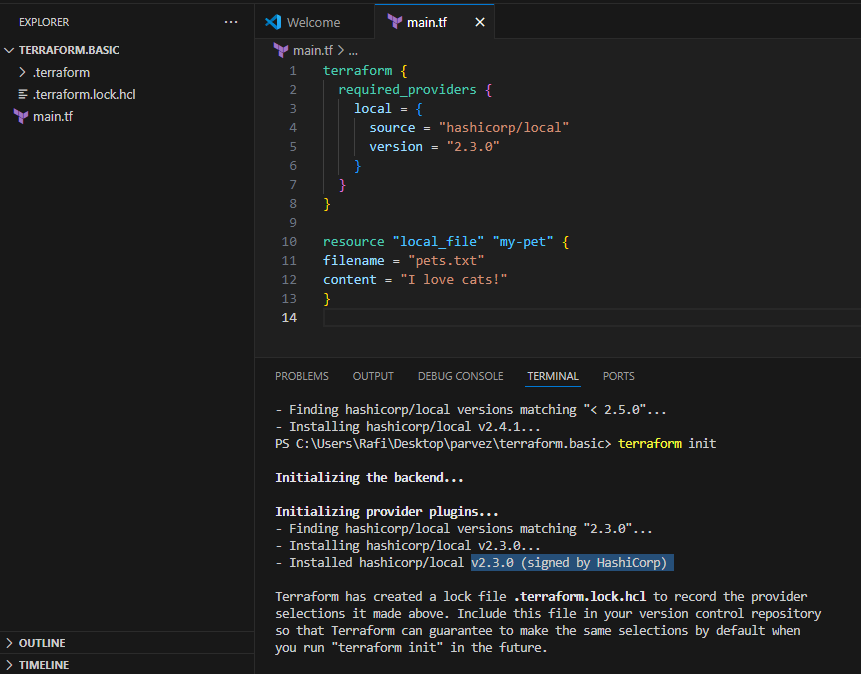
**Terraform Day-04**

***Mohammed Rafiyoodin – 29-8-05-2024***

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

Downloaded the latest version of terraform



Example:

=======

terraform {

required\_providers {

local = {

source = "hashicorp/local"

version = "2.3.0"

}

}

}

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

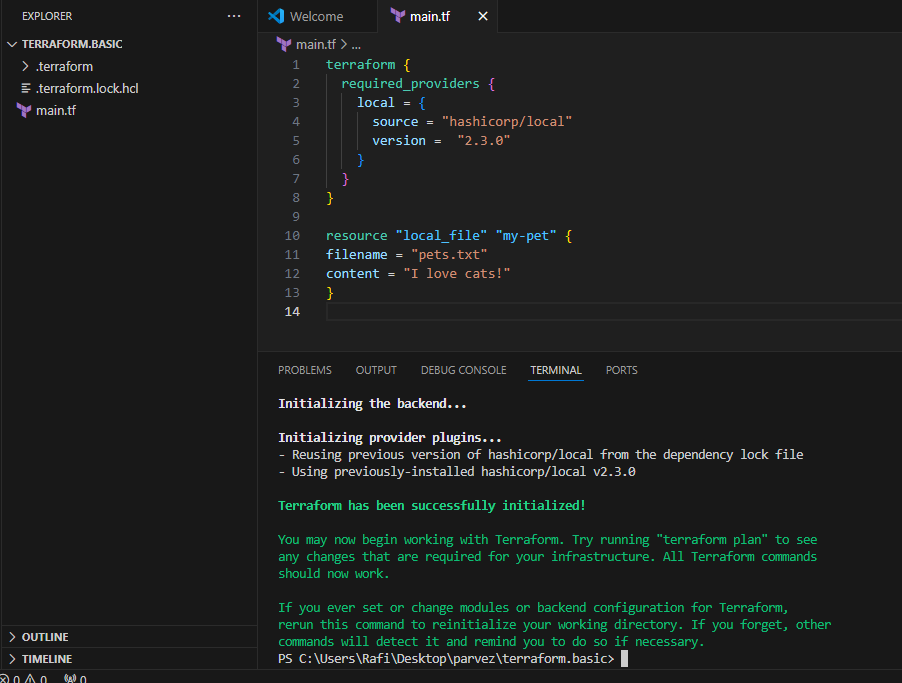
filename = "pets.txt"

content = "I love cats!"

