

## Jenkins class -3

We may feel burden to start the Jenkins server via executing every command step by step. So we can start the Jenkins with a single command by following steps.

We need to start the instance then create the file with **.sh** extension and enter all the commands which commands you have to execute in terminal. Then give the execution permission(`chmod +x file_name.sh`) to the file and execute it by following command

`./file_name.sh`

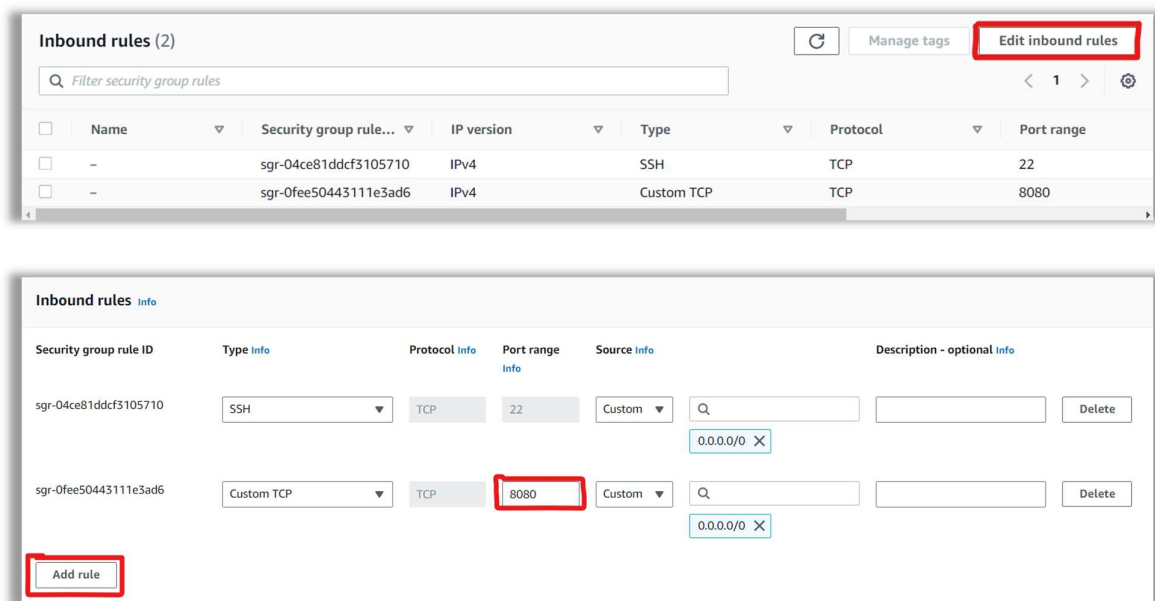
```
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

amazon-linux-extras install java-openjdk11 -y

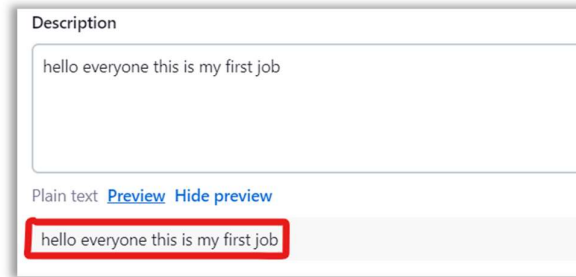
yum install jenkins -y

systemctl start jenkins
```

If the instance doesn't launch, then make sure the security groups are allowed the port 8080 in inbound rules. If we forget to add the port 8080 earlier, we can add it now. Just open the security group which you had selected to the current instance. Click on edit inbound rules then add the rule as port 8080 as shown below.

