A complete GCP environment with Terraform

[](https://medium.com/@zuhaibraja?source=post_page-----c087190366f0----------------------)

[Zuhaib Raja](https://medium.com/@zuhaibraja?source=post_page-----c087190366f0----------------------)

Follow

[Jan 8, 2019](https://medium.com/slalom-technology/a-complete-gcp-environment-with-terraform-c087190366f0?source=post_page-----c087190366f0----------------------) · 10 min read





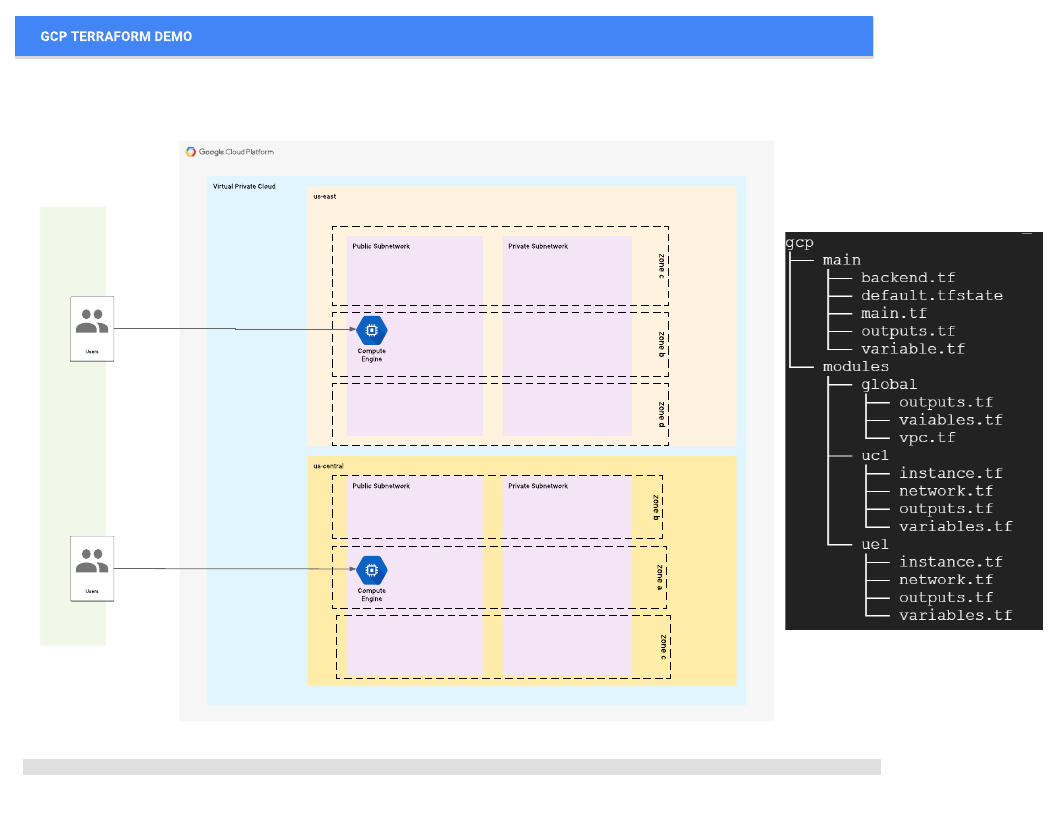
Photo by [Samuel Zeller](https://unsplash.com/@samuelzeller?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

**INTRODUCTION**

The purpose of this article is to show a full Google Cloud Platform (GCP) environment built using Terraform automation. I’ll walk through the setup process to get Google Cloud Platform and Terraform. I will be creating everything from scratch: VPC network, four sub-networks — two in each region (labeling private and Public), firewall rules allowing HTTP traffic and ssh access, and finally creating two virtual instances one in each sub-network running as a web server.

At the end of my deployment, I will have a Google Cloud Platform (GCP) environment setup with two web servers running in different regions as shown below:





GCP Environment and Terraform directory structure

Let’s get started with defining some terms and technology:

**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.

**You might be asking — Why use Terraform?**

**Terraform** is a tool and has become more popular because it has a simple syntax that allows easy modularity and works against multi-cloud. One important reason people consider **Terraform** is to manage their infrastructure as code.

