Implement DevOps in Google Cloud: Challenge Lab

Author: Vedant Kakde | **GitHub Profile:** github.com/vedant-kakde | **LinkedIn Profile:** linkedin.com/in/vedant-kakde/

Task 1: Configure a Jenkins pipeline for continuous deployment to Kubernetes Engine

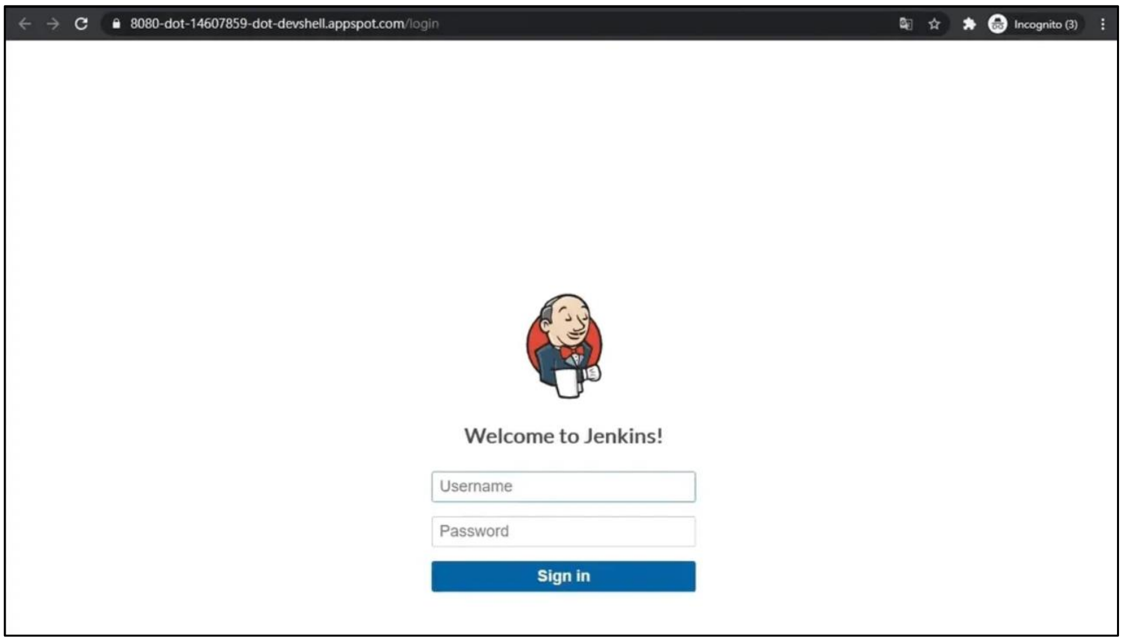
Run these commands one by one

```
gcloud config set compute/zone us-east1-b
```

Here replace project id in command then run.

```
git clone https://source.developers.google.com/p/<YOUR_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" -l "app.kubernetes.io/instance=cd" -o
jsonpath="{.items[0].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
```

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



4. You should now be able to log in with username `admin` and your auto-generated password.

Clone the Sample App Repository and Create the Production Namespace

1. Go back to the SSH window, clone the `sample-app` source code from the Cloud Source Repository by running the following command:

```
gcloud source repos clone sample-app --project=<YOUR-PROJECT-ID>
```

Remember to replace `<YOUR-PROJECT-ID>` with your GCP project ID

2. Change to the sample application directory and then create the Kubernetes namespace to logically isolate the deployment:

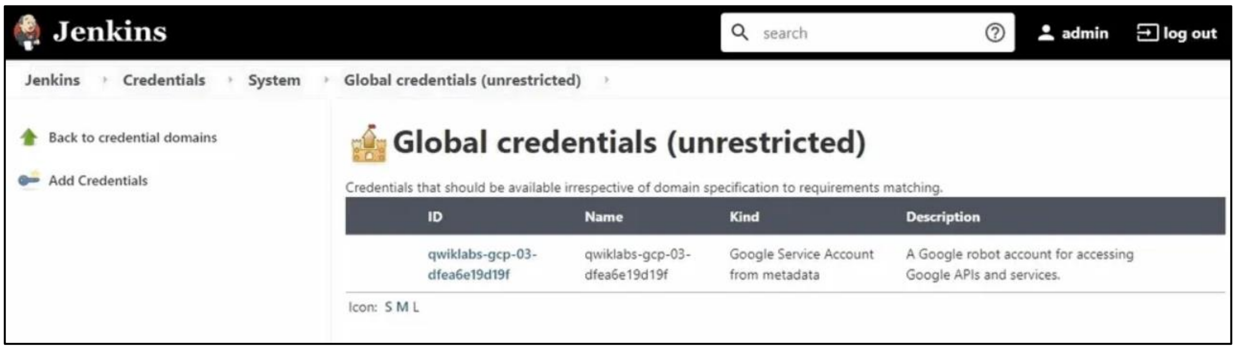
```
cd sample-app

kubectl create ns production
```

