Google Cloud

Terraform as an IaC tool

Previously, you learned that infrastructure as code (IaC) is the practice of provisioning and managing infrastructure using reusable scripts. Organizations that adopt IaC use different tools to help with automating these scripts. One of those tools is Terraform. In this reading, you'll explore Terraform as an IaC tool that automates the creation of infrastructure, and review an example of Terraform in action.

Terraform and IaC

Terraform is a tool that automates the creation and deployment of infrastructure in the cloud. And, it's widely used for many reasons. First, Terraform is a cloud-agnostic tool, which means it's compatible with several cloud platforms. Next, developers are able to use it to track resource and code changes throughout deployment. For example, implementing IaC can help developers detect invalid inputs in the build process. This means that IaC's automation capabilities can alert developers when they make an error.

Consider this example: A developer is using Terraform as an IaC tool to create server instances. When they create the variable to determine the number of instances to create, they define that it must range from 1-10. Terraform would then alert developers to any number inserted outside of that range.

IaC also helps reduce configuration drift, or the deviation of the system's configuration from the way it was configured. Configuration drift often happens when administrators make changes to systems after they've been deployed. These changes are not captured in the documented configuration, which makes it harder to rebuild the system. With IaC, instead of making manual changes to a system, developers update the IaC code and re-run the IaC tool, like Terraform, to apply the changes. This helps maintain the consistency between the IaC code and the actual state of the system, and helps eliminate configuration drift.

Terraform configuration files

Terraform uses configuration files to store the source code that defines the desired infrastructure. A configuration file is a document that contains the code needed to build the infrastructure. Cloud service providers (CSPs) can use Terraform to automate the creation of virtual machines (VMs), virtual private clouds (VPCs), and networks to align with the desired state. For example, you can use Google's Compute Engine and Terraform to create VMs.

Resource blocks and resources

Resources are a building block of configuration files. In the configuration file, resources define what Terraform should create. For example, if a developer is using Terraform to create VMs, they would need to determine the VM's resources, variables, and location in the configuration file.

This list describes each component of the configuration file:

- Resource: defines what Terraform is creating (e.g., a network or VM)
- Variable: defines the name of the resource
- Location: names the resource's deployment location, designated by region and zone

First resource block

Terraform labels each block of code in a configuration file a resource block, where the block begins with the `resource` keyword. In the first resource block, a resource is defined for Terraform to create. The first value after `resource` is the type of resource being created. The third value in the line is the variable name assigned to the resource. For example, the variable could be named `vpc_network`.

```
Unset
resource “google_compute_network” “vpc_network” {
      name                          = “my-custom-mode-network”
      auto_create_subnetworks       = false
      mtu                           = 1460
}
```

Second resource block

In the second resource block, another resource is created. In this configuration file, you're creating a subnetwork and IP range. You also assign the resource to a region.

```
Unset
resource “google_compute_subnetwork” “default” {
      name              = “my-custom-subnet”
      ip_cidr_range     = “10.0.1.0/24”
```

```
      region           = “us-west1”
      network          = google_compute_network.vpc_network.id
}
```

Third resource block

In the third resource block, you create another resource. For example, a Compute Engine instance, which is a VM. In this block, you assign the resource’s zone.

Unset

```
# Create a single Compute Engine instance
resource “google_compute_instance” “default” {
      name             = “flask-vm”
      machine_type     = “f1_micro”
      zone             = “us-west1-a”
      tags             = [“ssh”]
```

As a cloud security analyst, you might encounter the need to change the parameters of the script to meet organizational needs. For example, you may be asked to change the location of VMs to fulfill user requests, or to adjust to shifting organizational requirements. To change the location, select a new zone and its corresponding region.

Key takeaways

IaC makes it easier for development teams to develop and deploy infrastructure. With IaC tools like Terraform, infrastructure like VMs or VPCs can quickly be created and distributed. And, the configuration can be revised to accommodate any updates needed for the deployment. As a cloud security professional, you’ll work with tools like Terraform to orchestrate IaC. Knowing how resources are created with IaC provides a foundation for provisioning cloud infrastructure.

Resources for more information

Check out this resource to learn more about Terraform:

- This Terraform page explains resource blocks, syntax, and types.