publishing a website with GCP

I’ve always wanted to publish static sites to something other than Media Temple. Don’t get me wrong… I love MT, for the most part, but there are instances when their one-size-fits-all approach is too cumbersome for what I want to do and I would love to have a faster way to create sites and contents.

There are many ways to create static sites and to serve sites created with a framework in an(y number of languages… some of the ones that come to mind are Now, Github Pages and Heroku are the ones that come to mind the most.

But there are ways to upload your content to AWS and Google Cloud and publish it directly from there. I will cover the Google Cloud way to do it and will discuss some of the shortcomings and how to work around them, and if it’s worth working around them.

Configuration and CNAME record

Before we start working with Google Cloud we need to set up a Canonical Name (CNAME) Record to alias the bucket in Google Cloud to a subdomain of my host. In this case we’ll use `http://labs2.rivendellweb.net`.

According to Wikipedia:

> A Canonical Name record (abbreviated as CNAME record) is a type of resource record in the Domain Name System (DNS) used to specify that a domain name is an alias for another domain, which is the ‘canonical’ domain.
>
> … One can, for example, point ftp.example.com and www.example.com to the DNS entry for example.com, which in turn has an A record which points to the IP address. Then, if the IP address ever changes, one only has to record the change in one place within the network: in the DNS A record for example.com.
>
> CNAME records must always point to another domain

> name, never directly to an IP address.

So in our primary DNS host configuration we create the following record:

```
labs2.rivendellweb.net CNAME c.storage.googleapis.com.
```

This will point the URL `http://labs2.rivendellweb.net` to the host for our bucket.

We don’t have to host the DNS in Google any more than we have to host in Github to use Github Pages. I’ve kept my DNS in Media Temple and this record allows me to host the content in Google Cloud. In a later article we’ll discuss how to move a static site to Github Pages and how to use the CNAME record there.

With the CNAME record ready we can move to the actual mechanics of creating the bucket and setting it up.

Create the bucket

These steps assume that you've already created a Google account and have associated it with Google Cloud. If you haven't you need to do that first.

1. Select or create a Cloud Platform project.
2. Enable billing for your project.
3. Have a domain that you own or manage. If you don’t have an existing domain, there are many services through which can you can register a new domain, such as Google Domains.

   - This tutorial uses the domain `labs2.rivendellweb.net`

4. Verify that you own or manage the domain you will be using. I’ve completed this step when working on setting up analytics for a blog in the domain.

   - If you own the domain you are associating to a bucket, you might have already performed this step in the past
   - If you purchased your domain through Google Domains, verification is automatic.

Figure 1: Settings for a bucket that will hold the site's content

Upload content to the bucket

- If you haven’t already select your project in the Google Cloud Console
- Open the Cloud Storage browser in the Google Cloud Platform Console.
- In the list of buckets, click on the bucket you created.
- Click Upload files.
- In the file dialog, browse to the desired file and select it.
- After the upload completes, you should see the file name, size, type, and last modified date in the bucket.

The result would look something like this:

Figure 2: What the result of uploading files looks like

Alternative methods if you’re uploading large numbers of files

Rather than use the Google Cloud Console we can use command line tools and additional steps to build processes. For this example I’ll illustrate using Google Cloud’s `rsync` command and a Gulp task built around the gulp-gcloud-publish plugin.

gsutil rsync uses the Rsync utility to copy data from the source to the target by adding new files, updating files with updated content and, if so configured, deleting files that are no longer present in the source directory. At its most basic it works by removing files that are no longer needed (with the `-d` flag) and travels through children directories. The command looks like this:

```
gsutil rsync -d -r src_url gs://my_bucket/data
```

To copy only new/changed files without deleting extra files from gs://mybucket/data leave off the -d option:

```
gsutil rsync -r data gs://mybucket/data
```

If you have a large number of objects to copy you might want to use the gsutil -m option, to perform parallel (multi-threaded/multi-processing) synchronization:

```
gsutil -m rsync -d -r data gs://mybucket/data
```

The -m option typically will provide a large performance boost if either the source or destination (or both) is a cloud URL. If both source and destination are file URLs the -m option will typically thrash the disk and slow synchronization down.

For more details about the utility check the gsutil rsync docs.

gulp-gcloud-publish assumes you're using Gulp to process and package assets already. If not you'll have to either run rsync (discussed above) with a task to run shell commands.

If you're already using Gulp, you can install the plugins like normal. I'm adding them as dev dependencies.

```
npm i -D gulp-gcloud-publish gulp-gzip
```

Then we can use the task below to push our content to the bucket we've configured before. The following informaton needs to be modified:

- `bucket-name` need to be changed to the name of the bucket you created
- `path/to/keyFile.json` needs to point to the key file you generated for your site
- `projectId` must be equal to the name of the project you created
- base is the root of the site you want to publish

```
var gulp = require('gulp');
var gcPub = require('gulp-gcloud-publish');
v'gulp-gcloud-publish'lp-gzip'); // optional

gulp.task('publish', function() {
    return gulp.src('public/css/example.css')
      .pipe(gzip()) 'publish'// optionalipe(gcPub({
      bucket: 'bucket-name',
      keyFilename: 'path/to/keyFile.json',
      projectId: 'my-project-id',
      base: '/css',
      public: true,
      transformDestination: function(path) {
        return path.toLowerCase();
      },
      metadata: {'bucket-name'cheControl: 'max-age=315360000, no-transfor
      }
  })); // => Fil'max-age=315360000, no-transform, public'// => File will
});
```

This method will require the generation of a service account key. Instructions for generatingg your key can be found in the documentation about [creating and managing account keys]

These two ways give you the flexibility you need to publish your static content. We're not covering GCP functions and other technologies available to create static and serverless sites and apps.

