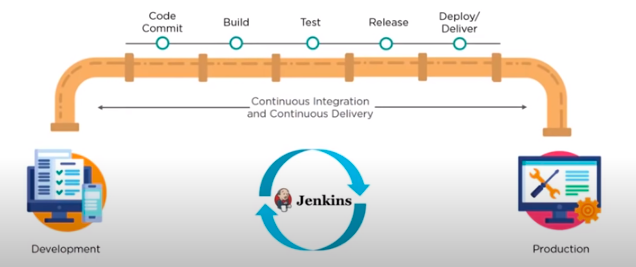
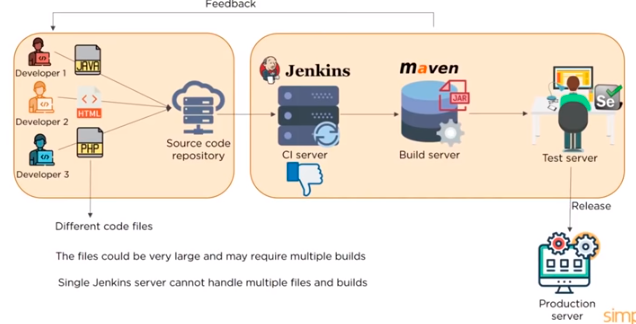
**What is Upstream and down stream job?**

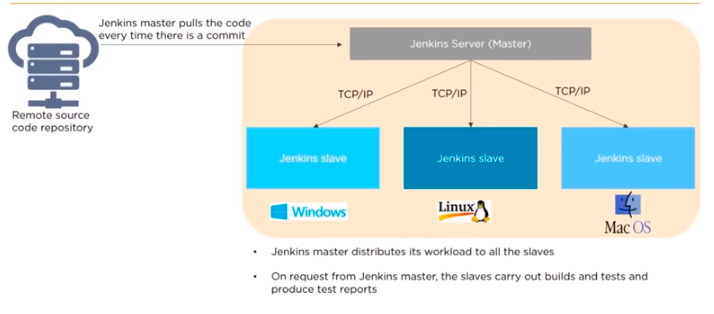
The upstream job is the one that is triggered before the actual job is triggered. The downstream job is the one that is triggered after the actual job is triggered. We can configure the actual job not to be triggered if the upstream job is failed. In the same way, we can configure the downstream job not to be triggered if the actual job is failed.

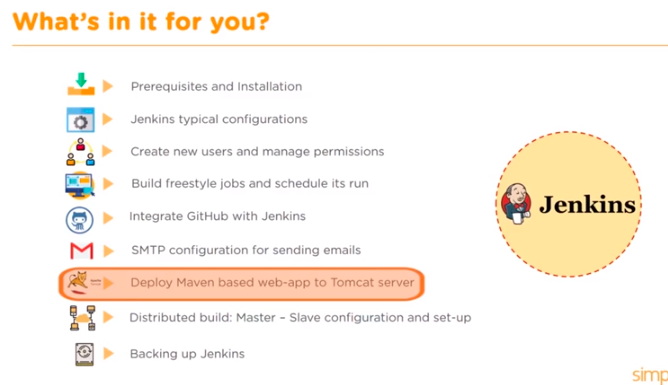
Order of job triggers:

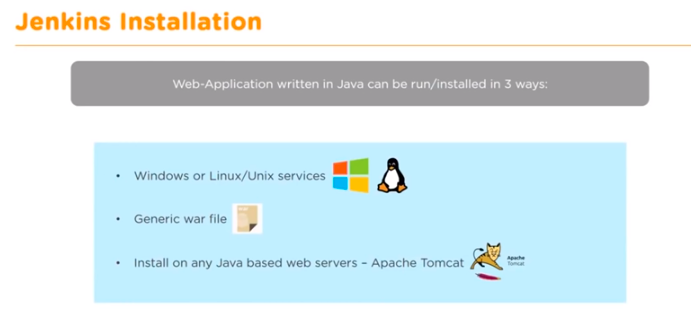
Upstream Job -> Actual Job -> Downstream Job

