**Terraform-04**

1. **Watch Terraform-04 video.**
2. **Execute the script shown in the video.ss**

Version Constraints:

===================

Changing in terraform providers version may get us into incompatability issues.

By default terraform will always try to downlaod the latest version of provider available from registry.

To make sure to use the specific version provider we can add the provider block in configuration.

Example:

=======

terraform {

required\_providers {

local = {

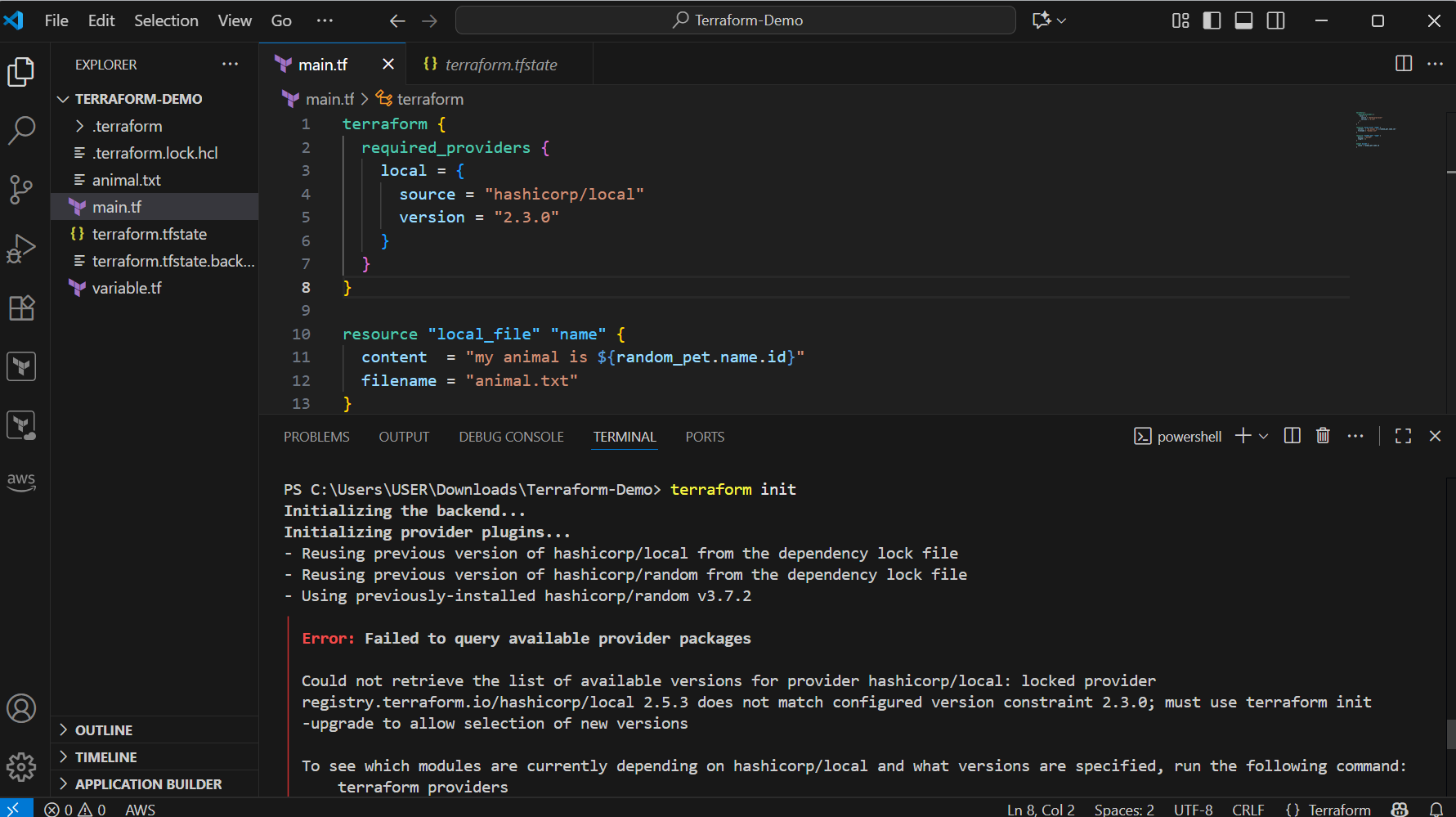
source = "hashicorp/local"

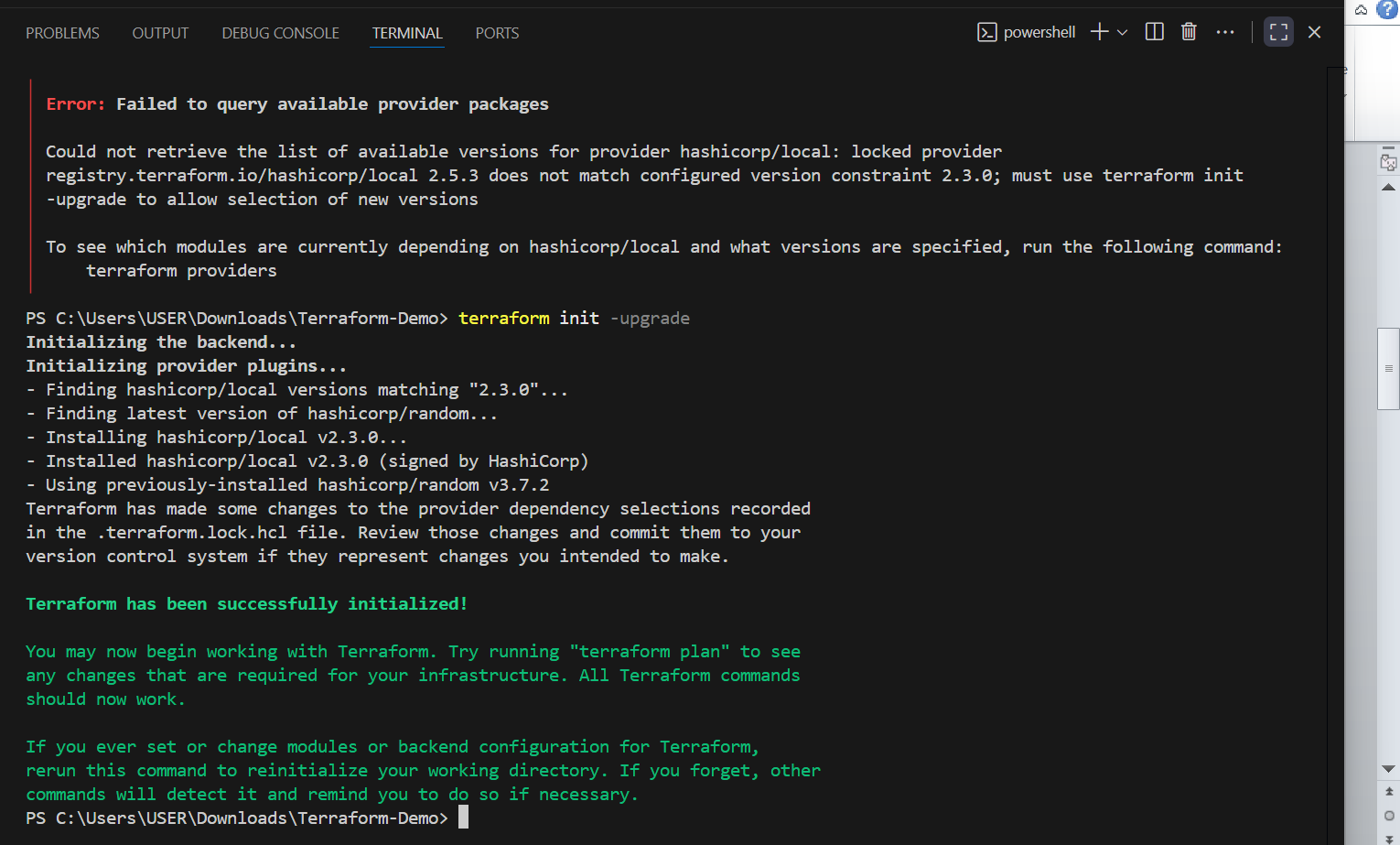
version = "2.3.0"

}

}

}

****

****

=========

version = "2.3.0" --> download the exact version

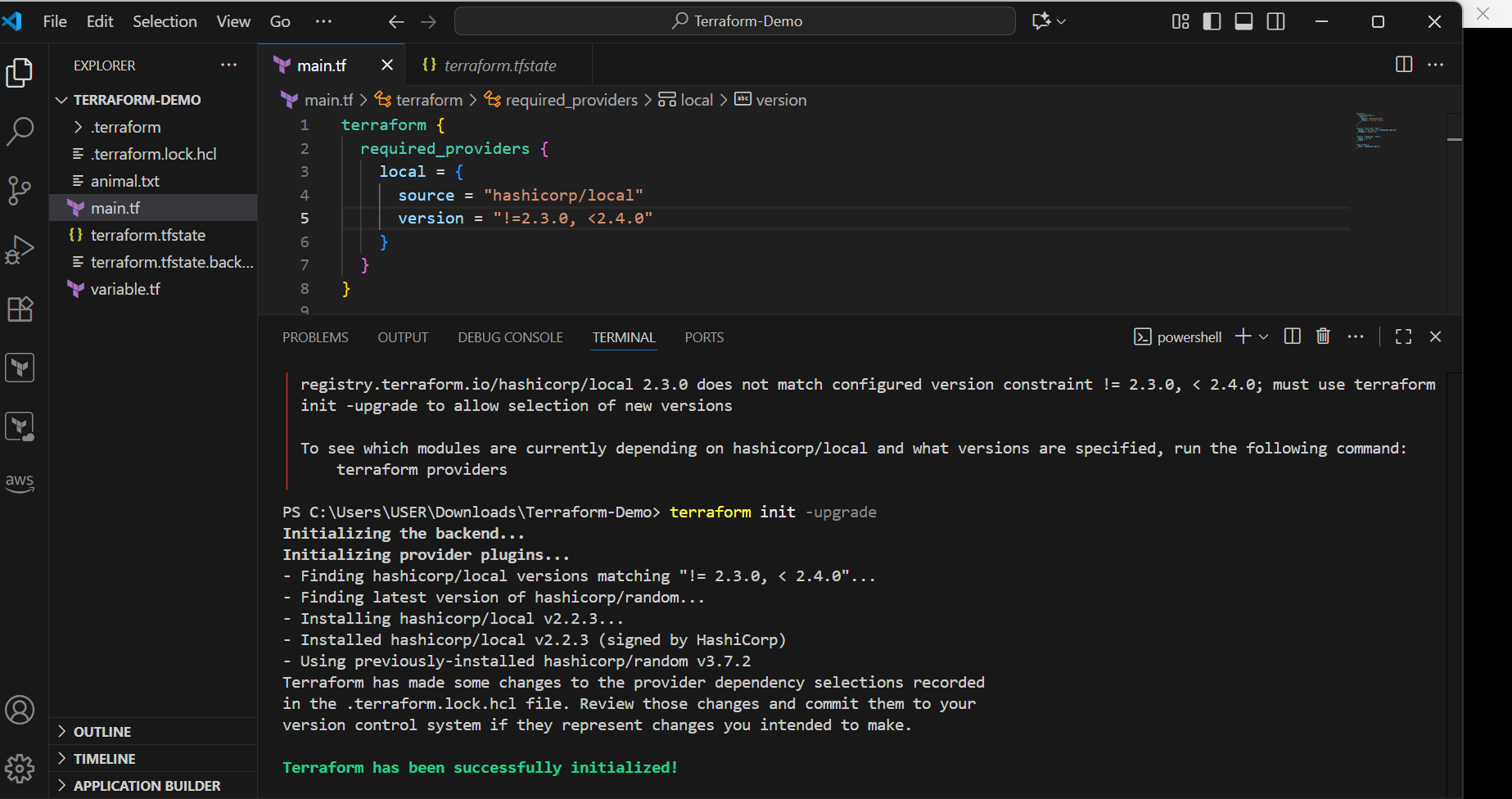
version = "!=2.3.0" --> will not use the mentioned version

version = "< 2.3.0" --> lesses than the mention version

version = "> 2.3.0" --> greater than the given version

version = "~> 2.3.0" --> specific version or higher version.

