**Workshop CloudFormation**

1. **Woord vooraf**

Als je deze workshop via Gitlab gevonden hebt en geen presentatie vooraf gehad hebt, dan kan het lastig zijn om een begin te vinden. Bekijk de slides en lees de toelichting in het word document in de presentatie folder.

1. **Opdracht 1**

* Probeer zonder de rest van deze hand-out op basis van het volgende diagram de CloudFormation template te maken:

Timeline

Description automatically generated

Het gaat dus om:

* + 1 VPC
  + 1 Internet Gateway
  + 2 Subnets
  + 1 Route Table

Op de volgende pagina’s loop ik er stap-door-stap doorheen. Je leert het meest door hier niet naar te kijken en alles zelf te proberen. Gebruik de handout (of onze hulp) als je vast komt te zitten.

* 1. Internet Gateway
* Voeg aan de eerdere code een Internet Gateway toe. Geef de internet gateway een naam waar zowel igw (of internetgateway) als je eigen naam in voorkomt.
* Controleer dat het werkt: deploy je stack. Je kunt je template deleten en opnieuw aanmaken, je kunt ook de Update button gebruiken.
  + Wordt je stack succesvol uitgerold?
  + Zie je je VPC?
  + Zie je je Internet Gateway?
* Je ziet een Internet Gateway, maar deze is detached.

Graphical user interface, application

Description automatically generated

* Hoe kun je dit oplossen? Je leert het meeste door te doen. Ik geef bovenaan de volgende pagina een hint, daarna onderaan de pagina de oplossing.

**Hint:** Documentatie:



Je hebt dus een resource nodig die de Internet Gateway koppelt aan de VPC. Google is je vriend!

**Oplossing:**

Google naar “AWS CloudFormation Internet Gateway attach VPC” en je ziet de oplossing:



De volgende vraag is: welke parameters moet je invullen voor de resource VPCGatewayAttachment? En wat moet je bij die parameters invullen? Opnieuw: denk hier eerst zelf over na. Op de volgende pagina weer eerst een hint en daarna de oplossing.

**Hint:** we proberen om een Internet Gateway te koppelen aan de VPC. Je zoekt naar een InternetGatewayId en naar een VpcId. Het gaat om de VPC en de Internet Gateway in hetzelfde CloudFormation template. De ID’s zijn -of- het default resultaat van een resource, -of- een attribuut van een resource. Voor beide hebben we een functie gezien in de slides. Je weet waar je de documentatie van de resources zelf kunt vinden…

**Oplossing:** zowel bij de Internet Gateway als bij de VPC krijg je het id terug als standaard resultaat van de resource. Je kunt hier dus de functie **!Ref** gebruiken:

  WordPressVpc:

    Type: AWS::EC2::VPC

    Properties:

      CidrBlock: 10.0.0.0/16

      Tags:

        - Key: Name

          Value: !Sub "wordpress-vpc-${Name}"

  InternetGateway:

    Type: AWS::EC2::InternetGateway

    Properties:

      Tags:

        - Key: Name

          Value: !Sub " wordpress-igw-${Name}"

  InternetGatewayAttachment:

    Type: AWS::EC2::VPCGatewayAttachment

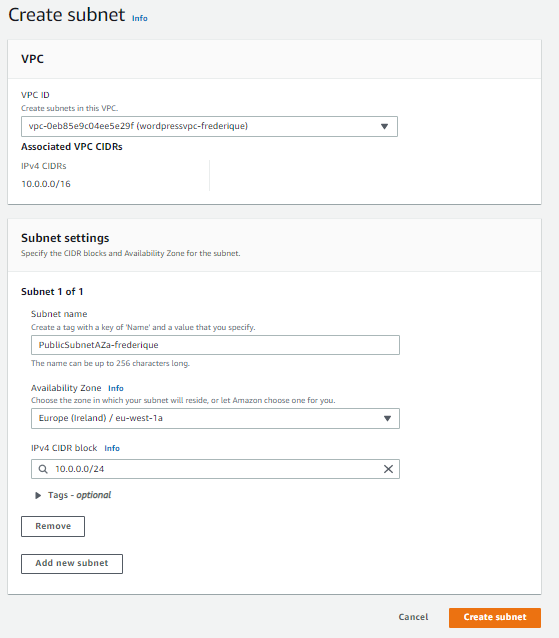
    Properties:

      InternetGatewayId: !Ref InternetGateway

      VpcId: !Ref WordPressVpc

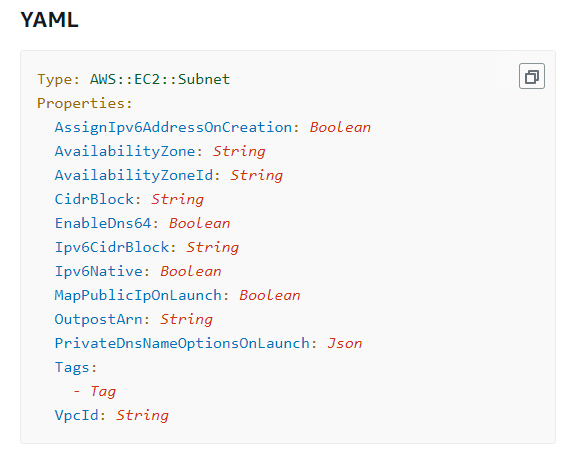
* Deploy je template opnieuw: controleer dat je Internet Gateway nu wel aan de (juiste) VPC hangt.

* 1. Subnets
* Wanneer je in de GUI een subnet aanmaakt dan zie je het volgende scherm:



Je moet dus het VPC Id invullen, een naam (via een tag), een Availability Zone (die er bijvoorbeeld uitziet als “eu-west-1a”) en een CIDR block (in dit voorbeeld: 10.0.0.0/24).

* Laten we de documentatie voor subnets in CloudFormation zoeken: google naar AWS CloudFormation subnet.



Voor ons zijn van belang: AvailabilityZone, CidrBlock, Tags en VpcId. We willen twee Subnets, beide in een andere AvailabilityZone. Laten we AZa en AZb nemen. Als we naar de GUI kijken, dan betekent dit dat we eu-west-1a en eu-west-1b moeten invullen bij de AvailabilityZone. Dus: de regio waar we in deployen met een ‘a’ of een ‘b’ er achter. Er was een trucje om de huidige regio in het template te verwerken, zie eventueel ook de slides…

Probeer het eerst zelf, mocht dit niet lukken dan vind je de oplossing onderaan deze pagina.

**Oplossing:** (copieer en plak dit zodanig dat je ook een public subnet voor AZb hebt)

  PublicSubnetAZa:

    Type: AWS::EC2::Subnet

    Properties:

      AvailabilityZone: !Sub "${AWS::Region}a"

      CidrBlock: 10.0.0.0/24

      VpcId: !Ref WordPressVpc

      Tags:

        - Key: Name

          Value: !Sub "wordpress-public-subnet-aza-${Name}"

* 1. Route table

Als je de beschrijving tot dusver gevolgd hebt zal dit geen enkel probleem opleveren: je krijgt een route table die er als volgt uit ziet:

  PublicRouteTable:

    Type: AWS::EC2::RouteTable

    Properties:

      Tags:

        - Key: Name

          Value: !Sub "wordpress-public-routetable-${Name}"

      VpcId: !Ref WordPressVpc

Binnen de Route Table staat standaard al de route voor je local network (in dit voorbeeld: 10.0.0.0/16 – dus voor de hele VPC). Die hoef je binnen je CloudFormation template dus ook niet aan te maken. We maken wel een extra route aan voor het publieke internet:

  PublicRoutePublicInternet:

    Type: AWS::EC2::Route

    Properties:

      RouteTableId: !Ref PublicRouteTable

      DestinationCidrBlock: 0.0.0.0/0

      GatewayId: !Ref InternetGateway

Als je dit uitrolt en je gaat naar je subnet en klikt op de Route table, dan zie je dat deze 0.0.0.0/0 niet is toegevoegd. Dit komt, doordat je de nieuwe Route Table nog niet gekoppeld hebt aan de subnets. Probeer dit eerst weer zelf, ik geef een hint bovenaan de volgende pagina en de oplossing onderaan de volgende pagina.

**Hint:** Als je een route table opent in de GUI, dan zie je onderaan het scherm meerdere tabs. Een ervan heet “Subnet associations”. Hier kun je in de GUI je route table koppelen aan de GUI.

