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

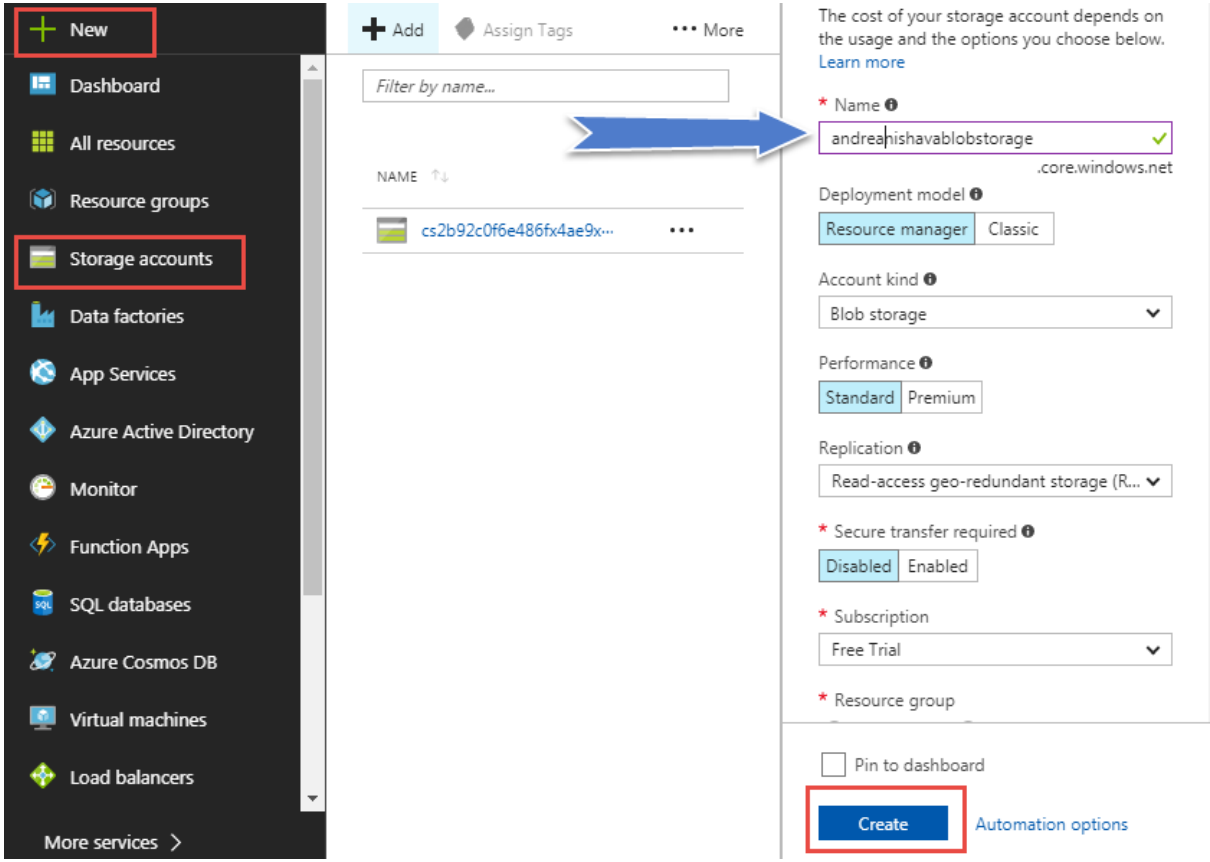
Create a Storage Account

Steps:

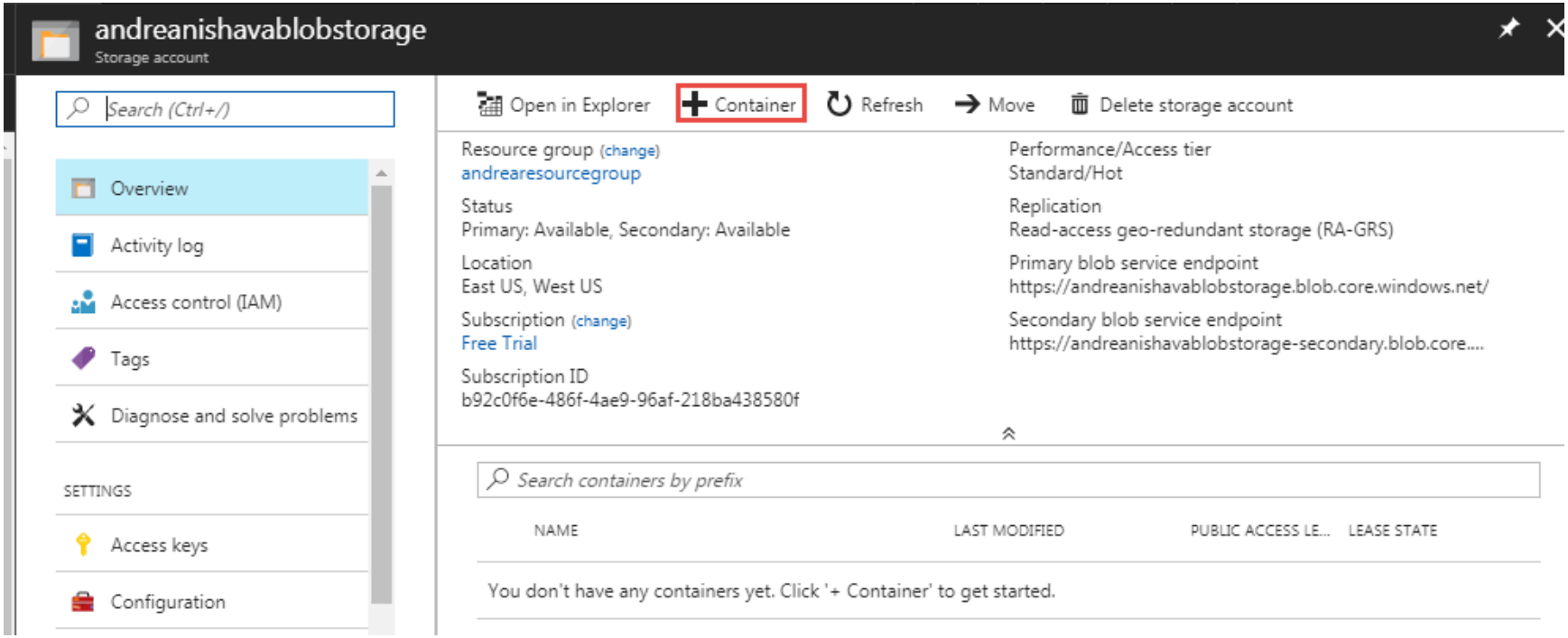
1. In Azure select New ->Storage accounts
2. Name: Unique within Azure.
3. Account Kind: Blob or General Purpose
4. Create new resource group or use an existing one.
5. Location: select any location. (I use East US).

