1. **Create a Cloud SQL instance, database, user (and password)**

Here are the links to the Terraform examples for

* [creating a Cloud SQL instance](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/sql_database_instance),
* [creating a database](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/sql_database) and
* [creating user and password](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/sql_user)

**Note:** As pointed out in [creating a Cloud SQL instance](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/sql_database_instance): ” For private IP instance setup, note that the *google\_sql\_database\_instance* does not actually interpolate values from *google\_service\_networking\_connection*. You must explicitly add a *depends\_on* reference …” And to create the *google\_service\_networking\_connection*, we need to:

* enable the Service Networking API which can be done in the console or via gcloud cli or terraform code
* grant *roles/servicenetworking.networksAdmin* role to the service account used by Terraform code. Here is an example:

gcloud projects add-iam-policy-binding gft-bike-insurance \

--member="serviceAccount:shared-components-installation@gft-bike-insurance.iam.gserviceaccount.com" \

--role="roles/servicenetworking.networksAdmin"

1. **Create an image from the existing VM**

Graphical user interface, text

Description automatically generated

use this machine image to create a new VM using terraform following the [gcloud command line example](https://cloud.google.com/compute/docs/machine-images/create-instance-from-machine-image#create-instance-from-different-project) or the [Terraform code example](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/compute_instance_from_machine_image)

Note:

1. There are some prerequisites as listed in the link above:

* *compute.instances.create* on the project
* *compute.machineImages.readOnly* on the machine image you want to use (see the gcloud clis in the [gcloud command line example](https://cloud.google.com/compute/docs/machine-images/create-instance-from-machine-image#create-instance-from-different-project) to see how to grant access to the machine images that is stored in a different project to a service account)
* Any additional permissions needed to set attributes on the instance, for example, the setTags permission

1. The boot disk size is fixed with the original image, so for instance, if the original VM has a boot size of 10GB, we can only create a VM with size of 10GB
2. some requirements for VM:

* If the cloudSQL only has private IP, the VM should be in the same network as the cloudSQL instance which store the backend database for Mlflow
* And the cloudSQL should be PostgreSQL
* We should grant the access to all google API services to VM

Once the VM is created, if we cannot ssh into the VM, and it shows the error to connect to port 22, we should add firewall rules in VPC network as shown in screenshot to allow ssh access



1. **Once the VM is created, enter into VM, change the configuration file of the systemd unit service in directory /usr/lib/systemd/system/**

* cloudsqlproxy.service
* gcs-airflow.service
* mlflow-webserver.service

1. **after modifying all these three \*.service files, run following commands to reload the service files and start the service.**

sudo systemctl daemon-reload

sudo systemctl restart gcs-airflow

sudo systemctl restart cloudsqlproxy

sudo systemctl restart mlflow-webserver

1. **check the status of the service units, run the following command:**

sudo systemctl status gcs-airflow

sudo systemctl status cloudsqlproxy

sudo systemctl status mlflow-webserver

sudo systemctl status airflow-scheduler

sudo systemctl status airflow-webserver

1. **and modify the /etc/airflow file.**

Change the project name to the current one in the file.