version = "!=2.3.0, <2.4.0"

****

Data sources:

===========

Data sources are used to read the content of the infrastructure

for example if we want terraform to read the content of the file which has been created by any other tool.

create a file called in dogs.txt in the same terrafrom working directory.

main.tf

======

resource "local\_file" "my-pet" {

filename = "pets.txt"

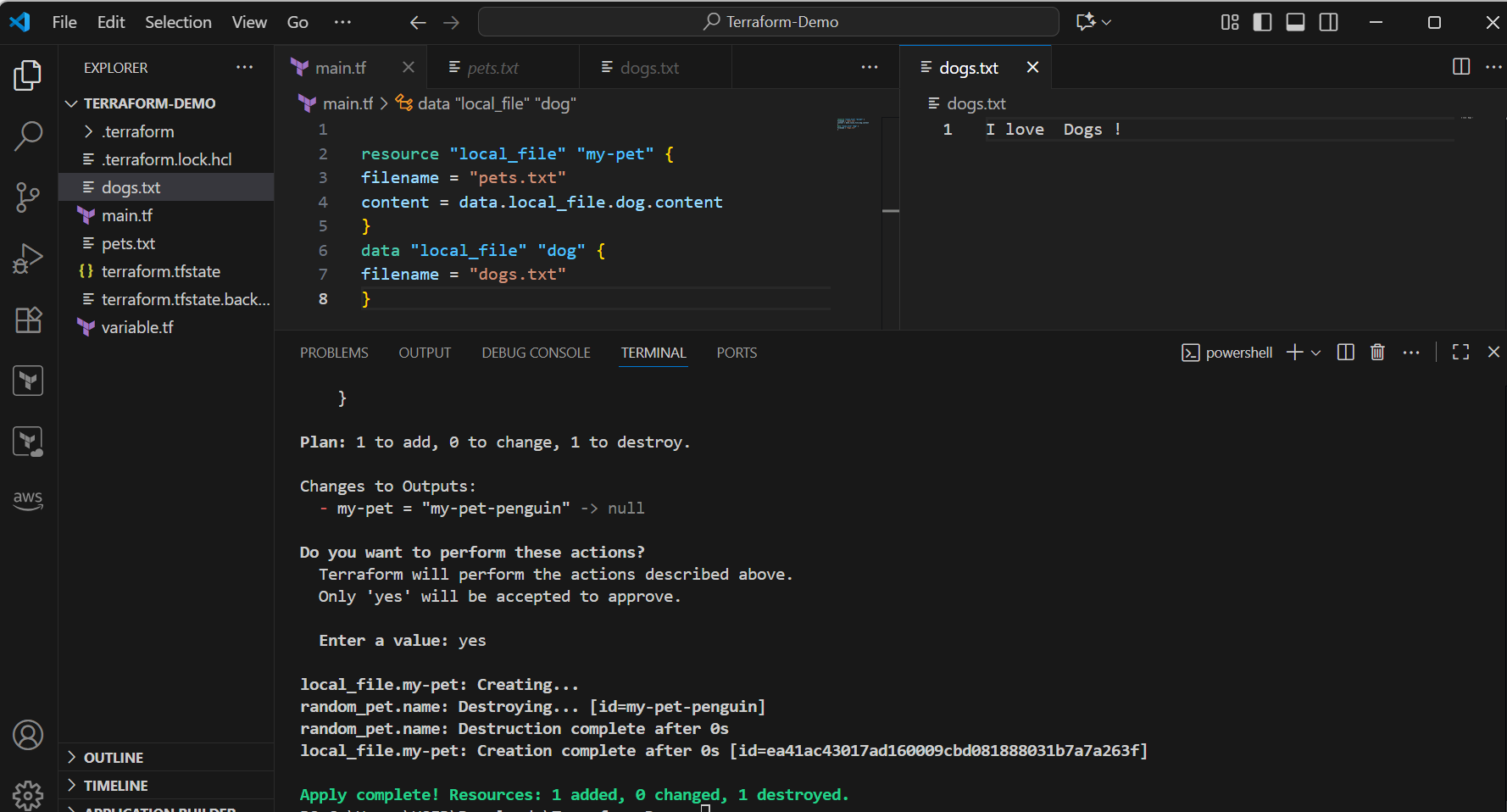
content = data.local\_file.dog.content

}

data "local\_file" "dog" {

filename = "dogs.txt"

}

****

Meta-Arguments:

==============

Meta arguments are used if we want to create multiple resources.

Meta arguments can be used within any resource block to change the behaviour of the resources.

Examples for meta arguments:

1) Depends\_on

2) lifecycle rules

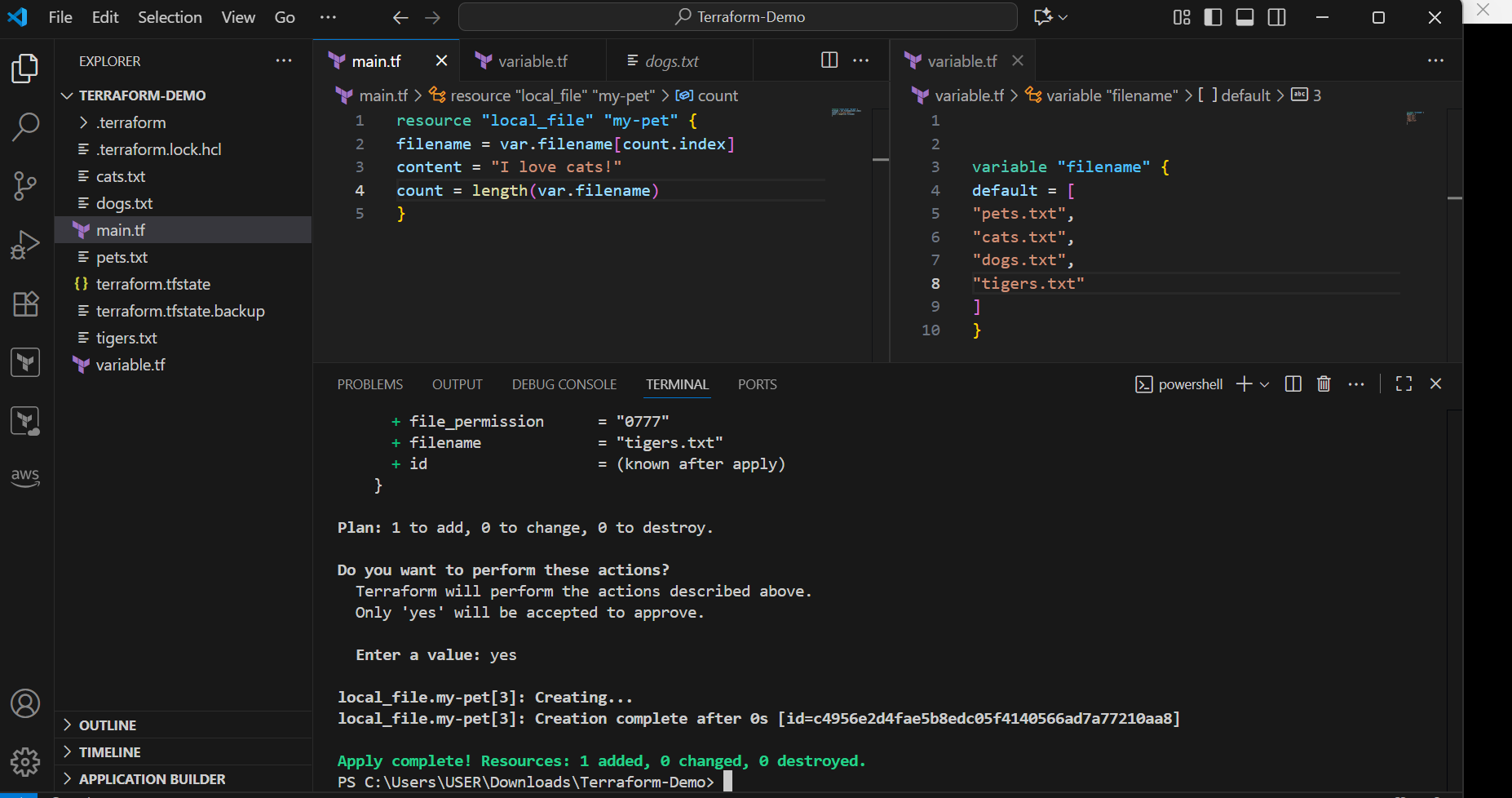
3) Count

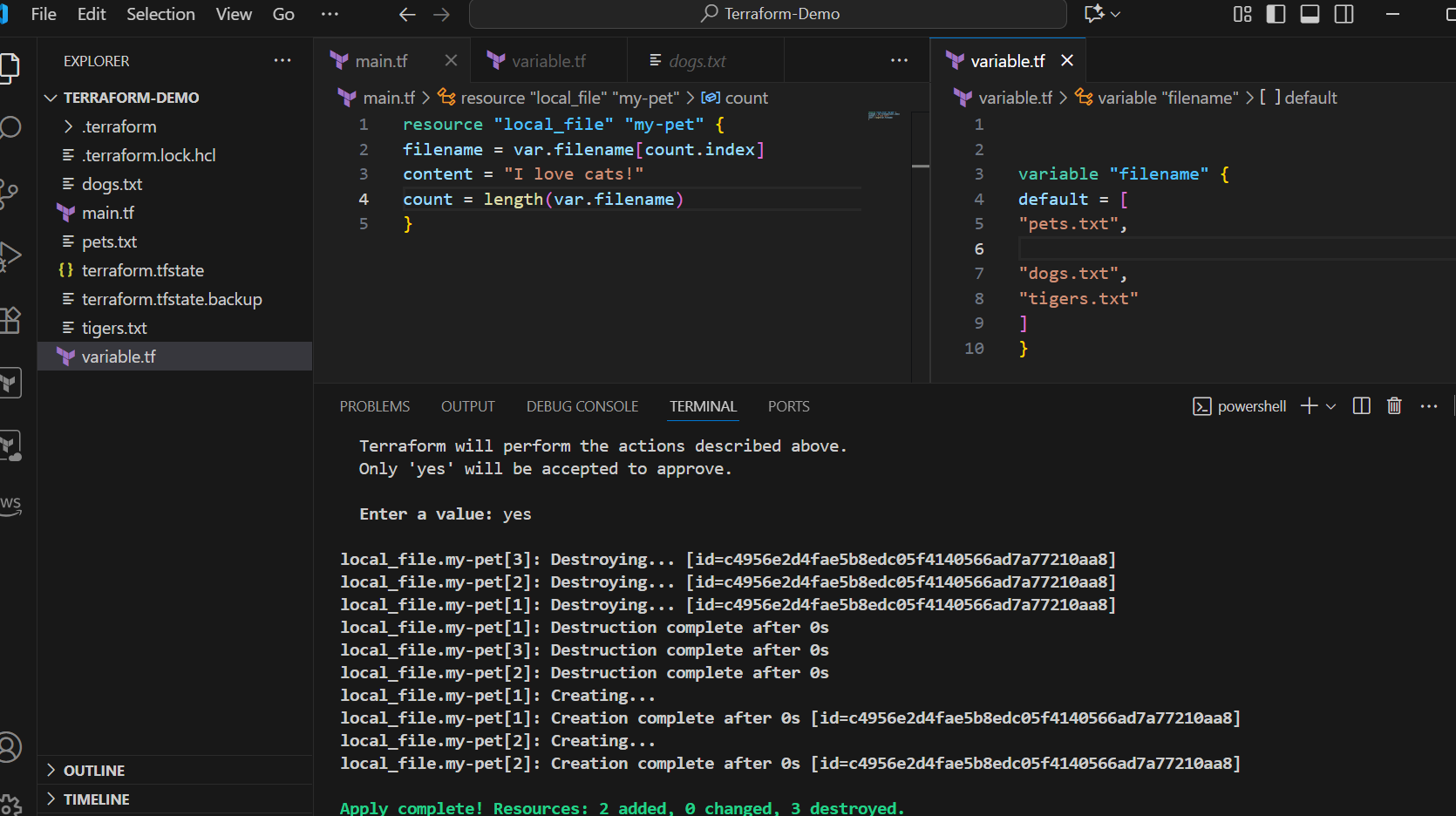
4) For\_each

Example of count:

================

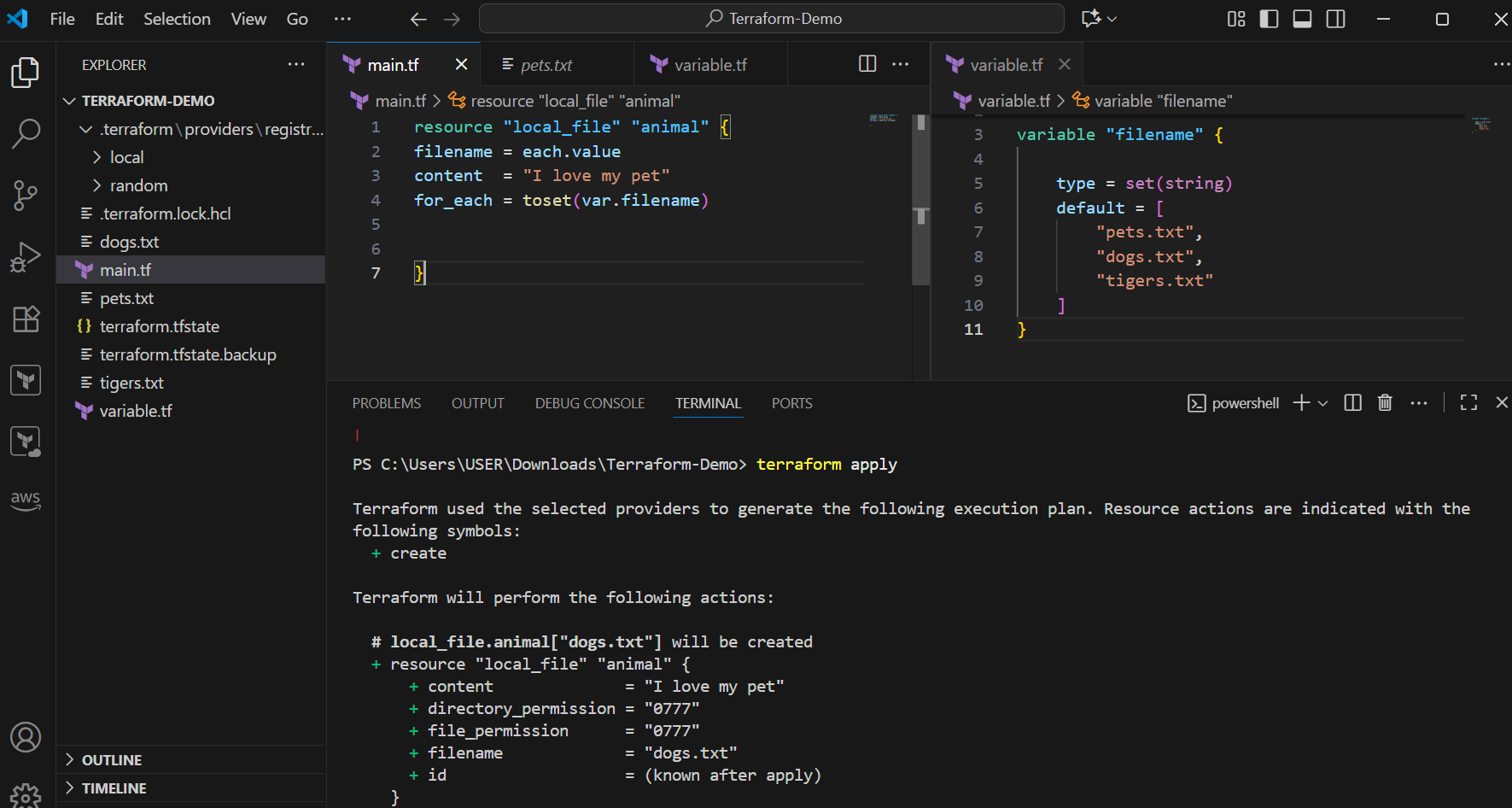
If we use count as 3 then it will create 3 files with pet[0],pet[1],pet[2]

****

****

Example of for\_each:

====================

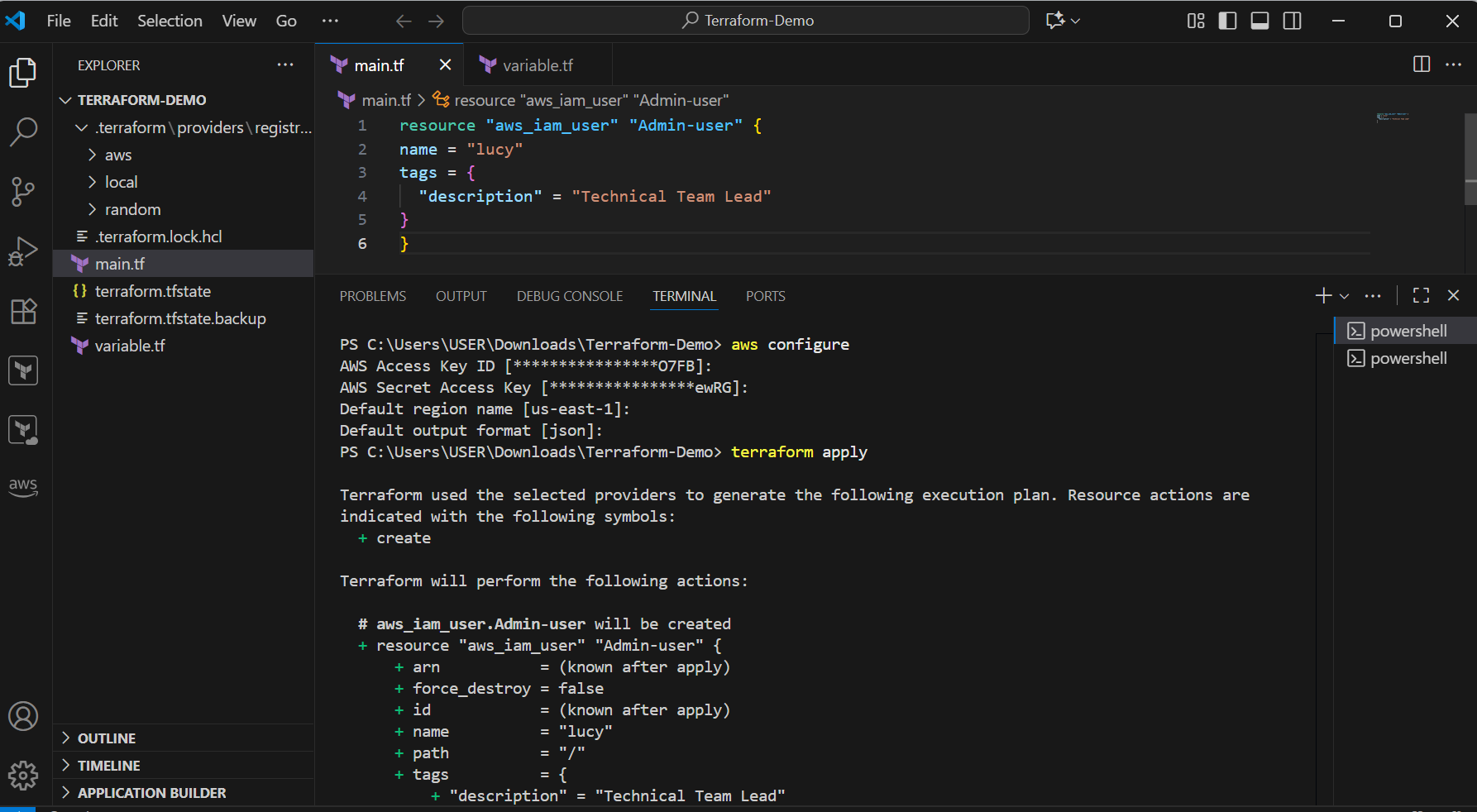
****

Terraform with AWS:

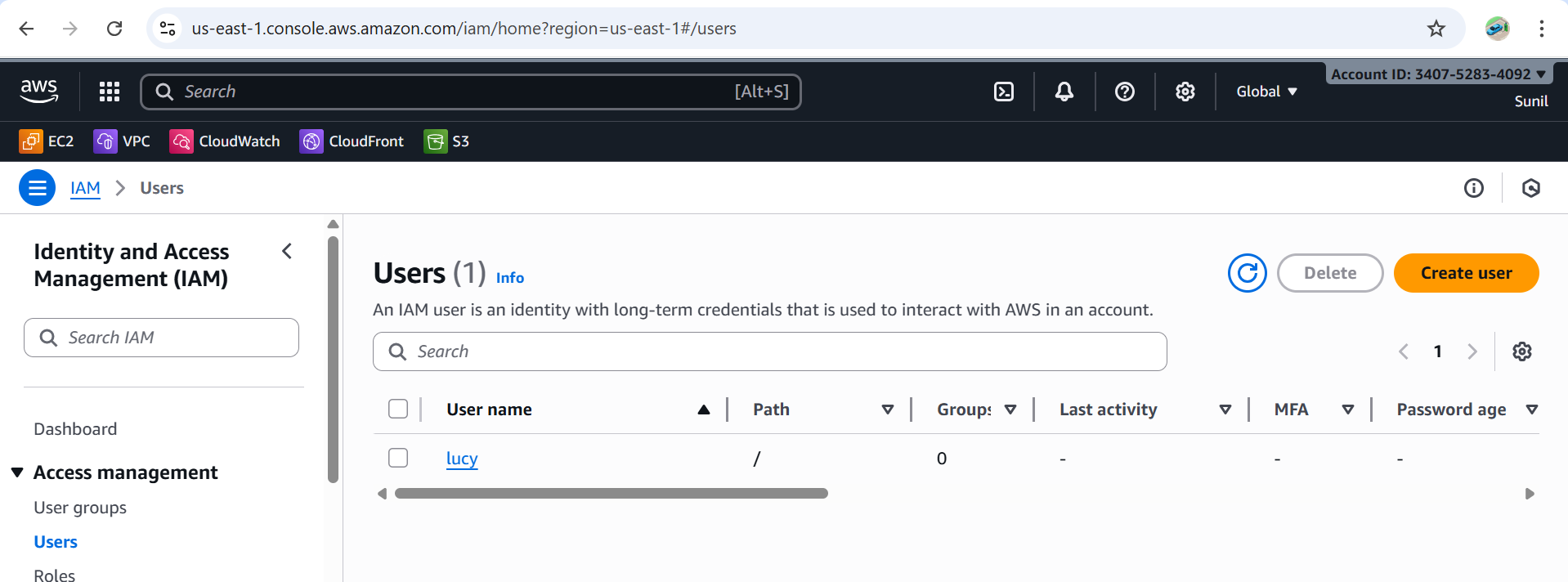
==================

--> first we need to create a secret key and access key and configure then in laptop.

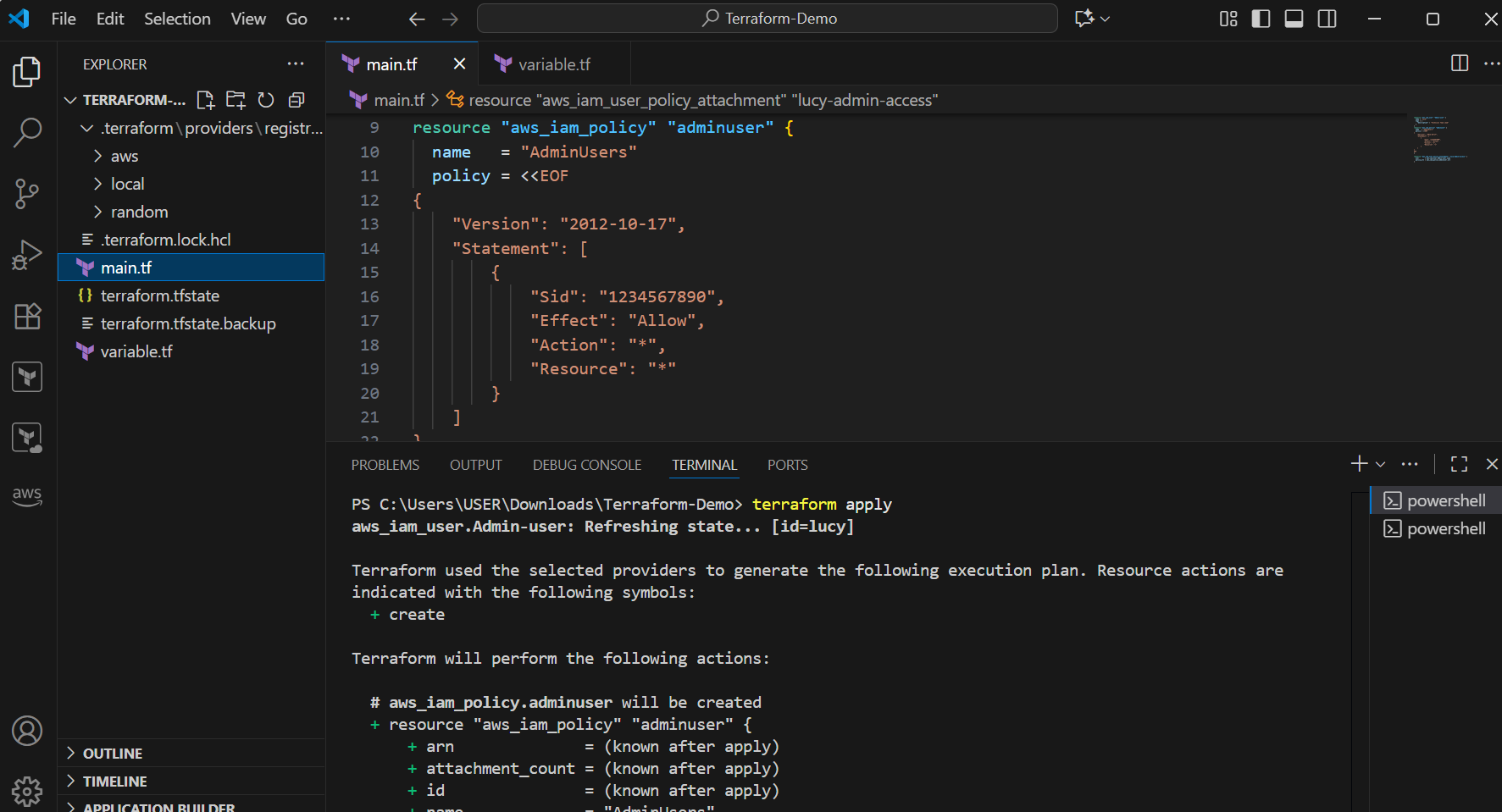
Example script to create aws iam:

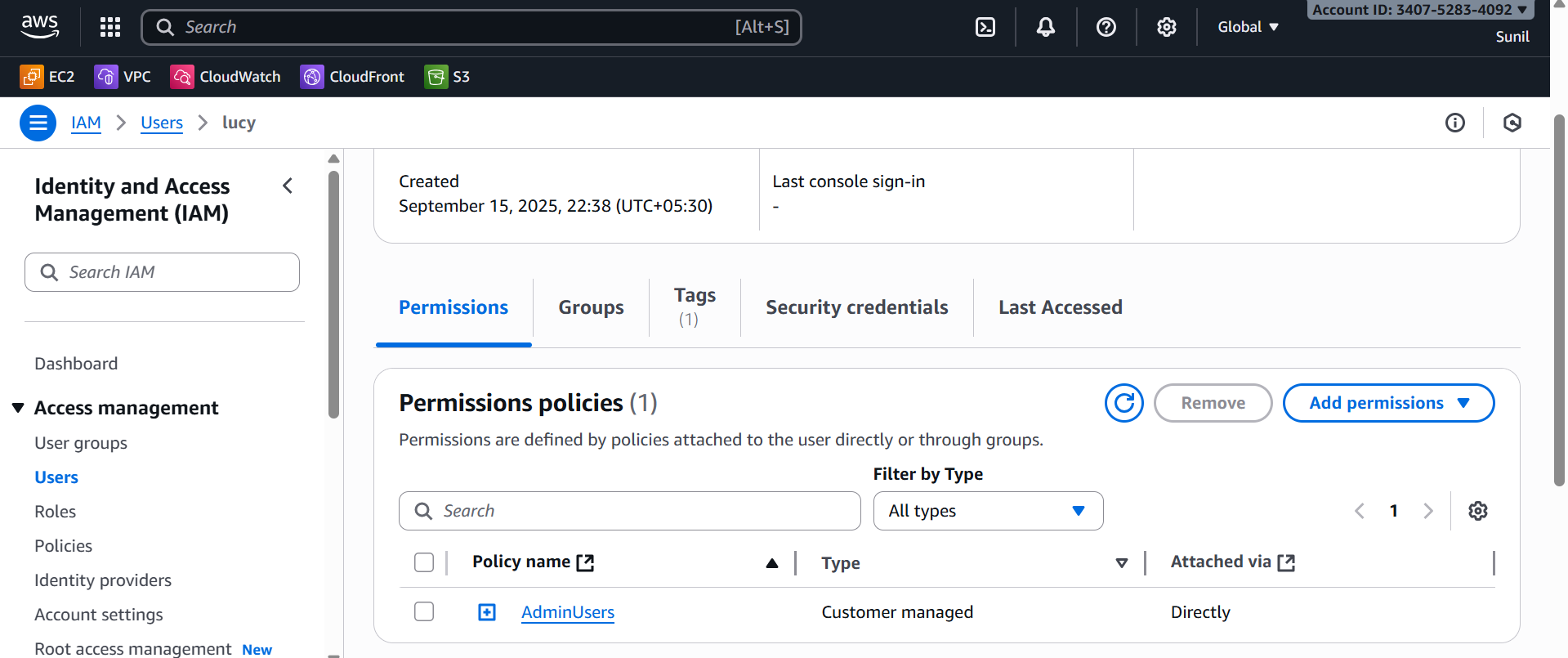
****

**User with name lucy has created in aws**

****

**Policy attach to the user**

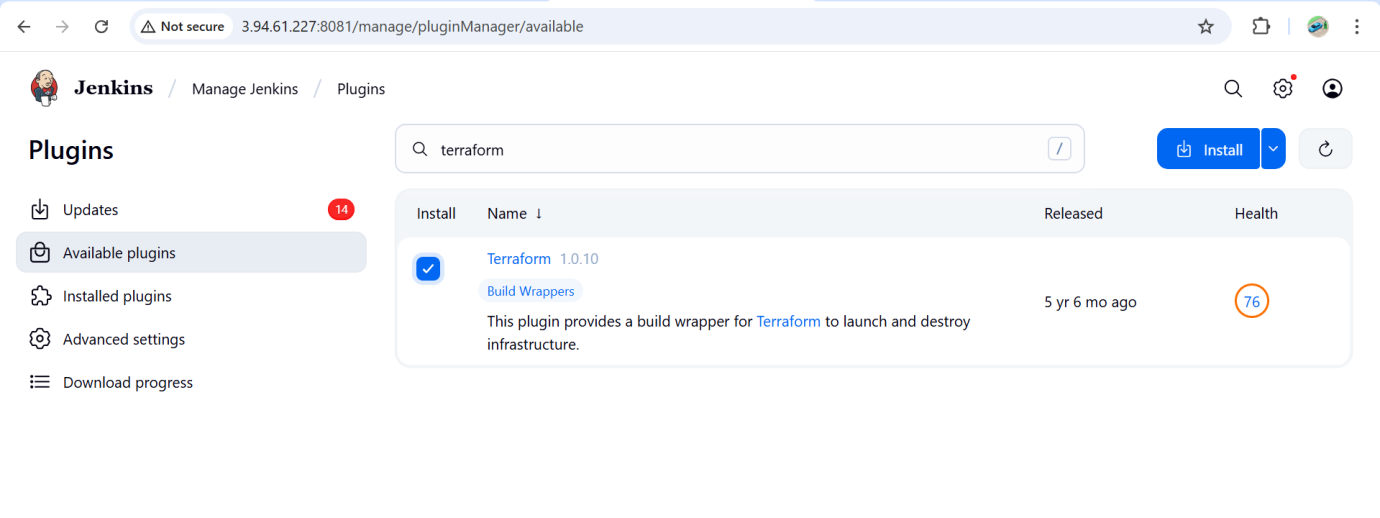
****

****

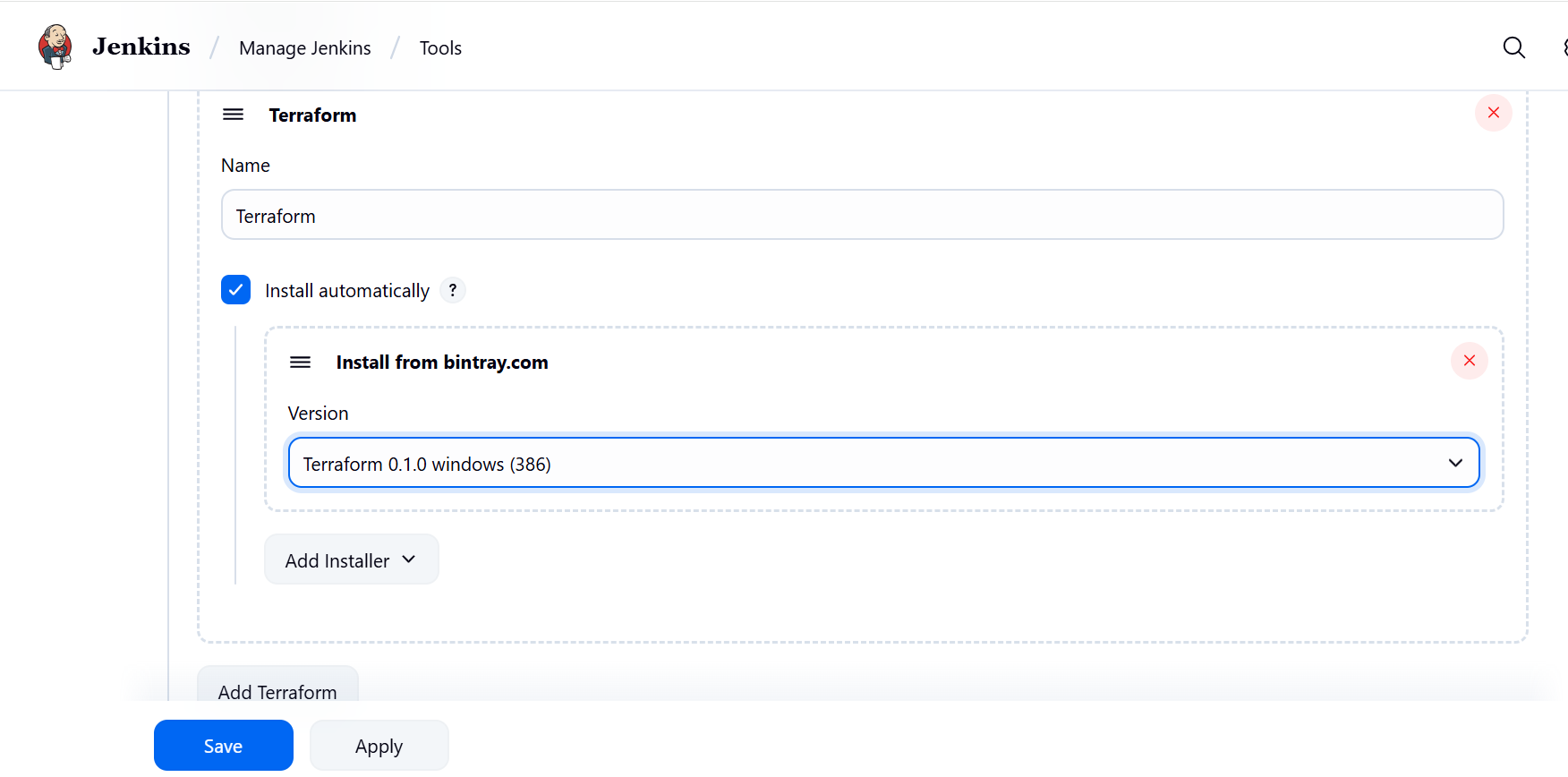
**3.Integrate Terraform in Jenkins using the Terraform plugin.**

**Step 1: Install the Terraform Plugin in Jenkins**

1. Go to **Jenkins Dashboard → Manage Jenkins → Plugins → Available Plugins**.
2. Search for **Terraform**.
3. Install **Terraform Plugin**
4. Restart Jenkins if required.

**Step 2: Configure Terraform in Jenkins**

1. Go to **Manage Jenkins → Tools**.
2. Find **Terraform installations**.
3. Click **Add Terraform**:
   * Name: terraform
   * Install automatically: ✅ (Jenkins will download it, or you can specify the path if already installed).
4. Save.

