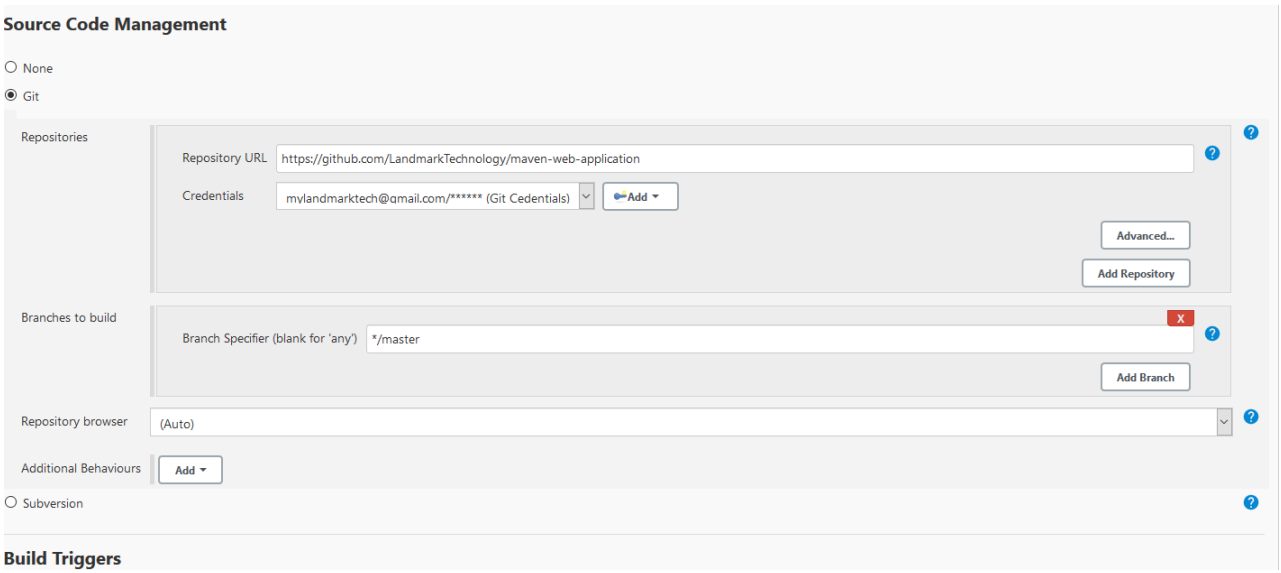
Monitor an external job: The “Monitor an external job” build job lets you keep an eye on non- interactive processes, such as cron jobs.





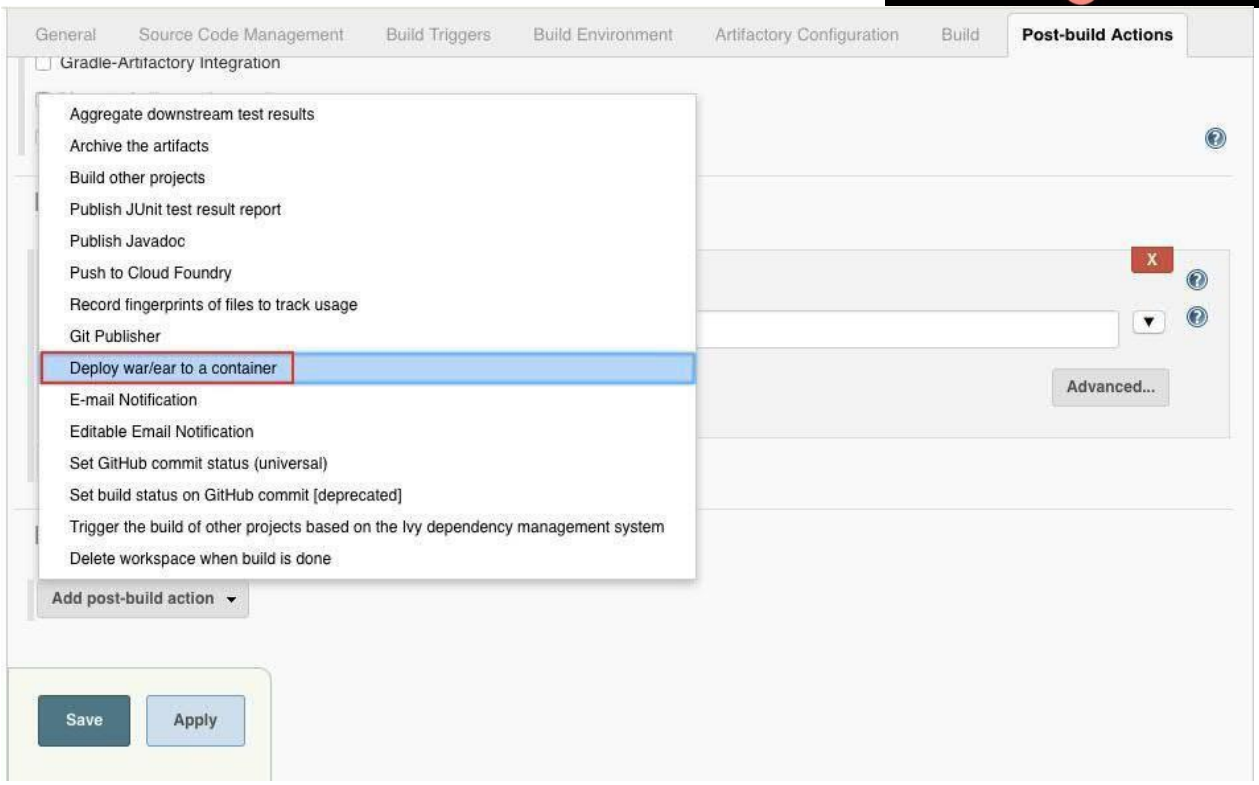


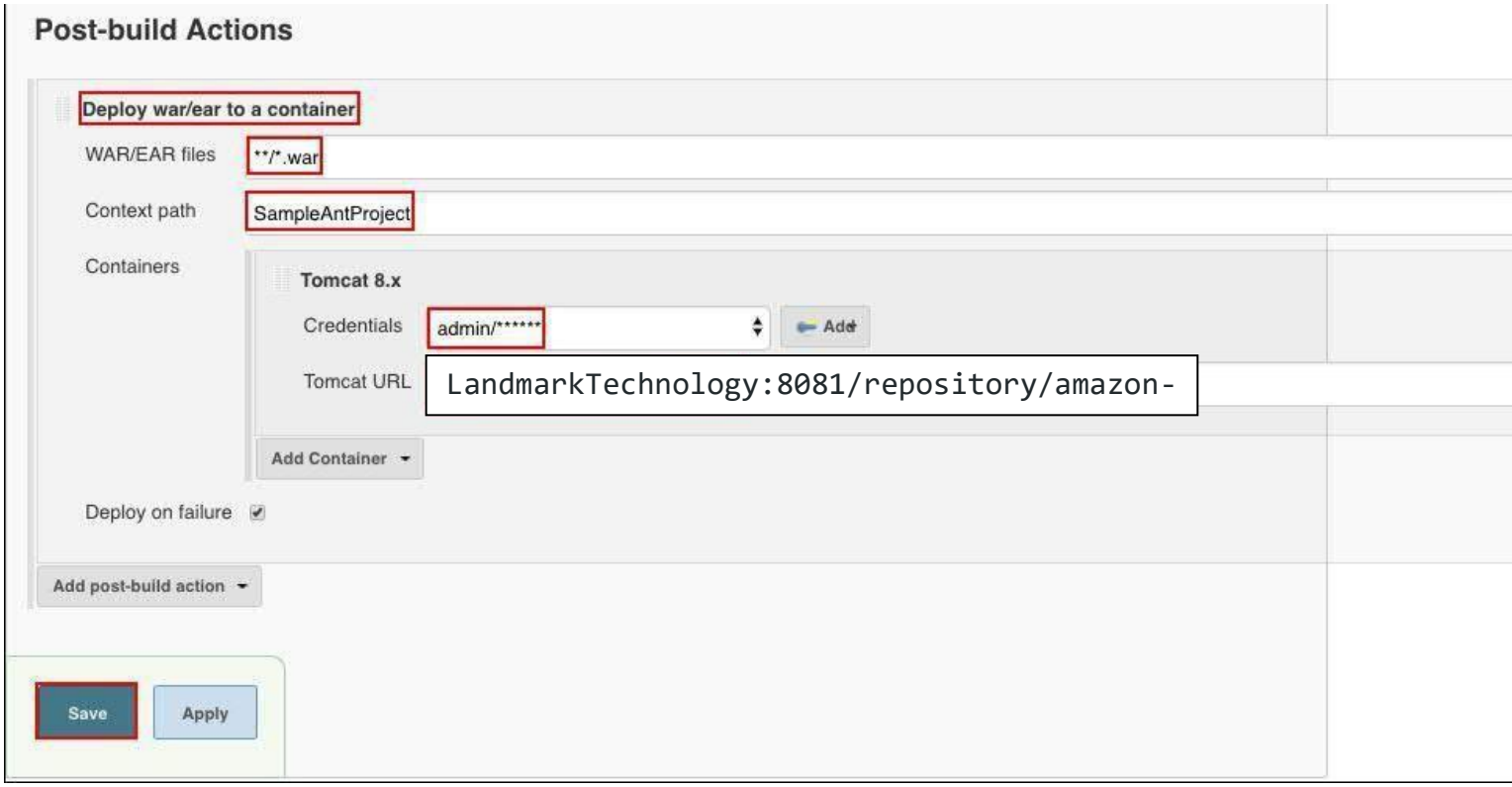
Specify when and how your build should be triggered. The following example polls the Git repository every 5 min. It triggers a build, if something has changed in the repo.



Deploy the application into Tomcat

Install the **"Deploy to container"** plugin.
Open the job which you want to configure deploy, and click on Configure and in **Post-build actions** tab, click on **ADD POST-BUILD ACTION** and select the **Deploy war/ear to container** as follows.


General
Source Code Management
Build Triggers
Build Environment
Artifactory Configuration
Build
Post-build Actions
Gradle-Artifactory Integration
Aggregate downstream test results
Archive the artifacts
Build other projects
Publish JUnit test result report
Publish Javadoc
Push to Cloud Foundry
Record fingerprints of files to track usage
Git Publisher
Deploy war/ear to a container
E-mail Notification
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Trigger the build of other projects based on the Ivy dependency management system
Delete workspace when build is done
Advanced...
Add post-build action
Save
Apply


Post-build Actions
Deploy war/ear to a container
WAR/EAR files
**/*.war
Context path
SampleAntProject
Containers
Tomcat 8.x
Credentials
admin/******
Add
Tomcat URL
LandmarkTechnology:8081/repository/amazon-
Add Container
Deploy on failure
Add post-build action
Save
Apply

Error:

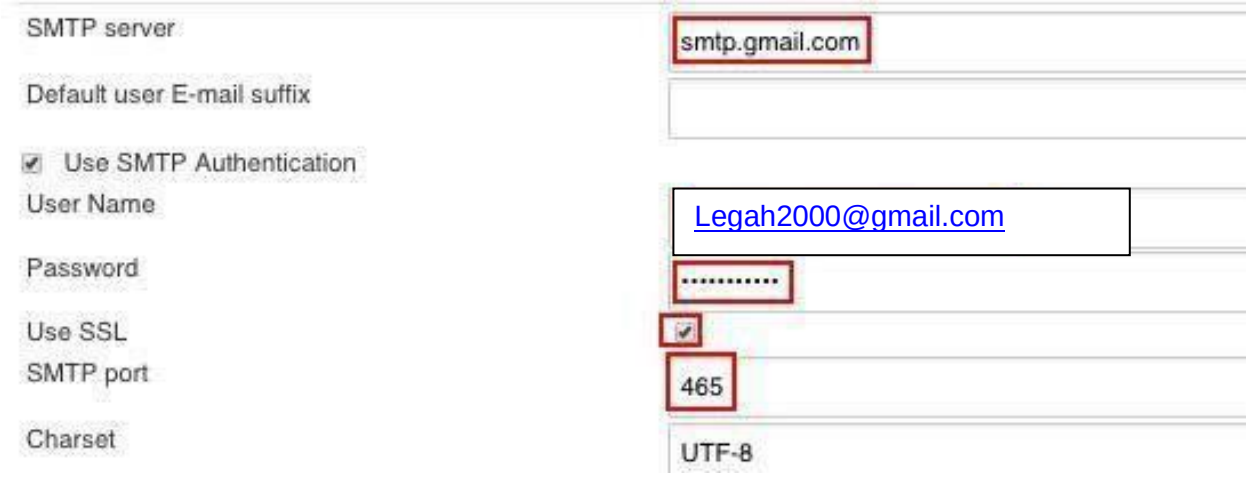
```
Caused by: org.codehaus.cargo.container.tomcat.internal.TomcatManagerException: The username you provided is
not allowed to use the text-based Tomcat Manager (error 403)
        at org.codehaus.cargo.container.tomcat.internal.TomcatManager.invoke(TomcatManager.java:704)
        at org.codehaus.cargo.container.tomcat.internal.TomcatManager.list(TomcatManager.java:876)
        at org.codehaus.cargo.container.tomcat.internal.TomcatManager.getStatus(TomcatManager.java:889)
        at
org.codehaus.cargo.container.tomcat.internal.AbstractTomcatManagerDeployer.redeploy(AbstractTomcatManagerDeplo
yer.java:173)
        ... 17 more
Caused by: java.io.IOException: Server returned HTTP response code: 403 for URL:
http://localhost:8085/manager/text/list
        at sun.net.www.protocol.http.HttpURLConnection.getInputStream0(HttpURLConnection.java:1894)
        at sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1492)
        at org.codehaus.cargo.container.tomcat.internal.TomcatManager.invoke(TomcatManager.java:571)
        ... 20 more
```

Solution: Need to add rule in tomcat-users.xml file as follows.

