**Terraform task – 03  
===================**

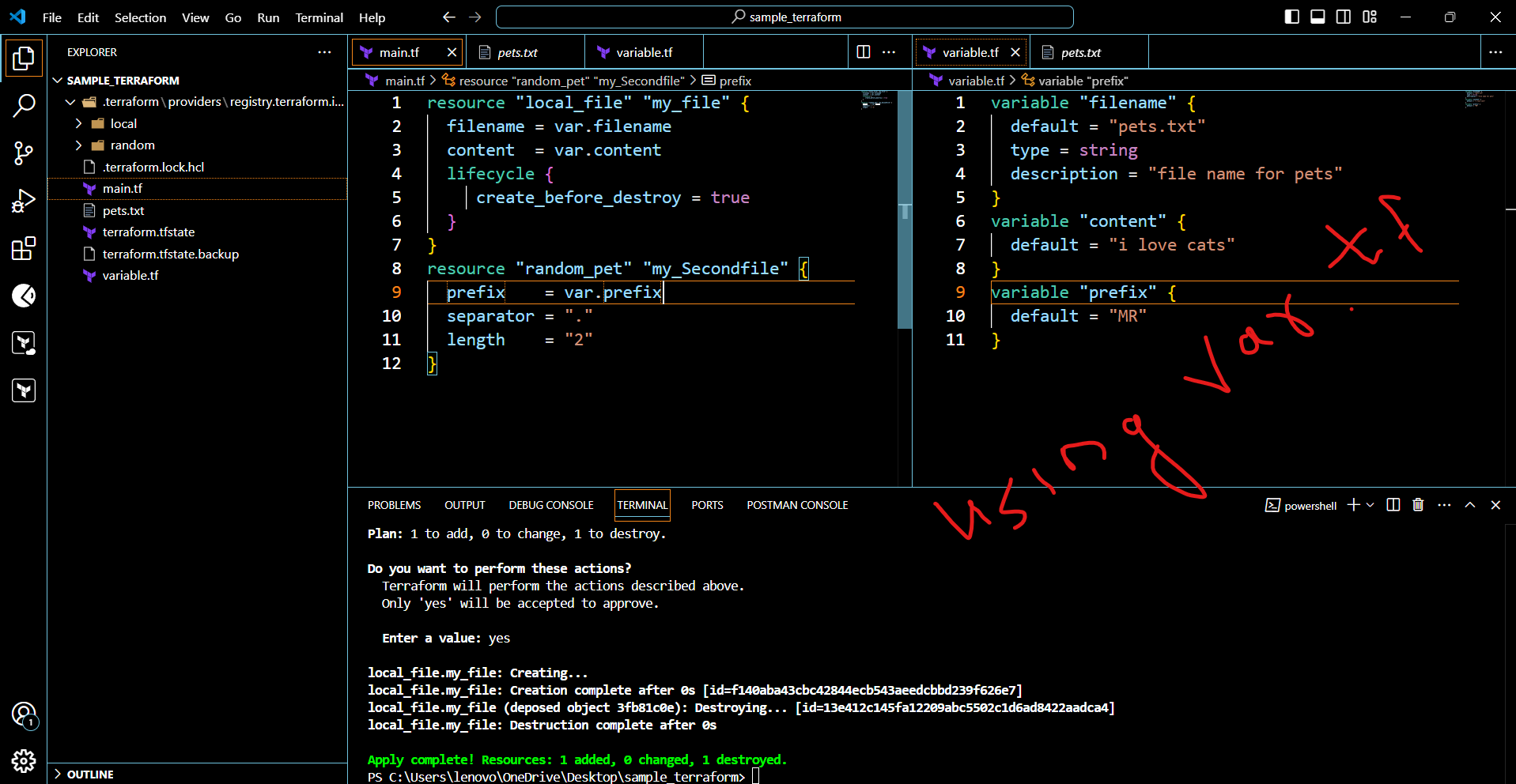
**1) Watch terraform-03 video.**

Completed

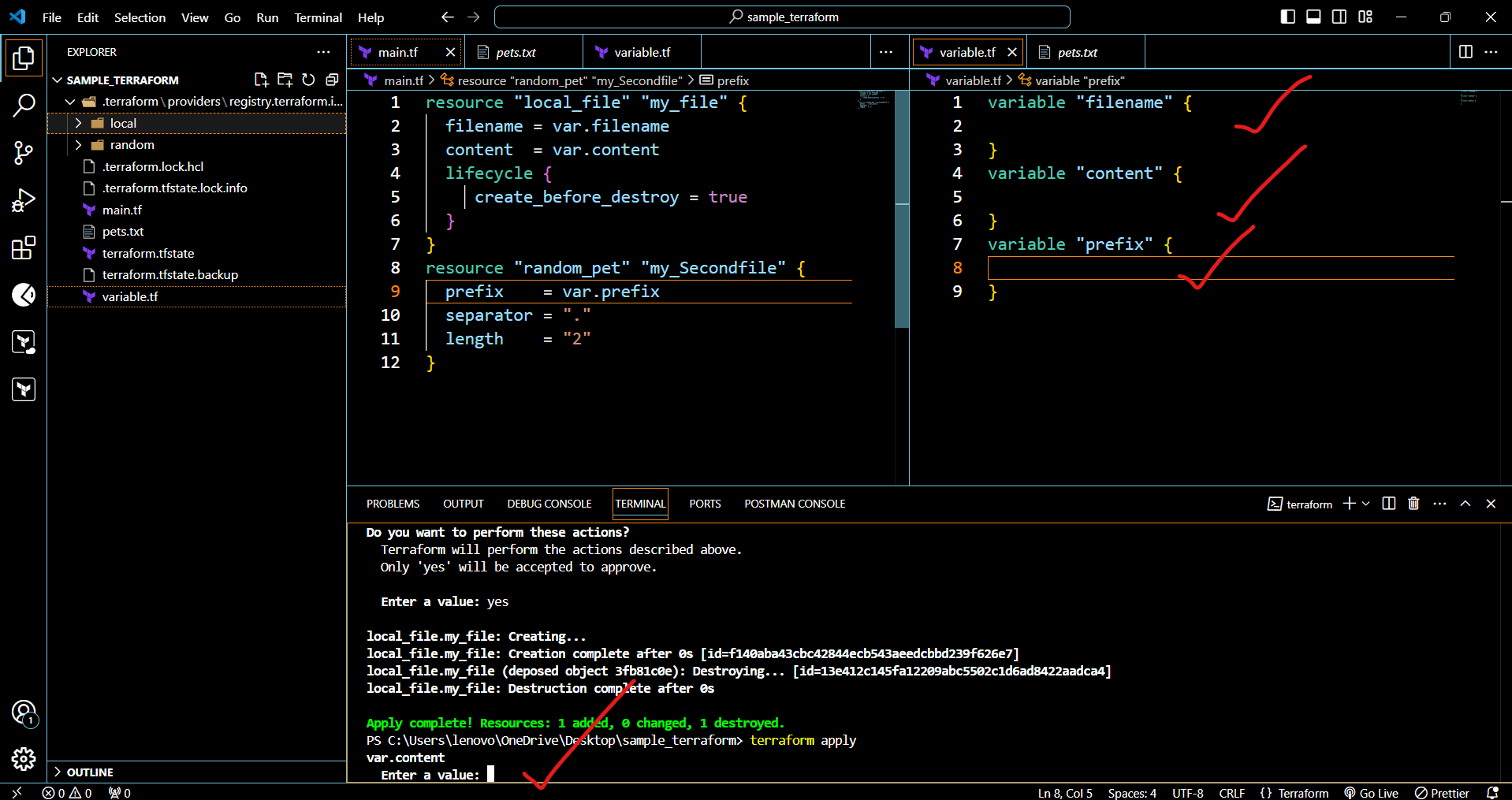
**2) Execute the script shown in video.**

