

Research-Project

Gebruikershandleiding Lagast Sean

Contents

Inleiding	3
Start gebruik	3
1: Opbouw applicatie	3
1.1: Setupterraform.sh	3
1.2: CSVFiles	3
1.3: Replace-varstf.ps1	5
1.4: Deleteter.sh	5
1.5: Overige mappen(Web,VM,Infrastructure)	5
2: Start script	6
2.1: Keuze omgeving:	6
2.2: Confirmatie	6
2.3 CSV kiezen	6
2.4: Tijdsduur.....	7
2.5: Resultaat.....	8
3: Aanpasbare factoren	9
3.1: Terraform omgevingen.....	9
3.2: CSV bestanden.....	11
3.3: Tijdsduur.....	13
4: Belangrijke elementen.....	17
4.1: Vars.tf files.....	17
4.2: Terraform files	19
4.3: CSVFiles directory	21
4.4: Setupterraform.sh	21
4.5: Deleteter.sh	21
4.6: Replace-varstf.ps1	21
4.7: Nieuwe environment toevoegen.....	21

Inleiding

Na de opstelling van het project kan er gestart worden met het gebruik. Hier wordt uitgelegd hoe je van start kunt gaan met de omgeving die er al is. Later wordt ook omschreven welke stukken van het project je kunt gaan aanpassen om het jezelf later makkelijker te maken.

Start gebruik

De applicatie die jij als user gebruikt is eigenlijk enkel een simpel bash script waar je een prompt krijgt welke gegevens je wilt gaan gebruiken.

Eerst even een woordje uitleg wat er nu bedoeld wordt met omgevingen. Indien je met dit project aan de slag gaat heb je waarschijnlijk wel al enige voorkennis van het gebruik van terraform en hoe je een terraform file kunt opzetten.

1: Opbouw applicatie

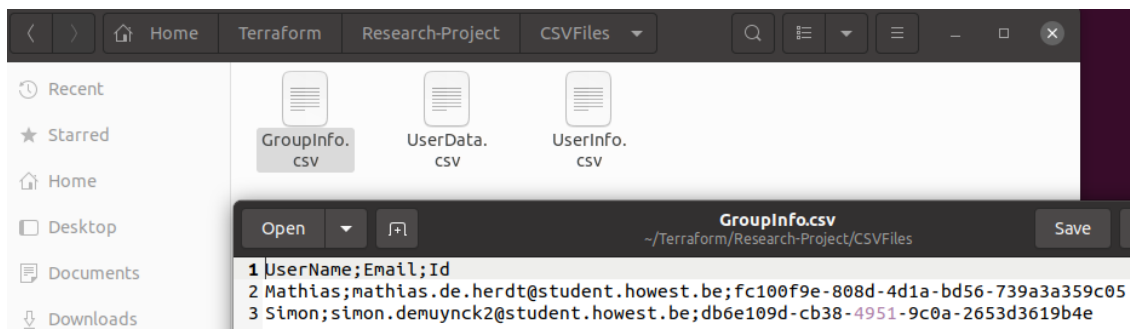
De gedownloade git repo bevat dus een aantal elementen. Deze worden nu elk uitgelegd.

1.1: Setupterraform.sh

Dit script is de 'kern' van alles. Jij als gebruiker hoeft enkel dit uit te voeren om de juiste omgeving op te zetten. De andere delen in de repo zijn componenten die aan dit script vast hangen om meerdere keuzes te kunnen maken.

1.2: CSVFiles

Tijdens het uitvoeren van het Setup script word je gevraagd welke csv file je wilt gaan gebruiken. Deze csv files die je wilt kunnen toepassen hoor je onder deze CSVFiles directory te zetten. Nu wat meer info over hoe je de csv file moet opbouwen.



De csv file hoort altijd volgens deze opbouw gemaakt te worden. Lijn 1 moet hetzelfde zijn als lijn 1 in de foto hierboven. Elke lijn die volgt stelt lijn 1 voor met unieke waarden.

UserName;Email;Id

UserName -> De naam van de gebruikers(mag zelf bepaald worden) -> Aangeraden om alles aan mekaar te schrijven

Email -> De email van de gebruikers

Id -> (BELANGRIJK!) Het ID van de gebruiker in azure

Ook belangrijk is dat je steeds ; gebruikt om UserName, Email en Id te onderscheiden.

De id's van de gebruikers vind je door het volgende te gaan doen.
In de azure portal navigeer je naar Access control.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with the Microsoft Azure logo and a search bar. Below the navigation bar, the breadcrumb trail reads 'Home > Azure for Students'. The main heading is 'Azure for Students | Access control (IAM)'. On the left, there's a sidebar with navigation links: Overview, Activity log, Access control (IAM) (selected), Tags, Diagnose and solve problems, Security, Events, and Billing. The main content area has a search bar and several action buttons: Add, Download role assignments, Edit columns, Refresh, Remove, and Got feedback?. Below these, there are tabs: Check access (selected), Role assignments, Roles, Deny assignments, and Classic administrators. The 'Check access' section shows 'My access' with a 'View my access' button and 'Check access' with a description and a 'Learn more' link. On the right, there's a 'Grant access to this resource' section with a description and an 'Add role assignment' button.

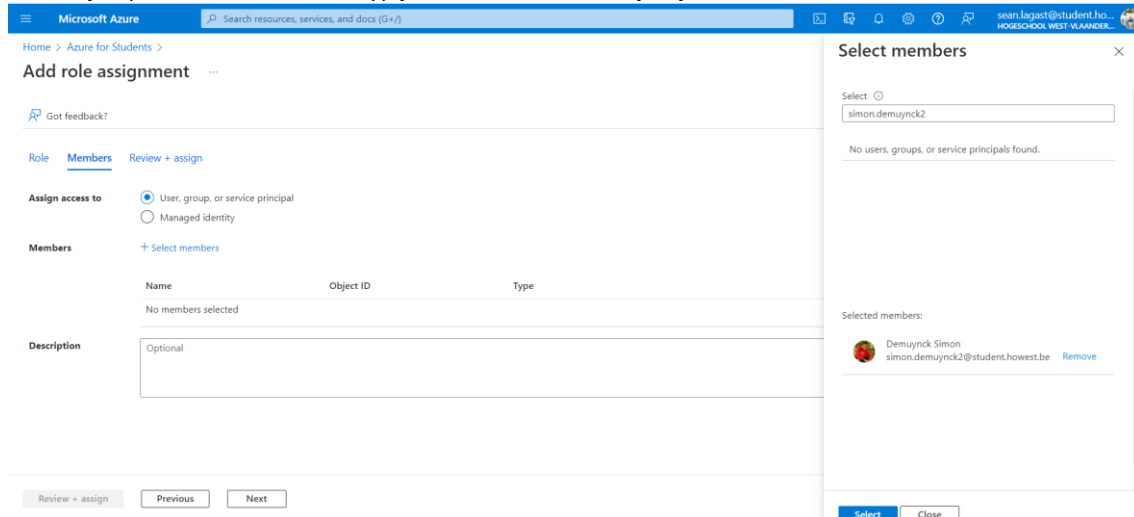
Daarna klik je op Add role assignment.

