

Jenkins

What is jenkins

Jenkins is a automation tool used to automate and monitor various tasks

For jenkins we need to tell what it has to do, when it has to do and where it has to do.

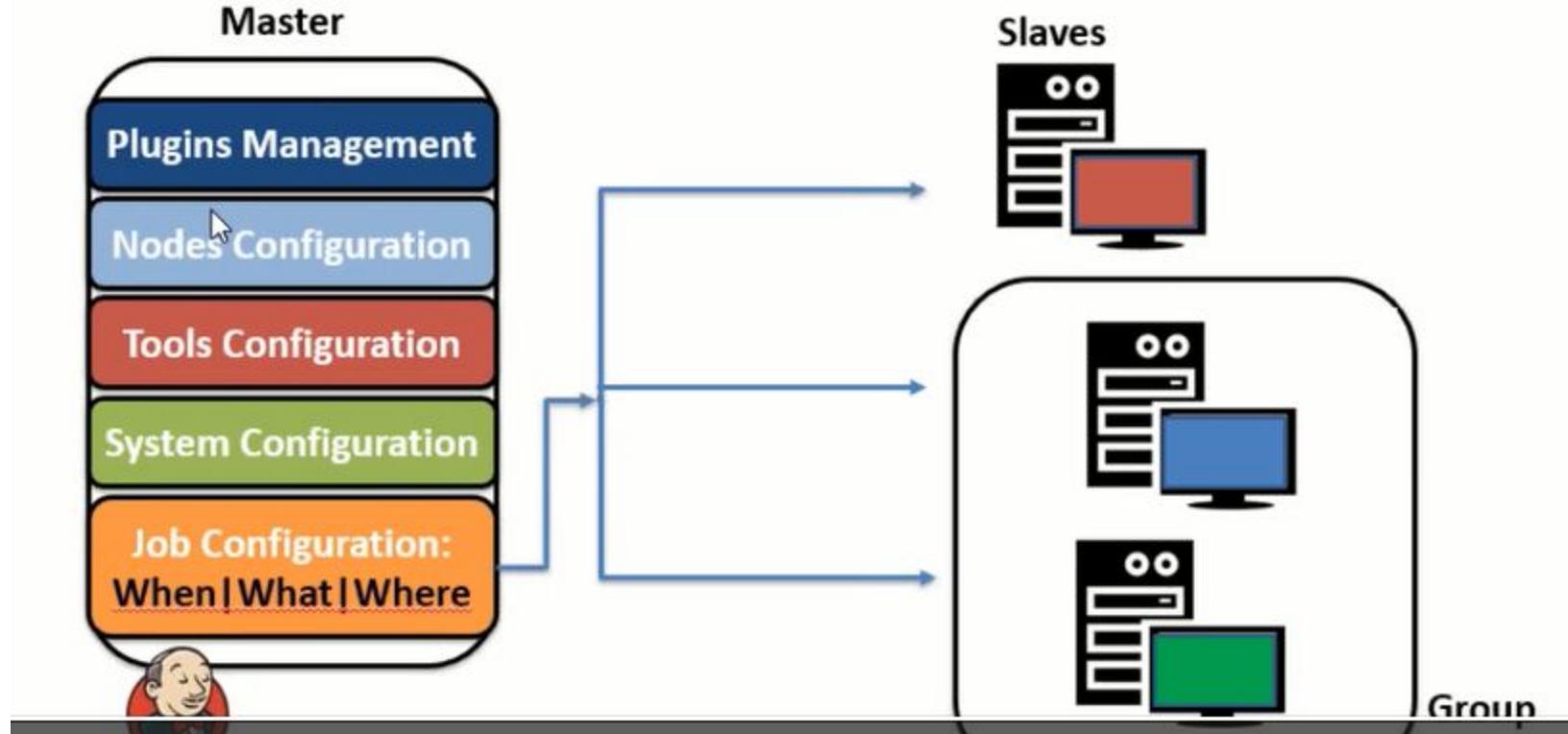
That job will be like infrastructure automating job or configuration management job or it will build job or deployment job.

Advantages- dashboard(to automate and monitor will be at one place and we can know job are running and which jobs are not running status also we can find.

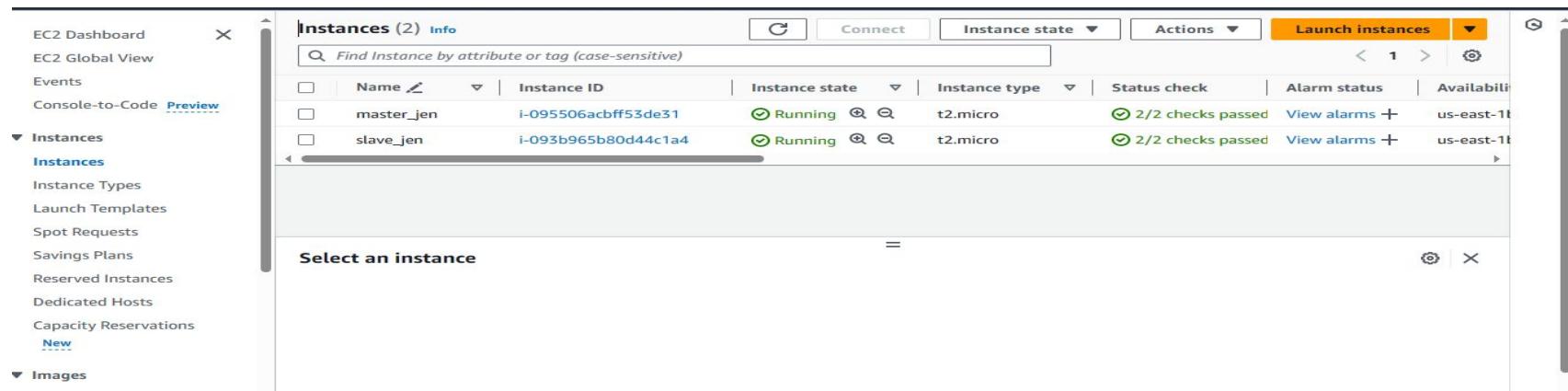
Group of machines to run the job- create label for that

Different plugins are available

Jenkins Architecture



Creating 2 instances in aws and by through ip address login to putty and install Java and all by using the link as reference: [Linux \(jenkins.io\)](https://jenkins.io) and commands as well



The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links like EC2 Dashboard, EC2 Global View, Events, Console-to-Code (Preview), Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations (New), and Images.

The main area displays a table titled "Instances (2) Info". The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. Two instances are listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
master_jen	i-095506acbff53de31	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1
slave_jen	i-093b965b80d44c1a4	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1

Below the table, a modal window titled "Select an instance" is open, showing the same two instances: master_jen and slave_jen.

```
# Download the Jenkins key and save it to /usr/share/keyrings/jenkins-keyring.asc
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

# Add the Jenkins repository to the list of trusted keys
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee
/etc/apt/sources.list.d/jenkins.list > /dev/null

# Update the package list
sudo apt-get update

# Install Jenkins
sudo apt-get install jenkins
```

1. Install jdk
2. Add repo key to the system(from above links)
3. Append the debian package
4. Install jenkins
5. Check the status of it –all these steps are common to both master and slave
6. To communicate master with slave we need to generate ssh from master
7. We need pem file for that but we have ppk file
8. Within master ,exit the root dir and give below command to generate private key
9. `ssh-keygen -t rsa`
10. Now copy that authorized key and go to slave and paste that key in below file
11. `vi ~/.ssh/authorized_keys`
12. To get host name to connect with master→ `hostname -i`

13. Now come back to master to connect that slave within master

Sudo apt install putty → create file to get ppk file that name must be matched

vim new_key.ppk

puttygen new_key.ppk -O privateOpenssh -o key.pem

Now file got converted from ppk to pem

ssh -i key.pem ubuntu@172.31.23.2 → slave come into master

After completion of operation exit from slave to master.

Once refer the steps in below page

```
with in master
1 exit
2 pwd
3 ssh-keygen -t rsa
4 ls -al
5 cd .ssh/
6 ls
7 cat authorized_keys    copy it and paste it in slave

with in slave
1. exit
2. vi ~/.ssh/authorized_keys  ---> append the key from master
3. hostname -i

with in master

1 sudo apt update
2 sudo apt install putty
3 vim new_key.ppk  --> new_key is downloaded name only we need to mention
4 puttygen new_key.ppk -O private-openssh -o key.pem
5 ssh -i key.pem ubuntu@172.31.23.2
```

Login to jenkins page

By through public ip address:8080 we can login to jenkins home page through user name and password.

The screenshot shows the Jenkins Home page. On the left, there's a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below that is a 'Build Queue' section stating 'No builds in the queue.' Further down is a 'Build Executor Status' section showing '1 Idle' and '2 Idle'. The main content area has a 'Welcome to Jenkins!' message, a 'Start building your software project' button, and a 'Create a job' button. It also features sections for 'Set up a distributed build', 'Set up an agent', and 'Configure a cloud'. At the bottom right, there's an 'Activate Windows' link.

Dashboard >

+ New Item

People

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

Activate Windows
Go to Settings to activate Windows.

Click on manage page

New Item

People

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Set up agent **Set up cloud** **Dismiss**

Java 11 end of life in Jenkins

You are running Jenkins on Java 11, support for which will end on or after Sep 30, 2024. Refer to [the documentation](#) for more details.

More Info **Ignore**

System Configuration

System Configure global settings and paths.

Tools Configure tools, their locations and automatic installers.

Plugins Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds Add, remove, and configure cloud instances to provision agents on-demand.

Activate Windows
Go to Settings to activate Windows.

plugins

To manage the plugins

Go to plugins

Advanced- if network is not available at that time we need to upload jenkins through http proxy. Directly upload from downloads.so we need to download it before from jenkins page.

Installed- what are the plugins we already installed

Available- available plugins within jenkins we can install based on necessary

Plugins



git



Install

 Updates Available plugins Installed plugins Advanced settings

Install

Name ↴

Released

 Git server 99.va_0826a_b_cdfa_d

git Library plugins (for use by other plugins)

1 yr 4 mo ago

Allows Jenkins to act as a Git server.

 Pipeline: Deprecated Groovy Libraries 609.vd95673f149b_b

Miscellaneous

Hosting of Pipeline Groovy libraries inside a Jenkins Git server. **Deprecated**. Use [Pipeline: Groovy Libraries](#) instead. If you see this plugin installed just because you upgraded, you can probably uninstall it now. This plugin should only be used if you have historically *pushed* libraries to a Git server inside Jenkins.

1 yr 1 mo ago

This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. [Learn more](#).