**Using of variables :  
===============**

**1) By using variables.tf file**

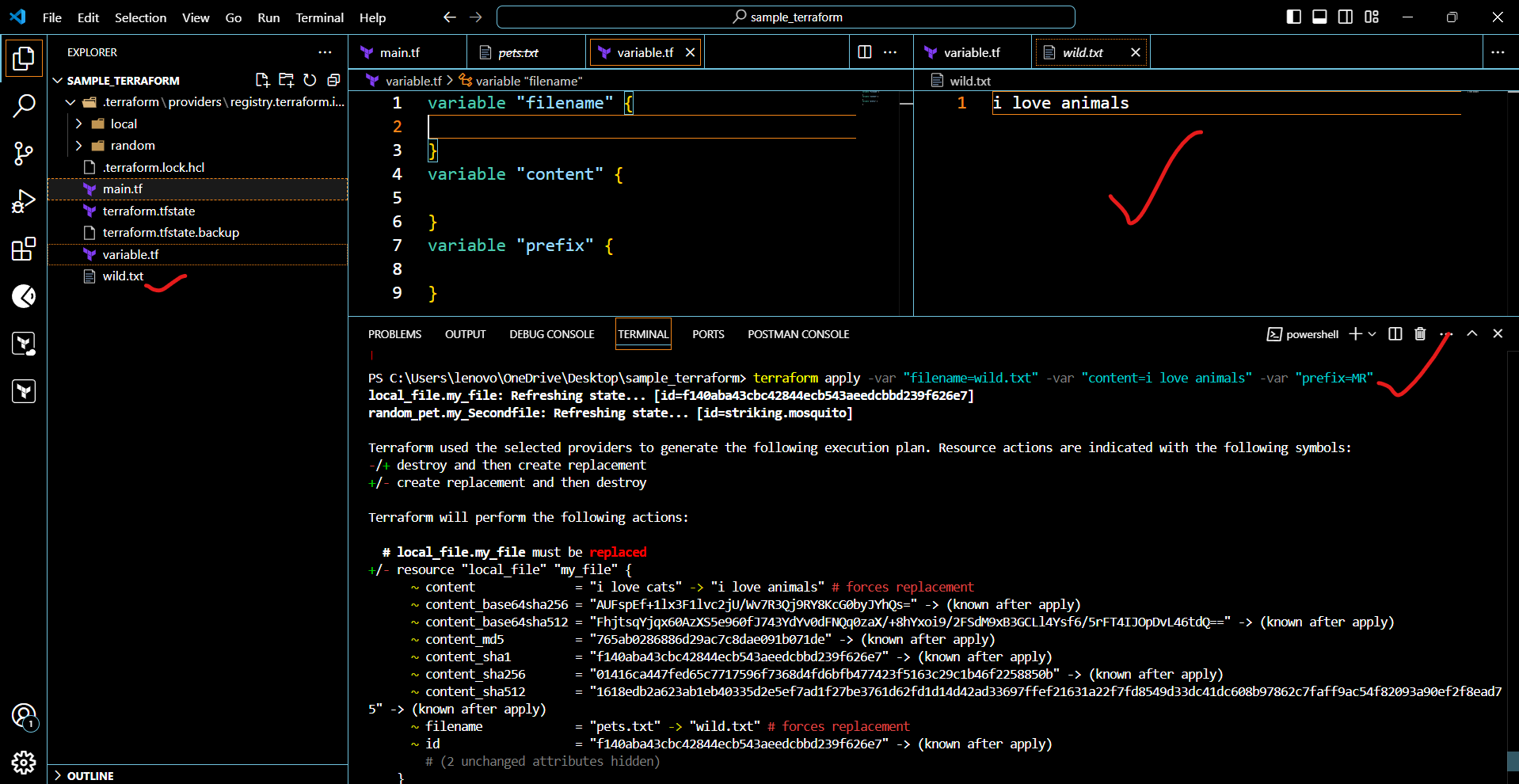


**By using interactive mode (This will get activated if we dont pass defa;ult value in variable.tf file)**

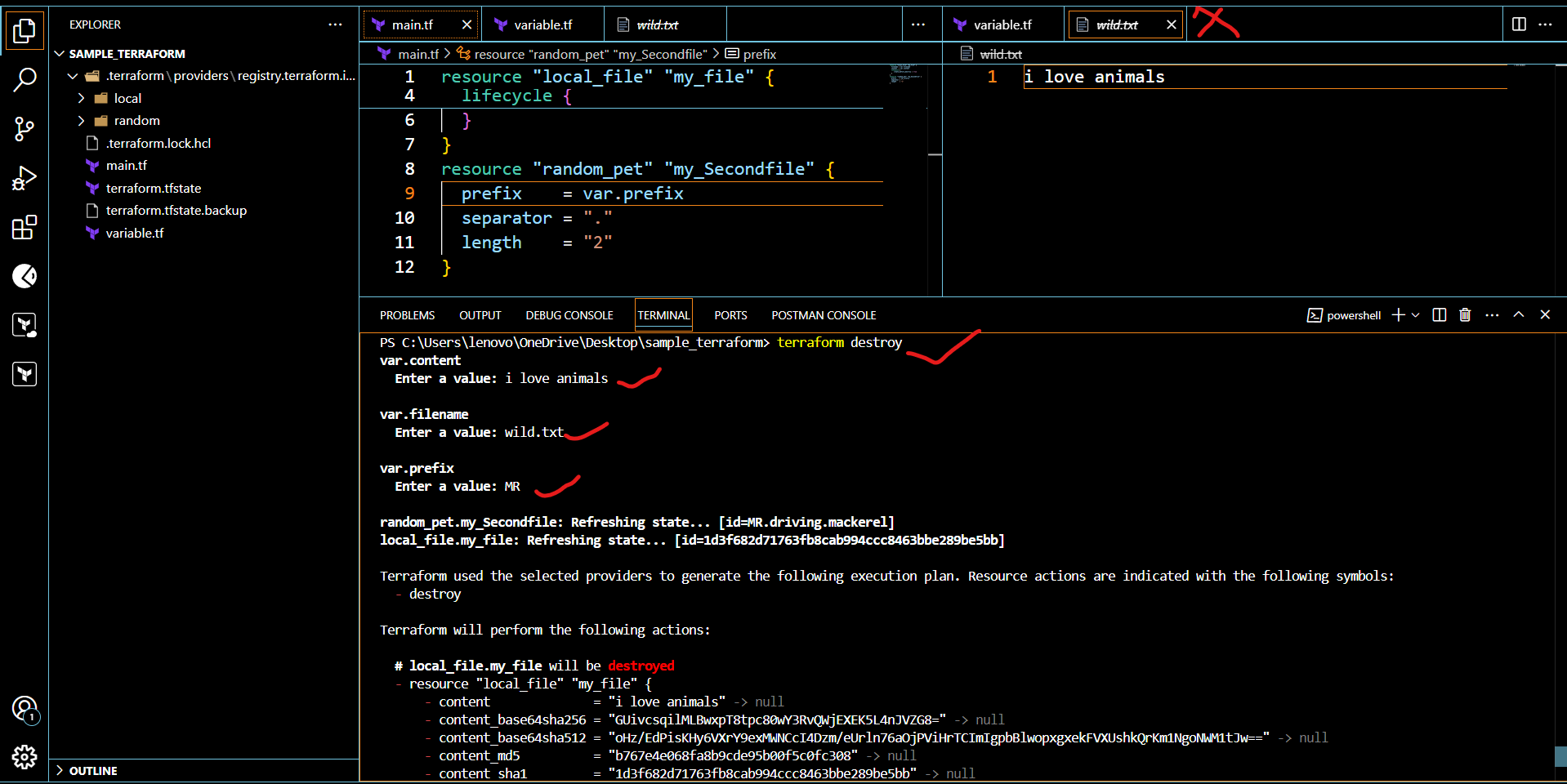


**Command line flags**

**--> terraform apply - var "filename=/root/pets.txt" -var "prefix=MR"**

****

**Terraform destroy**

****

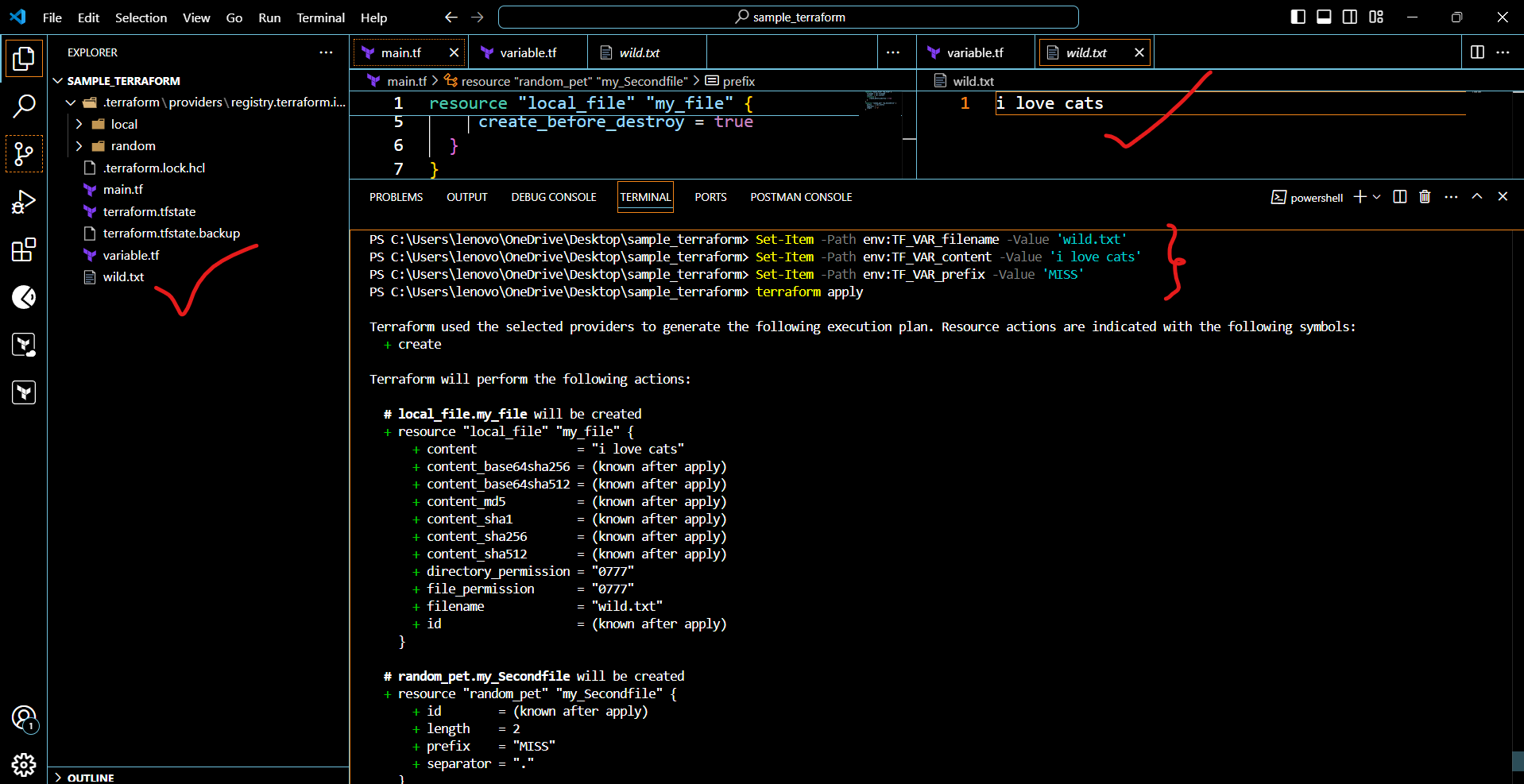
**Environment variables**

**--> export TF\_VAR\_filename="/root.pets.txt" [linux]**

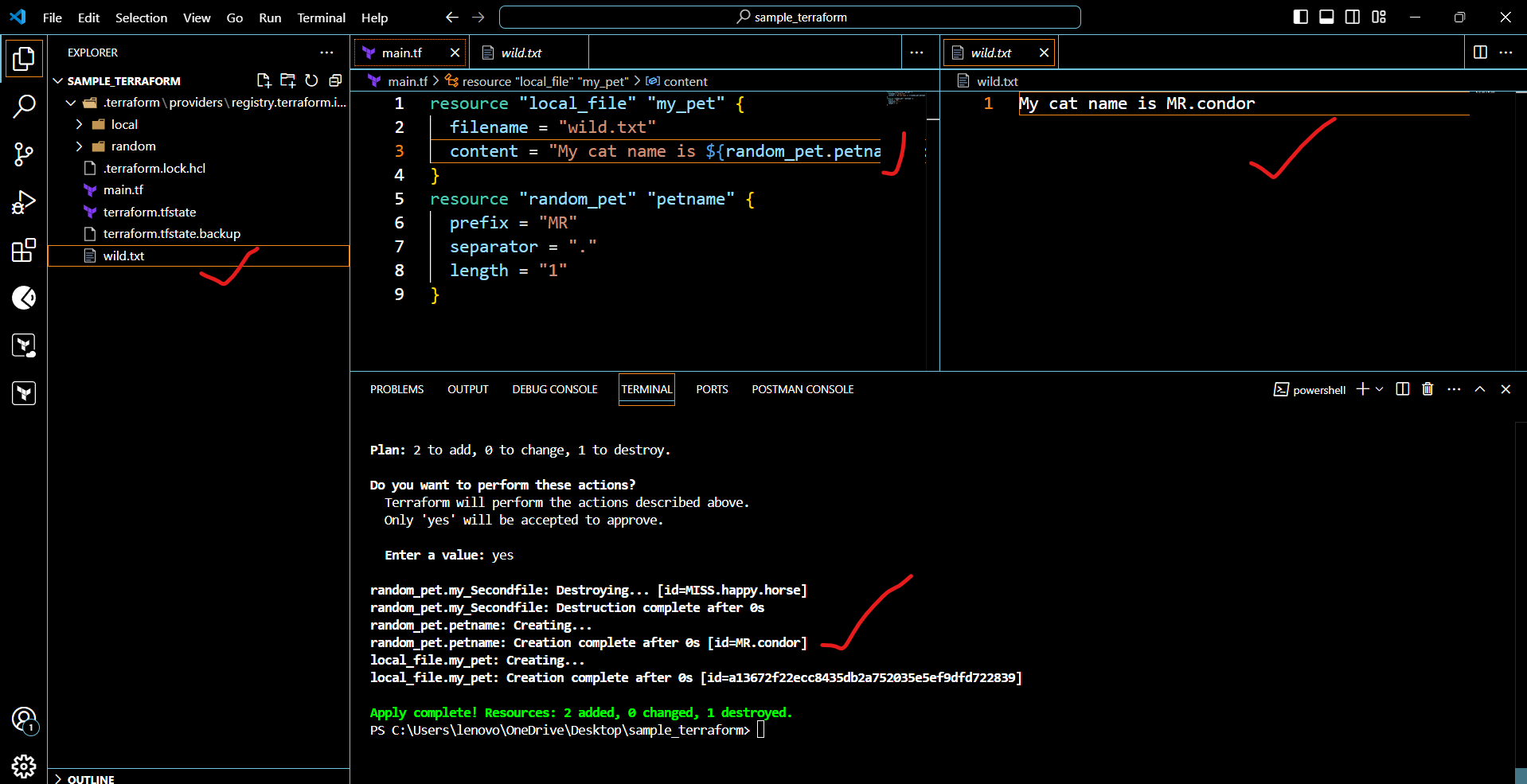
**--> export TF\_VAR\_prefix= "MR" [windows]**

**--> Set-Item -Path env:TF\_VAR\_filename -Value 'wild.txt'**

**terraform apply**

****

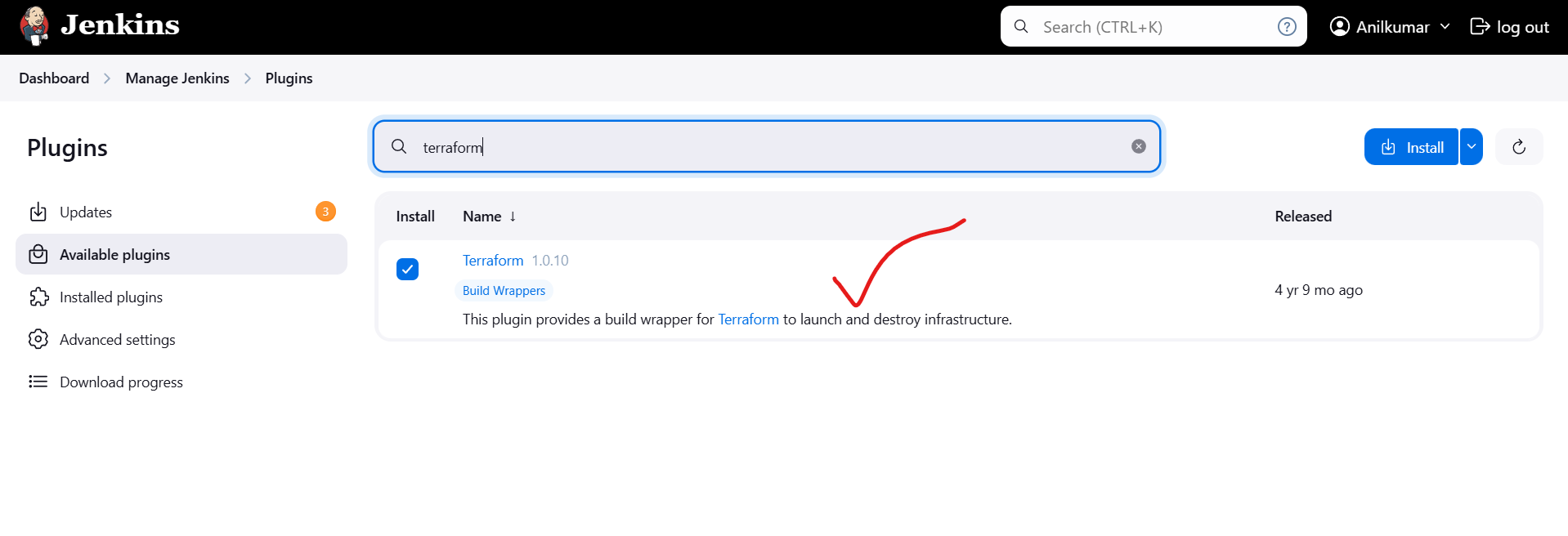
**Explicit**

****

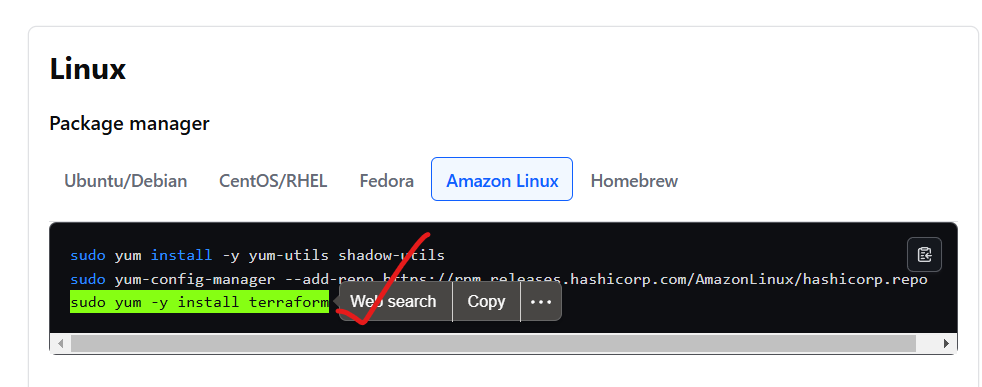
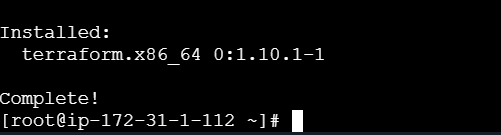
3) Intergrate terrafrom in jenkins using Terraform plugin.

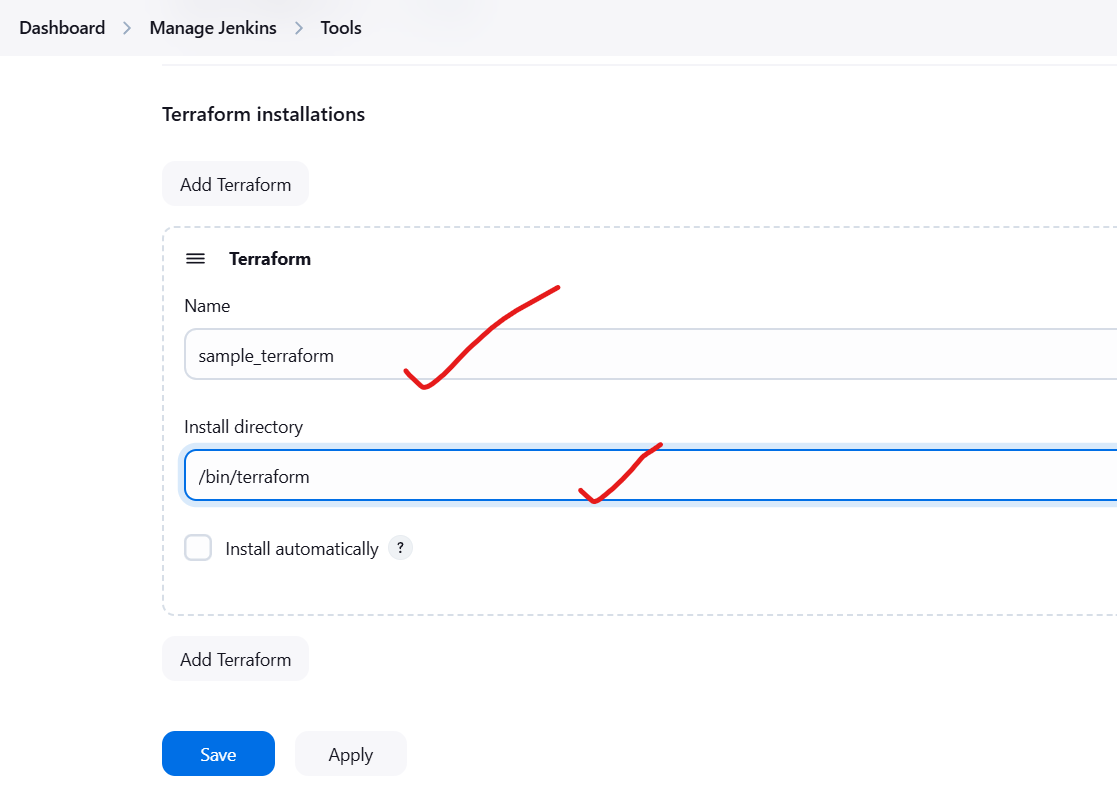
# To integrate a sample Terraform template in Jenkins.

