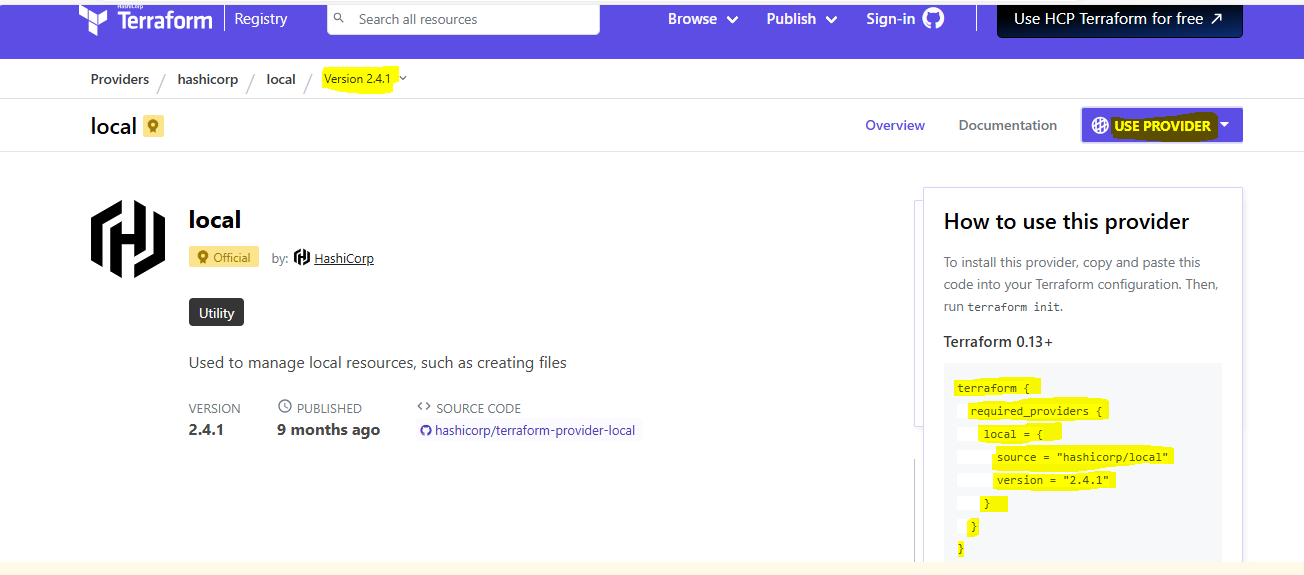
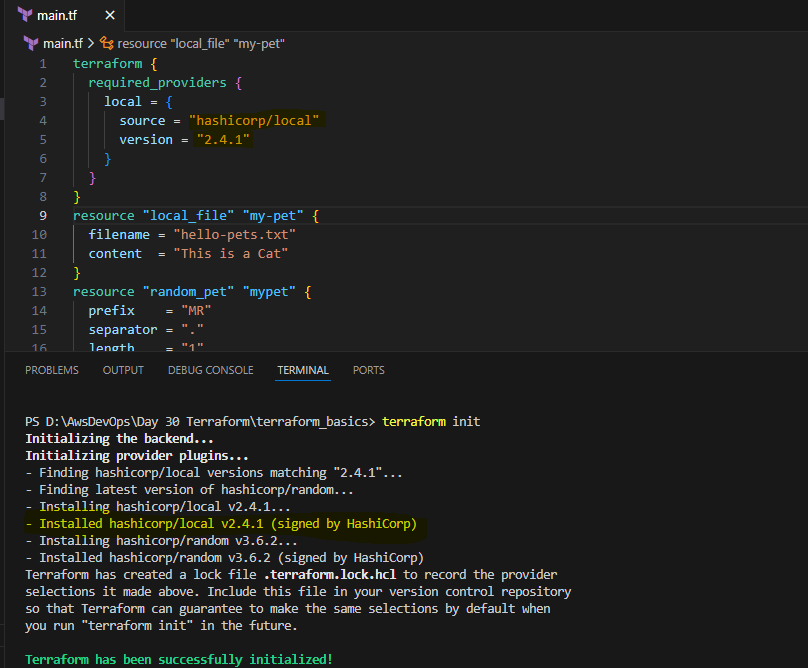
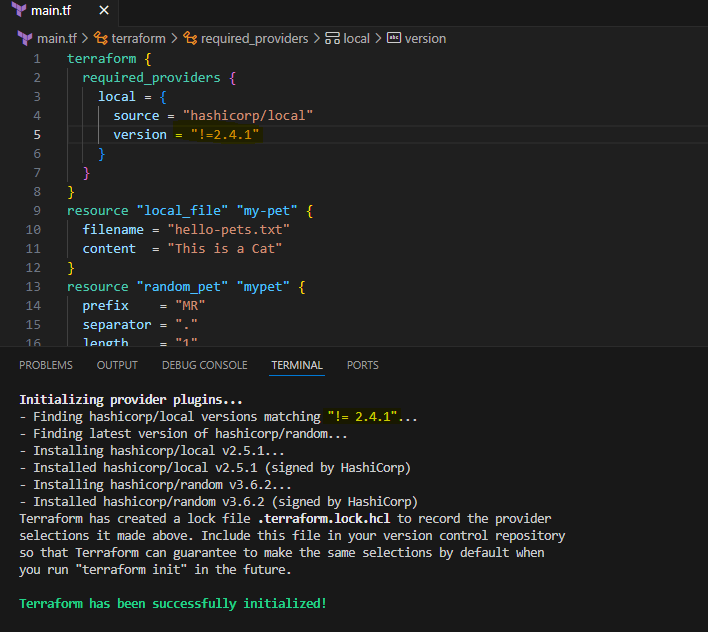
1) Watch terraform-04 video.

2) Execute the script shown in video.

If we want a specific version of local then:  
we need to go to the registry and copy the code of that provider and paste it at the start of the code.  


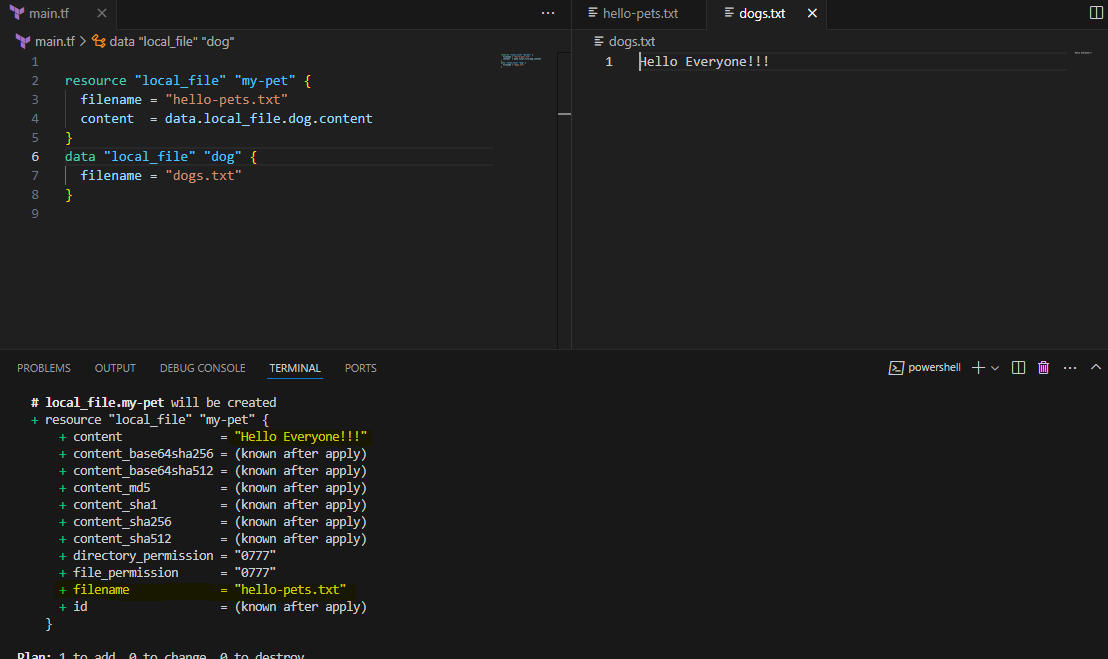


As I have mentioned **version = "!=2.4.1"** it won’t download that specific version:  


