



**POLYTECHNIQUE
MONTRÉAL**

UNIVERSITÉ
D'INGÉNIERIE

Devoir/Homework (CI/CD pipeline Terraform, Git, Azure)

Contents

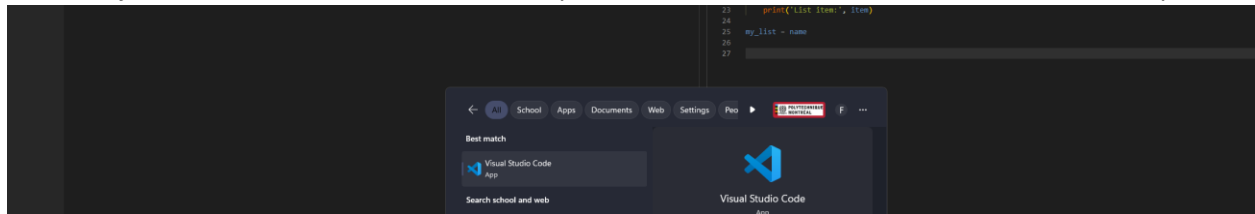
Devoir 1 (CI/CD pipeline)	1
Introduction	3
Partie 1 Installation et configuration de VS code	3
Partie 2 Installation et configuration de GitHub Desktop.....	5
Partie 3 Installation et configuration de Terraform CLI et Terraform cloud.....	6
Partie 4 Configuration du compte Microsoft Azure.....	15
Partie 5 Arrimage et connexion entre Github et Terraform	16
Partie 6 Arrimage et connexion entre Terraform cloud et MS Azure.....	17
Partie 7 Déploiement de ressource groupe à partir de votre pipeline dans MS Azure	20
Partie 8 Déploiement de Réseau Virtuel à partir de votre pipeline dans MS Azure.....	20
Partie 9 Déploiement d'une VM à partir de votre pipeline dans MS Azure	21
Partie 10 Déploiement d'un container Docker à partir de votre pipeline dans MS Azure	23
Conclusion.....	28
Références	28

Introduction

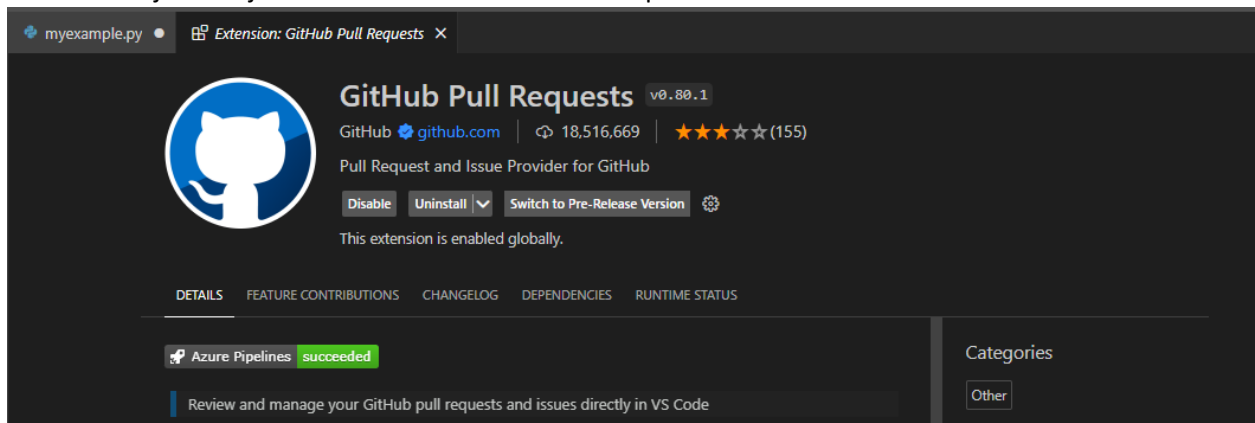
Dans le cadre de ce projet, nous avons entrepris une série d'étapes visant à configurer un environnement de développement et de déploiement utilisant plusieurs outils essentiels pour la gestion des infrastructures et des ressources cloud. Notre objectif principal était de mettre en place un flux de travail efficace pour le déploiement d'infrastructures dans Microsoft Azure, en utilisant des outils tels que Visual Studio Code, GitHub, Terraform CLI, Terraform Cloud et Azure itself. Ce projet nous a permis de comprendre les étapes nécessaires à la mise en place d'un pipeline de déploiement continu dans le cloud.

Partie 1 Installation et configuration de VS code

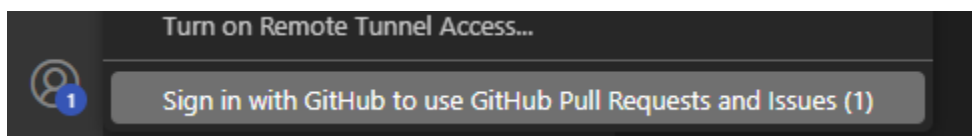
J'avais déjà VS Code d'installé donc il me manque seulement à mettre l'extension Github Pull Requests

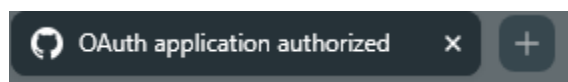
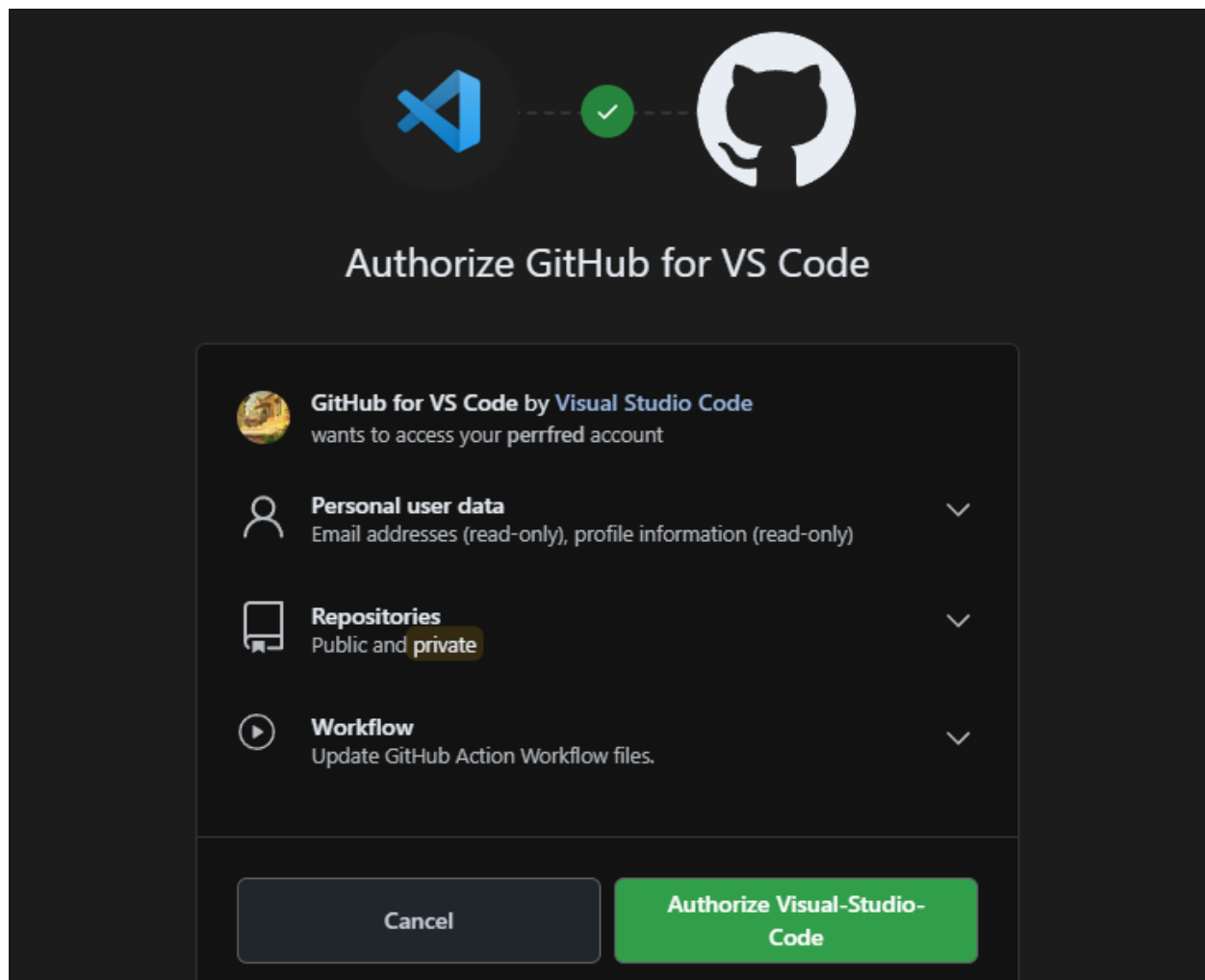


Maintenant je vais ajouter l'extension Github Pull Requests :

