**Terraform Task-3**

1. **Watch the Terraform-03 video.**

**Completed watching video**

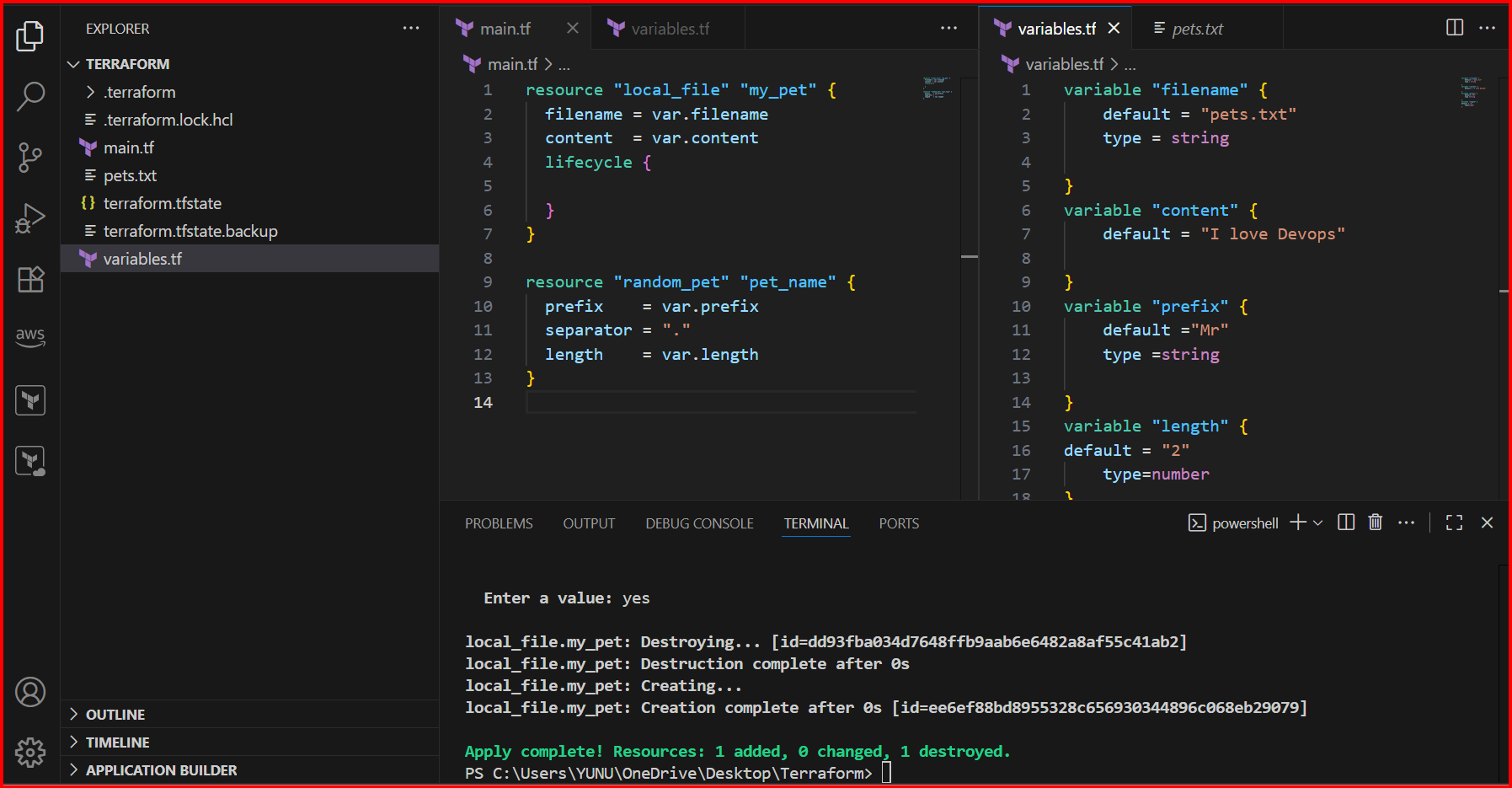
1. **Execute the script shown in the video**

**Varibles.tf**

**variables.tf is a file where we define input variables for a Terraform project.**

**Instead of hardcoding values,we declare them as variables.**

**This makes Terraform code reusable, dynamic, and easy to manage.**

****

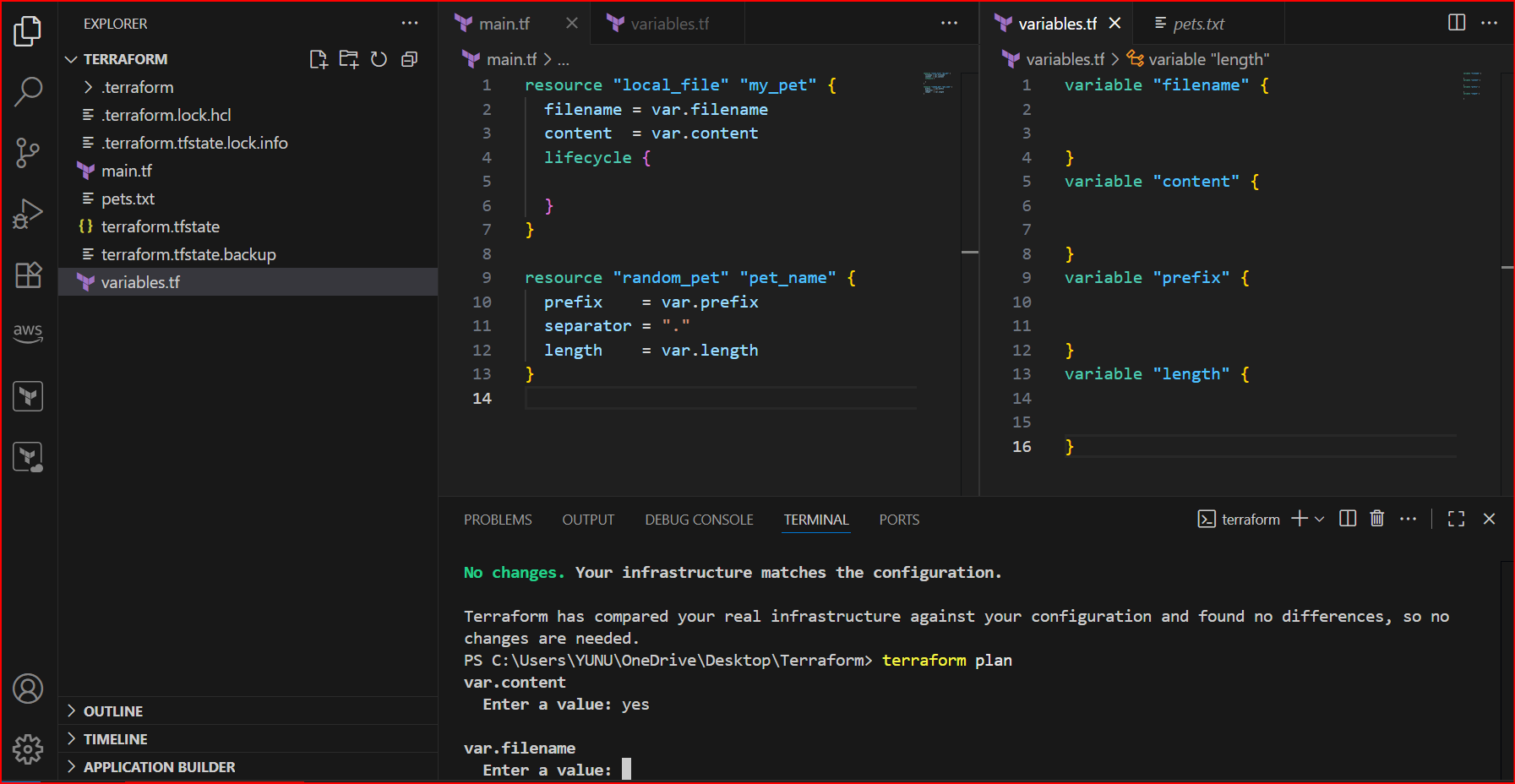
**Interactive mode:**

**Interactive mode happens when Terraform asks you for variable values at runtime (while running terraform plan or terraform apply) because:**

