DevOps Runbook

CICD for Microservices

# **INTRODUCTION**

This document is intended to serve as the basis for creating run book for creating a generic Jenkins pipeline for the microservices (tibco).

This document begins to explain how one can build and deploy CICD for microservices via a generic Jenkins pipeline and deploy them to GKE in Google cloud platform via Kubernates. The documents covers all the pre-requisites needed for creating Jenkins pipelines, GCP.

# **PRE REQUISITIES**

**2.1. ACCESS**

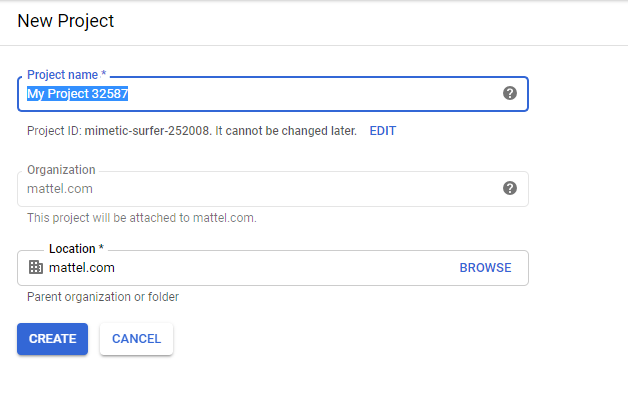
GCP: <https://console.cloud.google.com>

Jenkins: http://<server-ip>:8080/ [to be installed in the build server]

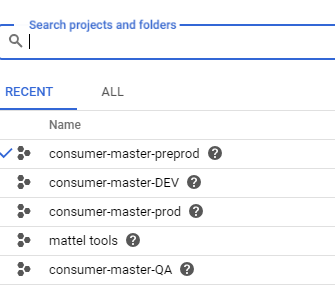
GitHub: https://github.com/<repository-name>

**2.2. GCP CONFIGURATION**

* Account in GCP –Login to GCP “<https://console.cloud.google.com>”
* Create new project in GCP or get access permissions to existing projects in GCP

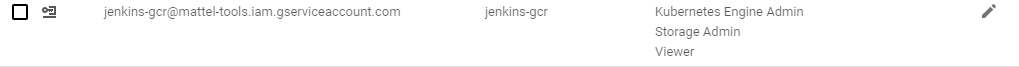


Eg: Projects Access:



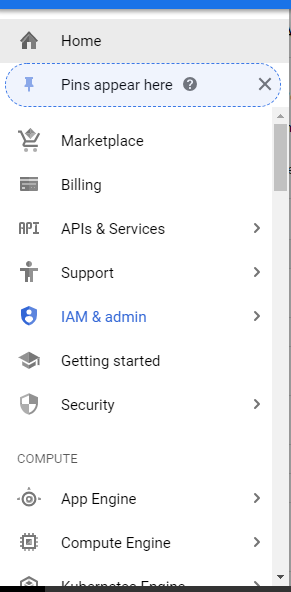
* Service accounts for GCR of each Project (environment) should be created

1. Service account to access a particular project is configured in Jenkins.



1. Service account to access GitHub for checking out the code is configured as well.

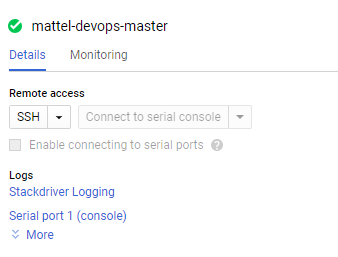
* Kubernates Admin Access, Storage Viewer ,Storage Admin - “IAM Role and Access” should be provided to each project (environment) in regards to the generic project(Mattel tools)



**2.3. SOFTWARE REQUIREMENTS**

**Mattel tools (Generic Project) - Build Server**

The entire DevOps project is present in [an](https://console.cloud.google.com/compute/instancesDetail/zones/us-central1-a/instances/mattel-devops-master?project=mattel-tools) instance, which is present in Mattel Tools project, in GCP.