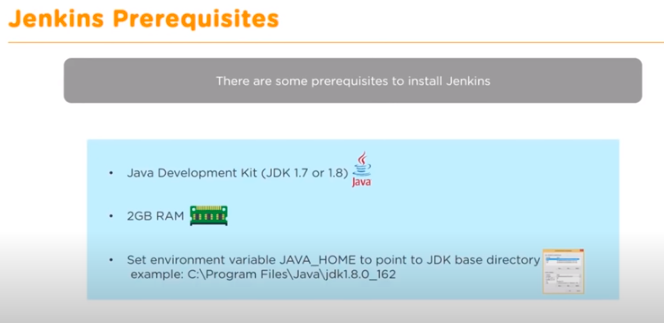


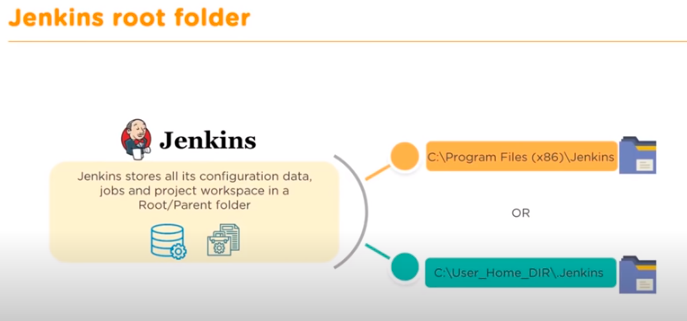












**Manage Jenkins**

Here we can see all options

Manage Jenkins 🡪

**System Configuration 🡪**

**Configure Systems 🡪**

Home Directory: Here all configuration and workspace Jenkins related things stored.

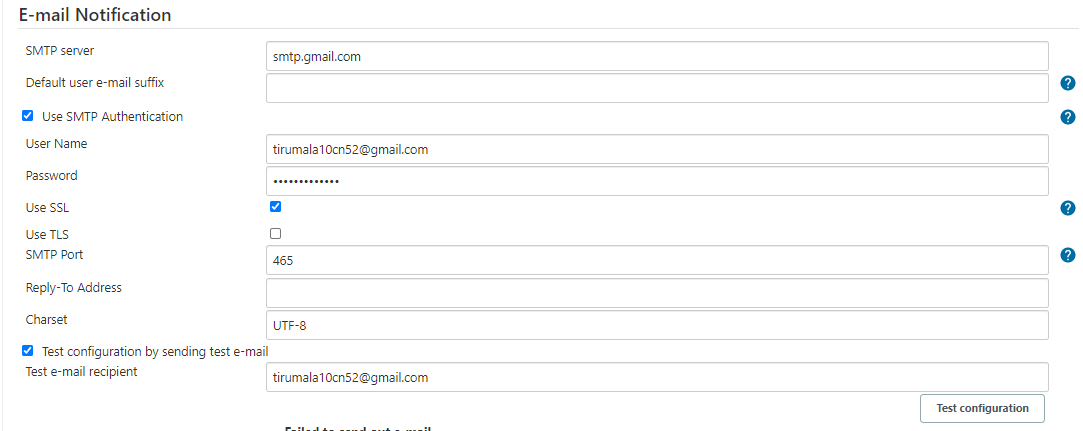
System Message: View this message in Jenkins page

Send Email Notification:

SMTP SERVER: Provide smtp server email id or if it is in local provide smtp.gmail.com

Port: 465

Test configuration mail id: [tirumala10cn52@gmail.com](mailto:tirumala10cn52@gmail.com)



**Global Tool Configuration** 🡪

* Need to configure the build tools like GitHub, Maven, ANT JDK to run the Code.

**Manage Jenkins 🡪**

* I can able to install plugins.
* Install Role -based strategy plugins and see it in Global Tool Configuration

**Security 🡪**

**Configure Global Security 🡪**

By default Jenkins provides user database for authentication. I can able to configure LDAP server, AD or other authentication.

**Jenkins Architecture**

Jenkins helps us to integrate different Devops tools using plugins. You can connect to the Testinf server and Production server from Jenkins Master .

