Real-life applicatie voor Azure Services Platform

Applicaties in de wolken - Deel 1

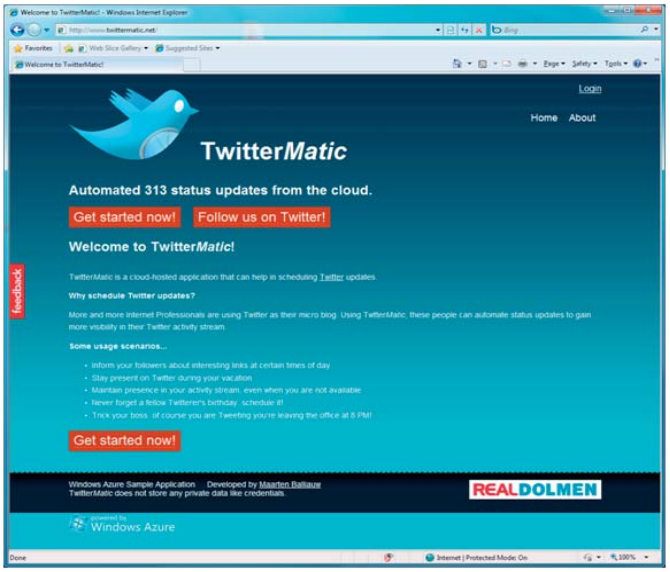
Maarten Balliauw

Het Windows Azure Services platform werd vorig jaar aangekondigd op de PDC in Los Angeles. Op de PDC van dit jaar werd het platform officieel losgelaten op het grote publiek, na een Community Technology Preview die ontwikkelaars gedurende een jaar hebben kunnen gebruiken om het platform te testen. In twee artikelen ga ik dieper in op het Azure Services Platform met het bouwen van een real-life applicatie voor het platform. Hier is deel 1.

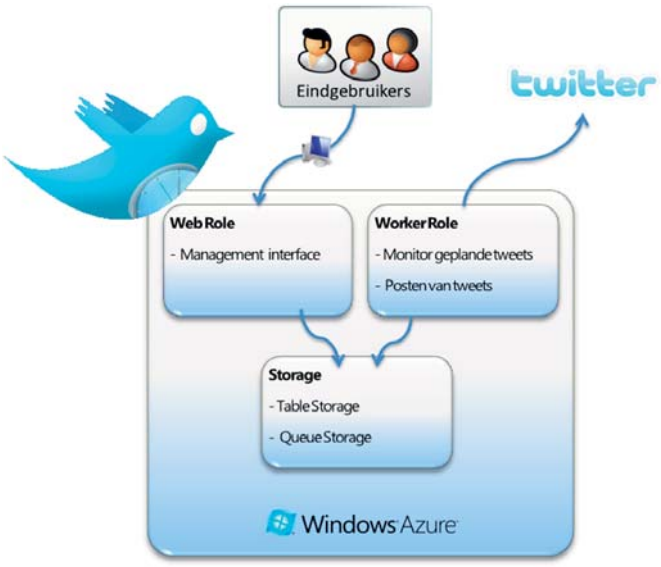
De applicatie die in deze reeks artikelen gebruikt wordt is TwitterMatic, een applicatie die ik bij RealDolmen het afgelopen jaar heb gebouwd voor de new CloudApp(); wedstrijd die werd uitgeschreven door Microsoft. Een eerste plaats hebben we hier niet gehaald, meedoen aan deze wedstrijd heeft wel een mooie Windows Azure voorbeeldapplicatie opgeleverd. TwitterMatic is een applicatie die gebruik maakt van de sociale netwerksite Twitter, waar je met een bericht van maximaal 140 tekens aan de wereld kan vertellen waarmee je bezig bent op dit moment. Dit kanaal wordt ook gebruikt door bedrijven om onder een groot publiek snel 'buzz' te genereren over nieuwe producten. Het is deze doelgroep die TwitterMatic wil bedienen: TwitterMatic laat toe om zogenaamde 'tweets' te gaan plannen en automatisch te posten op Twitter, zelfs als er niemand aanwezig is op kantoor. TwitterMatic kan je uiteraard ook gebruiken om een tweet te posten waarin je vermeld dat je overuren klopt, ook al hang je in de kroeg of zit je thuis in je luie stoel.

