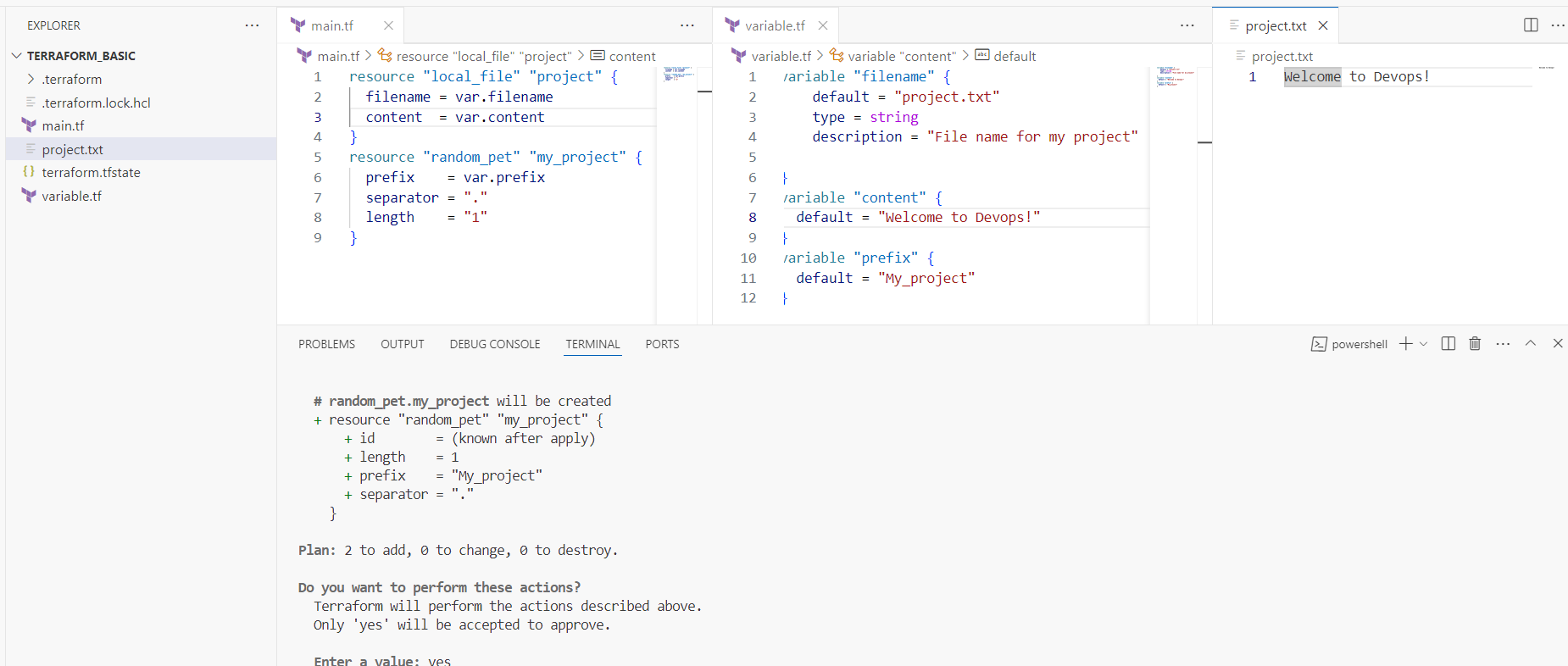
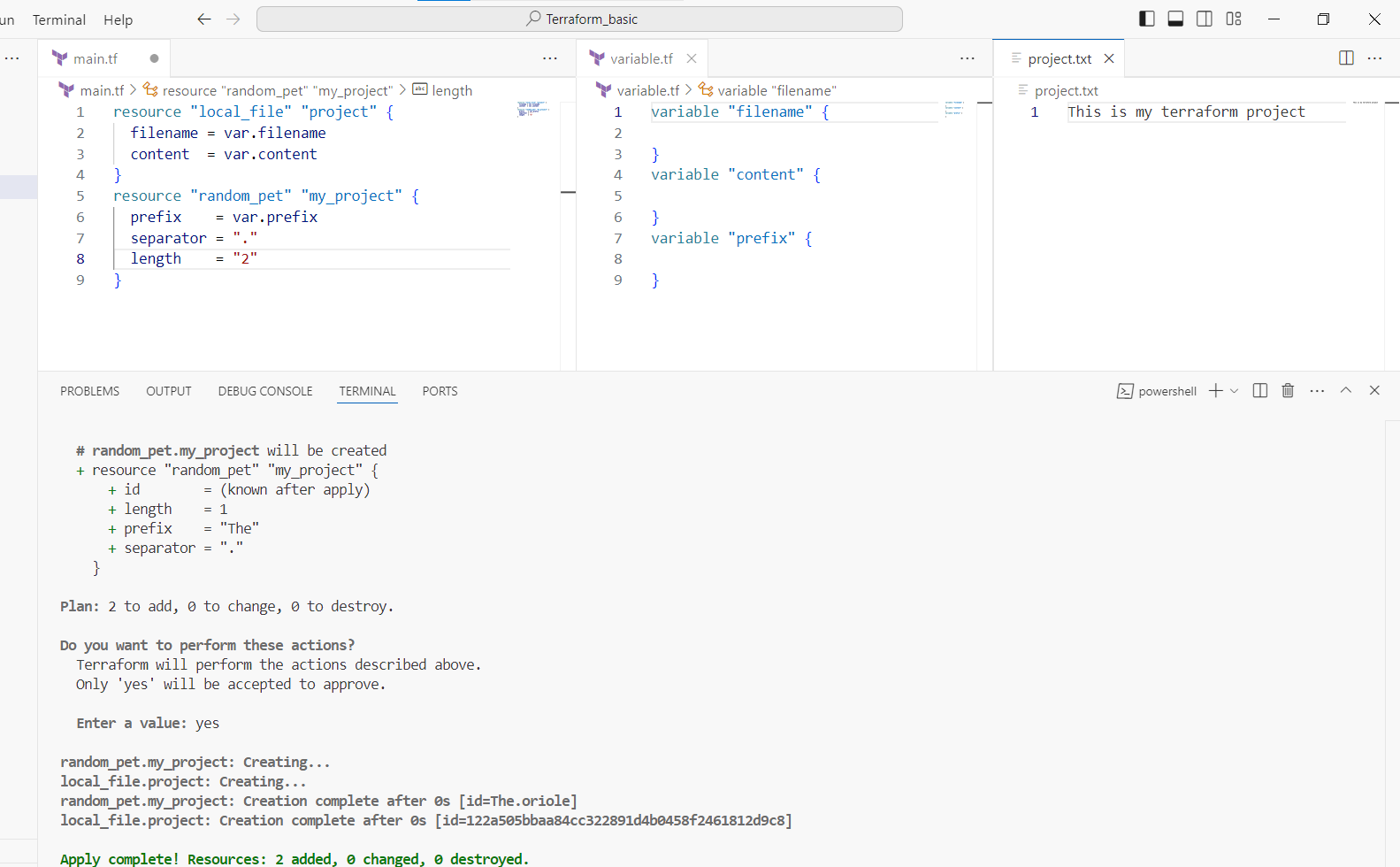
**Terraform -03-04- Tasks:**

