howest
hogeschool

Research-Project

Technisch-document Lagast Sean

Contents

Inleiding

Nu je zowel het project zelf kan gaan opzetten als gaan aanpassen wil ik ook nog even de code van alle script gaan becommentariëren om een inzicht te geven hoe alles werkt. Persoonlijk vind ik dat er veel aangepast kan worden. Ik had liever nog gewerkt met een echte kleine applicatie via python maar door de tijdspanne is dit er voor het moment niet van gekomen. Ik ben wel van plan deze code wat te updaten in mijn vrije tijd om een nog beter resultaat te bekomen zodat de gebruiker het nog makkelijker krijgt.

1: Inhoud scripts

De code zal bij elk deel weergegeven worden, en dan daaronder zal commentaar geschreven worden.

1.1: Deleteter.sh

```
1.  #!/bin/bash
2.
3.  #Positional variables
4.  currentDir=$1
5.  directory=$2
6.
7.  #Go into correct terraform environment
8.  cd "$currentDir/$directory"
9.
10. #Destroy terraform environment
11. /usr/local/bin/terraform destroy -auto-approve
12.
13. #Delete crontab
14. crontab -u $(whoami) -l | grep -v "#$directory" | crontab -u $(whoami) -
```

1. De shebang zorgt ervoor dat je systeem weet dat het met een script zal werken, in dit geval dus een bash script.

4. De variabel currentDir is een positional variabel dat bijhoudt wat de huidige directory is. Deze is afkomstig uit de crontab entry die is aangemaakt wanneer je het setup script start.

5. Hetzelfde als lijn 4, maar neemt de naam van de environment op, dus bijvoorbeeld; (Web/Infrastructure/VM).

8. Met de 2 variabelen die we hebben kunnen we nu specifiek gaan duiken in de omgeving die we willen verwijderen.

11. Dit commando zorgt ervoor dat de terraform omgeving afgebroken wordt, en dat alle resources dus verwijdert worden.

14. Ten slotte moet de crontab regel die dit script activeert ook verwijderd worden en daar dient deze regel voor.

1.2: Replace-varstf.ps1

```
1.  #!/usr/bin/pwsh -Command
2.
3.  #Variables send by the Setup script
4.  $usedEnvironment=$args[0]
5.  $csv=$args[1]
6.
7.  #Creating correct path to vars.tf file
8.  $string1=".\"
9.  $string2="$usedEnvironment\"
10. $string3="vars.tf"
11. $Varspath=$string1+$string2+$string3
12.
13. #Replacing / syntax by \
14. $CSVPath = $csv.replace('/','\')
15.
16. #Importing csv file
17. $UserList = Import-Csv -Path $CSVPath -Delimiter ";"
18.
19. #Getting the content of the Varspath
20. $Content = Get-Content $Varspath
21.
22. #Initiating variables for later use
23. $UserNameString = ""
24. $EmailString = ""
25. $IdString = ""
26. $AmountUsers = 0
27. $Environment = ""
28.
29. #Looping through the csv file
30. Foreach ($User in $UserList)
31. {
32.     $AmountUsers = ($AmountUsers + 1)
33.     $UserName = $User.UserName
34.     $Email = $User.Email
35.     $Id = $User.Id
36.
37.     $UserNameString = "$UserNameString" + "`"$UserName`","
38.     $EmailString = "$EmailString" + "`"$Email`","
39.     $IdString = "$IdString" + "`"$Id`","
40. }
41.
42. $UserNameString = $UserNameString.TrimEnd(',')
43. $EmailString = $EmailString.TrimEnd(',')
44. $IdString = $IdString.TrimEnd(',')
45. $Environment = $usedEnvironment
46.
47. $Content | ForEach-Object { $_ -replace ".+#UserName","    default =
    [$UserNameString] #UserName" } | ForEach-Object { $_ -replace
    ".+#Email","    default = [$EmailString] #Email" } | ForEach-Object { $_ -replace
    ".+#Id","    default = [$IdString] #Id" } | ForEach-Object { $_ -replace
    ".+#Amount","    default = $AmountUsers #Amount" } | ForEach-Object { $_ -replace
    ".+#Environment","    default = `"$Environment`" #Environment" } | Set-Content
    $Varspath
```

1. Opnieuw de shebang dat aantoont wat voor een script dit voorstelt.

4. Tijdens het runnen van het setup script wordt dit powershell script uitgevoerd. Hier worden 2 variabelen meegegeven. De eerste zijnde het type omgeving, dus de voorbeelden (Web/Infrastructure/VM).