 Git Pipeline for Blue Ocean 1.27.9[External Site/Tool Integrations](#) [User Interface](#)

Activate Windows

Go to Settings to activate Windows.

1 mo 17 days ago

System configurations

If we want to change any configurations like

Url of jenkins, urls of gitlab and environment variables etc.,

What ever be the changes we made those will be reflected globally to all jobs, nodes etc.

System

Home directory ?
By default, Jenkins stores all of its data in this directory on the file system
`/var/lib/jenkins`

System Message
This message will be displayed at the top of the Jenkins main page. This can be useful for posting notifications to your users

Plain text [Preview](#)

of executors
-

[Activate Windows](#)
Go to Settings to activate Windows.

tools

We need to add tools like git,jdk and maven as below commands

```
ubuntu@ip-172-31-46-45:~$ sudo su -  
root@ip-172-31-46-45:~# which java  
/usr/bin/java  
root@ip-172-31-46-45:~# ls -l /usr/bin/java  
lrwxrwxrwx 1 root root 22 Dec 19 06:31 /usr/bin/java -> /etc/alternatives/java  
root@ip-172-31-46-45:~# ls -l /etc/alternatives/java  
lrwxrwxrwx 1 root root 43 Dec 19 06:31 /etc/alternatives/java -> /usr/lib/jvm/java-11-openjdk-amd64/bin/java  
root@ip-172-31-46-45:~# cd /usr/lib/jvm/java-11-openjdk-amd64/  
root@ip-172-31-46-45:/usr/lib/jvm/java-11-openjdk-amd64# ls  
bin conf docs include jmods legal lib man release  
root@ip-172-31-46-45:/usr/lib/jvm/java-11-openjdk-amd64# which git  
/usr/bin/git  
root@ip-172-31-46-45:/usr/lib/jvm/java-11-openjdk-amd64# which mvn  
/usr/bin/mvn  
root@ip-172-31-46-45:/usr/lib/jvm/java-11-openjdk-amd64#
```

JDK installations

Add JDK

≡ JDK

Name

jdk-11

JAVA_HOME

/usr/lib/jvm/java-11-openjdk

⚠ /usr/lib/jvm/java-11-openjdk is not a directory on the Jenkins controller (but perhaps it exists on some agents)

Install automatically ?

Git installations

Git

X

Name

mygit

Path to Git executable ?

/usr/bin/git

Install automatically ?

Add Git ▾

Maven

Name

mvn_file

MAVEN_HOME

/usr/bin/mvn

⚠ /usr/bin/mvn is not a directory on the Jenkins controller (but perhaps it exists on some agents)

Install automatically ?

Add Maven

Save

Apply

Activate W
Cloud Foundry

nodes

We can also add nodes to jenkins

Defaultly it has master , if we didnot connect to any server then jenkins will run job in master itself.

Here server means node (machine) to run job in jenkins

The screenshot shows the Jenkins 'Nodes' page under 'Manage Jenkins > Nodes'. The page title is 'Nodes'. On the left, there are navigation links for 'Dashboard', 'Manage Jenkins', 'Nodes', 'Clouds', and 'Build Queue'. The 'Build Queue' section indicates 'No builds in the queue.' Below the title, there is a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. One row is present: 'Built-In Node' (Architecture: Linux (amd64), State: In sync, Free Disk Space: 5.02 GB, Free Swap Space: 0 B, Free Temp Space: 5.02 GB, Response Time: 0ms). At the bottom, there are sections for 'Build Executor Status' (1 Idle) and 'Build Queue' (2 Idle).

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.02 GB	0 B	5.02 GB	0ms

Now create the node

No. of executors - how many no. of jobs need to be executed in a machine

Remote root directory - folder where slave machines are there which is used by jenkins.create as below in this folder jenkins will run that job

```
root@ip-172-31-46-45:/usr/lib/jvm/java-11-openjdk-amd64# mkdir tmp  
root@ip-172-31-46-45:/usr/lib/jvm/java-11-openjdk-amd64# chmod 777 tmp/  
root@ip-172-31-46-45:/usr/lib/jvm/java-11-openjdk-amd64# |
```

Jenkins connects to machine and it checks executors are available or not if available then job runs otherwise it won't even though Jenkins connected to machine.

Create label means group of machines in a single group with name label

If any one of the jobs wants machine to run is possible only when that machine is under the same demo group otherwise it won't run the job.

Click on new node – new name – no. of executors – labels – remote root directory

Launch method through ssh

Host name — hostname -i from master

Credentials - (jenkins)

Kind-ssh with private key

And give name and add pem.key(private key) in credentials add it finally

Select manually trusted key verification.save it

We created one node click on that to get sync

If we want to create another node again take new node and use node1 in it to get all info to node1 which node 1 have

Creating jobs

Enter an item name

first_job
» Required field

 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a

OK

Activate Windows
Go to Settings to activate Windo

Where we want to run job — general

If we want to run the job of jenkins to connect any version control systems – SCM

When we want run the job (periodically,some scenarios) -- build trigger

What exactly we have to run – build

After u run the task what task we need to do – post build action

Job is created Build now Build history

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

Icon: S M L

Icon legend

Atom feed for all

Atom feed for failures

Atom feed for just latest builds

- [Workspace](#)
- [Build Now](#)
- [Configure](#)
- [Delete Project](#)
- [Rename](#)

Permalinks

- [Last build \(#1\), 26 sec ago](#)
- [Last stable build \(#1\), 26 sec ago](#)
- [Last successful build \(#1\), 26 sec ago](#)
- [Last completed build \(#1\), 26 sec ago](#)

[Build History](#) trend ▾

Filter builds... /

#1
Dec 20, 2023, 11:18 AM ↑ ↑

Click on build now to get console output



Console Output

```
Started by user admin
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins/workspace/first_job
[first_job] $ /bin/sh -xe /tmp/jenkins4146319082096168097.sh
+ echo -----
-----
+ echo hi friends
hi friends
+ echo -----
-----
Finished: SUCCESS
```

If we did not assign to any machine means it run on master and we can also assign as below where it will run

The screenshot shows the 'Restrict where this project can be run' section of a Jenkins project configuration. It includes three checkboxes: 'Throttle builds', 'Execute concurrent builds if necessary', and 'Restrict where this project can be run' (which is checked). Below this is a 'Label Expression' input field containing 'demo'. A note below states 'Label demo matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' At the bottom are 'Advanced' and 'Save' buttons, along with an 'Apply' button.

Throttle builds ?

Execute concurrent builds if necessary ?

Restrict where this project can be run ?

Label Expression ?

demo

Label demo matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Advanced ▾

Save Apply

Activate Wi
Go to Settings

Now the console output be like

```
Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/first_job
[first_job] $ /bin/sh -xe /tmp/jenkins15762561790500283325.sh
+ echo -----
-----
+ echo hi friends
hi friends
+ echo -----
-----
Finished: SUCCESS
```

When it will be run is decided by build triggers

Periodically

The screenshot shows a configuration interface for build triggers. At the top, there is a dropdown menu labeled "Build after other projects are built". Below it, a checkbox labeled "Build periodically" is checked, with a tooltip icon next to it. A "Schedule" button is also present. A text input field contains the cron expression "*****". A warning message below the schedule field states: "⚠ Do you really mean 'every minute' when you say '*****'? Perhaps you meant 'H *****' to poll once per hour". It also mentions the last run time and the next scheduled run time. There are three other trigger options listed below: "Build when a change is pushed to GitLab" (with a webhook URL), "GitHub hook trigger for GITScm polling", and "Bitbucket SCM".

Build after other projects are built

Build periodically ?

Schedule ?

⚠ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H *****" to poll once per hour
Would last have run at Wednesday, December 20, 2023 at 12:06:50 PM Coordinated Universal Time; would next run at Wednesday, December 20, 2023 at 12:06:50 PM Coordinated Universal Time.

Build when a change is pushed to GitLab. GitLab webhook URL: http://34.203.30.174:8080/project/first_job ?

GitHub hook trigger for GITScm polling ?

Bitbucket SCM ?

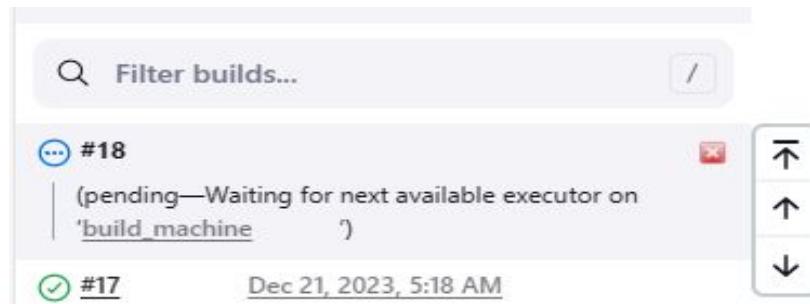
Activate Windows

0 12 21 1 1	-----> 0 min 12 hour 21 date 1 month monday
0 12 21 1 1,2,3,4,5	-----> 0 min 12 hour 21 date 1 month mon to fri
0 12 21-28 1 0-6	-----> 0 min 12 hour 21 to 28 date 1 month all week days
0 12 21-28 1 *	-----> as above
0 12 * 1 *	-----> 0 min 12 hour all days in a 1 month in all weeks
0 9,16 * 1*	-----> 0 min 9 and 4 time all days in a 1 month in all weeks
0 * * * *	-----> 0 min every hour
*/5 * * * *	-----> every min once
* * * * *	-----> every min

I have created one new job with sleep 30 statement in build ... if i run that second job as below

