Porten webapplicaties in Microsoft Azure

Hoe gaat het in zijn werk?

Jonathan van der Molen en Tinus Pool

Cloudcomputing is een hot topic en veel bedrijven hebben interesse om hun webapplicaties naar een Cloud-omgeving te migreren. Dankzij Microsoft Azure is het mogelijk om zonder al te veel moeite een cloudapplicatie op te zetten.

In opdracht van Windesheim Hogeschool hebben we een onderzoek uitgevoerd over cloudcomputing in het algemeen en Microsoft Azure in het bijzonder. De opdracht omvatte het porten van een complexe webapplicatie. Daarbij zijn tijdens de voorbereiding verscheidene kleinere applicaties gemigreerd. Hierdoor is ons vrij snel duidelijk geworden dat er een vast traject is om te migreren. De invulling van het traject verschilt per applicatie, maar het proces is vrijwel altijd hetzelfde.

We behandelen de processen die moeten plaatsvinden om een webapplicatie te migreren (porten) naar een Azure-omgeving. Tenslotte worden er ook wat algemeen toepasbare adviezen gegeven over het uitvoeren van migraties.

Ter ondersteuning is hieronder een lijst van de begrippen die worden gebruikt in dit artikel:

- *Azure*: Framework van Microsoft, dat het mogelijk maakt om te ontwikkelen op de Cloud omgeving van Microsoft.
- *Azure Emulator*: Een applicatie die de cloud omgeving nabootst op de pc van de designer (zodat er los van de cloud kan worden ontwikkeld).
- *Web role*: Een module die alle benodigde onderdelen bevat om een website of collectie van webpagina's te kunnen tonen. Een Role is compleet zelfstandig en kan niet (direct) met andere Roles communiceren.
- *Worker role*: Een module die extra functionaliteit geeft aan een applicatie. Bijvoorbeeld een module die de onderliggende infrastructuur ondersteunt. Een Role is compleet zelfstandig en kan niet (direct) met andere Roles communiceren.
- *Blob (Binary large object)*: Een bestand op de cloud. Vergelijkbaar met bestanden op een pc.
- *CloudTable/Table*: Een tabel die informatie bevat. Vergelijkbaar met een database tabel.
- *SQLAzure*: Database systeem voor de Microsoft cloud.
- *Porting*: Ander woord voor het migratie proces dat beschreven wordt.

De voorbereiding

Voordat we beginnen met het porten van de applicatie naar Azure, moet een aantal zaken zijn geregeld:

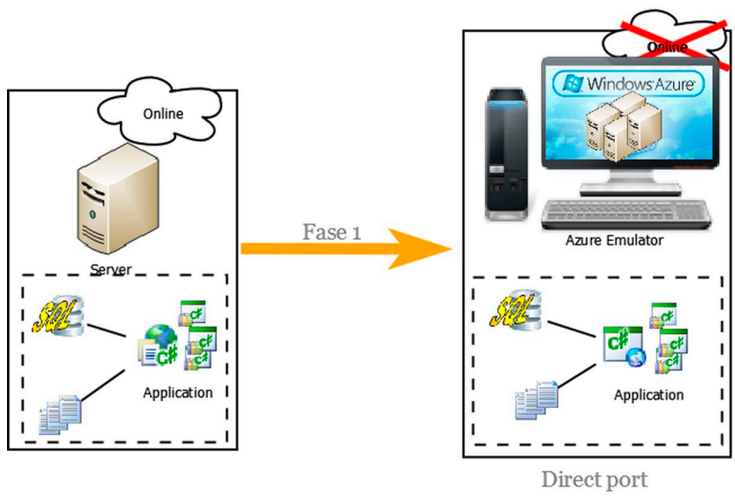
1. Er moet een geschikte Azure-ontwikkelomgeving zijn, die bestaat uit Visual Studio 2010 met 'Windows Azure Tools for Microsoft Visual Studio'.
2. Je moet weten hoe Azure met data omgaat en hoe je hier met code op inspeelt.
3. Ook is het handig om te kijken hoe Azure de kosten berekent en op den duur is er een Azure-account nodig.

