**Terraform -02**

**1) Watch the terraform-02 video.**

Topics in video-2

**Mutable vs Immutable Infrastructure**

**1. Mutable Infrastructure**

* **Definition**: Existing infrastructure is updated/modified to reach the new desired state.
* Example: You update a running EC2 instance to install a new version of software.
* **Problem**: Over time, changes may cause **configuration drift** (system differs from code definition).

**2. Immutable Infrastructure**

* **Definition**: Instead of modifying existing infrastructure, Terraform destroys the old resource and **creates a fresh one** with the new configuration.
* Example: Replace an EC2 instance with a new one that has the updated software.
* **Advantage**: Ensures consistency and avoids drift.
* **Trade-off**: May cause downtime if not carefully managed.

**Terraform Lifecycle Rules**

Terraform allows you to control how resources are created, destroyed, or modified with the lifecycle block.

**Variables in Terraform**

Variables make Terraform configurations **dynamic and reusable**.

**Basic Structure:**

variable "filename" {

default = "/root/pets.txt"

type = string

description = "Path to the pets file"

}

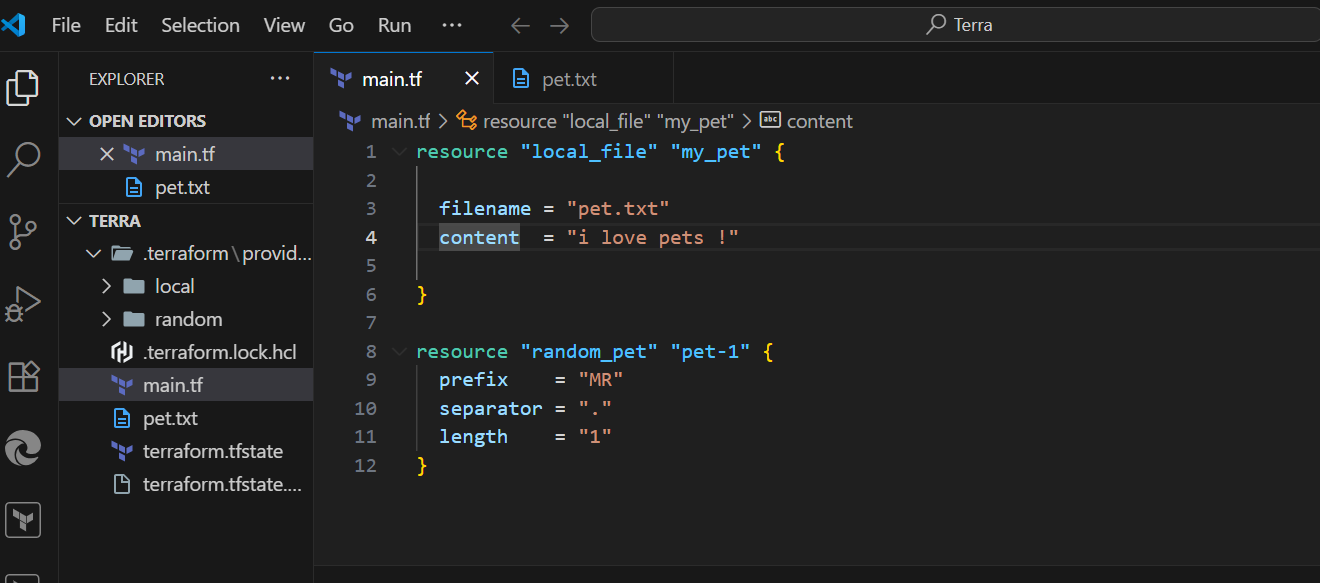
* **default** → Optional default value.
* **type** → Ensures variable type safety.
* **description** → Documentation for humans.

**Variable Types**

| **Type** | **Example** | **Use Case** |
| --- | --- | --- |
| **string** | "I love pets" | Text values |
| **number** | 1 | Numeric values (CPU, port numbers) |
| **bool** | true / false | Flags, toggles |
| **any** | Accepts any type | Flexible use |
| **list** | ["cat", "dog"] | Ordered collection of items |
| **map** | { pet1 = "cat", pet2 = "dog" } | Key-value pairs |
| **object** | { name = "dog", age = 5 } | Complex structured data |
| **tuple** | ["dog", 5, true] | Mixed data types in fixed order |

