

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.

Complete

Your SQL Server 2014 installation completed successfully with product updates.

- Product Key
- License Terms
- Global Rules
- Product Updates
- Install Setup Files
- Install Rules
- Setup Role
- Feature Selection
- Feature Rules
- Instance Configuration
- Server Configuration
- Database Engine Configuration
- Feature Configuration Rules
- Ready to Install
- Installation Progress
- Complete**

Information about the Setup operation or possible next steps:

Feature	Status
Management Tools - Complete	Succeeded
Management Tools - Basic	Succeeded
Database Engine Services	Succeeded
SQL Browser	Succeeded
SQL Writer	Succeeded
Setup Support Files	Succeeded

Details:

Viewing Product Documentation for SQL Server

Only the components that you use to view and manage the documentation for SQL Server have been installed. By default, the Help Viewer component uses the online library. After installing SQL Server, you can use the Help Library Manager component to download documentation to your local computer. For more information, see Use Microsoft Books Online for SQL Server (<http://go.microsoft.com/fwlink/?LinkId=299578>).

Microsoft Update

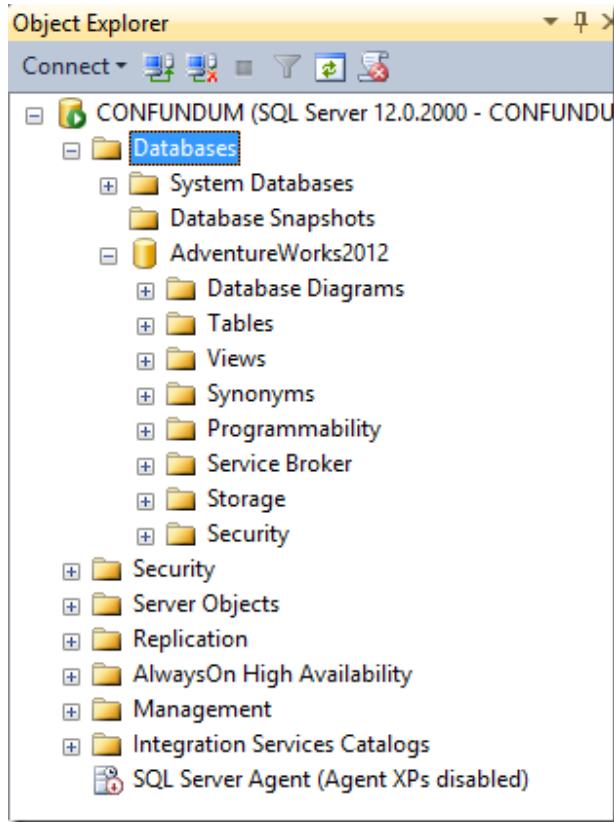
Summary log file has been saved to the following location:

C:\Program Files\Microsoft SQL Server\120\Setup Bootstrap\Log\20140404_204853\Summary.confundum.20140404_204853.txt

