Terraform and docker

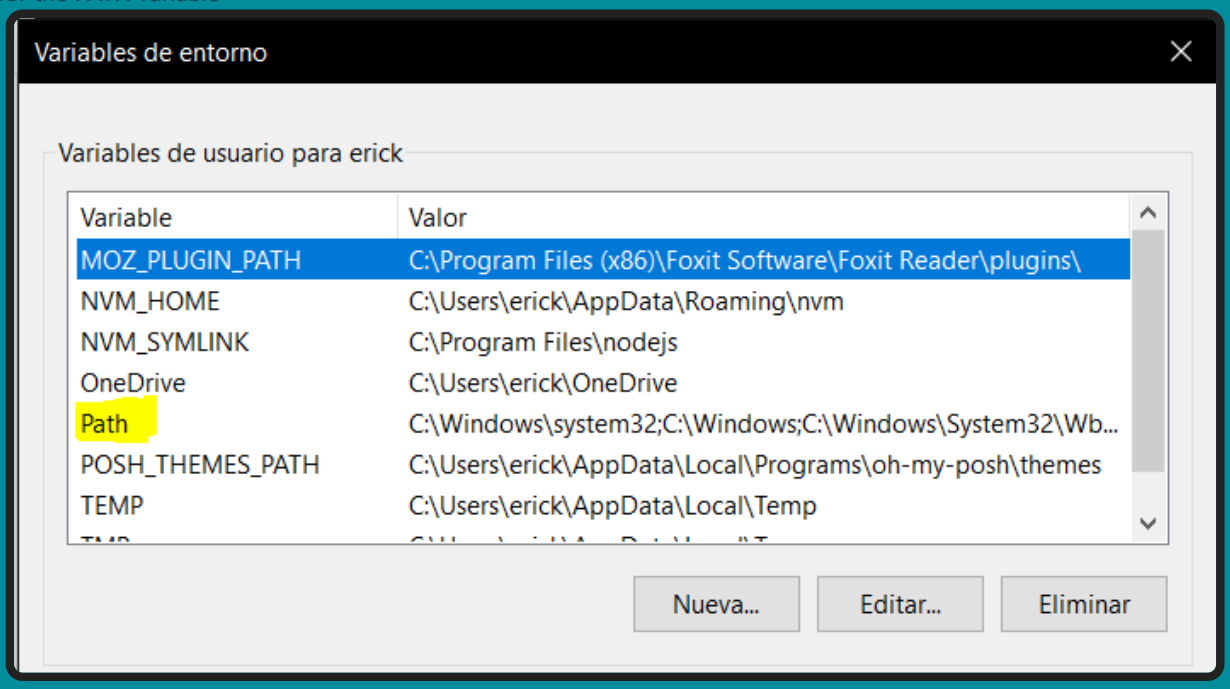
Abstract

Create a terraform config with any provider of choice (Docker if you prefer) and share the terraform config files (with .tf extension) and state file (with .tfstate extension)

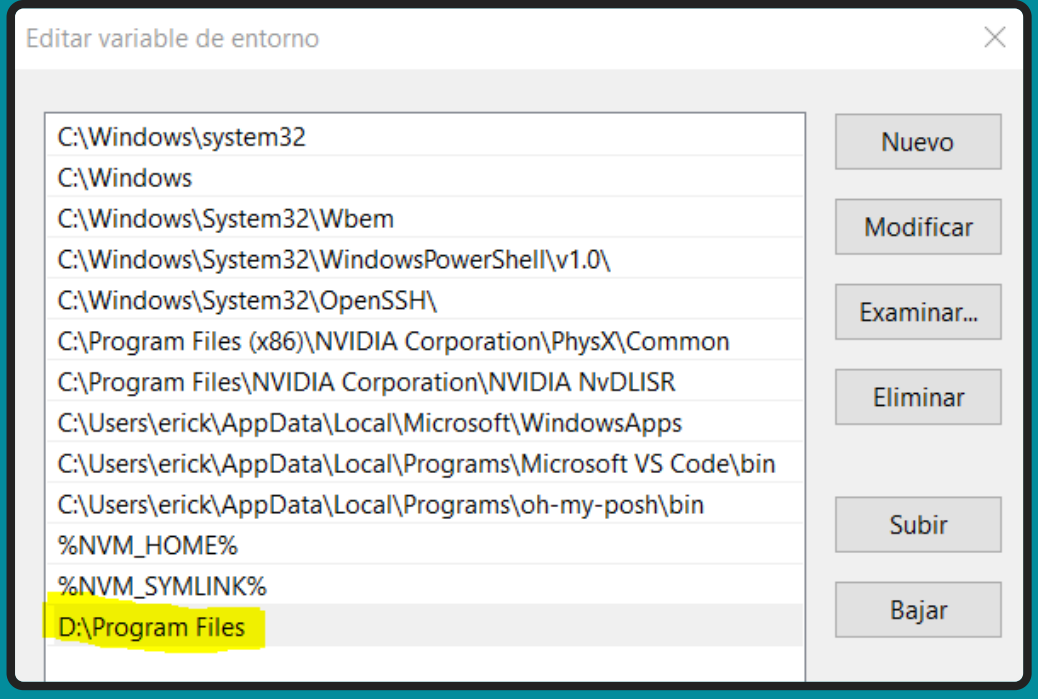
Installing Terraform (Windows 10)

