

TF file and is used for writing configuration files in Terraform using HCL.

```
1 1 ✓ resource "local_file" "games" {  
2   file      = "/root/favorite-games"  
3   content   = "FIFA 21"  
4 }  
5
```

The Terraform configuration block you provided declares a resource of type `local_file` with the name `games`. This resource creates a local file on the machine where Terraform is executed, with the content "FIFA 21".

Here's a breakdown of the different parts of the configuration block:

**resource:** This is a Terraform keyword that indicates that you're defining a new resource.

**local\_file:** This is the resource type, which creates a local file on the machine where Terraform is executed.

**games:** This is the name of the resource block, which you can use to reference this resource elsewhere in your configuration.

**file:** This parameter specifies the file path where the local file should be created. In this case, the file will be created at `/root/favorite-games`.

**content:** This parameter specifies the content that should be written to the local file. In this case, the content will be "FIFA 21".

Note that creating files in the `/root` directory requires elevated privileges, so you may need to run Terraform with elevated privileges (e.g. using `sudo`) to create the file. Also note that this configuration creates the file with the specified content every time you run Terraform, even if the file already exists. If you want to create the file only if it doesn't exist or update it with new content only when necessary, you can use a provisioner or a more advanced resource like `file`.

Provider for above resource is `local`. `file` and `content` are arguments in resource block.

```
iac-server $ pwd  
/root/terraform-projects/HCL  
iac-server $ terraform plan
```

**Error: Could not load plugin**

Plugin reinitialization required. Please run "terraform init".

Plugins are external binaries that Terraform uses to access and manipulate resources. The configuration provided requires plugins which can't be located, don't satisfy the version constraints, or are otherwise incompatible.

Terraform automatically discovers provider requirements from your configuration, including providers used in child modules. To see the requirements and constraints, run "terraform providers".

Failed to instantiate provider "registry.terraform.io/hashicorp/local" to obtain schema: unknown provider "registry.terraform.io/hashicorp/local"

Terraform plan will not work directly.

```

State Advanced State Management
iac-server $ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.4.0...
- Installed hashicorp/local v2.4.0 (self-signed, key ID 34365D9472D7468F)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/plugins/signing.html

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* hashicorp/local: version = "~> 2.4.0"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
iac-server $

```

Initialize terraform .

What was the version of the local provider plugin that was downloaded?

```

State Advanced State Management
iac-server $ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Installing hashicorp/local v2.4.0...
- Installed hashicorp/local v2.4.0 (self-signed, key ID 34365D9472D7468F)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/plugins/signing.html

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* hashicorp/local: version = "~> 2.4.0"

```

We can see local version .

commands will detect it and remind you to do so if necessary.

```
iac-server $ terraform plan
```

**Error: Missing required argument**

```
on main.tf line 1, in resource "local_file" "games":
 1: resource "local_file" "games" {
```

The argument "filename" is required, but no definition was found.

**Error: Unsupported argument**

```
on main.tf line 2, in resource "local_file" "games":
 2:   file      = "/root/favorite-games"
```

An argument named "file" is not expected here.

Invalid arguments are given in resource block so terraform plan is failed . file is not valid .

Fix the argument :

terraform-projects > HCL > main.tf > ...

```
0 references
1  resource "local_file" "games" {
2    filename      = "/root/favorite-games"
3    content       = "FIFA 21"
4  }
5
```

```
iac-server $ terraform plan
```

Refreshing Terraform state in-memory prior to plan...

The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

-----

An execution plan has been generated and is shown below.  
Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

```
# local_file.games will be created
+ resource "local_file" "games" {
+   content              = "FIFA 21"
+   content_base64sha256 = (known after apply)
+   content_base64sha512 = (known after apply)
+   content_md5          = (known after apply)
+   content_sha1         = (known after apply)
+   content_sha256       = (known after apply)
+   content_sha512       = (known after apply)
+   directory_permission = "0777"
+   file_permission      = "0777"
+   filename             = "/root/favorite-games"
+   id                   = (known after apply)
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

-----

Note: You didn't specify an "-out" parameter to save this plan, so Terraform can't guarantee that exactly these actions will be performed if "terraform apply" is subsequently run.