5. De tweede variabele is de naam van welk csv bestand jij wilt gebruiken om je omgeving op te zetten, dit is het volledig pad, dus het CSVFiles/csvbestand.csv in zijn geheel.

11. De lijnen 8/9/10 vormen samen het volledig pad naar de vars.tf file waar je omgeving staat, dit kon eenvoudiger gedaan zijn. Het is om duidelijk aan te tonen dat er een pad gemaakt wordt uit die waarden. De waarden uit het csv bestand worden straks in deze vars.tf file gezet.

14. In powershell moet je naam van paden gaan noteren met een \ en niet met een /, daarom wordt tijdens het sturen van deze variabele de / tekens hier omgezet naar een \.

17. Hier wordt het csv bestand ingeladen en omgezet naar een variabele die straks bruikbaar is.

20. Hier kunnen we nu de waarden uit de correcte vars.tf file gaan halen, later zetten we de nieuwe waarden die erin moeten komen erbij.

23/24/25/26/27. Dit zijn de waarden die in de vars.tf file zullen komen staan, hier worden ze eerst eens gedeclareerd.

30. Vanaf hier wordt elke lijn uit het csv bestand doorlopen, hier worden dan acties op uitgevoerd.

32. De eerste is een waarde die niet rechtstreeks uit het csv bestand komt. Om ons terraform script te laten werken moeten we meegeven met hoeveel users het script moet werken. Dus telkens er een waarde passeert telt deze regel er 1 persoon bij, tot de laatste user ingelezen is.

33/34/35. Deze waarden worden gelezen uit het csv bestand. Daarom dat de benamingen UserName/Id/Email zo ingevuld moeten staan in de eerste rij van het csv bestand, anders herkent het script niet de waarden die gebruikt moeten worden.

37/38/39. Als eindresultaat moeten alle users dus in 1 string ingevuld worden als variabele in het vars.tf bestand, daarom wordt er telkens wanneer er een nieuwe user bijkomt deze toegevoegd aan de al reeds bestaande string gevolgd door een komma, om een volgend persoon te kunnen toevoegen.

42/43/44. Eenmaal stappen 37/38/39 gedaan zijn moet de laatste komma verwijderd worden omdat er niets meer zal volgen.

45. Dit is in principe een nutteloze stap, en kon bij stap 4 al recht gezet worden, maar tijdens de overgang van het ene script naar het ander gebruik ik graag dezelfde variabel namen zodat ik weet wat van waar komt.

47. Nu nemen we de variabel uit stap 20, en gaan we elke variabel in het vars.tf bestand gaan aanpassen om de beurt. Door de gebruikte #'s weet het script waar iets moet aangepast worden. Het einde van deze lijn zorgt dat de aanpassingen worden opgeslagen.

