Deploy a Hugo Website with Cloud Build and Firebase Pipeline

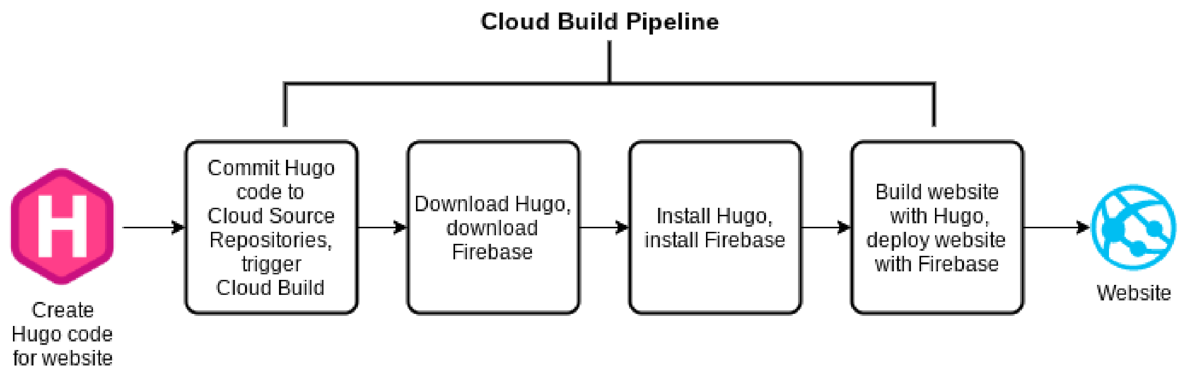
Objectives

In this lab you perform the following:

- Read a static website overview.
- Set up a website with Hugo.
- Store website content in a GitHub repository.
- Deploy the website with Firebase
- Create a build pipeline with Cloud Build to automate the deployment.

Process overview

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


Cloud Build Pipeline
Create Hugo code for website
Commit Hugo code to Cloud Source Repositories, trigger Cloud Build
Download Hugo, download Firebase
Install Hugo, install Firebase
Build website with Hugo, deploy website with Firebase
Website

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.

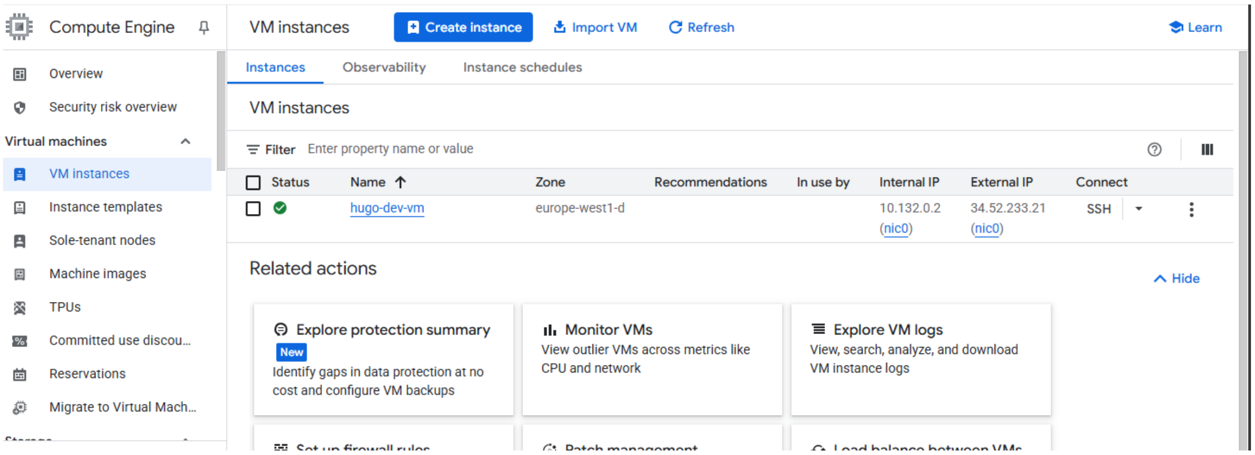
Connect to the Linux instance

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.



Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	hugo-dev-vm	europe-west1-d			10.132.0.2 (nic0)	34.52.233.21 (nic0)	SSH

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.

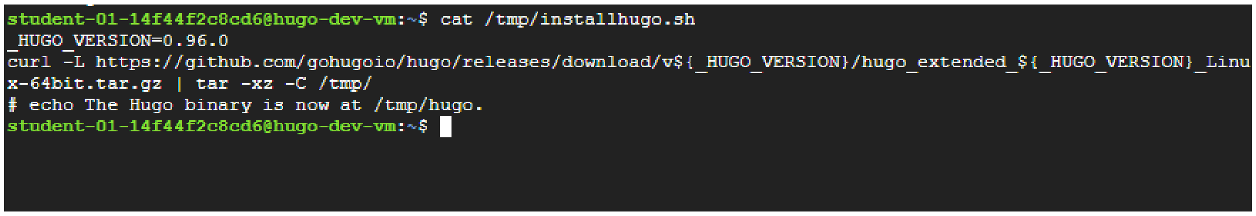
1. In the Linux instance shell, examine the file installhugo.sh:

cat /tmp/installhugo.sh

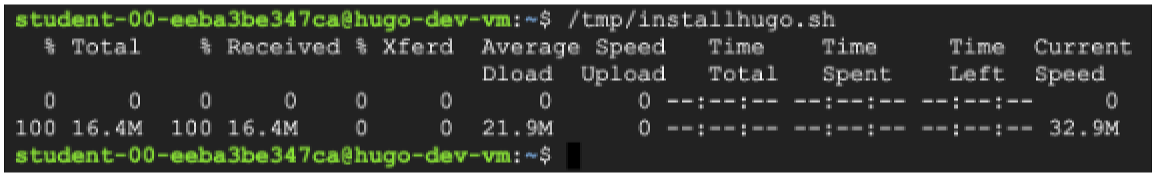
2. Enter the commands below to run the script and install Hugo:

cd ~

/tmp/installhugo.sh



```
student-01-14f44f2c8cd6@hugo-dev-vm:~$ cat /tmp/installhugo.sh
_HUGO_VERSION=0.96.0
curl -L https://github.com/gohugoio/hugo/releases/download/v${_HUGO_VERSION}/hugo_extended_${_HUGO_VERSION}_Linu
x-64bit.tar.gz | tar -xz -C /tmp/
# echo The Hugo binary is now at /tmp/hugo.
student-01-14f44f2c8cd6@hugo-dev-vm:~$ 
```



```
student-00-eeba3be347ca@hugo-dev-vm:~$ /tmp/installhugo.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 16.4M  100 16.4M    0     0  21.9M      0 --:--:-- --:--:-- --:--:-- 32.9M
student-00-eeba3be347ca@hugo-dev-vm:~$
```