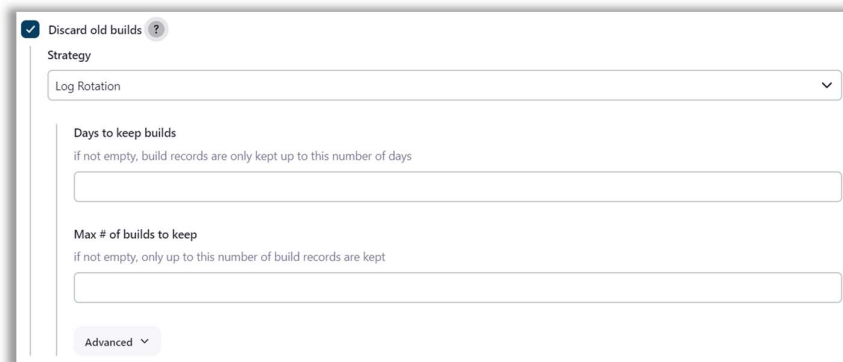


Once launched the Jenkins we can see the Description to describe anything related to the current job. If you want to preview it just click on preview option. It will reflect in below.



## **Discard old builds**

This option is used to discard the old build. Which means it will maintain the current active builds and discard the old build which is not in use.



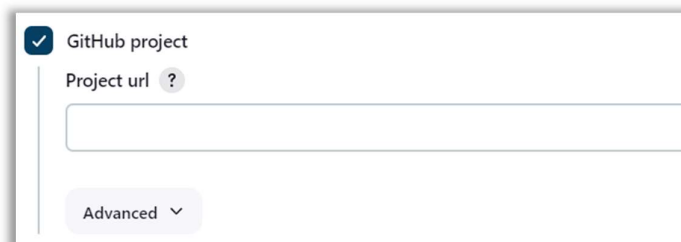
### **Days to keep builds**

This option is used to keep our set of build to be active for how long. For example, I want to keep last 8 days builds should be active, but no need to keep active previous builds. We can enter 8 in this field.

### **Max # of builds to keep**

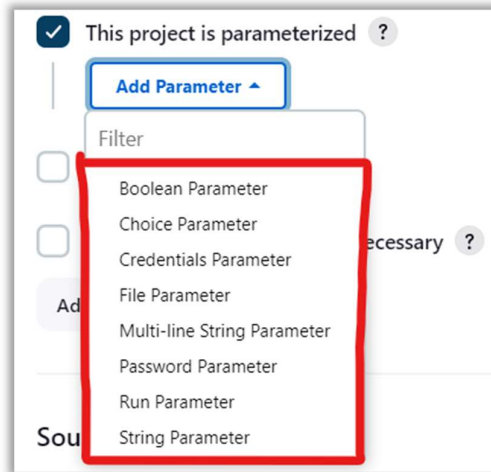
This field is also the same as **Days to keep builds**. We can maintain the certain number of builds by this field.

## **GitHub project**



This field is used to interact with GitHub account. We can integrate our public/private GitHub repo with this Jenkins. If we forget to include the GitHub repo details here, it doesn't matter because we have maintain the SCM with GitHub account.

## This project is parameterized



In this field you can use the multi parameter to interact with projects. This field have 8 options.

### Boolean Parameter

We can give the value as a **true/false** or **0/1** like that.

A screenshot of a "Boolean Parameter" configuration window. The window has a title bar with a hamburger menu icon, the text "Boolean Parameter", and a close button. Below the title bar is a light gray box containing a description: "Defines a simple boolean parameter, which you can use during a build, either as an environment variable, or through variable substitution in some other parts of the configuration. The string value will be 'true' or 'false'." Below this is a "Name" field with a question mark icon and an empty text input box. Underneath the name field is another light gray box with the text: "The name of the parameter. These parameters are exposed to build as environment variables." Below that is a "Set by Default" checkbox, which is currently unchecked, with a question mark icon. Underneath this checkbox is a light gray box with the text: "Specifies the default value of the field." At the bottom is a "Description" field with a question mark icon and a large empty text area.

### Choice parameter

We can make multi-choice over this field.

Choice Parameter ?

Defines a simple string parameter, which can be selected from a list. You can use it during a build, either as an environment variable, or through variable substitution in some other parts of the configuration..

