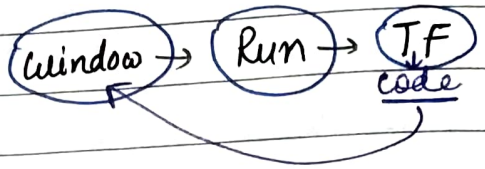
DAY-10

→ If you want Terraform to go to your base OS and run any program in your OS.
→ OS can be Windows, Linux, macos, etc.



→ If you want to provision any local command.
(Run command in local/remote system)
↓
"Command Provisioner" → run a command.
↓ local
local-exec
↓ remote
remote-exec

□ provisioner "local-exec" {
command = "echo hello"
}
↳ (fail)
↓
keyword/block → is a part of resource keyword

□ resource
↓
"null-resource"
→ predefined resources.
↓
This resource also requires a plugin.

↳ just run command without any resource
↓
Null Resource

```
resource "null_resource" "local1" {
  provisioner "local-exec" {
    command = "echo hello"
  }
}
```

or command = "chrome www.yahoo.com"

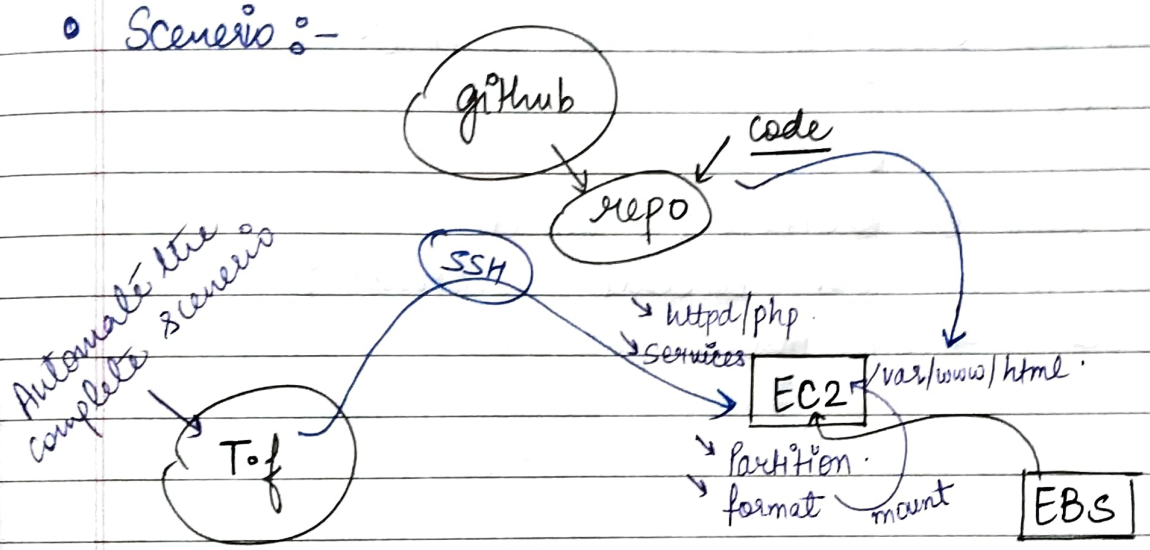
↓

Terraform Apply -auto-approve

↓

dosent ask for yes/no

- Scenario :-



→ Complete Terraform setup
→ end to end automation.

- Github - repository (new) → name
  → Initialize with readme

↓

index.php

↓

Clone / download ← url where you have code.

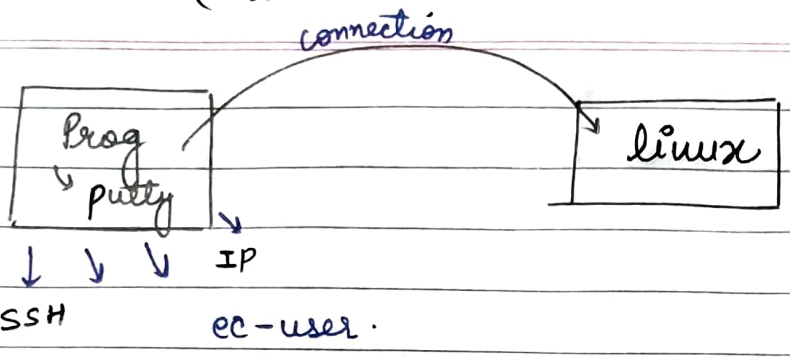
- launch EC2
  ↓
  Terraform AWS - EC2
  ↓
  write terraform code.
  ↓
  go to the instance → download softwares and github files.
  ↓
  I will tell Terraform code to SSH to the EC2
  ↓
  After this OS launches, I want terraform go an install softwares in EC2. → yum install php (manual)