**Installing Terraform:**

It is easy to install it, if you haven’t already. I am using Linux:

*sudo yum install -y zip unzip (if these are not installed)*

*wget*[*https://releases.hashicorp.com/terraform/0.X.X/terraform\_0.X.X\_linux\_amd64.zip*](https://releases.hashicorp.com/terraform/0.X.X/terraform_0.X.X_linux_amd64.zip)*(replace x with your version)*

*unzip terraform\_0.11.6\_linux\_amd64.zip*

*sudo mv terraform /usr/local/bin/*

*Confirm terraform binary is accessible: terraform — version*

*Make sure Terraform works:*

*$ terraform -v*

*Terraform v0.11.6*

**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 infrastructure.

**Download and install Google Cloud SDK:**

*$ curl*[*https://sdk.cloud.google.com*](https://sdk.cloud.google.com/)*| bash*

**Initialize the gcloud environment:**

*$ gcloud init*

You’ll be able to connect your Google account with the gcloud environment by following the on-screen instructions in your browser. If you’re stuck, try checking out the [**official documentation**](https://cloud.google.com/sdk/downloads).

**Configuring our Service Account on Google Cloud Platform**

Next, I will create a project, set up a service account and set the correct permissions to manage the project’s resources.

*· Create a project and name it whatever you’d like.*

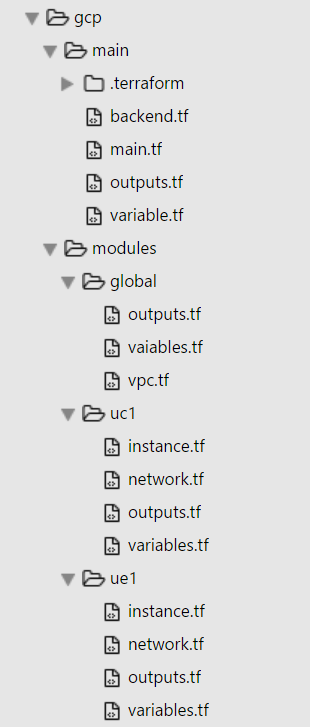
*· Create a service account and specify the compute admin role.*

*· Download the generated JSON file and save it to your project’s directory.*

**TERRAFORM PROJECT FILE STRUCTURE**

Terraform elaborates all the files inside the working directory so it does not matter if everything is contained in a single file or divided into many, although it is convenient to organize the resources in logical groups and split them into different files. Let’s look at how we can do this effectively:





*Terraform File Structure*

***Root level:****All tf files are contained in GCP folder*

***main.tf :****This is where I execute terraform from. It contains following sections:*

*a) Provider section: defines Google as the provider*

*b) Module section: GCP resources that points to each module in module folder*

*c) Output section: Displaying outputs after Terrafrom apply*

***variable.tf:****This is where I am defining all my variables that goes into main.tf. Modules variable.tf contains static values such as regions other variables that I am passing through main variables.tf.*

*Only main variable.tf needs to be modified. I kept it simple so I don’t have to modify every variable file under each module.*

***backend.tf****: For capturing and saving tfstate on Google Storage bucket, that I can share with other developers.*

***Module Folders****: I am using three main modules here. Global, ue1 and uc1*

***\* global****module has resources that are not region specific such as VPC Network, firewall, rules*

***\* uc1 and ue1****module(s) has resources that are region based. The module creates four sub-networks (two public and two private network) two in each region and creating one instance of each region*

Within my directory structure, I have packaged regional-based resources under one module and global resources in a separate module, that way I have to define Variable for a given region, once per module. IAM is another resource that you can define under the global module.

*I am running terraform init, plan and apply from main folder where I have defined all GCP resources. I will post another article in the future dedicated to Terraform modules, when & why it is best to use modules and which resources should be packaged in a module.*

**Main.tf**

Main.tf creates all GCP resources that are defined under each module folder. You can see the *source* is pointing to a relative path with my directory structure. You can also store modules on VCS such as GitHub.

