## **Module 1: Understanding Provisioners in Terraform**

1.1 Understanding the Challenge

Till now we have been working only on the creation and destruction of infrastructure scenarios.

Let’s take an example:

We created a web-server EC2 instance with Terraform.

Problem: It is only an EC2 instance, it does not have any software installed.

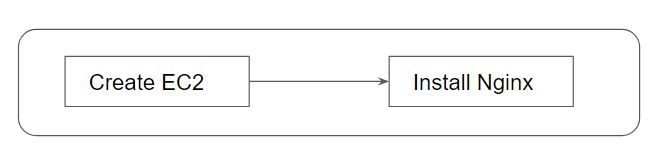
What if we want a complete end to end solution?

1.2 Introducing Terraform Provisioners

Provisioners are used to execute scripts on a local or remote machine as part of resource creation or destruction.

Let’s take an example:

On creation of Web-Server, execute a script which installs Nginx web-server.



## 

## 

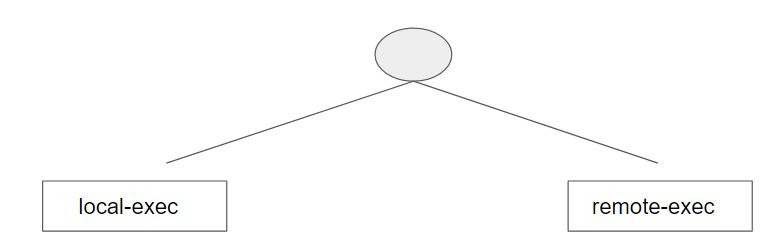
## 

## 

## **Module 2: Types of Provisioners**

Terraform has the capability to turn provisioners both at the time of resource creation as well as destruction.

There are two main types of provisioners:



2.1 Local Exec Provisioners

local-exec provisioners allow us to invoke a local executable after the resource is created.

One of the most used approaches of local-exec is to run ansible-playbooks on the created server after the resource is created.

Let’s take an example:

provisioner "local-exec" {

command = "echo ${aws\_instance.web.private\_ip} >> private\_ips.txt"

}

2.2 Remote Exec Provisioners

Remote-exec provisioners allow invoking scripts directly on the remote server.

Let’s take an example:

resource "aws\_instance" "web" {

# …

provisioner "remote-exec" {

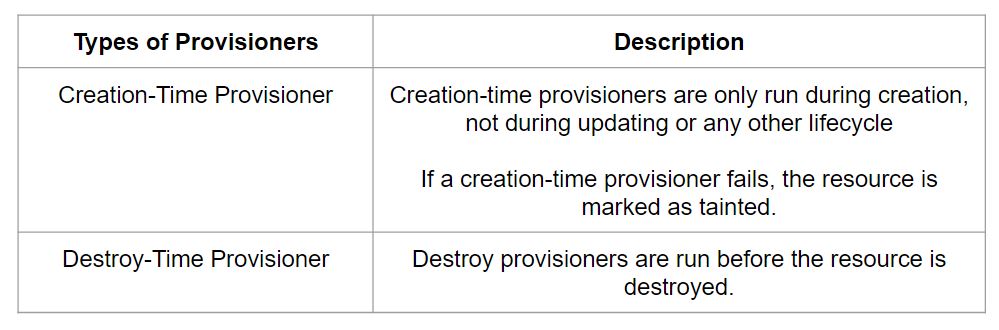
…………………………...

}

}

## **Module 3: Provisioner Types**

There are two primary types of provisioners:



If when = destroy is specified, the provisioner will run when the resource it is defined within is destroyed.



**Module 4: Failure Behavior**

By default, provisioners that fail will also cause the terraform apply itself to fail.

The on\_failure setting can be used to change this. The allowed values are:

