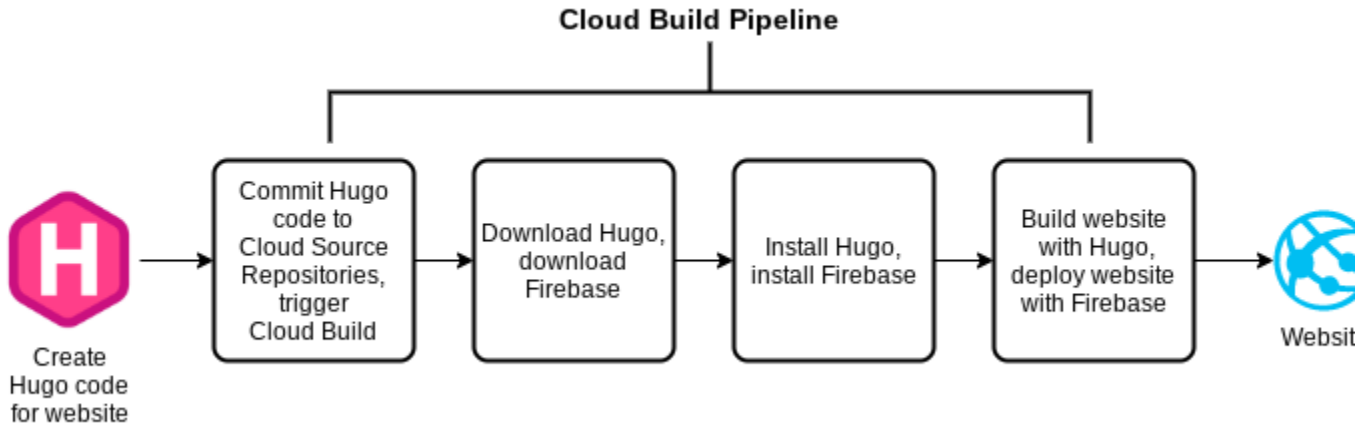
Deploy a Hugo Website with Cloud Build and Firebase Pipeline

Process overview

Here's a diagram of what you are going to build:



The goal is to be able to commit code and have it trigger the pipeline, which in turn deploys the website. Your journey is in two parts. First, you build the website locally and manually deploy it to Firebase. Second, you automate the process by building a pipeline with Cloud Build.

Task 1. Manual deployment

First build the website manually on a Linux instance to learn the end-to-end process. You also use the Linux instance to perform some of the one-time tasks needed to get Firebase up and running.

Connect to the Linux instance

1. From the **Navigation menu** (☰) select **Compute Engine** > **VM Instances**. Notice the instance that has been built for you.

At the end of the line you should see an External IP address and an SSH button as shown in the figure below. If these are obscured by an information panel, close that panel so you can see the entire line.

2. Record the External IP address for later use.
3. Click **SSH**. A window opens and you see a shell prompt.

Install Hugo

Now install Hugo in the Linux instance to locally test the website before you deploy it with Firebase. This lab provides a shell script to make this easier.

1. In the Linux instance shell, examine the file `installhugo.sh`:

```
cat /tmp/installhugo.sh
```

Copied!
content_copy
Your output should be similar to the sample output below:

Output:

```
#!/bin/bash

# Copyright 2020 Google Inc. All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
```

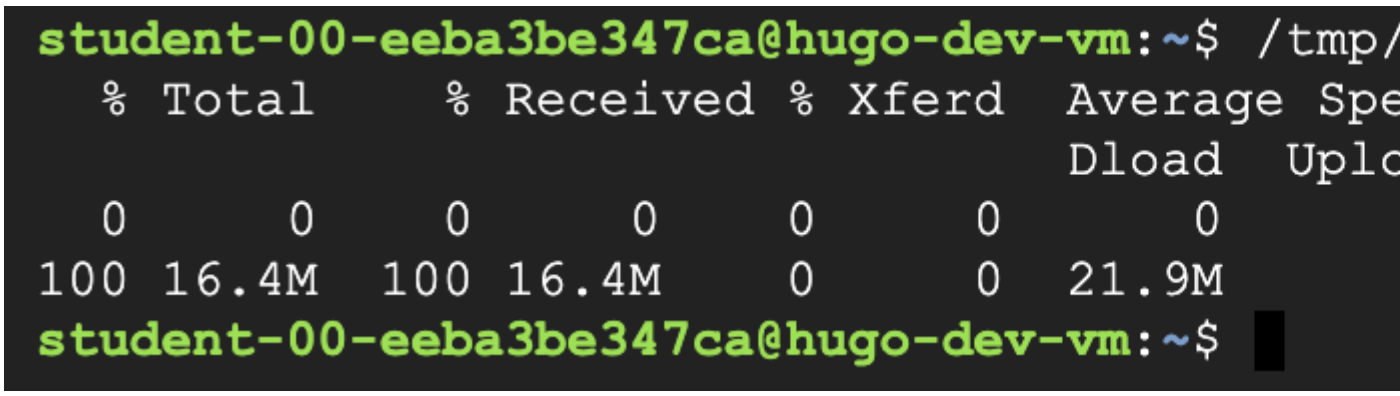
```
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.

_HUGO_VERSION=0.96.0

echo Downloading Hugo version $_HUGO_VERSION...

wget \
  --quiet \
  -O hugo.tar.gz \

https://github.com/gohugoio/hugo/releases/download/v${_HUGO_VERSION}/hu
go_extended_${_HUGO_VERSION}_Linux-64bit.tar.gz

echo Extracting Hugo files into /tmp...

mv hugo.tar.gz /tmp
tar -C /tmp -xzf /tmp/hugo.tar.gz

echo The Hugo binary is now at /tmp/hugo.
```