The screenshot shows the 'Add role assignment' page in the Microsoft Azure portal. The navigation bar and breadcrumb trail are the same as in the previous screenshot. The main heading is 'Add role assignment'. Below the heading, there's a 'Got feedback?' link. The page has three tabs: Role (selected), Members, and Review + assign. The main content area explains that a role definition is a collection of permissions and provides a 'Learn more' link. Below this, there's a search bar and two filters: Type: All and Category: All. A table with two columns, Name and Description, lists the available roles. The first row shows the 'Owner' role, which grants full access to manage all resources, including the ability to assign roles in Azure RBAC.

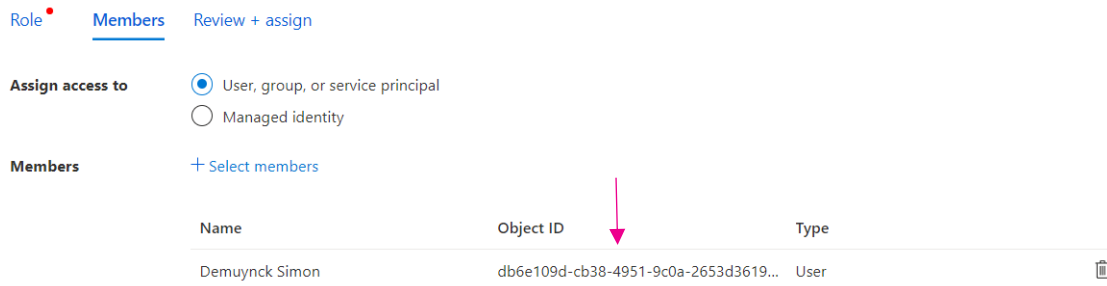
Daarna klik je op Members.

The screenshot shows the 'Members' tab in the 'Add role assignment' page. The navigation bar and breadcrumb trail are the same. The main heading is 'Add role assignment'. Below the heading, there's a 'Got feedback?' link. The page has three tabs: Role, Members (selected), and Review + assign. The main content area has an 'Assign access to' section with two radio buttons: 'User, group, or service principal' (selected) and 'Managed identity'. Below this, there's a 'Members' section with a '+ Select members' button.

Nu klik je op select members en typ je de user in waarvan je zijn id wilt weten.



Klik dan op select en je krijgt het id te zien van de user die je wilt.



Omdat ikzelf geen admin ben in mijn school omgeving, is dit de eenvoudigste manier hoe ik aan de id's kan komen van klasgenoten.

Indien een admin nieuwe users aanmaakt kan deze op een veel snellere manier alle members zien, en dus ook hun Object Id raadplegen.

1.3: Replace-varstf.ps1

Wanneer je het setup script start, wordt ergens in deze code het powershell script gestart. Wat deze gaat doen is alle variabelen in de door jouw geselecteerde csv file, gaan toepassen op de ook door jouw geselecteerde omgeving. In het technisch document wordt elk stukje code nog eens in detail uitgelegd.

1.4: Deleteter.sh

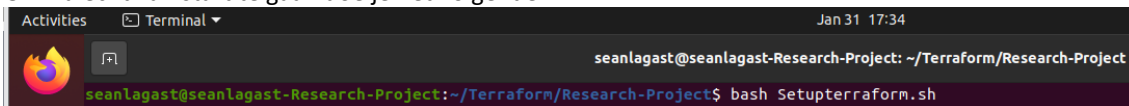
Het Deleteter.sh script wordt uitgevoerd in cron. Wanneer je door het setup script loopt is er een moment dat je moet kiezen hoelang je je omgeving wilt laten bestaan. De reden hierachter is kostenbesparing. Wanneer je keuze gemaakt is wordt er in crontab een lijn code toegevoegd dat dan op het gekozen tijdstip het Deleteter.sh script zal laten uitvoeren.

1.5: Overige mappen(Web,VM,Infrastructure)

Deze 3 mappen zijn voorbeelden van hoe je een terraform omgeving er laat uit zien. In zo'n map zit een vars.tf file die de variabelen van je users bijhoudt. De andere file is je terraform file zelf, die de opbouw doet van je omgeving. Later wordt uitgelegd wat de key componenten zijn van je terraform file.

2: Start script

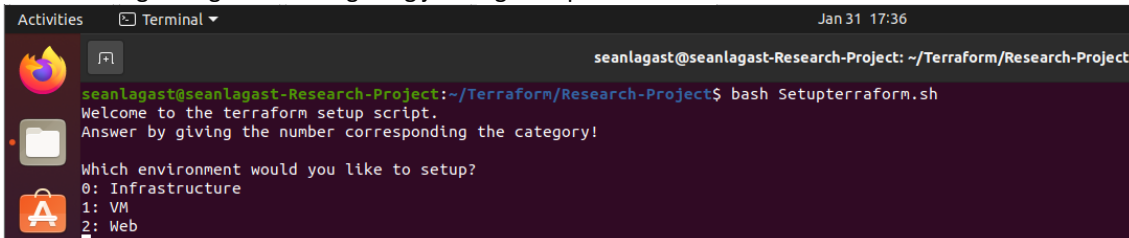
Om nu echt van start te gaan doe je het volgende.



```
Activities Terminal Jan 31 17:34
seanlagast@seanlagast-Research-Project: ~/Terraform/Research-Project
seanlagast@seanlagast-Research-Project:~/Terraform/Research-Project$ bash Setupterraform.sh
```

2.1: Keuze omgeving:

Eerst wordt gevraagd welke omgeving je wilt gaan opzetten.



```
Activities Terminal Jan 31 17:36
seanlagast@seanlagast-Research-Project: ~/Terraform/Research-Project
seanlagast@seanlagast-Research-Project:~/Terraform/Research-Project$ bash Setupterraform.sh
Welcome to the terraform setup script.
Answer by giving the number corresponding the category!

Which environment would you like to setup?
0: Infrastructure
1: VM
2: Web
```

Geef telkens het nummer in dat van toepassing is. Zoals daarnet besproken zijn deze 3 keuzes, test keuzes. Het is natuurlijk de bedoeling dat jij als gebruiker jouw eigen omgevingen zult gebruiken in dit script, straks wordt aangetoond hoe.