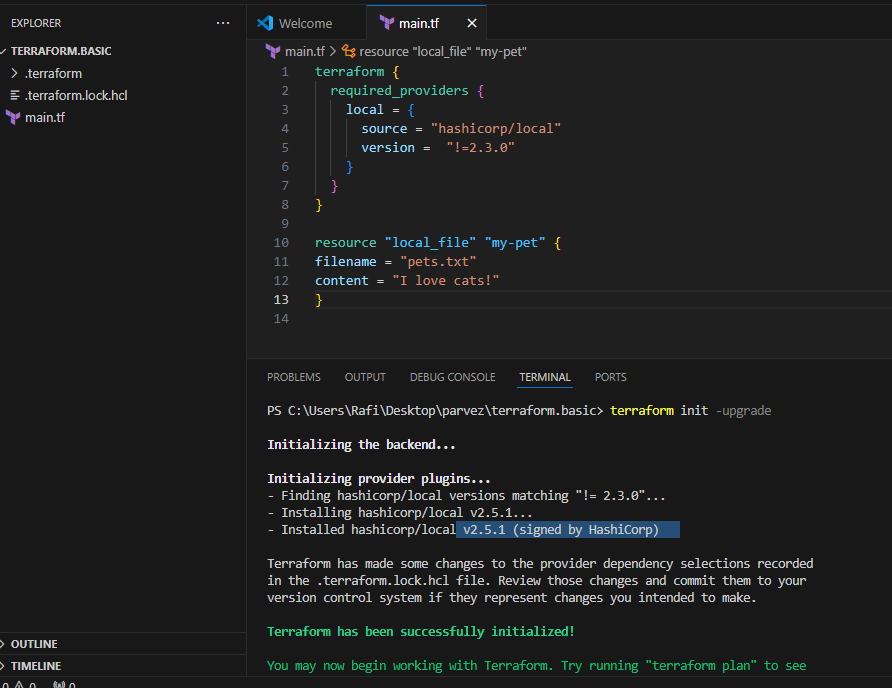
}

=========

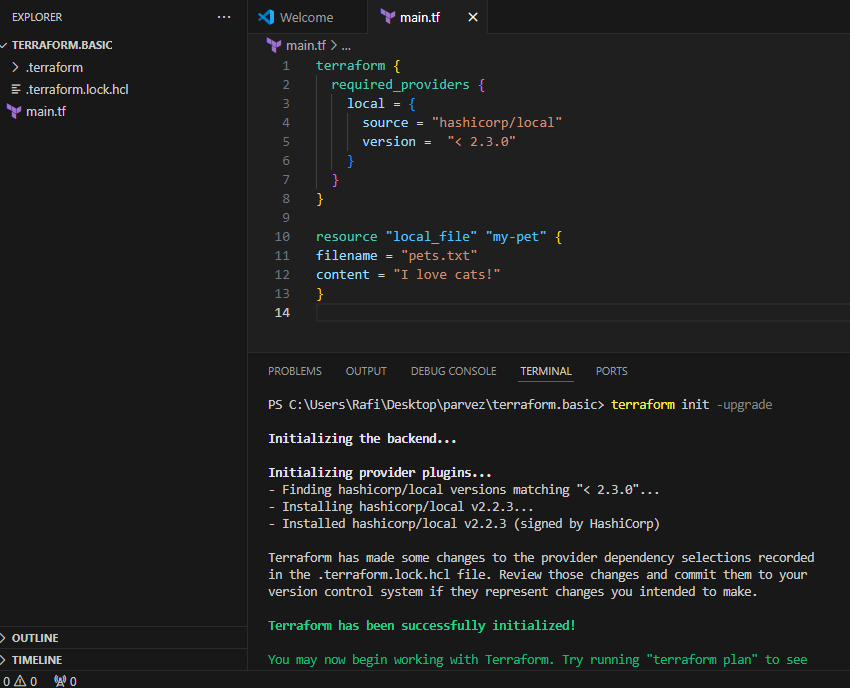
version = "2.3.0" --> download the exact version



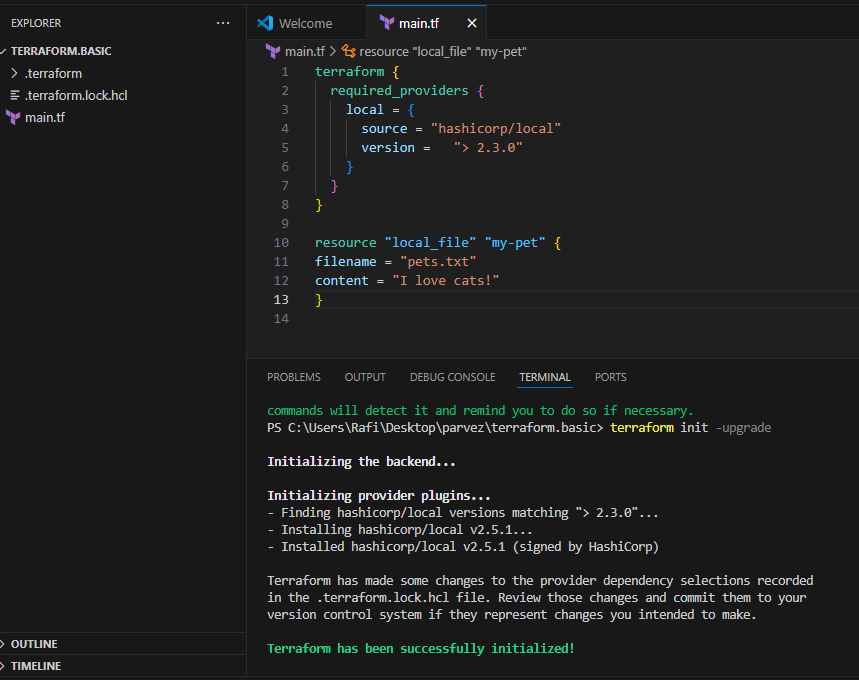
version = "!=2.3.0" --> will not use the mentioned version