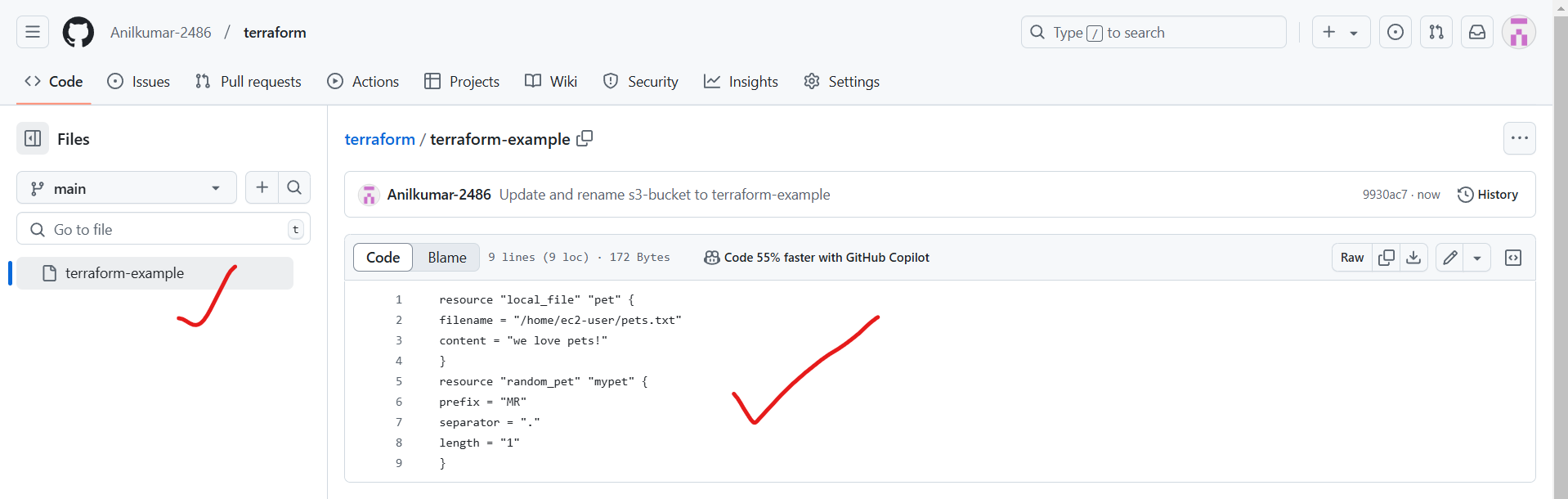
* + First we need to launch ec2 server.
  + With in the server you need to install Jenkins.
  + Go to the Jenkins GUI.
  + Go to manage Jenkins > Plugins.
  + Click on Available plugins there you need to search terraform plugin.
  + Click on check box and install.

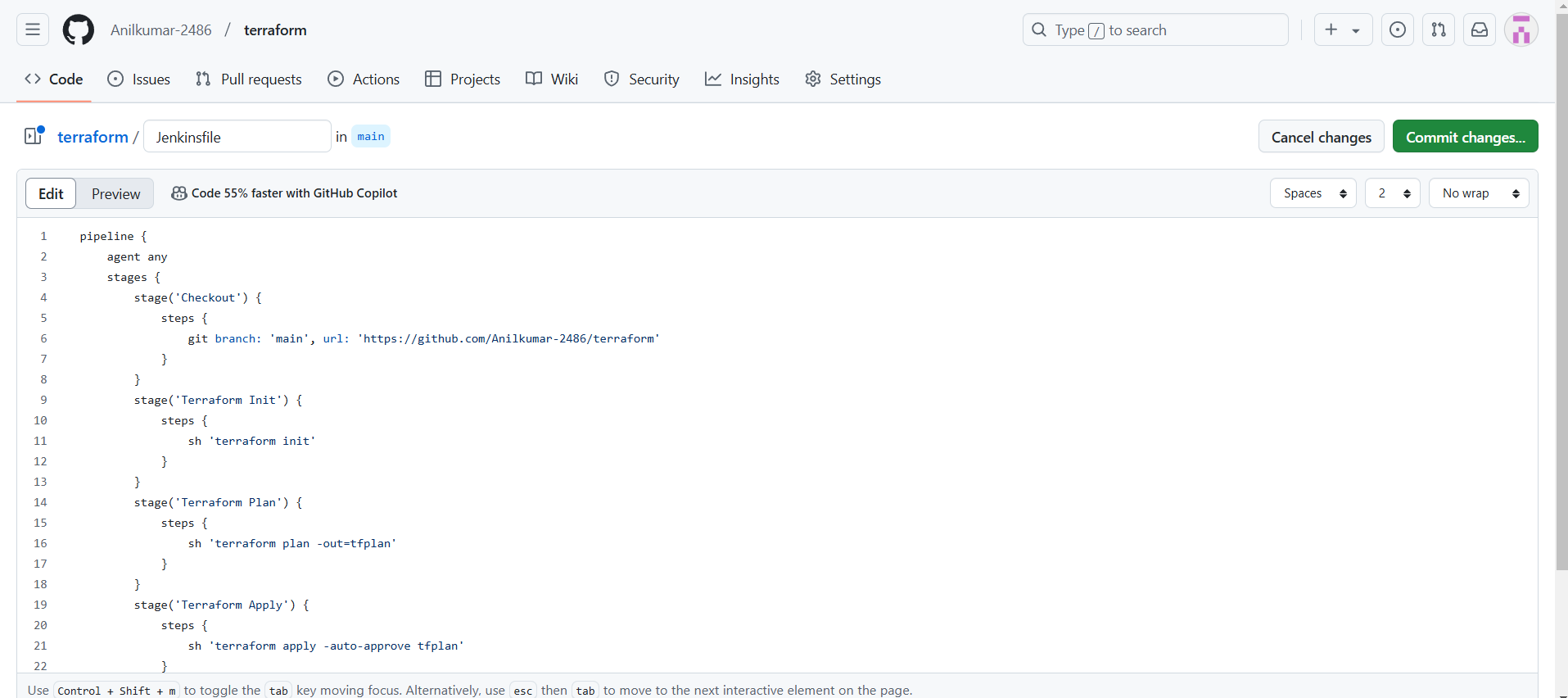


sudo yum install -y yum-utils shadow-utils

sudo yum-config-manager --add-repo <https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>  
sudo yum -y install terraform  


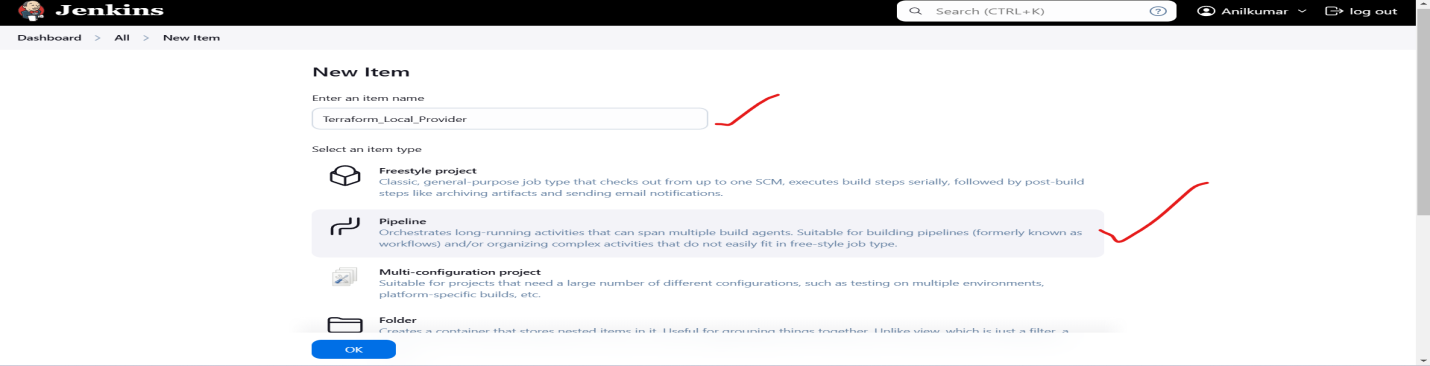


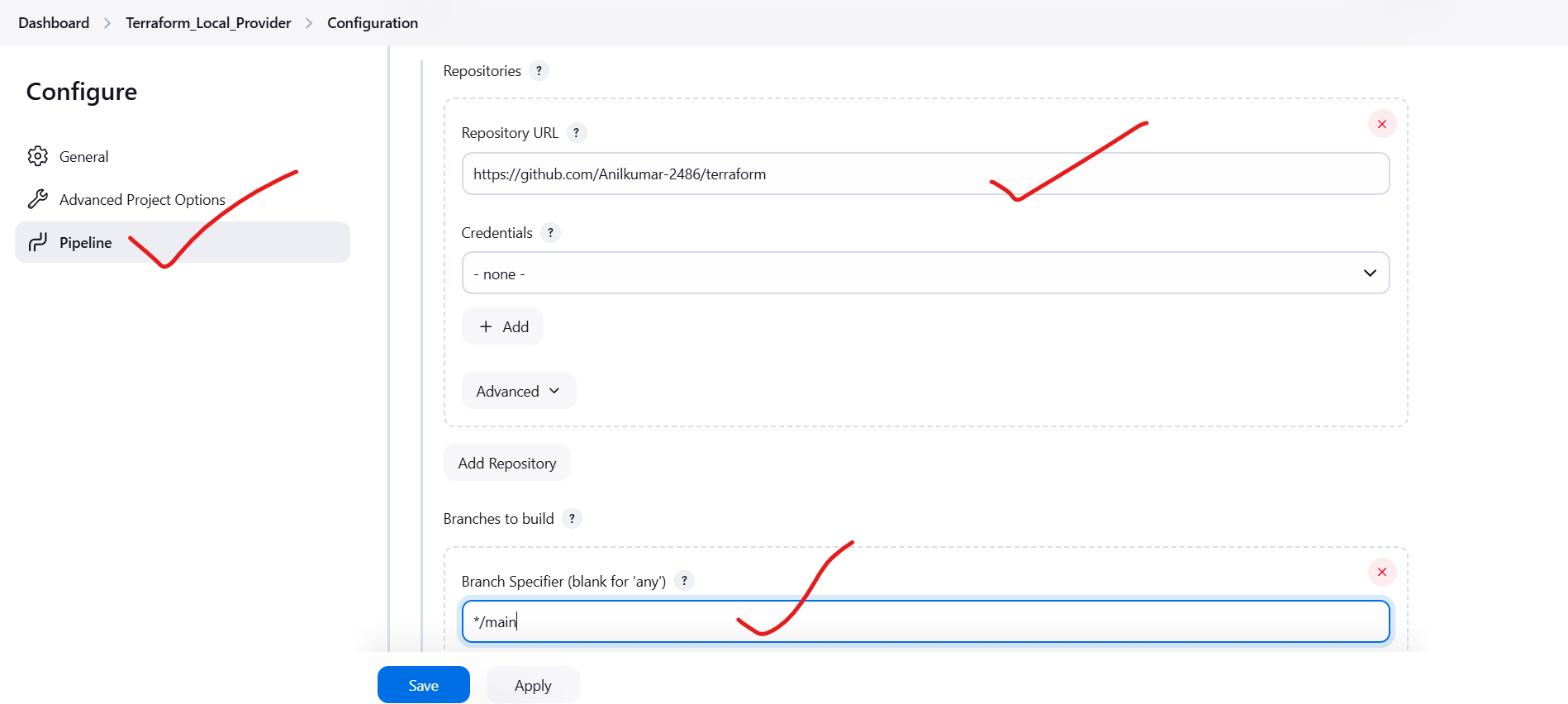


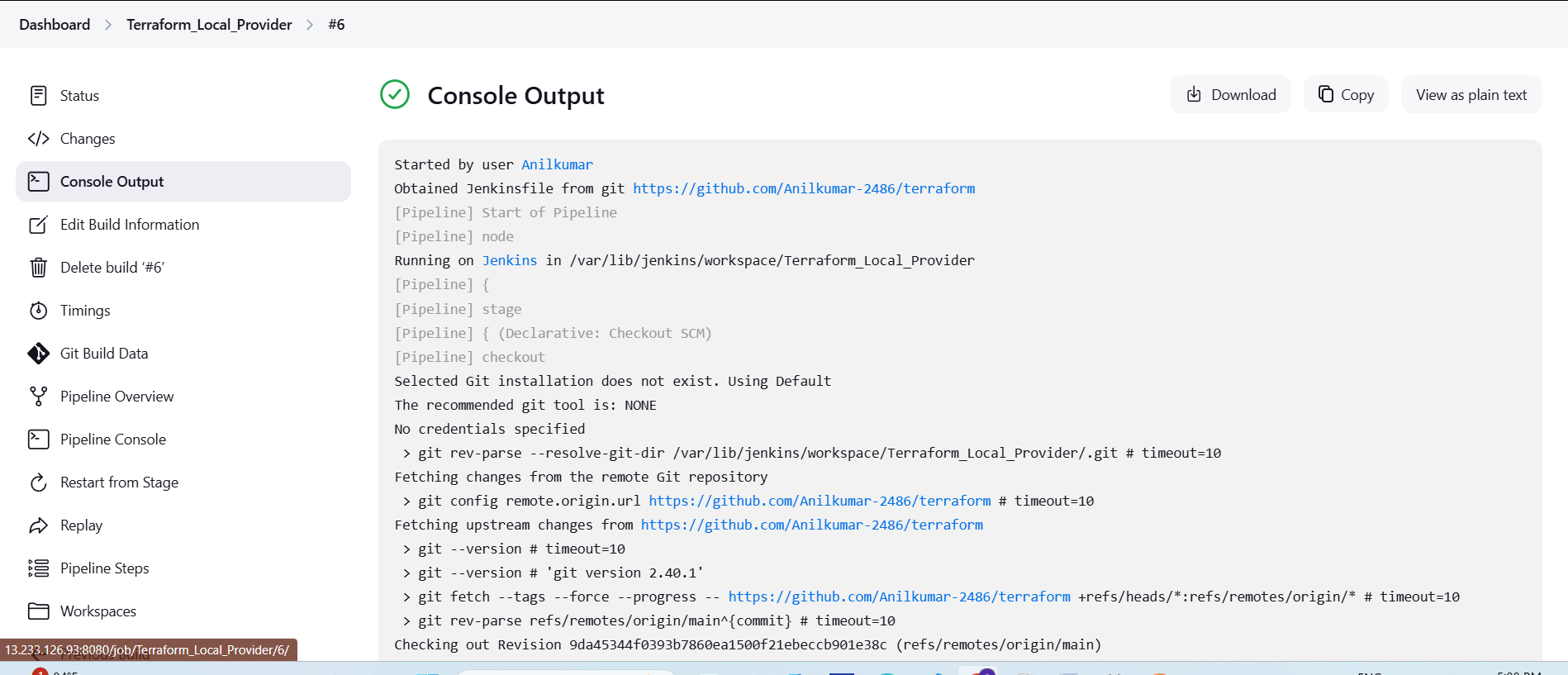


Now I went to the Jenkins GUI and I am created the one pipelinejob.