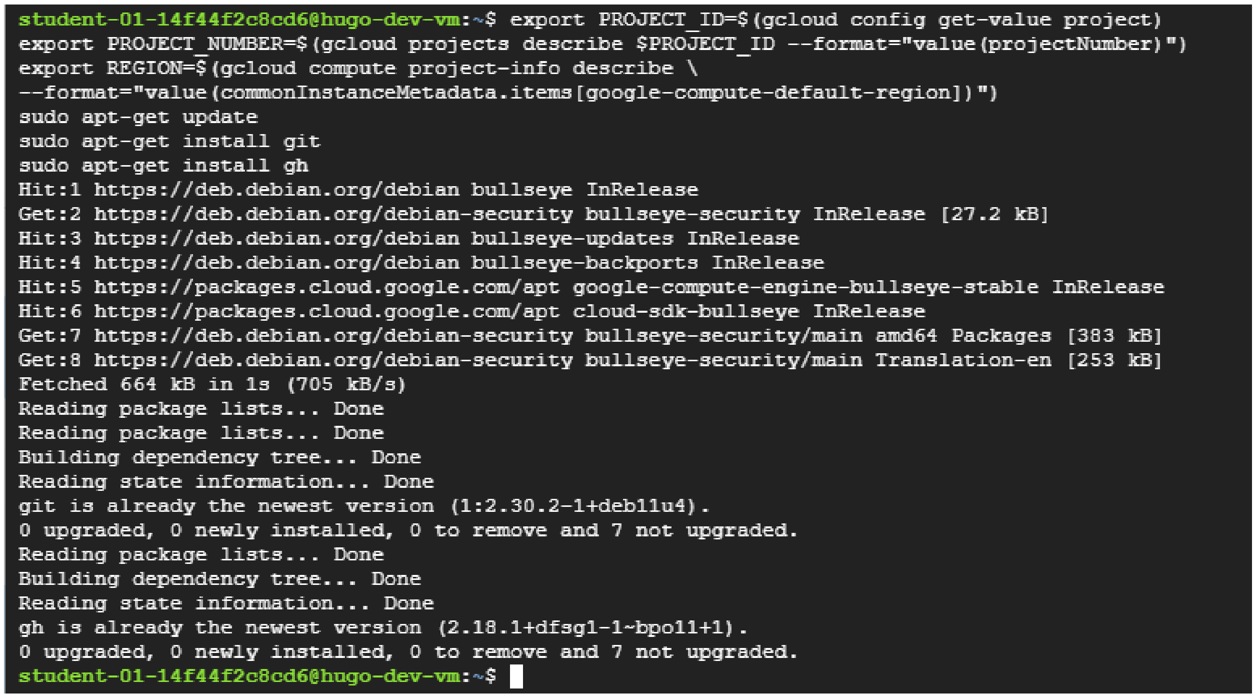
Create a repository and the initial web site

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.

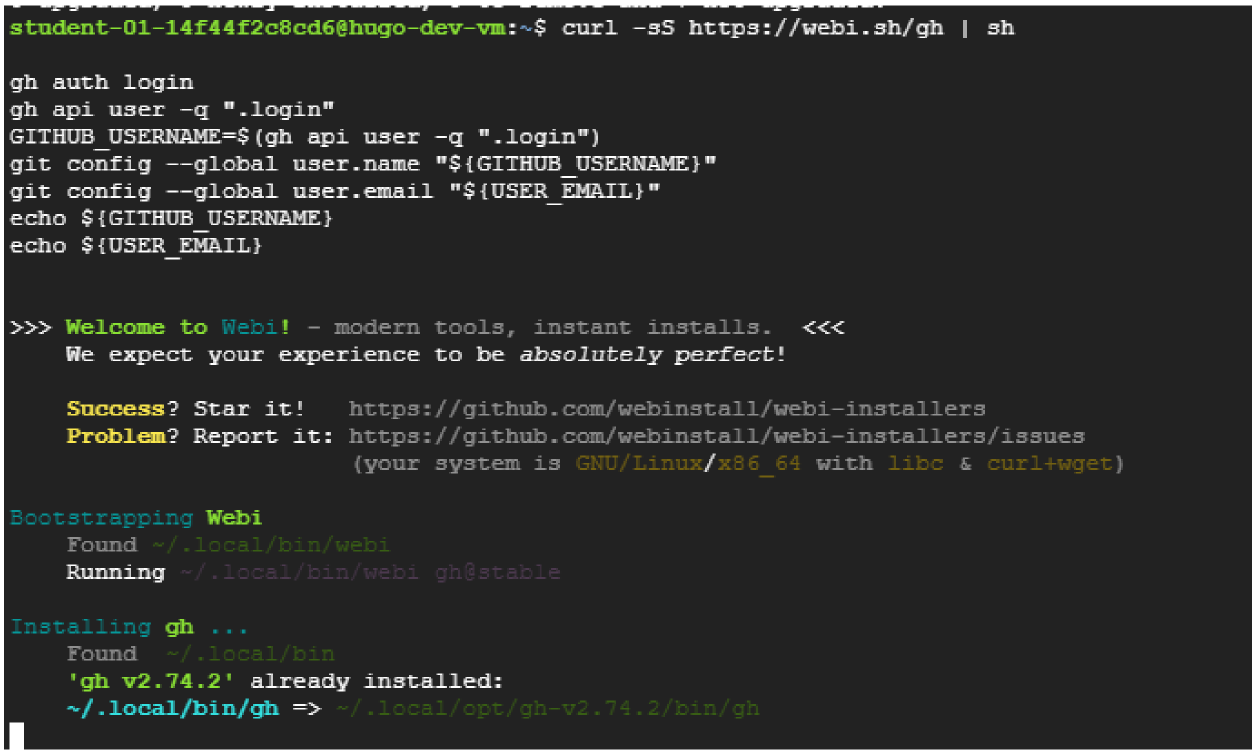
1. Install git and GitHub CLI on the Linux VM and set your project ID , project number and region. Save them as PROJECT_ID, PROJECT_NUMBER and REGION variables.