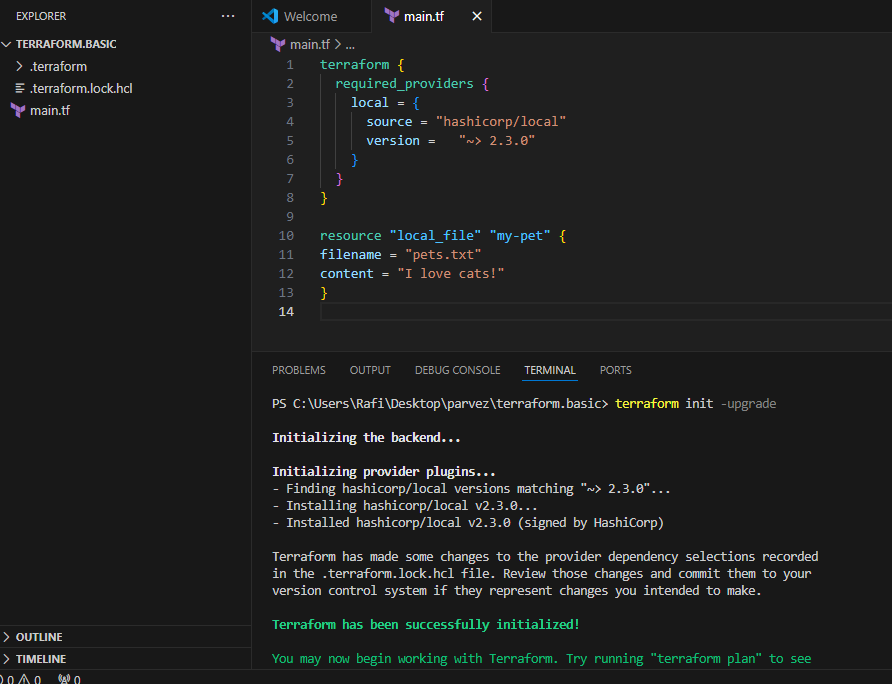
version = "< 2.3.0" --> lesses than the mention version



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



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



**I want to create No.of files for example3  
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.**

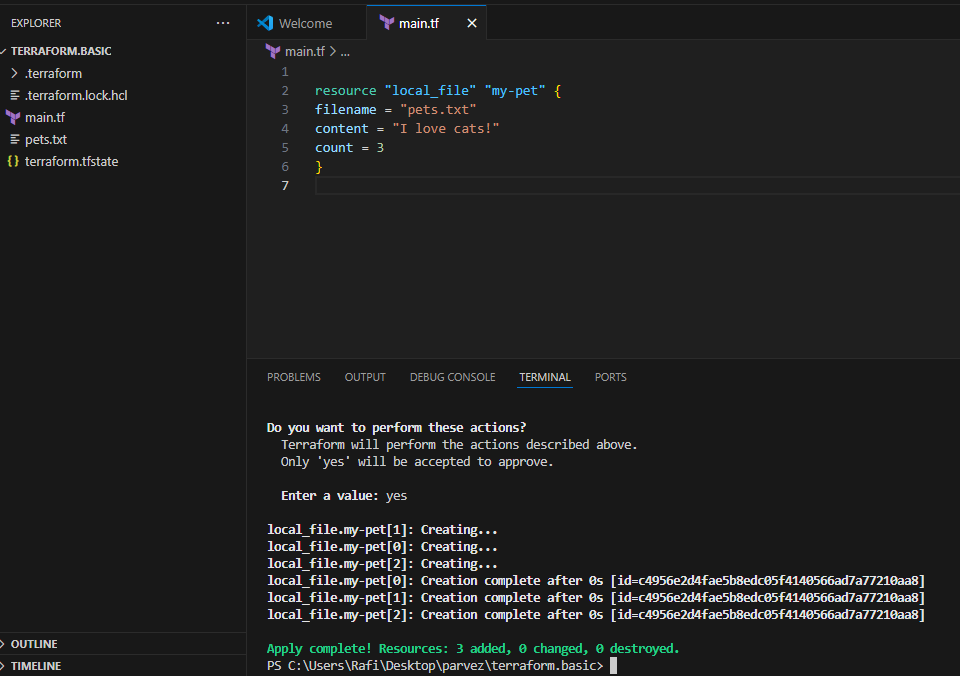
**Examples for meta arguments:**

**1) Depends\_on**

**2) lifecycle rules**

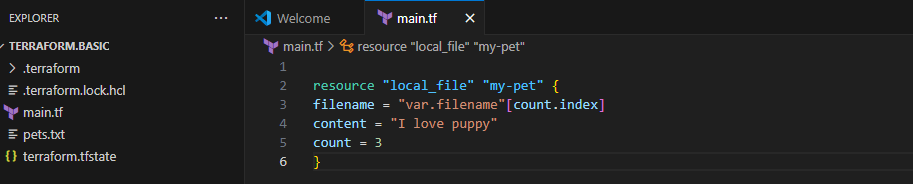
**3) For\_each**

**4) Count-----🡪by using count meta arqu we can create no of files**

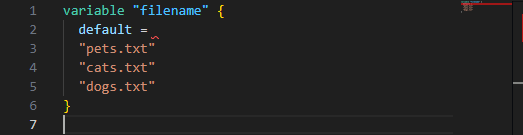


**Terraform is trying to create files but its not showing the files because of this Is not proper way to do the creating of no of files .its getting over ride .**

