**TERRAFORM TASKS  
  
  
Name : Varsha Yadav  
Date: 18-07-2025  
batch 11**  
  
  
  
Question 1 : subsequent apply, destroy

Question 2 : explore these providers -> local, random, null  
  
**Question 1 : subsequent apply, destroy  
  
1. terraform apply (Subsequent runs)**

When you run:

terraform apply

again **after** an initial apply:

* Terraform reads your **.tfstate** file (which stores the current real-world infrastructure state).
* It **compares** the existing resources to your .tf config.
* If **nothing changed**, it shows:

No changes. Your infrastructure matches the configuration.

But if you **change your config** (e.g., change length = 12), it will plan updates:

resource "random\_string" "example" {

length = 12 # changed from 8

}

Terraform output:

~ length: 8 => 12

Then you type yes to apply the update.

**2. terraform destroy**bash

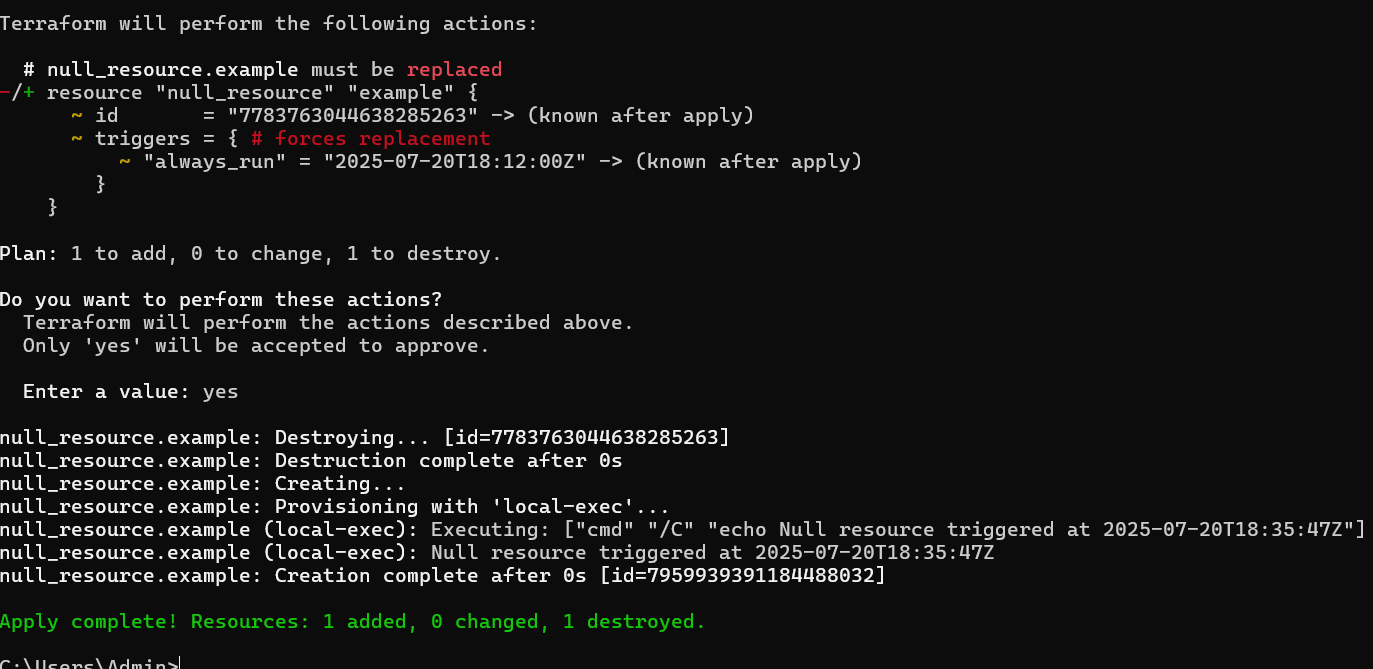
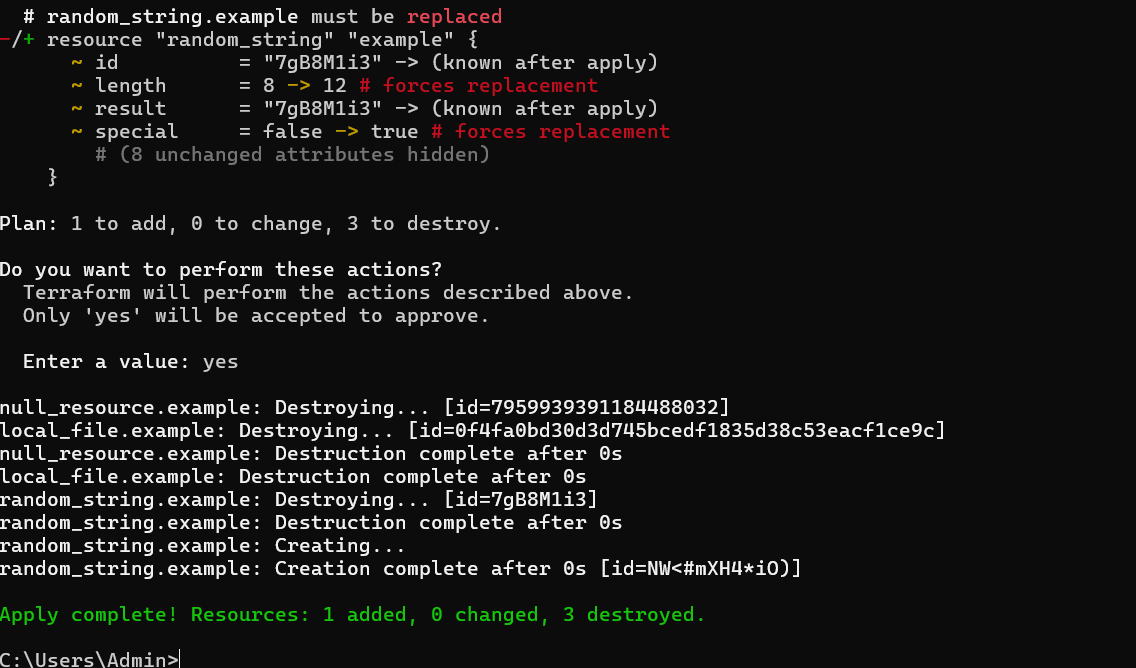
terraform destroy