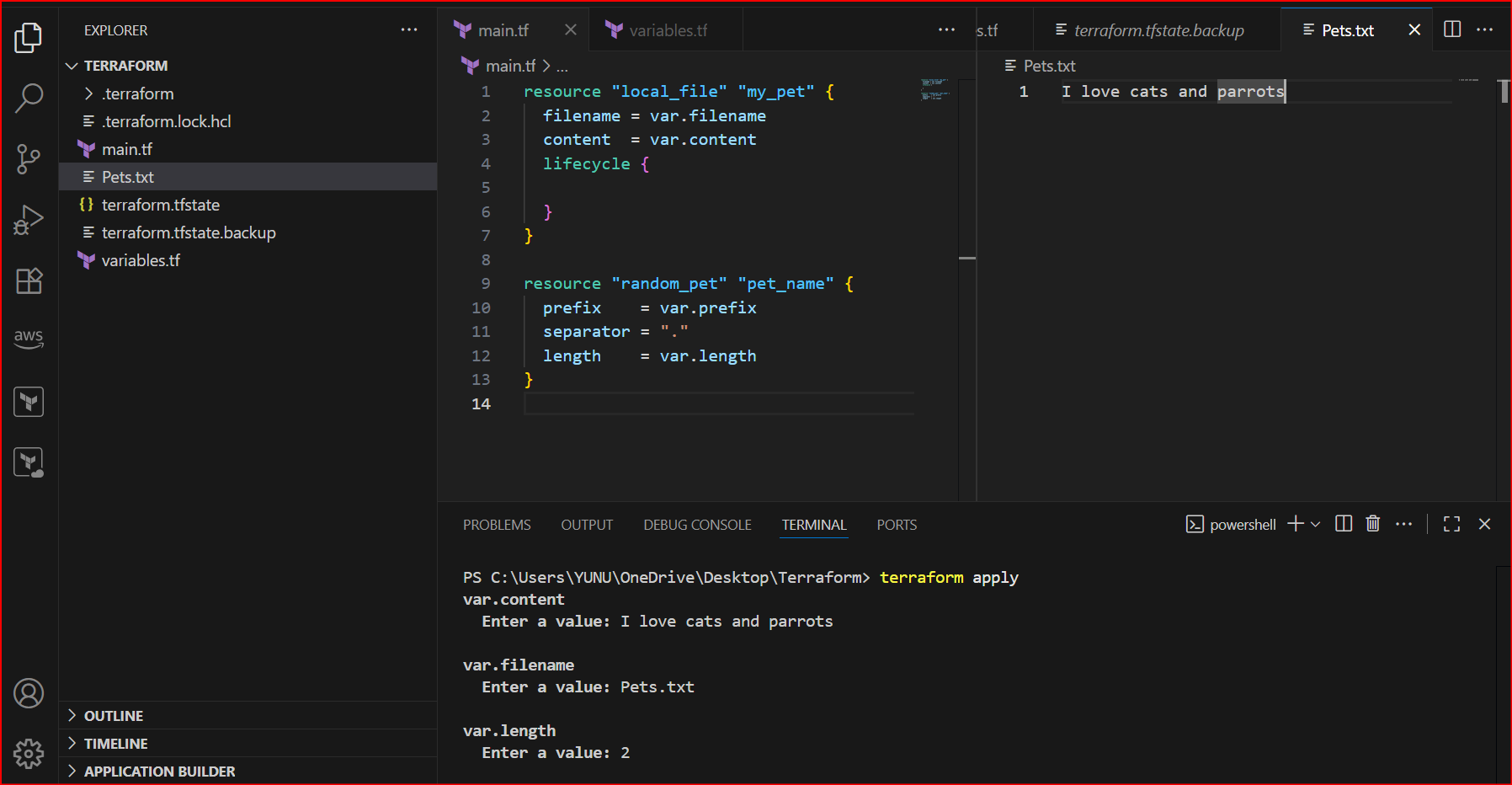
**The variable has no default value**

**You didn’t provide a value via .tfvars, CLI, or environment variables**

**In other words, Terraform will pause execution and prompt you to enter input.**

****

**This will ask me to enter the values**

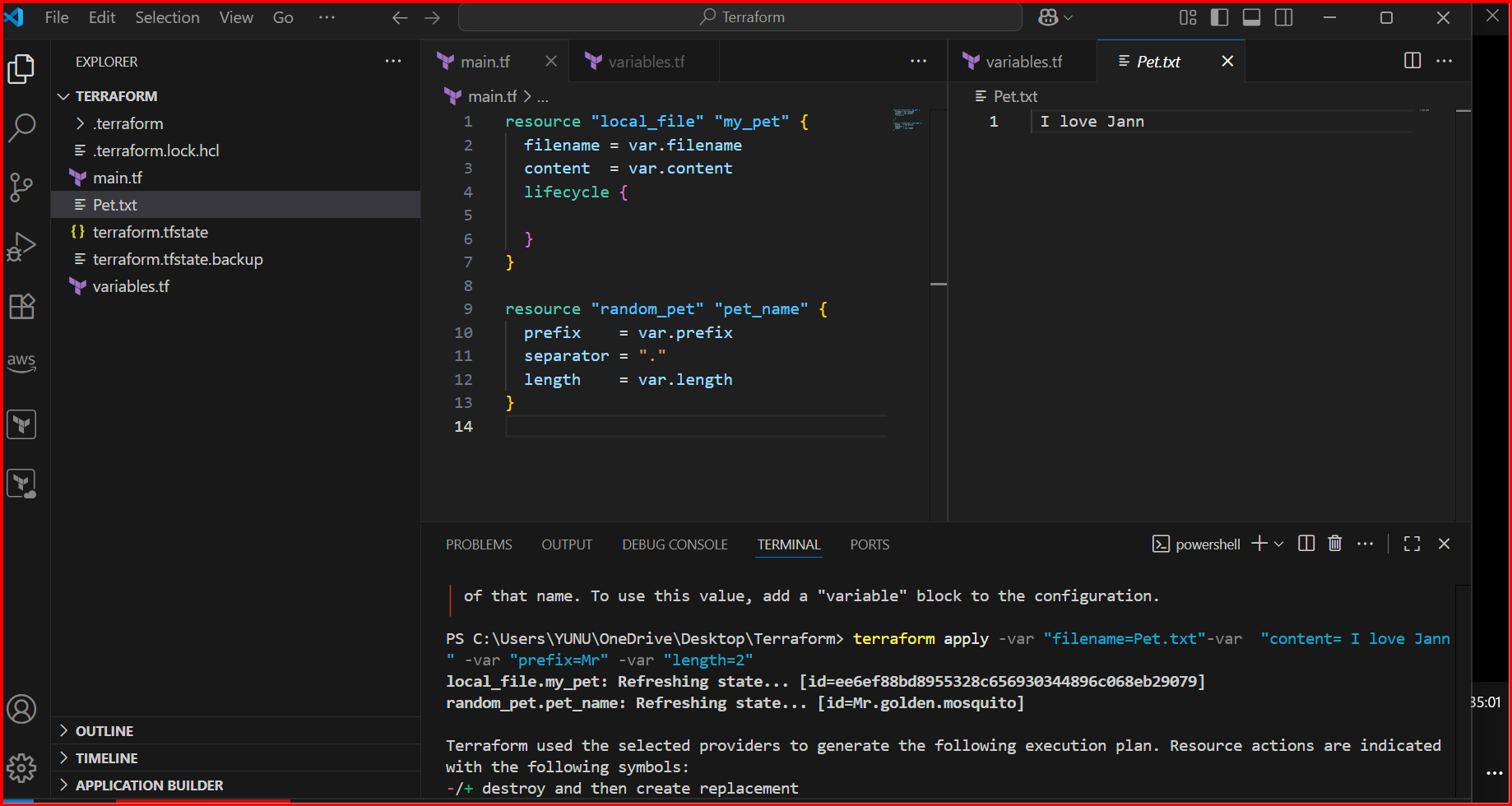
****

**Command line flag:**

**You can pass variables to Terraform directly when running commands like terraform plan or terraform apply using the -var flag.**

**Command:**

**terraform apply -var "filename=Pet.txt"-var "content= I love Jann" -var "prefix=Mr" -var "length=2"**

****

**Environment Variables:**

**In Terraform, environment variables are a way to provide values to input variables without hardcoding them in .tf files or .tfvars.**

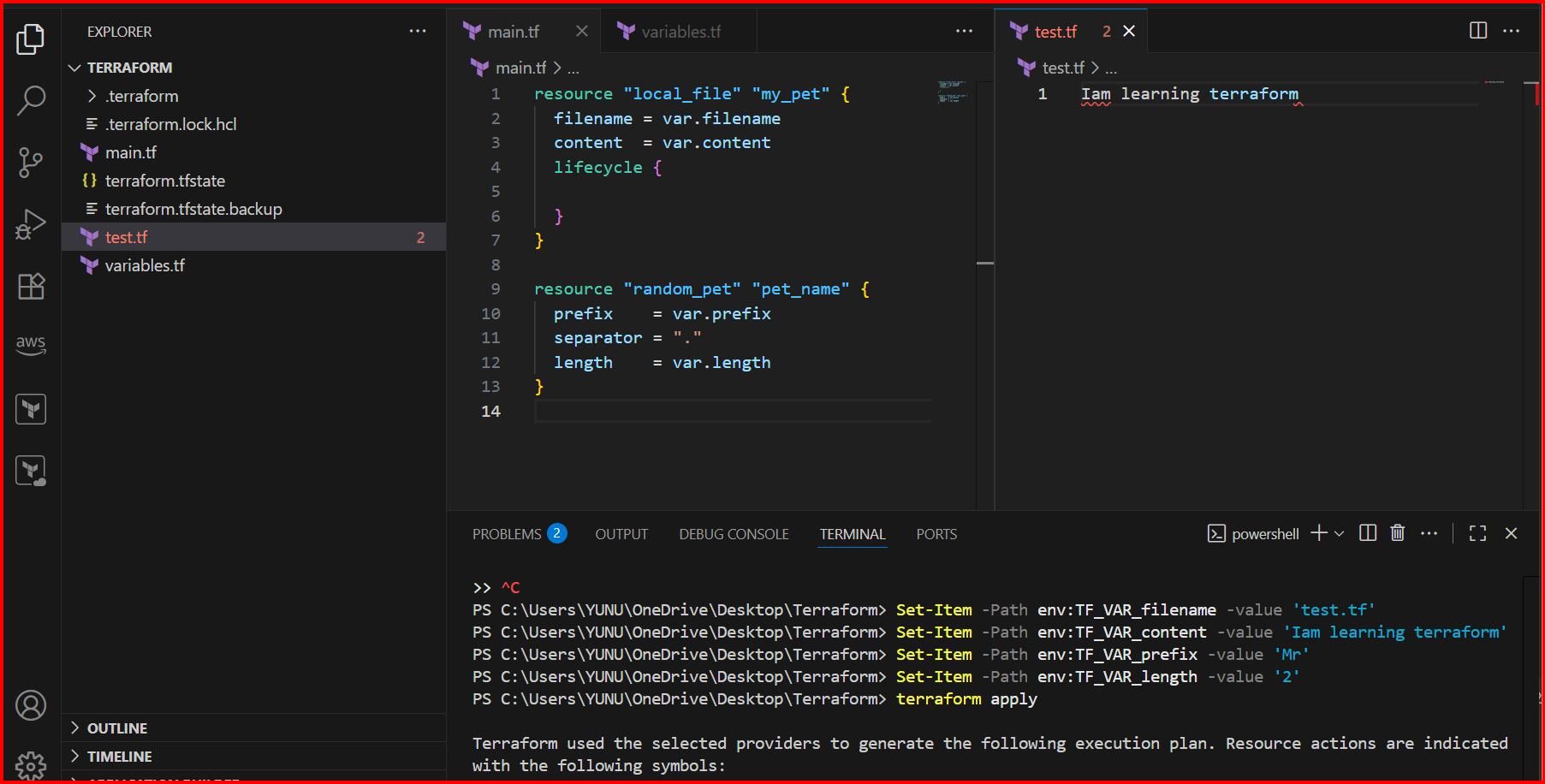
**Terraform looks for environment variables in the format:**