Optional: Pin to Dashboard

6. Select Create.

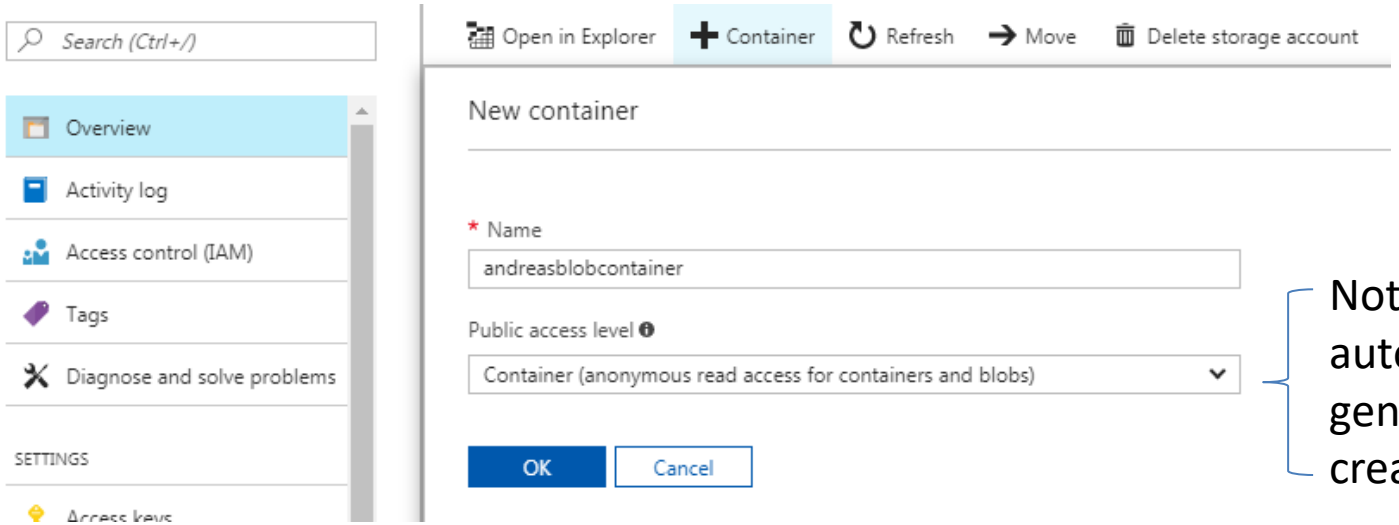


Create a Container for your Storage

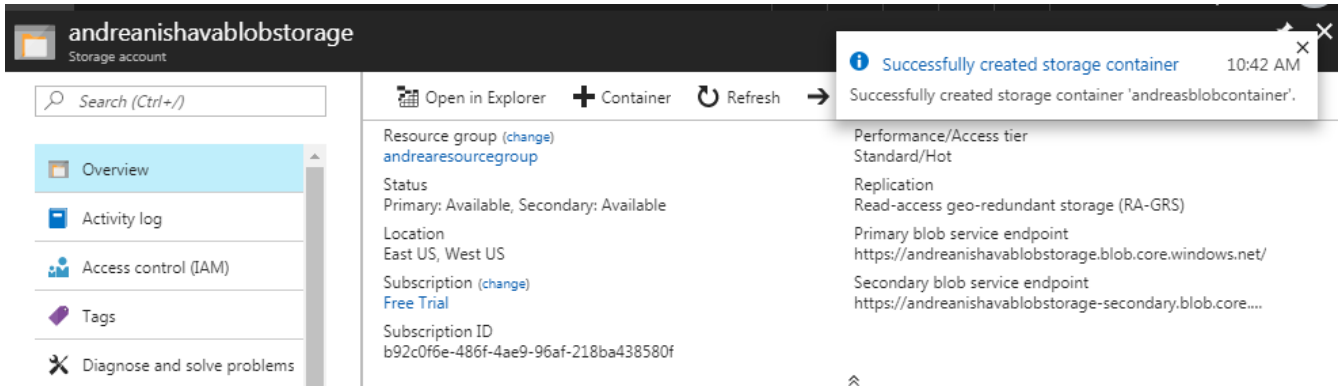


Create a Container for your Blob Storage

- You must set the access level to Container

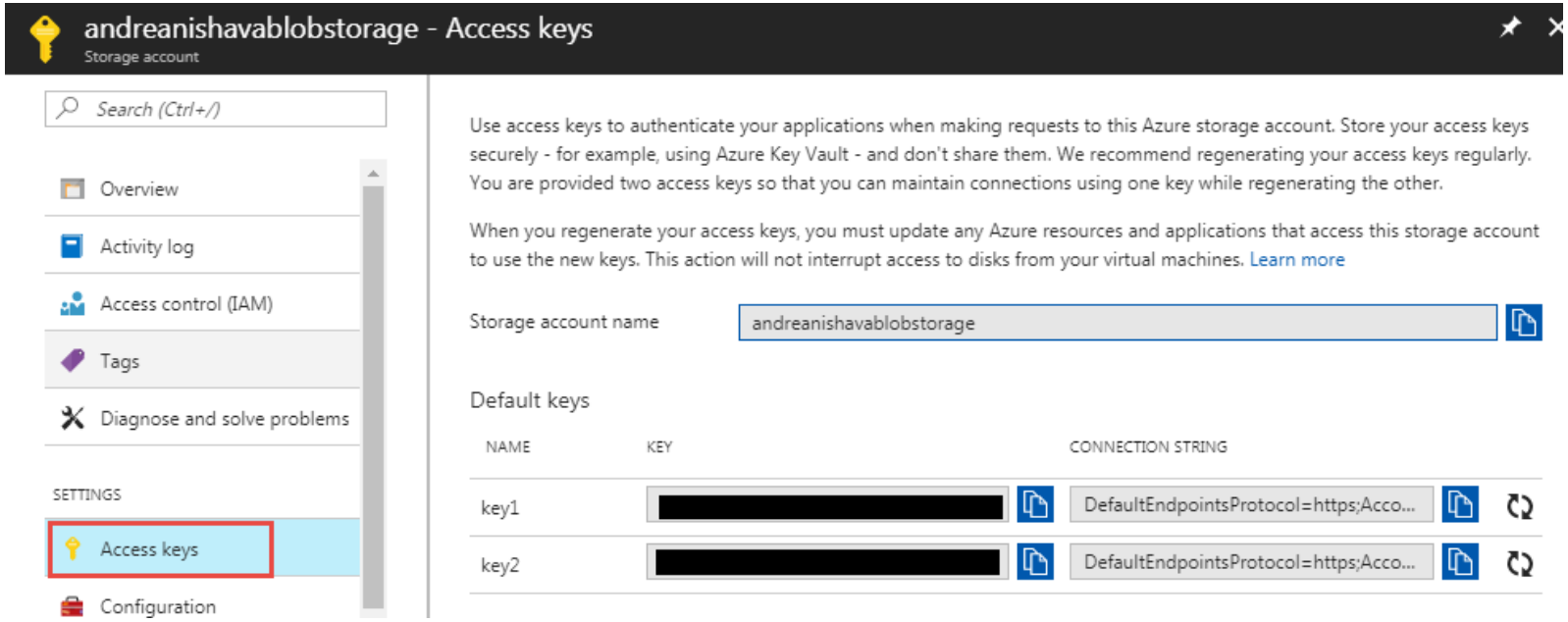


Note: Keys are automatically generated when you create your storage.



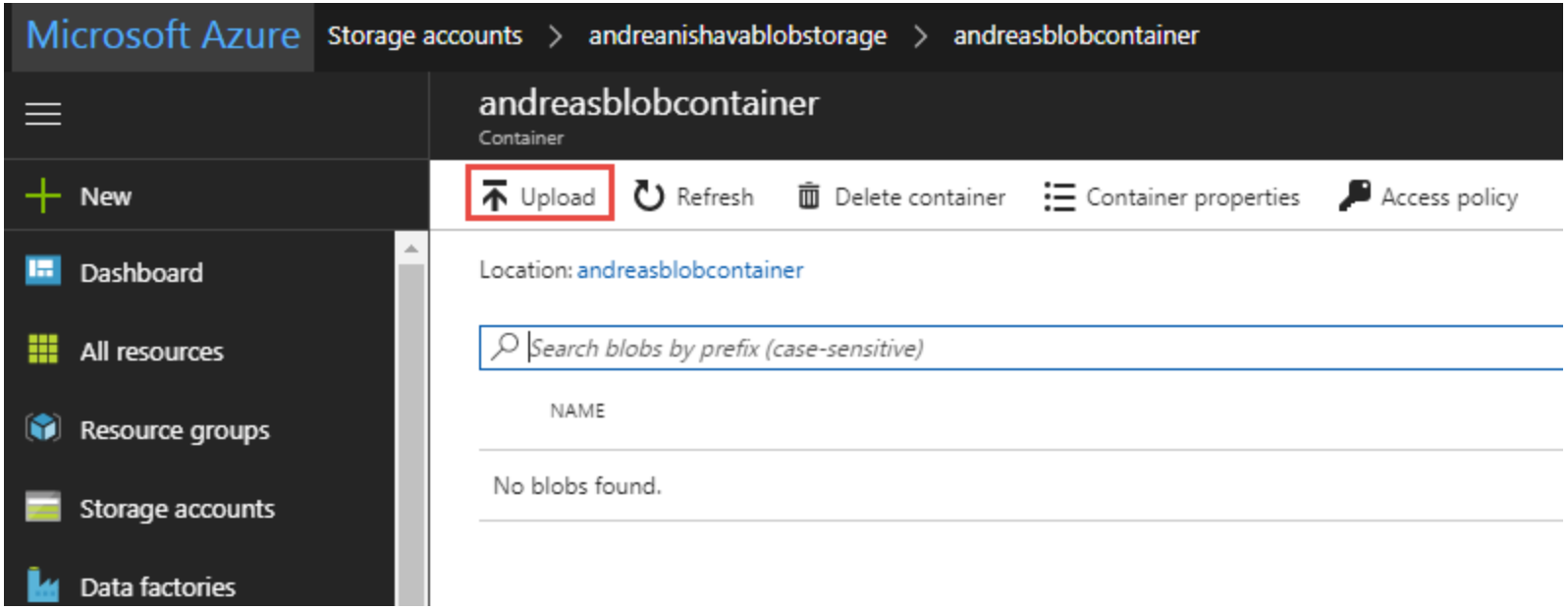
Storage Access Keys

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



Upload Data File to your Blob

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

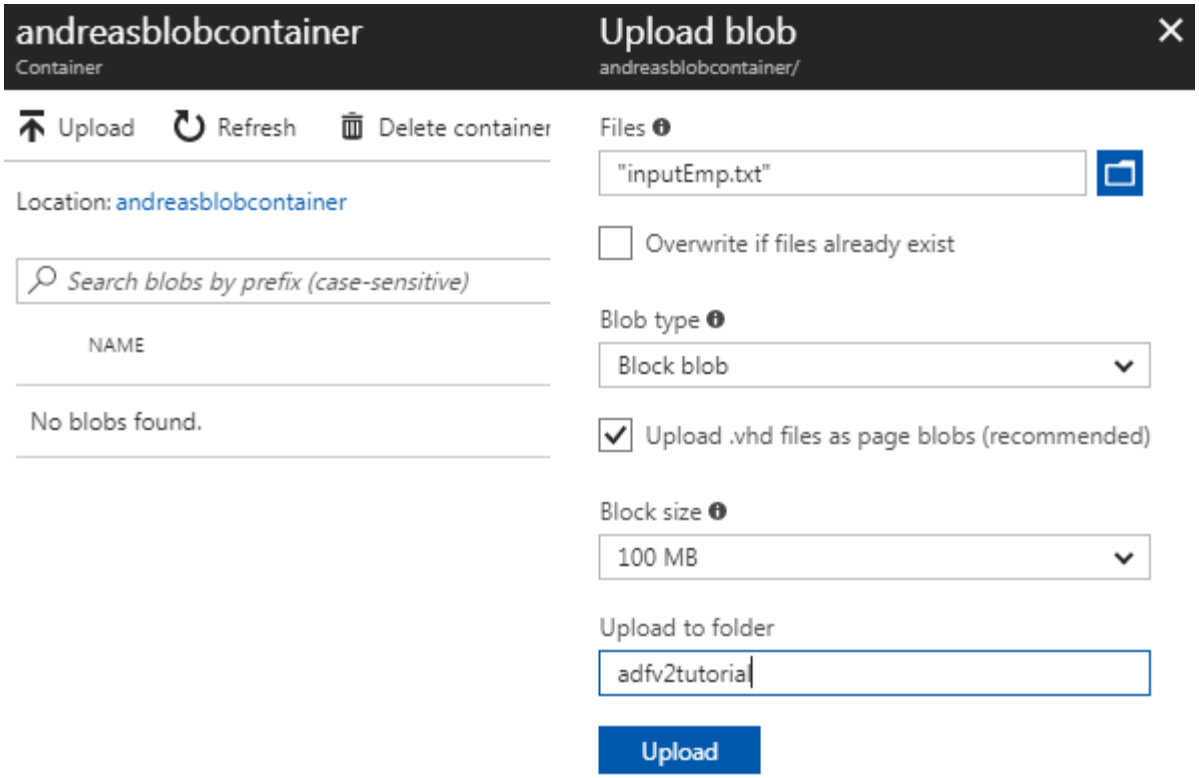


Upload Data File to your Blob

- Create a data file named **inputEmp.txt**
  - **Add some data to file such as:**

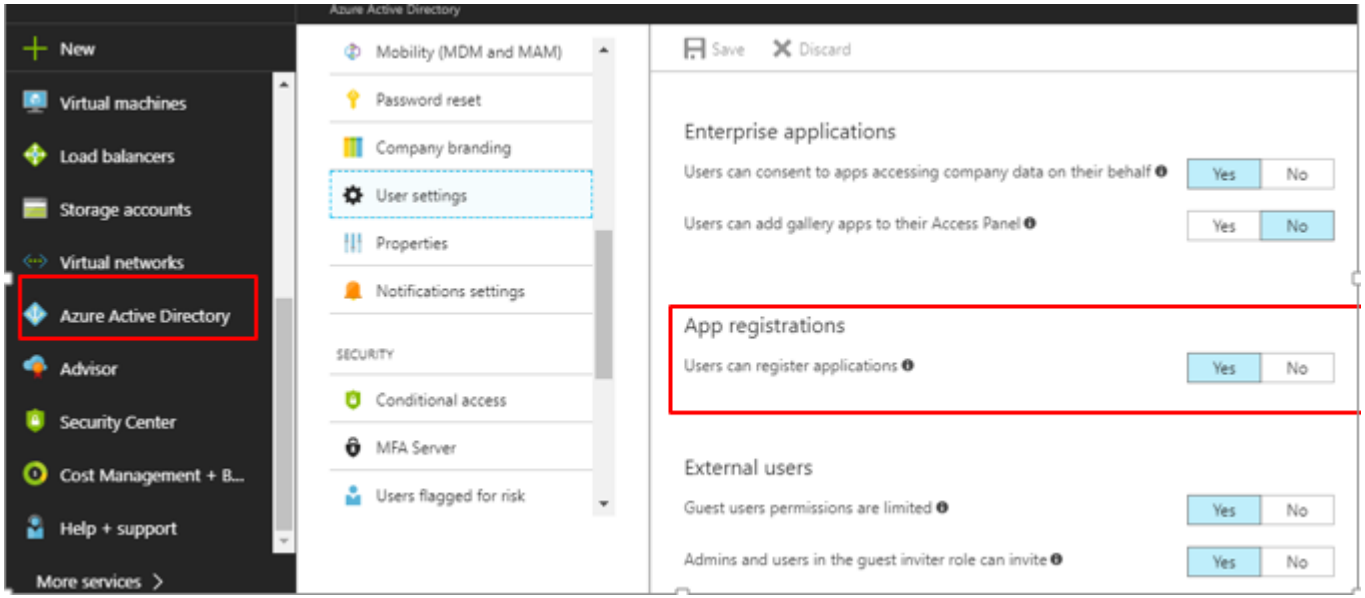
  John|Doe

  Jane|Doe
- Load this into your Blob by selecting the File and selecting Advanced and Upload it to a folder
- Select ‘Upload’



