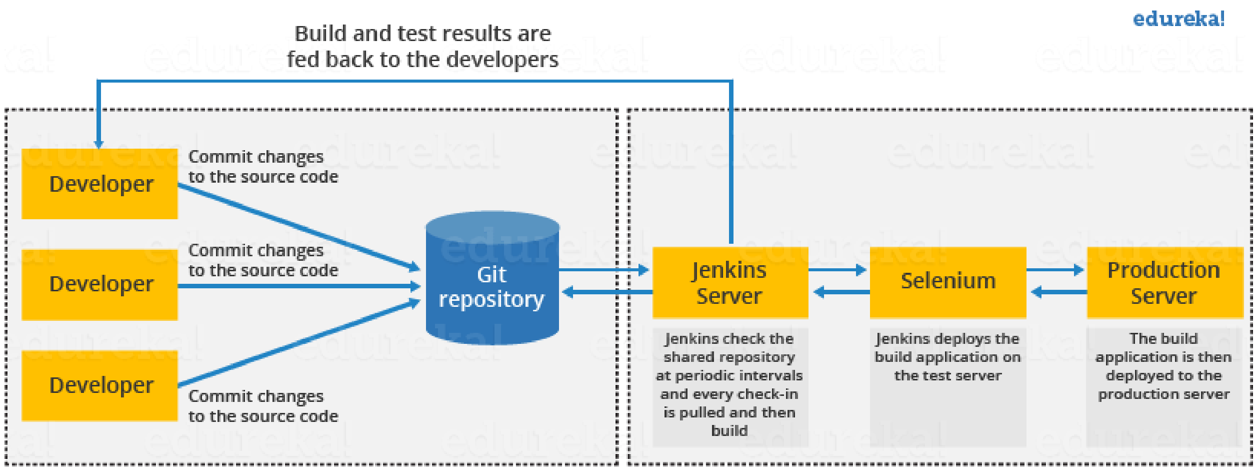
JENKINS:

Jenkins is a **Continuous Integration (CI) server** or tool which is written in java. It provides Continuous Integration services for software development, which can be started via command line or web application server.

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. It is a process of running your tests on a non-developer (say testers) machine automatically when someone pushes new code into the source repository. The below diagram shows the CI workflow.


edureka!
Build and test results are fed back to the developers
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Git repository
Jenkins Server
Jenkins check the shared repository at periodic intervals and every check-in is pulled and then build
Selenium
Jenkins deploys the build application on the test server
Production Server
The build application is then deployed to the production server

The above diagram is depicting the following functions:

- First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.
- Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.
- If the build fails, then the concerned team will be notified.
- If built is successful, then Jenkins deploys the built in the test server.

- After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.
- It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.

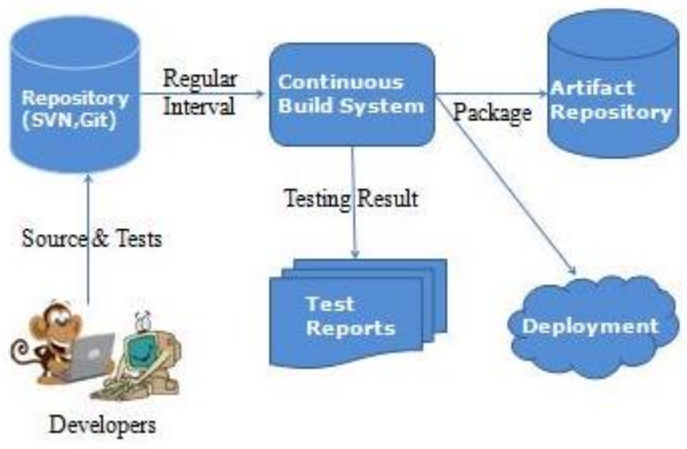


fig: The Continuous Build System here is Jenkins

Some of the attractive reasons why you need automate build testing and integration are:

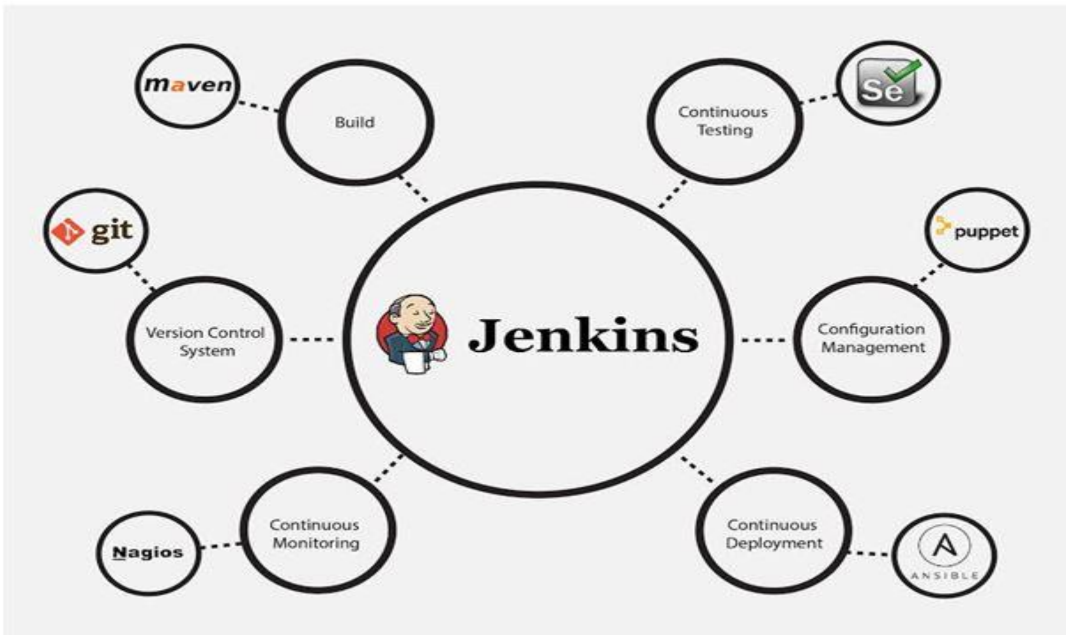
- **Developer time is concentrated on work that matters:** the developer's time is saved without wasting.
- **Software quality is made better:** Issues are detected and resolved almost right away.
- **Makes development faster:** Most of the integration work is automated. Hence integration issues are less.