2.2: Confirmatie

Dit is een tussenstap of je zeker bent met de gekozen omgeving.



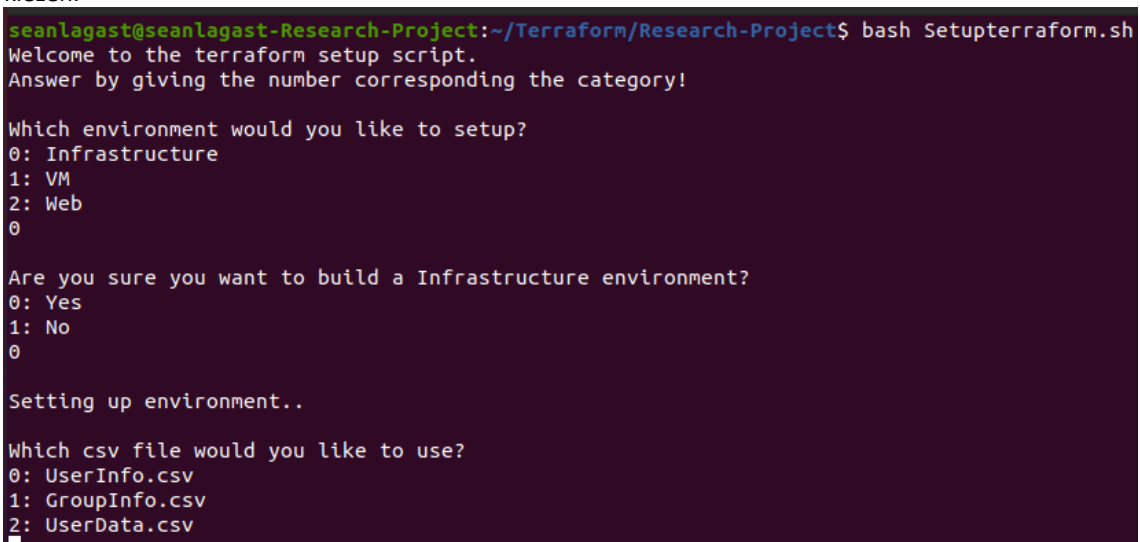
```
seanlagast@seanlagast-Research-Project:~/Terraform/Research-Project$ bash Setupterraform.sh
Welcome to the terraform setup script.
Answer by giving the number corresponding the category!

Which environment would you like to setup?
0: Infrastructure
1: VM
2: Web
0

Are you sure you want to build a Infrastructure environment?
0: Yes
1: No
```

2.3 CSV kiezen

In de volgende stap kies je voor welke users je een omgeving wilt opzetten, door de juiste csv file te kiezen.



```
seanlagast@seanlagast-Research-Project:~/Terraform/Research-Project$ bash Setupterraform.sh
Welcome to the terraform setup script.
Answer by giving the number corresponding the category!

Which environment would you like to setup?
0: Infrastructure
1: VM
2: Web
0

Are you sure you want to build a Infrastructure environment?
0: Yes
1: No
0

Setting up environment..

Which csv file would you like to use?
0: UserInfo.csv
1: GroupInfo.csv
2: UserData.csv
```

2.4: Tijdsduur

Ten slotte kies je hoelang je omgeving moet bestaan.

```
seanlagast@seanlagast-Research-Project:~/Terraform/Research-Project$ bash Setupterraform.sh
Welcome to the terraform setup script.
Answer by giving the number corresponding the category!

Which environment would you like to setup?
0: Infrastructure
1: VM
2: Web
0

Are you sure you want to build a Infrastructure environment?
0: Yes
1: No
0

Setting up environment..

Which csv file would you like to use?
0: UserInfo.csv
1: GroupInfo.csv
2: UserData.csv
1

How long do you want the environment to exist?
0: Forever
1: 2 hours
2: 1 day
3: 1 week
4: Fill in own crontab syntax
```

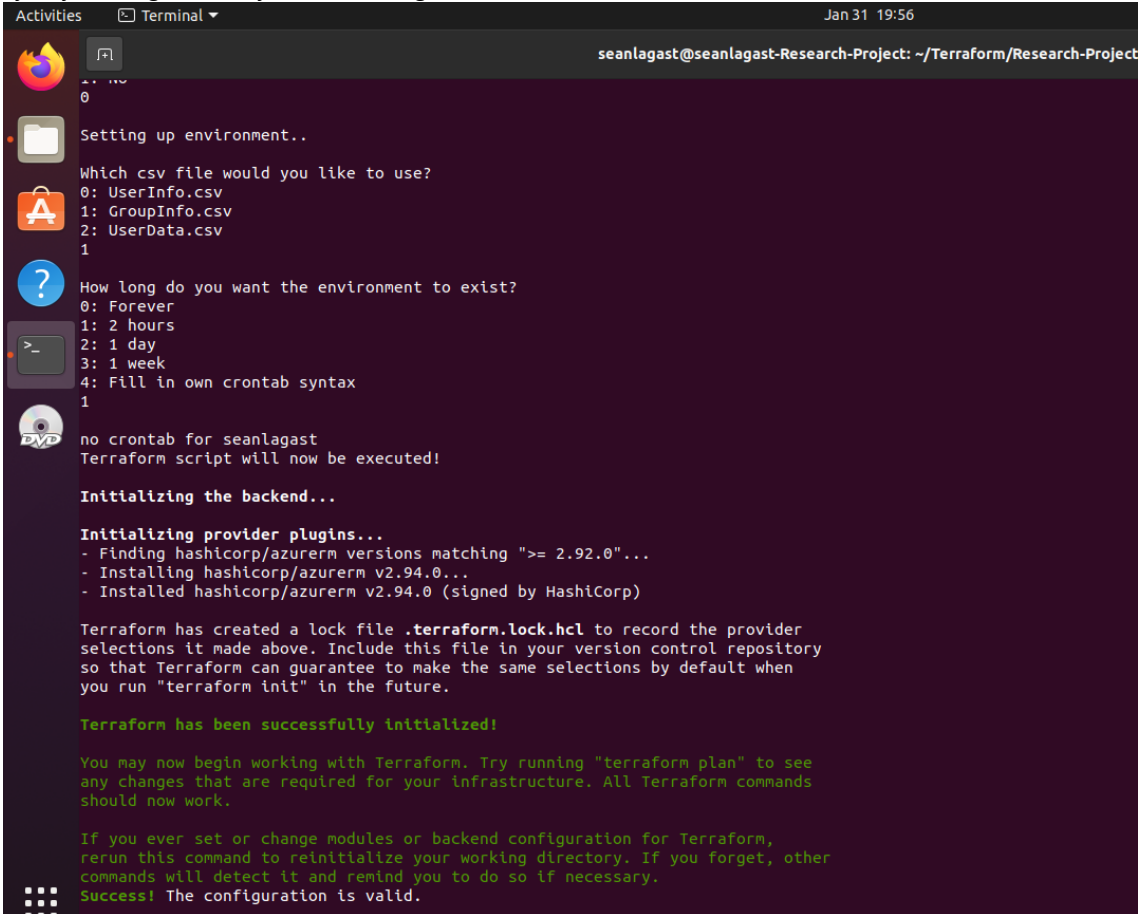
Dit zijn opnieuw keuzes die door mij gemaakt zijn, straks toon ik aan hoe je dit zelf kunt aanpassen.