Architectuur

De architectuur voor TwitterMatic is relatief eenvoudig. Het hostinggedeelte van Windows Azure biedt ondersteuning voor een zogenaamde 'web role', waarin een webapplicatie kan worden gehost, en voor 'worker roles', waarin achtergrondprocessen gehost kunnen worden. Vergelijk de web role met IIS, de worker role met een Windows service die op de achtergrond draait. Binnen de web role is een ASP.NET MVC applicatie ontwikkeld waarin gebruikers tweets kunnen plannen en archiveren. De worker role krijgt binnen TwitterMatic de taak om geplande tweets te overlopen en indien nodig effectief naar Twitter te posten. Alle gegevens worden bewaard in twee opslagsystemen: Windows Azure Table Storage, een erg eenvoudige en gelimiteerde database, en de Windows Azure Queue Service, een gelimiteerde manier om berichten in een wachtrij te plaatsen voor verwerking.



OVERUREN KLOPPEN VANUIT JE LUIE STOEL?



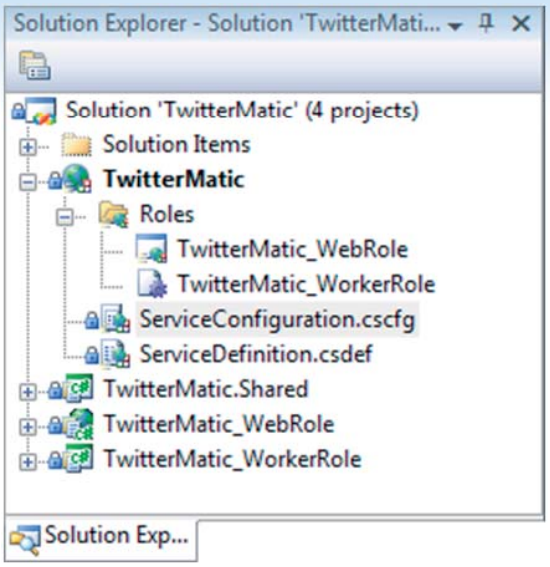
HET HOSTINGGEDEELTE BIEDT ONDERSTEUNING VOOR 'WEB' EN 'WORKER ROLES'.

Windows Azure features

Het is moeilijk om elke beschikbare feature van Windows Azure in een voorbeeldapplicatie te verwerken omdat de applicatie anders te uitgebreid wordt om te bespreken. Toch wil ik even kort de overige features en de nieuwigheden die dit jaar op de PDC 2009 werden voorgesteld toelichten. Allereerst werden de worker roles enorm uitgebreid: het is nu mogelijk om een worker role ook 'internet-facing' in te zetten en er bijvoorbeeld een MySQL of Apache server in te hosten. Zelfs JAVA hosten in Tomcat is perfect mogelijk! Daarnaast werden de logging en diagostische tools verder ontwikkeld en uitgebreid en komt voor SQL Azure het project met codenaam 'Vidalia' naar voren: Vidalia laat toe om confidentiële gegevens te bewaren in SQL Azure. De .NET Services werden hernoemd naar Windows Azure AppFabric waar nu de .NET Service Bus en de authentication services vallen. Nog niet live op PDC maar wel in de komende maanden zijn onder andere de 'X-drive', een virtual hard disk die je in alle roles kan mounten als gedeelde hard disk en de mogelijkheid om directe toegang te verkrijgen op een virtual machine via command-line en zelfs via remote desktop. De sessies rond Windows Azure kan je terugvinden op http://microsoftpdc.com/Sessions#/tags/WindowsAzure.