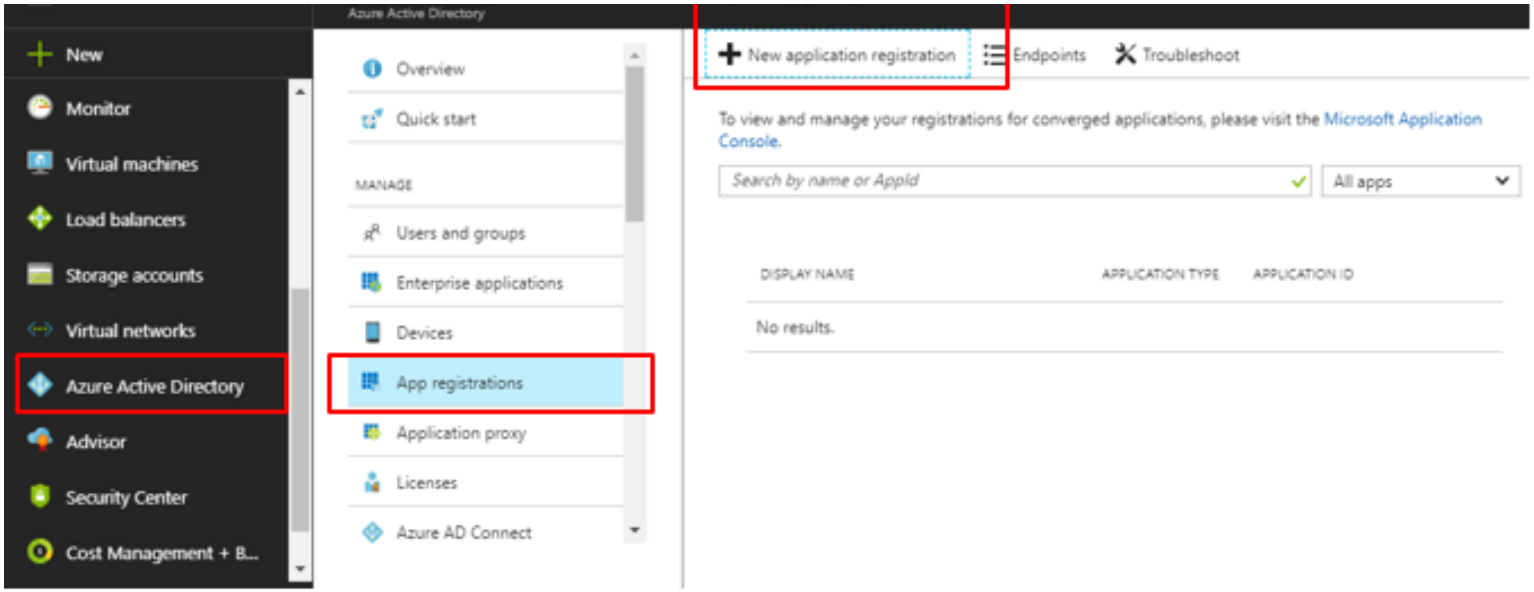
Create an application in Azure Active Directory

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



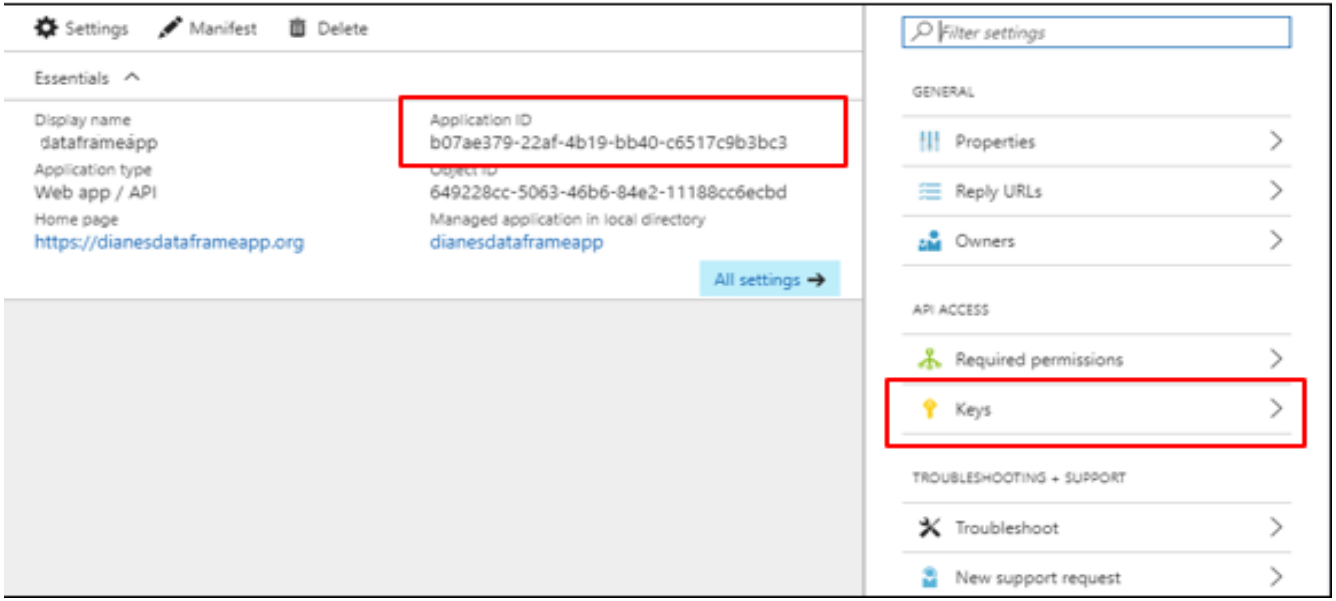
Create an Azure Active Directory application

- 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



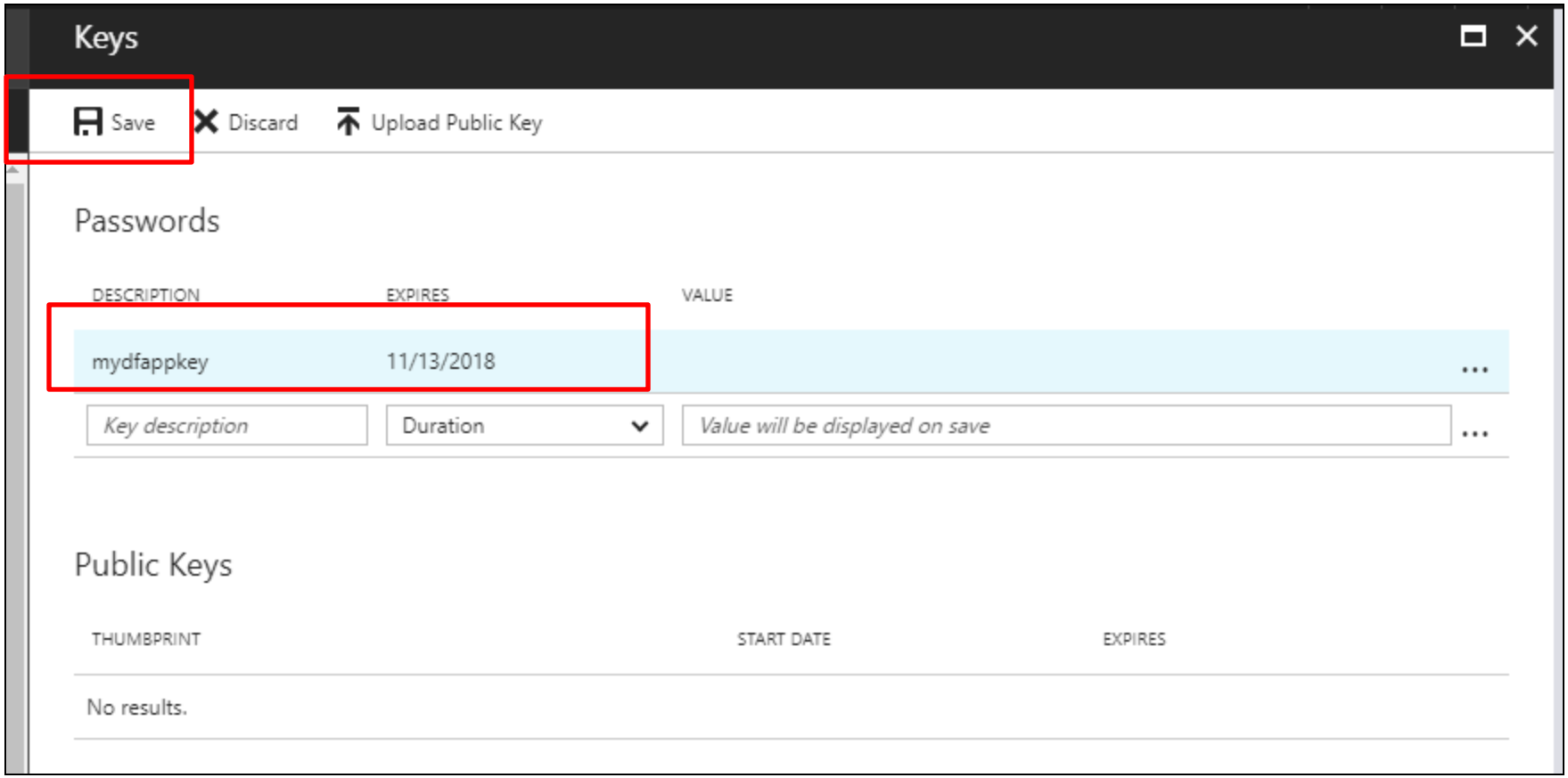
Create an Azure Active Directory application

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



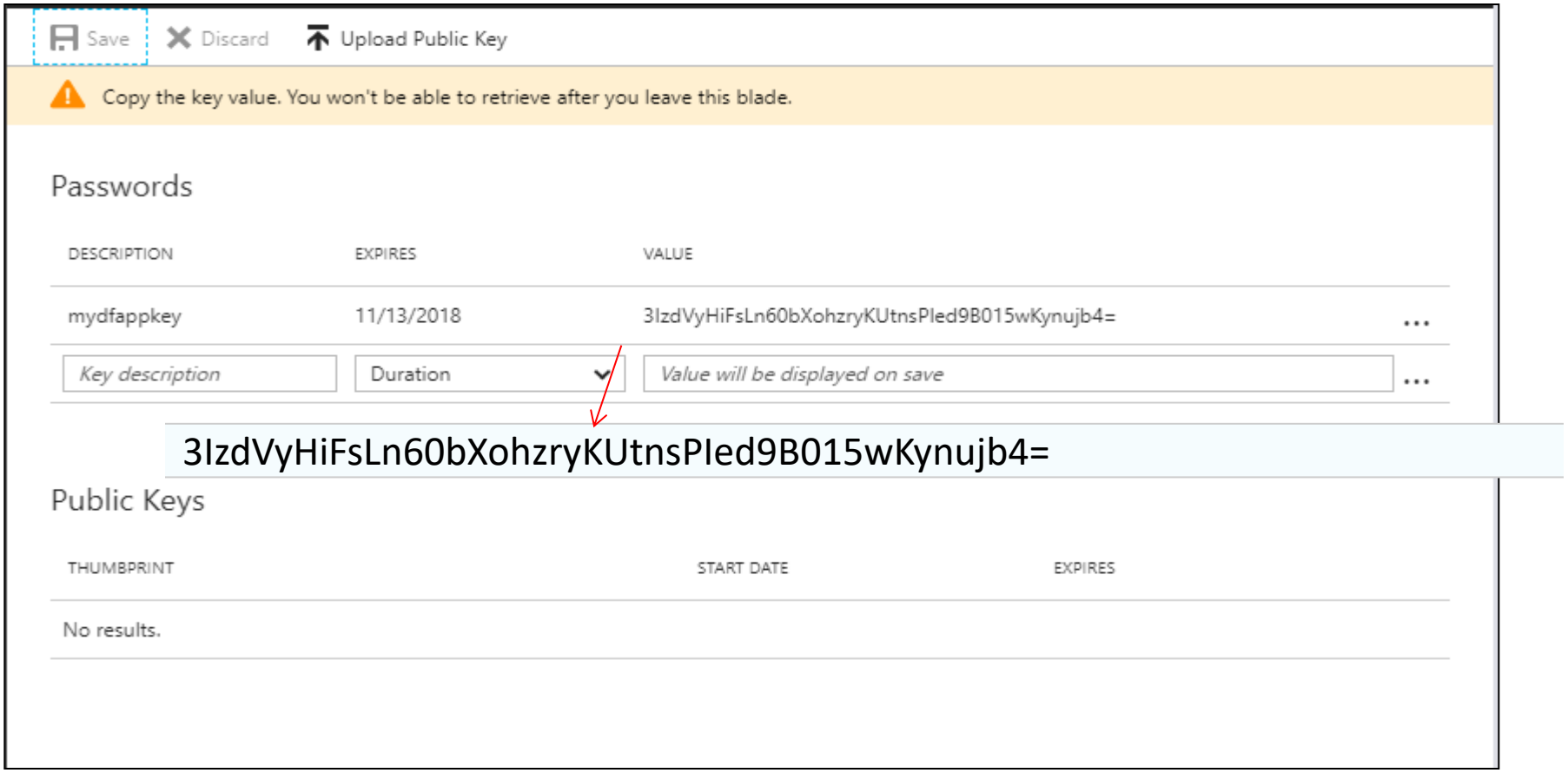
Create Authentication (Secret Key)