**Linux**

**export TF\_VAR\_aws\_region="ap-south-1"**

**Windows**

**Set-Item -Path env :TF\_VAR\_filename -var ‘test.tx’**

****

**Definition precedence:**

**If we use multiple ways to define varibles for the same file then terraform uses varible definition precedence.**

**Precedence order:**

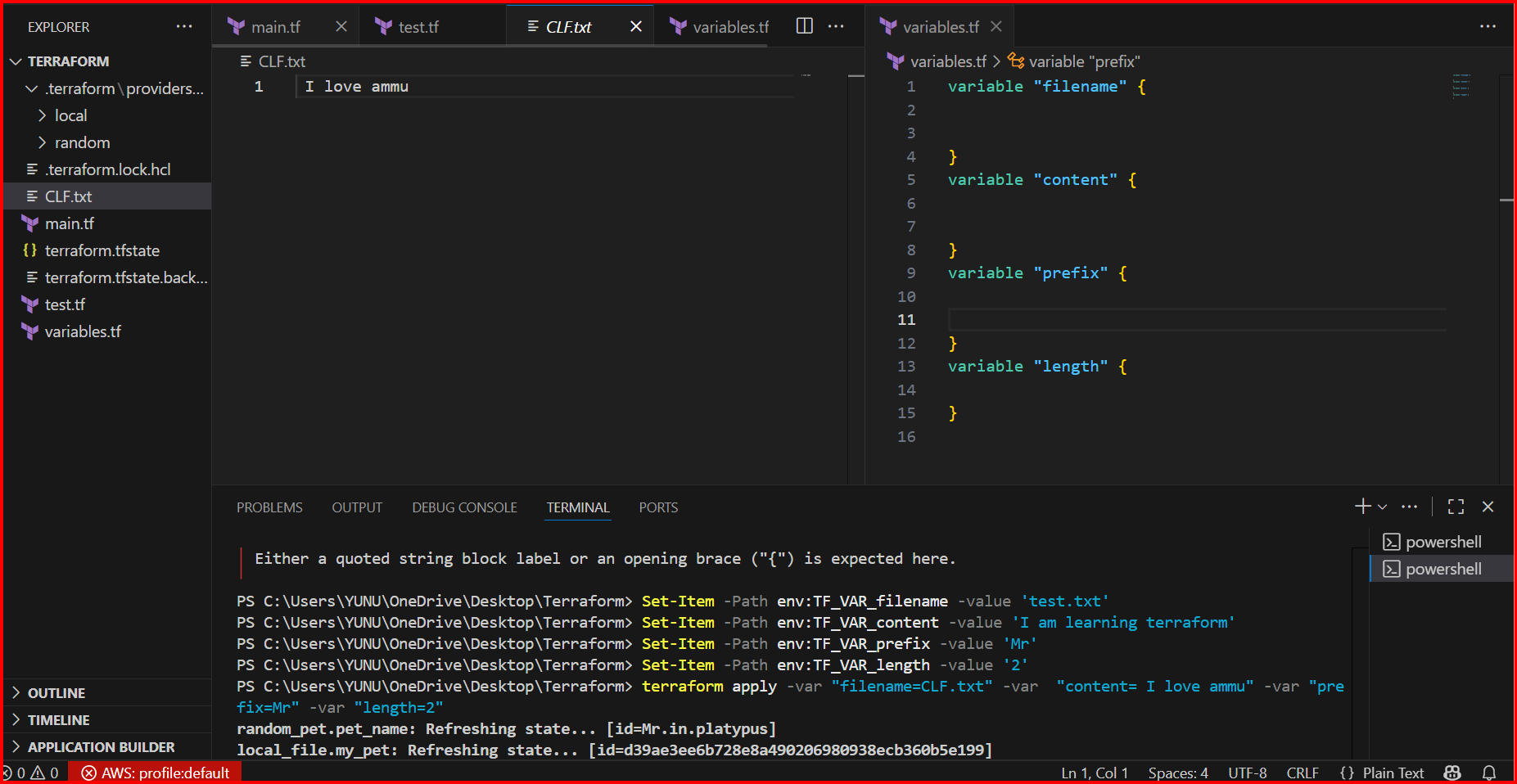
**Order Option**

**1 Environment variables**

**2 Terraform.tfvars**

**3 \*.auto.tfvars(alphabetical order)**

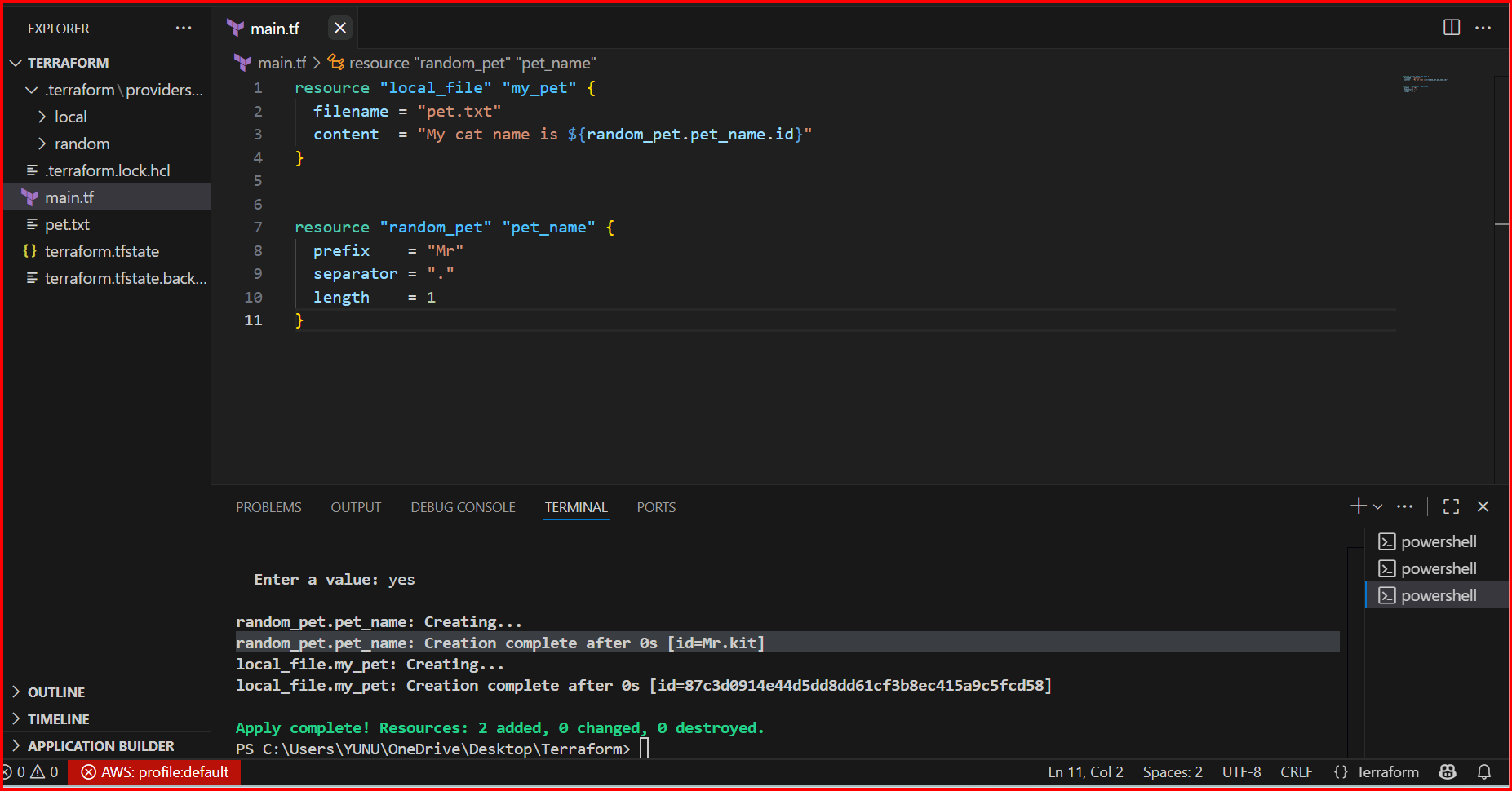
**4 -var or -var-file (Command line flags)**

****

**What is Implicit Behaviour in Terraform?**

**Terraform automatically figures out the order of resource creation by analyzing references between resources.  
You don’t always need to manually define depends\_on — Terraform builds a dependency graph from your code.**

**👉 This automatic ordering is called implicit dependency management (or implicit behaviour).**

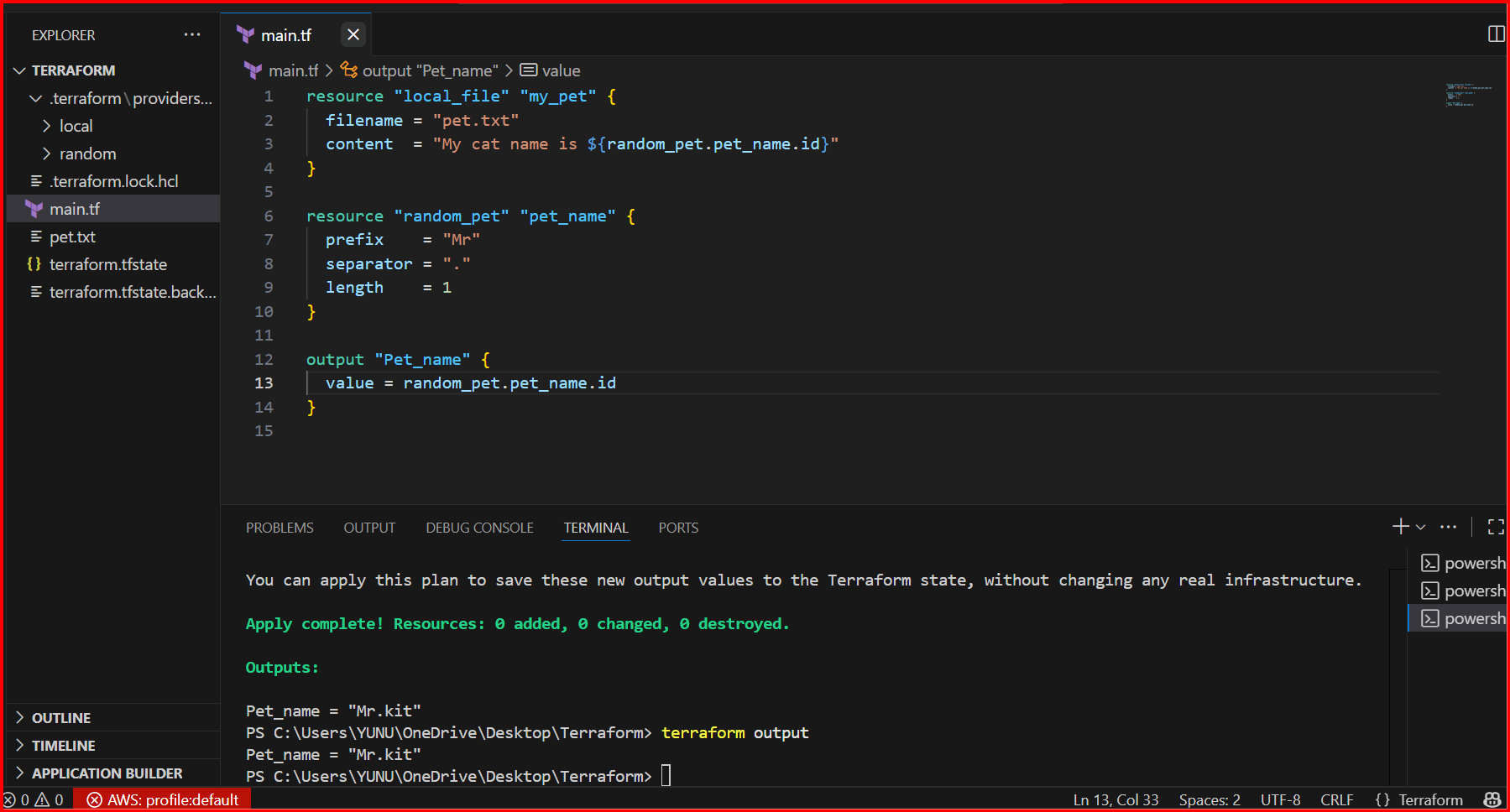
****

**Terraform out:**

**The terraform output command is used to display the values of outputs that you have defined in your Terraform configuration.**

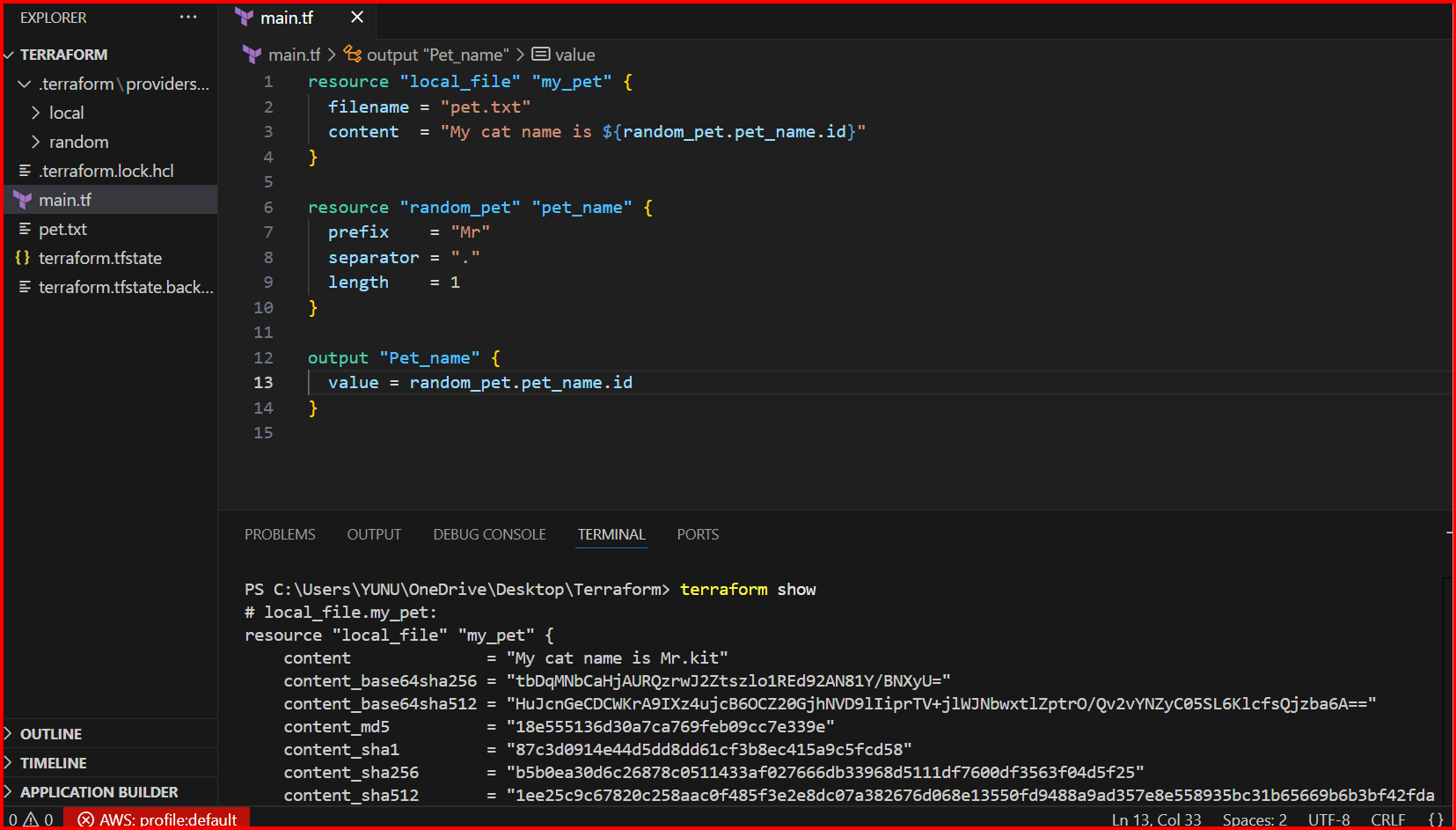
**Shows the values from your output blocks after terraform apply has been run.**

**Useful for retrieving important info like IP addresses, DNS names, resource IDs, passwords, etc.**

****

**Terraform show:**

**The terraform show command displays the full Terraform state — meaning all resources managed by Terraform and their current attributes.**

****

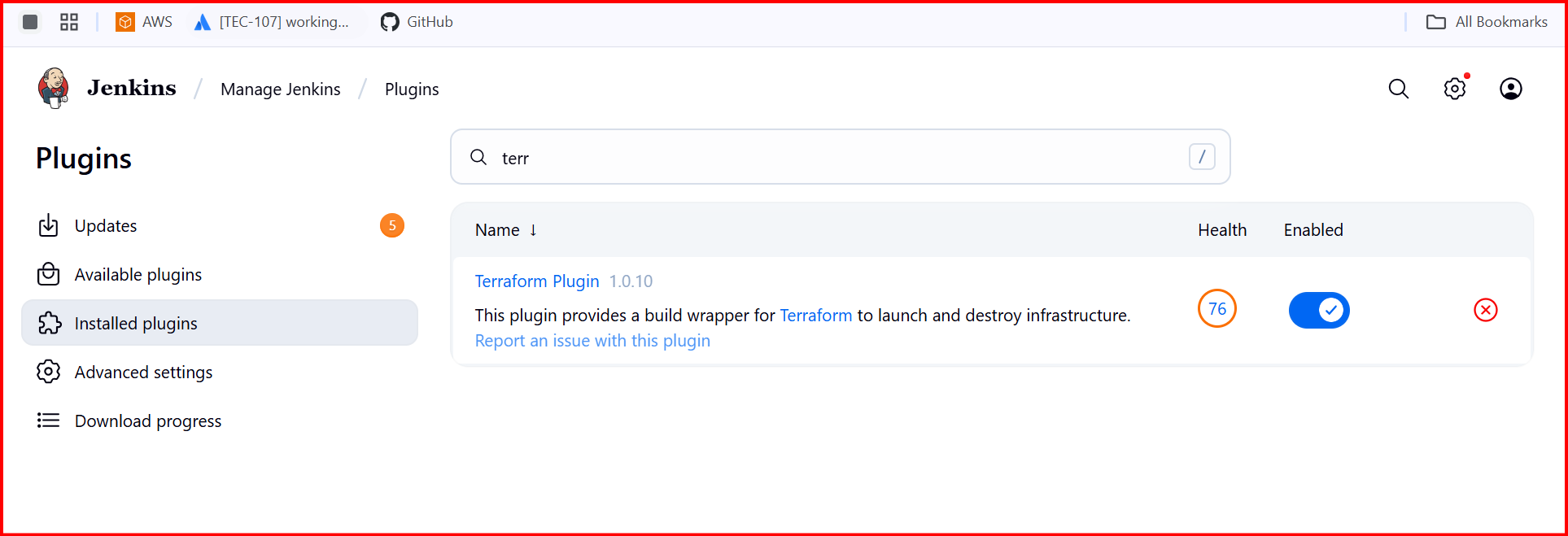
**3.Integrate Terraform in Jenkins using the Terraform plugin.**

