New backup to Azure feature in SQL Server 2014

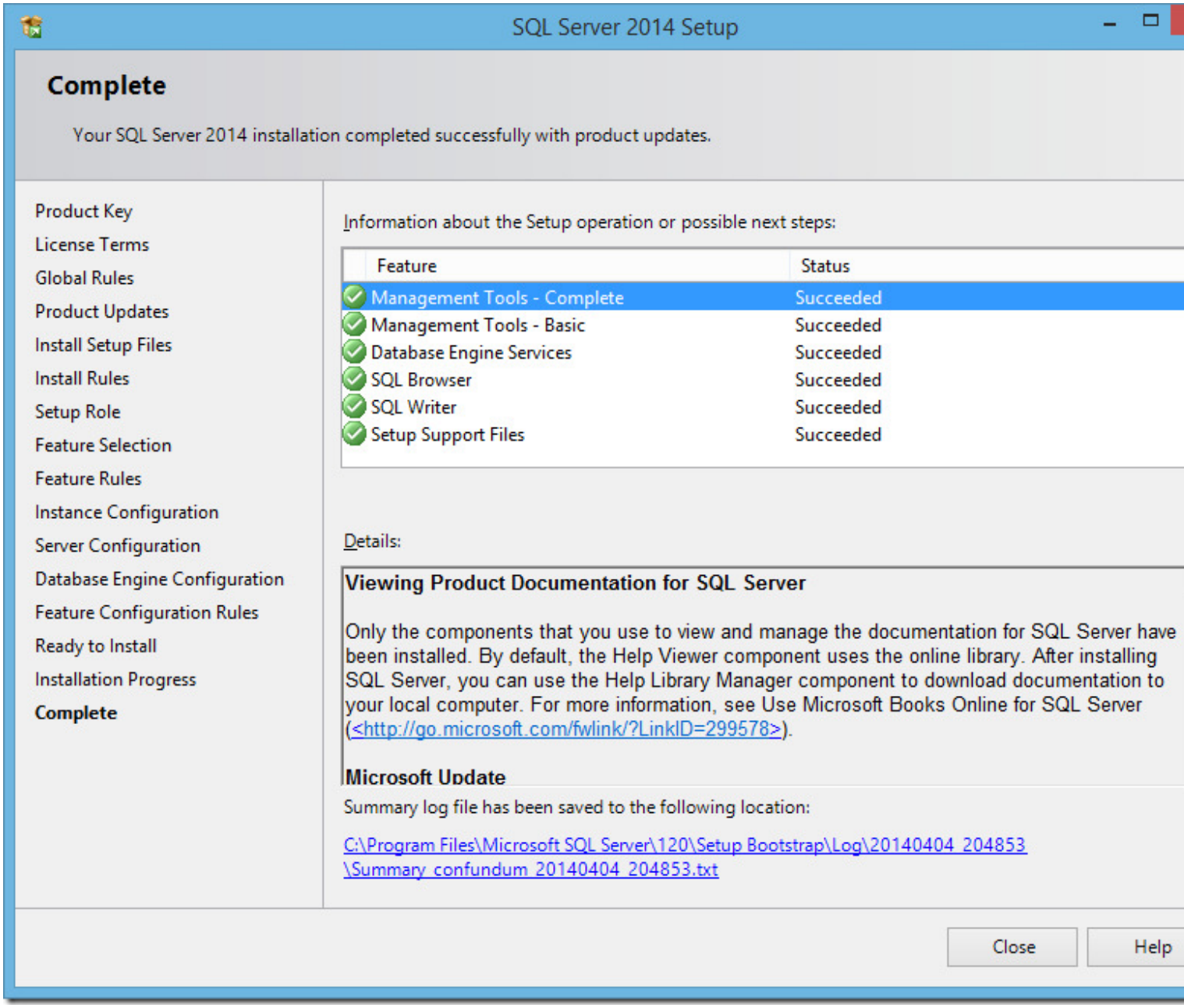
10TH OF APRIL, 2014 / CHRIS FULSTOW

SQL Server 2014 went RTM a couple of weeks ago and today it hits general availability for on-premises users via MSDN download, and for the cloud, where you can now spin one up as a VM from the Azure image gallery. So let's saddle up and take Microsoft's new Pegasi workhorse for a flying gallop through the hybrid cloud. Giddyup!