```
iac-server $ terraform apply
```

An execution plan has been generated and is shown below.  
Resource actions are indicated with the following symbols:  
+ create

Terraform will perform the following actions:

```
# local_file.games will be created
+ resource "local_file" "games" {
  + content                = "FIFA 21"
  + content_base64sha256  = (known after apply)
  + content_base64sha512 = (known after apply)
  + content_md5           = (known after apply)
  + content_sha1          = (known after apply)
  + content_sha256        = (known after apply)
  + content_sha512        = (known after apply)
  + directory_permission = "0777"
  + file_permission       = "0777"
  + filename              = "/root/favorite-games"
  + id                   = (known after apply)
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.

Enter a value: yes

local\_file.games: Creating...

local\_file.games: Creation complete after 0s [id=f68b901eb16aff12e9458bdb656a7df8d3425d4c]

We have now created our very first resource using Terraform! Next, let's work on updating the resource. If you look at the output produced by the terraform plan and terraform apply commands closely, we can see that the file content is printed on the screen. Since we do not want this to happen, we have updated the resource type. What is the resource type that we have updated?

```
resource "local_sensitive_file" "games" {
  filename      = "/root/favorite-games"
  content       = "FIFA 21"
  sensitive_content = "FIFA 21"
}
```

That's right, we have made use of the local\_sensitive\_file resource type to mask the contents of the file from the execution plan.

However, something is wrong. If we run terraform plan or terraform apply now we see an error!

```
iac-server $ terraform plan
```

**Error: Unsupported argument**

on main.tf line 4, in resource "local\_sensitive\_file" "games":  
4: sensitive\_content = "FIFA 21"

An argument named "sensitive\_content" is not expected here.

```
iac-server $
```

Delete the line containing the argument called sensitive\_content and then run terraform plan and then terraform apply to re-create the file.

```
terraform-projects > HCL > main.tf > games
```

```
0 references
```

```
1 ∨ resource "local_sensitive_file" "games" {
2   filename      = "/root/favorite-games"
3   content       = "FIFA 21"
4 }
```

```
main.tf: terraform:crstate
```

```
iac-server $ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
```

```
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
```

```
local_file.games: Refreshing state... [id=f68b901eb16aff12e9458bdb656a7df8d3425d4c]
```

```
-----
An execution plan has been generated and is shown below.
```

```
Resource actions are indicated with the following symbols:
```