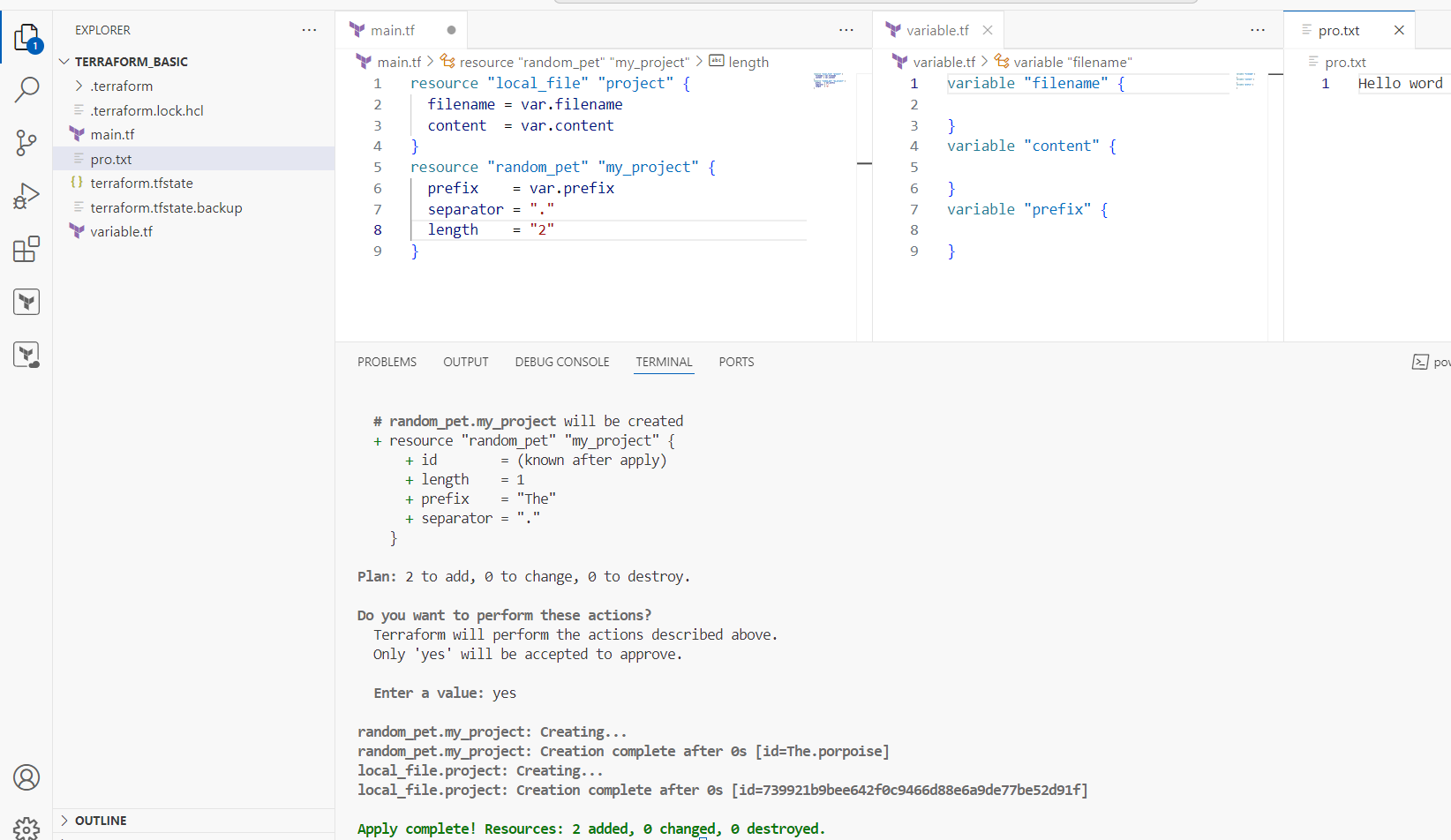
1. **Watch terraform-03 video.**

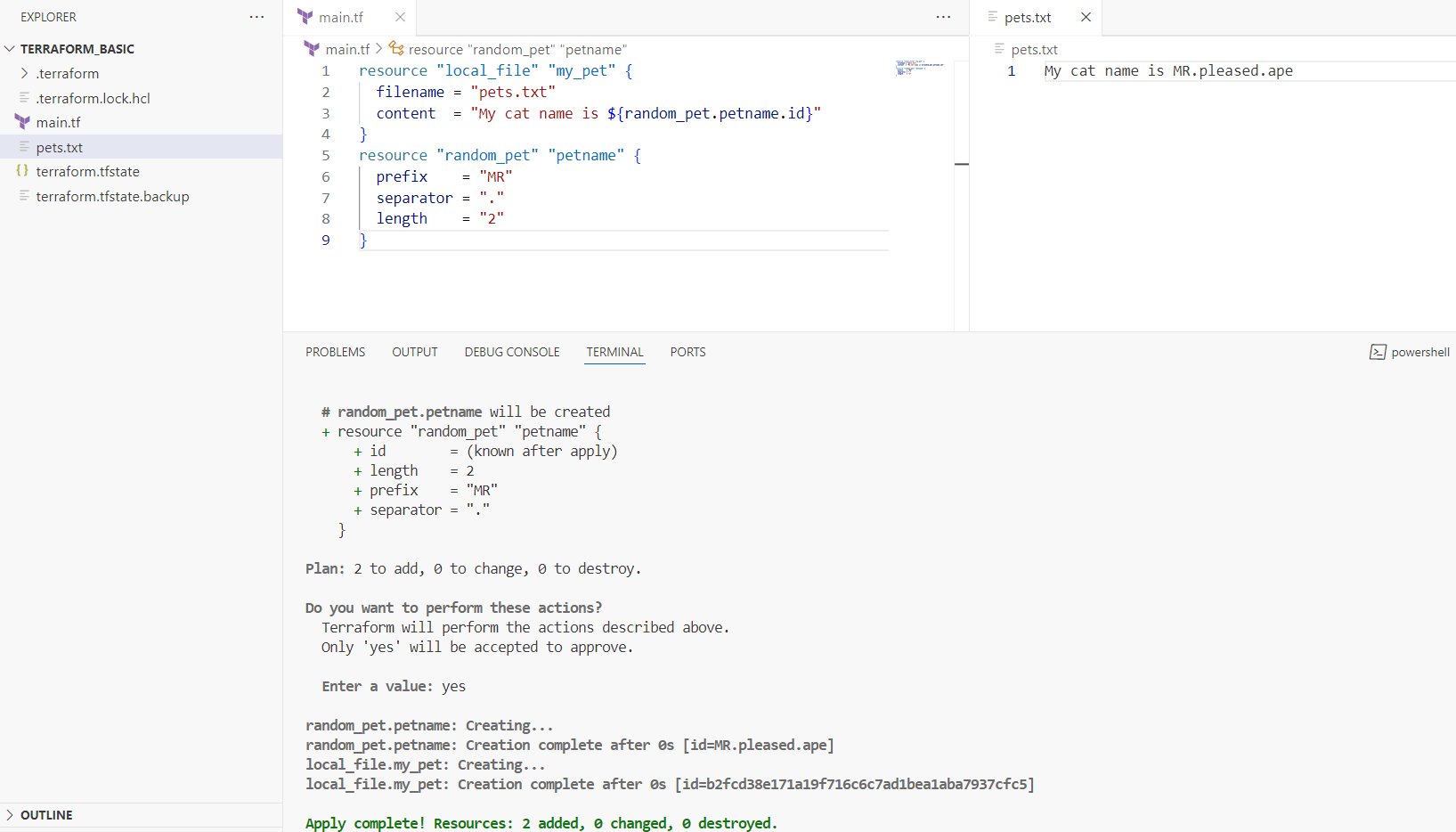
Done

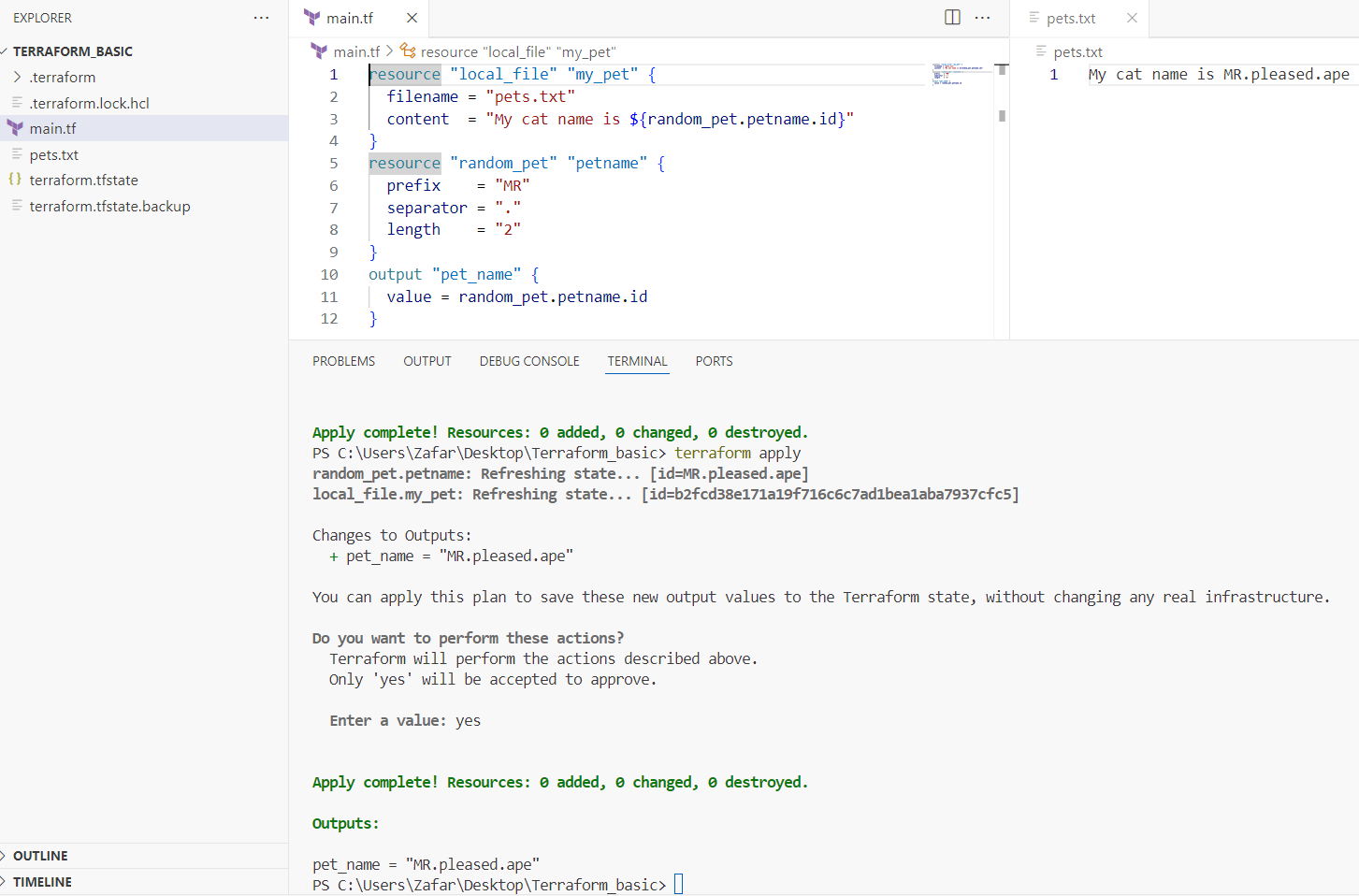
**2) Execute the script shown in video.**



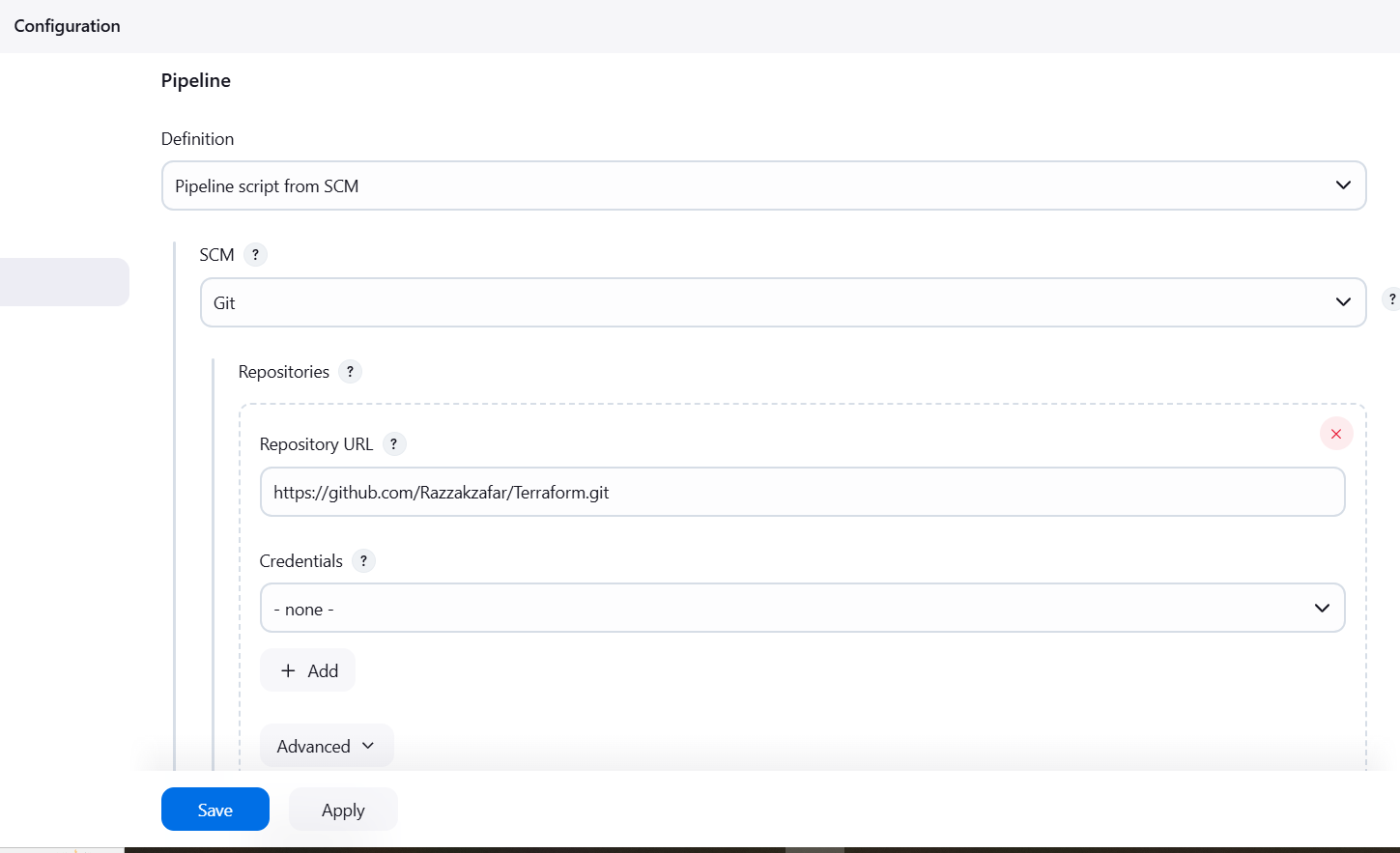


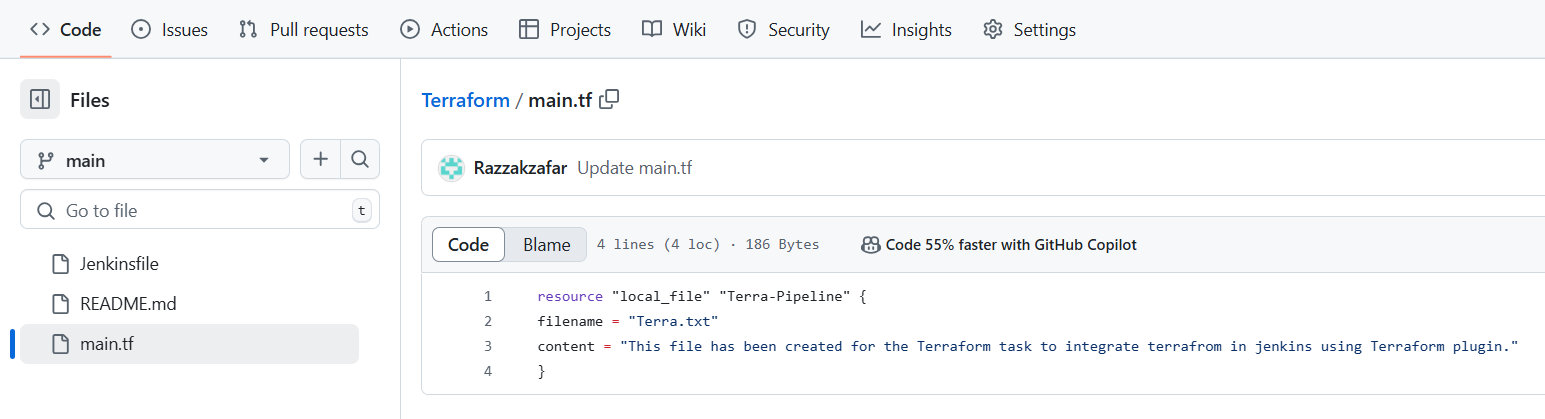


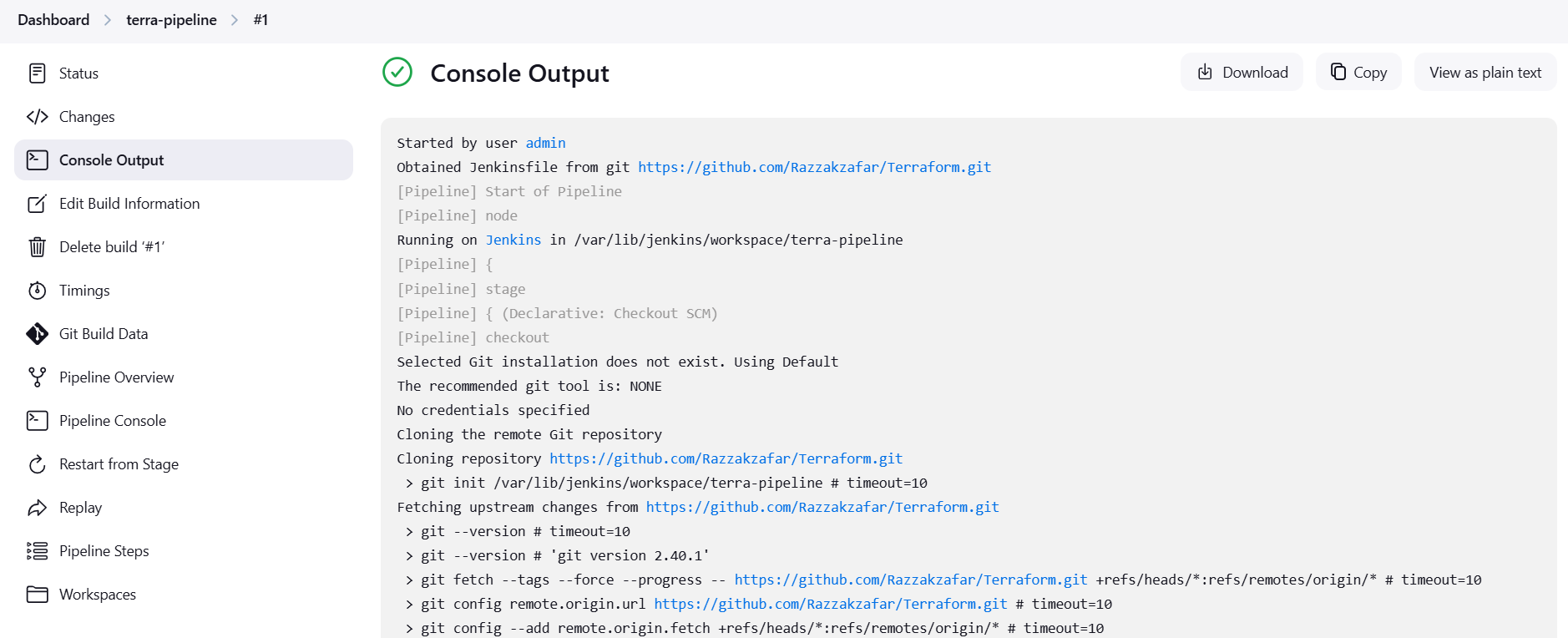




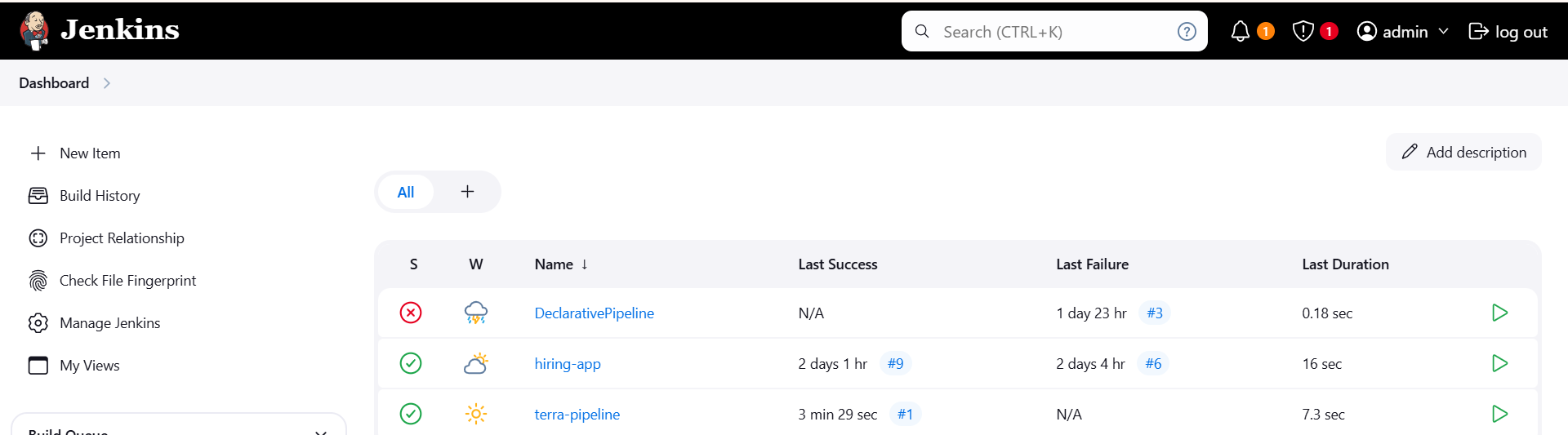
**3) Intergrate terrafrom in jenkins using Terraform plugin.**