<user username="admin" password="passw0rd" roles="admin-gui,manager-gui,**manager-script**"/>

Enable email notification

Step 1) Install Email Extension Plugin as follows.
Manage Jenkins ---> Manage Plugins ---> Install **"Email Extension Plugin "**
Step 2) Add the smtp server host as follows.
Click on Manage Jenkins ---> Configure System --->



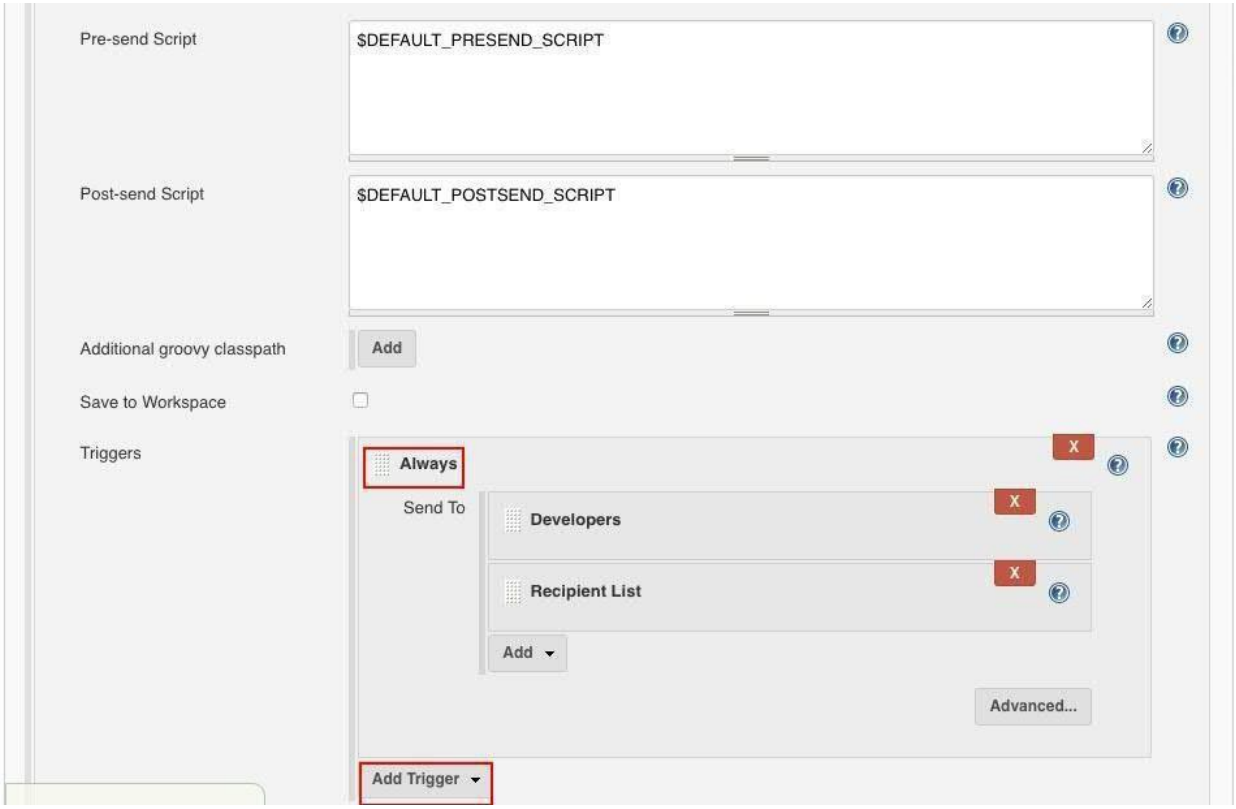
Step 3: In Job configure Editable Email as follows.

Select any Job, which we need to configure Email notification ---> Click on Configure ---> Select the

Post-build Actions section.

Click on Advanced Settings …

It will expand and will show more settings and click on **Add Trigger** and select the **Always**.



We can enable to attach the build logs while sending mail, as follows.

Output mail is like below.

We can enable to Compress and Attach Build Log to email as follows.

Output mail is like below.

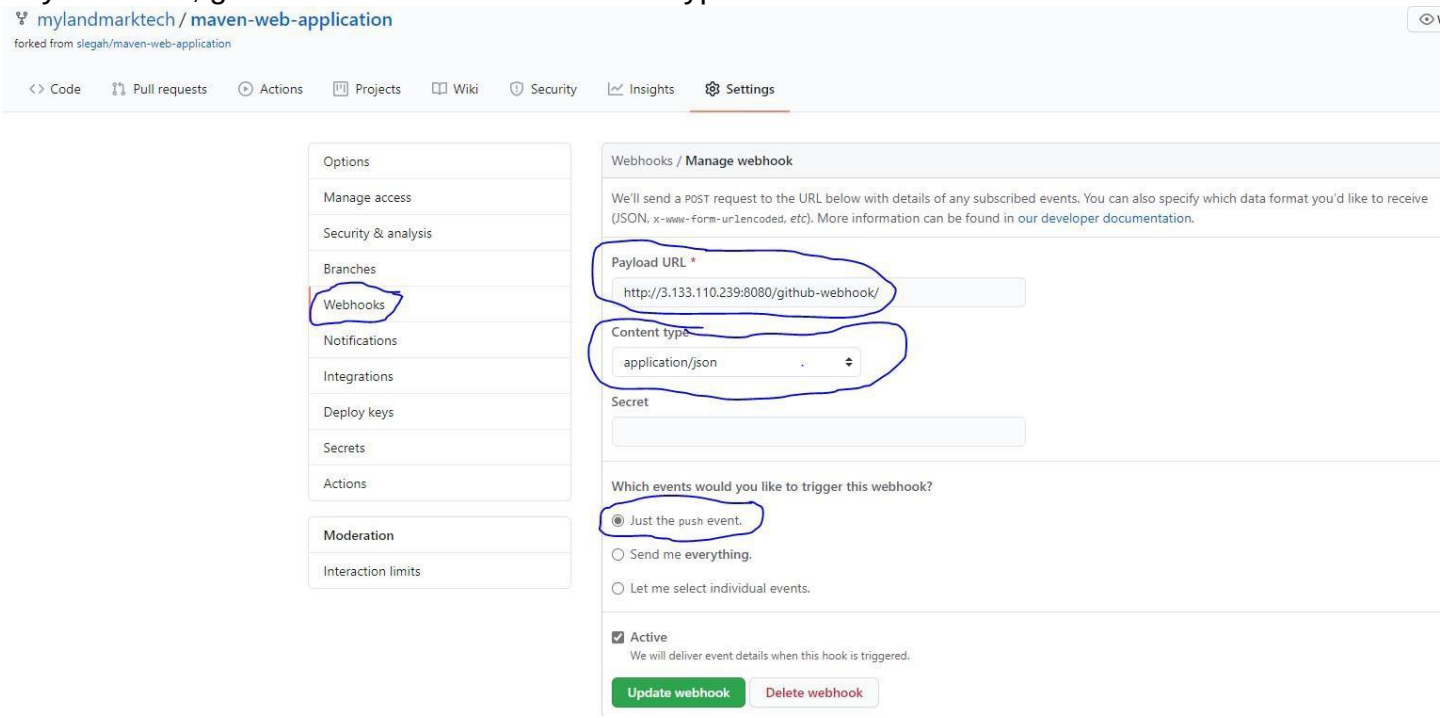
How to enable the Poll SCM in Jenkins?

Step 1: Install the "**Git plugin**" in Jenkins.

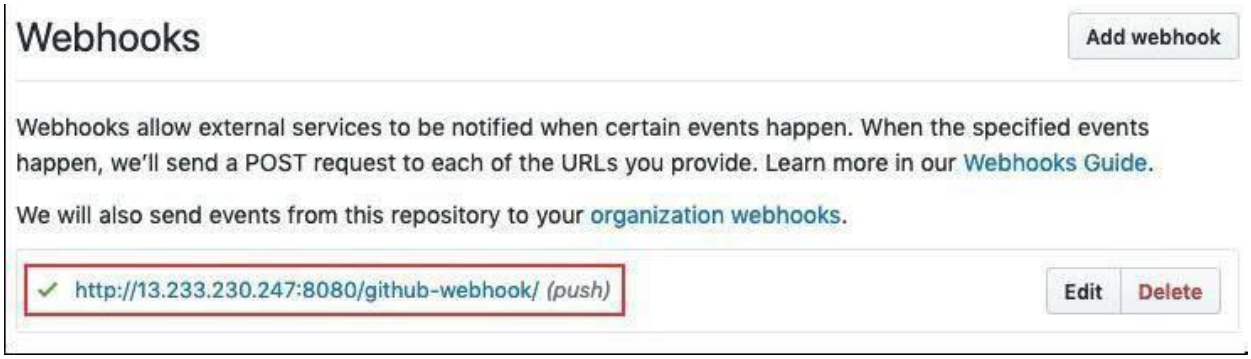
Step 2: Select the job which you need to enable hook and click on Configure ---> In **Build Triggers**
Section enable the **Poll SCM**
And provide the values as follows.

GitHub webhook

Settings --> Webhooks --> Add webhook , Once you click on Add webhook url, it will ask the Payload URL, give the Jenkins url and Content type as follows.



Once you have configured successfully, you will see as follows.



To restart Jenkins manually, you can use either of the following URLs:

(jenkins_url)/safeRestart - Allows all running jobs to complete. New jobs will remain in the queue to run after the restart is complete.
Ex: http://13.233.230.247:8080/safeRestart
(jenkins_url)/restart - Forces a restart without waiting for builds to complete.
Ex: http://13.233.230.247:8080/restart

(OR)

You can install one plug called **SafeRestart**, once installed it will give one option Jenkins dashboard as follows.

Disable Build:
A disabled Build will not be executed until you enable it again. This option often comes in handly to suspend a build during maintenance work or major refactoring.

Once the project is configured in Jenkins then all future builds are automated. It has basic reporting features like status and weather reports (job health).

Jenkins Directory Structure

jenkins : This is the default Jenkins home directory (may be .hudson in older installations) and it will be placed in user's home directory (C:\Users\LANDMARK_ADMIN\ ---> Windows & /Users/simonlegah/ --> MAC and /var/lib/jenkins ➐ Linux).

Jenkins home directory contains the below sub directories and configuration files (.xml).

+- jobs
+- [JOBNAME] :Sub directory for each job
+- config.xml : Job configuration file
+- latest : Symbolic link to the last successful build)
+- builds
+- [BUILD_ID] : for each build one build id
+- build.xml : build result summary
+- log : log file
+- changelog.xml (change log)

+- logs ()
+- nodes ()
+- plugins : This directory contains all the plugins that you have installed.
+- secrets ()
+- updates : This is an internal directory used by Jenkins to store information about available plugin updates.

+- userContent : You can use this directory to place your own custom content onto your Jenkins server. You can access files in this directory at http://localhost/jenkins/userContent (if you are running Jenkins on an application server) or http://localhost:8080/userContent (if you are running in stand-alone mode).

+- users : If you are using the native Jenkins user database, user accounts will be stored in this directory.

+- war : This directory contains the expanded web application. When you start Jenkins as a stand-alone application, it will extract the web application into this directory.

+- config.xml (jenkins root configuration)
+- *.xml (other site-wide configuration files)
+- fingerprints (stores fingerprint records)
+-workspace: This directory contains all jobs source code.

http://localhost:8080/configure

Home directory: By default, Jenkins stores all of its data in this directory on the file system. Under the Advanced section, you can choose to store build workspaces and build records elsewhere.

