

Data Factory using VS

Lab 06

by

Andrea Hatch, Nishava Inc.

Deep Azure @McKesson

Overview

- Set-up all Prerequisites on Slide 3
- Create a Console App
- Install NuGet Packages
- Add all code for your Data Factory Demo
- Run the Code
- View and monitor your run in Azure
- Check your output in SQL database

Objective of Demo

- Create a Data Factory in Azure using Visual Studio /.NET that will ingest data from an Excel spreadsheet (or Blob), perform an identify transformation (identical copy) and transfer data to SQL Server data store as a sink.
- This is accomplished via a workflow (job) initialized within the Data Factory.

Prerequisites

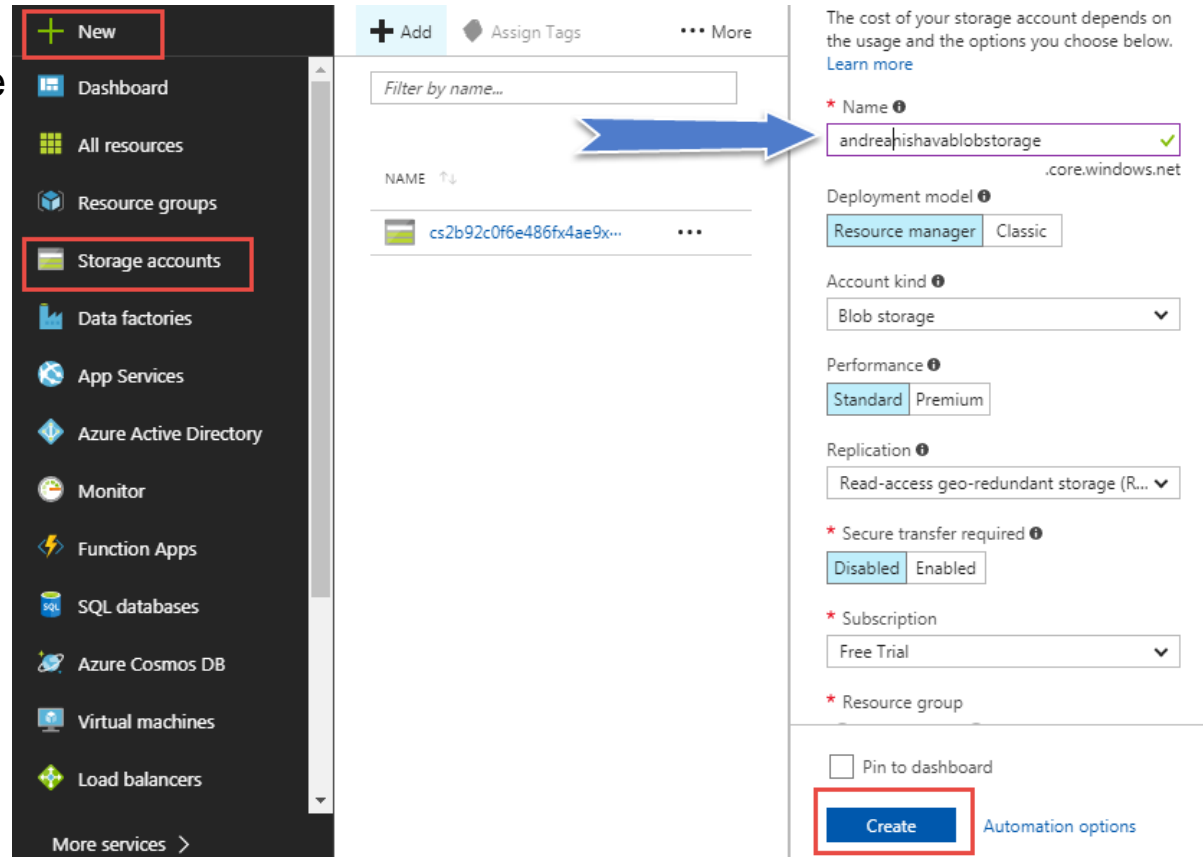
- You will need to do the following before creating a Data Factory:
 - Ensure you have Azure .NET SDK downloaded for VS (shown in week 3 lab)
 - Visual Studio (set-up shown in week 3)
 - Create a Blob Storage
 - Create an application in Azure Active Directory
 - Create an Azure Storage account
 - Create an Azure SQL Database
 - Create a SQL table in your SQL Database
 - Allow Azure services to access your SQL Server

Create a Storage Account

Steps:

1. In Azure select New ->Storage accounts
2. Name: Unique within Azure.
3. Account Kind: Blob or General Purpose
4. Create new resource group or use an existing one.
5. Location: select any location.
(I use East US).

Optional: Pin to Dashboard
6. Select Create.



The screenshot displays the Azure portal interface for creating a new storage account. On the left, the 'New' button is highlighted in the navigation pane, and 'Storage accounts' is selected. The main area shows the 'Create Storage Account' form. A blue arrow points from the 'Storage accounts' selection to the form. The form includes fields for Name (andreshavablobstorage), Deployment model (Resource manager), Account kind (Blob storage), Performance (Standard), Replication (Read-access geo-redundant storage), Secure transfer required (Disabled), Subscription (Free Trial), and Resource group. The 'Create' button is highlighted in red at the bottom.

New

- Dashboard
- All resources
- Resource groups
- Storage accounts**
- Data factories
- App Services
- Azure Active Directory
- Monitor
- Function Apps
- SQL databases
- Azure Cosmos DB
- Virtual machines
- Load balancers
- More services >

+ Add **Assign Tags** **More**

Filter by name...

NAME ↑↓

cs2b92c0f6e486fx4ae9x...

The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

* Name **andreshavablobstorage** ✓
.core.windows.net

Deployment model **Resource manager** Classic

Account kind **Blob storage**

Performance **Standard** Premium

Replication **Read-access geo-redundant storage (R...)**

* Secure transfer required **Disabled** Enabled

* Subscription **Free Trial**

* Resource group

☐ Pin to dashboard

Create Automation options

Create a Container for your Storage

The screenshot shows the Azure portal interface for a storage account named 'andreanishavablobstorage'. The left sidebar contains navigation links: Overview (selected), Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS, Access keys, and Configuration. The main pane displays account details and a '+ Container' button, which is highlighted with a red rectangle. Below the details is a table for containers, which is currently empty.

Storage account details:

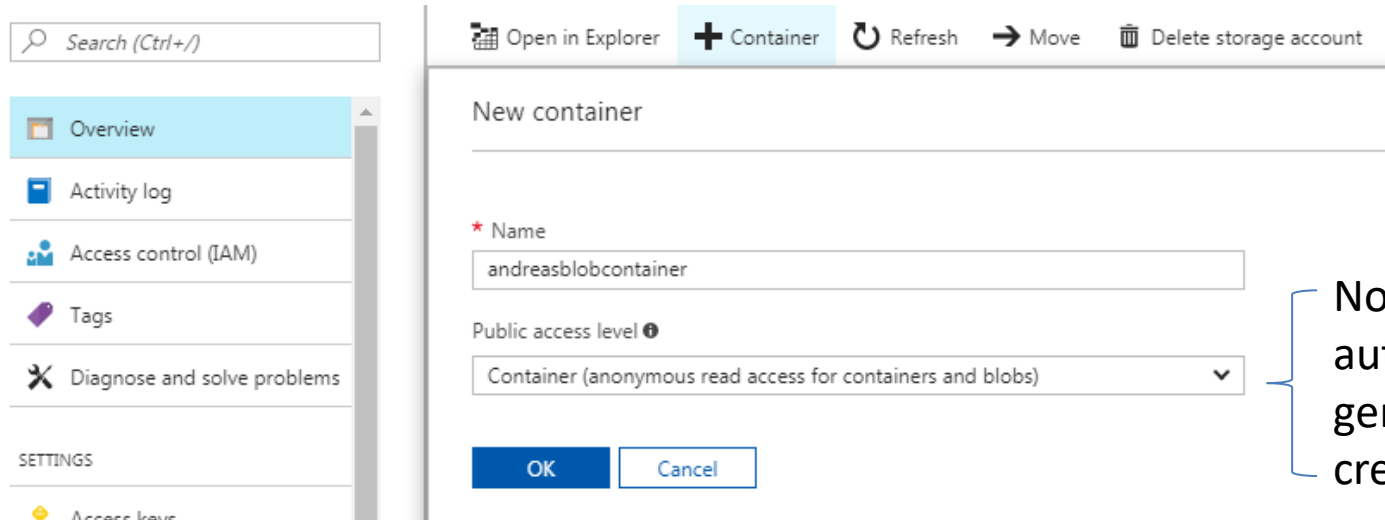
- Resource group: [andrearesourcegroup](#) (change)
- Status: Primary: Available, Secondary: Available
- Location: East US, West US
- Subscription: [Free Trial](#) (change)
- Subscription ID: b92c0f6e-486f-4ae9-96af-218ba438580f
- Performance/Access tier: Standard/Hot
- Replication: Read-access geo-redundant storage (RA-GRS)
- Primary blob service endpoint: <https://andreanishavablobstorage.blob.core.windows.net/>
- Secondary blob service endpoint: <https://andreanishavablobstorage-secondary.blob.core....>

Containers table:

NAME	LAST MODIFIED	PUBLIC ACCESS LE...	LEASE STATE
You don't have any containers yet. Click '+ Container' to get started.			

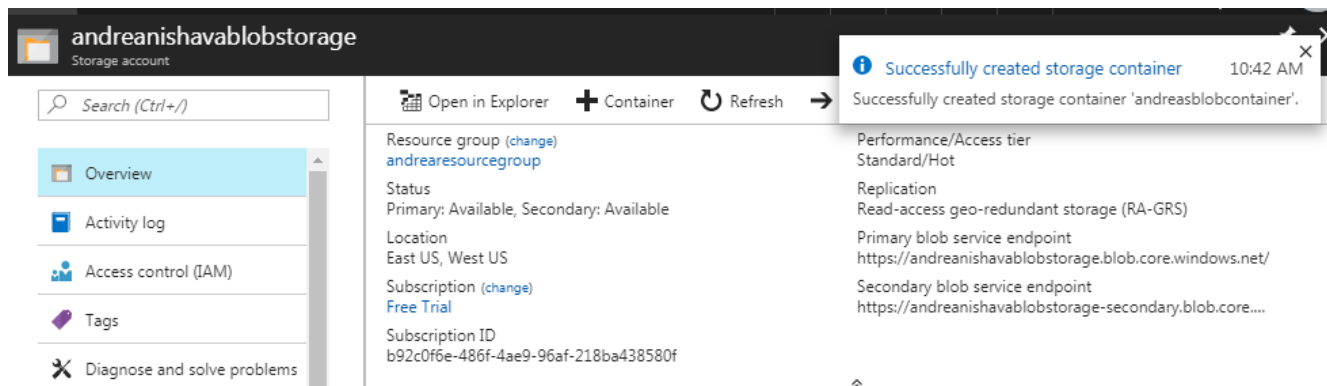
Create a Container for your Blob Storage

- You must set the access level to Container



The screenshot shows the 'New container' dialog box in the Azure portal. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS. The main area has a search bar and action buttons: Open in Explorer, + Container, Refresh, Move, and Delete storage account. The 'New container' form includes a 'Name' field with the value 'andreasblobcontainer', a 'Public access level' dropdown menu set to 'Container (anonymous read access for containers and blobs)', and 'OK' and 'Cancel' buttons.

Note: Keys are automatically generated when you create your storage.



The screenshot shows the 'andreasblobstorage' storage account page in the Azure portal. The left sidebar is the same as in the previous image. The main area displays the storage account details, including the resource group 'andrearesourcegroup', status 'Primary: Available, Secondary: Available', location 'East US, West US', subscription 'Free Trial', and subscription ID 'b92c0f6e-486f-4ae9-96af-218ba438580f'. A success message is displayed in a toast notification: 'Successfully created storage container' at 10:42 AM, with the text 'Successfully created storage container 'andreasblobcontainer'.' Below the details, there are links for 'Performance/Access tier' (Standard/Hot), 'Replication' (Read-access geo-redundant storage (RA-GRS)), 'Primary blob service endpoint' (https://andreasblobstorage.blob.core.windows.net/), and 'Secondary blob service endpoint' (https://andreasblobstorage-secondary.blob.core....).

Storage Access Keys

- In your storage under Settings select Access Keys
 - Note: Never share your storage access keys!

The screenshot shows the 'Access keys' page for the storage account 'andreanishavablobstorage'. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and a 'SETTINGS' section with 'Access keys' (highlighted with a red box) and 'Configuration'. The main content area includes a search bar, a warning about securely storing access keys, and instructions on regenerating keys. Below this, the 'Storage account name' is 'andreanishavablobstorage'. A table titled 'Default keys' lists two keys, 'key1' and 'key2', each with a masked key value, a copy icon, and a connection string starting with 'DefaultEndpointsProtocol=https;Acco...'. Each key row also has a refresh icon.

andreanishavablobstorage - Access keys
Storage account

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

SETTINGS

Access keys

Configuration

Use access keys to authenticate your applications when making requests to this Azure storage account. Store your access keys securely - for example, using Azure Key Vault - and don't share them. We recommend regenerating your access keys regularly. You are provided two access keys so that you can maintain connections using one key while regenerating the other.

