gcloud config set compute/zone us-east1-b

git clone [https://source.developers.google.com/p/$DEVSHELL\_PROJECT\_ID/r/sample-app](https://source.developers.google.com/p/%24DEVSHELL_PROJECT_ID/r/sample-app)

gcloud container clusters get-credentials jenkins-cd

kubectl create clusterrolebinding cluster-admin-binding --clusterrole=cluster-admin --user=$(gcloud config get-value account)

export POD\_NAME=$(kubectl get pods --namespace default -l "[app.kubernetes.io/component=jenkins-master](http://app.kubernetes.io/component%3Djenkins-master)" -l "[app.kubernetes.io/instance=cd](http://app.kubernetes.io/instance%3Dcd)" -o jsonpath="{.items[0].[metadata.name](http://metadata.name/)}")

kubectl port-forward $POD\_NAME 8080:8080 >> /dev/null &

printf $(kubectl get secret cd-jenkins -o jsonpath="{.data.jenkins-admin-password}" | base64 --decode);echo

//To get to the Jenkins user interface, click on the Web Preview button in cloud shell, then click Preview on port 8080

//You should now be able to log in with username admin and your auto-generated password. cd sample-app

kubectl create ns production

kubectl apply -f k8s/production -n production kubectl apply -f k8s/canary -n production kubectl apply -f k8s/services -n production kubectl get svc

kubectl get service gceme-frontend -n production

//Check the IP in a new browser window, you will notice the color and the version of the app if associated with version 1.0.0. git init

git config credential.helper gcloud.sh

git remote add origin [https://source.developers.google.com/p/$DEVSHELL\_PROJECT\_ID/r/sample-app](https://source.developers.google.com/p/%24DEVSHELL_PROJECT_ID/r/sample-app)

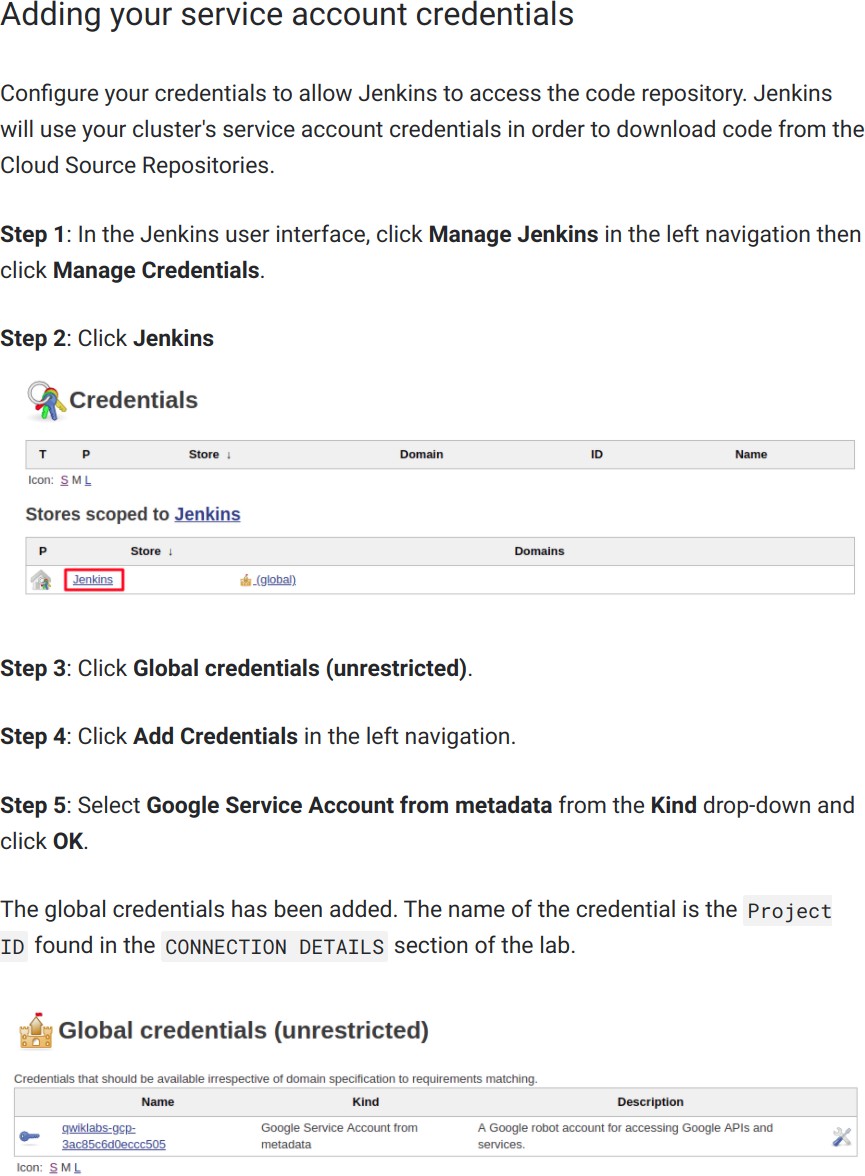
git config --global user.email "[EMAIL\_ADDRESS]" git config --global [user.name](http://user.name/) "[USERNAME]"