There are a few ways to change the Jenkins home directory:

- Edit the JENKINS_HOME variable in your Jenkins configuration file (e.g. /etc/sysconfig/jenkins on Red Hat Linux).
- Use your web container's admin tool to set the JENKINS_HOME environment variable.

- Set the environment variable JENKINS_HOME before launching your web container, or before launching Jenkins directly from the WAR file.
- Set the JENKINS_HOME Java system property when launching your web container, or when launching Jenkins directly from the WAR file.
- Modify web.xml in jenkins.war (or its expanded image in your web container). This is not recommended.

This value cannot be changed while Jenkins is running.
It is shown here to help you ensure that your configuration is taking effect.

Ex: /Users/SimonLegah/.jenkins is for my Jenkins which is installed in my local MAC.

Workspace Root Directory: Specifies where Jenkins will store workspaces for builds that are executed on the master.

Build Record Root Directory: Specifies where Jenkins will store build records on the filesystem. This includes the console output and other metadata generated by a build.

System Message: This message will be displayed at the top of the Jenkins main page.

# of executors: It shows how many builds run at a time. E.g.: If you give 2 here, then two builds are running.

Labels:

Usage: Controls how Jenkins schedules builds on this node.

Quiet period:

SCM checkout retry count:

Restrict project naming:

Naming Strategy

Strategy

Default ---> This is the default configuration and allows the user to choose any name they like.

Pattern-- > Define a pattern (regular expression) to check whether the job name is valid or not. Forcing the check on existing jobs, will allow you to enforce a naming convention on existing jobs - e.g. even if the user does not change the name, it will be validated with the given pattern at every submit and no updates can be made until the name confirms.

This option does not affect the execution of jobs with non-compliant names. It just controls the validation process when saving job configurations.

Global properties

Environment variables

Tool Locations
SonarQube servers
etc….

To Install any Jenkins Plugin, follow below steps
Manage Jenkins ---> Manage Plugins ---> Select the Plugin name (HTML Publishedplugin)
---------- >
Install Without Restart

Plugin Management

- Safe Restart
- Next Build Number
- Email Extension
- SonarQube Scanner
- Maven Integration
- Schedule Build
- Artifactory Plugin
- Cloud Foundry
- Blue Ocean
- Deploy to container
- Maven Integration
- JACOC
- SSH Agent
- Publish Over SSH
- ThinBackup
- Build Name Setter
- Convert To Pipeline
- **JobConfigHistory:** This plugin saves a copy of the configuration file of a job (config.xml) for every change made and of the system configuration. You can also see what changes have been made by which user if you configured a security policy.
- Repository browser
- Role-based Authorization Strategy:
- Slack Notification Plugin:
- Cobertura Plugin: In UI we will see as Coverage Trend.
- Hudson global-build-stats plugin:
- Delivery Pipeline View:
- Enable project-based security

Port number change for Jenkins

By default, 8080 is the default port, change from 8080 something like 8082 as

follow. In Ubuntu update the below file.

#vi /etc/default/jenkins

then restart the service with below command.
service jenkins restart

In RHEL/CentOS update the below file.#vi /etc/sysconfig/jenkins

```
## Type:        integer(0:65535)
## Default:     8080
## ServiceRestart: jenkins
#
# Port Jenkins is listening on.
# Set to -1 to disable
#
JENKINS_PORT="8080"
```

Once you change the port, restart the jenkins service by using below command.
#service jenkins restart

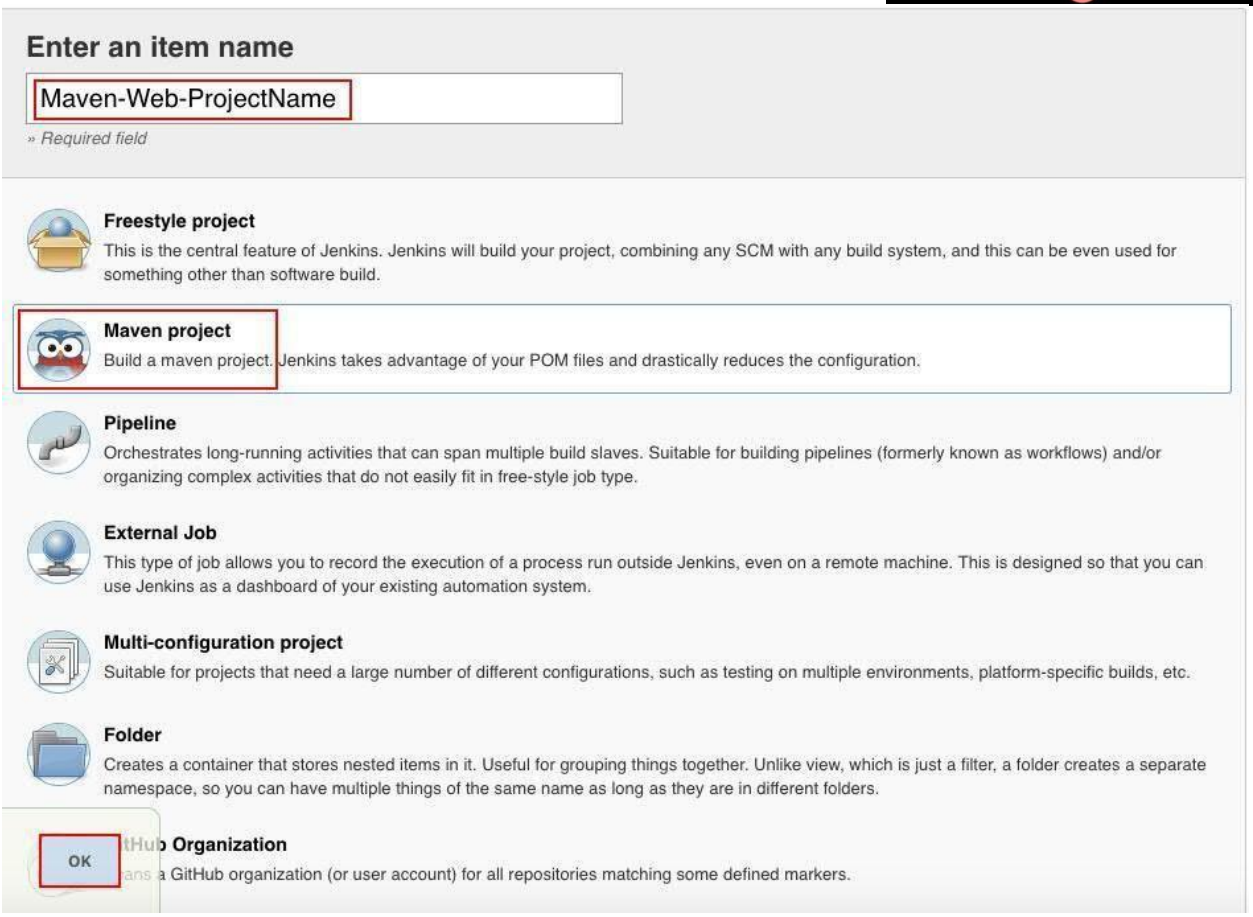
Create the Maven project/job in Jenkins

Method 1:

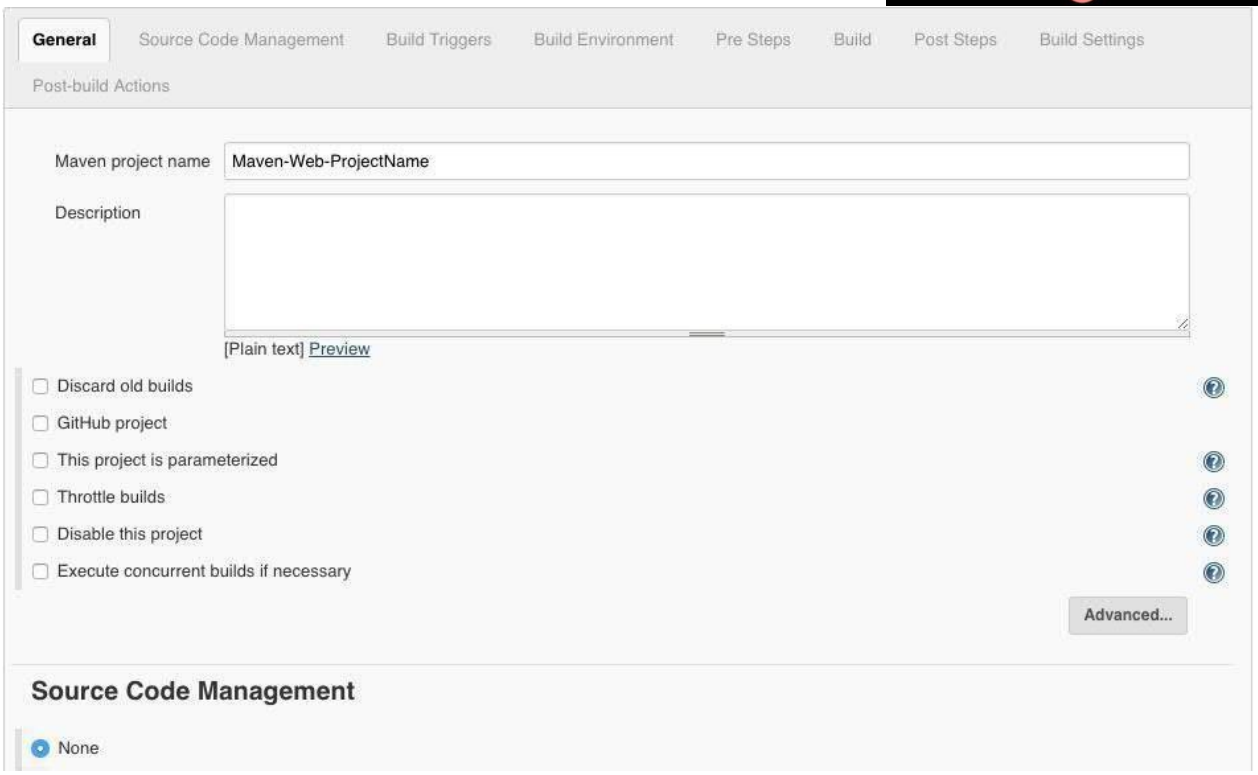
Install the **Maven Integration Plugin** and follow the below steps.

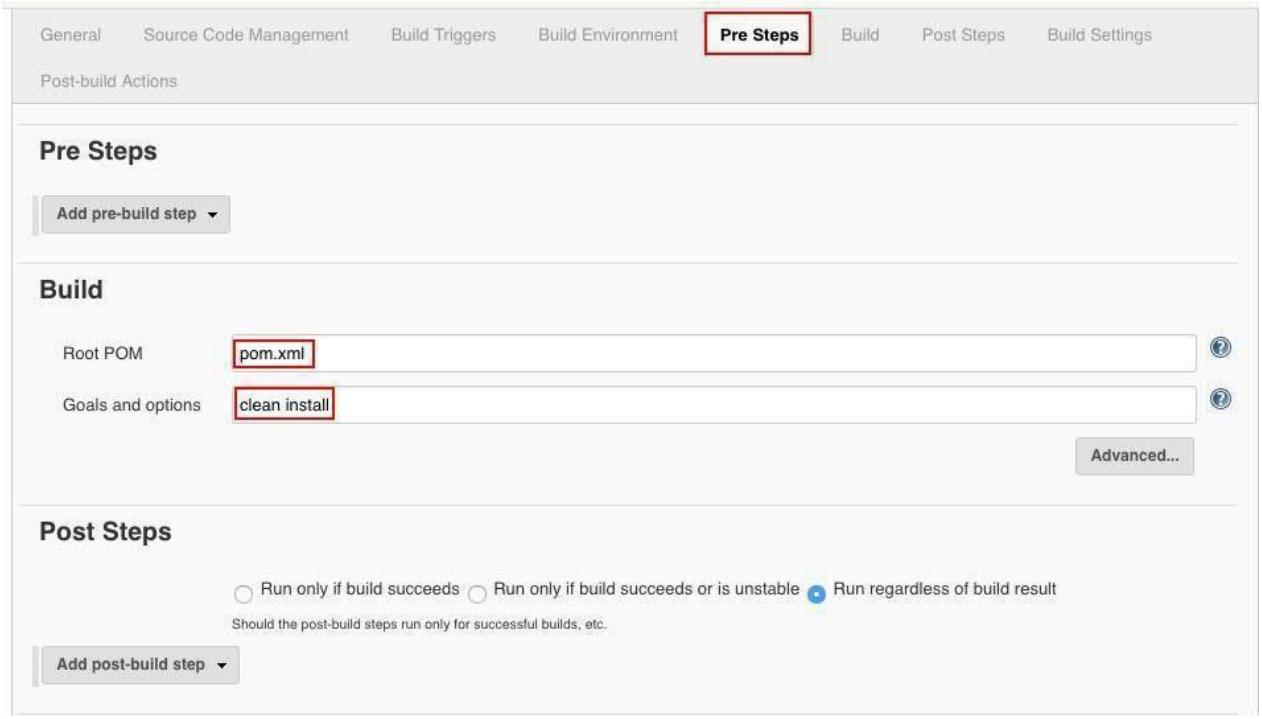
Create the Job using Freestyle project and in the Build section click on Add build step and select the Invoke Top level Maven targets.