Name ?

The name of the parameter  
These parameters are exposed to build as environment variables.

Choices ?

Requires Choices.

The potential choices, one per line. The value on the first line will be the default.

Description ?

Plain text [Preview](#)

## Credential Parameter

This parameter will use to store the credentials.

Credentials Parameter ?

Name ?

Credential type  

Any

☐ Required ?

Default Value ?  

- none -


Add

Description ?

Plain text [Preview](#)

## File parameter

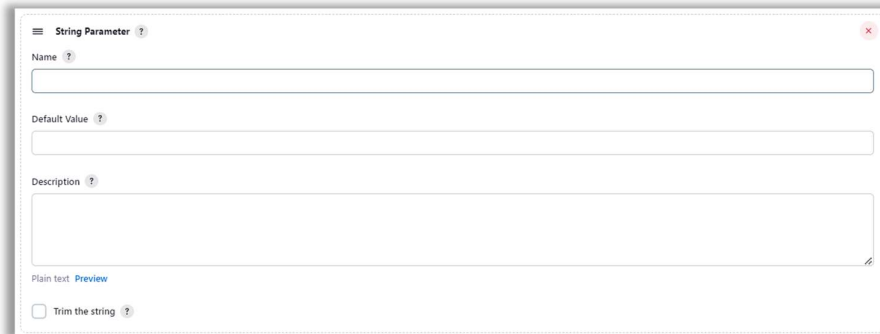
This field will help to upload the local files into the specific directory. We can specify path in the file location field. Then we can describe any description in the description field.



The screenshot shows a 'File Parameter' configuration window. It has a title bar with a hamburger menu, the text 'File Parameter', and a question mark icon. There is a red 'X' icon in the top right corner. The main area contains two input fields: 'File location' and 'Description'. Below the 'Description' field, there are links for 'Plain text' and 'Preview'. A small icon is visible in the bottom right corner of the text area.

## String parameter

This field is used to specify the string which will use to call as a variable in build steps field.



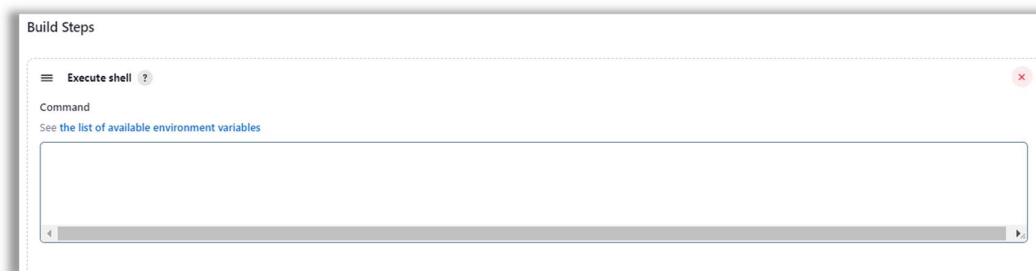
The screenshot shows a 'String Parameter' configuration window. It has a title bar with a hamburger menu, the text 'String Parameter', and a question mark icon. There is a red 'X' icon in the top right corner. The main area contains three input fields: 'Name', 'Default Value', and 'Description'. Below the 'Description' field, there are links for 'Plain text' and 'Preview'. At the bottom, there is a checkbox labeled 'Trim the string' with a question mark icon.

for example, we can select the Execute shell in the Build steps field. Then declare the variable in string parameter(file). It will help to execute the command(mkdir or touch) when you build the job.

Ex, mkdir \$file

Touch \$file

Note: (mkdir/touch \$file is declare in the execute shell in build steps. Then might be declare the variable name file is declare in string parameter field)



The screenshot shows a 'Build Steps' configuration window. It has a title bar with a hamburger menu, the text 'Execute shell', and a question mark icon. There is a red 'X' icon in the top right corner. The main area contains a 'Command' input field. Below the input field, there is a link that says 'See the list of available environment variables'. A scrollbar is visible at the bottom of the command field.

