WIKIMEDIA FOUNDATION

How to ~~Terraform~~ OpenTofu on Cloud VPS

Taavi Väänänen
Developer Experience 2023 SF offsite

Cloud VPS

- Mostly based on upstream OpenStack components
- Some components maintained by us (proxy service, Puppet ENC)
- Both OpenStack and our own software exposes HTTP APIs to interact with

Authentication

- OpenStack keystone deals with developer accounts
- Everything else deal with Keystone issued tokens
  - Previously, our custom components did not
- Explicit goal: interact with developer account passwords as little as possible
  - Solution*: application credentials

OpenTofu

- Declarative infrastructure as code tool
- Abstraction level: providers (“OpenStack”) that have resources (“compute instance”)
- Custom configuration language, HCL
- Linux Foundation project, fork of HashiCorp’s Terraform
- Apache 2.0 licensed

Example

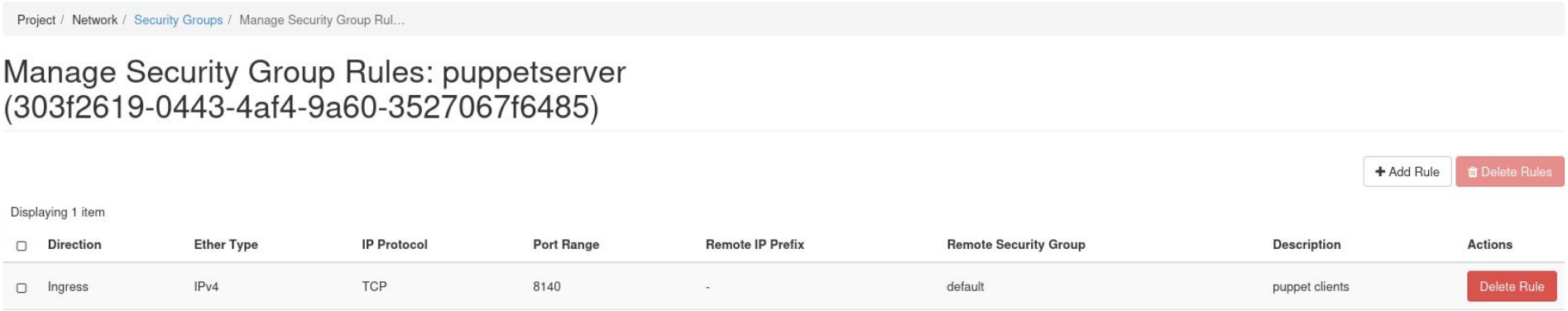
```
data "openstack_networking_secgroup_v2" "default" {
  name = "default"
}

resource "openstack_networking_secgroup_v2" "puppetserver_security_group" {
  name = "puppetserver"
}

resource "openstack_networking_secgroup_rule_v2" "puppetserver_access" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "tcp"
  port_range_min    = 8140
  port_range_max    = 8140
  security_group_id = openstack_networking_secgroup_v2.puppetserver_security_group.id
  remote_group_id   = data.openstack_networking_secgroup_v2.default.id
  description       = "puppet clients"
}
```

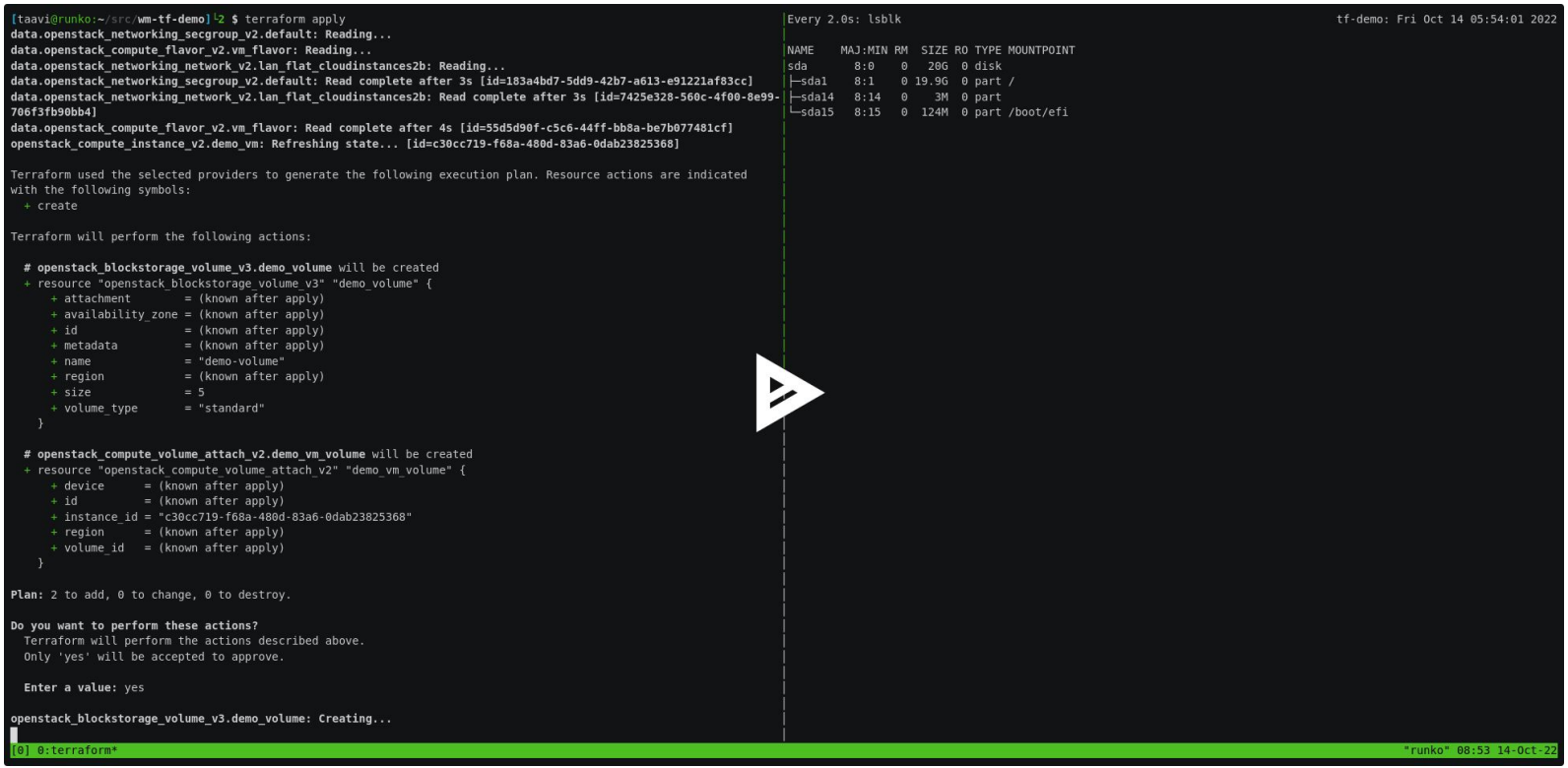
Example

```
data "openstack_networking_secgroup_v2" "default" {
  name = "default"
}
```




```
  port_range_max    = 8140
  security_group_id = openstack_networking_secgroup_v2.puppetserver_security_group.id
  remote_group_id   = data.openstack_networking_secgroup_v2.default.id
  description       = "puppet clients"
}
```

