# Components

Each Azure Data Factory has three fundamental components, source/destination dataset, an activity inside of a pipeline, and one or more linked services. Usually, activities require a compute resource. This could be an HDInsight cluster, or an Azure SQL Database. In this chapter, we will go over each of these components and experiment with some of the basic functionality of ADF.

## Linked Services

Every ETL application, including SQL Server Integration Services, starts with defining the data sources. ADF is no exception. In ADF, connection managers are called “Linked Services”. There are five different types of Linked Services available in the current version of ADF.

* Azure storage: This linked service type is used to connect to Azure Blob storage resources.
* Azure SQL Database: This linked service is designed to connect to Azure SQL database. This is different from on premise SQL Server resources.
* On-premises SQL Server database: This connection manager is utilized along with Data Management Gateway to access, SQL Server(s) that are residing on premise.
* On-premises Oracle database: As the name implies, this connection manager is designed to connect to Oracle server(s).
* On-premises file system: This connection manager is for connecting to file shares that are residing on premise systems. Again Data Management Gateway is utilized to access on premise resources.

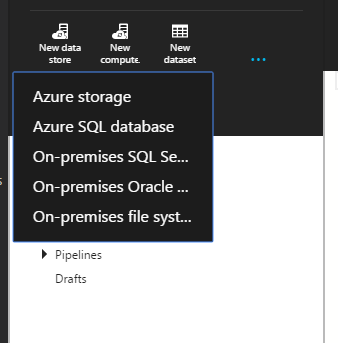


Fig 2-1

Figure ‑

In this first demonstration, let’s create an Azure storage Linked Service. We will be utilizing the web interface. In the upcoming chapters, we will explore the other ways to create Azure Data Factory pipelines. First open your favorite browser.

* *Go to https://portal.azure.com, this URL is for previewing Azure components.*
* *Click on the plus sign at the bottom left corner.*
* *Select “Data Analytics” from the “Create” menu and click on “Data Factory” (fig 2-2).*

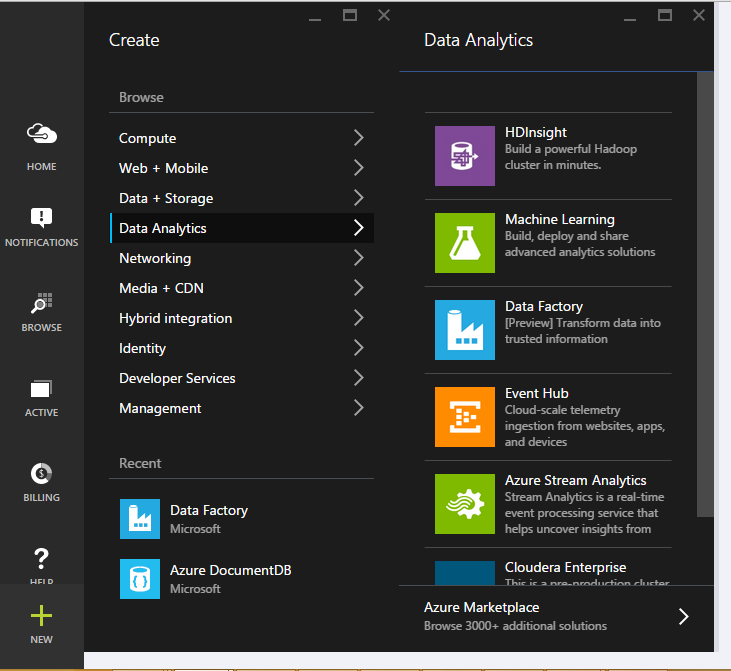


Figure 2‑1‑2

* ***Name****: Chpt2-DataFactory (This is the name of the Data Factory and it should be unique among ADF users.)* 
  + ***Resource Group****: Default-Storage-WestUS (In this step, you could utilize your own resource groups as well by following the steps below.)*
  + ***Create a new resource group***
  + ***Name:*** *MyEastCoastResourceGroup*
* ***Subscription Name:*** *Pay-As-You-Go (This parameter is based on the resource group selected. As the resource group is an entity that combines storage account and location.)*
* ***Region Name:*** *West US (This parameter, like the subscription name, could be locked as well. They both depend on the Resource Group selection.)*