Enter the following commands in the Linux instance shell:



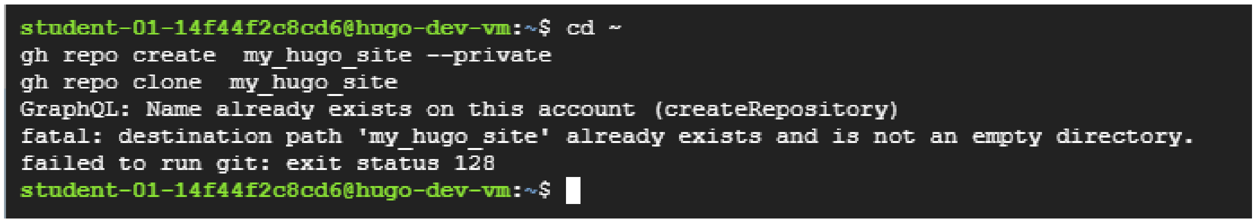
```
student-01-14f44f2c8cd6@hugo-dev-vm:~$ export PROJECT_ID=$(gcloud config get-value project)
export PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --format="value(projectNumber)")
export REGION=$(gcloud compute project-info describe \
--format="value(commonInstanceMetadata.items[google-compute-default-region])")
sudo apt-get update
sudo apt-get install git
sudo apt-get install gh
Hit:1 https://deb.debian.org/debian bullseye InRelease
Get:2 https://deb.debian.org/debian-security bullseye-security InRelease [27.2 kB]
Hit:3 https://deb.debian.org/debian bullseye-updates InRelease
Hit:4 https://deb.debian.org/debian bullseye-backports InRelease
Hit:5 https://packages.cloud.google.com/apt google-compute-engine-bullseye-stable InRelease
Hit:6 https://packages.cloud.google.com/apt cloud-sdk-bullseye InRelease
Get:7 https://deb.debian.org/debian-security bullseye-security/main amd64 Packages [383 kB]
Get:8 https://deb.debian.org/debian-security bullseye-security/main Translation-en [253 kB]
Fetched 664 kB in 1s (705 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.30.2-1+deb11u4).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gh is already the newest version (2.18.1+dfsg1-1~bpo11+1).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
student-01-14f44f2c8cd6@hugo-dev-vm:~$
```