Create the New Item as follows.
Provide the item name and select the Maven project and click on OK.



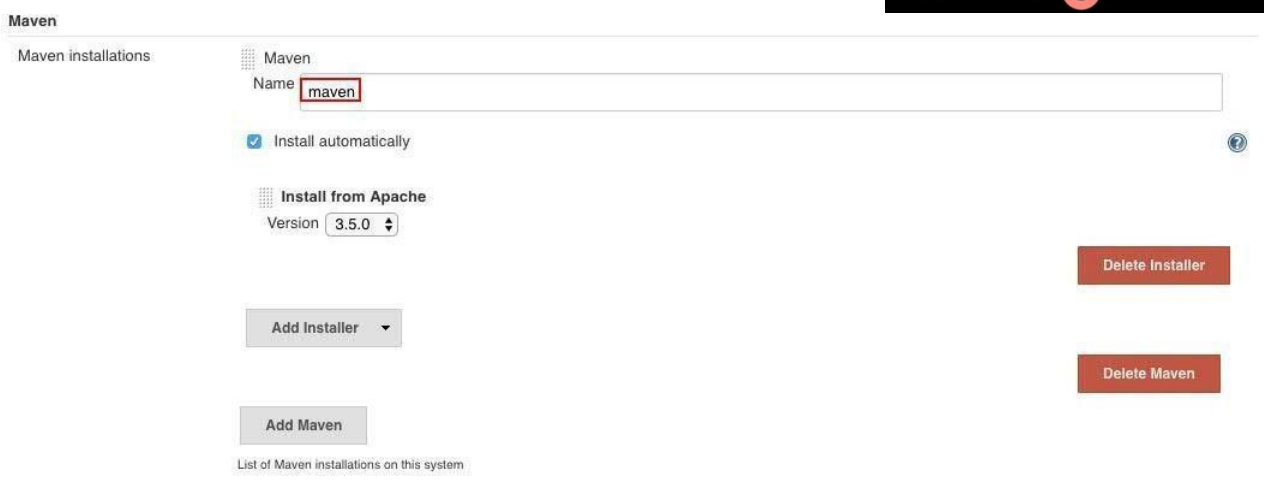
Once you click on OK, you will come to jobs configuration page as follows.





Once you provide all the details click on Save.

http://localhost:8080/configureTools/



Possible Errors

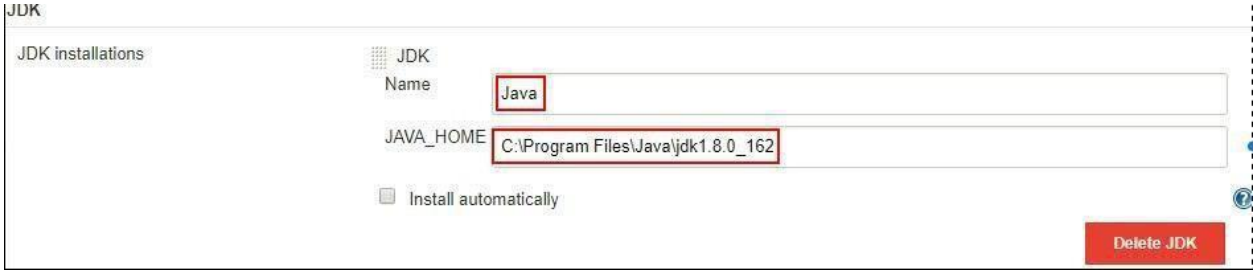
```
[ERROR] COMPILATION ERROR :
[INFO] -------------------------------------------------------------
[ERROR] No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK?
```

Solution1

Set the class path for Java.

Solution2

Go to the Jenkins Dashboard ---> Click on Manage Jenkins ---> Global Tool Configuration ---> in **JDK** section give the full path where u have installed the Java.



Jenkins - Security

How to create the users in Jenkins?

Click on Manage Jenkins ---> Manage Users ---> Create User ---> Provide the below

details Username:

Password: & Confirm password:

Full name:

E-mail address:

Click on Create User

How to see the list of Users in Jenkins?

Once you logged into Jenkins Dashboard
Go to Left Side Navigation Bar ---> Click on People
You will see list of users available in Jenkins.

How to remove/delete the User in Jenkins?

Click on Manage Jenkins ---> Manage Users ---> click on below Gear icon one circle with cross symbol

It will ask Are you sure about deleting the user from Jenkins? confirmation message Click on ---> Yes

Now User is deleted successfully.

How to change the password for existing users?

Note: TBD

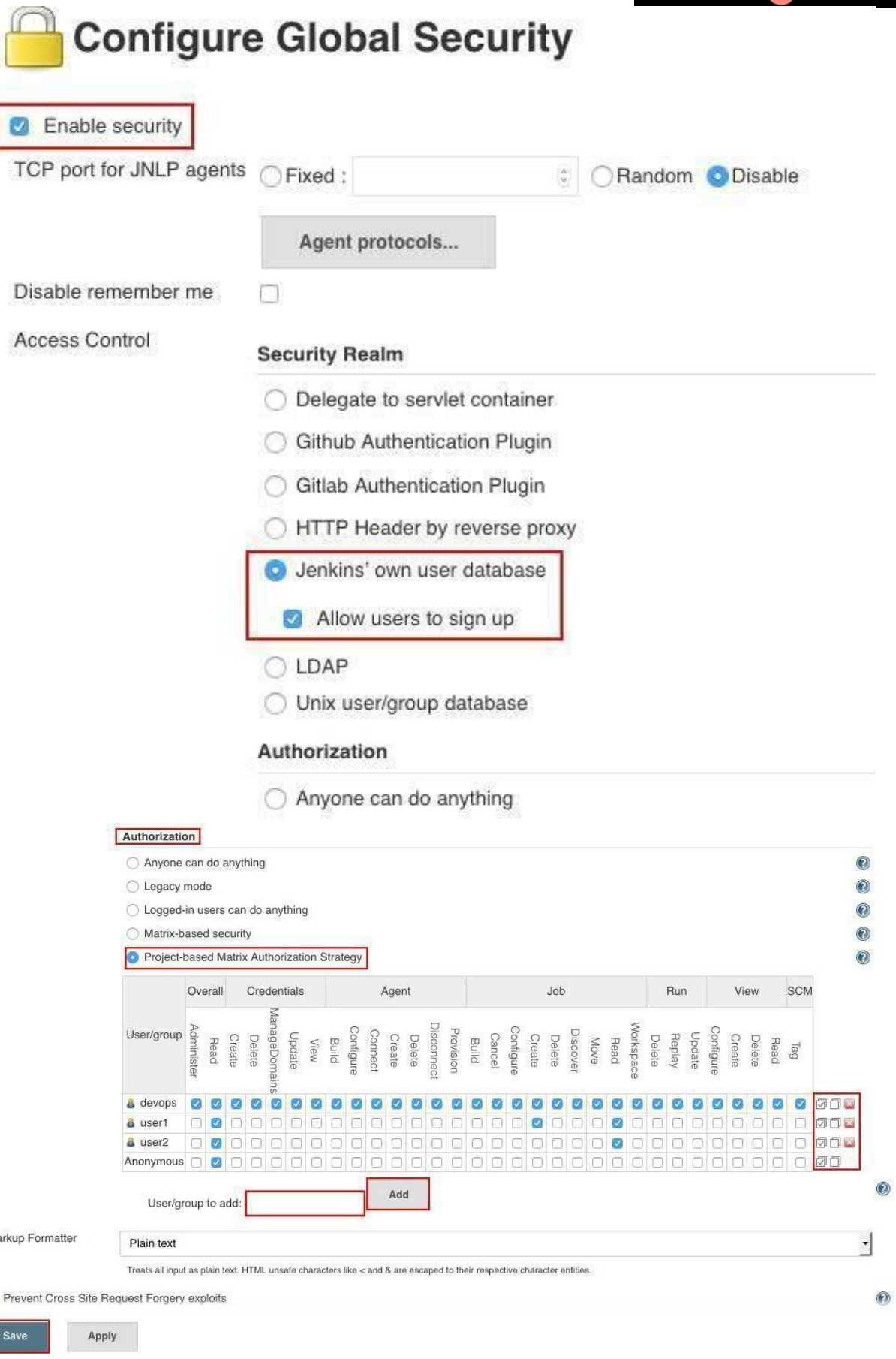
Project-based Matrix Authorization Strategy is an authorization method using which we can define which user or group can do what actions on which job. This gives us a fine-grained control over user/group permissions per project.

To Enable the Project-based Matrix Authorization Strategy need to configure in Jenkins as follows:

Step 1: Click on Manage Jenkins and choose the 'Configure Global Security' option.

Step 2: Click on Enable Security option.
As an example, let's assume that we want Jenkins to maintain it's own database of users, so in the Security Realm, Select the radio button of 'Jenkins' own user database'.

Step 3: Under Authorization, select "Project-based Matrix Authorization Strategy" and add 2 or 3 users, one administrator (say devops) and a regular user (say user1 and user2).


Configure Global Security
Enable security
TCP port for JNLP agents
Fixed :
Random
Disable
Agent protocols...
Disable remember me
Access Control
Security Realm
Delegate to servlet container
Github Authentication Plugin
Gitlab Authentication Plugin
HTTP Header by reverse proxy
Jenkins' own user database
Allow users to sign up
LDAP
Unix user/group database
Authorization
Anyone can do anything
Authorization
Anyone can do anything
Legacy mode
Logged-in users can do anything
Matrix-based security
Project-based Matrix Authorization Strategy
User/group
Overall
Credentials
Agent
Job
Run
View
SCM
Administer
Read
Create
Delete
ManageDomains
Update
View
Build
Configure
Connect
Create
Delete
Disconnect
Provision
Build
Cancel
Configure
Create
Delete
Discover
Move
Read
Workspace
Delete
Replay
Update
Configure
Create
Delete
Read
Tag
devops
user1
user2
Anonymous
User/group to add:
Add
Markup Formatter
Plain text
Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.
Prevent Cross Site Request Forgery exploits
Save
Apply

All the checkboxes present besides users are for setting global permissions. Select all checkboxes against admin user to give admin full permissions.

For user1, we are selecting read permissions under jobs. With this, user1 would now have read permission to view all jobs which we would be creating later on.

We have to provide read permission under “Overall” category to any regular user otherwise the user won’t be able to see anything after login.

All the checkboxes present besides users are for setting global permissions. Select all checkboxes against admin user to give admin full permissions. For user1, we are selecting read permissions under jobs. With this, user1 would now have read permission to view all jobs which we would be creating later on. We have to provide read permission under “Overall” category to any regular user otherwise the user won’t be able to see anything after login.

Finally, you can click on Save button.

Below scenario will applicable in Matrix based security

Error : Access Denied

<<User>> is missing the Overall/Readpermission

If you get this error, Pease follow below steps.

Solution:
Click on Manage Jenkins ---> Configure Global Security ---> User/group to add: Enter the user Name and click on Add button and --->
Enable the appropriate feature ---> Click on Save Button.

