**Introductie**

In deze SIG gaan we kijken naar CI/CD binnen AWS: het uitrollen en onderhouden van Lambda functies binnen een CI/CD straat. We kijken eerst naar een voorbeeld van CI/CD zoals AWS ons dat aanbiedt (iedereen kent dit voorbeeld, dus na deze SIG wij ook). Daarna gaan we CI/CD inrichten voor onze eigen objecten.

We zullen voor onze eigen objecten een simpel voorbeeld nemen: een API Gateway met een Lambda functie daarachter. De API Gateway maakt gebruik van een DNS entry in het domein cloudhotel.org. Het plaatje ziet er als volgt uit:

<<plaatje>>

Het certificate bestaat al, de DNS-entries zullen aangemaakt worden binnen deze SIG.

We gaan e.e.a. uitrollen via terraform. Voor mensen die terraform niet kennen, zullen we dit eerst doen vanaf de VM, pas daarna vanuit de CI/CD omgeving van AWS.

**SIG omgeving**

Binnen deze SIG heb je een user-id gekregen. In deze beschrijving ga ik uit van AMIS1, in de regio eu-west-1, vervang dit door de naam en de regio die je van mij hebt gekregen. Bij elk user-id hoort ook een eigen VM. Je ziet een overzicht van user-id’s met bijbehorende IP-adressen in de presentatie. Om op de VM te komen heb je een key nodig, die heb je ook van mij gekregen. Niet iedereen zit in dezelfde regio (dat zou het allemaal te onoverzichtelijk maken). Let op: de key verschilt per regio. Op de sheet zie je ook naar welke regio je toe moet.

Je kunt aanloggen naar de VM met

ssh -i AMIS-eu-west-1 ec2-user@1.2.3.4

Gebruiken van CodeCommit binnen je virtual machine

Het is mogelijk om binnen een VM gebruik te maken van git zonder credentials op te geven. Je gebruikt dan de gegevens van de gebruiker waarmee je aangelogd bent. Dit kan, door het package **git-remote-codecommit** te installeren op de VM [1]. Binnen deze SIG is dat al voor jou geïnstalleerd.

Configureren van je VM

Je moet nog wel een paar kleine aanpassingen doen binnen je VM: gebruik het commando **aws configure** om je AWS instellingen aan te passen. Sla de eerste drie vragen over (alleen enter) en vul bij de vierde json in:

AWS Access Key ID [None]:

AWS Secret Access Key [None]:

Default region name [None]:

Default output format [None]: json

Clonen van je repository

Je bent nu klaar om je repository voor het eerst te clonen. Dit kan via het standaard git clone commando. Omdat dit een AWS repo is, heeft het een iets andere vorm dan je bij github zou gebruiken:

git clone codecommit::eu-west-1://AMIS1-repo

Je ziet nu een waarschuwing dat deze repo leeg is. Dat klopt, we gaan hem nu vullen. Maar voordat we dat doen, voegen we onze naam en e-mail toe:

cd AMIS1-repo

git config --global user.name “<jouw naam>”

git config --global user.email “<jouw e-mail>”

We gaan de bestanden onder /start kopiëren naar de AMIS1-repo:

cd ~

cp -r start/\* AMIS1-repo

Laten we ze ook toevoegen aan de repository binnen AWS:

cd AMIS1-repo

git status

git add --all .

git commit -m “First commit”

git push origin master

Ga in de GUI naar de AWS CodeCommit service. Klik op de link van je repository, je ziet nu het bestand terraform\_sig.tf en de directory lambdas, met daarin de file sig.py.

Kijk in het linker menu naar Commits, je ziet nu je eigen naam bij de First commit. Als je op het commit-id klikt, kun je ook zien wat er gewijzigd is.

Terraform

Laten we, voordat we de repo vullen, eerst kijken wat we uit gaan rollen. In je home directory zie je een directory “start”. Ga met **cd start** naar deze directory. In deze directory staat een terraform file met de naam terraform\_sig.tf. Open deze file met vi of met nano Een van de variabelen is user\_prefix, deze heeft de waarde “AMIS1”. Wijzig dit in de naam van de user-id die je gekregen hebt. Alles wat door jou uitgerold wordt, begint met dit voorvoegsel. Controleer ook of de aws\_region de goede regio heeft (met de juiste afkorting) en pas dit aan indien nodig. Sla de gewijzigde file op.

