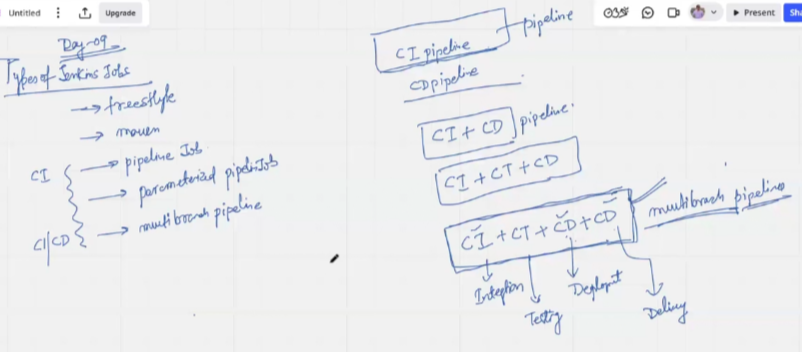
Types of Jenkins jobs and their use in different phases of continuous delivery



On system reboot, update the Jenkins system tab with new jenkins public ip which was overridden on top of old ip to troubleshoot Jenkins slowness

Steps to work with CI using parameterized job

Take help from Satyam GIT

Goto Github >> satyam88 >> Repositories >> para\_job >> URL

Goto Jenkins >> New item >> pipeline=Parameterized job >> Configure>> General >> Description=”This is a parametrized jenkins job” >> Pipeline >> Definition=Pipeline script from SCM >> SCM=GIT>> Repo URL=para\_job >> Branches to build= \*/master >> Script path=Jenkinsfile >> Build now. >> Stage view

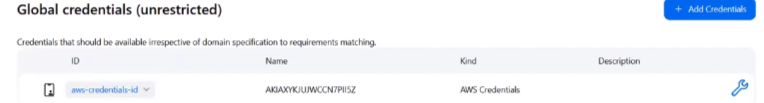
Build with parameters

**Repeat with job2 with name=list\_resources\_in\_aws**

Credentials will be missing >> Goto IAM >> Create user >> aws-credentials >> Attach policies directly >> AdministratorAccess >> Next >> Create >> View user >> Security credentials >> Access keys >> Application running outside AWS >> Next >> Create access key



Add key to jenkins >> Goto Manage jenkins >> Security >> Credentials >> global >> Add credentials >> AWS credentials (plugin needed) >> ID(check in the job definition) >> Access key ID(IAM) >> Secret access key(IAM) >> Create …… shows error of unknown ID

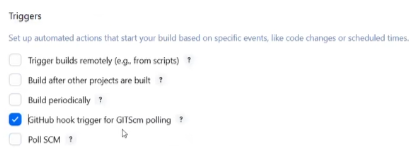


Delete and create new access key with similar kind - username with password

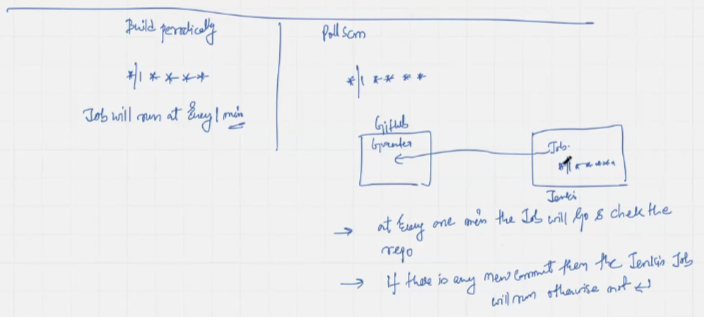
Then

Build with parameters

Triggers in Jenkins



* Build periodically >> CRONTAB >> trigger=”demo” & build steps >> echo “Hello World”
* Poll SCM >> CRONTAB >> trigger= ”demoSCM & build steps” >> Source code management=Git=repo url to poll >> every min >> echo “Hello World”

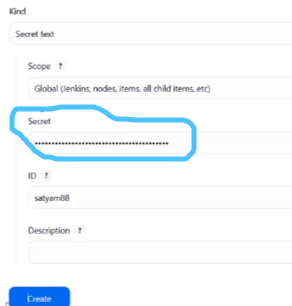


* GitHub hook trigger for GITScm polling - used at webhook level whenever code is pushed from IntelliJ IDE

^^ When we change the visibility of repository settings to Private, & build fails ….displays authentication failure, so add credentials

- Goto Jenkins >> Manage jenkins > Credentials >> System >> Global credentials >> Kind=secret text >> ID=github user >> secret = point A >> Create

1. Github name=pophale-viraj >> Settings >> Developer settings >> Personal access tokens >> tokens(classic) >> Generate new token >> classic >> Note=mygithubtoken >> Expiration = no expiration >> Select scopes = all >> Generate token >> Copy



Configure the jenkins job to use the key at Source Code Management >> Credentials …displays credentials without secret message

**Replacement step as secret text is incorrect >>** Update the credentials of job to use with token >> Goto Jenkins >> Add credentials >> Username with password >> Add password=token

Then

Delete secret key

Then

Build Now …displays that username with password is used between GitHub and Jenkins

Continue pipeline example from 10 May 2025: <https://github.com/satyam88/greentea/blob/master/may-11> < point A

**A:**

====================================

pipeline {

agent any

options {

buildDiscarder(logRotator(numToKeepStr: '3', artifactNumToKeepStr: '3'))

}

tools {

maven 'mvn\_3.9.9' // Make sure this Maven tool is defined in Jenkins global tools

}

stages {

stage('Code Compilation') {

steps {

echo 'Starting Code Compilation...'

sh 'mvn clean compile'

echo 'Code Compilation Completed Successfully!'