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



```
student-01-14f44f2c8cd6@hugo-dev-vm:~$ 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}


>>> Welcome to Webi! - modern tools, instant installs.  <<<
    We expect your experience to be absolutely perfect!

    Success? Star it!   https://github.com/webinstall/webi-installers
    Problem? Report it: https://github.com/webinstall/webi-installers/issues
                        (your system is GNU/Linux/x86_64 with libc & curl+wget)

Bootstrapping Webi
    Found  ~/.local/bin/webi
    Running ~/.local/bin/webi gh@stable

Installing gh ...
    Found  ~/.local/bin
    'gh v2.74.2' already installed:
    ~/.local/bin/gh => ~/.local/opt/gh-v2.74.2/bin/gh
```

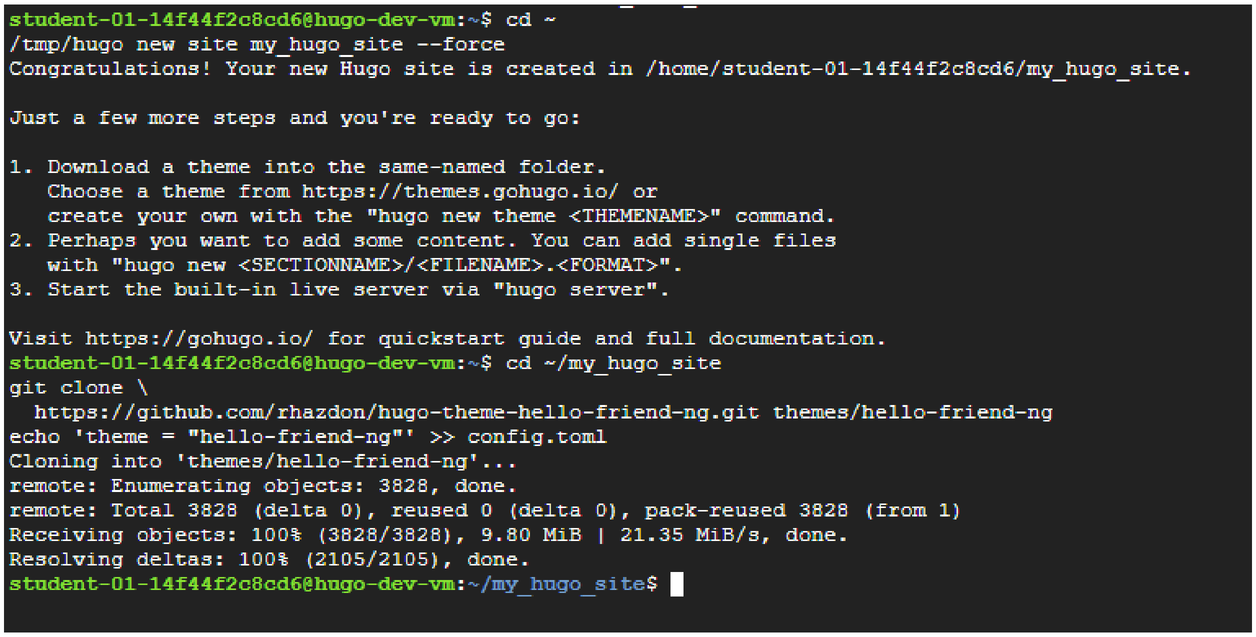
3. Enter the following commands to create and clone a code repository:



```
student-01-14f44f2c8cd6@hugo-dev-vm:~$ cd ~
gh repo create  my_hugo_site --private
gh repo clone  my_hugo_site
GraphQL: Name already exists on this account (createRepository)
fatal: destination path 'my_hugo_site' already exists and is not an empty directory.
failed to run git: exit status 128
student-01-14f44f2c8cd6@hugo-dev-vm:~$
```

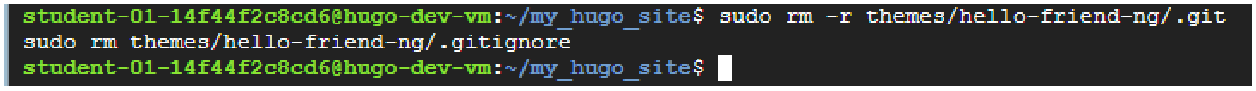
4. Enter the following commands below in the Linux shell:

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



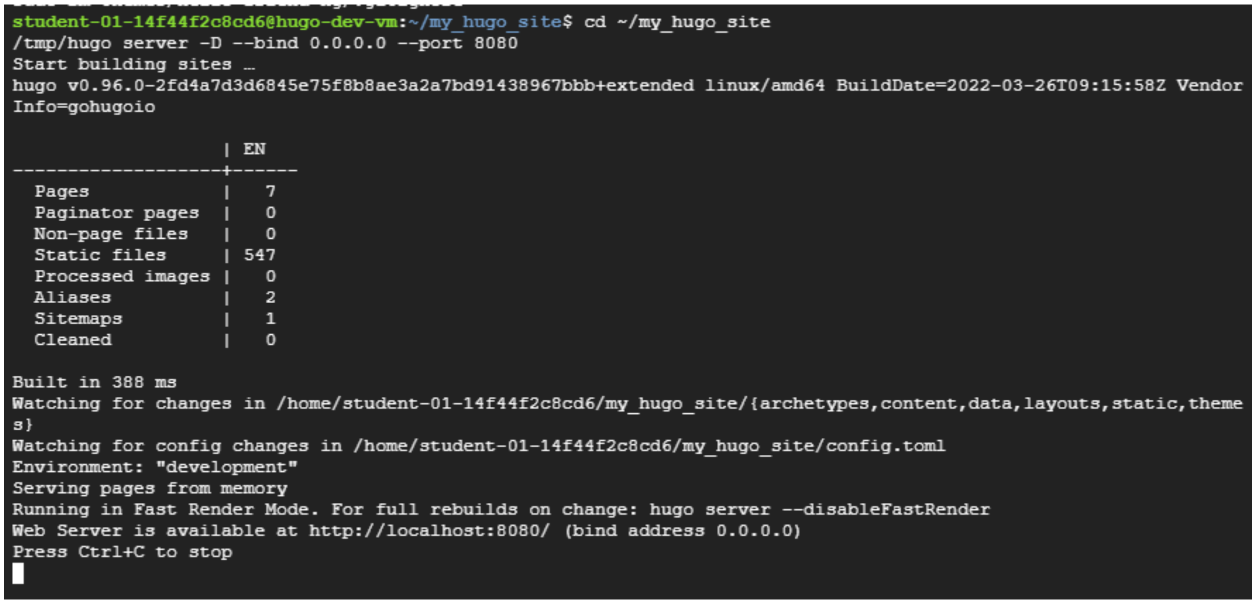
```
student-01-14f44f2c8cd6@hugo-dev-vm:~$ cd ~
/tmp/hugo new site my_hugo_site --force
Congratulations! Your new Hugo site is created in /home/student-01-14f44f2c8cd6/my_hugo_site.

Just a few more steps and you're ready to go:

1. Download a theme into the same-named folder.
   Choose a theme from https://themes.gohugo.io/ or
   create your own with the "hugo new theme <THEMENAME>" command.
2. Perhaps you want to add some content. You can add single files
   with "hugo new <SECTIONNAME>/<FILENAME>.<FORMAT>".
3. Start the built-in live server via "hugo server".

Visit https://gohugo.io/ for quickstart guide and full documentation.
student-01-14f44f2c8cd6@hugo-dev-vm:~$ 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
Cloning into 'themes/hello-friend-ng'...
remote: Enumerating objects: 3828, done.
remote: Total 3828 (delta 0), reused 0 (delta 0), pack-reused 3828 (from 1)
Receiving objects: 100% (3828/3828), 9.80 MiB | 21.35 MiB/s, done.
Resolving deltas: 100% (2105/2105), done.
student-01-14f44f2c8cd6@hugo-dev-vm:~/my_hugo_site$
```