Advantages of Jenkins

- **Jenkins** is an open source tool with much support from its community.

- Installation is easier.
- It has more than 1000 plug-in to make the work easier.
- It is easy to create new Jenkins plugin if one is not available.
- It is a tool which is written in Java. Hence it can be portable to almost all major platforms.

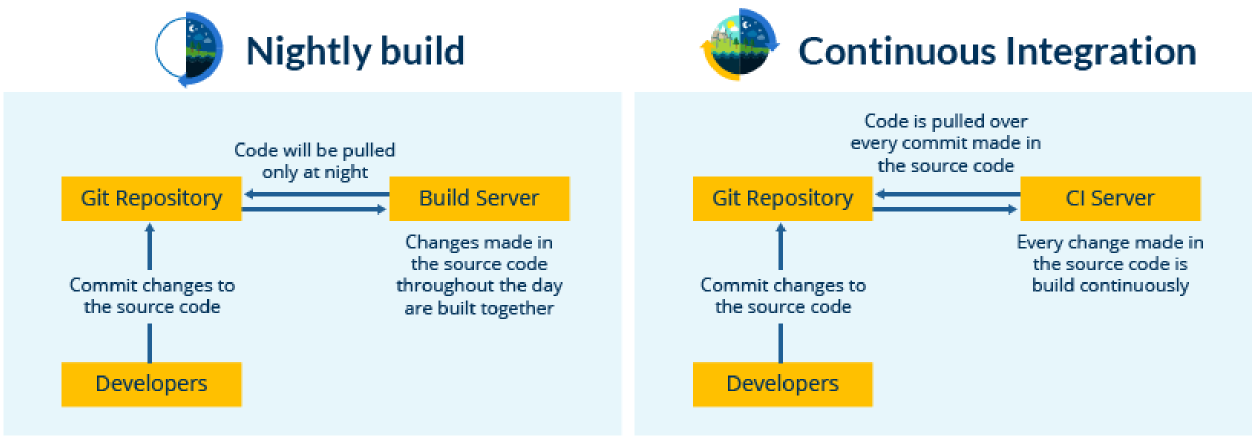
The diagram below depicts that Jenkins is integrating various DevOps stages:



Once the project is configured in Jenkins then all future builds are automated. It has basic reporting features like

1) status
2) weather reports (job health) .

So overall to say, Jenkins integrates development life-cycle processes of all kinds which include building, documenting, testing, packaging, staging, deploying, static analysis and plenty more.


Nightly build
Code will be pulled only at night
Git Repository
Build Server
Commit changes to the source code
Changes made in the source code throughout the day are built together
Developers
Continuous Integration
Code is pulled over every commit made in the source code
Git Repository
CI Server
Commit changes to the source code
Every change made in the source code is build continuously
Developers

Steps:

1) go to google type Jenkins and open the first link
2) then find **.war** file to download
3) then copy that file to a folder
4) open command in windows and open that folder where .war file is stored
5) type this command : **java -jar jenkins.war**
6) this will run and return an admin password copy it somewhere
7) now Jenkins is running at **localhost:8080**
8) go there and enter the copied password
9) now it will redirect to next page with two options ,to install suggested plugins and to install by your own
10) install suggested plugins
11) then new page opens, where we can create new users or continue as admin(note: admin is username and the password is the generated one)
12) after logging Welcome to Jenkins appear

Installing Tomcat(not necessary) :

1) What's needed : Tomcat 5 or above and Java 7 or above and also set PATH variable for java jdk

2) Download and unzip tomcat in any folder and then copy the Jenkins.war file in **tomcat/webapps** folder

3) Open command prompt and goto path **tomcat /bin** (whatever the full path is)

and type startup.bat

4) Tomcat would run on localhost:8080 and Jenkins would be on

localhost:8080/Jenkins

NOTE: to directly run Jenkins now (as Jenkins cant run on 8080 now as on that tomcat is running) we need to use different port :

Java –jar Jenkins.war --httpPort=9090

1) How to change home directory in Jenkins?

C:\Users\moabbasi\.jenkins

For instance the above will be the default home directory for any **job details, logs, plugins and configuration** (in my case) .

So why to change it:

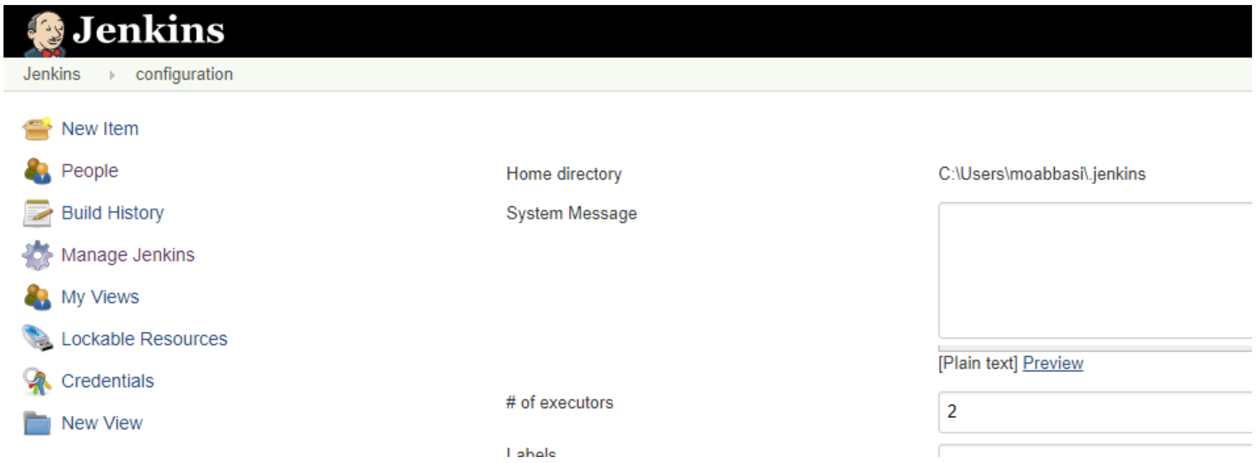
1) Project requirement

2) Enough disk space is required

How to change:

1) Check your current home directory by running Jenkins and then go to

Manage Jenkins -> Configure System



Home directory C:\Users\moabbasi\.jenkins

System Message

[Plain text] Preview

# of executors 2

Labels

Here we can see the Home directory as: C:\Users\moabbasi\jenkins

Step 1: Check your current home directory

Step 2: Create a new folder (which will be new home directory)

Step 3: Copy all data from old directory to new directory

Step 4: change environment variable - JENKINS_HOME and set to new directory Windows - change environment variable

Step 5: restart Jenkins (from command prompt or type **localhost:8080/restart**)

2) How to use cli Jenkins?