Bouwen van TwitterMatic

In deze editie van .NET magazine ga ik dieper in op hoe een Windows Azure applicatie binnen Visual Studio 2008 wordt opgezet. Daarna is het de beurt aan verschillende opslagsystemen in Windows Azure: ik laat zien hoe je gebruik maakt van Windows Azure Table Storage en de Windows Azure Queue Service. Om de visie van Microsoft in Software + Services te volgen, waarbij ervaringen worden gecreëerd over verschillende platformen, diensten en devices, wordt het authenticatiesysteem van TwitterMatic gewoon uitbesteed: aangezien we via Twitter de tweets zullen posten kunnen we net zo goed ook via Twitter inloggen.
In de volgende editie ga ik dieper in op wat zichtbaar is in TwitterMatic: de ASP.NET MVC applicatie die wordt gehost in een web role op Windows Azure, de worker role welke het posten van tweets verzorgt en uiteindelijk het deployment van TwitterMatic in de cloud, inclusief een eigen domeinnaam.
Opzetten van een Windows Azure project
Na het installeren van de Windows Azure Tools for Visual Studio is een nieuw project template beschikbaar binnen Visual Studio. Start Visual Studio 2008 met administrator rechten (rechts klikken onder Windows Vista en Windows 7, uitvoeren als administrator). Onder het menu File | New project… | Visual C# (of Visual Basic) | Cloud Service staat het project template Cloud Service te wachten op actie. Een nuttige wizard laat toe om aan deze Cloud Service meteen een web role en/of een worker role toe te voegen. Voor TwitterMatic is een ASP.NET MVC applicatie voorzien. De OK knop genereert nu een leeg Cloud Service project. Dit project is een 'manifest', een configuratiebestand waamee Windows Azure herkent welke roles er gestart moeten worden en met welke configuratie. Deze configuratie kan worden toegevoegd binnen de ServiceDefinition.csdef en ServiceConfiguration.cscfg bestanden, later hierover meer.
Binnen de solution heb ik nog een ASP.NET MVC project toegevoegd en 2 class libraries: een ervan is gedeelde code tussen web role en worker role, de tweede dient als worker role. Na het toevoegen van deze projecten binnen de solution moeten deze nog worden toegevoegd aan het Cloud Service project. Dit kan door rechts te klikken op de map Roles. De solution ziet er nu uit als volgt:



DE SOLUTION ITEMS VOOR TWITTERMATIC.

Wanneer dit project wordt gestart binnen Visual Studio wordt de Development Fabric gestart. Dit is een simulatie van de Windows Azure omgeving waarin 1 of meerdere web of worker roles uitgevoerd kunnen worden, een 'lokale cloud'. De Development Fabric is voor een Windows Azure applicatie wat de ASP.NET Development Server is voor een regulier ASP.NET project.