When you regenerate your access keys, you must update any Azure resources and applications that access this storage account to use the new keys. This action will not interrupt access to disks from your virtual machines. [Learn more](#)

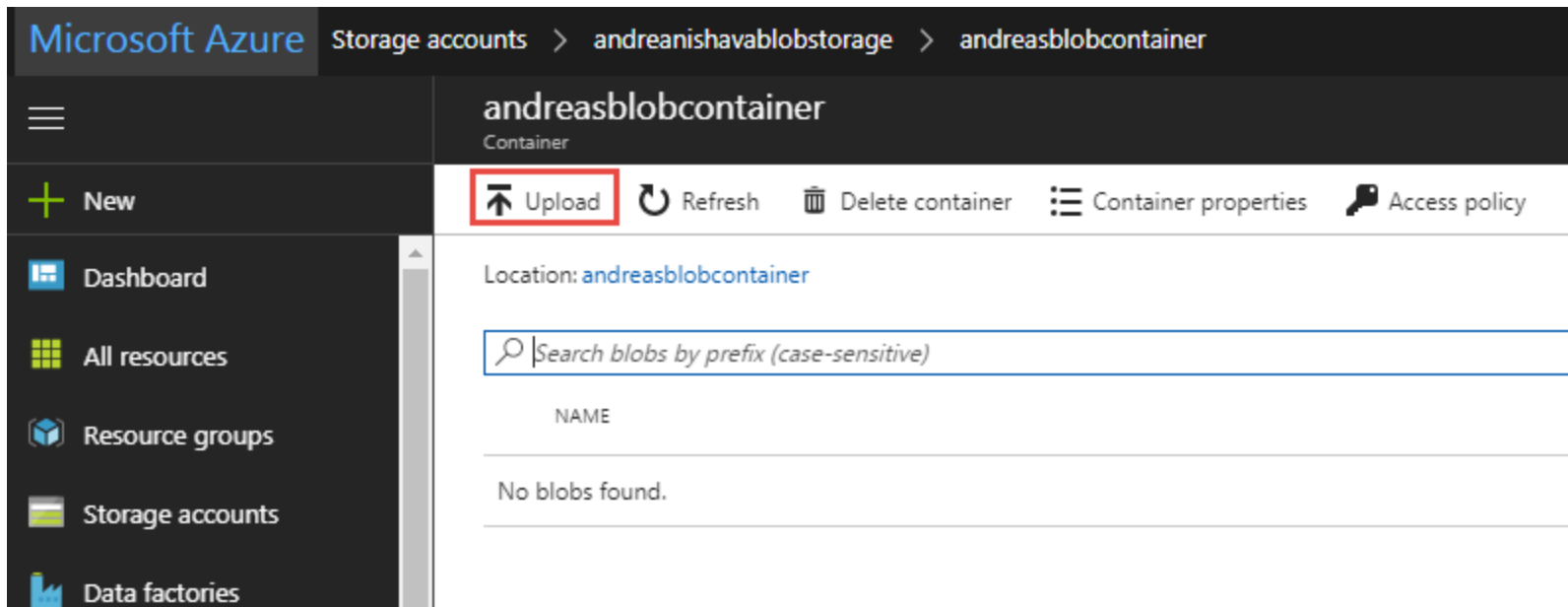
Storage account name: andreanishavablobstorage

Default keys

NAME	KEY	CONNECTION STRING
key1	[REDACTED]	DefaultEndpointsProtocol=https;Acco... [Copy] [Refresh]
key2	[REDACTED]	DefaultEndpointsProtocol=https;Acco... [Copy] [Refresh]

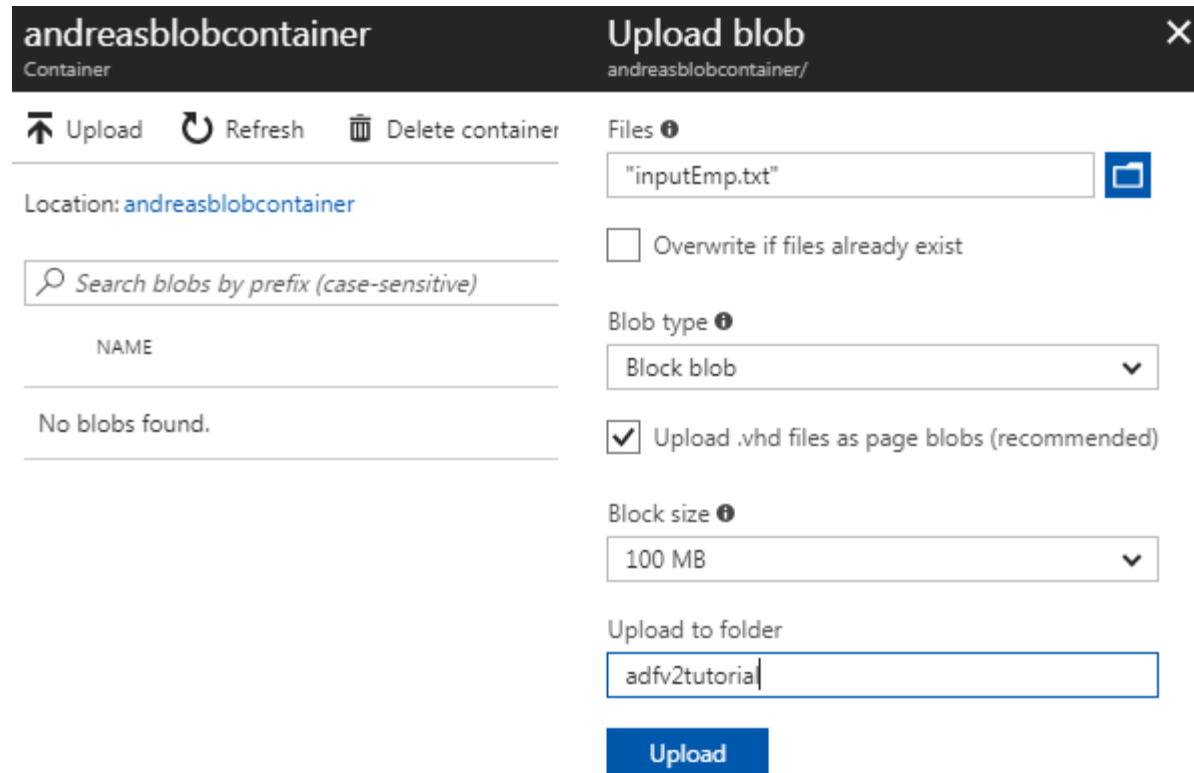
Upload Data File to your Blob

- In your Blob storage select your Blob container
- Select Upload



Upload Data File to your Blob

- Create a data file named **inputEmp.txt**
 - **Add some data to file such as:**
John|Doe
Jane|Doe
- Load this into your Blob by selecting the File and selecting Advanced and Upload it to a folder
- Select 'Upload'






The screenshot shows the 'Upload blob' interface for the 'andreasblobcontainer'. The interface is divided into two main sections: a left sidebar for container management and a right main area for file upload details.

Left Sidebar (Container Management):

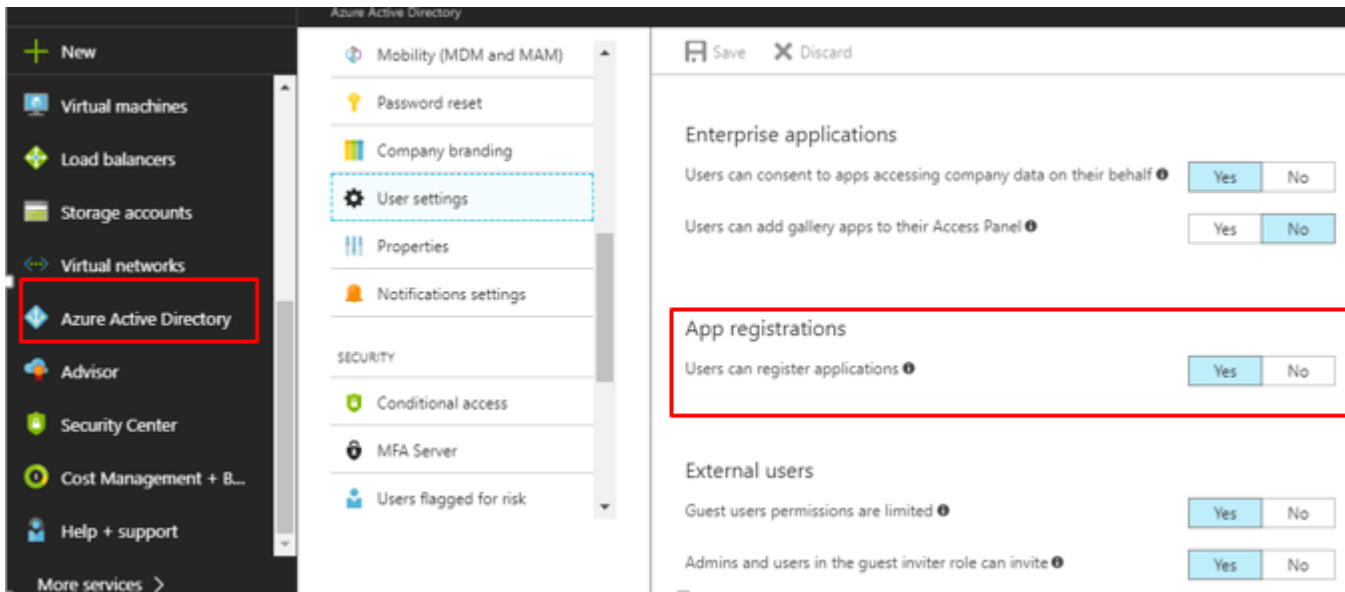
- Header: **andreasblobcontainer** (Container)
- Actions: Upload, Refresh, Delete container
- Location: [andreasblobcontainer](#)
- Search: Search blobs by prefix (case-sensitive)
- Table Header: NAME
- Content: No blobs found.

Right Main Area (Upload Details):

- Header: **Upload blob** (andreasblobcontainer/)
- Files: 
- Overwrite: ☐ Overwrite if files already exist
- Blob type: 
- Advanced Options: ☒ Upload .vhd files as page blobs (recommended)
- Block size: 
- Upload to folder:
- Upload Button: **Upload**

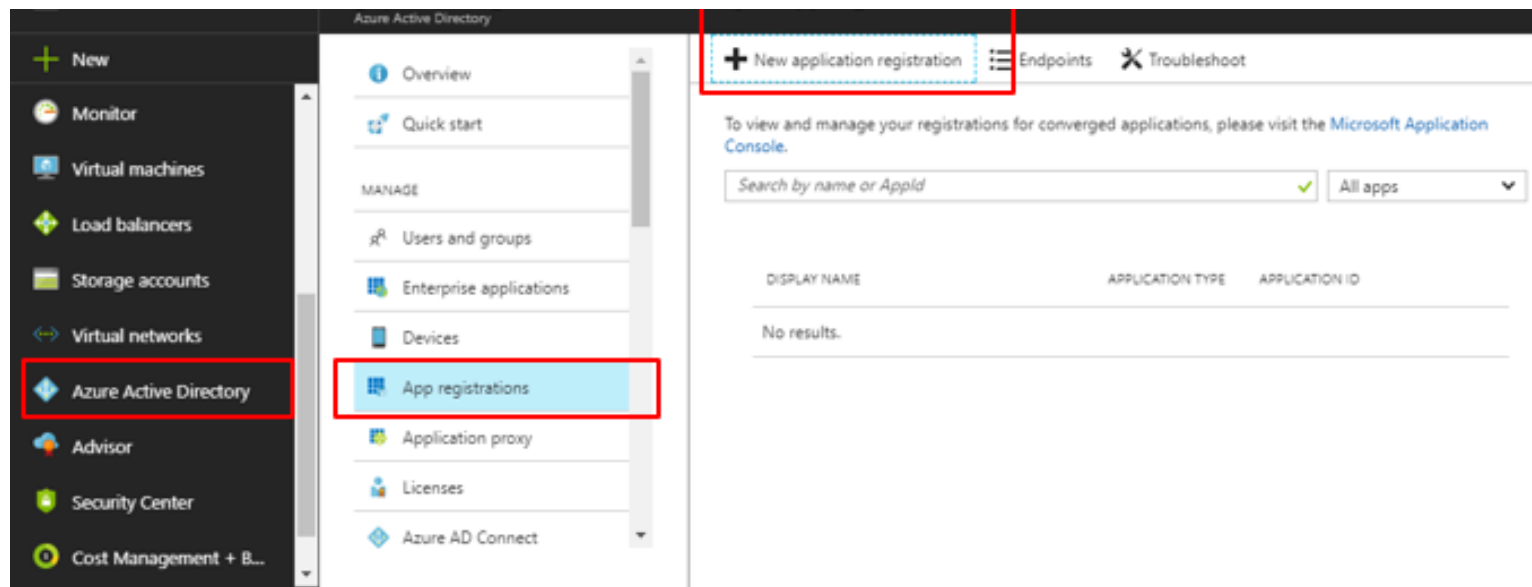
Create an application in Azure Active Directory

- Go to Active Directory to check if can register an App.
- Default is set to Yes!