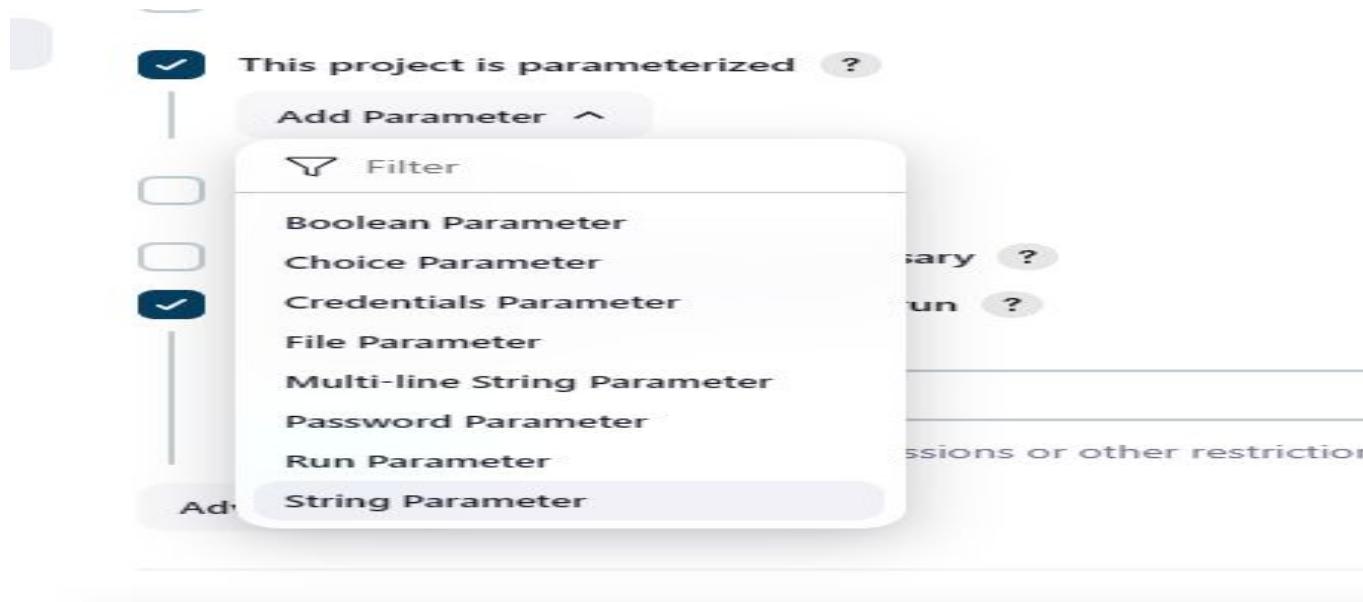


At the same time iam running first job i got below thing



If machine is not available then job need to be wait for use that machine to run the job

Now taking inputs from user and build or run the job With in general



If the job is periodic then we cant give input all the time so it takes default name

This project is parameterized ?

String Parameter ?

Name ?

user_string

Default Value ?

default_string

Description ?

enter the string

Plain text [Preview](#)



- 📁 Workspace
- ▷ Build with Parameters
- ⚙️ Configure
- 🗑 Delete Project
- ✍️ Rename

user_string
enter the string

jenkins

▷ Build

Cancel

Build Steps

Execute shell

Command

See [the list of available environment variables](#)

```
echo "string=$user_string"
```

The output will be like this



Console Output

```
Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/first_job
[first_job] $ /bin/sh -xe /tmp/jenkins8503828154197154283.sh
+ echo string=jenkins
string=jenkins
Finished: SUCCESS
```

We can also give choice as input by selecting choice parameterized

≡ Choice Parameter ?

Name ?

state

Choices ?

Possible
not possible

Description ?

enter the state

user_string

enter the string

default_string

state

enter the state

Possible

Possible

not possible

▷ Build

Cancel

The output will be like this

Console Output

```
Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/first_job
[first_job] $ /bin/sh -xe /tmp/jenkins8201237542472561967.sh
+ echo string=default_string --extra-vars state=Possible
string=default_string --extra-vars state=Possible
Finished: SUCCESS
```

Now select boolean to give true or false defaultlt it is true ...and edit it in build for print the output

≡ Boolean Parameter ?

Name ?

Set by Default ?

Description ?

want to dispaly true or false

Project first_job

This build requires parameters:

user_string

enter the string

default_string

state

enter the state

Possible



option

want to display true or false

The output will be like this

```
Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/first_job
[first_job] $ /bin/sh -xe /tmp/jenkins15871549887388270544.sh
+ echo string: default_string --extra-vars state=Possible
string: default_string --extra-vars state=Possible
+ echo option : true
option : true
Finished: SUCCESS
```

We can also give password

≡ **Password Parameter** ?

Name ?

password

Default Value ?

.....

Description ?

enter the password

This build requires parameters:

user_string

enter the string

default_string

state

enter the state

Possible



option

want to display true or false

password

enter the password



Concealed

Change Password

▷ Build

Cancel

The output will be like this

```
Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/first_job
[first_job] $ /bin/sh -xe /tmp/jenkins3377500542970597465.sh
+ echo string: default_string --extra-vars state=Possible
string: default_string --extra-vars state=Possible
+ echo option : true
option : true
+ echo password : nopassword
password : nopassword
Finished: SUCCESS
```

Post build actions

Select build other projects

Build other projects ?

Projects to build

second_job,

! No such project 'sec'. Did you mean 'second_job'?

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

- If the first build is not stable then the second build will not run depends upon given condition it will works.
- With in build option, if we give exit 1 as command then 1st build will not execute because exit 0 command only will represent that 1st build is stable other than that it will assume like 1st job will not stable.
- we can call other job within present job in this way
- here 2nd job is downstream

 **first_job**

The output will be like this

✓ Console Output

```
Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/first_job
[first_job] $ /bin/sh -xe /tmp/jenkins1754054900008258895.sh
+ echo string: default_string --extra-vars state=Possible
string: default_string --extra-vars state=Possible
+ echo option : true
option : true
+ echo password : nopassword
password : nopassword
Triggering a new build of second_job
Finished: SUCCESS
```

- let us assume that we have 2 jobs , as we know that 1st job calls 2nd job to execute.
- if 2nd job has parameterized ,individually we can give parameters while building 2nd job only but here we downstream it with 1st job
- we need parameterized trigger plugin so install it in dashboard
- first of all do string parameterized for second job
- after installed above plugin ,on post build actions select trigger parameterized build on other projects and select predefined parameters , give name to it based on second job variable name then it reflects it in that job.

Trigger parameterized build on other projects ?



Build Triggers

Projects to build ?

second_job,

! No such project 'second'. Did you mean 'second_job'?

Trigger when build is ?

Stable



Trigger build without parameters ?



Predefined parameters

Parameters ?

new_name=suma



The output will be like this for 2nd job

```
Started by upstream project "first_job" build number 26
originally caused by:
  Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/second_job
[second_job] $ /bin/sh -xe /tmp/jenkins15247226639377715764.sh
+ echo its running
its running
+ echo my name is suma
my name is suma
Finished: SUCCESS
```

→ we can also dynamically call the parameter which is declared in 1st job here
user_string is 1st job parameter

The screenshot shows a Jenkins job configuration page. At the top, there is a checkbox labeled "Trigger build without parameters" and a help icon. Below this is a section titled "Predefined parameters" with a "Parameters" link and a help icon. A parameter named "new_name=suma-\$User_string" is listed. There is also a red "X" icon in the top right corner of the parameter input field.

→ the output will be like this

```
+ echo its running
its running
+ echo my name is suma-default_string
my name is suma-default_string
Finished: SUCCESS
```

- whatever the parameters in 1st job those will be copied to second job but both parameters names must be matched with in 1st and 2nd jobs. For this select current build parameters in 1st job
- we need to remove predefined parameters in 1st job and copy the parameter in 1st job
- now add that parameter to 2nd job and update the build for printing statement.

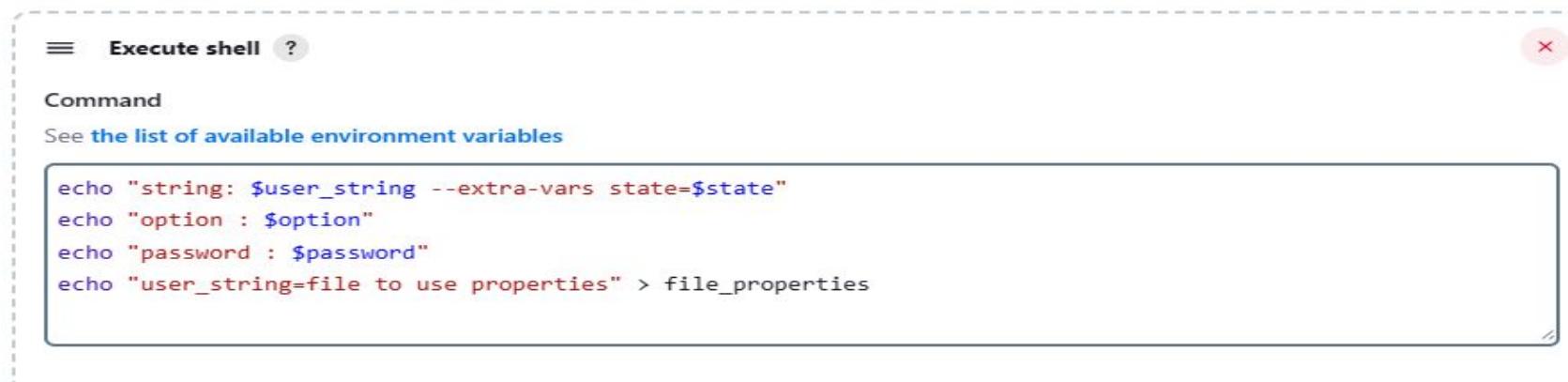
2nd job output will be like this

```
Started by upstream project "first_job" build number 31
originally caused by:
    Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/second_job
[second_job] $ /bin/sh -xe /tmp/jenkins6095129031414225303.sh
+ echo its running
its running
+ echo my name is noname
my name is noname
+ echo my name is hema
my name is hema
Finished: SUCCESS
```

By through file properties we can call parameters from 1st job to 2nd job

I dont have any file so create a file dynamically as below

Build Steps



The screenshot shows a build step configuration in a CI pipeline. The step is titled "Execute shell". The "Command" field contains the following shell script:

```
echo "string: $user_string --extra-vars state=$state"
echo "option : $option"
echo "password : $password"
echo "user_string=file to use properties" > file_properties
```

Now select that option in post build actions



Now run 1st job that is reflected to 2nd job and output will be



Console Output

```
Started by upstream project "first_job" build number 34
originally caused by:
  Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/second_job
[second_job] $ /bin/sh -xe /tmp/jenkins7439065573024615741.sh
+ echo its running
its running
+ echo my name is noname
my name is noname
+ echo my name is file to use properties
my name is file to use properties
Finished: SUCCESS
```

Within workspace we can see that file

Workspace of first_job on build_machine

first_job /

[file_properties](#)

Dec 27, 2023, 7:52:45 AM

35 B



[\(all files in zip\)](#)

First we have single parameter in 2nd job but after creating file and invoked it to 2nd job then 2nd job will have 2 parameters



Parameters

user_string

file to use properties

new_name

enter the string

noname

Build on same node option is also available in post build action

It is used to run the jobs which are in parent as well as child (1st and 2nd)on a same machine.

→the advantage of jenkins is to store all the history logs of the builds.

→go to putty and give cd var/lib/jenkins then we can get all folders, files plugins everything — to get only folders—ls -l |grep ^d

→we can see all jobs,plugins ,builds, logs , configuration files etc.,

Within general we can decide how many days build can be visible and how many latest build s can be visible as below

Discard old builds ?