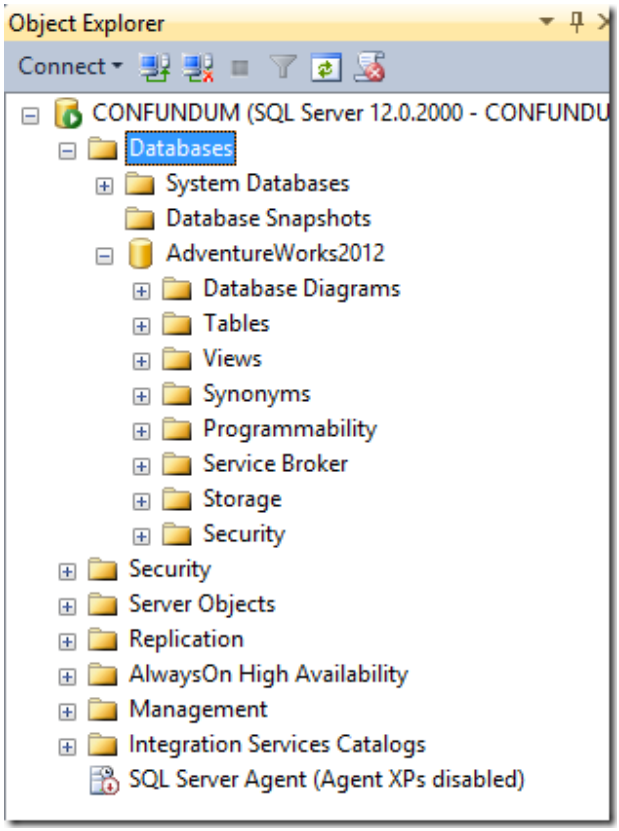
Installation

First to download the SQL Server 2014 Developer Edition (x64) from MSDN. It's a hefty 2.6 GB image, meaning it's time to put the new Kloud HQ 100 Mbps fibre link through its paces… four minutes later and we're done, nice. If you don't have access to an MSDN subscription there's also a fully featured 180 day evaluation version available for your convenience.

Installation of a default instance "on-premises" to my steadfast 4 GB MacBook Air is smooth, uneventful and very much like every other preceding version of SQL Server for the last 15 years.

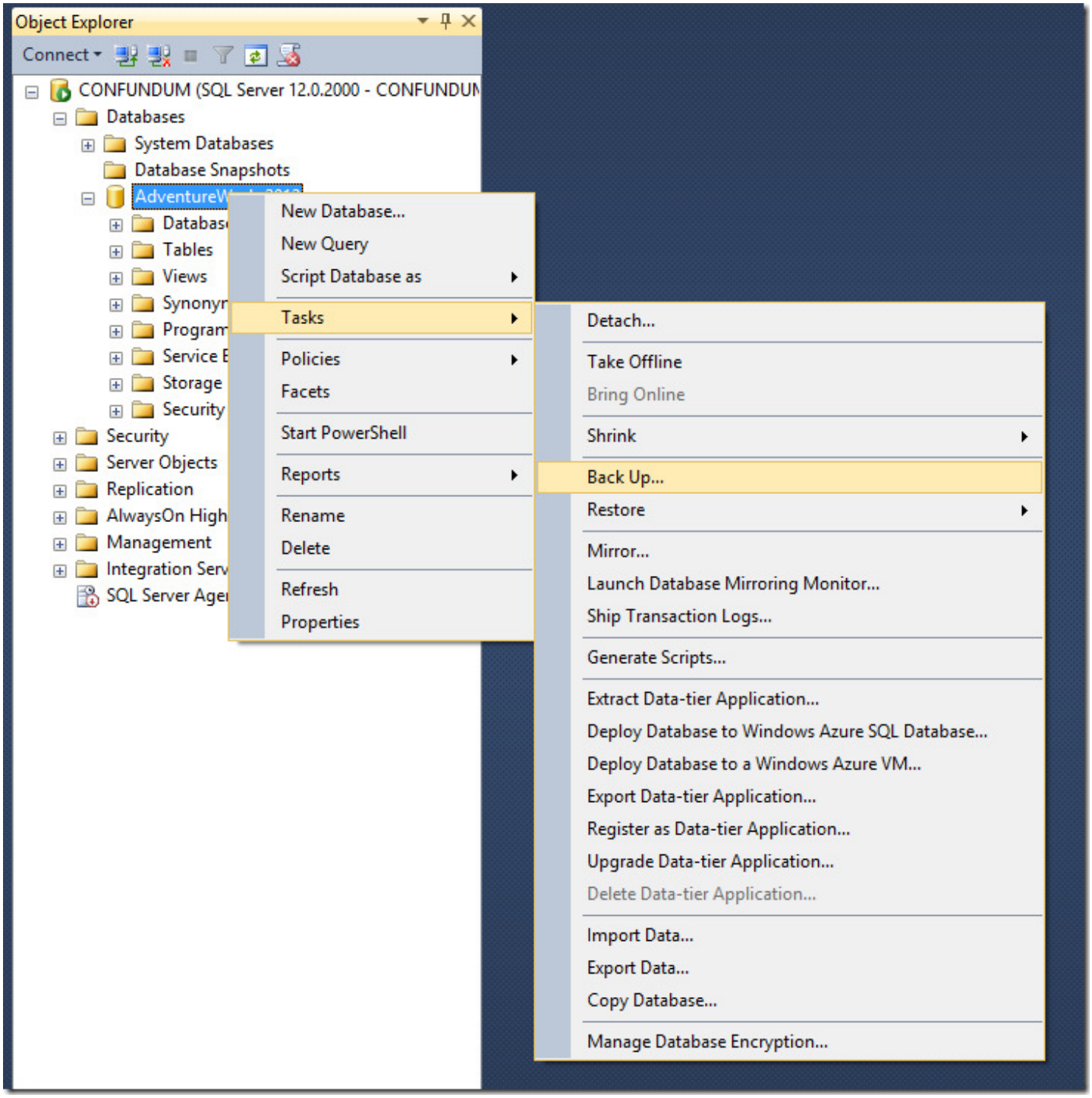


Now to fire up the new SQL Server Management Studio and connect to my shiny new instance. I'm going to need a database, so rather than spending time creating my own I'm going to download and attach the sample AdventureWorks database from Microsoft.