## Multi-line String Parameter

This is similar to the string parameter but we can give multiple string values here.



The image shows a configuration window titled "Multi-line String Parameter". It contains three input fields: "Name", "Default Value", and "Description". Each field has a small question mark icon to its right. The "Name" field is a single-line text box. The "Default Value" and "Description" fields are multi-line text boxes. At the bottom left, there is a "Plain text" label and a "Preview" link. At the bottom right, there is a small icon of a document with a pencil. A red "X" icon is in the top right corner.

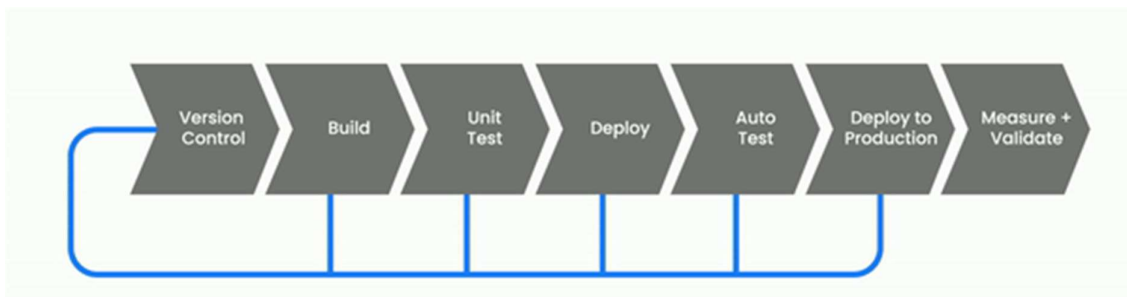
We can choose multiple parameter at a time.

For more must watch the tutorial.

## Pipeline

The pipeline is used to create the automate the thing which will made by manual. For example. As a devops cycle we have to plan, code, build, test, deploy, monitor.

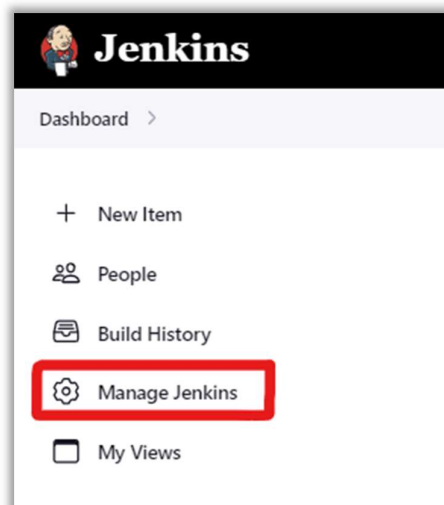
if the code created successfully, we have to push the code to build stage then it will move it to next stages. Meanwhile if any stage get fails, it will go to the previous stage to troubleshoot then it will fix then it will move to next stage with fixes. As same as it will get the feedback from current stage if there was an error then revert back to previous stage.

