Terraform plugin caching

Dominik Żyła
Feb 24 · 2 min read

Terraform became a standard and tool of choice for many for Infrastructure as Code (IaC) these days. Its learning curve isn't steep, and engineers can start using it relatively quickly.

Terraform uses provider plugins which enable interaction with the given cloud provider or a service, such as AWS, GCP, Azure or MySQL.

Using providers

All the providers used by the project have to be declared in the following way:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
    mysql = {
      source  = "terraform-providers/mysql"
      version = ">= 1.6"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Demo VPC that will hold MySQL RDBMS
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}

# …more resources…

# Configure the MySQL provider
provider "mysql" {
  endpoint = "my-database.example.com:3306"
  username = "app-user"
  password = "app-password"
}

# Create a Database
resource "mysql_database" "app" {
  name = "my_awesome_app"
}
```

Once declared, they have to be downloaded into the project. The `terraform init` command will download and store them inside of the `.terraform` in your project's root directory. Pretty straightforward, innit?

The catch

The scenario presented above works well but the providers have to be downloaded for each project you're working with and these are rather big files (hundreds of megabytes binaries). On a slower internet connection this could become a problem.

Plugins caching

Luckily Terraform allows for plugins caching. So, whenever plugin has to be downloaded and is present in the cache directory, it will be copied into the project instead. This can save some time and bandwidth.

Put this in your `~/.terraformrc` and enjoy:

```
plugin_cache_dir = "$HOME/.terraform.d/plugin-cache"
```

Then create the cache directory:

```
$ mkdir -p $HOME/.terraform.d/plugin-cache
```

Beyond local execution

As much as caching is useful for local development, you definitely want to run your terraform inside of some CI system.

Having many pipelines and running `terraform init` at the beginning of each run can generate quite some traffic and take some extra time.

This is really where plugins caching comes in handy.

I'm a consultant at The Scale Factory where we empower technology teams to deliver and secure workloads on the AWS cloud, through consultancy, engineering, support, and training. If you'd like to find out how we can support you and your team to secure your AWS account, get in touch.

Terraform Infrastructure As Code


Download on the
App Store


GET IT ON
Google Play