//Set the username and email address for your Git commits. Replace [EMAIL\_ADDRESS] with your Git email address and [USERNAME] with your

Git username git add .

git commit -m "Initial commit" git push origin master

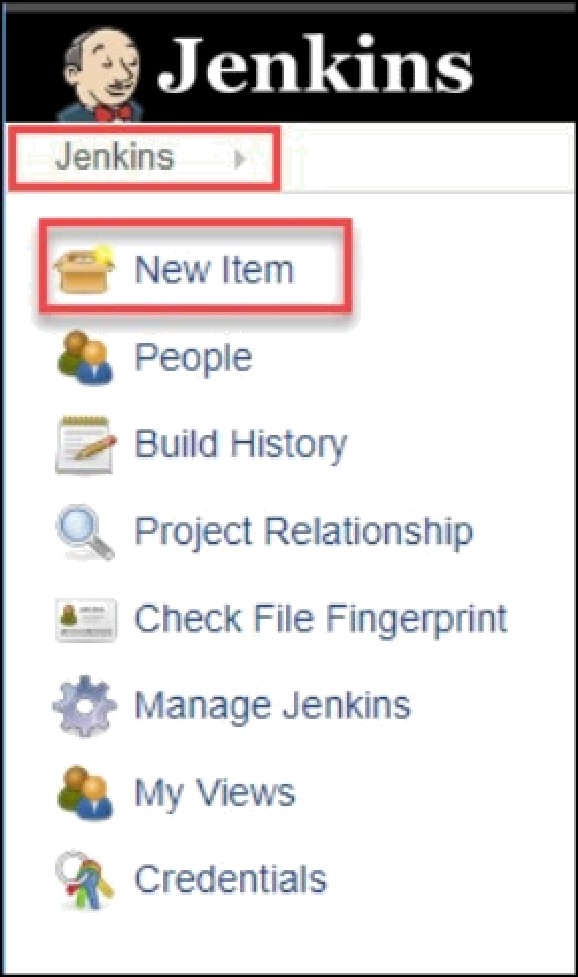
//Now create the Jinkins pipeline as per the images below.



# Creating the Jenkins job

Navigate to your Jenkins user interface and follow these steps to configure a Pipeline job.

Step 1: Click **Jenkins** » New Item in the left navigation:



Step 2: Name the project sample-app, then choose the Multibranch Pipeline option and click OK.

Step 3: On the next page, in the Branch Sources section, click Add Souree and select