This command:

* **Deletes all resources** tracked in terraform.tfstate
* Shows a plan to **remove** everything
* Asks for confirmation:

Do you really want to destroy all resources?

* After typing yes, it cleans up and shows:

Destroy complete! Resources: 3 destroyed.  
  
  
  
  
  
A screen shot of a computer

AI-generated content may be incorrect.  
 **Question 2 : explore these providers -> local, random, null  
  
1. local Provider**

**Purpose:**

Writes files **locally on your system**, not to the cloud.

**Use Cases:**

* Save output to .txt, .json, etc.
* Simulate output files for testing
* Create shell scripts from Terraform

**Example:**

resource "local\_file" "example" {

filename = "${path.module}/message.txt"

content = "Hello from Terraform!"

}

This will create a file message.txt in your project folder.

**2. random Provider**

**Purpose:**

Generates **random values** like strings, passwords, integers, UUIDs, etc.

**Use Cases:**

* Temporary credentials or passwords
* Unique resource names
* Random IDs during testing

**Example:**

resource "random\_string" "example" {

length = 12

special = false

}

Use ${random\_string.example.result} in other resources.

**3. null Provider**

**Purpose:**

Creates a "do-nothing" resource. Often used with **provisioners** or to force re-runs.

**Use Cases:**

* Run local scripts (via local-exec)
* Use triggers to force execution
* Wait or test lifecycle logic

**Example:**

resource "null\_resource" "example" {

triggers = {

always\_run = "${timestamp()}"

}

provisioner "local-exec" {

command = "echo Running null resource"

}

}

null\_resource runs even though it doesn’t manage infrastructure.

**Summary Table**

| **Provider** | **What It Does** | **Common Resources** | **Use Case Example** |
| --- | --- | --- | --- |
| local | Writes local files | local\_file | Generate a script or config file |
| random | Creates random data | random\_string, uuid | Generate unique name or secret |
| null | Run logic/scripts | null\_resource | Run shell commands or triggers |