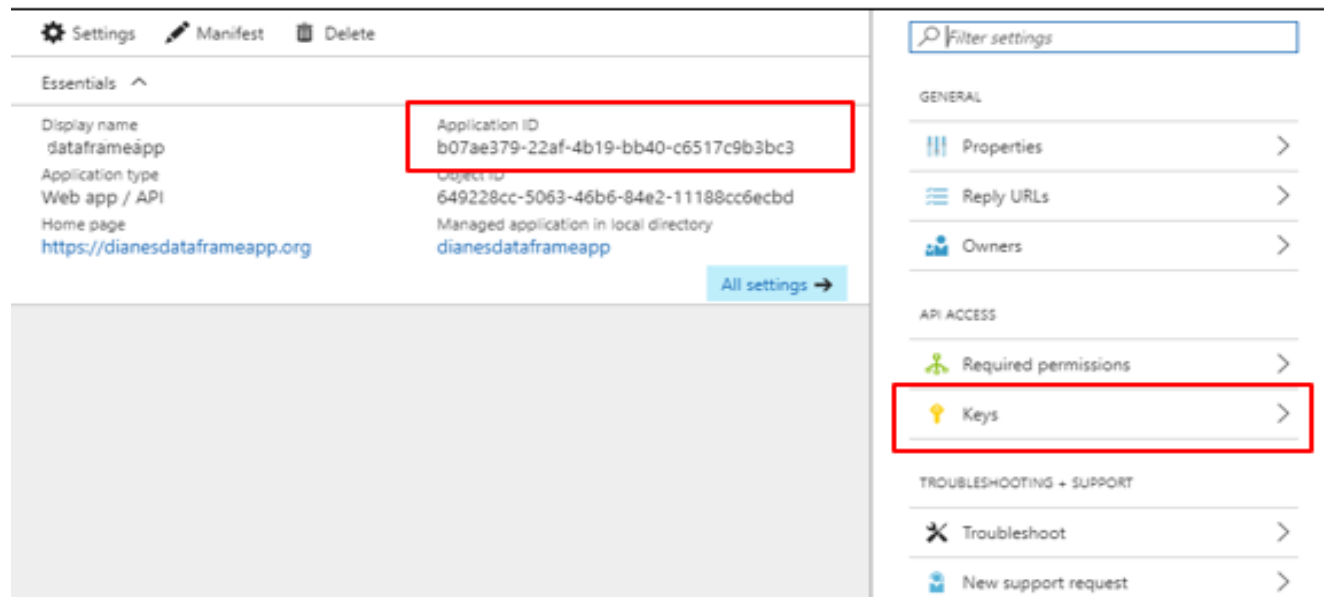
Create an Azure Active Directory application

- Set up an Azure Active Directory (AD) application and assign the required permissions
- Uses by an application that will need to access or modify resources



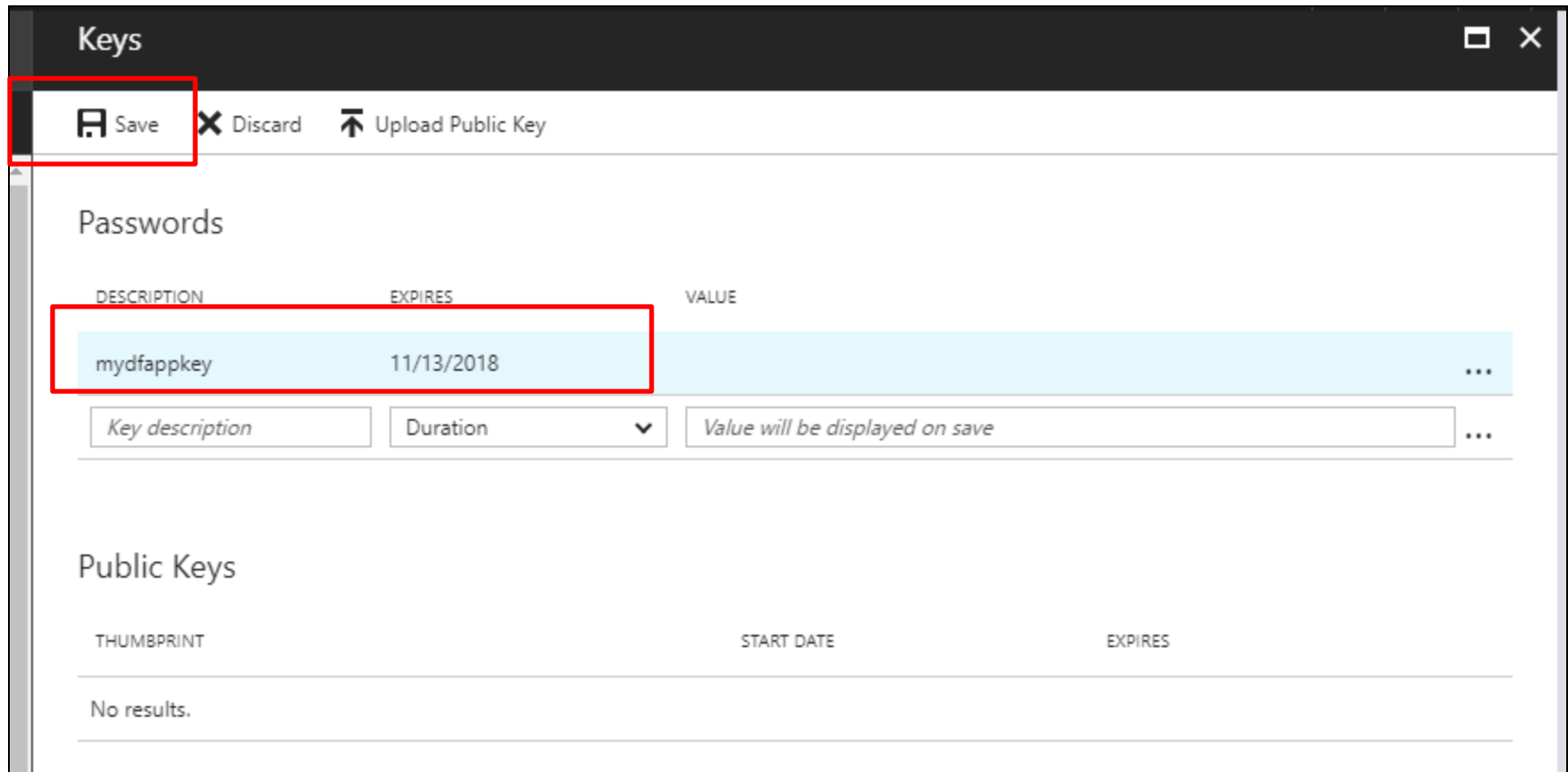
Create an Azure Active Directory application

- Need the Application ID for your .NET code
- Save it into Notepad as you will need to add it to your .NET code.



Create Authentication (Secret Key)

- Set up your Secret Key






The screenshot shows a web application titled "Keys". At the top, there is a dark header bar with the title "Keys" and window control icons. Below the header, there is a toolbar with three buttons: "Save" (highlighted with a red box), "Discard", and "Upload Public Key". The main content area is divided into two sections. The first section is titled "Passwords" and contains a table with three columns: "DESCRIPTION", "EXPIRES", and "VALUE". The first row of the table is highlighted with a light blue background and a red border. The "DESCRIPTION" column contains the text "mydfappkey", and the "EXPIRES" column contains the date "11/13/2018". Below the table, there are three input fields: "Key description", "Duration" (with a dropdown arrow), and "Value will be displayed on save". The second section is titled "Public Keys" and contains a table with three columns: "THUMBPRINT", "START DATE", and "EXPIRES". The table is currently empty, and the text "No results." is displayed below it.


DESCRIPTION	EXPIRES	VALUE
mydfappkey	11/13/2018	

THUMBPRINT	START DATE	EXPIRES
No results.		

Copy your Secret Key Value


- Appears one time only!
- This is also known as your Tenant ID.
- Save your Secret Key in Notepad!

 Save  Discard  Upload Public Key

 Copy the key value. You won't be able to retrieve after you leave this blade.

Passwords

DESCRIPTION	EXPIRES	VALUE
mydfapkey	11/13/2018	3lzdVyHiFsLn60bXohzryKUtnsPled9B015wKynujb4= ...

 ...

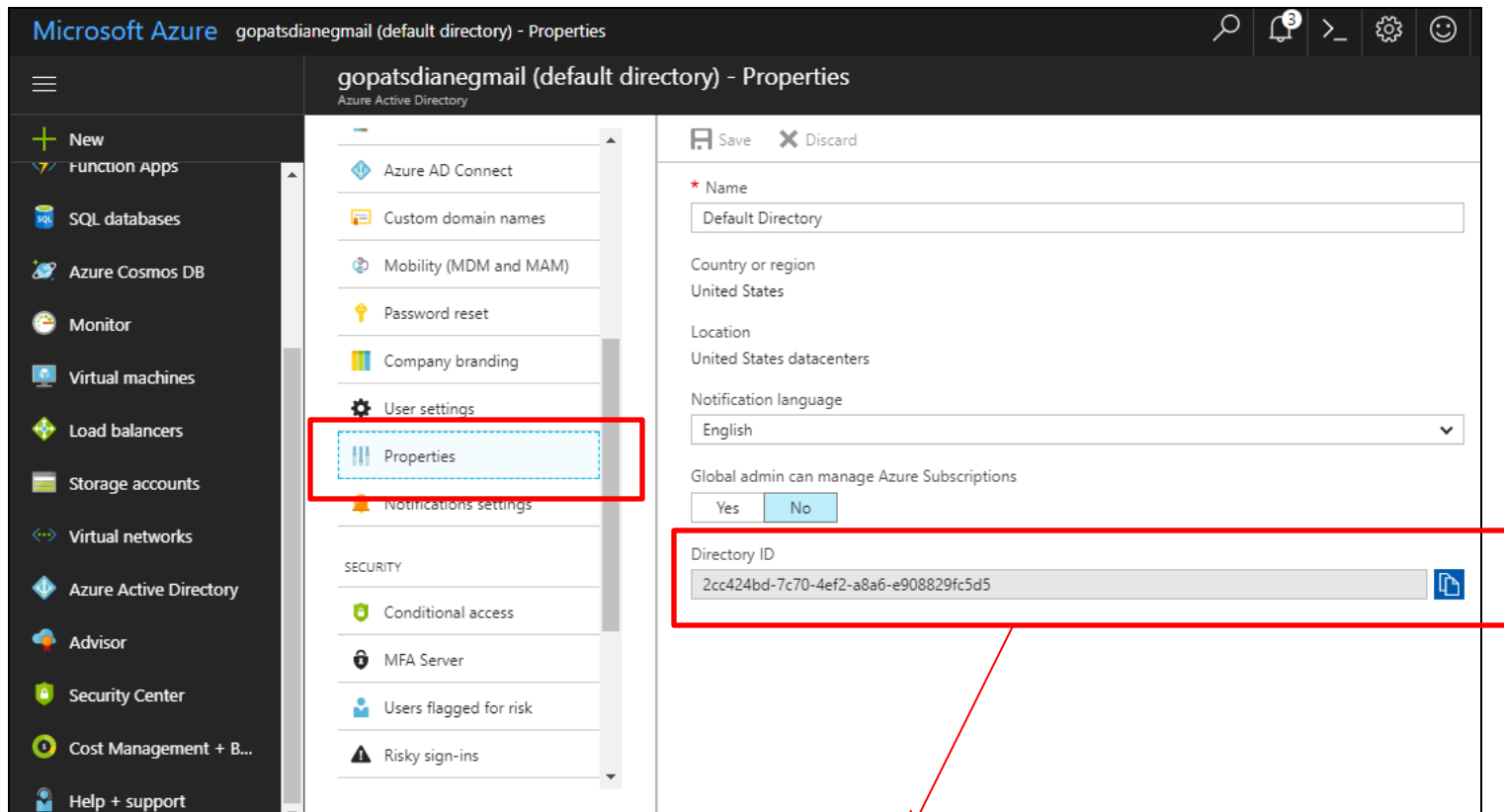
3lzdVyHiFsLn60bXohzryKUtnsPled9B015wKynujb4=

Public Keys

THUMBPRINT	START DATE	EXPIRES
No results.		