**2) Execute all the templates shown in video.**

**Main.tf -- main configuration file containing resource definition**

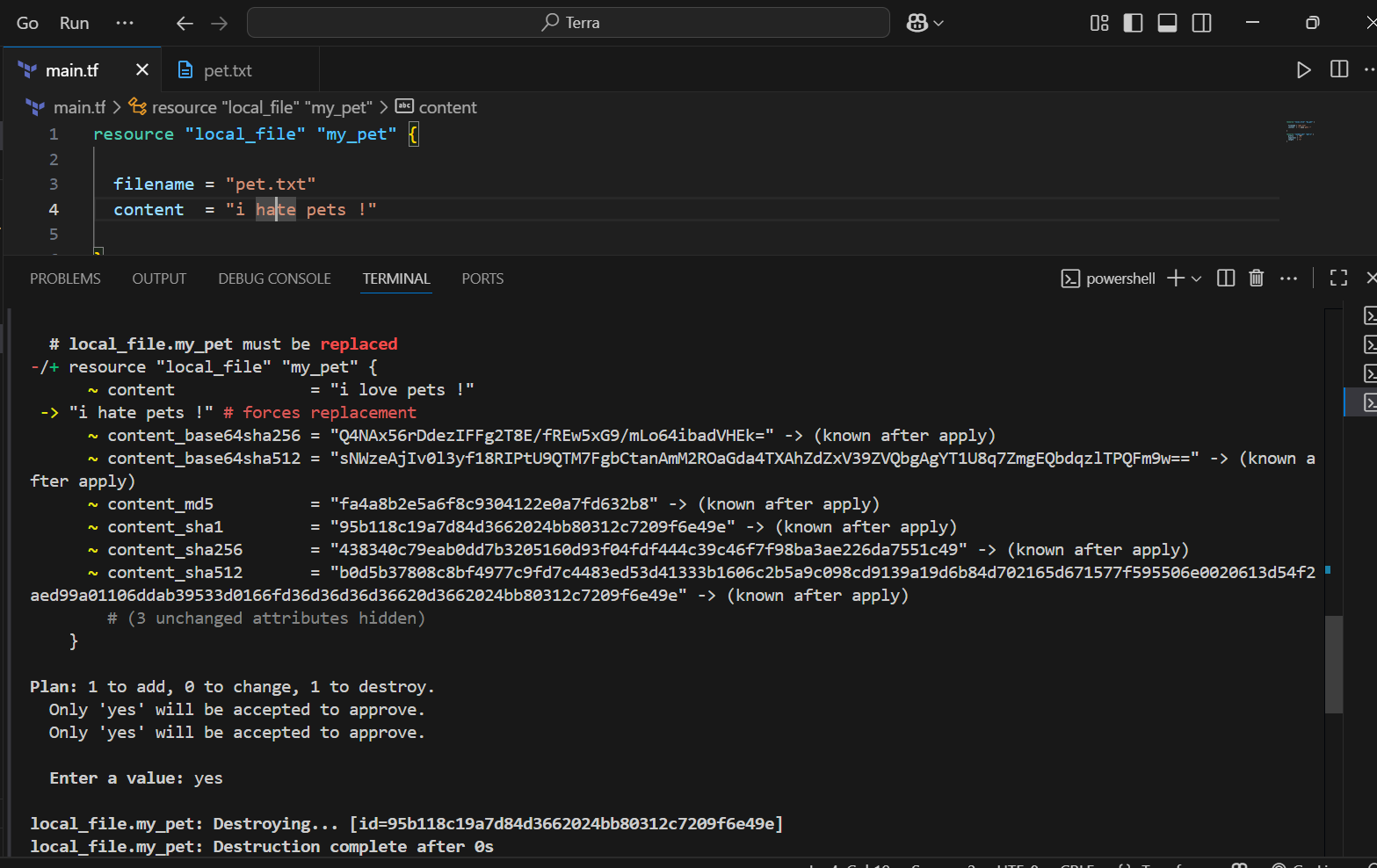
****

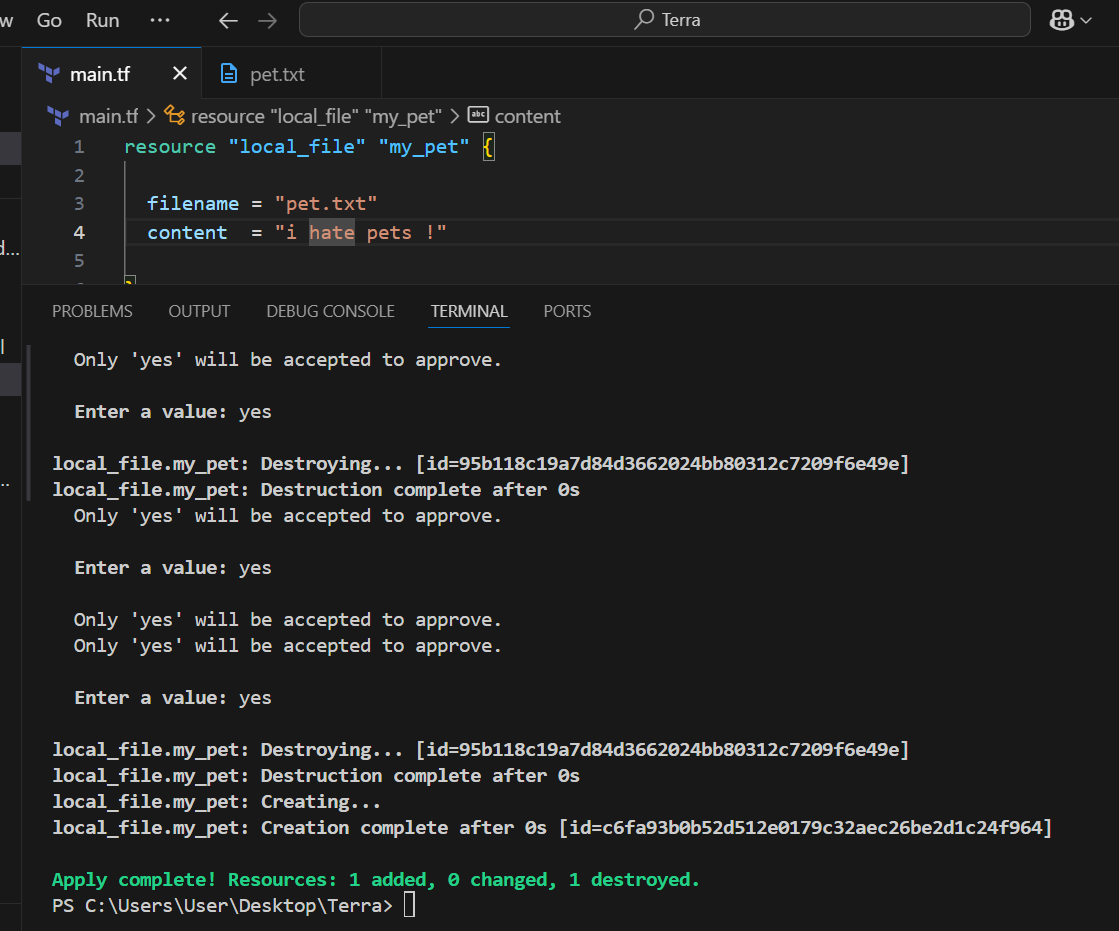
**Terraform mutable vs immutable infrastructure:**