**Oplossing:** Als je google’t naar AWS CloudFormation Route Table Subnet association, dan vind je de juiste resource AWS::EC2::SubnetRouteTableAssociation. Als je deze invult, dan krijg je voor subnet A:

  PublicSubnetAZaRouteTableAssociation:

    Type: AWS::EC2::SubnetRouteTableAssociation

    Properties:

      RouteTableId: !Ref PublicRouteTable

      SubnetId: !Ref PublicSubnetAZa

1. **Opdracht 2**

Ook hier geldt weer: probeer het eerst zelf uit te zoeken, op basis van het volgende diagram:

Diagram

Description automatically generated with medium confidence

Je moet de volgende resources toevoegen:

* Een virtual machine (EC2 instance)
* Een Security Group, open poorten 80 en 22 voor iedereen op het publieke internet
* Gebruik voor de parameters de volgende tekst (vervang de defaults door de juiste namen, je hebt de username, password, endpoint van de presentator van de workshop gekregen), of kopieer de inhoud van de file “parameters\_opdracht\_2.yaml” naar je eigen template:

Parameters:

  Name:

    Type: String

    Default: frederique

  WordpressUserName:

    Type: String

    Default: '---vul-hier-het-user-id-van-de-database-in---'

  WordpressPassword:

    Type: String

    Default: '---vul-hier-het-password-van-de-database-in---'

    NoEcho: true

  WordpressEndpoint:

    Type: String

    Default: '---vul-hier-het-ip-adres-van-de-database-in---'

  EC2InstanceType:

    Description: Instance type for EC2

    Type: String

    Default: t2.micro

  EC2KeyName:

    Description: Key name for EC2

    Type: String

    Default: frederiquer-20221102

  AmazonLinux2AMI:

    Description: Amazon Linux AMI

    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>

    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86\_64-ebs

* Gebruik voor de User Data van de EC2 Instance de volgende tekst (let op: tekst loopt over naar volgende regels, knip en plak onderstaande tekst of kopieer de inhoud van de file “userdata\_opdracht\_2.yaml” naar je eigen template):

      UserData:

        "Fn::Base64": !Sub |

            #!/bin/bash

            export WORDPRESS\_DATABASE\_NAME="wp\_database\_${Name}"

            export WORDPRESS\_USERNAME="${WordpressUserName}"

            export WORDPRESS\_PASSWORD="${WordpressPassword}"

            export WORDPRESS\_DATABASE\_SERVER\_ENDPOINT="${WordpressEndpoint}"

            export WWW\_DIR="/var/www/html"

            yum update -y

            echo "TRACE Install tar, http and mariadb"

            curl https://frpublic2.s3.eu-west-1.amazonaws.com/Xforce/Code/install\_tar\_http\_mariadb.sh -o ~/install\_tar\_http\_mariadb.sh

            . ~/install\_tar\_http\_mariadb.sh

            echo "TRACE Download WordPress in /opt/wordpress"

            curl https://frpublic2.s3.eu-west-1.amazonaws.com/Xforce/Code/download\_wordpress\_in\_opt\_wordpress.sh -o ~/download\_wordpress\_in\_opt\_wordpress.sh

            . ~/download\_wordpress\_in\_opt\_wordpress.sh

            echo "TRACE Install PHP 8"

            curl https://frpublic2.s3.eu-west-1.amazonaws.com/Xforce/Code/install\_php8.sh -o ~/install\_php8.sh

            . ~/install\_php8.sh

            echo "TRACE Change config file"

            curl https://frpublic2.s3.eu-west-1.amazonaws.com/Xforce/Code/change\_wordpress\_config\_file.sh -o ~/change\_wordpress\_config\_file.sh

            cd /opt/wordpress

            . ~/change\_wordpress\_config\_file.sh

            cd /root

            echo "TRACE Create database"

            curl https://frpublic2.s3.eu-west-1.amazonaws.com/Xforce/Code/create\_database.sh -o ~/create\_database.sh

            . ~/create\_database.sh

            echo "TRACE Copy files and start httpd"

            cp -p -r /opt/wordpress/\* $WWW\_DIR

            systemctl enable --now httpd

* Geef ook het public IPadres terug als output**.**

Woord vooraf: als je virtual machine goed start, dan zul je zien dat de CloudFormation template snel klaar is, maar je ziet dan nog geen website via je browser. Dat kan komen doordat bovenstaande commando’s in de userdata nog uitgevoerd worden. Je kunt dit controleren door aan te loggen naar de Virtual Machine:

**ssh -i frederiquer-20221102.pem ec2-user@1.2.3.4**

en te kijken in de file /var/log/cloud-init-output.log

**sudo -i**

**tail -f /var/log/cloud-init-output.log**

De uitrol is klaar als je als laatste regels:

**TRACE Change config file**

**% Total % Received % Xferd Average Speed Time Time Time Current**

**Dload Upload Total Spent Left Speed**

**100 317 100 317 0 0 4624 0 --:--:-- --:--:-- --:--:-- 4661**

**TRACE Create database**

**% Total % Received % Xferd Average Speed Time Time Time Current**

**Dload Upload Total Spent Left Speed**

**100 174 100 174 0 0 2259 0 --:--:-- --:--:-- --:--:-- 2289**

**TRACE Copy files and start httpd**

**Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.**

**Cloud-init v. 19.3-45.amzn2 finished at Sun, 06 Nov 2022 10:14:56 +0000. Datasource DataSourceEc2. Up 245.89 seconds**

ziet.

**Uitwerking opdracht 2:**

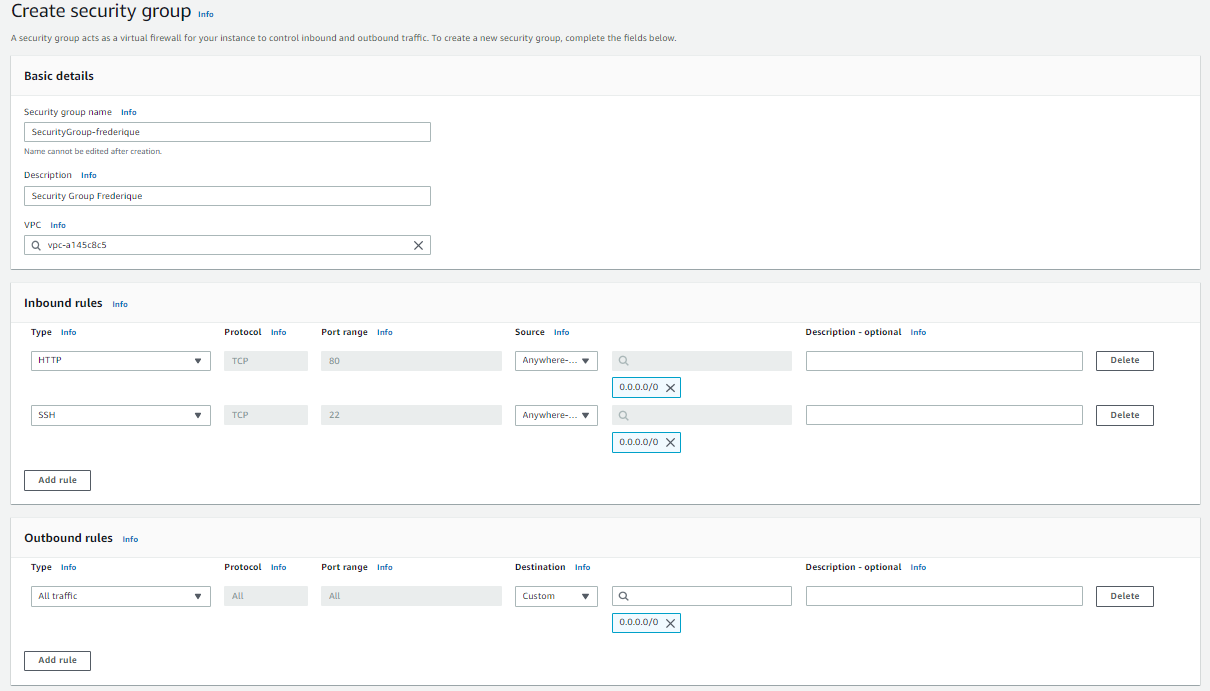
(probeer het eerst zelf!)