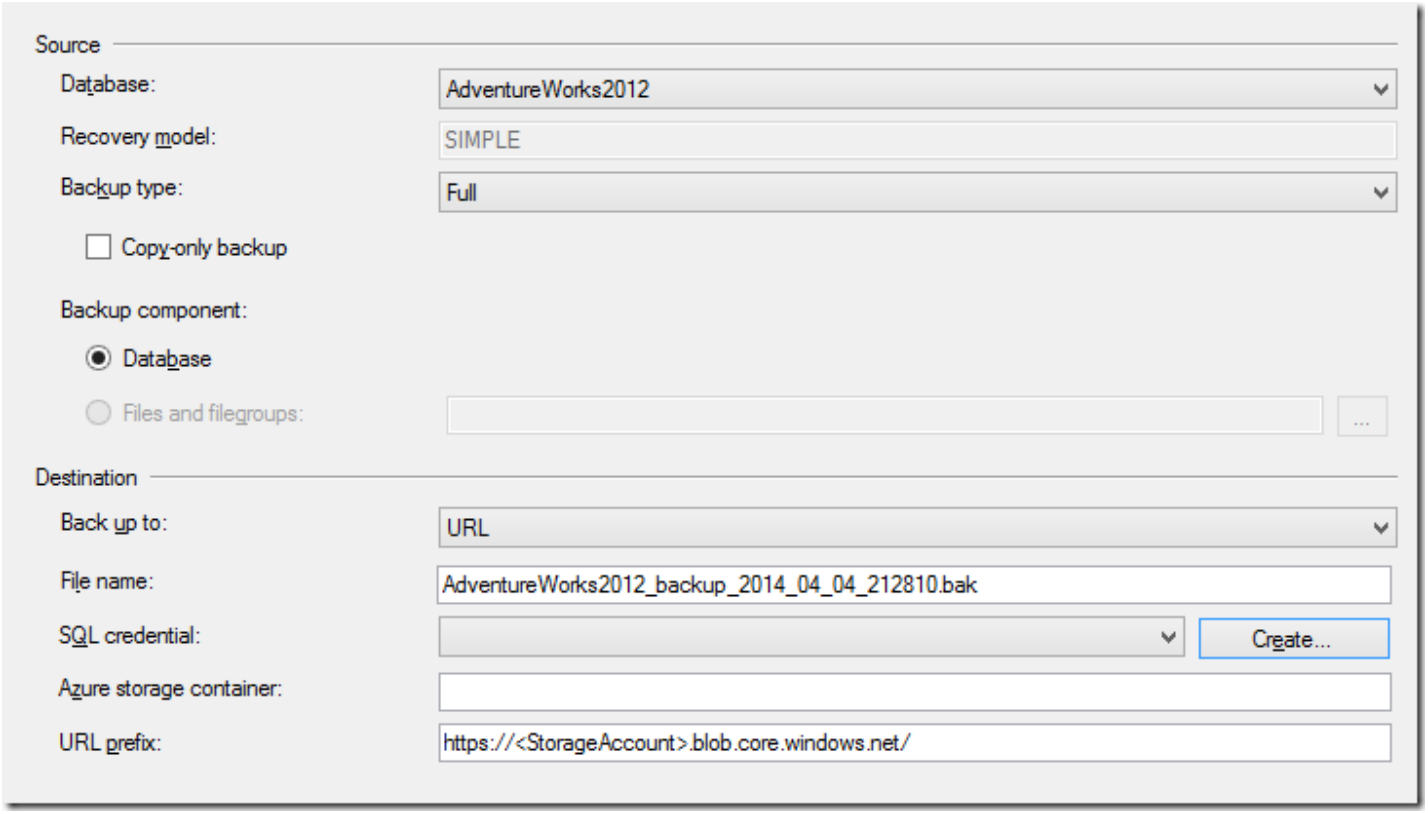


Backup to Azure Blob Storage

The first new cloud feature I want to try is backing up directly to Azure. I'm guessing this'll be an additional option in the Back Up Database dialog, so that's where I'll head. Already there're a couple of tantalising new options further down the context menu to deploy my database into the Azure SQL Database service or a virtual machine. I'll come back to explore these in a later blog post.



It's not immediately obvious where to select Azure as the backup destination, but expanding the *Back up to* dropdown reveals a new option for backing up to a URL. To me, URL feels like a strange hypernym given the address must point to an Azure storage blob endpoint – maybe a *Cloud* media type would've been more descriptive.