Note the use of the `wget` command to download Hugo and the `tar` command to unpack the Hugo archive. You will see similar commands later in this lab when you create the pipeline.

2. Enter the commands below to run the script and install Hugo:

```
cd ~
/tmp/installhugo.sh
```

When complete, you receive a message saying that Hugo has been installed into the `/tmp` directory as shown below.



You are now ready to build the website infrastructure.

Create a repository and the initial web site

Create a GitHub repository to hold the web site and then clone the repository to the Linux instance.

Cloning a repository creates a mirror of it in the shell. This allows you to implement the web site while in the shell, and then later commit your changes to the file system. Later in this lab, you will set up a pipeline that responds to these commits to the repository.

1. Install git and GitHub CLI on the Linux VM and set your project ID , project number and region. Save them as `PROJECT_ID`, `PROJECT_NUMBER` and `REGION` variables.

Enter the following commands in the Linux instance shell:

```
export PROJECT_ID=$(gcloud config get-value project)
export PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --
format="value(projectNumber)")
export REGION=$(gcloud compute project-info describe \
--format="value(commonInstanceMetadata.items[google-compute-default-
region])")
sudo apt-get update
sudo apt-get install git
sudo apt-get install gh
```

If prompted, answer `Yes` to all prompts.

2. Run the following commands to configure Git and GitHub:

```
curl -sS https://webi.sh/gh | sh

gh auth login
gh api user -q ".login"
GITHUB_USERNAME=$(gh api user -q ".login")
git config --global user.name "${GITHUB_USERNAME}"
git config --global user.email "${USER_EMAIL}"
echo ${GITHUB_USERNAME}
echo ${USER_EMAIL}
```

3. Enter the following commands to create and clone a code repository:

```
cd ~
gh repo create  my_hugo_site --private
gh repo clone  my_hugo_site
```

The output provides confirmations about the creation of the repository and the cloning of the repository as shown in the figure below. Ignore the two warning messages about the repository charge and that the repository is empty.

Now you are ready to create the site structure.

4. Enter the following commands below in the Linux shell:

```
cd ~
/tmp/hugo new site my_hugo_site --force
```

Normally the `hugo` command creates the directory. The `--force` option will create the site in the repository directory, which already exists. This allows you to keep the Git-related information in the directory that you just cloned.

When complete, you see messages that the site has been created.

4. Now install the **hello-friend-ng** theme to provide a layout for your site. Enter the following commands in the Linux instance shell:

```
cd ~/my_hugo_site
git clone \
  https://github.com/rhazdon/hugo-theme-hello-friend-ng.git
themes/hello-friend-ng
echo 'theme = "hello-friend-ng"' >> config.toml
```

When complete, you see messages indicating that the theme has been cloned.

5. Remove the git files from the themes directory:

```
sudo rm -r themes/hello-friend-ng/.git
sudo rm themes/hello-friend-ng/.gitignore
```

Note: The git files should be removed so that GitHub repository adds the theme files to version control.

6. With the web site structure set up, you can now preview it. Enter the command below to launch the site at TCP port 8080:

```
cd ~/my_hugo_site
/tmp/hugo server -D --bind 0.0.0.0 --port 8080
```

Hugo builds the site and serves it for access on TCP port 8080. The server runs until it is stopped by pressing CTRL+C.