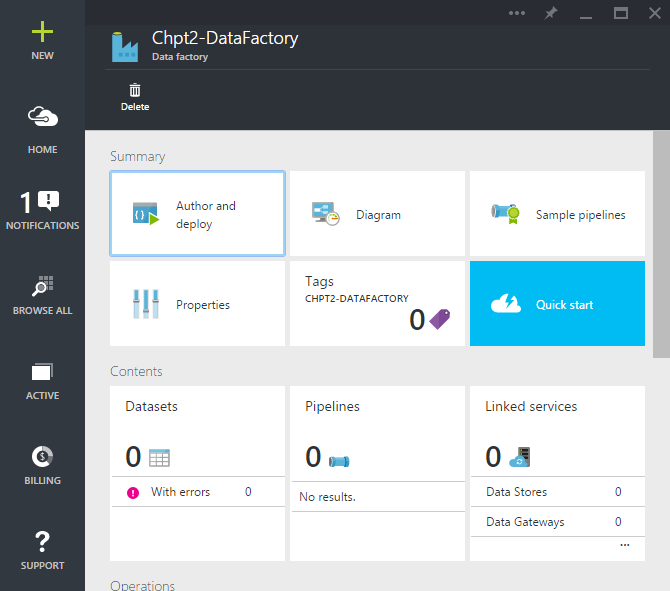


Figure 2‑3

* *fter a few minutes, our new Chpt2-DataFactory should be ready. (Fig 2-3)*
* *Click on “****Linked Services****”. A new blade should open on the right side.*
* *Click on “****New Data Store****” icon at the top.*

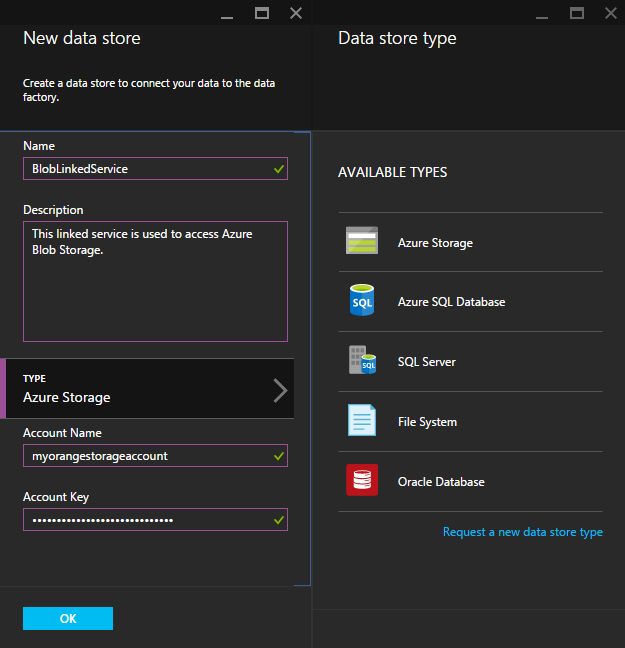
**

Figure 2‑4

* ***Name:*** *Type “****LinkedServiceBlob****” as the name of the Linked Service. It is always a good idea to name common objects meaningfully as those objects could be referred by other developers.*
* ***Type:*** *Click on “****Azure Storage****” as the type of the linked service.*
* ***Account Name:*** *Here set your Azure Storage Account Name. In this case, I set it to myorangestorageaccount.*
* ***Account Key:*** *This is the storage account key to access to your storage account. The user interface of the Azure Data Factory masks this sensitive information for security reasons.*

*Once, all the fields are completed, click on the “OK” button. This action will create a new linked service for you, if the settings are correct. You should be able to see your new Linked Service at the data factory management user interface.*