**Terraform as a IAC tool uses immutable infrastructure strategy.**

**Immutable means deleting the older infra and creating a newer one with a new update.**

**Mutable means using the existing infra and updating the system with newer versions.**

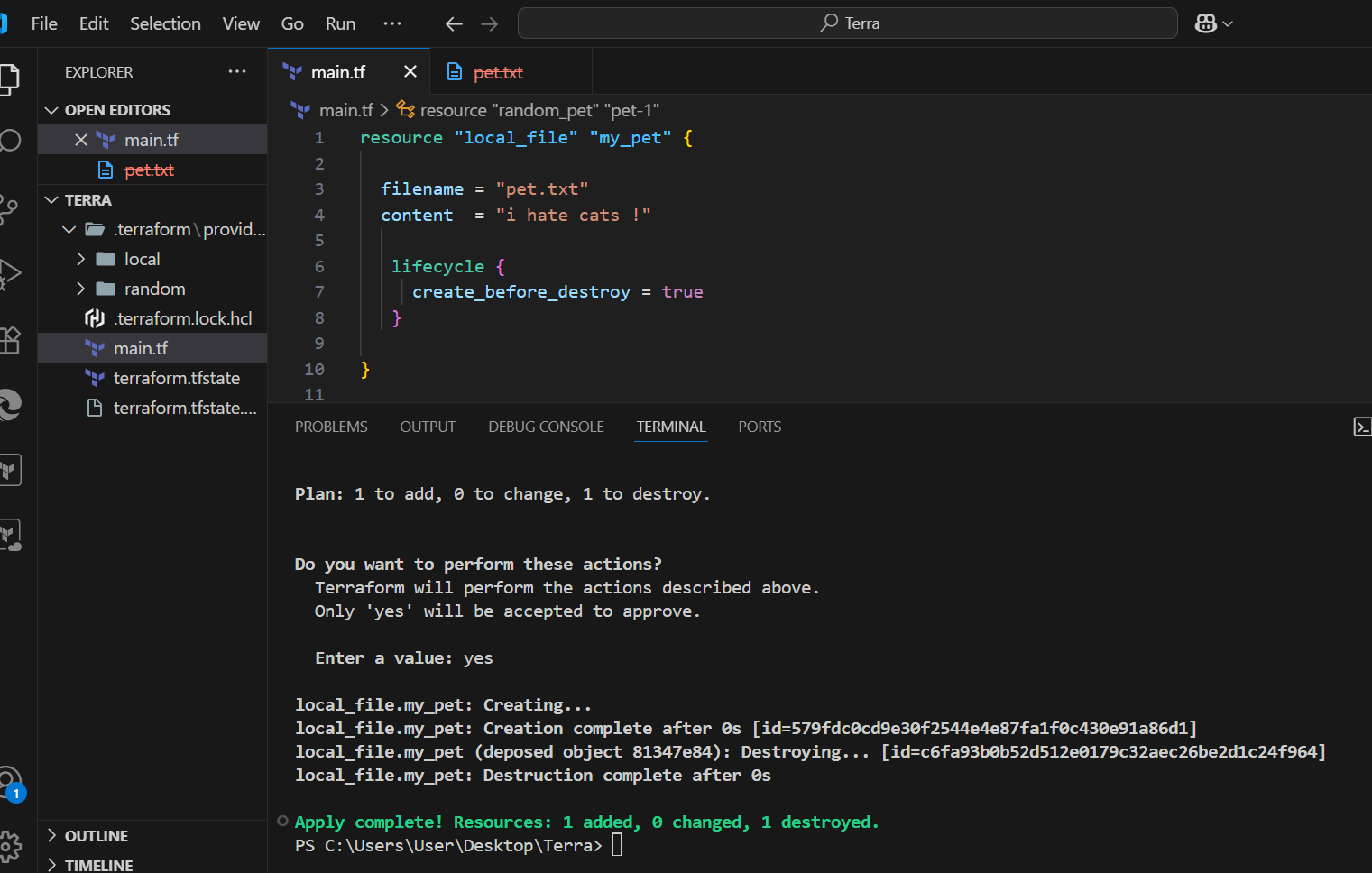
****

****

**Lifecycle**

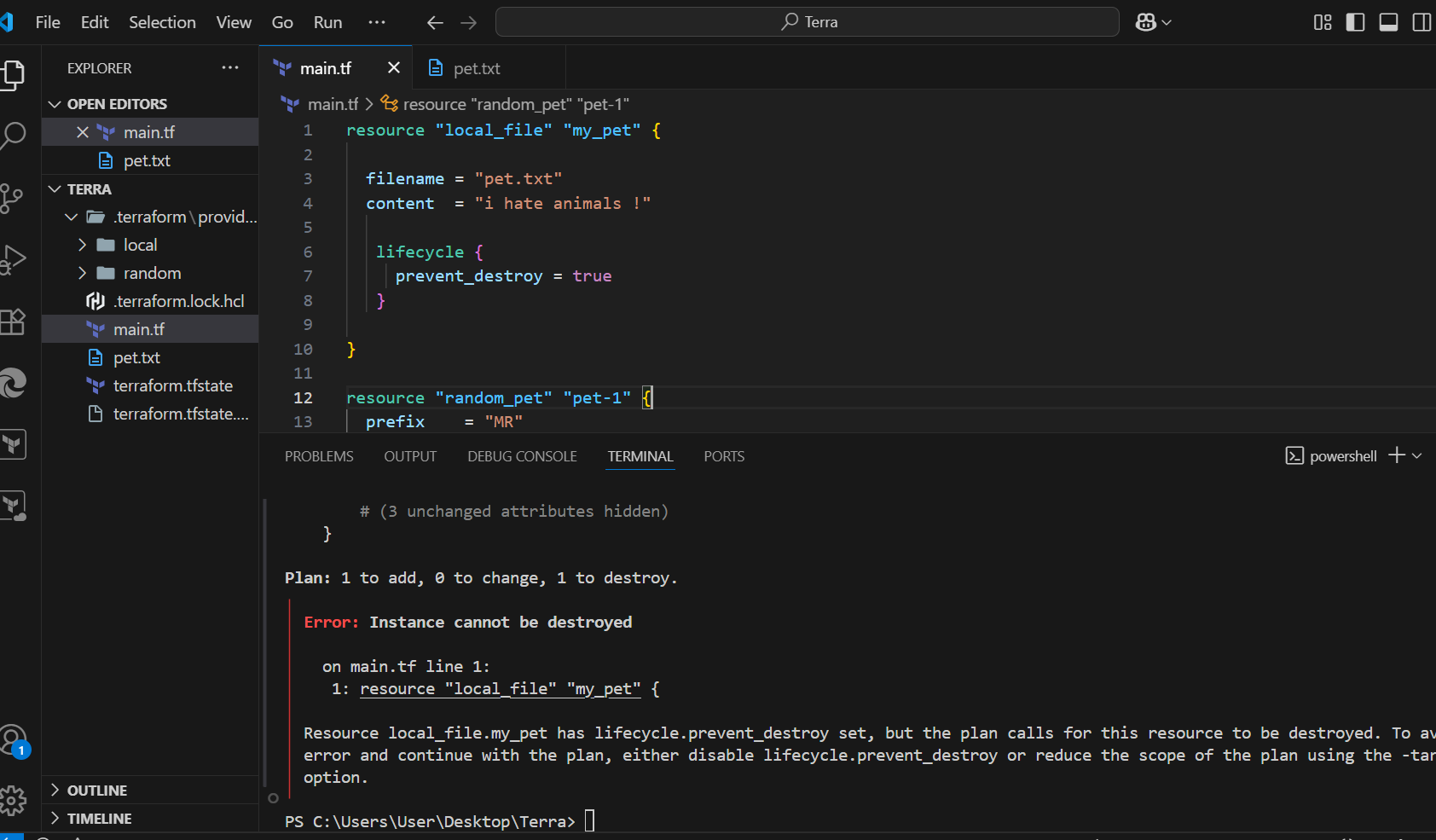
**1. Create\_before\_destroy**

* Ensures **new resource is created first** before destroying the old one.
* Prevents downtime when updating resources.
* Example use: Load balancers, production servers.

****

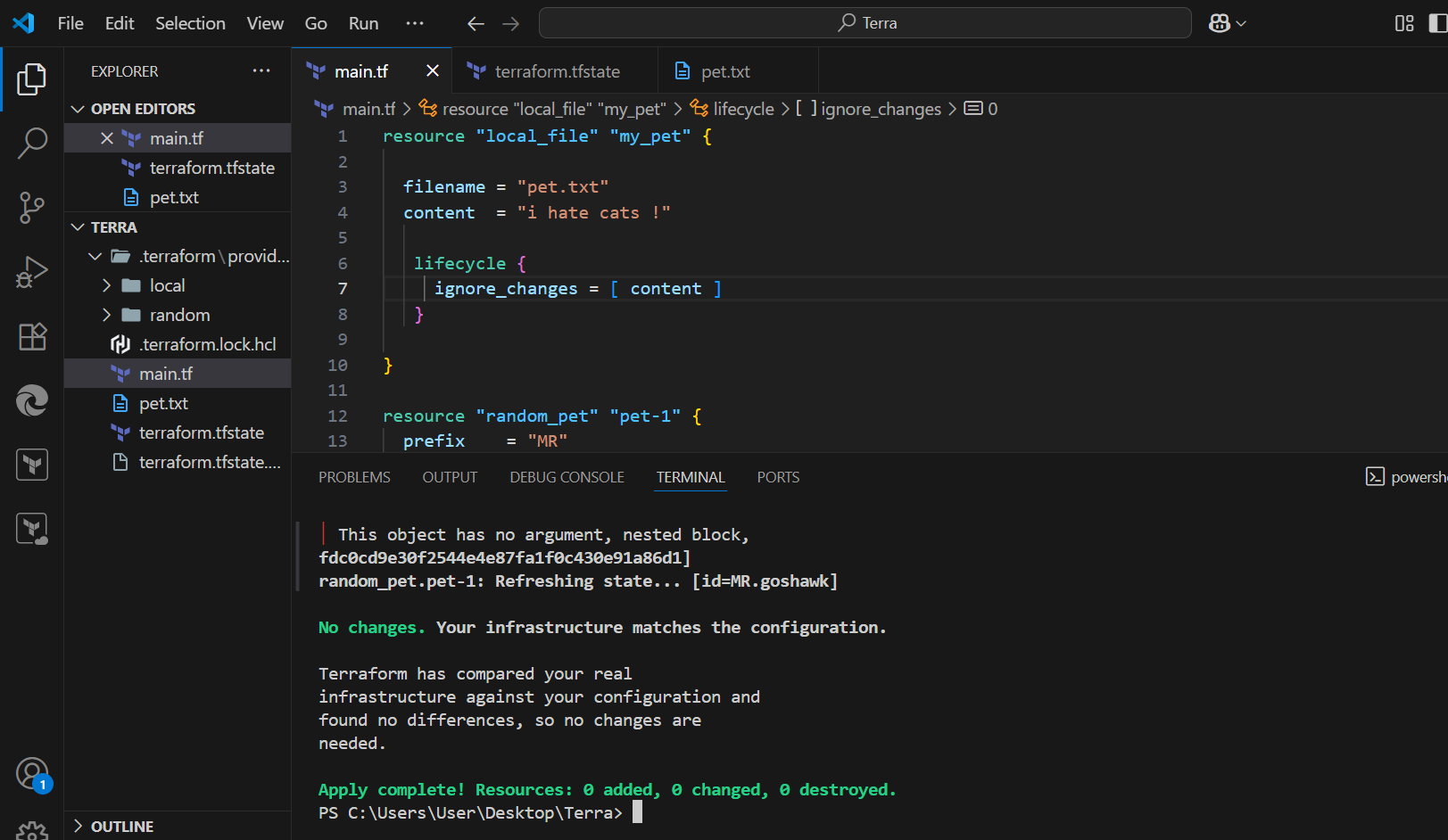
**2. Prevent\_destroy**

* Prevents accidental deletion of critical resources (e.g., DB, VPC).
* Terraform will throw an error if terraform destroy tries to delete it.