**Install Jenkins**

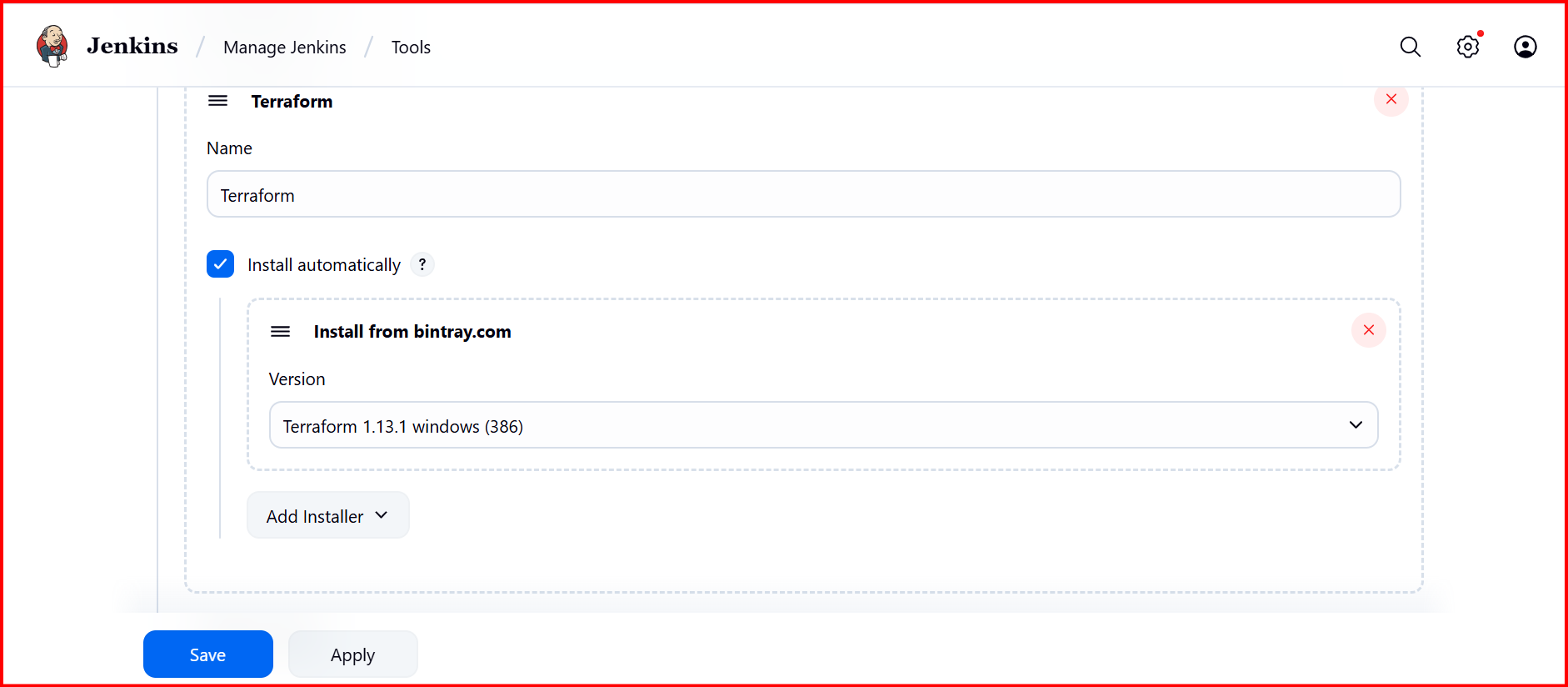
**Install terraform plugin in Jenkins**

Jenkins → **Manage Jenkins** → **Manage Plugins** → **Available**.

Search **Terraform** → Install (restart if asked).

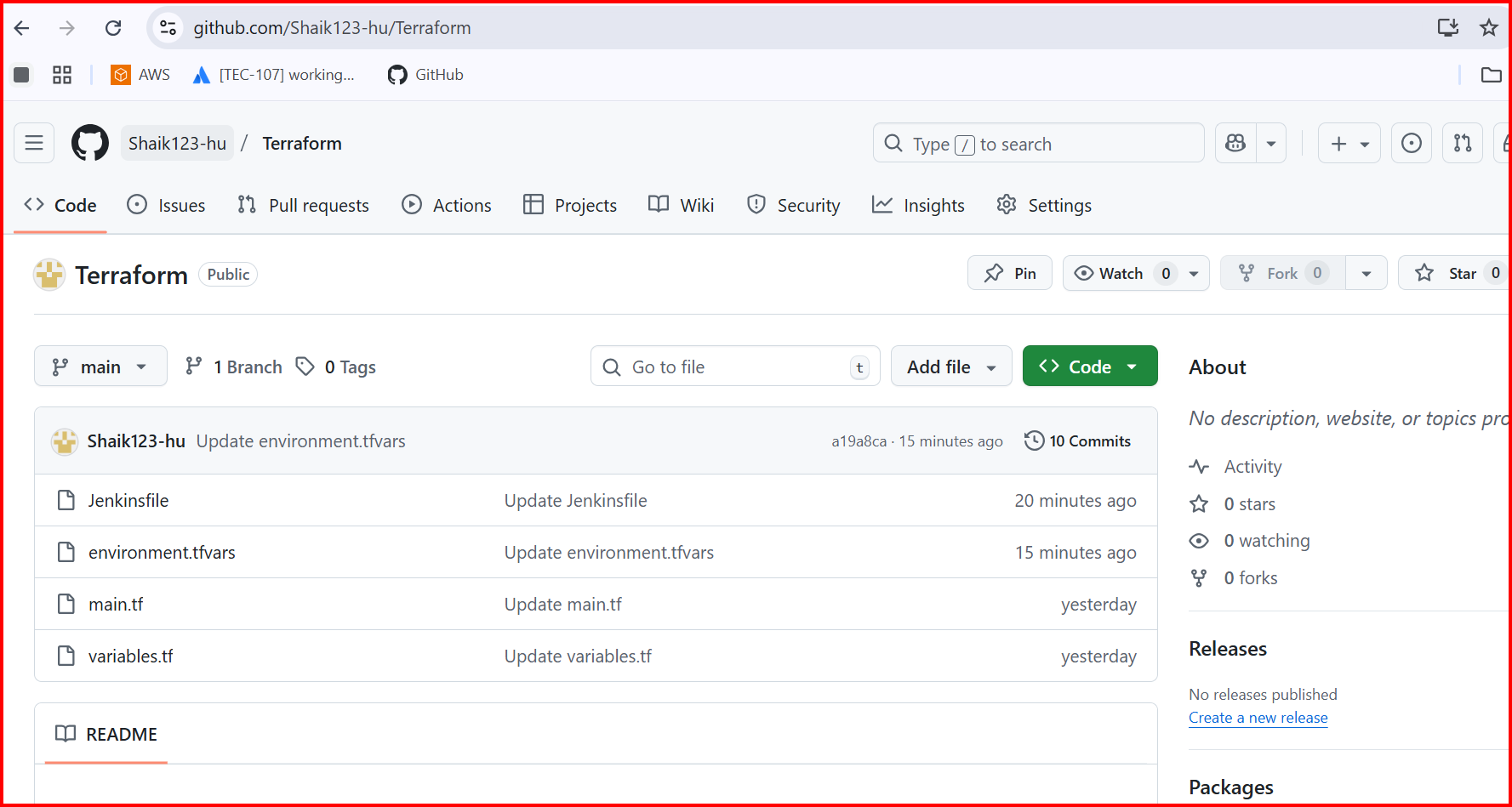
****

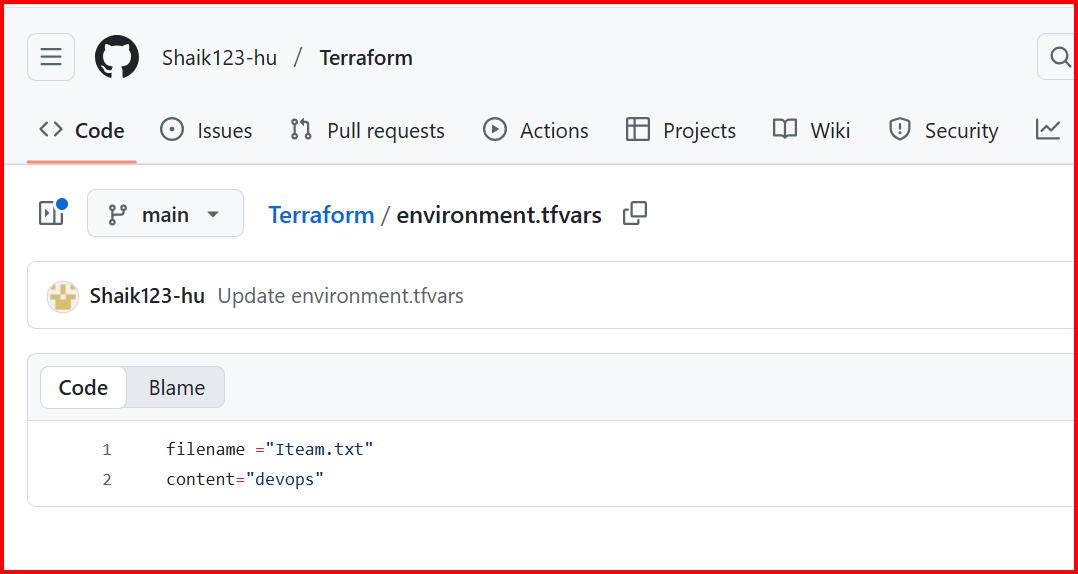
**Jenkins management 🡪 Tools 🡪 Configure the Terraform**

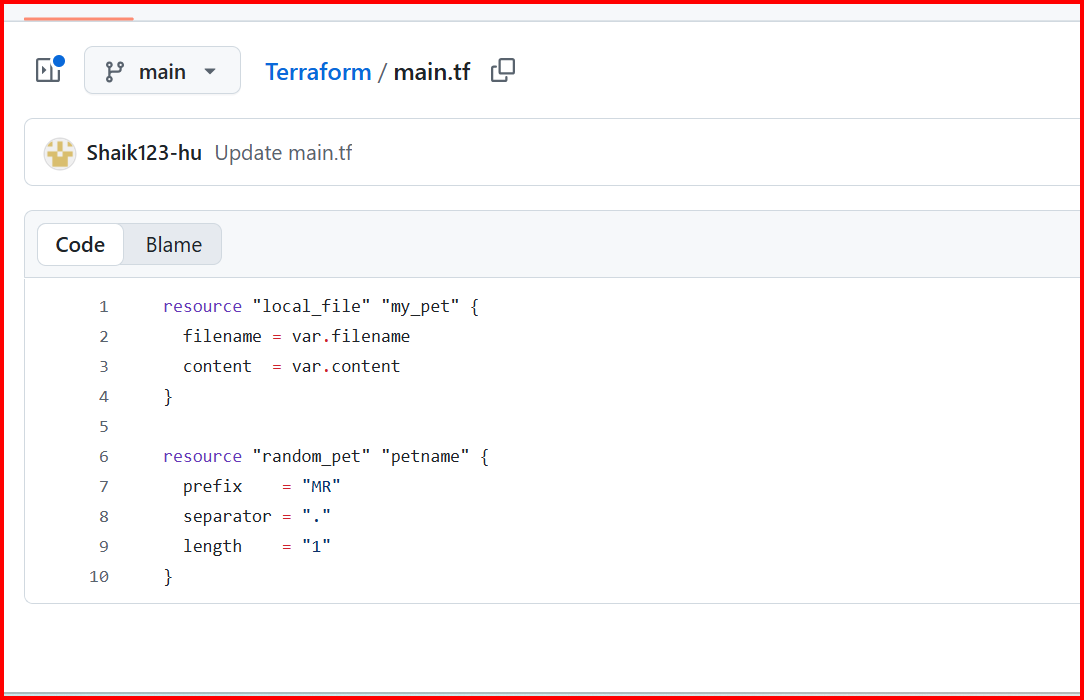
****

**GIT REPO:**

[**https://github.com/Shaik123-hu/Terraform**](https://github.com/Shaik123-hu/Terraform)

