Standing Up a GCP Instance with Genesis

#work #guide #genesis #gcp

Get you a GCP!

Initial Setup

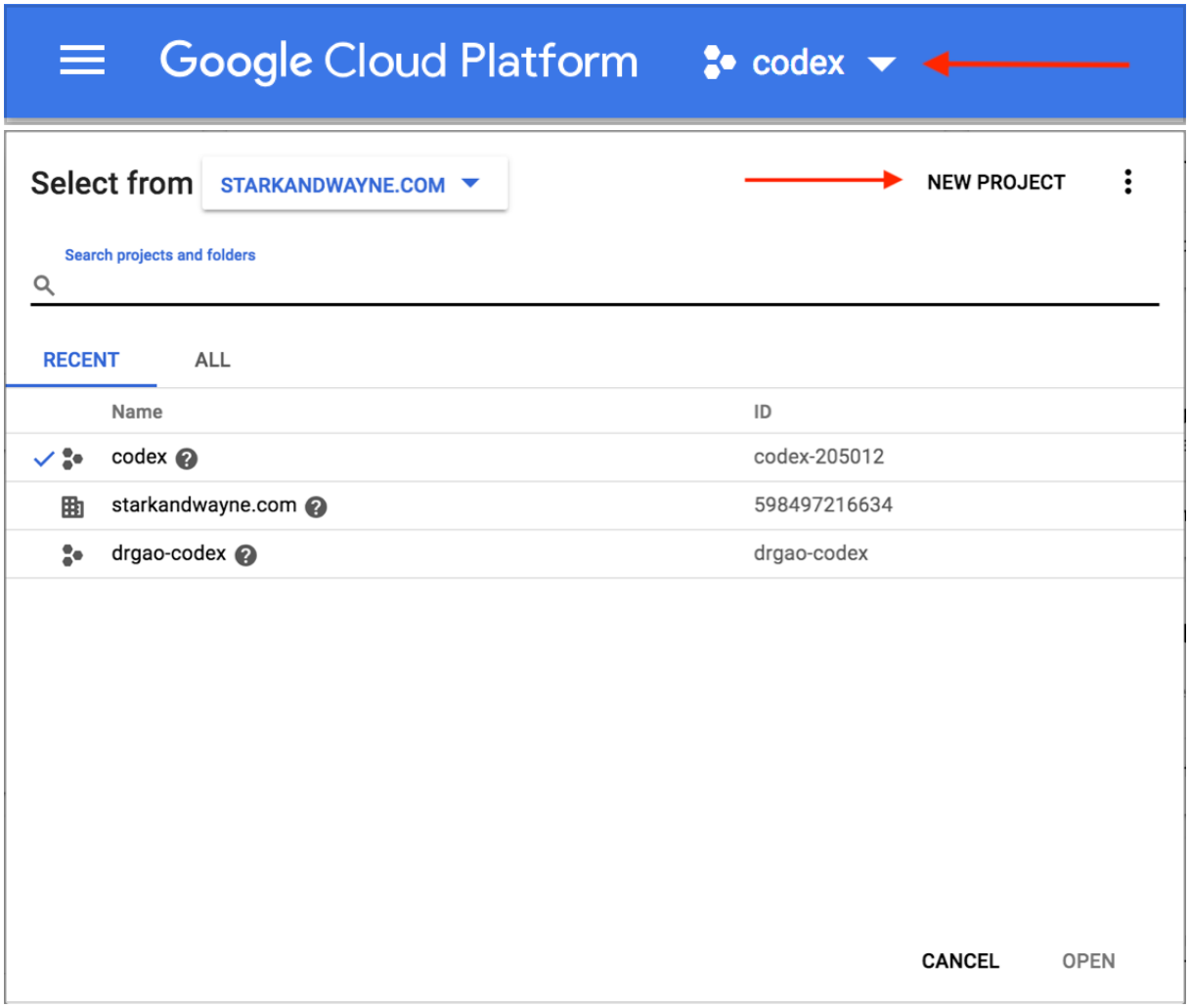
First, we'll create a GCP project and clone some necessary utilities to get started. You'll want a credit card handy, as GCP isn't free. You also won't be able to use a trial GCP project for this, as we require more than one IP in use (among other things)