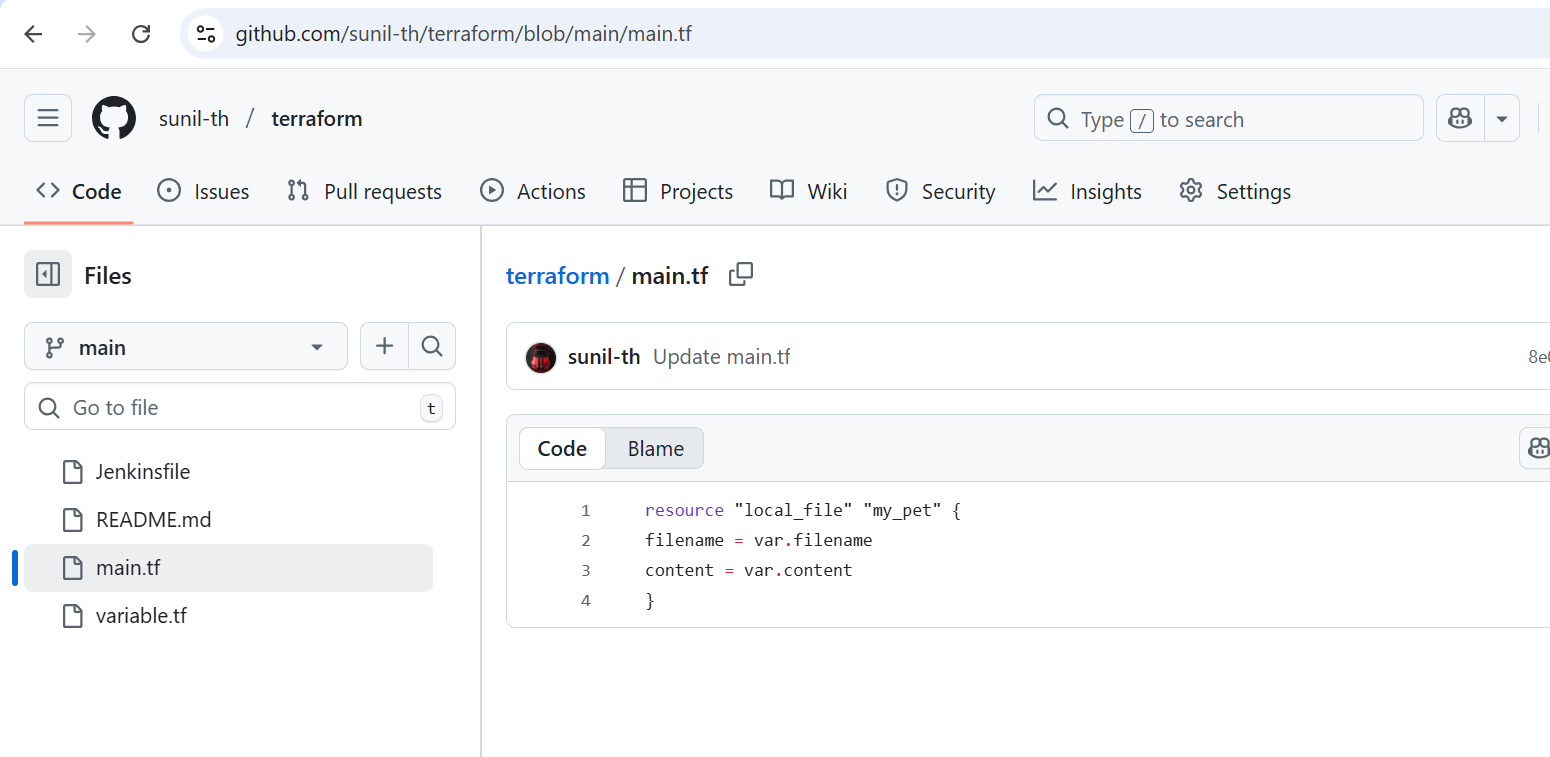


## ****Step 3: Create a Jenkins Job (Pipeline or Freestyle)****

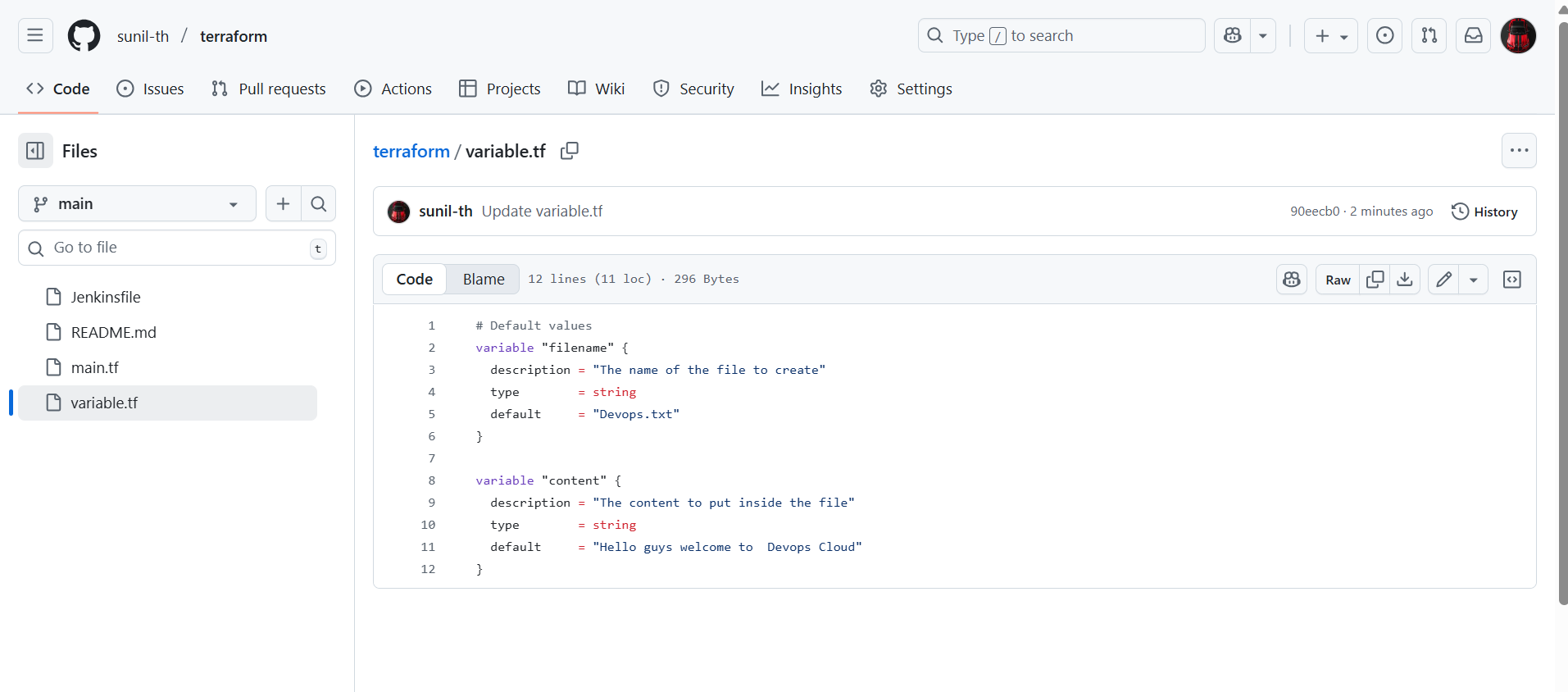
You can use either **Freestyle** or **Pipeline**.

