Lab 7.1: Terraform
ID720 Virtualisation

Introduction

We have been learning about the OpenStack API and accessing it with the Python SDK. One thing we keep in mind is that *any* cloud service will have similar tools available. Given that, the next logical step is to ask if we could have a general tool that abstracts out the differences between these various platforms. Then we could write out one description of some virtual machines we want to run and apply it to any cloud platform.

It turns out that we can have tools that are **close** to meeting that goal, but they don't quite work on any platform. There are too many differences between these platforms to allow us to write one script and run it anywhere, but we get pretty close. One such tool is called *Terraform* (`https://www.terraform.io`). In this lab we will write a simple Terraform configuration and use it to create and destroy a VM instance on OpenStack.

All of the steps below must be carried out on our `cloudapi` server.

1 Setup

When we run Terraform it will read all the files in the working directory ending in `.tf`, so start by creating a subdirectory of your home directory called `terraform`. Place any files for this lab in that directory.

When using other OpenStack tools we loaded credentials from either an RC file or a `clouds.yaml` file. For today's lab we will use both. (This doesn't exactly make sense, but it does work.) Your OpenStack RC file may still be in your home directory, or you can download a new one. Copy the `clouds.yaml` file from your Python code directory into your new Terraform directory.

2 Configuring our provider

Create a new file in your Terraform directory named `lab7.tf`. This file will contain two elements, a *provider* and a *resource*.

A Terraform provider tells Terraform what sort of cloud service you are using. Our provider configuration is simple, and looks like this:

```
terraform {
required_version = ">= 0.14.0"
 required_providers {
   openstack = {
     source  = "terraform-provider-openstack/openstack"
     version = "~> 1.35.0"
     }
   }
 }
```

```
provider "openstack" {
  cloud = "catalystcloud"
}
```

The first section tells terraform that we will use the OpenStack provider and provides details about the provider resources that will need to be downloaded from the provider repository.

The next section tells Terraform to use this provider, and tells it to look for the "`catalystcloud`" cloud in our `clouds.yaml` file. Once you've added this element, save your file and return to your shell inside your Terraform directory.

3 Initialising the project

Now that we have some minimal configuration, we can initialise our project. First, don't forget to activate your RC file with the command

```
source ~/tom-clark-openrc.sh
```

to populate some needed environment variables. Now, run the command

```
terraform init
```

This will create a `.terraform` directory inside our project directory, and will also ensure that the OpenStack plugin is installed since we indicates that we are using an OpenStack provider.

4 Configure a resource

To actually manage OpenStack items with Terraform, we need to define *resources*. For this lab we will configure a resource defining a virtual machine instance. Add the following to your `.tf` file.

```
resource "openstack_compute_instance_v2" "<your username>" {
  name = "<your username>"
  image_id = "2b3ca33a-7587-49e0-95b8-c22206f74481"
  flavor_id = "99fb31cc-fdad-4636-b12b-b1e23e84fb25"
  key_pair = "<your keypair name>"
  security_groups = ["default"]
  network {
    name = "private-net"
  }
}
```

This configuration tells Terraform that we want a compute instance. The image id and flavor id are the GUIDs of items defined in our OpenStack instance and were found be checking the web dashboard. You may should verify that those are correct values for the Ubuntu 10.04 minimal image and the `c1.c1r05` flavor.

Once this resource is defined in your file, return to your shell and run the command

```
terraform apply
```

Terraform will display it's *plan* for what changes will be made to our OpenStack project to satisfy the configuration. Once you say "yes" to this proposed plan Terraform will carry out the planned steps. Once this is done, check the web dashboard to verify that your VM instance was created.

Next re-run `terraform apply` and check the plan. Since the VM instance you configured is already running, Terraform concludes that there is nothing to do.

5 Clean up

I keep reminding people to not leave unneeded items running in OpenStack. Terraform makes it easy in this case. Simply run

```
terraform destroy
```

and confirm the plan to remove any resources configured in our `.tf` files from our OpenStack project.

More information is available at

- https://www.terraform.io/language (information about writing Terraform configurations)
- https://www.terraform.io/cli (information about the CLI)