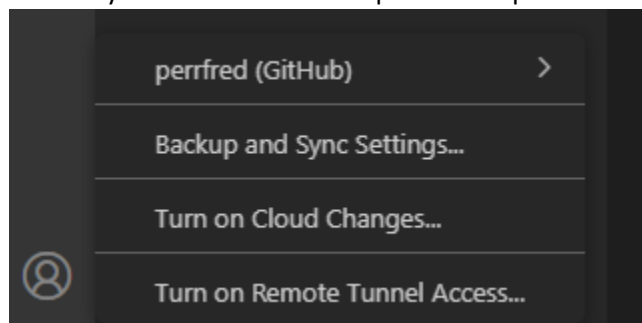


Ensuite je dois connecter mon compte Github





Nous voyons ensuite mon compte GitHub personnel connecté sur VSCode

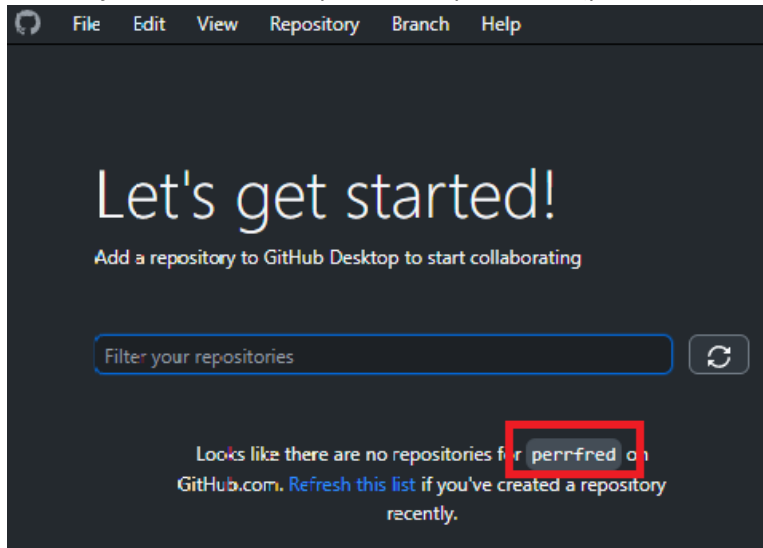


Partie 2 Installation et configuration de GitHub Desktop

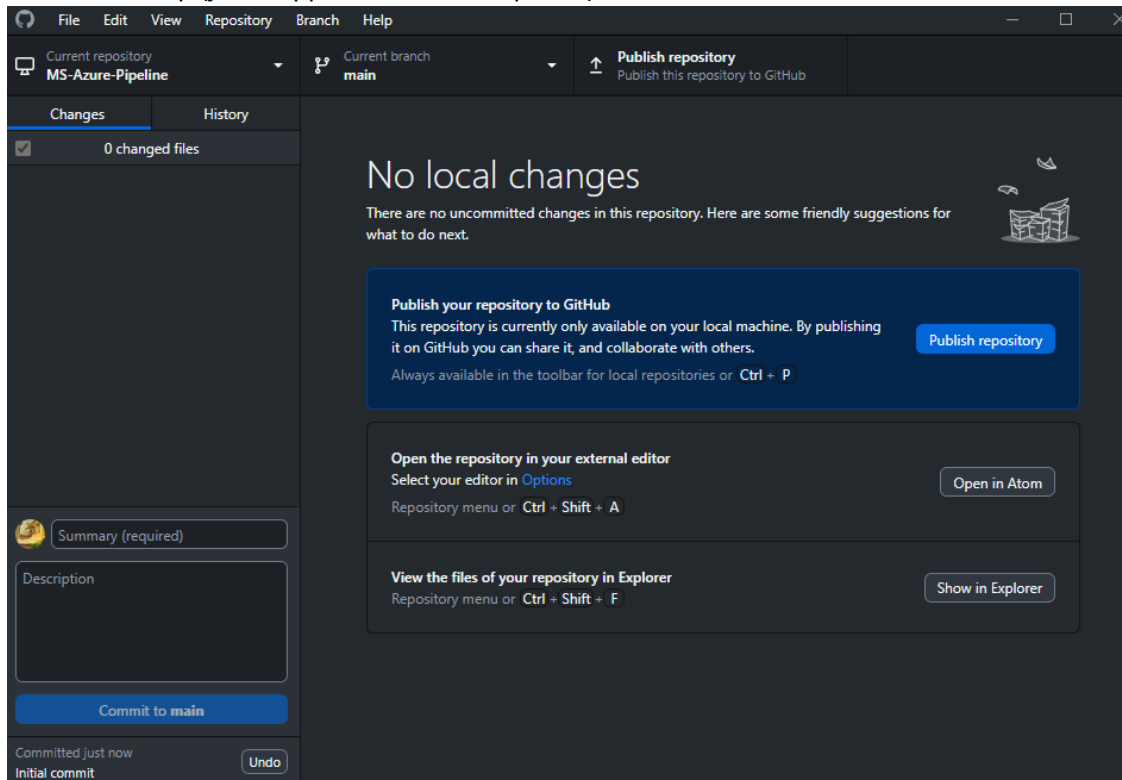
J'ai commencé par installer GitHub desktop à partir de leur site



Ensuite j'ai lié mon compte Github personnel (perrfred) sur l'application Github Desktop

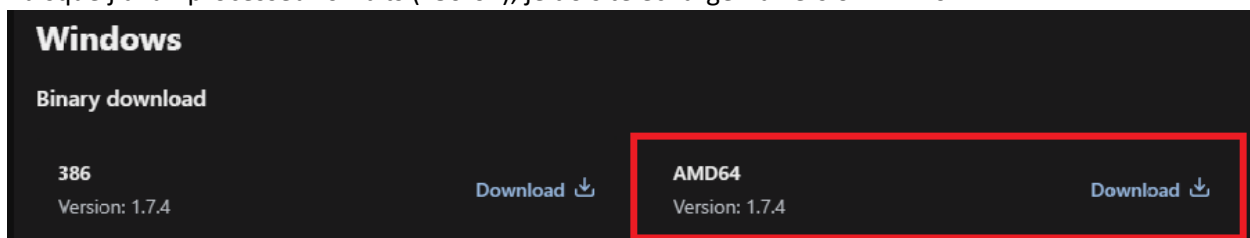


Nous créons ensuite un New Repository pour le cours et nous allons apparaître sur la page principal de Github Desktop (je l'ai appelé MS-Azure-Pipeline)

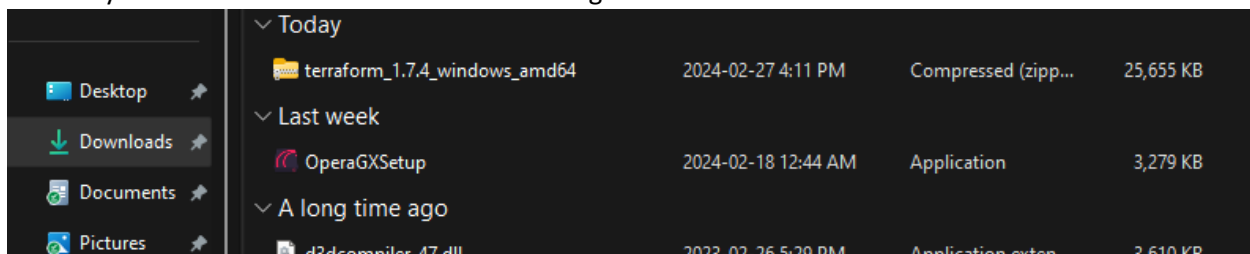


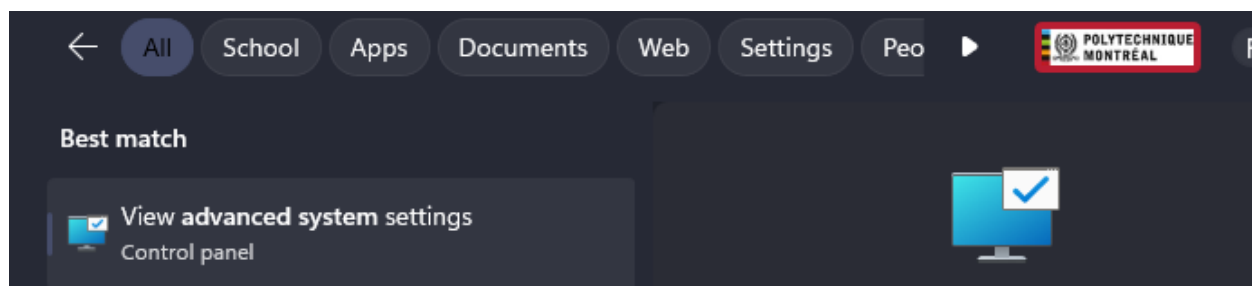
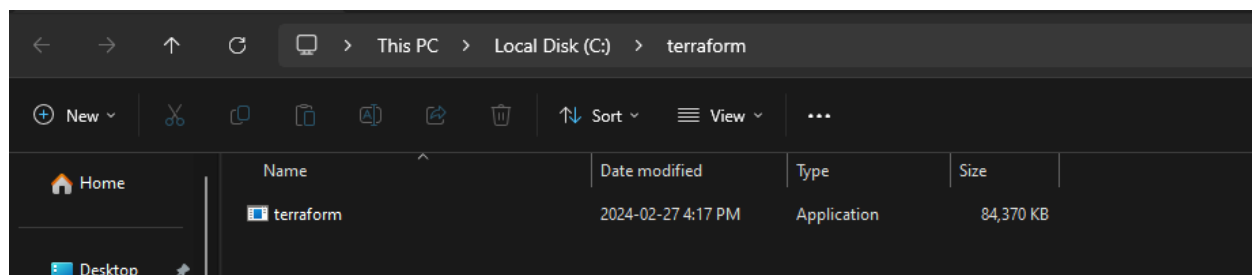
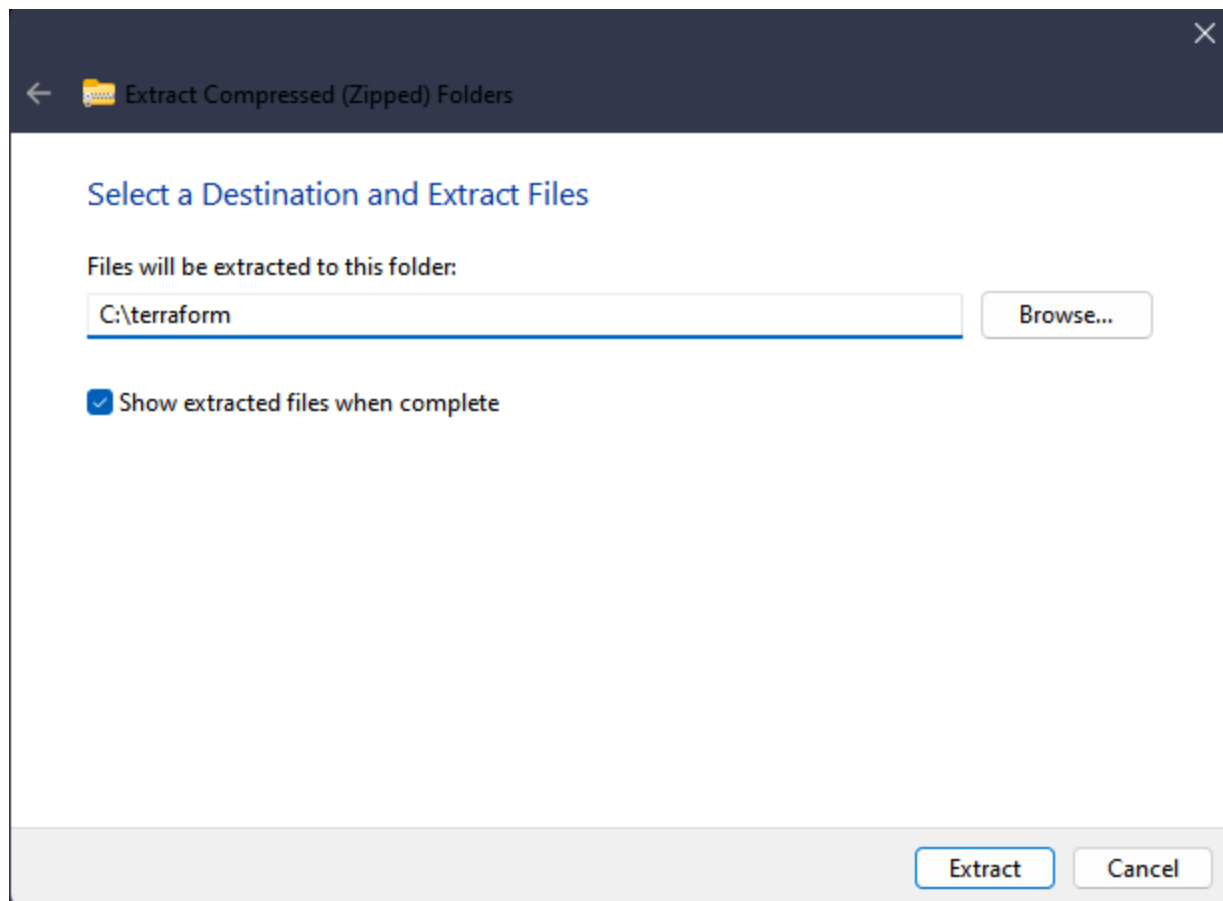
Partie 3 Installation et configuration de Terraform CLI et Terraform cloud