Strategy

Log Rotation

Days to keep builds
if not empty, build records are only kept up to this number of days
2

Max # of builds to keep
if not empty, only up to this number of build records are kept
4

Advanced ▾

scm(source code management)

→it is used to clone the data from git hub or git lab and those data will be stored in workspace

Source Code Management

None

Git ?

Repositories ?

Repository URL ?

https://github.com/ch-hemalatha/repo



Credentials ?

hemalatha.chandaka@thundersoft.com/*****



+ Add ▾

Advanced ▾

Add Repository

Activate Windows

Give ls in build then output will be as below after ls we get no of files in that folder

```
Started by user admin
Running as SYSTEM
Building remotely on build_machine (demo) in workspace /tmp/jenkins/workspace/scm_job
The recommended git tool is: NONE
using credential git_repo
Cloning the remote Git repository
Cloning repository https://github.com/ch-hemalatha/repo
> /usr/bin/git init /tmp/jenkins/workspace/scm_job # timeout=10
Fetching upstream changes from https://github.com/ch-hemalatha/repo
> /usr/bin/git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_ASKPASS to set credentials
> /usr/bin/git fetch --tags --force --progress -- https://github.com/ch-hemalatha/repo +refs/heads/*:refs/remotes/origin/* # timeout=10
> /usr/bin/git config remote.origin.url https://github.com/ch-hemalatha/repo # timeout=10
> /usr/bin/git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> /usr/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision ff2a81055e63200608e93c47f5cf01a01e461b4 (refs/remotes/origin/master)
> /usr/bin/git config core.sparsecheckout # timeout=10
> /usr/bin/git checkout -f ff2a81055e63200608e93c47f5cf01a01e461b4 # timeout=10
Commit message: "dictionary methods"
> /usr/bin/git rev-list --no-walk ff2a81055e63200608e93c47f5cf01a01e461b4 # timeout=10
[scm_job] $ /bin/sh -xe /tmp/jenkins16098213995169580453.sh
+ ls
```

Activate Windows
Go to Settings to activate Windows.

- Now consider one master download java and jenkins in it and generate ssh key in it(ssh-keygen -t rsa).
- copy that id_rsa.pub which is in .ssh folder with in the master
- go to slave and open authorized key file and by append that id_rsa.pub and close it and come back to master
- with in master copy give ssh ubuntu@host id

Or

- instead of above 3 steps just give below command

Ssh-copy-id ubuntu@hostid

- same process for any no. of slaves connected to masterfor suppose i have created 2 slaves (2 machines)
- now coming to jenkins page through master public ip addr to create 2 nodes for 2 slave machines

Now create one job and use scm method by cloning git hub link and perform the operations.

Source Code Management

None

Git [?](#)

Repositories [?](#)

Repository URL [?](#)

`https://github.com/ch-hemalatha/repo`

Credentials [?](#)

ubuntu (server1)

[+ Add](#) ▾

Advanced ▾

Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```
ls  
echo "_____"
```

Now save it. The output will be like it is cloned all the files and list them because we have given ls command.

We have given 2 machines(created 2 nodes) for each server instance. Created single label for 2 machines. While running a job it checks for available machine out of 2 machines which are created under same group or label.

Periodically run the build

Specify the branch

Branch Specifier (blank for 'any') ? ×

Add Branch

We can specify periodically when the job will run

Build Triggers

- Trigger builds remotely (e.g., from scripts) [?](#)
- Build after other projects are built [?](#)
- Build periodically [?](#)

Schedule [?](#)

```
0 13 * * *
```

In build statements will be

[≡](#) Execute shell [?](#)



Command

See [the list of available environment variables](#)

```
echo "start build"  
ls  
echo "stop build"
```

Now output will be

```
Started by user admin
Running as SYSTEM
Building remotely on node1 (label1) in workspace /home/ubuntu/new_jen/workspace/scm_job1
```

```
First time build. Skipping changelog.
[scm_job1] $ /bin/sh -xe /tmp/jenkins946272408723872649.sh
+ echo start build
start build
+ ls
add.c
praciced cherry_pick command
practice of git amend command
+ echo stop build
stop build
Finished: SUCCESS
```

Within workspace also we can get all the files within that specifies branch

Workspace of scm_job1 on node1

scm_job1 /		→	
📁	.git		
📄	add.c	Dec 29, 2023, 7:29:45 AM	79 B
📄	practiced cherry_pick command	Dec 29, 2023, 7:29:45 AM	3.63 KB
📄	practice of git amend command	Dec 29, 2023, 7:29:45 AM	3.38 KB

 (all files in zip)

→ normally we get incremental build but we need full build for that we need to enable the option in build environment

Build Environment

- Delete workspace before build starts**
- Advanced ▾
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans
- Terminate a build if it's stuck
- With Ant ?

Jenkins pipeline

- pipeline is a group of jobs which are interlinked to each other in a sequence manner.
- jenkins pipeline is a combination of plugins that supports integration and implementation of continuous delivery
- pipeline provides a set of tools for simple and complex deliveries as code via pipeline domain specific language(groovy dsl)
- instead of building several jobs just code the entire workflow as single script called pipeline as code.

→jenkinsfile is a text file that stores the entire workflow as code and it can be checked it into a scm, this enables the developers to access, edit and check the code at any time.

→jenkinsfile can written in

1. Scripted - traditional way ,stricter groovy based syntax,written on jenkins UI interface
2. Declarative - more recent features, richer syntactical features,writing and reading pipeline code easier.

Pipeline Concepts

Pipeline:

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

Syntax:

```
Pipeline {  
    // DSL  
}
```

Node/Agent:

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

Syntax:

```
Pipeline {  
    agent {    // DSL      }  
}
```

Reason for going freestyle to pipeline

- in freestyle we can create multiple job individually but those are not easily reusable
- maintanence is also a prlbm .dedicated way of running and we cant customize
- instead of giving all job individually we can integrate all the jobs in a single pipeline code . then it will define where and what it has to be run by each job .
- we can decide easily and modify the configurations within the code.

jenkinsfile

```
Pipeline{           — set of freestyle jobs
  Agent { }        — machine where to run the job(it is global block available for all stages
  Stages{
    stage("name of the job"){
      Steps { }      — what task it will do
    }
    stage("name of job"){
      Steps {}        what task it will do
    }
  }
}
```

If we need to give diff agents for diff job jobs means within stages we have to give agents then it becomes local.

Example for pipeline job

Now create a job (pipeline)

Enter an item name

demo_pipeline
» Required field

Freestyle project
 This is the central feature of Jenkins. Jenkins will build your project, combinir for something other than software build.

Pipeline
 Orchestrates long-running activities that can span multiple build agents. Suit and/or organizing complex activities that do not easily fit in free-style job ty

Multi-configuration project
 Suitable for projects that need a large number of different configurations, su builds, etc.

OK

Cancel

Cancel contains the cancel button that stops created items in it. Help for creating things

Now we get 3 options only those are

Within pipe line we have 2 options

Through scm means take the jenkinsfile from git hub

Through script means we can directly write the script there only

Pipeline

Definition

Pipeline script

Pipeline script

Pipeline script from SCM

1

try sample Pipeline... ▾

Configure

General

Advanced Project Options

Pipeline

Now write the script manually. Save it and build now

Pipeline

Definition

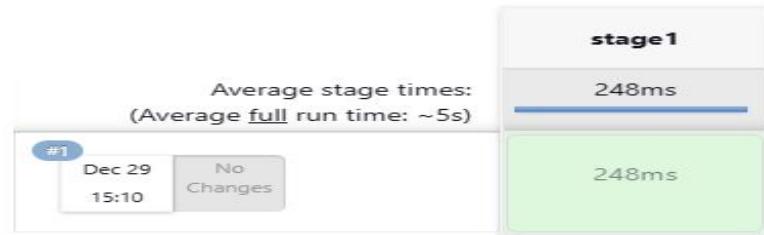
Pipeline script

Script ?

```
1 pipeline{
2     agent any      //it will take any available machine (@default master)
3     stages{
4         stage("stage1"){
5             steps {
6                 echo "first job"
7             }
8         }
9     }
10 }
```

try sample Pipeline... ▾

Stage View



The output will be like this

Each stages indicates each job

The console output will be like

```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/demo_pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] echo
first job
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

We can also see the output like below as well

The screenshot shows a CI pipeline interface with the following details:

- Stage Logs (stage1):** Displays the log entry: "Print Message -- first job (self time 11ms)" followed by "first job".
- Stage View:** Shows the execution times for the stage.
 - Average stage times: 248ms
 - (Average full run time: ~5s)
 - Build #1: Dec 29, 15:10, No Changes
 - Build #2: 248ms
- Actions:** Includes options to Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, Build History, and Filter builds... A "trend" dropdown is also present.
- Permalinks:** A section for sharing links.

Writing script for 2 jobs

Definition

Pipeline script

Script ?

```
1 < pipeline{
2     agent {label "label1"}
3     stages{
4         stage("stage1"){
5             steps {
6                 echo "first job"
7             }
8         }
9         stage("stage2"){
10            steps {
11                echo "second job"
12            }
13        }
14    }
15 }
```

```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (stage1) (hide)
[Pipeline] echo
first job
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
[Pipeline] echo
second job
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Diff agents with diff stages

Definition

Pipeline script

Script ?