- + create
- destroy

```
Terraform will perform the following actions:
```

```
# local_file.games will be destroyed
```

```
- resource "local_file" "games" {
  - content              = "FIFA 21" -> null
  - content_base64sha256 = "0QatlFV19H412mXNB5/Y9evwrDIxEW4ooJpmph2eoUY=" -> null
  - content_base64sha512 = "F0zLI9RS+tFB53xwISp1R3wvRQ/Sw4hsxwysWamsAk/cNyHr/X/pmmy
  - content_md5          = "44a271e06ddd134cdbeab299288422f3" -> null
  - content_sha1         = "f68b901eb16aff12e9458bdb656a7df8d3425d4c" -> null
  - content_sha256       = "d106ad95f565f47e35da65cd079fd8f5ebf0ac3231116e28a09a66a
  - content_sha512       = "174ccb23d452fad141e77c70212a75477c2f450fd2c3886cc70cac5
c9fce6d1457d6622" -> null
  - directory_permission = "0777" -> null
  - file_permission      = "0777" -> null
  - filename              = "/root/favorite-games" -> null
  - id                    = "f68b901eb16aff12e9458bdb656a7df8d3425d4c" -> null
}
```

```
# local_sensitive_file.games will be created
```

```
+ resource "local_sensitive_file" "games" {
  + content              = (sensitive value)
  + content_base64sha256 = (known after apply)
```

```

iac-server $ terraform apply
local_file.games: Refreshing state... [id=f68b901eb16aff12e9458bdb656a7df8d3425d4c]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
  - destroy

Terraform will perform the following actions:

# local_file.games will be destroyed
- resource "local_file" "games" {
  - content          = "FIFA 21" -> null
  - content_base64sha256 = "0QatlFV19H412mXNB5/Y9evwrDIxEW4ooJpmpH2eoUY=" -> null
  - content_base64sha512 = "F0zLI9RS+tFB53xwISp1R3wvRQ/Sw4hsxwysWamsAk/cNyHr/X/pm
  - content_md5        = "44a271e06ddd134cdbeab299288422f3" -> null
  - content_sha1       = "f68b901eb16aff12e9458bdb656a7df8d3425d4c" -> null
  - content_sha256     = "d106ad95f565f47e35da65cd079fd8f5ebf0ac3231116e28a09a6
  - content_sha512     = "174ccb23d452fad141e77c70212a75477c2f450fd2c3886cc70ca
c9fce6d1457d6622" -> null
  - directory_permission = "0777" -> null
  - file_permission      = "0777" -> null
  - filename             = "/root/favorite-games" -> null
  - id                   = "f68b901eb16aff12e9458bdb656a7df8d3425d4c" -> null
}

# local_sensitive_file.games will be created
+ resource "local_sensitive_file" "games" {
  + content          = (sensitive value)
  + content_base64sha256 = (known after apply)
  + content_base64sha512 = (known after apply)
  + content_md5        = (known after apply)
  + content_sha1       = (known after apply)
  + content_sha256     = (known after apply)
  + content_sha512     = (known after apply)
  + directory_permission = "0700"
}

```

Notice that the content of the file was not displayed when using `local_sensitive_file` instead of the `local_file` resource.

Also note that as Terraform follows an immutable infrastructure approach, the file was recreated although the contents are the same.

```

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
iac-server $ terraform destroy
local_sensitive_file.games: Refreshing state... [id=f68b901eb16aff12e9458bdb656a7df8d3425d4c]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# local_sensitive_file.games will be destroyed
- resource "local_sensitive_file" "games" {
  - content                = (sensitive value)
  - content_base64sha256 = "0QatlFVl9H412mXNB5/Y9evwrDIXEW4ooJpmpH2eoUY=" -> null
  - content_base64sha512 = "F0zLI9RS+tFB53xwISp1R3wvRQ/Sw4hsxwysWamsAk/cNyHr/X/pmmykTTCuvkI" -> null
  - content_md5           = "44a271e06ddd134cdbeab299288422f3" -> null
  - content_sha1          = "f68b901eb16aff12e9458bdb656a7df8d3425d4c" -> null
  - content_sha256        = "d106ad95f565f47e35da65cd079fd8f5ebf0ac3231116e28a09a66a61d9ea14" -> null
  - content_sha512        = "174ccb23d452fad141e77c70212a75477c2f450fd2c3886cc70cac59a9ac02" -> null
  - directory_permission = "0700" -> null
  - file_permission      = "0700" -> null
  - filename              = "/root/favorite-games" -> null
  - id                    = "f68b901eb16aff12e9458bdb656a7df8d3425d4c" -> null
}

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

local_sensitive_file.games: Destroying... [id=f68b901eb16aff12e9458bdb656a7df8d3425d4c]
local_sensitive_file.games: Destruction complete after 0s

