* **Terraform:** a tool used to turn infrastructure development into code.
* **Google Cloud SDK:**command line utility for managing Google Cloud Platform resources.
* **Google Cloud Platform:** cloud-based infrastructure environment.
* **Google Compute Engine:**resource that provides virtual systems to Google Cloud Platform customers.

**Downloading, installing and configuring Terraform**

1.SSH into your cloud server

2.wget https://releases.hashicorp.com/terraform/0.11.3/terraform\_0.11.3\_linux\_amd64.zip

3.unzip terraform\_0.9.8\_linux\_amd64.zip

4.sudo mv terraform /usr/local/bin/

5. Set the PATH variable for Terraform:

export PATH="/usr/local/terraform:$PATH"

vim /etc/bashrc

source /etc/bashrc

6.Confirm terraform binary is accessible: terraform --version

**Downloading and configuring Google Cloud SDK**

Now that we have Terraform installed, we need to set up the command line utility to interact with our services on Google Cloud Platform. This will allow us to authenticate to our account on Google Cloud Platform, and subsequently use Terraform to manage our infrastructure.

1. Download and install Google Cloud SDK:

**$ curl https://sdk.cloud.google.com | bash**

. . .

2. Initialize the gcloud environment:

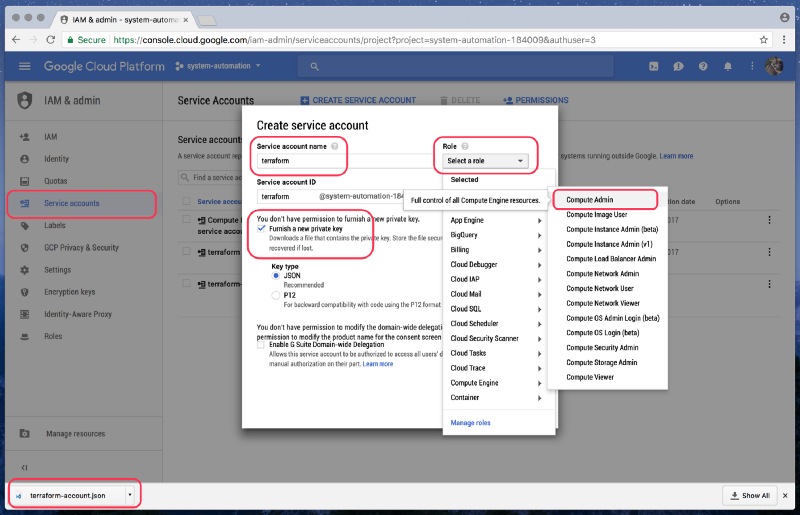
**$ gcloud init**

You’ll be able to connect your Google account with the gcloud environment.

**Configuring our Service Account on GCP**

In the following few paragraphs I’ll explain how to create a project, set up a service account and set the correct permissions to manage our project’s resources.

1. Create a project and name it whatever you’d like.
2. Create a service account and specify the compute admin role.
3. Download the generated JSON file and save it to your project’s directory.



Be warned, the JSON file you just downloaded should be protected from non-authorized users. Think of this as a private key or password to manage your infrastructure’s resources. For development purposes we can add a **.gitignore**file to our project, adding **terraform-account.json**so that it’s not committed to our repository.

**Creating a Compute Engine Instance with Terraform**

You’ll first need to create a few files to work with. The most important thing is that each file ends in **file.tf.** This allows Terraform to know what files to work with when initializing, planning, applying, and destroying.

Here are some examples:

**$ vim provider.tf**

# Specify the provider (GCP, AWS, Azure)  
provider “google” {  
credentials = “${file(“terraform-account.json”)}”  
project = “system-automation-184009”  
region = “us-central1”  
}

**$ vim compute.tf**

*# Create a new instance*resource "google\_compute\_instance" "ubuntu-xenial" {  
 name = "ubuntu-xenial"  
 machine\_type = "f1-micro"  
 zone = "us-west1-a"  
 boot\_disk {  
 initialize\_params {  
 image = "ubuntu-1604-lts"  
 }  
}

network\_interface {  
 network = "default"  
 access\_config {}  
}

service\_account {  
 scopes = ["userinfo-email", "compute-ro", "storage-ro"]  
 }  
}

We’ll also need to follow the proceeding steps to make Terraform aware of the files we just created. Your screen should be similar to what’s presented below.

**$ terraform init**

Initializing provider plugins…  
- Checking for available provider plugins on [https://releases.hashicorp.com](https://releases.hashicorp.com/)...  
- Downloading plugin for provider “google” (1.1.1)…

The following providers do not have any version constraints in configuration,  
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking  
changes, it is recommended to add version = “…” constraints to the  
corresponding provider blocks in configuration, with the constraint strings  
suggested below.

\* provider.google: version = “~> 1.1”

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running “terraform plan” to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.

If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.

You’ll now want to run the following command to see what we’re about to create and configure. Once again, your output should be similar.

**$ terraform plan**

Refreshing Terraform state in-memory prior to plan…  
The refreshed state will be used to calculate this plan, but will not be  
persisted to local or remote state storage.

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

An execution plan has been generated and is shown below.  
Resource actions are indicated with the following symbols:  
 + create

Terraform will perform the following actions:

google\_compute\_instance.ubuntu-xenial  
 id: <computed>  
 boot\_disk.#: “1”  
 boot\_disk.0.auto\_delete: “true”  
 boot\_disk.0.device\_name: <computed>

. . .

