Experiment - Provisioning and connecting to an Azure Synapse SQL pool using PowerShell

In this experiment, we'll provision an Azure Synapse SQL pool using PowerShell. Provisioning a Synapse SQL pool uses the same commands that are used for provisioning an Azure SQL database, but with different parameters.

Getting ready

Before you start, log in to Azure from PowerShell. To do this, execute the following command and follow the instructions to log in to Azure:

```
Connect-AzAccount
```

How to do it...

Follow the given steps to provision a new Azure Synapse SQL pool:

1. We could execute the following command to create a new resource group (We created this already so this is just a reference):

```
#Create resource group
New-AzResourceGroup -Name AzureDataEngineeringxx -location
centralus
```

Note: Remember that anywhere we see xx we replace that with our student number, student1 would replace xx with 01 and student12 would replace xx with 12.

2. Execute the following command to create a new Azure SQL server:

```
    #create credential object for the Azure SQL Server
                    admin credential
$sqladminpassword = ConvertTo-SecureString 'Sql@
Server@1234' -AsPlainText -Force
$sqladmincredential = New-Object System.Management.
Automation.PSCredential ('sqladmin', $sqladminpassword)
# create the azure sql server
New-AzSqlServer -ServerName adesqlserverxx
-SqlAdministratorCredentials $sqladmincredential
-Location "central us" -ResourceGroupName
AzureDataEngineeringxx
```

Note: Remember that anywhere we see xx we replace that with our student number, student1 would replace xx with 01 and student12 would replace xx with 12.

You should get an output as shown in the following screenshot:

```
PS C:\Users\kubernetes\Downloads> $sqladminpassword = ConvertTo-SecureString 'Sql@Server@1234' -AsPla
inText -Force
PS C:\Users\kubernetes\Downloads> echo $sqladminpassword
System.Security.SecureString
PS C:\Users\kubernetes\Downloads> $sqladmincredential = New-Object System.Management.Automation.PSCre
dential ('sqladmin', $sqladminpassword)
PS C:\Users\kubernetes\Downloads> New-AzSqlServer -ServerName adesqlserver -SqlAdministratorCredentia
ls $sqladmincredential -Location "central us" -ResourceGroupName AzureDataEngineering

ResourceGroupName             : AzureDataEngineering
ServerName                    : adesqlserver
Location                      : centralus
SqlAdministratorLogin         : sqladmin
SqlAdministratorPassword      :
ServerVersion                 : 12.0
Tags                          :
Identity                      :
FullyQualifiedDomainName      : adesqlserver.database.windows.net
ResourceId                    : /subscriptions/b7e892b5-020c-44c1-8d40-99e608f1b2d6/resourceGroups/A
                                zureDataEngineering/providers/Microsoft.Sql/servers/adesqlserver
MinimalTlsVersion             :
PublicNetworkAccess           : Enabled
RestrictOutboundNetworkAccess : Disabled
Administrators                :
PrimaryUserAssignedIdentityId :
KeyId                         :
FederatedClientId             :
```

3. Execute the following command to create a new Azure Synapse SQL pool:

```
New-AzSqlDatabase -ServerName adesqlserverxx -DatabaseName adesqldwxx
-Edition DataWarehouse -RequestedServiceObjectiveName DW100c -
ResourceGroupName AzureDataEngineeringxx
```

Note: Remember that anywhere we see xx we replace that with our student number, student1 would replace xx with 01 and student12 would replace xx with 12.

To understand Data Warehouse Units (DW100c) you can read further here https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/what-is-a-data-warehouse-unit-dwu-cdwu