```
1 ~ pipeline{
2     agent none
3     stages{
4         stage("stage1"){
5             agent {label "label1"}
6             steps {
7                 echo "first job"
8             }
9         }
10        stage("stage2"){
11            agent any
12            steps {
13                echo "second job"
14            }
15        }
16    }
17 }
```

Started by user admin

```
[Pipeline] Start of Pipeline
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] node
Running on node1 in /home/ubuntu/new_jen/workspace/demo_pipeline
[Pipeline] {
[Pipeline] echo
first job
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
[Pipeline] node
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
[Pipeline] echo
second job
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Instead of running 1st job on workspace it runs on customworkspace

```
Pipeline{  
    agent none  
    stages{  
        stage("stage1"){  
            agent {  
                node { label "label1"  
                    customWorkspace "/tmp"  
                }  
            }  
        }  
    }  
}
```

```
}

steps {
    echo "first job"
}

}

stage("stage2"){

    agent any

    steps {
        echo "second job"
    }

}

}
```

```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] node
Running on node1 in /home/ubuntu/new_jen/workspace/demo_pipeline
[Pipeline] {
[Pipeline] ws
Running in /tmp
[Pipeline] {
[Pipeline] echo
first job
[Pipeline] }
[Pipeline] // ws
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
[Pipeline] node
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
[Pipeline] echo
second job
```

Using environmental variables

```
pipeline {  
    agent { label 'demo' }  
    environment {  
        MYNAME = 'Adam'  
    }  
    stages {  
        stage('stage1') {  
            steps {  
                sh "echo 'Your name: $MYNAME'"  
            }  
        }  
        stage('stage2') {  
            steps {  
                echo env.MYNAME  
            }  
        }  
    }  
}
```

```
[Pipeline] node
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] sh
+ echo my name is hemalatha
my name is hemalatha
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/demo_pipeline
[Pipeline] {
[Pipeline] echo
hemalatha
[Pipeline] }
```

Example for local and global env variables

Script ?

```
1 -> pipeline{
2     agent {label "label1"}
3     environment{name = "global"}
4 ->     stages{
5         >         stage("stage1"){
6             environment{name = "local"}
7             steps {
8                 >                 sh "echo 'my name is $name' "
9             }
10            }
11        >         stage("stage2"){
12            agent any
13            steps {
14                >                echo env.name
15            }
16        }
17    }
18 }
```

First preference is given to local , if local is not available then it given to global

```
[Pipeline] sh
+ echo my name is local
my name is local
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/demo_pipeline
[Pipeline] {
[Pipeline] echo
global
[Pipeline] }
[Pipeline] // node
```

Using parameters

```
pipeline{
    agent {label "label1"}
    parameters{
        string(name:"person", defaultValue: "hema", description: "enter the name")
        text(name:"biography", defaultValue: " ", description: "enter the name")
        booleanParam(name:"toggle", defaultValue: "true", description: "enter the toggle value")
        choice(name:"CHOICE", choices: ["1","2","3"], description: "enter the choice")
        password(name:"pwd", defaultValue: "no_pwd", description: "enter the password")
        file(name:"file_prop",description: "enter the file name")
    }
    stages{
        stage("stage1"){
            steps {
                echo "hello ${params.person}"
                echo "biography : ${params.biography}"
                echo "boolean option : ${params.toggle}"
                echo "choice : ${params.CHOICE}"
                echo "password : ${params.pwd}"
                echo "filename : ${params.file_prop}"
            }
        }
    }
}
```

```
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] echo
hello hemalatha
[Pipeline] echo
biography : good biography
[Pipeline] echo
boolean option : true
[Pipeline] echo
choice : 2
[Pipeline] echo
Warning: A secret was passed to "echo" using Groovy String interpolation, which is insecure.
          Affected argument(s) used the following variable(s): [pwd]
          See https://jenkins.io/redirect/groovy-string-interpolation for details.
password : 98789
[Pipeline] echo
filename : null
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
```

→when we created pipeline job we have seen pipeline syntax,it is used generate syntax only for step block

The screenshot shows the Snippet Generator interface. On the left, a sidebar lists navigation options: Snippet Generator (selected), Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSL. The main content area has a heading '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.)'. Below this is a section titled 'Steps' with a 'Sample Step' example. A dropdown menu shows 'archiveArtifacts: Archive the artifacts'. The configuration for 'archiveArtifacts' includes fields for 'Files to archive' (with a placeholder '[]') and an 'Advanced' button. At the bottom is a large blue 'Generate Pipeline Script' button.

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

archiveArtifacts ?

Files to archive ?

Advanced ▾

Generate Pipeline Script

Activate Windows
Go to Settings to activate Windows

Option block

It will give lastest 5 builds

```
pipeline {  
    agent { label "label1" }  
    options {  
        buildDiscarder(logRotator(numToKeepStr: '5'))  
    }  
    stages {  
        stage("stage1") {  
            steps {  
                echo "first job"  
            }  
        }  
    }  
}
```

Full Stage View

Rename

Pipeline Syntax



Build History

trend ▾

Filter builds...

/



#

23

| Dec 29, 2023, 12:02 PM



#

22

| Dec 29, 2023, 12:00 PM



#

21

| Dec 29, 2023, 11:59 AM



#

20

| Dec 29, 2023, 11:57 AM



#

19

| Dec 29, 2023, 11:55 AM

Atom feed for all

Atom feed for failures

Average stage times: (Average full run time: ~720ms)		
#23	Dec 29 17:32	No Changes
#22	Dec 29 17:30	No Changes
#21	Dec 29 17:29	No Changes
#20	Dec 29 17:27	No Changes
#19	Dec 29 17:25	No Changes

Retry option

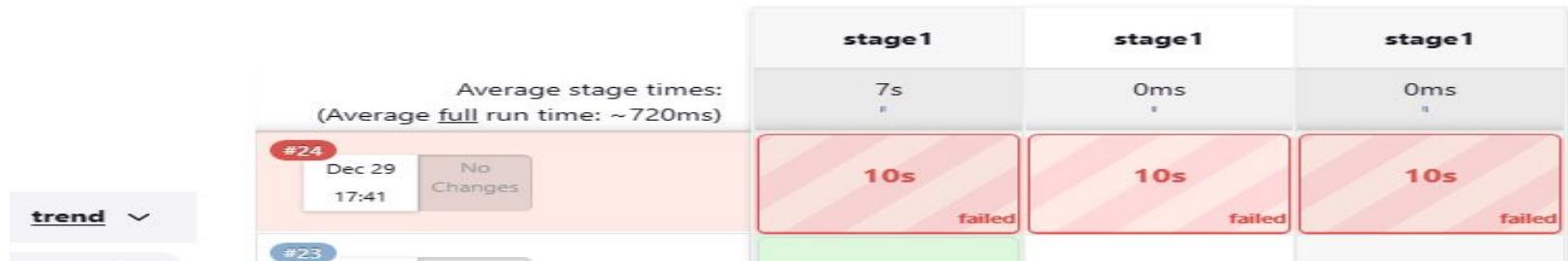
→ this option is not available in freestyle jobs

It retry for 3 times if stage is failed even

Though it fails it wont try again.

```
1 pipeline {  
2     agent { label "label1" }  
3     options {  
4         retry(3)  
5     }  
6     stages {  
7         stage("stage1") {  
8             steps {  
9                 sh "sleep 10;exit 1"  
10            }  
11        }  
12    }  
13 }
```

Stage View



2nd method

Script ?

```
1 -> pipeline {
2     agent { label "label1" }
3     options {
4         retry(3)
5     }
6     stages {
7         stage("stage1") {
8             steps {
9                 sh "echo hello"
10            }
11        }
12        stage("stage2") {
13            steps {
14                sh "sleep 10;exit 1"
15            }
16        }
17    }
18 }
```

	stage1	stage2	stage1	stage2	stage1	stage2
Average stage times:	366ms 	10s 	0ms 	0ms 	0ms 	0ms
#26 Dec 29 17:50 No Changes	382ms	10s failed	371ms	10s failed	346ms	10s failed
#25 Dec 29 17:49 No Changes					Activate Windows Go to Settings to activate Windows	

Timeout option

It runs the stage for specific time only after that it automatically terminates

script :

```
1 < pipeline {
2     agent { label "label1" }
3     options {
4         timeout(time: 5,unit: "SECONDS")
5     }
6     stages {
7         stage("stage1") {
8             steps {
9                 sh "sleep 10"
10            }
11        }
12    }
13}
14}
```

	stage1	stage2	stage1	stage2	stage1	stage2
Average stage times:	1s 	15s 	0ms 	0ms 	0ms 	0ms
#29 Dec 29 18:01 No Changes	5s aborted					
#28						

```
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
[Pipeline] timeout
Timeout set to expire in 5 sec
[Pipeline] {
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] sh
+ sleep 10
Cancelling nested steps due to timeout
Sending interrupt signal to process
Terminated
script returned exit code 143
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Timeout has been exceeded
org.jenkinsci.plugins.workflow.actions.ErrorAction$ErrorId: 7b88686d-d6be-4259-970c-f6c315ac1bb5
Finished: ABORTED
```

2nd method

```
1 > pipeline {
2     agent { label "label1" }
3     options {
4         timeout(time: 15,unit: "SECONDS")
5     }
6     stages {
7         stage("stage1") {
8             steps {
9                 sh "sleep 10"
10            }
11        }
12        stage("stage2") {
13            steps {
14                sh "sleep 10"
15            }
16        }
17    }
18 }
```

	stage1	stage2	stage1	stage2	stage1	stage2
Average stage times:	2s "	13s "	0ms "	0ms "	0ms "	0ms "
#31 Dec 29 18:05 No Changes	10s	5s aborted				
#30						

Timestamp option

```
1 > pipeline {  
2     agent { label "label1" }  
3 >     options {  
4         timestamps()  
5         timeout(time: 15,unit: "SECONDS")  
6     }  
7 >     stages {  
8         stage("stage1") {  
9             steps {  
10                sh "sleep 10"  
11            }  
12        }  
13         stage("stage2") {  
14             steps {  
15                sh "sleep 10"  
16            }  
17        }  
18    }  
19}
```

```
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
[Pipeline] timestamps
[Pipeline] {
[Pipeline] timeout
12:40:50 Timeout set to expire in 15 sec
[Pipeline] {
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] sh
12:40:51 + sleep 10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
[Pipeline] sh
12:41:01 + sleep 10
12:41:05 Cancelling nested steps due to timeout
12:41:05 Sending interrupt signal to process
12:41:06 Terminated
12:41:06 script returned exit code 143
[Pipeline] }
```

Disableconcurrent option

→ normal we can execute same job for multiple times simultaneously ... to avoid this we can use this option...if we run multiple times same job after given this option ... it is under pending only

```
1 > pipeline {  
2     agent { label "label1" }  
3     options {  
4         disableConcurrentBuilds()  
5         timestamps()  
6         timeout(time: 25,unit: "SECONDS")  
7     }  
8     stages {  
9         stage("stage1") {  
10            steps {  
11                sh "sleep 10"  
12            }  
13        }  
14        stage("stage2") {  
15            steps {  
16                sh "sleep 10"  
17            }  
18        }  
19    }
```

Stage View

 Delete Pipeline

 Full Stage View

 Rename

 Pipeline Syntax

 Build History

trend ▼

 Filter builds...