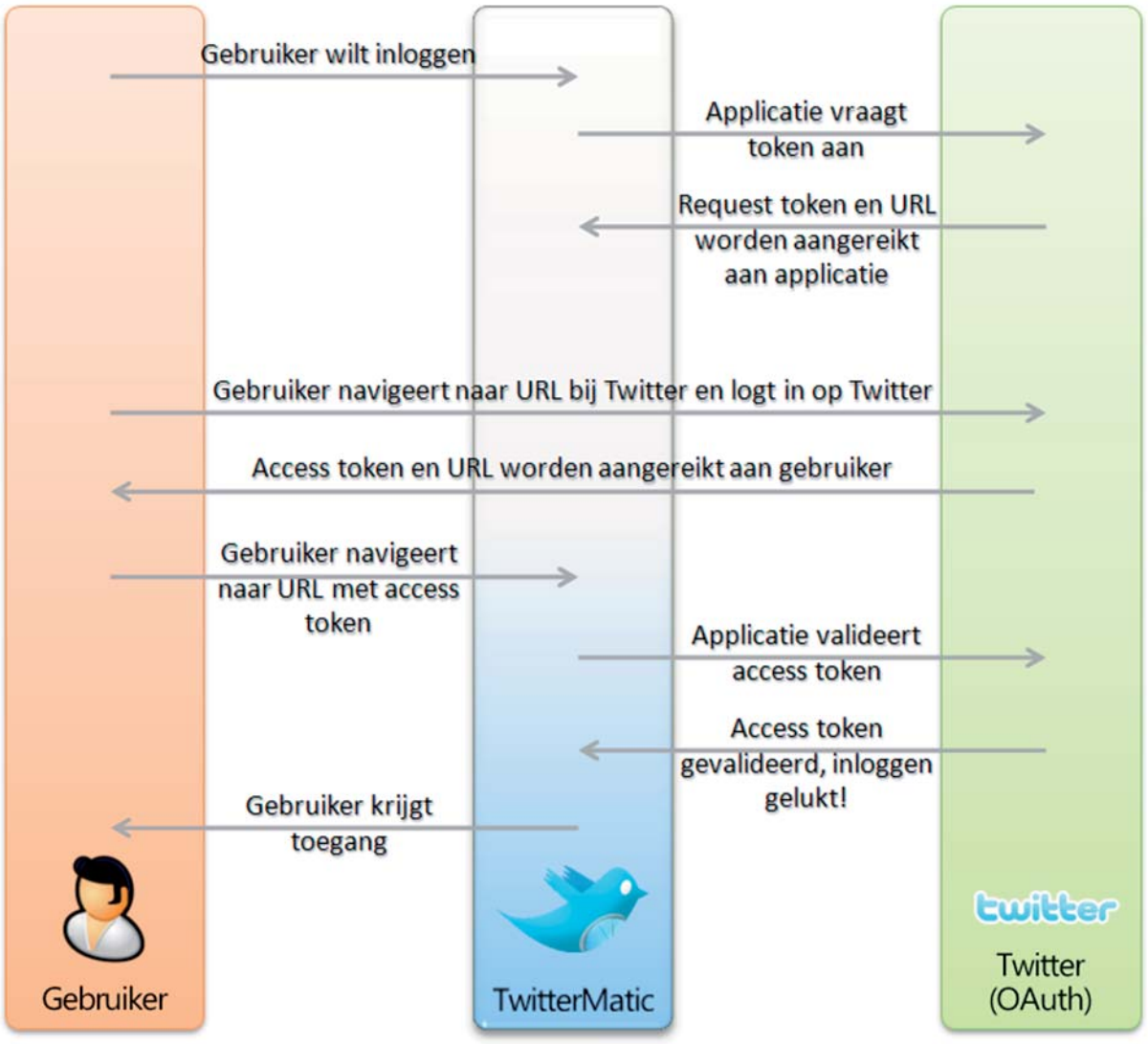
Gegevensopslag in de cloud

'Out-of-the-cloud' biedt Microsoft bij Windows Azure drie verschillende opslagdiensten aan, elk met hun eigen specifieke kenmerken: blob storage, table storage en de queue service. Blob storage is te vergelijken met een bestandssysteem. Table storage is een beperkte database: zaken als foreign keys, joins en grouping worden niet ondersteund, echter kan er wel sterk geschaald worden over verschillende locaties zonder enorme investeringen. Wie een meer geavanceerde database nodig heeft kan overigens altijd SQL Azure inzetten. De derde opslagdienst op Windows Azure is de queue service, waarin messages volgens een first-in-first-out principe in een wachtrij geplaatst worden. De queue service is voornamelijk bedoeld om taken aan te bieden aan een worker role.
De drie opslagdiensten worden aangeboden via REST: gegevens opslaan, opvragen en verwijderen gebeurt via eenvoudige HTTP instructies. Om gegevens op te slaan wordt het HTTP PUT of HTTP POST keyword gebruikt, om gegevens op te halen wordt het HTTP GET keyword gebruikt. Dit maakt dat de opslagdiensten die door Windows Azure worden aangeboden door elke programmeertaal gebruikt kunnen worden: de talen van het .NET platform maar ook Java, PHP, Ruby, Phyton en zelfs VB6! Als de taal HTTP requests kan uitvoeren, dan kan de taal gebruikt worden om gegevens te bewaren in de Windows Azure cloud.
Het gebruik van REST als protocol voor de Windows Azure opslagdiensten is niet toevallig: het kan enerzijds gebruikt worden in verschillende programmeertalen, anderzijds biedt het Microsoft de mogelijkheid om de opslagdienst oneindig te schalen. De interface blijft steeds dezelfde, de echte opslag in de cloud wordt redundant uitgevoerd, kan op verschillende locaties worden geplaatst.
Na het aanmaken van een storage account op http://windows.azure.com worden verschillende gegevens meegedeeld: een reeks storage endpoints, een account name en twee access keys. Een storage endpoint is de URL waarnaar REST aanvragen moeten