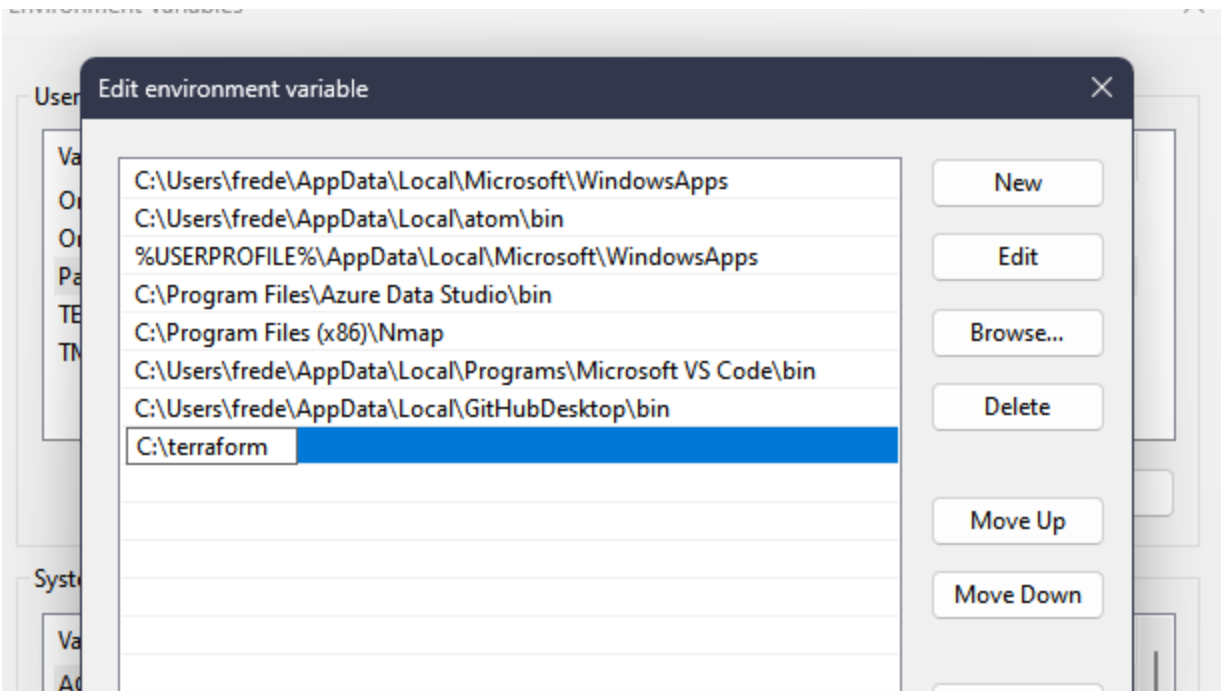
Puisque j'ai un processeur 64 bits (x86-64), je dois télécharger la version AMD64



Nous voyons ensuite le nouveau dossier téléchargé dans mes Downloads








```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\frede>terraform --version
Terraform v1.7.4
on windows_amd64
```


Nous créons ensuite notre compte Terraform cloud


Create an account

Have an account? [Sign in](#)


 **Continue with HCP account**

OR



Username



Email



Password



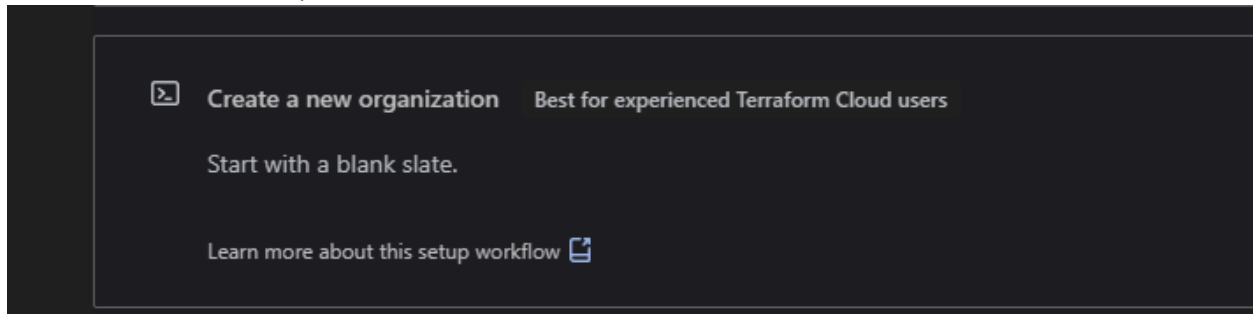
☒ I agree to the Terms of Use.

☒ I acknowledge the Privacy Policy.

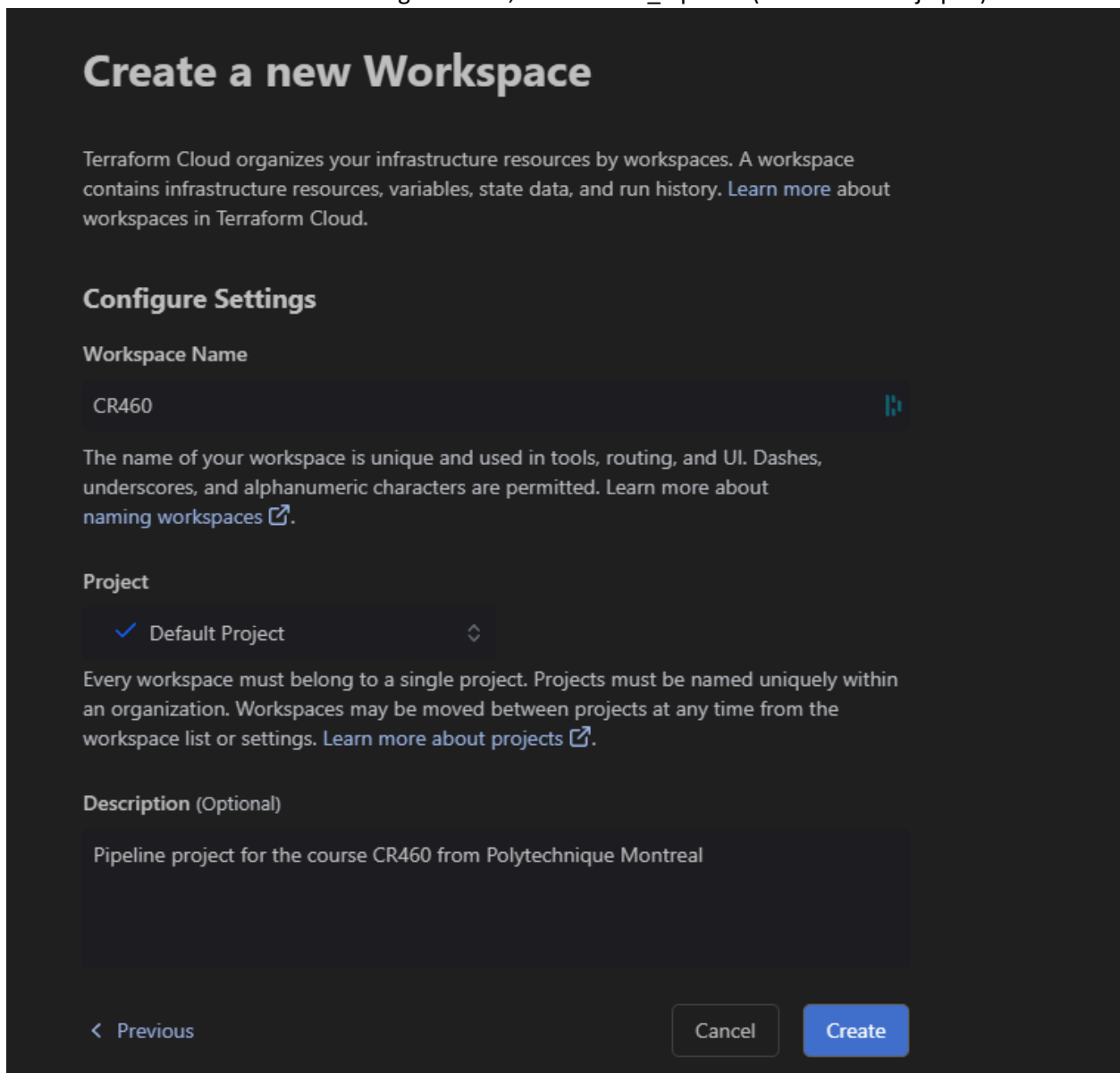
Please review the [Terms of Use](#) and [Privacy Policy](#).