/

 #39

X

| (pending—Build #38 is already in progress (ETA: 10 sec))

 #38

X

| Dec 29, 2023, 12:53 PM

Average stage times:
(Average full run time: ~21s)

#38

Dec 29
18:23

No
Changes

stage1

5s

||

stage2

11s

||

stage1

0ms

||

stage2

0ms

||

stage1

0ms

||

stage2

0ms

||

2s

||

#37

Dec 29
18:23

No
Changes

10s

10s

#36

Dec 29
18:23

No
Changes

10s

10s

Activate Windows
Go to Settings to activate Windows

Using triggers

We can schedule it when it will run Below job will run every in every hour every date and ever week day in every month

Script ?

```
1 - pipeline {
2     agent { label "label1" }
3     triggers {
4         cron('* * * * *')
5     }
6     stages {
7         stage("stage1") {
8             steps {
9                 sh "echo 'hello' "
10            }
11        }
12    }
13 }
14 }
```

In build triggers(poll scm) for freestyle job

It triggers the job when new commit come in to git hub repository otherwise it wont build the job

For every min it search for new commit and get logs of it based on it.

The screenshot shows a Jenkins job configuration page. At the top left, there is a checked checkbox labeled "Poll SCM" with a question mark icon. Below it, there is another section labeled "Schedule" with a question mark icon. A large text input field contains the cron expression "*****".

⚠ Do you really mean "every minute" when you say "***"? Perhaps you meant "H *****" to poll once per hour**

Would last have run at Sunday, December 31, 2022 at 7:00:00 AM Coordinated Universal Time; would next run at Sunday, December 31,

 Status

 Changes

 Workspace

 Build Now

 Configure

 Delete Project

 Git Polling Log

 Rename

 Build History

trend ▼

 Filter builds...

Git Polling Log

Started on Dec 31, 2023, 7:29:00 AM

Using strategy: Default

[poll] Last Built Revision: Revision 3185d3b2430c14eaeb57f2fe6c935586480e4b1f (refs/remotes/origin/git-amend-command)

The recommended git tool is: NONE

using credential server1

> git --version # timeout=10

> git --version # 'git version 2.34.1'

using GIT_SSH to set credentials server1

Verifying host key using known hosts file

You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.

> git ls-remote -h -- <https://github.com/ch-hemalatha/repo> # timeout=10

Found 2 remote heads on <https://github.com/ch-hemalatha/repo>

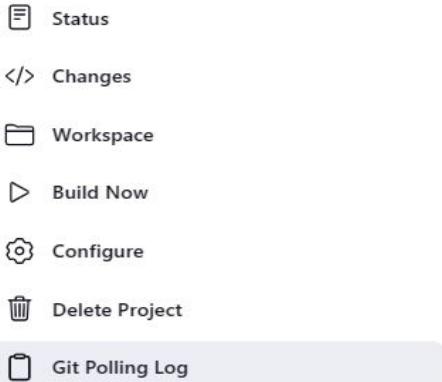
[poll] Latest remote head revision on refs/heads/git-amend-command is: 3185d3b2430c14eaeb57f2fe6c935586480e4b1f - already built by 4

Done. Took 0.15 sec

No changes

Activate Windows

For suppose if i made any changes in git repo then the next min output will be like this

 Status
</> Changes
Workspace
▷ Build Now
⚙ Configure
Delete Project
Git Polling Log
Rename

 Build History **trend** ▾
Filter builds... /
#5

Git Polling Log

```
Started on Dec 31, 2023, 7:40:00 AM
Using strategy: Default
[poll] Last Built Revision: Revision 3185d3b2430c14eaeb57f2fe6c935586480e4b1f (refs/remotes/origin/git-amend-command)
The recommended git tool is: NONE
using credential server1
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_SSH to set credentials server1
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.
> git ls-remote -h -- https://github.com/ch-hemalatha/repo # timeout=10
Found 2 remote heads on https://github.com/ch-hemalatha/repo
[poll] Latest remote head revision on refs/heads/git-amend-command is: 14de11d1605e620a29bcb6a27a6d6b4830e1364f
Done. Took 0.13 sec
Changes found
```

Activate Windows
Go to Settings to activate Window

→similar to freestyle job pollscm option is there in pipeline job

script :

```
1 < pipeline {  
2     agent { label "label1" }  
3     triggers{  
4         pollSCM('* * * * *')  
5     }  
6     stages {  
7         stage("stage1") {  
8             steps {  
9                 sh "echo 'hello' "  
10            }  
11        }  
12    }  
13}  
14 |
```

Upstream option

It checks the upstream's last build is success then it build the downstream build

Scm_job1 is upstream job it is successful then this pipeline job is triggered

Script ?

```
1 > pipeline {
2     agent { label "label1" }
3 >     triggers{
4         upstream(upstreamProjects:'scm_job1',threshold:hudson.model.Result.SUCCESS)
5     }
6 >     stages {
7         stage("stage1") {
8             steps {
9                 sh "echo 'hello' "
10            }
11        }
12    }
13}
14}
```

- in freestyle we can clone the git hub repo and build the job
- in same way pipeline also have an option to clone the git hub link
- but in freestyle first we need to clone and build but in pipeline we can clone anywhere or anytime based on req
- in pipeline we have pipeline syntax go through it to give url and all and get syntax of it.. In pipeline we have flexibility to call the steps wherever we want.

Now paste that syntax in which job u want that url.

In this we can generate script for git hub

Sample Step

git: Git



git ?

Repository URL ?

<https://github.com/ch-hemalatha/repo>

Branch ?

`*/git-amend-command`

Credentials ?

ubuntu (server1)



+ Add ▾

Include in polling? ?

Activate Windows
Go to Settings to activate Window

Include in changelog? ?

```
pipeline {
    agent { label "label1" }
    triggers{
        upstream(upstreamProjects:'scm_job1',threshold:hudson.model.Result.SUCCESS)
    }
    stages {
        stage("stage1") {
            steps {
                sh "echo 'hello' "
            }
        }
        stage("stage2") {
            steps {
                git changelog: false, credentialsId: 'server1', poll: false, url: 'https://github.com/ch-hemalatha/repo'
                sh 'ls'
            }
        }
    }
}
```

1st job completes then only it executes 2nd job

```
pipeline {
    agent { label "label1" }

    stages {
        stage("stage1") {
            steps {
                echo "1st job starts"
                build 'scm_job1'
                echo "1st job completed"
            }
        }
        stage("stage2") {
            steps {
                echo "running"
            }
        }
    }
}
```

2nd job executes even though 1st job will not

```
pipeline {
    agent { label "label1" }

    stages {
        stage("stage1") {
            steps {
                echo "1st job starts"
                build job:'scm_job1' , wait:false
                echo "1st job completed"
            }
        }
        stage("stage2") {
            steps {
                echo "running"
            }
        }
    }
}
```

Automatic mail

```
pipeline {  
    agent { label "label1" }  
    stages {  
        stage("mail") {  
            steps {  
                mail bcc:"",hi hema",cc:"",from:"",replyTo:"",subject:'Test Mail',to:  
            }  
        }  
    }  
}
```

Changing directory

Normally jobs are executed or build under jenkins workspace folder if we want to change the folder as per our req then go to change the directory in steps

```
pipeline {
    agent { label "label1" }

    stages {
        stage("stage1") {
            steps {
                sh 'mkdir dir1' //this directory will be created in jenkins workspace
                dir('/tmp/lenkins/'){ //this will be created in new directory
                    sh 'mkdir dir2'
                }
                sh 'mkdir dir3' //this directory will be created in jenkins workspace
            }
        }
    }
}
```

Catch error option

It is like exception handling in python

→within first job if we got any error means it exit from pipeline it wont execute 2nd job

→so avoid it we have catcherror blockwe can see the syntax in pipeline syntax as well.

→in the below way we can generate syntax

Sample Step

catchError: Catch error and set build result to failure



catchError ?

Message ?

Error

Build result on error ?

UNSTABLE



Stage result on error ?

FAILURE



Catch Pipeline interruptions ?

Generate Pipeline Script

```
catchError(buildResult: 'UNSTABLE', message: 'Error', stageResult: 'FAILURE') {  
    // some block
```

Activate Windows

Go to Settings to activate Window

```
pipeline {
    agent { label "label1" }

    stages {
        stage("stage1") {
            steps {
                //if we comment out below line then entire build will not trigger
                //if we uncomment below line then stage1 failes it thoroughs error msg and it buils stage2 as well
                //catchError(buildResult: 'UNSTABLE', message: 'Error', stageResult: 'FAILURE'){
                    sh "exit 1"
                }
            }
        }
        stage("stage2") {
            steps {
                echo "this is 2nd stage"
            }
        }
    }
}
```

65 -without catcherror block

66 - with catcherror block



Based on when condition env variables are printed

```
pipeline {
    agent { label "label1" }
    environment { deploy_to = "dep" }
    stages {
        stage("stage1") {
            when {
                environment name: "deploy_to", value:"dep"
            }
            steps {
                echo "1st job starts"
            }
        }
        stage("stage2") {
            when {
                environment name: "production", value:"pro"
            }
            steps {
                echo "2nd jobrunning"
            }
        }
    }
}
```

Only first stage got executed if we change the condition means 2nd stage got executed in above scripts.

```
[Pipeline] 
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] echo
1st job starts
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
Stage "stage2" skipped due to when conditional
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
```

When condition based on boolean parameters

```
pipeline {
    agent { label "label1" }
    parameters {
        booleanParam(name:"toggle",defaultValue:false,description:"toggle this value")
        //if it is true means build normally, false means it wont build
    }
    stages {
        stage("stage1") {
            when {
                expression{return params.toggle}
            }
            steps {
                echo "1st job starts"
            }
        }
    }
}
```

Output will be like this based on false condition

```
Running on node2 in /home/ubuntu/new_jen2/workspace/demo_pipeline
[Pipeline] {
  [Pipeline] stage
  [Pipeline] { (stage1)
    Stage "stage1" skipped due to when conditional
  [Pipeline] }
  [Pipeline] // stage
  [Pipeline] }
  [Pipeline] // node
  [Pipeline] End of Pipeline
Finished: SUCCESS
```

When condition based on string parameters

Based on condition (string matches) it is executed ..if it doesnot match stage got skipped.

```
1 pipeline {
2     agent { label "label1" }
3     parameters {
4         string(name:"person",defaultValue:"hema",description:"who r u?")
5     }
6     stages {
7         stage("stage1") {
8             when {
9                 equals expected:"hemalatha",actual:params.person
10            }
11            steps {
12                echo "hi hemalatha"
13            }
14        }
15    }
16 }
```

If strings are not match then stage got executed otherwise if strings got matched stage wont execute

```
pipeline {
    agent { label "label1" }
    parameters {
        string(name:"person",defaultValue:"hema",description:"who r u?")
    }
    stages {
        stage("stage1") {
            when {
                not{equals expected:"hemalatha",actual:params.person}
            }
            steps {
                echo "hi hemalatha"
            }
        }
    }
}
```