}

}

stage('Code QA Execution') {

steps {

echo 'Running JUnit Test Cases...'

sh 'mvn clean test'

echo 'JUnit Test Cases Completed Successfully!'

}

}

stage('Code Package') {

steps {

echo 'Creating WAR Artifact...'

sh 'mvn clean package'

echo 'WAR Artifact Created Successfully!'

}

}

}

post {

success {

echo '✅ Build completed successfully.'

}

failure {

echo '❌ Build failed.'

}

}

}

======================

pipeline {

agent any

parameters {

string(name: 'maven\_version', defaultValue: '3.9.3', description: 'Pass the version of Maven')

string(name: 'terraform\_version', defaultValue: '1.8.5', description: 'Pass the version of Terraform')

}

stages {

stage('Download Maven') {

steps {

sh '''

cd /var/lib/jenkins/

sudo wget https://dlcdn.apache.org/maven/maven-3/${maven\_version}/binaries/apache-maven-${maven\_version}-bin.tar.gz

'''

}

}

stage('Download Terraform') {

steps {

sh '''

cd /opt

sudo wget https://releases.hashicorp.com/terraform/${terraform\_version}/terraform\_${terraform\_version}\_linux\_amd64.zip

'''

}

}

}

}

========================

pipeline {

agent any

parameters {

choice(

name: 'AWS\_REGION',

choices: ['us-east-1', 'us-west-1', 'us-west-2', 'eu-central-1', 'eu-west-1', 'ap-south-1'],

description: 'Select the AWS region'

)

}

environment {

AWS\_REGION = "${params.AWS\_REGION}"

}

stages {

stage('List AWS Resources') {

steps {

script {

echo "Listing resources in the region: ${AWS\_REGION}"

// Use withCredentials to securely access AWS credentials

withCredentials([usernamePassword(credentialsId: 'aws-credentials-id', usernameVariable: 'AWS\_ACCESS\_KEY\_ID', passwordVariable: 'AWS\_SECRET\_ACCESS\_KEY')]) {

// List all EC2 instances

echo "Fetching EC2 Instances..."

sh """

aws ec2 describe-instances --region ${AWS\_REGION} --query "Reservations[\*].Instances[\*].{InstanceId:InstanceId, State:State.Name, InstanceType:InstanceType, PublicIpAddress:PublicIpAddress}" --output table

"""

// List all EBS volumes

echo "Fetching EBS Volumes..."

sh """

aws ec2 describe-volumes --region ${AWS\_REGION} --query "Volumes[\*].{VolumeId:VolumeId, Size:Size, State:State}" --output table

"""

// List all Security Groups

echo "Fetching Security Groups..."

sh """

aws ec2 describe-security-groups --region ${AWS\_REGION} --query "SecurityGroups[\*].{GroupName:GroupName, GroupId:GroupId, Description:Description, VpcId:VpcId}" --output table

"""

// List all Elastic IPs

echo "Fetching Elastic IPs..."

sh """

aws ec2 describe-addresses --region ${AWS\_REGION} --query "Addresses[\*].{PublicIp:PublicIp, InstanceId:InstanceId}" --output table

"""

}

}

}

}

}

}

Home Directory for Jenkins

/var/lib/jenkins

service jenkins restart

service jenkins status

[root@ip-172-31-37-131 jenkins]# service jenkins

The service command supports only basic LSB actions (start, stop, restart, try-restart, reload, reload-or-restart, try-reload-or-restart, force-reload, status, condrestart). For other actions, please try to use systemctl.

Enable Logs in jenkins

======================

mkdir -p /etc/systemd/system/jenkins.service.d

sudo vim /etc/systemd/system/jenkins.service.d/override.conf

[Service]

StandardOutput=append:/var/log/jenkins/jenkins.log

StandardError=append:/var/log/jenkins/jenkins.log

save and exit -> :wq!

sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl restart jenkins

tail -f /var/log/jenkins/jenkins.log

========how to find plugin binary===============================

https://plugins.jenkins.io/gcp-secrets-manager-credentials-provider/

https://github.com/jenkinsci/gcp-secrets-manager-credentials-provider-plugin/releases/tag/v0.3.1

wget https://github.com/jenkinsci/gcp-secrets-manager-credentials-provider-plugin/archive/refs/tags/v0.3.1.zip

===========install plugin in jenkins using CLI=========================================

wget https://updates.jenkins.io/download/plugins/gcp-secrets-manager-credentials-provider/0.3.1/gcp-secrets-manager-credentials-provider.hpi

sudo mv gcp-secrets-manager-credentials-provider.hpi /var/lib/jenkins/plugins/

sudo chown jenkins:jenkins /var/lib/jenkins/plugins/gcp-secrets-manager-credentials-provider.hpi

sudo chmod 644 /var/lib/jenkins/plugins/gcp-secrets-manager-credentials-provider.hpi

sudo systemctl restart jenkins

================================