* 1. Security Group

In de documentatie zie je een rood blok dat je waarschuwt voor circular dependencies. Hier hoef je je niets van aan te trekken. Je ziet de volgende Yaml in de documentatie:



Als je de GUI volgt, dan zie je het volgende:



Je ziet dat de Outbound Rules er standaard al stonden. Hier hoef je dus ook niets voor toe te voegen in je CloudFormation template. De overige velden (Security group name, description, inbound rules) moet je nog wel toevoegen. Probeer het eerst zelf, kijk daarna pas naar de oplossing op de volgende pagina:

**Oplossing:**

  EC2WordpressSecurityGroup:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupName: EC2WordpressSecurityGroup

      GroupDescription: !Sub "wordpress-sg-${Name}"

      VpcId: !Ref WordPressVpc

      SecurityGroupIngress:

        - IpProtocol: TCP

          Description: SSH

          CidrIp: 0.0.0.0/0

          FromPort: 22

          ToPort: 22

        - IpProtocol: TCP

          Description: HTTP

          CidrIp: 0.0.0.0/0

          FromPort: 80

          ToPort: 80

      Tags:

        - Key: Name

          Value: !Sub "wordpress-sg-${Name}"

N.B: je zult bij veel bedrijven zien dat ze wel degelijk andere resources gebruiken voor AWS::EC2::SecurityGroup en voor de rules binnen de security group, dus voor AWS::EC2::SecurityGroupIngress/SecurityGroupEgress. Dit komt omdat je bijvoorbeeld mag zeggen dat je alleen SSH verkeer vanaf een andere security group toestaat. Bijvoorbeeld: sta alleen poort 22 toe van een VM die security group BastionHostSecurityGroup heeft. Dat is veiliger dan wat we hier doen. In de oplossingen maak ik gebruik van twee SecurityGroupIngress resources voor poort 22 en poort 80 (beide open voor de wereld).

* 1. Virtual Machine

Bij het uitrollen van een Virtual Machine in de GUI zie je dat je best veel dingen kunt invullen, maar dat er ook veel defaults zijn. Probeer eerst in de GUI te ontdekken wat je moet aanpassen om jouw uitrol in jouw VPC goed te laten verlopen. Probeer het eerst zelf, zie daarna de hint bovenaan de volgende pagina en de oplossing onderaan de volgende pagina.

**Hint-1:** je vult de volgende velden in en wijzigt de volgende velden:

* Name: de naam van je VM
* Bij Application and OS Images (Amazon Machine Images) zie je dat standaard Amazon Linux 2 ingevuld is. Zie bij de opdracht beschrijving hierboven hoe je de meest recente versie zelf kunt invullen (via een parameter)
* Instance type is t2.micro. Zie ook de parameters die je kunt gebruiken hierboven.
* Key pair name: zie de parameters die je kunt gebruiken hierboven
* Network settings: in de GUI zal hij standaard je VM in de “default VPC” zetten. In ons geval willen we hem in onze eigen VPC.
* De default in de GUI is om een nieuwe security group aan te maken. Wij hebben in de vorige stap zelf een security groep aangemaakt.
* Onder Advanced Details zie je ook een blok met User data

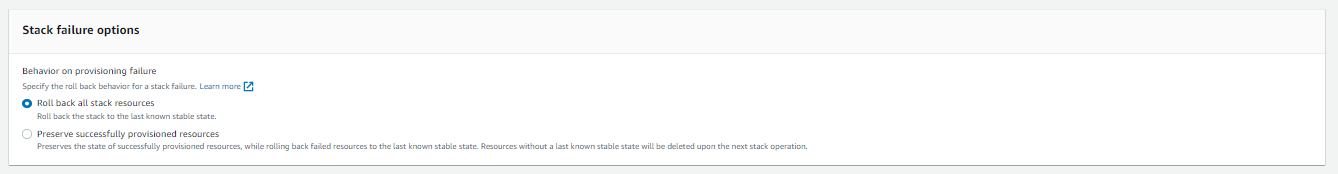
**Hint-2:** je zult zien dat je standaard geen public IP adres krijgt als je op deze manier een VM uitrolt. Je kunt dit op twee manieren oplossen:

* Binnen de definitie van de VM zelf: via NetworkInterfaces > AssociatePublicIpAddress: true
* Binnen de definitie van het subnet: MapPublicIpOnLaunch: true

Dit laatste is verreweg het gemakkelijkst, ik heb in de oplossing ook voor deze tweede gekozen.