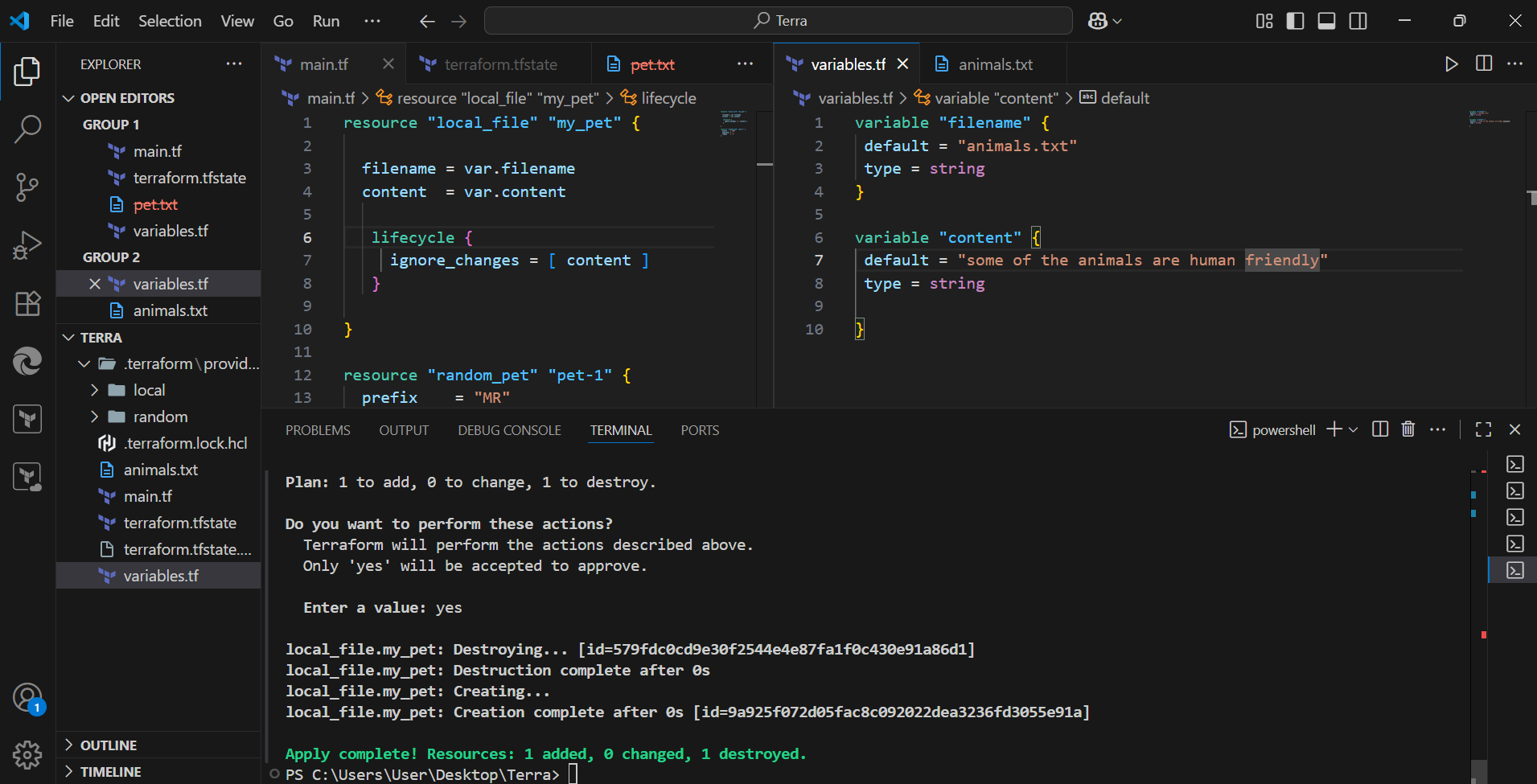


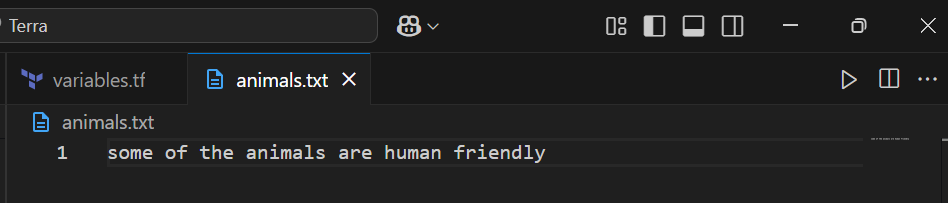
**3. Ignore\_changes**

* Ignores specific attributes from being managed by Terraform.
* Useful when certain values are changed **outside Terraform** (e.g., tags added by another tool).
* Example: Ignore tags or content.

