Azure Automation vs. DevOps Release, 0-1

Automatische SQL database back-ups achter firewall

Automatische back-ups van een Azure SQL database maken doen we normaal gesproken met Azure Automation. Vanwege het probleem dat zich aandiende omdat het back-up proces via een ander IP adres loopt dan het IP adres van het automation proces, hebben we dit keer gekozen om het eens met behulp van Azure DevOps te proberen.

In Azure kun je een point-in-time restore doen van een database. Dit kan voor een bepaalde periode terug in de tijd. De lengte van de periode is afhankelijk van de gekozen pricing tier.

De wens was om verder terug te kunnen in de tijd en niet afhankelijk te zijn van de gekozen pricing tier. De oplossing die daarvoor is bedacht is om elke dag een back-up van de database te maken en periodiek de back-ups op te schonen.

Azure automation en het gebruik van 'runbooks' is daarvoor de aangewezen manier. Je kunt PowerShell scripts schrijven en deze invoegen in een runbook. Je kunt vervolgens parameters meegeven aan het runbook en deze scripts op een tijdschema laten uitvoeren. Waarbij de mogelijkheden bij het maken van een tijdschema (schedule) uitgebreid zijn.

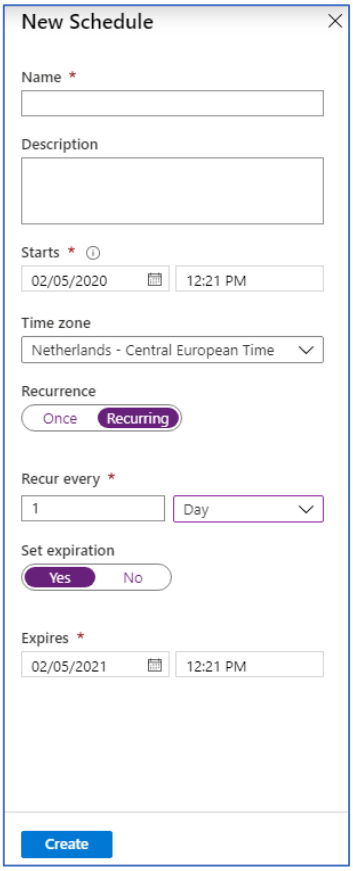

Figure 1 - Automation schedule

De automation resource en de runbook containers kunnen met een ARM template worden uitgerold. De PowerShell scripts kunnen door middel van een link in het ARM template worden aangegeven. Echter hebben we (in dit stadium van het project) ervoor gekozen om de inrichting van het runbook handmatig te doen.

Wat dan verder nog nodig is, is een PowerShell script en een 'Run as account' om binnen de Azure context het PowerShell script te mogen uitvoeren.

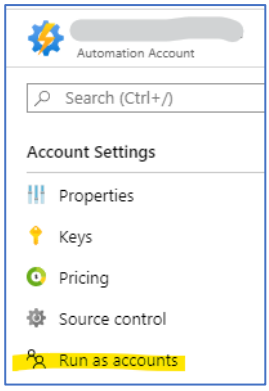

Figure 2 - Run as account

Kort samengevat is de procedure in het PowerShell script als volgt:

Gebruik het 'Run as account'
```
Get-AutomationConnection
Add-AzAccount
```

Vraag het huidige IP adres op
```
Invoke-RestMethod http://ipinfo.io/json | Select -exp ip
```

Voeg deze toe als rule aan de SQL firewall
```
New-AzSqlServerFirewallRule
```

Maak de database backup en plaats deze in blobstorage
```
New-AzSqlDatabaseExport
```

Verwijder de toegevoegde firewall rule
```
Remove-AzSqlServerFirewallRule
```

Omdat SQL server is afgeschermd door een firewall moet het IP adres worden toegevoegd aan de rules.