Add slave node to Jenkins using JNLP connection

1. **Create Files in GitHub**
2. **Create DockerFile**

In this docker file copy the code files into Docker file

1. **Run the DockerFile in Jenkins by creating the New job**

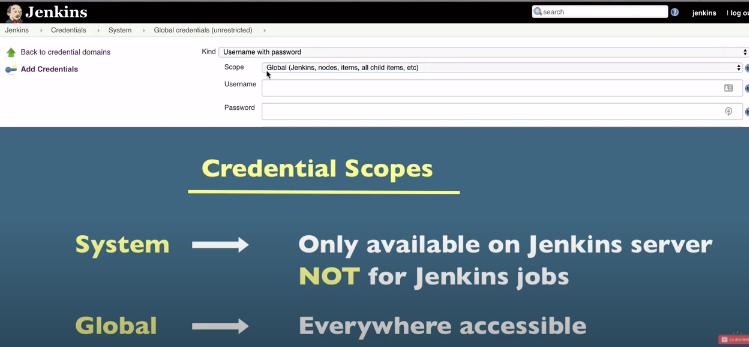
Run the below commands in Execute shell 🡪

$Sudo docker rm -f $(sudo docker ps -a -q)

$sudo docker build /home /ubuntu/Jenkins/Git-Job/ -t website

$sudo docker run --name dockerwebsite -d -p 8080:80 website

If you want to add notification to github webhook 🡪 go to webhook and give the url of the instance like http://<Ip>:port /github-webhook/



def notifySlack(String buildStatus = 'STARTED') {  
  
// Build status of null means success.  
buildStatus = buildStatus ?: 'SUCCESS'

def color

if (buildStatus == 'STARTED') {  
color = '#FFFF00'  
} else if (buildStatus == 'SUCCESS') {  
color = '#00FF00'  
} else if (buildStatus == 'UNSTABLE') {  
color = '#FFFE89'  
} else {  
color = '#FF0000'  
}  
wrap([$class: 'BuildUser']) {  
sh 'echo "${BUILD\_USER}"'  
  
def msg = "${buildStatus}: `${env.JOB\_NAME}` #${env.BUILD\_NUMBER} by ${BUILD\_USER}\n More info at: ${env.BUILD\_URL}"  
  
slackSend(baseUrl: 'https://hooks.slack.com/services/', channel: '#jenkins-events-npr', tokenCredentialId: 'slack-webhook', color: color, message: msg)  
}  
}

node('master'){  
try {  
wrap([$class: 'AnsiColorBuildWrapper', colorMapName: "xterm"]) {  
notifySlack()  
docker.withRegistry('https://iad.ocir.io/','sa-ocir-login') {  
stage('Preparation') {   
git (url: 'https://github.com/qloudable/qloudable-training-labs.git', branch: 'tl-qld-stg', credentialsId: 'sa-github')  
sh "git status"  
sh "kubectl get nodes"  
}

stage('findchanges') {   
sh "dos2unix tl-mappinglist.txt"  
def changes = ""  
build = currentBuild  
while(build != null && build.result != 'SUCCESS') {  
for (changeLog in build.changeSets) {  
for(entry in changeLog.items) {  
for(file in entry.affectedFiles) {  
changes += " ${file.path}\n"  
}}}  
build = build.previousBuild  
}  
echo changes

writeFile file: "fileschanged", text: changes, encoding: "UTF-8"  
sh "cat fileschanged"  
sh "cat fileschanged | sed 's@/@ @g' | awk '{ print \$1 }' | sort -u | grep -v .gitignore | grep -v catagory-service | grep -v QloudableTrainingLabs.jmx | grep -v copy-pem.sh | grep -v dc.dev.stack.yml | grep -v dc.stg.stack.yml | grep -v dc.stg.stack1.yml | grep -v dee-tl-int.yaml | grep -v rdb.stack.yml | grep -v solution.alpha.stack.yml | grep -v solution.stack.yml | grep -v tl-in-mappinglist.txt | grep -v tl-mappinglist.txt | grep -v tl-st-mappinglist.txt | grep -v tl-pr-mappinglist.txt | grep -v vault.stack.yml | grep -v statemachine-services | grep -v ssh-keygen-service | grep -v nbproject | grep -v docker-alpine-curl | grep -v stream-services | grep -v extract-transform-load | grep -v reset-password-service | grep -v deploy-image-tfv2 | grep -v deploy-image-tfv3 | grep -v channel-code-service | grep -v badge-service | grep -v tl-mappinglist.txt | grep -v tl-qld-pint-mappinglist.txt | grep -v tl-qld-int-mappinglist.txt | grep -v deploy-router-service | grep -v cleanup-resource-azure | grep -v tl-qld-stg-mappinglist.txt | grep -v tl-qld-pint-mappinglist.txt | grep -v tl-qld-int-mappinglist.txt | grep -v tl-pint-mappinglist.txt | grep -v tl-in-mappinglist.txt | grep -v tl-st-mappinglist.txt | grep -v tl-pr-mappinglist.txt | grep -v deploy-image-azure | grep -v terminal-service | grep -v terminal-creation-service | grep -v cleanup-service | grep -v provider-setup-service > services"  
sh "cat services"  
sh "date +%F > date"  
def date = readFile('date').trim()  
println date  
}