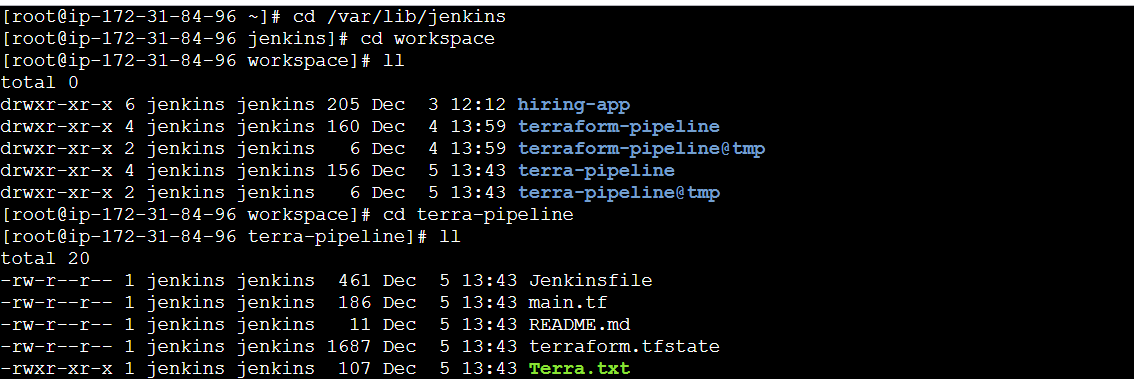












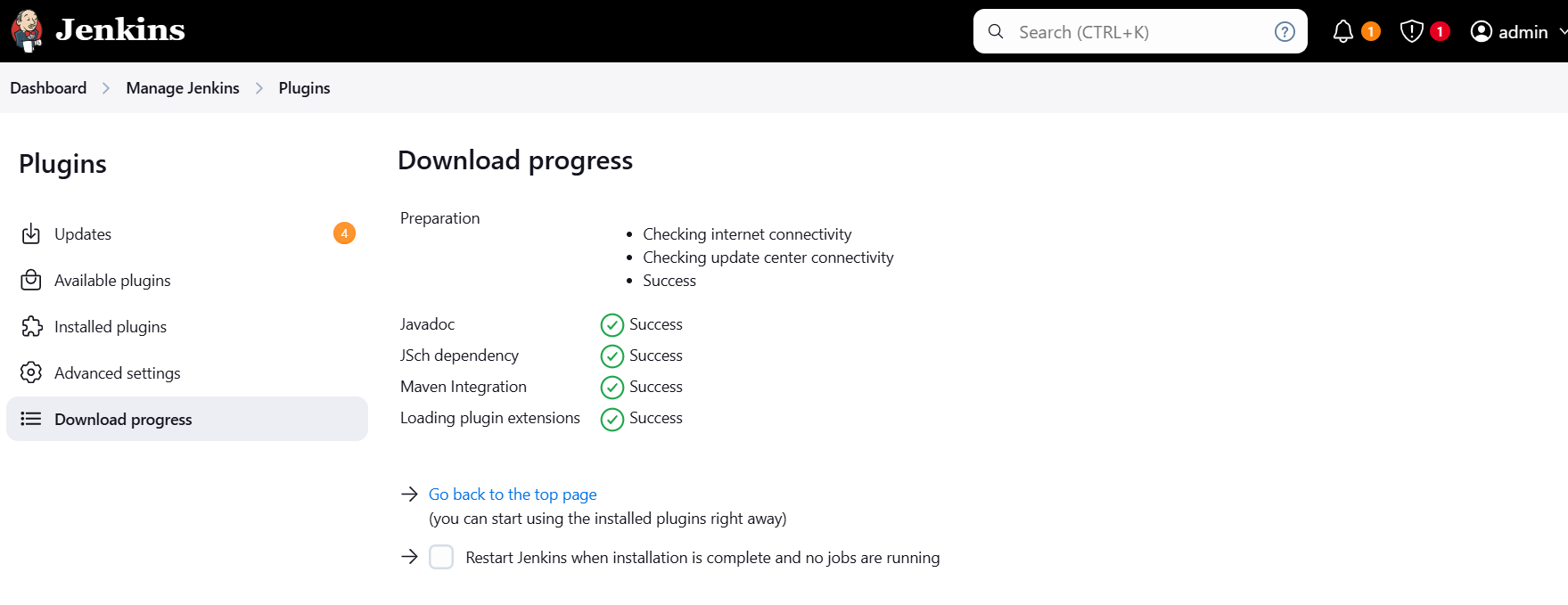
**4) Create one jenkins job using MAVEN PROJECT for the below code with two stages.**

**stage 1: Git clone**

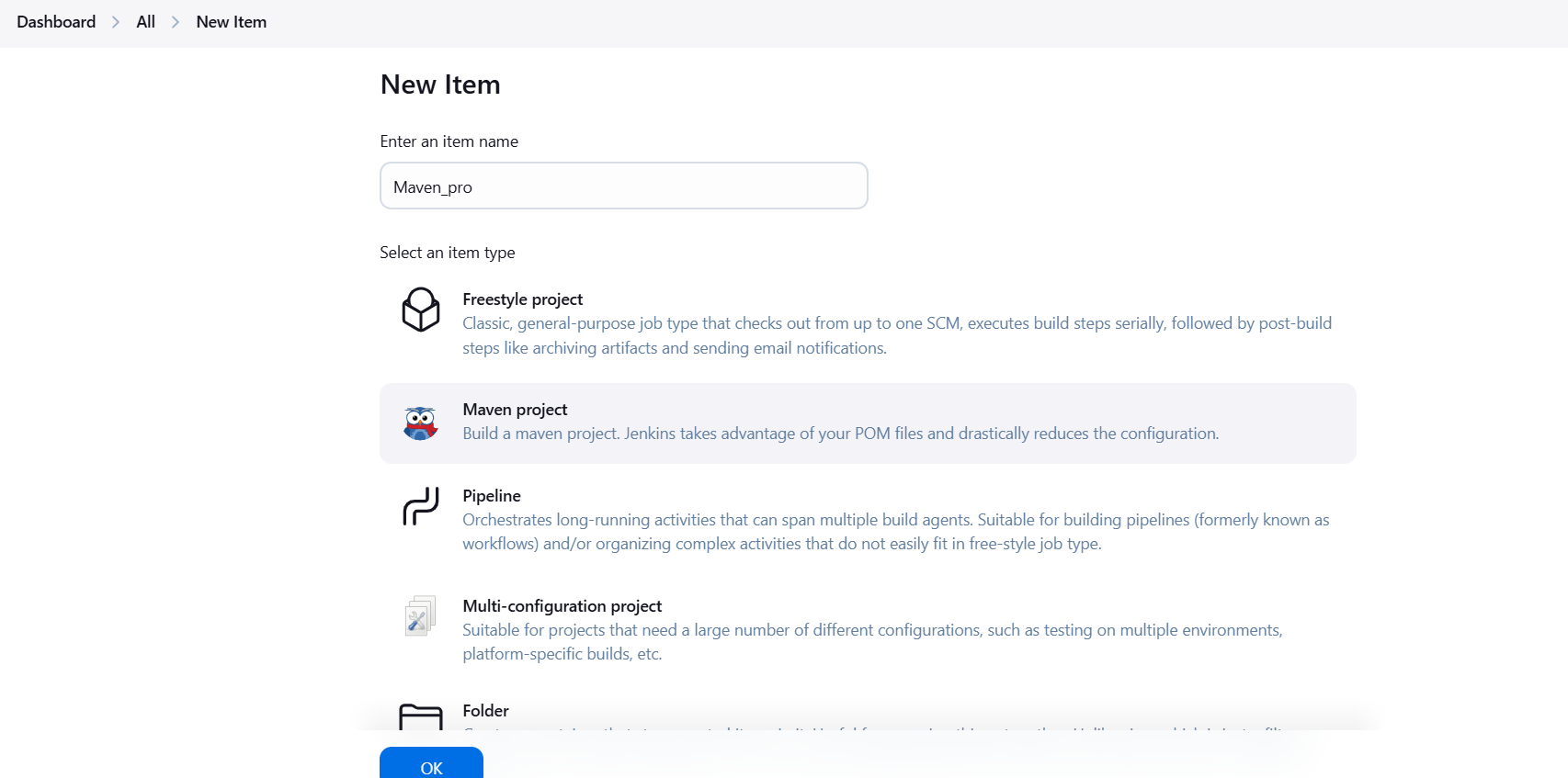
**stage 2: Maven Compilation**

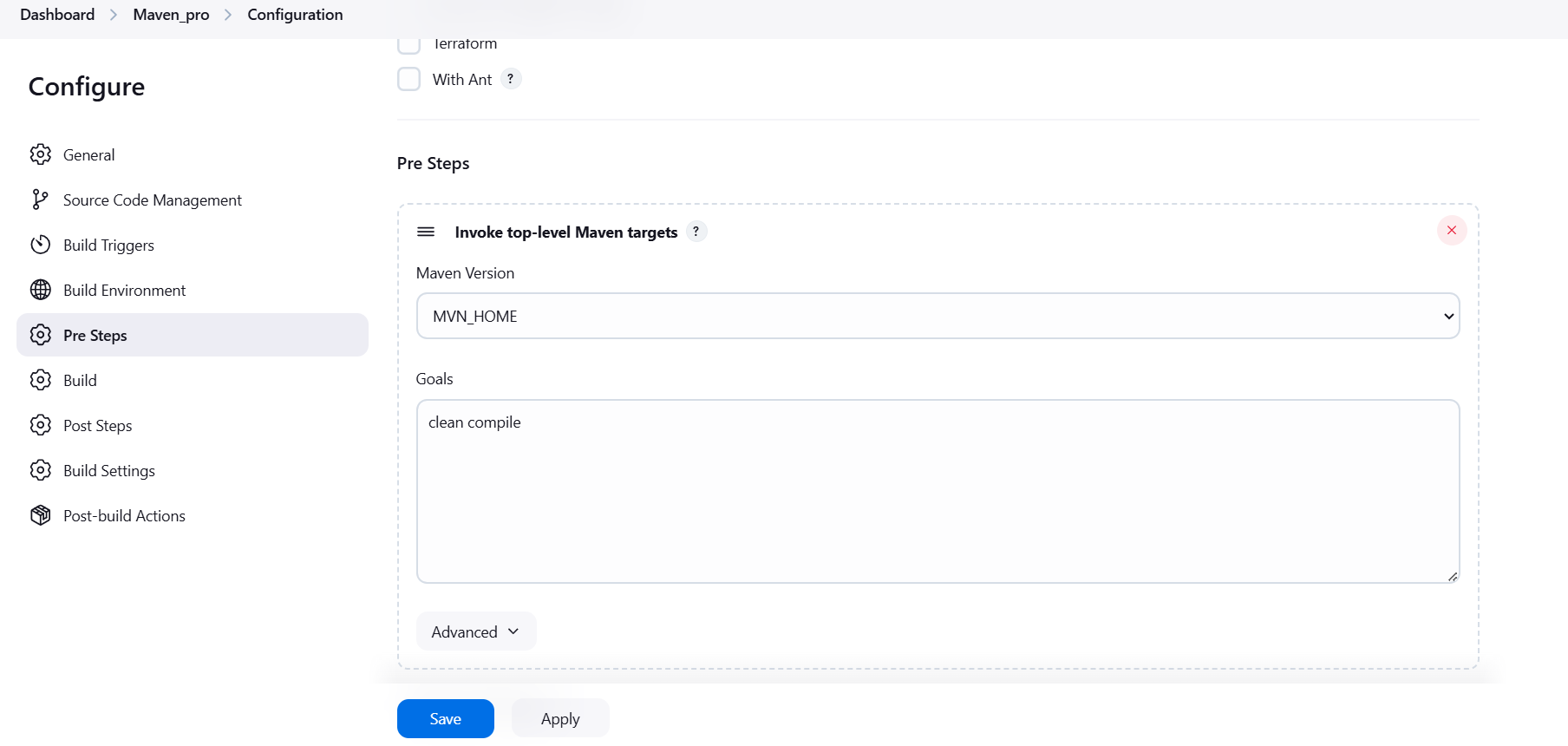
**Code**: <https://github.com/betawins/java-Working-app.git>