Below are the software installations done in the mattel-devops-master instance.

|  |  |  |
| --- | --- | --- |
| No | Software | Versions |
| 1 | Jenkins | 2.164.1 |
| 2 | Java | 1.8.0\_212 |
| 3 | Maven | 3.6.0 |
| 4 | Docker | 18.09.6 |
| 5 | Tibco-maven-plugin | bw6-maven-plugin 6.5 |
| 6 | BART | 1.0 |

**References:**

1. <https://linuxize.com/post/how-to-install-apache-maven-on-debian-9/>
2. <https://linuxize.com/post/how-to-install-and-use-docker-on-debian-9/>
3. <https://github.com/TIBCOSoftware/bw6-plugin-maven/wiki/Building-applications-for-TIBCO-BusinessWorks-Container-Edition>
4. <https://nl.devoteam.com/en/blog-post/cicd-with-tibco-bw6/>

**2.4. Best Practices Incorporated**

* Build Once Multiple Deploy
* Tag Based Deployment
* Quality Gates based on the code quality
* Pipeline as code based framework
* Generic pipeline for all micro services/APIs
* Requirement traceability based tagging

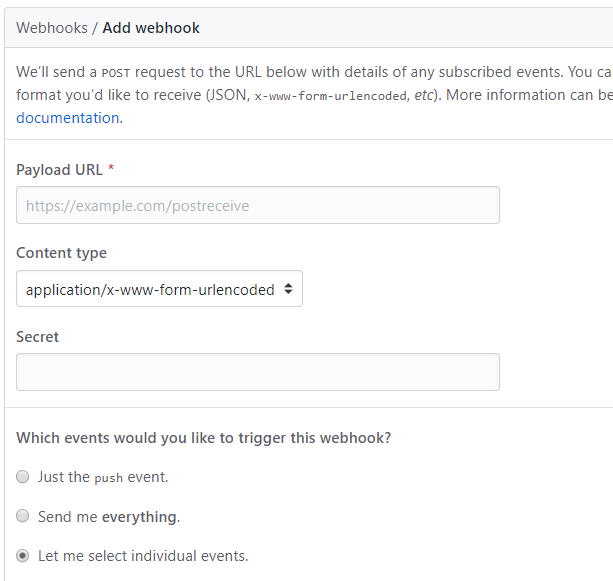
**2.5. GitHub Configuration**

* GitHub repository: [https://github.com/<repository-name>.git](https://github.com/%3crepository-name%3e.git)
* Branches :master,release,dev
* “GitHub-hook trigger” in above-mentioned repository for GITScm polling is configured in GitHub.

1. Under Settings 🡪Webhooks
2. The **payload url** : http://<jenkins-url>/github-webhook/ is provided

**Content Type**: “application/json” is selected

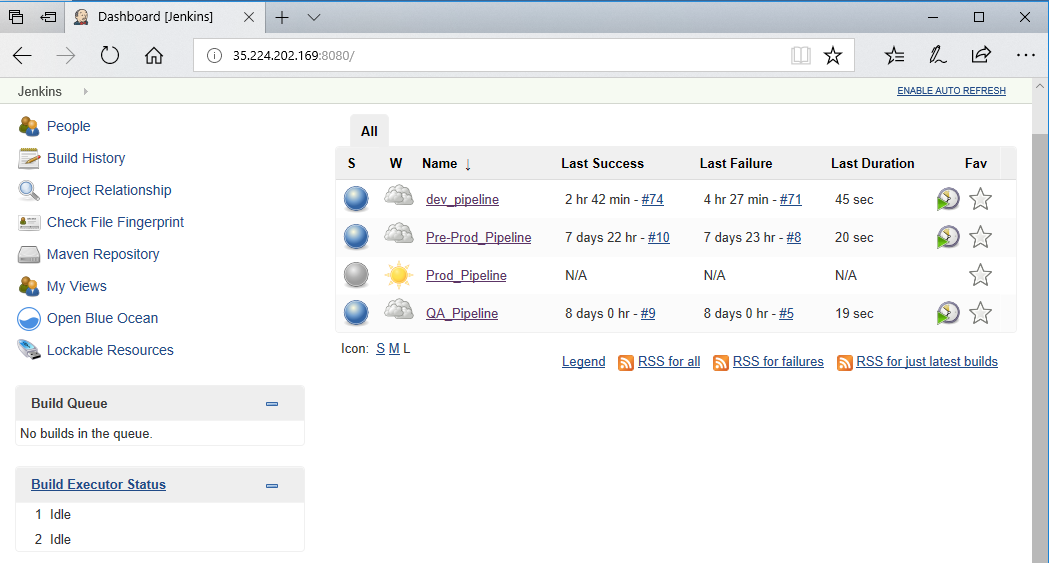
1. **Which events would you like to trigger this webhook? Is selected**