Step 1: start Jenkins

Step 2: goto Manage Jenkins - Configure Global Security - enable security

Step 3: goto - **http://localhost:8080/cli**

Step 4: download jerkins-cli jar

. Place at any location.

Step 5:Open command line and goto the path and copy the line displayed on **http://localhost:8080/cli** of your browser

```
For ex : java -jar jenkins-cli.jar -s http://localhost:8080/ help
```

If an error to authenticate comes try this:

Go to Manage Jenkins -> Configure Global Security ->

1. Enable Security Checked
2. Authorization -> Logged-in Users can do anything -> Checked allow anonymous read access

Restart Jenkins and try running

If asks for pass phrase goto : admin-> configure->SSH public keys

3) How to create Users + Manage + Assign Role ?

Step 1: Create new users, goto: manage Jenkins->manage users->create user

Step 2: Configure users goto: top right corner with user name written->configure

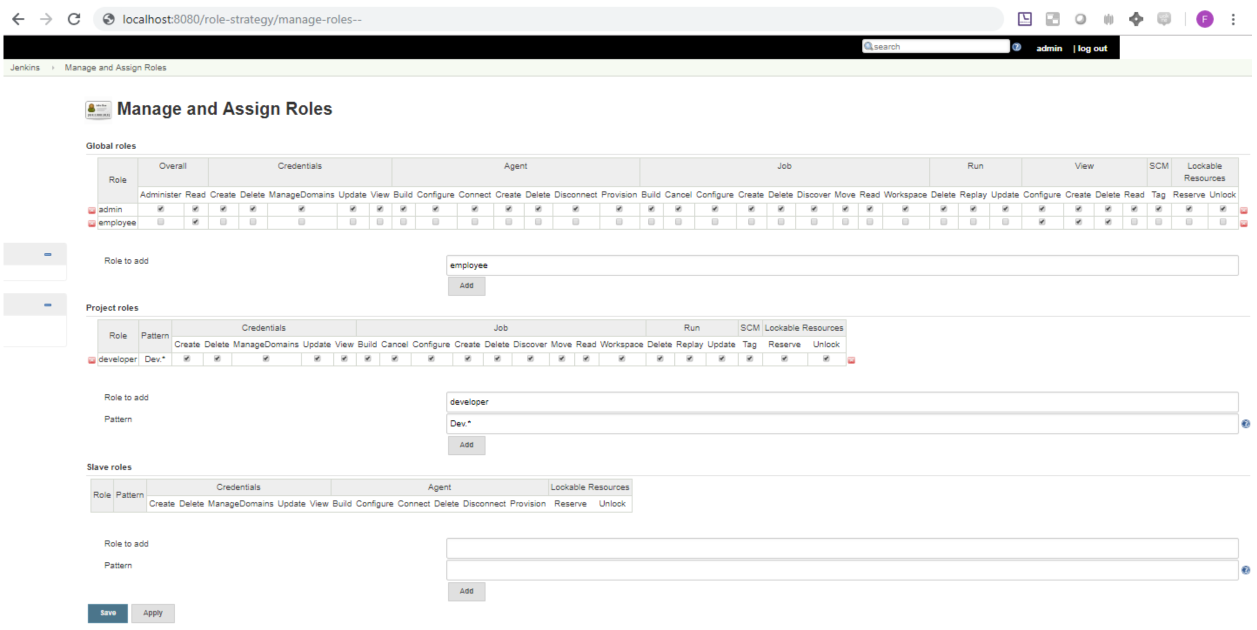
But Since every user has all the permissions so

Step 3: Create and manage user roles: manage Jenkins->manage plugins->in Available search for Role-based Authorization Strategy , **download** - restart jenkins and then goto step 4

Step 4: Manage Jenkins - Configure Global Security - Authorization - Role Based Strategy (do this as admin) and now when we login from as user we get error: **user1 is missing the Overall/Read permission**

So now goto step 5

Step 5: goto manage and Assign roles ->manage roles(create a new role) and control on the access


localhost:8080/role-strategy/manage-roles--

search admin | log out

Jenkins › Manage and Assign Roles

Manage and Assign Roles

Global roles

Role	Overall		Credentials					Agent							Job									Run			View				SCM	Lockable Resources	
	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	Reserve	Unlock
admin	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
employee	☐	✓	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	✓	✓	✓	☐	☐	☐	☐

Role to add: employee

Add

Project roles

Role	Pattern	Credentials					Job									Run			SCM	Lockable Resources	
		Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag	Reserve	Unlock
developer	Dev.*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Role to add: developer

Pattern: Dev.*

Add

Slave roles

Role	Pattern	Credentials					Agent						Lockable Resources	
		Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Delete	Disconnect	Provision	Reserve	Unlock