```

Terraform providers :

```

0 references
1  resource "local_file" "things-to-do" {
2      filename    = "/root/things-to-do.txt"
3      content     = "Clean my room before Christmas\nComplete the CKA Certification!"
4  }
0 references
5  resource "local_file" "more-things-to-do" {
6      filename    = "/root/more-things-to-do.txt"
7      content     = "Learn how to play Astronomia on the guitar!"
8  }

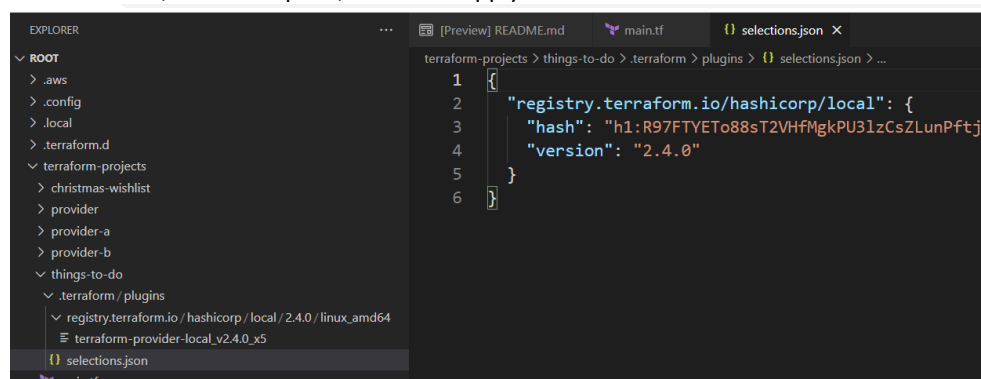
```

Using this configuration file we are configuring 2 resources .

Number of providers initialized in this resource – 0 .

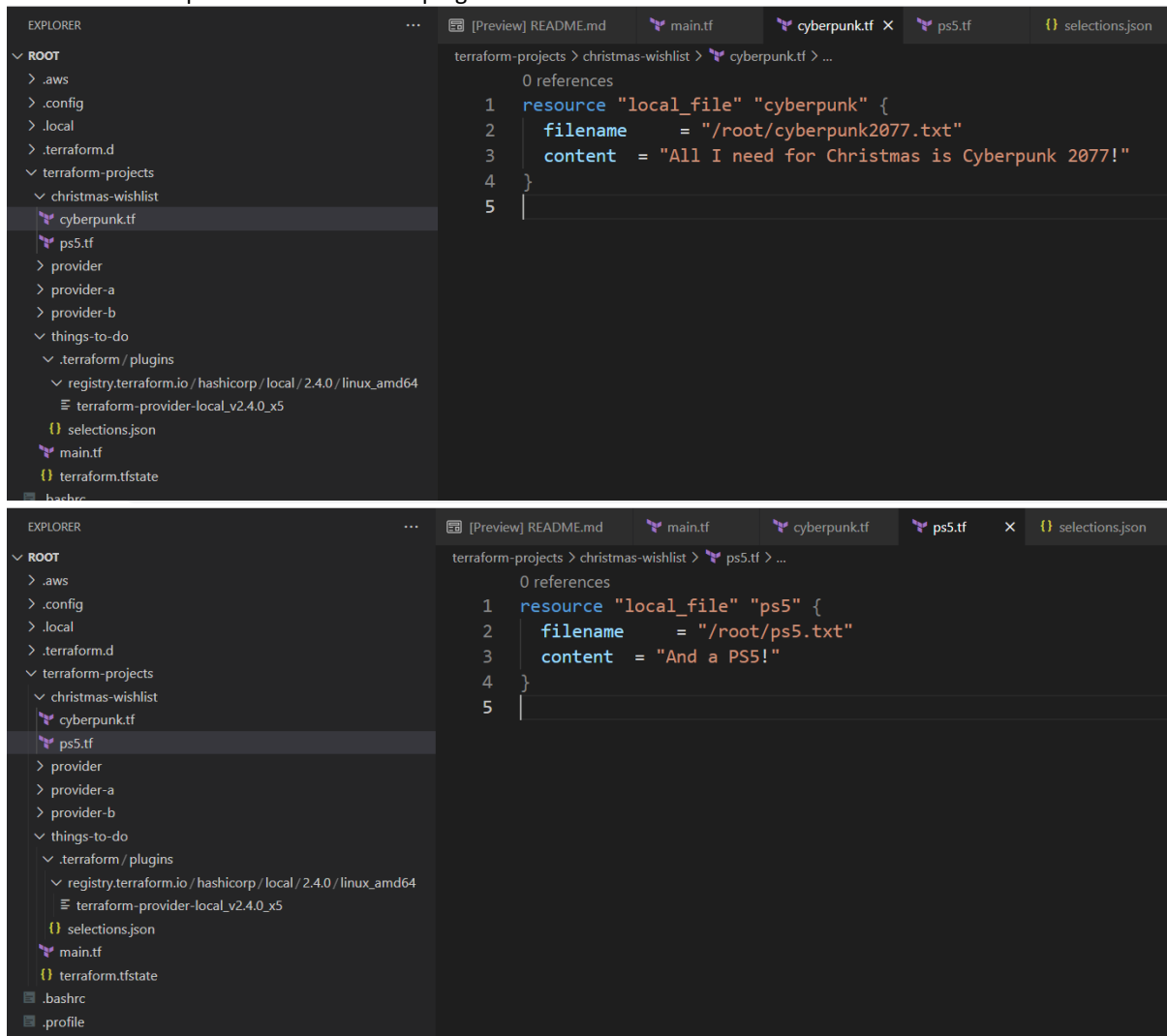
Now, go ahead and create these resources using terraform!