provider "google" {  
 project = "${var.var\_project}"  
}  
module "vpc" {  
 source = "../modules/global"   
 env = "${var.var\_env}"  
 company = "${var.var\_company}"  
 var\_uc1\_public\_subnet = "${var.uc1\_public\_subnet}"  
 var\_uc1\_private\_subnet= "${var.uc1\_private\_subnet}"  
 var\_ue1\_public\_subnet = "${var.ue1\_public\_subnet}"  
 var\_ue1\_private\_subnet= "${var.ue1\_private\_subnet}"  
}  
module "uc1" {  
 source = "../modules/uc1"  
 network\_self\_link = "${module.vpc.out\_vpc\_self\_link}"  
 subnetwork1 = "${module.uc1.uc1\_out\_public\_subnet\_name}"  
 env = "${var.var\_env}"  
 company = "${var.var\_company}"  
 var\_uc1\_public\_subnet = "${var.uc1\_public\_subnet}"  
 var\_uc1\_private\_subnet= "${var.uc1\_private\_subnet}"  
}  
module "ue1" {  
 source = "../modules/ue1"  
 network\_self\_link = "${module.vpc.out\_vpc\_self\_link}"  
 subnetwork1 = "${module.ue1.ue1\_out\_public\_subnet\_name}"  
 env = "${var.var\_env}"  
 company = "${var.var\_company}"  
 var\_ue1\_public\_subnet = "${var.ue1\_public\_subnet}"  
 var\_ue1\_private\_subnet= "${var.ue1\_private\_subnet}"  
}######################################################################  
# Display Output Public Instance  
######################################################################  
output "uc1\_public\_address" { value = "${module.uc1.uc1\_pub\_address}"}  
output "uc1\_private\_address" { value = "${module.uc1.uc1\_pri\_address}"}  
output "ue1\_public\_address" { value = "${module.ue1.ue1\_pub\_address}"}  
output "ue1\_private\_address" { value = "${module.ue1.ue1\_pri\_address}"}  
output "vpc\_self\_link" { value = "${module.vpc.out\_vpc\_self\_link}"}

**Variable.tf**

I have used variables for CIDR range for each sub-network, project name. I am also using variables to name resources gcp resources, so that I can easily identify which environment the resource belongs to. All variables are defined in the variables.tf file. Every variable is of type String.

variable "var\_project" {  
 default = "project-name"  
 }  
variable "var\_env" {  
 default = "dev"  
 }  
variable "var\_company" {   
 default = "company-name"  
 }variable "uc1\_private\_subnet" {  
 default = "10.26.1.0/24"  
 }variable "uc1\_public\_subnet" {  
 default = "10.26.2.0/24"  
 }variable "ue1\_private\_subnet" {  
 default = "10.26.3.0/24"  
 }variable "ue1\_public\_subnet" {  
 default = "10.26.4.0/24"  
 }

**VPC.tf**

In the VPC file, I have configured routing-type as global and I have disabled creation of sub-networks (automatically) as GCP creates sub-networks in every region during VPC creation if not disabled. I am also creating and attaching Firewall to the VPC along with firewall rules to allow icmp, tcp and udp ports within internal network and external ssh access to my bastion host.

resource "google\_compute\_network" "vpc" {  
 name = "${format("%s","${var.company}-${var.env}-vpc")}"  
 auto\_create\_subnetworks = "false"  
 routing\_mode = "GLOBAL"  
}  
resource "google\_compute\_firewall" "allow-internal" {  
 name = "${var.company}-fw-allow-internal"  
 network = "${google\_compute\_network.vpc.name}"  
 allow {  
 protocol = "icmp"  
 }  
 allow {  
 protocol = "tcp"  
 ports = ["0-65535"]  
 }  
 allow {  
 protocol = "udp"  
 ports = ["0-65535"]  
 }  
 source\_ranges = [  
 "${var.var\_uc1\_private\_subnet}",  
 "${var.var\_ue1\_private\_subnet}",  
 "${var.var\_uc1\_public\_subnet}",  
 "${var.var\_ue1\_public\_subnet}"  
 ]  
}  
resource "google\_compute\_firewall" "allow-http" {  
 name = "${var.company}-fw-allow-http"  
 network = "${google\_compute\_network.vpc.name}"allow {  
 protocol = "tcp"  
 ports = ["80"]  
 }  
 target\_tags = ["http"]   
}  
resource "google\_compute\_firewall" "allow-bastion" {  
 name = "${var.company}-fw-allow-bastion"  
 network = "${google\_compute\_network.vpc.name}"  
 allow {  
 protocol = "tcp"  
 ports = ["22"]  
 }  
 target\_tags = ["ssh"]  
 }