- Set up your Secret Key



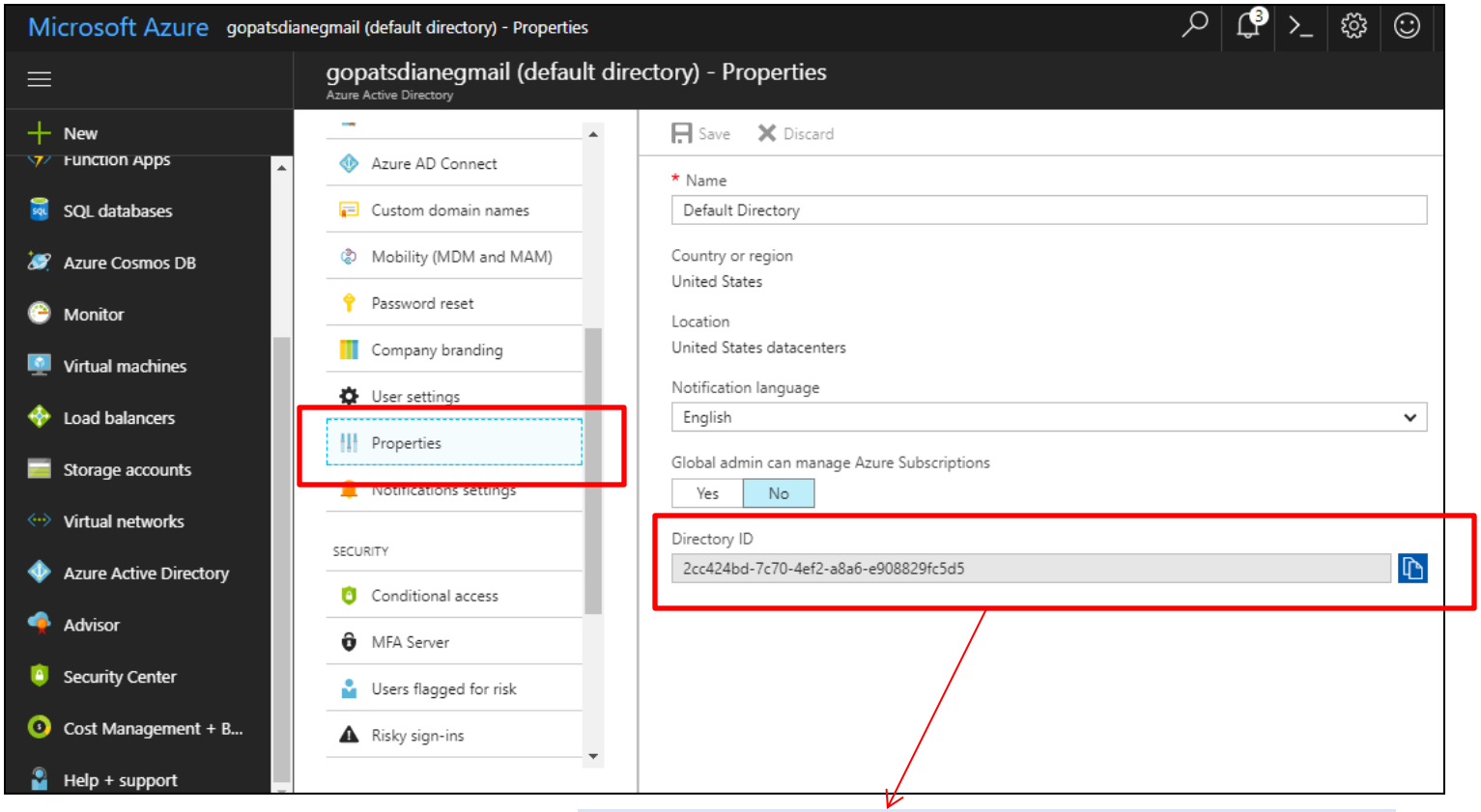
Copy your Secret Key Value

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



Get your Tenant ID

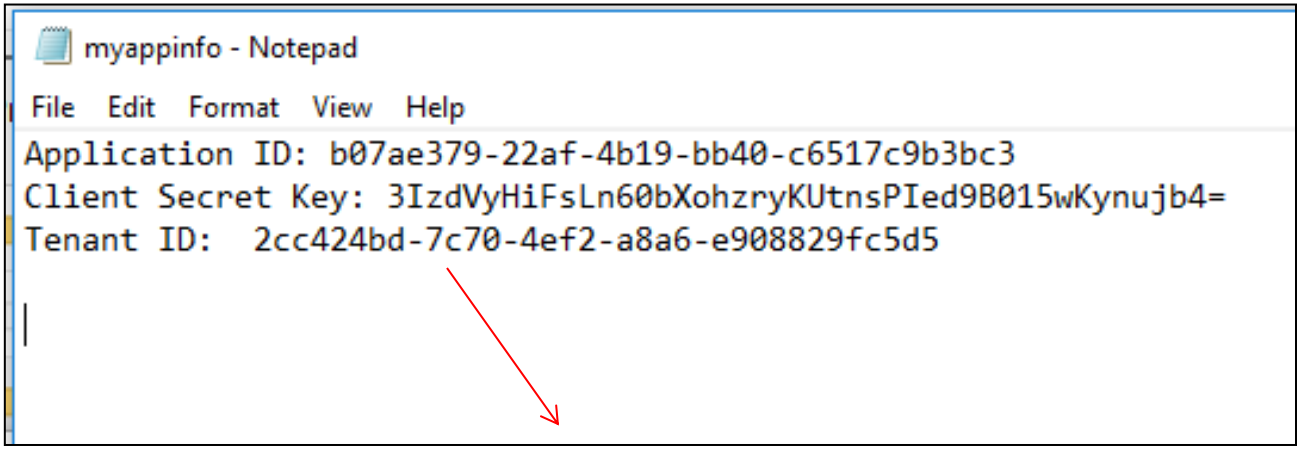
- The Directory ID = your Tenant ID.
- Copy it to Notepad



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

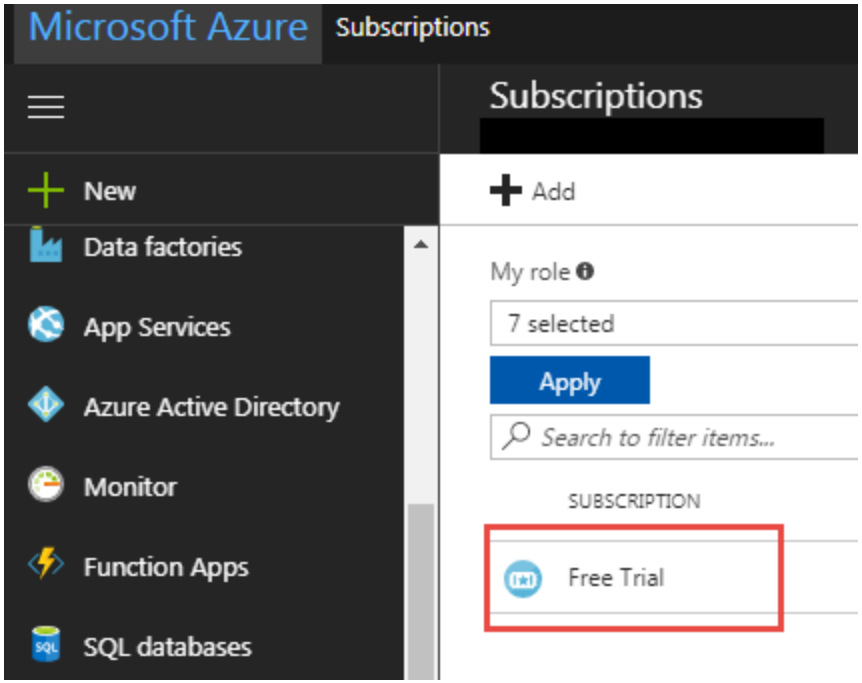
Save in Notepad

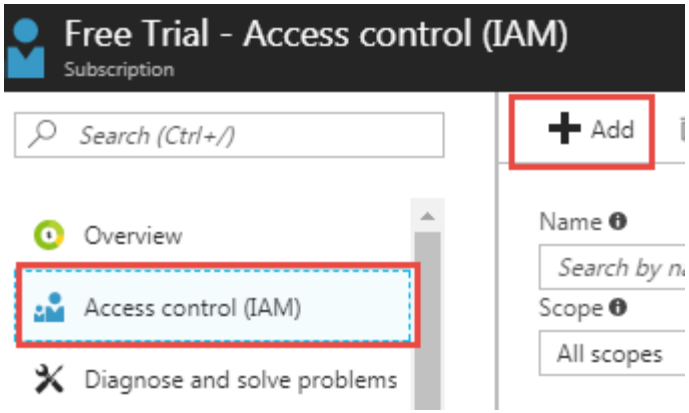
- Application ID: b07ae379-22af-4b19-bb40-c6517c9b3bc3
- Client Secret Key: 3IzdVyHiFsLn60bXohzryKUtnsPIed9B015wKynujb4=
- Tenant ID: 2cc424bd-7c70-4ef2-a8a6-e908829fc5d5
- We will need these values later in your .NET code.



Assign a Role

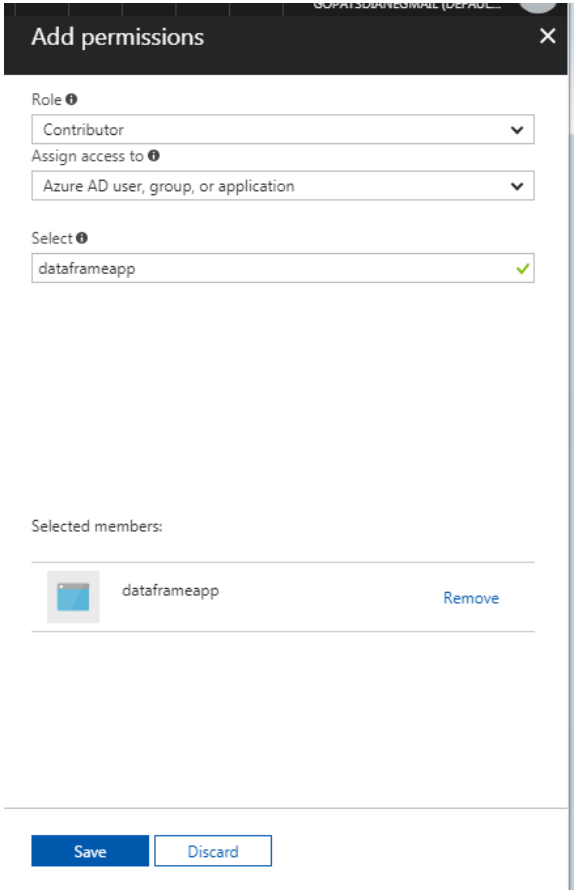
- Under New -> More Services -> Subscriptions
- Select your Subscription -> Access Control (IAM) -> Add