Get your Tenant ID

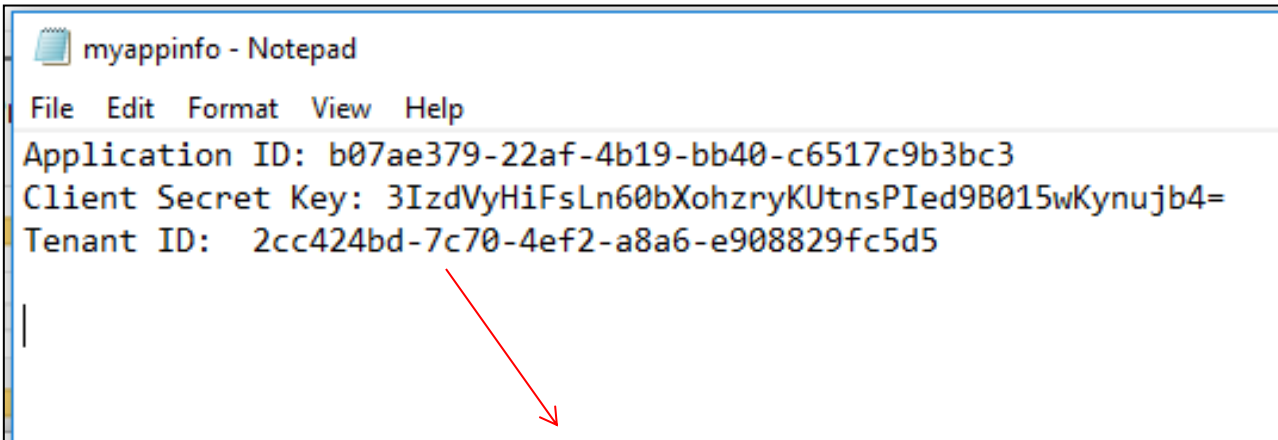
- The Directory ID = your Tenant ID.
- Copy it to Notepad



2cc424bd-7c70-4ef2-a8a6-e908829fc5d5

Save in Notepad

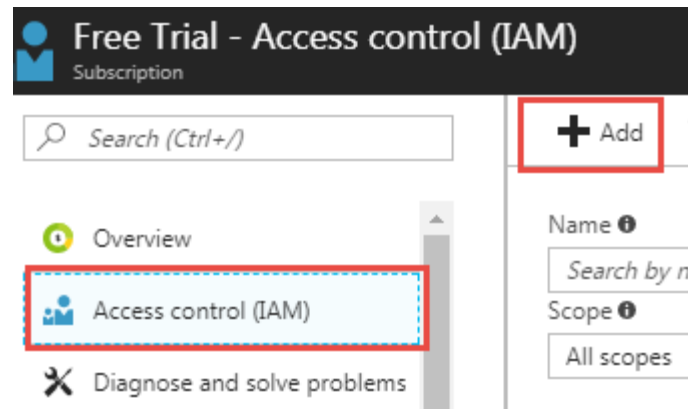
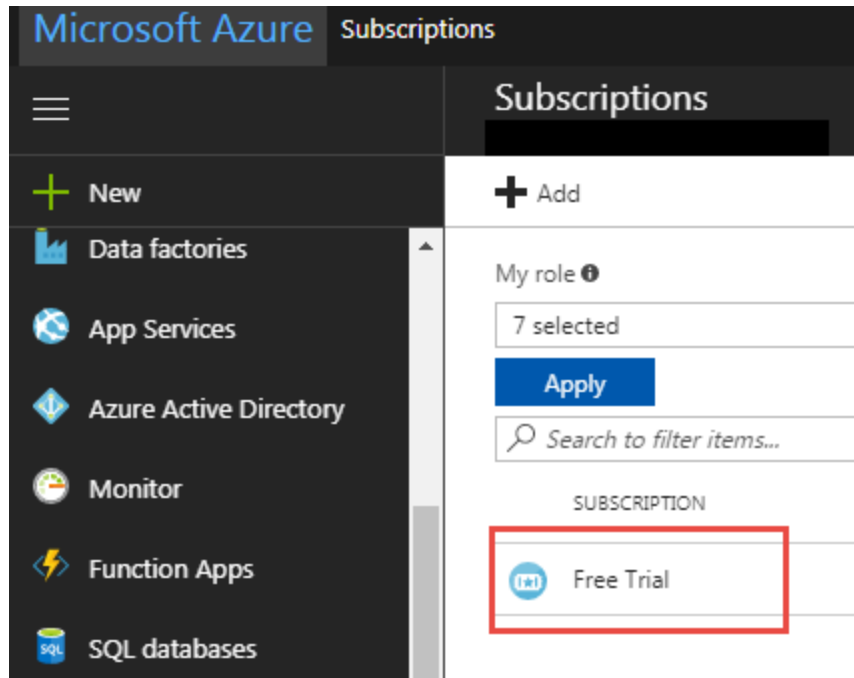
- Application ID: b07ae379-22af-4b19-bb40-c6517c9b3bc3
- Client Secret Key:
3IzdVyHiFsLn60bXohzryKUtnsPIed9B015wKynujb4=
- Tenant ID: 2cc424bd-7c70-4ef2-a8a6-e908829fc5d5
- We will need these values later in your .NET code.



```
myappinfo - Notepad
File Edit Format View Help
Application ID: b07ae379-22af-4b19-bb40-c6517c9b3bc3
Client Secret Key: 3IzdVyHiFsLn60bXohzryKUtnsPIed9B015wKynujb4=
Tenant ID: 2cc424bd-7c70-4ef2-a8a6-e908829fc5d5
```

Assign a Role

- Under New -> More Services -> Subscriptions
- Select your Subscription -> Access Control (IAM) -> Add



Assign a Role

- Under Role select Contributor
- Search for your data frame app, select it and then select Save


Add permissions [X]

Role ⓘ
Contributor ▼

Assign access to ⓘ
Azure AD user, group, or application ▼

Select ⓘ
dataframeapp ✓

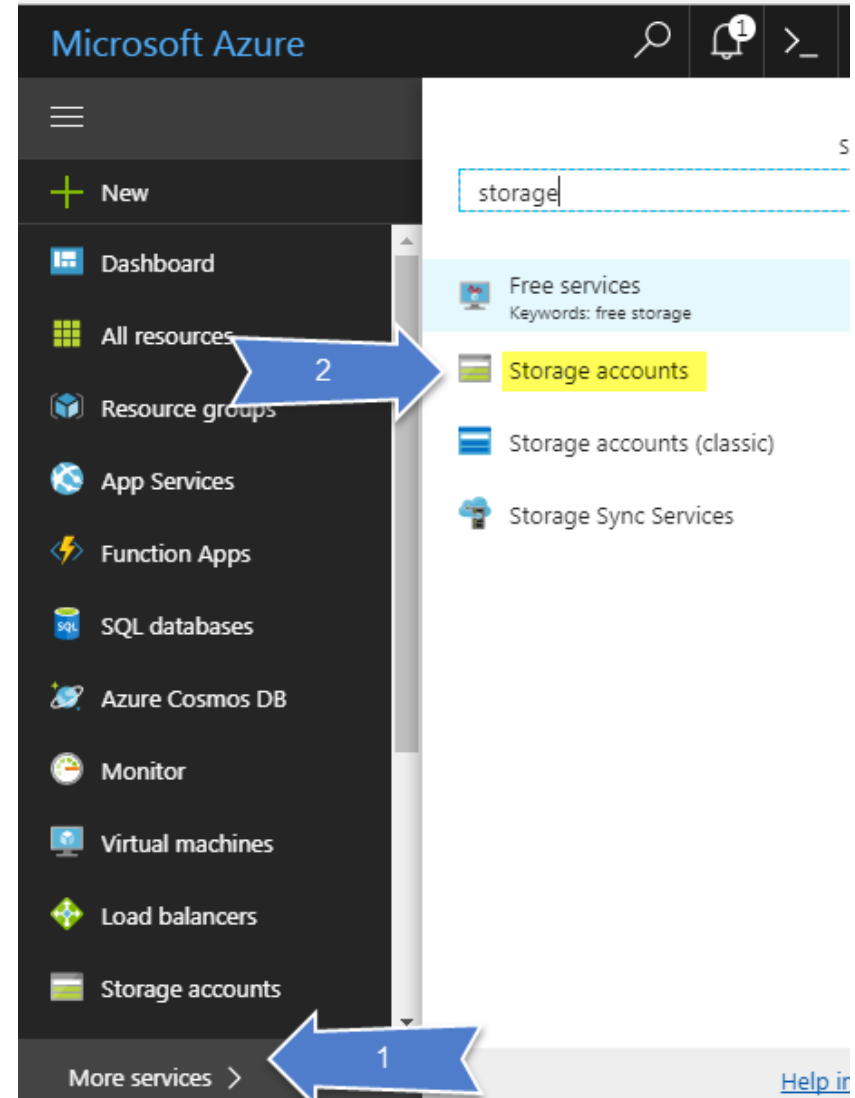
Selected members:

	dataframeapp	Remove
---	--------------	------------------------

[Save](#) [Discard](#)

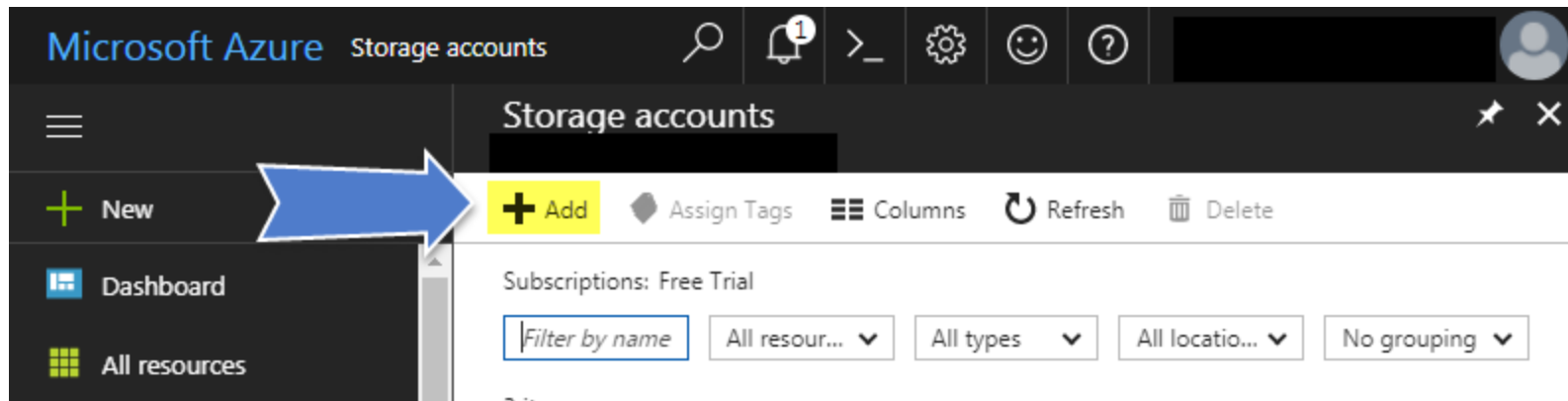
Create an Azure Storage Account

- In Azure select More Services
- Scroll until you find Storage and select Storage Accounts



Create an Azure Storage Account

- Select Add to create a Storage Account



Create an Azure Storage Account

- Enter a Name for the Storage Account
- Select the resource group that you used with your Blob creation
- Select Create

Create storage account

The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

* Name ⓘ
andreasblobnish ✓
core.windows.net

Deployment model ⓘ
Resource manager Classic

Account kind ⓘ
General purpose ▼

Performance ⓘ
Standard Premium

Replication ⓘ
Read-access geo-redundant storage (R... ▼

* Secure transfer required ⓘ
Disabled Enabled

☐ Pin to dashboard

Create [Automation options](#)

Create storage account

Replication ⓘ
Read-access geo-redundant storage (R... ▼

* Secure transfer required ⓘ
Disabled Enabled

* Subscription
Free Trial ▼

* Resource group
☒ Create new ☐ Use existing
andreaResourceGroup ✓

* Location
South Central US ▼

Virtual networks (Preview)
Configure virtual networks ⓘ
Disabled Enabled

☐ Pin to dashboard

Create [Automation options](#)