Jenkins Build Status Icon Colours

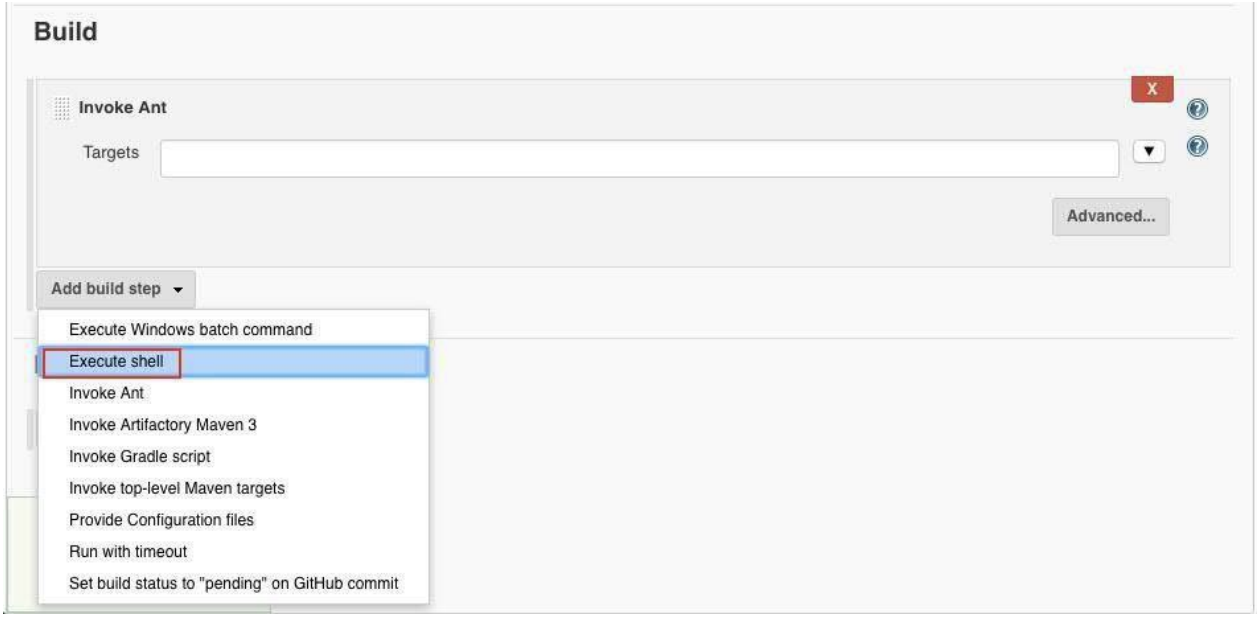
Status of the build	Description
	Failed
	Unstable
	Success
	Pending
	Disabled
	Aborted

Figure a: Build status

Job health	Description
	No recent builds failed
	20-40% of recent builds failed
	40-60% of recent builds failed
	60-80% of recent builds failed
	All recent builds failed
	Unknown status

Figure b: Weather reports

Deploy the Application Through Script



Add the below script in **Execute shell**

Linux/MAC for Tomcat

```
#!/bin/sh
echo "Starting to copy the build artifact"

cp $WORKSPACE/war/SampleAntProject.war
/Users/SimonLegah/Softwares/Running/apache-tomcat-9.0.37/webapps/ echo
"Deployed the build artifact into tomcat server successfully"
```

Windows

```
echo "Starting to copy the build"
copy %WORKSPACE%\war\SampleAntProject.war C:\\apache-tomcat-
8.5.23 \webapps\ echo "Copied the build to tomcat"
```

Linux/MAC for WIldFly

```
#Deploy in WildFly server
#!/bin/sh
 echo "Starting to copy the build"
 cp $WORKSPACE/war/SampleAntProject.war
/Users/bhaskarreddyl/SimonLegahN/Softwares/Running/wildfly11.0.0.Final/standalone/deploym
ent      s/
 echo "Copied the build to WildFly successfully"
```

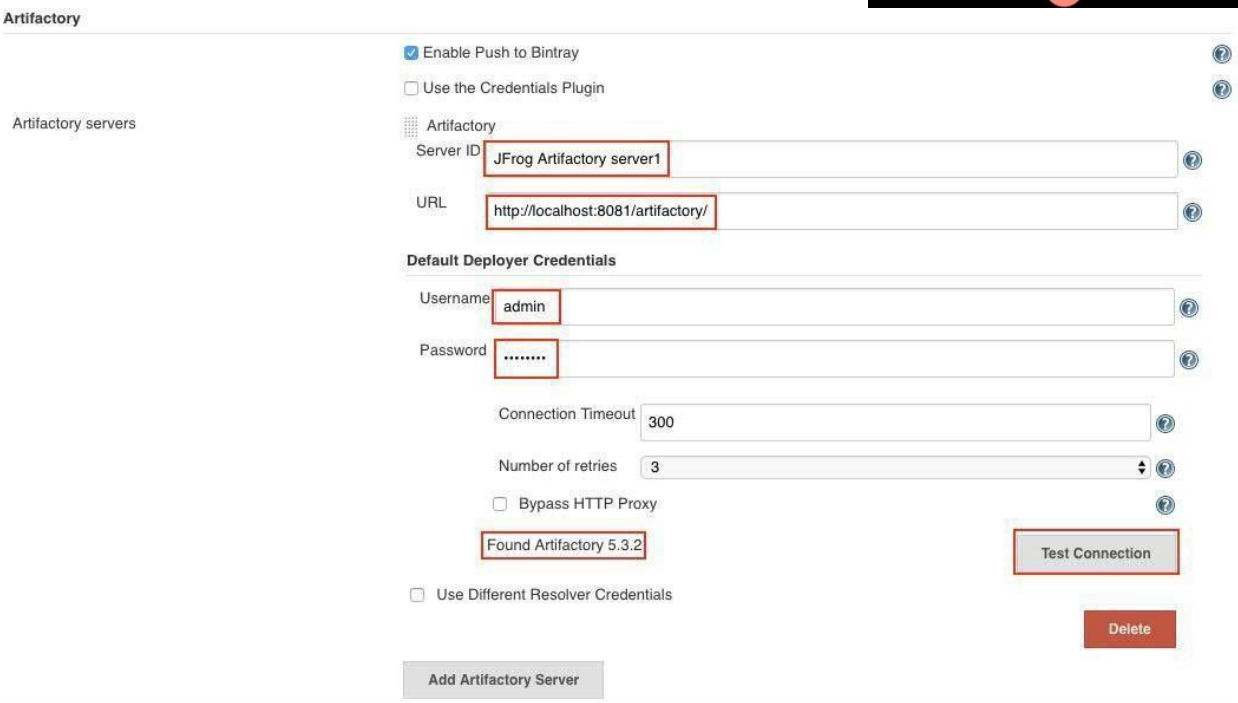
Note: If we want to deploy in Tomcat, which is installed in any remote machine, use below lines of code.

scp $WORKSPACE/war/SampleAntProject.war <<User Name>>@<<ServerIP>>:/opt/apache- tomcat-7.0.78/webapps

cp%JENKINS_HOME%\jobs\%JOB_NAME%\builds\%BUILD_NUMBER%\lo g C:\Users\windows7\Downloads\newfolder\

Integrate JFrog Artifactory with Jenkins

Install **"Artifactory Plugin"** plugin.
Got to the Manage Jenkins ---> Configure System --->
In the **Artifactory** section fill the below details and click on Save.



Note: Once you entered all the details click on **TEST CONNECTION**. IF connection is succeeded you will see the message like **Found Artifactory <<Version>>**.

Jenkins – Metrics and Trends

There are various plugins which are available in Jenkins to showcase metrics for builds which are carried out over a period of time. These metrics are useful to understand your builds and how frequently they fail/pass over time. As an example, let's look at the '**Build History Metrics plugin'**. This plugin calculates the following metrics for all of the builds once installed

Mean Time To Failure (MTTF) Mean Time To Recovery (MTTR) Standard Deviation of Build Times

Enable LDAP security to Jenkins

---

http://www.scmgalaxy.com/tutorials/complete-guide-to-use-jenkins-cli-command-line

Jenkins CLI

Jenkins has a built-in command line interface (CLI) that allows users and administrators to access Jenkins from a script or shell environment. This can be convenient for scripting of routine tasks, bulk updates, troubleshooting, and more.

Advantages of Jenkins CLI:

- Easier
- Faster
- Memory management
- Automation tasks.

Pre-Requisites

a) Jenkins server should run.

b) Enable security as follows.

Go to Jenkins dashboard in Home page (e.g http://localhost:8080/) -> Manage Jenkins -> Configure Global Security -> Click on “Enable security” checkbox

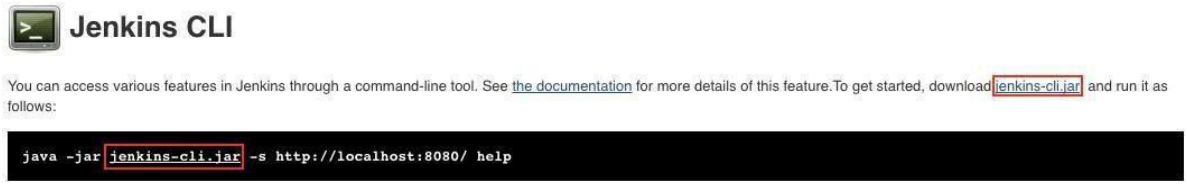
You can also configure “Access Control” and “Authorization” option in Global Security page.

Download the Jenkins CLI jar file as follows.

Method 1

Open the below url

http://localhost:8080/cli/



Click on Jenkins-cli.jar.

Method 2

Click on below url, it will automatically download the jar file.

http://<<Jenkins Server URL>>/jnlpJars/jenkins-cli.jar

Example: http://localhost:8080/jnlpJars/jenkins-cli.jar

Here

Copy into any folder as follows

#cp jenkins-cli.jar /opt/jenkins/

Go to the directory where Jenkins-cli.jar is there and run the below command to get the help.

Login Jenkins using username and Password

java -jar jenkins-cli.jar -s http://localhost:8080/ help --username devops --password passw0rd

Get the Version of Jenkins

#java -jar jenkins-cli.jar -s http://localhost:8080/ version --username devops --password passw0rd

Get all the jobs of Jenkins

#java -jar jenkins-cli.jar -s http://localhost:8080/ list-jobs --username devops --password passw0rd

Delete the Job

#java -jar jenkins-cli.jar -s http://localhost:8080/ delete-job ant-java-job-dev --username devops -- password passw0rd

#java -jar jenkins-cli.jar -s http://localhost:8080/ disable-job ant-web-job-dev --username devops -- password passw0rd

While executing above command if you see Enter passphrase, follow the below configuration. (Manage Jenkins ---> Configure Global Security ---> enable the Enable Security ---> Apply and Save.)
Manage Jenkins ---> Configure System --->SSH Public Keys (Enter here any value, same value u can use in CLI)

Jenkins Pipeline Project

Required Plugins

1) Pipeline Maven Integration Plugin
2) JUnit Attachments Plugin
3) Task Scanner Plugin

In Jenkins Pipeline project, we will use one file called Jenkinsfile, in this file we will write groovy code to build process.
We will write Jenkinsfile in 2 ways.
1) Declarative way
2) Scripted way.