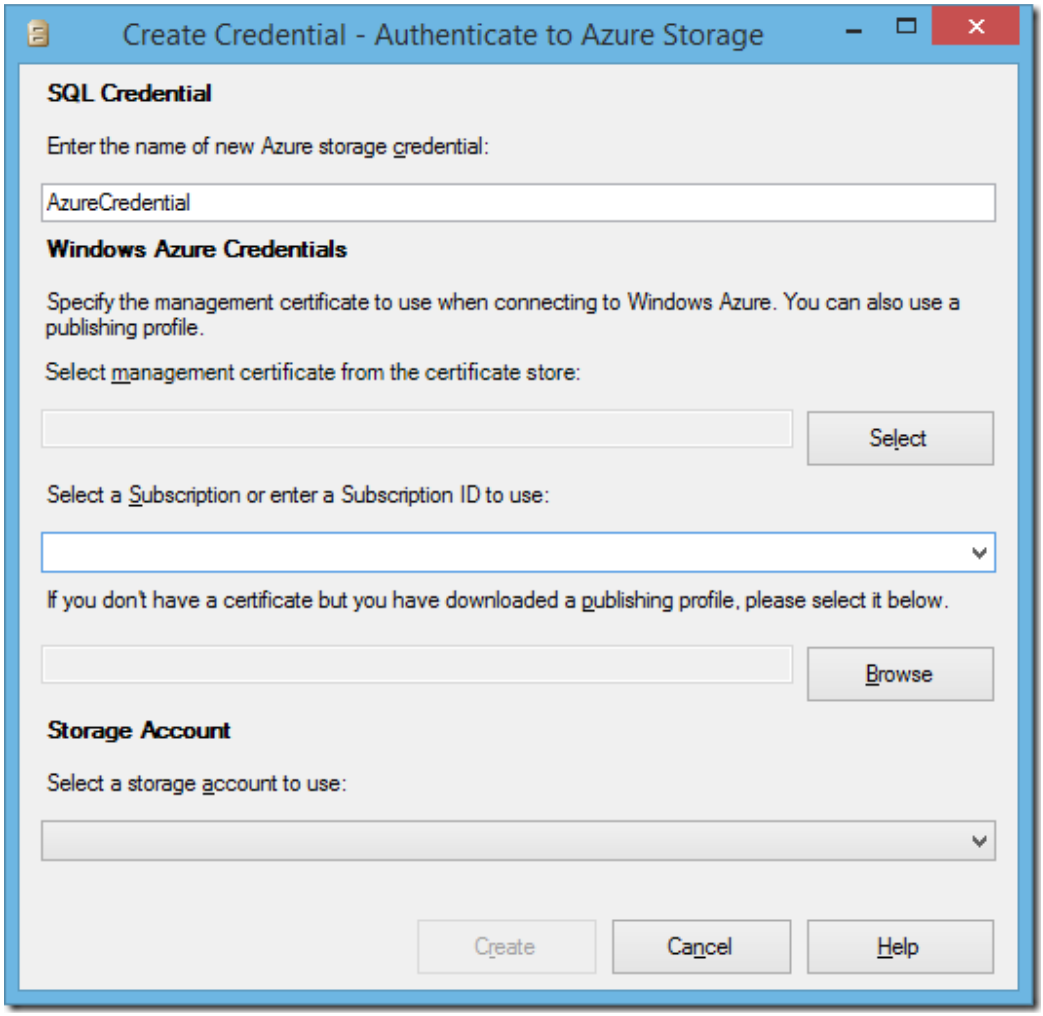
Before going any further at this stage, I'll need to create a new Azure storage account and blob container to hold my database backups. I could do this through the Azure management portal but PowerShell is quicker and more fun.

```
New-AzureStorageAccount `
     -StorageAccountName "sql2014kloud" `
     -Location "Southeast Asia"

Set-AzureSubscription `
     -SubscriptionName "Kloud Solutions" `
     -CurrentStorageAccount "sql2014kloud"

New-AzureStorageContainer -Name "sqlbackups"
```

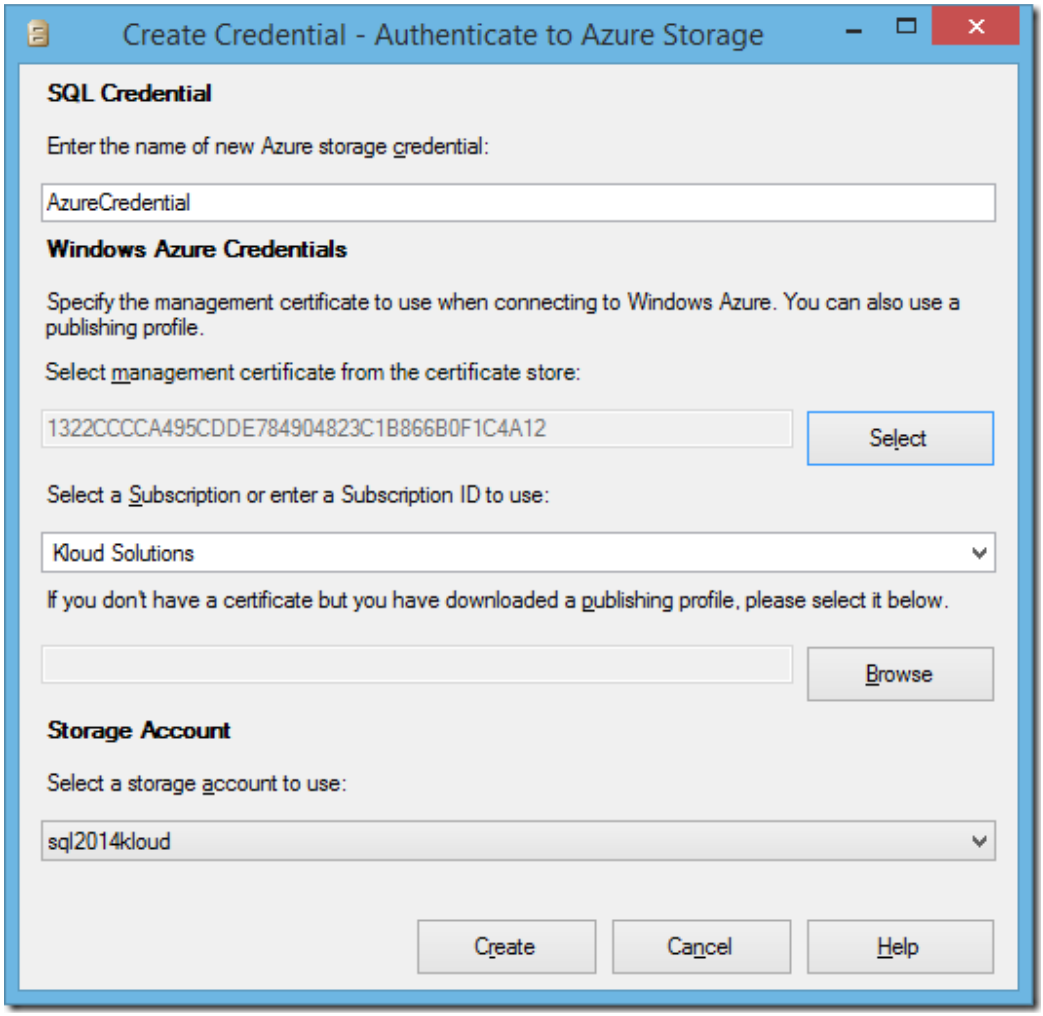
Now that's done, I'll continue configuring my Azure backup destination. I'll stick with the default backup filename, then move on to authentication where again the UI is slightly unintuitive. The *SQL credential* field refers to an object SQL Server uses to store authentication information for connecting to an external resource, in this case an Azure account. I'll click *Create* to set up a new one.