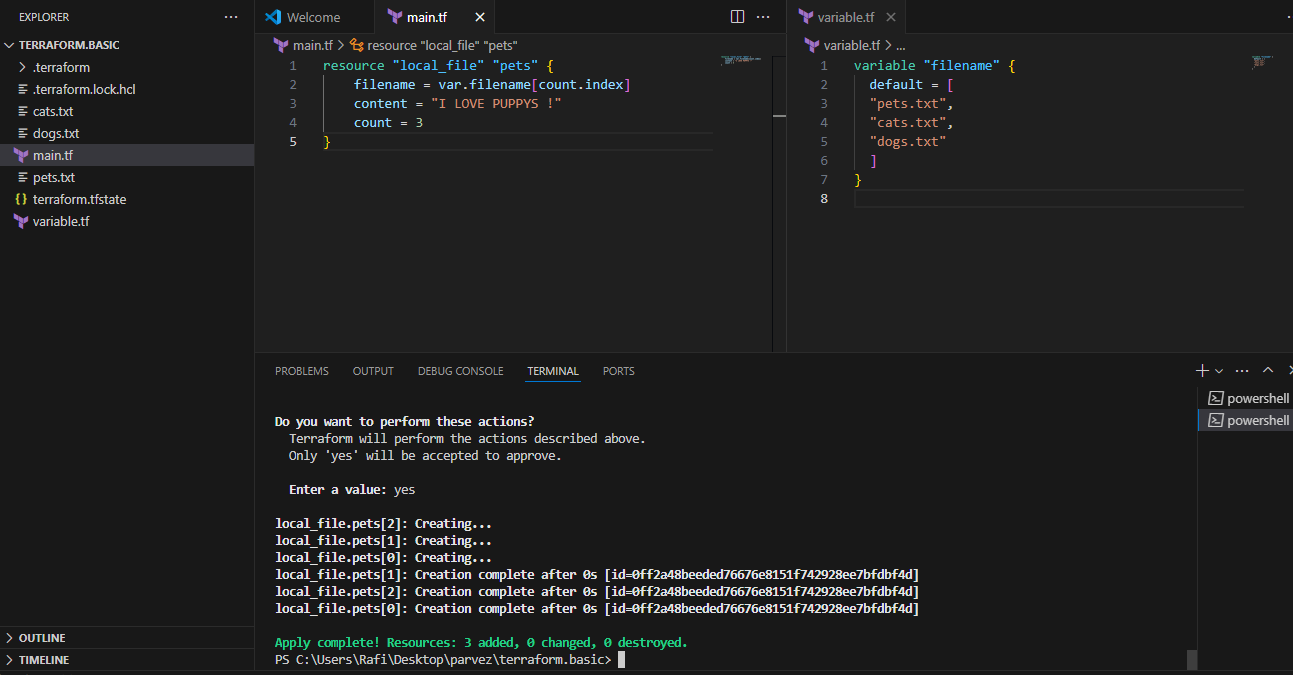
**Main.tf**



**variables.tf ------------🡪list type variable**

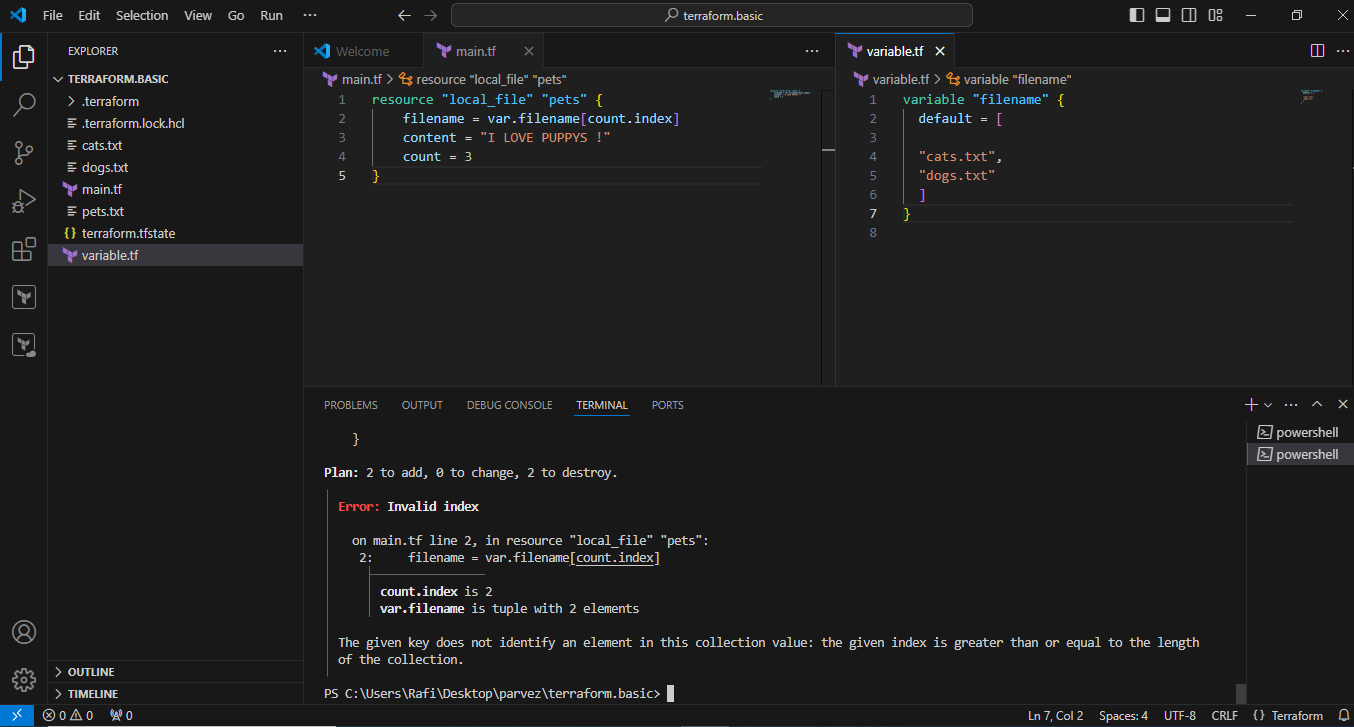


**Terraform apply it wil create 3 files as per variable .tf file and the based upon the syntax of main.tf**

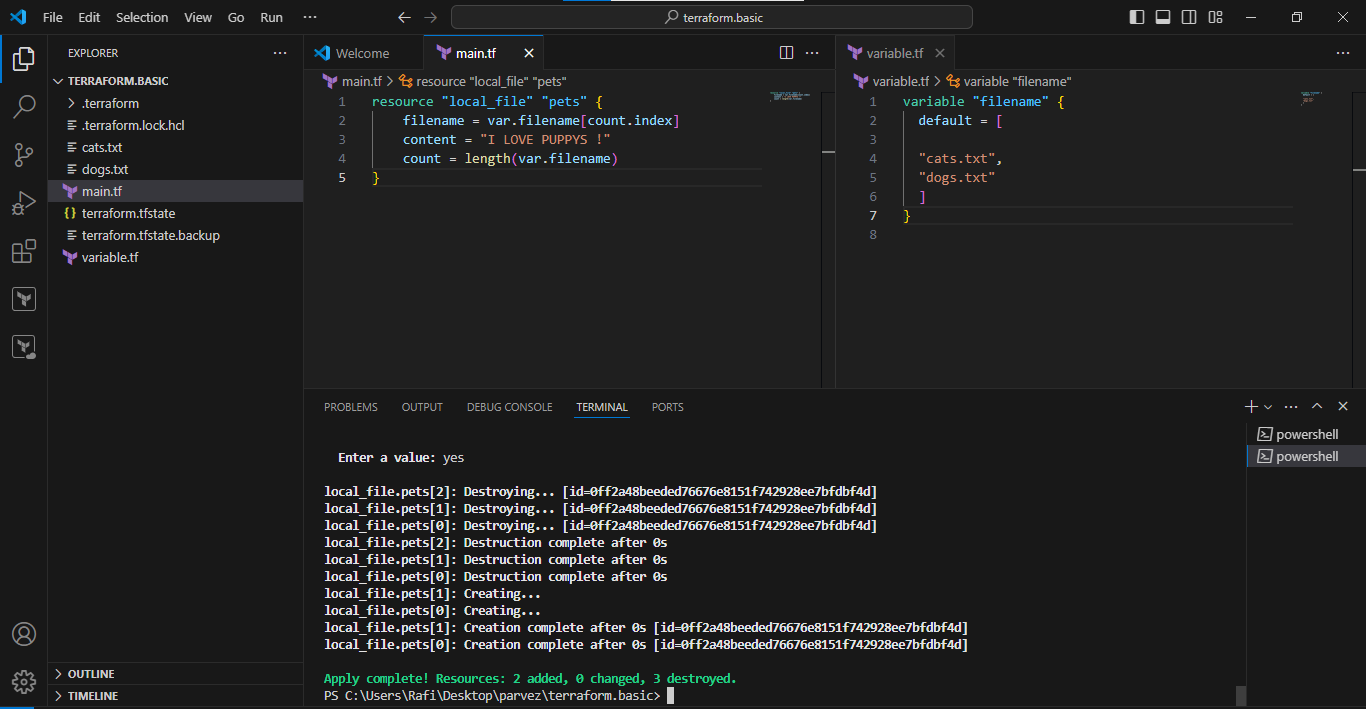


**3 files created.**

**By any chance 1 file is deleted from variable.tf file it will show error after apply because the count is mismatch to over come this we need to pass generic code for count instead of hard code.**



**Generic code passed below.it help us to check the length of the varible.tf file ---  
main.tf**



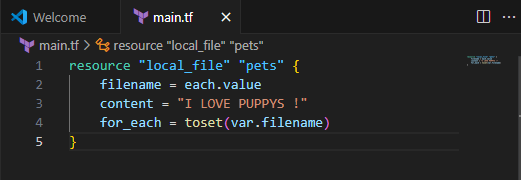