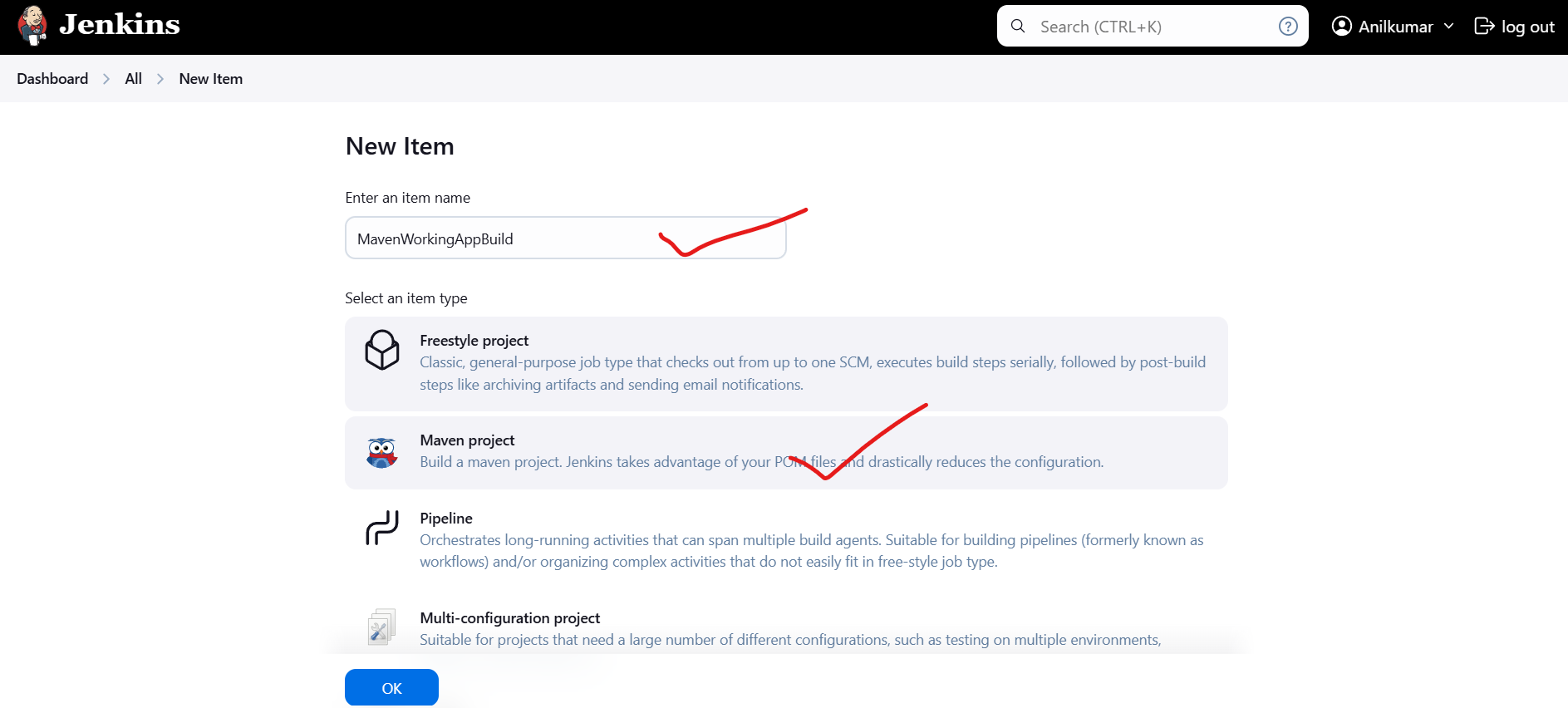
* + The job name given as Terraform\_Local\_provider
  + Select job – pipeline
  + Click on ok.
  + After that go to the pipeline and select pipeline script from SCM
  + Give repo URL
  + Credentials are not required because this is not private repo.

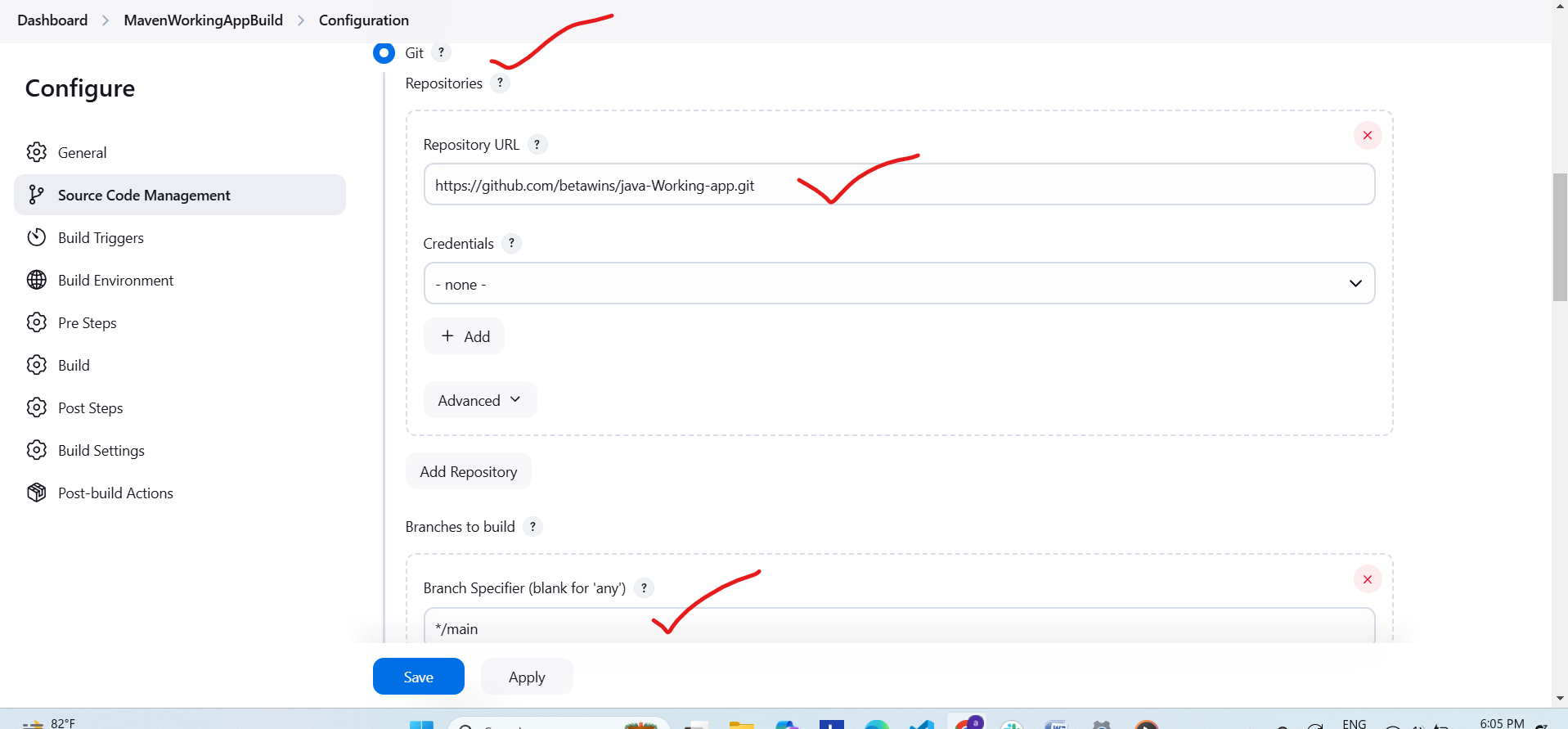


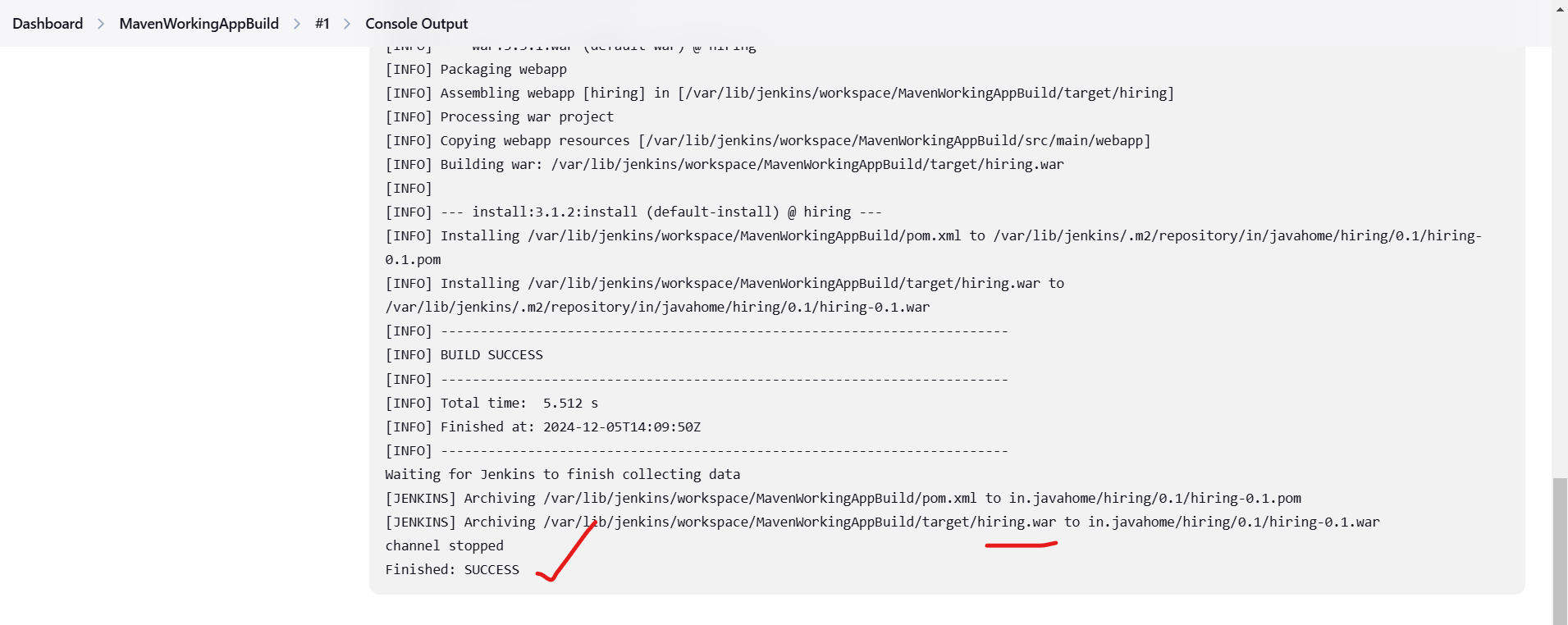




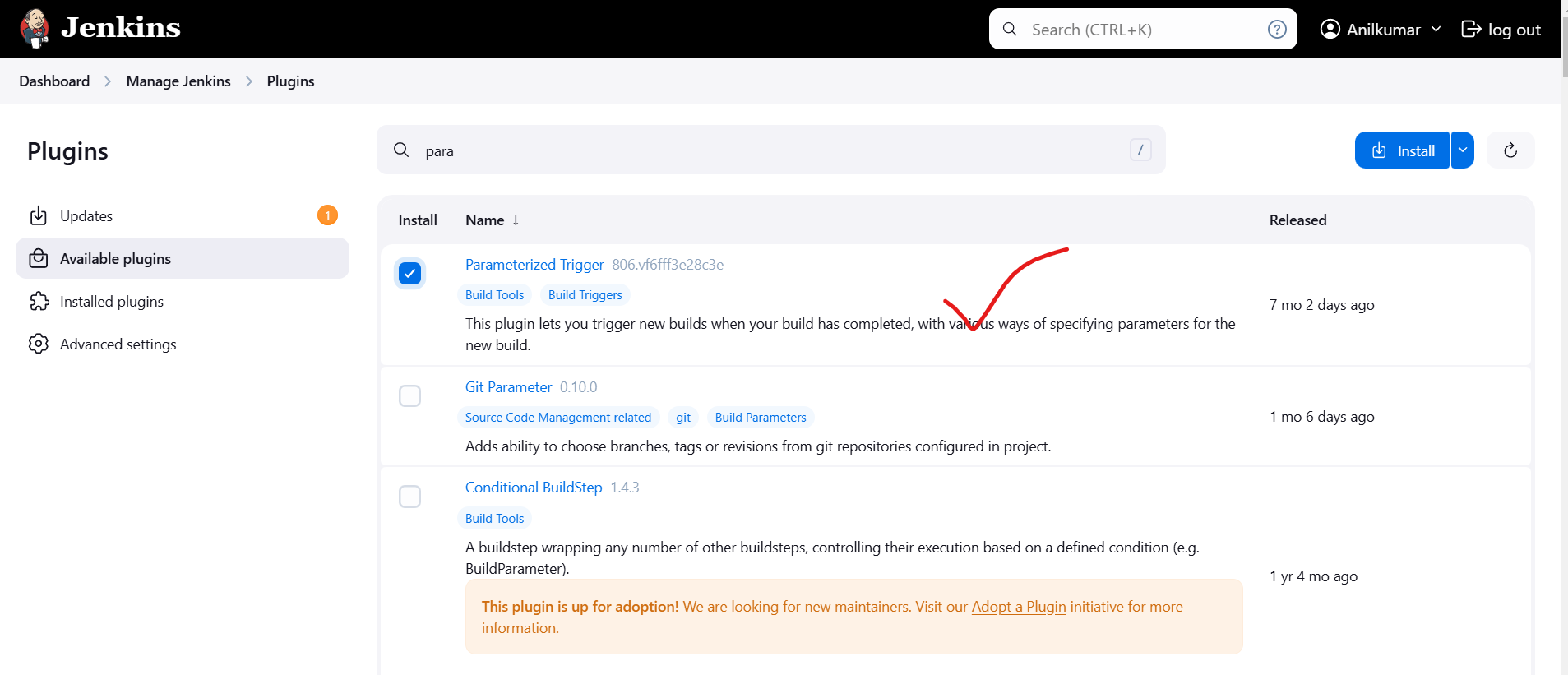
4) Create one jenkins job using MAVEN PROJECT for the below code with two stages.   
stage 1: Git clone  
stage 2: Maven Compilation Code: <https://github.com/betawins/java-Working-app.git>