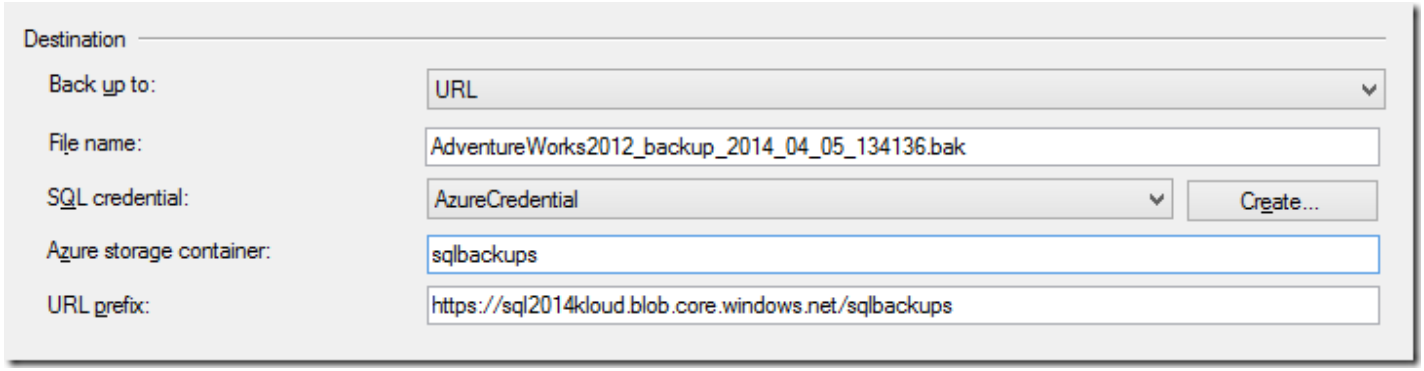
These credentials can be either an Azure management certificate or a storage access key, but the *Create Credential* dialog only offers the certificate options. Here, I can either create my own self-signed management certificate using makecert.exe and upload the public key to Azure. Or much easier, have Azure automatically create me a management certificate, then download it in a publishing profile, which is what I'll do.

To generate and download a publish profile settings, I'll go directly to the Azure Management Portal, but I could also invoke the Get-AzurePublishSettingsFile cmdlet to launch a browser to the same page. This prompts a file called *[Subscription].[Date]-credentials.publishsettings* to start downloading which contains my new management certificate and details of all the Azure subscriptions it's linked to. It's important to look after this file carefully because it's an unsecured key to all my Azure subscriptions.

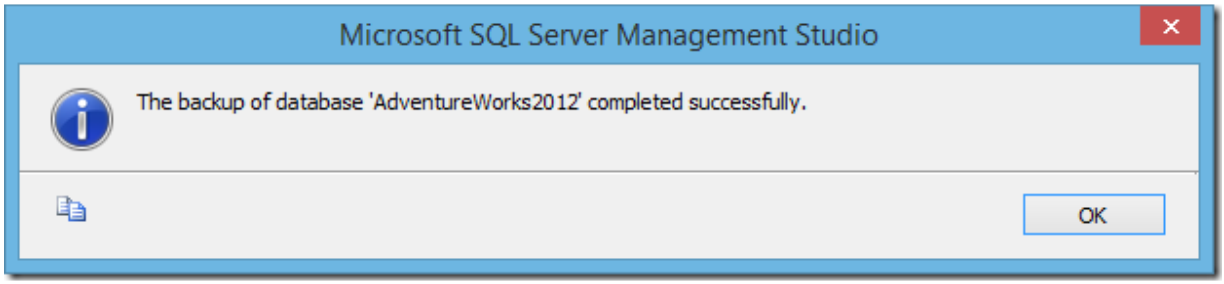
Next, I'll *Browse* to my publish settings file from the *Create Credential* window and select an Azure subscription from the dropdown. Now I can open the storage account dropdown, triggering SQL Server to verify my management certificate with Azure and retrieve a list of storage accounts, from which I'll choose the*sql2014kloud* account created above.



Lastly, I'll choose the *sqlbackups* storage container I created earlier, and we're good to go.