Zodra de voorbereiding gedaan zijn, kan het porten beginnen. De port bestaat uit de volgende fasen:

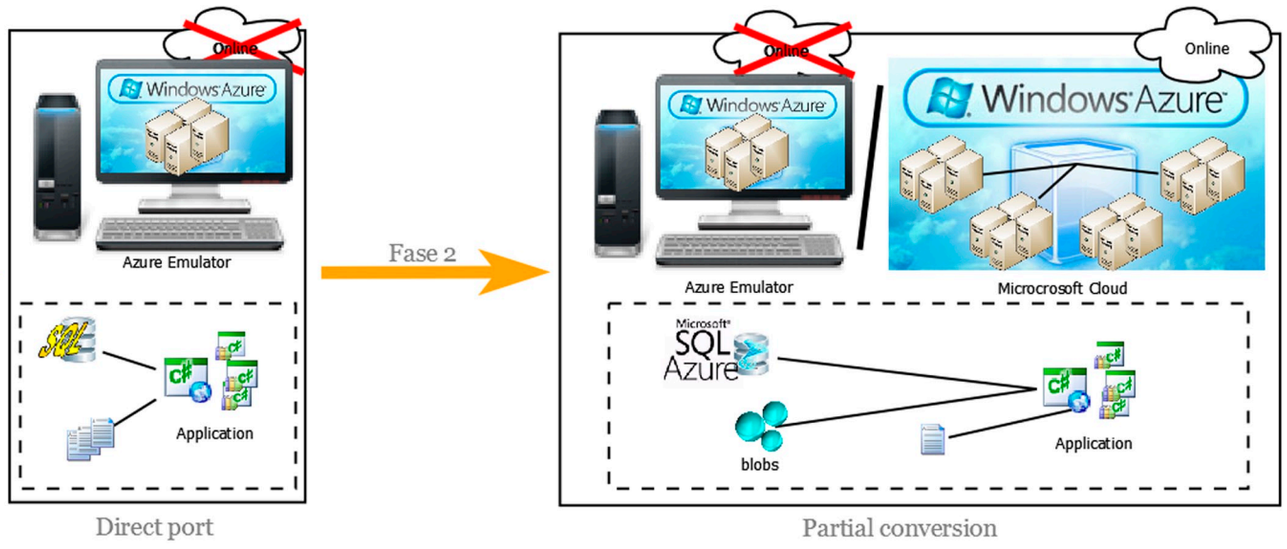
1. Direct port
2. Partial conversion
3. Complete conversion.

Fase 1: Direct port

Direct port wil zeggen dat het WPF/ADO basis platform vervangen wordt door een Azure-platform. De functionaliteit en werking van de applicatie blijft hierbij zoveel mogelijk hetzelfde. Wat we in deze fase alleen doen is de applicatie aanpassen, zodat zij op een Azure-emulator gelanceerd wordt. Hiermee zet je een begin op, waarop we in de volgende fasen verder kunnen bouwen. De emulator zelf bootst de cloud-omgeving na, zodat je lokaal



FIGUUR 1: FASE 1



FIGUUR 2: FASE 2

kunt testen en niet elke keer de applicatie naar de Azure-servers hoeft te uploaden. Zodra je de applicatie wilt publiceren kan deze dan makkelijk naar de Microsoft cloud worden gestuurd.

De emulator van Visual Studio biedt geen ondersteuning voor het hosten zelf en ook niet voor het lokaal hosten van SQL-Azure. Het bootst echter wel na hoe de Azure Cloud op de applicatie zal reageren en de SQL-Azure Database, die online op de Cloud draait, is hier wel op aan te sluiten.

Het resultaat dat deze fase oplevert kan beter nog niet op de Microsoft Cloud worden gehost. Het maakt niet optimaal gebruik van wat Azure te bieden heeft. Je maakt bijvoorbeeld nog geen gebruik van de dataopslagmethodes die Azure gebruikt en loopt daarmee het risico onnodig hoge kosten te maken.