When condition with allOf block

→ if we are giving number of parameters like boolean string etc., we r going to use all those at a time with some condition..all those will be keep in allof block then it executed when all the parameters condition is satisfied ..if any one of the condition fails then entire build will not done

→ in below scenario both conditions true then only build performs

 allOf — and operator

Anyof — or operator if any one condition satisfies it generate the build

 For that step

```
pipeline {
    agent { label "label1" }
    parameters {
        string(name:"person",defaultValue:"hema",description:"who r u?")
        booleanParam(name:"toggle",defaultValue:false,description:"toggle this value")
    }
    stages {
        stage("stage1") {
            when {
                allOf{
                    equals expected:"hemalatha",actual:params.person
                    expression{return params.toggle}
                }
            }
            steps {
                echo "hi hemalatha"
            }
        }
    }
}
```

anyOf option

```
pipeline {
    agent { label "label1" }
    parameters {
        string(name:"person",defaultValue:"hema",description:"who r u?")
        booleanParam(name:"toggle",defaultValue:false,description:"toggle this value")
    }
    stages {
        stage("stage1") {
            when {
                anyOf{
                    equals expected:"hemalatha",actual:params.person
                    expression{return params.toggle}
                }
            }
            steps {
                echo "hi hemalatha"
            }
        }
    }
}
```

Parallel execution of stages

- normally stages within pipeline are executed in sequential manner
- if we want to execute stages in parallel manner then proceed the below way



```
pipeline {
    agent { label "label1" }
    stages {
        stage("stage1") {
            steps {
                sh "sleep 10"
            }
        }
        stage("stage2") {
            steps {
                sh "sleep 10"
            }
        }
        stage("stage3") {
            parallel {
                stage("parallel 1") {
                    steps {
                        sh "sleep 10"
                    }
                }
                stage("parallel 2") {
                    steps {
                        sh "sleep 10"
                    }
                }
            }
        }
    }
}
```

Post build actions

Post

- The post section defines one or more additional steps that are run upon the completion of a Pipeline's or stage's run (depending on the location of the post section within the Pipeline)
- Post can support any of the following post-condition blocks:
 - **always:** Steps are executed regardless of the completion status.
 - **changed:** Executes only if the completion results in a different status than the previous run.
 - **fixed:** Executes only if the completion is successful and the previous run failed
 - **regression:** Executes only if current execution fails, aborts or is unstable and the previous run was successful.
 - **aborted:** Steps are executed only if the pipeline or stage is aborted.
 - **failure:** Steps are executed only if the pipeline or stage fails.
 - **success:** Steps are executed only if the pipeline or stage succeeds.
 - **unstable:** Steps are executed only if the pipeline or stage is unstable.

Post build with always option for entire pipeline



```
pipeline {  
    agent { label "label1" }  
    stages {  
        stage("stage1") {  
            steps {  
                echo "2nd stage"  
            }  
        }  
        stage("stage2") {  
            steps {  
                echo "2nd stage"  
            }  
        }  
    }  
    post {  
        always {  
            echo "post build actions"  
        }  
    }  
}
```

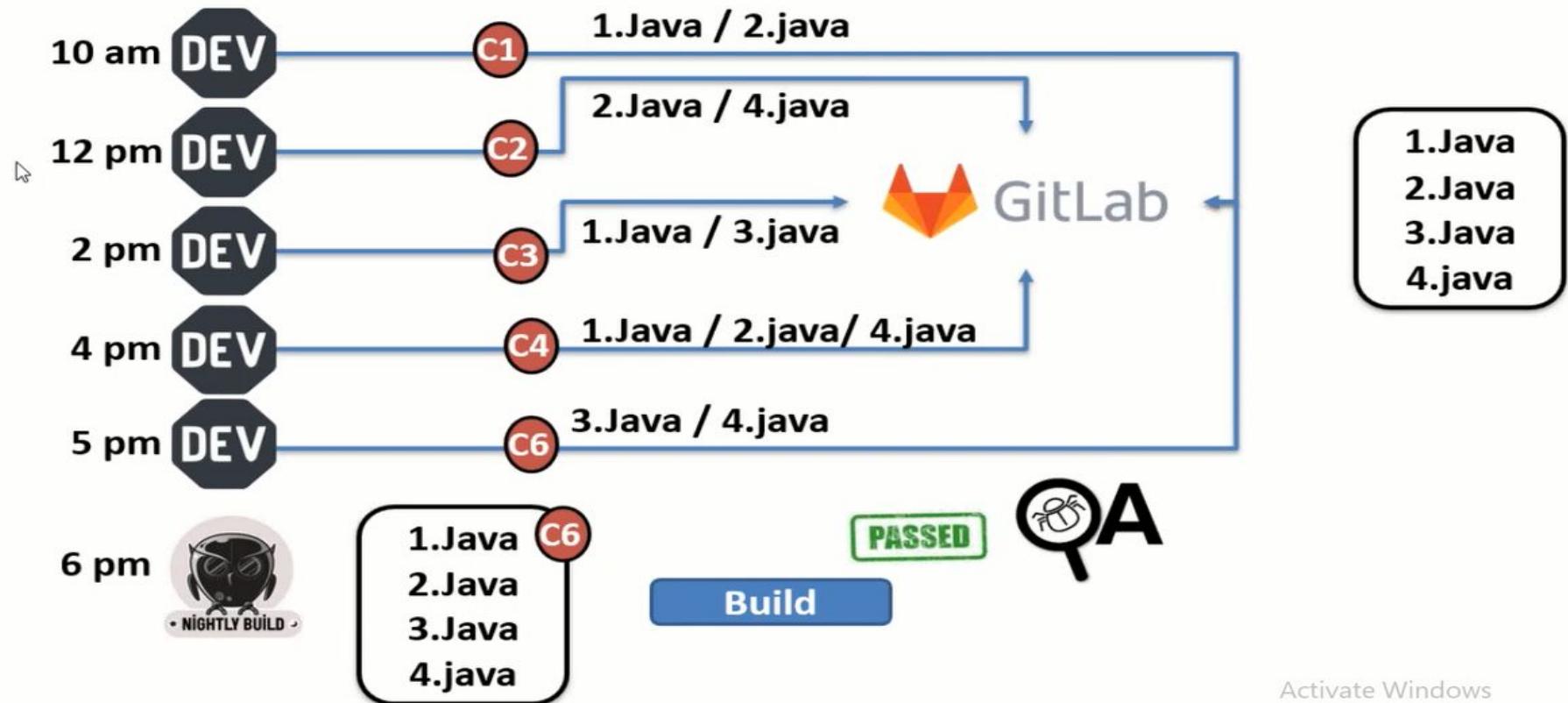
Always option for individual stages

Stage view



```
pipeline {  
    agent { label "label1" }  
    stages {  
        stage("stage1") {  
            steps {  
                echo "2nd stage"  
            }  
            post {  
                always {  
                    echo "post build actions"  
                }  
            }  
        }  
        stage("stage2") {  
            steps {  
                echo "2nd stage"  
            }  
            post {  
                always {  
                    echo "post build actions"  
                }  
            }  
        }  
    }  
}
```

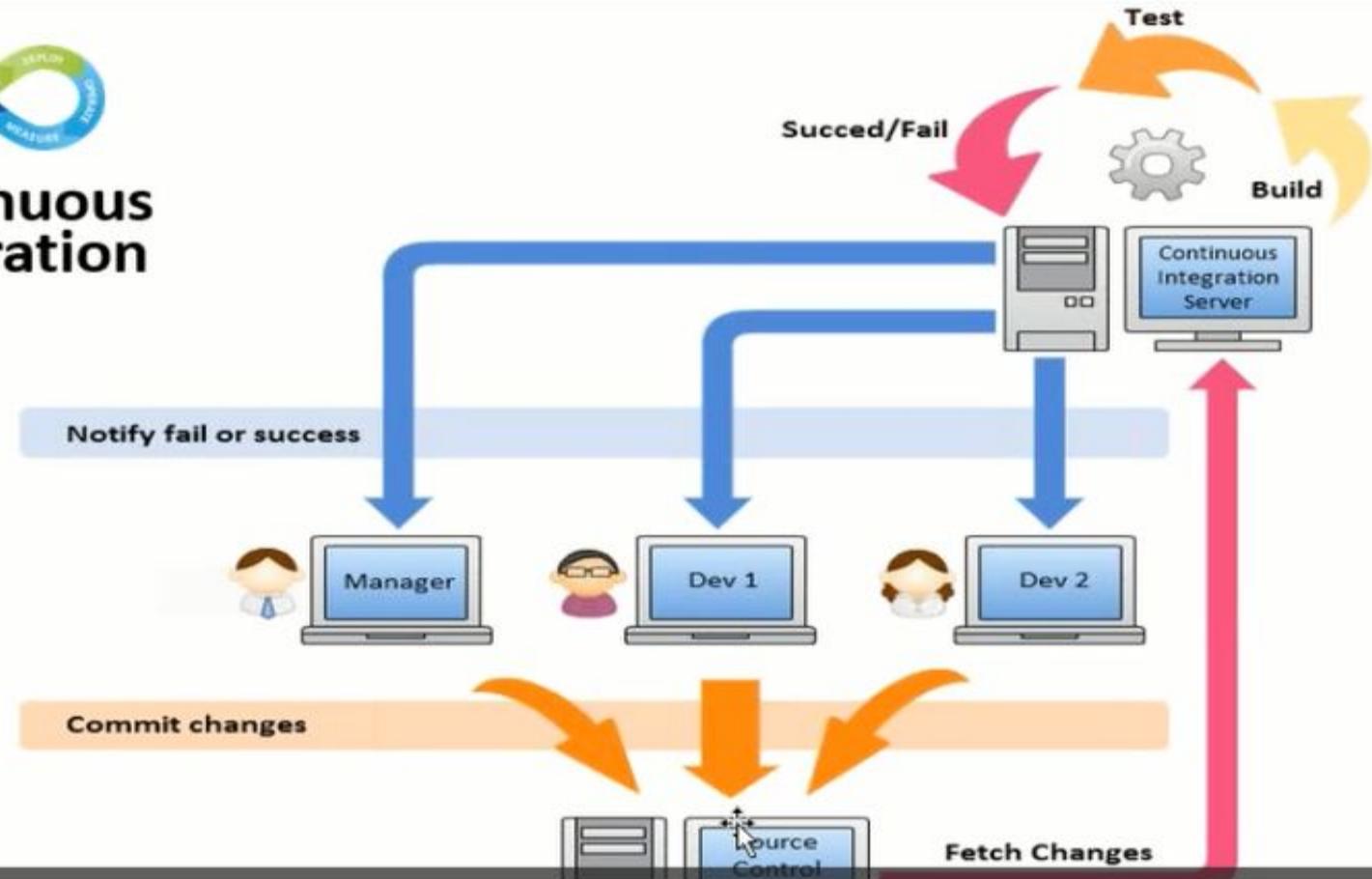
Continuous integration



- whenever the developer checks in , how we are going to tell him that his code is a good code which can be integrated with other code , that same thing will be given to QA.so that process where continuously integrate the developer codes into the actual repo is called as continuous integration.
- instead of doing a build once or twice a day and giving to QA ,in order to findout a bad commit or error in a code,do the build immediately whenever new code changes are check in or new push come within a repo.
- continuous integration helps us to identify error early,it helps us to do faster releases(qa team will not wait), developers have more time tofix the code



Continuous Integration



Continuous Integration



- Build Automation
- Developer Builds and QA Builds
- Implement Unit Tests
- Setup CI System
- Generate Build Artifacts
- Generate Code Coverage & Analysis
- Setup Build Artifactory System
- Publish results
- Maintenance

Tools: **Gitlab** – Git – JDK – Maven – Jenkins – JIRA – SonarQube – Artifactory – Scripting

Process to configure jenkins with github

→create webhook url by going into project settings and webhook and click on add webhook ----- <http://54.164.43.157:8080/github-web>(webhook) –within project setting–webhook

The screenshot shows the 'GitHub project' configuration section of a Jenkins job. A checkbox labeled 'GitHub project' is checked. Below it, a 'Project url' input field contains the URL 'https://github.com/ch-hemalatha/hello-maven.git/'. There is also an 'Advanced' dropdown menu.