Make the files publically visible

If everything has worked up to this point we now have a bucket ready to point to our subdomain. Now all we need to do is make our files publically accessible.

1. Open the Cloud Storage browser in the Google Cloud Platform Console.
2. In the list of buckets, click on the bucket that contains the objects you uploaded.
3. For each object you want to share publicly, click the checkbox in the Share publicly column.

Note that you must do this for all files individually. Cloud Storage will not let you select a folder and automatically make the files inside public.

SSL Certificate and HTTPS hosting

The one thing that Google Gloud doesn't do, out of the box, is provide SSL support for secure hosting. If you're working with PWAs this is a show stopper, right there. Service Workers will not work without SSL/HTTPS and the default setup for static hosting doesn't provide it, at least not easily.

1. Verify that you are the owner of your domain through the Webmaster Central page:

- In the Google Cloud Platform Console, go to App Engine > Settings > Custom Domains
- Click Add a custom domain to display the Add a new custom domain form:
- In the Select the domain you want to use section, enter the name of the domain that you want to use, for example example.com, and then click Verify to open a new tab to the Webmaster Central page
    - Use Webmaster Central to verify ownership of your domain.
    - **Important**: Verifying domain ownership by using a CNAME record is the preferred option for App Engine. If you choose to use a TXT record, you must avoid configuring your domain's DNS with a CNAME record because the CNAME record overrides the TXT record and causes your domain to appear unverified
- If the verification methods for your domain do not offer the CNAME record option, you can select Other as your domain provider and then choose Add a CNAME record:
    - Click Alternate methods and then Domain name provider.
    - In the menu, select Other
    - In the Having trouble section, click Add a CNAME record and then following the instructions to verify ownership of your domain
    - **Remember: It might take a minute before your CNAME is set at your domain registrar** Return to the Add new custom domain form in the Cloud Platform Console. Ensure that your domain has been verified:

CONSOLEGCLOUDAPI If your domain is not already listed, click Refresh domains.

Important: The domain verification is automatically re-confirmed about every 30 days. So if you remove the verification string from your DNS settings, you will lose the ability to change the configuration within the Cloud Platform Console. However, if this happens, the serving setup for the domain does not change and the app continues to serve over the custom domain. If you need to delegate the ownership of your domain to other users or service accounts, you can add permission through the Webmaster Central page:

Opening the following address in your web browser:

https://www.google.com/webmasters/verification/home Under Properties, click the domain for which you want to add a user or service account. Scroll down to the Verified owners list, click Add an owner, and then enter a Google Account email address or service account ID.

To view a list of your service accounts, open the Service Accounts page in the Cloud Platform Console:

GO TO SERVICE ACCOUNTS PAGE After you verify ownership of your domain, you can map that domain to your App Engine app:

CONSOLEGCLOUDAPI Continue to the next step of the Add new custom domain form to select the domain that you want to map to your App Engine app:

Specify the domain and subdomains that you want to map. The naked domain and www subdomain are pre-populated in the form. A naked domain, such as example.com, maps to http://example.com. A subdomain, such as www, maps to http://www.example.com. Click Submit mappings to create the desired mapping. In the final step of the Add new custom domain form, note the resource records that are listed, including their type and canonical name (CNAME), because you need to add these details to the DNS configuration of your domain. In the example below, CNAME is one of the types listed, and ghs.googlehosted.com is its canonical name.

Add a custom domain Add the resource records that you receive to the DNS configuration of your domain registrar:

Log in to your account at your domain registrar and then open the DNS configuration page. Locate the host records section of your domain's configuration page and then add each of the resource records that you received when you mapped your domain to your App Engine app.

Typically, you list the host name along with the canonical name as the address. For example, if you registered a Google Domain, then one of the records that you add to your DNS configuration is the www host name along with the ghs.googlehosted.com address. Alternatively, to specify a naked domain, you would instead use @ with the ghs.googlehosted.com address.

For more information about mapping your domain, see the following Using subdomains and Wildcard mappings sections. Save your changes in the DNS configuration page of your domain's account. It can take a while for these changes to take effect. Test for success by browsing to your app via its new domain URL, for example www.example.com.

Using subdomains

If you set up a wildcard subdomain mapping for your custom domain, then your application serves requests for any subdomain that matches.

If the user browses a domain that matches an application version name or service name, the application serves that version. If the user browses a domain that matches a service name, the application serves that service. Wildcard mappings

Note that wildcard mappings will work with your services in App Engine.

You can use wildcards to map subdomains at any level, starting at third-level subdomains. For example, if your domain is example.com and you enter text in the web address field:

Entering * maps all subdomains of example.com to your app. Entering *.private maps all subdomains of private.example.com to your app. Entering *.nichol.sharks.nhl maps all subdomains of nichol.sharks.nhl.example.com to your app. Entering *.excogitate.system maps all subdomains of excogitate.system.example.com to your app. If you use G Suite with other subdomains on your domain, such as sites and mail, those mappings have higher priority and are matched first, before any wildcard mapping takes place. In addition, if you have other App Engine apps mapped to other subdomains, those mappings also have higher priority than any wildcard mapping.

Note that some DNS providers might not work with wildcard subdomain mapping. In particular, a DNS provider must permit wildcards in CNAME host entries.

The above wildcard routing rules also apply to URLs that contain components for services, versions, and instances, following the service routing rules for the App Engine.

Other alternatives inside the Google family

Firebase hosting provides a similar service to that of Google Cloud