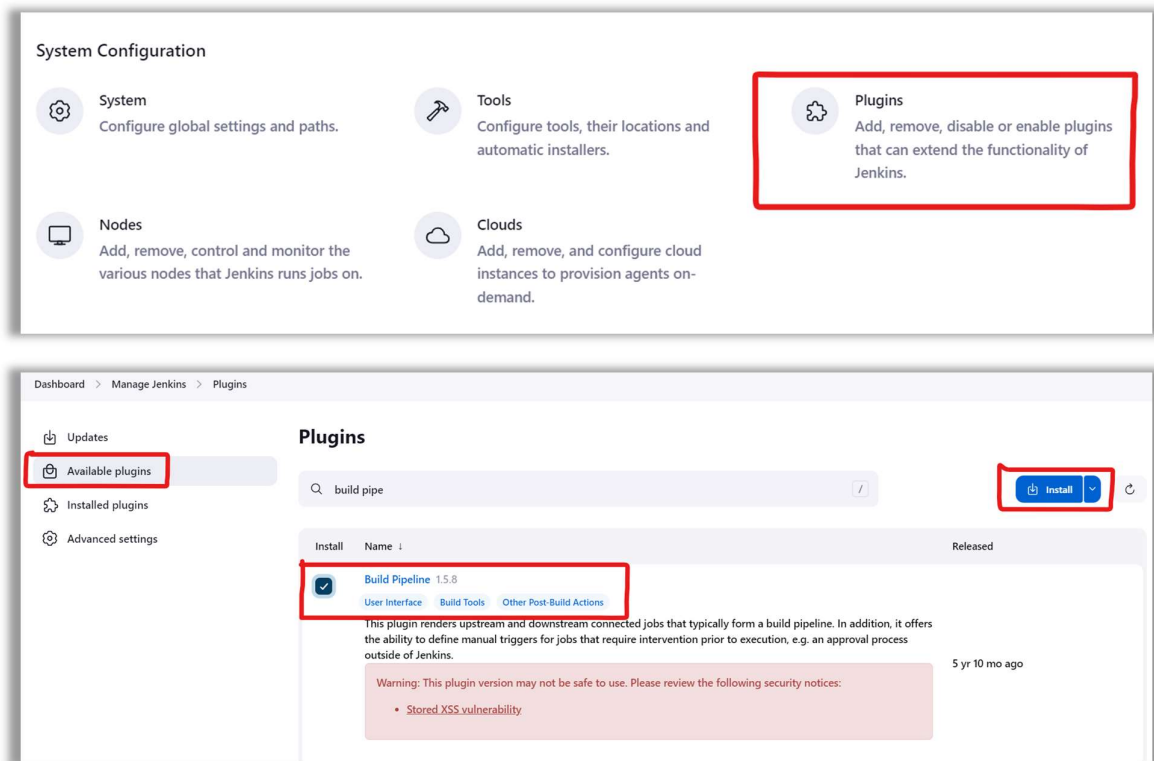


to make the pipeline view follow the below steps.

We need to install Build Pipeline from plugins by below path.

Dashboard -> Manage Jenkins -> plugins -> Available plugins -> search “**Build pipeline**” -> select the checkbox of **Build Pipeline** and click on install.