7. Open a browser tab and browse to the external IP address at port 8080. Use the following URL, replacing `[EXTERNAL IP]` with the external IP address of your instance:

```
http://[EXTERNAL IP]:8080
```

The web site should look like this.

8. Go back to the Linux shell and press CTRL+C to stop the Hugo server.

Deploy the site to Firebase

1. Install Firebase CLI in the Linux instance shell:

```
curl -sL https://firebase.tools | bash
```

Copied!
content_copy

2. Now you need to initialize Firebase. Enter the command below into the shell:

```
cd ~/my_hugo_site
firebase init
```

3. Select **Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys** using the arrow keys and spacebar and press ENTER.

- When asked for a project option, select **Use an existing project**, then use the arrow keys, spacebar, and the ENTER key to select the Project ID, `qwiklabs-gcp-01-befdb27da3b2`.
- For the public directory, select the default value **public**.
- For configuring as a single page application, select the default value of **N**.

For setting up automatic builds and deploys with GitHub, select **N**.

If asked to overwrite any existing files, select Y.

4. You are ready to deploy the application. Enter the commands below into the Linux instance shell to rebuild the site with Hugo and to deploy it with Firebase:

```
/tmp/hugo && firebase deploy
```

5. After the application has been deployed, you receive a hosting URL. Click on it and to see the same website being served from the Firebase CDN (content delivery network).

If you receive a generic welcome message, wait a few minutes for the CDN to initialize and refresh the browser window. Record this hosting URL for later use.

You have now performed the entire deployment locally. Next, you automate the process from end to end using Cloud Build.

Task 2. Automate the deployment

Perform the initial commit

The goal of building the pipeline is to be able to trigger builds when changes are made to the repository. You start by performing an initial commit to the repository to validate your ability to make future changes.

1. Configure the git commands global parameters by entering the commands below into the Linux shell. Make sure to include the quotation marks:

```
git config --global user.name "hugo"
git config --global user.email "hugo@blogger.com"
```

2. In the Linux shell, enter the commands below to create a `.gitignore` file to exclude certain directories from the repository:

```
cd ~/my_hugo_site
echo "resources" >> .gitignore
```

3. Perform the initial commit to the repository:

```
git add .
git commit -m "Add app to GitHub Repository"
git push -u origin master
```

You have now committed (uploaded) the initial version of the website to Google Cloud.

Configure the build

Cloud Build uses a file named cloudbuild.yaml in the root directory of the repository to perform the build. The file is in YAML format. Spacing and indentation are important, so it has already been placed on the Linux instance for you.

1. In the Linux shell, enter the following command. Note the final period (".") at the end of the `cp` command:

```
cd ~/my_hugo_site
cp /tmp/cloudbuild.yaml .
```

2. Run the following command to see what the `cloudbuild.yaml` file looks like.

```
cat cloudbuild.yaml
```

Some lines in the output have wrapped because of their length.

Output:

```
# Copyright 2020 Google Inc. All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.

steps:
- name: 'gcr.io/cloud-builders/curl'
  args:
  - '--quiet'
  - '-O'
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

```
steps:
- name: 'gcr.io/cloud-builders/curl'
  args:
  - '--quiet'
  - '-O'
  - 'firebase'
  - 'https://firebase.tools/bin/linux/latest'

- name: 'gcr.io/cloud-builders/curl'
  args:
  - '--quiet'
  - '-O'
  - 'hugo.tar.gz'
  -
'https://github.com/gohugoio/hugo/releases/download/v${_HUGO_VERSION}/h
ugo_extended_${_HUGO_VERSION}_Linux-64bit.tar.gz'
  waitFor: ['-']

- name: 'ubuntu:20.04'
  args:
  - 'bash'
  - '-c'
  - |
    mv hugo.tar.gz /tmp
    tar -C /tmp -xzf /tmp/hugo.tar.gz
    mv firebase /tmp
    chmod 755 /tmp/firebase
    /tmp/hugo
    /tmp/firebase deploy --project ${PROJECT_ID} --non-interactive --
only hosting -m "Build ${BUILD_ID}"

substitutions:
  _HUGO_VERSION: 0.96.0
options:
  defaultLogsBucketBehavior: REGIONAL_USER_OWNED_BUCKET
```

3. Here are some observations about the `cloudbuild.yaml` file:

- There are three named steps in this file, each of which is performed by a container image. The first two steps use a Google-supported builder to use `curl` to download the Hugo and Firebase tools. These two steps run in parallel. Using the curl builder is faster than installing `curl` manually.
- The third step uses a standard Ubuntu container to install Hugo and Firebase after which the site is built and deployed. Installing Hugo and Firebase for each deployment allows you to change the version of Hugo whenever you desire while also using the latest version of Firebase.
- The `tar` and `wget` commands are nearly identical to those used earlier in the `installhugo.sh` script.
- The file also uses a custom substitution variable (_HUGO_VERSION) and a Google-provided substitution variable (PROJECT_ID) to allow for this template to be used in different environments.
- The Hugo and Firebase binaries are created and installed in a temporary directory so that they do not inadvertently get deployed to the website itself.

Connect to a GitHub repository and create a Cloud Build repository

1. Initiate a connection to your GitHub repository in the Linux instance.