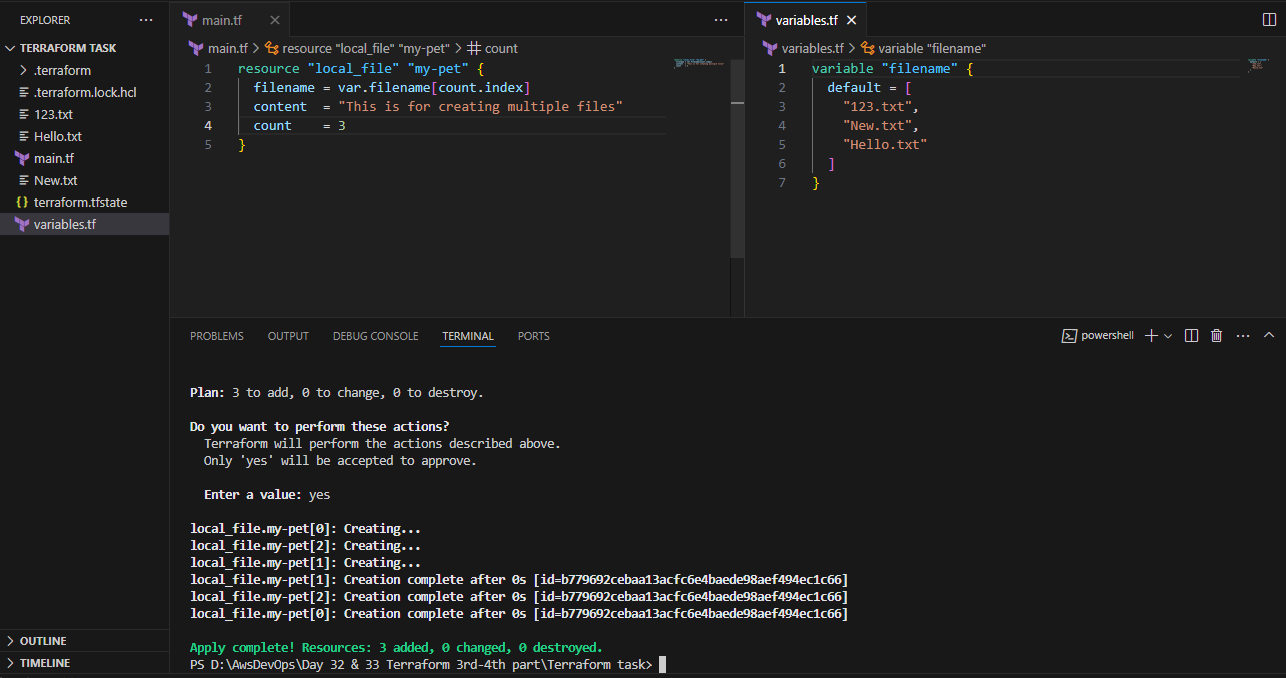
Similarly,

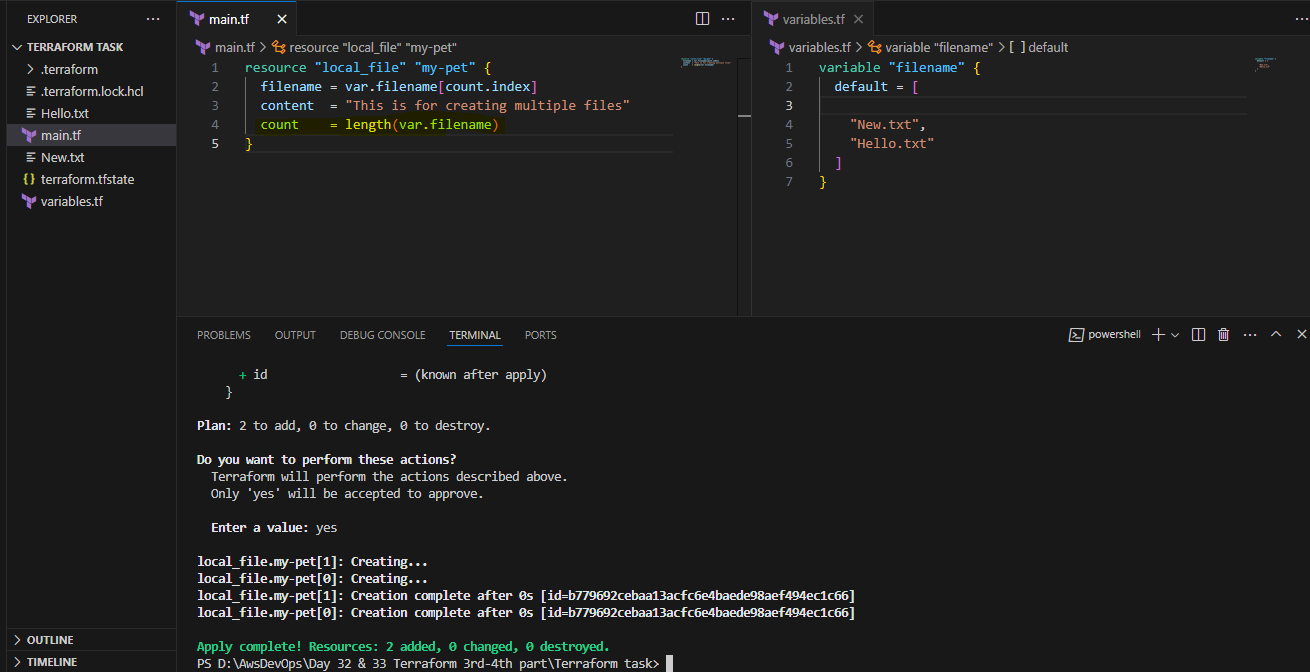
**Version = "2.4.1” it will use the exact same version.**  
**Version = "<2.4.1” it will use less than the mentioned version.**

**Version = ">2.4.1” it will use greater than the mentioned version.   
Version = "~>2.4.1” it will use the specific version or higher version.**