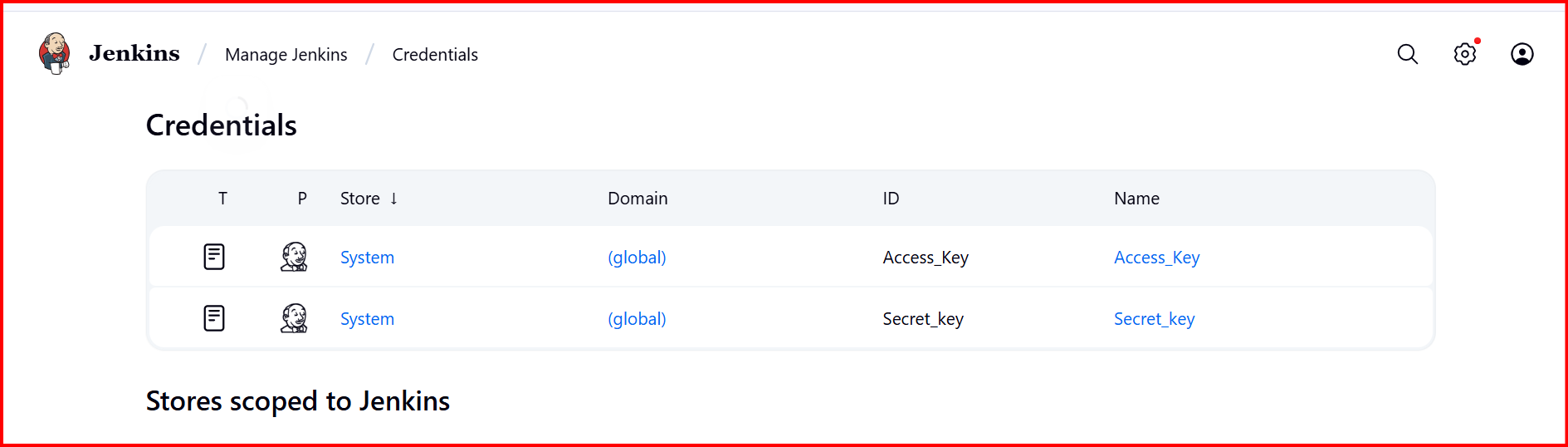
****

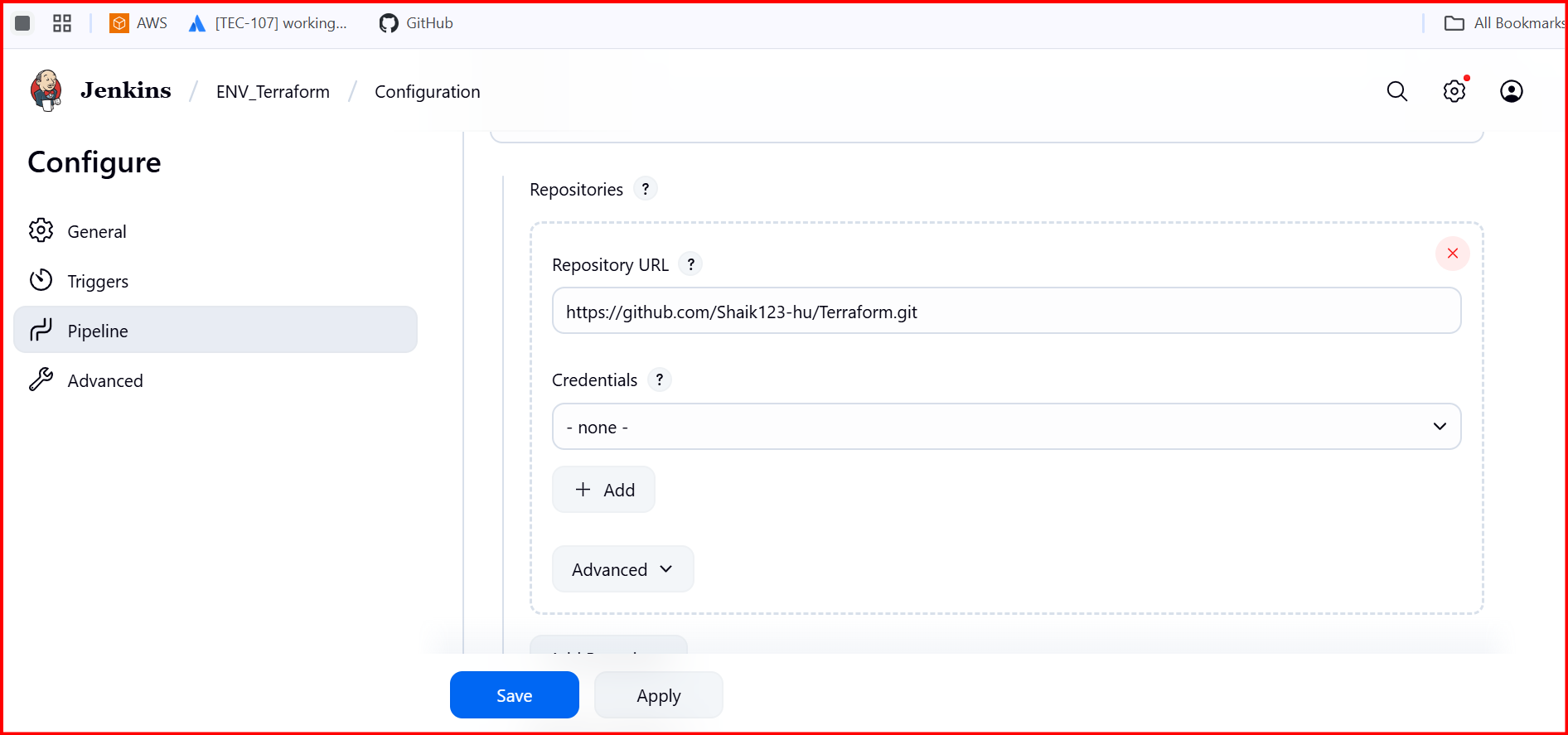
****

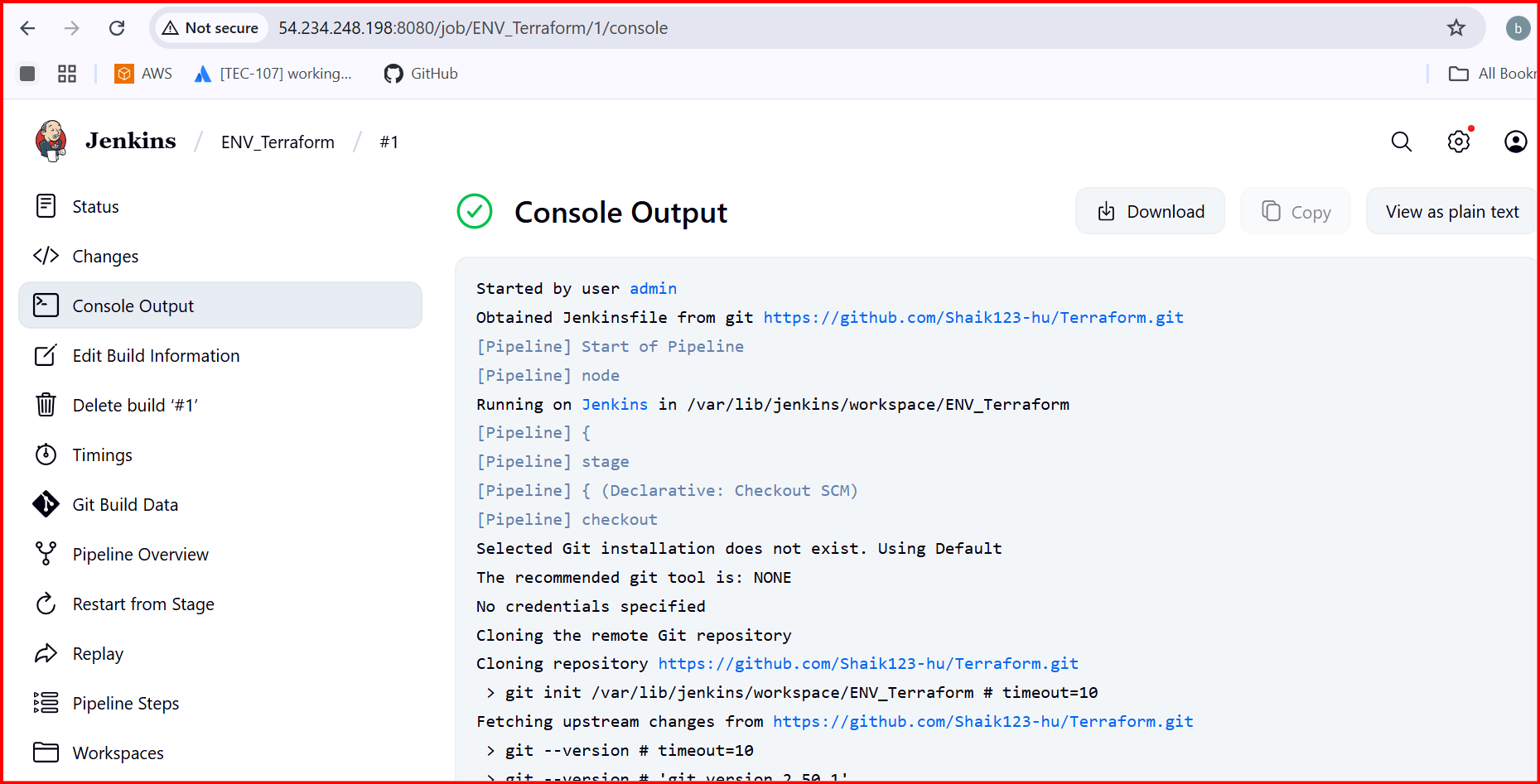
****

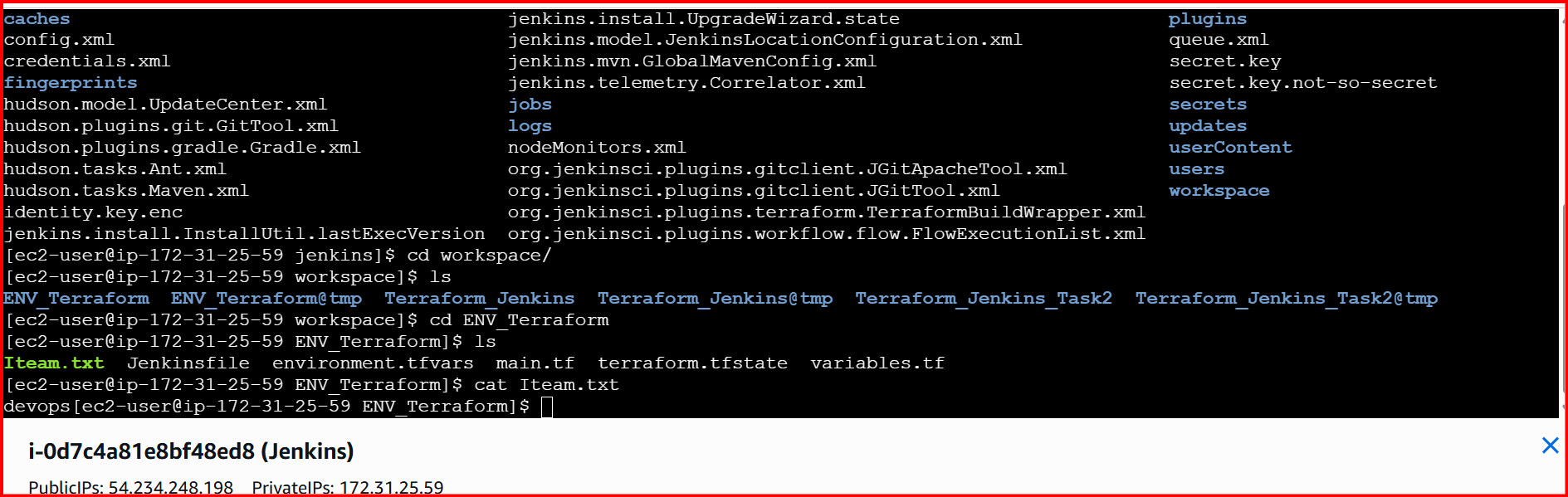
**Jenkins → Credentials → add credentials:**

* **For AWS: you can store as Username with password (username = AWS\_ACCESS\_KEY\_ID, password = AWS\_SECRET\_ACCESS\_KEY)**