Create account

Nous choisissons cette option



J'ai choisi le nom du cours comme organization, donc CR460_Pipeline (CR460 était déjà pris)



On choisi ensuite la deuxième méthode pour créer un workspace (VCS)

CR460_Pipeline / Projects & workspaces / New Workspace

Create a new Workspace

Terraform Cloud organizes your infrastructure resources by workspaces. A workspace contains infrastructure resources, variables, state data, and run history. [Learn more about workspaces in Terraform Cloud.](#)


1 Connect to VCS


2 Choose a repository


3 Configure settings


Connect to a version control provider

Choose the version control provider that hosts the Terraform configuration for this workspace.

 GitHub ^

 GitLab v

 Bitbucket v

 Azure DevOps v

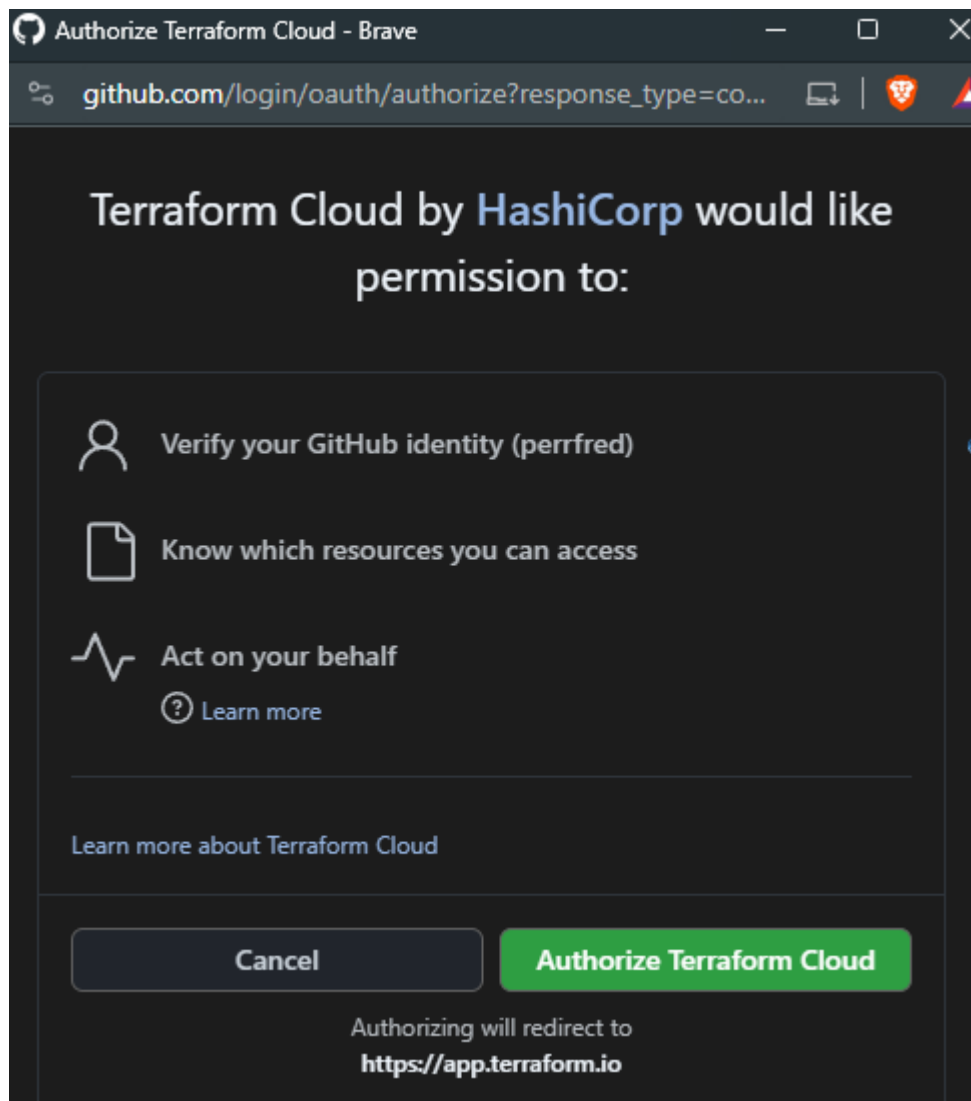
Version

GitHub.com

GitHub Enterprise

GitHub.com (Custom)

Cancel



Repository access

☐ **All repositories**


This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

☒ **Only select repositories**

Select at least one repository.
Also includes public repositories (read-only).

 **Select repositories** ▼

Selected 1 repository.

 perrfred/CR460


×

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 perrfred ▾



Repository name *

/ CR460

✔ CR460 is available.

Great repository names are short and memorable. Need inspiration? How about **sturdy-system** ?

Description (optional)

- ☐  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☒  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: Terraform ▾


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

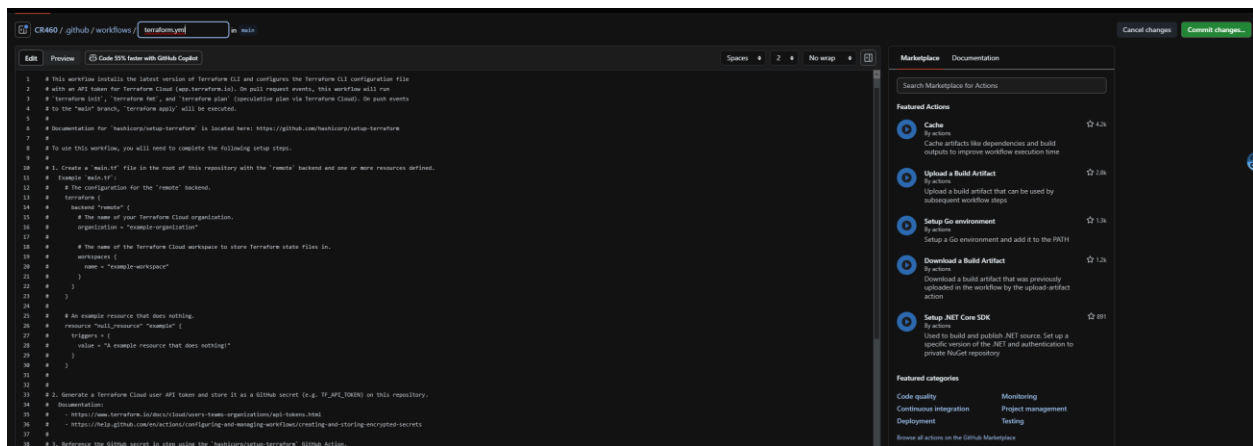
License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

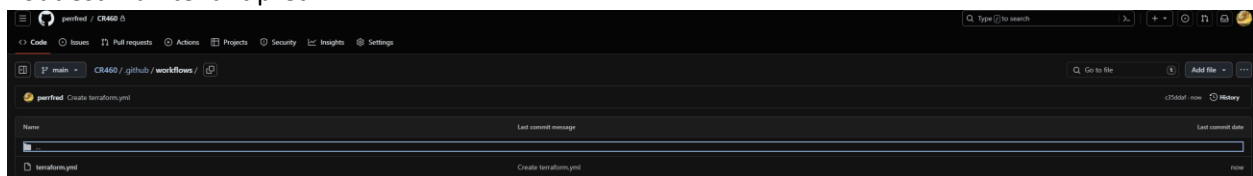
This will set  **main** as the default branch. Change the default name in your settings.

 You are creating a private repository in your personal account.