Save your Storage Account Subscription ID

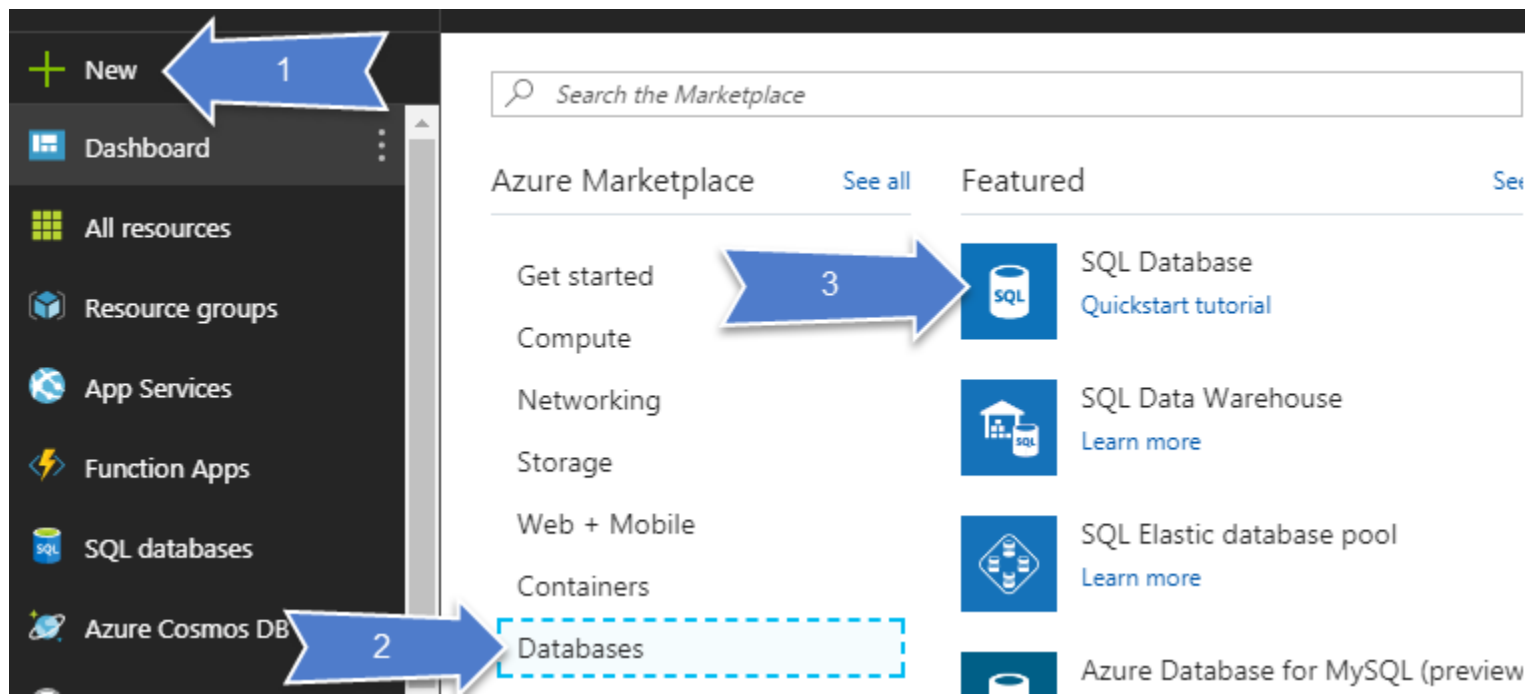
The screenshot shows the Azure portal interface for a storage account. The left sidebar contains navigation links: Overview (selected), Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main content area displays the account details. A red box highlights the 'Subscription ID' field, which contains the value 'b92c0f6e-486f-4ae9-96'. A red arrow points from the text 'Save your Subscription ID as you will need it' below to this field.

Property	Value
Resource group	andrearesourcegroup
Status	Primary: Available, Secondary: Available
Location	East US, West US
Subscription	Free Trial
Subscription ID	b92c0f6e-486f-4ae9-96
Performance/Access tier	Standard/Hot
Replication	Read-access geo-redundant storage (RA-GRS)
Primary blob service endpoint	https://andreanishavablobstorage.blob.core.windows.net/
Secondary blob service endpoint	https://andreanishavablobstorage-secondary.blob.core.windows.net/

Save your Subscription ID as you will need it

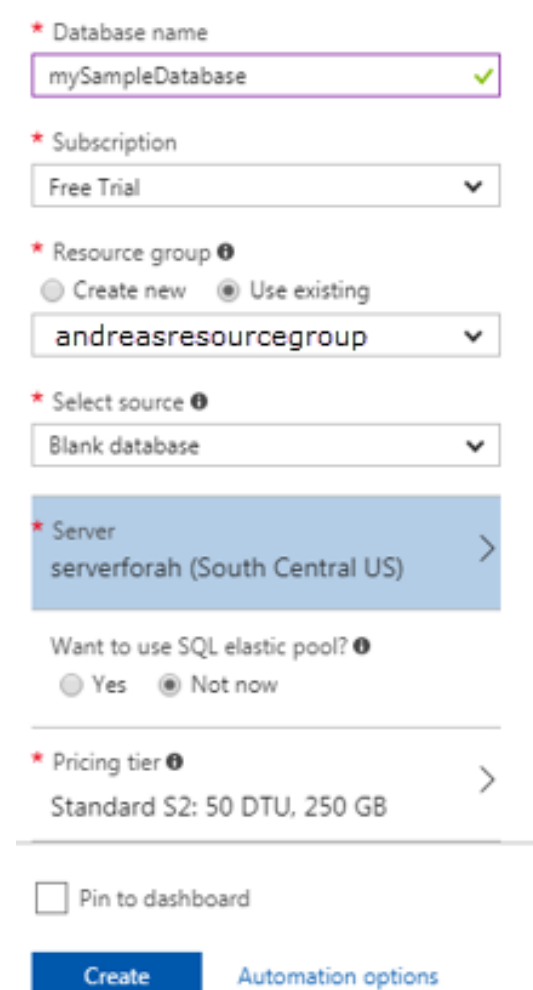
Create an Azure SQL Database

- In Azure select New -> Databases -> SQL Database



Create an Azure SQL Database

- Fill in details for your database
 - Database name
 - Subscription
 - The resource group that you have been using
 - Source



The screenshot shows the 'Create new SQL database' form in the Azure portal. The form is partially filled out with the following details:

- Database name:** mySampleDatabase (with a green checkmark icon)
- Subscription:** Free Trial (dropdown menu)
- Resource group:** andreasresourcegroup (dropdown menu, with radio buttons for 'Create new' and 'Use existing', where 'Use existing' is selected)
- Select source:** Blank database (dropdown menu)
- Server:** serverforah (South Central US) (dropdown menu, highlighted in blue)
- Want to use SQL elastic pool?:** Not now (radio buttons, where 'Not now' is selected)
- Pricing tier:** Standard S2: 50 DTU, 250 GB (dropdown menu)

At the bottom of the form, there is a checkbox for 'Pin to dashboard' which is unchecked. Below the form are two buttons: 'Create' (a blue button) and 'Automation options' (a link).

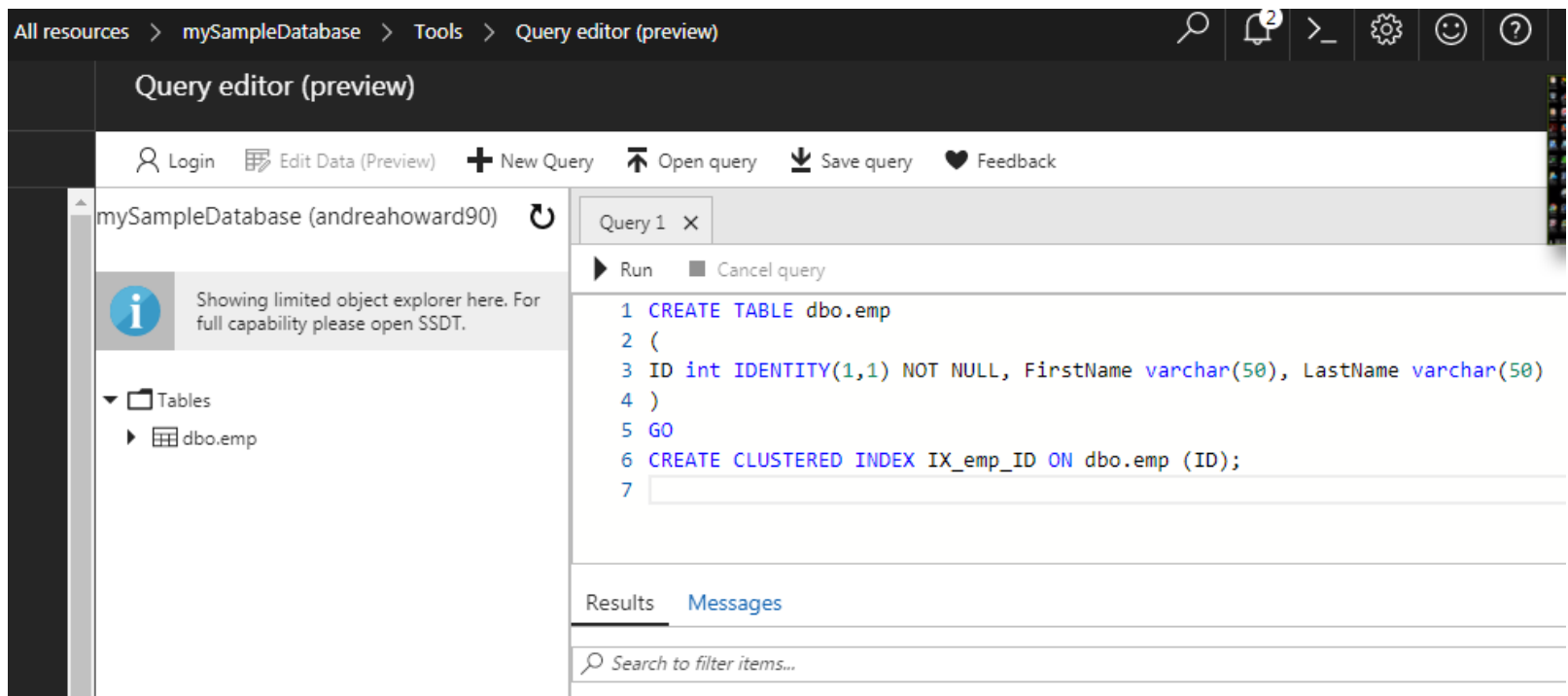
Create an Azure SQL Database

- Select Server and fill in the following:
 - Server Name (any name)
 - Login
 - Password
 - Resource group
- Keep the default storage and select Create

The image shows two side-by-side windows from the Azure portal. The left window, titled 'SQL Database', contains the following fields: 'Database name' (mySampleDatabaseAH), 'Subscription' (Free Trial), 'Resource group' (Use existing, andreasresourcegroup), 'Select source' (Blank database), 'Server' (serverforah (South Central US)), 'Want to use SQL elastic pool?' (Not now), 'Pricing tier' (Standard S2: 50 DTU, 250 GB), and a 'Pin to dashboard' checkbox. The right window, titled 'New server', contains: 'Server name' (serverforah), 'Server admin login' (redacted), 'Password' (redacted), 'Confirm password' (redacted), 'Location' (South Central US), and a checked checkbox for 'Allow azure services to access server'. Both windows have a 'Create' or 'Select' button at the bottom.

Create a table on your SQL Database

- In your resources select your SQL Database and then select Tools
- You can then either use Azure to create your table or Open Visual Studio
- Select the option you would like and run the statement to create the table on the next screen



Create a SQL table in your Database

- Run the following sql statement to create a table

```
CREATE TABLE dbo.emp
```

```
(
```

```
ID int IDENTITY(1,1) NOT NULL, FirstName varchar(50), LastName  
varchar(50)
```

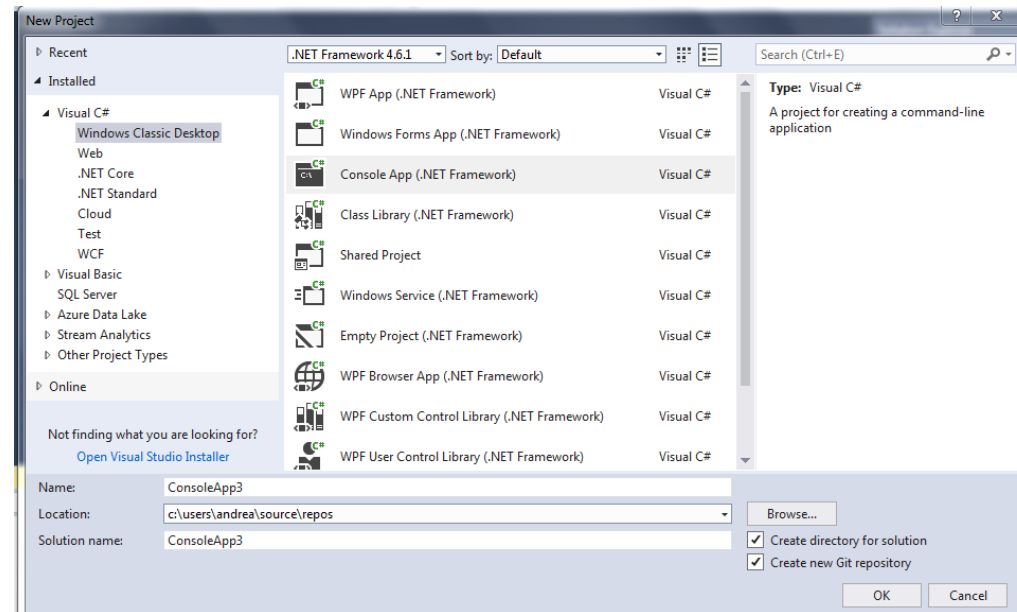
```
)
```

```
GO
```

```
CREATE CLUSTERED INDEX IX_emp_ID ON dbo.emp (ID);
```