stage('build\_n\_push') {   
// script {  
// // capture the approval details in approvalMap.  
// approvalMap = input id: 'test', message: 'Hello', ok: 'Proceed?', submitter: 'umar,asebastian,srekapalli', submitterParameter: 'APPROVER'  
// }  
sh "date +%F > date"  
def date = readFile('date').trim()  
println date  
def sers = readFile('services').trim()  
def dockerfile = 'Dockerfile.kube'  
String[] arraysers = sers.split("\n");  
for (String eachser : arraysers) {  
print(eachser);  
def img = docker.build('jumpstart/tl-qld-stg', "-f ${eachser}/Dockerfile.kube ${eachser}").push "${eachser}-stg-cikube-${date}.$BUILD\_NUMBER"   
}  
}

stage('deploy') {  
sh '''  
today=`date +%F`  
echo $today  
while read eachser  
do  
echo "${eachser} has changed"  
dockerservice=`cat ${WORKSPACE}/tl-mappinglist.txt | grep ^"${eachser}" | awk '{print $2}'`  
kubectl -n tl-qld-stg set image deployment/"$dockerservice" "$dockerservice"=iad.ocir.io/jumpstart/tl-qld-stg:"$eachser"-stg-cikube-"$today"."$BUILD\_NUMBER"  
echo $dockerservice  
done<services  
'''  
}  
// stage("Jmeter\_test") {  
// git (url: 'https://github.com/sysgain/qloudable-stackfiles.git', branch: 'tl-jmeter', credentialsId: 'git-login')  
// sh "git status"  
// sh "sleep 1m"  
// // remove previous loadtest result file.  
// sh "rm result.jtl"  
// // Prepare a list and write to file  
// sh "echo \"ten\_users.jmx\ntwo.jmx\nthree.jmx\" > ${WORKSPACE}/list"

// // Load the list into a variable  
// env.LIST = readFile (file: "${WORKSPACE}/list")

// // Show the select input  
// env.RELEASE\_SCOPE = input message: 'Select no.users Jmx file', ok: 'Test!',  
// parameters: [choice(name: 'JMX\_file', choices: env.LIST, description: 'Selected Jmx file to run load test')]  
// echo "Users for load test: ${env.RELEASE\_SCOPE}"  
// // sh "cat ${env.RELEASE\_SCOPE}"  
// sh "jmeter -Jjmeter.save.saveservice.samplerData=true -Jjmeter.save.saveservice.response\_data=true -Jjmeter.save.saveservice.output\_format=xml -n -t ./qld-stg/${env.RELEASE\_SCOPE} -j jmetertest.log -l result.jtl"  
// sh "cat result.jtl"  
  
// }  
stage('ashburn\_tag') {   
sh '''  
date=`date +%F`  
for eachser in $(cat $WORKSPACE/services)  
do  
echo "${eachser} is being tagged"  
docker tag iad.ocir.io/jumpstart/tl-qld-stg:"$eachser"-stg-cikube-"$date"."$BUILD\_NUMBER" iad.ocir.io/jumpstart/tl-qld-stg:"$eachser"-kube-stg  
docker push iad.ocir.io/jumpstart/tl-qld-stg:"$eachser"-kube-stg  
sleep 3  
done  
'''  
}

stage('delete\_ci\_image') {   
sh '''  
date=`date +%F`  
for eachser in $(cat $WORKSPACE/services)  
do  
echo "${eachser} is being deleted"  
docker rmi iad.ocir.io/jumpstart/tl-qld-stg:"$eachser"-stg-cikube-"$date"."$BUILD\_NUMBER"   
done  
'''  
}  
}  
}  
}  
catch (e) {  
currentBuild.result = 'FAILURE'  
throw e  
} finally {  
notifySlack(currentBuild.result)  
}  
}