Als het een niet al te uitgebreide applicatie is, kan de port met behulp van een tool in Visual Studio worden uitgevoerd. Deze tool voert de port geautomatiseerd uit. Deze tool is verre van perfect. Het systeem herkent geen XML-bestanden en deze worden niet mee geport en het systeem verliest ook de referenties naar andere projecten. Daarnaast is de port zelf ook niet perfect en kunnen er fouten ontstaan na gebruik van de tool. Bij kleinere systemen zijn de fouten snel op te lossen. Dit is geen goede oplossing voor een groter systeem. Hoe complexer het systeem, hoe groter de kans dat er iets fout gaat en dat het systeem niet meer werkt. Aan de hand van zijn complexiteit wordt het dus ook steeds moeilijker het systeem te repareren en er is een kleine kans dat het project zelf corrupt is.

Het alternatief op het geautomatiseerde porten van Visual Studio is om de port handmatig uit te voeren. Er is een solution waarin een applicatie staat. Verder bevat de solution ook een Azure Cloud service project en een Azure Web role project. De Web role zal zodanig worden aangepast, dat hij de exacte functionaliteit krijgt als de applicatie. Hiervoor zal er heel veel implementatie direct kunnen worden gekopieerd. Ook al zullen sommige dingen anders moeten worden geïmplementeerd door de manier waarop Azure werkt. De code van de Web role moet werkend te krijgen zijn, zoals die op de applicatie werkt. Databases of fileopslag komen later aan de beurt. Eerst moet de oude implementatie werken op de Azure Emulator. De Direct port is gelukt, als de Web role dezelfde functionaliteit en *look and feel* heeft. Dit kun je testen door de testcases van de oude applicatie erbij te pakken. Performance zou lokaal een issue kunnen zijn, maar bedenk wel dat de virtuele emulator ertussen zit die niet al te licht is.

Fase 2: Partial conversion

In de partial conversion wordt er gekeken naar de dataverwerking methodieken van Azure en hoe die het beste in de applicatie kunnen worden toegepast. Voor files uploaden en dowloaden kan er gekozen worden voor een Blob. Voor XML heeft Azure helaas geen functionaliteit ingebouwd, maar het is natuurlijk wel mogelijk om hier zelf een implementatie voor te schrijven. Tenslotte zijn er natuurlijk de databases.

Er zijn twee manieren waarop de database storage binnen de Cloud kan worden aangepakt.

- Een SQL database omzetten naar een SQLAzure database
- Een SQL database omzetten naar een CloudTable database

Werkt de applicatie met een relationele database, dan is er de mogelijkheid te kijken naar SQL Azure. In de meeste gevallen kan de huidige database worden geconverteerd naar een SQL Azure database en daarmee blijft de datastructuur en implementatie hetzelfde. Het is immers nog steeds SQL. Maar er zitten nadelen aan SQL Azure. Bijvoorbeeld de beperkingen op het gebied van de grootte. Een SQL Azure database mag maar 1 GB groot zijn. Als je bereid bent om extra te investeren is de maximale grootte 10 GB, maar dat is nog steeds weinig. Daar bovenop is er het probleem dat er maximaal 50 GB aan databases op de Cloud mag staan. Er is voor zover bekend geen manier om dit te vergroten buiten het aanschaffen van een extra account.

Een alternatief voor SQL Azure is CloudTable storage. Als je kiest voor Tables, bedenk dan dat een bestaande database compleet herbouwd moet worden. Daarnaast zijn Tables niet relationeel, een Table is helemaal zelfstandig. Ook hier kun je zelf een implementatie schrijven die Tables wel relationeel maakt, maar er is nog geen interne functionaliteit voor.
Tenslotte kan performance een issue zijn, omdat de queries gegenereerd worden op basis van LINQ, iets wat relatief traag werkt in vergelijking met SQL. Tables is een mooie techniek voor een

applicatie die van de grond af aan wordt opgebouwd. SQL Azure is de waarschijnlijke keuze bij het porten van een applicatie.

Buiten database storage is er ook de filestorage. Dit is vrij simpel te doen. Door middel van Blob storage kan complexe data, bijvoorbeeld een grote collectie van gegevens of een file, opgeslagen worden op de Cloud. Er is geen andere manier om grote bestanden op te slaan in het systeem. Sterker nog, in vergelijking met de andere vormen van datastorage kan storage een enorme hoeveelheid data opslaan. Eén enkel bestand mag wel 1TB groot worden!