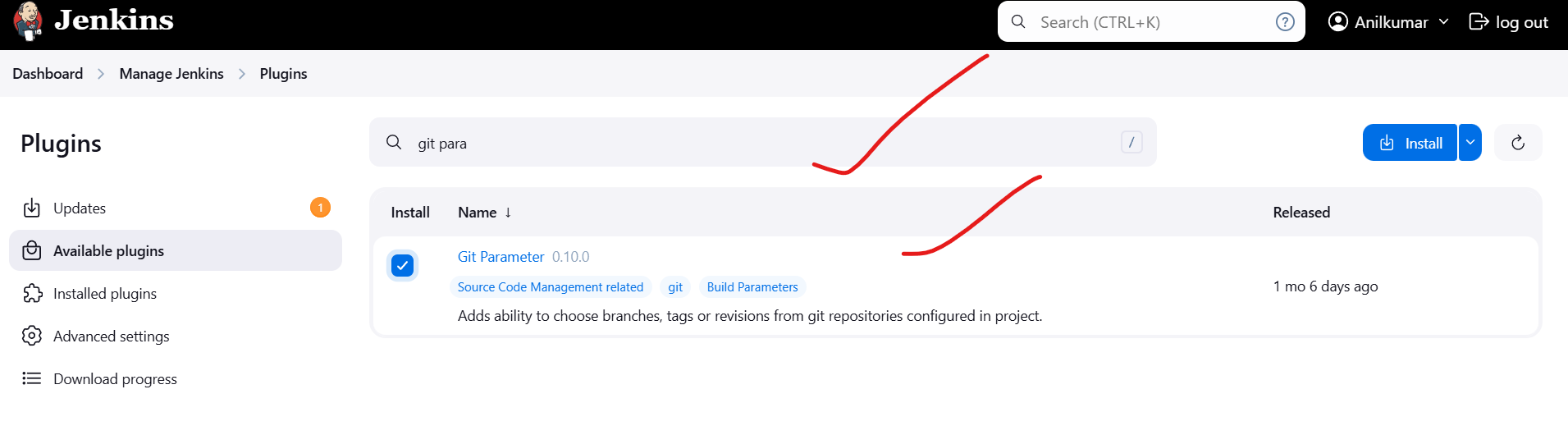


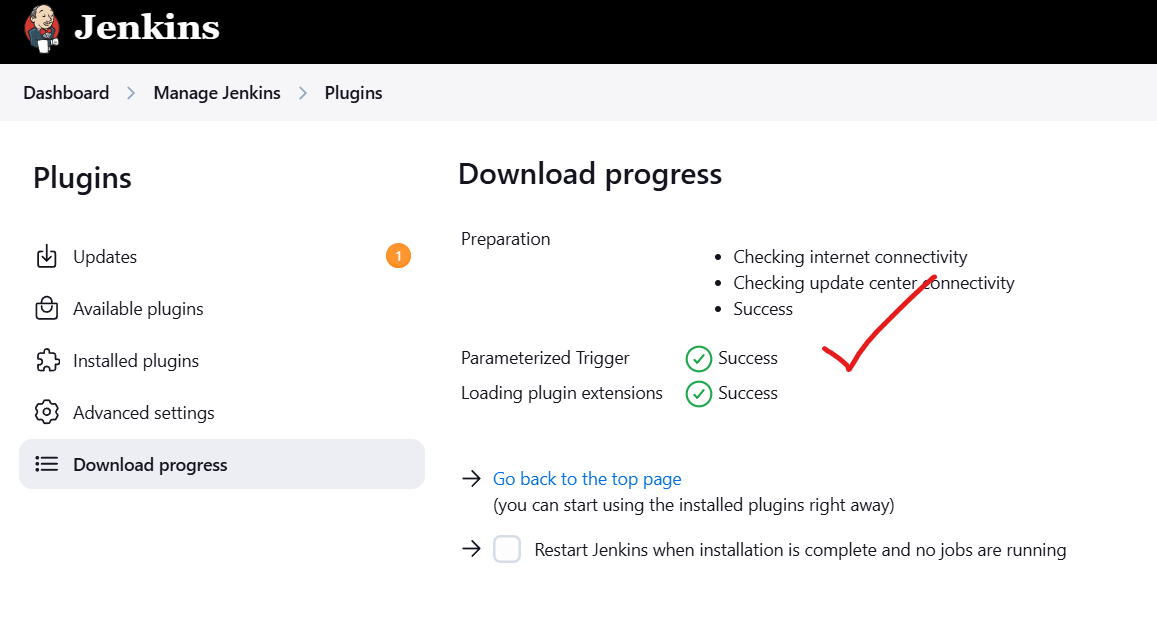


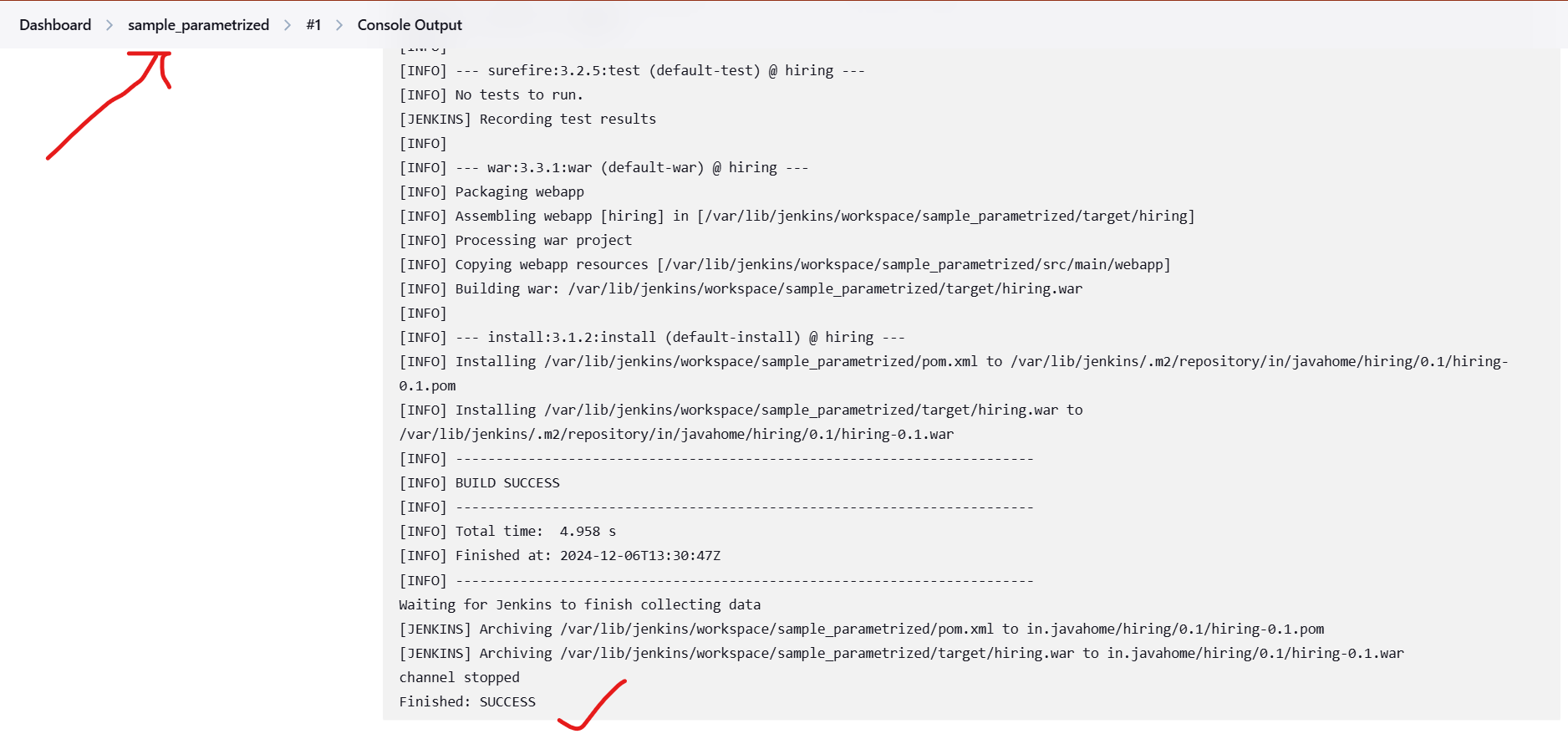
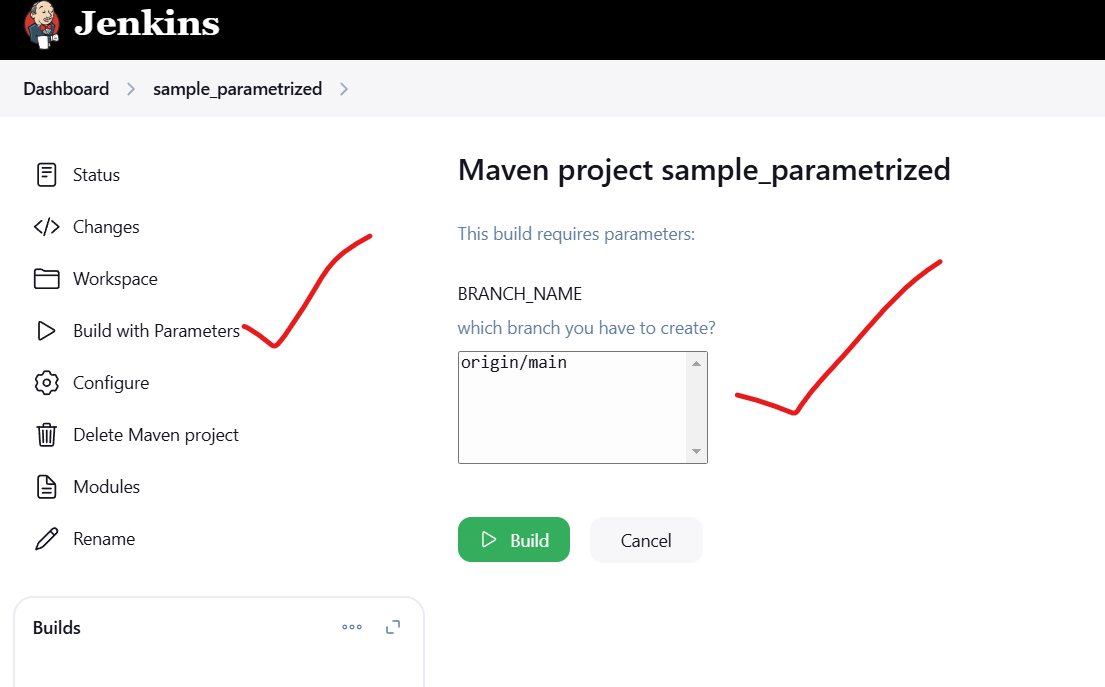
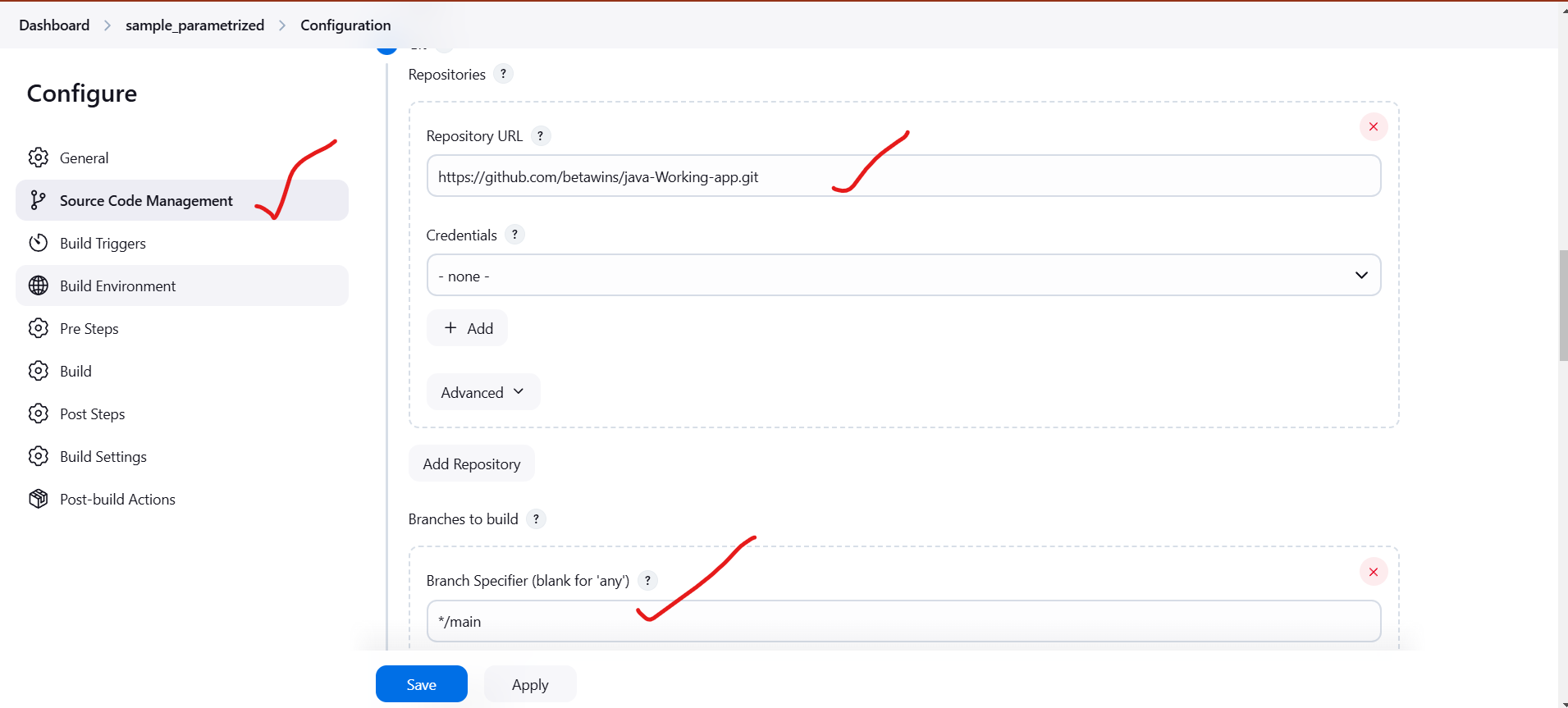
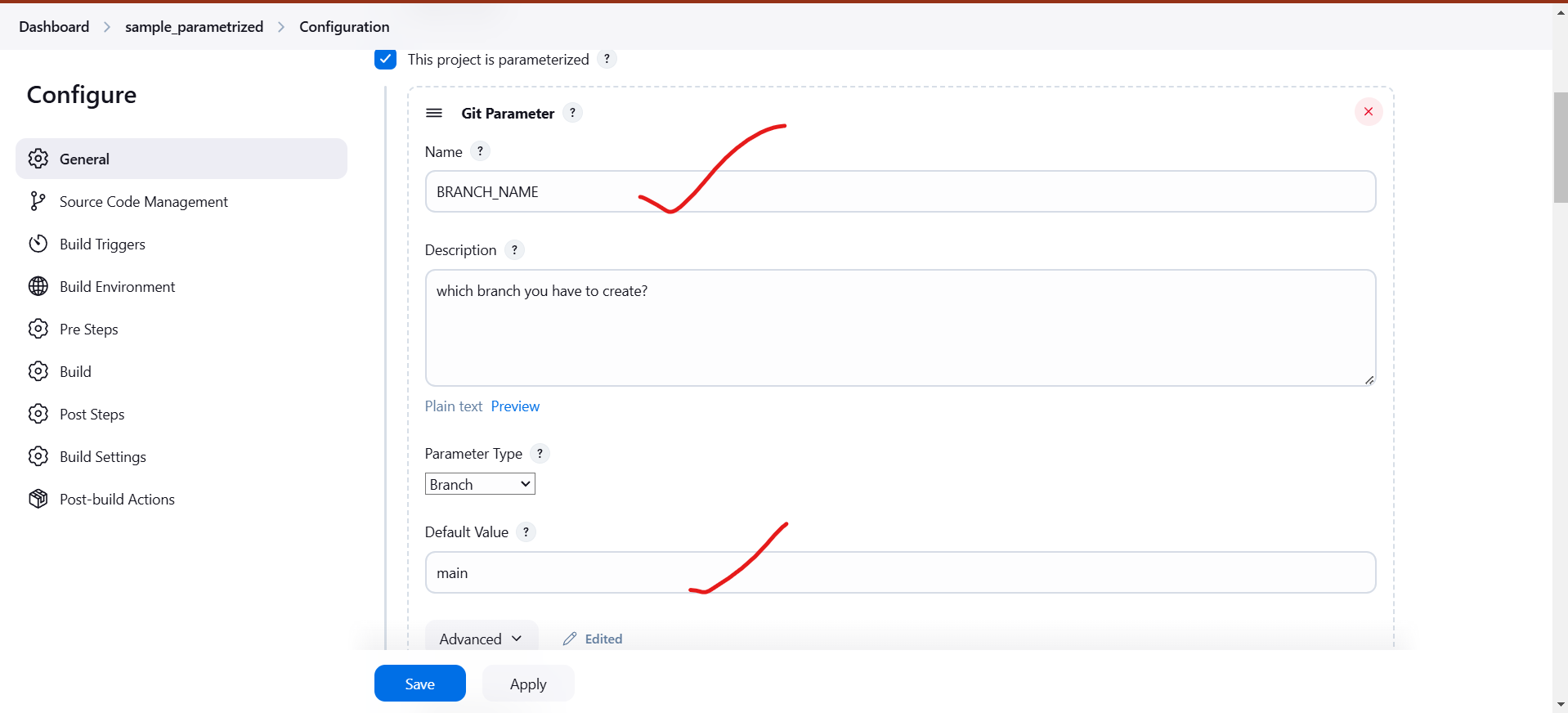


5) Use the below code and create a parameterized job in jenkins   
stage 1: Git clone   
stage 2: Maven Compilation Code: <https://github.com/betawins/java-Working-app.git>