terraform init , terraform plan , terraform apply .





We can see local provider is installed in plugins folder in terraform .



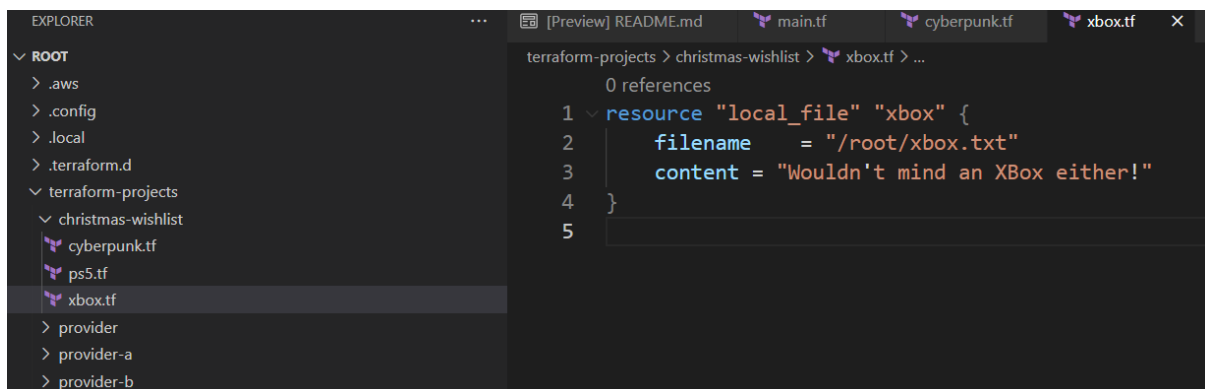
Create a new configuration file within the same directory called xbox.tf. This file should make use of the same local\_file resource type with the below requirements:

Resource Name: xbox

filename: /root/xbox.txt

content: Wouldn't mind an XBox either!