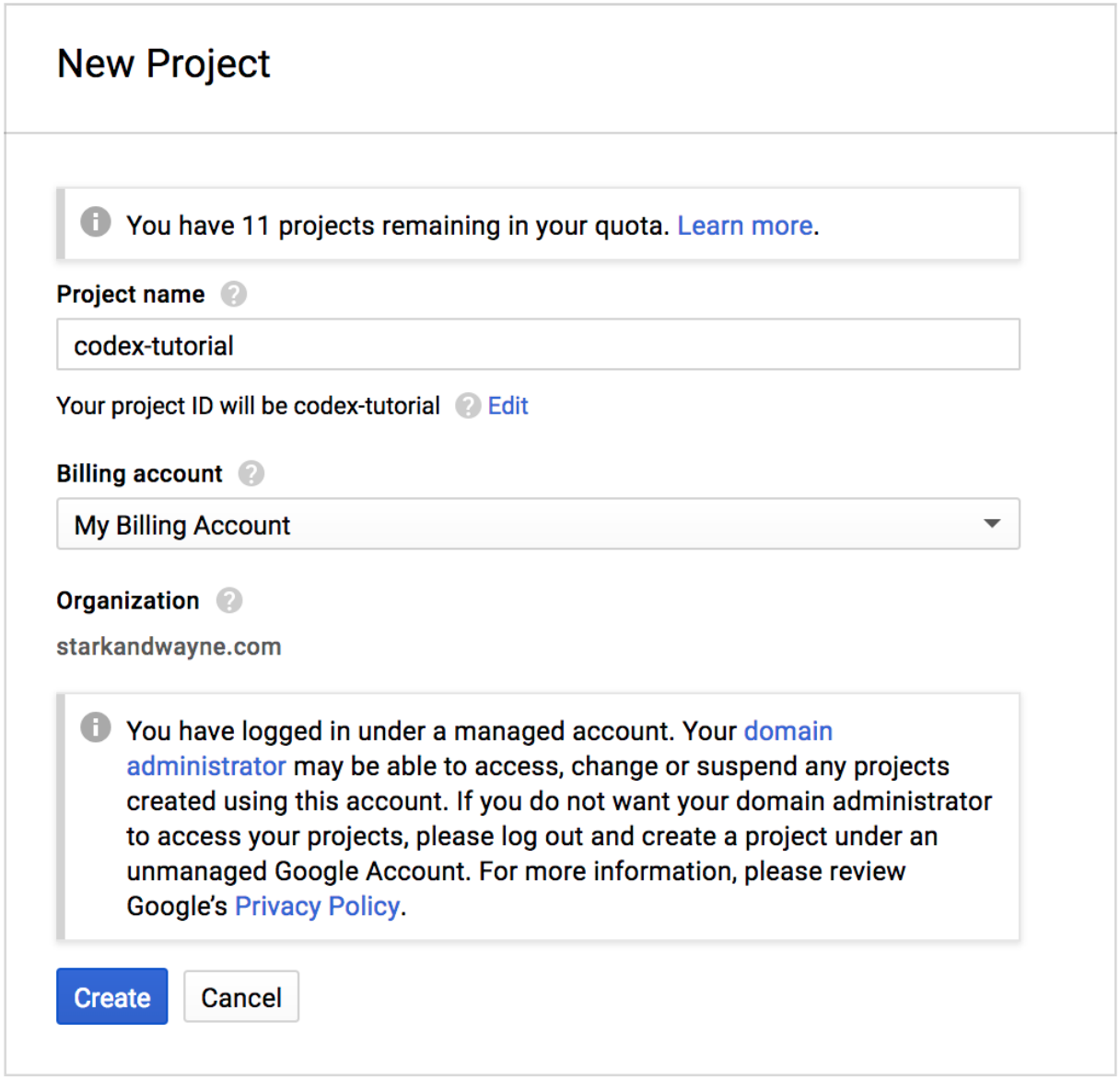
Some things to decide before getting started:

- What region are we going to use to deploy GCP? If you're unsure or just want to muck around with GCP, use `us-east1`
    - East coast, beast coast

Creating GCP Project

Visit Google Cloud Console and create a new project:



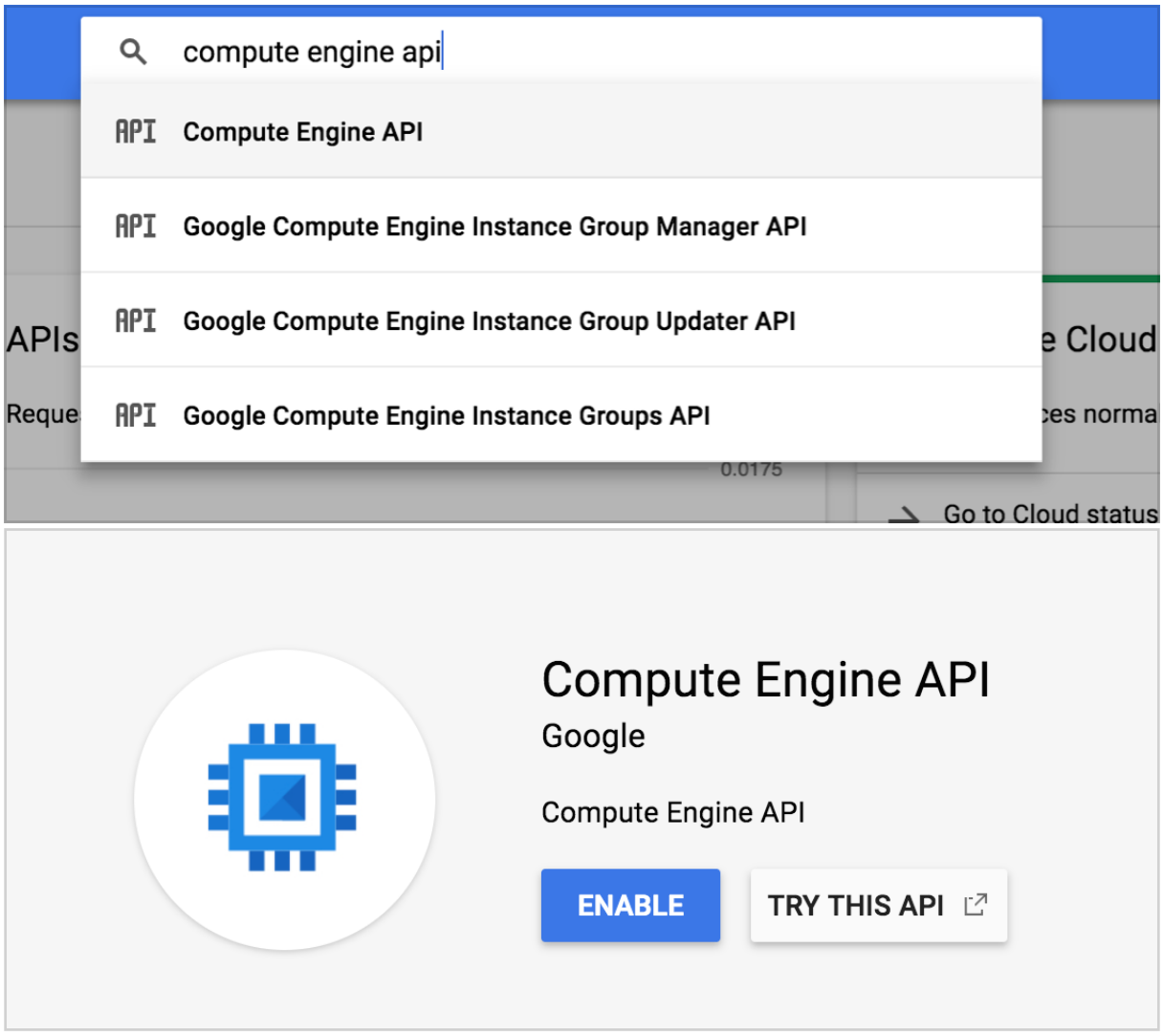


If you don't have a billing account setup, you'll be asked to input your information.