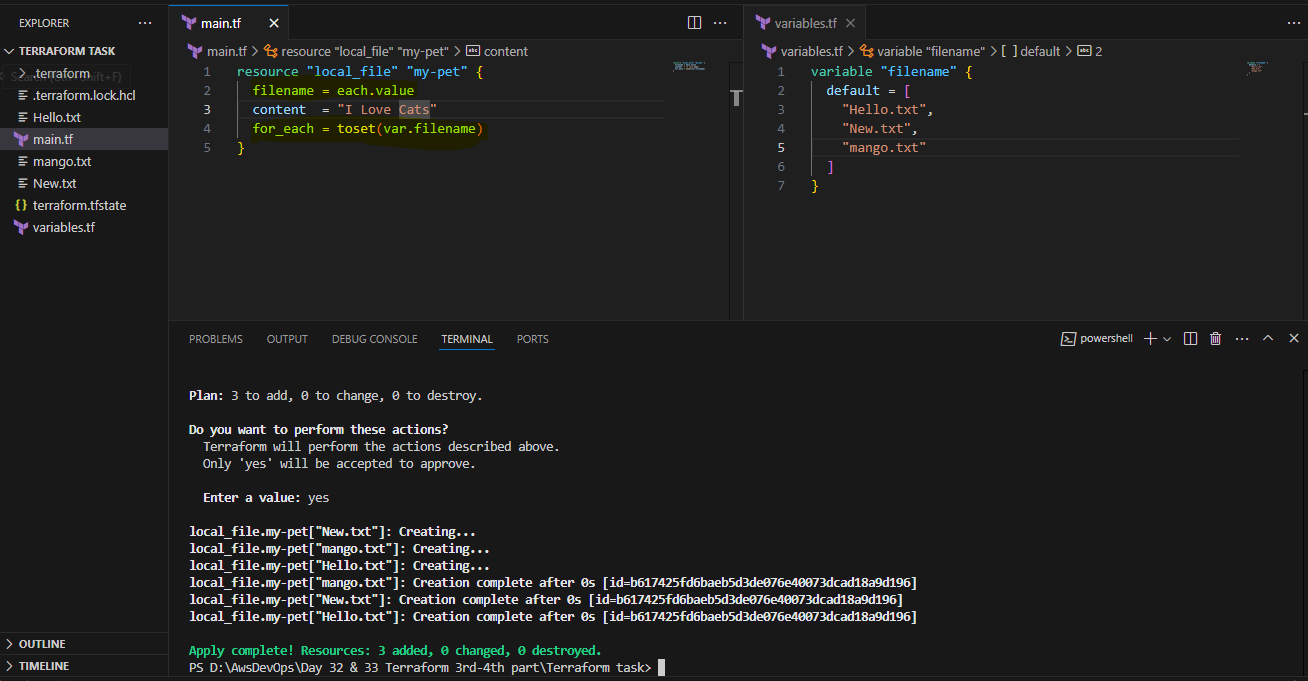
**Data source:**It is used to read the content of the infrastructure only to read the content  


Meta arguments:

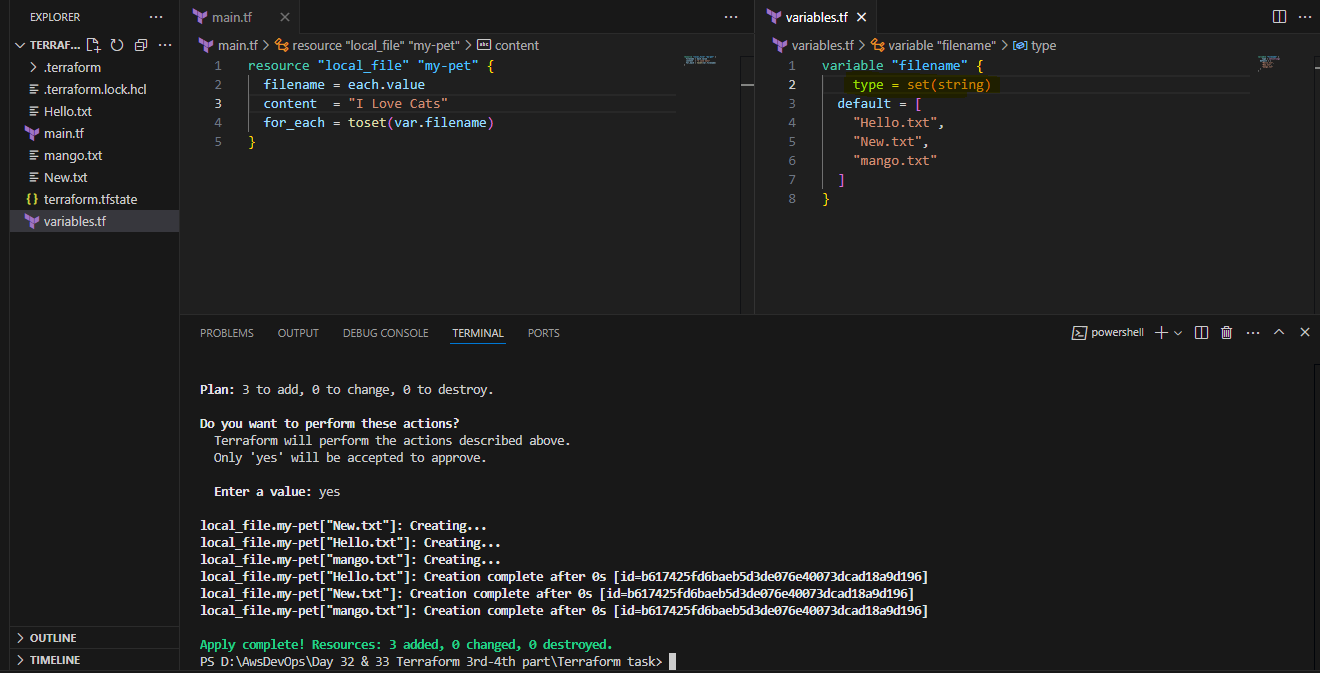
1) By using Count:  


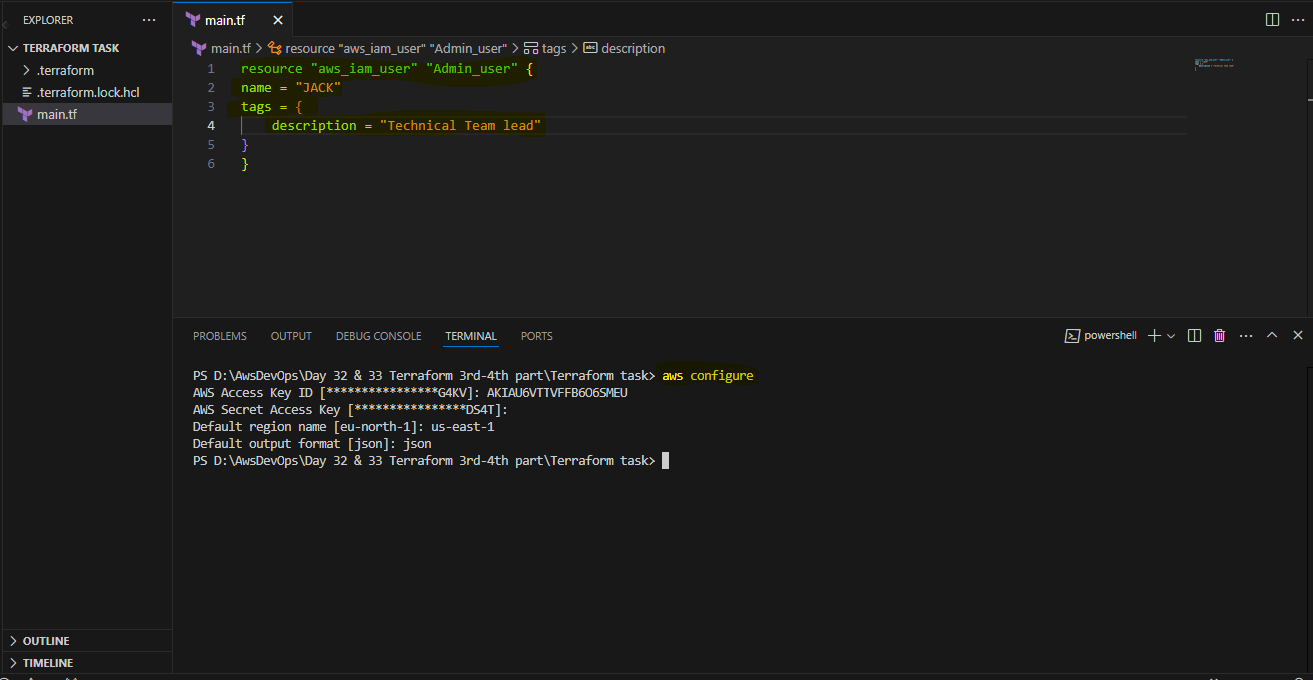
What if we don’t know the count we will use length function:  