Creating the Jenkins Pipeline

Adding your service account credentials

Configure your credentials to allow Jenkins to access the code repository. Jenkins will use your cluster's service account credentials in order to download code from the Cloud Source Repositories.

1. In the Jenkins user interface, click **Manage Jenkins** in the left navigation then click **Manage Credentials**.
2. Click **Jenkins**
3. Click **Global credentials (unrestricted)**.
4. Click **Add Credentials** in the left navigation.
5. Select **Google Service Account from metadata** from the **Kind** drop-down and click **OK**.


Jenkins
search
admin
log out
Jenkins › Credentials › System › Global credentials (unrestricted) ›
Back to credential domains
Add Credentials

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
qwiklabs-gcp-03-dfea6e19d19f	qwiklabs-gcp-03-dfea6e19d19f	Google Service Account from metadata	A Google robot account for accessing Google APIs and services.

Icon: S M L

Configure the Jenkins job

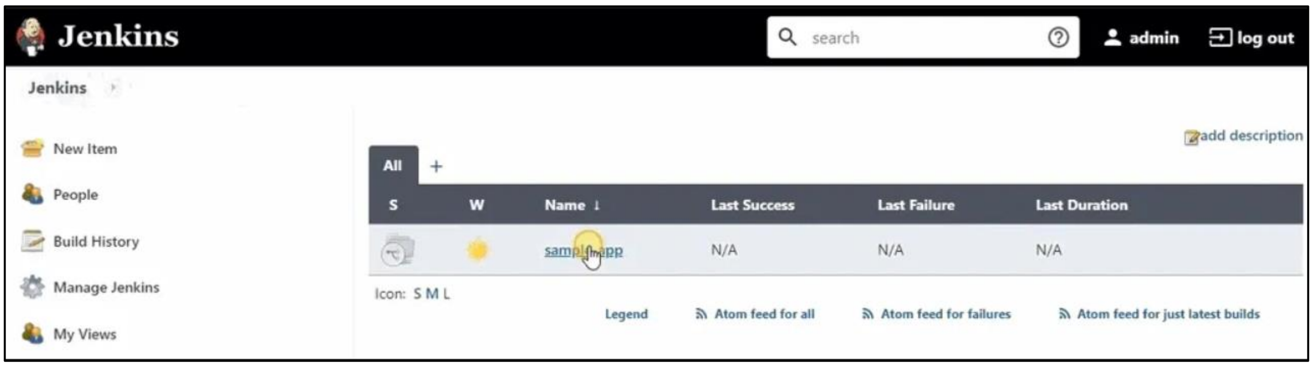
After configuring your credentials, follow these steps to configure a Pipeline job.