**Hint-3:** Je kunt bij de uitrol een foutmelding krijgen die lijkt op: Security group sg-0997f605ffcab8623 and subnet subnet-e0046b84 belong to different networks. (Service: AmazonEC2; Status Code: 400; Error Code: InvalidParameter; Request ID: bfa39bc2-b783-43e4-a028-cc21b2f04560; Proxy: null)

Bij het uitrollen van een stack kun je kiezen voor de default Roll back all stack resources:

****

Bij het uitzoeken van deze foutmelding kun je beter “Preserve successfully provisioned resources” kiezen. Controleer bij welke VPC de genoemde security group hoort en bij welk subnet het genoemde subnet hoort. Wat zou je hieraan kunnen doen?

**Oplossing:**

  EC2WordPress:

    Type: AWS::EC2::Instance

    Properties:

      ImageId: !Ref AmazonLinux2AMI

      InstanceType: !Ref EC2InstanceType

      KeyName: !Ref EC2KeyName

      SubnetId: !Ref PublicSubnetAZa

      Tags:

        - Key: Name

          Value: !Sub "wordpress-ec2instance-${Name}"

      SecurityGroupIds:

        - !Ref EC2WordpressSecurityGroup

      UserData:

        "Fn::Base64": !Sub |

            #!/bin/bash

            export WORDPRESS\_DATABASE\_NAME="wp\_database\_${Name}"

...

            systemctl enable --now httpd

De reden voor de foutmelding in hint-3 is dat als je geen subnet meegeeft als parameter, hij een willekeurig subnet **van de default VPC** pakt. Dat is dus niet hetzelfde netwerk als waar de Virtual Machine in terecht moet komen. Daarom **moet** je een subnet meegeven als je een Virtual Machine in een niet-default VPC wilt uitrollen…

1. **Opdracht 3**

Tot dusver maakte je gebruik van een MySQL database in een andere CloudFormation stack (de stack die ik al voor je uitgerold heb). Het is natuurlijk beter als je je eigen database gaat gebruiken.

**Let op:** om CloudFormation te leren is het prima om dit in dezelfde stack te gaan doen als we ook de Virtual Machine, het netwerk etc. hebben uitgerold. In de praktijk zul je zien dat het netwerk, de VM en de databases in verschillende stacks uitgerold worden. De reden daarvoor is, dat als je in de toekomst ooit een stack wilt weggooien en opnieuw wilt opbouwen dat je dan niet de database wilt weggooien. Hoewel CloudFormation hier oplossingen voor heeft is het vanuit beheersmatige overwegingen verreweg het handigst om dit in verschillende stacks uit te rollen.

Wij houden het simpel – en rollen de database uit in dezelfde stack als de VM en het netwerk.

Diagram

Description automatically generated

Rol dus een nieuwe database uit: een Aurora MySQL Serverless cluster, met een database subnet group. Schrijf ook een nieuwe security group specifiek voor de database (sta inbound poort 3306 toe en outbound alles).

Volg eerst zelf vanuit de GUI de schermen om zelf een database te maken. Op de volgende pagina staat bovenaan weer een hint als je hier niet uit komt.

**Hint-1:** Aurora is onderdeel van de AWS RDS Service (RDS = Relational Database Service).

**Hint-2:** Je kunt bij het aanmaken vanuit de GUI een selectie opgeven voor Serverless. Dat is wat we willen…

**Hint-3:** Wanneer je verzuipt in het aantal opties: we maken een DBCluster aan, dit is wat we nodig hebben is:

* Een verbinding met onze eigen VPC
* Een nieuwe, door onszelf geschreven, subnet group
* Een door onszelf geschreven security group
* Gebruik password authentication (met een master user-id dat je via je template meegeeft en ook een master password dat je via je template meegeeft)
* Capacity range: deze mag uiteenlopen van 1 tot 128, voor ons is minimum 1 en maximum 16 voldoende. Het mooie van Aurora Serverless is dat je hem automatisch kunt stoppen als er geen gebruik meer van de database gemaakt wordt (dit heet AutoPause). Stel dit in op bijvoorbeeld 300 seconden (5 minuten).
* Backup retention period van 7 dagen is prima.
* Maak een nieuwe database aan met de naam wp\_database\_frederique (vervang frederique door de naam die je in je template meegeeft)
* Geef je cluster een cluster identifier met je eigen naam erin
* Zet EnableHttpEndpoint op true.