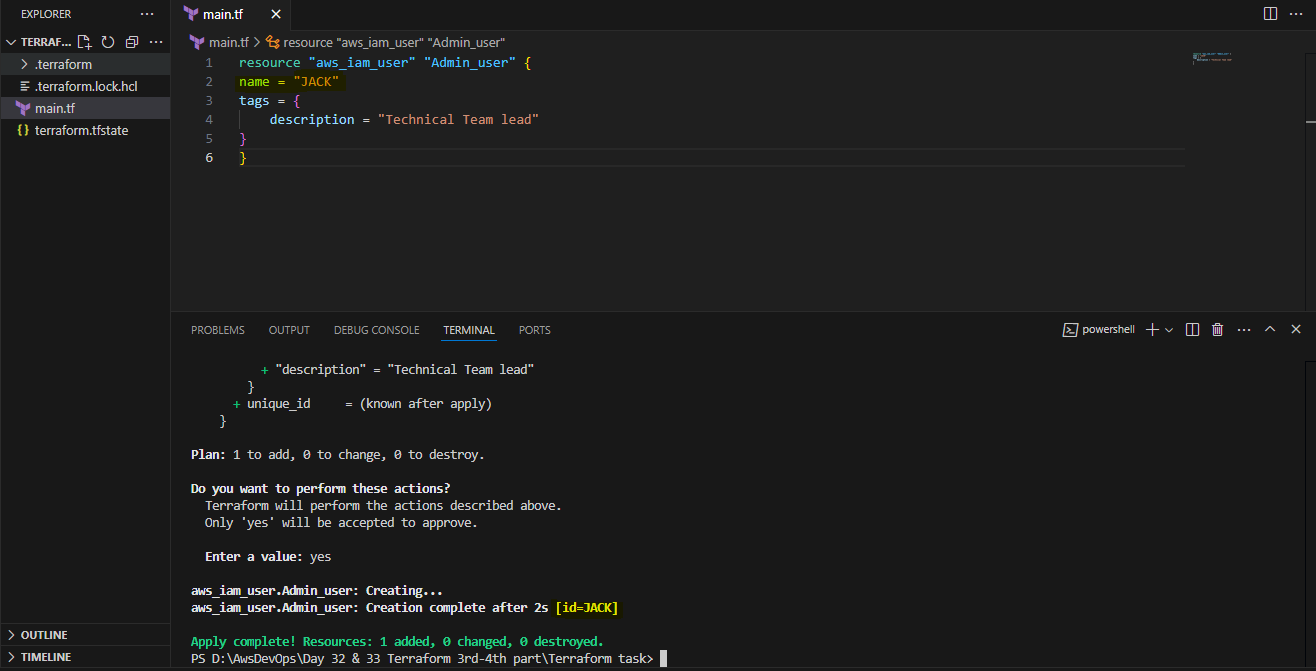
There’s one issue with **length** it will execute based on **index number** so to avoid such case we will use **for\_each** argument:

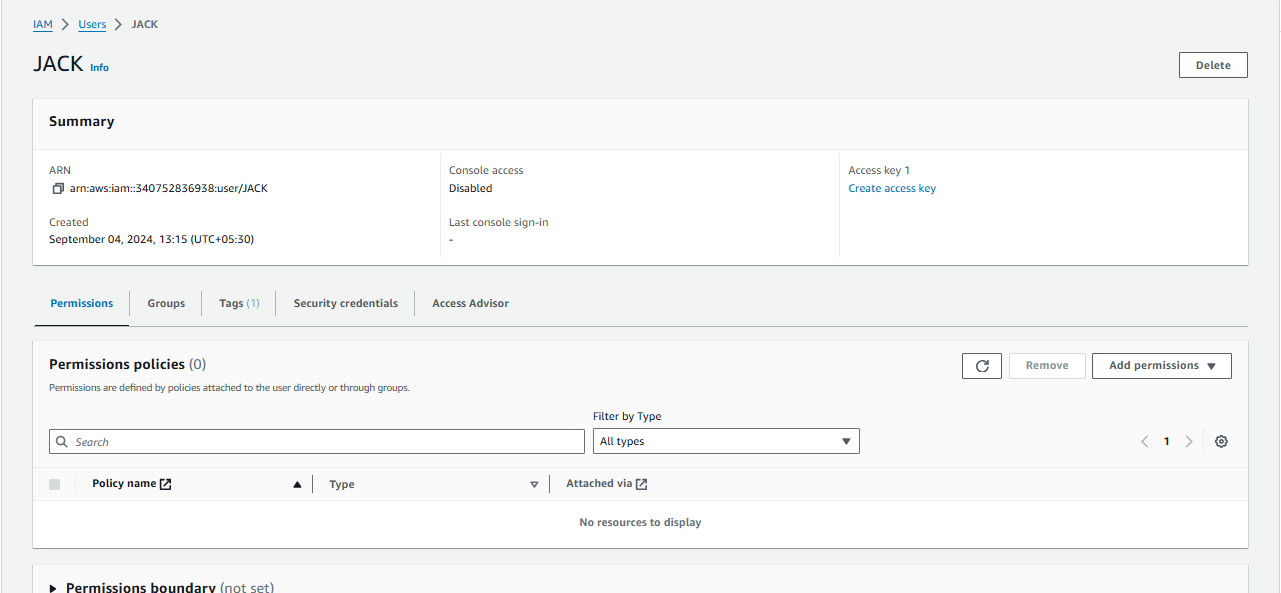


Also we can mention the **type** as well:

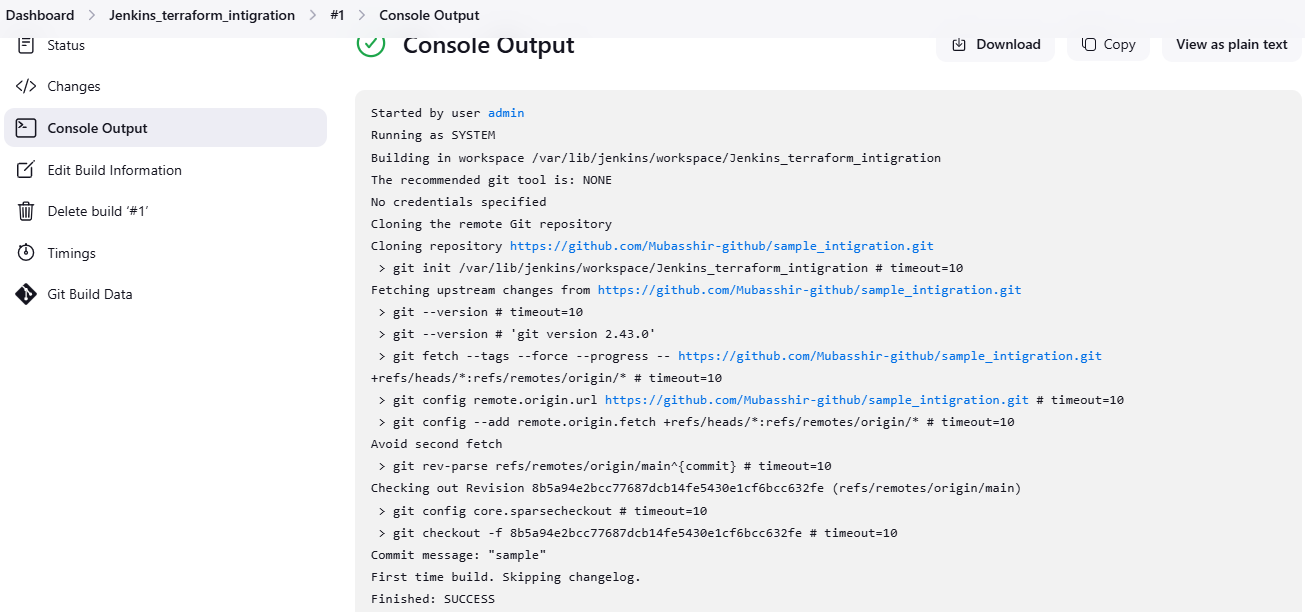


Let’stry to create an **AWS resource**: (Creating a user in aws) we need to configure the access key first  


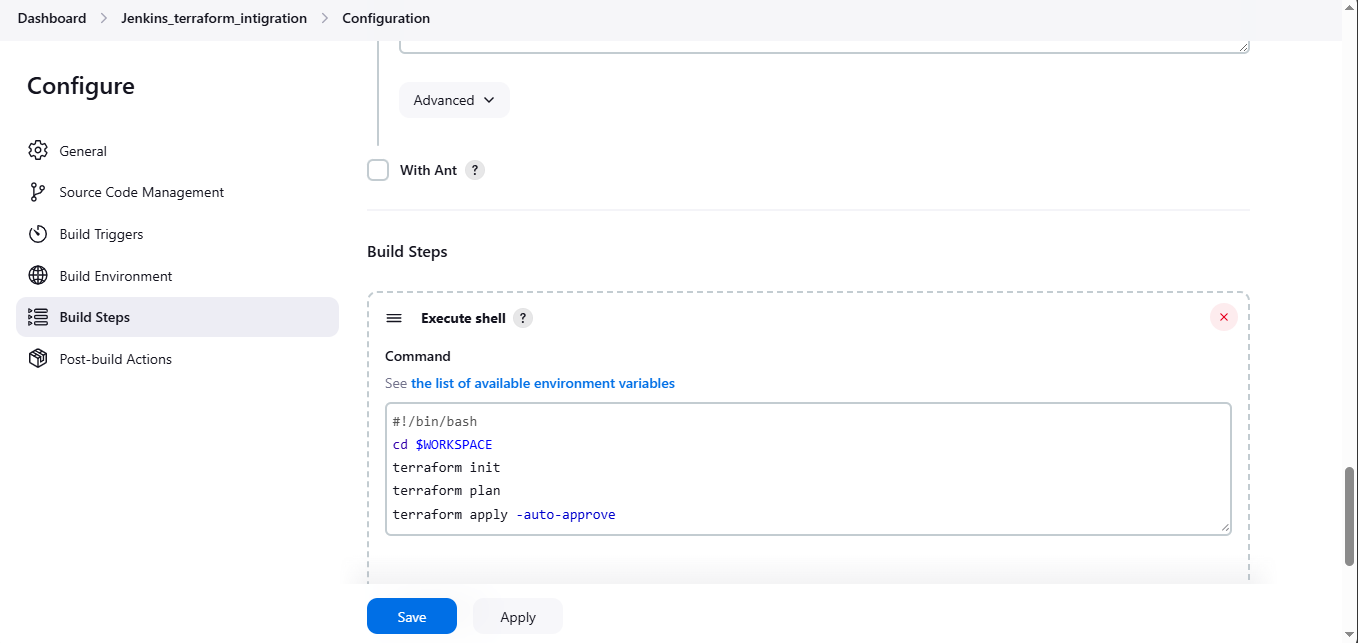
Created the user:  




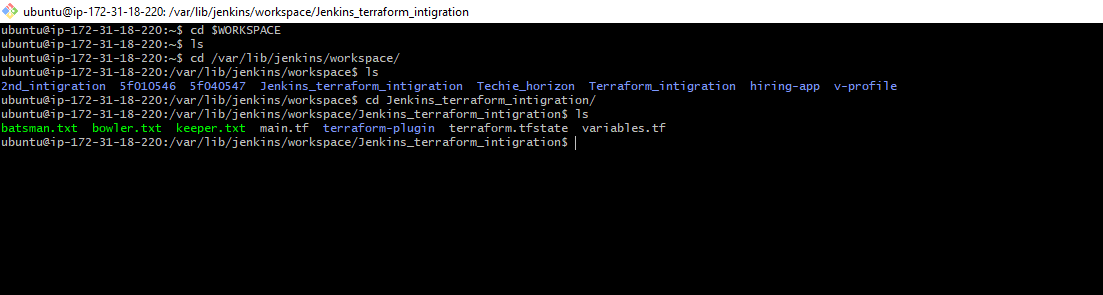
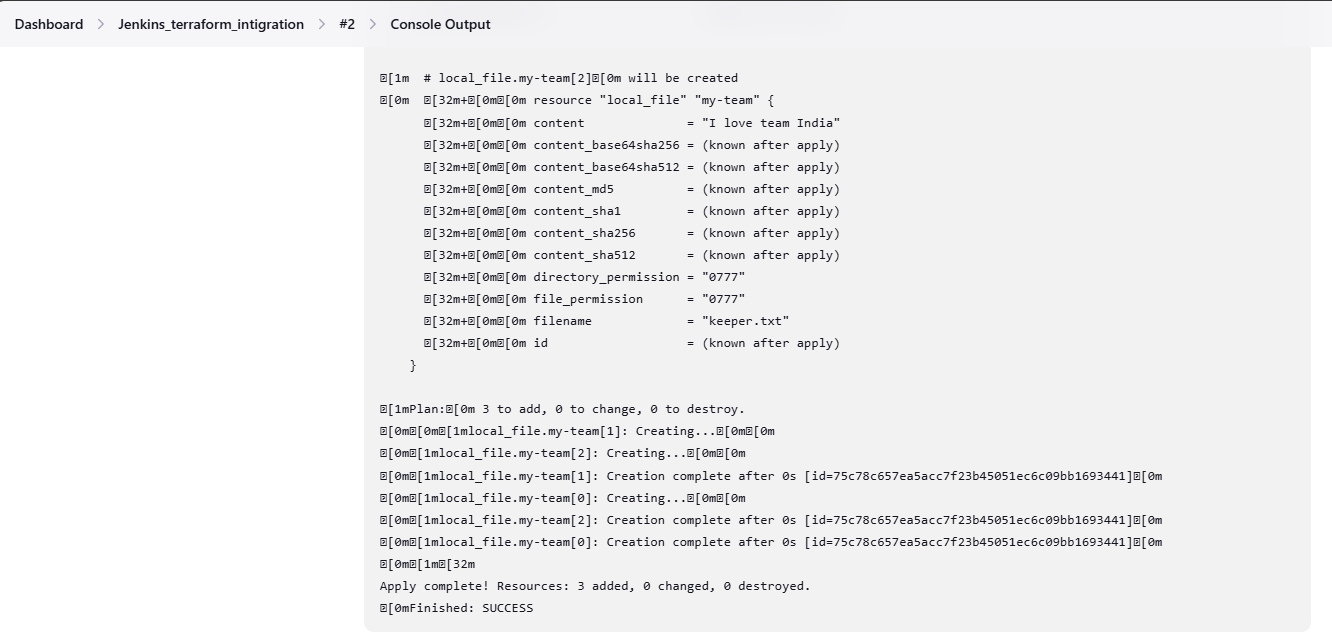
3) Integrate terrafrom in jenkins using Terraform plugin.