1. Click **Jenkins** to return the welcome page.

2. Reload the page a few times until you see a pipeline job named **sample-app** in the Jenkins user interface.

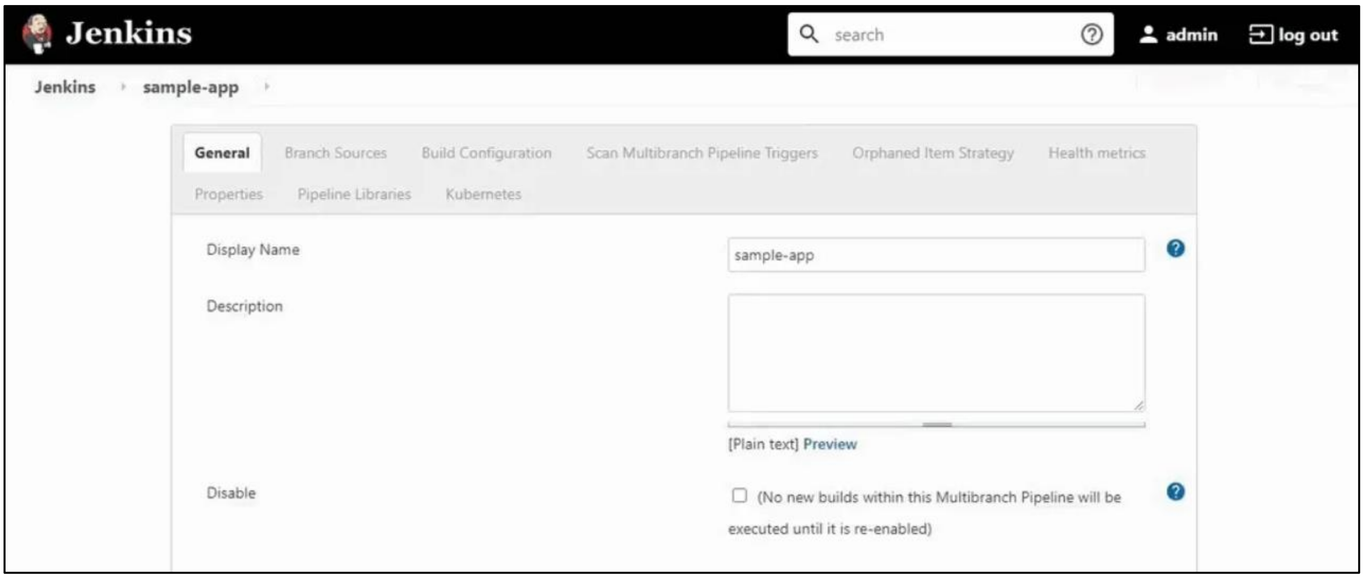
   (If you fail to get any job, click **Jenkins** > **New Item** in the left navigation to create the multibranch pipeline.)

3. Click on the job name of **sample-app**.



S	W	Name ↓	Last Success	Last Failure	Last Duration
		sample-app	N/A	N/A	N/A

4. Click **Configure** in the left pane.
5. In the configuration page, enter `sample-app` as the Display Name in the General section.



Display Name: sample-app

Description:

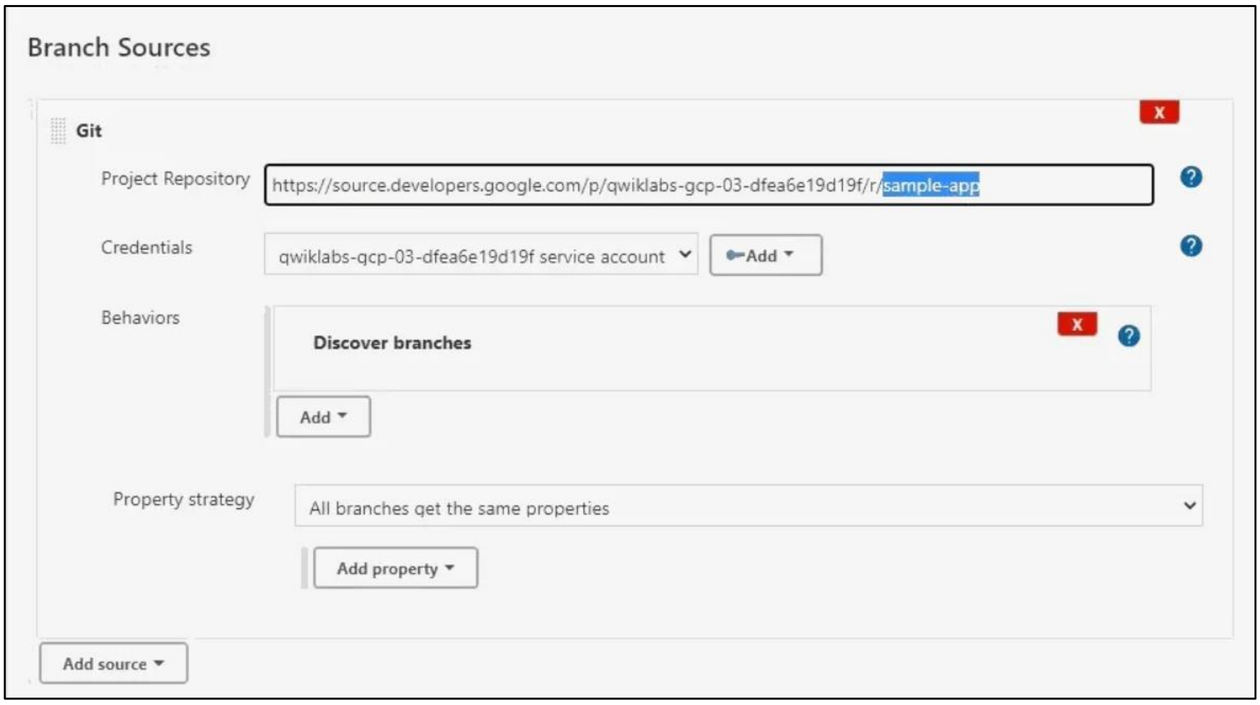
[Plain text] Preview

Disable: ☐ (No new builds within this Multibranch Pipeline will be executed until it is re-enabled)