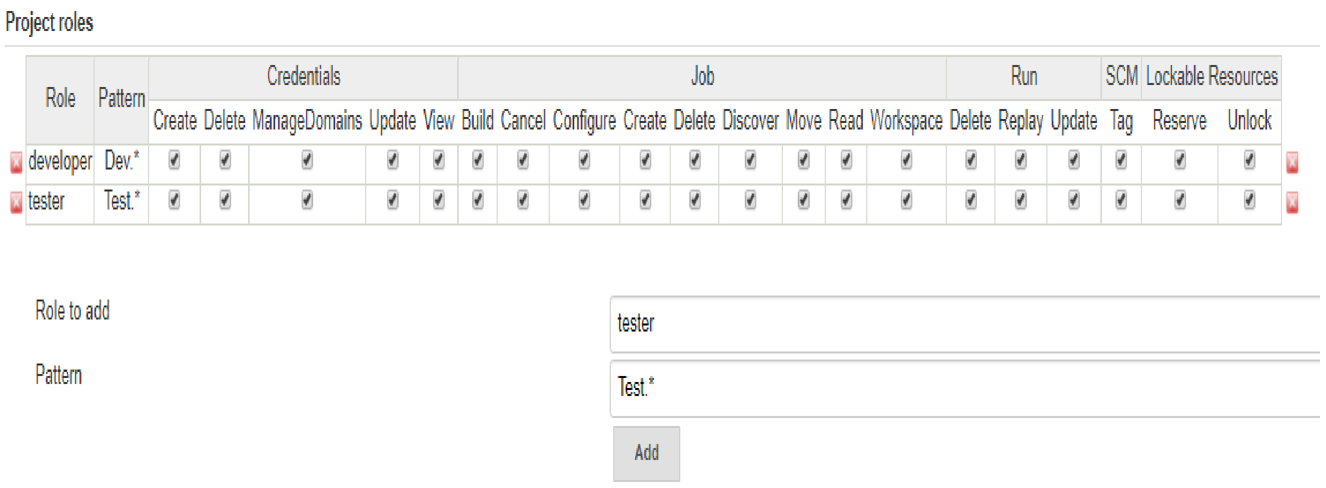
Role to add

Pattern

Add

Save Apply

In image: Here created a global Role (employee with access to read and view(complete)) and a project role(developer with access to all Dev.* files i.e files starting with Dev)


Project roles

Role	Pattern	Credentials					Job									Run			SCM	Lockable Resources	
		Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag	Reserve	Unlock
developer	Dev.*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
tester	Test.*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

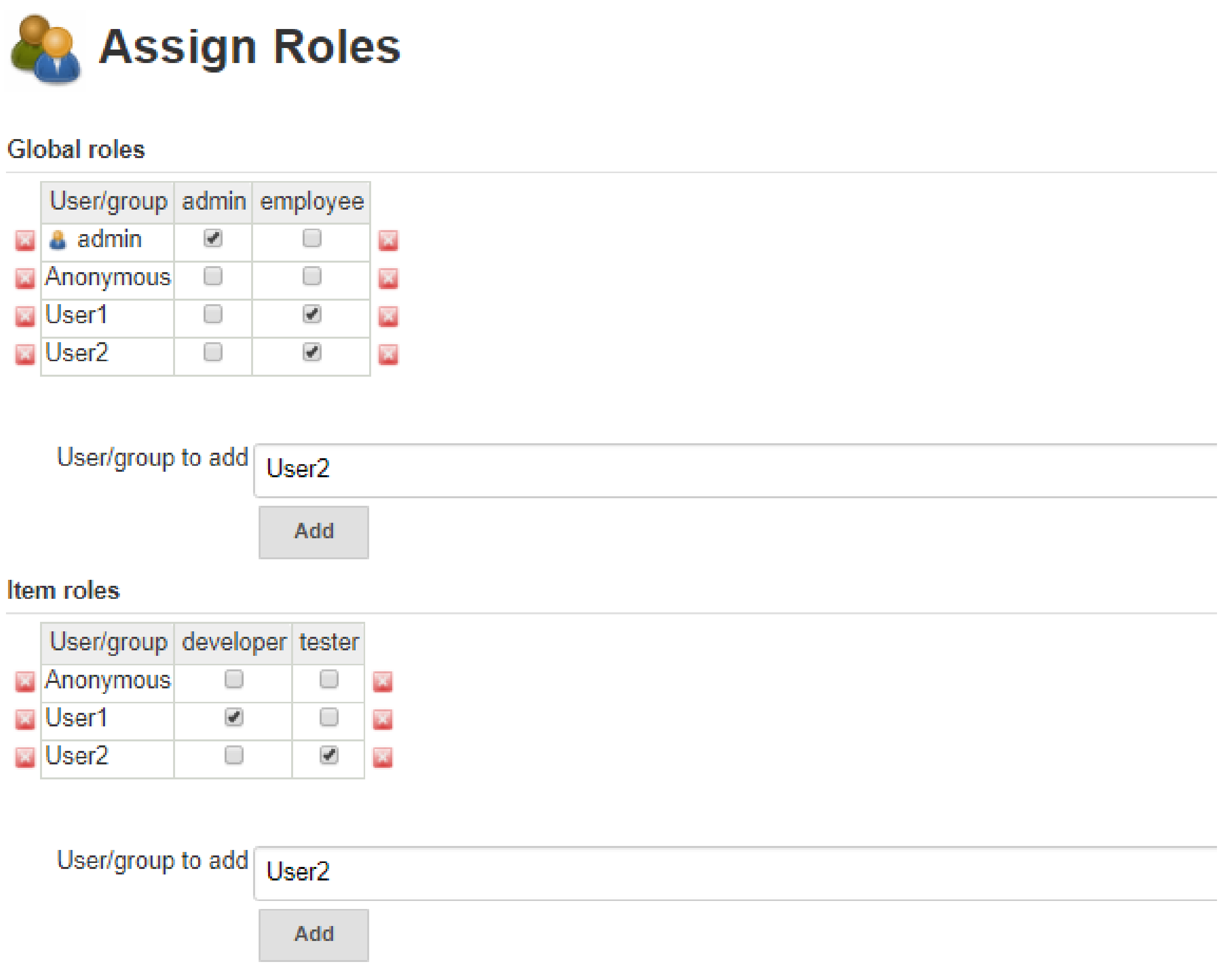
Role to add: tester

Pattern: Test.*

Add

In image: added one more project role named tester with access to all Test.* file