Ook kun je ervoor kiezen om zelf een cron syntax te gaan gebruiken. Hiervoor raad ik volgende link aan:

<https://crontab.guru/>

2.5: Resultaat

Indien alles goed verloopt aan de kant van azure, en indien de eigen terraform scripts in het algemeen werken zouden de omgevingen zichtbaar moeten zijn op de azure portal. Ook komt er in crontab -e een lijn bij die de gekozen tijdsduur zal volgen.



```
Activities Terminal Jan 31 19:56
seanlagast@seanlagast-Research-Project: ~/Terraform/Research-Project

Setting up environment..
Which csv file would you like to use?
0: UserInfo.csv
1: GroupInfo.csv
2: UserData.csv
1

How long do you want the environment to exist?
0: Forever
1: 2 hours
2: 1 day
3: 1 week
4: Fill in own crontab syntax
1

no crontab for seanlagast
Terraform script will now be executed!

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching ">= 2.92.0"...
- Installing hashicorp/azurerm v2.94.0...
- Installed hashicorp/azurerm v2.94.0 (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.

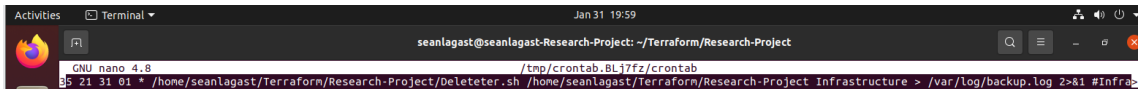
Terraform has been successfully initialized!

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.
Success! The configuration is valid.
```

<input type="checkbox"/> Name ↑↓	Subscription ↑↓
<input type="checkbox"/> cloud-shell-storage-west europe	Azure for Students
<input type="checkbox"/> DefaultResourceGroup-WEU	Azure for Students
<input type="checkbox"/> Mathias-Infrastructure	Azure for Students
<input type="checkbox"/> NetworkWatcherRG	Azure for Students
<input type="checkbox"/> Simon-Infrastructure	Azure for Students

Zoals je ziet is er een Infrastructure omgeving opgebouwd, met de users uit het GroupInfo.csv bestand.



```
Activities Terminal Jan 31 19:59
seanlagast@seanlagast-Research-Project: ~/Terraform/Research-Project

GNU nano 4.8 /tmp/crontab.BLj7fz/crontab
# 21 31 01 * /home/seanlagast/Terraform/Research-Project/Deleteter.sh /home/seanlagast/Terraform/Research-Project Infrastructure > /var/log/backup.log 2>&1 #Infro
```

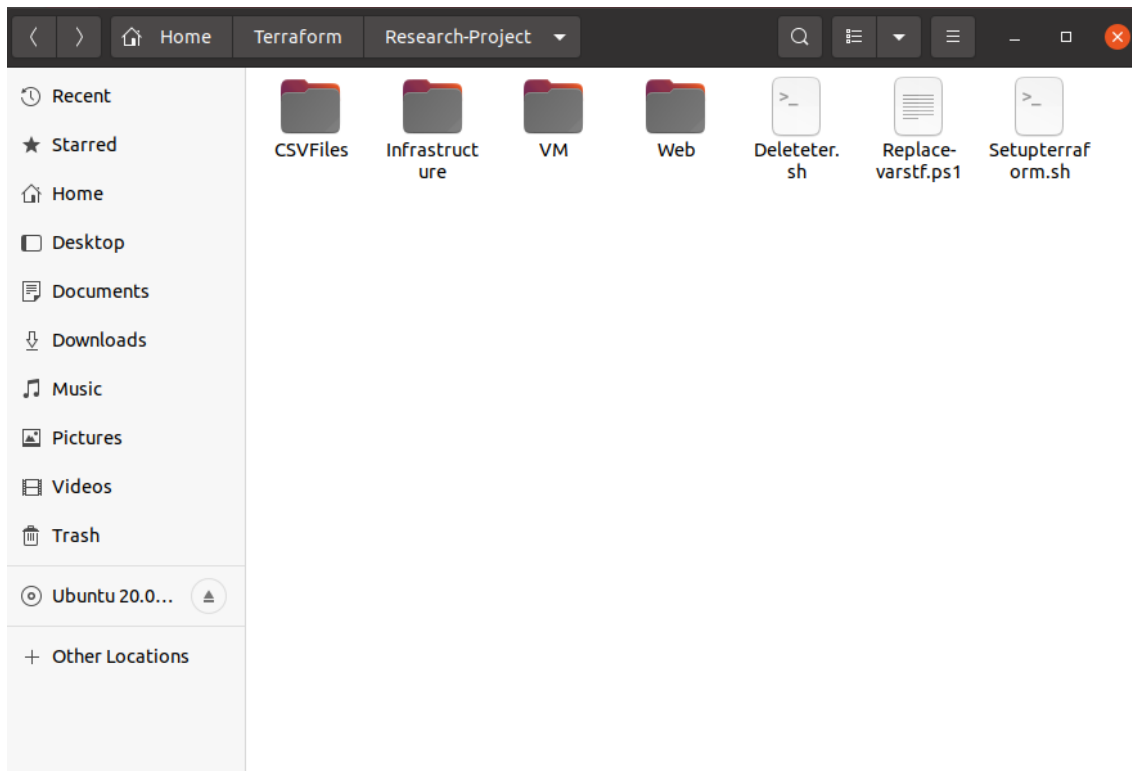
Ook is er in crontab een lijn aangemaakt die deze omgeving 2u laat bestaan.