worden verstuurd, voor blob storage is deze http://blob.core.windows.net. Het account name is een zelfgekozen, logische naam voor het account op Windows Azure. De twee access keys zijn base-64 encoded strings, bijvoorbeeld "aHR0cDovL2Jsb2cub-WFhcnRlbmJhbGxpYXV3LmJlIC0gaHR0cDovL3R3aXR-0ZXIuY29tL21hYXJ0
ZW5iYWxsaWF1dw==", waarmee aanvragen van clients naar het Windows Azure storage platform worden ondertekend. Op die manier kan enkel de eigenaar van het storage account aanvragen uitvoeren via REST.
Wanneer een Windows Azure cloud service wordt ontwikkeld met behulp van de Windows Azure Samples (zie resources), dienen account name, account key en storage endpoints in een configuratiebestand opgenomen te worden. De StorageClient klassen in de samples, een .NET interface naar de REST endpoints, gebruiken de gegevens uit de configuratiebestanden om aanvragen uit te voeren op het Windows Azure storage platform. In de Visual Studio solution welke eerder werd aangemaakt zijn in het Cloud Service project twee bestanden terug te vinden: ServiceDefinition.csdef en ServiceConfiguration.cscfg. Onderstaand is een voorbeeld van het ServiceDefinition.csdef bestand:

```
<?xmlversion="1.0"encoding="utf-8"?>
<ServiceDefinitionname="TwitterMatic"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/
  ServiceDefinition">
  <WebRolename="TwitterMatic _ WebRole"enableNativeCodeExecution=
  "false">
    <InputEndpoints>
      <InputEndpointname="HttpIn"protocol="http"port="80" />
    </InputEndpoints>
    <ConfigurationSettings>
      <Settingname="AccountName" />
      <Settingname="AccountSharedKey" />
      <Settingname="BlobStorageEndpoint" />
      <Settingname="QueueStorageEndpoint" />
      <Settingname="TableStorageEndpoint" />
      <Settingname="allowInsecureRemoteEndpoints" />
    </ConfigurationSettings>
  </WebRole>
  <WorkerRolename="TwitterMatic _ WorkerRole"enableNativeCode-
  Execution="false">
    <ConfigurationSettings>
      <Settingname="AccountName" />
      <Settingname="AccountSharedKey" />
      <Settingname="BlobStorageEndpoint" />
      <Settingname="QueueStorageEndpoint" />
      <Settingname="TableStorageEndpoint" />
      <Settingname="allowInsecureRemoteEndpoints" />
    </ConfigurationSettings>
  </WorkerRole>
</ServiceDefinition>
```

Binnen dit bestand wordt vastgelegd welke instellingen er nodig zijn om een bepaalde role binnen de cloud service te kunnen starten. Bovenstaande configuratie geeft aan dat voor zowel web role



DE ROL VAN DE PRAATAUTOMAAT IN HET TWITTERPROCES.

als worker role het account name, account key en de verschillende endpoints vereist zijn. In het ServiceConfiguration.cscfg bestand worden de waarden voor deze instellingen geplaatst, bv. <Settingn ame="AccountName"value="twittermatic" />. Dit alles lijkt op het eerste zicht dubbel werk: alle instellingen komen immers twee keer voor! Toch is deze manier van werken veilig naar deployment toe: bij deployment op Windows Azure kan het .cscfg bestand nog worden aangepast, om bijvoorbeeld naar productie storage te verwijzen. Windows Azure controleert dan of alle instellingen die in het .csdef bestand werden gedefiniëerd wel degelijk werden toegewezen in het .cscfg bestand, en laat ook enkel een deployment toe als dit het geval is. Op die manier worden configuratiefouten bij deployment gereduceerd tot nul, als de juiste gegevens ook nog worden ingevoerd, tenminste.