Now assigning the roles



Assign Roles

Global roles

User/group	admin	employee
admin	✓	
Anonymous		
User1		✓
User2		✓

User/group to add: User2

Add

Item roles

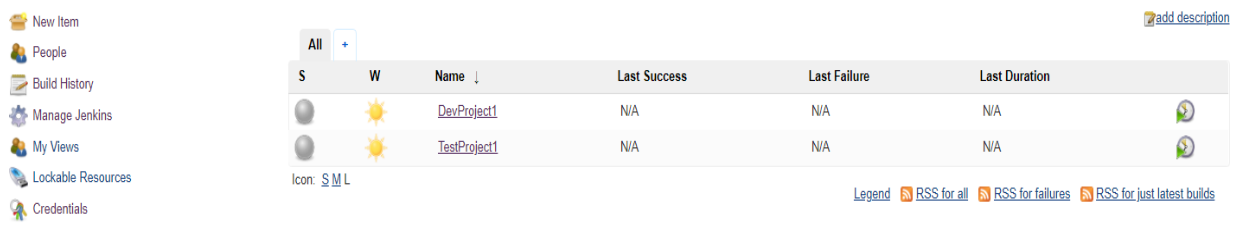
User/group	developer	tester
Anonymous		
User1	✓	
User2		✓

User/group to add: User2

Add

Image: Both User1 and User2 are employee but User1 is a developer (with access to Dev.* files) and User2 is a tester (with access to Test.* files)

Step 6: Validate authorization and authentication are working properly

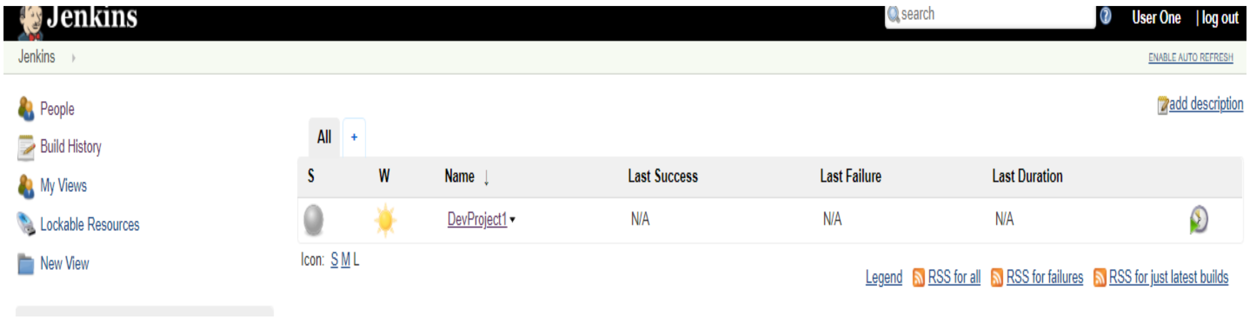


S	W	Name ↓	Last Success	Last Failure	Last Duration
		DevProject1	N/A	N/A	N/A
		TestProject1	N/A	N/A	N/A

Image: created two projects with Dev and Test prefix in names

Now login as User1 who is a developer and has access to Dev.* projects

(NOTE: for the User1 the global permission does not give permission to see manage Jenkins etc.)



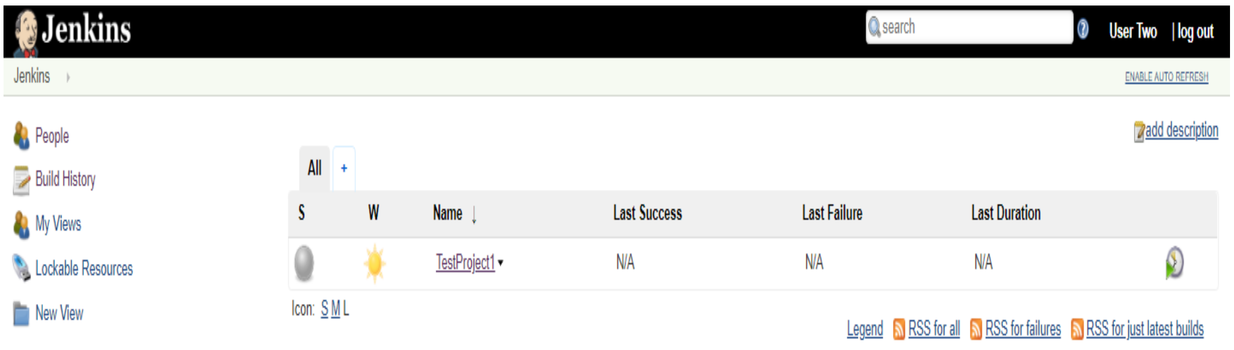
S	W	Name ↓	Last Success	Last Failure	Last Duration
		DevProject1	N/A	N/A	N/A

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Now login as User2 who is a tester and has access to Test.* projects

(NOTE: for the User2 also the global permission does not give permission to see manage Jenkins etc.)



S	W	Name ↓	Last Success	Last Failure	Last Duration
		TestProject1	N/A	N/A	N/A

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

4) How to create a new JOB ?

Click on new item->enter a name->select from templates->then follow the steps

Jenkins Integration with git scm :

1. Create a python program and run it through command line

2. Create a Jenkins job to run the python program by going in new item and then in build triggers -> execute windows batch command and copy the following:

cd path_of_name.py_file

python name.py

and then build the item.

3. Add this program/project to Git

4. Jenkins - add git plugin .Now create new Project in Jenkins , and configure it as:

Source Code Management: Git->Repositories->repository URL ->(paste the git file path)

Note: if error occurs goto Manage Jenkins->Global Tool Configuration->

Git->Path to Git executable->(here give the git.exe full path where it is installed in your pc) ->apply and save

5. Configure Jenkins job to trigger the execution when a change is pushed to GitHub

Build Triggers : Poll SCM (to trigger the job on any push in the github repo)

Apply and Save

6.Now make some changes in git repository by updating and pushing the file and notice that the job in Jenkins is automatically triggered is set in configure part.

5) What is catlight and how to use it?

CatLight is a notification app for developers. It shows the current status of continuous delivery, tasks and bugs in the project and informs when attention is needed.