Je ziet verder, dat er geen access key en secret access key gebruikt worden: deze waarden zijn uit gecommentarieerd. Als je van buiten AWS (bijv. vanaf een VM op je laptop of vanaf Windows) naar AWS gaat, dan zijn deze velden verplicht. Nu je op een virtual machine in AWS zit, nu hangt de policy aan de virtual machine en Terraform herkent dat je die permissies gaat gebruiken in plaats van de permissies die aan een access key of secret access key hangen.

Ga binnen AWS naar service EC2, ga naar Instances > instances en zoek uit welke rechten je virtual machine heeft. (Kom je er niet uit? Bekijk dan de antwoorden aan het eind van dit document, onder [1]).

Ga nu uit de editor. Laten we de objecten uitrollen:

cd lambdas

zip sig.zip sig.py

cd ..

../terraform init

../terraform plan -out terraform.tfplans

../terraform apply terraform.tfplans

API Gateway

Ga nu naar de service API Gateway. Je ziet (onder meer) een gateway met de naam AMIS1\_api\_gateway. Klik op deze link. Je ziet dat de API Gateway een resource /sig heeft. Klik op de POST onder de /sig:

<<Afbeelding post onder \_sig>>

Je ziet nu een plaatje hoe de stroom van aanroep van de API via de Lambda functie AMIS1\_sig terug gaat naar de client. Kies in het linker menu voor Stages, daarna voor prod. In het scherm dat nu naar voren komt, zie je een “Invoke URL”.

<<Afbeelding invoke url>>

Kopieer deze, voeg /sig toe en gebruik deze in je virtual machine:

curl -X POST <https://m1az0lc5th.execute-api.us-east-1.amazonaws.com/prod>/sig

Je ziet nu de tekst “Hello from Lambda”.

Lambda functions

Ga in de GUI naar de AWS Lambda service en klik op AMIS1\_sig, kijk naar de code. Dit is de standaard tekst voor een Lambda functie is die in Python geschreven is, je ziet nu ook waar de “Hello from Lambda” vandaan komt.

Opruimen

Ga terug naar je virtual machine en typ de volgende opdracht:

**../terraform destroy -auto-approve**

Je omgeving wordt nu (zonder verdere vragen) netjes opgeruimd. Controleer onder de API Gateway en onder Lambda dat je inderdaad geen AMIS1-objecten meer hebt.

Bijwerken van CodeCommit

Als je wijzigingen hebt gemaakt t.a.v. user-id of regio, kopieer je terraform\_sig.tf file van de start directory naar de AMIS1-repo directory en zet deze in de repository in AWS (zie evt. de **git add**, **git commit** en **git push** statements eerder in deze beschrijving).

Branching

Je kunt binnen AWS branchen, zoals je dat ook binnen github kunt. Maak een branch testbranch aan en check deze branch uit:

git branch testbranch

git checkout testbranch

Pas iets aan (voeg bijv. een commentaar regel toe aan de file terraform\_sig.tf) en check de file in (gebruik weer git add, git commit en gebruik **git push origin testbranch** om de branch in te checken).

Ga in het menu van CodeCommit naar Pull requests, kies bij destination voor “master” en bij source voor “testbranch”. Klik nu op Compare.

Je ziet dat de testbranch te mergen is met master, onder in het scherm zie je de wijzigingen en de commits die nodig waren om zover te komen. Bedenk een title en klik op “Create pull request”

p/m: beschrijven afdwingen meerdere mensen laten kijken

p/m: waarom codecommit gebruiken als je ook github kunt gebruiken? -> als je mensen hebt die al rechten op AWS hebben, dan heb je heel veel mogelijkheden om permissies uit te delen (zie image)

**Check, check – dubbelcheck FRA**

Is de aws configure werkelijk nodig?

**<FRA>**

**Antwoorden:**

Antwoord [1] op de vraag: welke rechten heeft de VM?

Kijk naar de eigenschappen van je VM. Je vindt de naam van de role die aan deze VM hangt achter IAM Role. Als je op deze link klikt, dan wordt een tweede tab geopend, met daarin de IAM role die aan de VM hangt. Binnen deze role zie je onder het eerste tabblad een link naar de policy. Als je op deze link klikt, dan zie je het volgende:

<<

**Links**

[1] <https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-git-remote-codecommit.html>