**Jenkins Intro**

+ Jenkins: Jenkins is a free and open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.

Important features in Jenkins:

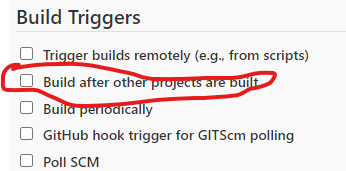
1. Plugins
2. We can create jobs.
3. We can link jobs.
4. We can create pipelines.

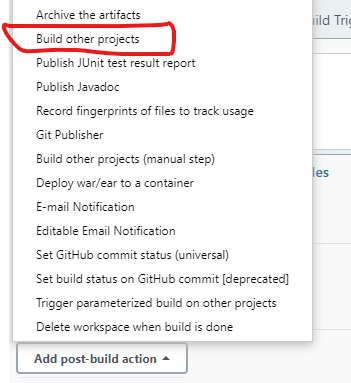
Note: to create a pipeline we need “Pipeline” Plugin.

**create pipelines from jobs**

1. A. From jobs you want to create a pipeline than we want to install plugin “[Build Pipeline](https://plugins.jenkins.io/build-pipeline-plugin)” and also “Delivery Pipeline”.

B. create Multiple job and link those by using “Build Triggers” and “Post Build Actions”. This is called linking or chain.





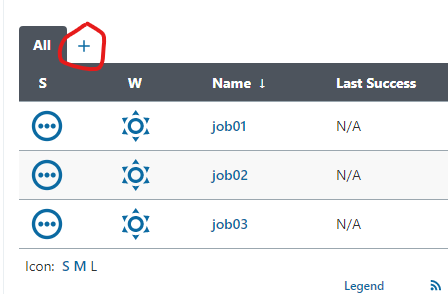
Note:

Upstream Project

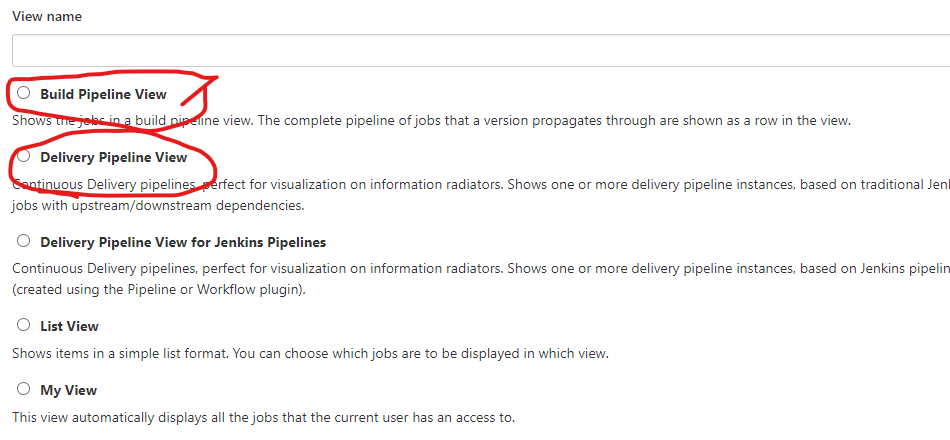
Downstream Project

C. After link completed, we can one job the other depended/linked jobs also build.

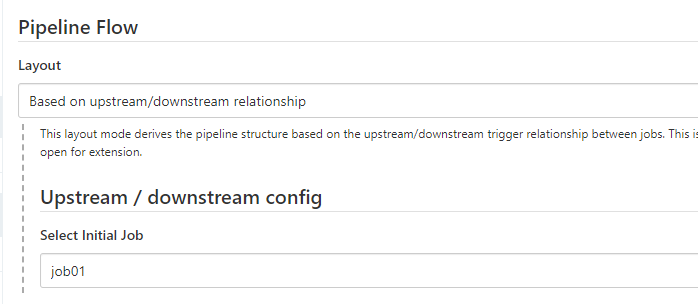
1. Here we can create a pipeline by using Jobs.
2. install plugin “[Build Pipeline](https://plugins.jenkins.io/build-pipeline-plugin)” and also “Delivery Pipeline”.
3. Clik on “+” symbol



1. We are getting below options



1. Select which pipeline you want
2. I selected “Build pipeline view” and select initial job.



1. Click on “apply and ok”
2. Click on run icon.

Note: Similar way we can build “Delivery Pipeline View”

**By using Groovy Script, create Pipelines**

By using groovy we are able to create 2 types of Pipelines:

1. **Scripted type**: we will write Script while creating project.
2. **Declarative type**: in this we maintain pipeline script in “jenkinsfile” in GitHub repo.

Note : the difference is “in this method we don’t need additional plugins”.

**Scripted Type:**

pipeline {

agent any

stages {

stage('Build') {

steps {

echo "build stage"

}

}

stage('Test') {

steps {

echo "test stage"

}

}

stage('Deploy') {

steps {

echo "deploy stage"

}

}

}

post{

always{

echo "always print this with respect to build"

}

}

}

pipeline {

agent any

stages {

stage('Build') {

steps {

echo "build stage"

}

}

stage('Test') {

when{

expression {

return env.BRANCH\_NAME=="master";

}

}

steps {

echo "test stage"

}

}

stage('Deploy') {

steps {

echo "deploy stage"

}

}

}

post{

always{

echo "always print this with respect to build"

}

}

}

Maven:

 a pipeline job by calling sh "mvn …​ or bat "mvn …​.

**This is the Pipeline to run Terraform in Jenkins**

pipeline {

agent any

stages {

stage("GitHub checkin"){

steps{

git branch: 'main', credentialsId: 'f618f6ea-5783-46e8-bce5-4197e880899c', url: 'https://github.com/Pavan-14/private\_terraform.git'

}

}

stage("terraform initilation"){

steps{

sh '''terraform init'''

sh '''terraform fmt'''

}

}

stage("terraform validate"){

steps{

sh "terraform validate"

sh "terraform plan "

}

}

stage('Approval') {

steps {

script {

def userInput = input(id: 'confirm', message: 'Apply Terraform?', parameters: [ [$class: 'BooleanParameterDefinition', defaultValue: false, description: 'Apply terraform', name: 'confirm'] ])

}

}

}

stage('TF Apply') {

steps {

sh 'terraform apply -input=false --auto-approve'

}

}

}

}