Installed the terraform in server also plugin in jnekins and attached to this job



Got the output as it got succeeded :



4) Create CICD pipeline for Nodejs Application.

<https://github.com/betawins/Trading-UI.git>

used a pipeline script:  
pipeline {

agent any

tools { nodejs "nodejs" }

stages {

stage('Build') {

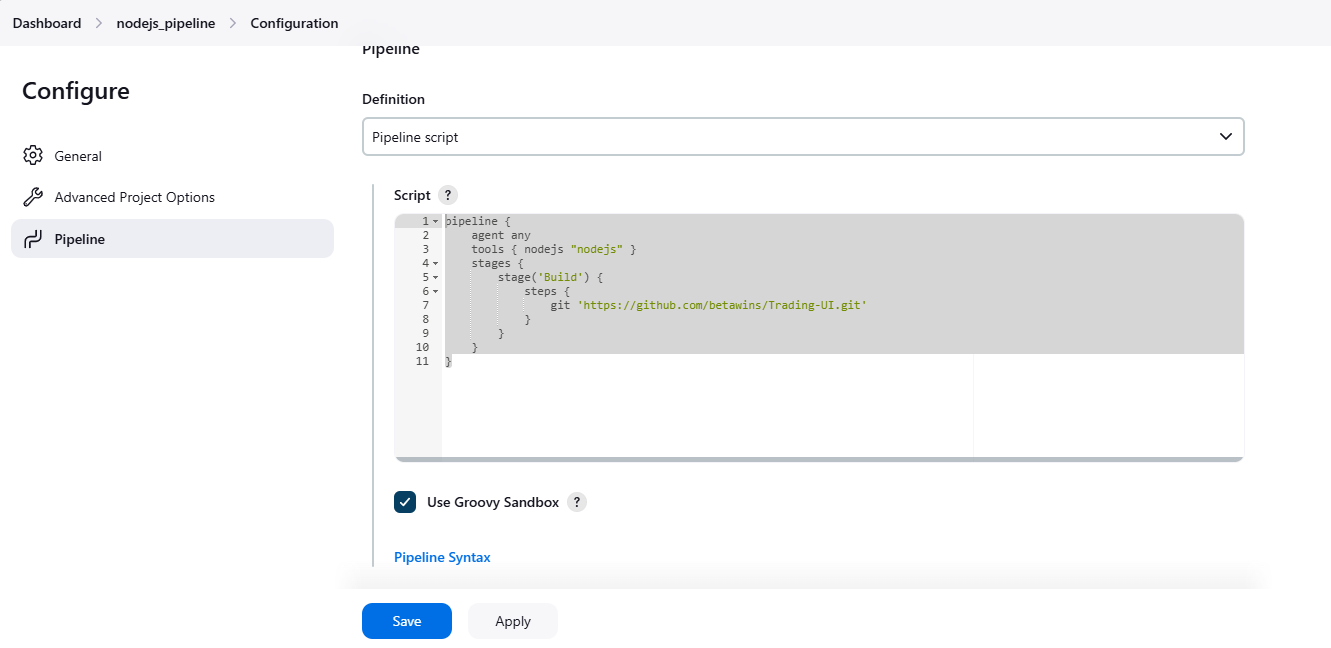
steps {

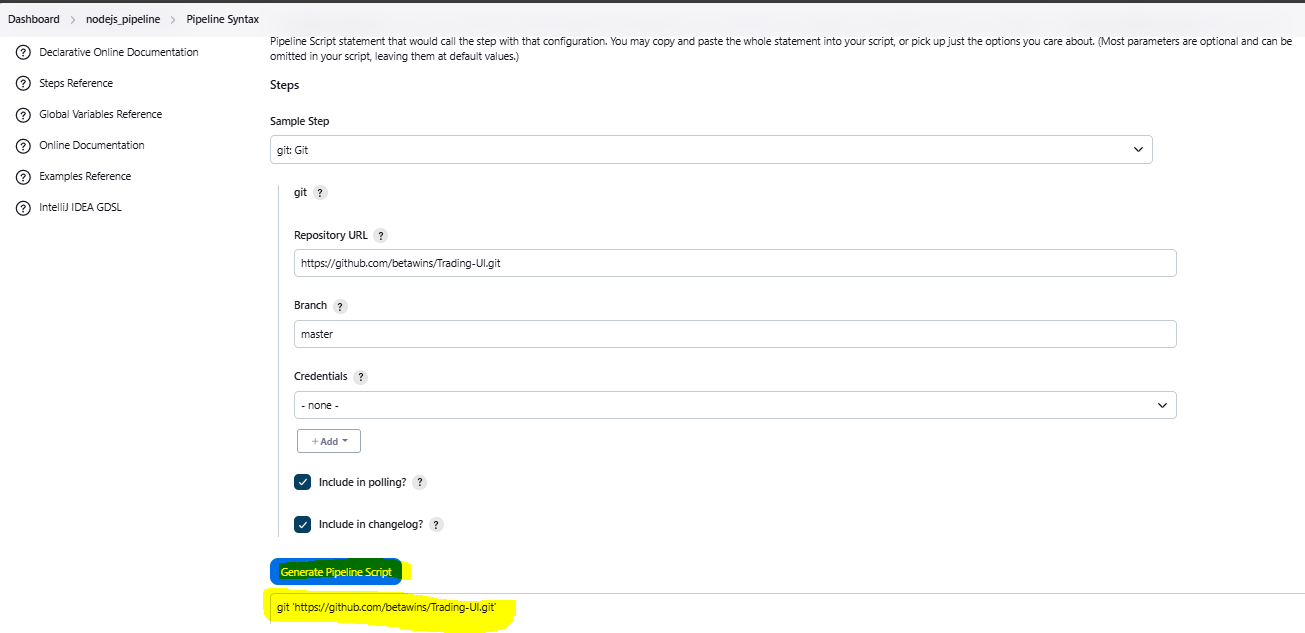
git 'https://github.com/betawins/Trading-UI.git'