Example


[taavi@runko:~/src/wm-tf-demo]└2 $ terraform apply
data.openstack_networking_secgroup_v2.default: Reading...
data.openstack_compute_flavor_v2.vm_flavor: Reading...
data.openstack_networking_network_v2.lan_flat_cloudinstances2b: Reading...
data.openstack_networking_secgroup_v2.default: Read complete after 3s [id=183a4bd7-5dd9-42b7-a613-e91221af83cc]
data.openstack_networking_network_v2.lan_flat_cloudinstances2b: Read complete after 3s [id=7425e328-560c-4f00-8e99-706f3fb90bb4]
data.openstack_compute_flavor_v2.vm_flavor: Read complete after 4s [id=55d5d90f-c5c6-44ff-bb8a-be7b077481cf]
openstack_compute_instance_v2.demo_vm: Refreshing state... [id=c30cc719-f68a-480d-83a6-0dab23825368]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:
# openstack_blockstorage_volume_v3.demo_volume will be created
+ resource "openstack_blockstorage_volume_v3" "demo_volume" {
+ attachment = (known after apply)
+ availability_zone = (known after apply)
+ id = (known after apply)
+ metadata = (known after apply)
+ name = "demo-volume"
+ region = (known after apply)
+ size = 5
+ volume_type = "standard"
}
# openstack_compute_volume_attach_v2.demo_vm_volume will be created
+ resource "openstack_compute_volume_attach_v2" "demo_vm_volume" {
+ device = (known after apply)
+ id = (known after apply)
+ instance_id = "c30cc719-f68a-480d-83a6-0dab23825368"
+ region = (known after apply)
+ volume_id = (known after apply)
}
Plan: 2 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes
openstack_blockstorage_volume_v3.demo_volume: Creating...
Every 2.0s: lsblk
tf-demo: Fri Oct 14 05:54:01 2022
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 20G 0 disk
├─sda1 8:1 0 19.9G 0 part /
├─sda14 8:14 0 3M 0 part
└─sda15 8:15 0 124M 0 part /boot/efi
[0] 0:terraform*
"runko" 08:53 14-Oct-22

State

- Mapping between defined resources and those that actually exist
- By default, stored in a file in the project directory
- Should be kept private, might contain secrets
- Needs to be shared with everyone working on the project
- The Cloud VPS object storage service seems to work fine for this with the S3 interface. See Wikitech for examples

The good

- Most OpenStack resources work fine with the OpenStack provider
- Cloud VPS proxy integration works fine

The bad

- Application credentials are a bit janky
  - Only allows access to a single project, except I would not rely on that isolation working
- Some OpenStack APIs (eg. Trove/DBaaS) aren't very stable, and a single failing resource will fail the entire run
- Too early to tell how the OpenStack Terraform provider will continue to be maintained after Terraform/OpenTofu fork

The ugly: Puppet integration

- Resource for managing Puppet ENC data (prefix roles and hiera)
- Hiera YAML formatting handled poorly by Terraform
- No support for the certificate signing dance

```
resource "cloudvps_puppet_prefix" "puppetserver" {
  name  = "metricsinfra-puppet-"
  roles = ["role::puppetserver::cloud_vps_project"]
  hiera = file("${path.module}/puppetserver_hiera.yaml")
}
```

Example users in Cloud VPS

- By me
  - terraform.wmcloud.org (the provider registry)
  - metricsinfra (Cloud VPS monitoring-as-a-service)
- By other WMCS admins
  - PAWS, superset.wmcloud.org, tf-infra-test
- By the community
  - account-creation-assistance, Quarry
- Others I don't know about? Please list yourself on Wikitech or otherwise make yourself known

WIKIMEDIA FOUNDATION

More details:

terraform.wmcloud.org[1]

[1] yes, the domain name change is WIP

taavi@wikimedia.org