1.3: Setupterraform.sh

```
#!/bin/bash

#Variables setup
cronSyntax=""

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

ListDir=$(ls -d */)
DirVar=()
i=0
for dir in $ListDir
do
    if [ "${dir::-1}" == "venv" ]
    then
        :
    elif [ "${dir::-1}" == "CSVFiles" ]
    then
        :
    else
        DirVar+=($dir)
    fi
done


#Start script
echo "Welcome to the terraform setup script."
echo "Answer by giving the number corresponding the category!"
echo ""
echo "Which environment would you like to setup?"
for dir in "${DirVar[@]}"
    do
        echo "$i: ${dir::-1}"
        i=$((i+1))
    done
read setup
echo ""
if ! [[ "$setup" =~ ^[0-9]+$ ]]
then
    echo "Sorry integers only!"
else
    if [ -z ${DirVar[$setup]} ]
    then
        echo "Pick a given number!"
    else
        usedEnvironment=${DirVar[$setup]}
        echo "Are you sure you want to build a ${usedEnvironment::-1} environment?"
        echo "0: Yes"
        echo "1: No"
        read answer
        echo ""
        if ! [[ "$answer" =~ ^[0-9]+$ ]]
        then
            echo "Sorry integers only!"
        elif [ "$answer" == "0" ]
        then
            echo "Setting up environment.."
            echo ""
            echo "Which csv file would you like to use?"
            i=0
            for csv in $csvfiles
            do
                CSVVar+=($csv)
                echo "$i: ${csv##*/}"
                i=$((i+1))
            done
```

```
67.             read csvnumber
68.             echo ""
69.             if ! [[ "$csvnumber" =~ ^[0-9]+$ ]]
70.             then
71.                 echo "Sorry integers only!"
72.             elif [ -z ${CSVVar[$csvnumber]} ]
73.             then
74.                 echo "Pick a given number!"
75.             else
76.                 i=0
77.                 check=0
78.                 for csv in "${CSVVar[@]}"
79.                 do
80.                     if [ $csvnumber -eq $i ]
81.                     then
82.                         echo "How long do you want the environment to exist?"
83.                         echo "0: Forever"
84.                         echo "1: 2 hours"
85.                         echo "2: 1 day"
86.                         echo "3: 1 week"
87.                         echo "4: Fill in own crontab syntax"
88.                         read duration
89.                         echo ""
90.                         if ! [[ "$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
107.                                newDate=$(date +"%Y-%m-%d %T" -d "+1 week")
108.                                cronSyntax="${newDate:14:2} ${newDate:11:2}
    ${newDate:8:2} ${newDate:5:2} *"
109.                            elif [ "$duration" -eq "4" ]
110.                            then
111.                                echo "Cron syntax must be exactly like this one
    between brackets, don't add the brackets"
112.                                echo ""
113.                                echo "(* * * * *)"
114.                                echo ""
115.                                echo "For more info check: https://crontab.guru/"
116.                                echo ""
117.                                echo "Fill in your cron syntax:"
118.                                read cronSyntax
119.                            else
120.                                echo "Pick a given number!"
121.                                check=1
122.                            fi
123.
124.                            if [ $check -eq 1 ]
125.                            then
126.                                :
127.                            else
128.                                if [ -z "$cronSyntax" ]
129.                                then
130.                                    echo "No crontab record will be made"
```

```
131.                                 else
132.                                     currentDir=$(pwd)
133.                                     line="$cronSyntax $currentDir/Deleteter.sh
    $currentDir ${usedEnvironment::-1} > /var/log/backup.log 2>&1 #${usedEnvironment::-
    1}"
134.                                     (crontab -u $(whoami) -l; echo "$line" ) |
    crontab -u $(whoami) -
135.                                 fi
136.
137.                                 echo "Terraform script will now be executed!"
138.                                 pwsh Replace-varstf.ps1 ${usedEnvironment::-1}
    $csv
139.
140.                                 cd $usedEnvironment
141.                                 terraform init
142.                                 terraform validate
143.                                 terraform plan -out="plan"
144.                                 terraform apply "plan"
145.                             fi
146.                         fi
147.                     else
148.                         :
149.                     fi
150.                 i=$((i+1))
151.                 done
152.             fi
153.         elif [ "$answer" == "1" ]
154.         then
155.             echo "Run script again to start over."
156.         else
157.             echo "No corresponding number given!"
158.         fi
159.     fi
160. fi
161. #End script
```

Eerst een klein beetje algemene info. In dit script worden er heel wat echo's uitgevoerd maar deze spreken wel voor zich, het is de tekst dat je te zien krijgt terwijl je het script runt. Ook zitten er veel checks in het script die nagaan of je wel een correcte keuze gemaakt hebt en of je al dan niet wel een nummer getypt hebt.

1. Opnieuw de shebang.

4. Declareren van variabele dat later zal gebruikt worden.

6. Deze variabele bevat een volledig commando dat straks van pas komt, dit commando op zichzelf gaat alle csv bestanden in de CSVFiles gaan opzoeken.

7. Declareren van een lege array, ook voor later gebruik.

9. Houd een lijst bij van alle mappen in de huidige map, dit wordt gebruikt om de omgevingen te gaan opslaan later zodat we ernaar kunnen navigeren.