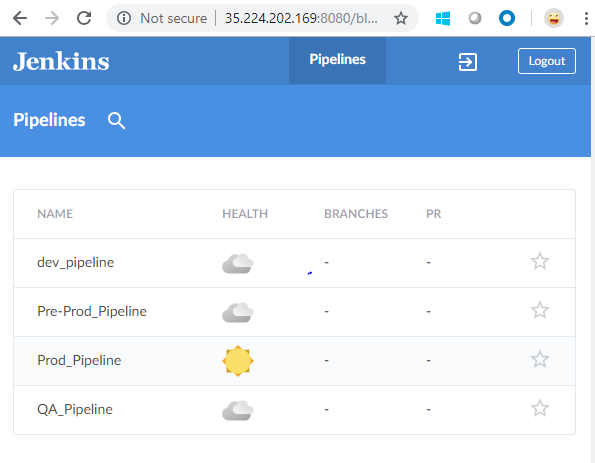
**2.6. Jenkins Configuration**

Login via Jenkins credentials

Once you had logged in to the Jenkins console using your credentials, you will be able to see the below page. Totally, there are four pipeline jobs present in the Jenkins console.



In order to view the various stages of the pipeline and it progress we need to select the option “Open Blue Ocean” in the left pane.



**2.6.1. Plugins To be installed**

**(Manage Jenkins>>Manage Plugins)**

* Google OAuth Credentials Plugin
* Google Container Registry Auth Plugin
* Build Pipeline Plugin
* Maven Integration plugins
* Pipeline maven integration

**2.6.2. Credentials**

* The GitHub credentials in Jenkins ( Credentials>>Add New) is configured
* The mattel-tools project service account credentials is configured

**2.7. WORKFLOW:**

**Sample Jenkinsfile:**

**The variables are externalized using properties file**

def getEnvVar(String paramName){

//get the env from properties file

return sh (script:"grep '${paramName}' /var/lib/jenkins/workspace/${JOB\_NAME}/propertiesfile/dev.properties|cut -d'=' -f2", returnStdout: true).trim();

}

pipeline{

agent any

environment {

SERVICE\_NAME = sh(script: "git describe --tags \$(git rev-list --tags --max-count=1)| cut -d'\_' -f1", , returnStdout: true).trim()

}

stages {

stage('Git Checkout') { // for display purposes

steps{

cleanWs()

checkout([$class: 'GitSCM', branches: [[name: 'refs/tags/\*\_dev\_\*']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'git-service-acc', refspec: '+refs/tags/\*:refs/remotes/origin/tags/\*', url: 'https://github.com/mattel-dig/ConsumerMaster--GSL-.git']]])

}

}

stage('Initialization'){

steps{

//checkout scm

script{

def CATEGORY\_CHECK= sh(script: "echo $SERVICE\_NAME|cut -d'-' -f2", ,returnStdout: true).trim()

if (CATEGORY\_CHECK=='ms') {

env.CATEGORY= sh(script: "echo 'Microservice'", ,returnStdout: true).trim()

env.SERVICE\_CHECK=sh(script: "echo $SERVICE\_NAME", ,returnStdout: true).trim()

}

else {

env.CATEGORY= sh(script: "echo 'API'", ,returnStdout: true).trim()

env.SERVICE\_CHECK=sh(script: "echo $SERVICE\_NAME-api", ,returnStdout: true).trim()

}

env.CODE\_FOLDER\_NAME = getEnvVar('CODE\_FOLDER\_NAME')

env.DEPLOY\_FOLDER\_NAME = getEnvVar('DEPLOY\_FOLDER\_NAME')

env.JENKINS\_GCLOUD\_PROJECT\_ID = getEnvVar('JENKINS\_GCLOUD\_PROJECT\_ID')

env.JENKINS\_GCLOUD\_K8S\_CLUSTER\_ZONE = getEnvVar('JENKINS\_GCLOUD\_K8S\_CLUSTER\_ZONE')

env.JENKINS\_GCLOUD\_K8S\_CLUSTER\_REGION = getEnvVar('JENKINS\_GCLOUD\_K8S\_CLUSTER\_REGION')

env.DEPLOY\_GCLOUD\_PROJECT\_ID\_DEV= getEnvVar('DEPLOY\_GCLOUD\_PROJECT\_ID\_DEV')

env.DEPLOY\_GCLOUD\_K8S\_CLUSTER\_NAME\_DEV = getEnvVar('DEPLOY\_GCLOUD\_K8S\_CLUSTER\_NAME\_DEV')

env.DEPLOY\_GCLOUD\_K8S\_CLUSTER\_ZONE\_DEV = getEnvVar('DEPLOY\_GCLOUD\_K8S\_CLUSTER\_ZONE\_DEV')

env.DEPLOY\_GCLOUD\_K8S\_CLUSTER\_REGION\_DEV = getEnvVar('DEPLOY\_GCLOUD\_K8S\_CLUSTER\_REGION\_DEV')

env.PROJECT\_NAME = getEnvVar(env.SERVICE\_CHECK)

