# Dynatrace and Terraform

By Riley McClelland

## What is Terraform?

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. This includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc. Terraform can manage both existing service providers and custom in-house solutions.

## Why do we need to know?

As Dynatrace Experts we need to know how best to integrate Dynatrace with IaC tools such as terraform so that we can install Dynatrace agents quickly and efficiently. If we can implement the Dynatrace agent through Terraform, then this code can be copied across multiple different environments allowing for easier implementation of Dynatrace. Additionally, clients will commonly have a ‘base build’ for their environment, these include OS, Box Specifications, and permissions. If we as Dynatrace experts can implement Dynatrace onto that ‘base build’ then Dynatrace will be included with every single box spun up in the client infrastructure.

## Tutorial (GCP)

These steps will detail how I was able to implement the Dynatrace Agent using Terraform on the GCP platform.

### Problem Statement:

I as a Dynatrace Expert want to implement the Dynatrace OneAgent as code using Terraform

### Best Practices

* I should be able to customize variables to easily select the Dynatrace Tenancy
* I should be able to customize variables to implement Host Groups and other Install arguments
* Dynatrace PAAS Token should never appear in plaintext in any code and instead should be input via argument

One of the best ways to learn how to use Terraform is to go through the tutorials and then use the registry as a reference to develop additional resources. Due to this I will be adding links to additional pages I used to solve this problem. Each page should be from either GCP or Terraform Documentation.

