Terraform Provisioners

Terraform Provisioners - Introduction

- Provisioners can be used to model specific actions on the local machine or on a remote machine in order to prepare servers or other infrastructure objects for service.
- A connection block nested in a provisioner block only affects that provisioner, and overrides any resource-level connection settings.
- **Built-in Provisioners:**
  - local-exec
  - remote-exec
  - Chef

Provisioners - Local-exec

- The local-exec provisioner invokes a local executable after a resource is created.
- This invokes a process on the machine running Terraform, not on the resource.

```
resource "aws_instance" "web" {
  # ...

  provisioner "local-exec" {
    command = "echo The server's IP address is ${self.private_ip}"
  }
}
```

```
resource "aws_instance" "web" {
  # ...

  provisioner "local-exec" {
    command = "echo first"
  }

  provisioner "local-exec" {
    command = "echo second"
  }
}
```

Provisioners - Remote-exec

- The remote-exec provisioner invokes a script on a remote resource after it is created.
- This can be used to run a configuration management tool, bootstrap into a cluster, etc.
- The remote-exec provisioner supports both ssh and winrm type connections.
- inline - This is a list of command strings. They are executed in the order they are provided.

```
provisioner "remote-exec" {
   inline = [
     "sudo amazon-linux-extras enable nginx1.12",
     "sudo yum -y install nginx",
     "sudo systemctl start nginx",
   ]
 }
```

Provisioners - Chef

- The chef provisioner installs, configures and runs the Chef Client on a remote resource.
- The chef provisioner supports both ssh and winrm type connections.

Requirements:

- For ssh type connections, cURL must be available on the remote host.
- For winrm connections, PowerShell 2.0 must be available on the remote host.
- Without these prerequisites, your provisioning execution will fail.

Chef Provisioner - Sample Code

```
resource "aws_instance" "web" {
  provisioner "chef" {
    attributes_json = <<EOF
      {
        "key": "value",
        "app": {
          "cluster1": {
            "nodes": [
              "webserver1",
              "webserver2"
            ]
          }
        }
      }
    EOF

    environment     = "_default"
    client_options  = ["chef_license 'accept'"]
    run_list        = ["cookbook::recipe"]
    node_name       = "webserver1"
    secret_key      = "${file("../encrypted_data_bag_secret")}"
    server_url      = "https://chef.company.com/organizations/org1"
    recreate_client = true
    user_name       = "bork"
    user_key        = "${file("../bork.pem")}"
    version         = "15.10.13"
    # If you have a self signed cert on your chef server change this to :verify_none
    ssl_verify_mode = ":verify_peer"
  }
}
```