Exploring Provisioners in Terraform

One powerful feature in Terraform is the use of provisioners, which enable you to execute scripts and commands on your infrastructure resources. This blog post will explore what provisioners are, their syntax, and practical use cases to help you leverage them effectively in your Terraform configurations.

What are Provisioners?

Provisioners in Terraform are used to execute scripts or commands on a resource after it has been created or before it is destroyed. They provide a way to customize and configure resources beyond what is possible with resource arguments alone. Common use cases for provisioners include bootstrapping instances, installing software, and running configuration management tools.

Types of Provisioners

Terraform supports several types of provisioners, including:

1. `local-exec`: Executes commands on the machine running Terraform.
2. `remote-exec`: Executes commands on the resource being provisioned, typically via SSH or WinRM.
3. `file`: Uploads files to the resource being provisioned.
4. **Third-party provisioners**: Custom provisioners provided by the community

1. Using `local-exec`

The `local-exec` provisioner executes commands on the machine running Terraform. This can be useful for running local scripts or commands after a resource is created.

```
resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
  provisioner "local-exec" {
    command = "echo ${aws_instance.example.id} > instance_id.txt"
  }
}
```

In this example, the `local-exec` provisioner runs a command to write the instance ID to a local file after the AWS EC2 instance is created.

2. Using `remote-exec`

The `remote-exec` provisioner executes commands on the resource being provisioned, typically via SSH or WinRM. This is useful for bootstrapping instances and running configuration scripts.

```
resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
  connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key = file("~/.ssh/id_rsa")
    host     = self.public_ip
  }
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get update",
      "sudo apt-get install -y nginx"
    ]
  }
}
```

3. Using `file`

The `file` provisioner uploads files to the resource being provisioned. This is useful for transferring configuration files or scripts to the resource.

```
resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
  connection {
    type     = "ssh"
    user     = "ec2-user"
    private_key = file("~/.ssh/id_rsa")
    host     = self.public_ip
  }
  provisioner "file" {
    source      = "myapp.conf"
    destination = "/etc/myapp/myapp.conf"
  }
}
```

In this example, the `file` provisioner uploads a configuration file to the EC2 instance.

Destroy-Time Provisioners

Destroy-time provisioners run when a resource is destroyed. They are used for cleanup tasks, such as removing temporary files, deregistering from external services, or sending notifications. **`local-exec` Provisioner for Destroy-Time**

You can configure the `local-exec` provisioner to run on resource destruction by using the `when = destroy` argument.

```
resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
  provisioner "local-exec" {
    when    = destroy
```

```
    command = "echo ${aws_instance.example.id} is being destroyed"
  }
}
```

`remote-exec` Provisioner for Destroy-Time

Similarly, you can configure the `remote-exec` provisioner to run on resource destruction.

When to Use Provisioners

Provisioners can be powerful, but they should be used sparingly and with caution. They are not idempotent and can introduce complexity and brittleness into your Terraform configurations. Use provisioners for:

- **Bootstrapping instances**: Installing initial software and configuration.
- **Configuration management**: Running scripts or configuration management tools.
- **Data transfer**: Uploading files or scripts to instances.