3: Aanpasbare factoren

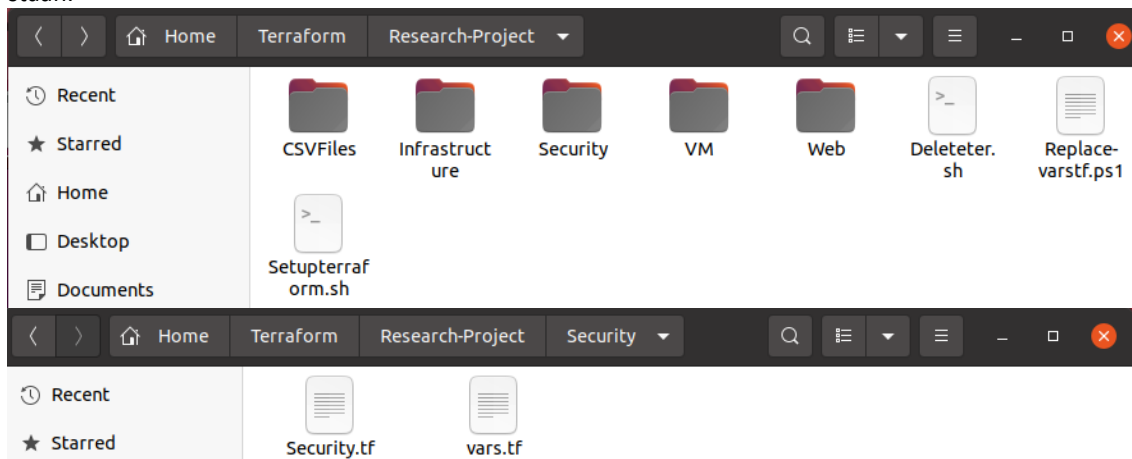
Zoals eerder besproken kun je heel wat zaken gaan personaliseren. Hier worden alle stukken die je kunt aanpassen opgesomd.

3.1: Terraform omgevingen

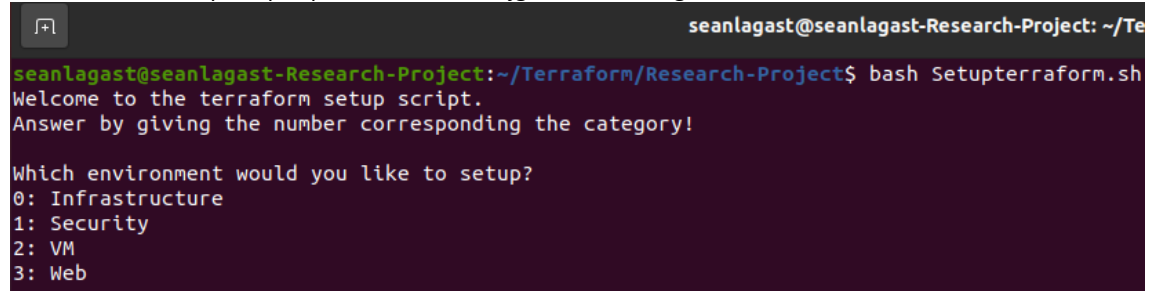
Het allerbelangrijkste is dat je natuurlijk je eigen gemaakte omgevingen kunt gaan gebruiken, hiervoor doe je het volgende.



Maak een nieuwe directory aan, en steek hier zowel je terraform script als je vars.tf file in. Hier een voorbeeld met security. Je kunt als je wilt de andere mappen verwijderen, maar laat CSVFiles zeker staan.



Als we nu het setup script opnieuw runnen krijgen we het volgende te zien.



```
seanlagast@seanlagast-Research-Project: ~/Terraform/Research-Project$ bash Setupterraform.sh
Welcome to the terraform setup script.
Answer by giving the number corresponding the category!

Which environment would you like to setup?
0: Infrastructure
1: Security
2: VM
3: Web
```

De omgeving is er dus bijgekomen. In een volgend punt haal ik enkel nog de belangrijke zaken aan die je terraform script en je vars.tf file moeten bevatten om je users dynamisch toe te kunnen voegen.

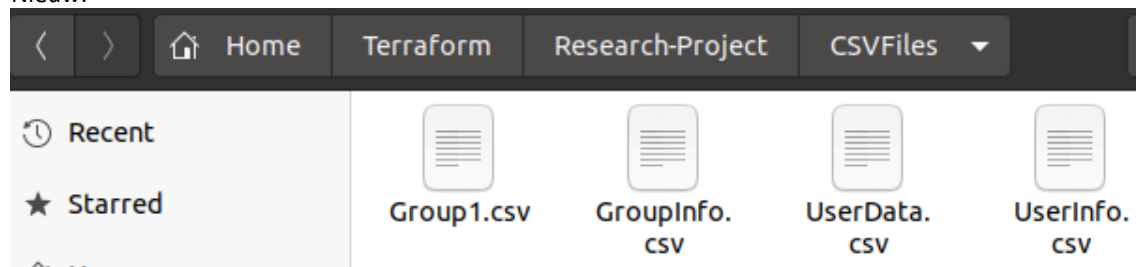
3.2: CSV bestanden

Ook csv bestanden kunnen worden toegevoegd, de opbouw is reeds gekend. Terug een voorbeeld van hoe je dit kunt doen.