1) **Scripted Pipeline Syntax**
2) **Declarative Pipeline Syntax**

Jenkins Multi Branch Pipeline Project

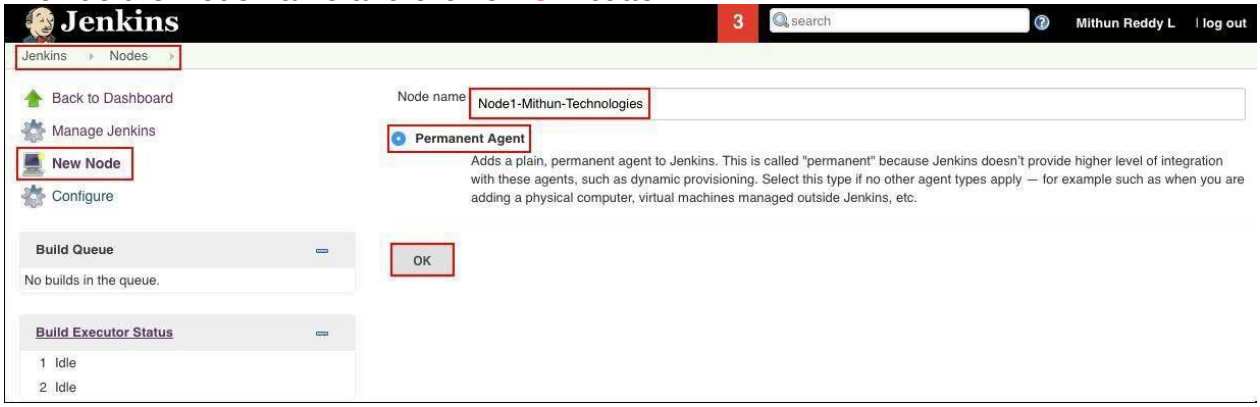
Required Plugins

1) Pipeline: Multibranch

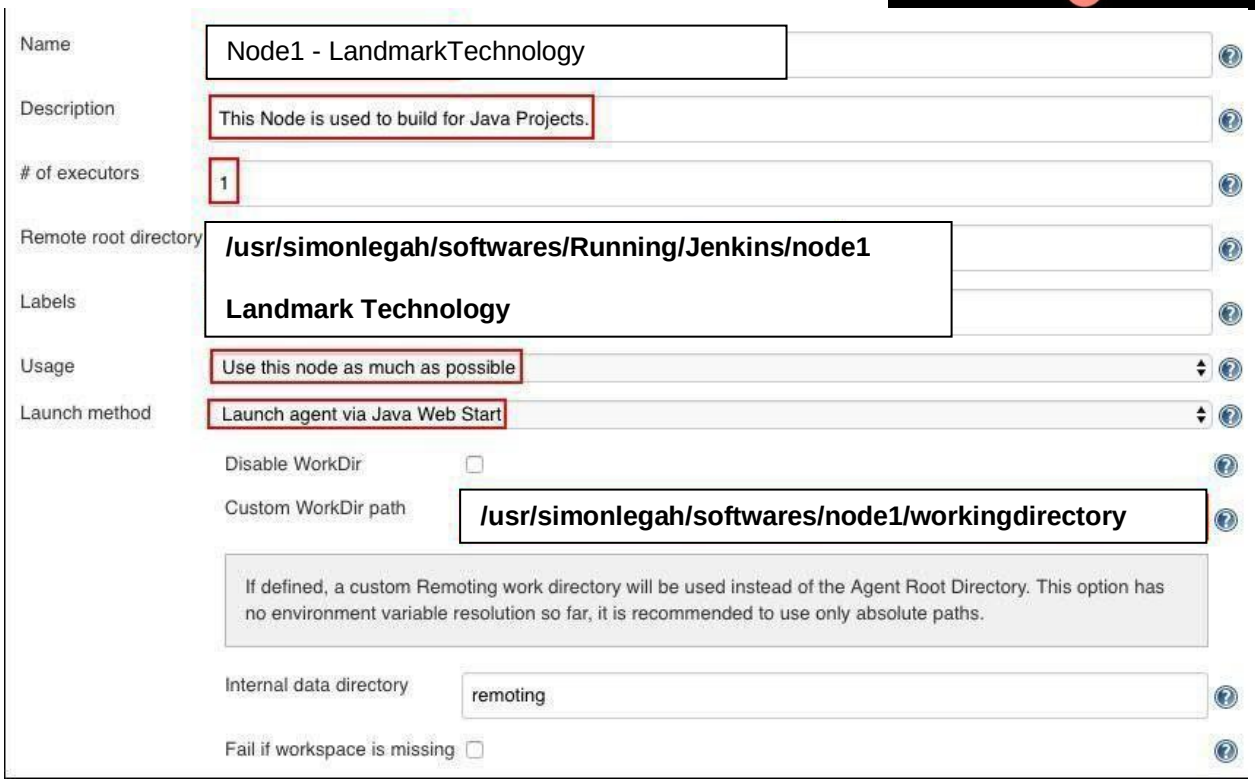
Blue Ocean Plugin

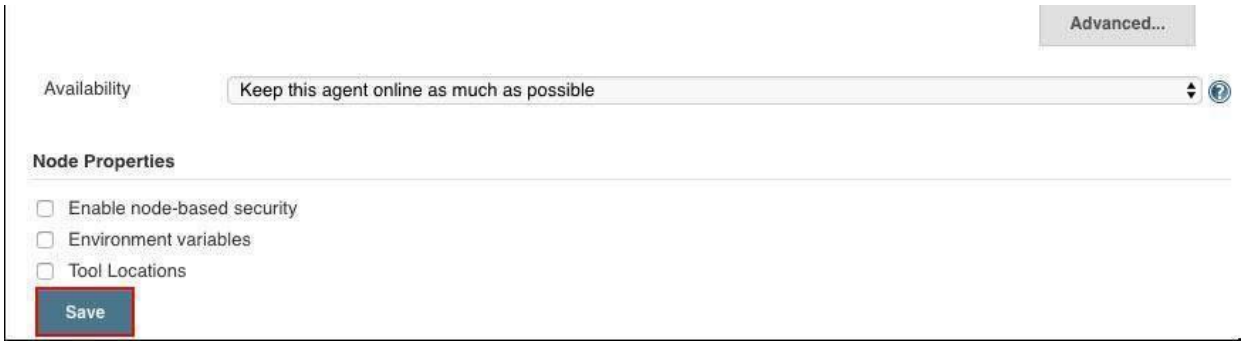
Jenkins Master-Slave setup

Manage Jenkins ---> Manage Nodes ---> New Node
Provide the Node name and click on **OK** button.



Provide all the details as follows and click on **Save** button.

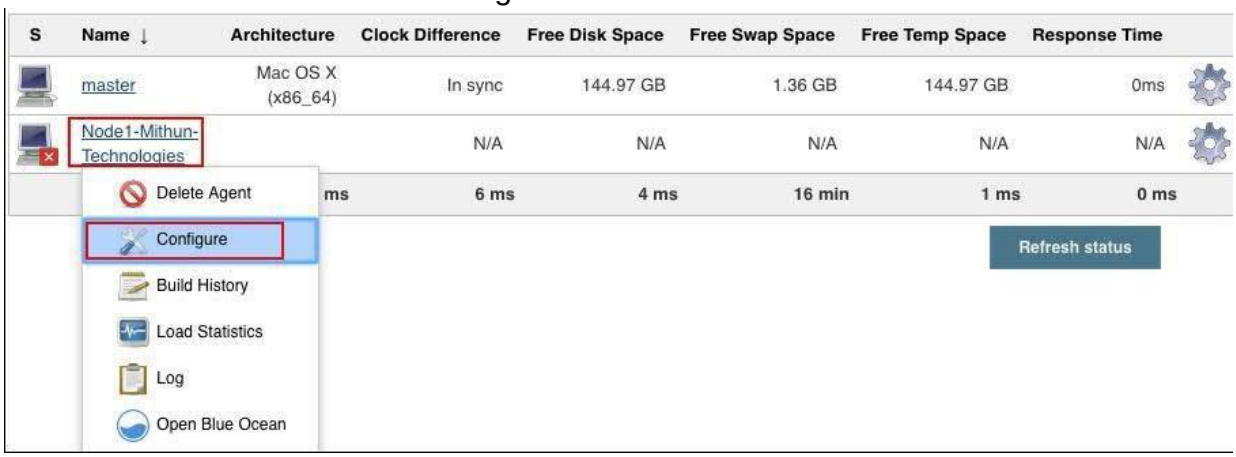




Note: Suppose if you don't see "**Launch agent via Java Web Start**" option, do the below configurations.
Manage Jenkins ---> Configure Global Security ---> enable the **TCP port for JNLP agents** (by default, it is Disabled.)

Once you click on Save you will see the Nodes and Master detail, and select the Node which

we have created and click on configure.



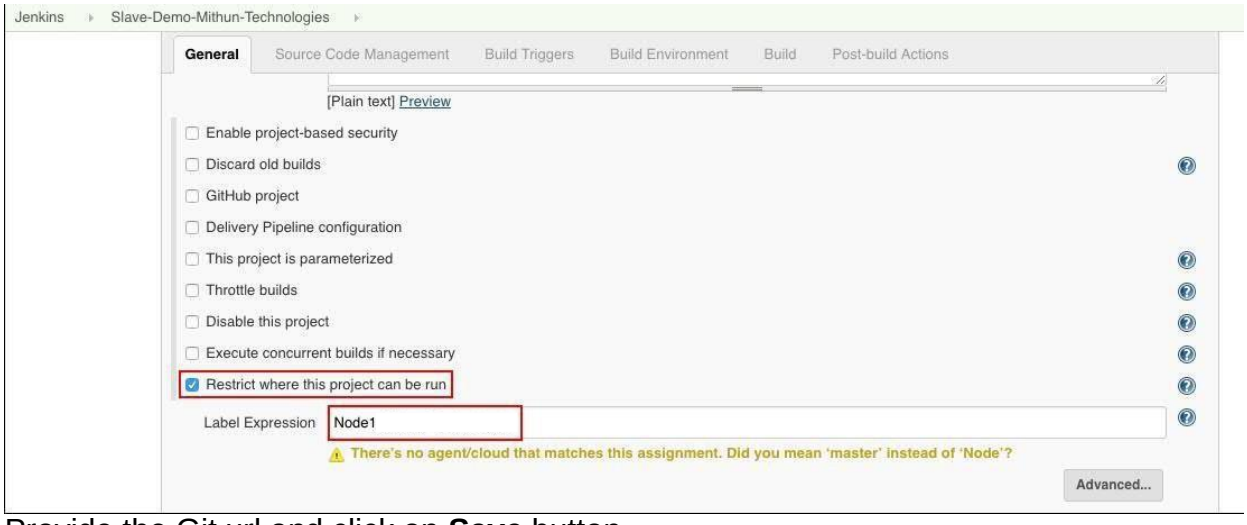
You will see below screen and click download the slave.jar file.
Copy slave.jar file into any directory (/Users/bhaskarreddyl/SimonLegah/Softwares/Running/jenkins/node1)
Go to the path where slave.jar copied and run the below command.

```
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/Node1-Lanmark-Technology/slave-agent.jnlp -secret 8e6c24c3e977342073d2184d051b1fb87f30d57acd0c63ae0a913008e65ad86f - workDir "/Users/simonlegah/SimonLegah/Softwares/Running/jenkins/node1/workdirectory"
```

Now slave become communicating to node and it is live.

Now you can use this slave for job creation.

Create one Freestyle project/any kind of project and select the Restrict where this project can be run and select the Node which you have created.



Provide the Git url and click on **Save** button.

Jenkins Home Directory Change in RHEL 7.5 Version

By Default, Jenkins home directory will be in /var/lib/jenkins in RHEL.
We can change the Jenkins default home directory to your custom directory(/opt/LandmarkTechnology/jenkins).

Stop the Jenkins service if it is running.

sudo su -
service jenkins status
service jenkins stop
Create a directory LandmarkTechnology in opt directory as follows.

#mkdir -p /opt/LandmarkTechnology

Copy the jenkins directory to
cp -r /var/lib/jenkins/ /opt/landmarktechnology/

##Change the ownership as follows.

chown -R jenkins:jenkins /opt/ landmarktechnology /jenkins/

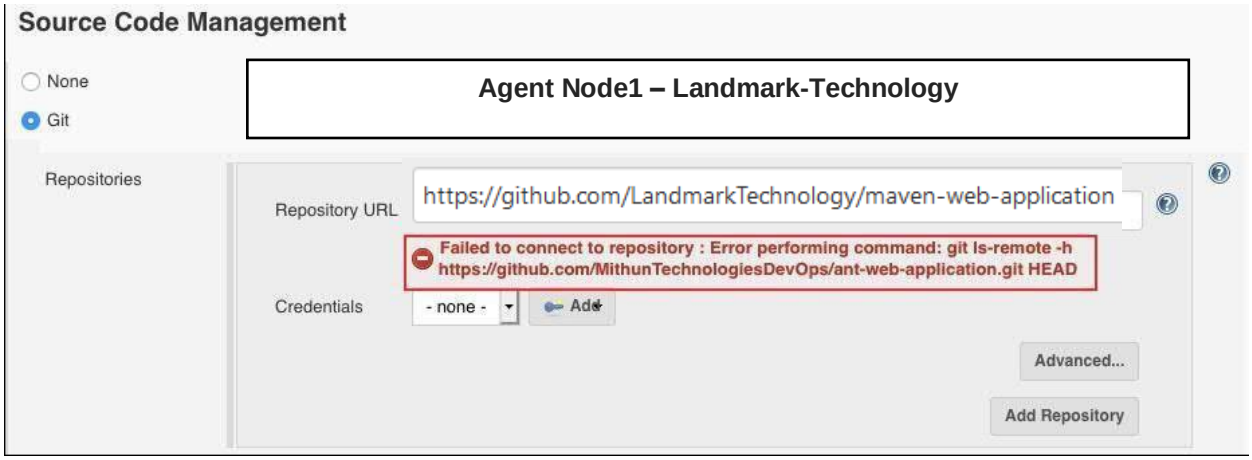
##Change the permissions as follows.
chmod -R 775 /opt/mylandtech/jenkins/

##Start the jenkins service as follows.

service jenkins start

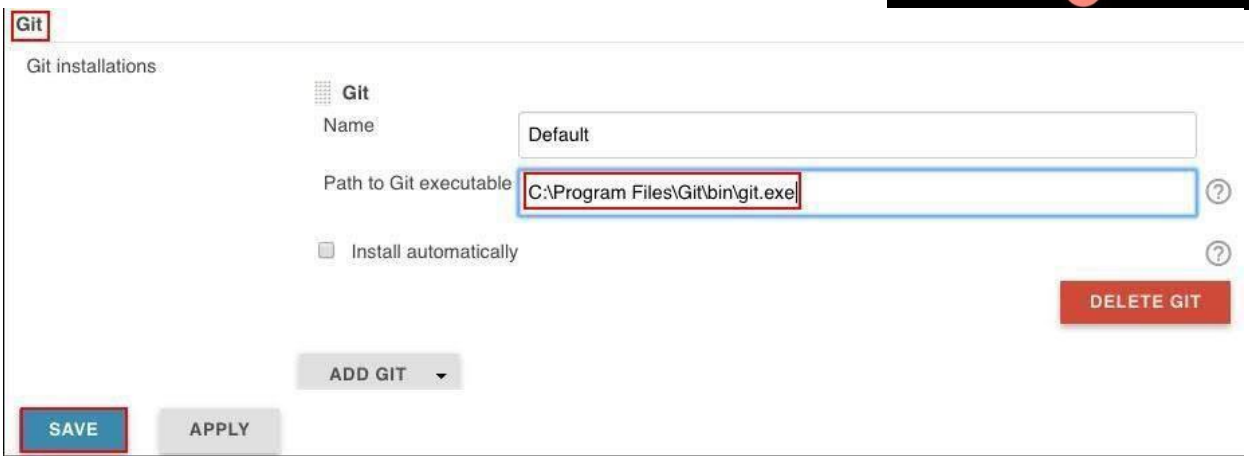
Possible Errors and Solutions:

Issue:



Solution – Windows OS

Go to the Jenkins dashboard, Click on Manage Jenkins - ❼ Global Tool Configuration In Git option, Give the Gitbash installed path in **Path to Git executable** text filed as follows.