6. In the Branch Sources section, click on **Add source** and choose **Git** from the dropdown menu.
7. Copy the Cloud Source URL of the **sample-app** repository to the **Project Repository** field. The URL should look like this:

```
https://source.developers.google.com/p/<YOUR-PROJECT-ID>/r/sample-app
```

Select the service account for your GCP project from the **Credentials** dropdown list.


Branch Sources

Git

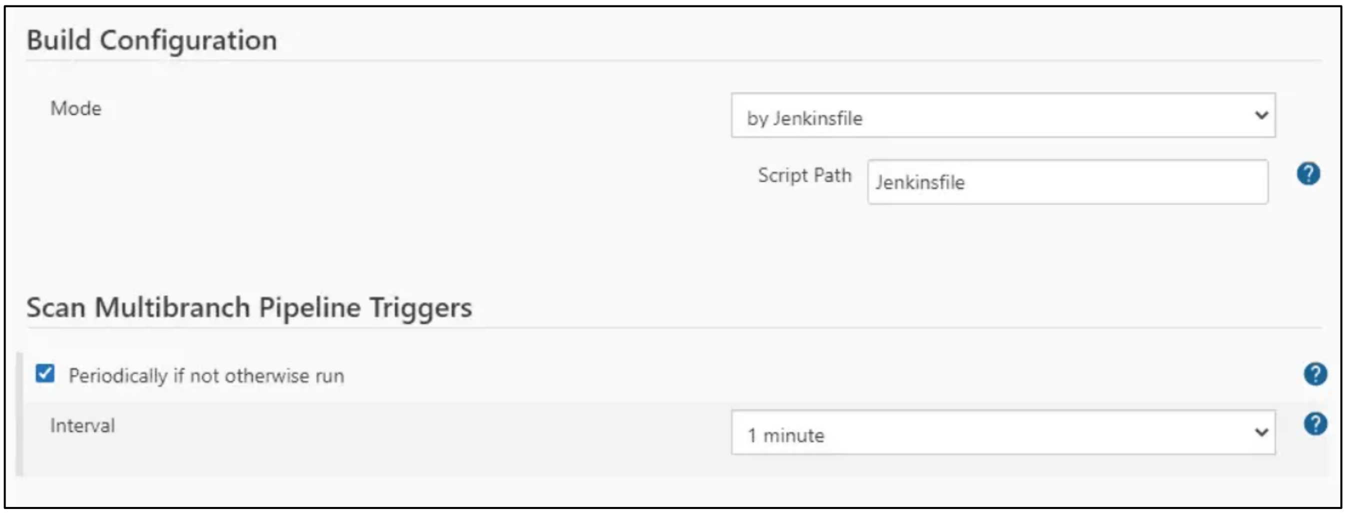
Project Repository: https://source.developers.google.com/p/qwiklabs-gcp-03-dfea6e19d19f/r/sample-app

Credentials: qwiklabs-gcp-03-dfea6e19d19f service account

Behaviors: Discover branches

Property strategy: All branches get the same properties

8. Check **Periodically if not otherwise run** in the **Scan Multibranch Pipeline Trigger** section, and then select **1 minute** for the interval.