Close

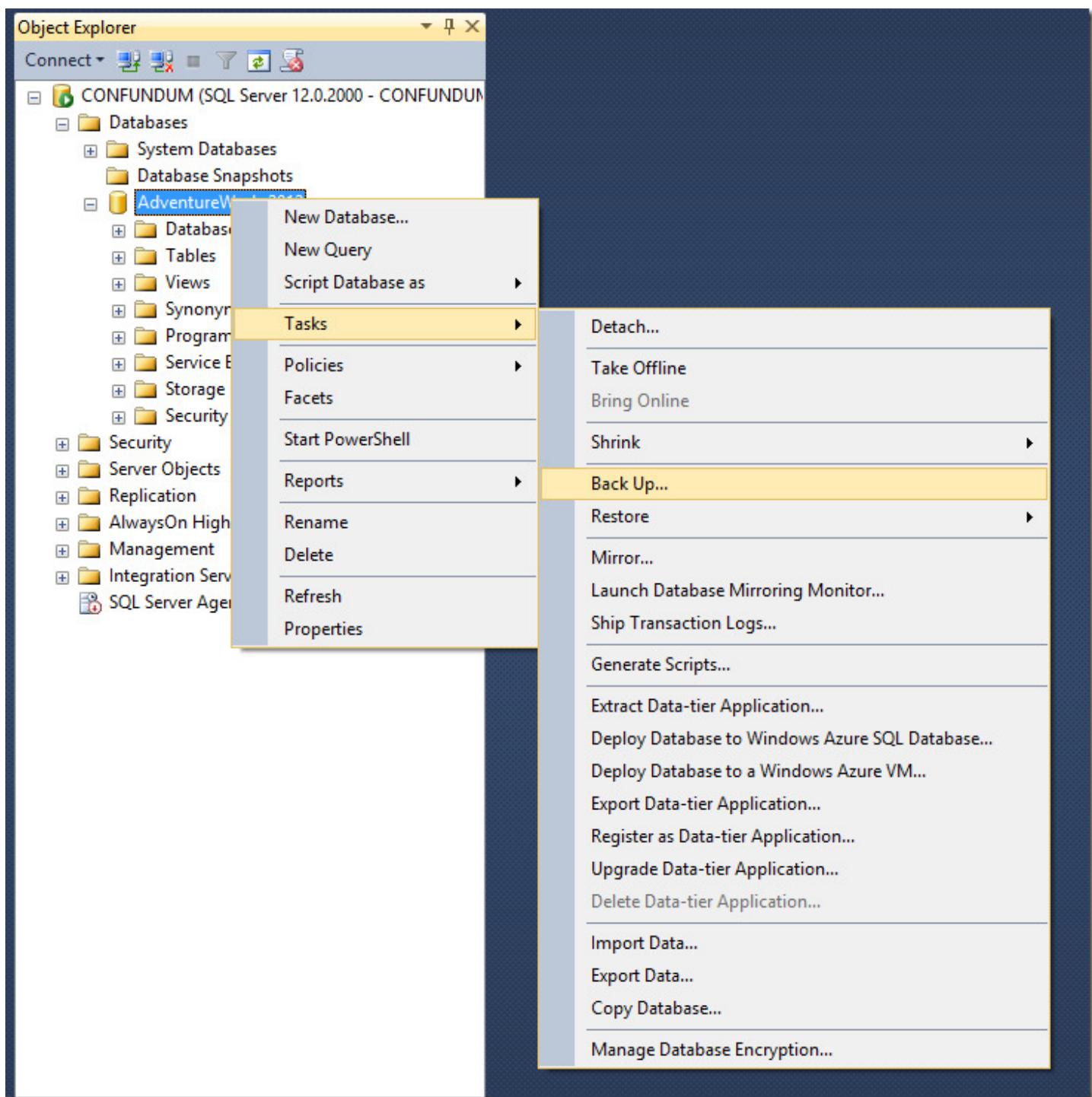
Help

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.