***>>>Download Maven Integration Plugin:***



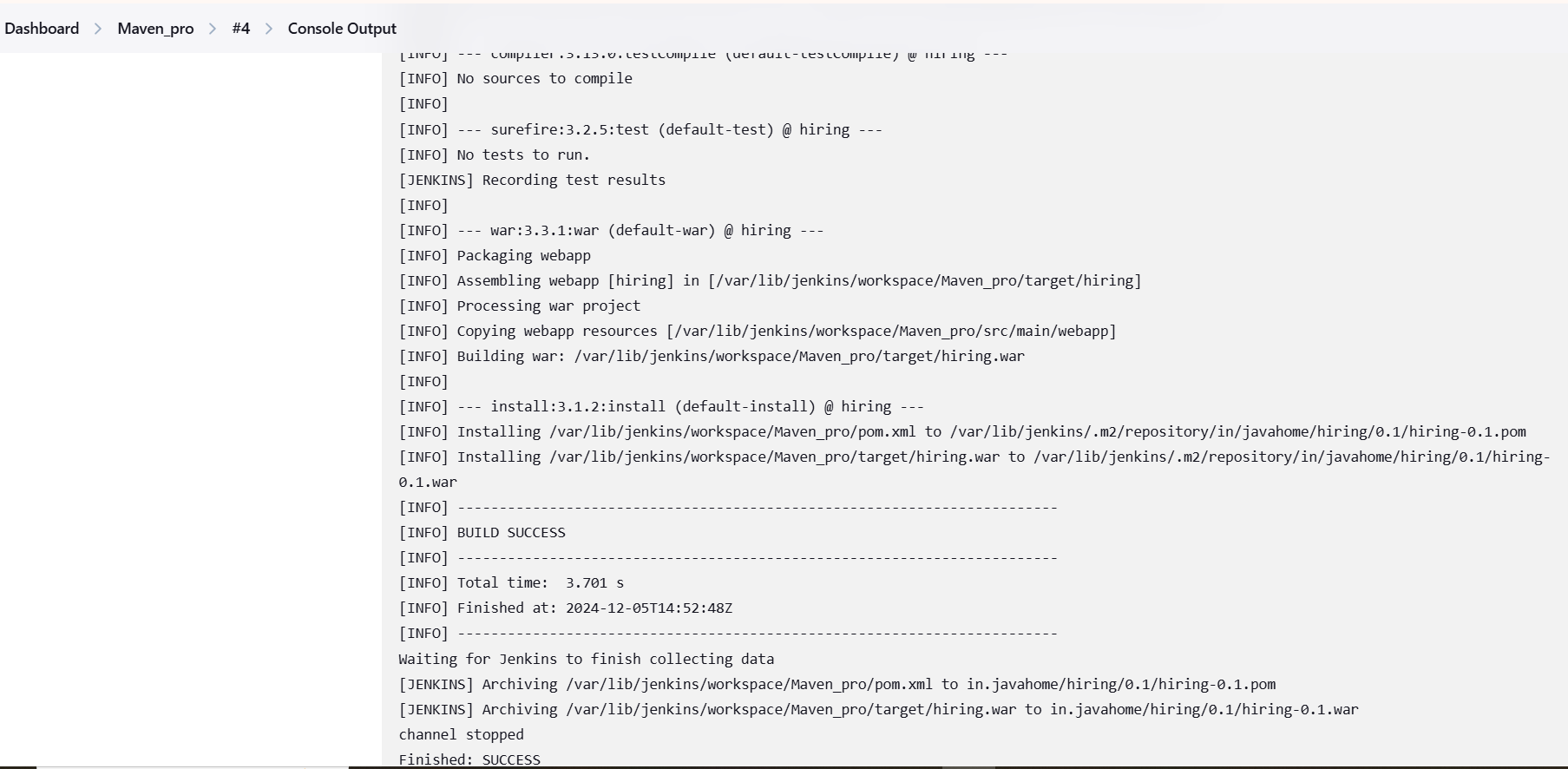
***>>>Select Maven Project:***



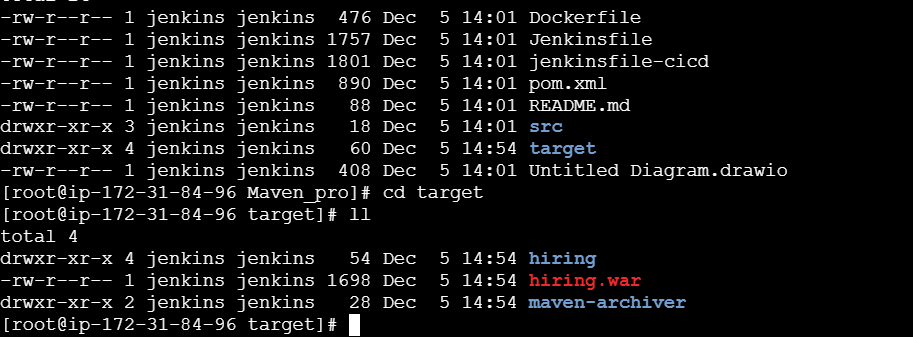


***>>>Console Output***:





***>>>Go to /var/lib/Jenkins/workspace 🡪 Maven\_pro:***

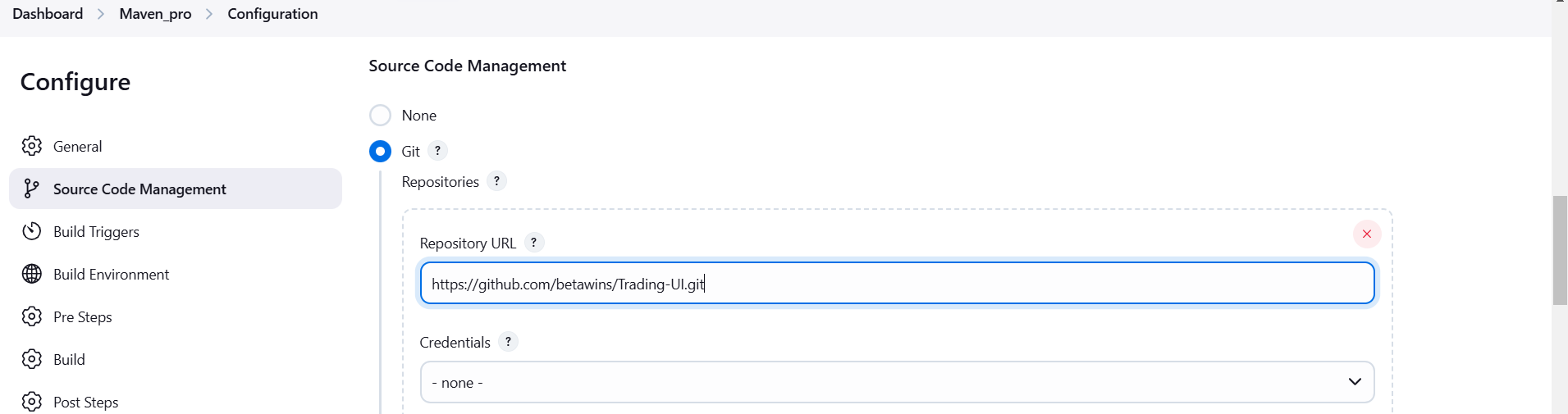


**5) Use the below code and create a parameterized job in jenkins**

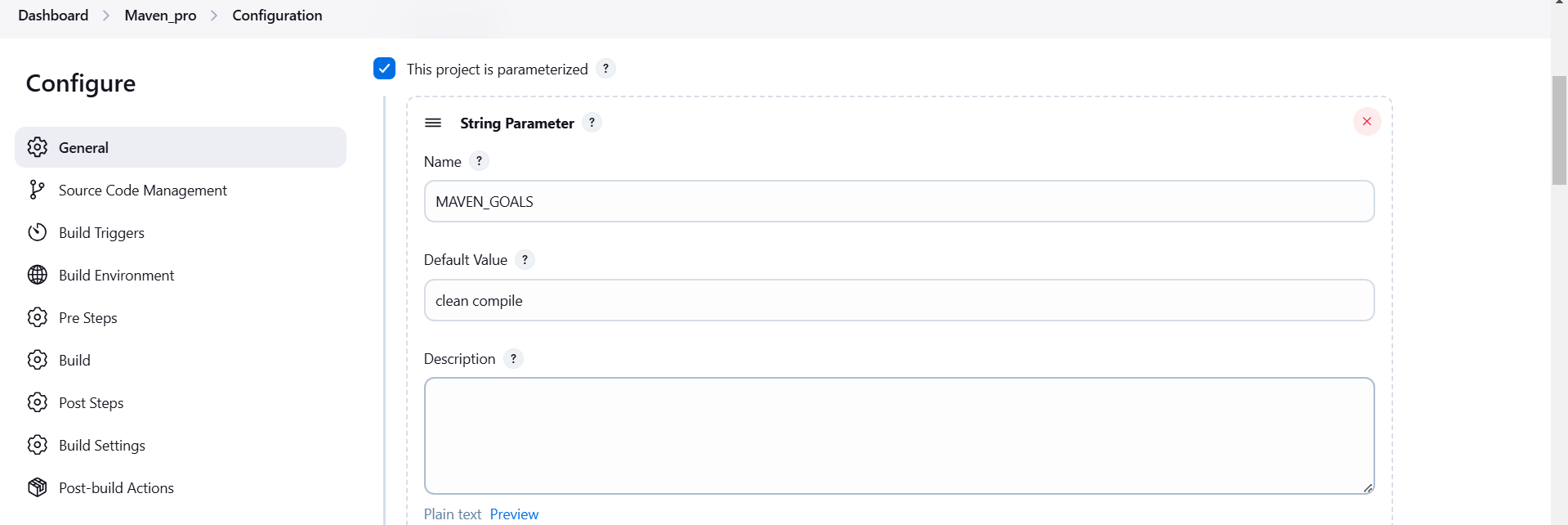
**stage 1: Git clone**

**stage 2: Maven Compilation**

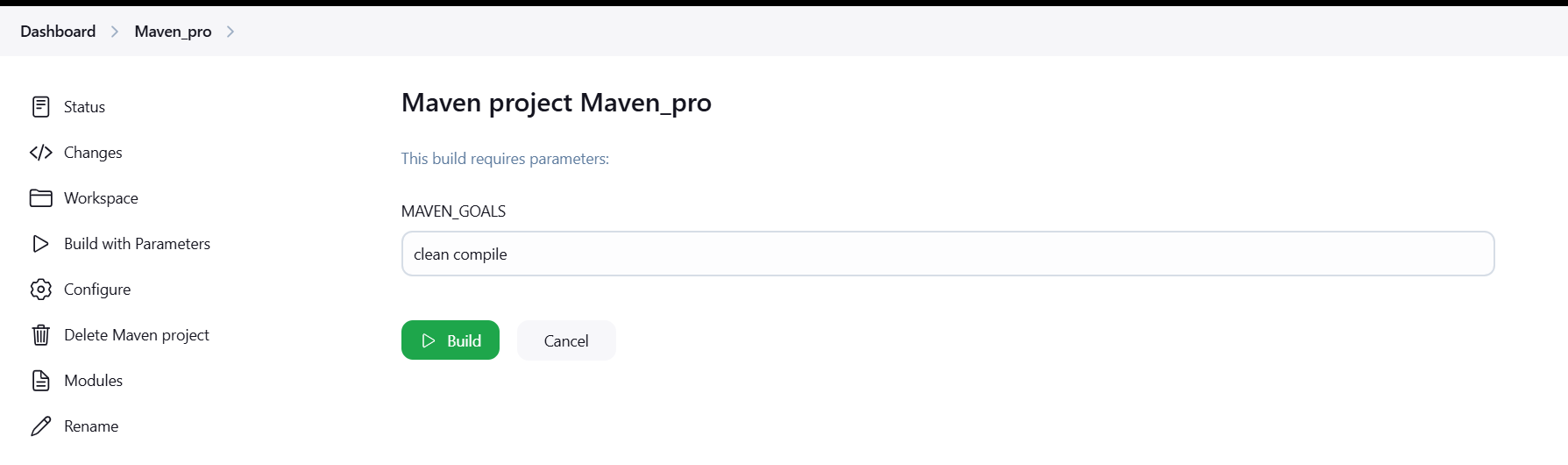
**Code**: <https://github.com/betawins/java-Working-app.git>