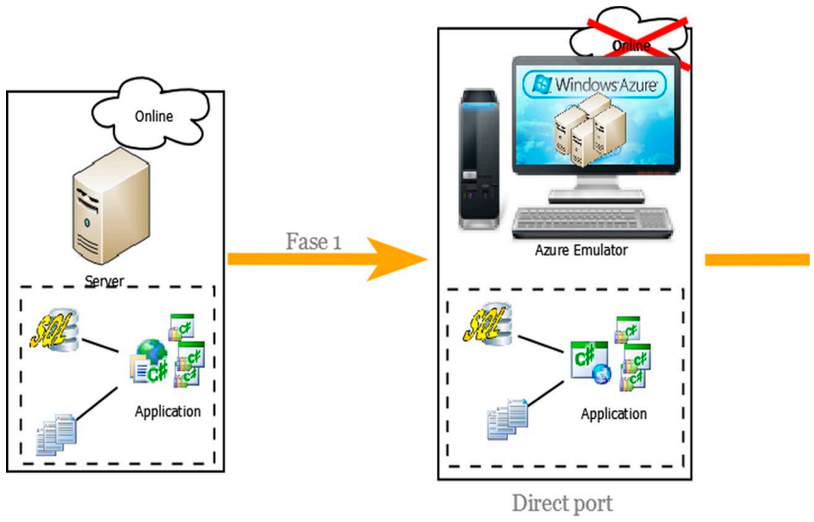
Fase 3: Complete conversion

Hier wordt het een stuk abstracter dan bij de andere fases. Wat er moet gebeuren is immers sterk afhankelijk van het systeem dat moet worden gemigreerd. Er moet dus eerst goed worden nagedacht over hoe het systeem werkt, voordat het systeem kan worden aangepast. Het opsplitsen van de applicatie naar Web/Worker rollen, is een manier waarop de voordelen van een Cloud kunnen worden benut.

Dit betekent niet dat de applicatie zo veel mogelijk moet worden opgesplitst. Er zijn zeker twee nadelen aan opsplitsen verbonden. Het eerste nadeel is dat rollen niet direct onderling of met elkaar kunnen communiceren. Het tweede is dat je dan per rol etxra zult moeten investeren. Hoe splits je een applicatie op?
Er zijn dus geen algemene regels voor en hoe dit het beste kan worden aangepakt verschilt per applicatie. Enkele zaken die in het algemeen bekeken kunnen worden zijn:

- De grootte van de applicatie. Hoeveel verantwoordelijkheden heeft het huidige systeem? Hoe groter en complexer een applicatie is, hoe makkelijker je deze kunt opdelen.
- Wat is voor iedere verantwoordelijkheid nodig om het te kunnen uitvoeren? Kan die verantwoordelijkheid opgesplitst worden? Is er veel overlap in de verantwoordelijkheden? Dataverkeer tussen modules is mogelijk maar is gelimiteerd, men zal dus de meeste benodigdheden lokaal moeten vervullen of via externe bronnen zoals databases.

Een mogelijk voorbeeld van het opsplitsen van verantwoordelijkheden is de spamfiltering van een mailserver. Een Software