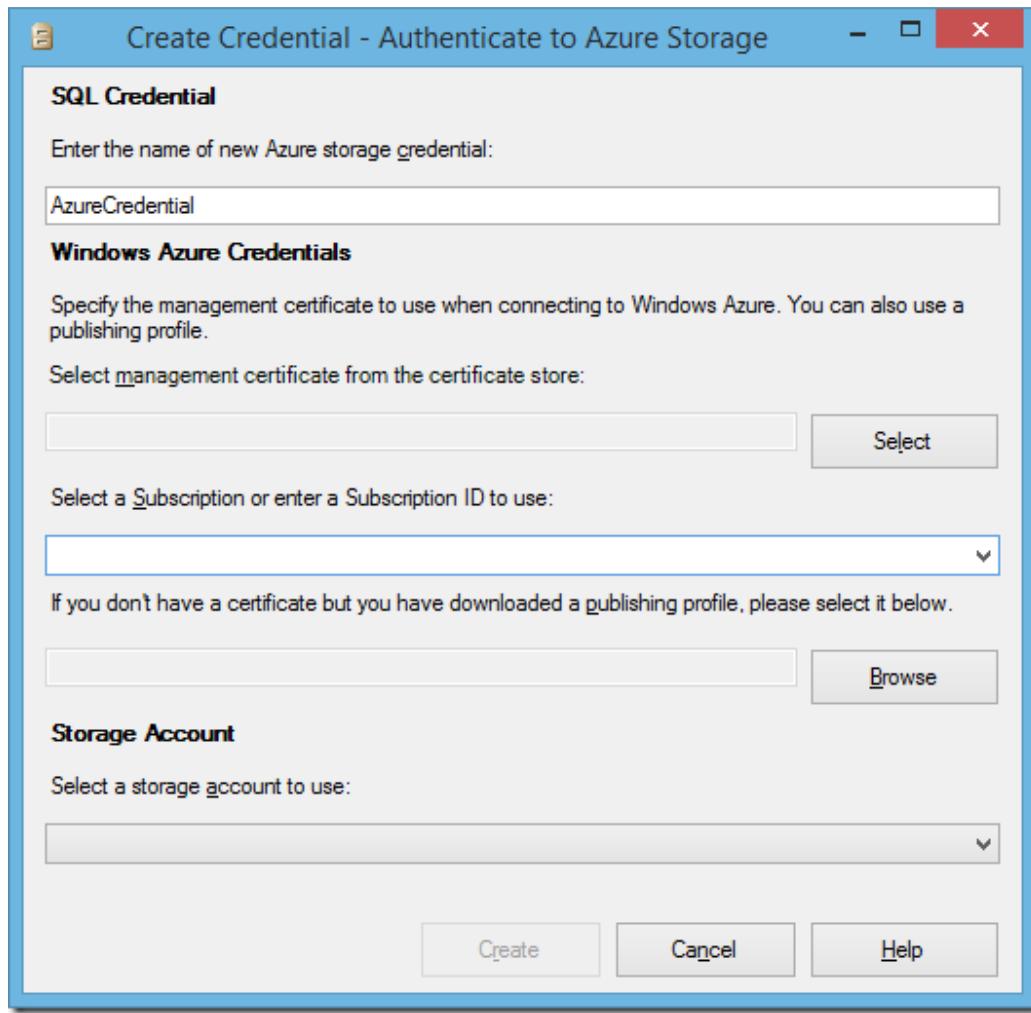
It's not immediately obvious where to select Azure as the backup destination, but expanding the *Back up* to dropdown reveals a new option for backing up to a URL. To me, URL feels like a strange hypernym given the address must point to an Azure storage blob endpoint – maybe a *Cloud* media type would've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

The screenshot shows the 'Backup Database' dialog in SSMS. In the 'Source' section, 'AdventureWorks2012' is selected as the database, 'SIMPLE' as the recovery model, and 'Full' as the backup type. Under 'Backup component:', 'Database' is chosen. In the 'Destination' section, the backup is set to go to a URL, with the file name being 'AdventureWorks2012_backup_2014_04_04_212810.bak'. The 'SQL credential' field has a 'Create...' button available. The 'Azure storage container' and 'URL prefix' fields are also visible.

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.

```
1 New-AzureStorageAccount `  
2     -StorageAccountName "sql2014kloud" `  
3     -Location "Southeast Asia"  
4  
5 Set-AzureSubscription `  
6     -SubscriptionName "Kloud Solutions" `  
7     -CurrentStorageAccount "sql2014kloud"  
8  
9 New-AzureStorageContainer -Name "sqlbackups"
```