Enabling GCP APIs

A few APIs are necessary to get started. Using the search feature in the blue navigation bar, search for (and enable) the following APIs:

- Cloud Resource Manager API
- Identity and Access Management API
- Compute Engine API
- CloudSQL API

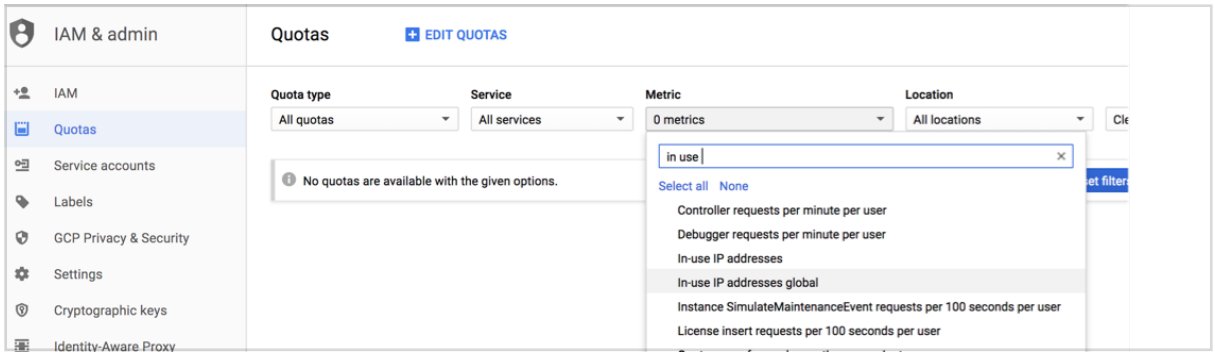


Raising GCP Quotas

It's also necessary to raise some too-low quota values that Google Cloud declares for new projects. Go to the menu (☰), and navigate to "IAM & Admin" ➔ "Quotas". You'll need to increase the following:

- Global In-use IP addresses
  - Set to 50
- Global CPUs
  - Set to 100
- Regional CPUs (the region you're deploying in)
  - Set to 100

Since there's a lot of quotas to scroll through, your best bet is to *select none* under the Metric dropdown, and search for the quotas you need and select them.



Raising quotas requires Google approval, so it may take some time to the new quotas to be set. You can continue with the rest of the setup while you wait, but you can't Terraform GCP (and anything after that) until they're set. *It can take up to 48 hours, but it's typically 5-40 minutes*

Clone Codex Repository

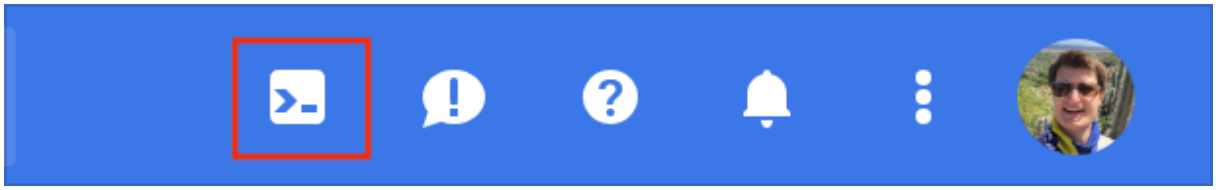
Codex is a repository we (Stark & Wayne) maintain as a knowledge DB. It contains scripts & information that helps get various projects setup (including this one). So, clone it in your typical work directory:

```
git clone https://github.com/starkandwayne/codex
```