Authenticatie

De visie van Microsoft op Software + Services wordt gekenmerkt door het combineren van nieuwe software en bestaande diensten. Een van deze diensten is Windows Live ID: het biedt de mogelijkheid om op een nieuwe applcatie in te loggen met een bestaand Windows Live ID. Erg handig, aangezien ik het zelf meer dan beu ben om meer dan honderd verschillende accounts te hebben op verschillende sites.
Omdat TwitterMatic toch gebruik maakt vanTwitter en Twitter ook zijn authenticatiemechanisme aanbiedt voor derde partijen onder de noemer OAuth, gebruikt TwitterMatic geen Windows Live ID maar gewoon het bestaande Twitter account van zijn gebruikers, zonder ooit rekening te hoeven houden met het vergeten van wachtwoorden en dat soort zaken.
OAuth is een protocol gebaseerd p het Open ID protocol, en werkt als volgt: wanneer een gebruiker wenst in te loggen op TwitterMatic, wordt op de achtergrond een token aangevraagd bij Twitter: "er komt zo dadelijk iemand inloggen, wat moet hij vertellen bij aankomst?". Twitter stuurt een token terug aan TwitterMatic, de gebruiker wordt naar Twitter doorgestuurd met dit token. Bij de site van Twitter logt de gebruiker in, waarna Twitter op zijn beurt een nieuw token meegeeft aan de gebruiker: "hier, geef die maar aan TwitterMatic als je daar weer bent".
Dit token wordt dan door TwitterMatic gebruikt om bij Twitter een extra controle uit te voeren: is deze gebruiker wel degelijk ingelogd?

CodePlex

Omdat ik zelf niet graag het wiel opnieuw uitvind, ben ik op CodePlex op zoek gegaan naar een library die deze manier van aanmelden kan ondersteunen. Het Linq-to-Twitter project (zie resources) biedt ondersteuning voor OAuth alsook voor het posten van tweets. Aangezien TwitterMatic beide features gebruikt, is Linq-To-Twitter een ideale kandidaat.
De volledige broncode voor TwitterMatic is te vinden op CodePlex, en bevat een APS.NET MVC controller waarmee OAuth op een makkelijke manier kan worden geïntegreerd in een ASP.NET MVC webapplicatie. Toch wil ik een stukje code laten zien waarin bovenstaande flow duidelijk te zien is:

```
public ActionResult Login(string oauth_token)
{
    IOAuthTwitter oAuthTwitter = new OAuthTwitter();
    oAuthTwitter.OAuthConsumerKey = configuration.ReadSetting
    ("OAuthConsumerKey");
    oAuthTwitter.OAuthConsumerSecret = configuration.ReadSetting
    ("OAuthConsumerSecret");
    if (string.IsNullOrEmpty(oauth_token)) {
        // Not authorized. Redirect to Twitter!
        string loginUrl = oAuthTwitter.AuthorizationLinkGet( ... );
        return Redirect(loginUrl);
    } else {
        // Should be authorized. Get the access token and secret.
        string userId = "";
        string screenName = "";

        oAuthTwitter.AccessTokenGet(oauth_token, configuration.
        ReadSetting("OAuthAccessTokenUrl"),
            out screenName, out userId);
        if (oAuthTwitter.OAuthTokenSecret.Length > 0)
        {
            // Store the user in membership of our own site
            // ...

            // All is well!
            FormsAuthentication.SetAuthCookie(screenName, true);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            // Not OK...
            return RedirectToAction("Login");
        }
    }
}
```

Conclusie

In dit artikel heb ik gefocussed op de verschillende componenten waarmee een Windows Azure applicatie als TwitterMatic gebouwd kan worden. De verschillende opslagdiensten werden toegelicht, alsook de features die dit jaar op de PDC werden voorgesteld aan het grote publiek.
In de volgende editie van .NET magazine ga ik aan de hand van TwitterMatic dieper in op de ontwikkeling zelf: hoe worden de opslagdiensten precies gebruikt, waar moet ik op letten bij ontwikkeling voor Windows Azure, en vooral: hoe krijg ik mijn applicatie in the cloud? Tot de volgende!

Referenties

http://www.azure.com
http://www.twittermatic.net
http://twittermatic.codeplex.com
http://blog.maartenballiauw.be
http://twitter.com/maartenballiauw
http://www.microsoft.com/downloads/details.aspx?FamilyID=6967ff37-813e-47c7-b987-889124b43abd&displaylang=en
http://www.microsoft.com/downloads/details.aspx?familyid=413E88F8-5966-4A83-B309-53B7B77EDF78&displaylang=en
http://code.msdn.microsoft.com/windowsazuresamples
http://linqtotwitter.codeplex.com
http://asp.net/mvc

Maarten Balliauw, is Software Engineer bij RealDolmen. Tevens is hij Microsoft MVP.