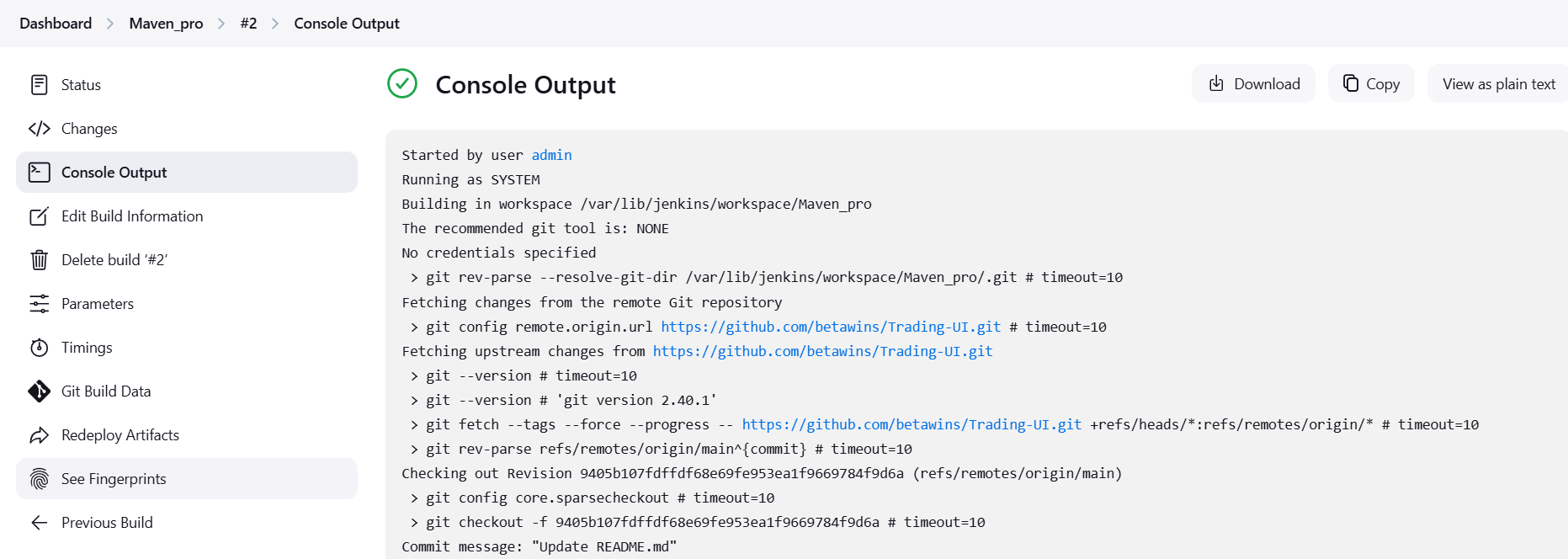


***>>>Select “This project is parameterized.”***

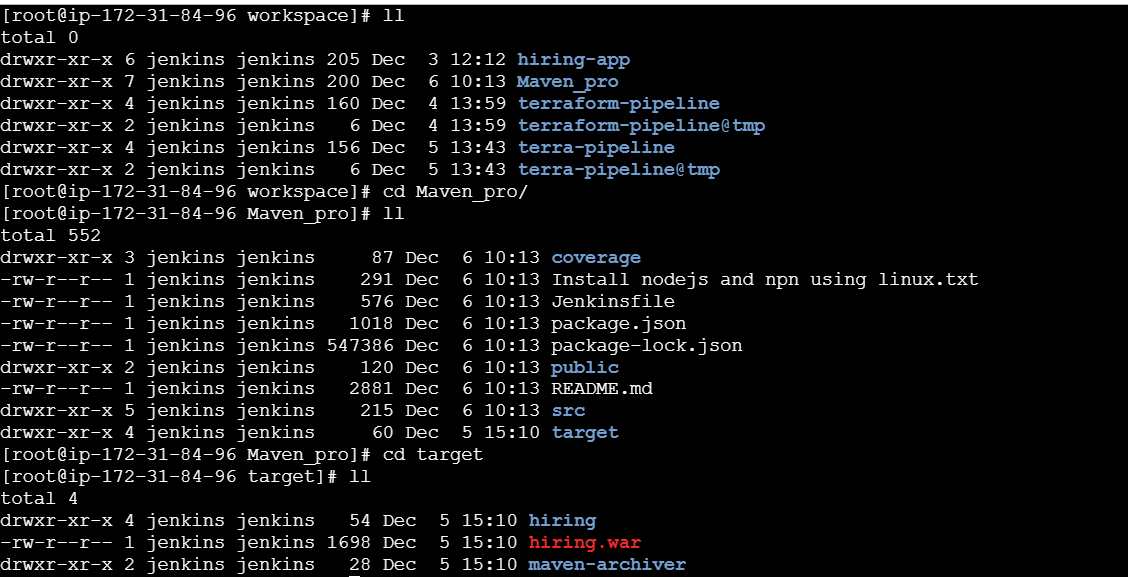


***>>>Build With Parameters:***





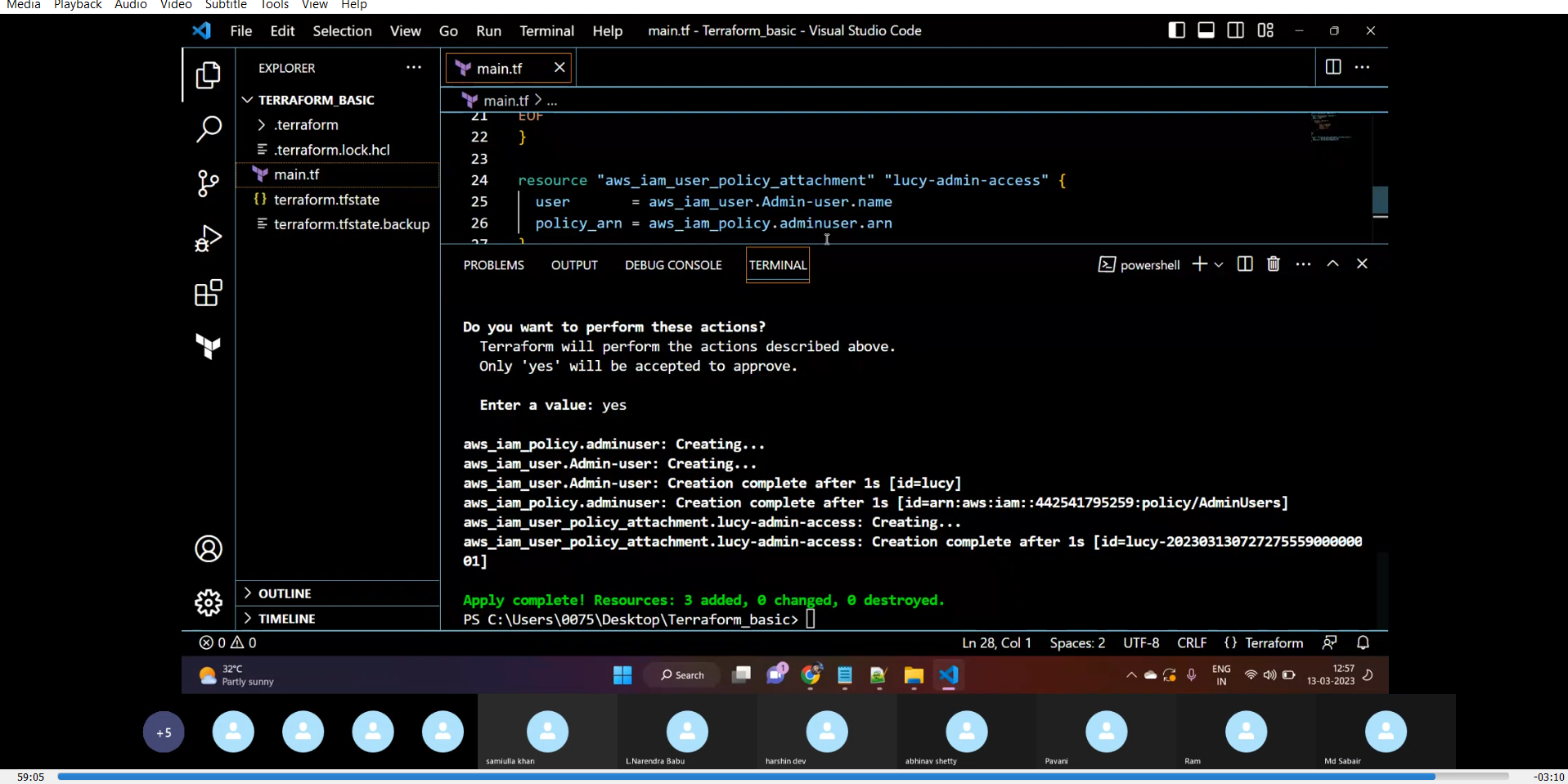


****

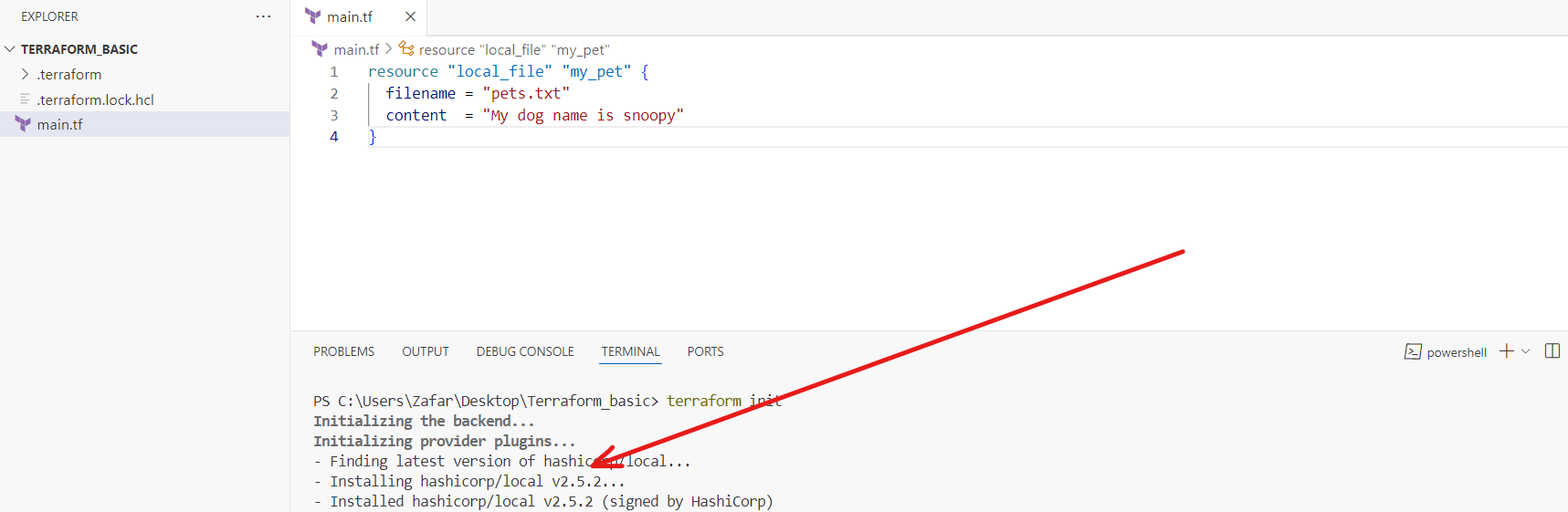
**6) What are the global varaiables in jenkins?**

1. **BUILD\_NUMBER**: This variable contains the current build number. It's automatically incremented with each build.
2. **BUILD\_ID**: A unique identifier for the current build, typically in the format "YYYY-MM-DD\_hh-mm-ss."
3. **JOB\_NAME**: The name of the current Jenkins job or project.
4. **WORKSPACE**: The directory where the current build is executed. This can be used to reference files and directories within the workspace.
5. **JENKINS**\_**HOME**: The path to the Jenkins installation directory.
6. **EXECUTOR**\_**NUMBER**: The unique number of the current build executor (e.g., 0, 1, 2).
7. **NODE**\_**NAME**: The name of the agent (slave) on which the current build is running. If the build is running on the master, this will be "master."
8. **JOB\_URL:** The URL of the current Jenkins job or project.
9. **BUILD\_URL:** The URL of the current build.
10. **BUILD\_TAG:** A unique tag for the current build, typically in the format "jenkins-${JOB\_NAME}-${BUILD\_NUMBER}."
11. **BUILD\_DISPLAY\_NAME:** The human-readable display name for the current build.
12. **BUILD\_CAUSE**: A description of the cause that triggered the current build (e.g., SCM change, manual start).
13. **CHANGE\_ID:** The ID of the specific change or commit that triggered the build (for SCM-triggered builds).
14. **CHANGE\_AUTHOR:** The author of the change or commit that triggered the build (for SCM-triggered builds).
15. **BUILD\_USER**: The username of the user who triggered the build (for builds triggered by users)