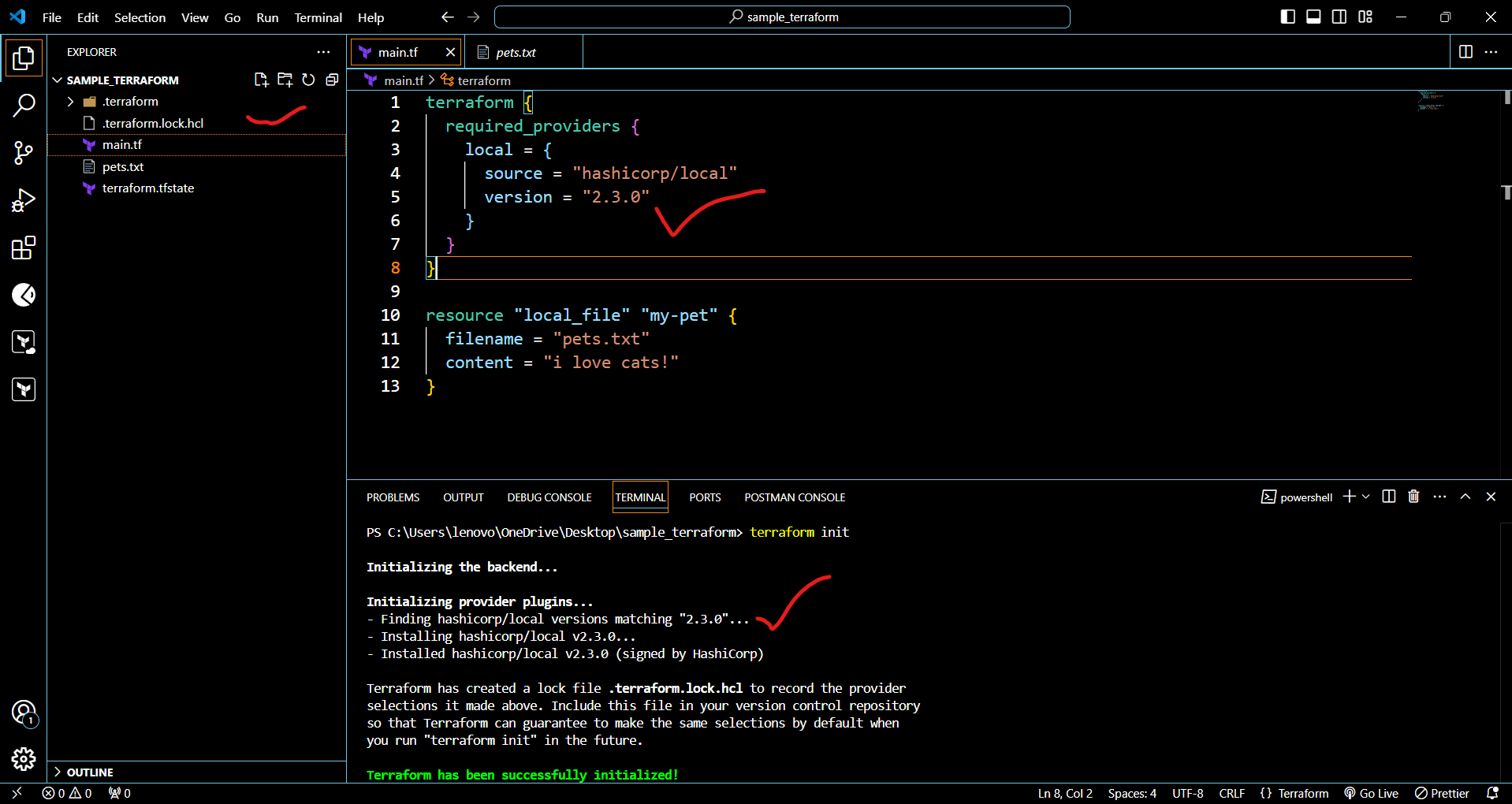
  
6) What are the global varaiables in jenkins?

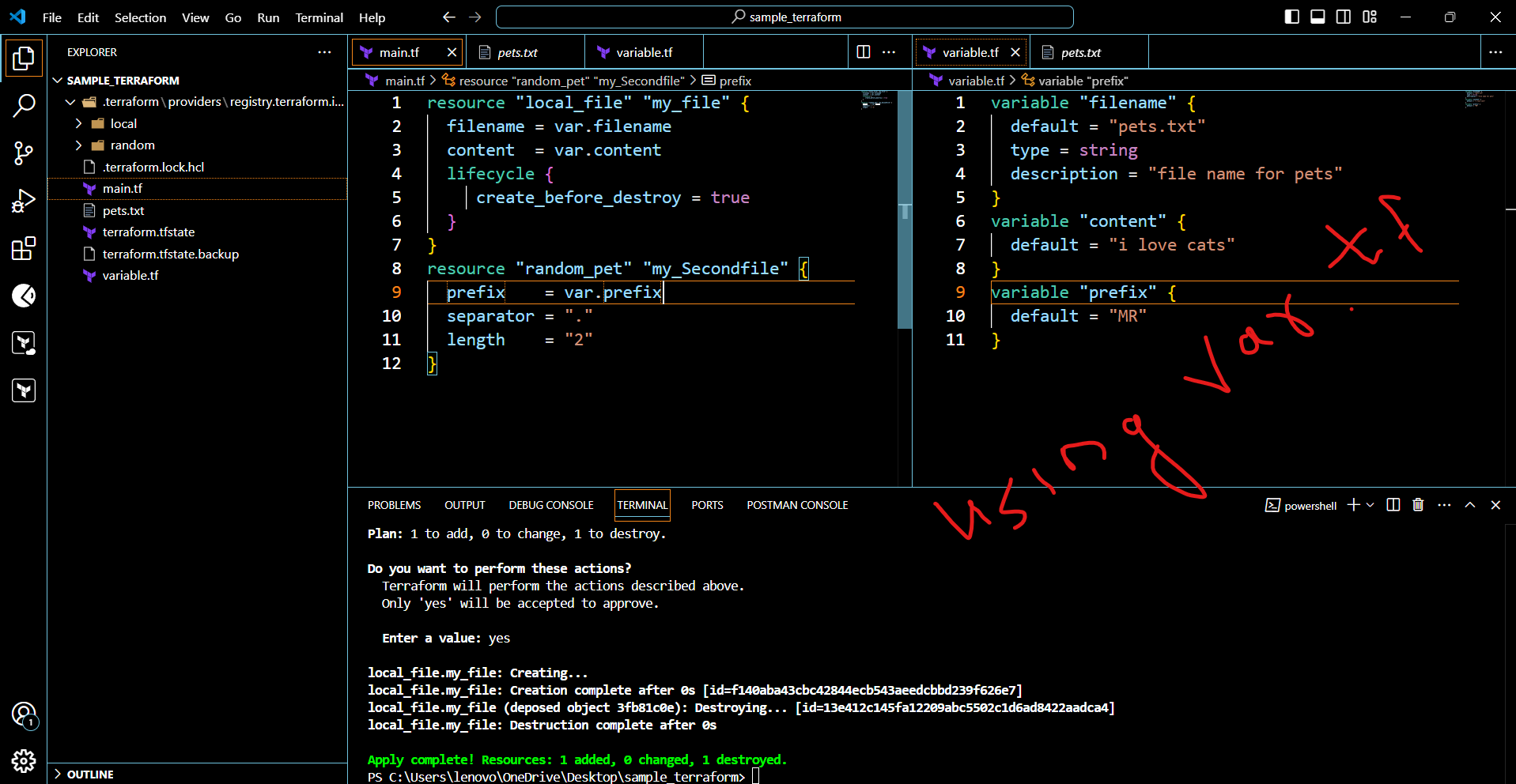
1. BUILD\_NUMBER: This variable contains the current build number. It's automatically incremented with each build.
2. BUILD\_ID: A unique identifier for the current build, typically in the format "YYYY-MM-DD\_hh-mm-ss."
3. JOB\_NAME: The name of the current Jenkins job or project.
4. WORKSPACE: The directory 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)

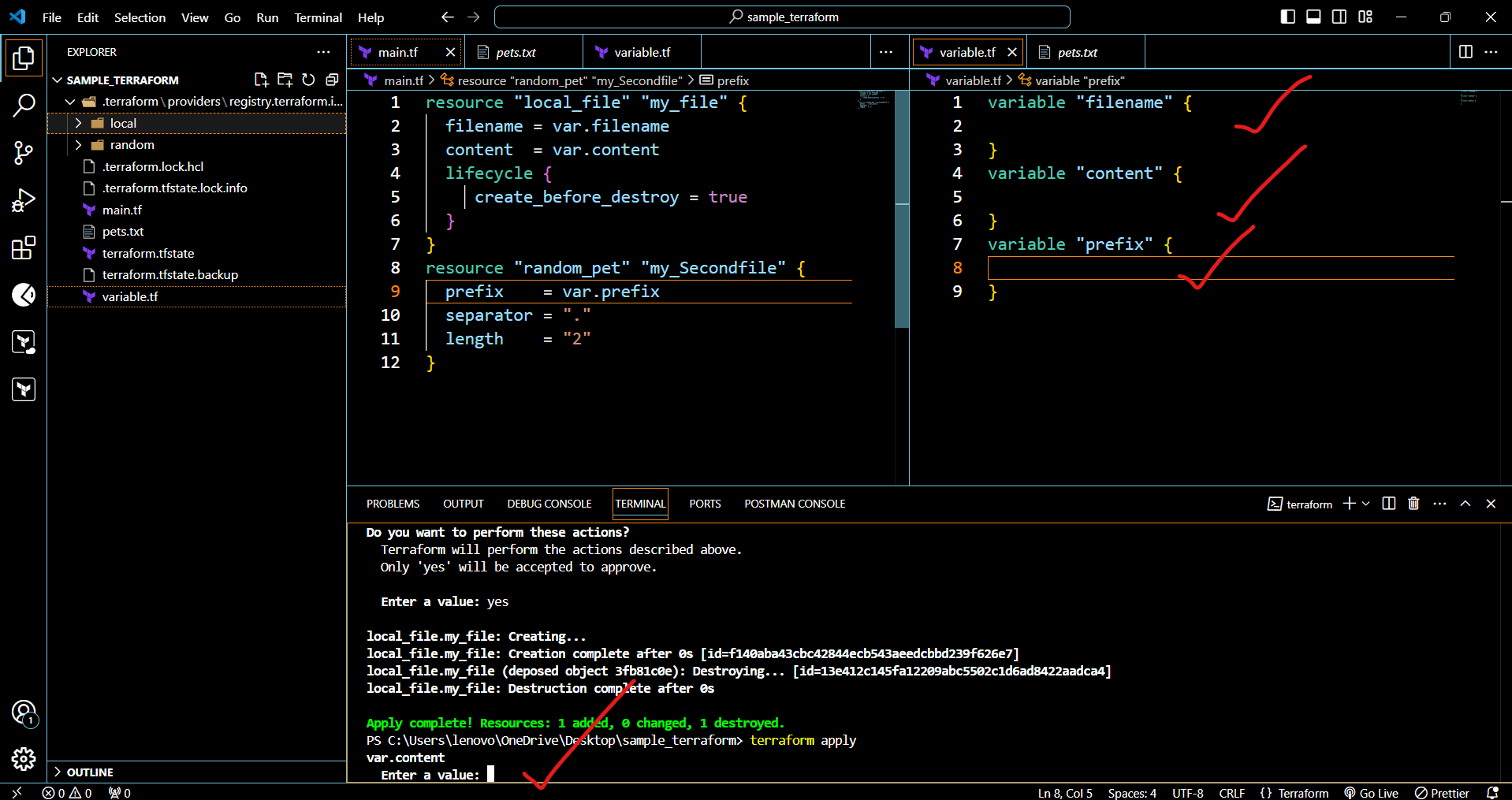
7) Watch terraform-04 video.

Completed

8) Execute the script shown in video.