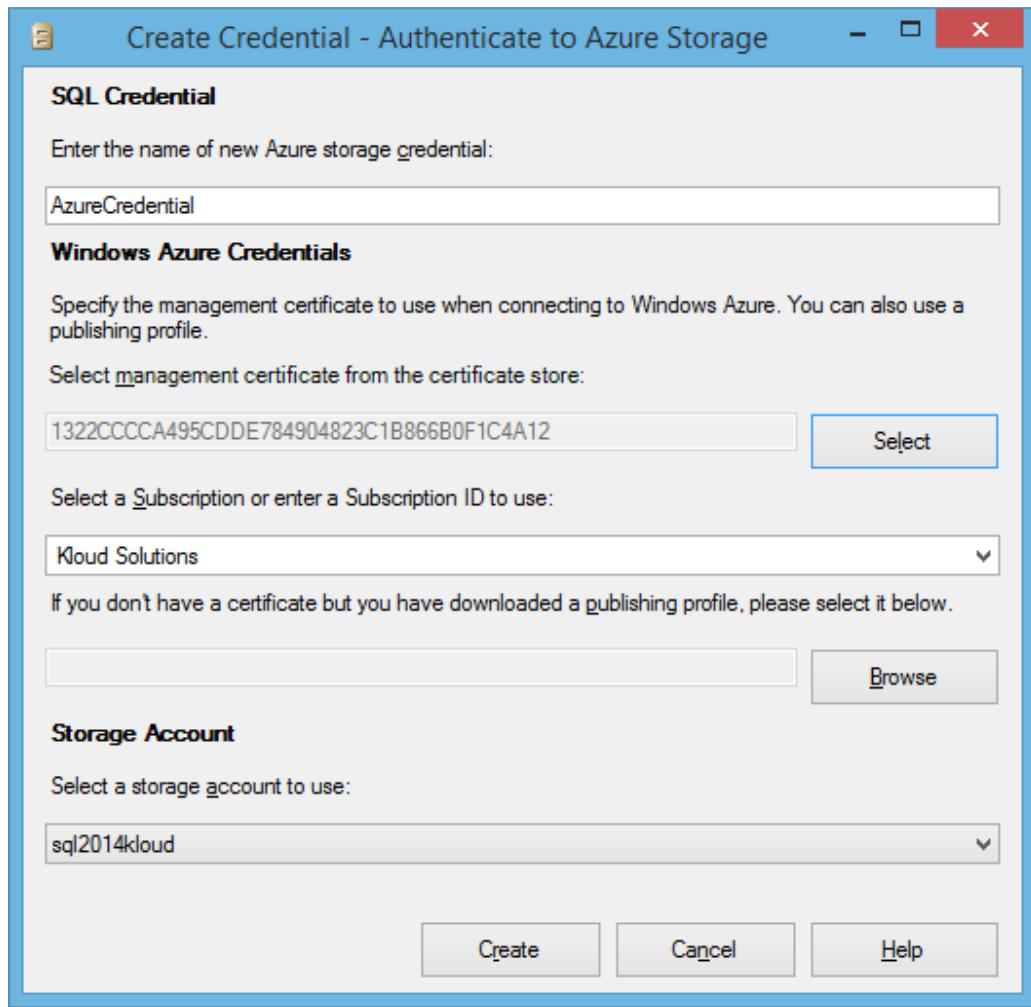
Now that's done, I'll continue configuring my Azure backup destination. I'll stick with the default backup filename, then move on to authentication where again the UI is slightly unintuitive. The *SQL credential* field refers to an object SQL Server uses to store authentication information for connecting 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 `thesql2014kloud` account created above.



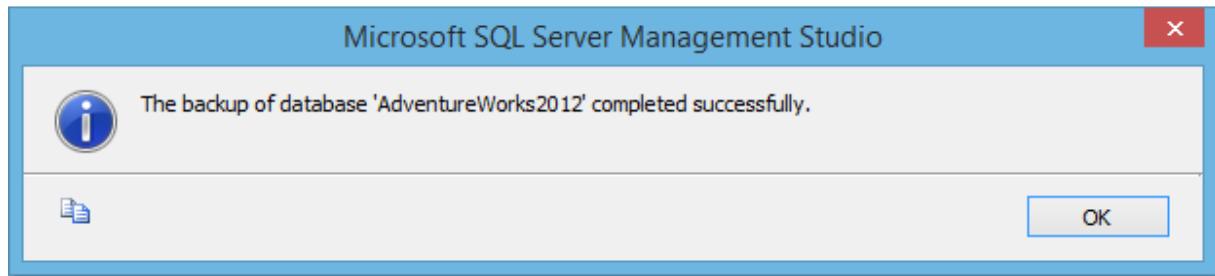
Lastly, I'll choose the *sqlbackups* storage container I created earlier, and we're good to go.