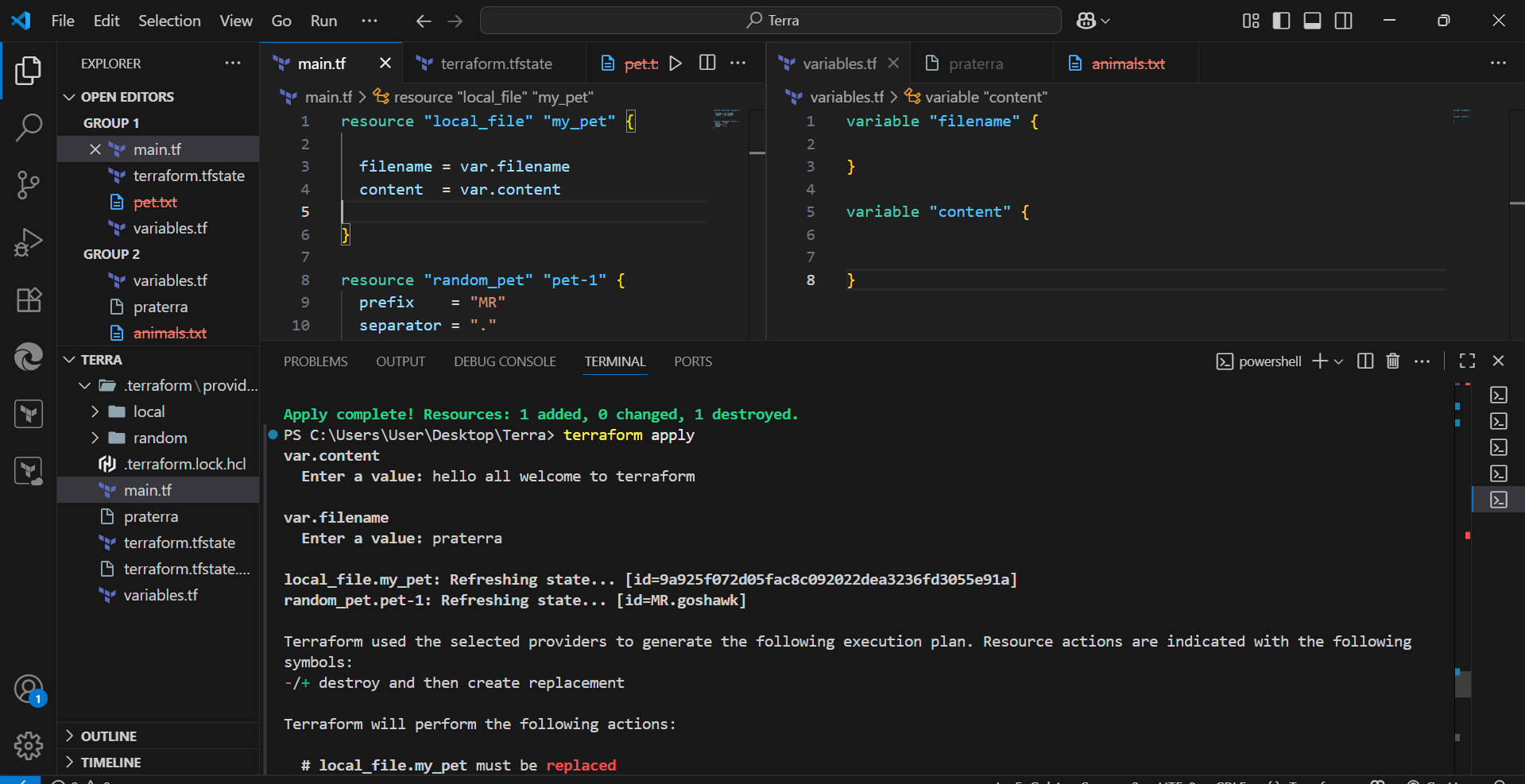
****

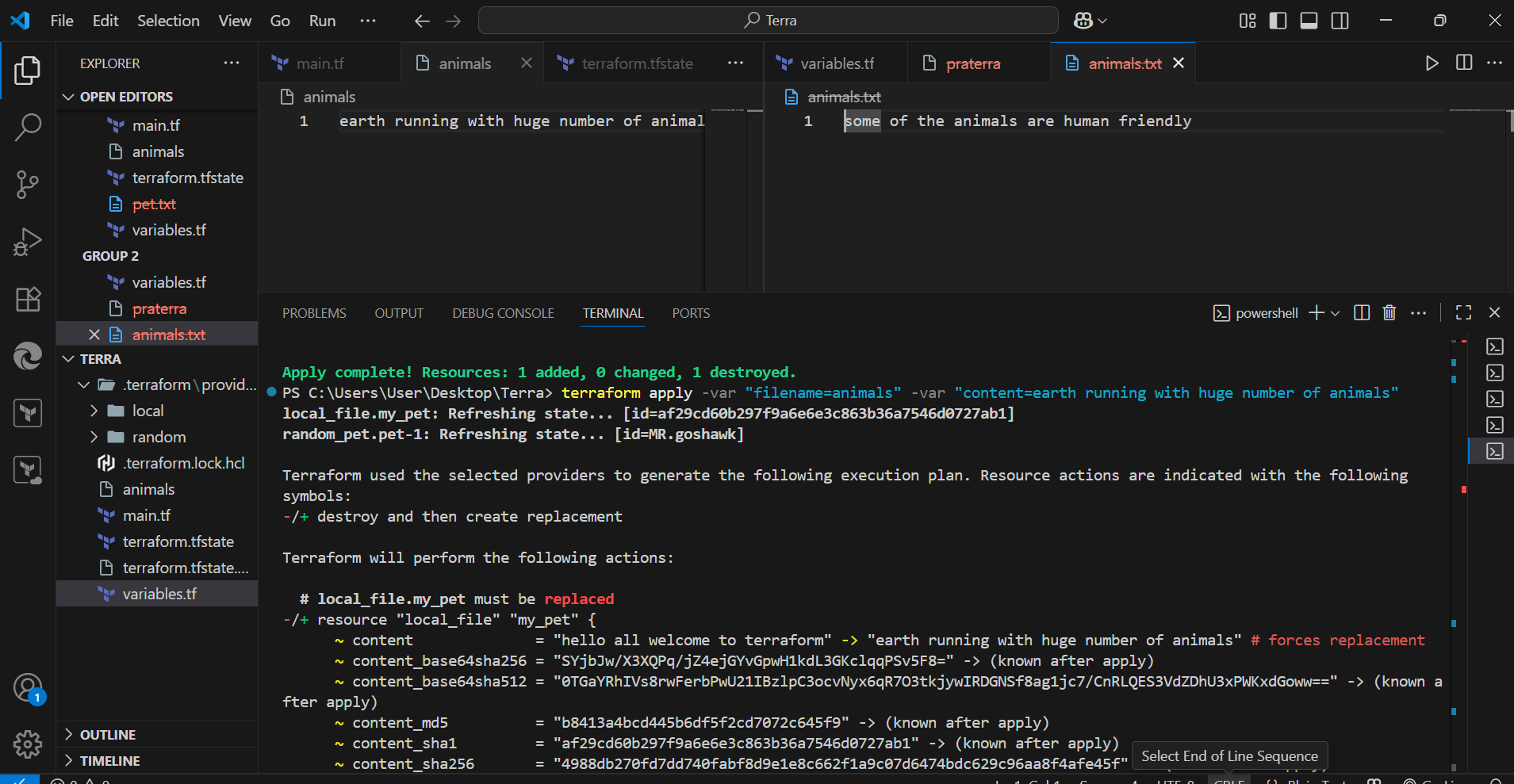
**Variables in Terraform**

****

****

**Passing values using command**

****

****

**3) Integrate terraform in Jenkins using Terraform plugin.**

**Step 1: Install Terraform on Jenkins Server**

On the Jenkins machine (EC2, VM, etc.):

# Install Terraform

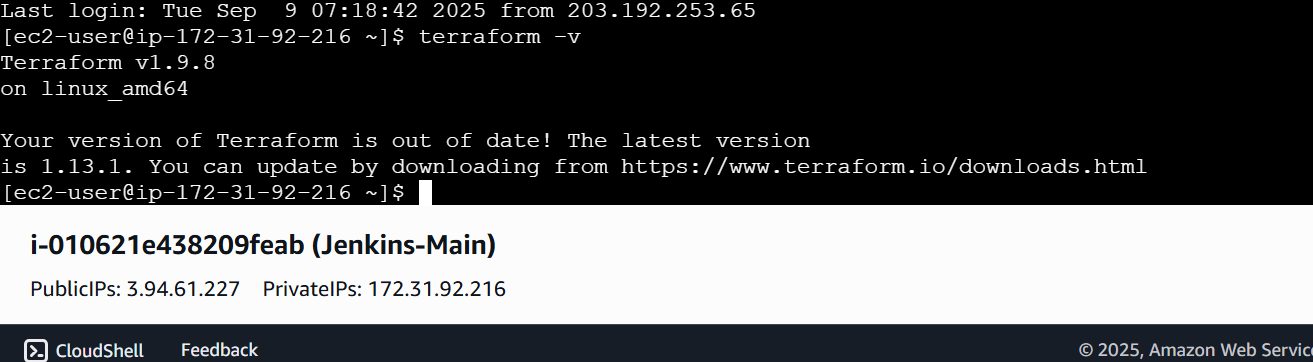
sudo yum install -y wget unzip # (Amazon Linux)

wget https://releases.hashicorp.com/terraform/1.8.5/terraform\_1.8.5\_linux\_amd64.zip