Create repository



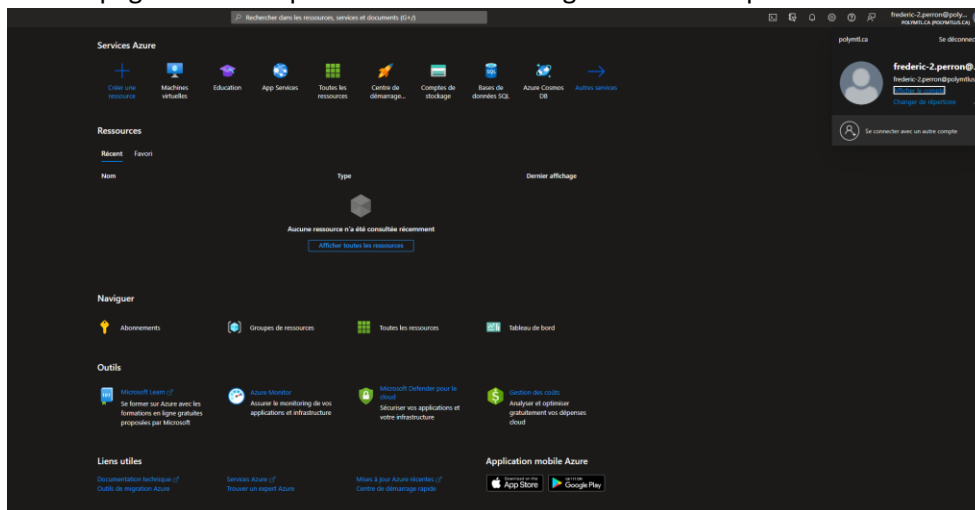
Tout est maintenant prêt



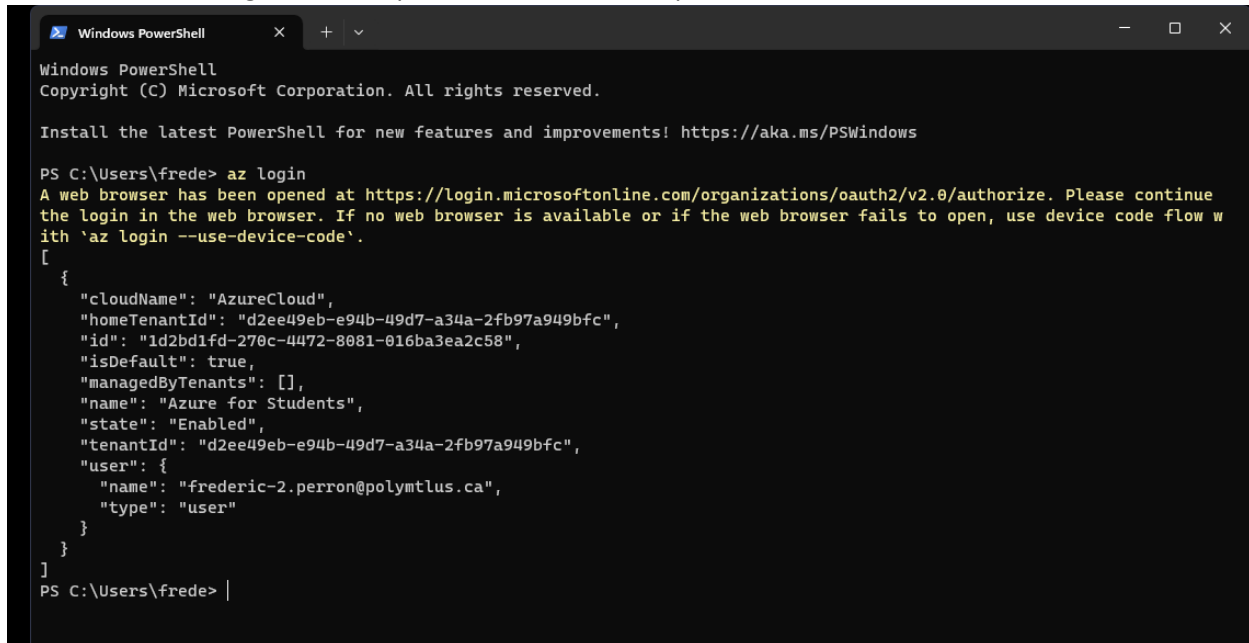
Partie 4 Configuration du compte Microsoft Azure

J'ai créé un compte avec le courriel de poly puisque ça nous procurait des services et autres gratuits.

Voici la page d'accueil après avoir créé et configuré mon compte Azure



J'ai ensuite téléchargé Azure CLI pour accéder le tout depuis mon terminal.



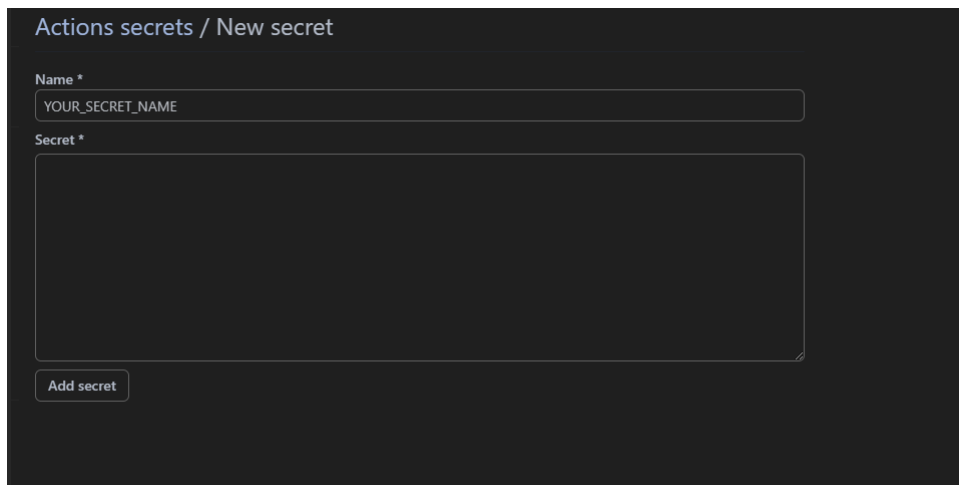
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\frede> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue
the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow w
ith 'az login --use-device-code'.
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "d2ee49eb-e94b-49d7-a34a-2fb97a949bfc",
    "id": "1d2bd1fd-270c-4472-8081-016ba3ea2c58",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure for Students",
    "state": "Enabled",
    "tenantId": "d2ee49eb-e94b-49d7-a34a-2fb97a949bfc",
    "user": {
      "name": "frederic-2.perron@polymtlus.ca",
      "type": "user"
    }
  }
]
PS C:\Users\frede> |
```

Partie 5 Arrimage et connexion entre Github et Terraform

Nous avons créé un API token, et nous créons par la suite un Actions secrets



Actions secrets / New secret

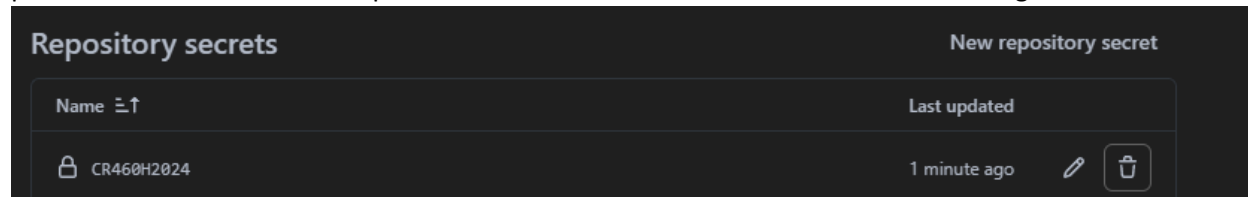
Name *

YOUR_SECRET_NAME

Secret *

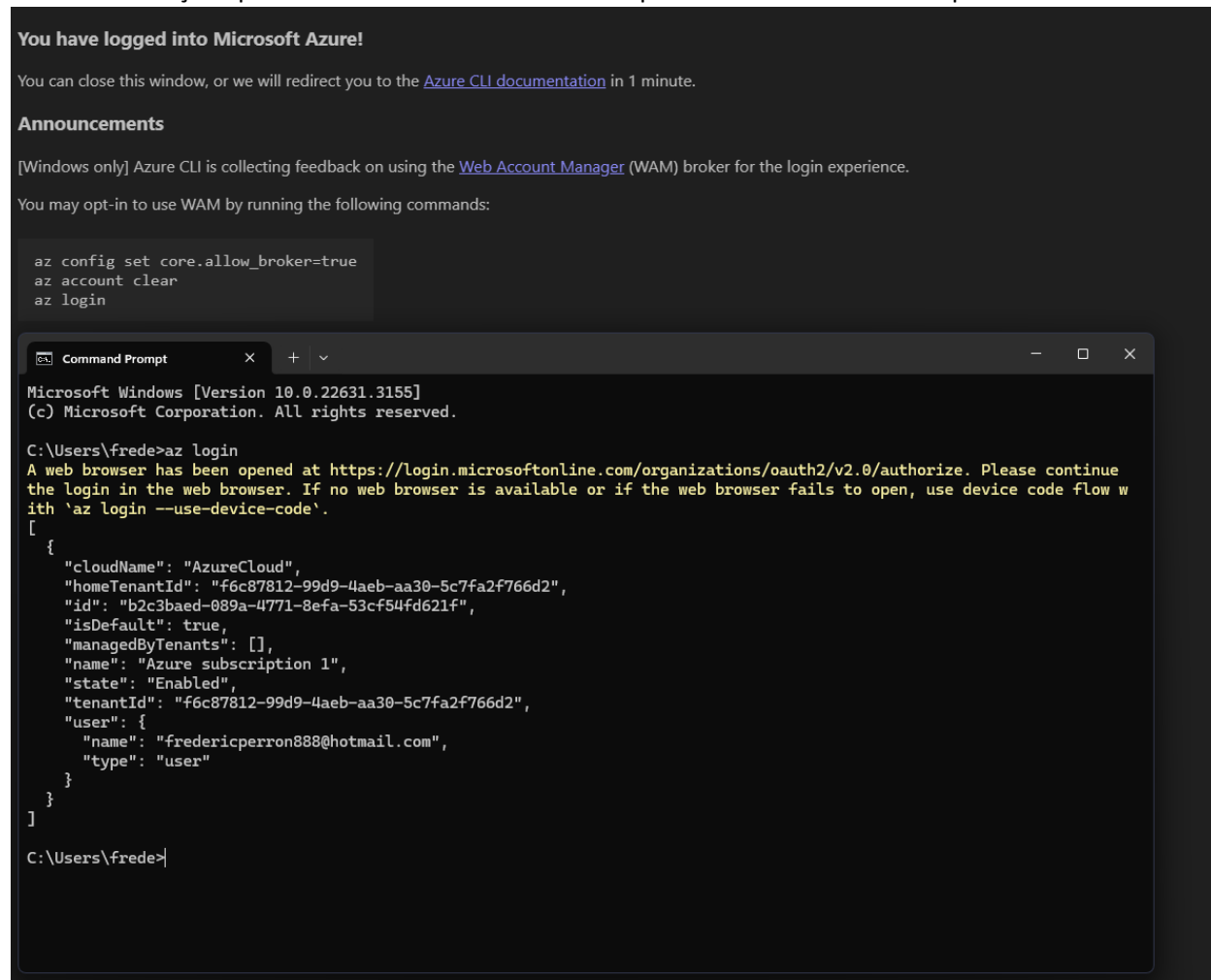
Add secret

Donc je mets comme nom CR460H2024, similaire à mes autres noms et repos, et j'entre le key dans la partie "Secret *". Nous avons par la suite créé le lien entre Terra et Github. Voir image ci-dessous



Partie 6 Arrimage et connexion entre Terraform cloud et MS Azure

Nous commençons par se connecter avec Azure CLI depuis notre Command Prompt



Nous pouvons voir lorsque nous faisons *init*, que tout est bien connecté ensemble.

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

- Finding hashicorp/azurerm versions matching "3.93.0"...
- Installing hashicorp/azurerm v3.93.0...
- Installed hashicorp/azurerm v3.93.0 (signed by HashiCorp)

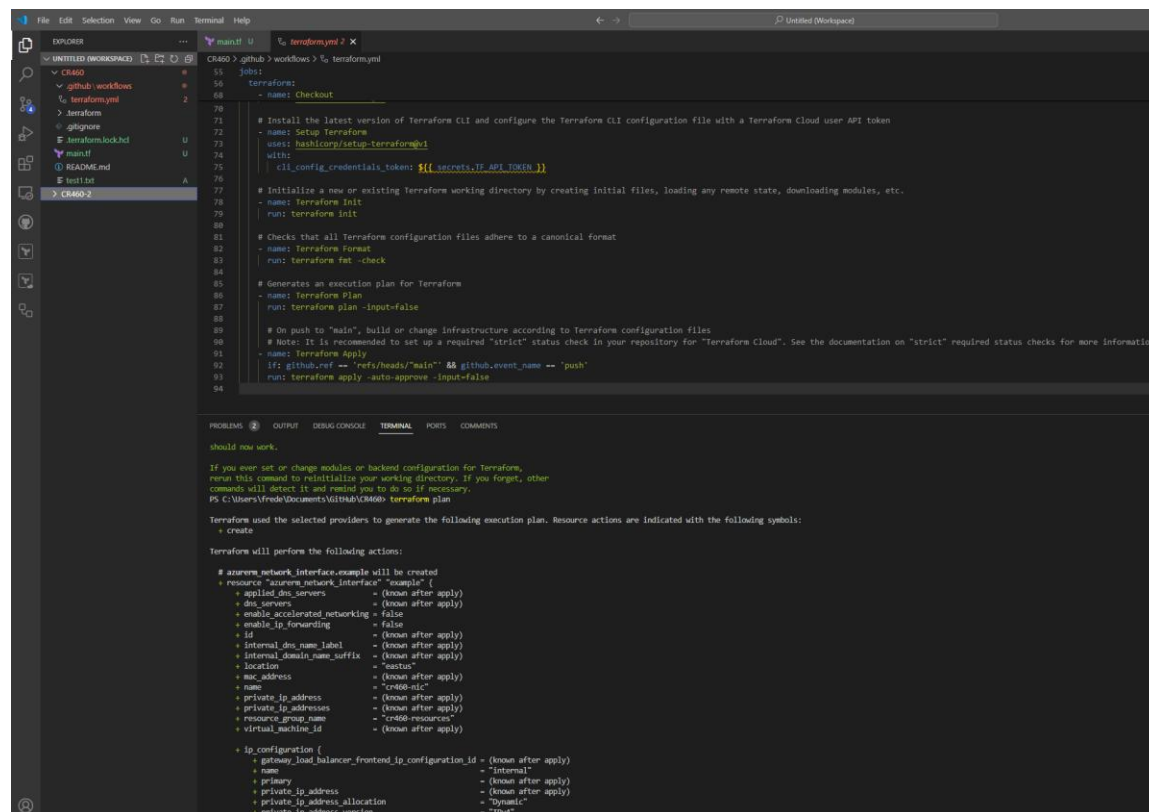
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
PS C:\Users\frede\Documents\GitHub\CR460>
```