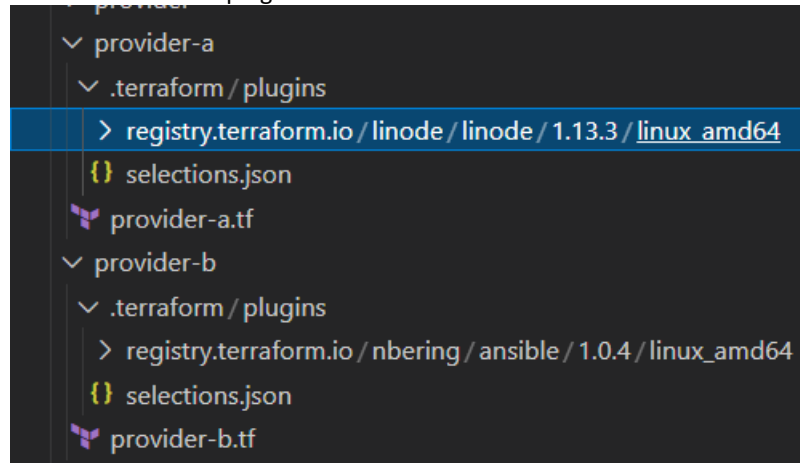
Once the configuration file has been created, use the terraform workflow to create this resource.



terraform init , terraform plan , terraform apply



Downloaded below plugins .



Multiple providers :

we can use as many providers as needed within the same configuration.

Now, run the terraform init command and inspect the .terraform/plugins directory. Count the number of plugins downloaded.

```
iac-server $ terraform init
```

**Initializing the backend...**

**Initializing provider plugins...**

- Finding latest version of hashicorp/local...
- Finding latest version of hashicorp/random...
- Installing hashicorp/local v2.4.0...
- Installed hashicorp/local v2.4.0 (self-signed, key ID 34365D9472D7468F)
- Installing hashicorp/random v3.5.1...
- Installed hashicorp/random v3.5.1 (self-signed, key ID 34365D9472D7468F)

Partner and community providers are signed by their developers.

If you'd like to know more about provider signing, you can read about it here:  
<https://www.terraform.io/docs/plugins/signing.html>

The following providers do not have any version constraints in configuration, so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking changes, we recommend adding version constraints in a `required_providers` block in your configuration, with the constraint strings suggested below.

```
* hashicorp/local: version = "~> 2.4.0"
* hashicorp/random: version = "~> 3.5.1"
```

Now, Navigate to the directory /root/terraform-projects/MPL. Create a new configuration file called pet-name.tf.

This file should make use of the local\_file and random\_pet resource type with the below requirements:

local\_file resource details:

Resource name = "my-pet"

File name = "/root/pet-name"

Content = "My pet is called finnegan!!"

random\_pet resource details:  
Resource name = "other-pet"  
Length = "1"  
Prefix = "Mr"  
Separator = "."

Once the configuration file has been created, use the terraform workflow to create this resource.

```
[Preview] README.md multi-provider.tf pet-name.tf X
terraform-projects > MPL > pet-name.tf > other-pet
0 references
1 resource "local_file" "my-pet" {
2   filename = "/root/pet-name"
3   content = "My pet is called finnegan!!"
4 }
0 references
5 resource "random_pet" "other-pet" {
6   length = "1"
7   prefix = "Mr"
8   separator = "."
9 }
```

```
[Preview] README.md multi-provider.tf pet-name.tf cloud-provider.tf X
terraform-projects > provider > cloud-provider.tf > ...
1
0 references
2 resource "aws_instance" "ec2_instance" {
3   ami = "ami-0eda277a0b884c5ab"
4   instance_type = "t2.large"
5 }
6
7
0 references
8 resource "aws_ebs_volume" "ec2_volume" {
9   availability_zone = "eu-west-1"
10  size = 10
11 }
12
```

What is the instance\_type configured with the resource type called aws\_instance?

```
EXPLORER
ROOT
  .aws
  .config
  .local
  .terraform.d
  terraform-projects
    MPL
      .terraform
      pet-name.tf
      terraform.tfstate
    multi-provider
      .terraform
      multi-provider.tf
    provider
      cloud-provider.tf
      kube.tf
    provider-a
  .bashrc
  .profile

[Preview] README.md multi-provider.tf pet-name.tf cloud-provider.tf kube.tf X
terraform-projects > provider > kube.tf > ...
1
0 references
2 resource "local_file" "data" {
3   filename = "/root/k8s.txt"
4   content = "kubernetes the hard way!"
5 }
6
7
0 references
8 resource "kubernetes_namespace" "dev" {
9   metadata {
10    name = "development"
11  }
12 }
13
```

Let's get some more practice! Now navigate to the directory path `/root/terraform-projects/provider-a`. Create a configuration file called `code.tf`.

