**1. Setting Up GCP and Terraform**

Before you can start provisioning resources with Terraform, you need to ensure that you have a working GCP account and the Terraform CLI installed.

**Prerequisites:**

* **GCP Account**: You need a Google Cloud Platform account. If you don’t have one, sign up at <https://cloud.google.com/>.
* **Google Cloud SDK**: It is helpful to have the Google Cloud SDK installed on your local machine for managing GCP resources. Download it from here.
* **Terraform**: Install Terraform on your local machine. You can download it from the Terraform website.

**Steps:**

1. **Install Terraform**:
   * Download the appropriate Terraform binary for your system from the official site.
   * After downloading, unzip it and place it in a directory that is included in your system's PATH.

To check if Terraform is installed correctly, run the following command:

bash

Copy

terraform --version

1. **Install Google Cloud SDK**:
   * If you don’t already have it installed, follow the installation steps here: Install the Google Cloud SDK.
   * Once installed, authenticate with your Google account by running:

bash

Copy

gcloud auth login

**2. Setting Up the GCP Service Account**

Terraform interacts with GCP using a **service account**, which needs appropriate permissions. Here’s how to set it up:

**Steps to Create a Service Account:**

1. **Go to the GCP Console**:
   * Visit the Google Cloud Console.
2. **Create a New Service Account**:
   * Navigate to IAM & Admin > Service Accounts.
   * Click on + CREATE SERVICE ACCOUNT.
   * Provide a name, e.g., terraform-service-account, and description.
   * Click **Create**.
3. **Assign Roles to the Service Account**:
   * In the "Grant this service account access to project" section, select the following roles:
     + Owner (for full access to GCP resources, adjust based on the principle of least privilege).
     + Alternatively, you can assign more granular roles, such as Editor, Compute Admin, Storage Admin, etc., depending on your use case.
   * Click **Continue**.
4. **Create and Download the Service Account Key**:
   * After creating the service account, click on it to open the details.
   * Go to the Keys tab.
   * Click on Add Key > Create new key and select the **JSON** key type.
   * Download the key file to your local machine. This will be used in Terraform for authentication.
5. **Set Environment Variable for Authentication**:
   * Set the GOOGLE\_APPLICATION\_CREDENTIALS environment variable to the path of the downloaded service account key file.

bash

Copy

export GOOGLE\_APPLICATION\_CREDENTIALS="/path/to/your/service-account-file.json"

* + This environment variable allows Terraform to authenticate with GCP using the service account.

**3. Preparing Your Terraform Environment**

Now that you have Terraform and GCP set up, it’s time to create your Terraform configuration files.

**Steps:**

1. **Create a New Directory for Terraform Configuration**:
   * Create a directory on your local machine to store your Terraform configuration files.

bash

Copy

mkdir terraform-gcp

cd terraform-gcp

1. **Create the provider.tf File**:
   * This file will define the provider (GCP in our case) and authenticate Terraform to use your GCP account.
   * Example content for provider.tf:

hcl

Copy

provider "google" {

credentials = file("path/to/your/service-account-file.json")

project = "your-gcp-project-id"

region = "us-central1"

}

1. **Create Other Terraform Configuration Files**:
   * Create other files like sql-instance.tf, vm-instance.tf, firewall.tf, etc., depending on the resources you want to provision (e.g., Cloud SQL, VM, Vault).

Example:

hcl

Copy

resource "google\_compute\_instance" "vault\_instance" {

name = "vault-instance"

machine\_type = "e2-medium"

zone = "us-central1-a"

network\_interface {

network = "default"

access\_config {

// External IP access

}

}

boot\_disk {

initialize\_params {

image = "debian-cloud/debian-10-buster-v20210129"

}

}

}

1. **Initialize Terraform**:
   * Run the following command in the terminal inside your Terraform configuration directory to initialize Terraform.

bash

Copy

terraform init

This command downloads the necessary provider plugins and sets up your working directory for use.

**4. Running Terraform Scripts on GCP**

Now that your configuration is set up, you are ready to run Terraform and provision resources on GCP.

**Steps:**

1. **Validate the Configuration**:
   * Run the terraform plan command to validate your configuration. This will show you what changes Terraform plans to make to your GCP infrastructure.

bash

Copy

terraform plan

* + Terraform will analyze the resources defined in your configuration files and show you a preview of what will be created, modified, or deleted.

1. **Apply the Configuration**:
   * To apply the changes and create the resources on GCP, run:

bash

Copy

terraform apply

* + Terraform will prompt you to confirm before it proceeds. Type yes to proceed with the resource creation.

1. **Monitor the Output**:
   * Terraform will output logs to the terminal showing the progress of resource creation.
   * Once it completes, Terraform will display the output values (e.g., public IP of your instance, unseal key for Vault, etc.).
2. **Verify the Resources**:
   * After Terraform finishes, you can go to the GCP Console to verify that the resources (VM instances, Cloud SQL, Firewall Rules, etc.) were created successfully.

**5. Accessing and Managing Resources**

Once your resources are created, you can interact with them directly from the GCP Console or by using command-line tools.

**Steps to Access Resources:**

1. **Access Cloud SQL**:
   * You can connect to your Cloud SQL instance using MySQL clients or via the Cloud SQL web interface.
   * For connecting from your local machine, use the public IP and authentication credentials you set up in your Terraform script.
2. **Access Vault**:
   * Use the Vault public IP from the output to access the Vault UI (if ui = true was enabled in the configuration).
   * Unseal the Vault instance and login with the root token provided in the output.

bash

Copy

vault operator unseal <UNSEAL\_KEY>

vault login <ROOT\_TOKEN>

1. **Managing Resources**:
   * To manage your resources with Terraform after they are created, you can modify your .tf files and run terraform apply again to make changes.
   * To destroy all resources, run:

bash

Copy

terraform destroy