Build Configuration

Mode: by Jenkinsfile

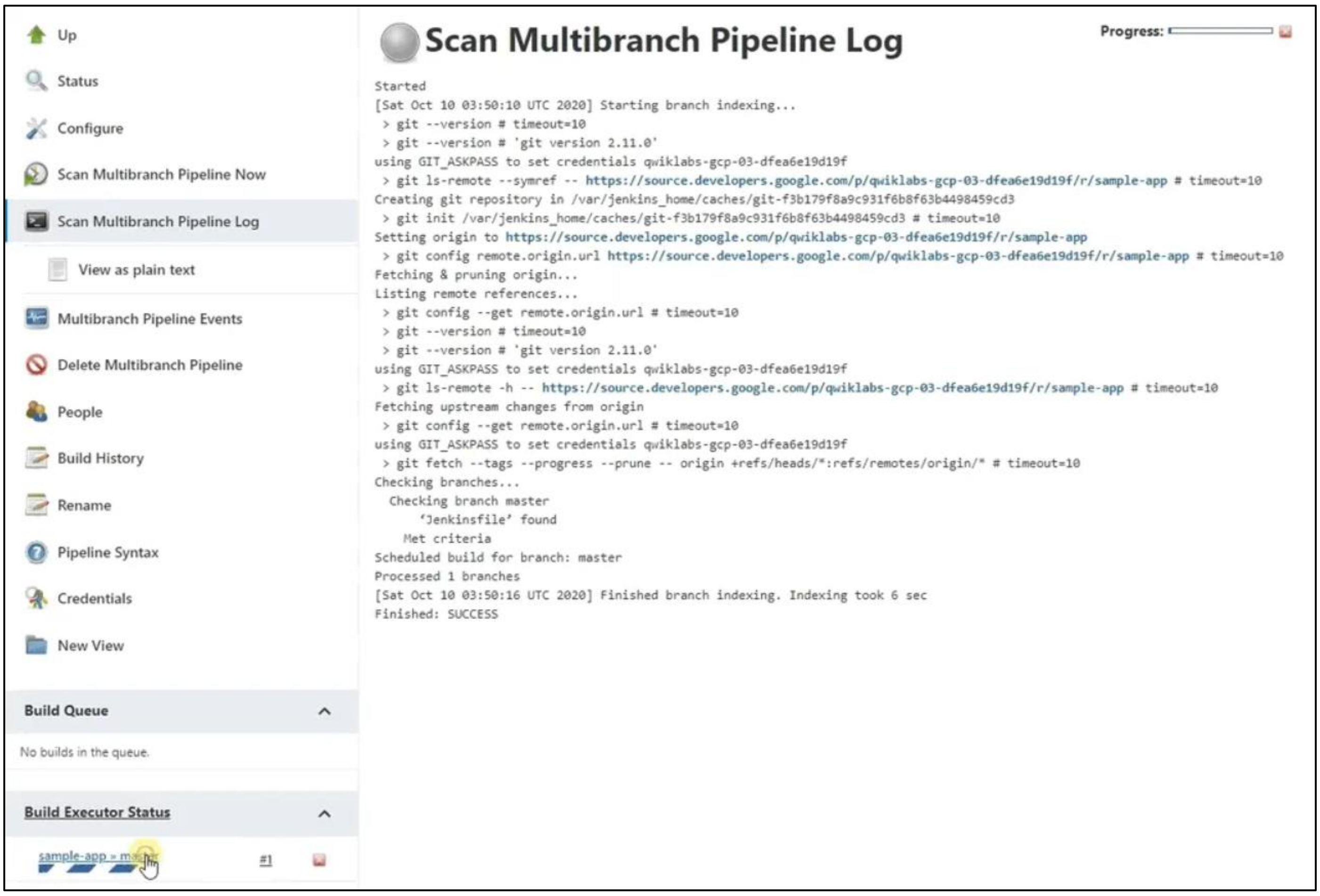
Script Path: Jenkinsfile

Scan Multibranch Pipeline Triggers

☑ Periodically if not otherwise run

Interval: 1 minute

9. Scroll to the bottom and click on **Save**.
10. If you correctly configured the pipeline, you will see **Finished: SUCCESS** at the end of the Scan Multibranch Pipeline Log.