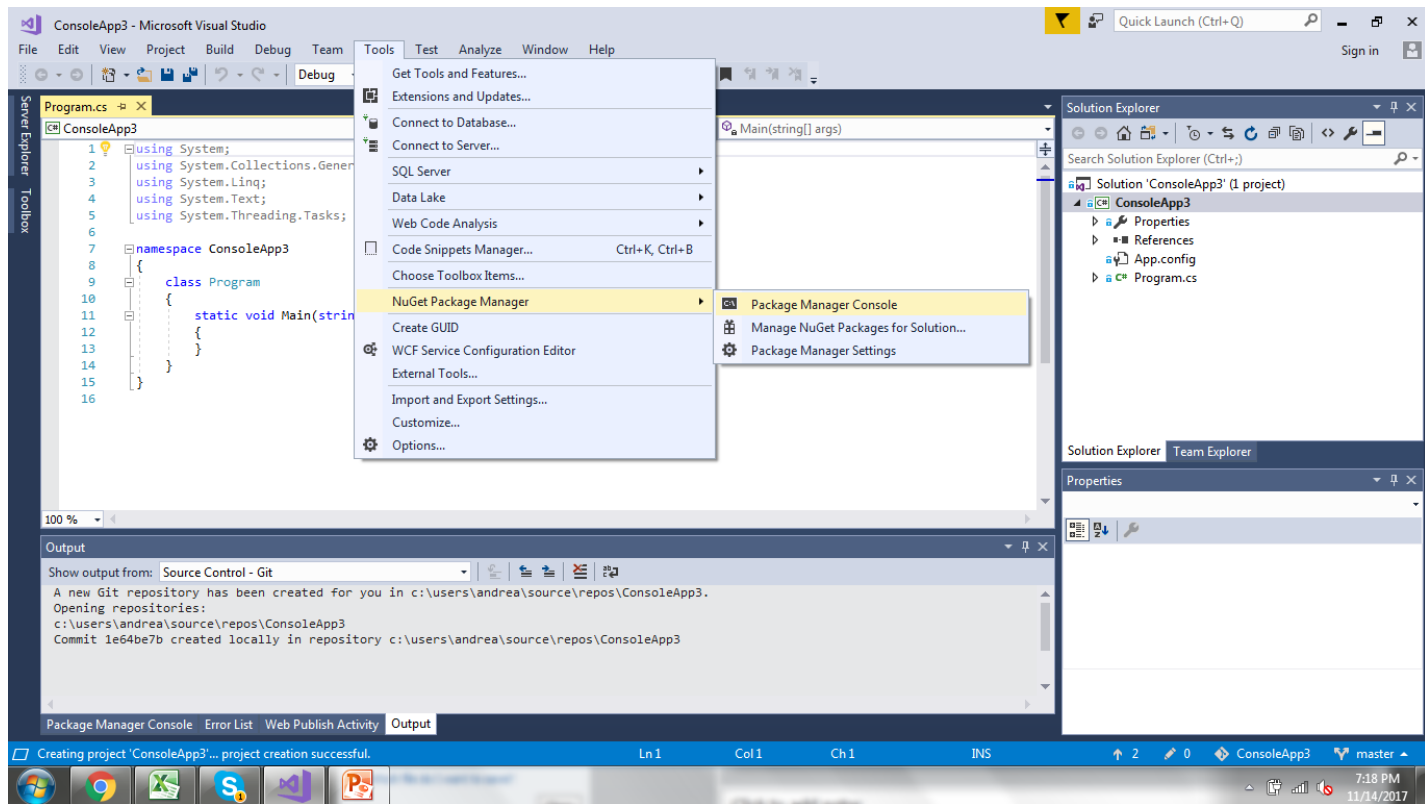
Create a Console App

- Open VS and create a new .NET Console Application
- File -> New -> Project -> Visual C# -> Console App (.NET Framework)
- Enter a name and select 'OK'



Install NuGet packages

- These are NuGet packages that you need to download to create a Data Factory using VS
- In Tools -> NuGet Package Manager -> Package Manager Console



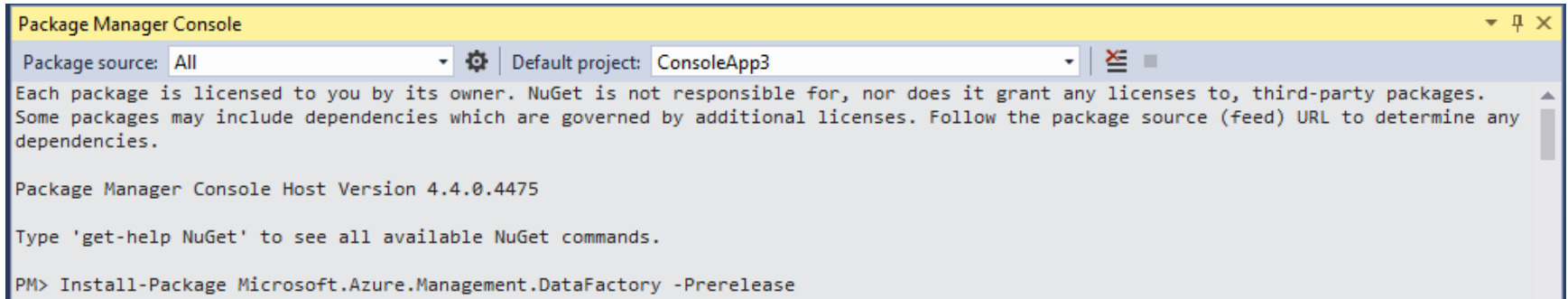
Install NuGet packages

- The Package Manager Console will be displayed most likely at the bottom of your screen and you need to run the three following commands:

Install-Package Microsoft.Azure.Management.DataFactory -Prerelease

Install-Package Microsoft.Azure.Management.ResourceManager -Prerelease

Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory

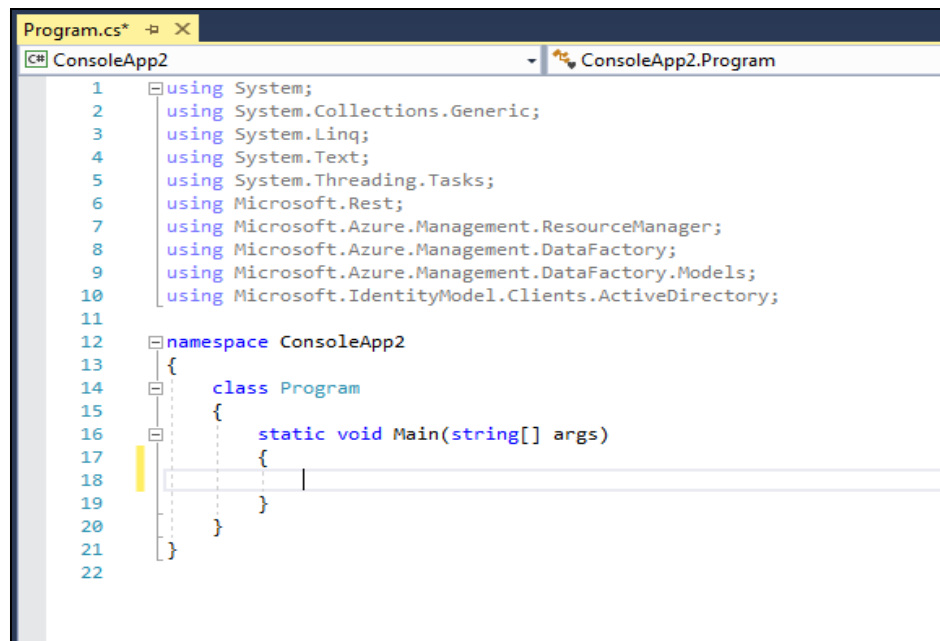


```
Package Manager Console
Package source: All
Default project: ConsoleApp3
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Some packages may include dependencies which are governed by additional licenses. Follow the package source (feed) URL to determine any
dependencies.
Package Manager Console Host Version 4.4.0.4475
Type 'get-help NuGet' to see all available NuGet commands.
PM> Install-Package Microsoft.Azure.Management.DataFactory -Prerelease
```

Specify Required Packages

- Open the Program.cs file of your App and add the below references to the top namespaces

```
using Microsoft.Rest;  
using Microsoft.Azure.Management.ResourceManager;  
using Microsoft.Azure.Management.DataFactory;  
using Microsoft.Azure.Management.DataFactory.Models;  
using Microsoft.IdentityModel.Clients.ActiveDirectory;
```



```
Program.cs* [X]  
[C#] ConsoleApp2 ConsoleApp2.Program  
1 using System;  
2 using System.Collections.Generic;  
3 using System.Linq;  
4 using System.Text;  
5 using System.Threading.Tasks;  
6 using Microsoft.Rest;  
7 using Microsoft.Azure.Management.ResourceManager;  
8 using Microsoft.Azure.Management.DataFactory;  
9 using Microsoft.Azure.Management.DataFactory.Models;  
10 using Microsoft.IdentityModel.Clients.ActiveDirectory;  
11  
12 namespace ConsoleApp2  
13 {  
14     class Program  
15     {  
16         static void Main(string[] args)  
17         {  
18             |  
19         }  
20     }  
21 }  
22
```


Set up Variables

- In the Main of your file you will need to add the following variables to create your Data Factory

```
// Set variables
string tenantID = "<your tenant ID>";
string applicationId = "<your application ID>";
string authenticationKey = "<your authentication key for the application>";
string subscriptionId = "<your subscription ID where the data factory
resides>";
string resourceGroup = "<your resource group where the data factory resides>";
string region = "East US 2";
string dataFactoryName = "<specify the name of data factory to create. It must
be globally unique.>";
```

Note: you can find where you would get your variables on each of the below slides

TenantID – found on slide 15

applicationID – found on slide 12

Authenticationkey – found on slide 7

subscriptionID – found on slide 22

resourceGroup (the name of the resource group you have used throughout) – found on slide 4

dataFactoryName – you create a unique name

Enter your IDs, Names

```
namespace ConsoleApp2
{
    class Program
    {
        static void Main(string[] args)
        {
            // Set variables
            string tenantID = "2cc424bd-7c70-4ef2-a8a6-e908829fc5d5";
            string applicationId = "b07ae379-22af-4b19-bb40-c6517c9b3bc3";
            string authenticationKey = "g7/wn9A/JUf+dXRFoTxRiGYmkJb0FsLZVV2Xl9kSnho=";
            string subscriptionId = "b92c0f6e-486f-4ae9-96af-";
            string resourceGroup = "andreaResourceGroup";

            string region = "East US";
            string dataFactoryName = "ah90dataFactory";
        }
    }
}
```

Specify Storage Defaults

- In the main of your file you then need to specify your Blob info

```
// Specify the source Azure Blob information
string storageAccount = "<your storage account name to copy
data>";
string storageKey = "<your storage account key>";
string inputBlobPath = "adfv2tutorial/";
string inputBlobName = "inputEmp.txt";
```

Default Values

```
static void Main(string[] args)
{
    // Set variables
    string tenantID = "2cc424bd-7c70-4ef2-a8a6-e908829fc5d5";
    string applicationId = "b07ae379-22af-4b19-bb40-c6517c9b3bc3";
    string authenticationKey = "g7/wn9A/JUf+dXRFoTxRjGYmkJh0F<LZVV2Xl9kSnho=";
    string subscriptionId = "b92c0f6e-486f-4ae9-96af-";
    string resourceGroup = "andreaResourceGroup";

    string region = "East US";
    string dataFactoryName = "ah90dataFacory";

    // Specify the source Azure Blob information
    string storageAccount = "andreaishavablobstorage";
    string storageKey = "MwB+17Qge+ChP43R5vYb2pEtCKKao6aV/ZfKqfefzVcLfHvGHMIXI6o6nI8o/zAH1KkJiZ5WkzR7GM7RRjIYaQ==";
    string inputBlobPath = "andreasblobcontainer/adfv2tutorial/";
    string inputBlobName = "inputEmp.txt";
}
```

Specify the Azure SQL DB Sink Info

- In the main of your file Specify the Azure SQL DB Info

```
// Specify the sink Azure SQL Database information
```

```
string azureSqlConnString = "Server=tcp:<your server  
name>.database.windows.net,1433;Database=<your database  
name>;User ID=<your username>@<your server  
name>;Password=<your  
password>;Trusted_Connection=False;Encrypt=True;Connection  
Timeout=30";  
  
string azureSqlTableName = "dbo.emp";  
string storageLinkedServiceName = "AzureStorageLinkedService";  
string sqlDbLinkedServiceName = "AzureSqlDbLinkedService";  
string blobDatasetName = "BlobDataset";  
string sqlDatasetName = "SqlDataset";  
string pipelineName = "Adfv2TutorialBlobToSqlCopy";
```

Specify the Azure SQL DB Info

```
string subscriptionId = "b92c0f6e-486f-4ae9-96af-";
string resourceGroup = "andreaResourceGroup";

string region = "East US";
string dataFactoryName = "ah90dataFacory";

// Specify the source Azure Blob information
string storageAccount = "andreanishavablobstorage";
string storageKey = "MwB+17Qge+ChP43R5vYb2pEtCKKao6aV/ZfKqfefzVcLfHVGHMIXI6o6nI8o/zAH1Kk3iZ5WKzR7GM7RRjIYaQ==";
string inputBlobPath = "andreasblobcontainer/adfv2tutorial/";
string inputBlobName = "inputEmp.txt";

// Specify the sink Azure SQL Database information
string azureSqlConnString = "Server=tcp:serverforah.database.windows.net,1433;Database=mySampleDatabase;User ID=
string azureSqlTableName = "dbo.emp";

string storageLinkedServiceName = "AzureStorageLinkedService";
string sqlDbLinkedServiceName = "AzureSqlDbLinkedService";
string blobDatasetName = "BlobDataset";
string sqlDatasetName = "SqlDataset";
string pipelineName = "Adfv2TutorialBlobToSqlCopy";
```