The screenshot shows the 'Destination' configuration dialog box. It contains the following fields:

Back up to:	URL
File name:	AdventureWorks2012_backup_2014_04_05_134136.bak
SQL credential:	AzureCredential
Azure storage container:	sqlbackups
URL prefix:	https://sql2014kloud.blob.core.windows.net/sqlbackups

There is also a "Create..." button next to the SQL credential dropdown.

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 container.

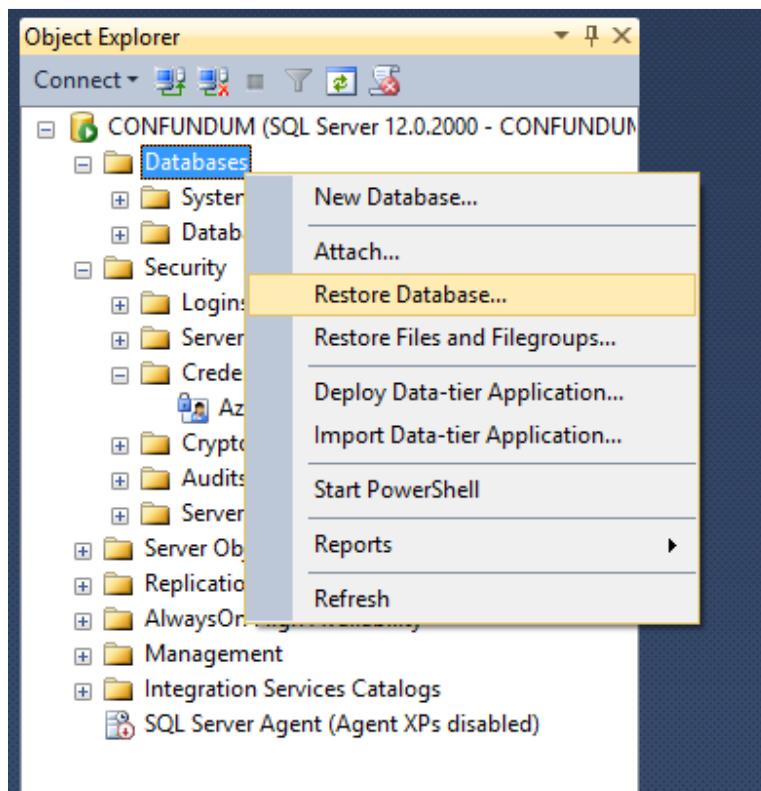
```
1 Get-AzureStorageBlob -Container "sqlbackups"
```