I'll click *OK* to start the backup, and a minute or so later it's finished.



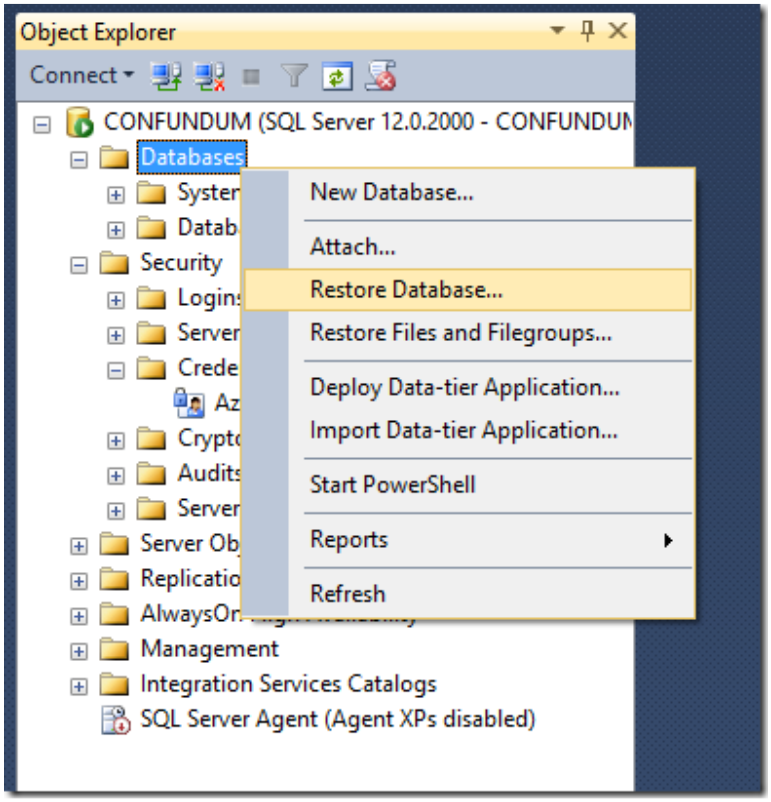
Restoring from Azure Blob Storage

To make sure the backup exists in blob storage, I'll use PowerShell to list the contents of my *sqlbackups* Azure blob conatiner.

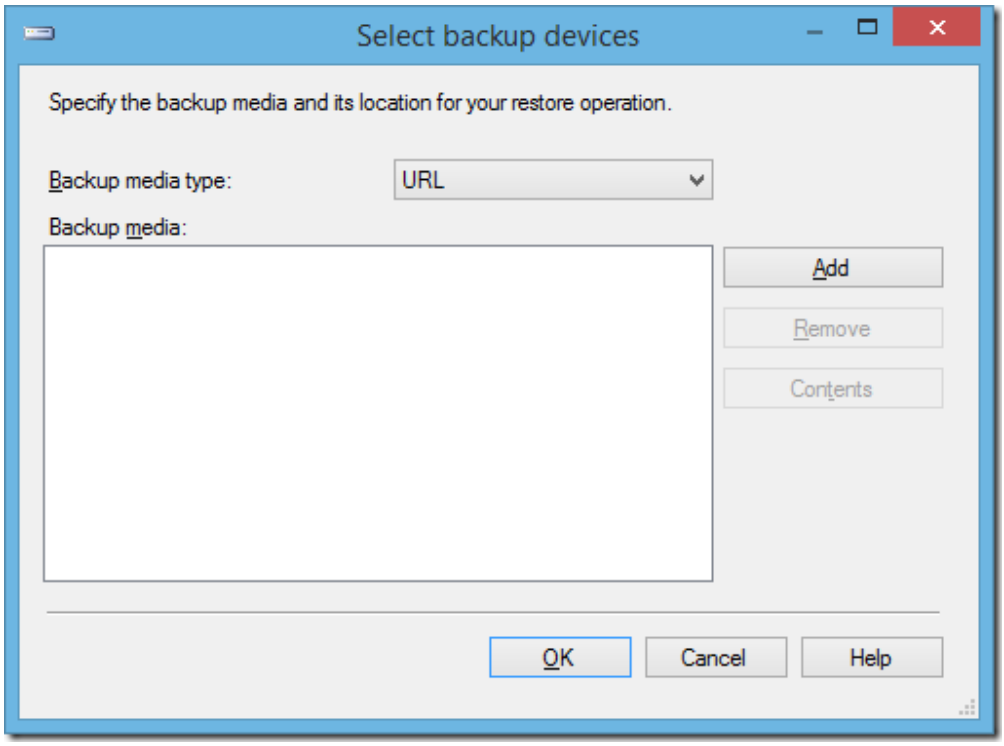
```
1    Get-AzureStorageBlob -Container "sqlbackups"
```

```
Name
--
AdventureWorks2012_backup_2014_04_05_134136.bak
```

And there it is, in my Azure storage account. To verify the backup worked properly, I'll drop the AdventureWorks database completely and attempt to restore it again from Azure.