}

}

}

stage('PreBuild'){

steps{

//Builds the container from Dockerfile

sh '''

#description : The script is used to fetch the dependent shared module with respect to the API.

#!/bin/bash

echo ${SERVICE\_NAME}

mkdir /var/lib/jenkins/workspace/${JOB\_NAME}/$SERVICE\_NAME

chmod -R 777 /var/lib/jenkins/workspace/${JOB\_NAME}/$SERVICE\_NAME

#Transfer of API and API files to the workspace

cp -r /var/lib/jenkins/workspace/${JOB\_NAME}/${CODE\_FOLDER\_NAME}/${CATEGORY}/$SERVICE\_NAME/\* /var/lib/jenkins/workspace/${JOB\_NAME}/$SERVICE\_NAME/

#Get the list of shared modules currently present

cd /var/lib/jenkins/workspace/${JOB\_NAME}/$SERVICE\_NAME

ls | grep 'Mattel.\*.parent' > ms\_parent.txt

B="`cat ms\_parent.txt`"

cd /var/lib/jenkins/workspace/${JOB\_NAME}/${CODE\_FOLDER\_NAME}/SharedModules/

ls > SM\_list.txt

A="`cat SM\_list.txt`"

shared\_module="`echo $A`"

for i in $shared\_module

do

module=`grep $i /var/lib/jenkins/workspace/${JOB\_NAME}/${CODE\_FOLDER\_NAME}/${CATEGORY}/${SERVICE\_NAME}/ReadMe.txt | wc -l`

if [ $module -eq 1 ]; then

echo "Shared module is present in the ReadMe.txt and has to be copied to the workspace"

cp -r $i /var/lib/jenkins/workspace/${JOB\_NAME}/$SERVICE\_NAME/

sed -i "s/Mattel.\*.parent/`echo $B`/g" /var/lib/jenkins/workspace/${JOB\_NAME}/${SERVICE\_NAME}/$i/pom.xml

else

echo "Shared Module not present in the ReadMe.txt"

fi

done

> SM\_list.txt

cd /var/lib/jenkins/workspace/${JOB\_NAME}/$SERVICE\_NAME

ls | grep 'Mattel.\*.application' > ms\_application.txt

App\_folder="`head -1 ms\_application.txt`"

'''

}

}

stage('Automated Code Review'){

steps {

step([$class: 'TibcoBartPipeline',

bartHome:'/opt/Bart\_home',

bartVer:'1.0',

projectName:"${PROJECT\_NAME}",

projectWorkSpace:"/var/lib/jenkins/workspace/${JOB\_NAME}/${SERVICE\_NAME}",

reportDir:"${workspace}"])

}

}

stage('Build') {

steps {

//build using pom.xml - specify the path of the parent pom

withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'subram',usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {

sh '''

mvn -f ${SERVICE\_NAME}/${PROJECT\_NAME}.parent/pom.xml clean install

cd /opt/git/dev/

if [ ! -d "${JOB\_NAME}" ]; then

mkdir -p ${JOB\_NAME}

fi

cd /opt/git/dev/${JOB\_NAME}/

rm -rf ConsumerMaster--GSL-

git init

git clone -b dev --single-branch https://$USERNAME:$PASSWORD@github.com/mattel-dig/ConsumerMaster--GSL-.git

cd /opt/git/dev/${JOB\_NAME}/ConsumerMaster--GSL-/${DEPLOY\_FOLDER\_NAME}/Earfiles/${CATEGORY}/

if [ ! -d "${SERVICE\_NAME}" ]; then

mkdir -p ${SERVICE\_NAME}

fi

cp /var/lib/jenkins/workspace/${JOB\_NAME}/${SERVICE\_NAME}/${PROJECT\_NAME}/target/${PROJECT\_NAME}\_1.0.0.ear /opt/git/dev/${JOB\_NAME}/ConsumerMaster--GSL-/${DEPLOY\_FOLDER\_NAME}/Earfiles/${CATEGORY}/${SERVICE\_NAME}/${PROJECT\_NAME}\_$(date +%Y%m%d\_%H%M%S).ear