  use provisioner → remote-exec

  - provisioner "remote-exec" {
      inline = [
        "sudo yum install httpd php git -y", (dosent require yes/no.)
        "sudo systemctl restart httpd",
        "sudo systemctl enable httpd",
        "sudo git clone (url of git files) /var/www/html/" (write to this folder)
      ]
    }

⇒ echo hi > Hello.txt

saves to file (info) ↑ → save this ↑



classmate Date

Terraform
↳ Resource
↓
Keywords.
Datasource
↓
all variables.

- Terraform connection

Connection Keyword in Terraform.

```
connection {
  type = "ssh"
  user = "ec2-user"
  ~~password~~ / private_key = file( "C:\Users\Kunal Daga\Downloads\xyz.pem" )
  host = "aws_instance.web.public_ip"
}
```

backslash changed to forward slash.

provide ip. ↘

"print it with output function"

```
output "myos_ip" {
  value = aws_instance.web.public_ip.
}
```

▷ terraform validate - checks syntax

```
resource "null_resource" "nulllocal1" {
  provisioner "local-exec" {
    command = "echo ${aws_instance.web.public_ip} > publicip.txt"
  }
}
```

- After EC2 (Linux Instance) launches it may fail to install due to root power
  ↓
  sudo su - root
  or
  ⇒ (sudo yum install httpd)
  ↑ root power

↓

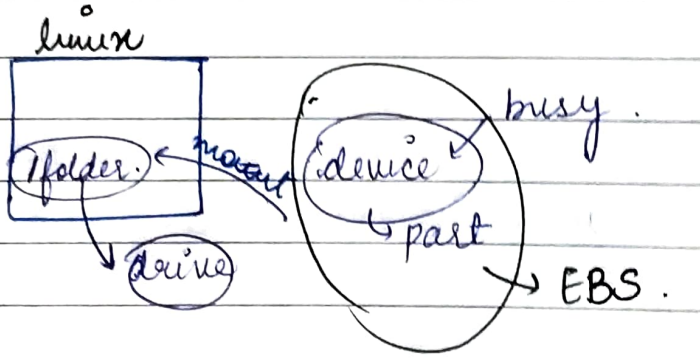
now creation of ebs using Terraform code

↓

[ebs volume attach to EC2]

↓

After you attach the volume, I would like to create a partition

- 

- for removing any volume you have to first detach it.

  ↓

  Delete the volume.

- If I say destroy (terraform destroy) →
  → disconnect hard disk → delete (behind scene)

→ If I mount this folder, then the command will fail

Hardisk creation → format → delete or destroy will fail.
→ partition.
→ and mount
(OS is using HDD)
↓
busy mode

- we have a option of force-detach for automation code (Terraform)
  - force_detach = true

- partition code terraform.

```
resource "null_resource" "nullremote3" {
  connection {
    type
    user
    private key
    host
  }
  provisioner "remote-exec" {
    inline = [
      "sudo mkfs.ext4  /dev/xvdh",
      "sudo mount /dev/xvdh /var/www/html",
      "sudo git clone url folder /var/www/html/"
    ]
  }
```

- Whenever you use provisioners (keyword), resource (keyword) → EBS volume
  → null
  ↳ provisioner
  ↳ remote

→ They can run any resource first
→ Null resource (especially remote) they run it first.

- Tells null resource that you have to [depend_on] ebs attachment

```
resource "null_resource" "name" {
  depends_on = [
    aws_volume_attachment.ebs_att
  ]
```