**If you add one no of file to the var.tf it will create as per the length function**

**For\_each**

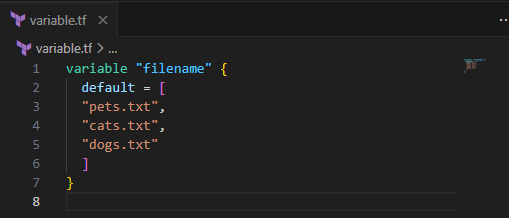
**Main.tf**

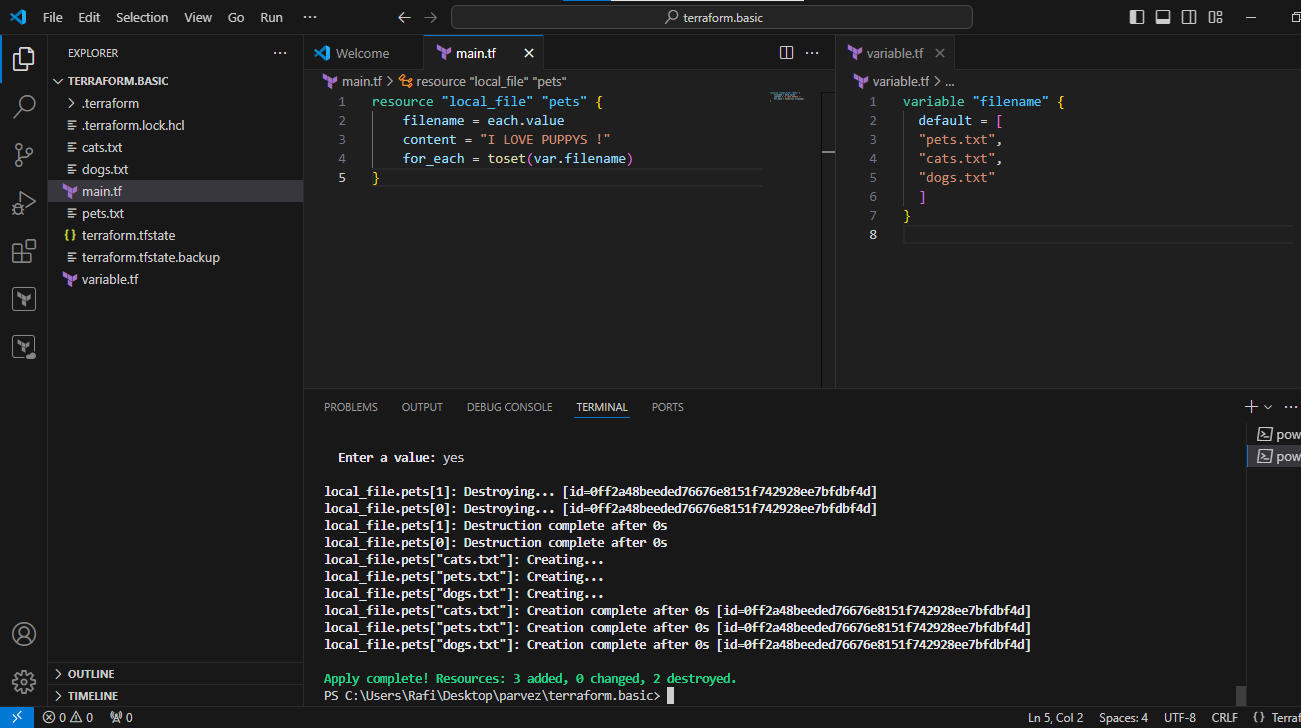
* **var.filename is expected to be a list or tuple variable containing filenames.**
* **toset(var.filename) converts the list or tuple into a set, ensuring each filename is unique.**
* **The for\_each meta-argument allows you to create multiple instances of a resource, each identified by a unique key. In this case, each filename in the set becomes a unique key for the local\_file resource.**
* **each.value represents the current element being processed during iteration. It will be replaced with each filename from the set, allowing Terraform to create a separate local\_file resource for each filename.**