5. Remove the git files from the themes directory:



```
student-01-14f44f2c8cd6@hugo-dev-vm:~/my_hugo_site$ sudo rm -r themes/hello-friend-ng/.git
sudo rm themes/hello-friend-ng/.gitignore
student-01-14f44f2c8cd6@hugo-dev-vm:~/my_hugo_site$
```

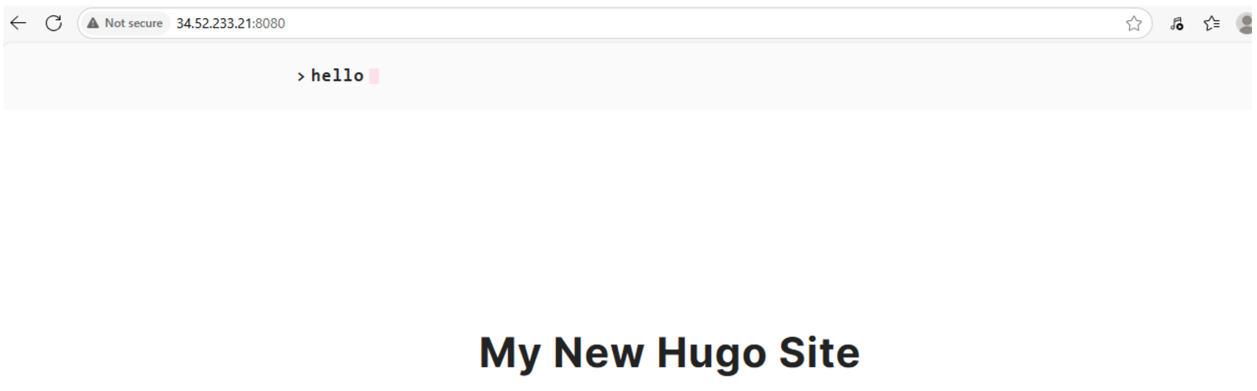
6. With the web site structure set up, you can now preview it. Enter the command below to launch the site at TCP port 8080:



```
student-01-14f44f2c8cd6@hugo-dev-vm:~/my_hugo_site$ cd ~/my_hugo_site
/tmp/hugo server -D --bind 0.0.0.0 --port 8080
Start building sites …
hugo v0.96.0-2fd4a7d3d6845e75f8b8ae3a2a7bd91438967bbb+extended linux/amd64 BuildDate=2022-03-26T09:15:58Z Vendor
Info=gohugoio

                   | EN
-------------------+------
  Pages            |   7
  Paginator pages  |   0
  Non-page files   |   0
  Static files     | 547
  Processed images |   0
  Aliases          |   2
  Sitemaps         |   1
  Cleaned          |   0

Built in 388 ms
Watching for changes in /home/student-01-14f44f2c8cd6/my_hugo_site/{archetypes,content,data,layouts,static,theme
s}
Watching for config changes in /home/student-01-14f44f2c8cd6/my_hugo_site/config.toml
Environment: "development"
Serving pages from memory
Running in Fast Render Mode. For full rebuilds on change: hugo server --disableFastRender
Web Server is available at http://localhost:8080/ (bind address 0.0.0.0)
Press Ctrl+C to stop
```

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


Not secure 34.52.233.21:8080

> hello

My New Hugo Site

Deploy the site to Firebase

1. Install Firebase CLI in the Linux instance shell:

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

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, Project ID.
- 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

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
   ```

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. Open the actionUri in a new browser tab.
3. 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 filled in at lab start.

5. Click **Save**.

6. Create a Cloud Build repository:

   ```
   gcloud builds repositories create "filled in at lab start" \
     --remote-uri="https://github.com/${GITHUB_USERNAME}/"filled in at lab start".git" \
     --connection="cloud-build-connection" --region=$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$'
   ```