Plan: 1 to add, 0 to change, 0 to destroy.

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

Note: You didn’t specify an “-out” parameter to save this plan, so Terraform  
can’t guarantee that exactly these actions will be performed if  
“terraform apply” is subsequently run.

Now that everything looks good, and is staged, we can apply those settings and create our VM instance.

**$ terraform apply**

google\_compute\_instance.ubuntu-xenial: Creating…  
 boot\_disk.#: “” => “1”  
 boot\_disk.0.auto\_delete: “” => “true”  
 boot\_disk.0.device\_name: “” => “<computed>”  
   
. . .

Error applying plan:

1 error(s) occurred:

\* google\_compute\_instance.ubuntu-xenial: 1 error(s) occurred:

\* google\_compute\_instance.ubuntu-xenial: The user does not have access to service account '[298693621029-compute@developer.gserviceaccount.com](mailto:298693621029-compute@developer.gserviceaccount.com)'. User: '[terraform@system-automation-184009.iam.gserviceaccount.com](mailto:terraform@system-automation-184009.iam.gserviceaccount.com)'. Ask a project owner to grant you the iam.serviceAccountActor role on the service account

Terraform does not automatically rollback in the face of errors.  
Instead, your Terraform state file has been partially updated with  
any resources that successfully completed. Please address the error  
above and apply again to incrementally change your infrastructure.

Let’s focus on the following error output:

\* google\_compute\_instance.ubuntu-xenial: The user does not have access to service account '[298693621029-compute@developer.gserviceaccount.com](mailto:298693621029-compute@developer.gserviceaccount.com)'. User: '[terraform@system-automation-184009.iam.gserviceaccount.com](mailto:terraform@system-automation-184009.iam.gserviceaccount.com)'. **Ask a project owner to grant you the iam.serviceAccountActor role on the service account**

This is basically saying that we need to grant our terraform@system-automation-184009.iam.gserviceaccount.com service account the same permissions as 298693621029compute@developer.gserviceaccount.com.

You should see a similar output as shown below:

**$ gcloud projects get-iam-policy system-automation-184009**

bindings:  
- members:  
 — serviceAccount:[terraform@system-automation-184009.iam.gserviceaccount.com](mailto:terraform@system-automation-184009.iam.gserviceaccount.com)  
 role: roles/compute.admin

. . .

- members:  
 — serviceAccount:[298693621029-compute@developer.gserviceaccount.com](mailto:298693621029-compute@developer.gserviceaccount.com)  
 role: roles/editor

. . .

version: 1

Let’s make some changes and see what happens:

**$ gcloud projects add-iam-policy-binding system-automation-184009 \  
 --member serviceAccount:**[**terraform@system-automation-184009.iam.gserviceaccount.com**](mailto:terraform@system-automation-184009.iam.gserviceaccount.com) **-- role roles/editor**

bindings:  
- members:  
 — serviceAccount:[terraform@system-automation-184009.iam.gserviceaccount.com](mailto:terraform@system-automation-184009.iam.gserviceaccount.com)  
 role: roles/compute.admin

. . .

- members:  
 — serviceAccount:[298693621029-compute@developer.gserviceaccount.com](mailto:298693621029-compute@developer.gserviceaccount.com)  
 role: roles/editor

. . .

— serviceAccount:[terraform@system-automation-184009.iam.gserviceaccount.com](mailto:terraform@system-automation-184009.iam.gserviceaccount.com)  
 role: roles/editor

. . .

version: 1

Everything should be good now that we’ve [made those modifications](https://cloud.google.com/iam/docs/granting-roles-to-service-accounts). Let’s get back to setting up our VM instance:

**$ terraform apply**

google\_compute\_instance.ubuntu-xenial: Creating…  
 boot\_disk.#: “” => “1”  
 boot\_disk.0.auto\_delete: “” => “true”

. . .

google\_compute\_instance.ubuntu-xenial: Still creating... (10s elapsed)  
google\_compute\_instance.ubuntu-xenial: Creation complete after 17s (ID: ubuntu-xenial)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

**$ gcloud compute instances list**

NAME ZONE MACHINE\_TYPE PREEMPTIBLE INTERNAL\_IP EXTERNAL\_IP STATUS  
ubuntu-1604-server us-central1-c g1-small 10.128.0.2 23.255.255.255 RUNNING

it works! Use below command to destroy it.

**$ terraform destroy**

google\_compute\_instance.ubuntu-xenial: Refreshing state… (ID: ubuntu-xenial)

An execution plan has been generated and is shown below.  
Resource actions are indicated with the following symbols:  
 — destroy

Terraform will perform the following actions:

- google\_compute\_instance.ubuntu-xenial

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy?  
 Terraform will destroy all your managed infrastructure, as shown above.  
 There is no undo. Only ‘yes’ will be accepted to confirm.

Enter a value: yes

google\_compute\_instance.ubuntu-xenial: Destroying… (ID: ubuntu-xenial)  
google\_compute\_instance.ubuntu-xenial: Still destroying… (ID: ubuntu-xenial, 10s elapsed)  
google\_compute\_instance.ubuntu-xenial: Still destroying… (ID: ubuntu-xenial, 20s elapsed)  
google\_compute\_instance.ubuntu-xenial: Still destroying… (ID: ubuntu-xenial, 30s elapsed)  
google\_compute\_instance.ubuntu-xenial: Destruction complete after 38s

Destroy complete! Resources: 1 destroyed.