|  |  |  |
| --- | --- | --- |
| # | Step | Example Image |
| 1 | First complete the following Terraform Tutorial so that a VM Instance is created within GCP:  <https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code?in=terraform/gcp-get-started> |  |
| 2 | Next define your variables in a variables.tf file. We need to define the following variables:   * Dynatrace Tenant * Agent Arguments * Dynatrace Tokens   *Note: There may be other variables set, I am only showing the Dynatrace related ones. Other variables might be the GCP Project ID set as part of the tutorial*  *Reference: https://learn.hashicorp.com/tutorials/terraform/google-cloud-platform-variables?in=terraform/gcp-get-started* | variable "dt\_tenant" {  type = string  description = "This is the ID of the Dynatrace tenant https://\_\_\_\_\_\_\_\_\_.live.dynatrace.com"  default = "<Put Dynatrace ID Here>"  }  variable "agent\_arg" {  type = string  description = "These are the agent arguments used upon installation of the Dynatrace Agent"  default = "<Put Default Agent Arguments Here>"  }  variable "dt\_token" {  type = string  description = "This is the Dynatrace PAAS Token used to download the agent file"  sensitive = true  }  #The sensitive argument here ensures that the value will not appear in any plans thereby exposing the token |
| 3 | Next we need to create our secret in GCP. GCP Secrets work through creating a base secret, and then the value of the secret is then held in a ‘version’. Due to this we need to do the same in Terraform creating both a ‘secret-basic’ and a ‘secret-version-basic’  *Note: In order to complete this step your terraform service account must have the ‘secret manager admin role’ please see these instructions for more help:* *https://cloud.google.com/secret-manager/docs/creating-and-accessing-secrets*  *Reference: https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/secret\_manager\_secret\_version* | resource "google\_secret\_manager\_secret" "secret-basic" {  secret\_id = "secret-version"  labels = {  label = "my-label"  }  replication {  automatic = true  }  }  resource "google\_secret\_manager\_secret\_version" "secret-version-basic" {  secret = google\_secret\_manager\_secret.secret-basic.id  secret\_data = var.dt\_token  } |
| 4 | Due to the complexity of the vm instance file I will split this out into two parts, the vm instance section and the startup script.  First the VM Instance itself, will need to have the correct OS and specifications to run a Dynatrace Agent. Due to this I changed the specifications from the Terraform Tutorial to ‘e2-micro’ and ‘rhel-8’  Next I ensured that the network interface was the default, this allows us to SSH onto the vm instance ourselves to verify if there are any issues with the Dynatrace agent. *Note this requires the default network firewall rules to have port 22 open.*  After this I set the service account to one of my service accounts in GCP. Note that this service account must have the secretAccessor role in order for it to access the Dynatrace Token Secret.  Finally I added metadata to allow for oslogin, this was another step designed to making logging onto the box easier. Reference can be found here: *https://cloud.google.com/compute/docs/instances/access-overview* | resource "google\_compute\_instance" "vm\_instance" {  name = "terraform-instance"  machine\_type = "e2-micro" #This sets the specifications of the VM Instance  allow\_stopping\_for\_update = true #This allows Terraform to tear down the instance for updates if needed    boot\_disk {  initialize\_params {  image = "rhel-cloud/rhel-8"  }  }  network\_interface {  network = "default"  access\_config {  }  }    service\_account{  email = "982955065423-compute@developer.gserviceaccount.com"  scopes = ["cloud-platform"]  }  metadata\_startup\_script = \*cut from this step\*  metadata = {  enable-oslogin = "TRUE"  }  } |
| 5 | Now for the startup script, GCP allows for a startup script to be set as metadata for the VM instance. Due to this we need to be careful to ensure the Dynatrace Token does not appear in the VM Instance Metadata.  *Note that everything between ‘<<SCRIPT SCRIPT’ is all one line, due to limitations of word they have been broken up. Please see screenshot for more info.*  The start-up script is 4 commands separated by a semicolon the first command is as follows:  TOKEN=$(gcloud secrets versions access latest --secret="secret-version" --format='get(payload.data)' | tr '\_-' '/+' | base64 -d);  This command extracts the Dynatrace token secret value from the secret manager in GCP. It is then assigned to the variable ‘TOKEN’ to be used in a future command | metadata\_startup\_script = <<SCRIPT  TOKEN=$(gcloud secrets versions access latest --secret="secret-version" --format='get(payload.data)' | tr '\_-' '/+' | base64 -d);curl -X GET "https://${var.dt\_tenant}.live.dynatrace.com/api/v1/deployment/installer/agent/unix/default/latest?flavor=default&arch=all&bitness=all&skipMetadata=false&networkZone=default" -H "accept: \*/\*" -H "Authorization: Api-Token $TOKEN" > /var/tmp/dynatrace-install.sh;chmod 755 /var/tmp/dynatrace-install.sh;sudo ./bin/sh /var/tmp/dynatrace-install.sh ${var.agent\_arg};  SCRIPT |
| 6 | The next command downloads the Dynatrace Agent and inserts it into a Dynatrace shell install file.  curl -X GET "https://${var.dt\_tenant}.live.dynatrace.com/api/v1/deployment/installer/agent/unix/default/latest?flavor=default&arch=all&bitness=all&skipMetadata=false&networkZone=default" -H "accept: \*/\*" -H "Authorization: Api-Token $TOKEN" > /var/tmp/dynatrace-install.sh;  Here Terraform will change the variable ‘dt\_tenant’ to the requisite value set in the variables file.  *Note: curl was used here as opposed to wget because the GCP RHEL VM Starts with curl. This could be any command allowing for download of the agent* |  |
| 7 | Next we change the permissions of the downloaded file so that it can be executed  chmod 755 /var/tmp/dynatrace-install.sh; |  |
| 8 | Finally we execute the downloaded shell script  sudo ./bin/sh /var/tmp/dynatrace-install.sh ${var.agent\_arg};  Terraform automatically turns the referenced variable ‘agent\_arg’ into the related string allowing us to insert as many agent install arguments as we want provided they are formatted correctly. |  |
| 9 | With all this done we should see the agent show up in Dynatrace after a few minutes: | Graphical user interface, application  Description automatically generated |

## Authenticating to GCP

Dynatrace would prefer that staff use their personal accounts for the use of Terraform rather than creating a service Account. This is primarily done to minimize the amount of service accounts created as sharing a single one would effectively be sharing credentials.

Due to this here is the steps required to authenticate against GCP using the google cloud sdk

|  |  |  |
| --- | --- | --- |
| # | Step | Screenshot |
| 1 | Install Terraform as per these instructions  <https://developer.hashicorp.com/terraform/tutorials/gcp-get-started/install-cli> |  |
| 2 | Install gcloud CLI as per these instructions  https://cloud.google.com/sdk/docs/install |  |
| 3 | In your Terraform location create the following main.tf | terraform {  required\_providers {  google = {  source = "hashicorp/google-beta"  version = "4.57.0"  }  }  }  provider "google" {  project = "dxs-apac"  region = "australia-southeast1"  zone = "australia-southeast1-a"  }  resource "google\_compute\_network" "vpc\_network" {  name = "terraform-network"  } |
| 4 | Run the command  gcloud auth application-default login |  |
| 5 | Step through the google cloud login process to authenticate your computer |  |
| 6 | You are now authenticated using your staff google cloud credentials! |  |