**Oplossing:**

  WordPressDBSubnetGroup:

    Type: AWS::RDS::DBSubnetGroup

    Properties:

      DBSubnetGroupDescription: !Sub "wordpress-dbsubnetgroup-${Name}"

      DBSubnetGroupName: !Sub "wordpress-dbsubnetgroup-${Name}"

      SubnetIds:

        - !Ref PublicSubnetAZa

        - !Ref PublicSubnetAZb

      Tags:

        - Key: Name

          Value: !Sub "wordpress-dbsubnetgroup-${Name}"

  WordPressDatabaseCluster:

    Type: AWS::RDS::DBCluster

    DeletionPolicy: Delete

    UpdateReplacePolicy: Delete

    Properties:

      BackupRetentionPeriod: 7

      DatabaseName: !Sub "wp\_database\_${Name}"

      DBClusterIdentifier: !Sub "wordpress-databasecluster-${Name}"

      DBSubnetGroupName: !Ref WordPressDBSubnetGroup

      EnableHttpEndpoint: true

      Engine: aurora-mysql

      EngineMode: serverless

      MasterUsername: !Ref WordpressUserName

      MasterUserPassword: !Ref WordpressPassword

      ScalingConfiguration:

        AutoPause: true

        MinCapacity: 1

        MaxCapacity: 16

        SecondsUntilAutoPause: 300

      VpcSecurityGroupIds:

        - !Ref WordPressDBSecurityGroup

      Tags:

        - Key: Name

          Value: !Sub "wordpress-databasecluster-${Name}"

Let op: de WordPressDBSecurityGroup heb ik hierboven niet opgenomen (lijkt sterk op wat we al hadden met dan poort 3306 en CIDR block 10.0.0.0/16 (hetzelfde CIDR block als je VPC).

Je moet daarnaast de parameters nog aanpassen om de endpoint daar uit te halen. Pas ook de AWS::EC2::Instance aan:

export WORDPRESS\_DATABASE\_SERVER\_ENDPOINT="${WordPressDatabaseCluster.Endpoint.Address}"

Verwijder ook het aanmaken van de database, dat gebeurt nu door AWS, niet door je UserData script…

**Bonus opdrachten bij opdracht 3:**

* Wacht 5 minuten, ga daarna opnieuw naar de website (of klik op de link met de naam van de website linksboven, of op de link “Sample Page” rechtsboven op de website). Je merkt dat dit best lang duurt (meer dan 30 seconden) maar dat de website uiteindelijk wel reageert.
* Kijk binnen RDS bij de settings van je database cluster naar monitoring: kun je terug vinden waar de database op/terugschaalt qua aantal Serverless Database Capacity Units?

1. **Opdracht 4**

Maak gebruik van de files in de gitlab repository en kijk hoe je de metadata zo optimaal mogelijk kunt gebruiken.

Documentatie over metadata: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-init.html>

Als je metadata toegevoegd hebt, houd je in UserData nog het volgende over:

      UserData:

        "Fn::Base64": !Sub |

            #!/bin/bash

            yum update -y

            amazon-linux-extras install php8.0 -y

            /opt/aws/bin/cfn-init --verbose --stack ${AWS::StackName} --resource EC2WordPress --region ${AWS::Region}

            /opt/aws/bin/cfn-signal --exit-code $? --stack ${AWS::StackName} --resource EC2WordPress --region ${AWS::Region}

Hierbij zorgt cfn-init voor het verwerken van de metadata en cfn-signal voor het doorgeven van de status aan CloudFormation.

Je kunt de UserData zoals je die had gebruiken om de shell scripts te downloaden en te bekijken, je vindt de bestanden ook in de Gitlab repository onder “Voorbereiding”.

**Hints:**

* Als je een foutmelding vanuit cfn-signal krijgt dat de status in CloudFormation al bekend is, dan komt dit vermoedelijk doordat je geen CreationPolicy aan de EC2 hebt toegevoegd. Zie voor de oplossing verder hieronder.
* Als je je VM wilt laten draaien na een foutmelding om e.e.a. verder te onderzoeken, vervang dan --exit-code $? door –exit-code 0 . De stack zal dan succesvol zijn (ook als e.e.a. nog niet goed uitgerold is). Kijk in logfiles in /var/log wat er aan de hand zou kunnen zijn.
* De logfiles van cfn-init heten /var/log/cfn-init.log en /var/log/cfn-init-cmd.log. In deze laatste file vind je de stdout en stderr van de commando’s die uitgevoerd zijn.