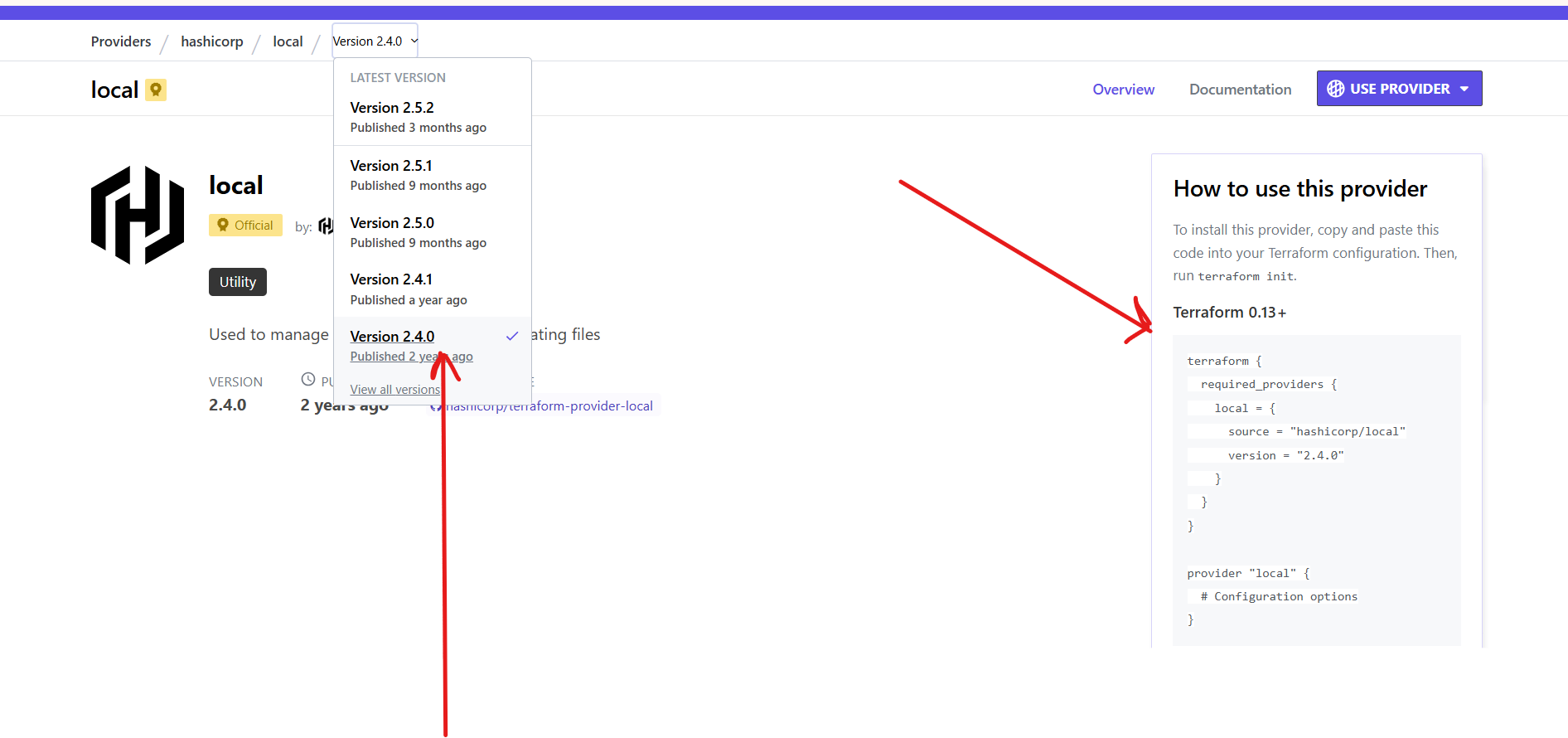
**7) Watch terraform-04 video.**

****

**8) Execute the script shown in video.**

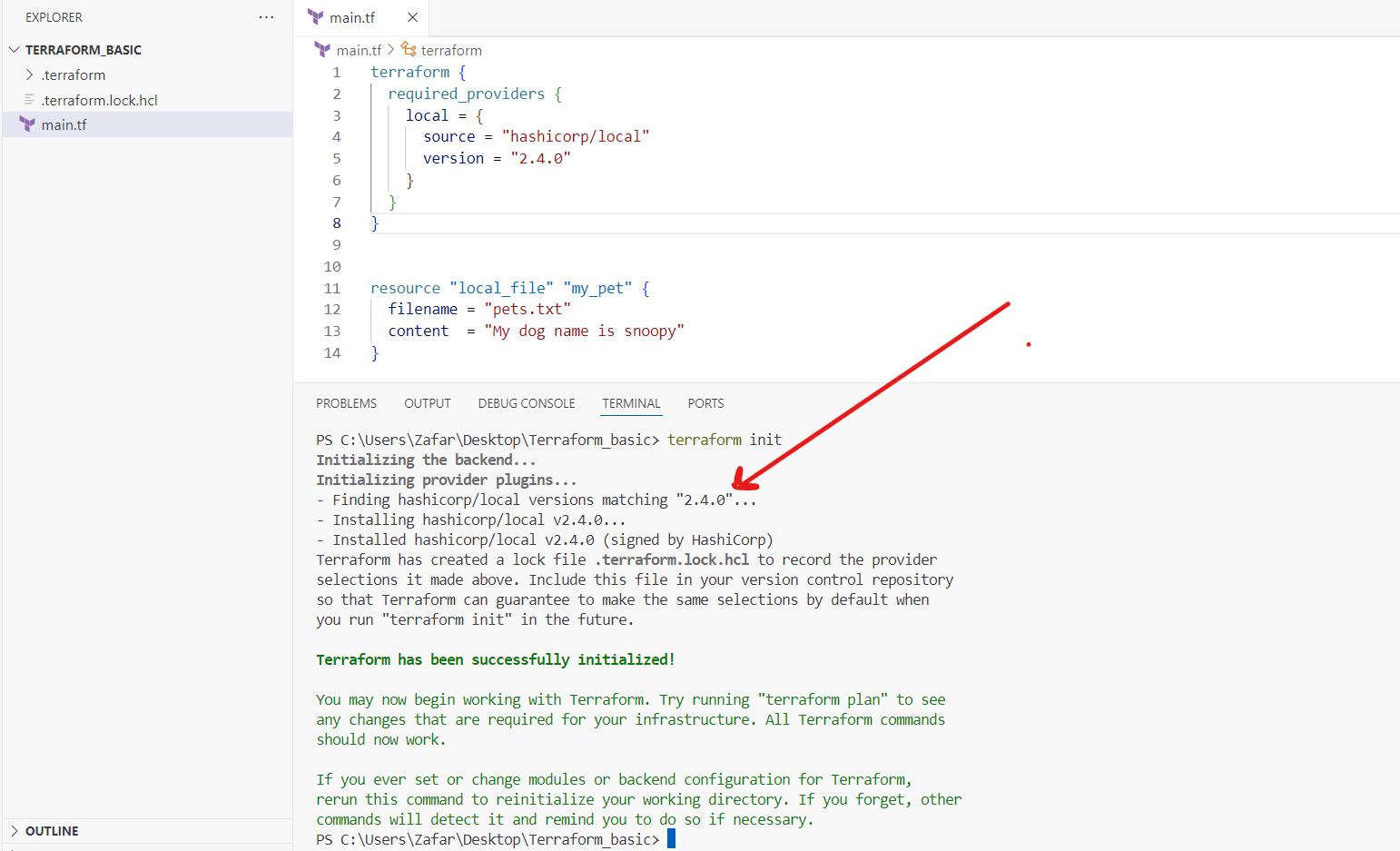
****

**Local Provider Version.**

****

Choosing Version 2.4.0

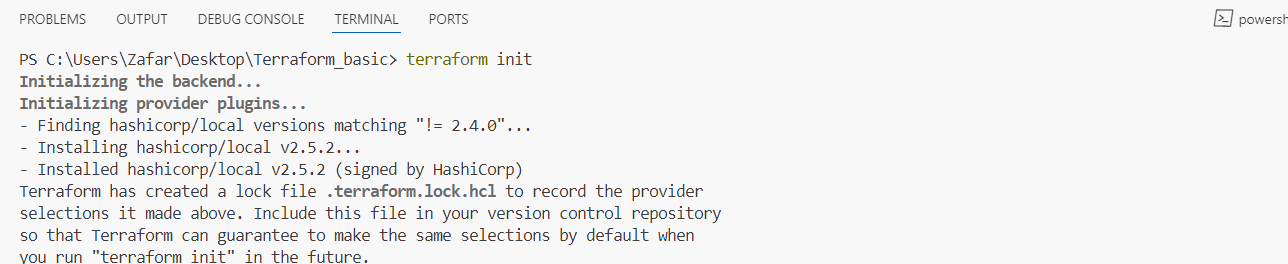
**Terraform Block**

****

**Version 2.4.0**

**>>>Will not use the mentioned version: !=2.4.0**

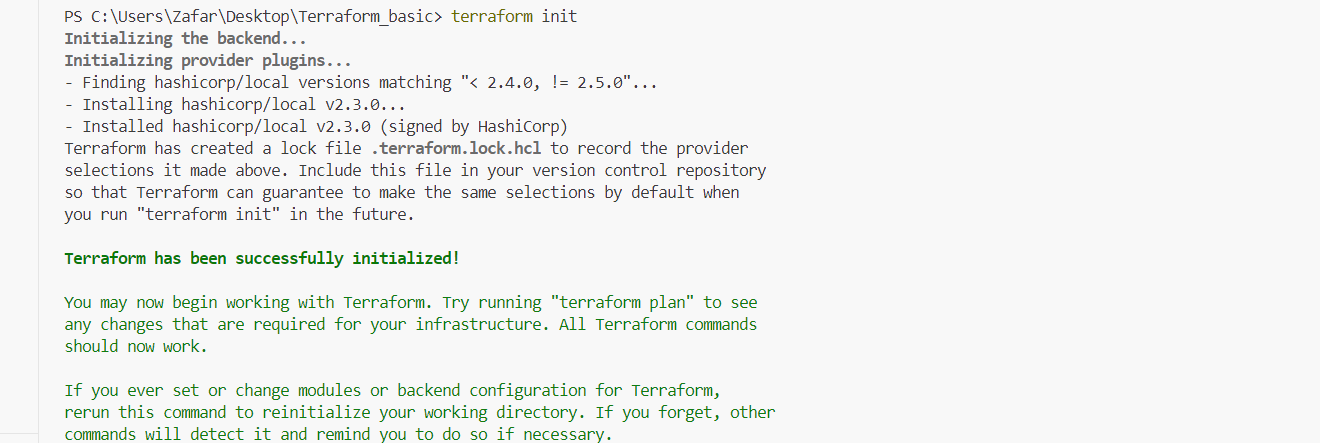
****

****

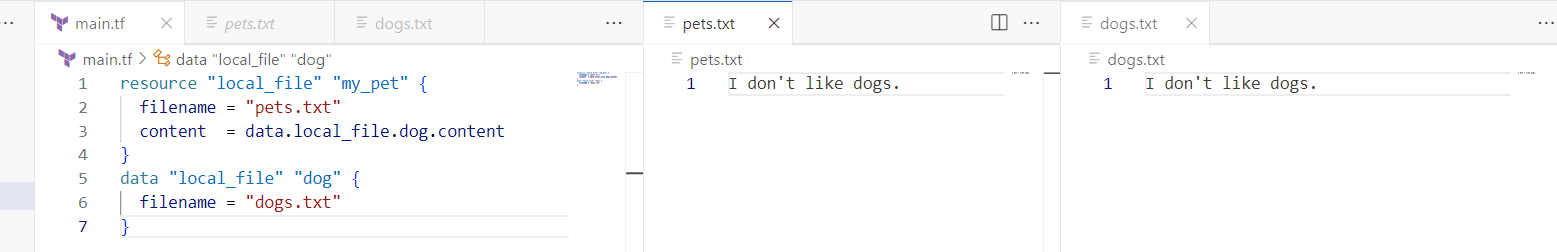
**>>>The Version should not be equal to 2.5.0 and it should be lesser than 2.4.0:**

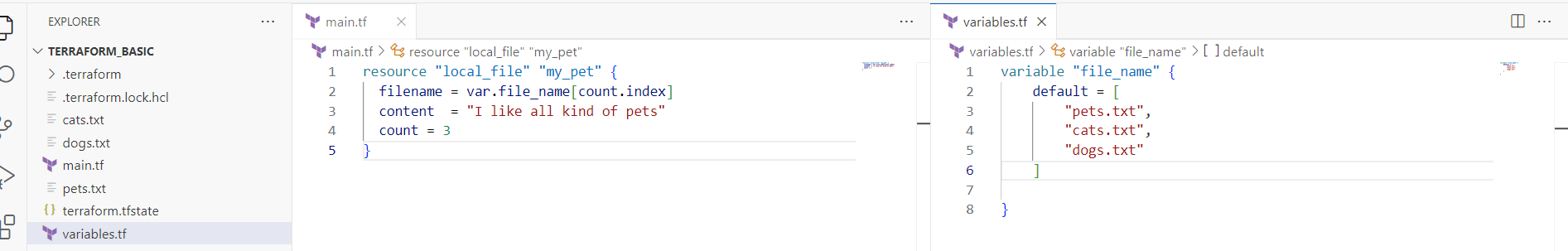
**>version = "!=2.5.0, <2.4.0"**

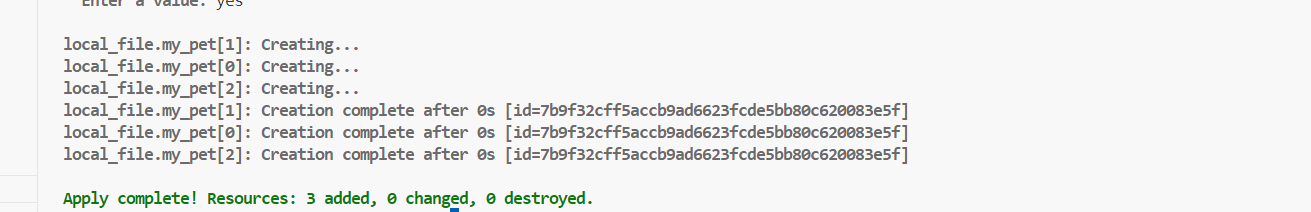
****

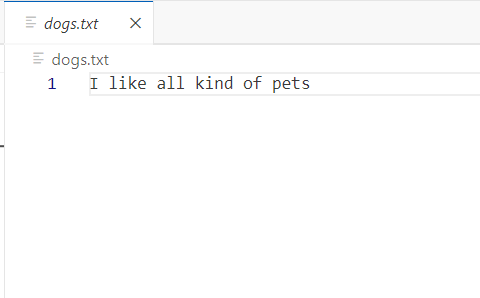
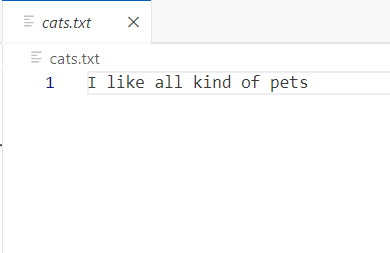
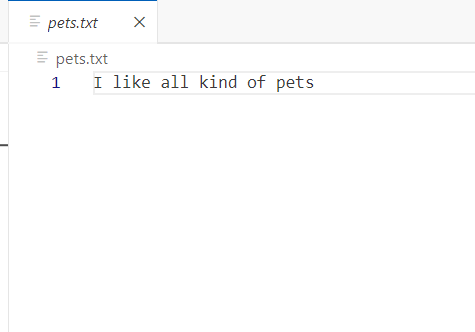
****

**>>Using Data Sources (Data Block):**

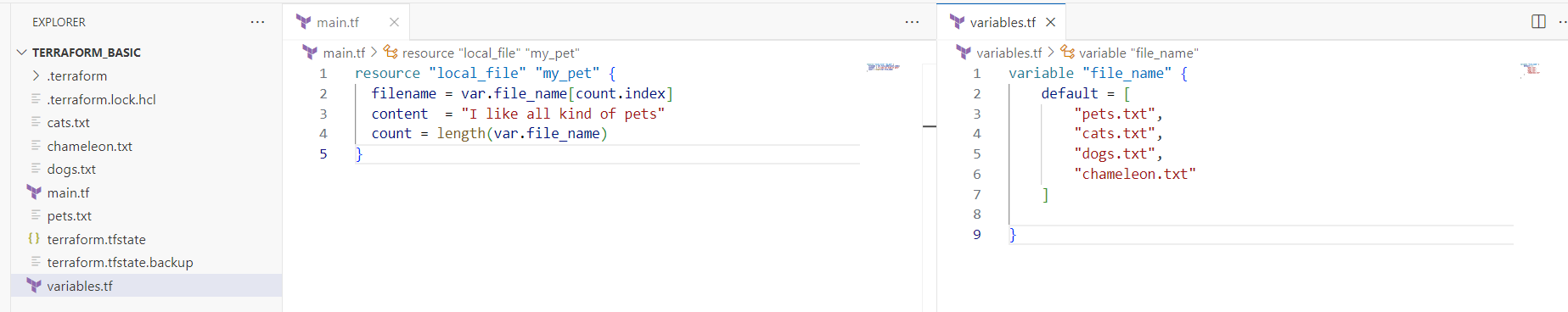
****

****

****

**** **** 

**>> Using Meta argument🡪  count = length(var.file\_name)**

****

**Created all the 4 files.**



**>> Using Meta argument 🡪 filename = each.value**

**for\_each = toset(var.file\_name)**

****

**>> Creating an IAM user:**

resource "aws\_iam\_user" "Admin-user" {

  name = "Lucy"

  tags = {

    "description" = "Technical Team Lead"

  }

}

resource "aws\_iam\_policy" "adminuser" {

  name   = "AdminUsers"

  policy = <<EOF

{

    "Version": "2012-10-17",

    "Statement": [

        {

            "Sid": "1234567890",

            "Effect": "Allow",

            "Action": "\*",

            "Resource": "\*"

        }

    ]

}

EOF

}

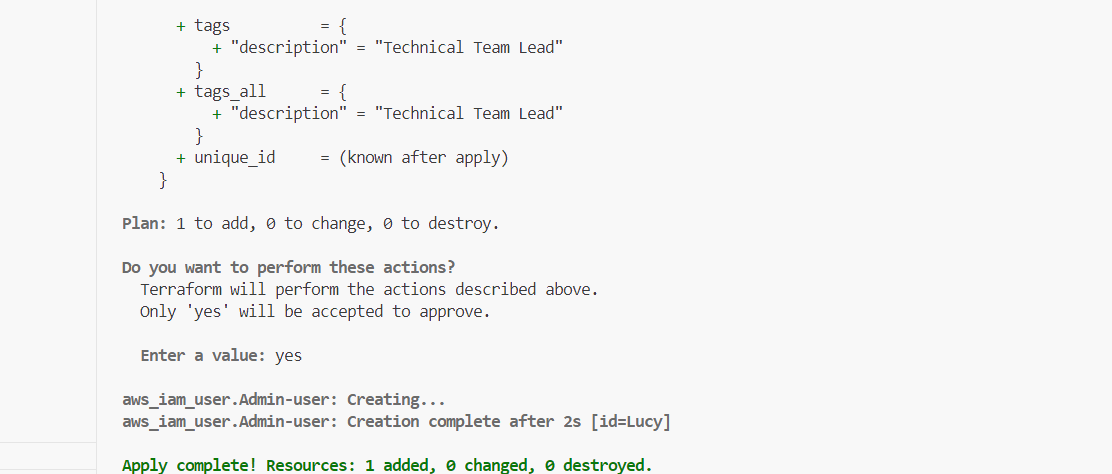
resource "aws\_iam\_user\_policy\_attachment" "lucy-admin-access" {

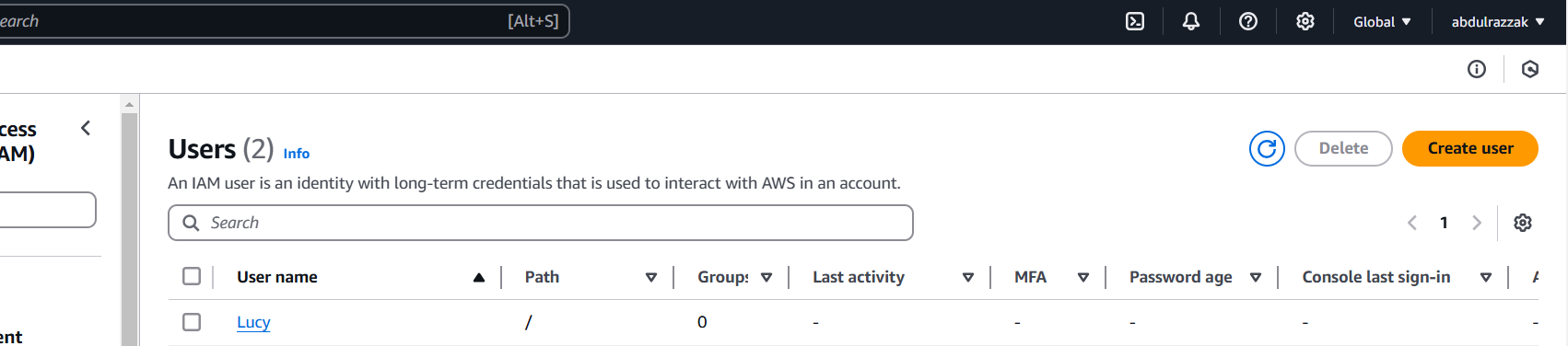
  user       = aws\_iam\_user.Admin-user.name