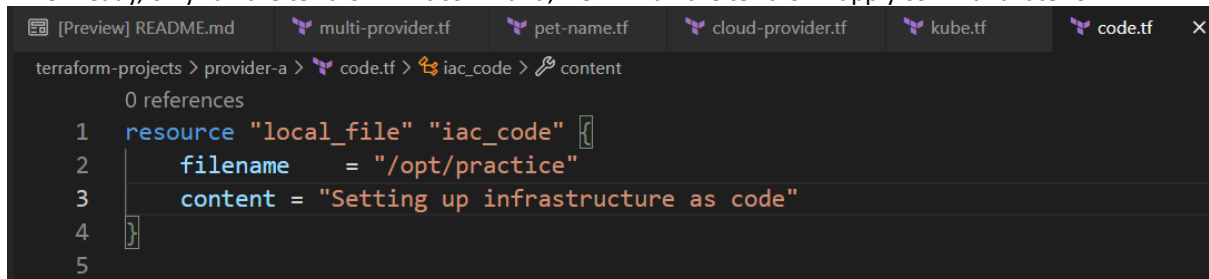
Using the `local_file` resource type, write the resource block with the below requirements into the file:

Resource name = `iac_code`

File name = `/opt/practice`

Content = Setting up infrastructure as code

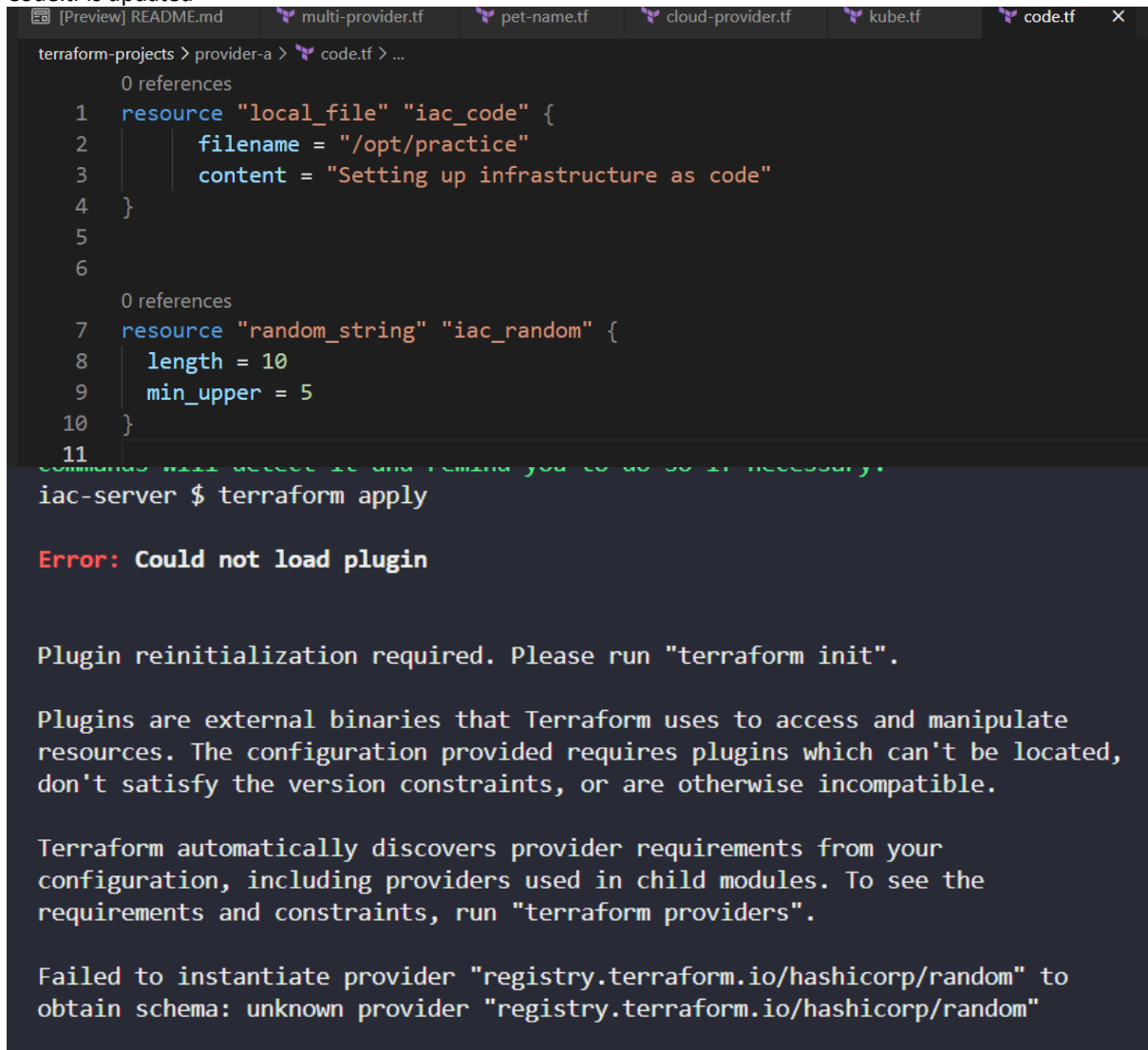
When ready, only run the `terraform init` command, we will run the `terraform apply` command later on.