cd /opt/git/dev/${JOB\_NAME}/ConsumerMaster--GSL-/${DEPLOY\_FOLDER\_NAME}/Earfiles/${CATEGORY}/${SERVICE\_NAME}/

git add ${PROJECT\_NAME}\_$(date +%Y%m%d\_%H)\*.ear

git commit -m "$(date +%Y%m%d\_%H%M)"

cd /opt/git/dev/${JOB\_NAME}/ConsumerMaster--GSL-/${DEPLOY\_FOLDER\_NAME}/Bart\_report/${CATEGORY}

if [ ! -d "${SERVICE\_NAME}" ]; then

mkdir -p ${SERVICE\_NAME}

fi

cp /var/lib/jenkins/workspace/${JOB\_NAME}/\*.html /opt/git/dev/${JOB\_NAME}/ConsumerMaster--GSL-/${DEPLOY\_FOLDER\_NAME}/Bart\_report/${CATEGORY}/${SERVICE\_NAME}/${SERVICE\_NAME}\_report\_$(date +%Y%m%d\_%H%M%S).html

cd /opt/git/dev/${JOB\_NAME}/ConsumerMaster--GSL-/${DEPLOY\_FOLDER\_NAME}/Bart\_report/${CATEGORY}/${SERVICE\_NAME}/

git add ${SERVICE\_NAME}\_report\_$(date +%Y%m%d\_%H%M)\*.html

git commit -m "$(date +%Y%m%d\_%H%M)"

git push https://$USERNAME:$PASSWORD@github.com/mattel-dig/ConsumerMaster--GSL-/ dev

rm -rf /opt/git/dev/${JOB\_NAME}/\*

'''

}

}

}

stage('Docker Containerisation'){

steps{

//Builds the container from Dockerfile

sh '''

cd /var/lib/jenkins/workspace/${JOB\_NAME}/${SERVICE\_NAME}/${PROJECT\_NAME}/target/

cp /var/lib/jenkins/workspace/${JOB\_NAME}/${DEPLOY\_FOLDER\_NAME}/dockerfiles/${CATEGORY}/${SERVICE\_NAME}/Dockerfile Dockerfile

docker build -t gcr.io/${JENKINS\_GCLOUD\_PROJECT\_ID}/cm-${SERVICE\_NAME}:$GIT\_COMMIT .

'''

}

}

stage('Image Publish to GCR'){

steps{

sh '''

#This gets the Git commit id

gcloud config set compute/zone ${JENKINS\_GCLOUD\_K8S\_CLUSTER\_ZONE}

gcloud config set compute/region ${JENKINS\_GCLOUD\_K8S\_CLUSTER\_REGION}

gcloud config set project ${JENKINS\_GCLOUD\_PROJECT\_ID}

gcloud auth configure-docker

#Pushes Docker images into GCR

docker push gcr.io/${JENKINS\_GCLOUD\_PROJECT\_ID}/cm-${SERVICE\_NAME}:$GIT\_COMMIT

'''

}

}

stage('Deployment to DEV GKE'){

steps{

withCredentials([file(credentialsId: 'mattelCreds', variable: 'mattel')]) {

sh '''

#Sets the env for gcloud

gcloud auth activate-service-account --key-file=${mattel}

gcloud config set compute/zone ${DEPLOY\_GCLOUD\_K8S\_CLUSTER\_ZONE\_DEV}

gcloud config set compute/region ${DEPLOY\_GCLOUD\_K8S\_CLUSTER\_REGION\_DEV}

gcloud config set project ${DEPLOY\_GCLOUD\_PROJECT\_ID\_DEV}

#Though --zone is mentioned for get-credentials ,provide the region

gcloud container clusters get-credentials ${DEPLOY\_GCLOUD\_K8S\_CLUSTER\_NAME\_DEV} --zone ${DEPLOY\_GCLOUD\_K8S\_CLUSTER\_REGION\_DEV}

sed -i s/latest/`echo $GIT\_COMMIT`/g /var/lib/jenkins/workspace/${JOB\_NAME}/${DEPLOY\_FOLDER\_NAME}/manifest.yml/DEV/${SERVICE\_NAME}\_dev.yml

