Terraform: Understand Terraform Basics II

➤ Section 3 : Understand Terraform Basics

- ➤ Describe how Terraform finds and fetches Providers.
- ➤ Explain when to use and not use Provisioners and when to use local-exec or remote-exec.

- **Provisioners :**
- Provisioners can be used to event specific actions in order to prepare servers for service.
- Passing data into virtual machines. Terraform have multiple Provisioners to pass data to public Cloud.
- user_data : AWS, Alibaba Cloud
- metadata : Google Cloud Platform
- custom_data : Microsoft Azure

- **local-exec :**
- local-exec provisioner is used to run the CLI for your target system in order to create, update, or interact with remote objects in that system.
- If you are trying to use a new feature of the remote system that isn't yet supported in its Terraform provider, **local-exec** might be the only option.

```
resource "aws_instance" “server" {
 # ...

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

- **local-exec :**
- Expressions in provisioner blocks cannot refer to their parent resource by name. Instead, they can use the special **self** object.
- All log output from the provisioner is automatically suppressed to prevent the sensitive values from being displayed.

- **remote-exec Provisioner :**
- **remote-exec** provisioner invokes a script on a remote resource after it is created.
- **inline** - This is a list of command strings. They are executed in the order they are provided.
- **script** - This is a path to a local script that will be copied to the remote resource and then executed.
- **scripts** - This is a list of paths to local scripts that will be copied to the remote resource and then executed. They are executed in the order they are provided.

- **remote-exec Provisioner :**
- How to execute Script with Arguments?
- User cannot pass any arguments to scripts using the **script** or **scripts** arguments to this provisioner. If you want to specify arguments, upload the script with the **file provisioner** and then use inline to call it.

```
resource "aws_instance" "server" {

 provisioner "file" {
  source      = “test_script.sh"
  destination = "/tmp/test_script.sh"
 }

 provisioner "remote-exec" {
  inline = [
    "chmod +x /tmp/test_script.sh",
    "/tmp/test_script.sh args",
  ]
 }
```

- **Creation-Time Provisioners :**
- By Default, provisioner run after the resource creation.
- Creation-time **provisioners** are only run during ***creation***, not during updating or any other lifecycle.
- Creation-time provisioner fails, the resource is marked as **tainted**. A tainted resource will be planned for destruction and recreation upon the next terraform **apply**.

- **Provisioners Failure Behaviour :**
- By default, provisioners that fail will also cause the Terraform apply itself to fail. The **on_failure** setting can be used to change this.
- **continue** - Ignore the error and continue with creation or destruction.
- **fail** - Raise an error and stop applying. If this is a creation provisioner, taint the resource.

```
resource "aws_instance" "server" {

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

Will see you in Next Lecture…

See you in next lecture …