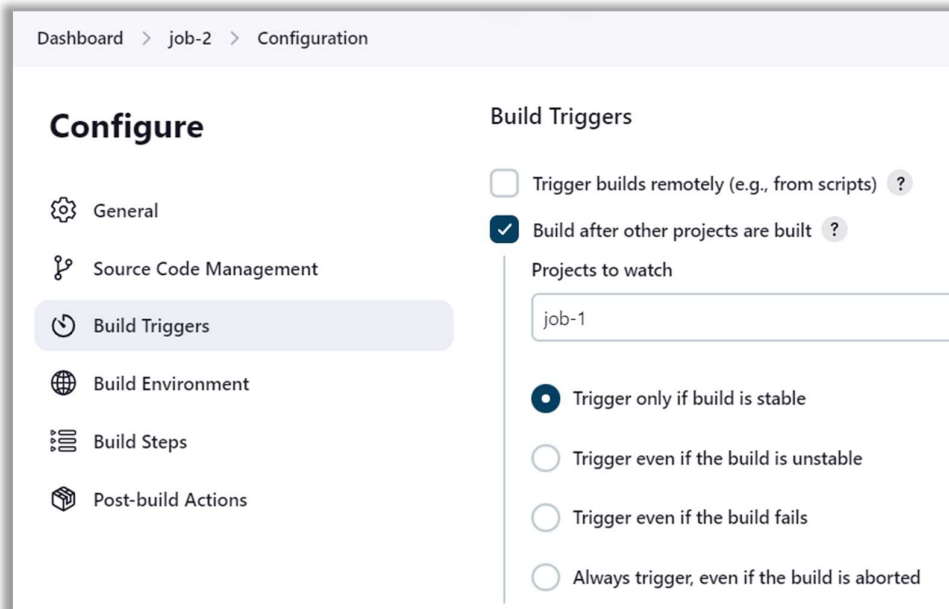




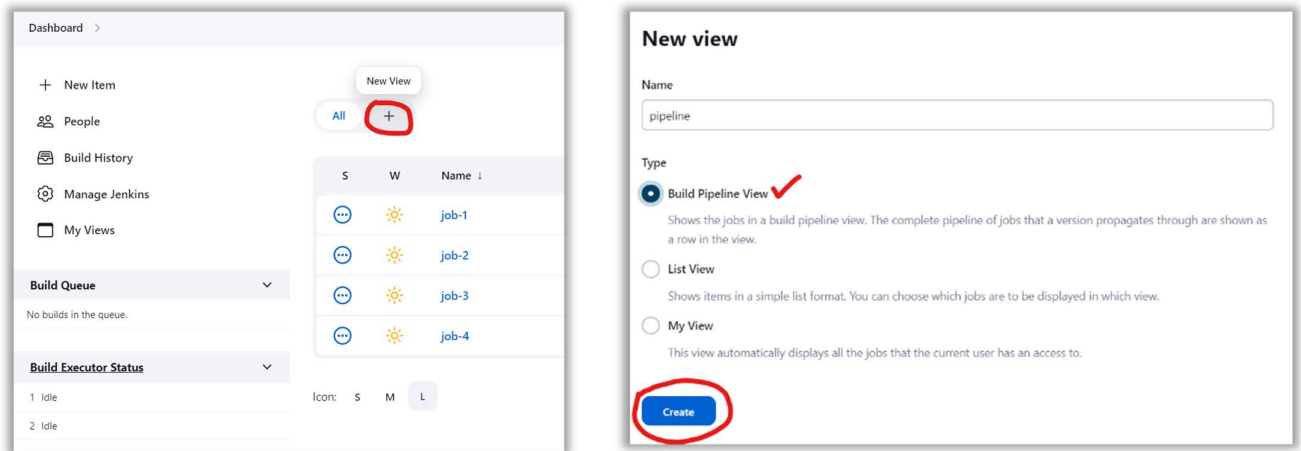
After the installation we need to following below steps.

If we want to create more than one jobs to build pipeline. Now need to link job-2 to job-1. If the job-1 build successfully then run the job-2. Like this we can link with others.

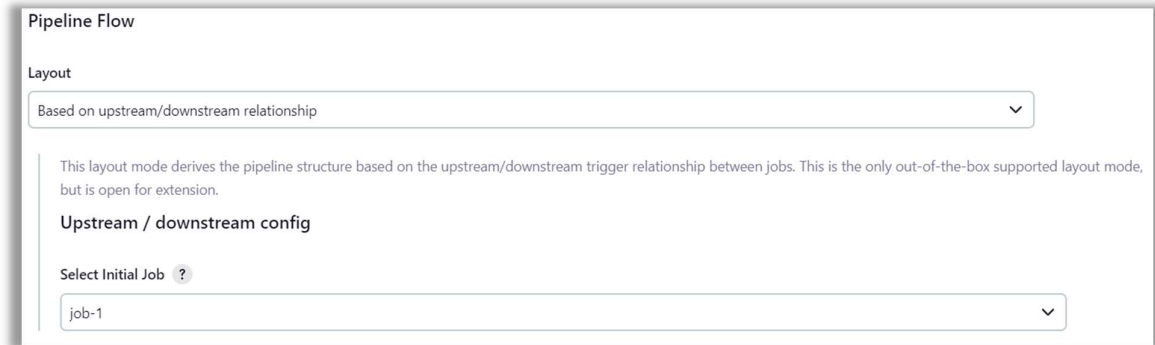
In this below picture we were in job-2. So, we can link with Job-1. So if the job build successfully the job-2 will execute.



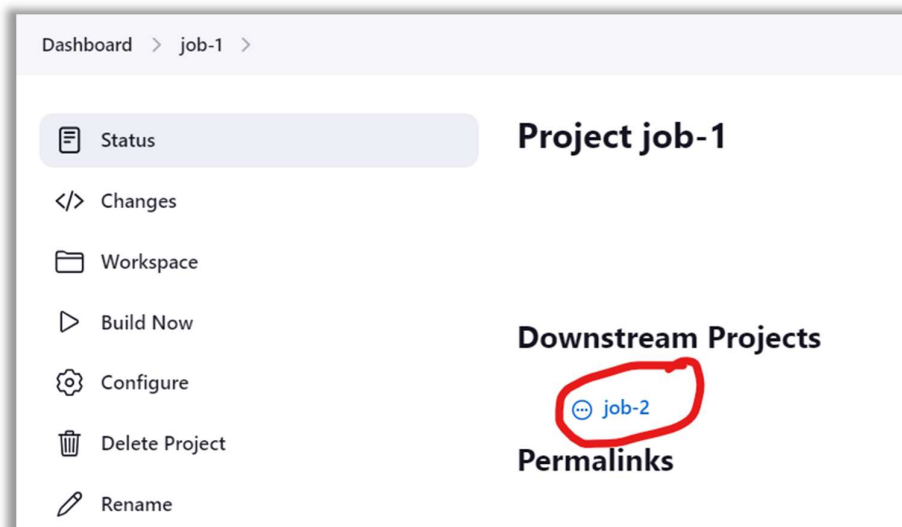




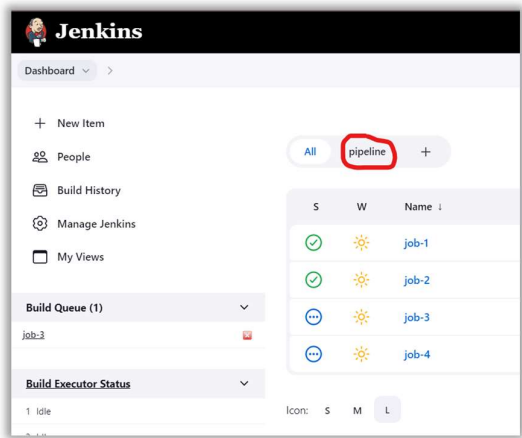
After following above steps, then launch the pipeline view, now we should make sure about the pipeline flow initial job which job you would like to start from.



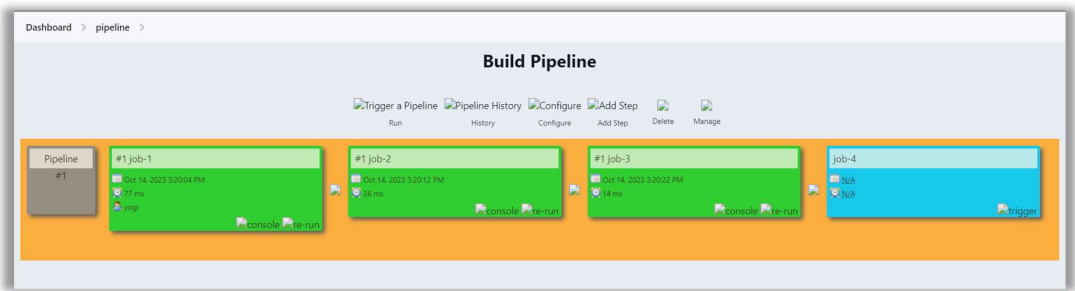
After select the job lines now go to the job-1. We can see the job-2 as a downstream. Which means if the current job get completed the next job will be execute which is job-2. Like wise this pipeline will work.



Now start the build in current job by click on **Build now/ build with parameter**.



Now we can clearly see the build has been started. But we can see more clearly by go to the pipeline view. Now we can see the build pipeline view.



The pipeline will look like below while building the jobs.