  policy\_arn = aws\_iam\_policy.adminuser.arn

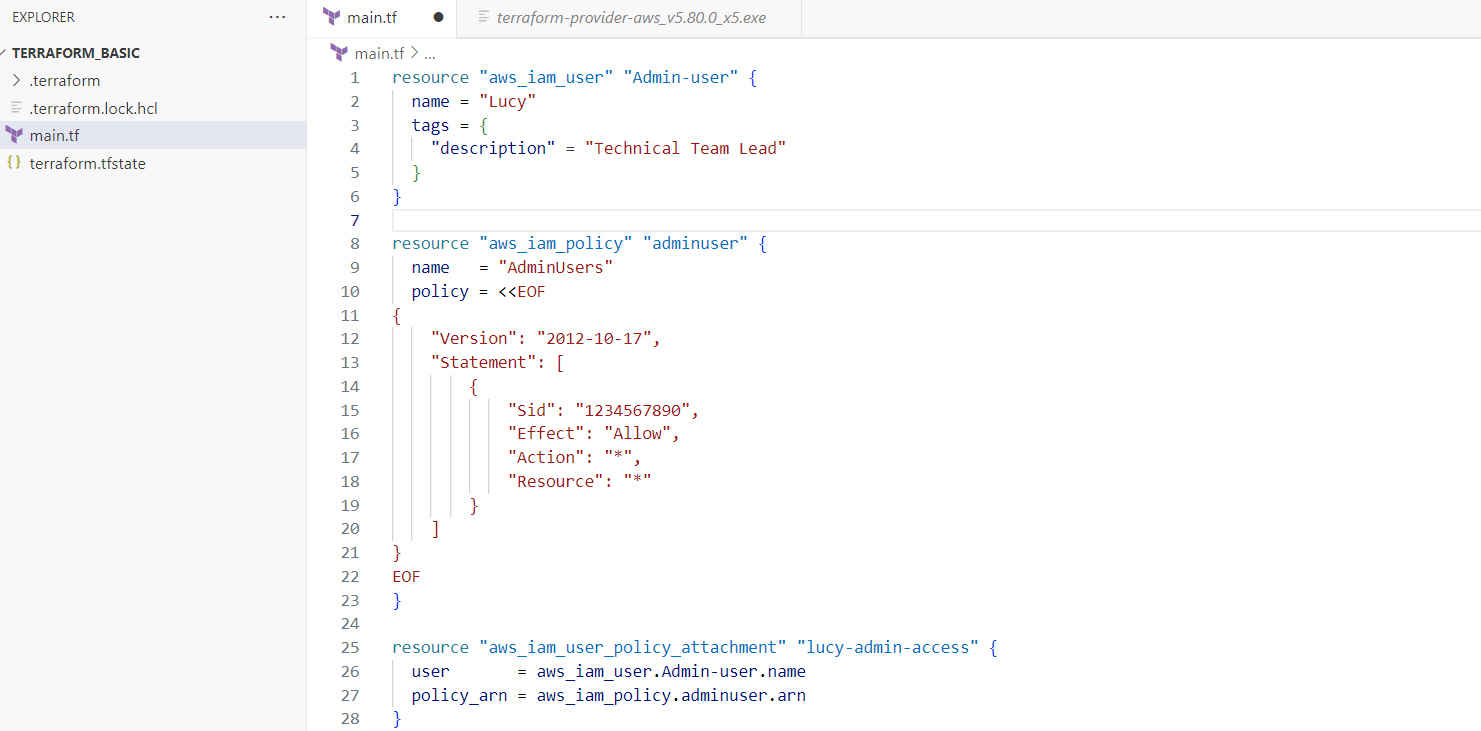
}

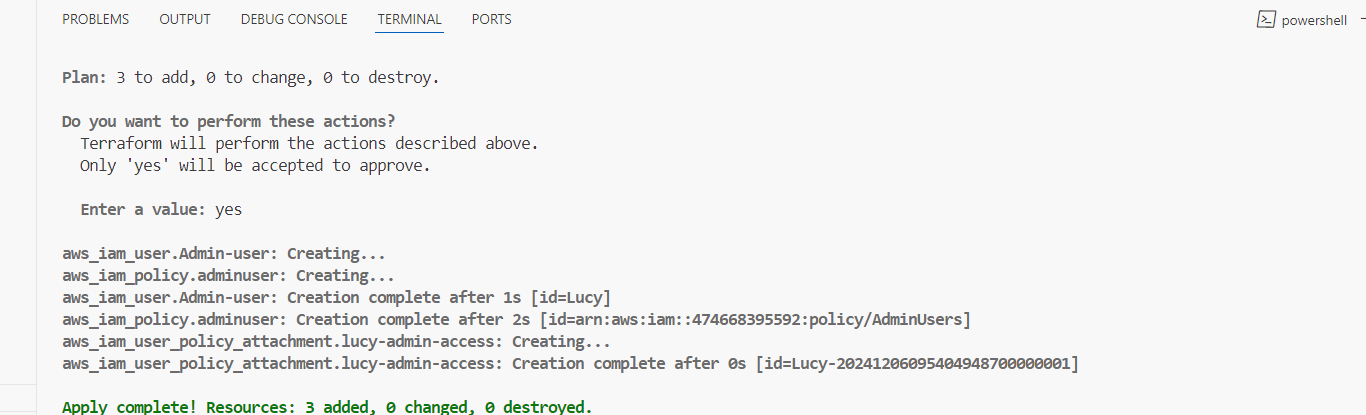
****

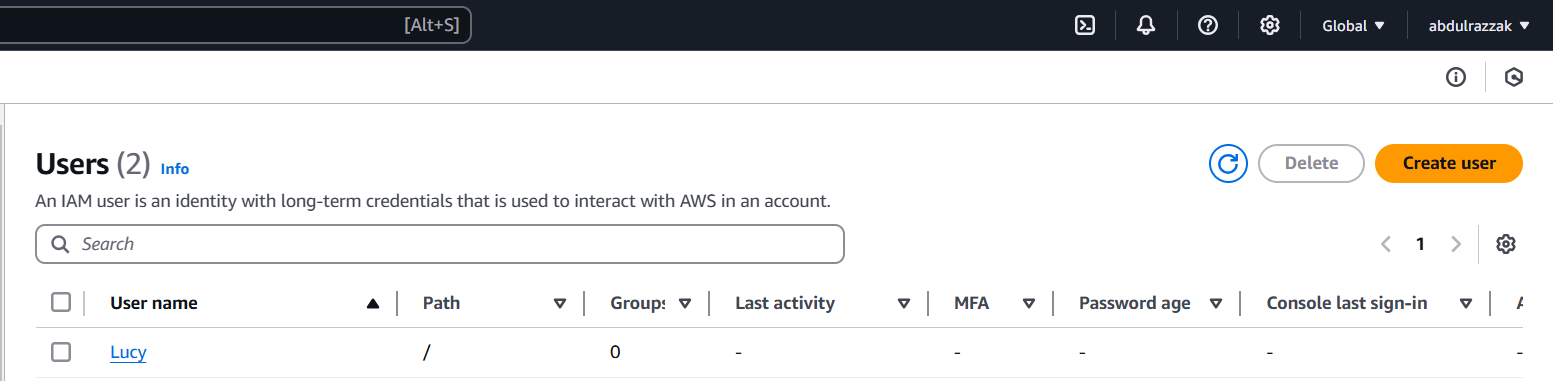
****

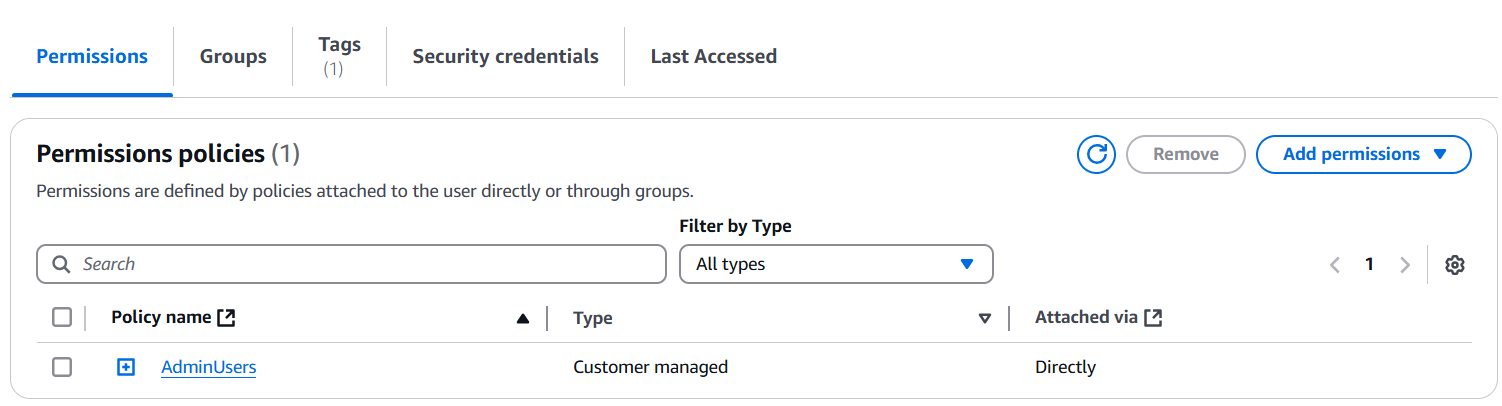
****

**>>>Attaching permission policy to the User:**

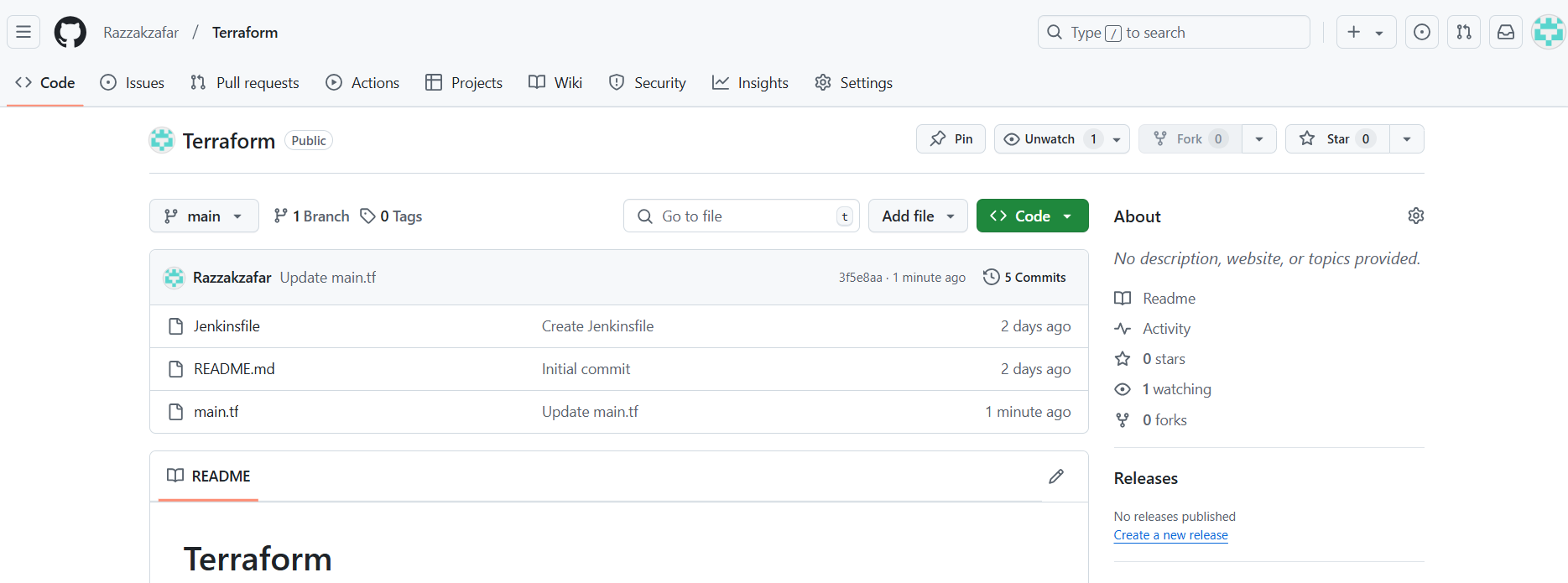
****

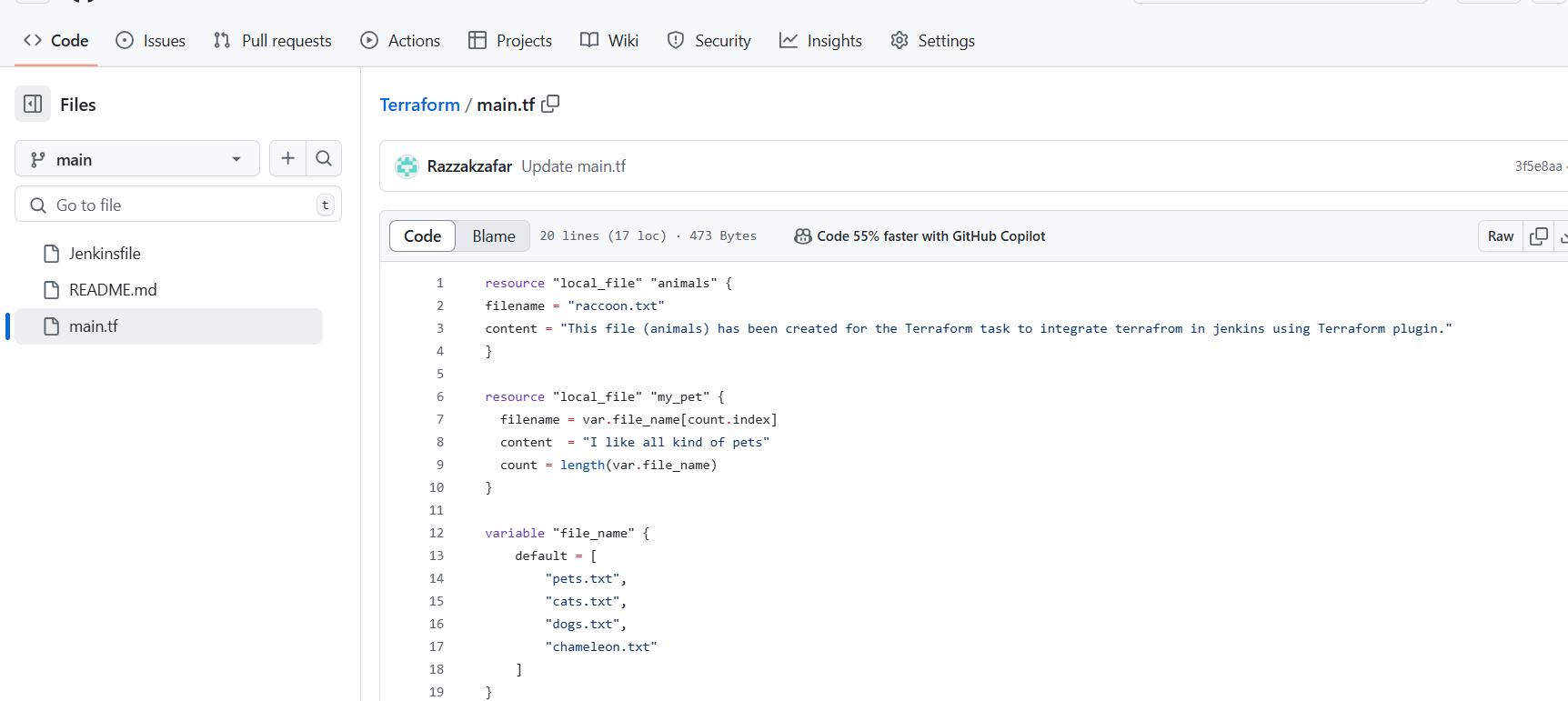
****

****

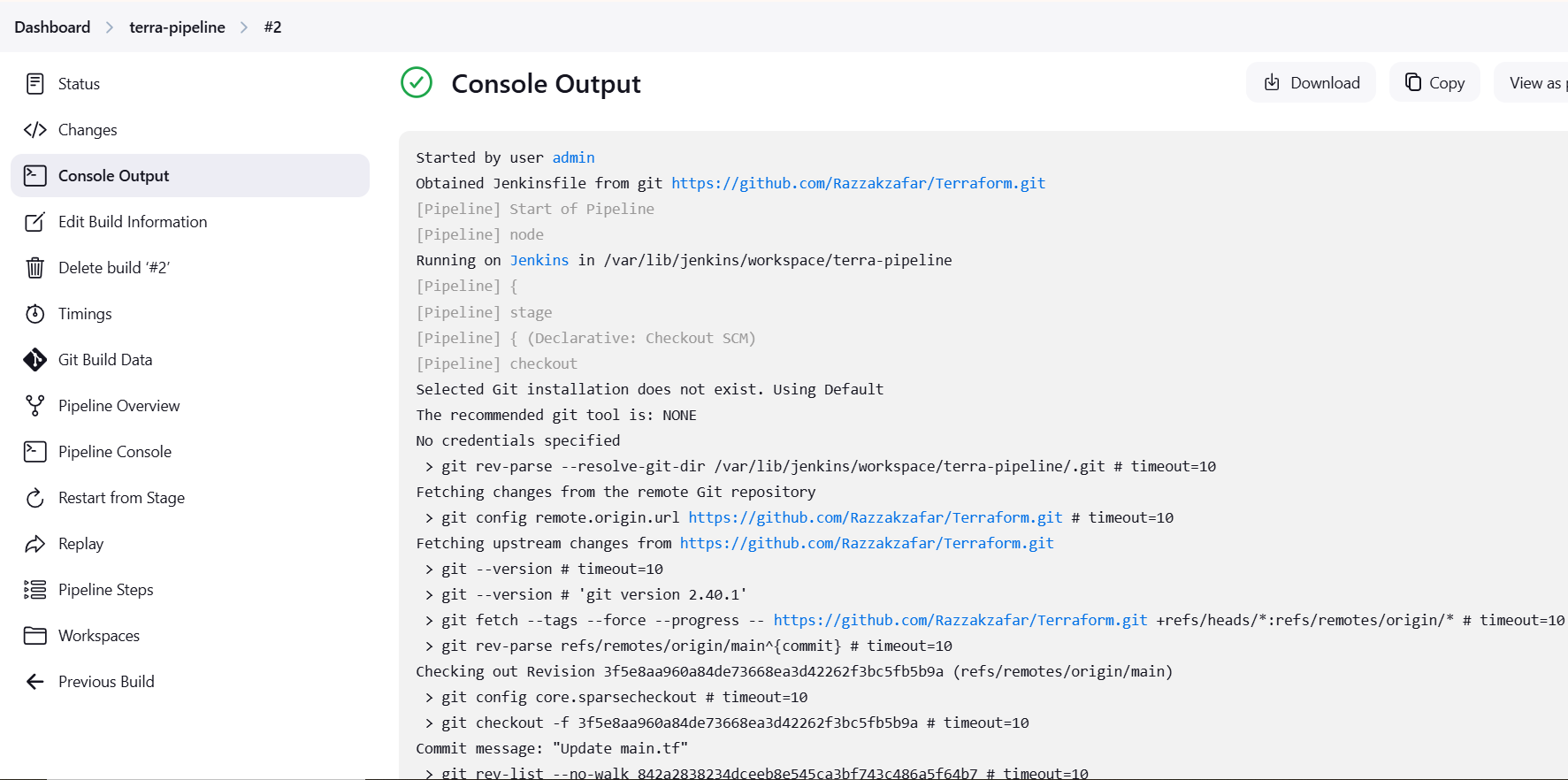
****

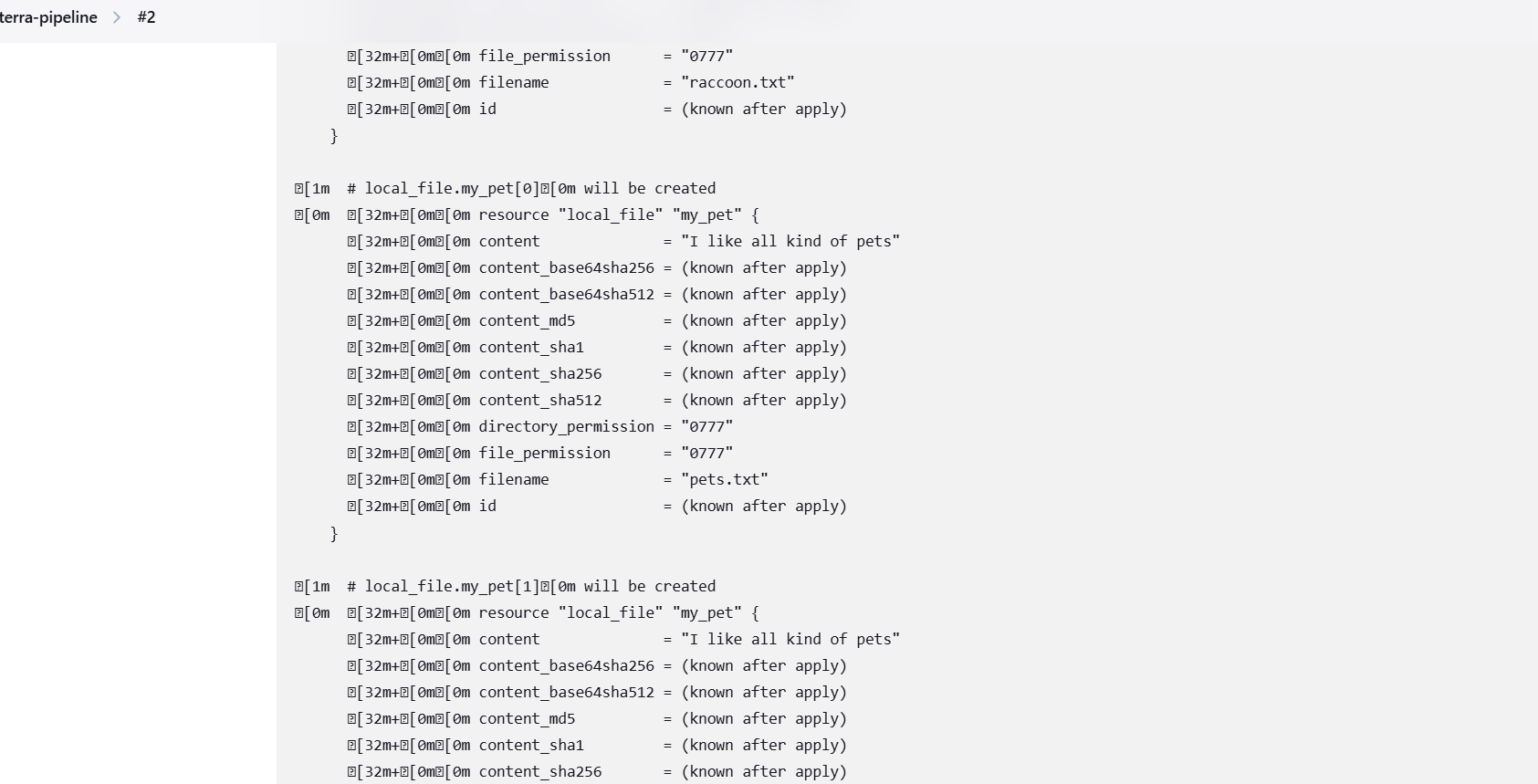
**9) Integrate terraform in Jenkins using Terraform plugin.**

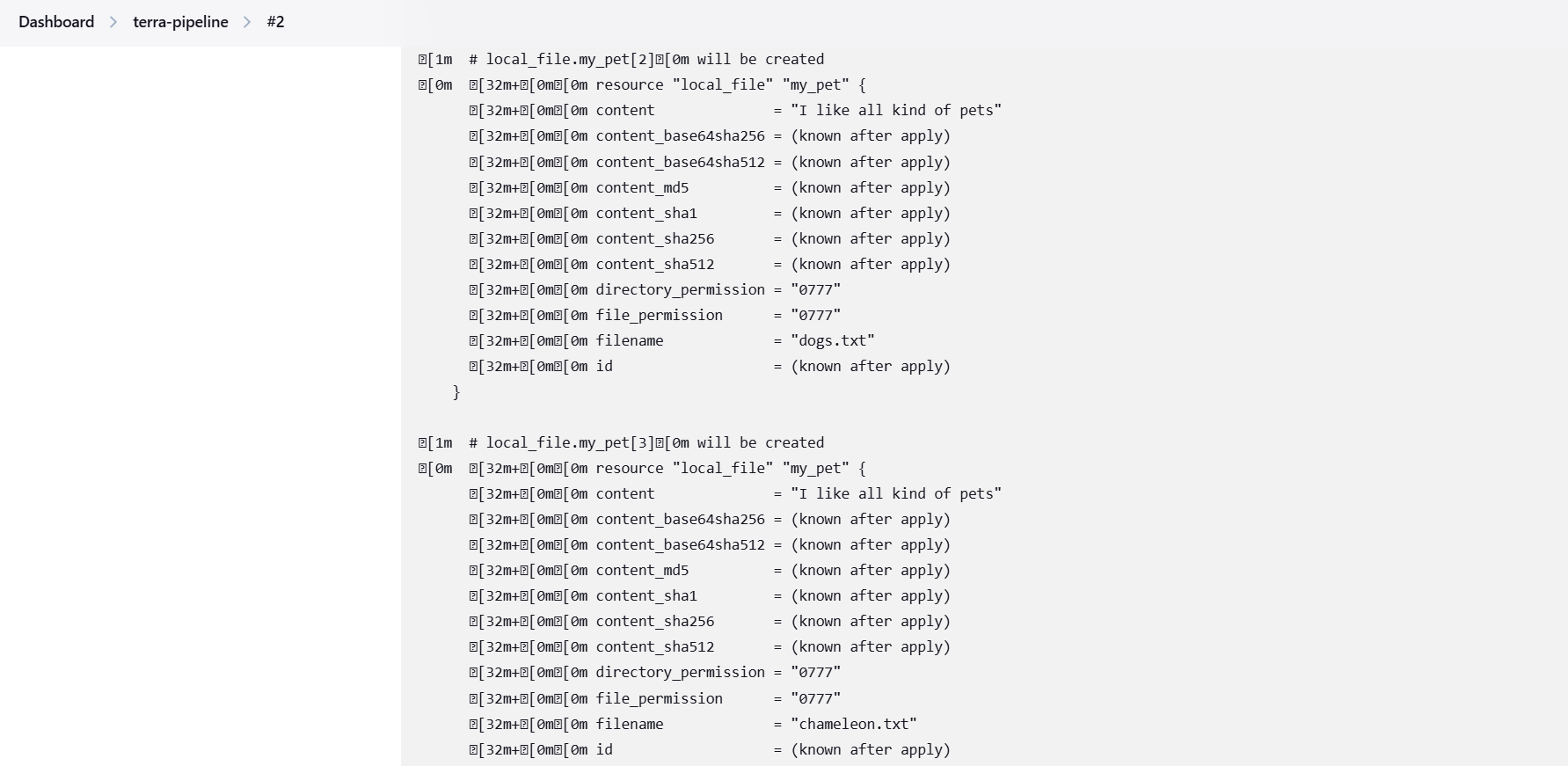
****

****

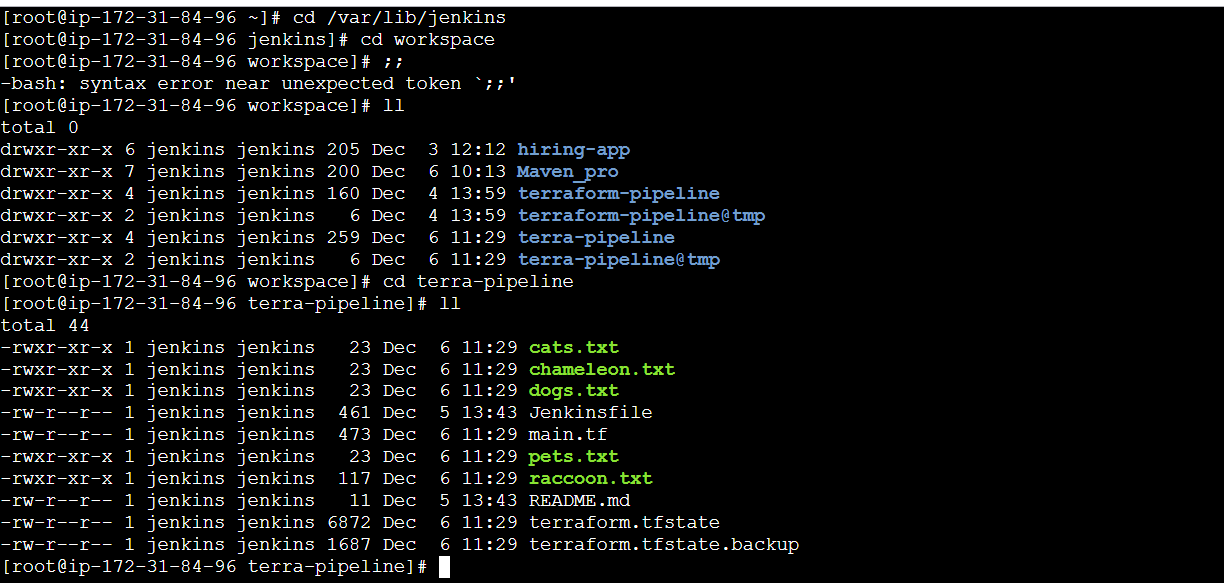
****

****

****

****



****

**11) Explain 10 Maven commands.**

**1. mvn clean: This command cleans the project by deleting the target directory, which contains compiled classes and build artifacts. It's useful when you want to start fresh with your build.  
2. mvn compile: Compiles your project's source code (located in the src/main/java directory by default) into bytecode. The compiled classes are placed in the target/classes directory.  
3. mvn test: Runs the tests in your project. Test classes should be placed in the src/test/java directory. Maven uses testing frameworks like JUnit to execute the tests.  
4. mvn package: Packages your project into a distributable format, like a JAR or WAR file. It compiles the code, runs tests, and creates the artifact in the target directory.  
5. mvn install: Builds the project, runs tests, and installs the artifact in the local Maven repository. This makes it available for other projects on the same machine to use.  
6. mvn dependency:tree: Generates a tree-like view of your project's dependencies, showing their relationships and versions. It helps you understand the entire dependency hierarchy.  
7. mvn clean install: A common combination of commands. It cleans the project, then builds, tests, and installs the artifact.  
8. mvn clean package -DskipTests: Builds the project and packages it, skipping the test execution. This is useful when you want a quick build without running tests.  
9. mvn dependency:resolve: Resolves and downloads project dependencies without building the project. This ensures that required dependencies are available locally.  
10. mvn archetype:generate: Generates a new Maven project from a predefined template (archetype). This is helpful to quickly set up a project structure with common configurations.**