**Main.tf**



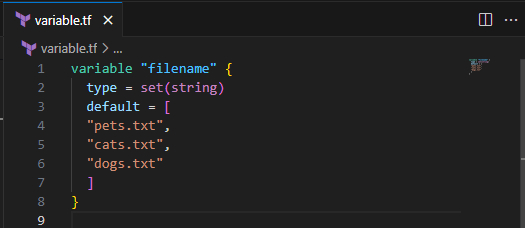
**Variable.tf**



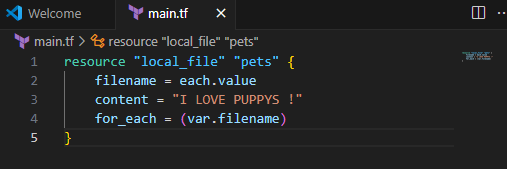


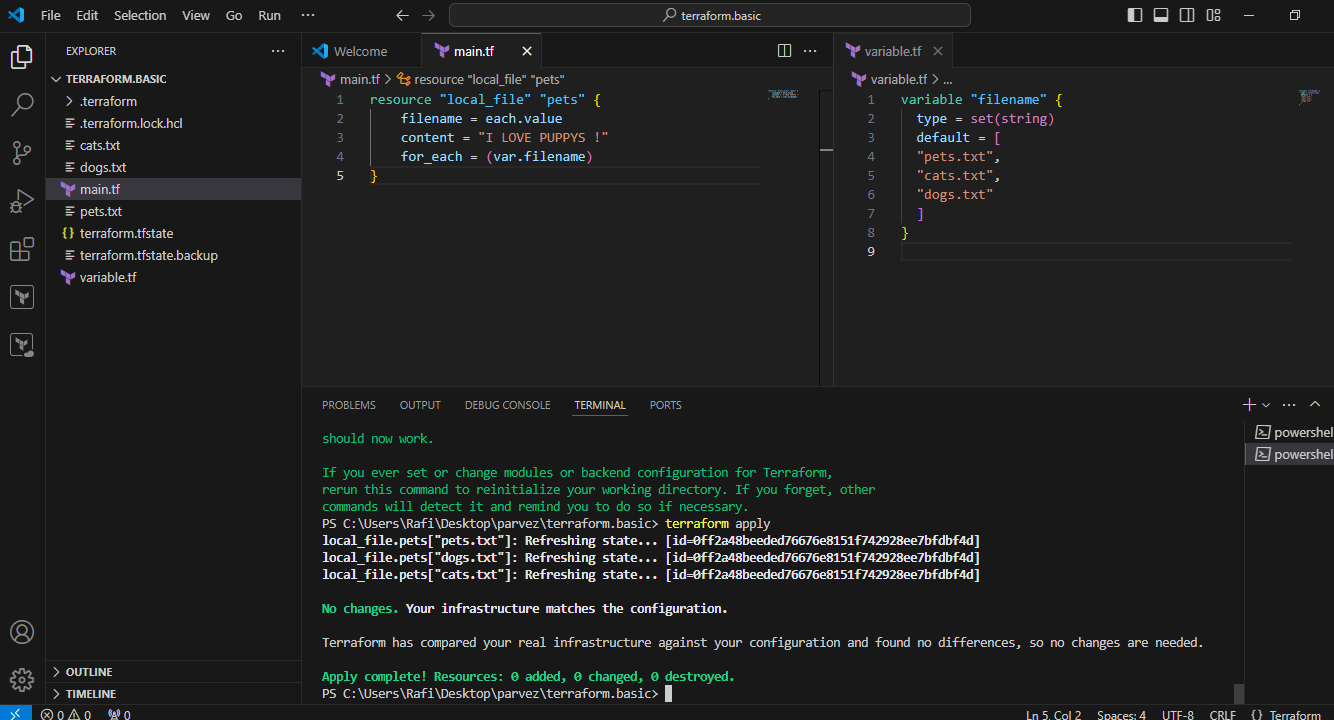
**If you don’t want to use inbuilt function toset(var.filename)  
then define the type in variable.tf like below**