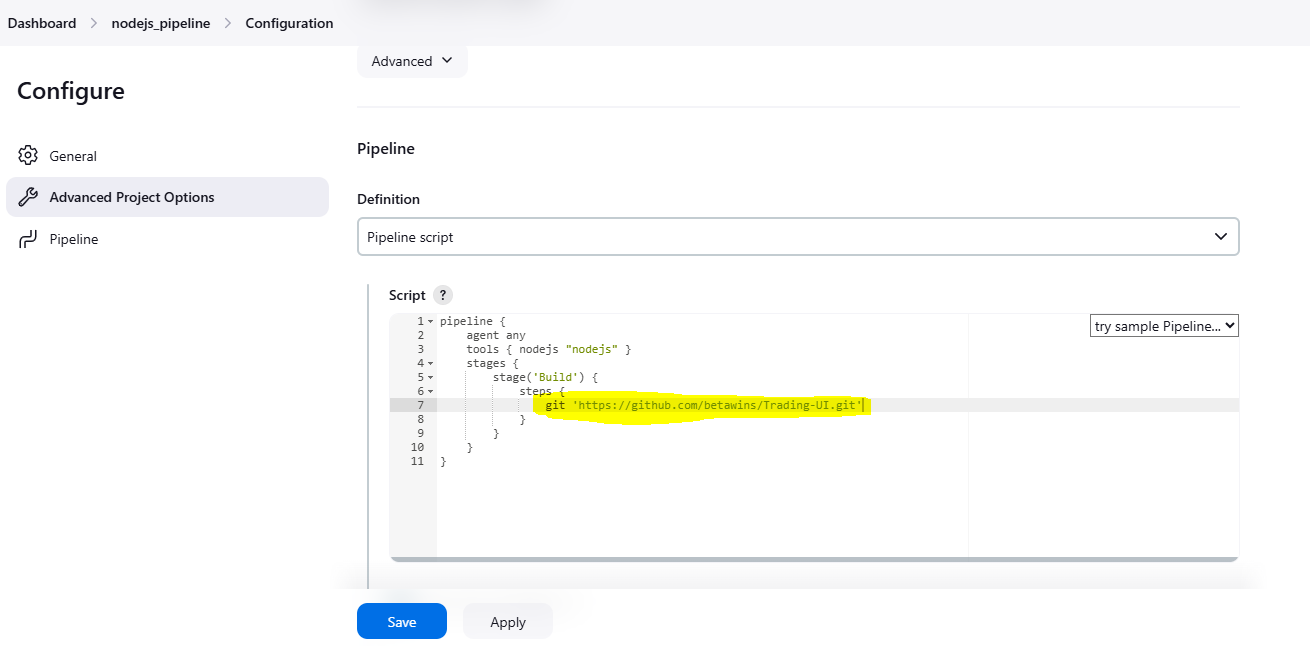
}

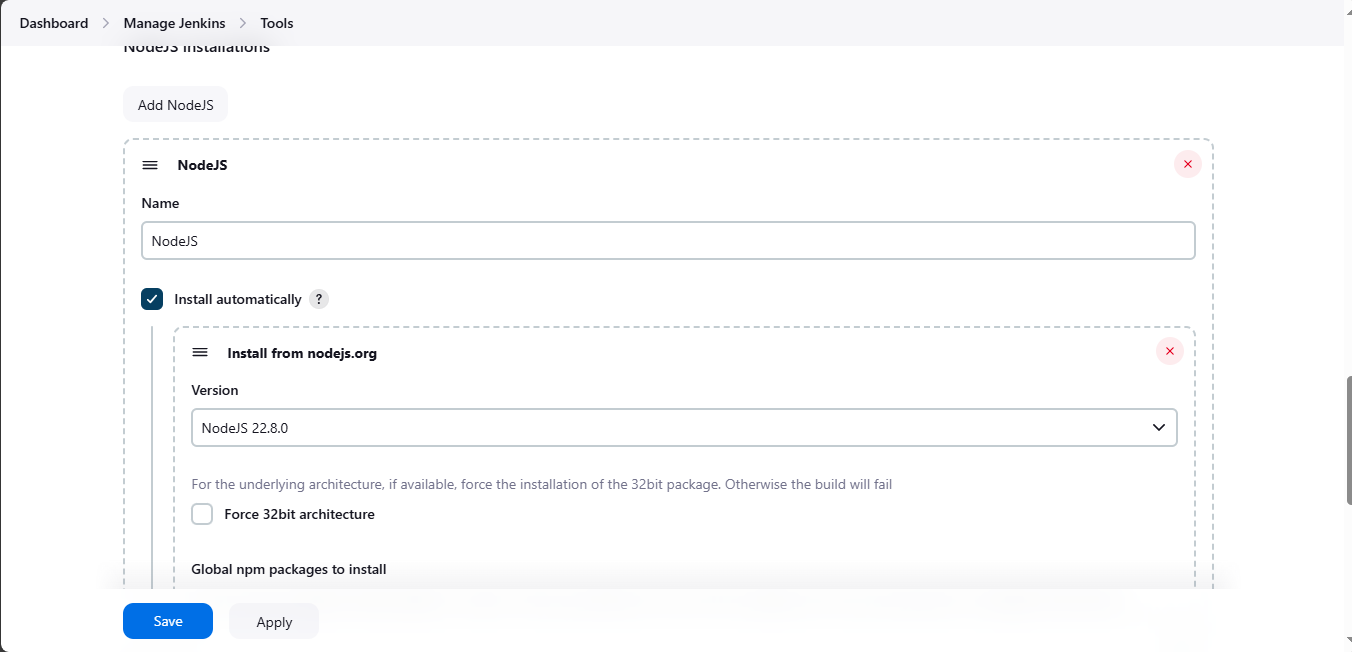
}

}

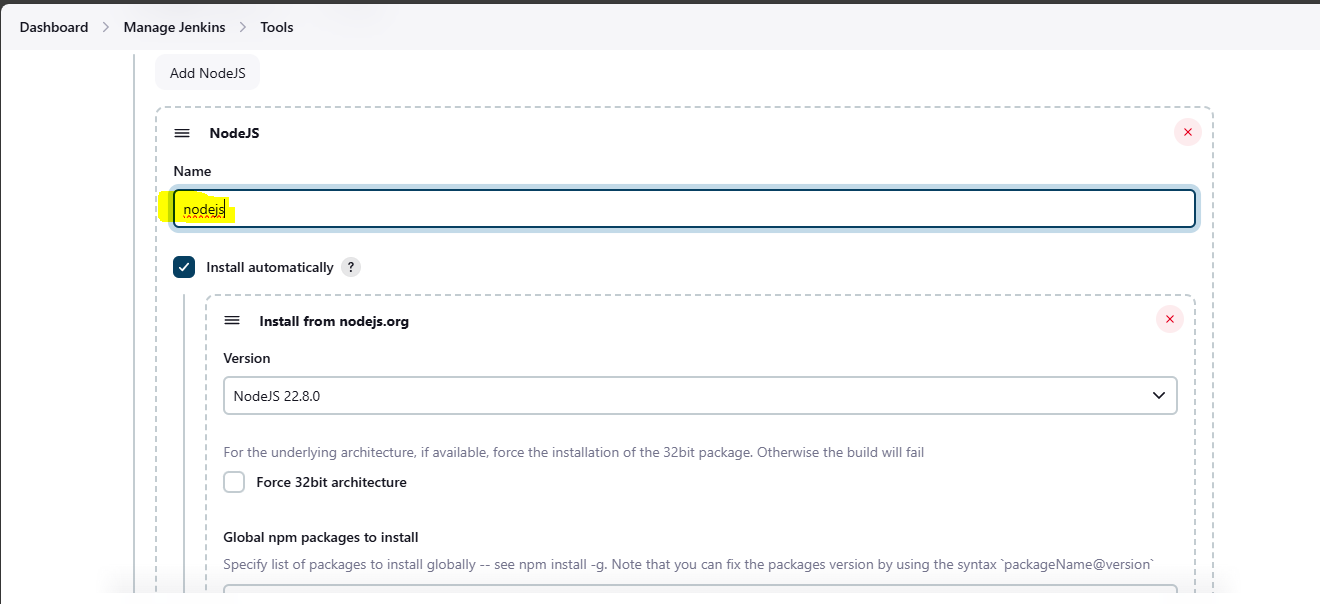
}  


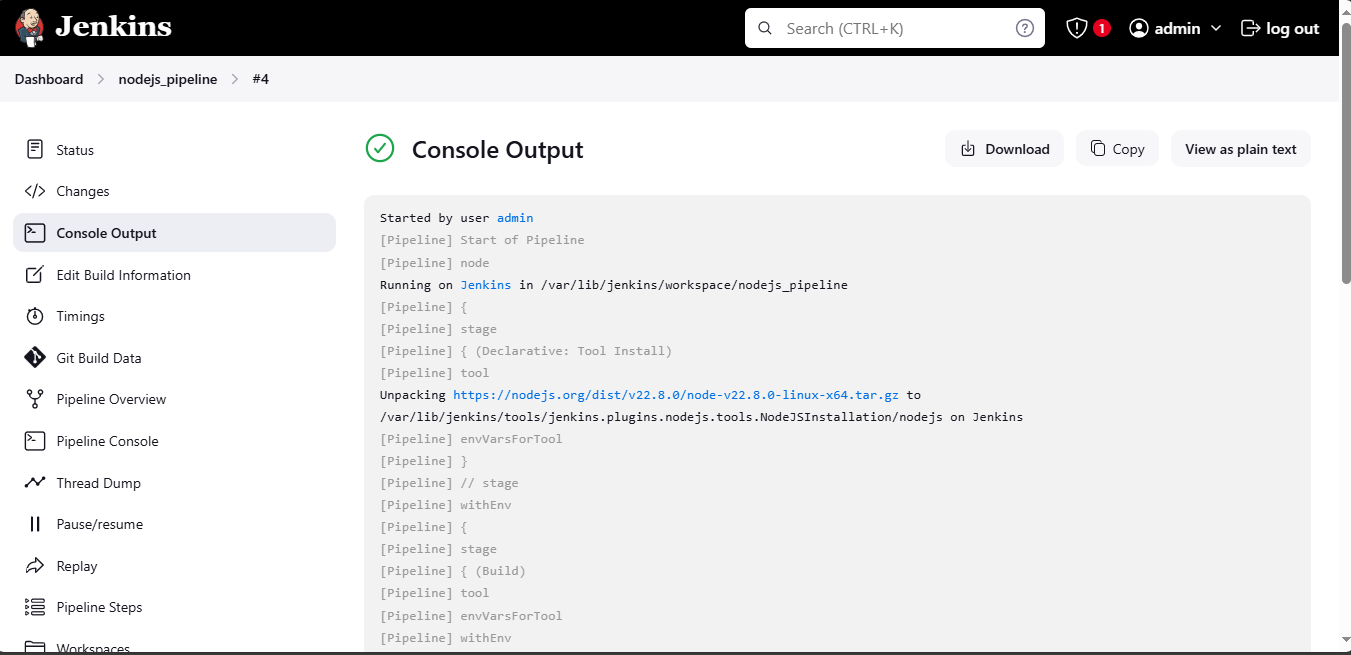
used the pipeline syntax for git clone and generated the pipeline script:  


added the link in to the pipeline script and save it: 

Install NodeJS plugin in out jenkins WebUI,Configure it in global tool configuration: 

Got error:

Solved it: 

Successfullty created the pipeline:  


5) Explain 10 Maven commands.

**1. mvn clean**

* **Purpose**: Removes all files generated by the previous build.
* **Usage**: mvn clean
* **Explanation**: This command deletes the target directory where Maven stores compiled classes, JARs, and other build artifacts. It’s often used to ensure that subsequent builds start from a clean slate.

**2. mvn compile**

* **Purpose**: Compiles the source code of the project.
* **Usage**: mvn compile
* **Explanation**: This command compiles the source code from the src/main/java directory and places the compiled classes in the target/classes directory.

**3. mvn test**

* **Purpose**: Runs the unit tests of the project.
* **Usage**: mvn test
* **Explanation**: This command executes the tests located in the src/test/java directory using a testing framework like JUnit or TestNG. The results are placed in the target/test-classes and target/surefire-reports directories.

**4. mvn package**

* **Purpose**: Compiles the code and packages it into a JAR, WAR, or another artifact type.
* **Usage**: mvn package
* **Explanation**: This command performs the compile and test phases and then packages the compiled code into a distributable format (such as JAR or WAR) in the target directory.

**5. mvn install**

* **Purpose**: Compiles, tests, and packages the project and installs the artifact into the local repository.
* **Usage**: mvn install