10. Opnieuw een lege array voor later gebruik.

12-23. Dit stuk code gaat elke map die bestaat gaan opslaan in de variabele van stap 10. De map CSVFiles bestaat natuurlijk ook en deze wordt gefilterd, tijdens tests werkte ik ook met een venv file dus deze staat ook gefilterd, dit stuk code mag gerust verwijderd worden.

31-35. Elke map uit de 'dir' array wordt weergegeven aan de gebruiker zodat deze een keuze kan maken.

36. Uit voorgaande stap wordt nu gevraagd welk van de omgevingen je kiest, dit wordt nu bijgehouden tot we het nodig hebben.

38. Zoals vermeld is dit 1 van de checks die gebeurt of je wel degelijk een nummer ingevoerd hebt bij stap 36.

42. Ook dit is een check dat kijkt of je een keuze gemaakt hebt binnen de keuzes die je kon maken.

46. Na het gekozen nummer moeten we ook de waarde gaan ophalen. Dit gebeurd door uit de array de waarde te halen die op dezelfde plaats staat als het ingevoerde nummer uit stap 36.

50. Er wordt dan gevraagd of je wel zeker bent van je keuze, deze stap is wat overbodig maar zal ik later verwijderen.

52. Opnieuw een check.

55. Indien je akkoord bent met je keuze gaat het script verder.

61-66. Hier krijg je nu elke csv die je hebt staan onder CSVFiles als keuze. Ook worden deze opgeslaan in de csv array.

67. Nu maak je opnieuw de keus welk nummer csv je wilt gebruiken.

69. Opnieuw een check.

72. Ook een check.

78/80/150. Elke csv uit de csv array wordt overlopen. Indien het csvnummer niet overeenkomt met het huidig $i nummer wordt $i +1 gedaan. Wanneer $i wel overeenkomt met het csvnummer dan gaat het script verder.

88. Hier wordt gevraagd hoelang je de terraform omgeving wilt laten bestaan.

90. Ook een check.

94-96. Indien je 0 gekozen hebt gebeurd er niets, want je koos er voor om de omgeving voor altijd te laten bestaan.

97-100. Indien je 1 nam koos je er voor de omgeving 2uur te laten bestaan. Cron heeft een specifieke syntax nodig om te kunnen werken, daarom gebeuren deze tussenstappen om het gewone uur om te zetten naar iets waar cron mee overweg kan.

101-104. Gelijk aan de stap hierboven maar voor een ander tijdstype.

105-108. Zelfde als de stap hierboven.

109-118. Je kon ook kiezen om een eigen cron syntax mee te geven, dat wordt dan hier ingelezen, let op, want je moet specifiek zijn in wat je meegeeft, dit is reeds uitgelegd in de gebruikershandleiding.

124-126. Opnieuw een check.

128-130. Ook een check.

131-135. Nu de tijdsduur gekozen is kan de lijn toegevoegd worden in crontab zelf. De lijn bestaat uit de cronsyntax gevolgd door het pad van het Deleteter.sh script, gevolgd door de 2 positionele variabelen dat het script nodig heeft, gevolgd met het zelf gemaakte log bestand. Op het einde wordt er een # toegevoegd samen met de benaming van de gemaakte omgeving.

138. Nu wordt het powershell script uitgevoerd met de nodige variabelen.

140. Hier verdiepen we ons in de map van de te maken omgeving.

141-144. Deze commando's zijn de setup om de terraform omgeving op te zetten.

153. Indien je bij stap 55 voor no koos stopt het script hier ook.

2: Gebruikte bronnen bij aanmaak scripts/terraform setup en research

2.1: Eigen gemaakte posts

https://www.reddit.com/r/Terraform/comments/shrg99/school_research_project/
https://discuss.hashicorp.com/t/school-research-project/34731
https://github.com/LagastSean/Research-Project

2.2: Extern geraadpleegde bronnen

[1]
'Add lines to cron from script', *Ask Ubuntu*. https://askubuntu.com/questions/58575/add-lines-to-cron-from-script (geraadpleegd 20 januari 2022).