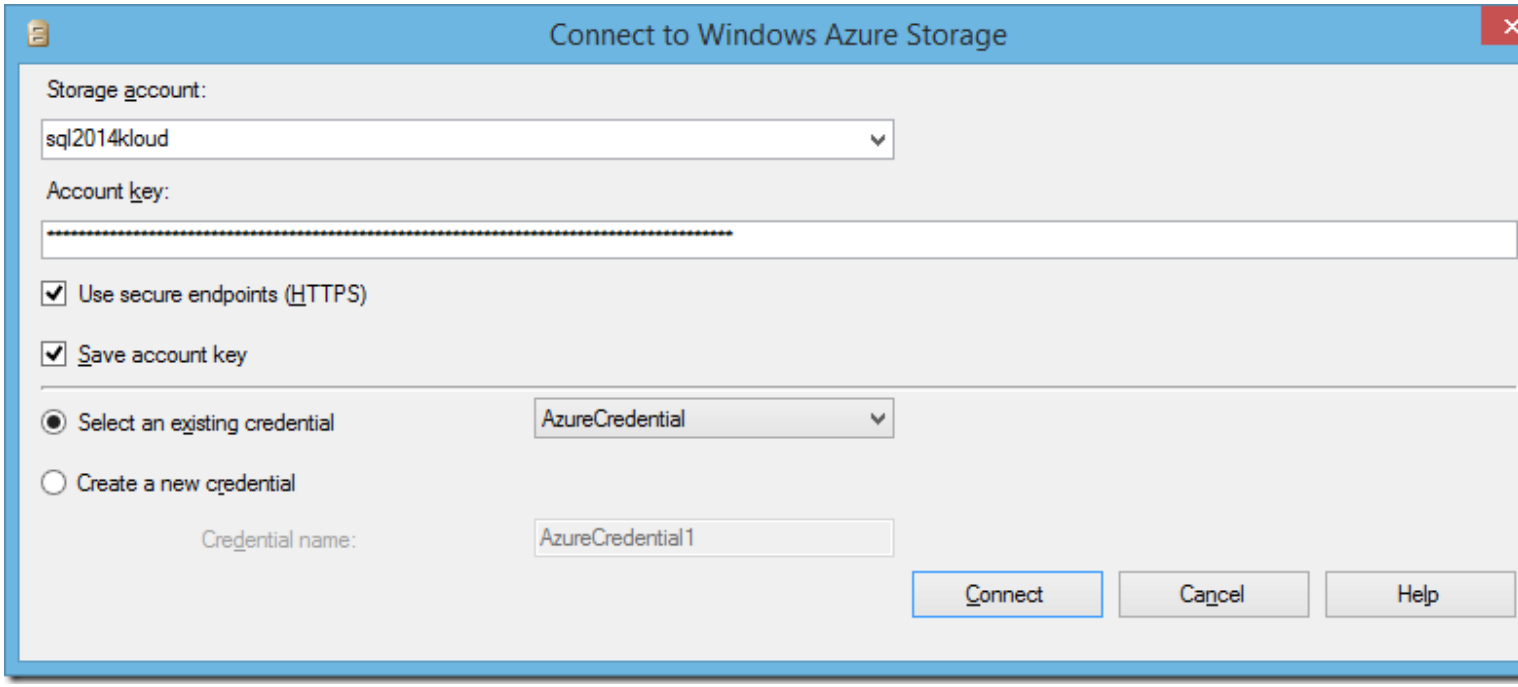
The Restore Database feature now has a new device media type of *URL*, which I'll select, then *Add* my Azure storage account:



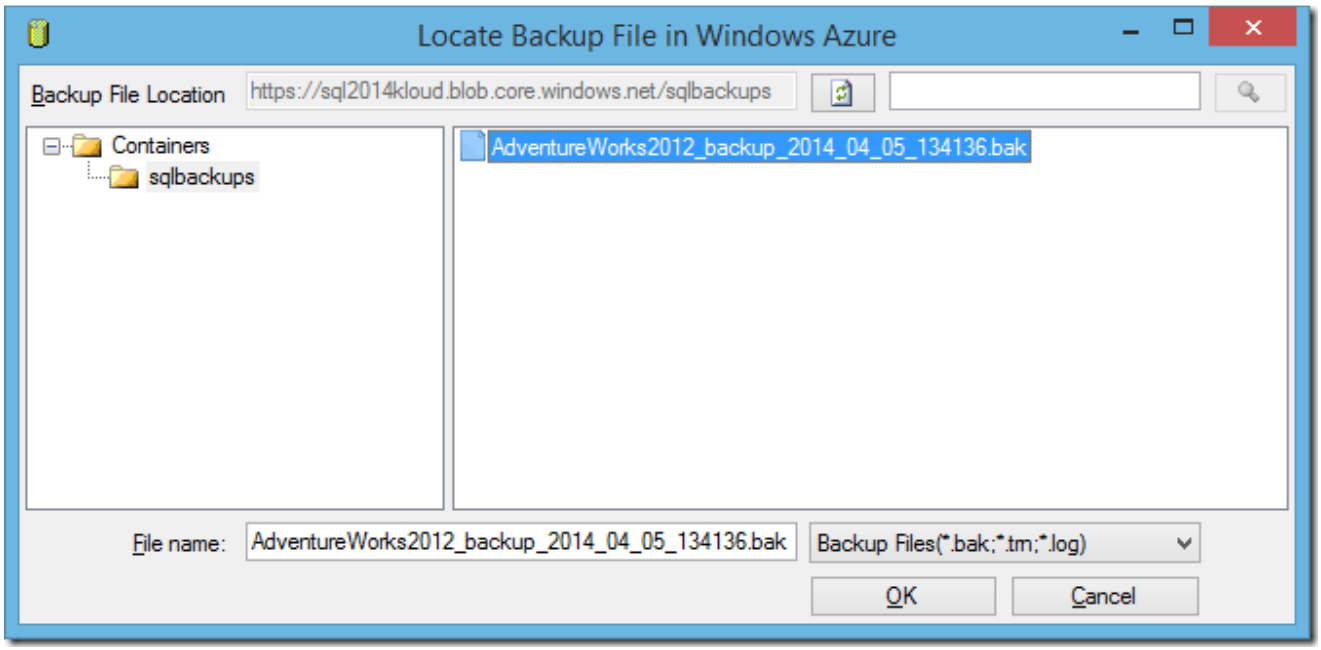
For this, I need my Azure storage access key, which is available from the Azure management portal or can be easily grabbed using PowerShell.

```
1    Get-AzureStorageKey -StorageAccountName "sql2014kloud"
```