**Network.tf**

In the network.tf file, I have set up public and private sub-network and attaching each sub-network to myVPC. The values for regions are coming out of variables.tf files defined within each sub-module folder (not shown here). I have two network.tf files one each module folder, the difference between the two is region *us-east vs us-central*.

resource "google\_compute\_subnetwork" "public\_subnet" {  
 name = "${format("%s","${var.company}-${var.env}-${var.region\_map["${var.var\_region\_name}"]}-pub-net")}"  
 ip\_cidr\_range = "${var.var\_uc1\_public\_subnet}"  
 network = "${var.network\_self\_link}"  
 region = "${var.var\_region\_name}"  
}resource "google\_compute\_subnetwork" "private\_subnet" {  
 name = "${format("%s","${var.company}-${var.env}-${var.region\_map["${var.var\_region\_name}"]}-pri-net")}"  
 ip\_cidr\_range = "${var.var\_uc1\_private\_subnet}"  
 network = "${var.network\_self\_link}"  
 region = "${var.var\_region\_name}"  
}

**Instance.tf**

Here, I am creating a Ubuntu virtual machine instance and a network interface within the sub-network and then I am attaching the network interface to the instance. I am also running a *userdata* script which installs nginx as part of the instance creation and boot. I have two interface.tf files one each module folder, the difference between the two is region *us-east vs us-central*.

resource "google\_compute\_instance" "default" {  
 name = "${format("%s","${var.company}-${var.env}-${var.region\_map["${var.var\_region\_name}"]}-instance1")}"  
 machine\_type = "n1-standard-1"  
 #zone = "${element(var.var\_zones, count.index)}"  
 zone = "${format("%s","${var.var\_region\_name}-b")}"  
 tags = ["ssh","http"]  
 boot\_disk {  
 initialize\_params {  
 image = "centos-7-v20180129"   
 }  
 }labels {  
 webserver = "true"   
 }metadata {  
 startup-script = *<<SCRIPT*  
 apt-get -y update  
 apt-get -y install nginx  
 export HOSTNAME=$(hostname | tr -d '\n')  
 export PRIVATE\_IP=$(curl -sf -H 'Metadata-Flavor:Google' <http://metadata/computeMetadata/v1/instance/network-interfaces/0/ip> | tr -d '\n')  
 echo "Welcome to $HOSTNAME - $PRIVATE\_IP" > /usr/share/nginx/www/index.html  
 service nginx start  
  *SCRIPT*  
 }   
network\_interface {  
 subnetwork = "${google\_compute\_subnetwork.public\_subnet.name}"  
 access\_config {  
 // Ephemeral IP  
 }  
 }  
}

**Executing Terraform scripts using GCloud SDK**

***$ Terraform init***

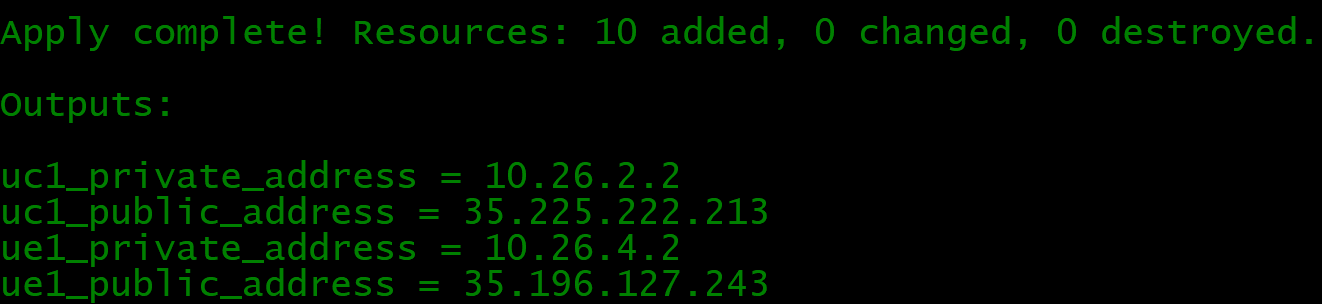
Initializing modules...  
- module.vpc  
- module.uc1  
- module.ue1Initializing provider plugins...The following providers do not have any version constraints in configuration,  
so the latest version was installed.To prevent automatic upgrades to new major versions that may contain breaking  
changes, it is recommended to add version = "..." constraints to the  
corresponding provider blocks in configuration, with the constraint strings  
suggested below.\* provider.google: version = "~> 1.20"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.