DEPLOY\_STATE=`kubectl get deployments|grep "cm-${SERVICE\_NAME} " | wc -l`

if [ $DEPLOY\_STATE -eq 1 ]; then

echo "Deployment already exists! so updating the deployment"

kubectl apply -f /var/lib/jenkins/workspace/${JOB\_NAME}/${DEPLOY\_FOLDER\_NAME}/manifest.yml/DEV/${SERVICE\_NAME}\_dev.yml

kubectl rollout status deployment cm-${SERVICE\_NAME}

else

echo "Creating a new deployment"

kubectl create -f /var/lib/jenkins/workspace/${JOB\_NAME}/${DEPLOY\_FOLDER\_NAME}/manifest.yml/DEV/${SERVICE\_NAME}\_dev.yml

kubectl rollout status deployment cm-${SERVICE\_NAME}

fi

'''

}

}

}

/\* stage('DEV-SanityTesting&checkforRollback'){

steps{

withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'devApiCreds',usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {

sh '''

sleep 140

#post deployment verification and rollback

#post deployment script for dev\_deploymentaddress.yml file

curl -X GET --header 'Accept: application/json' --header ${PARAMETERS} ${URL} -u $USERNAME:$PASSWORD > result.txt

status=`grep -E "Bad Request|Server Error" result.txt| wc -l`

echo $status

if [ $status -eq 0 ]; then

echo "Deployment has been rolled out successfully"

echo $GIT\_COMMIT

echo $GIT\_COMMIT >> /opt/docker\_tag/phase1b\_tag/dev\_docker\_tag/${CATEGORY}/${SERVICE\_NAME}\_tag.txt

else

echo "Deployment wasn't successfull, rolling back the deploy to the previous successfull image"

image=`tail -n 1 /opt/docker\_tag/phase1b\_tag/dev\_docker\_tag/${CATEGORY}/${SERVICE\_NAME}\_tag.txt`

echo "Last successfull $image image is going to be deployed"

kubectl apply -f /var/lib/jenkins/workspace/${JOB\_NAME}/${DEPLOY\_FOLDER\_NAME}/manifest.yml/DEV/${SERVICE\_NAME}\_dev.yml

kubectl rollout status deployment cm-${SERVICE\_NAME}-devops

fi

rm -rf result.txt

'''

}

}

} \*/

}

}

Deployment.yml file :

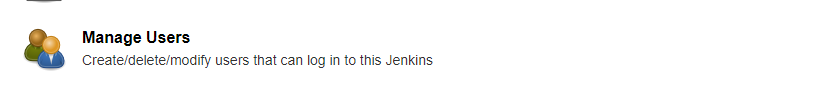
|  |
| --- |
| apiVersion: v1 |
|  | kind: Service |
|  | metadata: |
|  | name: cm-bulkupload |
|  | labels: |
|  | app: cm-bulkupload |
|  | spec: |
|  | type: LoadBalancer |
|  | ports: |
|  | - port: 80 |
|  | targetPort: 8080 |
|  | selector: |
|  | app: cm-bulkupload |
|  | --- |
|  | apiVersion: apps/v1 |
|  | kind: Deployment |
|  | metadata: |
|  | name: cm-bulkupload |
|  | spec: |
|  | replicas: 1 |
|  | selector: |
|  | matchLabels: |
|  | app: cm-bulkupload |
|  | template: |
|  | metadata: |
|  | name: cm-bulkupload |
|  | labels: |
|  | app: cm-bulkupload |
|  | spec: |
|  | containers: |
|  | - name: cm-bulkupload |
|  | image: gcr.io/mattel-tools/cm-bulkupload:latest |
|  | imagePullPolicy: Always |
|  | env: |

…………….

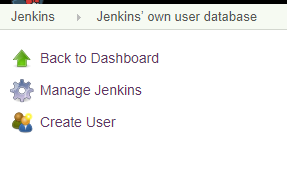
**2.8. ROLE BASED STATERGY:**

Create user accounts:

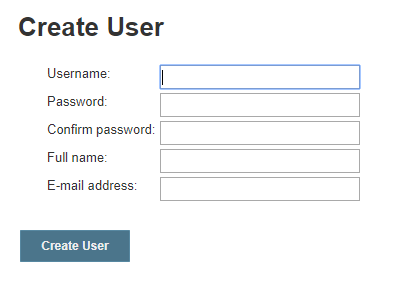
1.Click Manage users under manage jenkins