* **Explanation**: This command performs the build lifecycle phases (clean, compile, test, package) and installs the resulting artifact (JAR/WAR) into the local Maven repository (~/.m2/repository). This allows other projects on the same machine to use the artifact as a dependency.

**6. mvn deploy**

* **Purpose**: Deploys the artifact to a remote repository.
* **Usage**: mvn deploy
* **Explanation**: This command is used in continuous integration environments to deploy the packaged artifact to a remote repository (such as Nexus or Artifactory) so that it can be shared with other developers or used in other projects.

**7. mvn site**

* **Purpose**: Generates a site for the project.
* **Usage**: mvn site
* **Explanation**: This command creates a site for the project using the information from the pom.xml file. It generates reports and documentation (e.g., Javadoc, test coverage) and places them in the target/site directory.

**8. mvn validate**

* **Purpose**: Validates the project’s configuration.
* **Usage**: mvn validate
* **Explanation**: This command validates the project’s pom.xml and other configurations to ensure that they are correct and that all required parameters are available before starting the build.

**9. mvn dependency:tree**

* **Purpose**: Displays the project's dependency tree.
* **Usage**: mvn dependency:tree
* **Explanation**: This command shows a tree of all dependencies in the project, including transitive dependencies. It helps to visualize and debug dependency relationships and conflicts.

**10. mvn help:effective-pom**

* **Purpose**: Displays the effective pom.xml for the project.
* **Usage**: mvn help:effective-pom
* **Explanation**: This command generates and displays the effective pom.xml file after applying all inherited and overridden configurations. It’s useful for debugging configuration issues by showing the final configuration that Maven uses.