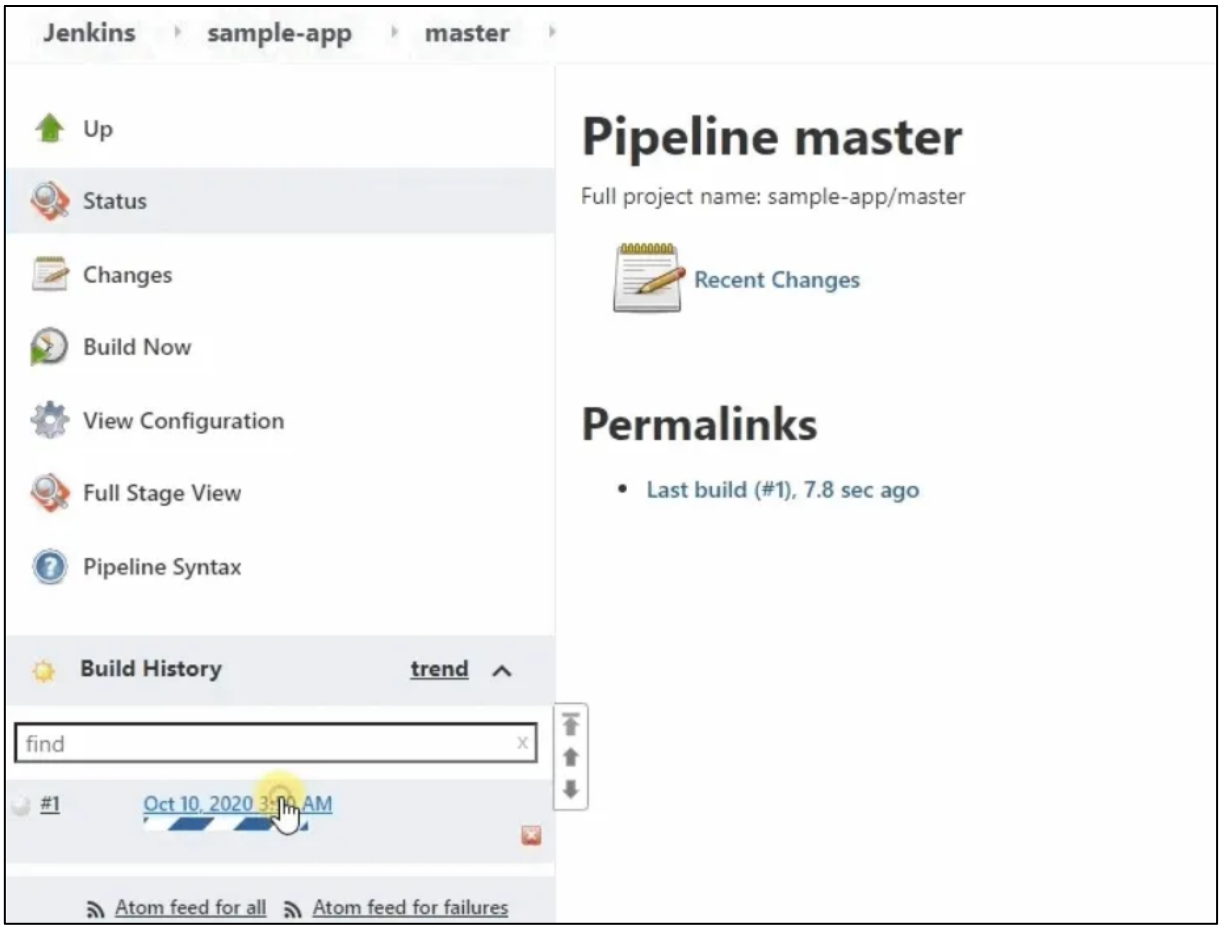


Scan Multibranch Pipeline Log

```
Started
[Sat Oct 10 03:50:10 UTC 2020] Starting branch indexing...
 > git --version # timeout=10
 > git --version # 'git version 2.11.0'
using GIT_ASKPASS to set credentials qwiklabs-gcp-03-dfea6e19d19f
 > git ls-remote --symref -- https://source.developers.google.com/p/qwiklabs-gcp-03-dfea6e19d19f/r/sample-app # timeout=10
Creating git repository in /var/jenkins_home/caches/git-f3b179f8a9c931f6b8f63b4498459cd3
 > git init /var/jenkins_home/caches/git-f3b179f8a9c931f6b8f63b4498459cd3 # timeout=10
Setting origin to https://source.developers.google.com/p/qwiklabs-gcp-03-dfea6e19d19f/r/sample-app
 > git config remote.origin.url https://source.developers.google.com/p/qwiklabs-gcp-03-dfea6e19d19f/r/sample-app # timeout=10
Fetching & pruning origin...
Listing remote references...
 > git config --get remote.origin.url # timeout=10
 > git --version # timeout=10
 > git --version # 'git version 2.11.0'
using GIT_ASKPASS to set credentials qwiklabs-gcp-03-dfea6e19d19f
 > git ls-remote -h -- https://source.developers.google.com/p/qwiklabs-gcp-03-dfea6e19d19f/r/sample-app # timeout=10
Fetching upstream changes from origin
 > git config --get remote.origin.url # timeout=10
using GIT_ASKPASS to set credentials qwiklabs-gcp-03-dfea6e19d19f
 > git fetch --tags --progress --prune -- origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking branches...
  Checking branch master
      'Jenkinsfile' found
    Met criteria
Scheduled build for branch: master
Processed 1 branches
[Sat Oct 10 03:50:16 UTC 2020] Finished branch indexing. Indexing took 6 sec
Finished: SUCCESS
```

Examine the Console Output from the Jenkins Pipeline

1. Click on **sample-app » master** under Build Executor Status in the left pane, then click on the job under Build History.



