Linux + Terraform Cloud Administrator

Full Step-by-Step Training Guide (Beginner → Job-Ready)

Audience: Linux beginners to junior cloud engineers
Outcome: You can deploy, secure, operate, automate, and destroy Linux cloud infrastructure using Terraform (AWS or Azure)

How to Use This Guide (IMPORTANT)

- Follow **in order** – do not skip days
- Spend **1.5–2 hours per day**
- Every section includes:
- What you are learning
- Exact commands to run
- What success looks like
- Always follow this cycle:

```
Write Terraform → terraform plan → terraform apply → SSH → verify → terraform destroy
```

PART 0 – Preparation (Day 0)

0.1 What You Need

Local Machine

- Windows 11 / macOS / Linux
- Internet connection

Install Tools

Terraform

```
https://developer.hashicorp.com/terraform/downloads
```

Verify:

```
terraform version
```

Git

```
git --version
```

SSH (Windows already included)

```
ssh -V
```

Cloud Account (Choose ONE)

Option A – AWS (Recommended) - AWS account - Create IAM user with: - EC2 - VPC - Security Groups

Option B – Azure - Azure subscription - Install Azure CLI - Login:

```
az login
```

PART 1 – Terraform Fundamentals (Days 1–3)

Day 1 – Terraform Basics

Goal

Understand Terraform workflow and commands

Steps

1. Create project folder

   ```
   mkdir terraform-linux-lab
   cd terraform-linux-lab
   ```

2. Create file `main.tf`

```
terraform {
  required_version = ">= 1.6.0"
}
```

3. Initialize Terraform

```
terraform init
```

4. Validate

```
terraform validate
```

Success Check

- No errors
- Terraform initialized successfully

Day 2 – Provider Configuration (AWS example)

Goal

Allow Terraform to talk to the cloud

Create `provider.tf`

```
provider "aws" {
  region = "ap-southeast-2"
}
```

Authenticate

```
aws configure
```

Initialize

```
terraform init
```

Success Check

- Provider downloads
- No authentication error

Day 3 – Create Your First Linux VM

Goal

Provision a real Linux server using code

Create `compute.tf`

```
resource "aws_instance" "linux" {
  ami           = "ami-0df4b2961410d4cff"
  instance_type = "t2.micro"

  tags = {
    Name = "terraform-linux"
  }
}
```

Deploy

```
terraform plan
terraform apply
```

Success Check

- EC2 visible in AWS Console
- Terraform reports instance created

PART 2 – Linux Administration Core (Days 4–7)

Day 4 – SSH Access

Goal

Access your Linux VM securely

Create SSH Key

```
ssh-keygen -t rsa -b 4096
```

Update EC2 to use key

Add to `aws_instance`:

```
key_name = "mykey"
```

Connect

```
ssh -i mykey.pem ubuntu@PUBLIC_IP
```

Linux Commands

```
uname -a
uptime
ls /
```

Success Check

- Logged into Linux shell

Day 5 – Linux Users & Permissions

Goal

Operate Linux like a system administrator

Commands

```
sudo adduser adminuser
sudo usermod -aG sudo adminuser
su - adminuser
```

Permissions

```
ls -l
chmod 644 file.txt
chown adminuser file.txt
```

Day 6 – Services & Logs

Goal

Control system services

Commands

```
systemctl status ssh
journalctl -xe
```

Install Nginx

```
sudo apt update
sudo apt install nginx -y
sudo systemctl enable nginx
```

Verify

```
systemctl status nginx
```

Day 7 – Networking & Firewall

Goal

Secure Linux networking

Firewall

```
sudo ufw allow OpenSSH
sudo ufw allow 80
```

```
sudo ufw enable
sudo ufw status
```

Test

- SSH still works
- Browser opens `http://PUBLIC_IP`

PART 3 – Terraform + Linux Automation (Days 8–11)

Day 8 – Variables

Goal

Make Terraform reusable

Create `variables.tf`

```
variable "instance_type" {
  default = "t2.micro"
}
```

Use variable:

```
instance_type = var.instance_type
```

Day 9 – Outputs

Goal

Display useful information

Create `outputs.tf`

```
output "public_ip" {
  value = aws_instance.linux.public_ip
}
```

Day 10 – Provisioning Linux Automatically

Goal

Install software automatically using Terraform

Create `scripts/setup.sh`

```
#!/bin/bash
apt update
apt install nginx -y
systemctl start nginx
```

Attach to Terraform

```
provisioner "remote-exec" {
  inline = ["bash setup.sh"]
}
```

Day 11 – Linux Hardening

Goal

Secure server like production

SSH Hardening

```
sudo nano /etc/ssh/sshd_config
PermitRootLogin no
```

```
sudo systemctl reload ssh
```

Fail2Ban

```
sudo apt install fail2ban -y
```

PART 4 – Real Cloud Operations (Days 12–14)

Day 12 – Terraform Modules

Goal

Professional Terraform structure

```
modules/linux_vm/
  main.tf
  variables.tf
```

Day 13 – Environments (Dev/Test)

Goal

Multiple environments

```
terraform apply -var="env=dev"
terraform apply -var="env=test"
```

Day 14 – Failure Recovery + Documentation

Break Things

```
sudo systemctl stop nginx
```

Fix:

```
sudo systemctl start nginx
```

Destroy Everything

```
terraform destroy
```

Write README

Include: - Architecture - Security - Automation - Commands

FINAL OUTCOME

After this training you can:

✔️Deploy Linux using Terraform
✔️Secure Linux servers
✔️Automate installs & backups
✔️Manage cloud infrastructure safely
✔️Explain your work in interviews

NEXT STEPS (OPTIONAL)

- Add Docker
- Add S3 / Blob backup
- Add GitHub Actions CI
- Prepare RHCSA or Terraform Associate

If you want next: - AWS-only full Terraform code - Azure-only full Terraform code - PDF export - Notion checklist - Interview Q&A based on this lab

Just tell me what you want next ♀