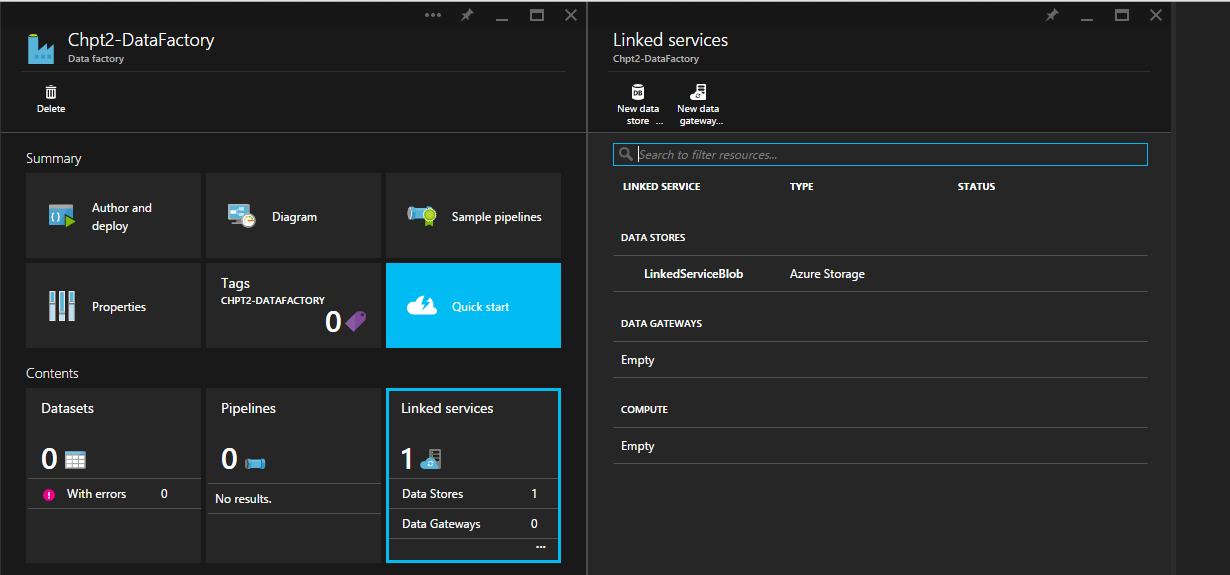


Figure 2-5

## Datasets

In Azure Data Factory, datasets are defined as the logical representation of data. Like any other component, it is defined in JSON format. Having said that, there are multiple ways to create them, hence no need to worry about creating JSON files manually. In this section, we will use the main user interface.

Each dataset has four main components. Depending on the linked service, these components may have additional fields.

1. **Structure**: The name of the columns and data types associated with them are defined here.
2. **Location**: Starting with the name of the Linked Service, location of the data, if applicable partition structure, and format of the data are set.
3. **Availability**: The frequency of data population and the interval associated with this frequency are defined. For instance, if we set the frequency to “Day” and interval to “1”, would mean our dataset is going to have one slicer per day. We will go over this slicer concept in later chapters.
4. **Policy:** Add definition of a policy.

For this example, we have created a fictitious dataset. It shows orange production between 1993 and 2013. The dataset has two columns, Year and Volume. You may access to the demo dataset from the link below. (GITHUB LOCATION)

Throughout the book, we will be using an open source tool called Azure Storage Explorer to access our Azure Storage. You may download the tool from the link following link. https://azurestorageexplorer.codeplex.com/

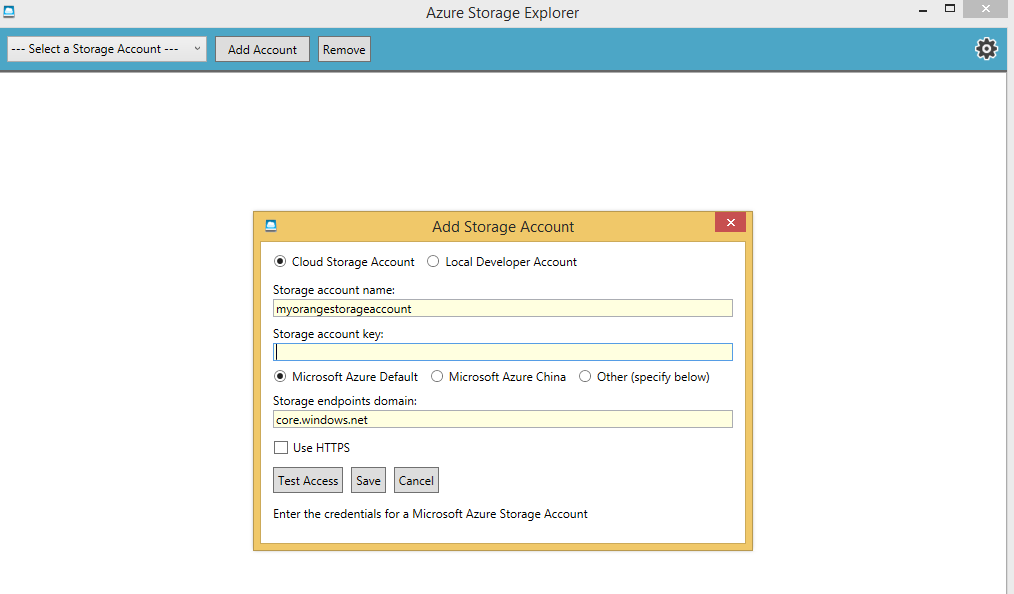
When you run the Azure Data Explorer, click on the “**Add Account**” icon at the top.

Figure 2‑6

Type your storage account name and your account key. Once you hit save, this is how it should look like.

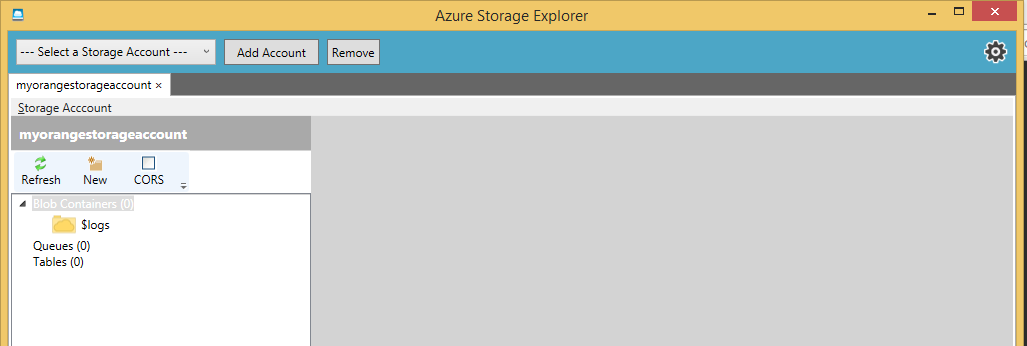


Figure 2-7

* Select the “**Blob Containers**” and click “**New**” icon at the top. This action will create a new blob container.
* ***Blob Container Name****: Type stage. Be aware, Azure does not like capital letters. Leave the access level as is and hit “****Create****”.*
* Select the “**stage**” container and click on upload icon on the ribbon at the top.
* Find “**DimCustomer.csv**” and hit “**Open**”.

Before going back to the Azure Portal, let’s create our raw container as well.