For the remainder of this tutorial, we'll be working in `codex/terraform/google`

Grabbing Credentials From GCP

Now that your project is setup, you need to grab some keys and information necessary for Terraform to work. Open the Google Cloud Console (the >_ icon from the upper right hand navbar) and run the following Bash commands in the Cloud Console:



```
export project_id=$(gcloud config get-value project)
export region=us-east1
export zone=us-east1-d
export service_account_email=terraform@${project_id}.iam.gserviceaccount.com
```

```
gcloud config set compute/zone ${zone}
gcloud config set compute/region ${region}

gcloud iam service-accounts create terraform --display-name terraform
gcloud iam service-accounts keys create ~/terraform.key.json \
  --iam-account ${service_account_email}

gcloud projects add-iam-policy-binding ${project_id} \
  --member serviceAccount:${service_account_email} \
  --role roles/owner
```

You've now created a user account for Terraform to use, and granted it owner status. You'll need to download the `terraform.key.json` file that was just generated, which you can do by clicking on the Console dropdown (⋮), selecting "Download file", and typing in `terraform.key.json`. Place that file in your Codex repository, as `terraform/google/keys/iam.json`.

⚠️ Keep iam.json secure! It has ownership-level access to your GCP project, which grants permission to do anything (including total deletion & access to billing information) to your project ⚠️

Setting Up Terraform

Install Terraform on your system. If you're on Mac and use Homebrew, it's `brew install terraform`, otherwise you'll need to visit Terraform Downloads and download the binary for your system.

Once you have Terraform, `cd` into the Codex repository, and into `terraform/google` and run `terraform init`. This will download the necessary plugins to use the GCP API.

Some of the Terraform files require an external library called cc-me, created by James. You'll need to install it:

```
wget https://raw.githubusercontent.com/jhunt/cc-me/master/cc-me
chmod +x cc-me
sudo mv cc-me /usr/bin/local/cc-me
```

A file containing per-project variables, named `google.tfvars` needs to be created within `terraform/google` and populated with:

```
google_project      = "<< project id >>"
google_region       = "<< gcp zone >> "
google_az1          = "b"
google_az2          = "c"
google_az3          = "d"
google_network_name = "codex"
google_credentials  = "keys/iam.json"
google_pubkey_file  = "keys/gce.pub"
bucket_prefix = " << random all lower-case string >> "
db_prefix = " << random all lower-case string >> "
```

Here's an example `tfvars`:

```
google_project      = "codex-tutorial"
google_region       = "us-east1"
google_az1          = "b"
google_az2          = "c"
google_az3          = "d"
google_network_name = "codex"
google_credentials  = "keys/iam.json"
google_pubkey_file  = "keys/gce.pub"
bucket_prefix = "2007178a2c3148f"
db_prefix = "8b21d03cb5c74ca8b"
```

Now, you'll need SSH keys to get into the bastion host and the NATs. You can generate a key with these commands:

```
mkdir keys
ssh-keygen -f keys/gce </dev/null
chmod 0400 keys/*
echo "/keys" >> .gitignore
```

⚠️ **Keep the generated SSH key secure. This key has total access to your bastion host, and by proxy, all of your GCP infrastructure. Ensure it's not committed to git, as you're currently working inside a git repository.** ⚠️

- download iam.json
- run mkdir keys
- terraform init
- uuidgen (make sure it's been lower-cased)
- update tfvars with uuid
- .to-the-basion,
- setup the git
- proto-bosh 10.4.1.3
- '

from

Definitions & "What...?"

GCP Google Cloud Platform. It's Google's cloud services offering, a la Amazon's AWS.

Terraform Tool used to setup the infrastructure surrounding all your cloud VMs. You *could* spend all week working with Google Console web UI to setup the network tiles & databases, or you could write a Terraform ruleset to do it for you in a consistent, predictable and automated way. Plus you can use it to generate cloud-configs easily.