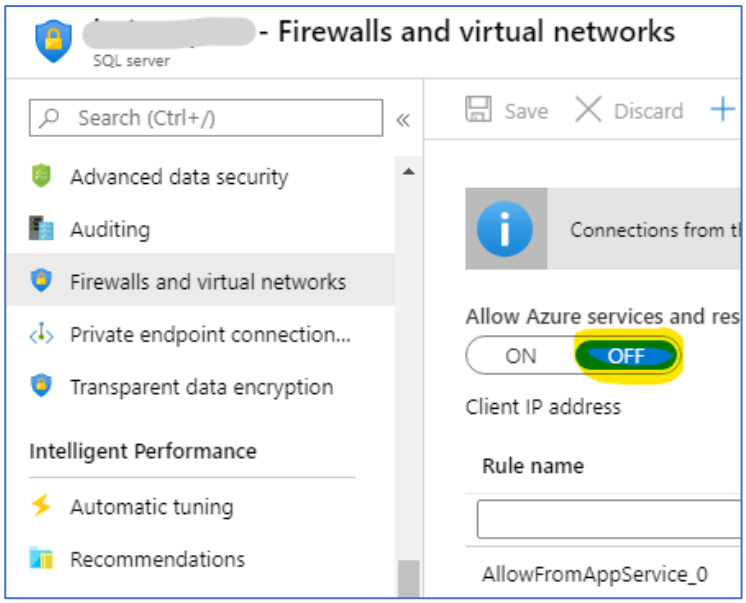

Figure 3 - SQL Firewall

Echter, wanneer het runbook wordt gestart, produceert het script een foutmelding:

```
0: There was an error that occurred during this operation : '<string
xmlns="http://schemas.microsoft.com/2003/10/Serialization/">Error encountered
during the service operation. ; Exception
Microsoft.SqlServer.Management.Dac.Services.ServiceException:Unable to
authenticate request; Inner exception System.Data.SqlClient.SqlException:Cannot
open server &amp;#39;[SQL resource naam]&amp;#39; requested by the login. Client
with IP address &amp;#39;[IP adres]&amp;#39; is not allowed to access the server.
To enable access, use the Windows Azure Management Portal or run
sp_set_firewall_rule on the master database to create a firewall rule for this IP
address or address range.  It may take up to five minutes for this change to take
effect.; </string>'
```

Het IP adres dat wordt genoemd in de foutmelding is niet gelijk aan het IP adres dat in het script is opgevraagd. En dus ongelijk aan het IP adres dat is ingesteld als firewall rule. Daarmee krijgen we geen toegang tot SQL-server en mogen we geen back-up maken.

Waar het IP adres uit de foutmelding vandaan komt is ons onbekend. Het IP adres is wel afkomstig van Microsoft. Wanneer dit IP adres wordt toegevoegd aan de firewall, werkt het script naar behoren. Ook hebben we gemerkt dat dit IP adres niet wijzigt.

Ter test hebben we besloten om de scripts vanuit Azure DevOps te draaien.

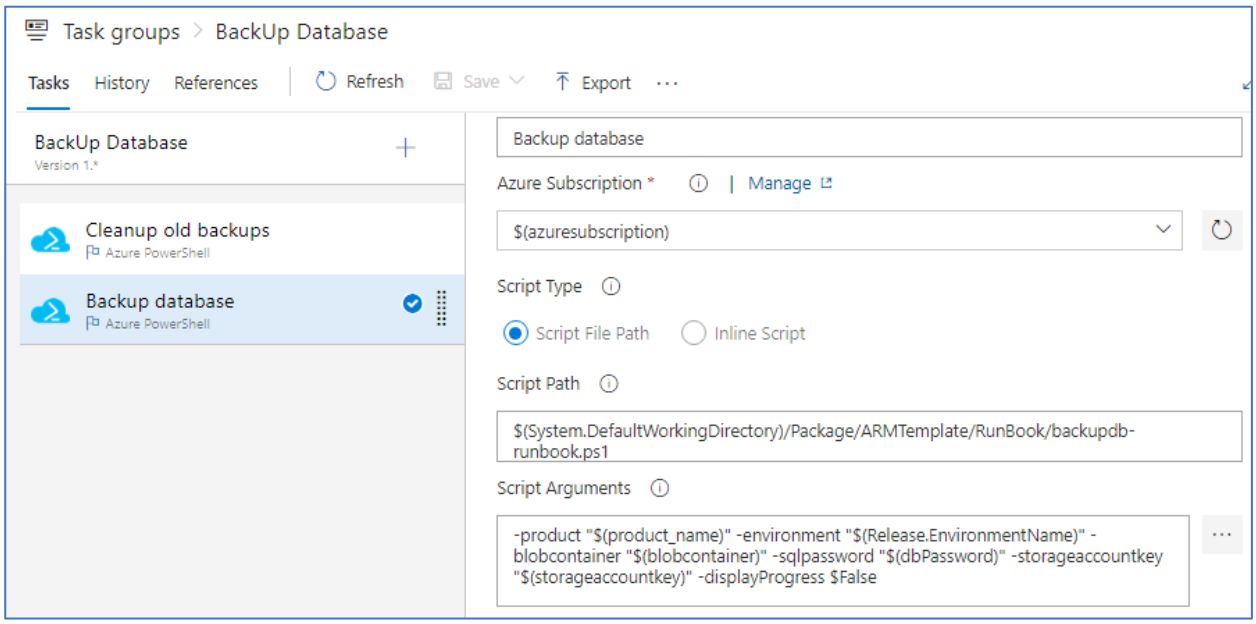

Figure 4 - Azure Powershell task in Azure DevOps taskgroup (for release)

Het resultaat was: exact dezelfde foutmelding, maar daarin hetzelfde IP adres als in de foutmelding vanuit het Azure Automation runbook.

