How to use Ansible with Terraform

Now, returning to the actual provisioning I found 3 ways to use Ansible with Terraform after reading the heated discussion at [this GitHub issue](https://github.com/hashicorp/terraform/issues/2661). Read on to find the one that’s most suitable for you.

Inline inventory with instance IP

One of the most obvious yet hacky solutions is to invoke Ansible within local-exec provisioner. Here is how it looks like:

provisioner "local-exec" {

command = "ansible-playbook -i '${self.public\_ip},' --private-key ${var.ssh\_key\_private} provision.yml"

}

Nice and simple, but there is a problem here. local-exec provisioner starts without waiting for an instance to launch, so in the most cases, it will fail because by the time it will try to connect there is nobody listening.

As a nice workaround, you can use preliminary remote-exec provisioner that will wait until the connection to the instance is established and then invoke the local-exec provisioner.

As a result, I have this thingy that plays the role of “Ansible provisioner”

provisioner "remote-exec" {

inline = ["sudo dnf -y install python"]

connection {

type = "ssh"

user = "fedora"

private\_key = "${file(var.ssh\_key\_private)}"

}

}

provisioner "local-exec" {

command = "ansible-playbook -u fedora -i '${self.public\_ip},' --private-key ${var.ssh\_key\_private} provision.yml"

}

To make ansible-playbook work you have to have an Ansible code in the same directory with Terraform code like this:

$ ll infra

drwxrwxr-x. 3 avd avd 4.0K Mar 5 15:54 roles/

-rw-rw-r--. 1 avd avd 367 Mar 5 15:19 ansible.cfg

-rw-rw-r--. 1 avd avd 2.5K Mar 7 18:54 main.tf

-rw-rw-r--. 1 avd avd 454 Mar 5 15:27 variables.tf

-rw-rw-r--. 1 avd avd 38 Mar 5 15:54 provision.yml

This inline inventory will work in most cases, except when you need multiple hosts in inventory. For example, when you setup Consul agent you need a list of Consul servers for rendering a config and that is usually found in the usual inventory. So but it won’t work here because you have a single host in your inventory.

Anyway, I’m using this approach for the basic things like setting up users and installing some basic packages.

Dynamic inventory after Terraform

Another simple solution for provisioning infrastructure created by Terraform is just don’t tie Terraform and Ansible together. Create infrastructure with Terraform and then use Ansible with dynamic inventory regardless of how your instances were created.

So you first create an infra with terraform apply and then you invoke ansible-playbook -i inventory site.yml, where inventory dir contains dynamic inventory scripts.

This will work great but has a little drawback – if you need to increase the number of instances you must remember to launch Ansible after Terraform.

That’s what I use complementary to the previous approach.

Inventory from Terraform state

There is another interesting thing that might work for you – generate static inventory from Terraform state.

When you work with Terraform it maintains the state of the infrastructure that contains everything including your instances. With a [local backend](https://www.terraform.io/docs/backends/index.html), this state is stored in a JSON file that can be easily parsed and converted to the Ansible inventory.

Here are 2 projects with examples that you can use if you want to go this way.

<https://github.com/adammck/terraform-inventory>

$ terraform-inventory -inventory terraform.tfstate

[all]

52.51.215.84

[all:vars]

[server]

52.51.215.84

[server.0]

52.51.215.84

[type\_aws\_instance]

52.51.215.84

[name\_c10k server]

52.51.215.84

[%\_1]

52.51.215.84

<https://github.com/express42/terraform-ansible-example/blob/master/ansible/terraform.py>

$ ~/soft/terraform.py --root . --hostfile

## begin hosts generated by terraform.py ##

52.51.215.84 C10K Server

## end hosts generated by terraform.py ##

IMHO, I don’t see a point in this approach.

Ansible plugin for Terraform that didn’t work for me

Finally, there are few projects that try to make a native looking Ansible provisioner for Terraform like builtin Chef provisioner.

<https://github.com/jonmorehouse/terraform-provisioner-ansible> – this was the first attempt to make such plugin but, unfortunately, it’s not currently maintained and moreover it’s not supported by the current Terraform plugin system.

<https://github.com/radekg/terraform-provisioner-ansible> – this one is more recent and currently maintained. It enables this kind of provisioning:

...

provisioner "ansible" {

plays {

playbook = "./provision.yml"

hosts = ["${self.public\_ip}"]

}

become = "yes"

local = "yes"

}

...

Unfortunately, I wasn’t able to make it work so I blew it off because first 2 solutions cover all of my cases.

Conclusion

Terraform and Ansible is a powerful combo that I use for provisioning cloud infrastructure. For basic cloud instances setup, I invoke Ansible with local-exec and later I invoke Ansible separately with dynamic inventory.