unzip terraform\_1.8.5\_linux\_amd64.zip

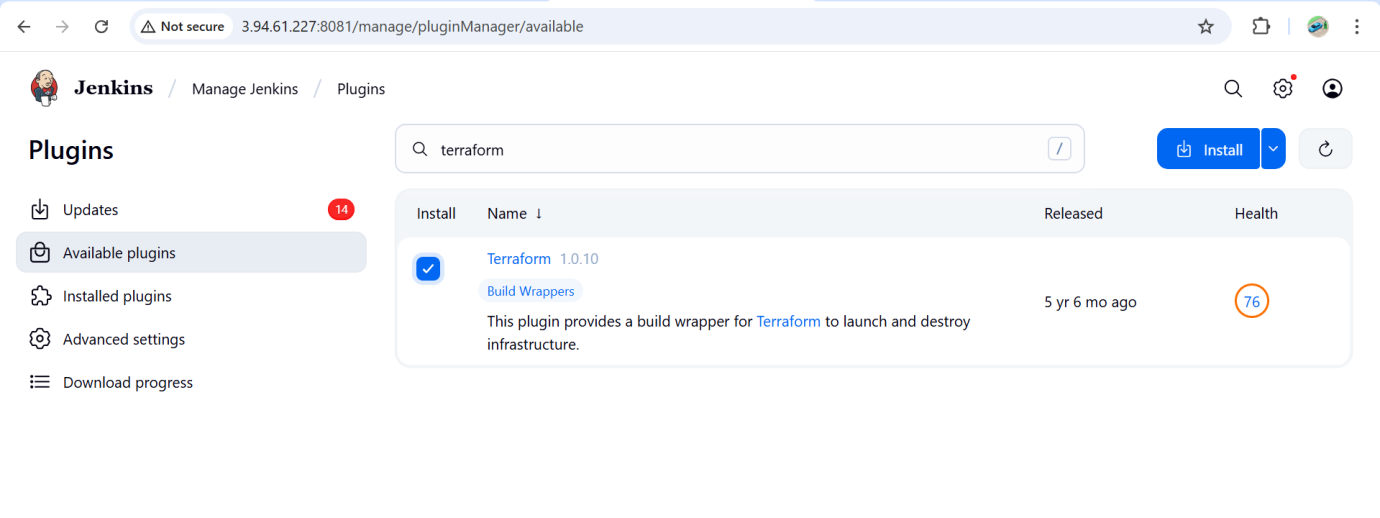
sudo mv terraform /usr/local/bin/

terraform -version

****

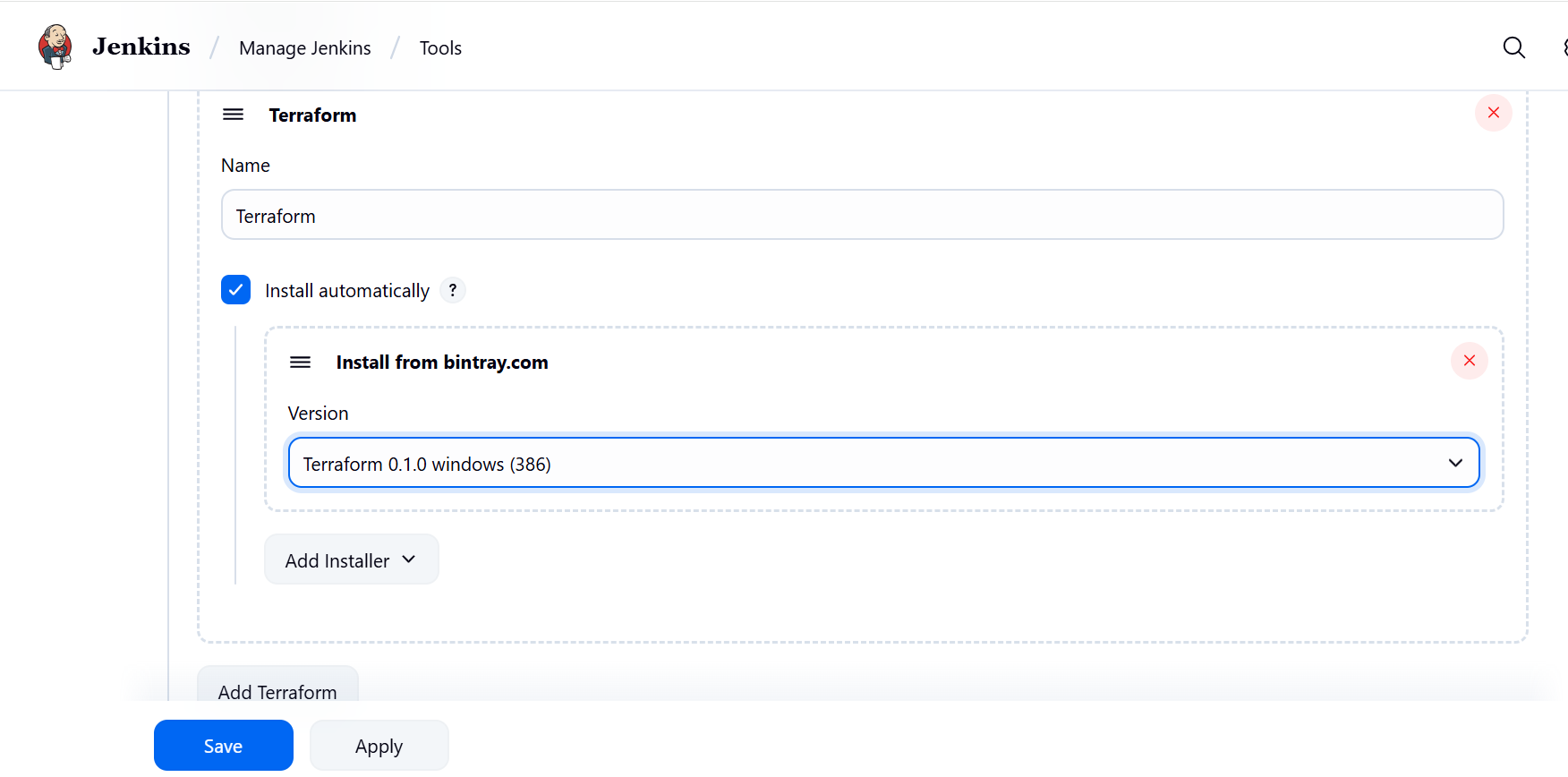
## Step 2: Install Terraform Plugin in Jenkins

1. Go to **Jenkins Dashboard → Manage Jenkins → Plugins → Available Plugins**.
2. Search for **“Terraform”** plugin.
3. Install it



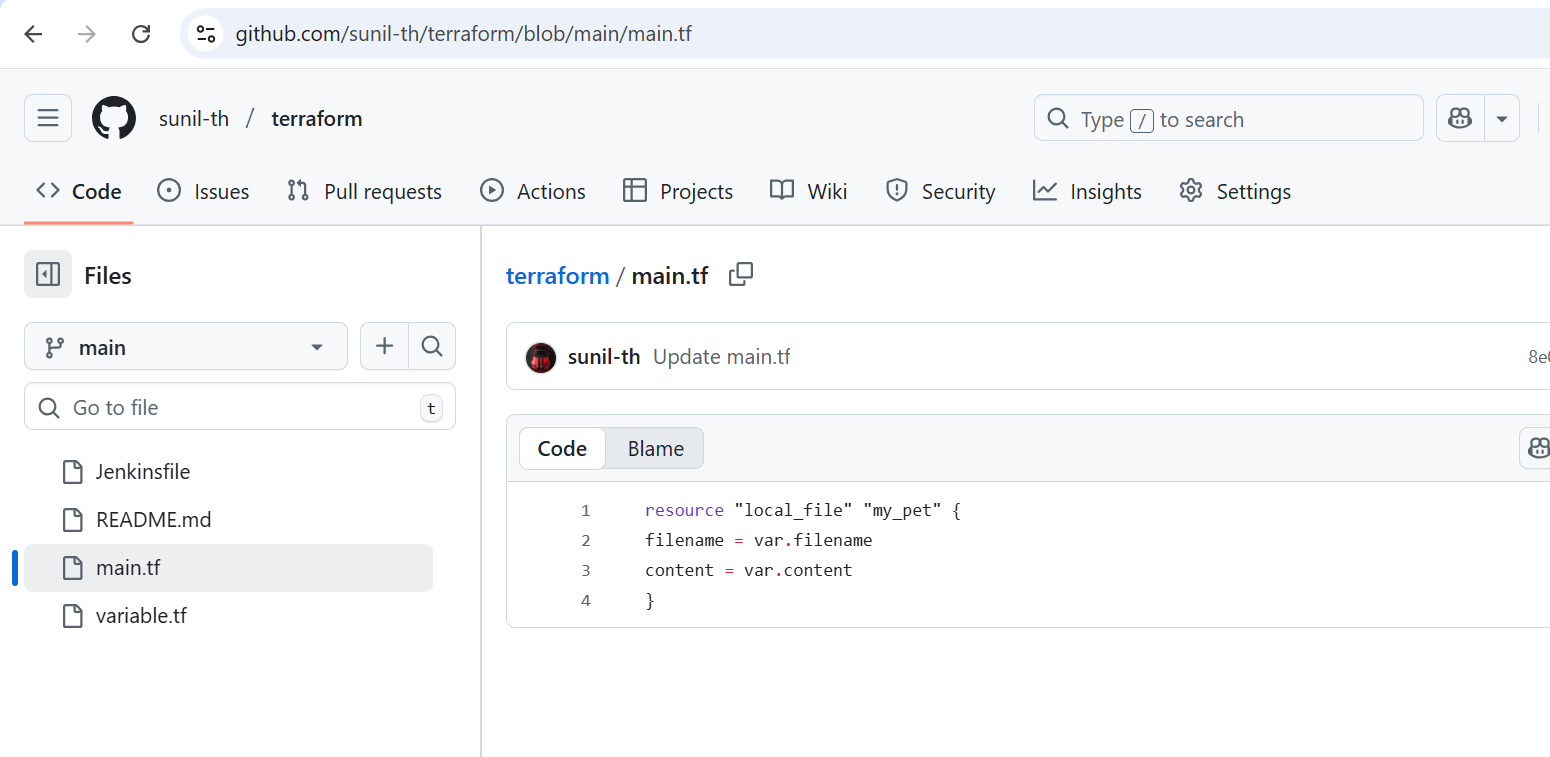
## Step 3: Configure Terraform in Jenkins