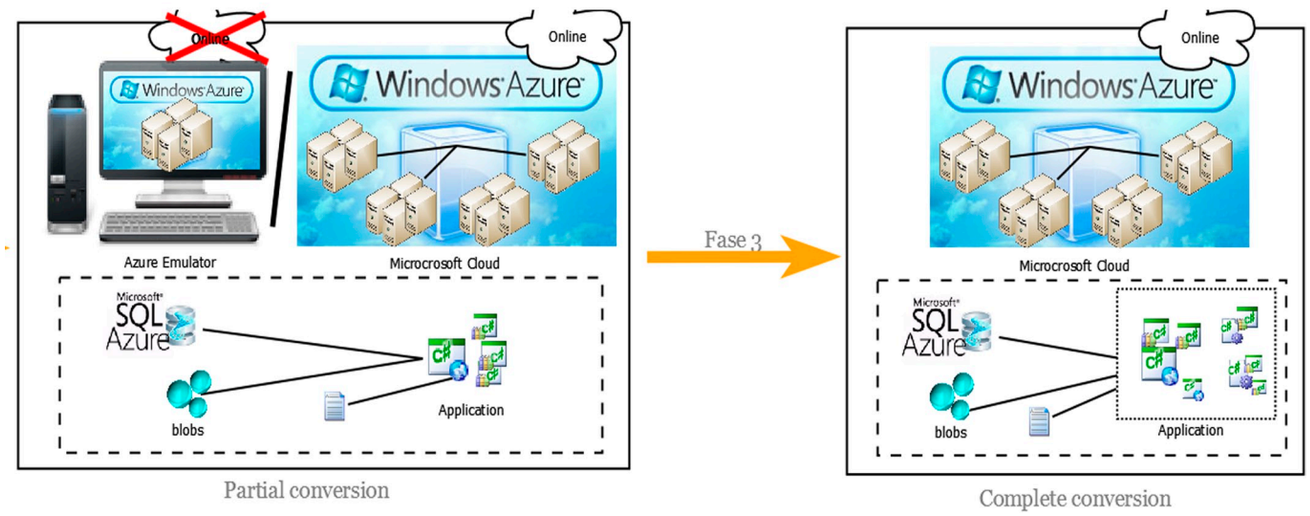


FIGUUR 4: TOTAAL BEELD

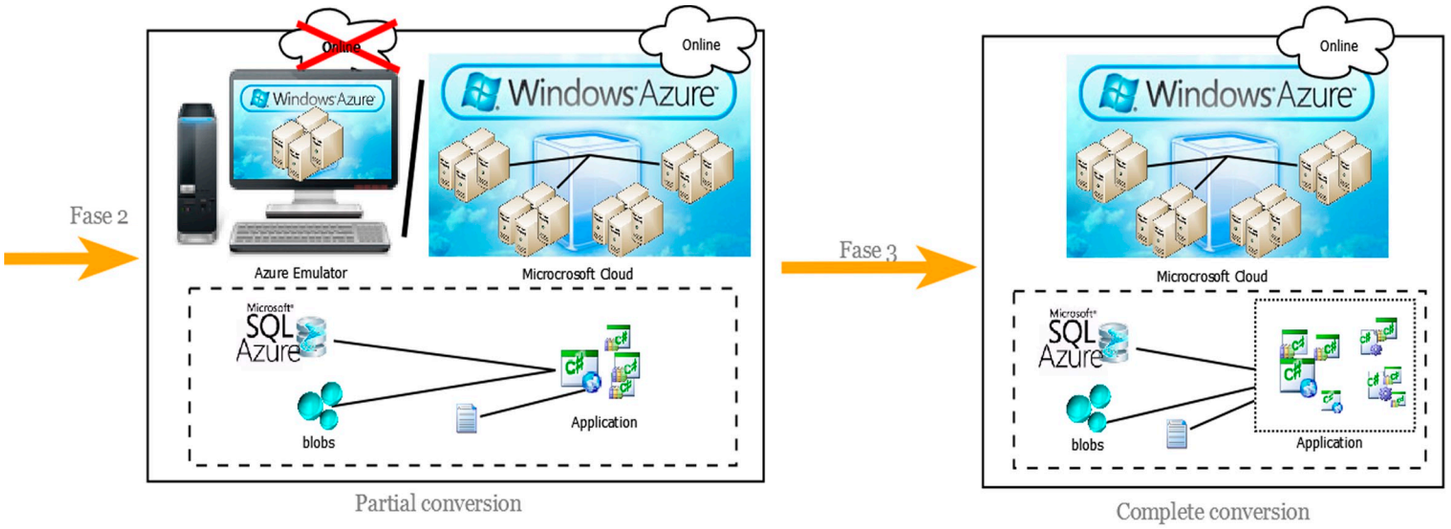
Architect van de KPN vertelde dat deze de mail onnodig vaak langs de spamfilter liet gaan. Bij het inkomen van de mail ging deze langs de spamfilter, dan ging hij ergens anders naar toe en daar vandaan ging hij weer langs de filter naar de mailbox van de gebruiker. Naarmate het mailverkeer toenam bij KPN moesten ze er steeds meer servers bijplaatsen, omdat het systeem het anders niet aan kon. Daarnaast ontwikkelden ze de spamfiltering ook mee en werd die intensiever. De Software Architect loste uiteindelijk het probleem op door met een team een aparte module voor de spamfilter te maken. De mail gaat nu alleen langs deze module bij het binnenkomen en uitgaan. Communicatie tussen mailserver en deze spamfilter-module loopt via messaging. Nog een voordeel voor KPN is dat ze deze module nu ook kunnen verhuren aan andere mailserver providers. Wel moeten we er bij zeggen dat deze mailserver niet op Azure draait.