You should get an output as shown in the following screenshot:

```
PS C:\Users\kubernetes\Downloads> New-AzSqlDatabase -ServerName adesqlserver -DatabaseName adesqldw -
Edition DataWarehouse -RequestedServiceObjectiveName DW100c -ResourceGroupName AzureDataEngineering
WARNING: Upcoming breaking changes in the cmdlet 'New-AzSqlDatabase' :
- The output type 'Microsoft.Azure.Commands.Sql.Database.Model.AzureSqlDatabaseModel' is changing
- The following properties in the output type are being deprecated : 'BackupStorageRedundancy'
- The following properties are being added to the output type : 'CurrentBackupStorageRedundancy' 'Req
uestedBackupStorageRedundancy'
- The change is expected to take effect from the version : '3.0.0'
Note : Go to https://aka.ms/azps-changewarnings for steps to suppress this breaking change warning, a
nd other information on breaking changes in Azure PowerShell.

ResourceGroupName               : AzureDataEngineering
ServerName                      : adesqlserver
DatabaseName                    : adesqldw
Location                        : centralus
DatabaseId                      : 3831ca8e-4566-4127-a10d-318e6ace514e
Edition                         : DataWarehouse
CollationName                   : SQL_Latin1_General_CP1_CI_AS
CatalogCollation                :
MaxSizeBytes                    : 263882790666240
Status                          : Online
CreationDate                    : 5/18/2022 12:30:54 AM
CurrentServiceObjectiveId       : 00000000-0000-0000-0000-000000000000
CurrentServiceObjectiveName     : DW100c
RequestedServiceObjectiveName   : DW100c
```

4. Execute the following command to whitelist the public IP of the client machine in the Azure SQL Server firewall:

```
#Get the client public ip
$clientip = (Invoke-RestMethod -Uri https://ipinfo.io/
json).ip
#Whitelist the public IP in firewall
New-AzSqlServerFirewallRule -FirewallRuleName "home"
-StartIpAddress $clientip -EndIpAddress $clientip
-ServerName adesqlserverxx -ResourceGroupName
AzureDataEngineeringxx
```

You should get the following output:

```
PS C:\Users\kubernetes\Downloads> $clientip = (Invoke-RestMethod -Uri https://ipinfo.io/json).ip
PS C:\Users\kubernetes\Downloads> New-AzSqlServerFirewallRule -FirewallRuleName "home" -StartIpAddres
s $clientip -EndIpAddress $clientip -ServerName adesqlserver -ResourceGroupName AzureDataEngineering

ResourceGroupName : AzureDataEngineering
ServerName        : adesqlserver
StartIpAddress    : 69.180.128.204
EndIpAddress      : 69.180.128.204
FirewallRuleName  : home
```

4. Install the PowerShell SQL Server module if not already installed

```
Install-Module SQLServer
```

5. Execute the following command to connect to the Synapse SQL pool:

```
Invoke-Sqlcmd -ServerInstance "adesqlserverxx.database.windows.net" -U
sqladmin -Password "Sql@Server@1234" -Database adesqldwxx  -Query "SELECT
GETDATE()"
```

```
PS C:\Users\kubernetes\Downloads> Invoke-Sqlcmd -ServerInstance "adesqlserver.database.windows.net" -
U sqladmin -Password "Sql@Server@1234" -Database adesqldw  -Query "SELECT GETDATE()"

Column1
-------
5/18/2022 1:09:30 AM
```

We could execute the following command to remove Azure SQL Server and the Azure Synapse SQL pool.

NOTE: DO NOT DELETE THESE RESOURCES SINCE WE WILL USE THEM

```
Remove-AzSqlServer -ServerName adesqlserverxx -ResourceGroupName
AzureDataEngineeringxx
```

How it works...

To provision an Azure Synapse SQL pool, we first need to create the logical Azure SQL server. To create an Azure SQL server, we use the **`New-AzSqlServer`** command. We provide the server name, resource group, location, and admin credentials as parameters.

To create an Azure Synapse SQL pool, we use the **`New-AzSqlDatabase`** command. This is the same command that is used to provision an Azure SQL database. We provide the server name, database name, resource group name, edition, and service objective. In the **Edition** parameter, we provide **Data warehouse**.

To add a new firewall rule, we use the **`New-AzSqlServerFirewallRule`** command. We provide a firewall rule name, server name, resource group name, and the start and end IP as parameters. Removing Azure SQL Server also removes the Azure Synapse SQL pool.

To connect to an Azure Synapse SQL pool, we can use **`sqlcmd`**, SSMS, Azure Data Studio, or any of the valid SQL drivers. The server name to be provided should be of the form **`<servername>.database.windows.net`**.

To remove an Azure SQL server, we use the **`Remove-AzSqlServer`** command. We provide the server name and resource group name as parameters.