6) What is Automated Deployment ?

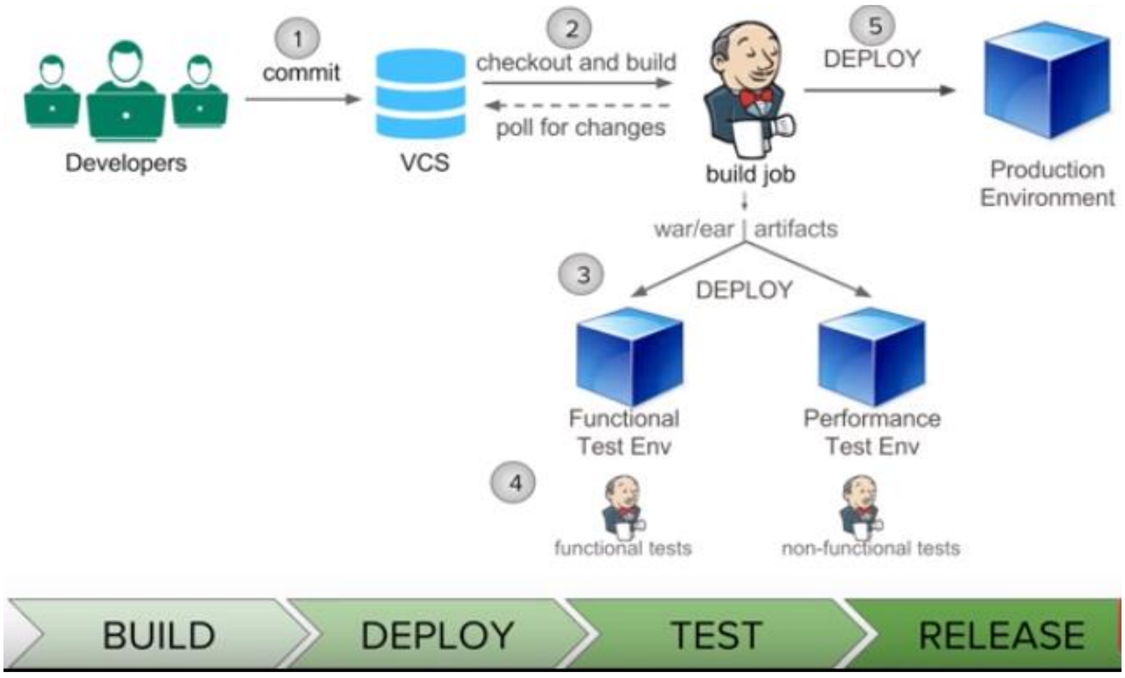
It is the process of automating the deployment process in CDS.Main stages in Continuous delivery and deployment pipeline :

a. BUILD

b. DEPLOY

c. TEST

d. RELEASE



7) How to do automated deployment ?

Step 1: Run Jenkins

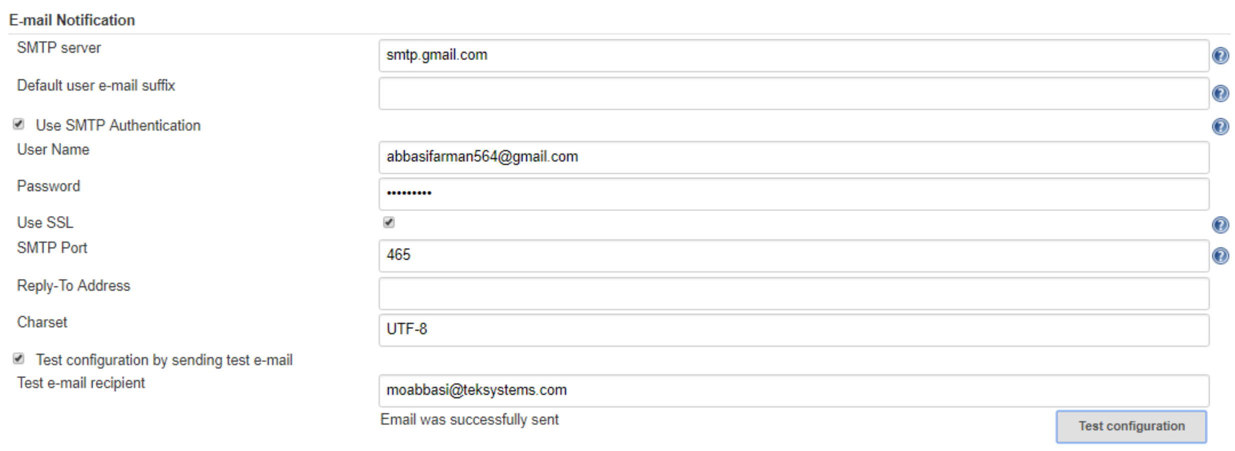
Step 2: Install plugin called : Deploy to container Plugin

Step 3: Create a new project (we can use one from internet for testing type : sample war file download in google) and copy the sample.war file in .jenkins folder-> workspace -> inside the project name we have created.

Note: war/ear file is created as a result of a build job here we are downloading one just for practice

Step 4:configure Post build Action note(we want tomcat running for this)

8) How to send email?