Wel of niet migreren?

Nu de mogelijkheden van Cloud bekend zijn, is de vraag of het slim is om daadwerkelijk naar de Cloud te migreren. Om te beginnen moet je je afvragen wat je met de migratie wilt bereiken? Hoe snel wil je resultaat hebben? Hoe zit het met restricties op dataverkeer (bijvoorbeeld bij banken)?



FIGUUR 3: FASE 3



Fase 1: Directe port

Bij simpele systemen is er geen enkele reden om het systeem niet te porten. Je kunt dit met één druk op de knop laten uitvoeren en de applicatie kan nog steeds op een gewone server draaien.
Bij complexere systemen is dit onderdeel wat minder aantrekkelijk. Het zal snel handmatig moeten gebeuren en daar kunnen nogal wat manuren in gaan zitten. Wel een leuke uitdaging natuurlijk, maar het wordt pas aantrekkelijk als je weet dat de Cloud deze extra uren aan besparing zou kunnen teruggeven.

Het voornaamste voordeel van de directe port is dat het een makkelijke manier is om schaalbaarheid te implementeren (ook al is deze schaalbaarheid waarschijnlijk niet op zijn efficiëntst).

Een bezoeker merkt zelf niet of nauwelijks verschil tussen een site die Cloud of niet-Cloud is, zolang de site maar bereikbaar is en zijn privacy gewaarborgd blijft. Het is vooral de ontwikkelaar die het verschil merkt. Aangezien hij nu met een ander dataopslagsysteem werkt.

Fase 2: Partial conversion

Bij deze fase aangekomen is het sterk aan te raden om SLQ Azure te gaan gebruiken om een database te vervangen. Zo hoeft er het minst aangepast te worden aan de database architectuur die je gebruikt. Je kunt dezelfde query gebruiken en er zijn tools die de database in één keer converteren. Het gebruik van CloudTables is alleen een optie als de dataopslag opnieuw wordt opgezet.

Het grote nadeel van Cloudstorage is dat data altijd op de server in de Cloud staat. Er kunnen redenen zijn waarom dit niet gewenst is, bijvoorbeeld als het over gevoelig informatie gaat.

Fase 3: Complete conversion

Bij kleine en zelfs middelgrote applicaties zal het niet nodig zijn om een complete conversie uit te voeren. Sterker nog, het wel uitvoeren van de fase met dergelijke applicaties kan nadelige effecten hebben.
Je krijgt extra Web rollen waarvoor moet worden betaald. Bedenk eerst welke onderdelen in het systeem, van de niet geporte versie, verbeterd kunnen worden. Vaak is het pas bij een uitbreiding van het systeem handig om de complete conversie uit te voeren.

Links

http://www.dotnetmag.nl/Artikel/1198/Azure-Platform-een-productiecloud
http://msdn.microsoft.com/en-us/library/ff803364.aspx
http://www.microsoft.com/windowsazure/tools/
http://www.microsoft.com/windowsazure/offers/
http://msdn.microsoft.com/en-us/library/gg433135.aspx

Jonathan van der Molen en Tinus Pool, zijn derdejaars studenten Informatica aan Windesheim Hogeschool in Zwolle. Ze hebben een half jaar onderzoek gedaan naar cloudcomputing en Microsoft Azure.