***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:  
 + createTerraform will perform the following actions:+ module.uc1.google\_compute\_instance.default  
 id: <computed>  
 boot\_disk.#: "1"  
 boot\_disk.0.auto\_delete: "true"  
 boot\_disk.0.device\_name: <computed>  
 boot\_disk.0.disk\_encryption\_key\_sha256: <computed>  
 boot\_disk.0.initialize\_params.#: "1"  
 boot\_disk.0.initialize\_params.0.image: "debian-9-stretch-v20180227"  
 boot\_disk.0.initialize\_params.0.size: <computed>  
 boot\_disk.0.initialize\_params.0.type: <computed>  
 can\_ip\_forward: "false"  
 cpu\_platform: <computed>  
 create\_timeout: "4"  
 deletion\_protection: "false"  
 guest\_accelerator.#: <computed>  
 instance\_id: <computed>  
 label\_fingerprint: <computed>  
 labels.%: "1"  
 labels.webserver: "true"  
 machine\_type: "n1-standard-1"  
 metadata\_fingerprint: <computed>  
 name: "company-dev-uc1-instance1"  
 network\_interface.#: "1"  
 network\_interface.0.access\_config.#: "1"  
 network\_interface.0.access\_config.0.assigned\_nat\_ip: <computed>  
 network\_interface.0.access\_config.0.nat\_ip: <computed>  
 network\_interface.0.access\_config.0.network\_tier: <computed>  
 network\_interface.0.address: <computed>  
 network\_interface.0.name: <computed>  
 network\_interface.0.network\_ip: <computed>  
 network\_interface.0.subnetwork: "company-dev-uc1-pub-net"  
 network\_interface.0.subnetwork\_project: <computed>  
 project: <computed>  
 scheduling.#: <computed>  
 self\_link: <computed>  
 tags.#: "2"  
 tags.2541227442: "http"  
 tags.4002270276: "ssh"  
 tags\_fingerprint: <computed>  
 zone: "us-central1-a"+ module.uc1.google\_compute\_subnetwork.private\_subnet  
 id: <computed>  
 creation\_timestamp: <computed>  
 fingerprint: <computed>  
 gateway\_address: <computed>  
 ip\_cidr\_range: "10.26.1.0/24"  
 name: "company-dev-uc1-pri-net"  
 network: "${var.network\_self\_link}"  
 project: <computed>  
 region: "us-central1"  
 secondary\_ip\_range.#: <computed>  
 self\_link: <computed>+ module.uc1.google\_compute\_subnetwork.public\_subnet  
 id: <computed>  
 creation\_timestamp: <computed>  
 fingerprint: <computed>  
 gateway\_address: <computed>  
 ip\_cidr\_range: "10.26.2.0/24"  
 name: "company-dev-uc1-pub-net"  
 network: "${var.network\_self\_link}"  
 project: <computed>  
 region: "us-central1"  
 secondary\_ip\_range.#: <computed>  
 self\_link: <computed>+ module.ue1.google\_compute\_instance.default  
 id: <computed>  
 boot\_disk.#: "1"  
 boot\_disk.0.auto\_delete: "true"  
 boot\_disk.0.device\_name: <computed>  
 boot\_disk.0.disk\_encryption\_key\_sha256: <computed>  
 boot\_disk.0.initialize\_params.#: "1"  
 boot\_disk.0.initialize\_params.0.image: "centos-7-v20180129"  
 boot\_disk.0.initialize\_params.0.size: <computed>  
 boot\_disk.0.initialize\_params.0.type: <computed>  
 can\_ip\_forward: "false"  
 cpu\_platform: <computed>  
 create\_timeout: "4"  
 deletion\_protection: "false"  
 guest\_accelerator.#: <computed>  
 instance\_id: <computed>  
 label\_fingerprint: <computed>  
 labels.%: "1"  
 labels.webserver: "true"  
 machine\_type: "n1-standard-1"  
 metadata\_fingerprint: <computed>  
 name: "company-dev-ue1-instance1"  
 network\_interface.#: "1"  
 network\_interface.0.access\_config.#: "1"  
 network\_interface.0.access\_config.0.assigned\_nat\_ip: <computed>  
 network\_interface.0.access\_config.0.nat\_ip: <computed>  