Create a data factory management client

- In the main of your file you will need to add the following code to create a Data Factory Management Client

```
// Authenticate and create a data factory management client
var context = new AuthenticationContext("https://login.windows.net/" +
tenantID);
ClientCredential cc = new ClientCredential(applicationId,
authenticationKey);
AuthenticationResult result =
context.AcquireTokenAsync("https://management.azure.com/", cc).Result;
ServiceClientCredentials cred = new TokenCredentials(result.AccessToken);
var client = new DataFactoryManagementClient(cred) { SubscriptionId =
subscriptionId };
```

Create a data factory management client

```
// Specify the sink Azure SQL Database information
string azureSqlConnString = "Server=tcp:serverforah.database.windows.net,1433;Database=mySampleDatabase;User ID=
string azureSqlTableName = "dbo.emp";

string storagelinkedServiceName = "AzureStorageLinkedService";
string sqlDbLinkedServiceName = "AzureSqlDbLinkedService";
string blobDatasetName = "BlobDataset";
string sqlDatasetName = "SqlDataset";
string pipelineName = "Adfv2TutorialBlobToSqlCopy";

// Authenticate and create a data factory management client
var context = new AuthenticationContext("https://login.windows.net/" + tenantID);
ClientCredential cc = new ClientCredential(applicationId, authenticationKey);
AuthenticationResult result = context.AcquireTokenAsync("https://management.azure.com/", cc).Result;
ServiceClientCredentials cred = new TokenCredentials(result.AccessToken);
var client = new DataFactoryManagementClient(cred) { SubscriptionId = subscriptionId };
```


Create the Data factory

- Add the following to your main to create the data factory:

```
// Create a data factory
Console.WriteLine("Creating a data factory " +
dataFactoryName + "..."); Factory dataFactory = new Factory
{
    Location = region, Identity = new FactoryIdentity()
};
client.Factories.CreateOrUpdate(resourceGroup,
dataFactoryName, dataFactory);
Console.WriteLine(SafeJsonConvert.SerializeObject(dataFactory
, client.SerializationSettings));

while (client.Factories.Get(resourceGroup,
dataFactoryName).ProvisioningState == "PendingCreation")
{
    System.Threading.Thread.Sleep(1000);
}
```

Create the Data factory

```
// Create a data factory
Console.WriteLine("Creating a data factory " + dataFactoryName + "...");
Factory dataFactory = new Factory
{
    Location = region,
    Identity = new FactoryIdentity()
};
client.Factories.CreateOrUpdate(resourceGroup, dataFactoryName, dataFactory);
Console.WriteLine(SafeJsonConvert.SerializeObject(dataFactory, client.SerializationSettings));

while (client.Factories.Get(resourceGroup, dataFactoryName).ProvisioningState == "PendingCreation")
{
    System.Threading.Thread.Sleep(1000);
}
```

Create an Azure Storage linked service

- Add the following to your main to create the linked service:

```
// Create an Azure Storage linked service
```

```
Console.WriteLine("Creating linked service " +  
storageLinkedServiceName + "...");
```

```
LinkedServiceResource storageLinkedService = new  
LinkedServiceResource(  
    new AzureStorageLinkedService  
    {  
        ConnectionString = new  
SecureString("DefaultEndpointsProtocol=https;AccountName=" +  
storageAccount + ";AccountKey=" + storageKey)  
    }  
);  
client.LinkedServices.CreateOrUpdate(resourceGroup, dataFactoryName,  
storageLinkedServiceName, storageLinkedService);  
Console.WriteLine(SafeJsonConvert.SerializeObject(storageLinkedService  
, client.SerializationSettings));
```

Create an Azure Storage linked service

```
// Create a data factory
Console.WriteLine("Creating a data factory " + dataFactoryName + "...");
Factory dataFactory = new Factory
{
    Location = region,
    Identity = new FactoryIdentity()
};
client.Factories.CreateOrUpdate(resourceGroup, dataFactoryName, dataFactory);
Console.WriteLine(SafeJsonConvert.SerializeObject(dataFactory, client.SerializationSettings));

while (client.Factories.Get(resourceGroup, dataFactoryName).ProvisioningState == "PendingCreation")
{
    System.Threading.Thread.Sleep(1000);
}

// Create an Azure Storage linked service
Console.WriteLine("Creating linked service " + storageLinkedServiceName + "...");

LinkServiceResource storageLinkedService = new LinkServiceResource(
    new AzureStorageLinkedService
    {
        ConnectionString = new SecureString("DefaultEndpointsProtocol=https;AccountName=" + storageAccount + ";A");
    }
);
client.LinkedServices.CreateOrUpdate(resourceGroup, dataFactoryName, storageLinkedServiceName, storageLinkedService);
Console.WriteLine(SafeJsonConvert.SerializeObject(storageLinkedService, client.SerializationSettings));
```

Create an Azure SQL Database linked service

Add the following to your main to create the next linked service:

```
// Create an Azure SQL Database linked service
Console.WriteLine("Creating linked service " +
    sqlDbLinkedServiceName + "...");

LinkedServiceResource sqlDbLinkedService = new
LinkedServiceResource(
    new AzureSqlDatabaseLinkedService
    {
        ConnectionString = new SecureString(azureSqlConnString)
    }
);
client.LinkedServices.CreateOrUpdate(resourceGroup,
dataFactoryName, sqlDbLinkedServiceName, sqlDbLinkedService);
Console.WriteLine(SafeJsonConvert.SerializeObject(sqlDbLinkedService, client.SerializationSettings));
```

Create an Azure SQL Database linked service

```
// Create an Azure Storage linked service
Console.WriteLine("Creating linked service " + storageLinkedServiceName + "...");

LinkedServiceResource storageLinkedService = new LinkedServiceResource(
    new AzureStorageLinkedService
    {
        ConnectionString = new SecureString("DefaultEndpointsProtocol=https;AccountName=" + storageAccount + ";A
    });
client.LinkedServices.CreateOrUpdate(resourceGroup, dataFactoryName, storageLinkedServiceName, storageLinkedService);
Console.WriteLine(SafeJsonConvert.SerializeObject(storageLinkedService, client.SerializationSettings));

// Create an Azure SQL Database linked service
Console.WriteLine("Creating linked service " + sqlDbLinkedServiceName + "...");

LinkedServiceResource sqlDbLinkedService = new LinkedServiceResource(
    new AzureSqlDatabaseLinkedService
    {
        ConnectionString = new SecureString(azureSqlConnString)
    });
client.LinkedServices.CreateOrUpdate(resourceGroup, dataFactoryName, sqlDbLinkedServiceName, sqlDbLinkedService);
Console.WriteLine(SafeJsonConvert.SerializeObject(sqlDbLinkedService, client.SerializationSettings));
```

Create an Azure Blob dataset

```
// Create a Azure Blob dataset
Console.WriteLine("Creating dataset " + blobDatasetName + "...");
DatasetResource blobDataset = new DatasetResource(
    new AzureBlobDataset
    {
        LinkedServiceName = new LinkedServiceReference
        {
            ReferenceName = storageLinkedServiceName
        },
        FolderPath = inputBlobPath,
        FileName = inputBlobName,
        Format = new TextFormat { ColumnDelimiter = "|" },
        Structure = new List<DatasetDataElement>
        {
            new DatasetDataElement
            {
                Name = "FirstName", Type = "String"
            },
            new DatasetDataElement
            {
                Name = "LastName", Type = "String"
            }
        }
    }
);
client.Datasets.CreateOrUpdate(resourceGroup, dataFactoryName, blobDatasetName, blobDataset);
Console.WriteLine(SafeJsonConvert.SerializeObject(blobDataset, client.SerializationSettings));
```

Create an Azure Blob dataset

```
// Create a Azure Blob dataset
Console.WriteLine("Creating dataset " + blobDatasetName + "...");
DatasetResource blobDataset = new DatasetResource(
    new AzureBlobDataset
    {
        LinkedServiceName = new LinkedServiceReference
        {
            ReferenceName = storageLinkedServiceName
        },
        FolderPath = inputBlobPath,
        FileName = inputBlobName,
        Format = new TextFormat { ColumnDelimiter = "|" },
        Structure = new List<DatasetDataElement>
        {
            new DatasetDataElement
            {
                Name = "FirstName",
                Type = "String"
            },
            new DatasetDataElement
            {
                Name = "LastName",
                Type = "String"
            }
        }
    }
);
client.Datasets.CreateOrUpdate(resourceGroup, dataFactoryName, blobDatasetName, blobDataset);
Console.WriteLine(SafeJsonConvert.SerializeObject(blobDataset, client.SerializationSettings)).
```


Create an Azure SQL Database dataset

```
// Create a Azure SQL Database dataset
Console.WriteLine("Creating dataset " + sqlDatasetName + "...");
DatasetResource sqlDataset = new DatasetResource(
    new AzureSqlTableDataset
    {
        LinkedServiceName = new LinkedServiceReference
        {
            ReferenceName = sqlDbLinkedServiceName
        },
        TableName = azureSqlTableName
    }
);
client.Datasets.CreateOrUpdate(resourceGroup, dataFactoryName,
sqlDatasetName, sqlDataset);
Console.WriteLine(SafeJsonConvert.SerializeObject(sqlDataset,
client.SerializationSettings));
```

Create an Azure SQL Database dataset

```
// Create a Azure SQL Database dataset
Console.WriteLine("Creating dataset " + sqlDatasetName + "...");
DatasetResource sqlDataset = new DatasetResource(
    new AzureSqlTableDataset
    {
        LinkedServiceName = new LinkedServiceReference
        {
            ReferenceName = sqlDbLinkedServiceName
        },
        TableName = azureSqlTableName
    }
);
client.Datasets.CreateOrUpdate(resourceGroup, dataFactoryName, sqlDatasetName, sqlDataset);
Console.WriteLine(SafeJsonConvert.SerializeObject(sqlDataset, client.SerializationSettings));
```

Create a pipeline for your copy from blob to SQL DB sink

// Create a pipeline with copy activity

```
Console.WriteLine("Creating pipeline " + pipelineName + "...");
PipelineResource pipeline = new PipelineResource
{
    Activities = new List<Activity>
    {
        new CopyActivity
        {
            Name = "CopyFromBlobToSQL", Inputs = new List<DatasetReference>
            {
                new DatasetReference()
                {
                    ReferenceName = blobDatasetName
                }
            },
            Outputs = new List<DatasetReference>
            {
                new DatasetReference
                {
                    ReferenceName = sqlDatasetName
                }
            },
            Source = new BlobSource { }, Sink = new SqlSink { }
        }
    }
};

client.Pipelines.CreateOrUpdate(resourceGroup, dataFactoryName, pipelineName, pipeline);
Console.WriteLine(SafeJsonConvert.SerializeObject(pipeline, client.SerializationSettings));
```

Create a pipeline

```
// Create a pipeline with copy activity
Console.WriteLine("Creating pipeline " + pipelineName + "...");
PipelineResource pipeline = new PipelineResource
{
    Activities = new List<Activity>
    {
        new CopyActivity
        {
            Name = "CopyFromBlobToSQL",
            Inputs = new List<DatasetReference>
            {
                new DatasetReference()
                {
                    ReferenceName = blobDatasetName
                }
            },
            Outputs = new List<DatasetReference>
            {
                new DatasetReference
                {
                    ReferenceName = sqlDatasetName
                }
            },
            Source = new BlobSource { },
            Sink = new SqlSink { }
        }
    }
};
client.Pipelines.CreateOrUpdate(resourceGroup, dataFactoryName, pipelineName, pipeline);
```

Create a pipeline run

```
// Create a pipeline run
Console.WriteLine("Creating pipeline run...");
CreateRunResponse runResponse =
client.Pipelines.CreateRunWithHttpMessagesAsync(resourceGroup,
dataFactoryName, pipelineName).Result.Body;
Console.WriteLine("Pipeline run ID: " + runResponse.RunId);
```

Create a pipeline run

```
// Create a pipeline run
Console.WriteLine("Creating pipeline run...");
CreateRunResponse runResponse = client.Pipelines.CreateRunWithHttpMessagesAsync(resourceGroup, dataFactoryName,
Console.WriteLine("Pipeline run ID: " + runResponse.RunId);
```

Monitor the pipeline run

```
// Monitor the pipeline run
Console.WriteLine("Checking pipeline run status...");
PipelineRun pipelineRun;
while (true)
{
    pipelineRun = client.PipelineRuns.Get(resourceGroup,
dataFactoryName, runResponse.RunId);
    Console.WriteLine("Status: " + pipelineRun.Status);
    if (pipelineRun.Status == "InProgress")
        System.Threading.Thread.Sleep(15000);
    else
        break;
}
```

Monitor the pipeline run

```
// Monitor the pipeline run
Console.WriteLine("Checking pipeline run status...");
PipelineRun pipelineRun;
while (true)
{
    pipelineRun = client.PipelineRuns.Get(resourceGroup, dataFactoryName, runResponse.RunId);
    Console.WriteLine("Status: " + pipelineRun.Status);
    if (pipelineRun.Status == "InProgress")
        System.Threading.Thread.Sleep(15000);
    else
        break;
}
```


