
🧾 Terraform Provisioners – Handwritten Notes

🌍 What are Provisioners?

- Provisioners are used to **run scripts or commands** (locally or remotely) after resources are created.
- Useful for **post-creation tasks** like installing software, starting services, or logging info.

💻 Types of Provisioners

1. Remote-Exec

Runs commands **on the remote resource** (e.g., EC2 instance).

✅ **Example:**

```
None
resource "aws_instance" "webserver" {
  ami           = "ami-0edad43b6fa892279"
  instance_type = "t2.micro"

  provisioner "remote-exec" {
    inline = [
      "sudo apt update",
      "sudo apt install nginx -y",
      "sudo systemctl enable nginx",
      "sudo systemctl start nginx"
    ]
  }

  connection {
    type        = "ssh"
    host        = self.public_ip
```

```
    user        = "ubuntu"
    private_key = file("/root/.ssh/web")
  }
}
```

📝 **Note:**

- Requires **network connectivity** (SSH for Linux, WinRM for Windows).
- Use **connection block** to define how Terraform connects to the instance.

---

2. Local-Exec

Runs commands **on your local machine** (where Terraform runs).

✅ **Example:**

```
None
resource "aws_instance" "webserver" {
  ami           = "ami-0edad43b6fa892279"
  instance_type = "t2.micro"

  provisioner "local-exec" {
    command = "echo ${self.public_ip} >> /tmp/ips.txt"
  }
}
```

💡 **Use case:** Save instance IPs, write logs, or trigger local scripts.

⚙️ Run Provisioners on Create & Destroy

```
None
provisioner "local-exec" {
```

```
  command = "echo Instance Created! > /tmp/instance_state.txt"
}

provisioner "local-exec" {
  when    = destroy
  command = "echo Instance Destroyed! > /tmp/instance_state.txt"
}
```

🧠 **When = destroy** → Runs before the resource is destroyed.

⚠️ Handling Provisioner Failures

Option	Description
`on_failure = "fail"`	Default – fails Terraform run if command fails.
`on_failure = "continue"`	Continue even if the command fails.

✅ **Example:**

```
None
provisioner "local-exec" {
  on_failure = "continue"
  command    = "echo Setup completed!"
}
```

🌟 Best Practices

- **Avoid overusing provisioners** — use them only when no better alternative exists.
- Prefer **native options** like `user_data` (for AWS) or **metadata startup scripts** (for GCP).

✅ **Example using user_data:**

None

```
user_data = <<-EOF
  #!/bin/bash
  sudo apt update
  sudo apt install nginx -y
  sudo systemctl enable nginx
  sudo systemctl start nginx
EOF
```

🎯 This is cleaner and runs automatically during instance boot.

🧩 Summary

Type	Runs On	Common Use	Example Command
`remote-exec`	Remote instance	Install/configure software	`apt install nginx`
`local-exec`	Local machine	Log or notify	`echo ${public_ip}`
`user_data`	Cloud-native	Bootstrap instance	`#!/bin/bash`

🧠 **Tip:** Use provisioners only when necessary — prefer declarative configuration!