1. Go to **Manage Jenkins → Global Tool Configuration**.
2. Find **Terraform** section.
3. Add a Terraform installation → give it a **Name** (example: Terraform-1.8.5) and path if manually installed.
   * If plugin supports auto-install → let Jenkins download it.
   * If already installed → point Jenkins to /usr/local/bin/terraform.

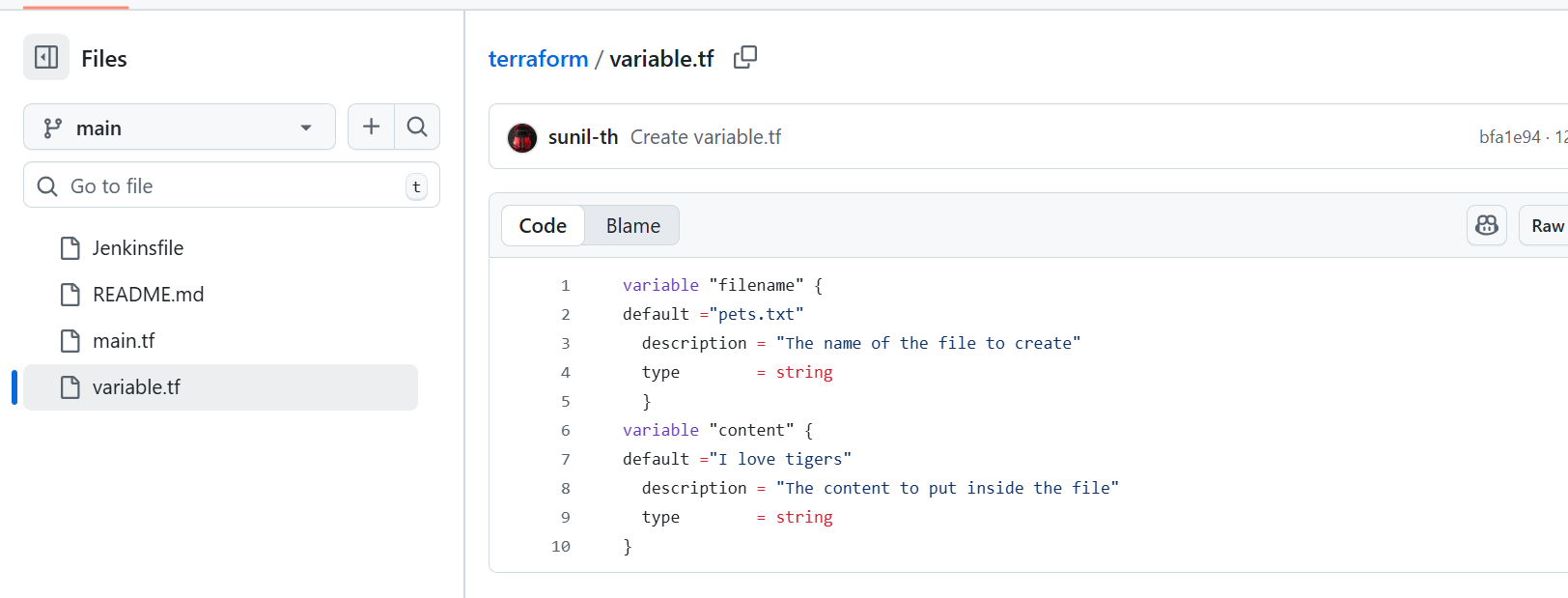