```
gcloud builds connections create github cloud-build-connection --
project=$PROJECT_ID  --region=$REGION

gcloud builds connections describe cloud-build-connection --
region=$REGION
```

2. Find the `actionUri`:

```
etag: yKV297keFBHzs1UcgMsbYJlEYvYdIkfFLJMYZfOADu8
githubConfig: {}
installationState:
  actionUri:
  https://accounts.google.com/AccountChooser?continue=https%3A%2F%2
  Fconsole.cloud.google.com%2Fm%2Fgcb%2Fgithub%2Flocations%2Fus-
  east4%2Foauth_v2%3Fconnection_name%3Dprojects%252F921646058273%25
  2Flocations%252Fus-east4%252Fconnections%252Fcloud-build-
  connection
  message: Please log in to https://github.com using a robot
  account and then follow
 this link to authorize Cloud Build to access that account. After
  authorization,
 your GitHub authorization token will be stored in Cloud Secret
  Manager.
   stage: PENDING_USER_OAUTH
 name: projects/qwiklabs-gcp-00-40e7d6bb49bb/locations/us-
  east4/connections/cloud-build-connection
 reconciling: false
updateTime: '2024-12-12T08:52:48.505263316Z'
```

13. Open the `actionUri` in a new browser tab.

14. Click **Continue**.

Install the Cloud Build GitHub App in your account or in an organization you own. Permit the installation using your GitHub account.

4. Under **Repository access**, choose **Only select repositories**. Click **Select repositories** and select the repository `my_hugo_site`.

5. Click **Save**.

6. Create a Cloud Build repository:

```
gcloud builds repositories create hugo-website-build-repository \
```

```
  --remote-uri="https://github.com/${GITHUB_USERNAME}/my_hugo_site.git"
\
  --connection="cloud-build-connection" --region=$REGION
```

Create the Cloud Build trigger

Now create a trigger that responds to commits to the master branch of the repository.

1. In the Linux instance shell, enter the following command:

```
gcloud builds triggers create github --name="commit-to-master-branch1"
\
   --
repository=projects/$PROJECT_ID/locations/$REGION/connections/cloud-
build-connection/repositories/hugo-website-build-repository \
   --build-config='cloudbuild.yaml' \
   --service-
account=projects/$PROJECT_ID/serviceAccounts/$PROJECT_NUMBER-
compute@developer.gserviceaccount.com \
   --region=$REGION \
   --branch-pattern='^master$'
```

The Cloud Build service account

The Cloud Build service account must have permissions to use Firebase to deploy the website.

Cloud Build	Role	Description
[PROJECT_NUMBER@cloudbuild.gserviceaccount.com	roles/firebasehosting.admin	Full read/write access to Hosting resources

Test the pipeline

Now that you have created the pipeline, make a change to the site then commit it to see if the change propagates.

1. In the Linux shell enter the command below to move to the repository directory:

```
cd ~/my_hugo_site
```

2. Edit the file **config.toml** and change the title:

```
Blogging with Hugo and Cloud Build
```

Copied!
content_copy

3. Enter the commands below to commit the changes to the repository and trigger the Cloud Build pipeline:

```
git add .
git commit -m "I updated the site title"
git push -u origin master
```

4. Check the build history to see the status of the build:

```
gcloud builds list --region=$REGION
```

5. Check the build logs for the current build:

```
gcloud builds log --region=$REGION $(gcloud builds list --
format='value(ID)' --filter=$(git rev-parse HEAD) --region=$REGION)
```

6. Grab the URL from the build performed:

```
gcloud builds log "$(gcloud builds list --format='value(ID)' --
filter=$(git rev-parse HEAD) --region=$REGION)" --region=$REGION | grep
"Hosting URL"
```

7. Browse to the hosting URL to see the results.

You can also go to the Firebase console and examine the project to find the domain name.

Note: It may take a few minutes for the CDN to update and display the updated site information.**Note:** The site has an SSL certificate and is accessed using the https (Hypertext Transfer Protocol Secure) protocol.