****

****

****

****

**5.Use the same code and create a parameterized job in Jenkins with:**

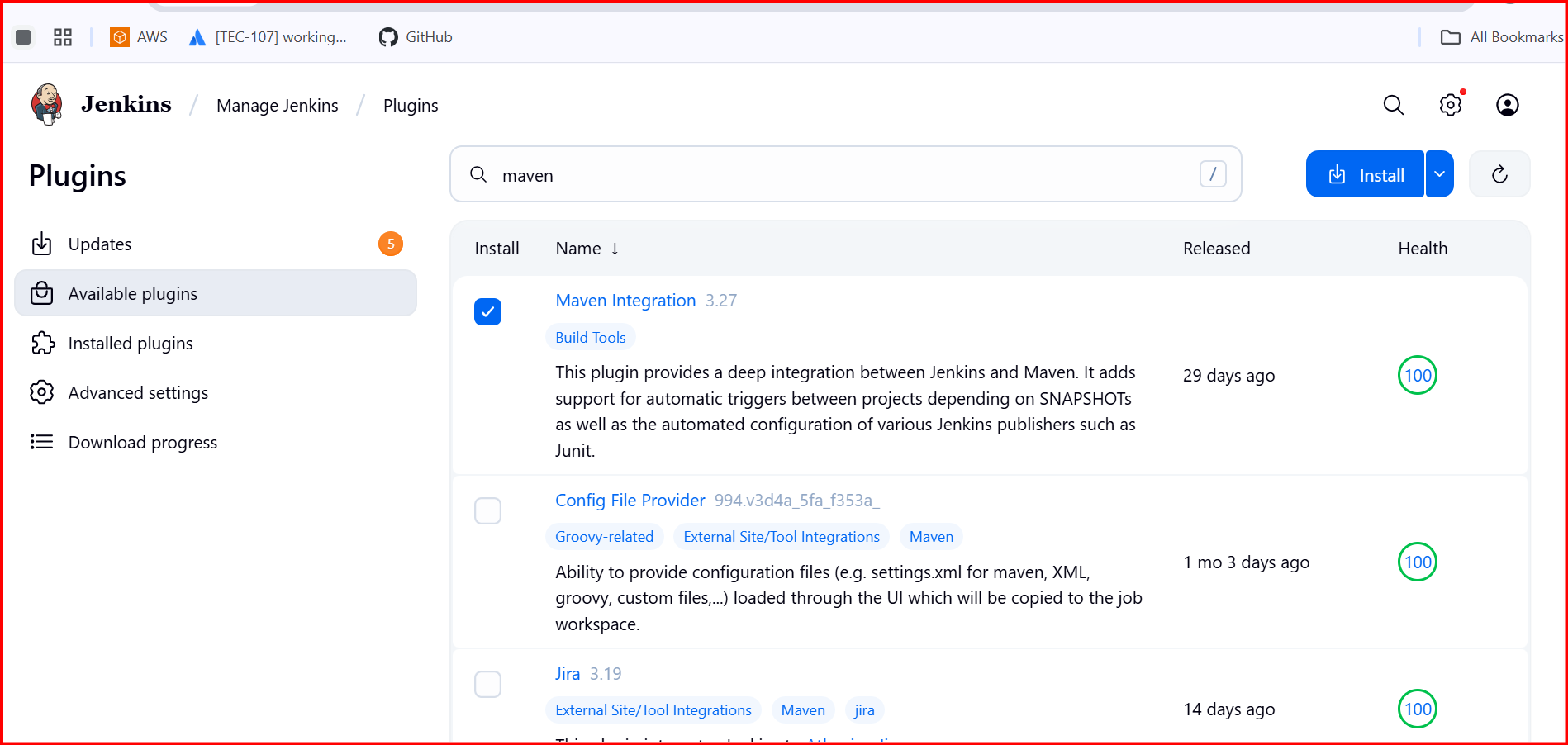
* + **Stage 1: Git clone**
  + **Stage 2: Maven Compilation Code:**[**https://github.com/betawins/java-Working-app.git**](https://github.com/betawins/java-Working-app.git)