E-mail Notification

SMTP server	smtp.gmail.com
Default user e-mail suffix	
☑ Use SMTP Authentication	
User Name	abbasifarman564@gmail.com
Password	•••••••••
Use SSL	☑
SMTP Port	465
Reply-To Address	
Charset	UTF-8
☑ Test configuration by sending test e-mail	
Test e-mail recipient	moabbasi@teksystems.com

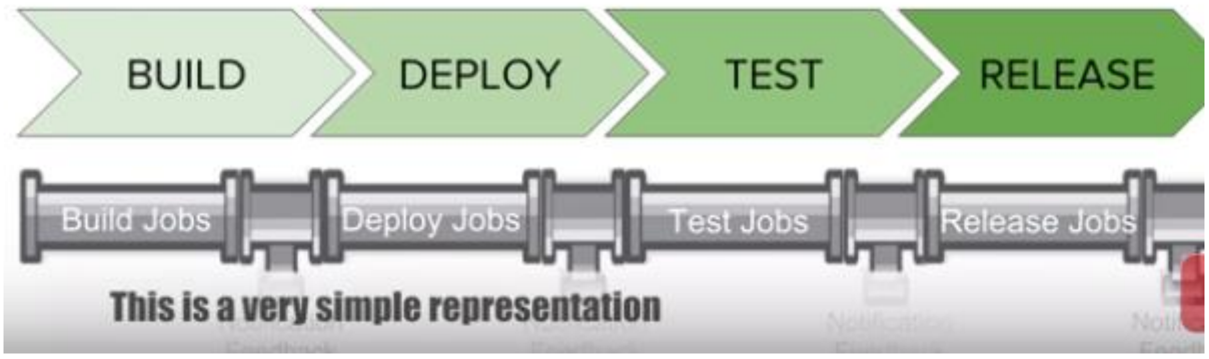
Email was successfully sent

Note: goto gmail sign in options and set less secure applications on for the sender email id

And then email can be configured for particular jobs->configure->post build actions-> email-notification

9) What is Pipeline in Jenkins ?

Pipeline - is a workflow with group of events or jobs that are chained and integrated with each other in sequence. Every job in a pipeline has some dependency on one or more other jobs

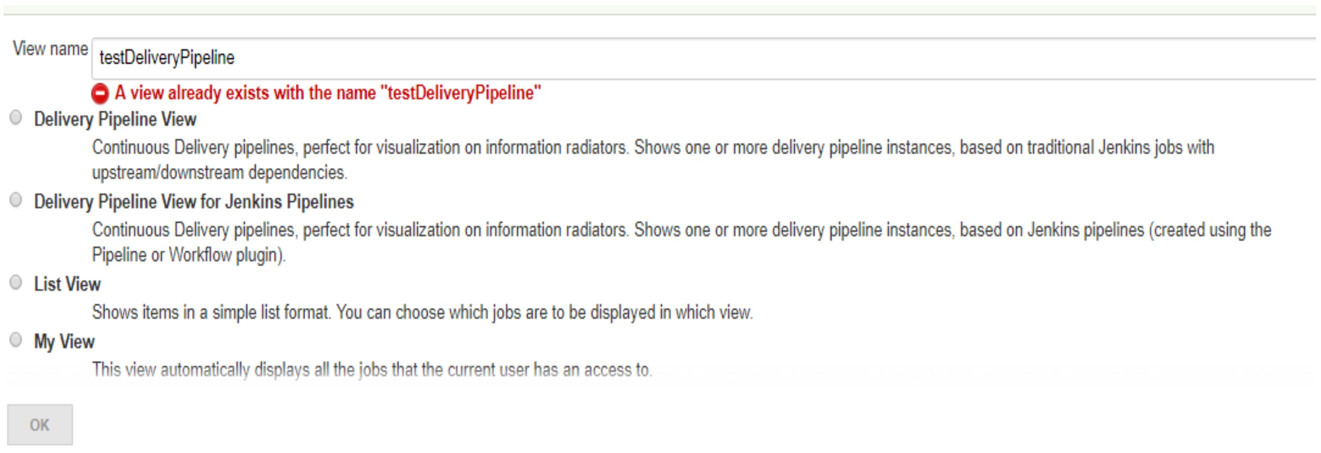


10) How to setup Delivery Pipeline in Jenkins?

Step 1 Chain required jobs in sequence Add upstream/downstream jobs

Step 2 Install Delivery Pipeline Plugin

Step 3 Add Delivery Pipeline View and configure the view



View name testDeliveryPipeline

A view already exists with the name "testDeliveryPipeline"

Delivery Pipeline View

Continuous Delivery pipelines, perfect for visualization on information radiators. Shows one or more delivery pipeline instances, based on traditional Jenkins jobs with upstream/downstream dependencies.

Delivery Pipeline View for Jenkins Pipelines

Continuous Delivery pipelines, perfect for visualization on information radiators. Shows one or more delivery pipeline instances, based on Jenkins pipelines (created using the Pipeline or Workflow plugin).

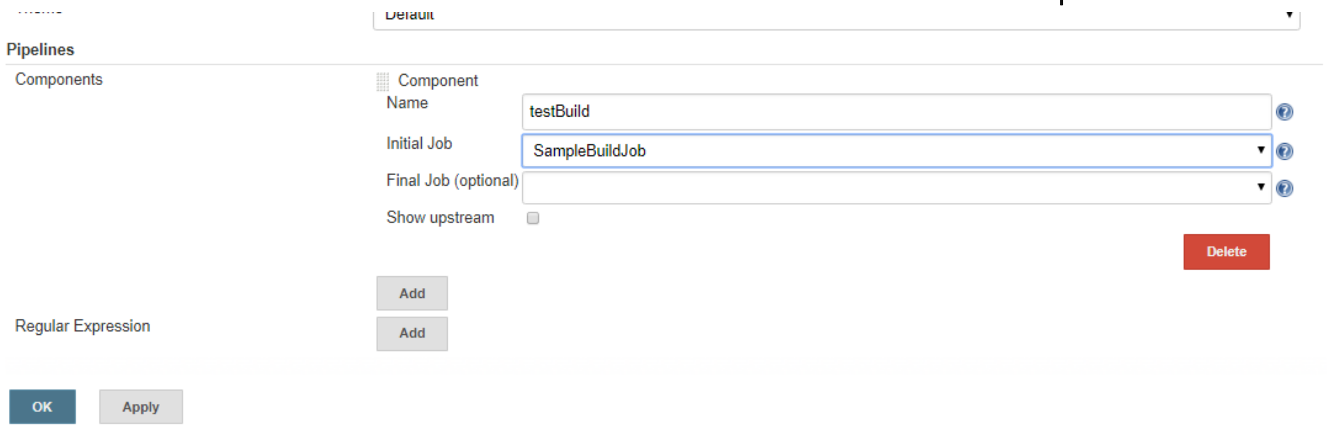
List View

Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View

This view automatically displays all the jobs that the current user has an access to.