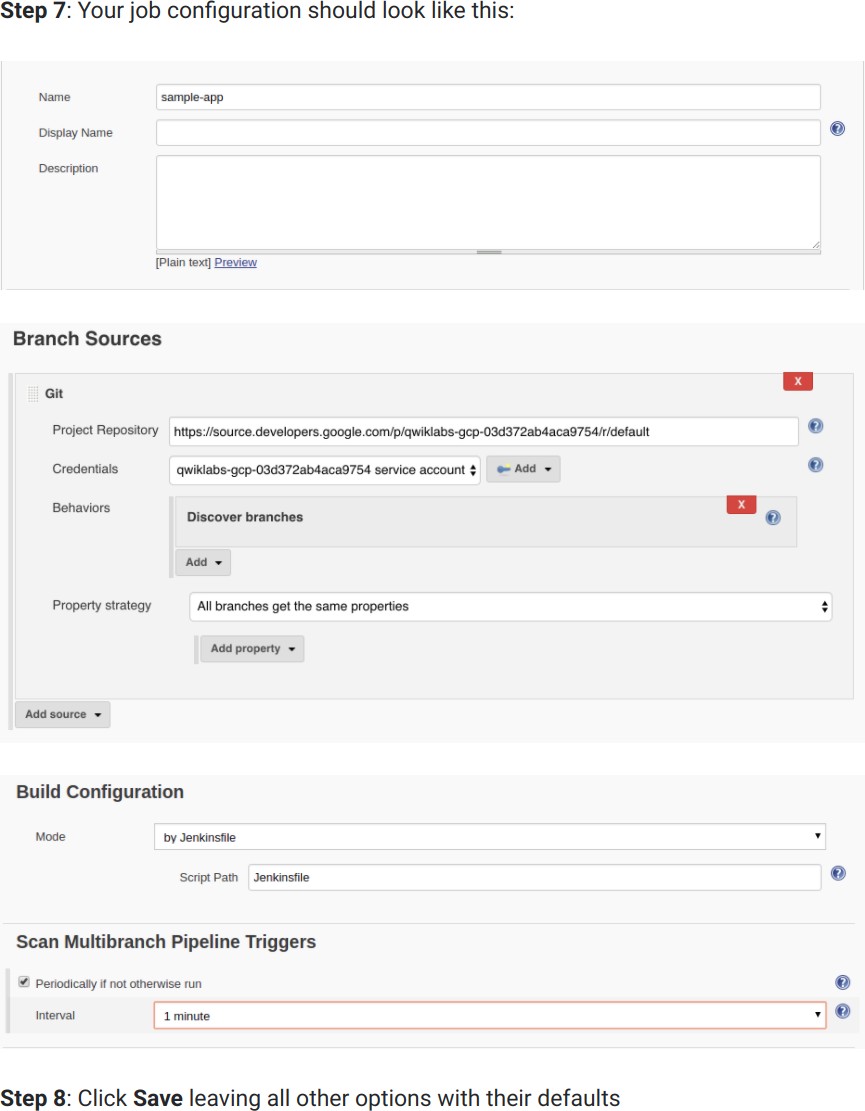
Step 4: Paste the HTTPS clone URL of your sample-app repo in Cloud Source Repositories into the Project Repository field. Replace [ ROJECT ID With your Project ID:

# https://source.developers.google.com/p/(PR0JECT\_ID]/r/sample-app

where [PROJECT ID ] must be replaced by your lab project ID.

# Step 5: From the Credentials drop-down, select the name of the credentials you created when adding your service account in the previous steps.

Step 6: Under Scan Multibranch Pipeline Triggers section, check the Periodically if not otherwise run box and set the Interval value to 1 minute.



NOTE: Wait for the Pipeline and the master to complete Successfully. To monitor the master, go to Jenkins homepage and click on the Sample- App option, inside that you will see the progress of master. Wait until it turns blue. Refresh the page to see the progress update. Make sure that both turn blue.

git checkout -b new-feature

vi html.go //Change the two instances of <div class="card blue"> to <div class="card orange"> vi main.go //Update the version to "const version string = "2.0.0" "

git add Jenkinsfile html.go main.go git commit -m "Version 2.0.0"

git push origin new-feature

//Wait for 1-2 mins and you will see the new-feature branch. Monitor the new-feature branch to be completed successfully. It will turn blue once succeeded.

kubectl proxy &

// If it stalls, press Ctrl + C to exit out.

curl \

http://localhost:8001/api/v1/namespaces/new-feature/services/gceme-frontend:80/proxy/version

kubectl get service gceme-frontend -n production

//Open the IP in a new browser tab to check the application. git checkout -b canary

git push origin canary

//Wait for 1-2 mins and you will see the canary branch. Monitor the canary branch to be completed successfully. It will turn blue once succeeded.

export FRONTEND\_SERVICE\_IP=$(kubectl get -o \

jsonpath="{.status.loadBalancer.ingress[0].ip}" --namespace=production services gceme-frontend)

while true; do curl http://$FRONTEND\_SERVICE\_IP/version; sleep 1; done

//If you keep seeing 1.0.0, try running the above commands again. Once you've verified that the above works, end the command with Ctrl + C. git checkout master

git merge canary

git push origin master

//Wait for 1-2 mins and you will see the master branch. Monitor the master branch to be completed successfully. It will turn blue once succeeded.

export FRONTEND\_SERVICE\_IP=$(kubectl get -o \

jsonpath="{.status.loadBalancer.ingress[0].ip}" --namespace=production services gceme-frontend)

while true; do curl http://$FRONTEND\_SERVICE\_IP/version; sleep 1; done

//Once again, if you see instances of 1.0.0 try running the above commands again. You can stop this command by pressing Ctrl + C.

kubectl get service gceme-frontend -n production

//Check the IP in a new browser window, you will notice that the color and the version of the app must be updated.