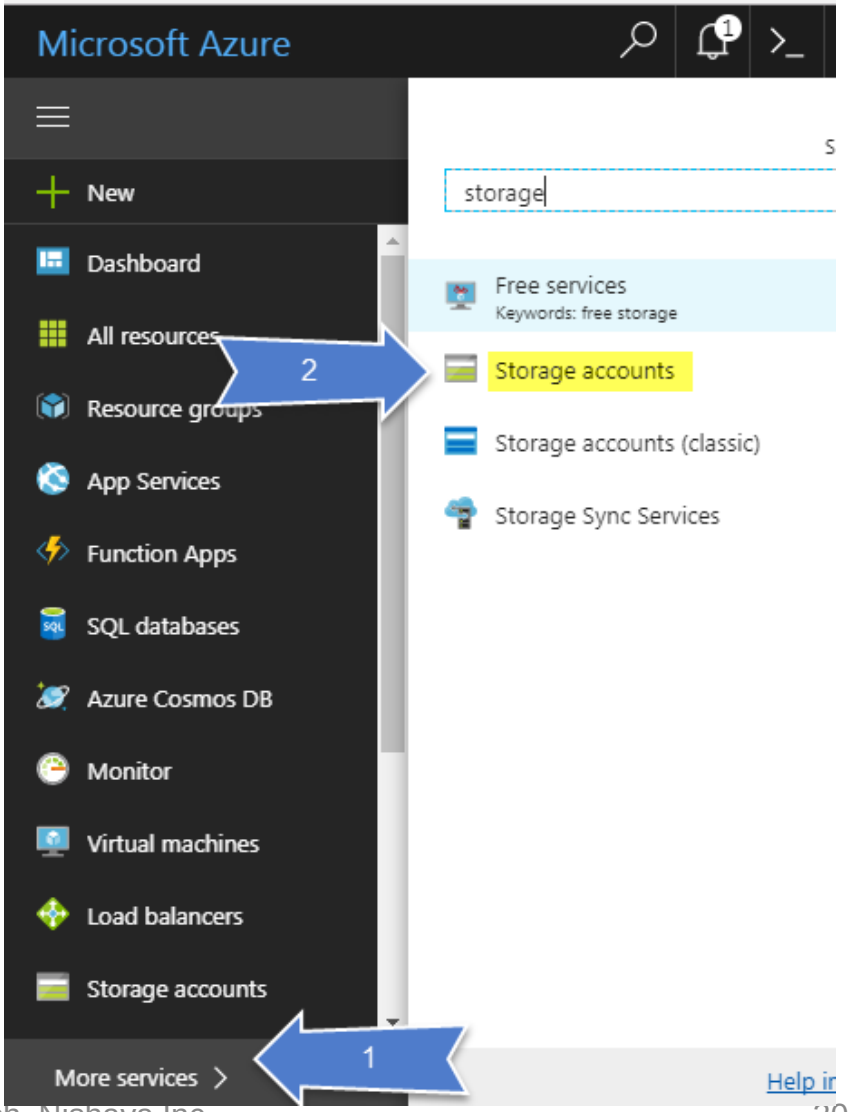
Assign a Role

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



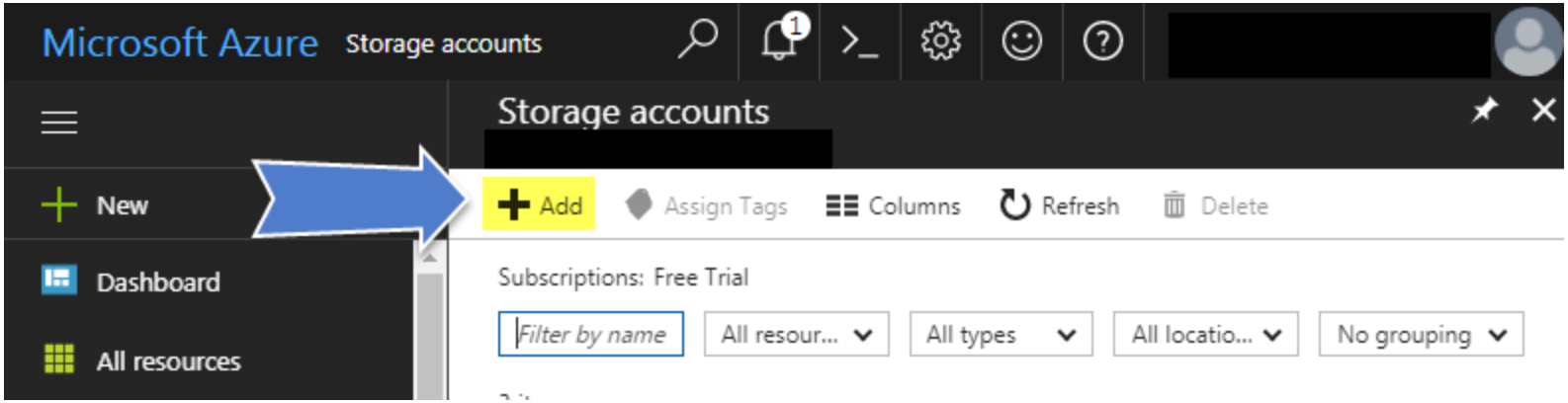
Create an Azure Storage Account

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



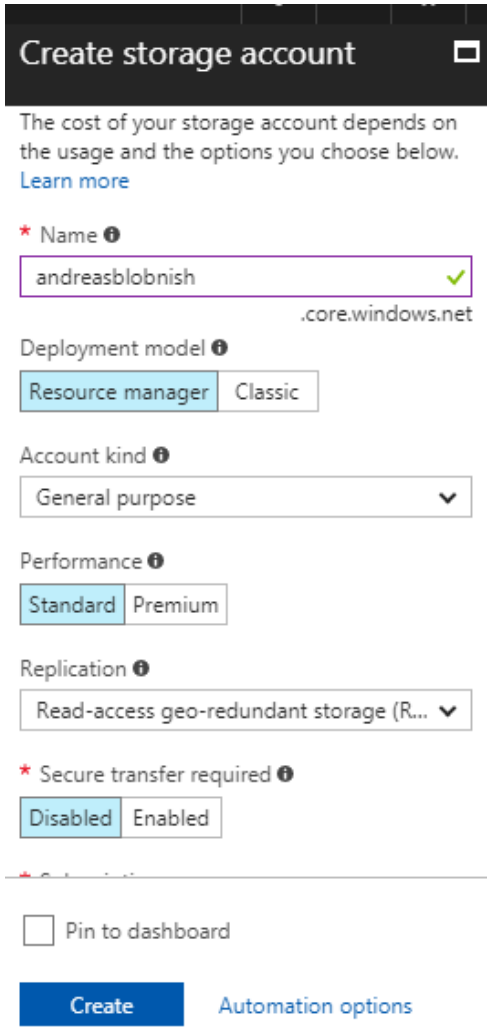
Create an Azure Storage Account

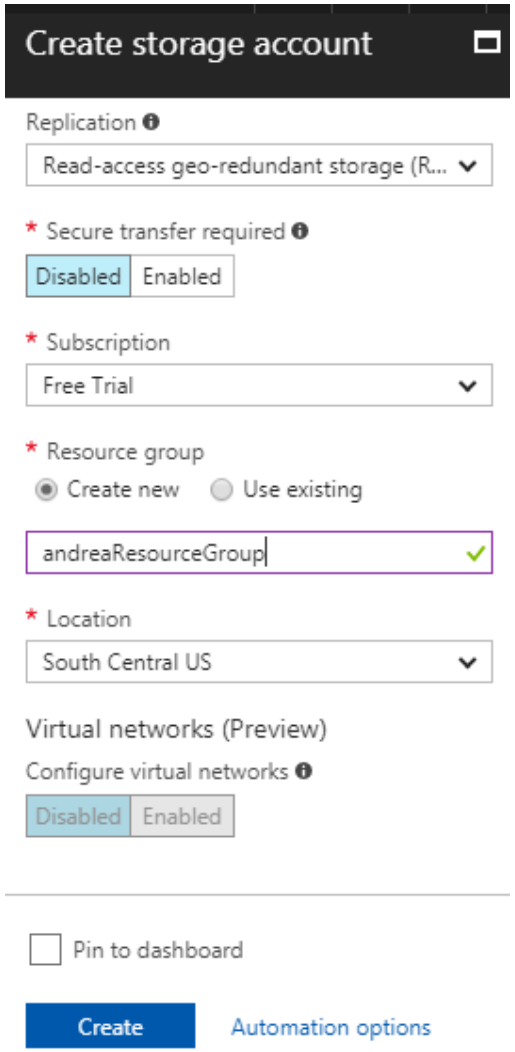
- Select Add to create a Storage Account