Voor dit probleem zijn geen oplossingen gevonden, alleen work-arounds en non-oplossingen (Hybrid Worker). En het probleem geldt ook voor andere resources zoals keyvault.

De work-around waar wij voor hebben gekozen is om het IP adres uit de foutmelding als variabele op te nemen in de Azure DevOps release. De variabele wordt als parameter door gegeven aan de ‘Azure resource group deployment’ task. Op deze manier wordt het IP adres m.b.v. het ARM template toegevoegd aan de SQL firewall rules.

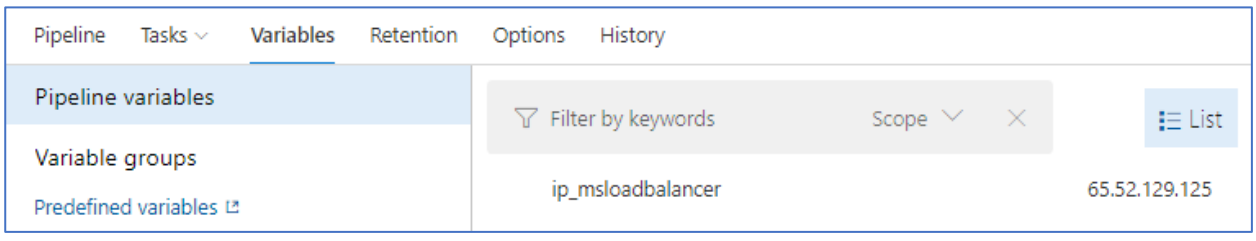

Figure 5 - Azure DevOps release variabele

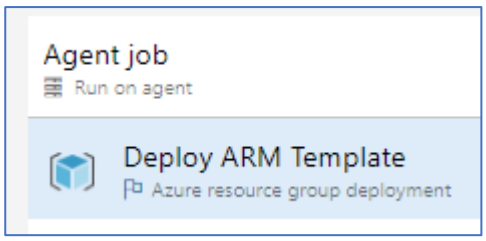

Figure 6 - Azure DevOps release task

ARM template snippet voor toevoegen van het IP adres aan de SQL firewall rules:

```
{
  "apiVersion": "2015-05-01-preview",
  "type": "Microsoft.Sql/servers/firewallRules",
  "comments": "Allow From DN Office",
  "name": "[concat(variables('sqlSettings').serverName, '/', 'AllowFromMSAutomation')]",
  "location": "[resourceGroup().location]",
  "properties": {
    "startIpAddress": "[variables('sqlSettings').fireWallRuleMSAutomationIP]",
    "endIpAddress": "[variables('sqlSettings').fireWallRuleMSAutomationIP]"
  },
  "dependsOn": [
    "[resourceId('Microsoft.Sql/servers', variables('sqlSettings').serverName)]"
  ]
}
```

Wanneer deze firewall rule is ingesteld kan het PowerShell back-up script goed functioneren. Het is niet een ideale situatie om dit IP adres als waarde in te voeren en permanent in de firewall rules te laten staan.

Een andere denkbare work-around -waarbij geen parameters nodig zijn- is een retry scenario. Probeer de database back-up te maken, filter vervolgens met PowerShell en een regular expression het IP adres uit de foutmelding, voeg dit IP adres tijdelijk toe aan de firewall rules en probeer daarna nogmaals de database back-up te creëren.

```
$regexPatternForIPAddress = "(\b(?:\d{1,3}\.){3}\d{1,3}\b)"
if($errorMessage -match $regexPatternForIPAddress){
    $clientIP = $Matches.0
    #add clientIP to firewall rules and try creating backup again
}
```

Of de scripts in Azure Automation of in Azure DevOps werden gedraaid: het maakte geen verschil. Toch hebben we dit keer gekozen om het in DevOps te laten staan om een aantal redenen.

- Ook in Azure DevOps kan de release ingepland worden om elke nacht uit te voeren
- De PowerShell scripts staan in source control, de DevOps release pakt automatisch de laatste versie van het script op
- Alle uitrol gerelateerde parameters en informatie staat bij elkaar op één plek
- Wanneer het IP adres wijzigt, faalt de nachtelijke back-up release. Hiervoor kunnen we een (gratis) mailnotificatie instellen naar het development team dat zich bezighoudt met dit project.