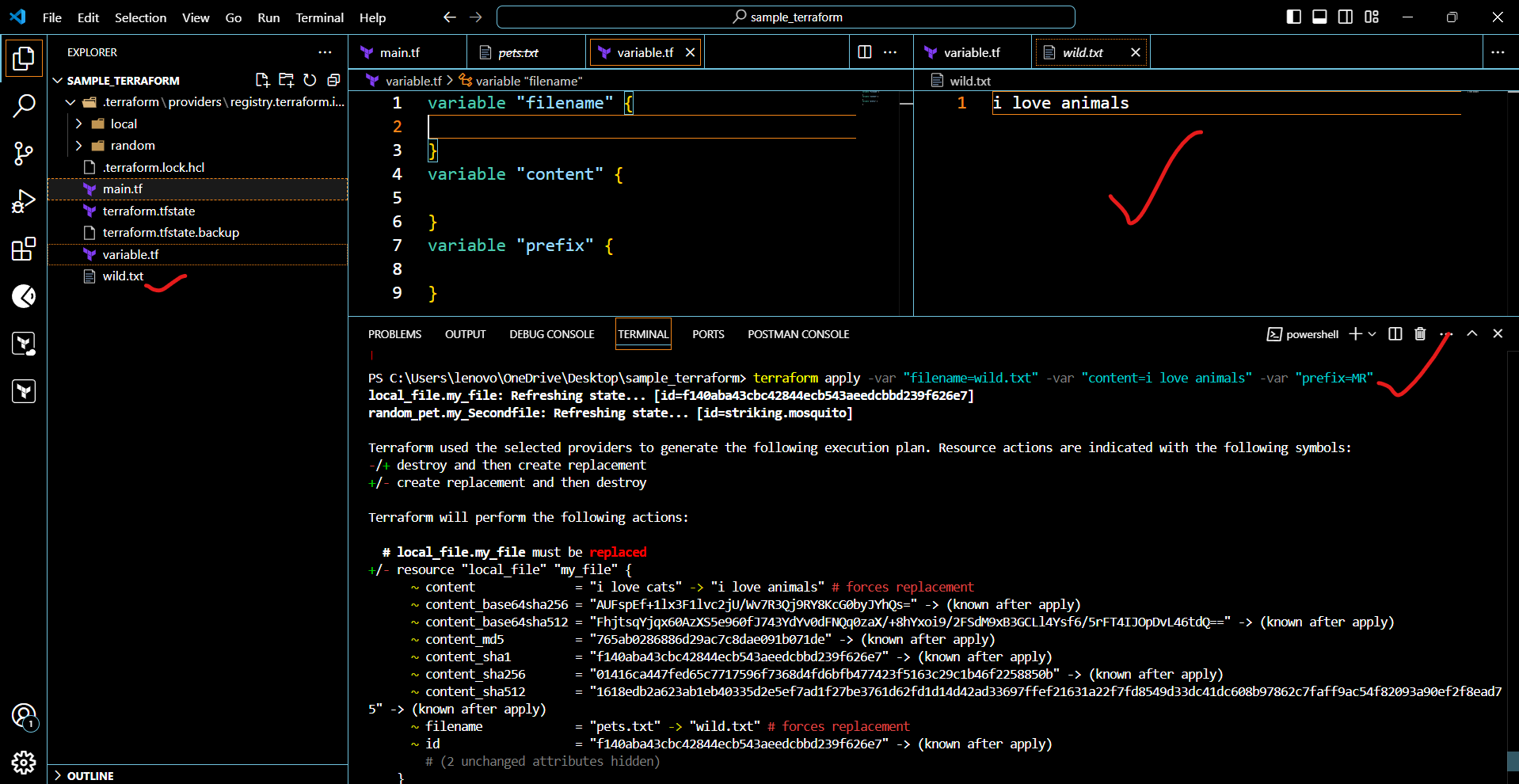




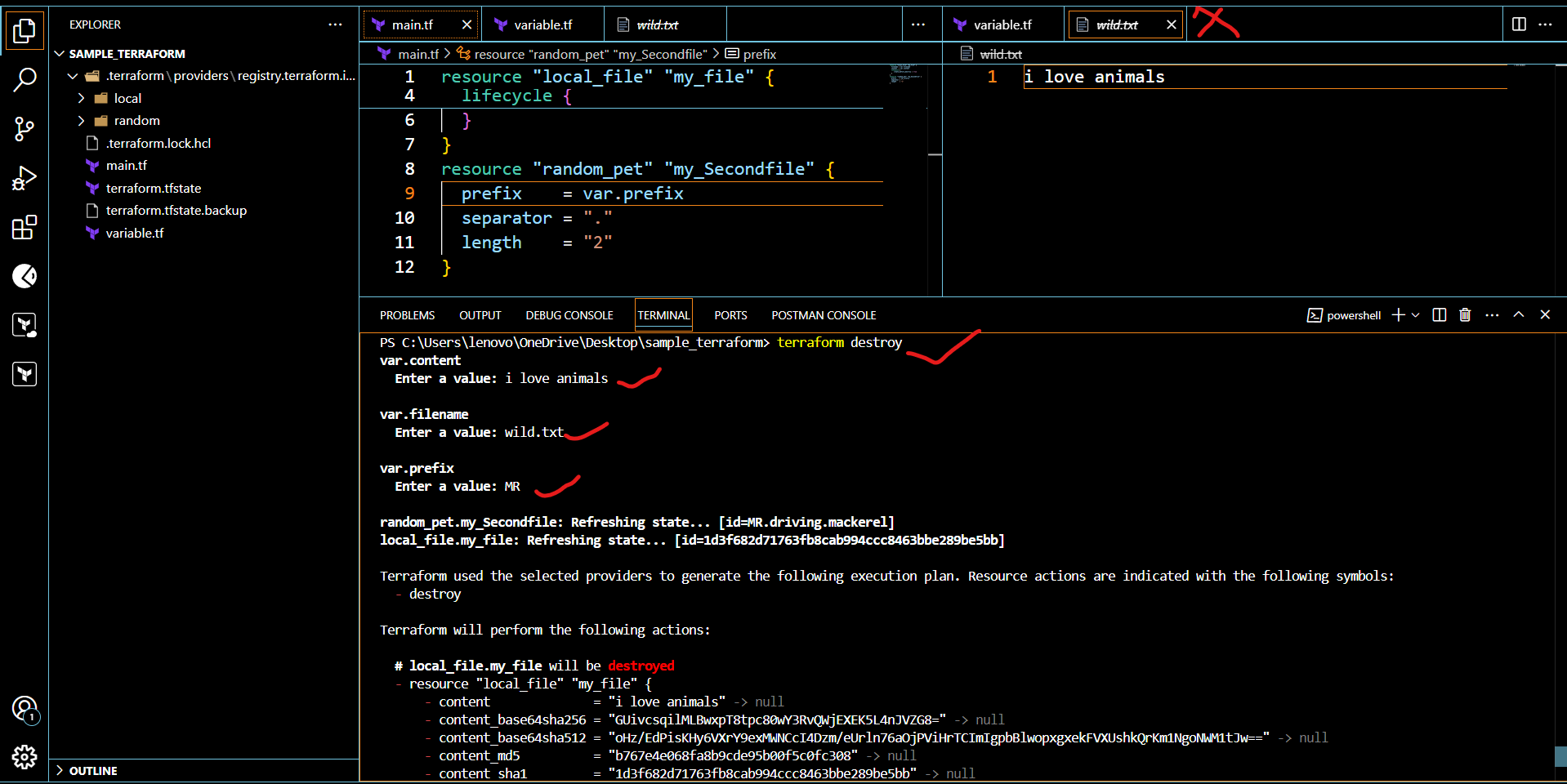


**Command line flags**

**--> terraform apply - var "filename=/root/pets.txt" -var "prefix=MR"**

****

**Terraform destroy**

****

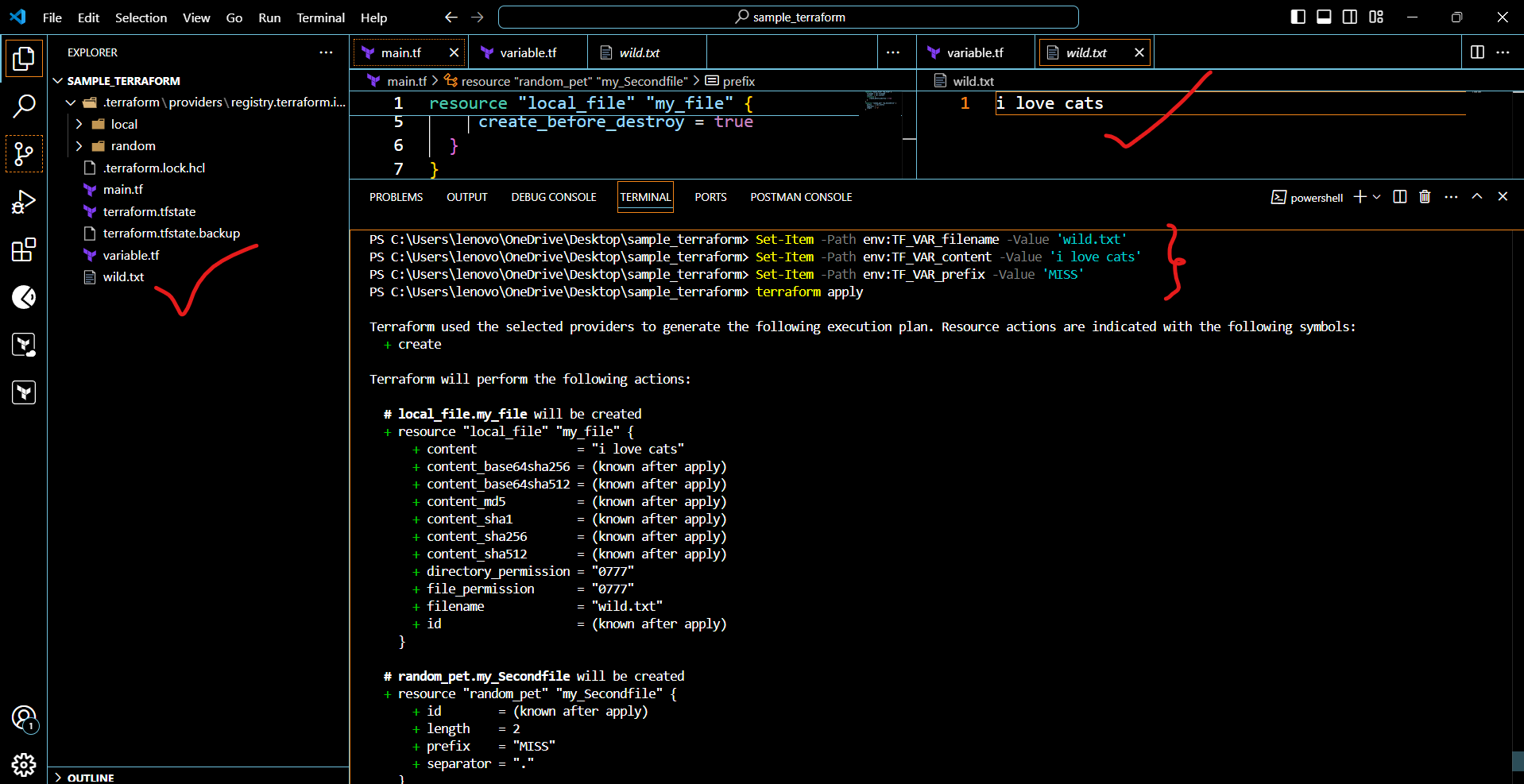
**Environment variables**

**--> export TF\_VAR\_filename="/root.pets.txt" [linux]**

**--> export TF\_VAR\_prefix= "MR" [windows]**

**--> Set-Item -Path env:TF\_VAR\_filename -Value 'wild.txt'**

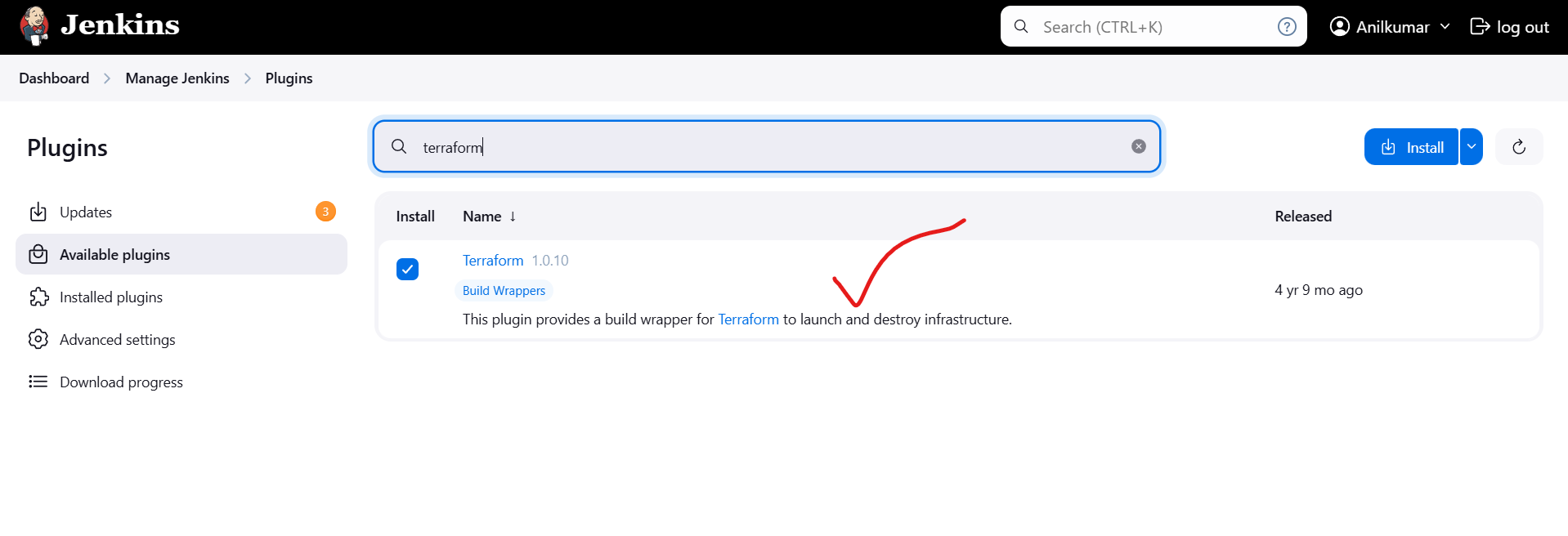
**terraform apply**

****

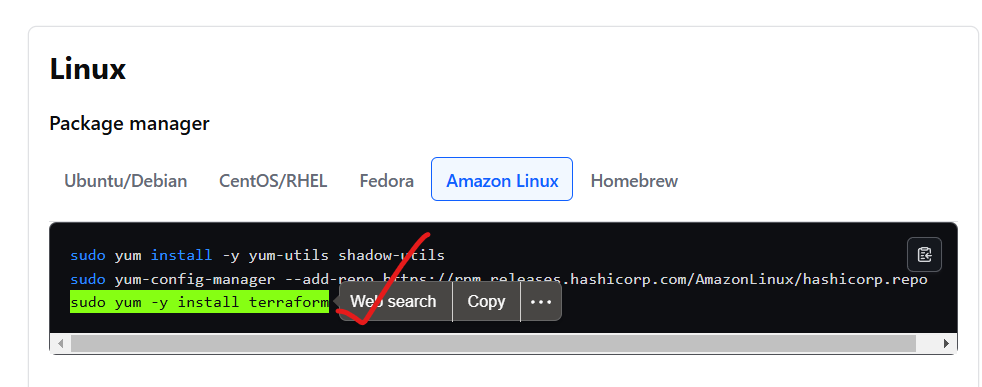
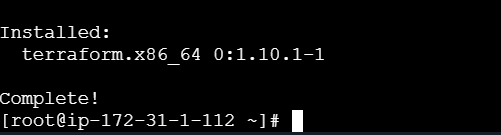
9) Integrate terrafrom in jenkins using Terraform plugin.

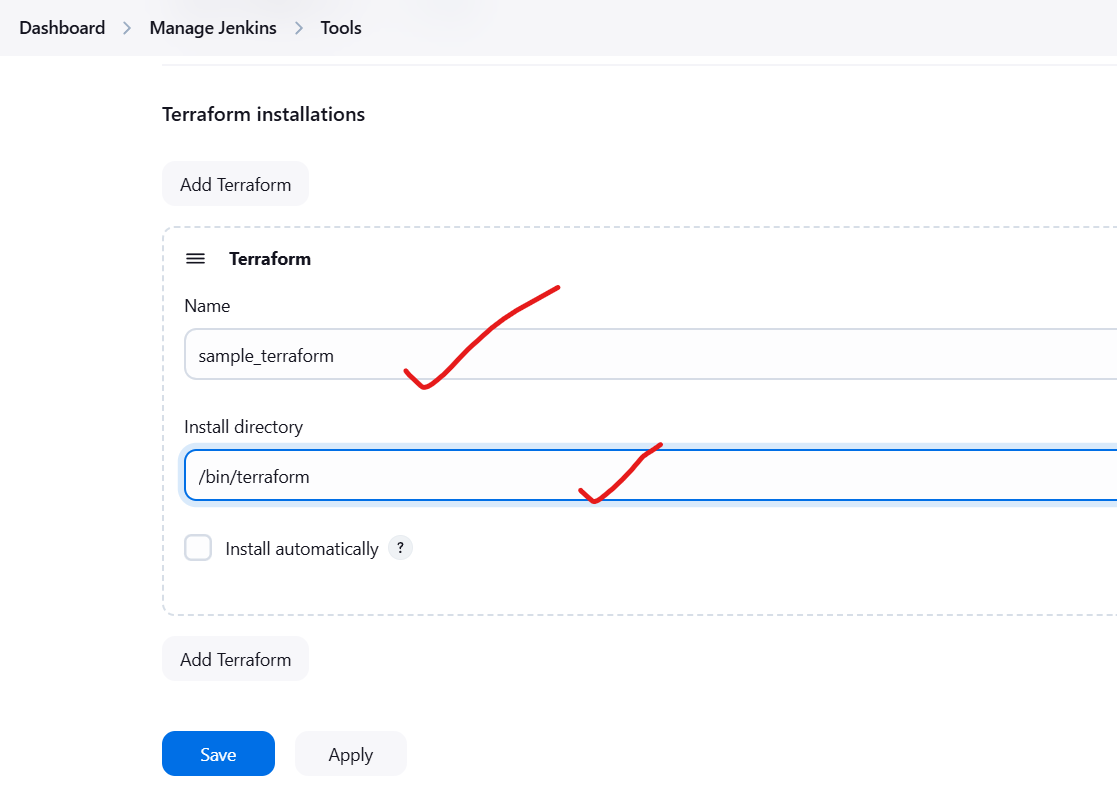
# To integrate a sample Terraform template in Jenkins.