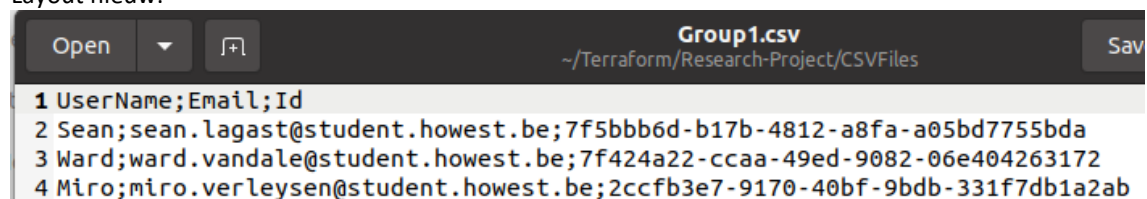
Huidig:



Nieuw:



Layout nieuw:



Opnieuw runnen we eens het setup script.

```
seanlagast@seanlagast-Research-Project:~/Terraform/Research-Project$ bash Setupterraform.sh
Welcome to the terraform setup script.
Answer by giving the number corresponding the category!

Which environment would you like to setup?
0: Infrastructure
1: Security
2: VM
3: Web
1

Are you sure you want to build a Security environment?
0: Yes
1: No
0

Setting up environment..

Which csv file would you like to use?
0: UserInfo.csv
1: GroupInfo.csv
2: Group1.csv
3: UserData.csv
```

De nieuw aangemaakte csv is nu ook beschikbaar.

3.3: Tijdsduur

Ten slotte kan de tijdsduur aangepast worden. Dit is een klein beetje moeilijker om toe te passen, maar daarom is dit document er ook.

Dit deel zit ingebakken in het setup script. Om dit dus aan te passen zullen we wat code moeten aanpassen.

Het deel waar je naar opzoek bent is vanaf lijn 82:

```
82 echo "How long do you want the environment to exist?"
83 echo "0: Forever"
84 echo "1: 2 hours"
85 echo "2: 1 day"
86 echo "3: 1 week"
87 echo "4: Fill in own crontab syntax"
88 read duration
89 echo ""
90 if ! [[ "$duration" =~ ^[0-9]+$ ]]
91 then
92     echo "Sorry integers only!"
93 else
94     if [ "$duration" -eq "0" ]
95     then
96         :
97     elif [ "$duration" -eq "1" ]
98     then
99         newDate=$(date +"%Y-%m-%d %T" -d "+2 hours")
100        cronSyntax="{newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
101    elif [ "$duration" -eq "2" ]
102    then
103        newDate=$(date +"%Y-%m-%d %T" -d "+1 day")
104        cronSyntax="{newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
105    elif [ "$duration" -eq "3" ]
106    then
```

Lijnen 82-87 zijn de keuzes die je kunt maken. Indien je meer keuzes wilt kun je dus een extra lijn toevoegen tussen deze met de tijd die je wilt. Bijvoorbeeld:

```
82 echo "How long do you want the environment to exist?"
83 echo "0: Forever"
84 echo "1: 30 minutes"
85 echo "2: 2 hours"
86 echo "3: 1 day"
87 echo "4: 1 week"
88 echo "5: Fill in own crontab syntax"
```

Op lijn 84 is de optie bijgekomen van 30 minuten. Natuurlijk pas je ook de rest van de volgende getallen aan in de rij.

Nu je deze optie hebt toegevoegd moet er nog 1 deel code bijkomen en een aantal delen worden aangepast. Het deel dat erbij komt ziet er zo uit.

```
elif [ "$duration" -eq "1" ]
then
    newDate=$(date +"%Y-%m-%d %T" -d "+30 minutes")
    cronSyntax="{newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *
```

De twee delen die aangepast moeten worden zijn degene aangegeven met de pijlen.

De roze pijl geeft aan welke keuze in de rij van keuzes dit stuk zich bevindt, wat in de foto boven deze dus op plaats 1 is. Het tweede deel bij de gele pijl gaat over de effectieve tijdsduur, aangezien we in de keuzelijst 30 minutes kozen voegen we dit hier ook toe na het + teken.

Dit deel kunnen we nu toevoegen in de huidige code.

```
echo "How long do you want the environment to exist?"
echo "0: Forever"
echo "1: 30 minutes"
echo "2: 2 hours"
echo "3: 1 day"
echo "4: 1 week"
echo "5: Fill in own crontab syntax"
read duration
echo ""
if ! [[ "$duration" =~ ^[0-9]+$ ]]
then
    echo "Sorry integers only!"
else
    if [ "$duration" -eq "0" ]
    then
        :
    elif [ "$duration" -eq "1" ]
    then
        newDate=$(date +"%Y-%m-%d %T" -d "+30 minutes")
        cronSyntax="${newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
    elif [ "$duration" -eq "1" ]
    then
        newDate=$(date +"%Y-%m-%d %T" -d "+2 hours")
        cronSyntax="${newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
    elif [ "$duration" -eq "2" ]
    then
        newDate=$(date +"%Y-%m-%d %T" -d "+1 day")
        cronSyntax="${newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
    elif [ "$duration" -eq "3" ]
    then
        newDate=$(date +"%Y-%m-%d %T" -d "+1 week")
        cronSyntax="${newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
    elif [ "$duration" -eq "4" ]
    then
        echo "Cron syntax must be exactly like this one between brackets, don't add the brackets"
        echo ""
        echo "(* * * * *)"
        echo ""
    fi
fi
```

Het deel tussen de lijnen is toegevoegd. Het laatste wat nu moet gebeuren is het aanpassen van de andere keuzes zodat hun corresponderend nummer klopt, deze staan aangegeven met een gele pijl.

Deze zullen er nu zo uit zien:

```
else
  if [ "$duration" -eq "0" ]
  then
    :
  elif [ "$duration" -eq "1" ]
  then
    newDate=$(date +"%Y-%m-%d %T" -d "+30 minutes")
    cronSyntax="{newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
  elif [ "$duration" -eq "2" ]
  then
    newDate=$(date +"%Y-%m-%d %T" -d "+2 hours")
    cronSyntax="{newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
  elif [ "$duration" -eq "3" ]
  then
    newDate=$(date +"%Y-%m-%d %T" -d "+1 day")
    cronSyntax="{newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
  elif [ "$duration" -eq "4" ]
  then
    newDate=$(date +"%Y-%m-%d %T" -d "+1 week")
    cronSyntax="{newDate:14:2} ${newDate:11:2} ${newDate:8:2} ${newDate:5:2} *"
  elif [ "$duration" -eq "5" ]
  then
    echo "Cron syntax must be exactly like this one between brackets, don't add the brackets"
    echo ""
    echo "(* * * * *)"
    echo ""
    echo "For more info check: https://crontab.guru/"
    echo ""
    echo "Fill in your cron syntax:"
    read cronSyntax
```

Wanneer dit aangepast is, is alles afgerond.