Name

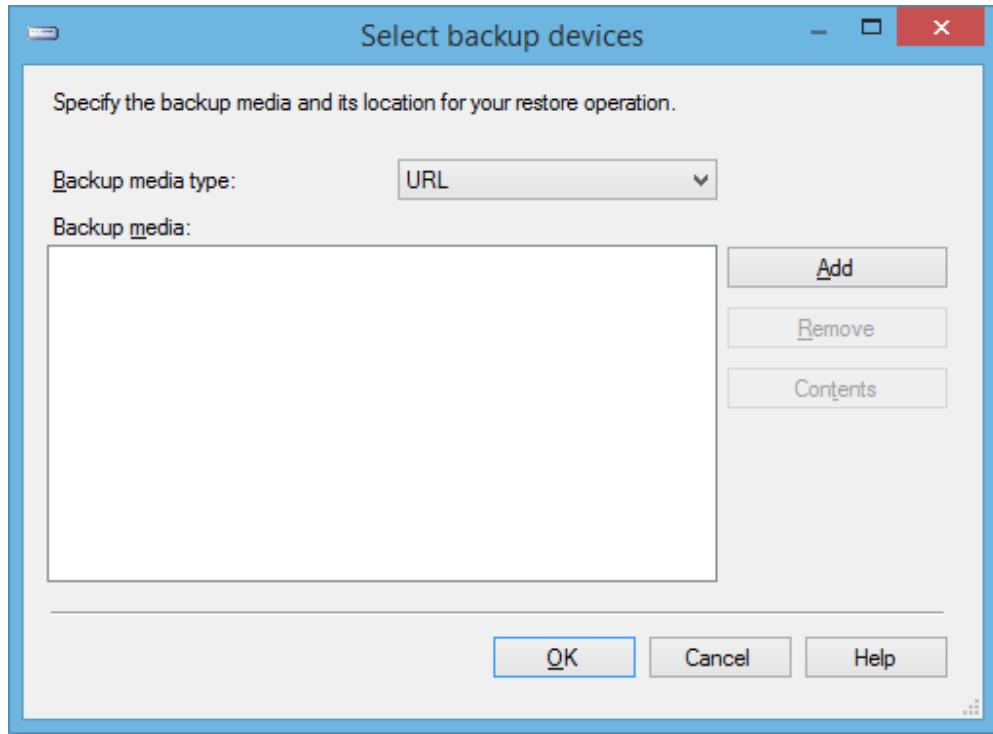
--

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.



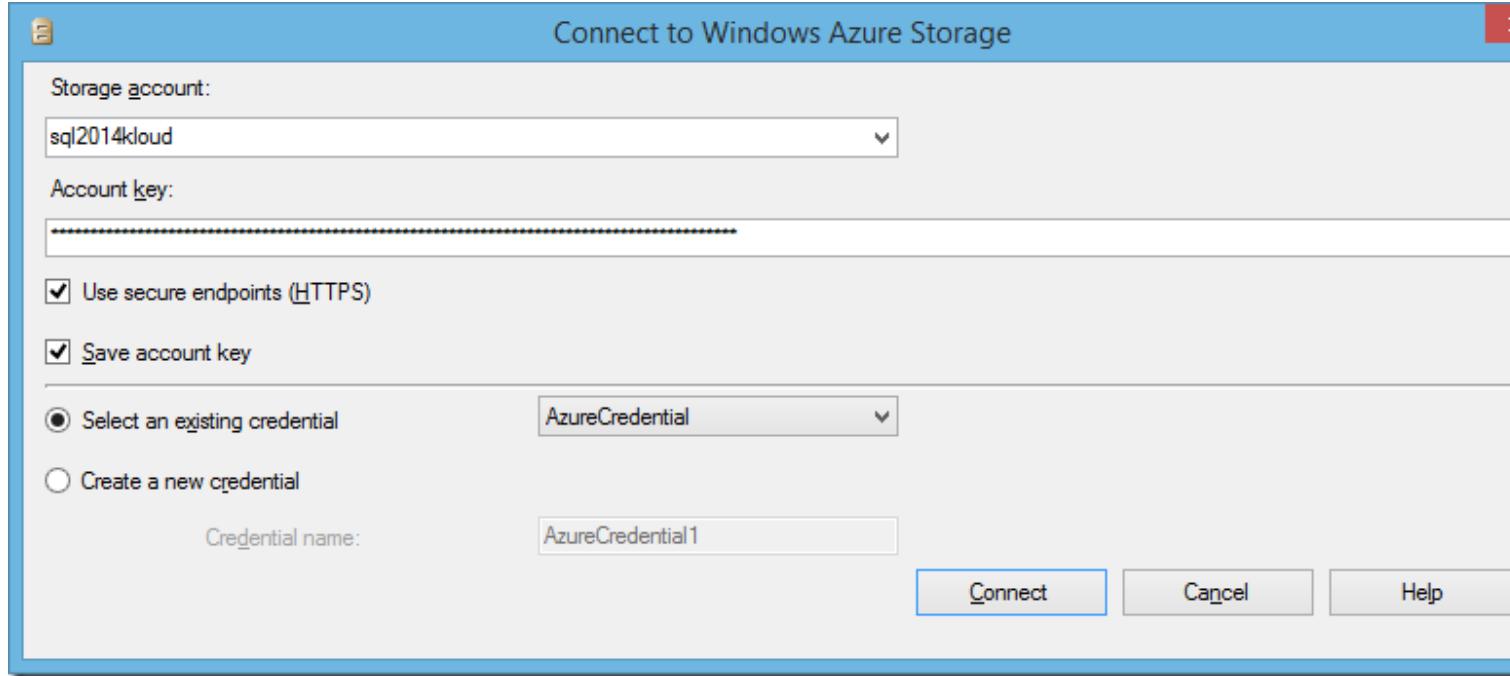
The Restore Database feature now has a new device media type of *URL*, which I'll select, then *Add* my Azure storage account:



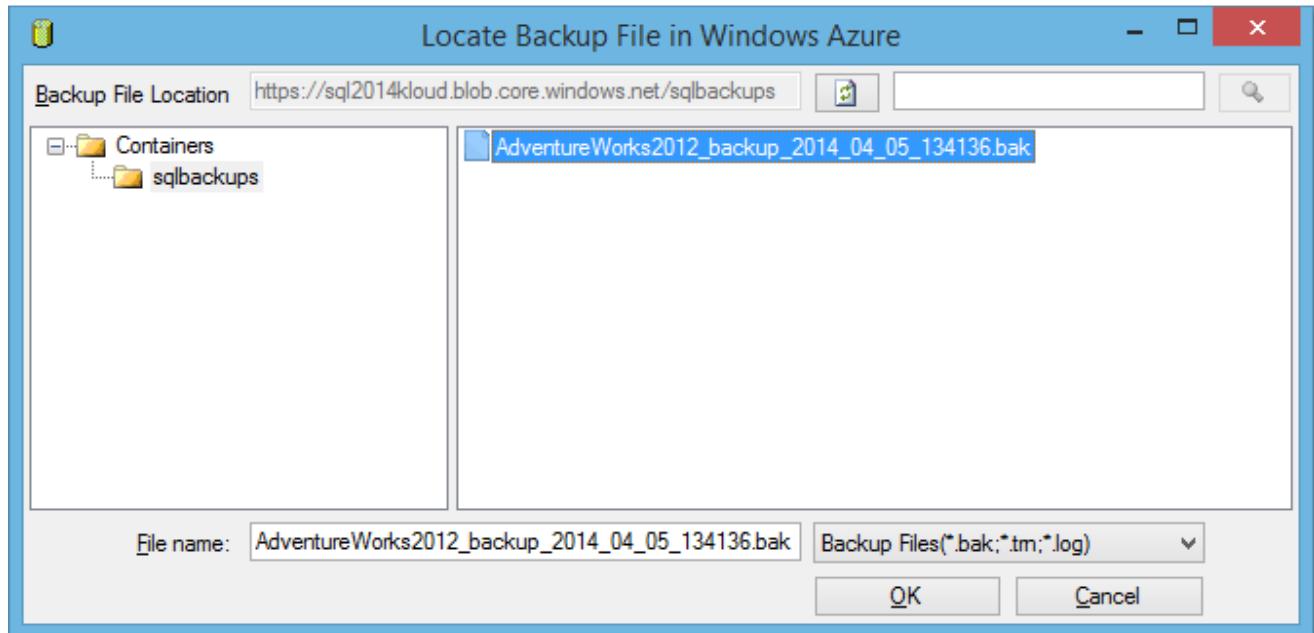
For this, I need my Azure storage access key, which is available from the Azure management portal or can be easily grabbed using PowerShell.

```
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.

Restore Database - AdventureWorks2012

Restoring: AdventureWorks2012-Full Database Backup

Select a page: General | Files | Options | Script | Help

Source:

Database: []

Device: [https://sql2014kloud.blob.core.windows.net/sqlbackups/AdventureWorks2012-Full Database Backup]

Database: AdventureWorks2012

Destination:

Database: AdventureWorks2012

Restore to: The last backup taken (Saturday, 5 April 2014 5:19:46 PM)

Restore plan:

Backup sets to restore:

Restore	Name	Component	Type	Server	Database
<input checked="" type="checkbox"/>	AdventureWorks2012-Full Database Backup	Database	Full	CONFUNDUM	AdventureWorks2012

Microsoft SQL Server Management Studio

i Database 'AdventureWorks2012' restored successfully.

OK

Connection: CONFUNDUM [CONFUNDUM\chris.fulstow]

[View connection properties](#)

Progress: [Circular Progress Bar]

Verify Backup

OK Cancel

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.