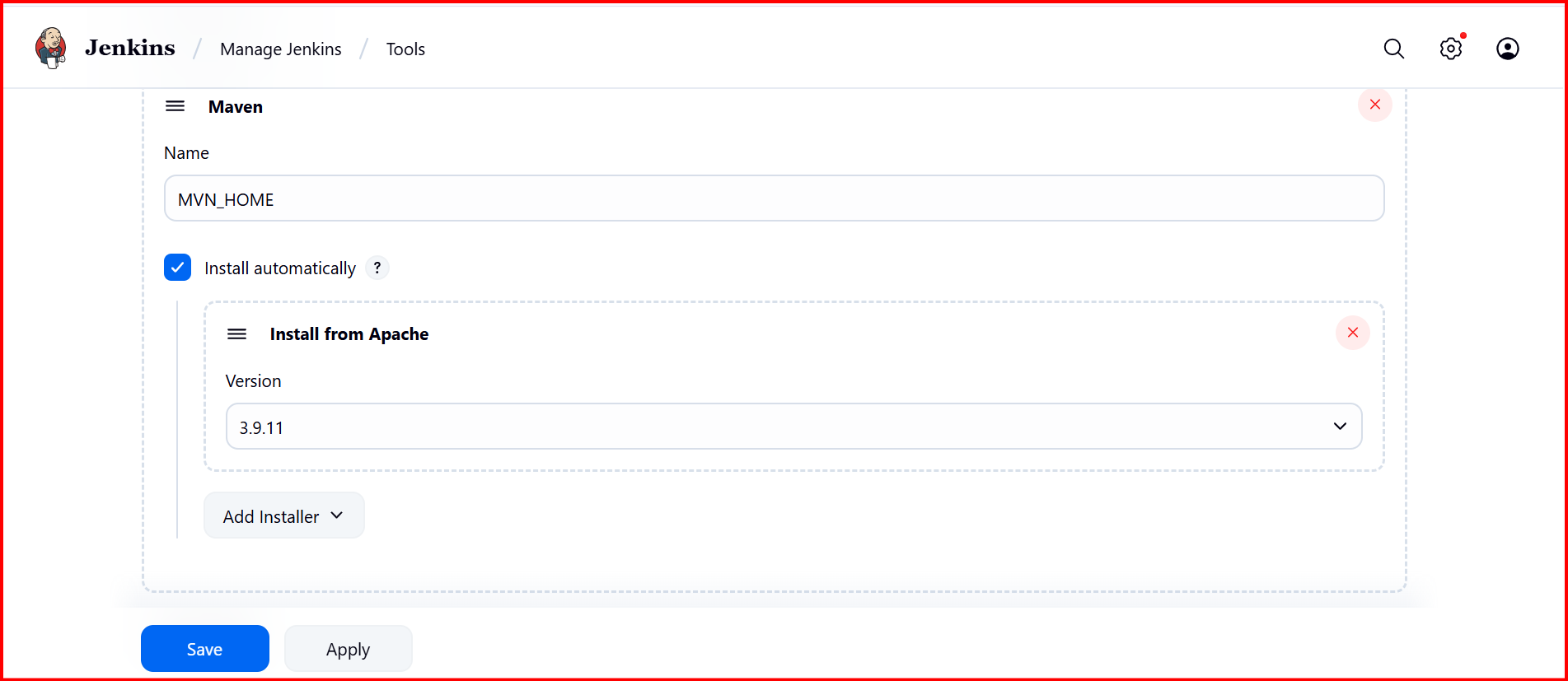
**Install Jenkins**

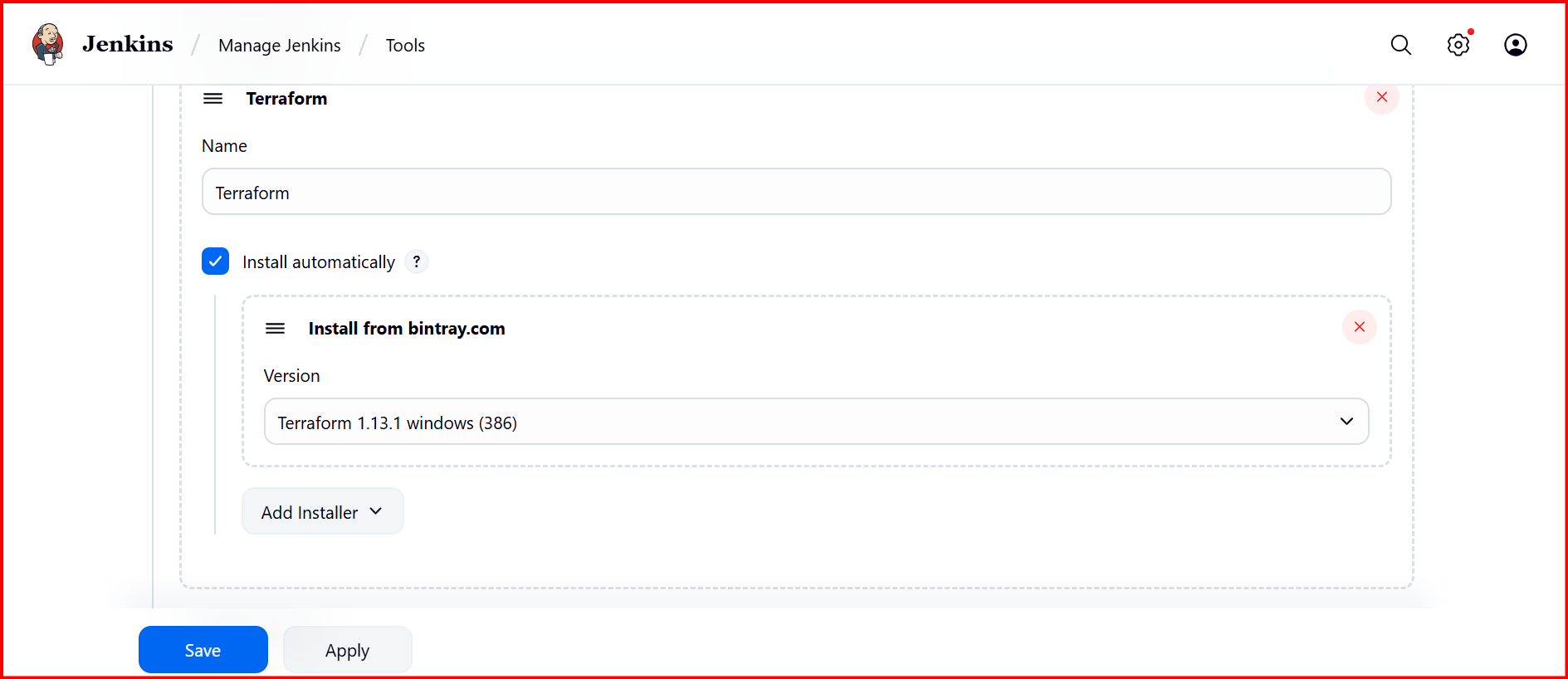
**Install Maven**

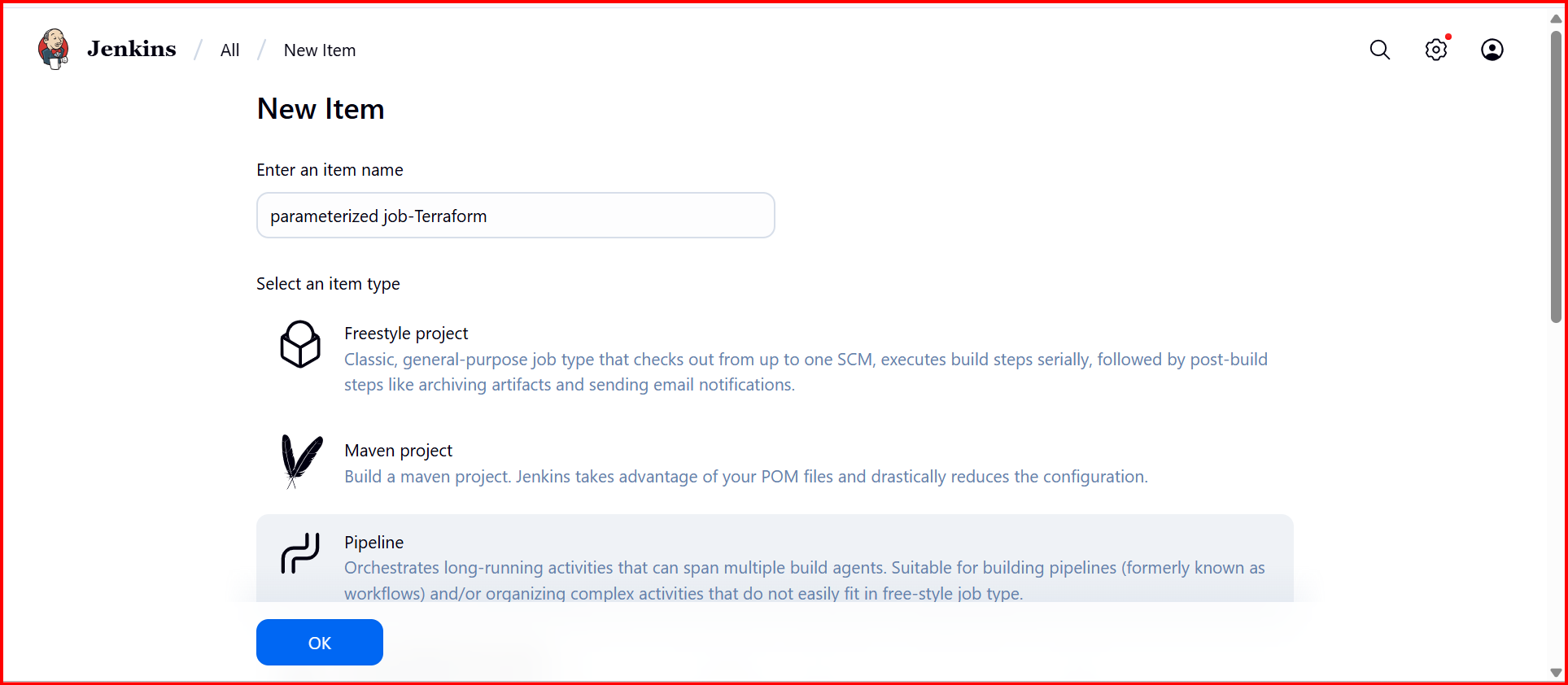
****

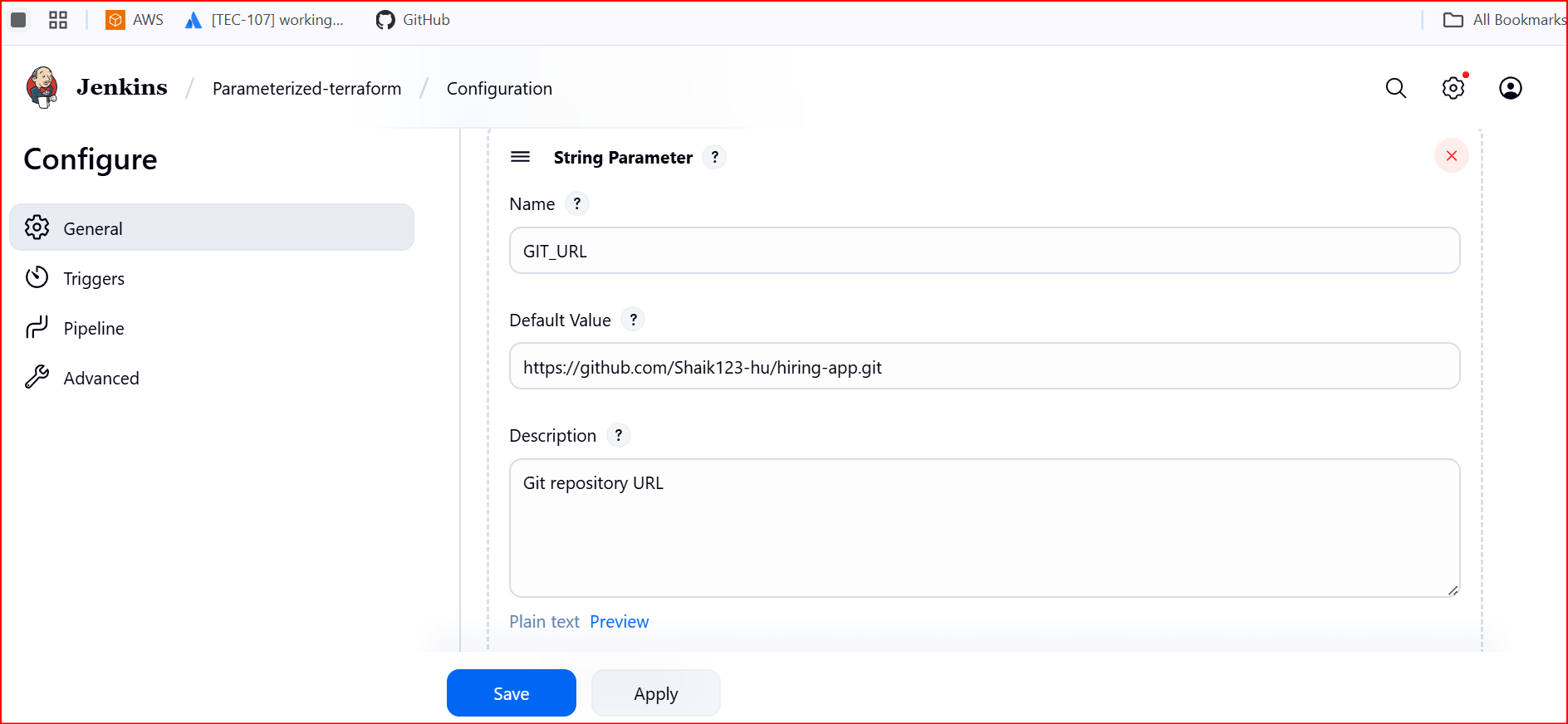
**Install Maven Integration plugin and Terraform Plugin**

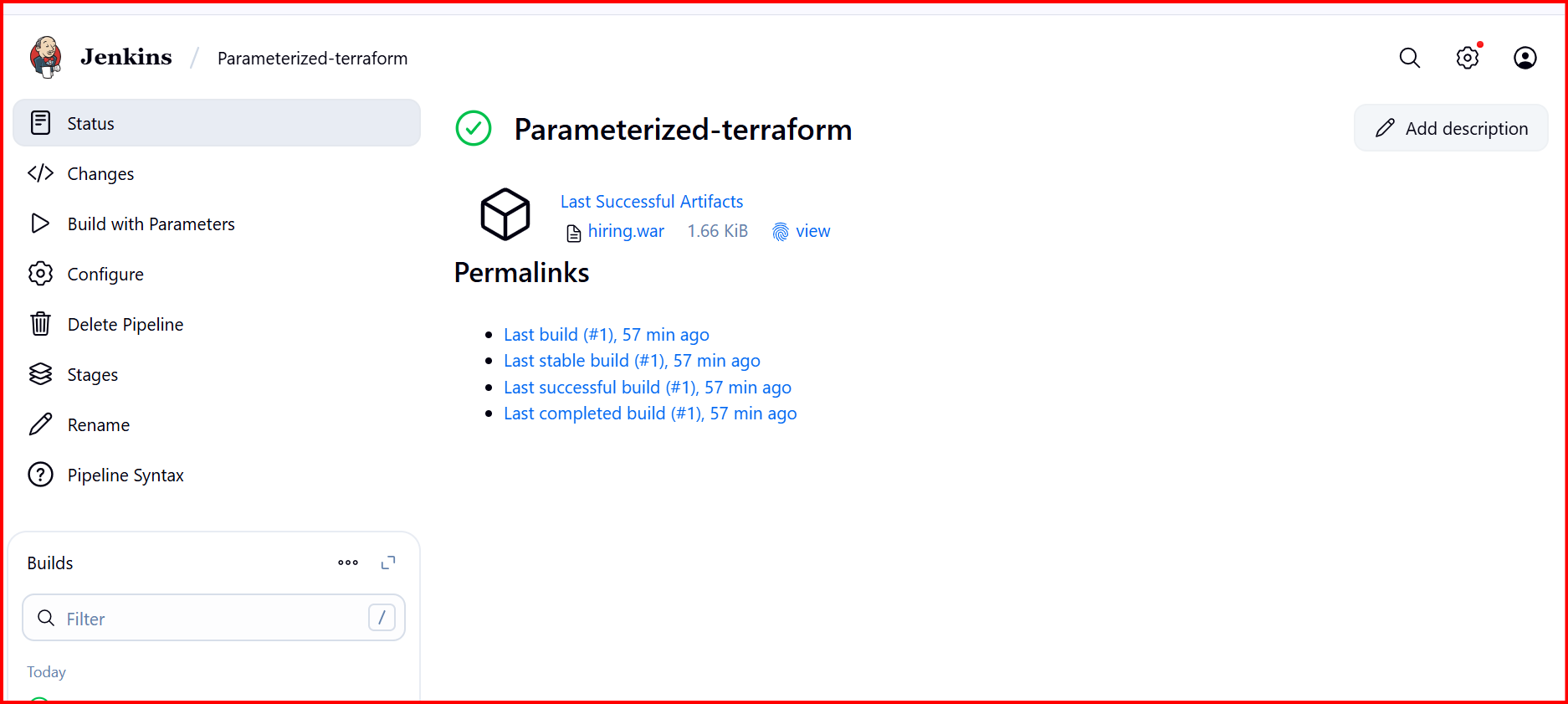
****

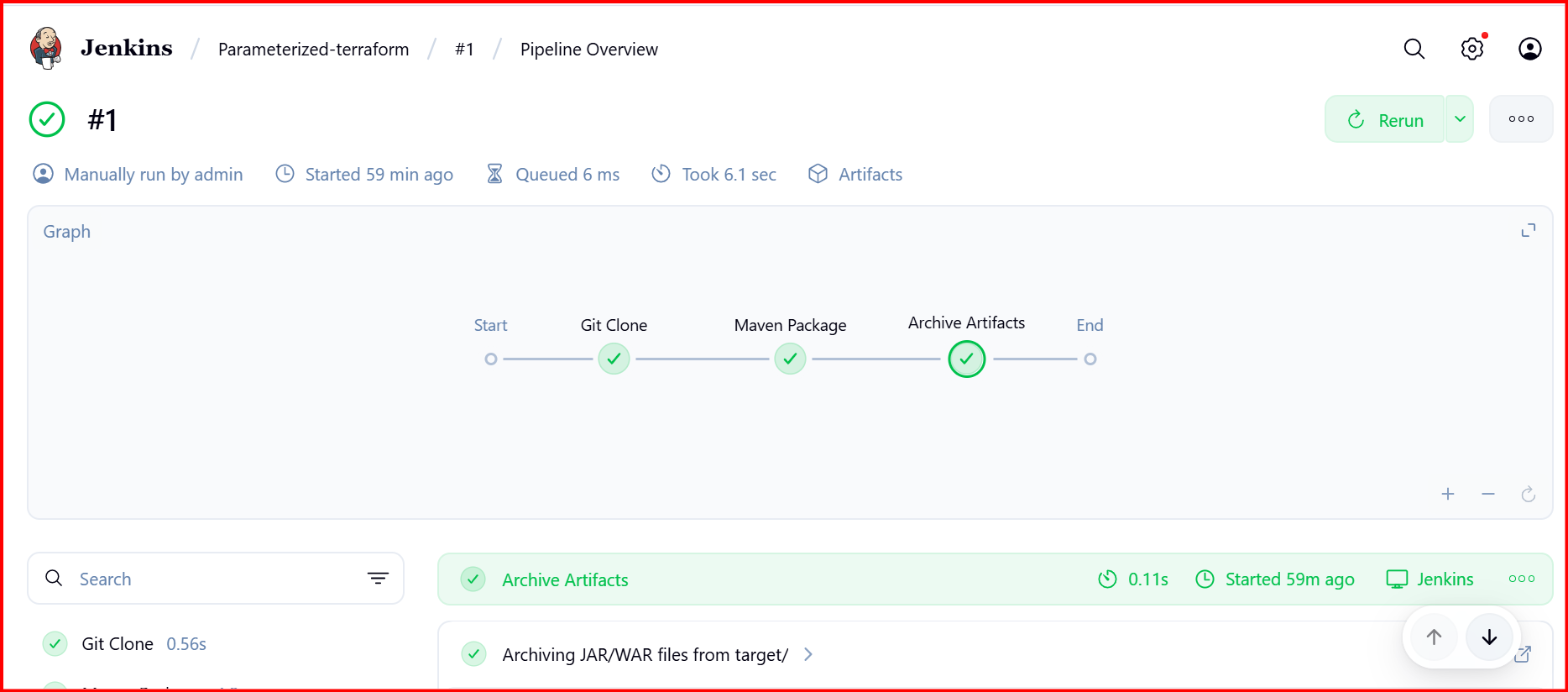
****

****

****

****

****

****

****

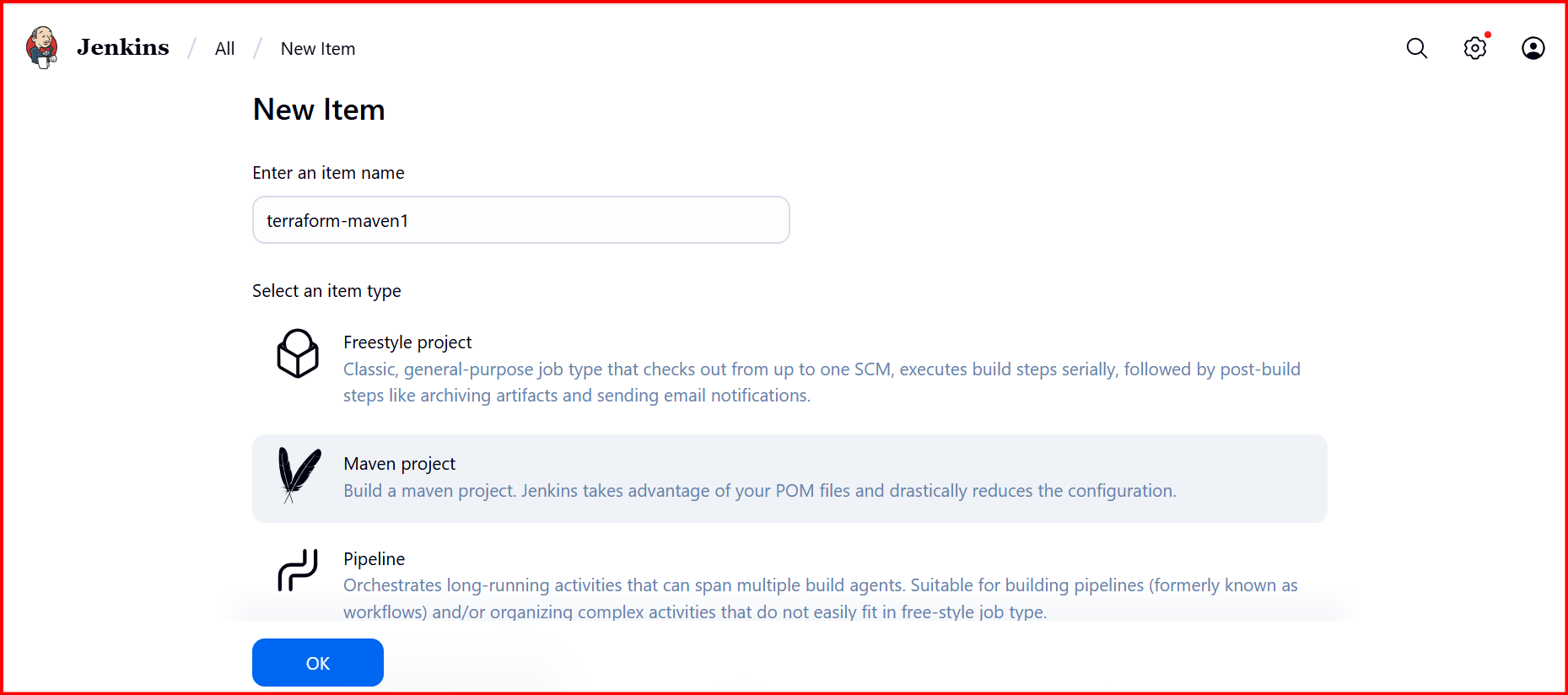
**4.Create one Jenkins job using Maven Project for the code below with two stages:**