Monitor the pipeline run

```
// Check the copy activity run details
Console.WriteLine("Checking copy activity run details...");

List<ActivityRun> activityRuns = client.ActivityRuns.ListByPipelineRun(
    resourceGroup, dataFactoryName, runResponse.RunId,
    DateTime.UtcNow.AddMinutes(-10), DateTime.UtcNow.AddMinutes(10)).ToList();

if (pipelineRun.Status == "Succeeded")
{
    Console.WriteLine(activityRuns.First().Output);
}
else
    Console.WriteLine(activityRuns.First().Error);

Console.WriteLine("\nPress any key to exit...");
Console.ReadKey();
```

Monitor the pipeline run

```
// Check the copy activity run details
Console.WriteLine("Checking copy activity run details...");

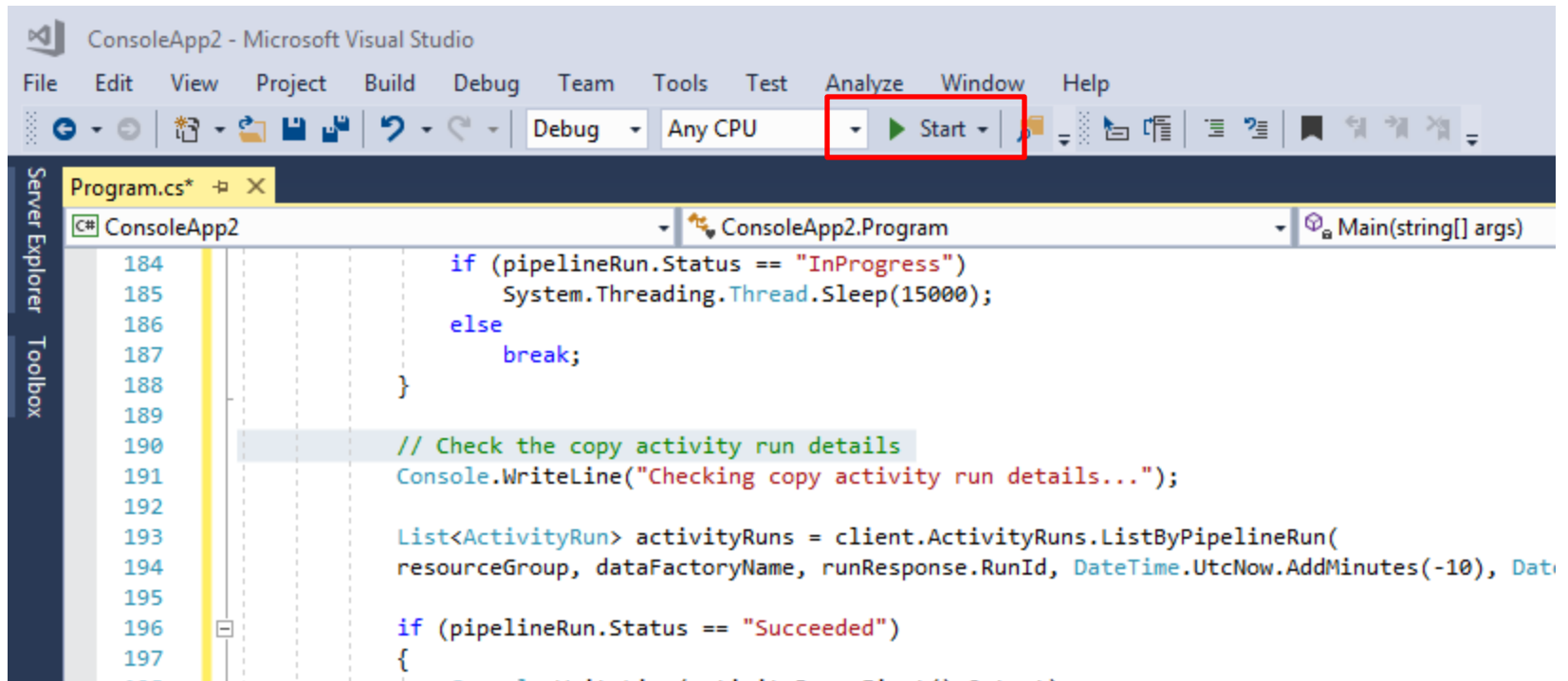
List<ActivityRun> activityRuns = client.ActivityRuns.ListByPipelineRun(
resourceGroup, dataFactoryName, runResponse.RunId, DateTime.UtcNow.AddMinutes(-10), DateTime.UtcNow.AddMinutes(10));

if (pipelineRun.Status == "Succeeded")
{
    Console.WriteLine(activityRuns.First().Output);
}
else
    Console.WriteLine(activityRuns.First().Error);

Console.WriteLine("\nPress any key to exit...");
Console.ReadKey();
```

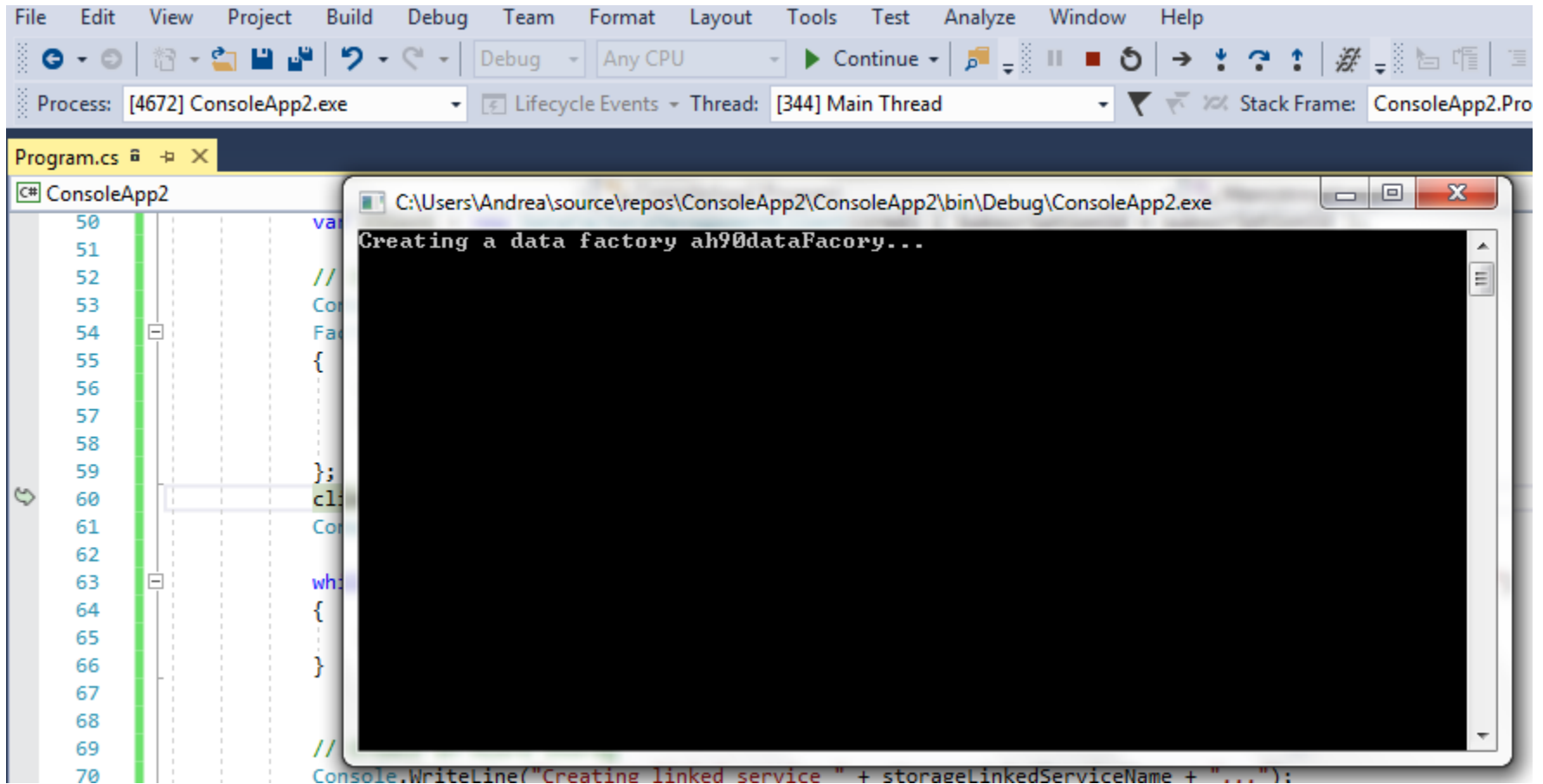
Run the Code

- Finally build the code by selecting the Green triangle that says Start



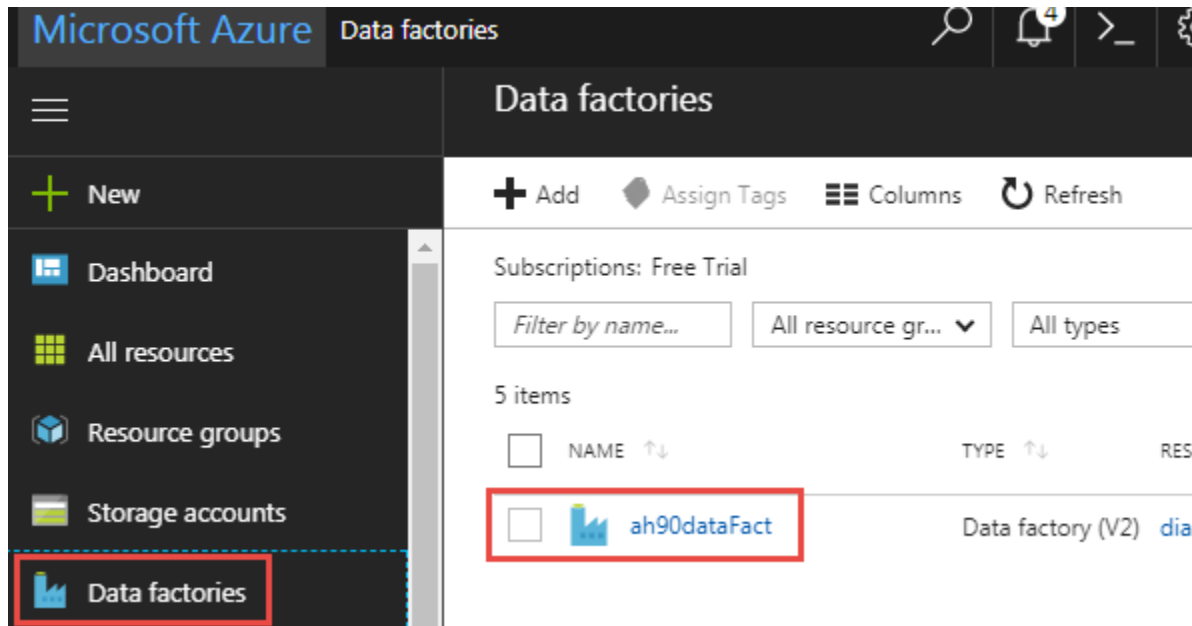
Run the Code

- A command prompt will pop up for you to watch the build



View the Run in Azure

- In Azure select Data Factories then select the Data Factory you just created and then you can see details about the data factory



After your run check your SQL Database

- Within the portal in SQL databases select your database that you just created
- Select Tools editor to query your table

The screenshot shows a web-based SQL database portal. At the top, there is a navigation bar with icons and labels for 'Login', 'Edit Data (Preview)', 'New Query', 'Open query', 'Save query', and 'Feedback'. Below this, the main interface is divided into three sections. On the left is the 'Object Explorer' showing a tree view of the database 'nySampleDatabase (andreahoward90)'. It contains a 'Tables' folder with a table 'dbo.emp' listed, showing its columns: 'ID (int, not null)', 'FirstName (varchar, null)', and 'LastName (varchar, null)'. A message box above the table list states: 'Showing limited object explorer here. For full capability please open SSDT.' The middle section is the 'Query Editor' for 'Query 1', containing the SQL query: '1 select * from emp'. Above the query is a toolbar with 'Run' and 'Cancel query' buttons. The right section is the 'Results' pane, which is active and displays the query output as a table with columns 'ID', 'FIRSTNAME', and 'LASTNAME'. The results show two rows: (1, John, Doe) and (2, Jane, Doe). A search bar at the top of the results pane says 'Search to filter items...'. The bottom of the interface has a footer with the text '@Andrea Hatch, Nishava Inc'.

nySampleDatabase (andreahoward90)

Showing limited object explorer here. For full capability please open SSDT.

Tables

- dbo.emp
 - ID (int, not null)
 - FirstName (varchar, null)
 - LastName (varchar, null)

Query 1 X

Run Cancel query

```
1 select * from emp
2
3
```

Results Messages

Search to filter items...

ID	FIRSTNAME	LASTNAME
1	John	Doe
2	Jane	Doe

Summary

- We created a simple Data Factory in VS using .Net for a small data file which was uploaded to a Blob and watched the progress of the job: *Copy the data file to our Blob Sink.*
- We used Azure to create:
 - a database
 - storage account
 - Blob
 - Active directory