Tout déploie comme prévu, pour démontrer que l'arrimage est bel et bien fait et que la connexion marche bel et bien.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a file tree with folders like 'main.tf' and 'terraform.lock.hcl'. The main editor area displays a workflow file named 'main.tf' with the following content:

```
jobs:
  terraform:
    - name: Checkout
    - name: Setup Terraform
      uses: hashicorp/setup-terraform@v1
      with:
        cli_config_credentials_token: ${secrets.TF_API_TOKEN}
    - name: Initialize a new or existing Terraform working directory by creating initial files, loading any remote state, downloading modules, etc.
      run: terraform init
    - name: Terraform Format
      run: terraform fmt -check
    - name: Terraform Plan
      run: terraform plan -input=false
    - name: Terraform Apply
      if: github.ref == 'refs/heads/main' && github.event_name == 'push'
      run: terraform apply -auto-approve -input=false
```

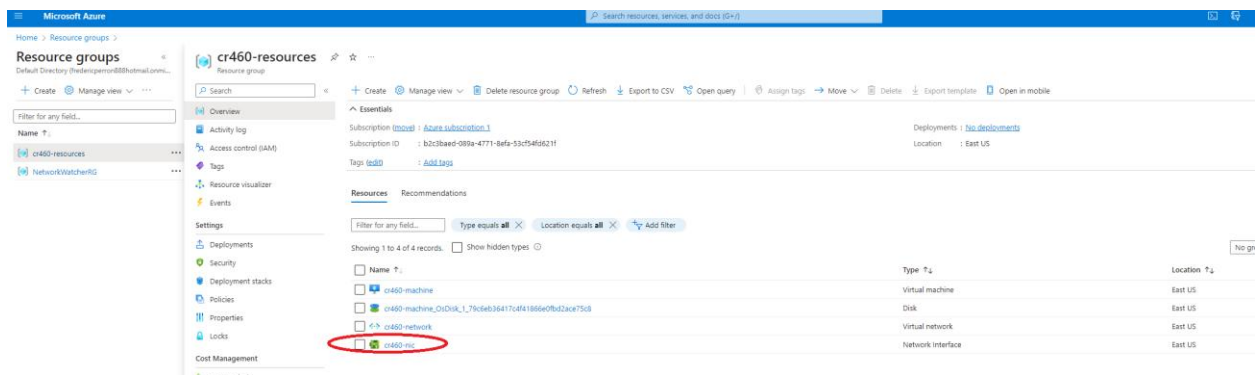
The terminal pane at the bottom shows the output of the 'terraform plan' command. It indicates that Terraform will create an 'azure_network_interface' resource and perform several actions on it, including applying, creating, and deleting. The output also shows the configuration for the 'ip_configuration' resource.



Partie 7 Déploiement de ressource groupe à partir de votre pipeline dans MS Azure

Nous voyons ici la partie du script pour les ressources groups, qui seront déployés en même temps que les autres ressources lorsque je ferai *terraform init*, *plan* et *apply*.

```
9
10
11 provider "azurerm" {
12   features {}
13 }
14
15 resource "azurerm_resource_group" "example" {
16   name       = "cr460-resources"
17   location   = "East US"
18 }
```



Partie 8 Déploiement de Réseau Virtuel à partir de votre pipeline dans MS Azure

Nous voyons ici la partie du script pour le réseau virtuel, qui sera déployé en même temps que les autres ressources lorsque je ferai *terraform init*, *plan* et *apply*.

```
20 resource "azurerm_virtual_network" "example" {
21   name             = "cr460-network"
22   address_space    = ["10.0.0.0/16"]
23   location          = azurerm_resource_group.example.location
24   resource_group_name = azurerm_resource_group.example.name
25 }
26
```

Filter for any field...			Type equals all	Location equals all	Add filter
Showing 1 to 4 of 4 records.			No grouping		
<input type="checkbox"/>	Name ↑		Type ↑		Location ↑
<input type="checkbox"/>	cr460-machine		Virtual machine		East US
<input type="checkbox"/>	cr460-machine_OsDisk_1_79c6eb36417c4f41866e0bd2ace75c8		Disk		East US
<input type="checkbox"/>	cr460-network		Virtual network		East US
<input type="checkbox"/>	cr460-nic		Network interface		East US

Partie 9 Déploiement d'une VM à partir de votre pipeline dans MS Azure

Nous voyons ici la partie du script pour la VM, qui sera déployée en même temps que les autres ressources lorsque je ferai *terraform init, plan et apply*.

```

46 resource "azurerm_windows_virtual_machine" "example" {
47     name                        = "cr460-machine"
48     resource_group_name        = azurerm_resource_group.example.name
49     location                    = azurerm_resource_group.example.location
50     size                        = "Standard_F2"
51     admin_username              = "adminuser"
52     admin_password              = "P@$$w0rd1234!"
53     network_interface_ids = [
54         azurerm_network_interface.example.id,
55     ]
56 }

```

Access control (IAM)
Tags
Resource visualizer
Events
Deployments
Security
Deployment stacks
Policies
Properties
Locks
Management
Cost analysis
Cost alerts (preview)

Filter for any field...

Type equals all

Location equals all

Add filter

Showing 1 to 4 of 4 records. ☐ Show hidden types

<input type="checkbox"/>	Name ↑		Type ↑
<input type="checkbox"/>	cr460-machine		Virtual machine
<input type="checkbox"/>	cr460-machine_OsDisk_1_79c6eb36417c4f41866e0bd2ace75c8		Disk
<input type="checkbox"/>	cr460-network		Virtual network
<input type="checkbox"/>	cr460-nic		Network interface

Après avoir fait *apply*, nous avons tous nos ressources disponibles dans Azure

Microsoft Azure

Home > Resource groups >

Resource groups

Default Directory (fredricperson88@hotmail.com)...

+ Create Manage view Refresh Export to CSV Open query Assign tags Move Delete Export template Open in mobile

Filter for any field...

Name ↑

- cr460-resources
- NetworkWatcherRG

cr460-resources

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Deployments

Security

Deployment stacks

Policies

Properties

Locks

Cost Management

Cost analysis

Cost alerts (preview)

Budgets

Advisor recommendations

Monitoring

Insights (preview)

Alerts

Metrics

Diagnostic settings

Logs

Advisor recommendations

Workbooks

cr460-resources

Subscription (moved) : Azure subscription 1

Subscription ID : b2c3baed-089e-4771-8efa-53c54fd621f

Tags (edit) : Add tags

Deployments : No deployments

Location : East US

Resources

Recommendations

Filter for any field...

Type equals all Location equals all Add filter

Showing 1 to 4 of 4 records. Show hidden types

Name ↑	Type ↑
cr460-machine	Virtual machine
cr460-machine_OsDisk_1_79c5eb36417c4f41866e0fb2aca75cd	Disk
cr460-network	Virtual network
cr460-nic	Network interface

Microsoft Azure

Home >

Resource groups

Default Directory (fredricperson88@hotmail.com)...

+ Create Manage view Refresh Export to CSV Open query Assign tags

Filter for any field...

Subscription equals all Location equals all Add filter

Showing 1 to 2 of 2 records.

Name ↑	Subscription ↑	Location ↑
cr460-resources	Azure subscription 1	East US
NetworkWatcherRG	Azure subscription 1	East US

Enfin, nous faisons *Destroy* pour détruire nos ressources.