2.Click create new user

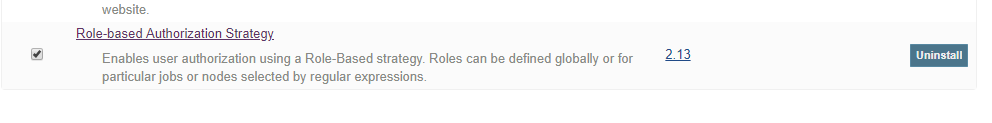


3.Populate the necessary details



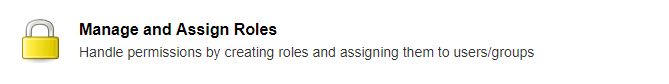
Install the plugin:

1.Install role based Authorization Stratergy

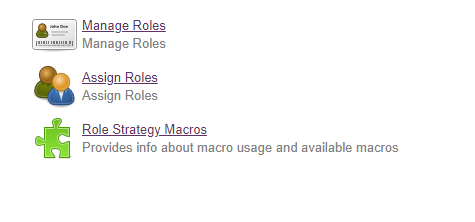


Assign role:

1. Manage and Assign Roles Appears Under manage Jenkins only after the plugin is installed



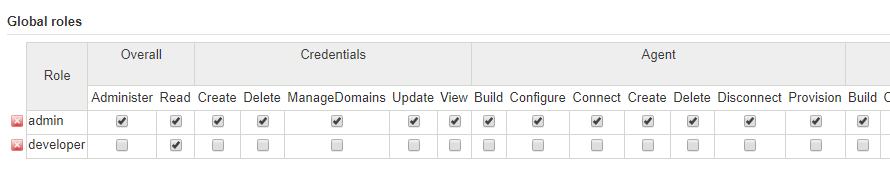
2.Click Manage Roles



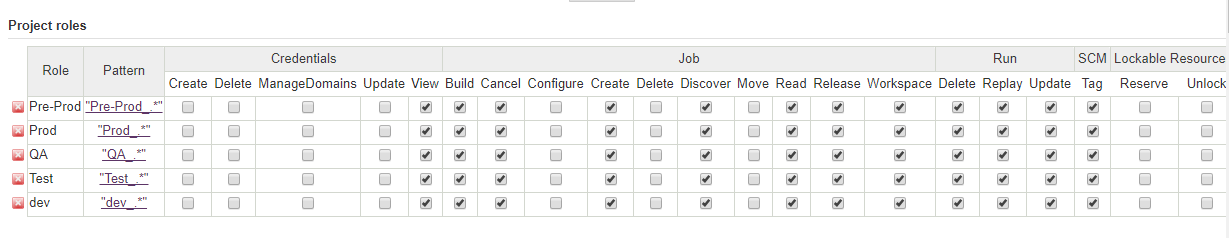
3.Manage Roles:

A.Give the permisions for the roles added

Provide read permision to developer

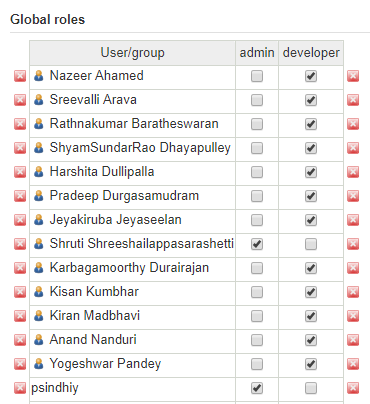


B.Add the patterns to access the jobs in Jenkins eg: dev\_.\*

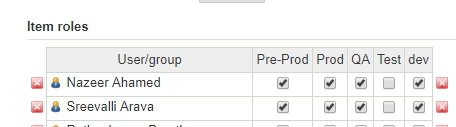


4.Assign Roles

A.Assign the roles to users added

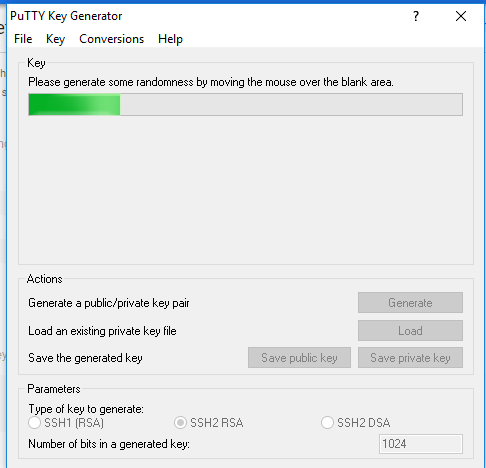


B.Give the permissions to the job patterns

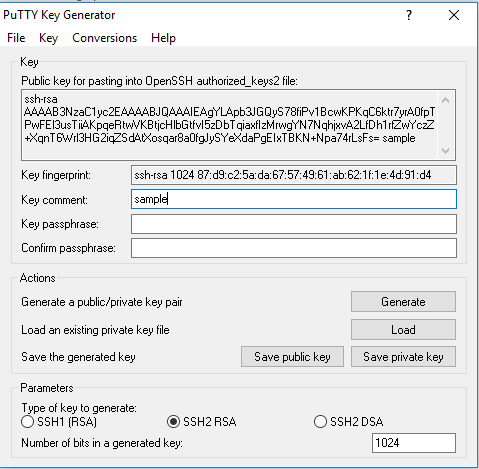


**2.9. CONNECT TO INSTANCE IN COMPUTE ENGINE:**

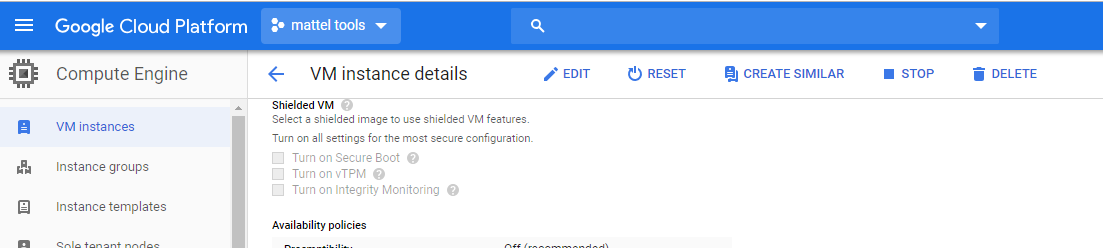
1.Open Putty gen



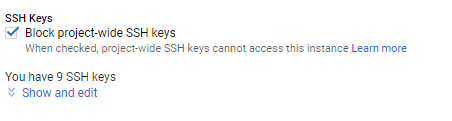
2.Provide the key comment and the passphrase (password)



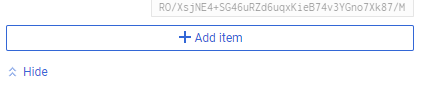
3.Click edit under the respective VM instance in GCP



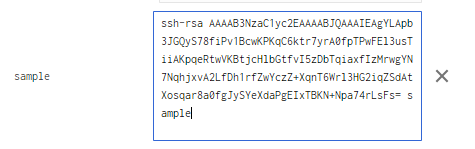
4.Click Show and edit



5.Click Add item

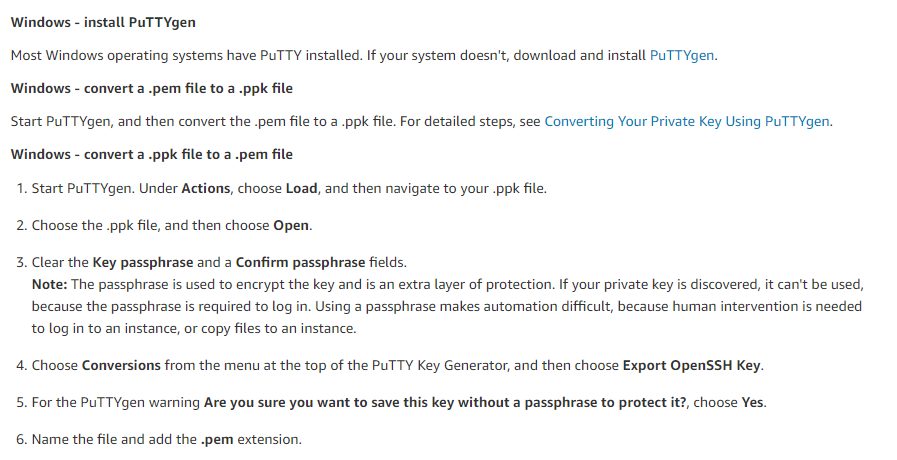


6.Paste the public key from puttygen to GCP



7. Save

8.Under Putty gen create the .ppk file usning .pem file of instance



9.use the .ppk to connect to instance via putty

10.login using the username and password.