Pipeline master

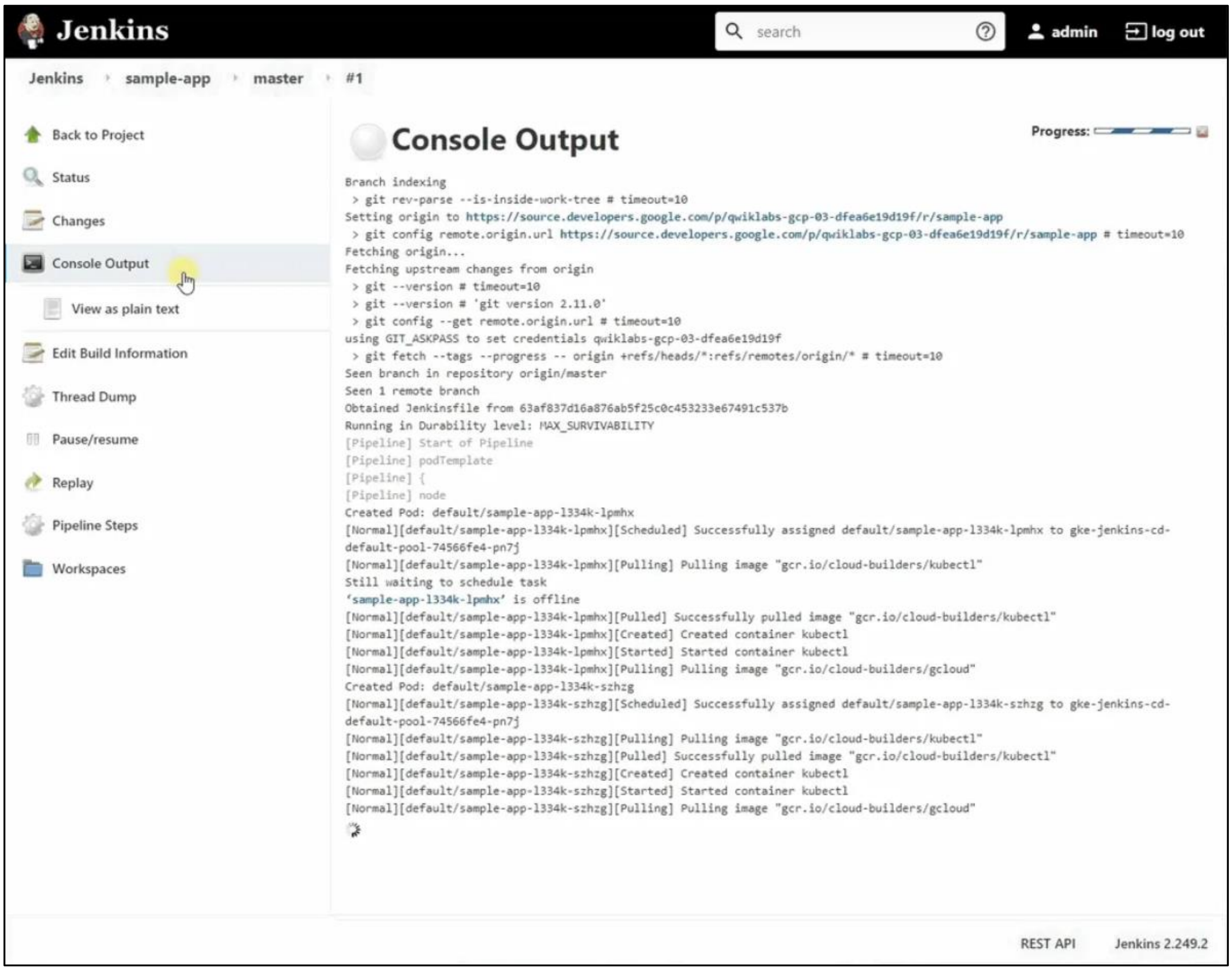
Full project name: sample-app/master

Recent Changes

Permalinks

- Last build (#1), 7.8 sec ago

2. Click **Console Output** in the left navigation to monitor the build process.



Console Output

```
Branch indexing
 > git rev-parse --is-inside-work-tree # timeout=10
Setting origin to https://source.developers.google.com/p/qwiklabs-gcp-03-dfea6e19d19f/r/sample-app
 > git config remote.origin.url https://source.developers.google.com/p/qwiklabs-gcp-03-dfea6e19d19f/r/sample-app # timeout=10
Fetching origin...
Fetching upstream changes from origin
 > git --version # timeout=10
 > git --version # 'git version 2.11.0'
 > git config --get remote.origin.url # timeout=10
using GIT_ASKPASS to set credentials qwiklabs-gcp-03-dfea6e19d19f
 > git fetch --tags --progress -- origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/master
Seen 1 remote branch
Obtained Jenkinsfile from 63af837d16a876ab5f25c0c453233e67491c537b
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] podTemplate
[Pipeline] {
[Pipeline] node
Created Pod: default/sample-app-1334k-lpmhx
[Normal][default/sample-app-1334k-lpmhx][Scheduled] Successfully assigned default/sample-app-1334k-lpmhx to gke-jenkins-cd-
default-pool-74566fe4-pn7j
[Normal][default/sample-app-1334k-lpmhx][Pulling] Pulling image "gcr.io/cloud-builders/kubectl"
Still waiting to schedule task
'sample-app-1334k-lpmhx' is offline
[Normal][default/sample-app-1334k-lpmhx][Pulled] Successfully pulled image "gcr.io/cloud-builders/kubectl"
[Normal][default/sample-app-1334k-lpmhx][Created] Created container kubectl
[Normal][default/sample-app-1334k-lpmhx][Started] Started container kubectl
[Normal][default/sample-app-1334k-lpmhx][Pulling] Pulling image "gcr.io/cloud-builders/gcloud"
Created Pod: default/sample-app-1334k-szhzg
[Normal][default/sample-app-1334k-szhzg][Scheduled] Successfully assigned default/sample-app-1334k-szhzg to gke-jenkins-cd-
default-pool-74566fe4-pn7j
[Normal][default/sample-app-1334k-szhzg][Pulling] Pulling image "gcr.io/cloud-builders/kubectl"
[Normal][default/sample-app-1334k-szhzg][Pulled] Successfully pulled image "gcr.io/cloud-builders/kubectl"