* + **Stage 1: Git clone**
  + **Stage 2: Maven Compilation Code:**[**https://github.com/betawins/java-Working-app.git**](https://github.com/betawins/java-Working-app.git)

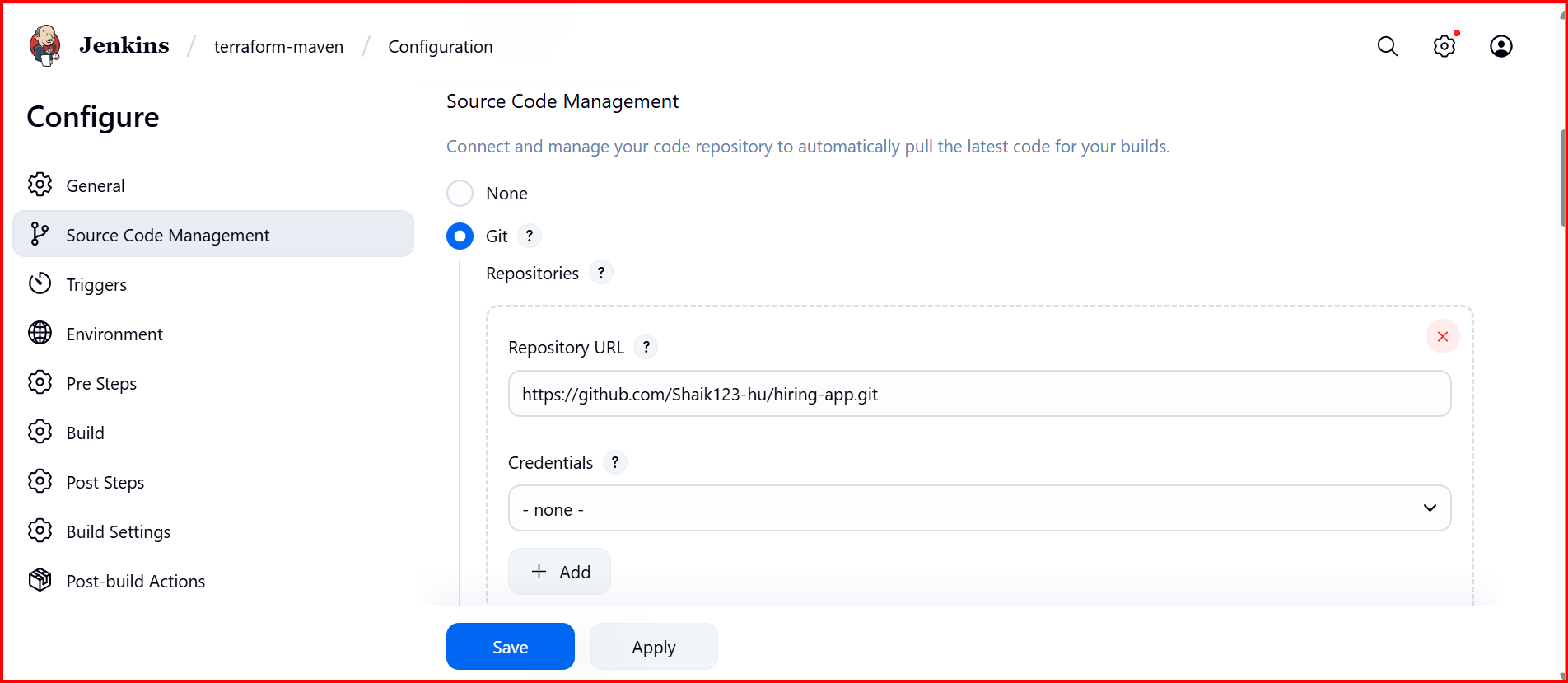
**Install Jenkins**

**Install Pipeline integration**

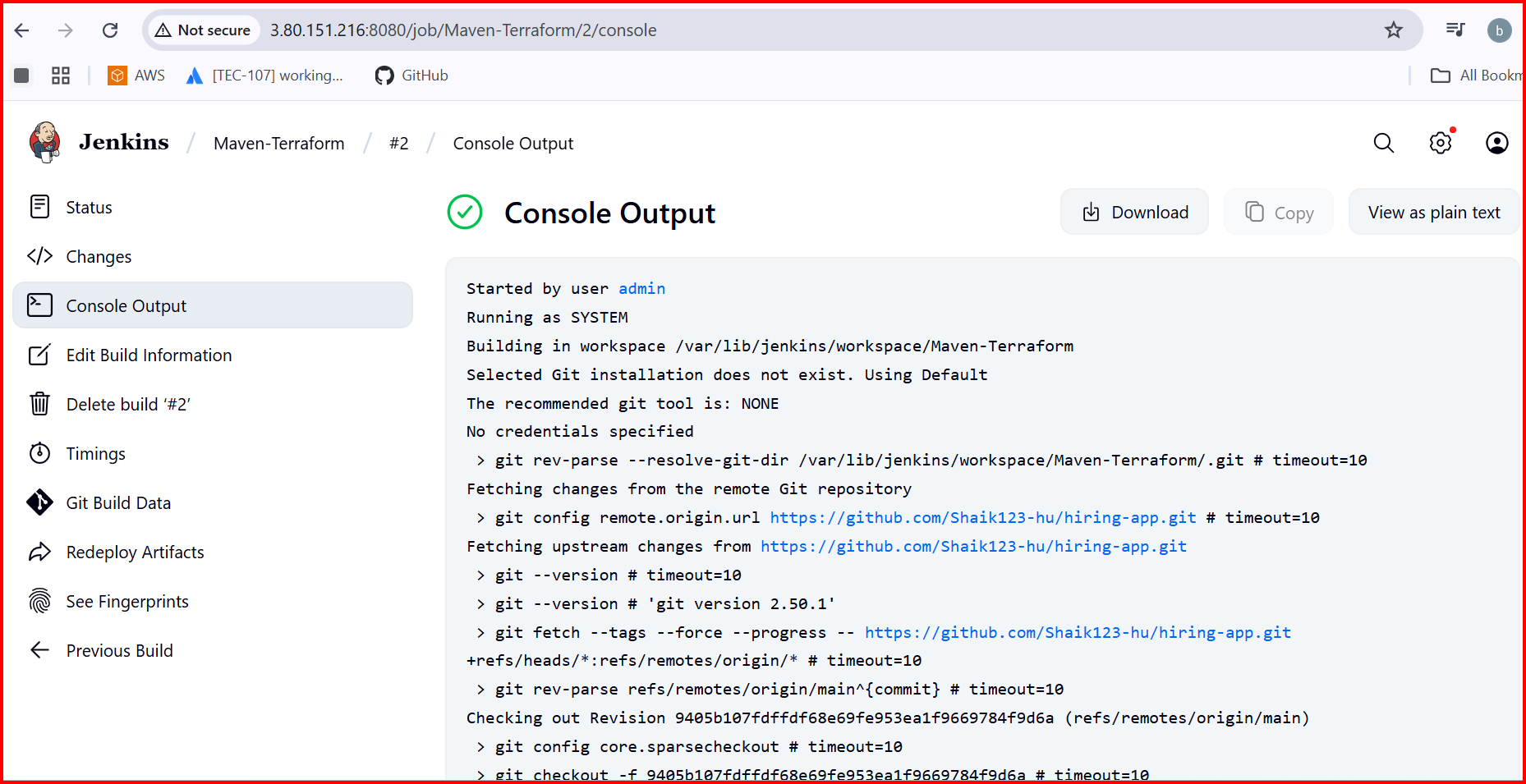
1. **Login to Jenkins.**
2. **Create a new job:**
   * **Click “New Item”.**
   * **Enter a name, e.g., terraform-maven.**
   * **Select Pipeline → Click OK.**

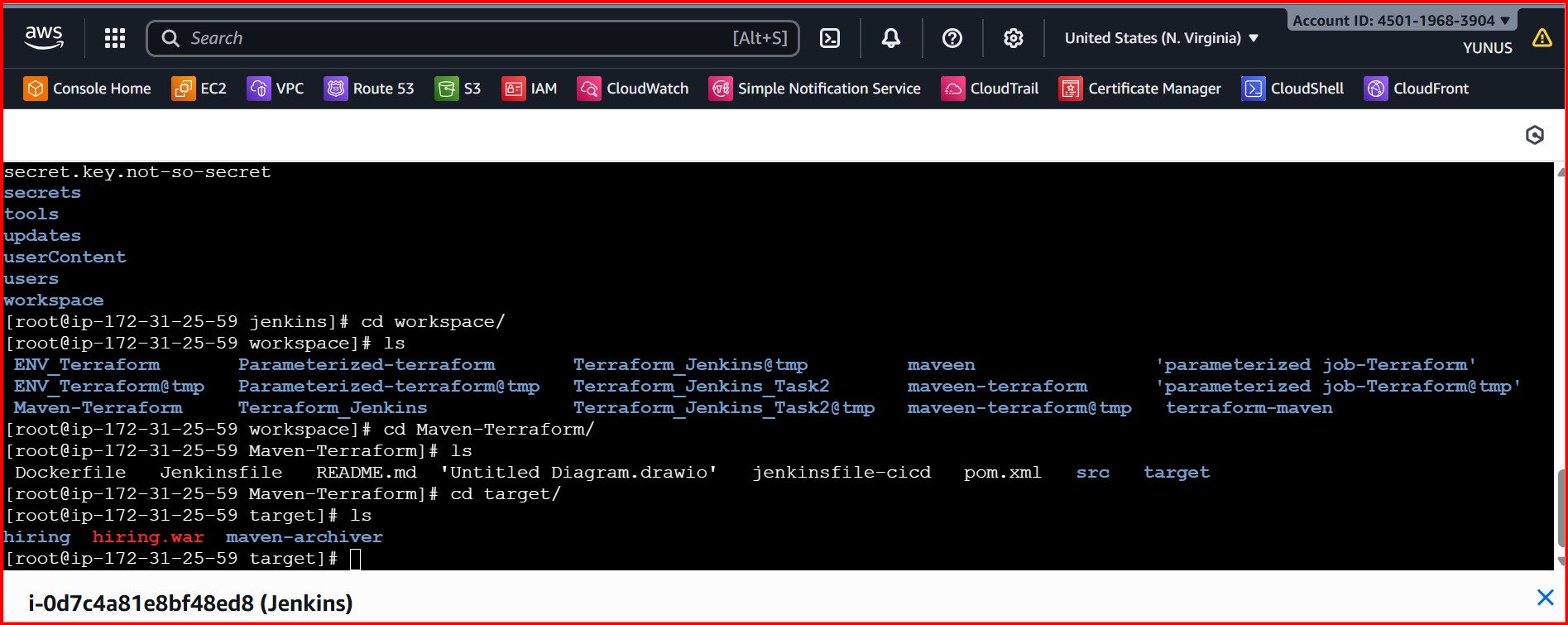
****

1. **Pipeline Configuration:**
   * **Under Pipeline section, select Pipeline script from SCM.**
   * **SCM: Git**
   * **Repository URL: https://github.com/Shaik123-hu/hiring-app.git**
   * **Branch: main**
   * **Script Path: Jenkinsfile-maven (your pipeline file).**

****

**Save And build it**

****

****

**6.** **What are the global variables in Jenkins?**

**1. Jenkins Environment Variables**

**These are provided automatically by Jenkins during a job/pipeline run.**

**Some important ones are:**

* **BUILD\_NUMBER → The current build number.**
* **BUILD\_ID → The unique build ID (often a timestamp).**
* **BUILD\_URL → URL of the current build.**
* **JOB\_NAME → Name of the Jenkins job.**
* **JOB\_BASE\_NAME → The short name of the job without path.**
* **WORKSPACE → Path to the workspace directory.**
* **JENKINS\_URL → Root URL of the Jenkins instance.**
* **NODE\_NAME → Name of the agent/node where the job is running.**
* **EXECUTOR\_NUMBER → Executor number of the build on the node.**
* **GIT\_COMMIT → Commit ID of the current build (if using Git plugin).**
* **GIT\_BRANCH → Branch name being built.**

**2. Pipeline Global Variables**

**In Declarative/Scripted Pipelines, Jenkins exposes some built-in global variables:**

* **env → Access to environment variables (env.BUILD\_NUMBER, env.WORKSPACE, etc.).**
* **params → Access to parameters defined for the job (params.BRANCH).**
* **currentBuild → Provides details about the current build (status, display name, etc.).**
* **scm → Refers to the SCM configuration of the job.**
* **pipeline → For pipeline-level configuration in declarative syntax.**