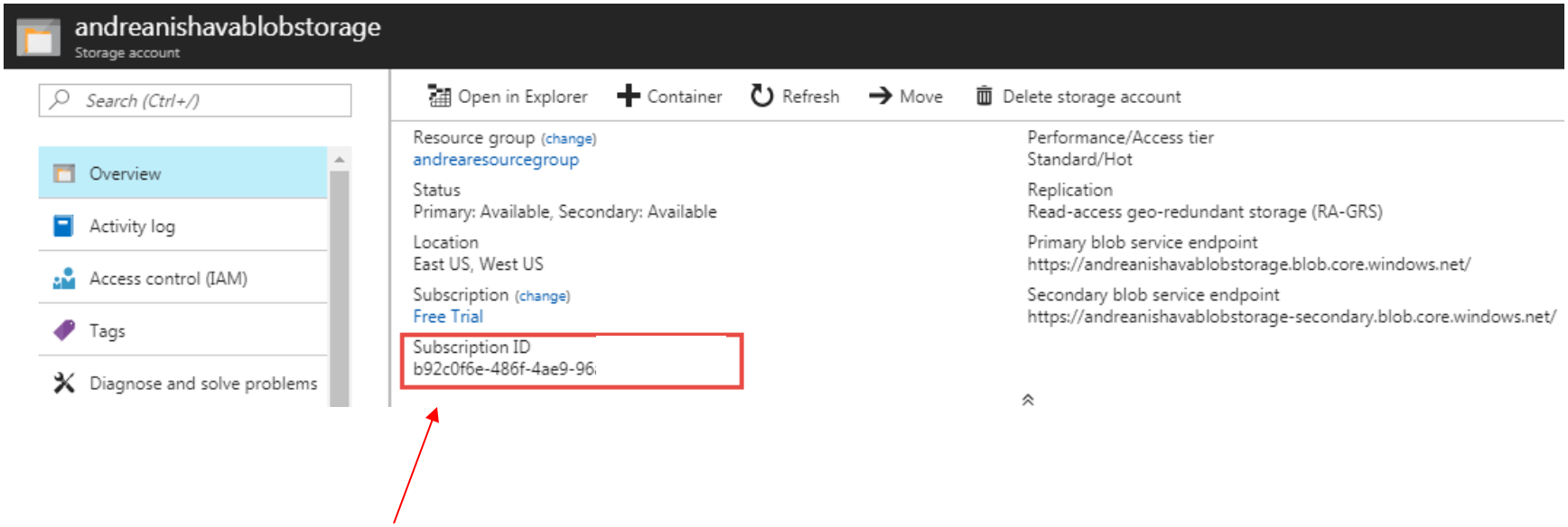
Create an Azure Storage Account

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





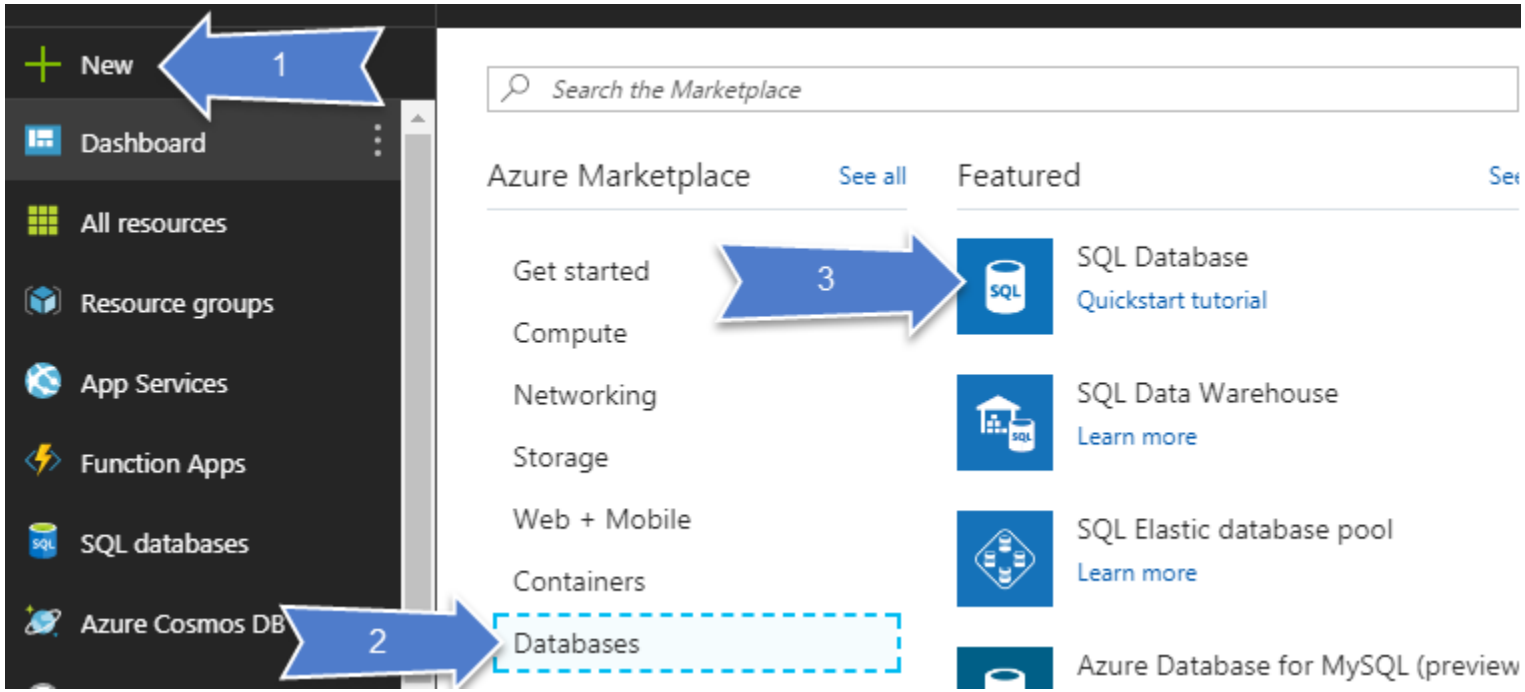
Save your Storage Account Subscription ID