The screenshot shows a VS Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with files like `main.tf`, `terraform.tfstate`, and `test.tf`. The main editor displays a Terraform configuration file `main.tf` with the following content:

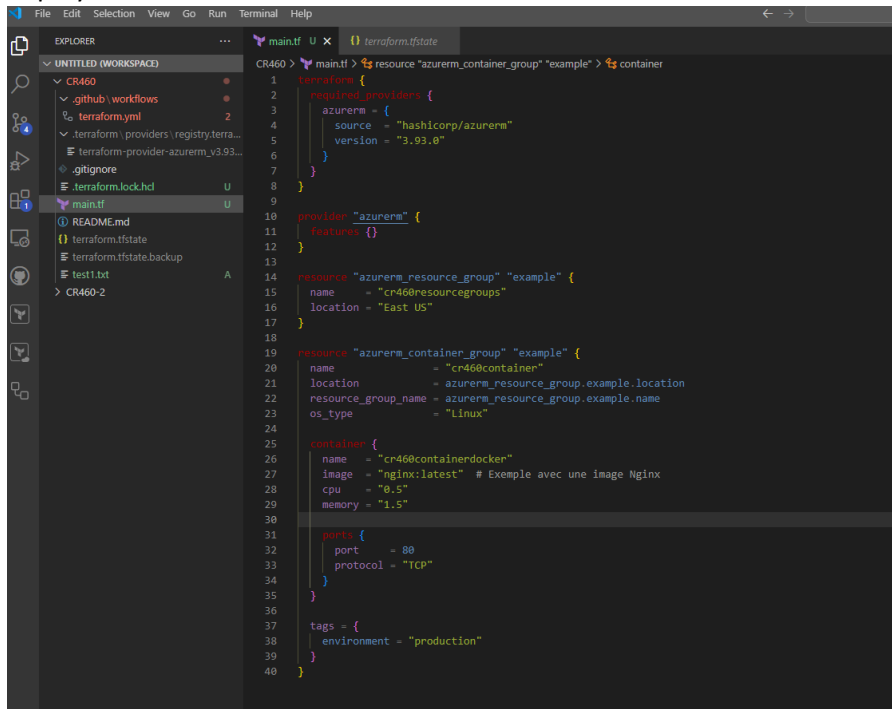
```
55: jobs:
56:   terraform:
57:     name: Checkout
58:
59:     # Install the latest version of Terraform CLI and configure the Terraform CLI configuration file with a Terraform Cloud user API token
60:     name: Setup Terraform
61:     uses: hashicorp/setup-terraform@v1
62:     with:
63:       cli_config_credentials_token: ${{ secrets.TF_API_TOKEN }}
64:
65:     # Initialize a new or existing Terraform working directory by creating initial files, loading any remote state, downloading modules, etc.
66:     name: Terraform Init
67:     run: terraform init
68:
69:     # Checks that all Terraform configuration files adhere to a canonical format
70:     name: Terraform Format
71:     run: terraform fmt -check
72:
73:     # Generates an execution plan for Terraform
74:     name: Terraform Plan
75:     run: terraform plan -input=false
76:
77:     # On push to "main", build or change infrastructure according to Terraform configuration files
78:     # Note: It is recommended to set up a required "strict" status check in your repository for "Terraform Cloud". See the documentation on "strict" required status checks for more information: https://help.github.com/en/github/administering-a-repository
79:     name: Terraform Apply
80:     if: github.ref == 'refs/heads/main' && github.event_name == 'push'
81:     run: terraform apply -auto-approve -input=false
```

The terminal at the bottom shows the output of the Terraform apply command. It starts with a warning about the `strict` status check, followed by a confirmation prompt. The user enters `yes`, and the terminal shows the destruction of various resources, including `azurerm_virtual_machine`, `azurerm_network_interface`, `azurerm_subnet`, `azurerm_virtual_network`, and `azurerm_resource_group`. The final output is:

```
Destroy complete! Resources: 5 destroyed.
PS C:\Users\francesco\Documents\GitHub\demo>
```

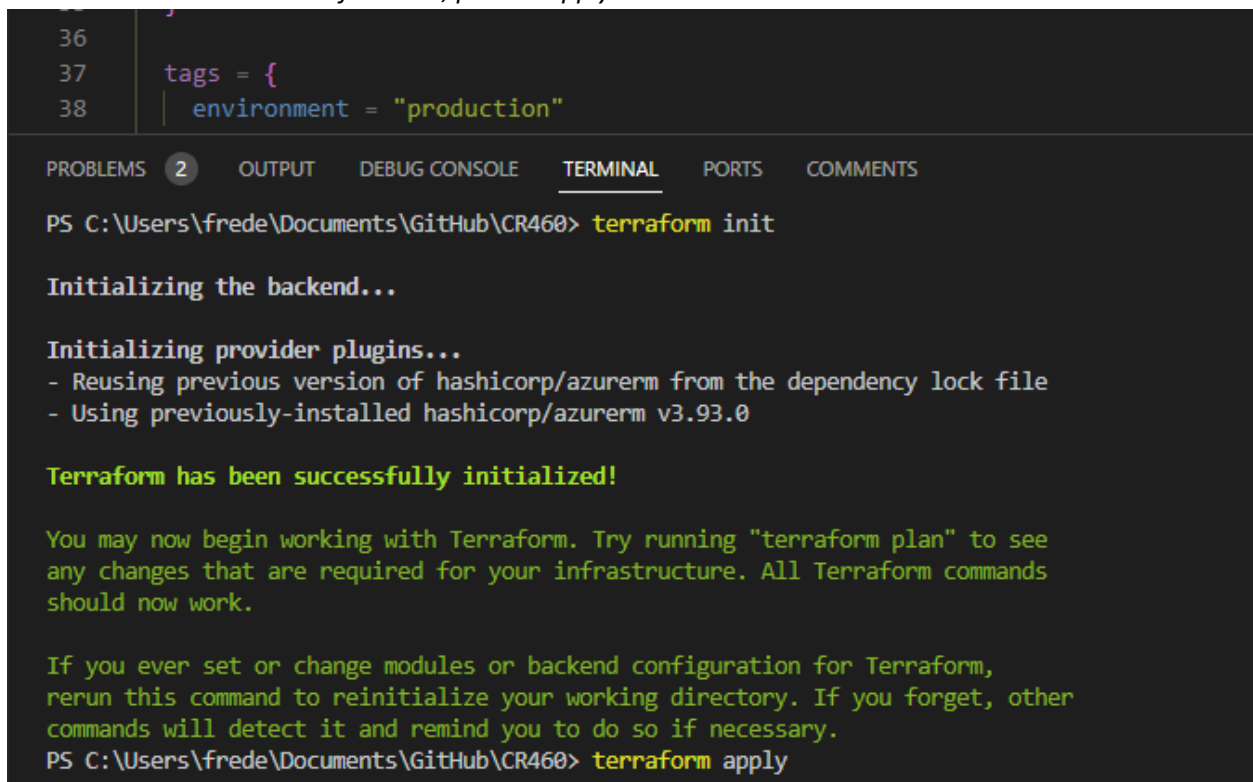
Partie 10 Déploiement d'un container Docker à partir de votre pipeline dans MS Azure

J'ai remplacé la partie des autres ressources, VM, etc. dans mon script par ce script ci-dessous pour déployer mon docker container dans Azure :



```
1  resource "azurerm_container_group" "example" {
2      terraform {
3          required_providers {
4              azurerm = {
5                  source = "hashicorp/azurerm"
6                  version = "3.93.0"
7              }
8          }
9      }
10     provider "azurerm" {
11         features {}
12     }
13
14     resource "azurerm_resource_group" "example" {
15         name     = "cr460resourcegroups"
16         location = "East US"
17     }
18
19     resource "azurerm_container_group" "example" {
20         name                = "cr460container"
21         location            = azurerm_resource_group.example.location
22         resource_group_name = azurerm_resource_group.example.name
23         os_type             = "Linux"
24
25         container {
26             name = "cr460containerdocker"
27             image = "nginx:latest" # Exemple avec une image Nginx
28             cpu   = "0.5"
29             memory = "1.5"
30
31             ports {
32                 port     = 80
33                 protocol = "TCP"
34             }
35         }
36
37         tags = {
38             environment = "production"
39         }
40     }
41 }
```