 network\_interface.0.access\_config.0.network\_tier: <computed>  
 network\_interface.0.address: <computed>  
 network\_interface.0.name: <computed>  
 network\_interface.0.network\_ip: <computed>  
 network\_interface.0.subnetwork: "company-dev-ue1-pub-net"  
 network\_interface.0.subnetwork\_project: <computed>  
 project: <computed>  
 scheduling.#: <computed>  
 self\_link: <computed>  
 tags.#: "2"  
 tags.2541227442: "http"  
 tags.4002270276: "ssh"  
 tags\_fingerprint: <computed>  
 zone: "us-east1-b"+ module.ue1.google\_compute\_subnetwork.private\_subnet  
 id: <computed>  
 creation\_timestamp: <computed>  
 fingerprint: <computed>  
 gateway\_address: <computed>  
 ip\_cidr\_range: "10.26.3.0/24"  
 name: "company-dev-ue1-pri-net"  
 network: "${var.network\_self\_link}"  
 project: <computed>  
 region: "us-east1"  
 secondary\_ip\_range.#: <computed>  
 self\_link: <computed>+ module.ue1.google\_compute\_subnetwork.public\_subnet  
 id: <computed>  
 creation\_timestamp: <computed>  
 fingerprint: <computed>  
 gateway\_address: <computed>  
 ip\_cidr\_range: "10.26.4.0/24"  
 name: "company-dev-ue1-pub-net"  
 network: "${var.network\_self\_link}"  
 project: <computed>  
 region: "us-east1"  
 secondary\_ip\_range.#: <computed>  
 self\_link: <computed>+ module.vpc.google\_compute\_firewall.allow-bastion  
 id: <computed>  
 allow.#: "1"  
 allow.803338340.ports.#: "1"  
 allow.803338340.ports.0: "22"  
 allow.803338340.protocol: "tcp"  
 creation\_timestamp: <computed>  
 destination\_ranges.#: <computed>  
 direction: <computed>  
 name: "company-fw-allow-bastion"  
 network: "company-dev-vpc"  
 priority: "1000"  
 project: <computed>  
 self\_link: <computed>  
 source\_ranges.#: <computed>  
 target\_tags.#: "1"  
 target\_tags.4002270276: "ssh"+ module.vpc.google\_compute\_firewall.allow-http  
 id: <computed>  
 allow.#: "1"  
 allow.272637744.ports.#: "1"  
 allow.272637744.ports.0: "80"  
 allow.272637744.protocol: "tcp"  
 creation\_timestamp: <computed>  
 destination\_ranges.#: <computed>  
 direction: <computed>  
 name: "company-fw-allow-http"  
 network: "company-dev-vpc"  
 priority: "1000"  
 project: <computed>  
 self\_link: <computed>  
 source\_ranges.#: <computed>  
 target\_tags.#: "1"  
 target\_tags.2541227442: "http"+ module.vpc.google\_compute\_firewall.allow-internal  
 id: <computed>  
 allow.#: "3"  
 allow.1367131964.ports.#: "0"  
 allow.1367131964.protocol: "icmp"  
 allow.2250996047.ports.#: "1"  
 allow.2250996047.ports.0: "0-65535"  
 allow.2250996047.protocol: "tcp"  
 allow.884285603.ports.#: "1"  
 allow.884285603.ports.0: "0-65535"  
 allow.884285603.protocol: "udp"  
 creation\_timestamp: <computed>  
 destination\_ranges.#: <computed>  
 direction: <computed>  
 name: "company-fw-allow-internal"  
 network: "company-dev-vpc"  
 priority: "1000"  
 project: <computed>  
 self\_link: <computed>  
 source\_ranges.#: "4"  
 source\_ranges.1778211439: "10.26.2.0/24"  
 source\_ranges.2728495562: "10.26.3.0/24"  
 source\_ranges.3215243634: "10.26.4.0/24"  
 source\_ranges.4016646337: "10.26.1.0/24"+ module.vpc.google\_compute\_network.vpc  
 id: <computed>  
 auto\_create\_subnetworks: "false"  
 gateway\_ipv4: <computed>  
 name: "company-dev-vpc"  
 project: <computed>  
 routing\_mode: "GLOBAL"  
 self\_link: <computed>Plan: 10 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.

**Terraform apply Outputs**





Output from Terraform apply