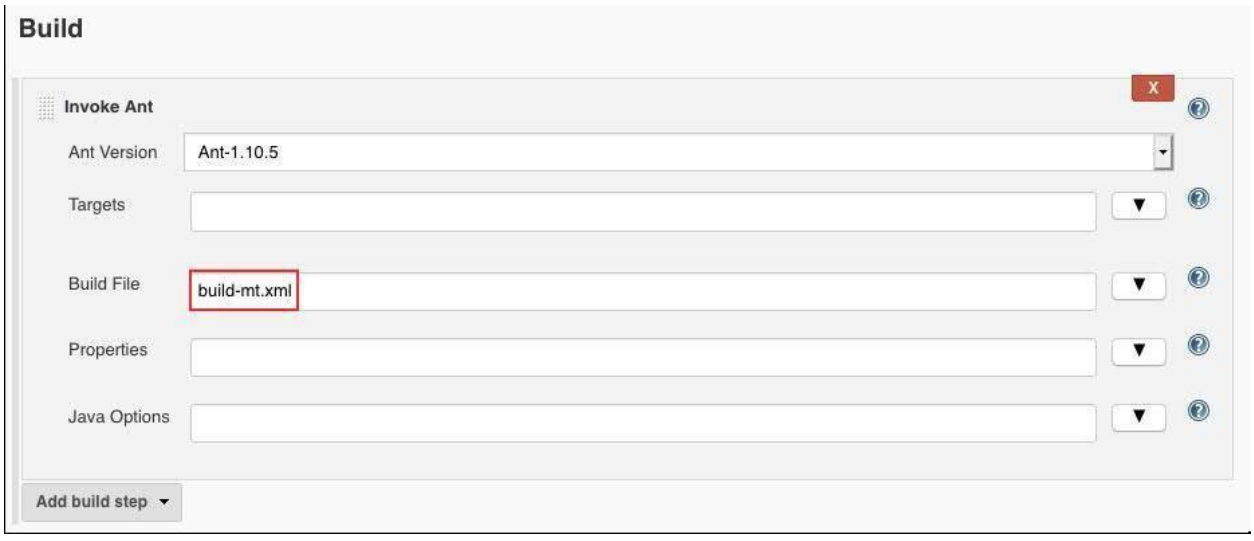
Solution – Linux

Install the git.

Issue:

```
Commit message: "Update home.jsp"
First time build. Skipping changelog.
ERROR: Unable to find build script at /var/lib/jenkins/workspace/flipkart-dev/build.xml
Finished: FAILURE
```

In Build step, give the build file name as in below screen shot.



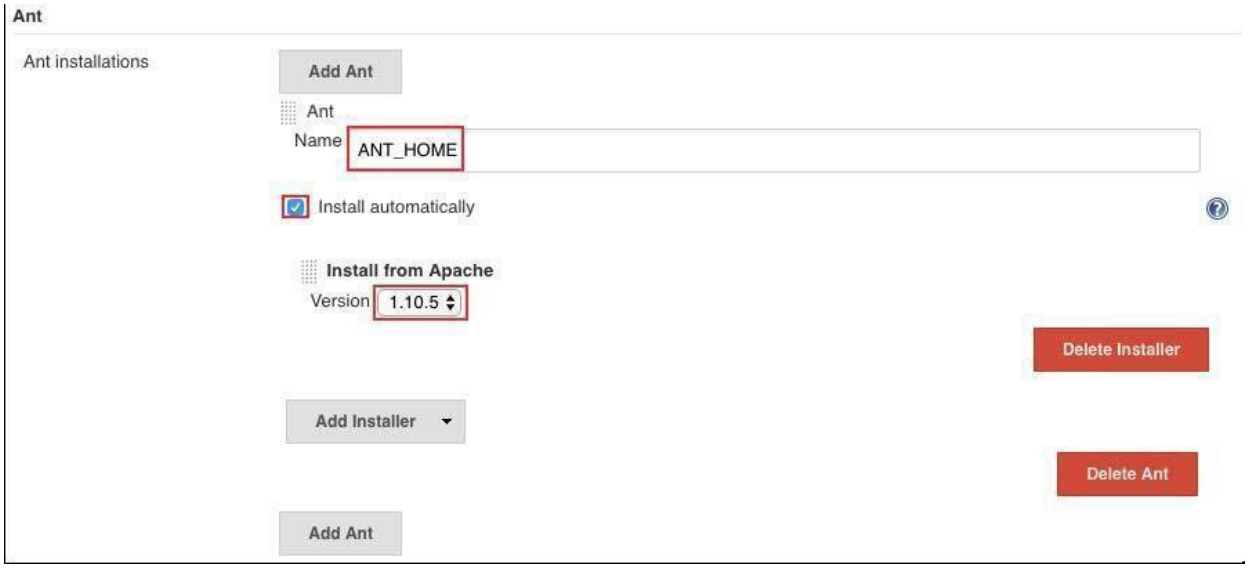
Issue:

While building if you see below error

```
[Test] $ ant -file build-mt.xml
ERROR: command execution failed.Maybe you need to configure the job to choose one of your Ant installations?
Finished: FAILURE
```

Solution:

Go to the Manage Jenkins ---> Global Tool Configuration ---> Ant ---> Ant Installations...

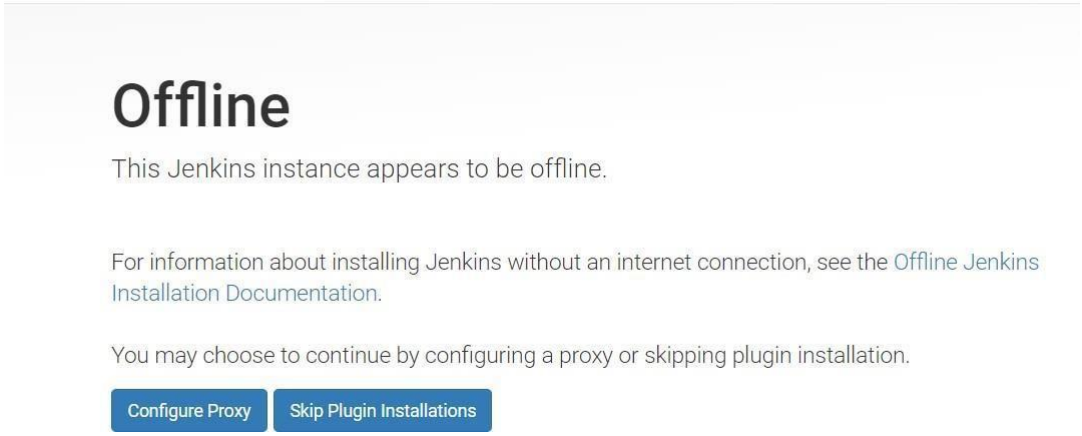


and in Job, select the Ant Versions as follows.

Installation Issues:

Issue 1: Offline

Offline



Solution

jenkinshomedir/hudson.model.UpdateCenter.xml and change url to use **http** instead of

https. Once you changed from https to http, you need to restart the Jenkins.

Issue

```
+refs/heads/*:refs/remotes/origin/*" returned status code 128:
stdout:
stderr: remote: Password authentication is not available for Git operations.
remote: You must use a personal access token or SSH key.
```

Solution

If you see this error, generate SSH or PAT and use these keys instead of password.

Issue Jenkins Start

#service Jenkins start

```
[root@ip-172-31-17-1 jenkins]# service jenkins start
Starting jenkins (via systemctl):  Job for jenkins.service failed because the control process exited with error code. See "systemctl status jen
kins.service" and "journalctl -xe" for details.
                                                           [FAILED]
[root@ip-172-31-17-1 jenkins]#
```

#journalctl -xe

```
Mar 10 11:33:17 ip-172-31-17-1.ap-south-1.compute.internal sshd[3035]: Disconnected from 218.92.0.198 port 44310 [preauth]
Mar 10 11:33:33 ip-172-31-17-1.ap-south-1.compute.internal sshd[3039]: Connection closed by 218.92.0.198 port 19055 [preauth]
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal polkitd[465]: Registered Authentication Agent for unix-process:3057:17319042 (system bus name
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal systemd[1]: Starting LSB: Jenkins Automation Server...
-- Subject: Unit jenkins.service has begun start-up
-- Defined-By: systemd
-- Support: http://lists.freedesktop.org/mailman/listinfo/systemd-devel
--
-- Unit jenkins.service has begun starting up.
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal runuser[3068]: pam_unix(runuser:session): session opened for user jenkins by (uid=0)
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal jenkins[3063]: Starting Jenkins bash: /usr/bin/java: No such file or directory
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal runuser[3068]: pam_unix(runuser:session): session closed for user jenkins
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal systemd[1]: jenkins.service: control process exited, code=exited status=1
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal jenkins[3063]: [FAILED]
Mar 10 11:33:51 ip-172-31-17-1.ap-south-1.compute.internal systemd[1]: Failed to start LSB: Jenkins Automation Server.
-- Subject: Unit jenkins.service has failed
-- Defined-By: systemd
-- Support: http://lists.freedesktop.org/mailman/listinfo/systemd-devel
--
-- Unit jenkins.service has failed.
--
-- The result is failed.
```

Solution

Install the java.

Issue:

Cloning: git clone https://github.com/mylandmarktech/maven-web-application.git

```
Cloning repository https://github.com/MithunTechnologiesDevOps/ant-web-application.git
 > git init /var/lib/jenkins/workspace/Test # timeout=10
ERROR: Error cloning remote repo 'origin'
hudson.plugins.git.GitException: Could not init /var/lib/jenkins/workspace/Test
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$5.execute(CliGitAPIImpl.java:813)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$2.execute(CliGitAPIImpl.java:605)
        at hudson.plugins.git.GitSCM.retrieveChanges(GitSCM.java:1152)
        at hudson.plugins.git.GitSCM.checkout(GitSCM.java:1192)
        at hudson.scm.SCM.checkout(SCM.java:504)
        at hudson.model.AbstractProject.checkout(AbstractProject.java:1208)
        at hudson.model.AbstractBuild$AbstractBuildExecution.defaultCheckout(AbstractBuild.java:574)
        at jenkins.scm.SCMCheckoutStrategy.checkout(SCMCheckoutStrategy.java:86)
        at hudson.model.AbstractBuild$AbstractBuildExecution.run(AbstractBuild.java:499)
        at hudson.model.Run.execute(Run.java:1810)
        at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:43)
        at hudson.model.ResourceController.execute(ResourceController.java:97)
        at hudson.model.Executor.run(Executor.java:429)
Caused by: hudson.plugins.git.GitException: Error performing command: git init /var/lib/jenkins/workspace/Test
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandIn(CliGitAPIImpl.java:2049)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandIn(CliGitAPIImpl.java:2010)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandIn(CliGitAPIImpl.java:2006)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommand(CliGitAPIImpl.java:1638)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$5.execute(CliGitAPIImpl.java:811)
        ... 12 more
Caused by: java.io.IOException: Cannot run program "git" (in directory "/var/lib/jenkins/workspace/Test"): error=2, No such file or
directory
```

Solution:

Install the Git.

Issue:

There is insufficient memory for the Java Runtime Environment to continue.

Solution:

Increase the JVM size as follows. vi

/etc/sysconfig/Jenkins

```
## Type: string
## Default:     "-Djava.awt.headless=true"
## ServiceRestart: jenkins
#
# Options to pass to java when running Jenkins.
#
JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true -Xmx1024m -XX:MaxPermSize=512m"
```

Resources:

https://jenkins.io/ ---> Download software
https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+as+a+Windows+service

http://www.tothenew.com/blog/jenkins-implementing-project-based-matrix-authorization-strategy/ ---> User Access

https://support.cloudbees.com/hc/en-us/articles/216118748-How-to-Start-Stop-or-Restart-your-Instance
https://www.jdev.it/deploying-your-war-file-from-jenkins-to-tomcat/ ---> Deploy into Tomcat

WHAT IS AGILE?