[Normal][default/sample-app-1334k-szhzg][Created] Created container kubectl
[Normal][default/sample-app-1334k-szhzg][Started] Started container kubectl
[Normal][default/sample-app-1334k-szhzg][Pulling] Pulling image "gcr.io/cloud-builders/gcloud"
```

This process takes about five minutes to complete.

Task 2: Push an update to the application to a development branch

1. Go back to the SSH window, run the following command to create a development branch in the `sample-app` directory.

```
git checkout -b new-feature
```

2. Open the **main.go** file with `vi`, `nano`, or the Cloud Shell Editor.
3. Update the version in the following line:

```
const version string = "1.0.0"
```

   to

```
const version string = "2.0.0"
```

   then save the change.

4. Open the **html.go** file and change the two instances of `<div class="card blue">` to `<div class="card orange">`.
5. Commit and push your changes by running the following commands:

```
git config --global user.email "you@example.com"

git config --global user.name "Your Name"

git add Jenkinsfile html.go main.go

git commit -m "Version 2.0.0"

git push origin new-feature
```

6. Go to the Jenkins user interface, click **Scan Multibranch Pipeline Now** if you want to trigger the build immediately.

Task 3: Push a Canary deployment to the production namespace

1. Go back to the SSH window, run the following command to create a canary branch in the `sample-app` directory.

```
git checkout -b canary
```

2. Merge the change from the development branch:

```
git merge new-feature
```

3. Push the canary to the Git server:

```
git push origin canary
```

4. Go to the Jenkins user interface, click **Scan Multibranch Pipeline Now** if you want to trigger the build immediately.

Task 4: Promote the Canary Deployment to production

1. Go back to the SSH window, run the following commands to merge the canary branch, and push it to the Git server.

```
git checkout master

git merge canary

git push origin master
```

2. Go to the Jenkins user interface, click **Scan Multibranch Pipeline Now** if you want to trigger the build immediately.

Congratulations! You completed this challenge lab.