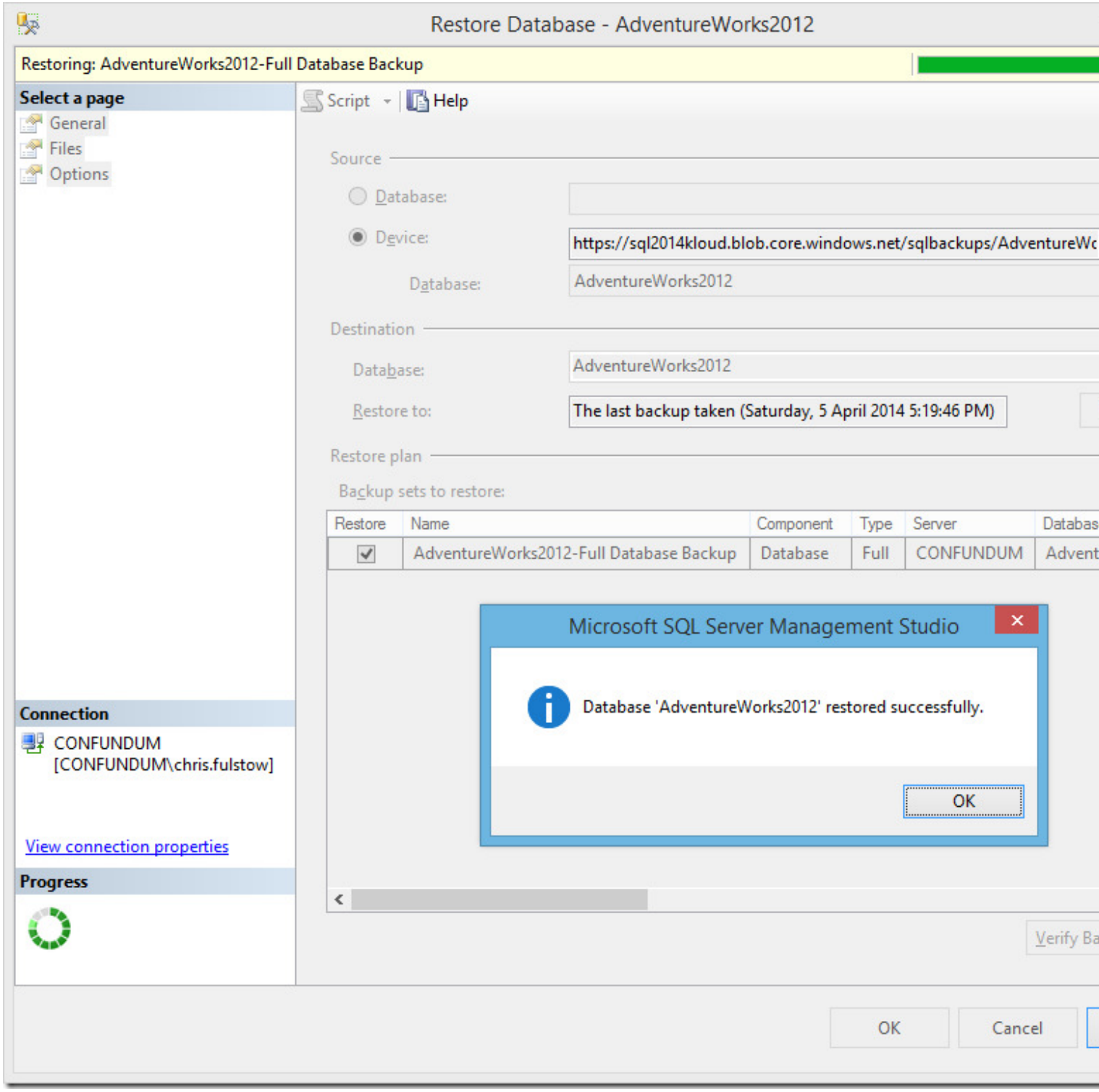
Now I'll select the storage account where I saved my backup, and paste in my storage account key.



Once connected to my storage account I can see the backup file in the blob container and select it to be restored.



And it’s done, easy as that, my AdventureWorks database is fully restored to my on-premises SQL Server instance directly from a blob stored in Azure.



Conclusion

No more messing about with physical backup disks, or even worse, tapes. Although the SSMS user interface is a bit clunky in places, everything is scriptable with TSQL, making the option to backup and restore a database directly to Azure blob storage very convenient. Each Azure storage account can hold up to 200TB with a maximum blob size of 1TB, giving me a lot of low cost, off-site, geo-redundant archive storage at my fingertips.