- Agile software development refers to a group of software development methodologies based on interactive development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams
- Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a set of engineering best practices intended to allow for rapid delivery of high-quality software/ and a business approach that aligns development with customer needs and company goals

WHAT IS SCRUM?

- Scrum is a subset of Agile. It is a lightweight process framework for agile development, and the most widely-used one
- A "process framework "is a particular set of practices that must be followed in order for a process to be consistent with the framework. (For example, the Scrum process framework requires the use of development cycles called Sprints
- "Lightweight" means that the overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done
- Scrum significantly increases productivity and reduces time to benefits relative to classic "waterfall" processes
- Scrum processes enable organizations to adjust smoothly to rapidly-changing requirements, and produce a product that meets evolving business goals

AGILE - SCRUM BENEFITS

- An agile Scrum process benefits the organization by helping it to:
- Increase the quality of the deliverables
- Cope better with change (and expect the changes)
- Provide better estimates while spending less time creating them
- Be more in control of the project schedule and state

Continuous Testing

DevOps

What is Continuous Testing (CT)?

- Execution of tests repeatedly against a code based on deployment environments
  - Automated unit integration, coded and load tests are common
  - Depth of testing often progresses as environment gets closer to Prod
- In practice, continuous testing is the most difficult part of a continuous delivery pipeline to keep up to date

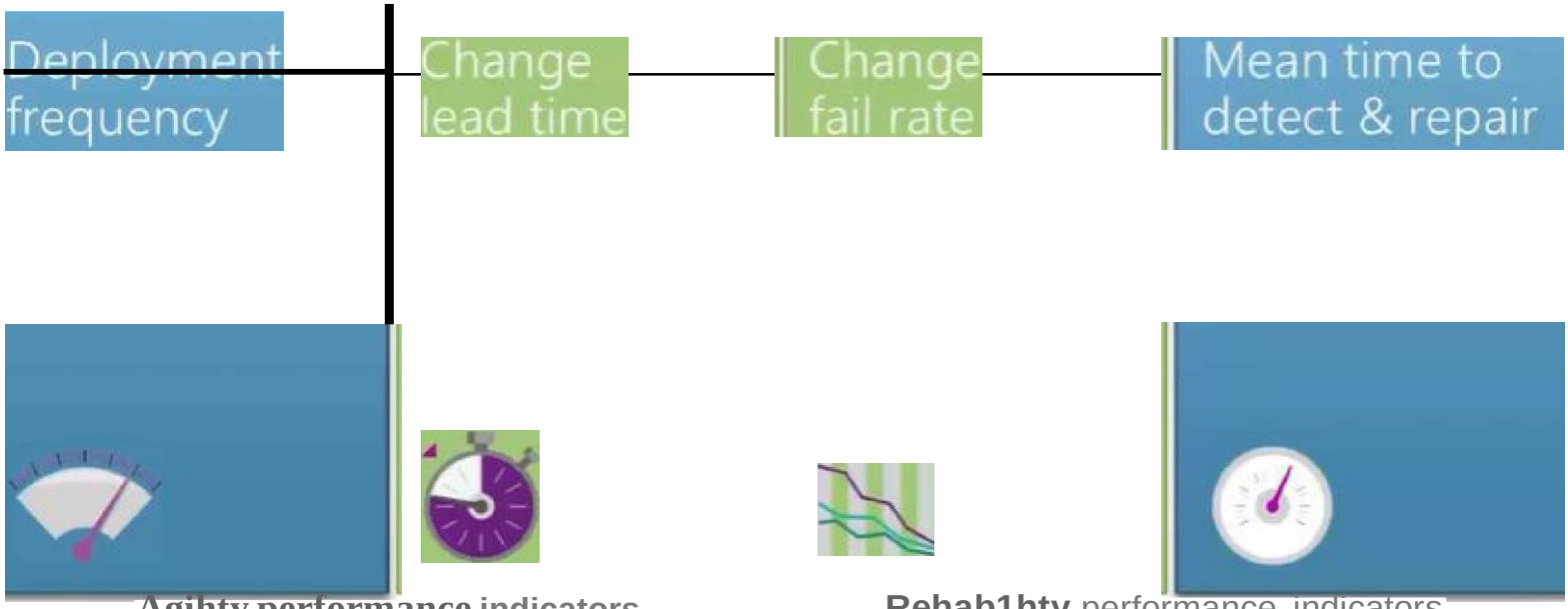
Types of Testing

- Unit Testing
  - Test units of system in isolation
- Integration Testing
  - Test components together in scenarios
- User Interface (Lil) Testing
  - Test components together in scenarios through the UI
- Regress ion Testing
  - Regression testing is a type of software testing which verifies that software which was previously developed and tested stills performs correctly after it was changed or interfaced with other software

WAIT - WHAT ABOUT DEV0 PS?

- Apart from working together as a cross- functional teams; designer, tester and developer as part of an Agile team, DevOps suggests adding operations as well in the definition of cross-functional teams
- DevOps strives to focus on the overall service o software fully delivered to the customer
- It emphasizes breaking do n barriers betwee develope sand operations teams, and getting them o collabora e in a ay he e hey benefit from combined skills
- DevOps brings some tools n econ igu ation management (puppet, chef, ansible), orchestration (zoo eeper, mesas), monilo rng, virtualization and containerization (AWS, Kuberne es - Goog le Cloud, Docker)

DEV0 PS METRICS



Agihty performance indicators

Rehab1hty performance indicators

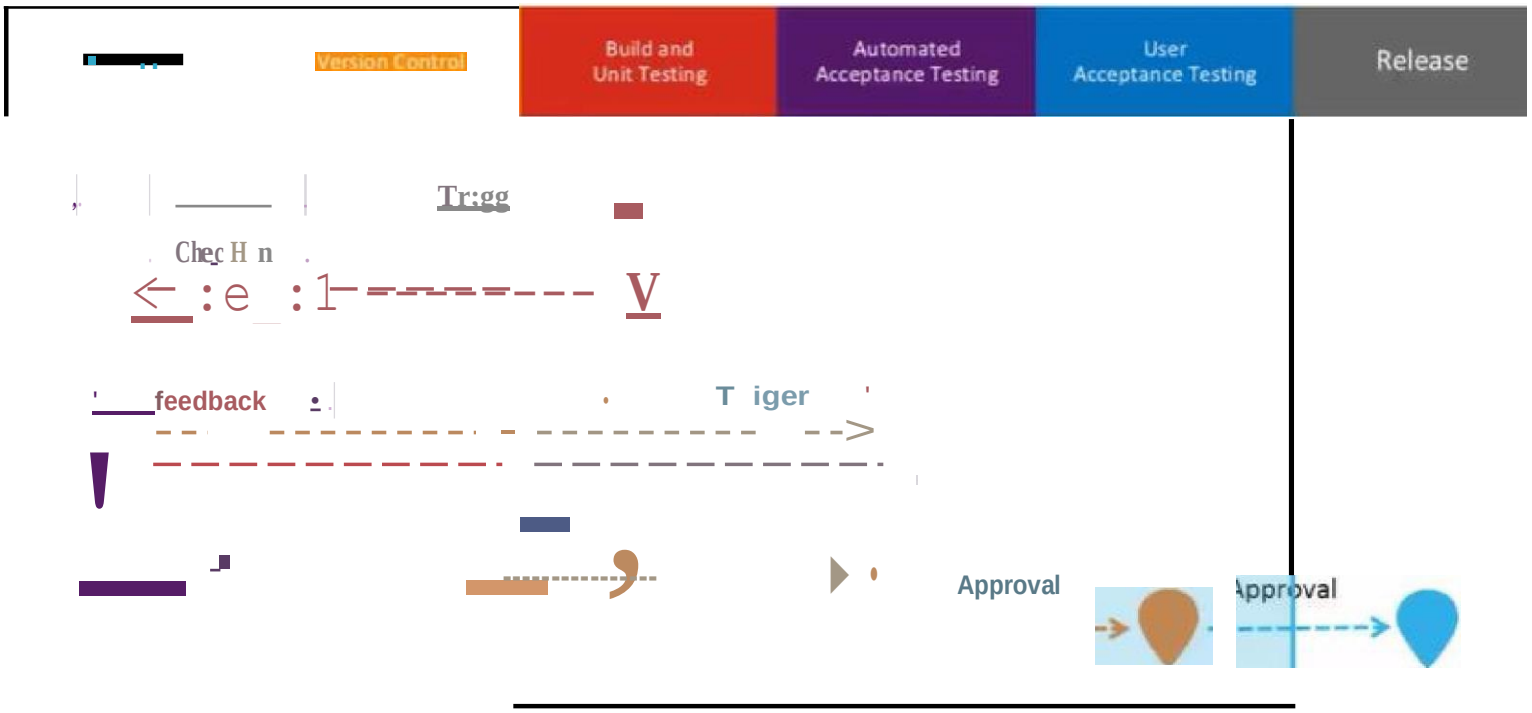
JENKINS BUILD ENVIRONMENT

- Deploy and host Jenkins, an open-source automation software predominantly used for CI/CD (Continuous Integration Continuous Deployment)
- You will use Amazon Elastic Compute Cloud (EC2) in a public subnet within your own Amazon Virtual Private Cloud (VPC) and you Will also set up an Amazon Elastic Block Store (EBS) volume

What is Continuous Delivery (C0)7

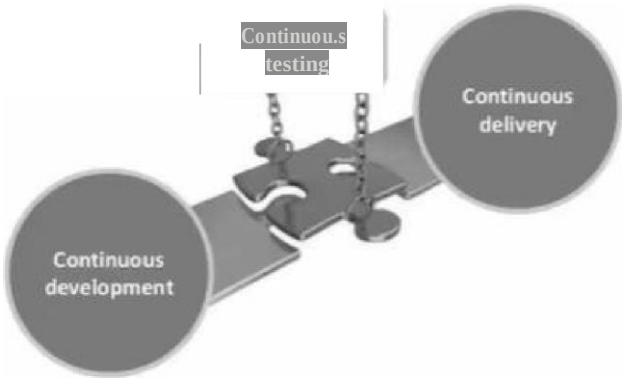
- A software engineering approach in which teams produce software in short cycles, ensuring that software can be reliably released at any time
  - Aims to build, test and release software faster and more frequently
  - Reduce the cost time and risk of delivering changes by allowing for more incremental updates to production
- In practice (continuous delivery focuses on an automated deployment pipeline
  - This may have one or more manual approval gates prior to reaching production

Continuous Testing



Benefits of Continuous Testing

- Quality gates throughout he pipeline
- Increases co nfidence in code long be e produc ion



Configurat ion Management

DevOps

What is Configuration Management?

- Management of configuration of all environments for an applicaron
  - Typically in he form of scripts
  - Version controlled
- Less formal that "traditional" configuration management
  - Emphasizes encapsulation of con iguratio ·n code over formal documentation

Benefits of Infrast ruct ure and Configuration as Code

- Allow s configuration to be version controlled
- , Detect s (and corrects) configura ion drift
- Treats infrast ructu re as filexible reso rce
- Facilitates automa ion
- Enables automa ed scale-up and sca le-ou
- Pr ovides envir onment consistency

Release Management

What is Release Management?

- Managing what is deployed where, and how
  - What which versions of components
  - Where: which environments
  - How: automation and scripting to configure environments and deploy applications
- Less formal that 'traditional' release management
  - Emphasizes quick and consistent delivery over formal change control

Benefits of Release Management

- Auditing and Traceability
- Automation
- Visibility
- Confidence through consistency
- Higher quality

Application Performance Monitoring (APM)

What is Application Performance Monitoring (APM)?

- Monitoring and learning from 'live site'
  - Diagnostics and error reporting
  - Usage
  - Notifications on application performance
- Rules for application performance and availability
  - High avaitability
  - Automated scale up/down or out/in
- Drive insights into backlog from production

APM Tools

- APM tools allow you to target bottlenecks with your application's framework
- New Relic is the reigning market leader which lets you pinpoints precisely where and when bottlenecks are occurring
- App Dynamics is also a great tool, enabling you to monitor Java, .NET, PHP, and Nodejs applications
- Compuware APM & Boundary are enterprise-geared APM tools which give you a clear view of the user experience, offering metrics like data transactions performance and user requests

Application Monitoring

- Hypothesis driven development requ res telemetry
- Proactive (not reactive) action
- Type of monitoring
  - Usage
  - Availabflity
  --- Performance
  - Custom telemetry

What Did We Learn?

- Underst anding DevOps
- Value of DevOps
- Agile and Scrum Methodologies-Framework
- DevOps in Action (Demo s - Kuberne es/Docker)
- Co ncep t of Git
- Continuous In egration and Delivery
- Configuration and ReleaseManagement
- Applicat ion Performance Monitoring