Save your Subscription ID as you will need it

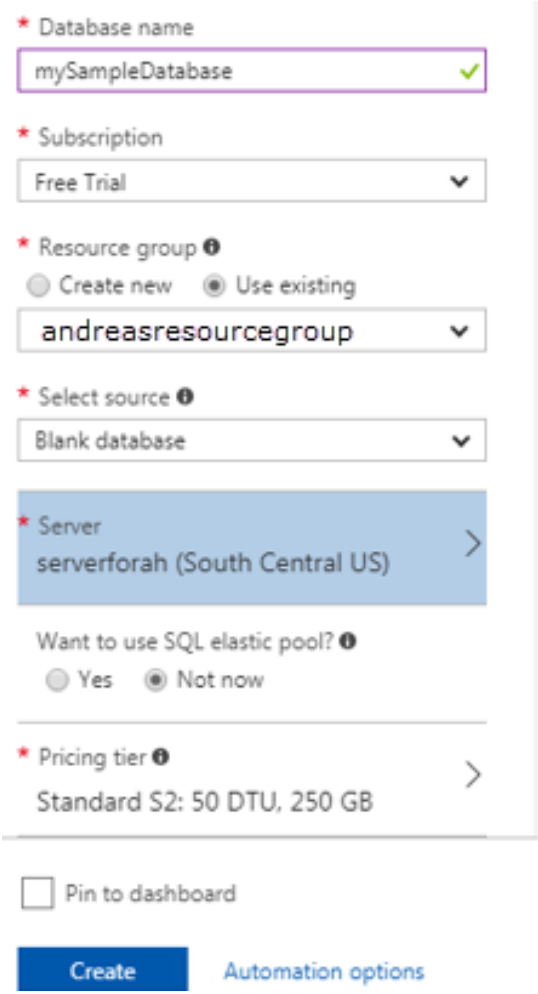
Create an Azure SQL Database

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



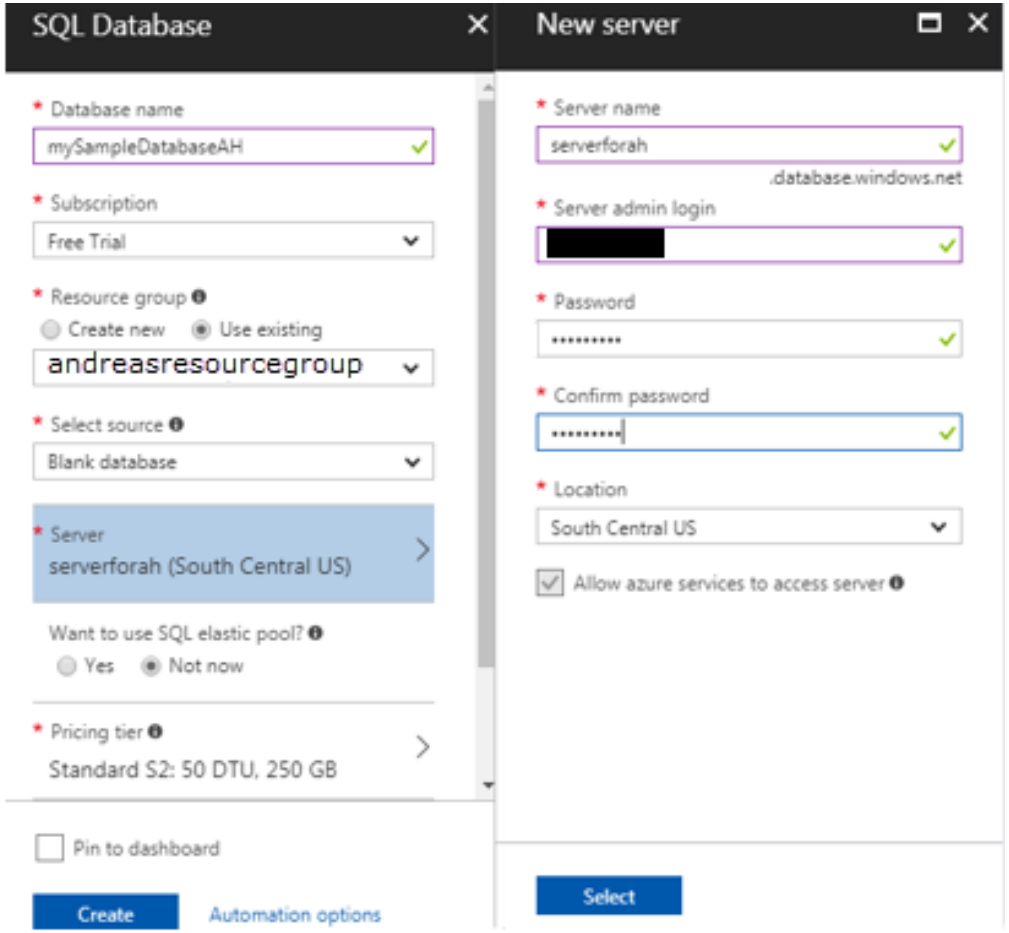
Create an Azure SQL Database

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



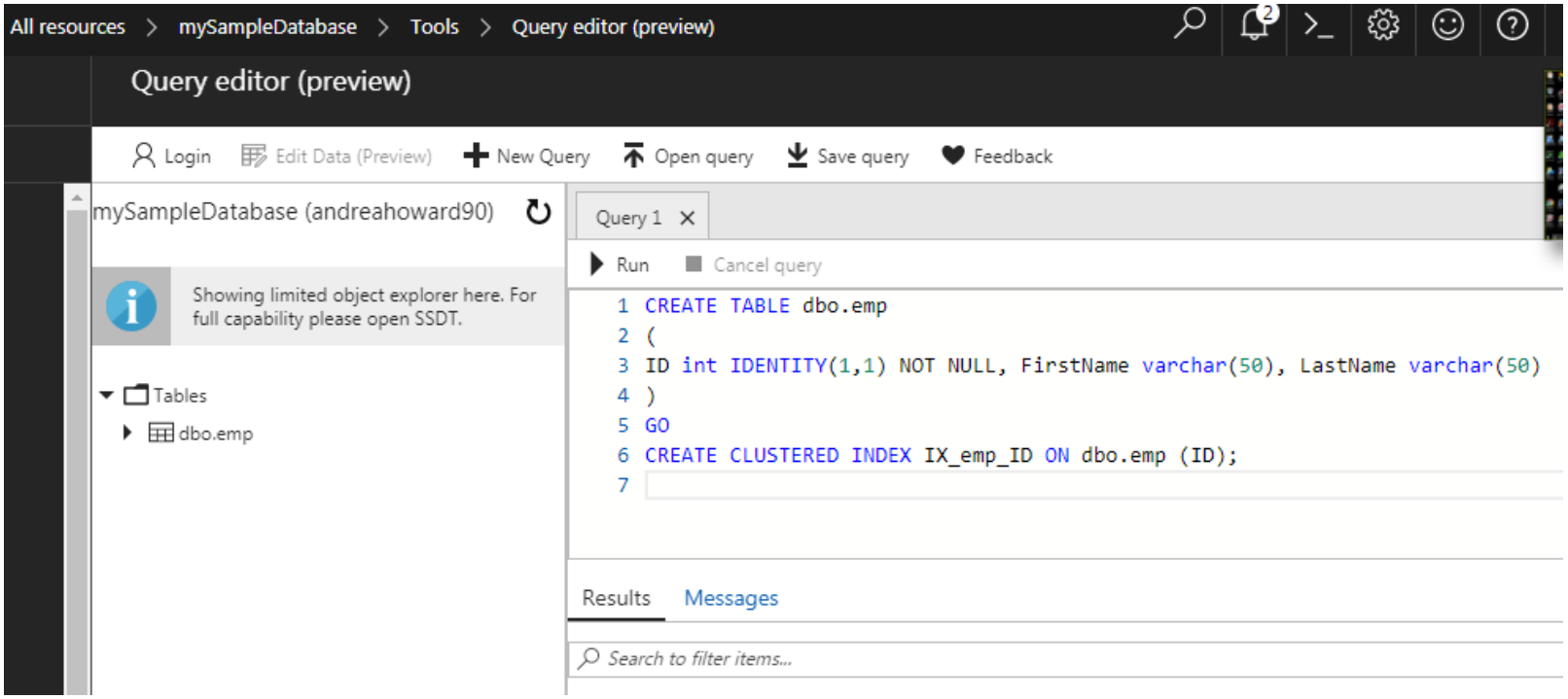
Create an Azure SQL Database

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



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



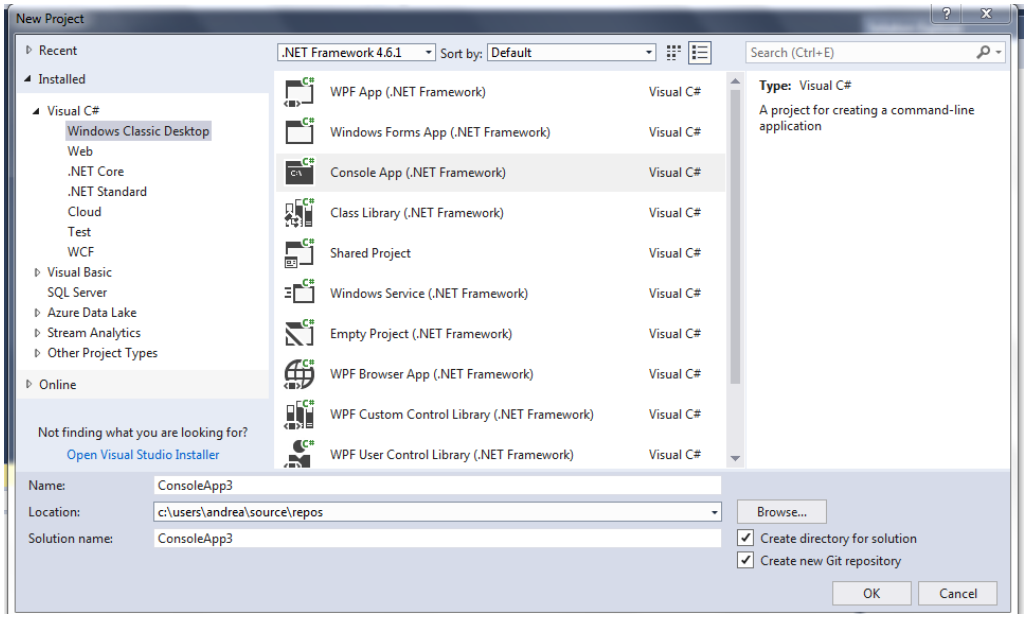
Create a SQL table in your Database

- 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);
```

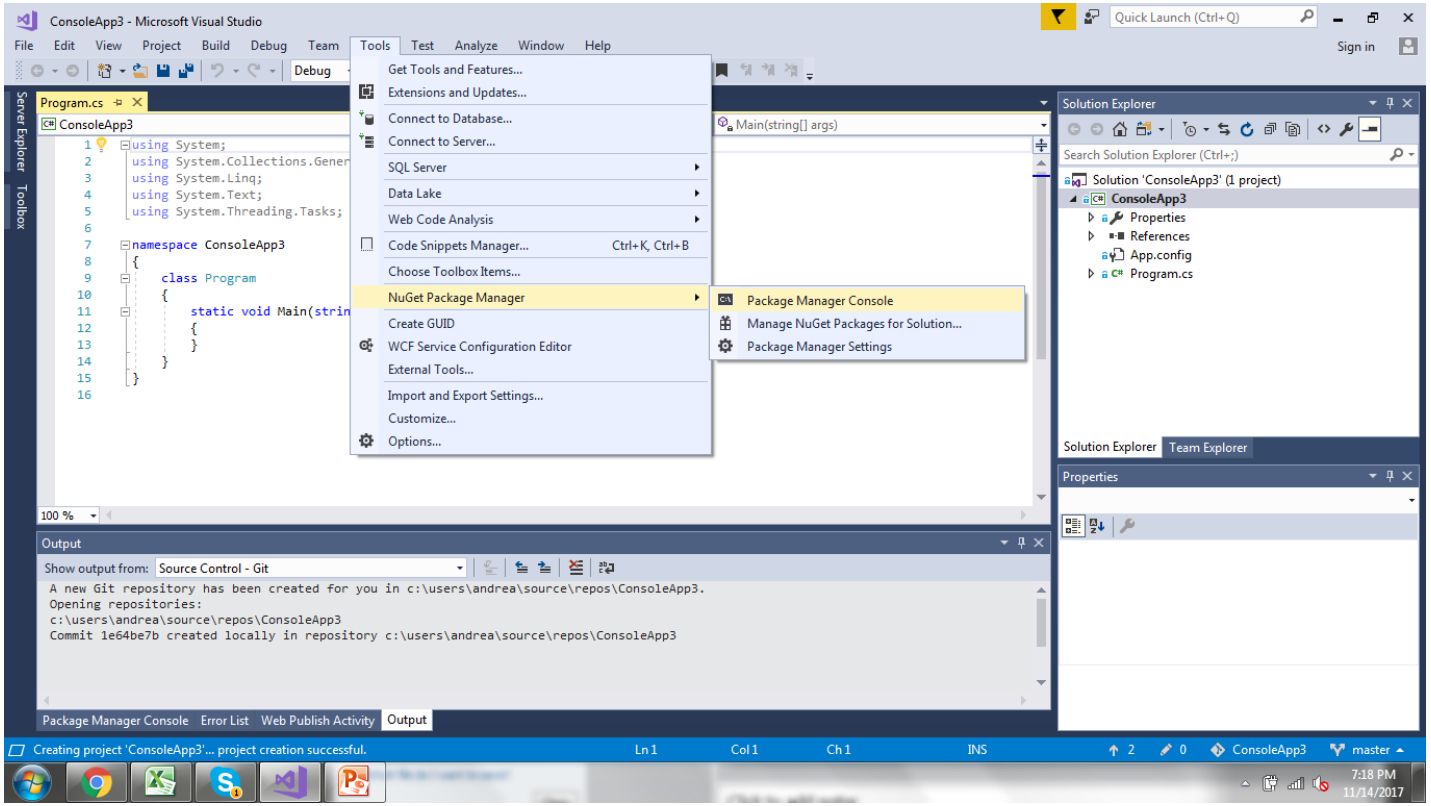
Create a Console App

- 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



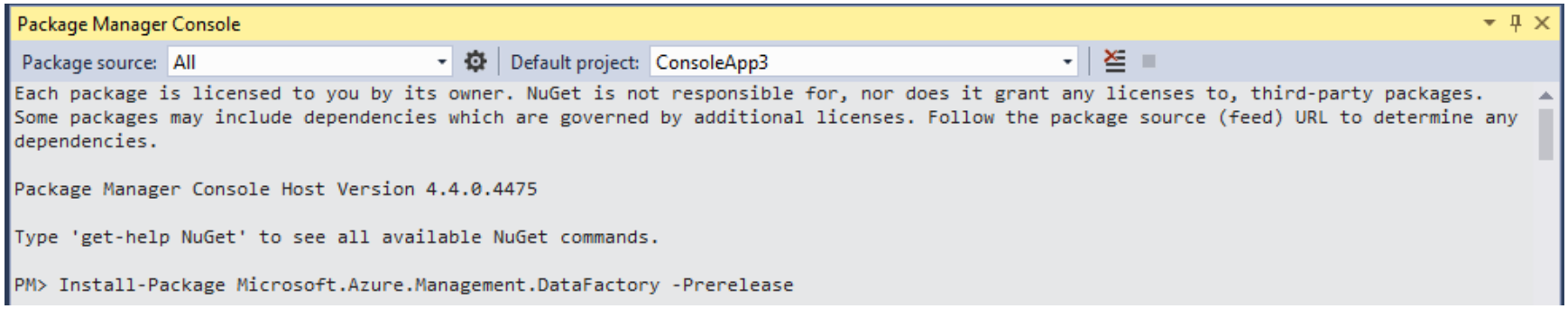
Install NuGet packages

- 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

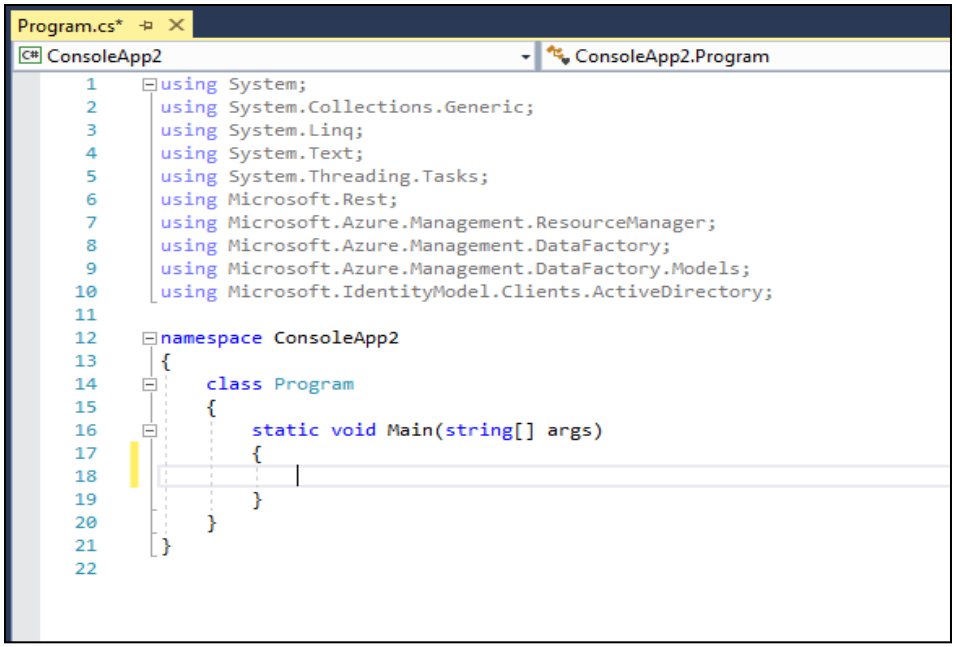
Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory



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;
```



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+dXRFoTxRiGYmkJbOFsLZVV2Xl9kSnho=";
            string subscriptionId = "b92c0f6e-486f-4ae9-96af-              ";
            string resourceGroup = "andreaResourceGroup";

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



        }
    }
}
```

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+dXRFoTxRjGYmkJbOFsLZVV2Xl9kSnho=";
    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/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/zAH1KkJiZSWKzR7GM7RRjIYaQ==";
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 + "...");

LinkedServiceResource storageLinkedService = new LinkedServiceResource(
    new AzureStorageLinkedService
    {
        ConnectionString = new SecureString("DefaultEndpointsProtocol=https;AccountName=" + storageAccount + ";A
    }
);
client.LinkedServices.CreateOrUpdate(resourceGroup, dataFactoryName, storageLinkedServiceName, storageLinkedServ
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(sqlDbLinkedServ
ice, 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, storageLinkedServ
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

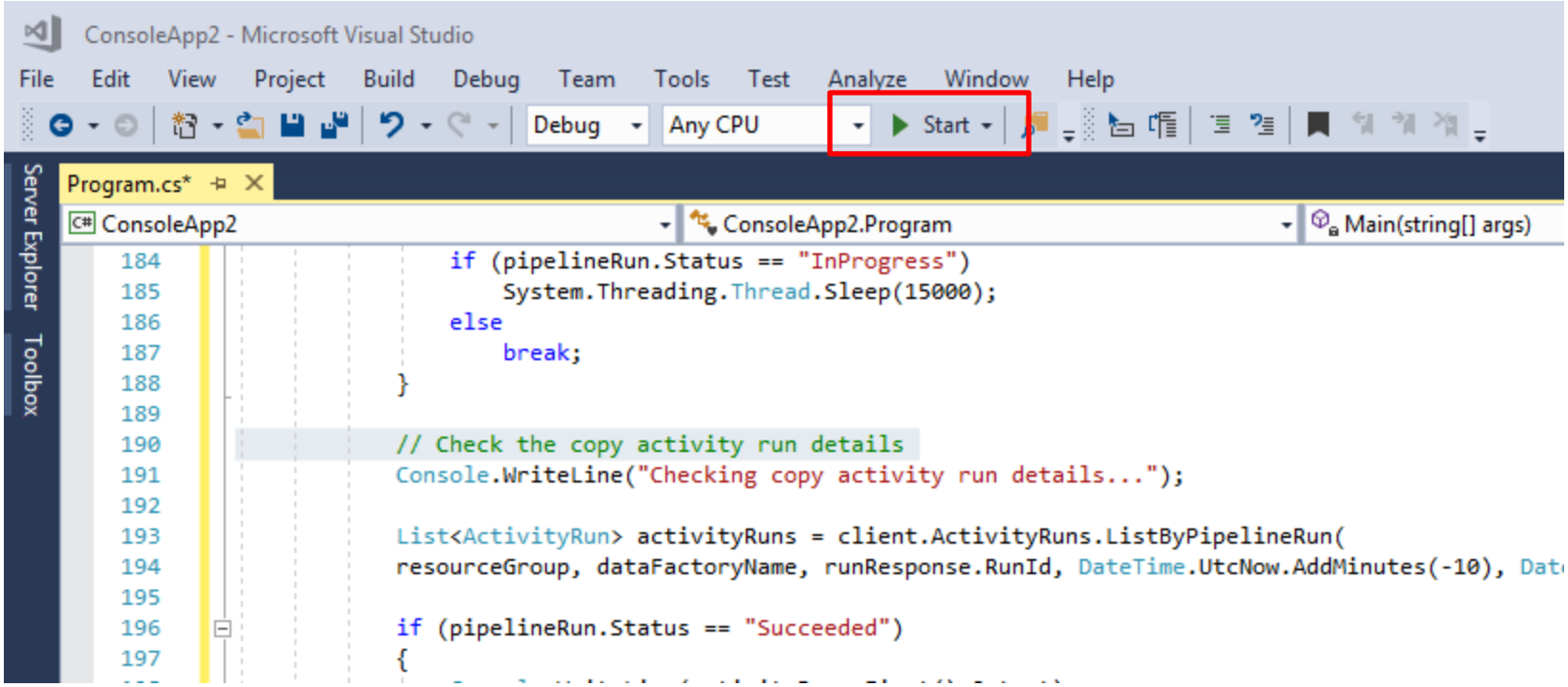
```
// Check the copy activity run details
Console.WriteLine("Checking copy activity run details...");