**Oplossing**

cfn-signal en CreationPolicies

cfn-signal en CreationPolicies werken samen. In z’n meest simpele vorm ziet dit er als volgt uit:

  EC2WordPress:

    Type: AWS::EC2::Instance

    CreationPolicy:

      ResourceSignal:

        Timeout: PT15M

...

Properties:

      UserData:

        "Fn::Base64": !Sub |

            #!/bin/bash

...

            /opt/aws/bin/cfn-signal --exit-code $? --stack ${AWS::StackName} --resource EC2WordPress --region ${AWS::Region}

De CreationPolicy bepaalt dat CloudFormation maximaal 15 minuten gaat wachten (let op: dit is inclusief de 2 minuten die het duurt om sowieso een EC2 Instance te starten).

De exit code bepaalt of CloudFormation de uitrol van de Virtual Machine succesvol vindt, of niet. De naam van de --resource parameter moet gelijk zijn aan de naam van de EC2-resource in de template.

cfn-init en metadata

Je zult merken dat er best veel scripts gebruikt zijn en het “voelt” misschien onnatuurlijk om dit allemaal in één groot metadata blok te zetten. Toch is dat wel de bedoeling: het blok metadata komt er als volgt uit te zien:

    Metadata:

      'AWS::CloudFormation::Init':

        config:

          packages:

            yum:

              tar: []

              httpd: []

              mariadb: []

              php: []

              php-cli: []

              php-fpm: []

...

              php-bcmath: []

          files:

            '/opt/install/download\_wordpress\_in\_opt\_wordpress.sh':

              content: |

                #!/bin/bash

                mkdir -p /opt/wordpress

                cd /opt

                curl -O https://wordpress.org/latest.tar.gz

                tar -xzvf latest.tar.gz

              mode: 000500

            '/opt/install/change\_wordpress\_config\_file.sh':

              content: |

                #!/bin/bash

                cp wp-config-sample.php wp-config.php

                sed "s/database\_name\_here/$WORDPRESS\_DATABASE\_NAME/g" -i wp-config.php

                sed "s/username\_here/$WORDPRESS\_USERNAME/g" -i wp-config.php

                sed "s/password\_here/$WORDPRESS\_PASSWORD/g" -i wp-config.php

                sed "s/localhost/$WORDPRESS\_DATABASE\_SERVER\_ENDPOINT/g" -i wp-config.php

              mode: 000500

          commands:

            01-download-wordpress-in-opt-wordpress:

              command: /opt/install/download\_wordpress\_in\_opt\_wordpress.sh

              ignoreErrors: true

            02-change\_wordpress\_config\_file:

              cwd: /opt/wordpress

              command: /opt/install/change\_wordpress\_config\_file.sh

              env:

                WORDPRESS\_DATABASE\_NAME: !Sub "wp\_database\_${Name}"

                WORDPRESS\_USERNAME: !Ref WordpressUserName

                WORDPRESS\_PASSWORD: !Ref WordpressPassword

                WORDPRESS\_DATABASE\_SERVER\_ENDPOINT: !GetAtt WordPressDatabaseCluster.Endpoint.Address

              ignoreErrors: true

            03-copy-files-to-var-www-html:

              command: cp -p -r /opt/wordpress/\* $WWW\_DIR

              env:

                WWW\_DIR: "/var/www/html"

              ignoreErrors: true

          services:

            systemd:

              httpd:

                enabled: true

                ensureRunning: true

                packages:

                  yum:

                    - httpd

                    - php

                    - php-cli

                    - php-fpm

...

                    - php-bcmath

AWS zorgt er zelf voor dat de **commando’s** in de juiste volgorde uitgevoerd wordt (bijv: eerst de yum install, pas daarna de services opstarten). De commando’s worden opgestart in alfabetische volgorde, om dit extra duidelijk te maken heb ik ze genummerd. ignoreErrors betekent dat de VM verder gaat met het volgende commando als er iets fout gaat en dat ook cfn-signal een succesvolle uitrol terug zal geven. Voor debugging is dat prima, zodra alles werkt zoals het ontworpen is is het beter om dit eruit te halen.

Bij de **services** zie je de namen van de packages genoemd worden: dit zorgt ervoor dat de httpd service herstart wordt als een nieuwe versie van deze packages wordt uitgerold.