J'ai ensuite encore fait *terraform init*, *plan* et *apply* et nous avons eu ces résultats-là :



```
36
37     tags = {
38         environment = "production"
39     }
40 }

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

PS C:\Users\frede\Documents\GitHub\CR460> terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/azurerm from the dependency lock file
- Using previously-installed hashicorp/azurerm v3.93.0

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.
PS C:\Users\frede\Documents\GitHub\CR460> terraform apply
```



```
main.tf U x terraform.tfstate
CR460 > main.tf > resource "azurem_container_group" "example" > container > memory

1 terraform {
2   required_providers {
3     azurem = {
4       source = "hashicorp/azurem"
5       version = "3.93.0"
6     }
7   }
8 }
9
10 provider "azurem" {
11   features {}
12 }
13
14 resource "azurem_resource_group" "example" {
15   name     = "cr460resourcegroups"
16   location = "East US"
17 }
18

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

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.
PS C:\Users\Frede\Documents\GitHub\CR460> terraform apply

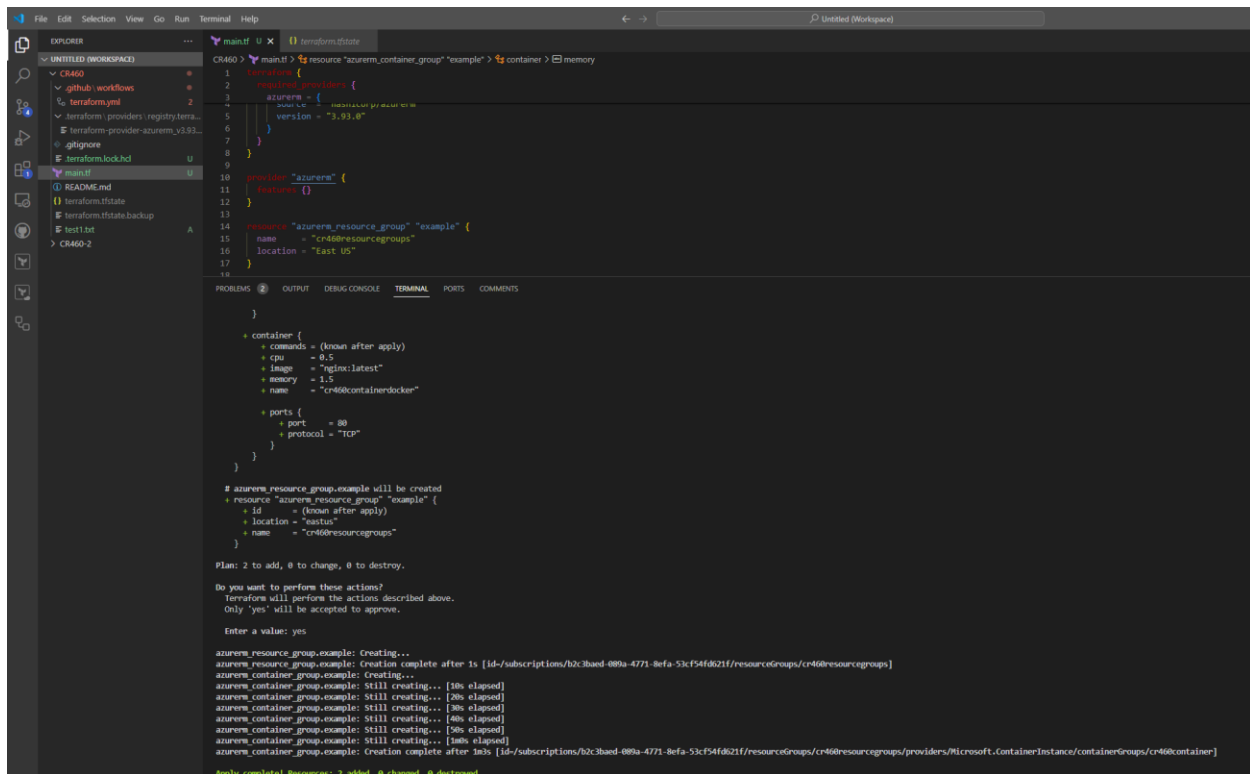
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# azurem_container_group.example will be created
+ resource "azurem_container_group" "example" {
+   dns_name_label_reuse_policy = "Unsecure"
+   exposed_port                = (known after apply)
+   fqdn                       = (known after apply)
+   id                         = (known after apply)
+   ip_address                 = (known after apply)
+   ip_address_type            = "Public"
+   location                   = "eastus"
+   name                       = "cr460container"
+   network_profile_id         = (known after apply)
+   os_type                    = "Linux"
+   resource_group_name        = "cr460resourcegroups"
+   restart_policy              = "Always"
+   sku                        = "Standard"
+   tags                       = {
+     "environment" = "production"
+   }
+ container {
+   + commands = (known after apply)
+   + cpu      = 0.5
+   + image    = "nginx:latest"
+   + memory   = 1.5
+   + name     = "cr460containerdocker"
+   + ports {
+     + port      = 80
+     + protocol = "TCP"
+   }
+ }
}

# azurem_resource_group.example will be created
+ resource "azurem_resource_group" "example" {
+   id       = (known after apply)
+   location = "eastus"
+   name     = "cr460resourcegroups"
+ }

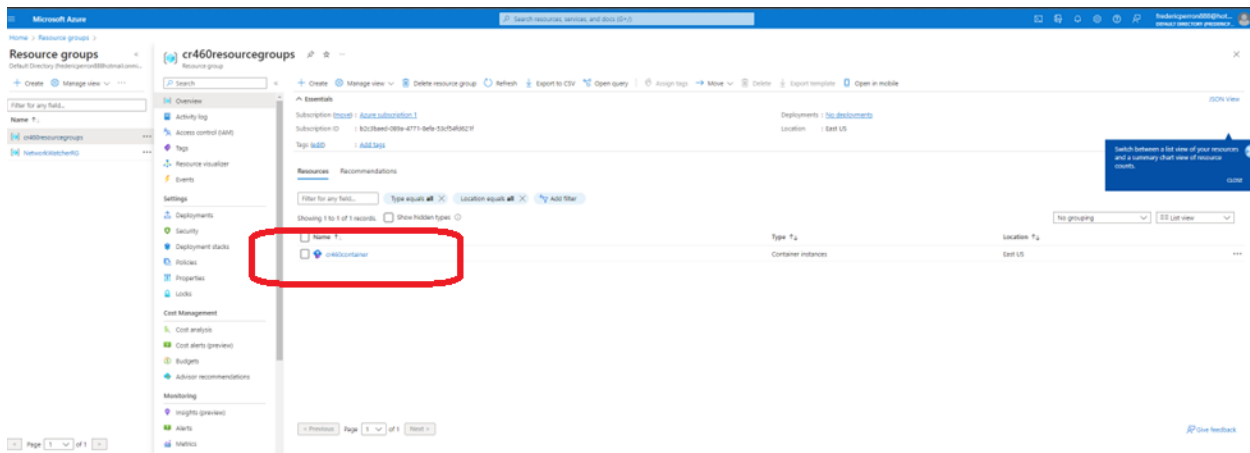
Plan: 2 to add, 0 to change, 0 to destroy.
```



```
CR460 > main.tf > resource "azurerm_container_group" "example" > container > memory

1 terraform {
2   required_providers {
3     azurerm = {
4       source = "hashicorp/azurerm"
5       version = "~>3.93.0"
6     }
7   }
8 }
9
10 provider "azurerm" {
11   features {}
12 }
13
14 resource "azurerm_resource_group" "example" {
15   name     = "cr460resourcegroups"
16   location = "East US"
17 }
18
19 container {
20   commands = [known after apply]
21   cpu       = 0.5
22   image     = "nginx:latest"
23   memory    = 1.5
24   name      = "cr460container"
25
26   ports {
27     port       = 80
28     protocol   = "TCP"
29   }
30 }
31
32 # azurerm_resource_group.example will be created
33 resource "azurerm_resource_group" "example" {
34   id       = [known after apply]
35   location = "eastus"
36   name     = "cr460resourcegroups"
37 }
38
39 Plan: 2 to add, 0 to change, 0 to destroy.
40
41 Do you want to perform these actions?
42 Terraform will perform the actions described above.
43 Only 'yes' will be accepted to approve.
44
45 Enter a value: yes
46
47 azurerm_resource_group.example: Creating...
48 azurerm_resource_group.example: Creation complete after 1s [id=/subscriptions/b2c3baed-089a-4771-befa-53c546d21f/resourceGroups/cr460resourcegroups]
49 azurerm_container_group.example: Creating...
50 azurerm_container_group.example: Still creating... [50s elapsed]
51 azurerm_container_group.example: Still creating... [1m0s elapsed]
52 azurerm_container_group.example: Still creating... [1m10s elapsed]
53 azurerm_container_group.example: Still creating... [1m20s elapsed]
54 azurerm_container_group.example: Still creating... [1m30s elapsed]
55 azurerm_container_group.example: Still creating... [1m40s elapsed]
56 azurerm_container_group.example: Still creating... [1m50s elapsed]
57 azurerm_container_group.example: Still creating... [2m0s elapsed]
58 azurerm_container_group.example: Creation complete after 1m5s [id=/subscriptions/b2c3baed-089a-4771-befa-53c546d21f/resourceGroups/cr460resourcegroups/providers/Microsoft.ContainerInstance/containerGroups/cr460container]
59
60 Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Nous pouvons par la suite voir dans l'interface Azure, que notre container nommé « cr460container » a bel et bien été créé.



Nous faisons ensuite *terraform Destroy* pour faire sûr que tout est bel et bien enlevé

```
PS C:\Users\frede\Documents\GitHub\CR460> terraform destroy
azure_rm_resource_group.example: Refreshing state... [id=/subscriptions/b2c3baed-089a-4771-8efa-53cf54fd621f/resourceGroups/cr460resourcegroups]
azure_rm_container_group.example: Refreshing state... [id=/subscriptions/b2c3baed-089a-4771-8efa-53cf54fd621f/resourceGroups/cr460resourcegroups/providers/Microsoft.ContainerInstance/containerGroups/cr460container]

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

Terraform will perform the following actions:

# azure_rm_container_group.example will be destroyed
- resource "azure_rm_container_group" "example" {
  - dns_name_label_reuse_policy = "Unsecure" -> null
}
```

```
39  }
40  }

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

# azure_rm_resource_group.example will be destroyed
- resource "azure_rm_resource_group" "example" {
  - id      = "/subscriptions/b2c3baed-089a-4771-8efa-53cf54fd621f/resourceGroups/cr460resourcegroups" -> null
  - location = "eastus" -> null
  - name     = "cr460resourcegroups" -> null
  - tags     = {} -> null
}

Plan: 0 to add, 0 to change, 2 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

azure_rm_container_group.example: Destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-53cf54fd621f/resourceGroups/cr460resourcegroups/providers/Microsoft.ContainerInstance/containerGroups/cr460container]
azure_rm_container_group.example: Destruction complete after 2s
azure_rm_resource_group.example: Destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-53cf54fd621f/resourceGroups/cr460resourcegroups]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 10s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 20s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 30s elapsed]
```

```
39  }
40  }

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

- tags     = {} -> null
}

Plan: 0 to add, 0 to change, 2 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

azure_rm_container_group.example: Destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-53cf54fd621f/resourceGroups/cr460resourcegroups/providers/Microsoft.ContainerInstance/containerGroups/cr460container]
azure_rm_container_group.example: Destruction complete after 2s
azure_rm_resource_group.example: Destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-53cf54fd621f/resourceGroups/cr460resourcegroups]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 10s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 20s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 30s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 40s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 50s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 1m0s elapsed]
azure_rm_resource_group.example: Still destroying... [id=/subscriptions/b2c3baed-089a-4771-8efa-...21f/resourceGroups/cr460resourcegroups, 1m10s elapsed]
azure_rm_resource_group.example: Destruction complete after 1m17s

Destroy complete! Resources: 2 destroyed.
PS C:\Users\frede\Documents\GitHub\CR460>
```

Conclusion

En conclusion, ce projet nous a permis d'acquérir une expérience précieuse dans l'installation, la configuration et l'intégration d'outils essentiels pour le développement et le déploiement dans le cloud. Nous avons réussi à installer et configurer avec succès Visual Studio Code, GitHub, Terraform CLI, Terraform Cloud et à les intégrer efficacement avec Microsoft Azure. De plus, nous avons réalisé plusieurs déploiements de ressources Azure, y compris des groupes de ressources, des réseaux virtuels, des machines virtuelles et des conteneurs Docker, en utilisant un pipeline de déploiement continu. Ce projet nous a permis de développer nos compétences techniques et notre compréhension des bonnes pratiques en matière de gestion d'infrastructure et de déploiement dans le cloud.

Références

- https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/windows_virtual_machine.html
- Cours Audio/Visuel #2,3,4,5,6 CR460 et leurs PDF respectifs