In build triggers enable it

The screenshot shows the 'Build Triggers' configuration section of a Jenkins job. It lists three triggers: 'GitHub Full Requests', 'GitHub hook trigger for GITScm polling' (which is checked), and 'Poll SCM'.

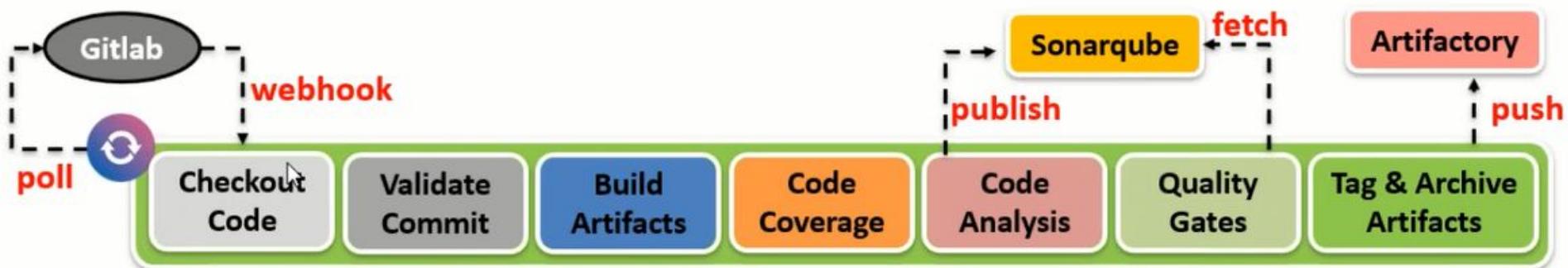
New commits checkin then build automatically

- write any script(lets simple echo script) in the pipeline option and save it
- now do the new commit in git hub then automatically build takes place
- the output will be as below



```
Started by GitHub push by ch-hemalatha
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/webhook_pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] git
```

Overview for CI



Code checkout from github repo

Any part of pipeline will take more than 1 hour for execution then it aborted.

Script

```
1 * pipeline {
2     agent none
3     options {
4         | timeout(time: 1, unit: 'HOURS')
5     }
6     stages {
7         stage('Checkout')
8     {
9         agent { label 'demo' }
10    steps {
11        | git credentialsId: 'GitlabCred', url: 'https://gitlab.com/scmlearningcentre/wezvatech-cicd.git'
12    }
13    }
14
15 }
16 }
17
18 }
```

Scenerio ---validate

Let us assume that take a project which checkout before

In that let us assume pom.xml and file1 are a dev codes then if we made any new changes with those files then only pipeline should trigger

Out of these files any other files may get new changes then pipeline will not trigger

If i did changes in read me file then output will be

```
[Pipeline] {
[Pipeline] }
[Pipeline] // node
Stage "validate" skipped due to when conditional
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] End of Pipeline
Finished: SUCCESS
```

If i did changes in file1

```
Running on Jenkins in /var/lib/jenkins/workspace/webhook_pipeline
[Pipeline] {
  [Pipeline] script
  [Pipeline] {
    [Pipeline] }
    [Pipeline] // script
    [Pipeline] }
    [Pipeline] // node
    [Pipeline] }
    [Pipeline] // stage
    [Pipeline] }
    [Pipeline] // timeout
  [Pipeline] End of Pipeline
Finished: SUCCESS
```

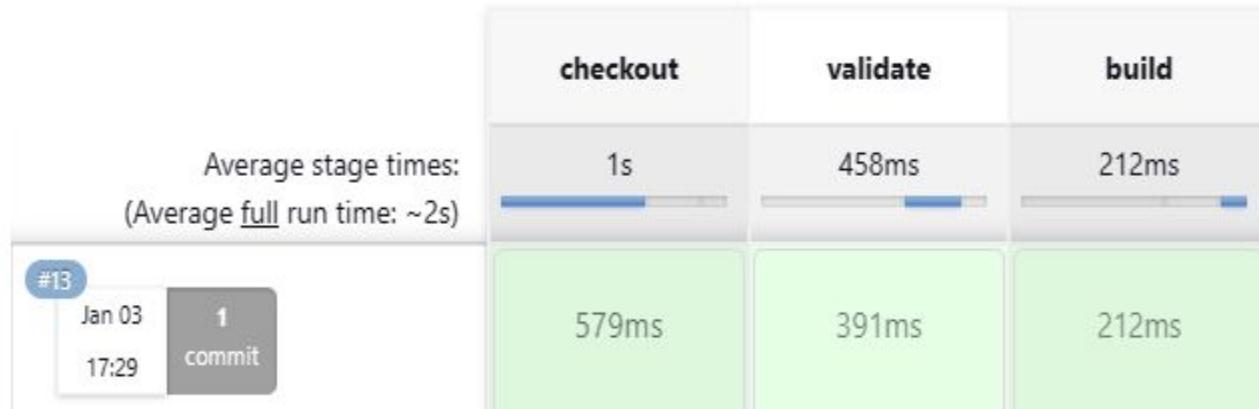
If there is no new commits then

```
Warning, empty changelog. Have you run checkout?  
[Pipeline] }  
[Pipeline] // node  
Stage "validate" skipped due to when conditional  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // timeout  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

Build artifacts

To check the validation whether it is working or not we are using build artifacts

If validation stage execute then only build stage executes which is failing but build artifacts



Using directory build artifacts

It enter into directory and run clean command

```
stage('Build')
{
    when {environment name: 'BUILDME', value: 'yes'}
    agent { label 'demo' }
    steps {
        echo "Building Jar Component ..."
        dir ("./samplejar") {
            sh "mvn clean package"
        }
        echo "Building War Component ..."
        dir ("./samplewar") {
            sh "mvn clean package "
        }
    }
}
```

For maven packages

Instead of maven clean command just give parameter and execute it

```
$ mvn package  
1. compile  
2. unit test  
3. package
```

```
$ mvn package -Dmaven.test.skip=true  
1. compile  
2. pack
```

Without unit test in maven packages we need to create parameters as below

```
parameters {  
    booleanParam(name: 'UNITTEST', defaultValue: true, description: 'Enable UnitTests ?')  
}  
.....
```

Within build we need to modify as below and compare with before scenario maven packages

.... pto.....

```
steps {
    script {
        |if (params.UNITTEST) {
            unitstr = ""
        } else {
            unitstr = "-Dmaven.test.skip=true"
        }

        echo "Building Jar Component ..."
        dir ("./samplejar") {
            sh "mvn clean package ${unitstr}"
        }

        echo "Building War Component ..."
        dir ("./samplewar") {
            sh "mvn clean package "
        }
    }
}
```

Multibranching pipeline

- create new multi branching job
- create new git hub repo
- in repo with in main/master create Jenkinsfile which is available for all branches
As like readme file
 - within jenkinfile just write some pipeline code as per our requirement.
 - create 5 branches within main ,all branches may contain readme and jenkins file as well

→ configure the multibranches pipeline as below, copy the url from git hub project and paste it as below

The screenshot shows the Jenkins Multibranch Pipeline configuration interface. The main section is titled "Branch Sources". Under "Git Project Repository", the URL `https://github.com/ch-hemalatha/multibranch.git` is entered. The "Credentials" section shows "- none -" selected. The "Behaviors" section contains a "Discover branches" option.

Branch Sources

Git Project Repository ?

`https://github.com/ch-hemalatha/multibranch.git`

Credentials ?

- none -

+ Add ▾

Behaviors

Discover branches ?

Save it

Automatically it got trigger the jobs

Dashboard > multibranching >

Status

Configure

Scan Multibranch Pipeline Now

Scan Multibranch Pipeline Log

Multibranch Pipeline Events

Delete Multibranch Pipeline

People

Build History

Project Relationship

Check File Fingerprint

Favorite

 **multibranching**

[Disable Multibranch Pipeline](#)

Branches (5)

S	W	Name ↓	Last Success	Last Failure	Last Duration	F
✓	☀️	branch1	19 sec #1	N/A	5.1 sec	 
✓	☀️	branch2	19 sec #1	N/A	5.1 sec	 
✓	☀️	dev	19 sec #1	N/A	6.8 sec	 
✓	☀️	main	19 sec #1	N/A	6.3 sec	 
✓	☀️	release	19 sec #1	N/A	6.3 sec	 

54.164.12.157:8080/cib/multibranching/

Activate Windows 

Go to Settings to activate Windows.

→within status we can see above triggered jobs

For example lets create a branch and rename jenkinsfile then it will not get trigger

Because jenkins file is not there

Refresh it check the status then that job wil not trigger in jenkins.

Code coverage

- **CodeCoverage** is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs
- It is a measurement of how many lines/blocks/arcs of your code are executed while an application is running

Benefits:

- Determine the amount of code being tested & not tested
- Eliminate Dead code, remove code which doesn't affect the program results
- Identifying additional test cases which might have been missed in the test suite
- Program with high code coverage has a lower chance of containing undetected bugs

Code-Coverage Tools : [JTest](#), [Cobertura](#), [Emma](#), [JCov](#), [JaCoCo](#)