[2]
'ARM Templates Or HashiCorp Terraform - What Should I Use?', *Azure DevOps Blog*, 24 november 2020. https://devblogs.microsoft.com/devops/arm-templates-or-hashicorp-terraform-what-should-i-use/ (geraadpleegd 13 januari 2022).

[3]
'bash - Checking if an input number is an integer', *Unix & Linux Stack Exchange*. https://unix.stackexchange.com/questions/151654/checking-if-an-input-number-is-an-integer (geraadpleegd 20 januari 2022).

[4]
'Best Infrastructure as Code Tools (IaC): The Top 10 for 2022'. https://bluelight.co/blog/best-infrastructure-as-code-tools (geraadpleegd 12 januari 2022).

[5]
'Cloud Computing Services | Microsoft Azure'. https://azure.microsoft.com/en-us/ (geraadpleegd 12 januari 2022).

[6]
'Cloud Services - Amazon Web Services (AWS)', *Amazon Web Services, Inc.* https://aws.amazon.com/ (geraadpleegd 13 januari 2022).

[7]
'cron - Verify if crontab works', *Ask Ubuntu*. https://askubuntu.com/questions/85558/verify-if-crontab-works (geraadpleegd 15 januari 2022).

[8]
'Cron job every 5 minutes'. https://crontab.guru/every-5-minutes (geraadpleegd 20 januari 2022).

[9]
Flipphones, 'Deploy and destroy after a period of time', *r/Terraform*, 22 maart 2020. www.reddit.com/r/Terraform/comments/fn8h35/deploy_and_destroy_after_a_period_of_time/ (geraadpleegd 21 januari 2022).

[10]
'Home - Terraform Cloud and Terraform Enterprise', *Terraform by HashiCorp*. https://www.terraform.io/cloud-docs (geraadpleegd 14 januari 2022).

[11]
M. says, 'How to automate Terraform deployments', *UpCloud*. https://upcloud.com/?p=8340 (geraadpleegd 17 januari 2022).

[12]
A. V. G. L. updated: April 5 en 2021 22 Comments, 'How To Bash Shell Find Out If a Variable Is Empty Or Not', *nixCraft*, 6 december 2012. https://www.cyberciti.biz/faq/unix-linux-bash-script-check-if-variable-is-empty/ (geraadpleegd 25 januari 2022).

[13]
'How to Check if a Cron Job has Run (Crontab Log)', *InMotion Hosting Support Center*, 25 januari 2013. https://www.inmotionhosting.com/support/edu/cpanel/did-cron-job-run/ (geraadpleegd 24 januari 2022).

[14]
'How to Use PowerShell to Escape Double Quotes', 25 juli 2019. https://adamtheautomator.com/powershell-escape-double-quotes/ (geraadpleegd 20 januari 2022).

[15]

‘Latest topics’, *HashiCorp Discuss*, 25 januari 2022. https://discuss.hashicorp.com/latest (geraadpleegd 23 januari 2022).

[16]
‘PowerShell Concatenate String | Different Examples of Concatenate String’, *EDUCBA*, 14 maart 2020. https://www.educba.com/powershell-concatenate-string/ (geraadpleegd 19 januari 2022).

[17]
‘Pulumi vs. Terraform’, *pulumi*. https://www.pulumi.com/docs/intro/vs/terraform/ (geraadpleegd 12 januari 2022).

[18]
‘r/Terraform - School Research Project’, *reddit*. https://www.reddit.com/r/Terraform/comments/shrg99/school_research_project/ (geraadpleegd 20 januari 2022).

[19]
‘syslog - No logs are written to /var/log’, *Ask Ubuntu*. https://askubuntu.com/questions/615457/no-logs-are-written-to-var-log (geraadpleegd 27 januari 2022).

[20]
‘ubuntu install powershell - Google Search’. https://www.google.com/search?q=ubuntu+install+powershell&rlz=1C1GCEU_nlBE936BE936&oq=ubuntu+install+&aqs=chrome.3.69i57j69i59l3j0i20i263i512j0i512l2j69i60.5733j0j4&sourceid=chrome&ie=UTF-8 (geraadpleegd 25 januari 2022).

howest
hogeschool

howest.be