The screenshot shows a code editor with a dark theme. The top bar has tabs for 'README.md', 'multi-provider.tf', 'pet-name.tf', 'cloud-provider.tf', 'kube.tf', and 'code.tf'. The breadcrumb navigation shows the path: 'terraform-projects > provider-a > code.tf > iac\_code > content'. The code content is as follows:

```
0 references
1 resource "local_file" "iac_code" {
2     filename = "/opt/practice"
3     content = "Setting up infrastructure as code"
4 }
5
```

Code.tf is updated



The screenshot shows a terminal window with a dark theme. The top bar has tabs for 'README.md', 'multi-provider.tf', 'pet-name.tf', 'cloud-provider.tf', 'kube.tf', and 'code.tf'. The terminal output is as follows:

```
terraform-projects > provider-a > code.tf > ...
0 references
1 resource "local_file" "iac_code" {
2     filename = "/opt/practice"
3     content = "Setting up infrastructure as code"
4 }
5
6
0 references
7 resource "random_string" "iac_random" {
8     length = 10
9     min_upper = 5
10 }
11
```

Commands will detect it and remind you to do so if necessary.

```
iac-server $ terraform apply
```

**Error: Could not load plugin**

Plugin reinitialization required. Please run "terraform init".

Plugins are external binaries that Terraform uses to access and manipulate resources. The configuration provided requires plugins which can't be located, don't satisfy the version constraints, or are otherwise incompatible.

Terraform automatically discovers provider requirements from your configuration, including providers used in child modules. To see the requirements and constraints, run "terraform providers".

Failed to instantiate provider "registry.terraform.io/hashicorp/random" to obtain schema: unknown provider "registry.terraform.io/hashicorp/random"

This is because whenever we add a resource for a provider that has not been used so far in the configuration directory, we have to initialize the directory by running `terraform init` command.

Let's do that now. Run `terraform init` followed by `terraform apply` command.