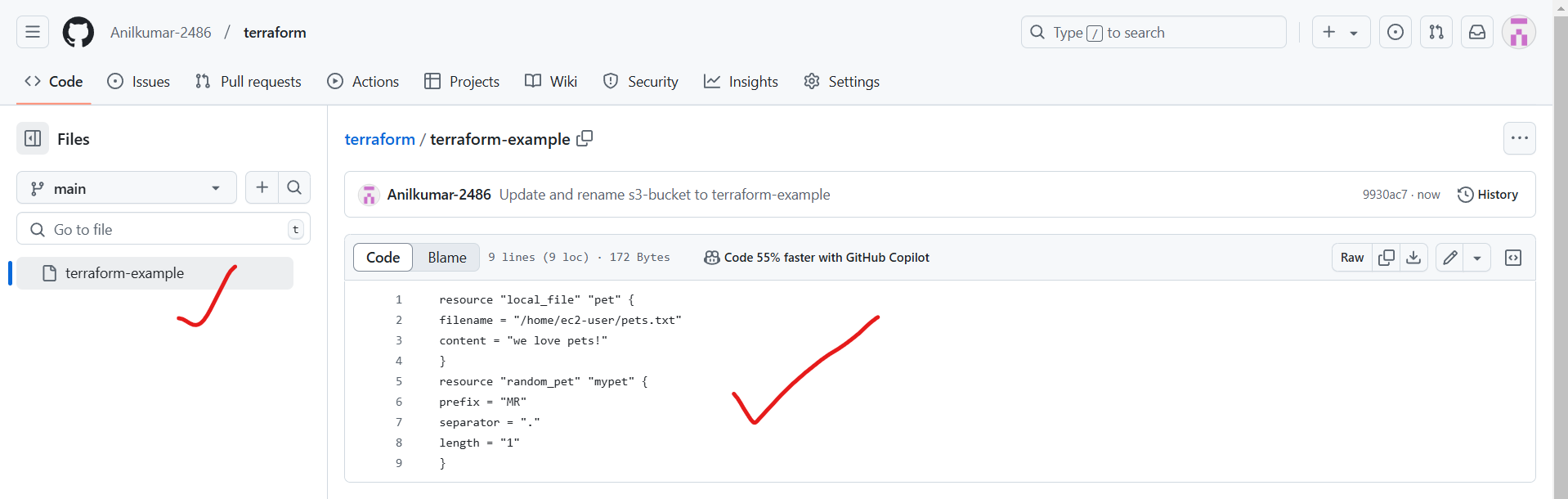
* + First we need to launch ec2 server.
  + With in the server you need to install Jenkins.
  + Go to the Jenkins GUI.
  + Go to manage Jenkins > Plugins.
  + Click on Available plugins there you need to search terraform plugin.
  + Click on check box and install.

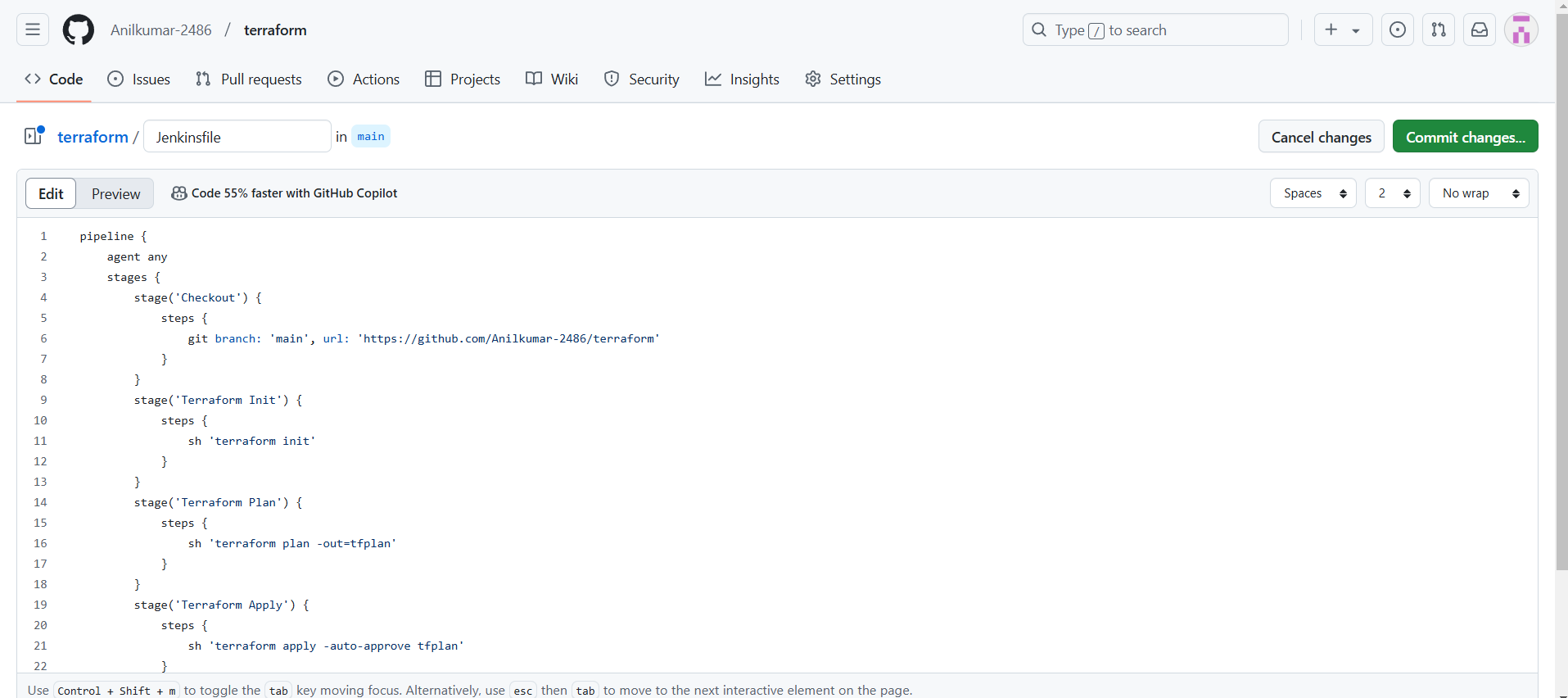


sudo yum install -y yum-utils shadow-utils

sudo yum-config-manager --add-repo <https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>  
sudo yum -y install terraform  


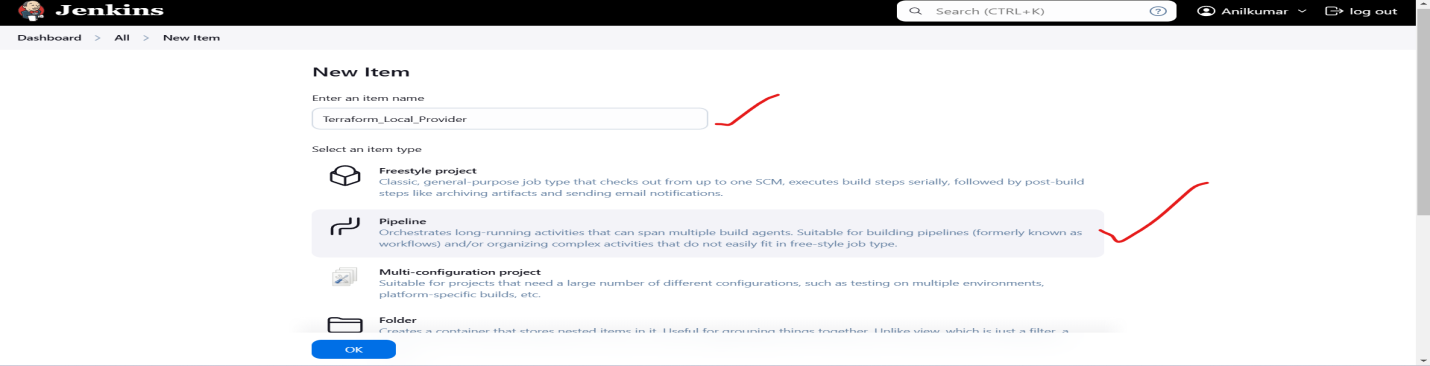


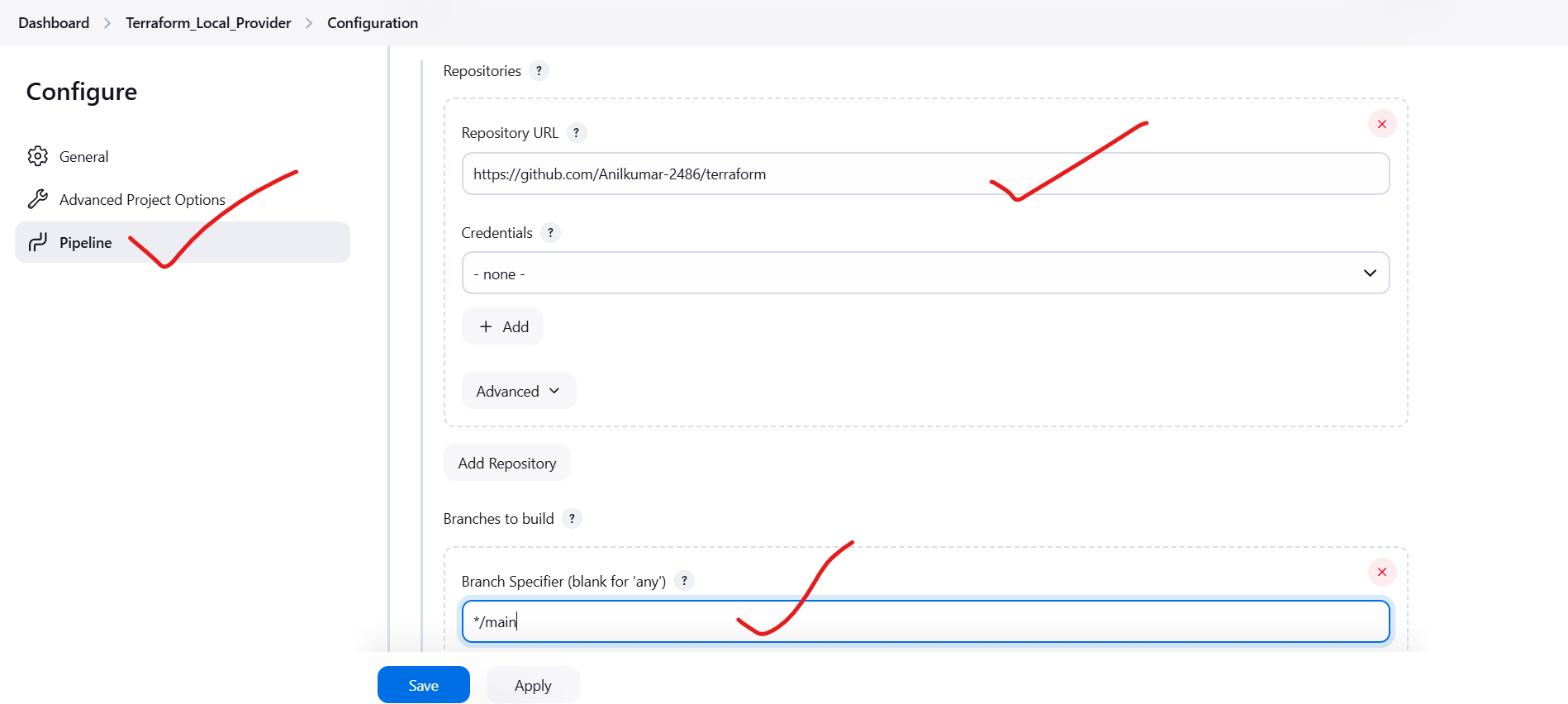


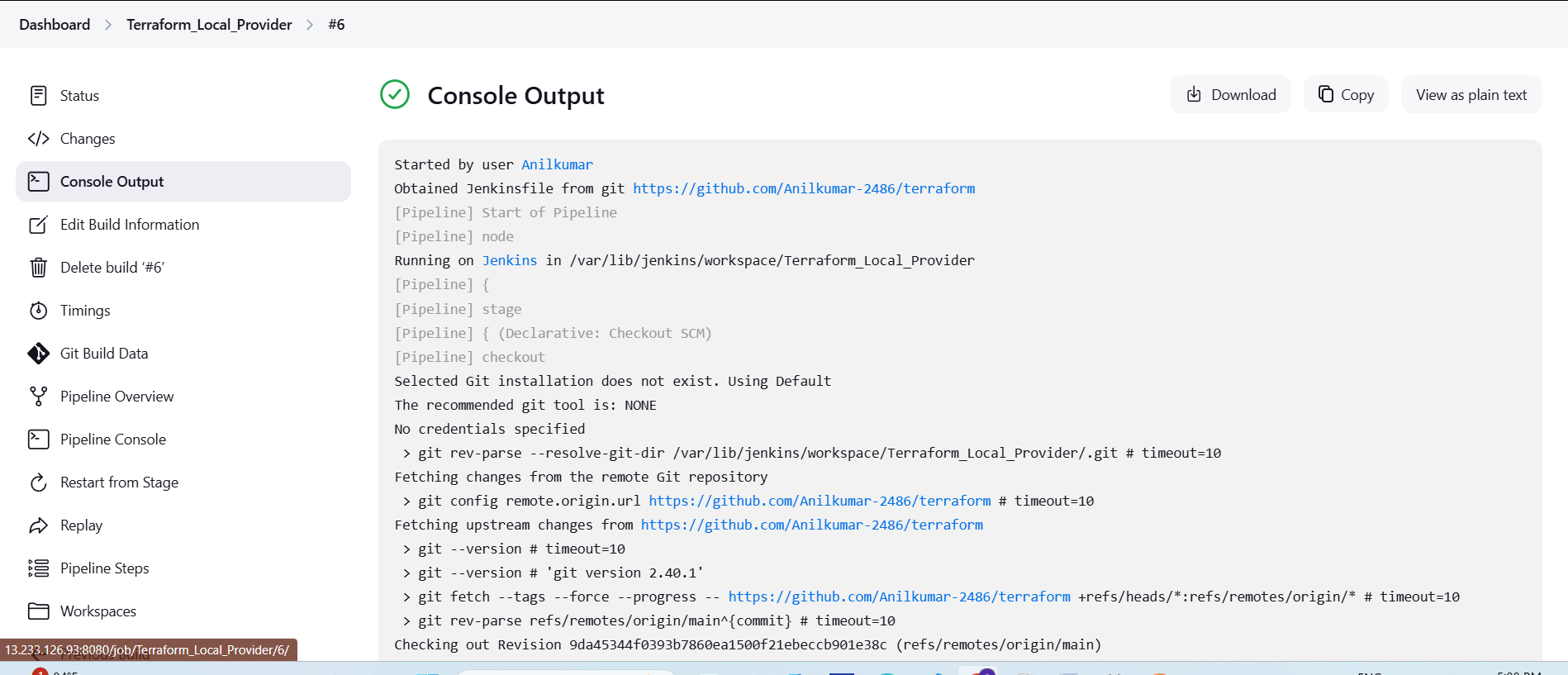


Now I went to the Jenkins GUI and I am created the one pipelinejob.

* + The job name given as Terraform\_Local\_provider
  + Select job – pipeline
  + Click on ok.
  + After that go to the pipeline and select pipeline script from SCM
  + Give repo URL
  + Credentials are not required because this is not private repo.







10) Create CICD pipeline for Nodejs Application. <https://github.com/betawins/Trading-UI.git>

11) Explain 10 Maven commands.

### 1. mvn clean : Before starting a fresh build to avoid conflicts with old files.

**2. mvn compile: To ensure the code compiles without errors before building the project.  
3. mvn test**: **After coding to verify the functionality of individual units of code.  
4. mvn package: To create the final artifact after confirming the code and tests are valid.  
5. mvn install:** **To make the artifact available for other local Maven projects.   
6. mvn deploy:** **For deployment to environments like staging or production  
7. mvn site: To create documentation for your project.  
8. mvn dependency:tree:**  **To troubleshoot dependency conflicts or understand the dependency hierarchy.  
9. mvn verify:** **To ensure the project passes all checks, including integration tests.  
10. mvn exec:java:**  **For testing or debugging Java classes during development.**