****

**In Git hub add these files**

**Main.tf**

****

**Variable.tf**

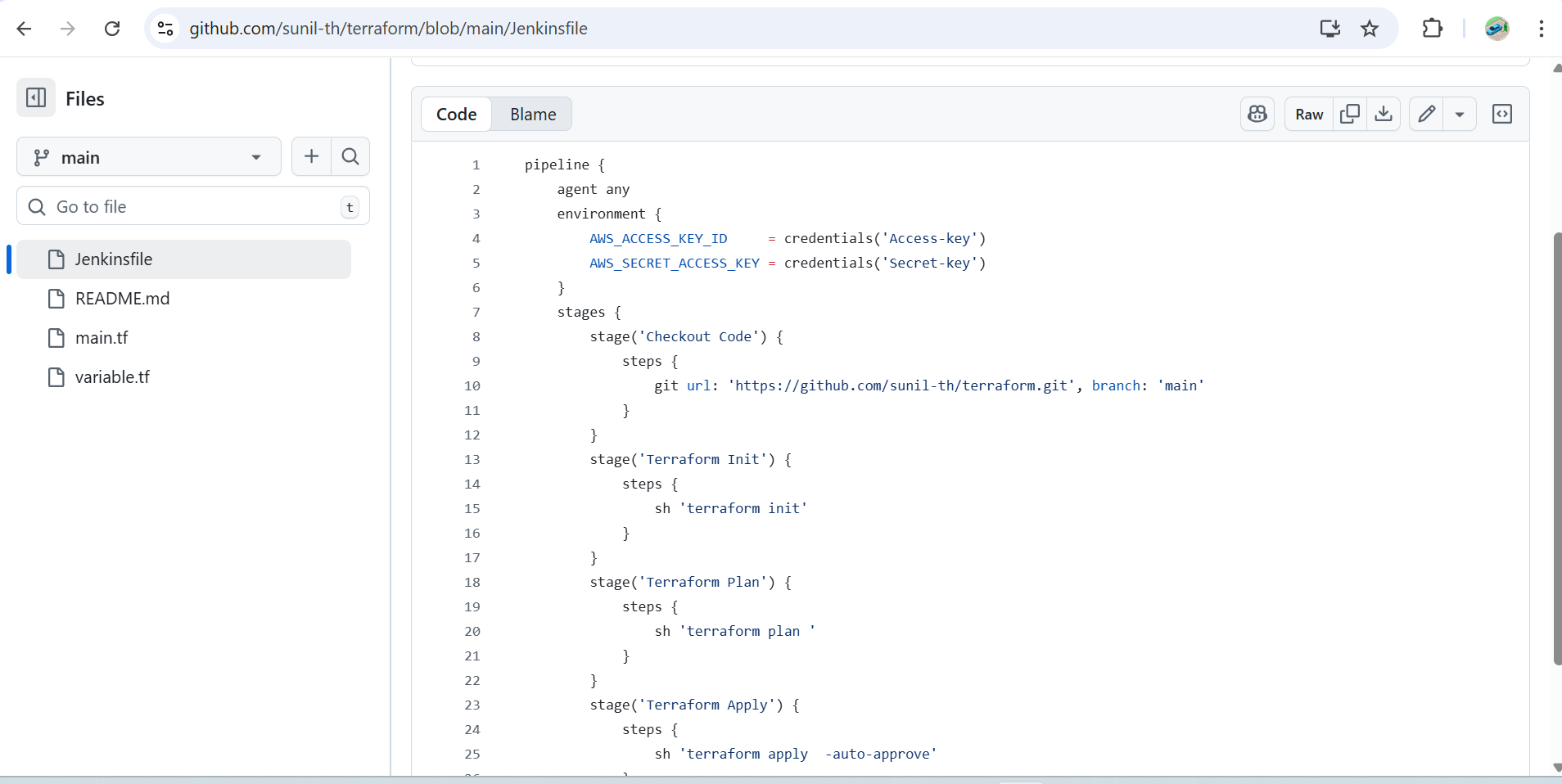
****

**Jenkinsfile**

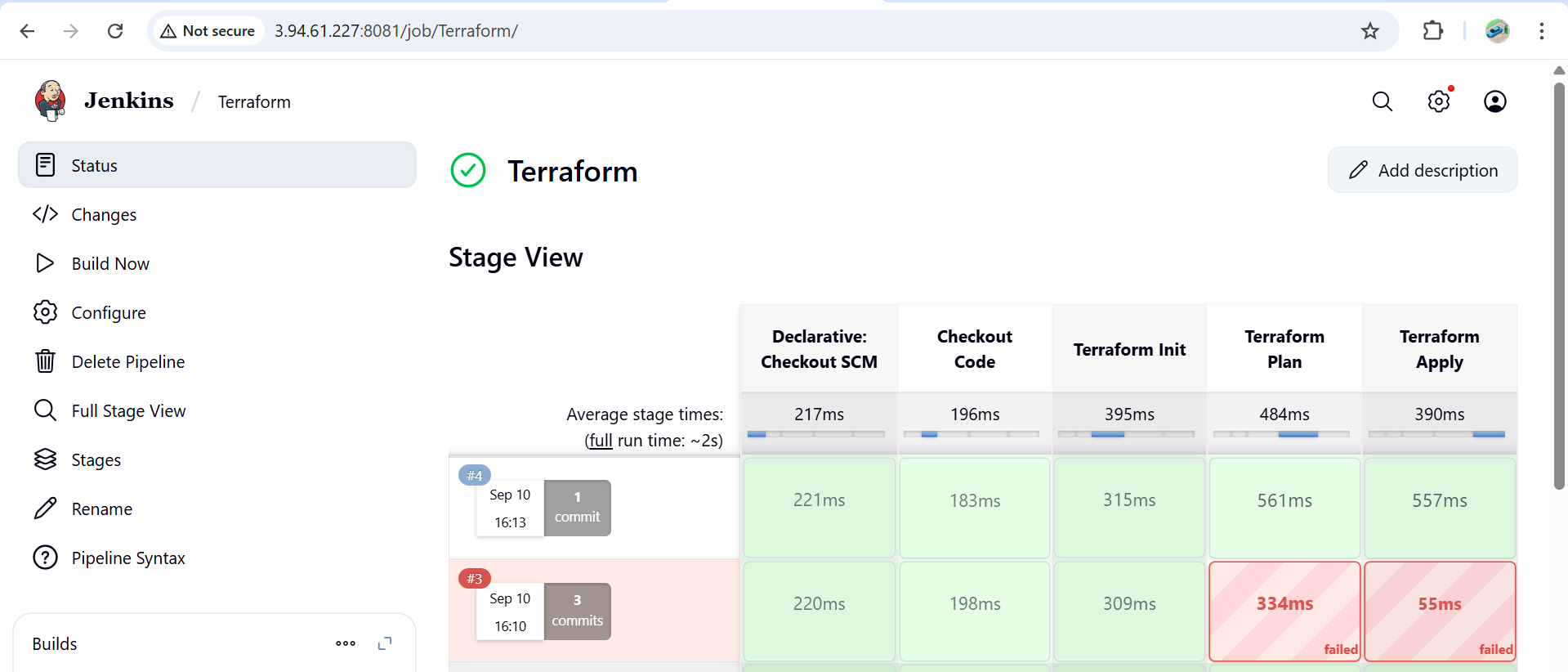
**In Jenkinsfile**

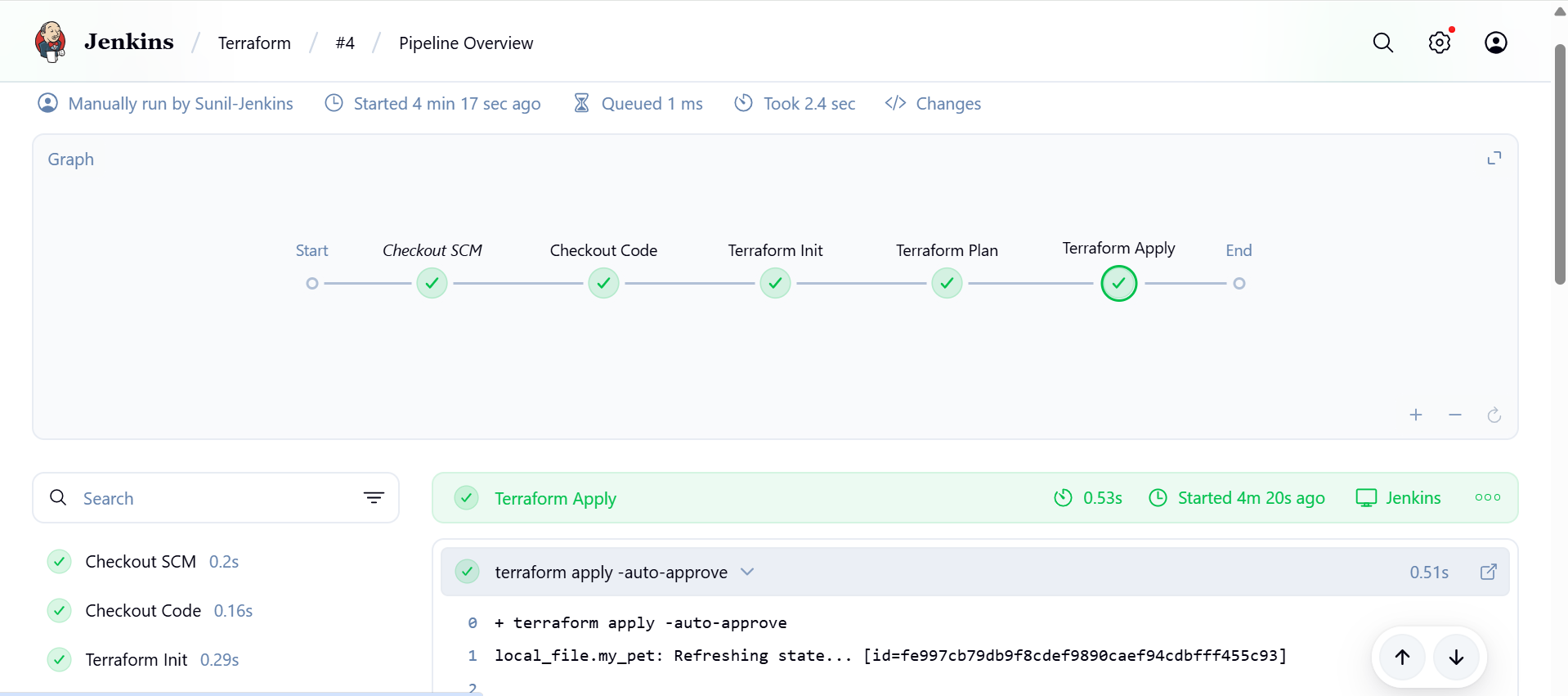
Give the aws credentials ids

And all the stages of the terraform

****

**Create a new job give the github url, save and buid**

****

****