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(1

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

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

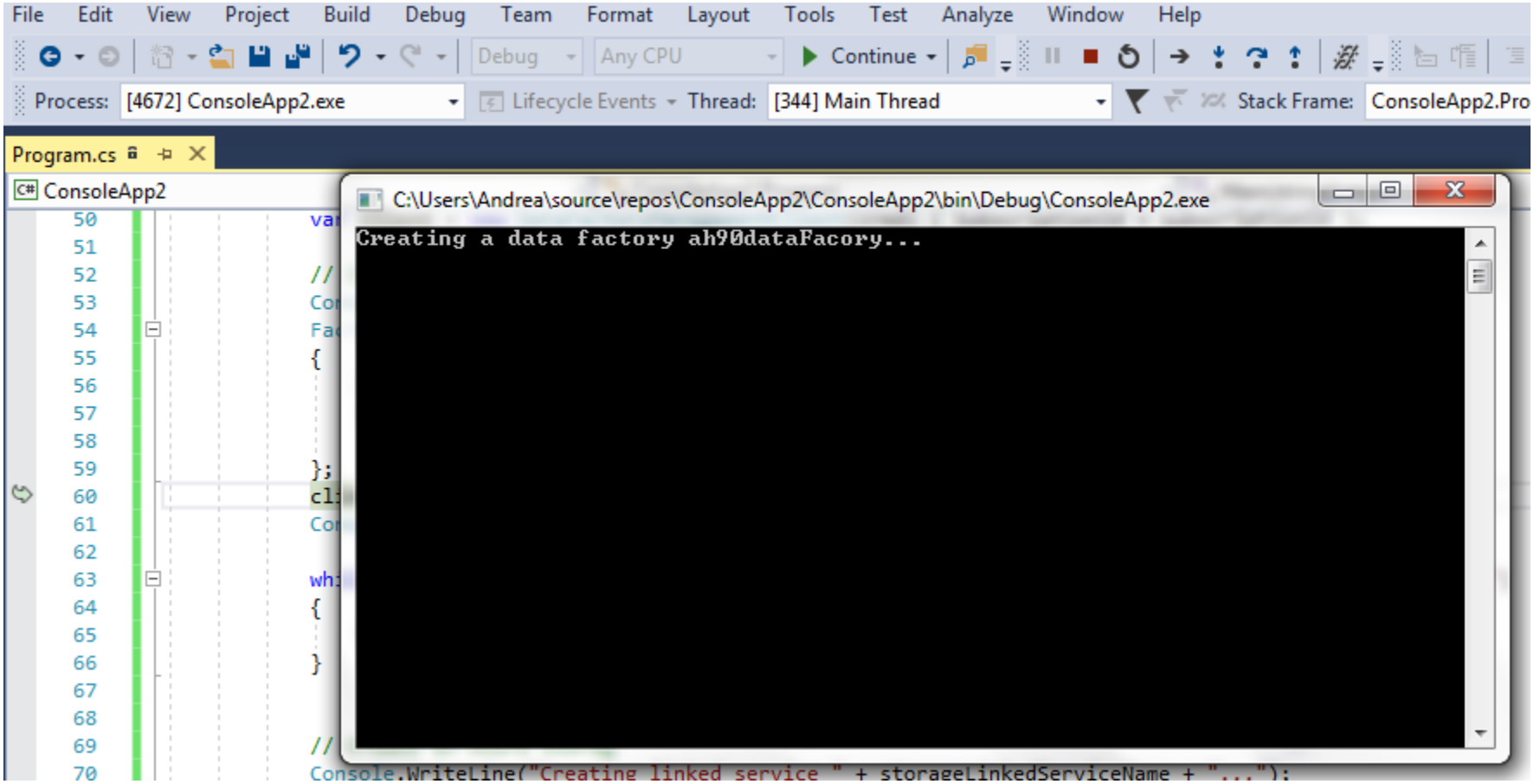
Run the Code

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



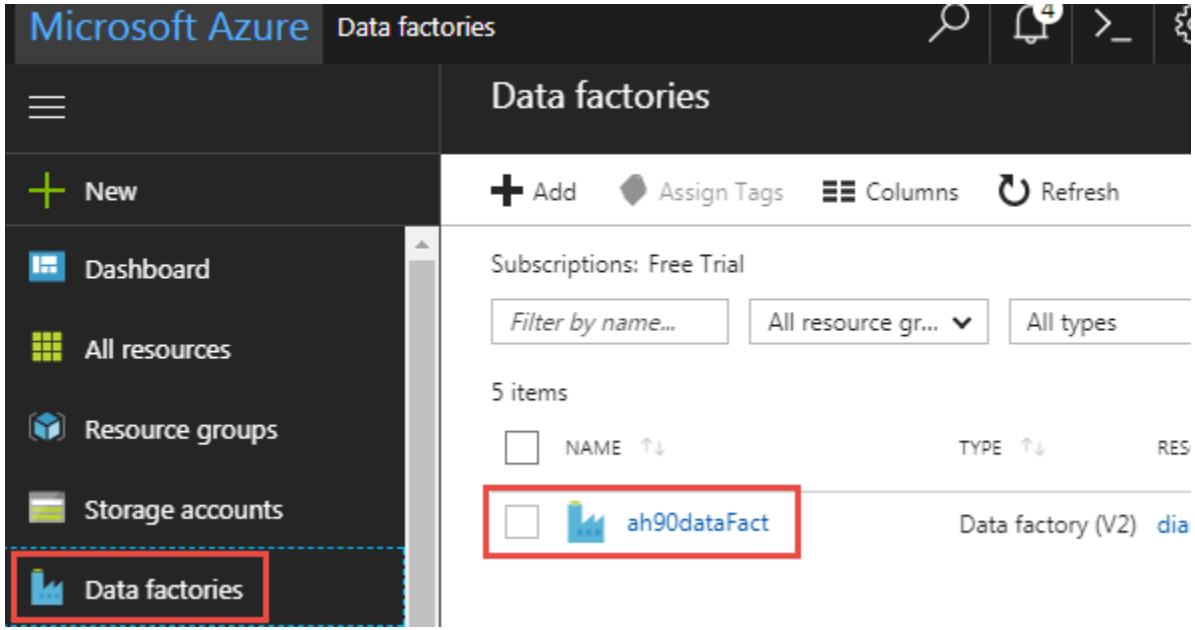
Run the Code

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



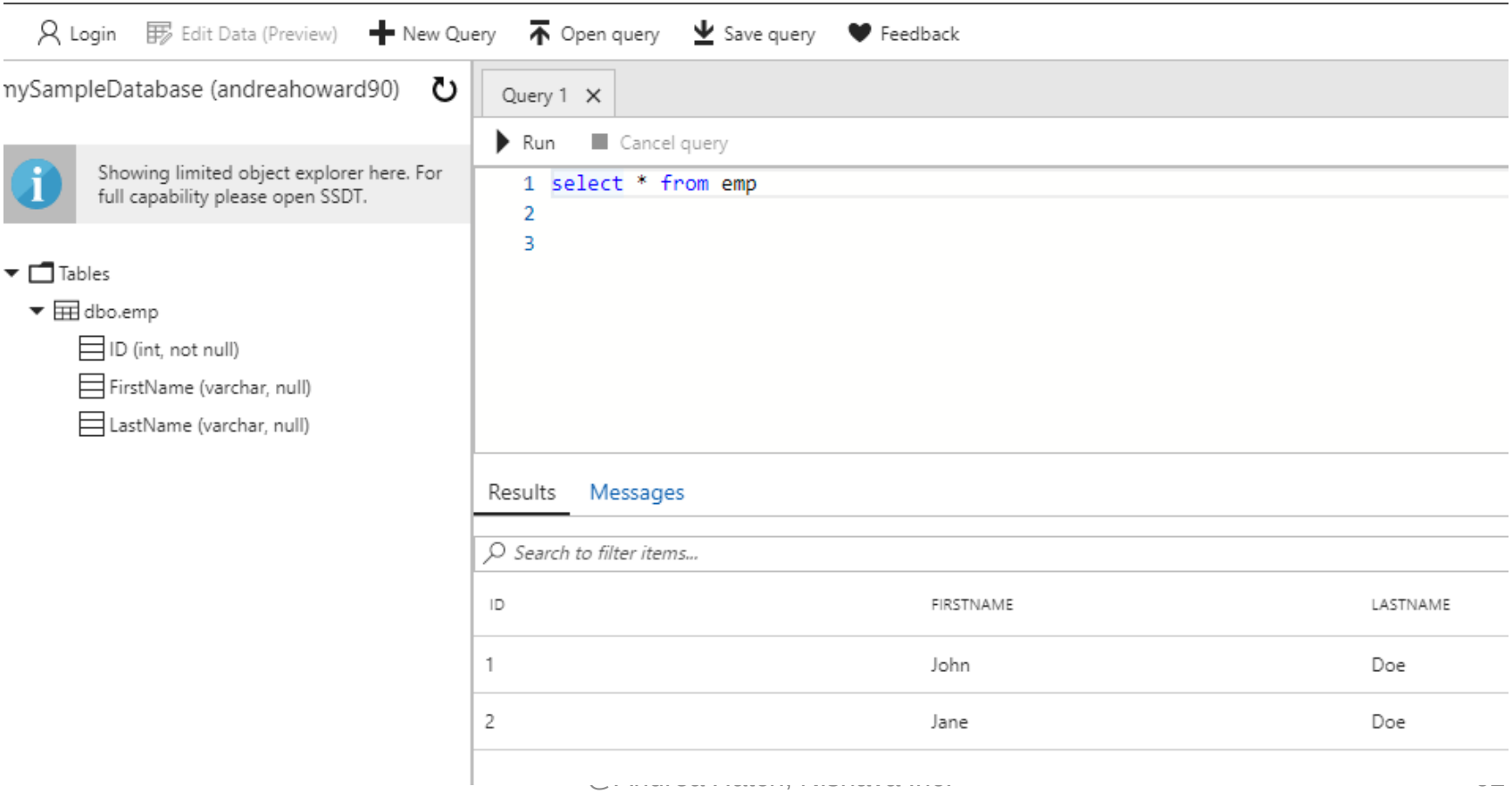
View the Run in Azure

- 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



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