For installing and using terreform, first I downloaded the .exe file from the web * **https://developer.hashicorp.com/terraform/downloads**. After I downloaded, I moved the file to my path **D:\Program Files**. In order to run the program in the cmd, we must set the path in the **path enviroment variable**. To set this, you must:

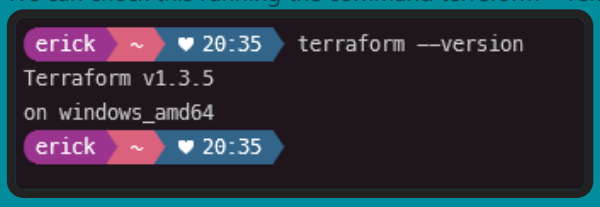
1. Go to **Control Panel->System->Sistem Settings->Environmet Variables**
2. Search for the **PATH** variable



3. Click **edit**, then **New** and add the path where terraform.exe is located



4. Terraform should be already installed. We can check this running the command terreform --version in the cmd.



Terraform *configuration.*

The set of files used to describe infrastructure in Terraform is known as a Terraform *configuration*.

Each Terraform configuration must be in its own working directory. We created a directory and the a file called **main.tf**
The main.tf contains Terraform settings, including the required providers Terraform will use to provision your infrastructure.

Here is the code from my main.tf

```
terraform {
        required_providers {
                docker = {
                        source = "kreuzwerker/docker"
                        version = ">= 2.21.0"
                }
        }
}

provider "docker" {
        host = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "jenkins" {
        name = "jenkins/jenkins:lts-jdk11"
}

resource "docker_container" "jenkins" {
        image = docker_image.jenkins.image_id
        name  = "jenkins_container_terreform"
        ports {
```

```
                    internal = 8080
                    external = 8080
            }
    }
```

- The **terraform {}** block contains Terraform settings, including the required providers Terraform will use to provision your infrastructure. In this example configuration, the docker provider's source is defined as **kreuzwerker/docker**
- The **provider block** configures the specified provider, in this case docker. A provider is a plugin that Terraform uses to create and manage your resources.
- **Resource blocks** define components of your infrastructure. A resource might be a physical or virtual component such as a Docker container, or it can be a logical resource such as a Heroku application. For this case, we are installing **jenkins** as a Docker container.

Once the main.tf is created, I initialized the directory with the command

```
terraform init
```

Initializing a configuration directory downloads and installs the providers defined in the configuration, which in this case is the docker provider.

In order to vallidate the configuration we made in the main.tf, you can run **terraform fmt** It automatically updates configurations in the current directory for readability and consistency.

```
terraform fmt
```

To validate the configuration, we use

```
terraform validate
```

It must show any mistake that you configuration have, or it shows you a message if there is not problem.

Shows me a warning

```
erick  main  → (main)  ♥ 20:51  terraform validate

Warning: Deprecated attribute

  on main.tf line 19, in resource "docker_container" "jenkins":
  19:   image = docker_image.jenkins.latest

The attribute "latest" is deprecated. Refer to the provider documentation for details.

Success! The configuration is valid, but there were some validation warnings as shown above.
```

After I corrected the main.tf file it just showed me a message

```
erick  main  → (main)  ♥ 20:52  terraform fmt
erick  main  → (main)  ♥ 20:52  terraform validate
Success! The configuration is valid.
```

Create infrastructure

Once the configuration is completed, qe can apply now the configuration with the **terraform apply**. It shows the actions Terraform will take in order to change your infrastructure to match the configuration and ask you if you want to proceed. You only need to write **yes** and the process will continue

```
terraform apply
```

```
erick  main  → (main)  ♥ 20:52  terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.jenkins will be created
  + resource "docker_container" "jenkins" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
```

We can use the command
terraform state list
to list of the resources in the project's state

```
erick  main  → (main)  ♥ 21:11  terraform state list
docker_container.jenkins
docker_image.jenkins
```

Also we can see in docker desktop that our container is already created

Only show running containers Search

NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
jenkins_container_terreform ab50eb7aa825	jenkins/jenkins:lts-jdk11	Created	8080:8080		