Then next add a component



Pipelines

Components

Component

Name: testBuild

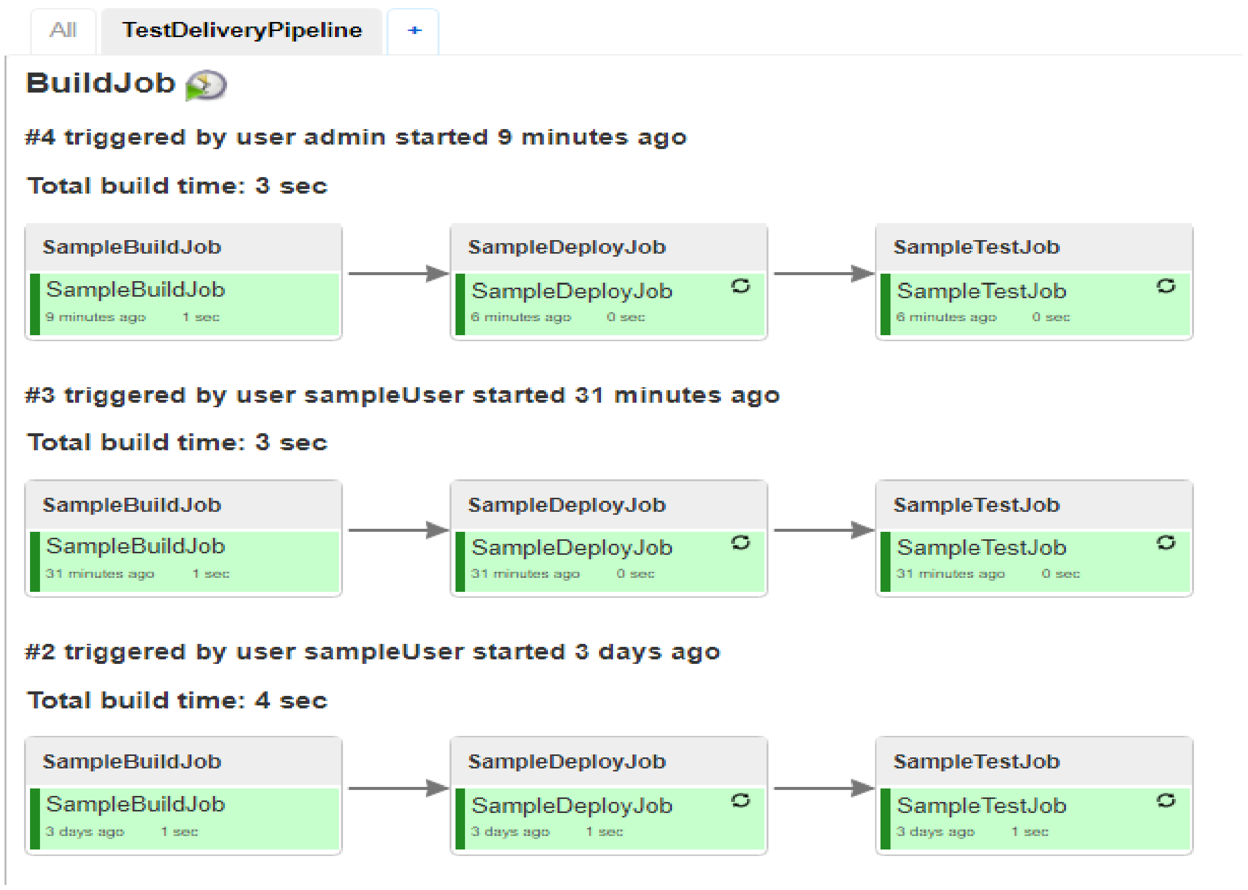
Initial Job: SampleBuildJob

Final Job (optional):

Show upstream: ☐

Regular Expression

Step 4 Run and Validate



BuildJob

#4 triggered by user admin started 9 minutes ago

Total build time: 3 sec

SampleBuildJob — SampleBuildJob, 9 minutes ago, 1 sec → SampleDeployJob — SampleDeployJob, 6 minutes ago, 0 sec → SampleTestJob — SampleTestJob, 6 minutes ago, 0 sec

#3 triggered by user sampleUser started 31 minutes ago

Total build time: 3 sec

SampleBuildJob — SampleBuildJob, 31 minutes ago, 1 sec → SampleDeployJob — SampleDeployJob, 31 minutes ago, 0 sec → SampleTestJob — SampleTestJob, 31 minutes ago, 0 sec

#2 triggered by user sampleUser started 3 days ago

Total build time: 4 sec

SampleBuildJob — SampleBuildJob, 3 days ago, 1 sec → SampleDeployJob — SampleDeployJob, 3 days ago, 1 sec → SampleTestJob — SampleTestJob, 3 days ago, 1 sec

11) How to setup build Pipelines in Jenkins?

Step 1 Chain required jobs in sequence Add upstream/downstream jobs

Step 2 Install Build Pipeline Plugin

Step 3 Add Build Pipeline View configure the view

Step 4 Run and Validate

12) Bitbucket pipelines vs Jenkins

Jenkins is written on groovy

Jenkins on the other hand requires some management. Somebody needs to be knowledgeable enough to install it, configure the necessary plugins, and configure the agent(s)

Jenkins is incredibly flexible in what it can do

Bitbucket is on cloud and uses docker image

Bitbucket requires almost zero management effort

Bitbucket Pipelines to work wonderfully for smaller projects that just need a basic build-test-deploy-forget pipeline.

13)Using Maven

Download maven .zip file and unzip in a folder.

Craete PATH in environment variables.

Now open cmd and type : **mvn –help to** to see if working fine

Type : mvn clean install to clean it from previous testings

14)Using DSL plugin

Install job DSL plugin from manage plugins

Create a new job which uses DSL groovy to create new job automatically

Configure the newly created job

And under build->Process Job DSLs ->use the provide DSL script

and type :

def jobName ='my-job'

job(jobName) {

}

This can be tested on link **:** http://job-dsl.herokuapp.com/

Save and apply

And on building this project a project called **my-job** will be created automatically