**In Git hub add these files**

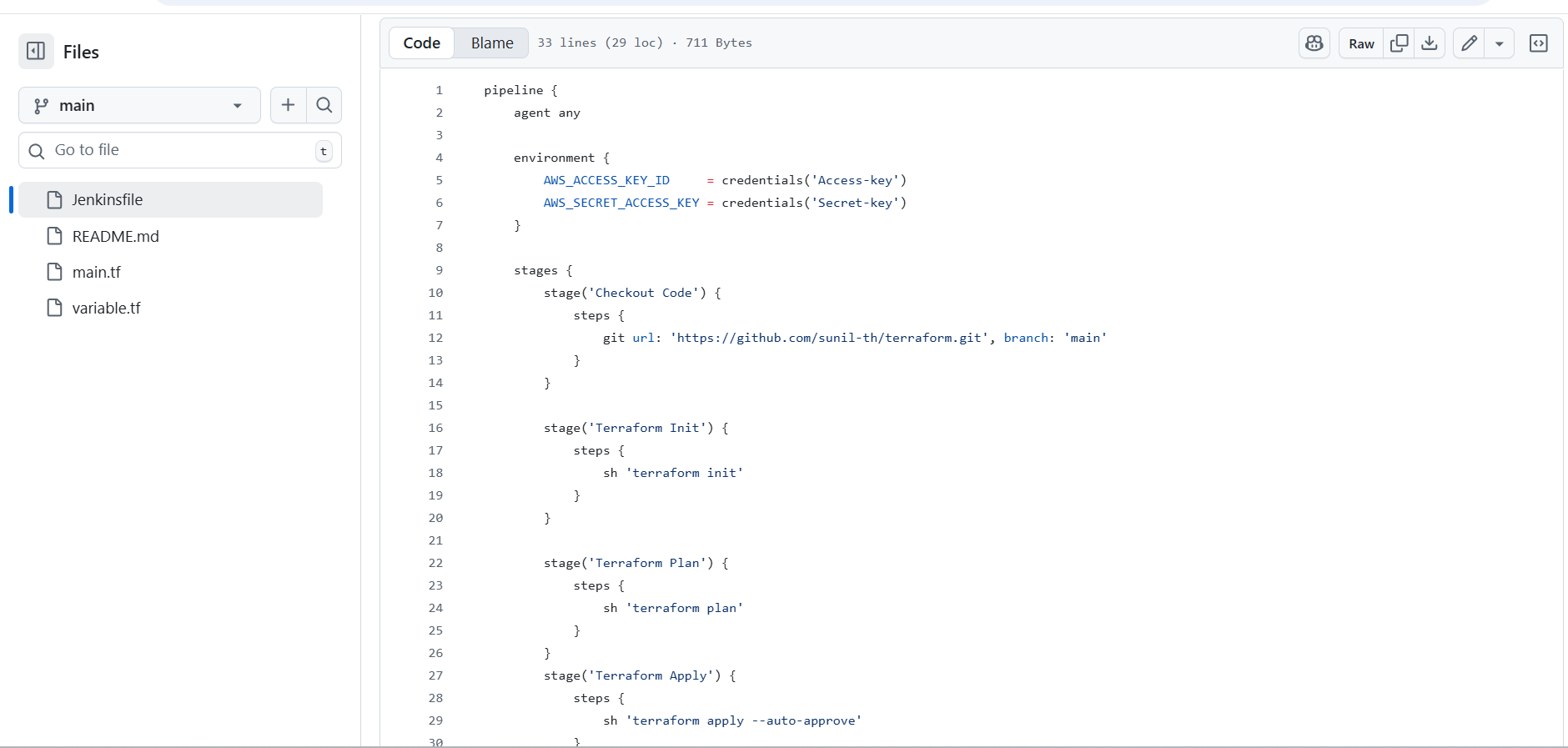
**Main.tf**

****

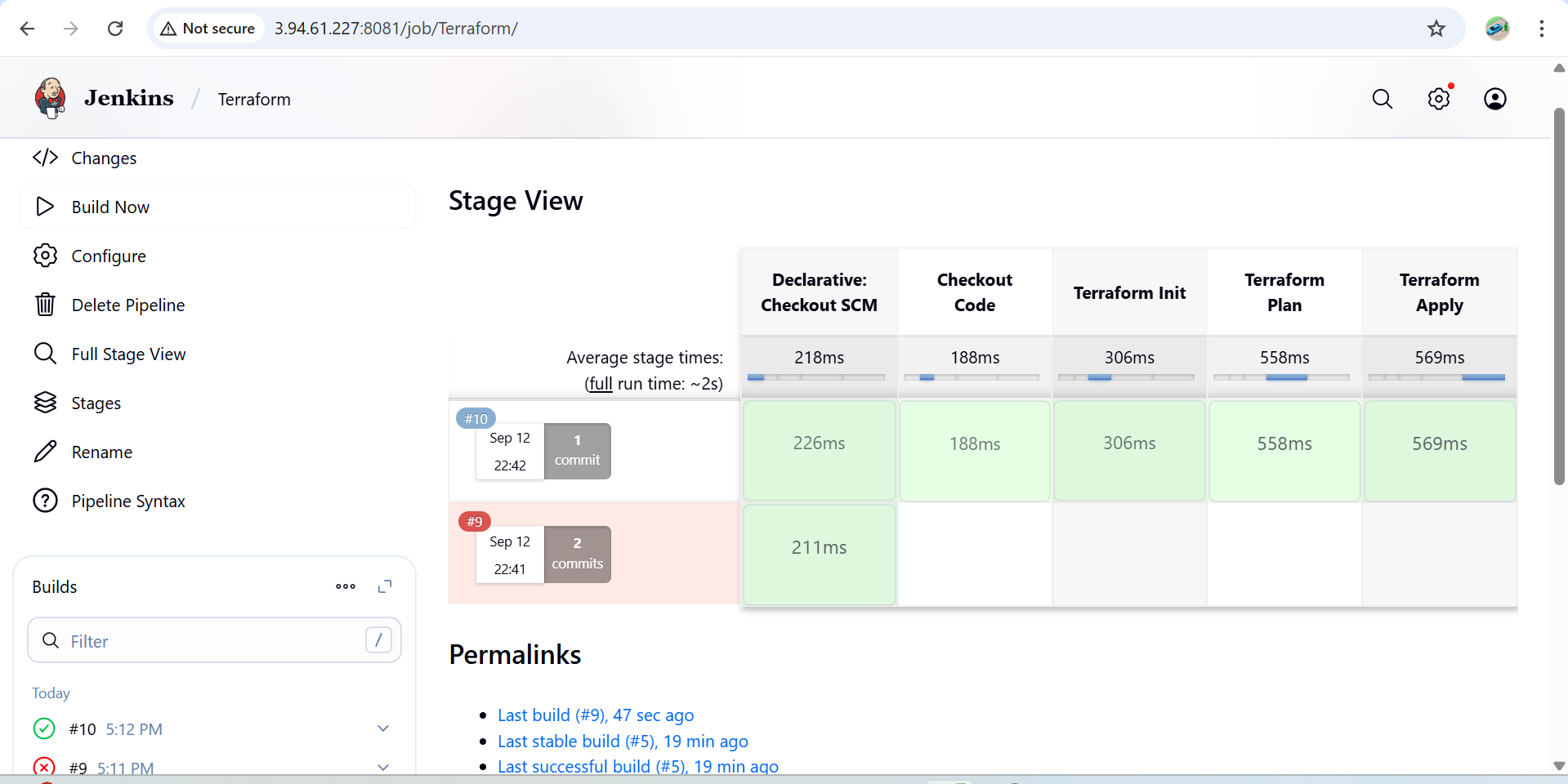
**Variable.tf**

****

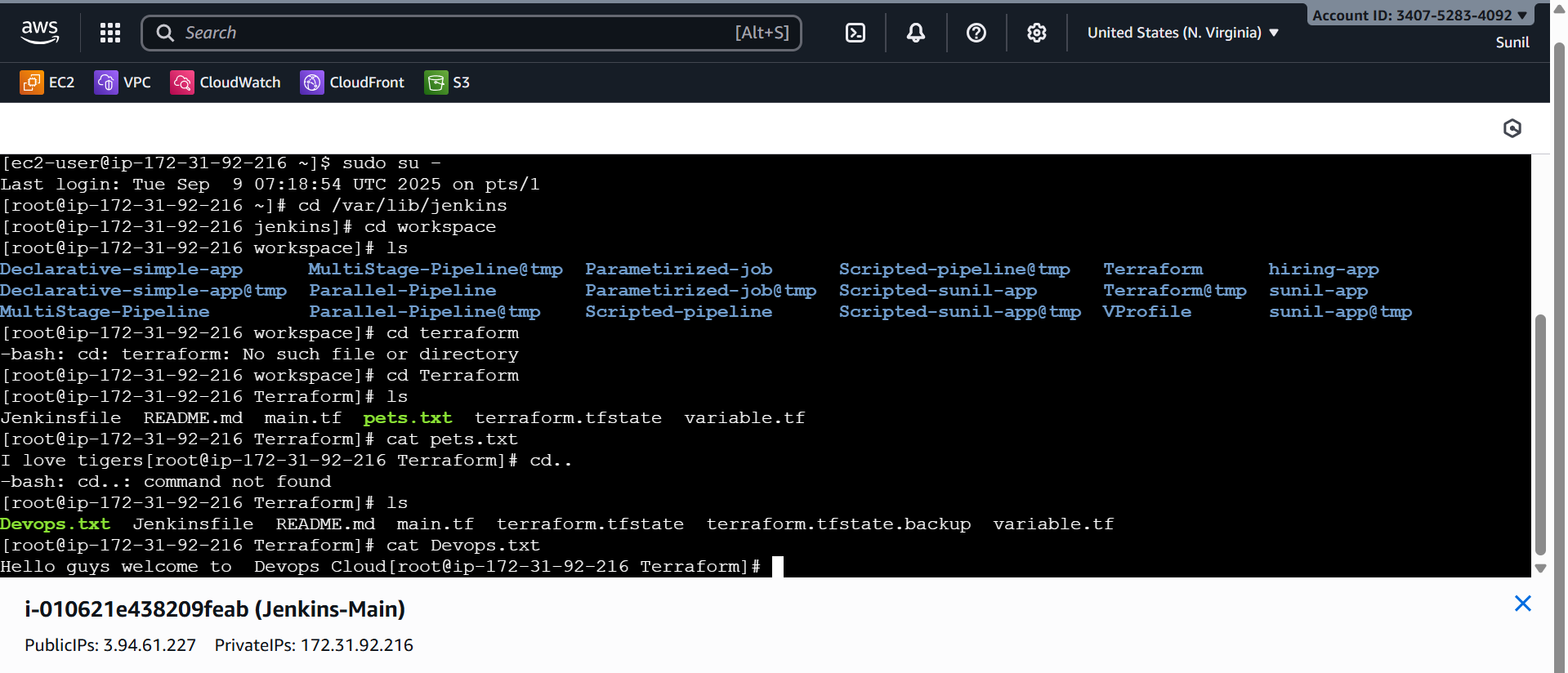
**Jenkinfile**

****

**Create a jenkinJob and build**

****

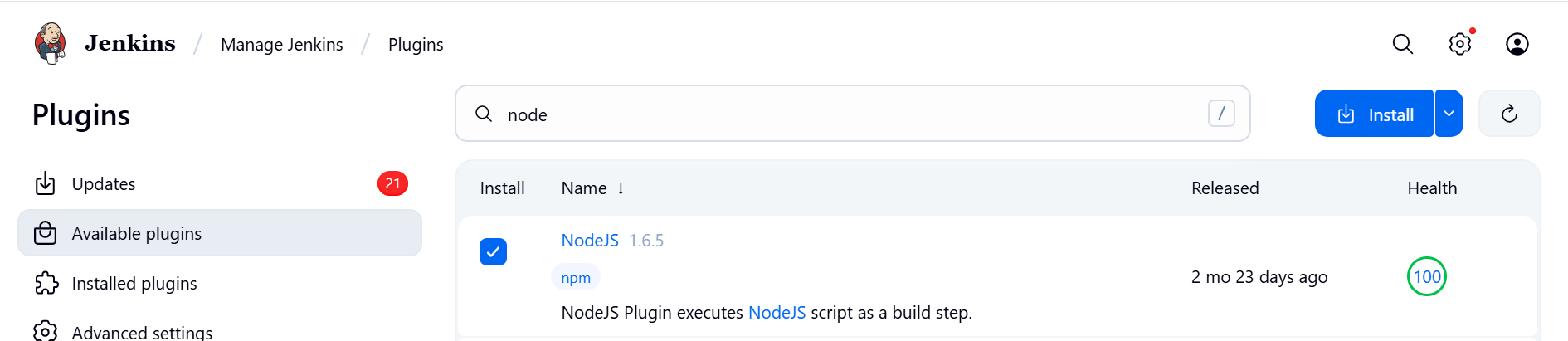
Check the Jenkins main workspace and cat the file inside the job you can view the content in the file



**4.) Create a CI/CD pipeline for a Nodejs Application:** [**https://github.com/betawins/Trading-UI.git**](https://github.com/betawins/Trading-UI.git)

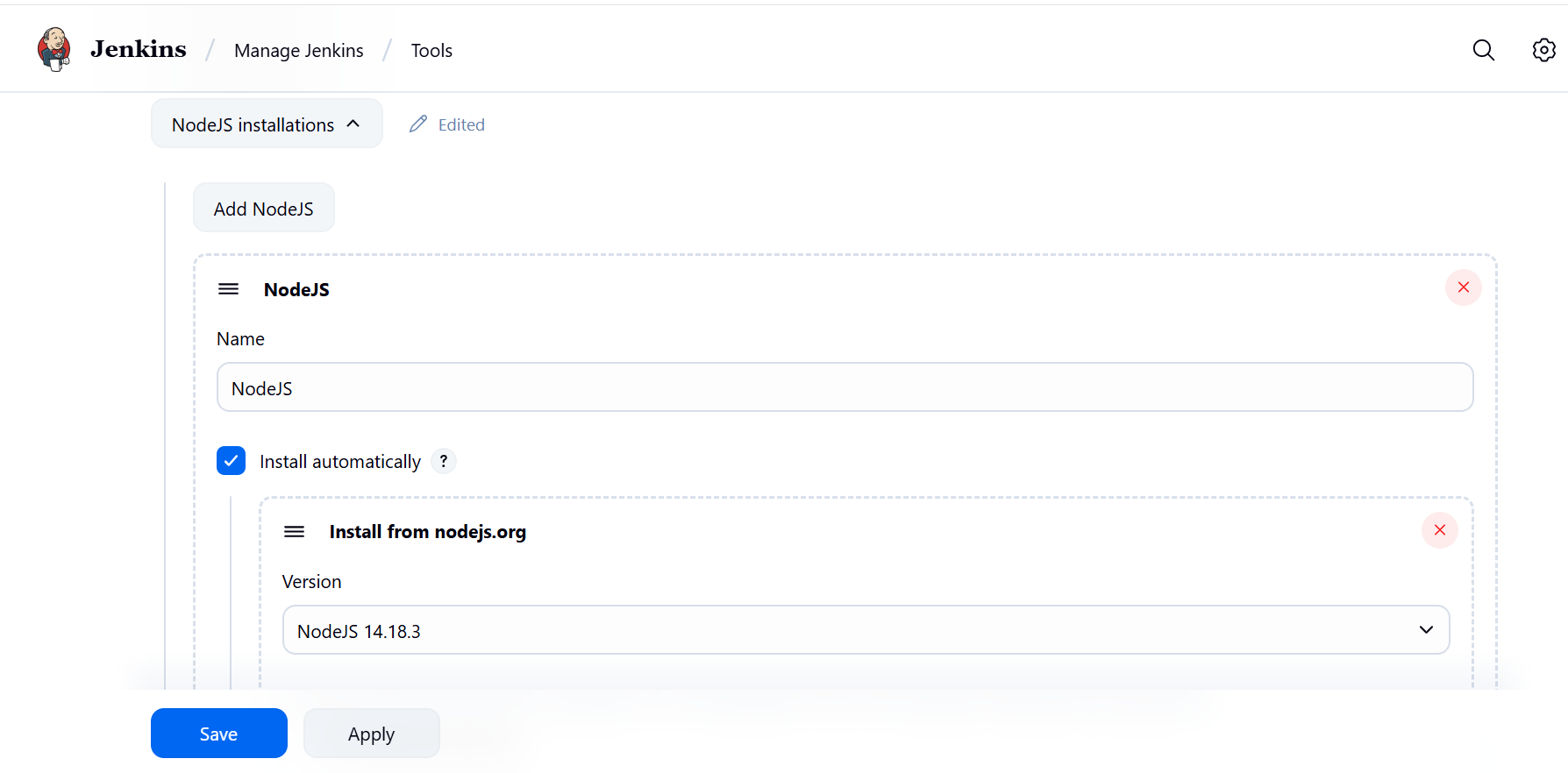
**Set Up Jenkins**

1. Install required plugins:
   * **NodeJS plugin** (to install Node.js runtime)

****

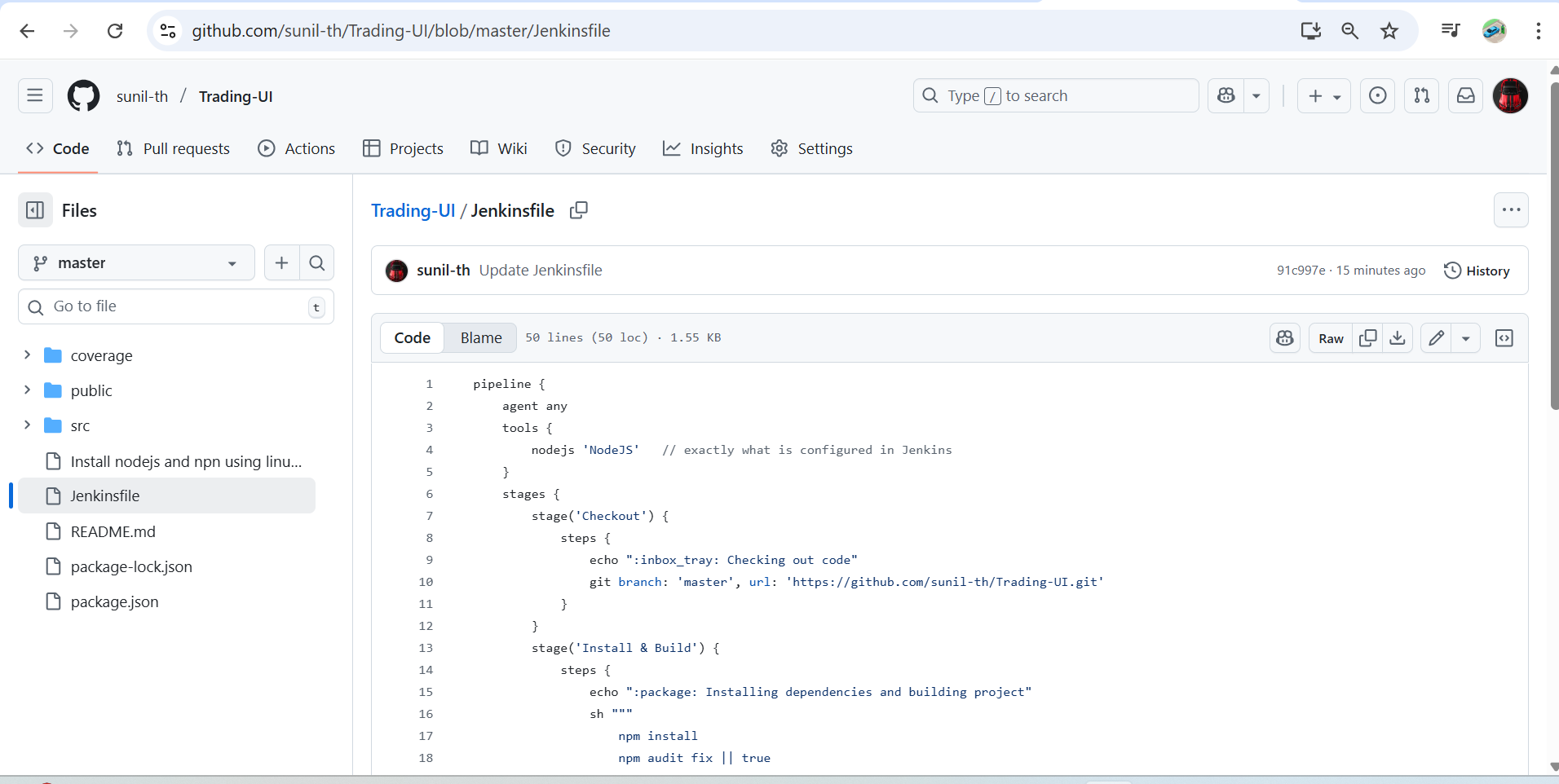
Configure **NodeJS Tool** in Jenkins:

* + Go to **Manage Jenkins → Global Tool Configuration → NodeJS installations**
  + Add a version (e.g., NodeJS 14.x).

****

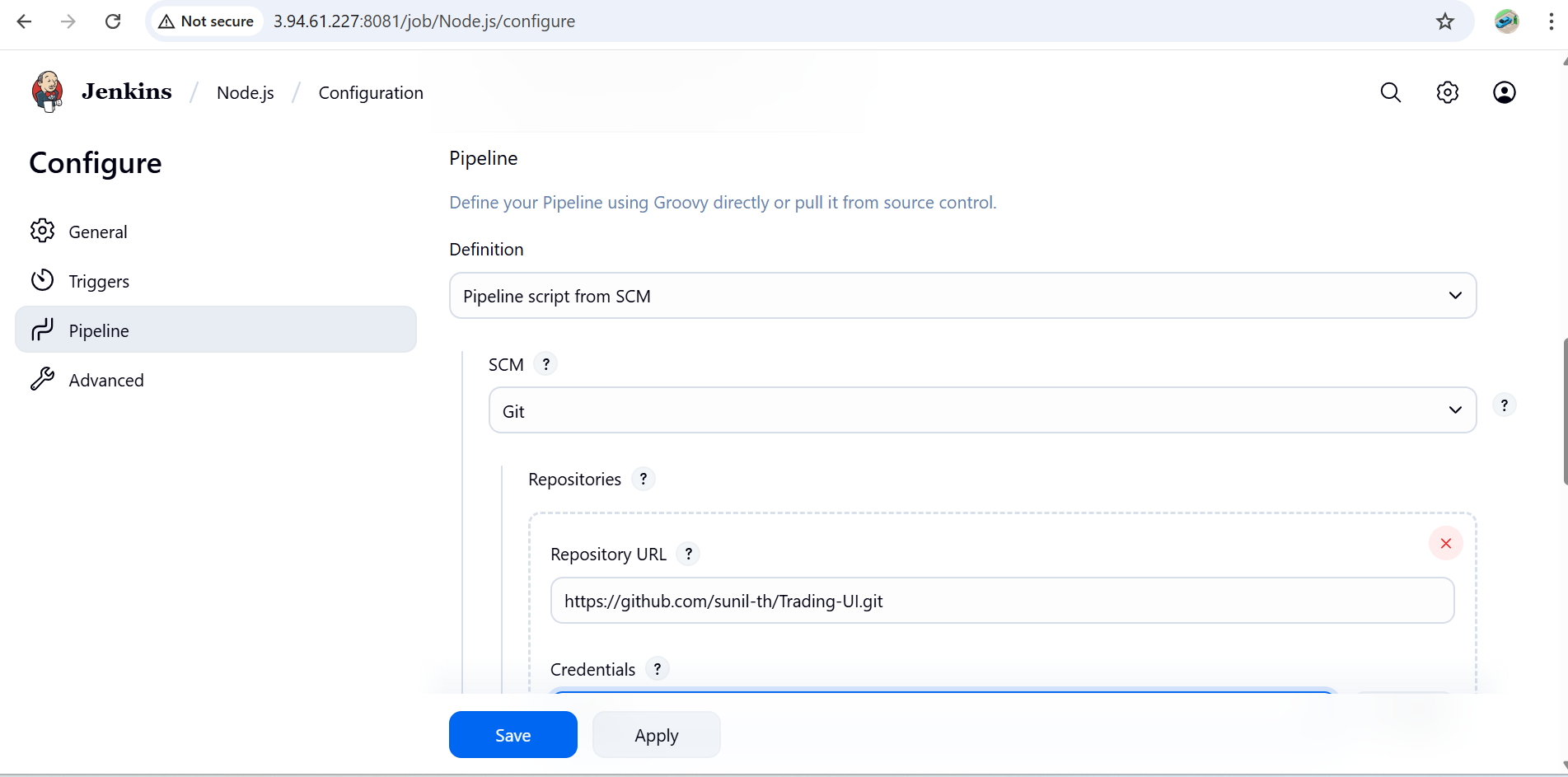
### ****Fork the github repo of the url****

### ****. Create Jenkinsfile in GitHub Repo****

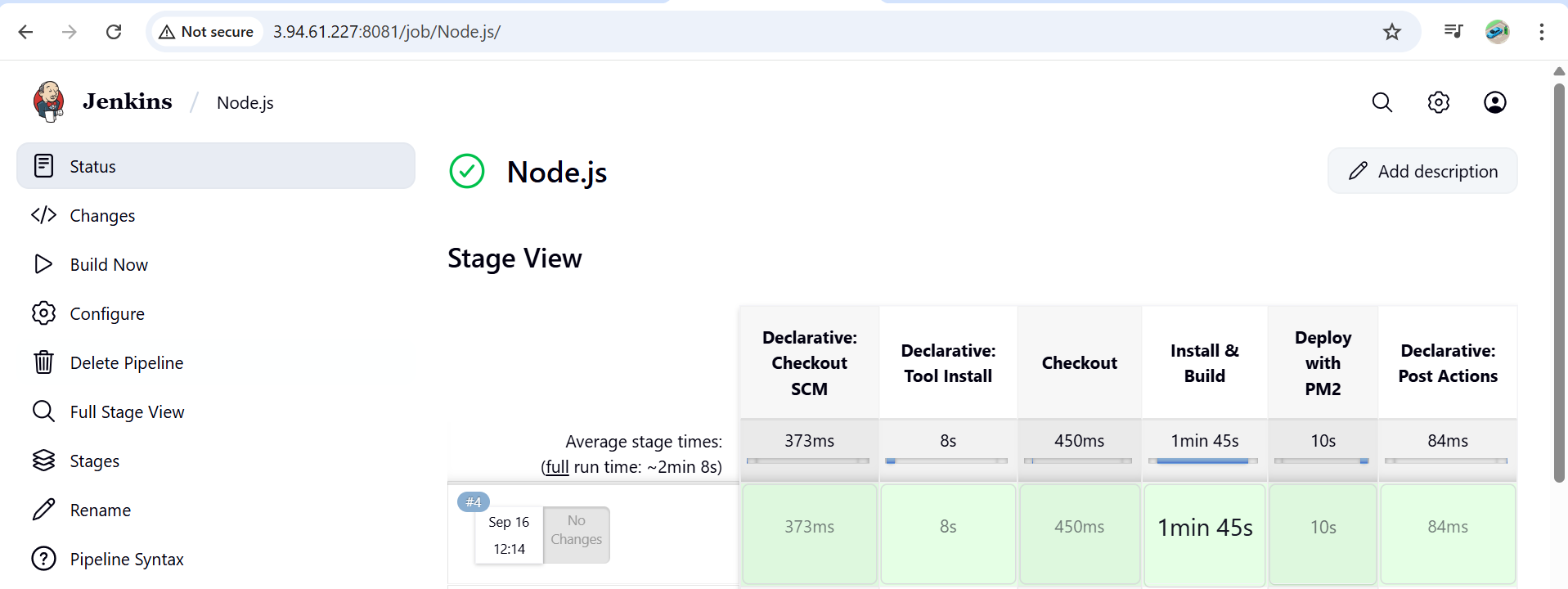
****

Create a **Jenkins Pipeline Job**:

* + New Item → Pipeline → Name it → Select "Pipeline from SCM"
  + Choose **Git** and provide repo URL.

****

**Save and build**

****

**5.) Explain 10 Maven commands.**

**1. mvn clean**

* Removes the **target/** directory (where compiled files, packaged JARs/WARs, etc. are stored).
* Ensures you start with a **clean build environment**.

mvn clean

**2. mvn compile**

* Compiles the **source code** of the project (from src/main/java).
* Output goes to target/classes.

mvn compile

**3. mvn test**

* Compiles and runs the **unit tests** (in src/test/java) using JUnit/TestNG.

mvn test

**4. mvn package**

* Packages the compiled code into a **JAR/WAR** file as defined in pom.xml.
* Example: target/my-app-1.0-SNAPSHOT.jar

mvn package

**5. mvn install**

* Installs the packaged project into the **local Maven repository** (~/.m2/repository).
* Makes the artifact available for use in other local projects.

mvn install

**6. mvn deploy**

* Copies the packaged code to a **remote repository** (like Nexus, Artifactory).
* Used in CI/CD pipelines for sharing artifacts with other developers.

mvn deploy

**7. mvn site**

* Generates a **site documentation** (reports, dependencies, plugins info, etc.) for the project in target/site.

mvn site

**8. mvn dependency:tree**

* Shows the **dependency hierarchy** of the project.
* Useful for identifying conflicts between library versions.

mvn dependency:tree

**9. mvn validate**

* Validates the project’s structure and checks if pom.xml is correct.
* Runs before compilation.

mvn validate

**10. mvn verify**

* Runs any **integration tests** and checks if the package is valid.
* Ensures the project is ready for deployment.

mvn verify

✅ Summary:

| **Command** | **Purpose** |
| --- | --- |
| mvn clean | Remove target/ build files |
| mvn compile | Compile source code |
| mvn test | Run unit tests |
| mvn package | Package into JAR/WAR |
| mvn install | Install into local repo |
| mvn deploy | Deploy to remote repo |
| mvn site | Generate project site reports |
| mvn dependency:tree | Show dependency tree |
| mvn validate | Validate project structure |
| mvn verify | Run integration tests & verify package |