Nu zal ik het script eens runnen met alle nieuwe toegevoegde waarden:

```
seanlagast@seanlagast-Research-Project:~/Terraform/Research-Project$ bash Setupterraform.sh
Welcome to the terraform setup script.
Answer by giving the number corresponding the category!

Which environment would you like to setup?
0: Infrastructure
1: Security
2: VM
3: Web
1

Are you sure you want to build a Security environment?
0: Yes
1: No
0









Setting up environment..

Which csv file would you like to use?
0: UserInfo.csv
1: GroupInfo.csv
2: Group1.csv
3: UserData.csv
2

How long do you want the environment to exist?
0: Forever
1: 30 minutes
2: 2 hours
3: 1 day
4: 1 week
5: Fill in own crontab syntax
1




Terraform script will now be executed!
```

Het eindresultaat van alles ziet er als volgt uit:

- ☐  cloud-shell-storage-westeurope
- ☐  DefaultResourceGroup-WEU
- ☐  Mathias-Infrastructure
- ☐  Miro-Security
- ☐  NetworkWatcherRG
- ☐  Sean-Security
- ☐  Simon-Infrastructure
- ☐  Ward-Security

```
seanlagast@seanlagast-Research-Project: ~/Terraform/Research-Project
GNU nano 4.8 /tmp/crontab.RwgB3G/crontab
# 21 31 01 * /home/seanlagast/Terraform/Research-Project/Deleteter.sh /home/seanlagast/Terraform/Research-Project Infrastructure > /var/log/backup.log 2>&1 #Infra
# 16 21 31 01 * /home/seanlagast/Terraform/Research-Project/Deleteter.sh /home/seanlagast/Terraform/Research-Project Security > /var/log/backup.log 2>&1 #Security
```

Zowel in azure als in crontab is de verwachte configuratie gebeurt. Om het nu wat te versnellen zal ik beide tijdstippen handmatig veranderen naar een tijdstip dicht bij het mijne om de verwijdering eens te zien.

<input type="checkbox"/> Name ↑↓	Subscription ↑↓	Location ↑↓	
<input type="checkbox"/>  cloud-shell-storage-westeurope	Azure for Students	West Europe	...
<input type="checkbox"/>  DefaultResourceGroup-WEU	Azure for Students	West Europe	...
<input type="checkbox"/>  NetworkWatcherRG	Azure for Students	West Europe	...

```
seanlagast@seanlagast-Research-Project: ~/Terraform/Research-Project
GNU nano 4.8 /tmp/crontab.gyuw0B/crontab
```

Zoals je ziet is alles verwijderd.

4: Belangrijke elementen

Om de applicatie goed te laten werken zijn er een paar zaken waar je op moet letten. Deze worden hier opgesomd.

4.1: Vars.tf files

We zullen beginnen met wat er specifiek noodzakelijk is om het terraform script dynamisch te laten werken.

```
1 #-----
2 #User-Information
3 variable "Amount" {
4     default = 0 #Amount
5     type = number
6 }
7
8 variable "Environment" {
9     default = "" #Environment
10    type = string
11 }
12
13 variable "Users_Email" {
14     type = list
15     default = [] #Email
16 }
17
18 variable "Users_Name" {
19     type = list
20     default = [] #UserName
21 }
22
23 variable "Users_Id" {
24     type = list
25     default = [] #Id
26 }
27 #-----
```

Dit deel code hoort zeker in je vars.tf file te zitten, dit zorgt voor het dynamisch gebeuren van je script. Let ook op dat de # na alle default waarden zeker blijft staan, anders zal bij het uitvoeren van het setup script niets in de vars.tf file aangepast worden.

Als het setup script nu 1x gerund wordt in deze omgeving zullen de default waarden aangepast worden met de gegevens van de 'cursisten'.

```

30
31 #-----
32 #Terraform specific info
33 variable "RG_Location" {
34     default = "westeurope"
35 }
36
37 variable "SP_Name" {
38     default = "WebAppServicePlan"
39 }
40
41 variable "SP_tier" {
42     default = "Free"
43 }
44
45 variable "SP_size" {
46     default = "F1"
47 }
48
49 variable "AS_bootstrap" {
50     default = "https://github.com/Azure-Samples/nodejs-docs-hello-world"
51 }
52
53 variable "AS_bootstrap_branch" {
54     default = "master"
55 }
56 #-----

```

Ook dit deel staat nog in mijn vars.tf file. Dit zorgt voor de resource configuratie van bepaalde zaken, dit is bij iedereen uniek en hoeft dus niet perse in je vars.tf file te zitten.