**Variable,tf**

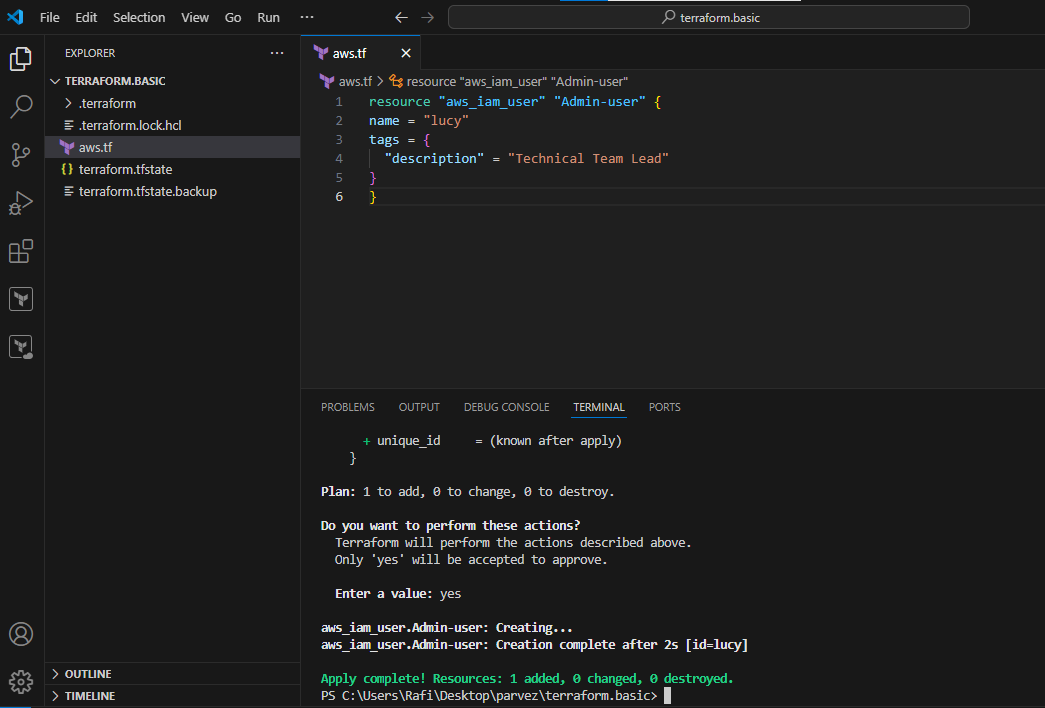


**Main.tf**

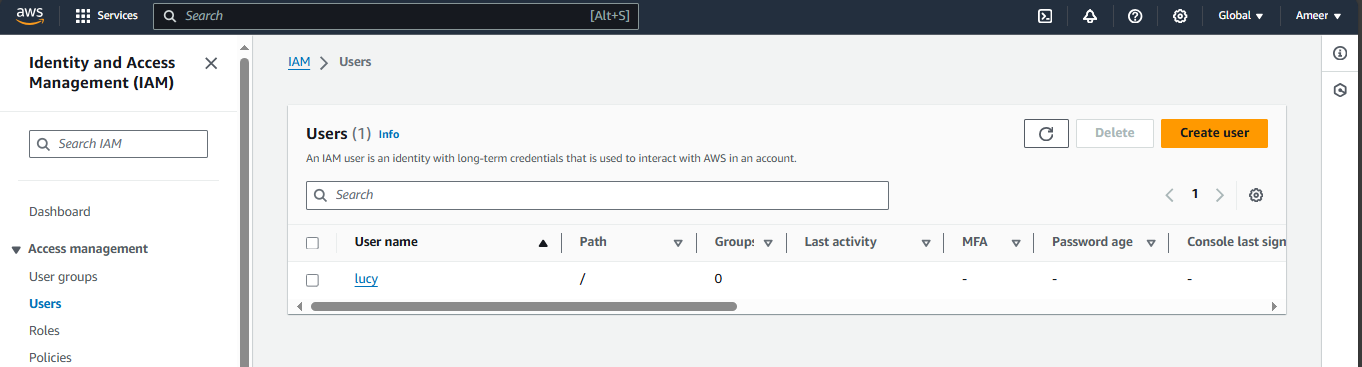
****



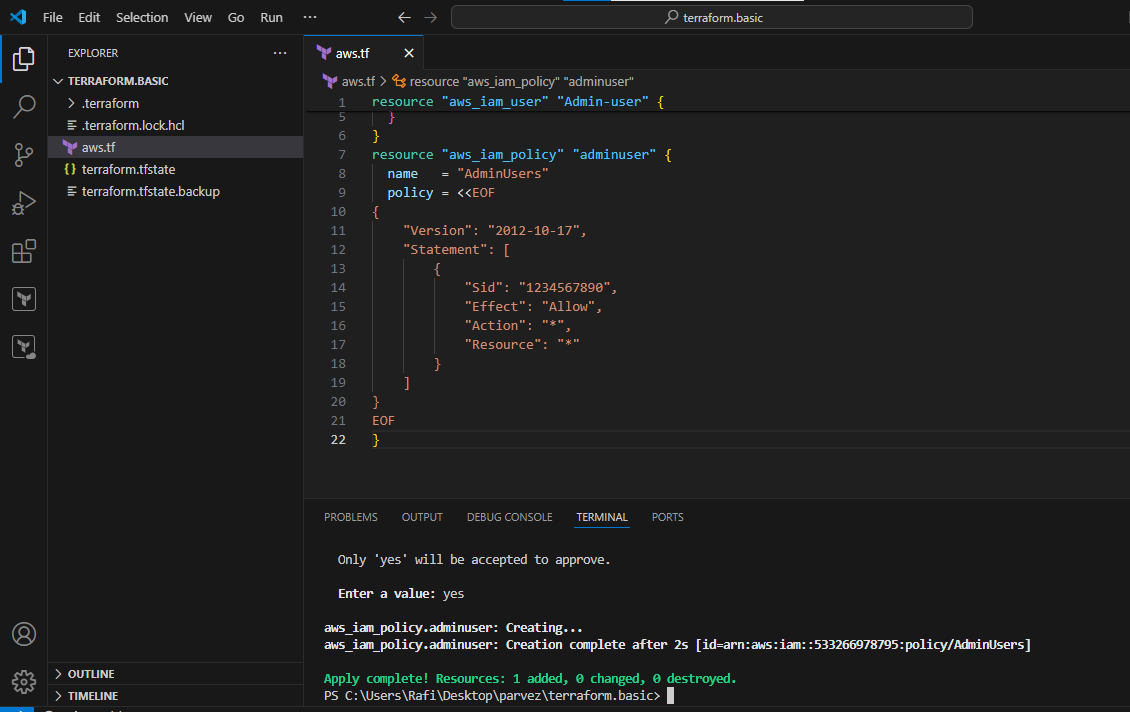
**Creating IAM user in Terraform**

****

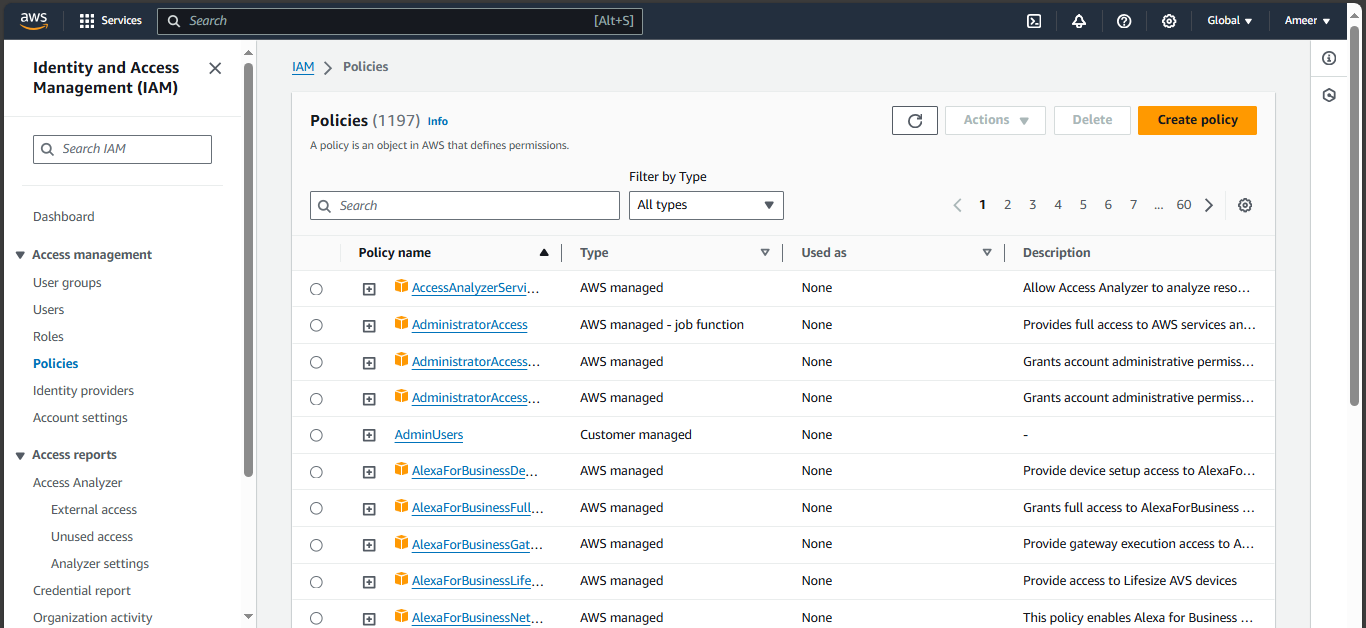
**IAM user created**

****

**Iam policy created Terraform**

****

**Created iam user policy in Aws**

****



**5) Explain 10 Maven commands.**

**mvn --version**  
 Prints out the version of Maven you are running.  
**mvn clean**  Clears the target directory into which Maven normally builds your project.  
**mvn package**  
 Builds the project and packages the resulting JAR file into the target directory.  
**mvn package -Dmaven.test.skip=true**  
 Builds the project and packages the resulting JAR file into the target directory - without running the unit tests during the build.  
**mvn clean package**  
 Clears the target directory and Builds the project and packages the resulting JAR file into the target directory.  
  **mvn clean package -Dmaven.test.skip=true**  
 Clears the target directory and builds the project and packages the resulting JAR file into the target directory - without running the unit tests during the build.  
**mvn verify**  
 Runs all integration tests found in the project.  
**mvn clean verify**  
 Cleans the target directory, and runs all integration tests found in the project.  
      **mvn install**  
 Builds the project described by your Maven POM file and installs the resulting artifact (JAR) into your local Maven repository  
**mvn install -Dmaven.test.skip=true** Builds the project described by your Maven POM file without running unit tests, and installs the resulting artifact (JAR) into your local Maven repository  
**mvn clean install**  
 Clears the target directory and builds the project described by your Maven POM file and installs the resulting artifact (JAR) into your local Maven repository

**s**