4.2: Terraform files

```
1 terraform {
2   required_providers {
3     azurerm = {
4       source = "hashicorp/azurerm"
5       version = ">= 2.92.0"
6     }
7   }
8 }
9
10 provider "azurerm" {
11   features {}
12 }
13
14 resource "azurerm_resource_group" "Server-RG" {
15   count = var.Amount
16   name = "${element(var.Users_Name, count.index)}-${var.Environment}"
17   location = var.RG_Location
18
19   tags = {
20     "Student-email" = "${var.Users_Email[count.index]}",
21     "Student-name" = "${var.Users_Name[count.index]}"
22   }
23 }
24
25 resource "azurerm_app_service_plan" "Server-SP" {
26   count = var.Amount
27   name = var.SP_Name
28   location = azurerm_resource_group.Server-RG[count.index].location
29   resource_group_name = azurerm_resource_group.Server-RG[count.index].name
30
31   tags = {
32     "Student-email" = "${var.Users_Email[count.index]}",
33     "Student-name" = "${var.Users_Name[count.index]}"
34   }
35
36   sku {
37     tier = var.SP_tier
38     size = var.SP_size
39   }
40
41   depends_on = [
42     azurerm_resource_group.Server-RG,
43   ]
44 }
45
46 resource "azurerm_app_service" "Server-AS0" {
47   count = var.Amount
48   name = "${var.Environment}-${var.Users_Name[count.index]}-0"
49   location = azurerm_resource_group.Server-RG[count.index].location
50   resource_group_name = azurerm_resource_group.Server-RG[count.index].name
51   app_service_plan_id = azurerm_app_service_plan.Server-SP[count.index].id
52
53   tags = {
54     "Student-email" = "${var.Users_Email[count.index]}",
55     "Student-name" = "${var.Users_Name[count.index]}"
56   }
57
58   source_control {
59     repo_url = var.AS_bootstrap
60     branch = var.AS_bootstrap_branch
61     manual_integration = true
62     use_mercurial = false
63   }
64
65   depends_on = [
66     azurerm_app_service_plan.Server-SP,
67   ]
68 }
69
70 resource "azurerm_app_service" "Server-AS1" {
71   count = var.Amount
72   name = "${var.Environment}-${var.Users_Name[count.index]}-1"
73   location = azurerm_resource_group.Server-RG[count.index].location
74   resource_group_name = azurerm_resource_group.Server-RG[count.index].name
75   app_service_plan_id = azurerm_app_service_plan.Server-SP[count.index].id
76
77   tags = {
78     "Student-email" = "${var.Users_Email[count.index]}",
79     "Student-name" = "${var.Users_Name[count.index]}"
80   }
81
82   source_control {
83     repo_url = var.AS_bootstrap
84     branch = var.AS_bootstrap_branch
85     manual_integration = true
86     use_mercurial = false
87   }
88
89   depends_on = [
90     azurerm_app_service_plan.Server-SP,
91   ]
92 }
93
94 resource "azurerm_role_assignment" "example" {
95   count = var.Amount
96   scope = "/subscriptions/bc9fe0f4-4aa6-4e8a-859a-2909dc00af8e/resourceGroups/${var.Users_Name[count.index]}-${var.Environment}"
97   role_definition_name = "Contributor"
98   principal_id = var.Users_Id[count.index]
99
100   depends_on = [
101     azurerm_resource_group.Server-RG,
102   ]
103 }
```

Alle zaken met een pijl aangeduid zorgen voor de dynamische werking van je script. De kleurencodes zal ik nu even uitleggen. Algemeen zie je bij elke pijl dat de term (var."iets") gebruikt wordt, dit is de verwijzing naar de variabelen in het vars.tf bestand.

-> De blauwe pijl toont aan dat bij elke resource die je voor elke user wilt aanmaken een count moet gaan gebruiken, gebruik je deze count niet zal dit terraform script maar voor 1 enkele user aangemaakt worden.

-> De roze pijlen tonen de tags aan, deze gebruik ik persoonlijk om makkelijk te kunnen filteren per user.

-> De zwarte pijl staat voor de benamingen van resources. We willen natuurlijk dat deze uniek is per user. Je kunt hier natuurlijk zo ver in gaan naar eigen zin. Indien je de user variabelen gebruikt uit het vars.tf bestand is het [count.index] zeer belangrijk als de variabele zelf meerdere waarden bevat. Bij {var.Environment} is er geen [count.index] nodig omdat deze 1 enkele waarde bevat.

-> De gele pijlen verwijzen naar een vooraf gebouwde resource. Bijvoorbeeld: de `azurerm_app_service_plan` resource verwijst naar de bijhorende resource group, dit moet je standaard ook doen, wat hier nu nog bijkomt is dat ook bij deze elementen een [count.index] bijkomt, omdat we specifiek naar de resource group willen vermelden van de correcte user. We willen niet dat er een service wordt aangemaakt met de naam Bram en dan in de resource group van Jan terecht komt.

-> De 2 groene pijlen op het einde zijn wat speciaal. Deze resource zorgt ervoor dat elke user zijn eigen omgeving te zien krijgt in azure, en enkel zijn eigen omgeving. De eerste pijl zorgt dat de correcte resource group genomen wordt, de tweede zorgt dan dat de correcte user toegang heeft tot die resource group. De tweede pijl maakt gebruik van de users hun object id in azure, er is voor het moment geen andere manier om via een terraform script dit dynamisch te doen.

4.3: CSVFiles directory

Om het setup script correct te laten werken hoort deze map te bestaan, en horen je csv files in deze map te zitten. Je kan indien je wilt de directory naam aanpassen, maar dan moet je deze ook aanpassen in het setup script:

```
6 csvfiles=$(find ./CSVFiles/ -type f -name "*.csv")
```

Op lijn 6 in het script kun je de benaming CSVFiles aanpassen naar iets anders, laat zeker de 2 '/' tekens staan. Wees ook zeker dat deze benaming hetzelfde is in het script en als map naam zelf.

4.4: Setupterraform.sh

In principe valt hier niet veel aan op te passen. Enkel wanneer je dus een cron waarde toevoegt/verwijdert/aanpast moet je op specifieke stukken in het setup script aanpassingen maken. Aan andere zaken hoeft je niet te komen. In het technisch-document wordt de code nog eens specifiek uitgelegd wat wat doet.

4.5: Deleteter.sh

Hier moet niets aangepast worden, de naam van het script moet ook hetzelfde blijven. Je kunt dit opnieuw aanpassen indien je wilt maar dan moet je dit ook weer aanpassen in het setup script:

```
138 line="$cronSyntax $currentDir/Deleteter.sh $currentDir ${usedEnvironment::-1} > /var/log/backup.log 2>&1 #&
    {usedEnvironment::-1}"
```

Op lijn 138 kun je het woord Deleteter.sh aanpassen naar de benaming die jij wilt. Het moet overeenkomen met de effectieve naam van dat deleteter script.

4.6: Replace-varstf.ps1

Opnieuw hetzelfde zoals het Deleteter.sh script. In het script zelf valt er niets aan te passen. De naam kun je opnieuw veranderen maar dan moet je dit ook hier doen:

```
143 pwsh Replace-varstf.ps1 ${usedEnvironment::-1} $csv
```

Op lijn 143 kun je Replace-varstf.ps1 aanpassen naar iets anders indien gewenst.

4.7: Nieuwe environment toevoegen

Zoals reeds besproken kun je je eigen terraform omgevingen gaan gebruiken met dit script. Het enige wat je zeker moet gaan doen is zorgen dat je omgeving in 1 enkele map zit, en niet nog dieper gaat in die map zelf, deze ene map zet je op dezelfde directory hoogte als de voorbeeldmappen: (Web/Infrastructure/VM).

howest
hogeschool