* Select the “**Blob Containers**” and click “**New**” icon at the top. This action will create a new blob container.
* ***Blob Container Name****: Type “****raw****”. Be aware, Azure does not like capital letters. Leave the access level as is and hit “****Create****”.*

*Now, let’s go back to portal. Our objective is to create our first dataset.*

* *Click on “Author and deploy” icon at the top left.*
* *Click on “****New dataset****” and from the drop down menu choose “****Azure Blob Storage****”. This action will open up a JSON base document template.*

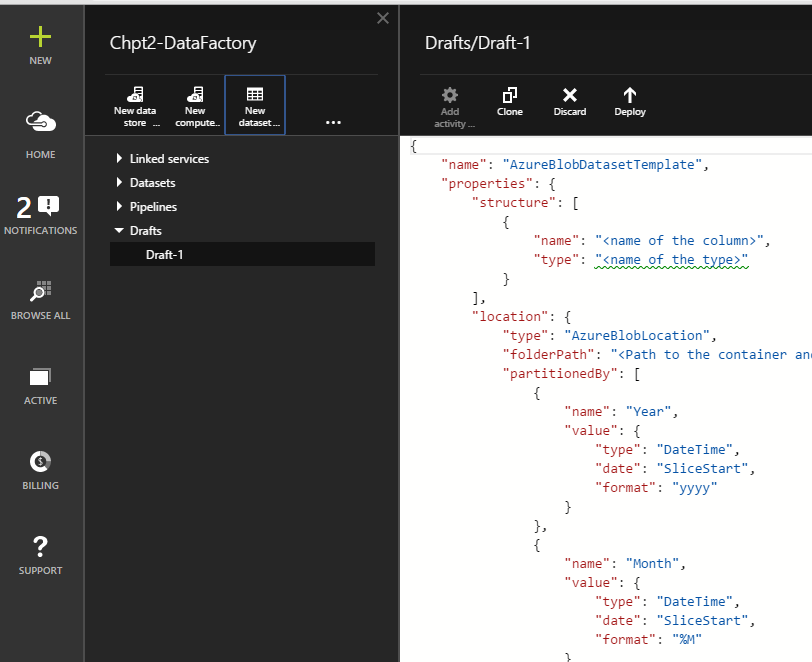


Figure 2‑8

* ***Name:*** *Change “AzureBlobDatasetTemplate” to “****DimCustomer****” as the name of the dataset. This is how the object will be referred to from now on.*
* *Under the structure element, dataset definition is defined. Each column has two attributes, a name and a type. Just delete the template existing section and paste the following code.*

*{"name":"IDCustomer","type":"Int"},*

*{"name":"CustomerID","type":"Int"},*

*{"name":"Title","type":"String"},*

*{"name":"FirstName","type":"String"},*

*{"name":"MiddleName","type":"String"},*

*{"name":"LastName","type":"String"},*

*{"name":"Suffix","type":"String"},*

*{"name":"CompanyName","type":"String"},*

*{"name":"ADWSalesPerson","type":"String"},*

*{"name":"EmailAddress","type":"String"},*

*{"name":"Phone","type":"String"},*

*{"name":"CustomerEffDate","type":"Date"},*

*{"name":"CustomerEndDate","type":"Date"},*

*{"name":"CustomerCurrent","type":"String"},*

*{"name":"LoadID","type":"Int"}*

After the modification the structure element of our template should look like figure 2-9.



Figure 2‑9

ADF supports couple more data types like, decimal, guid, boolean, and enumurator. For this example, we are just using strings, integers, and date data types.

* **folderPath**: Type “***stage***/". There are two parts in a folder path. First the container is defined, in our case the container name is “stage”. Then the location of the blob is added. Since, we have DimCustomer.csv at the root of the container, we just add “/” after the container name.
* Add a **comma** after **folderPath** and add the following line

**"fileName": "DimCustomer.csv",**

* For this dataset, our data is not partitioned. Hence, let’s remove the “partitionedBy” section all together.
* There are two types of data formats supported by ADF, TextFormat and AvroFormat. Change the **format** element to “***TextFormat***”.
* Since we have a csv file, our column delimiter is comma. Update the **columnDelimiter** attribute value to “*,*”.
* Set the **rowDelimiter** to “*\n*”, new line notation.
* ADF supports escape characters for columns. For this exercise our data has already been escaped. Hence, delete “EscapeChar” element.
* Different line of business applications may represent null values differently. ADF supports conversion between these representations of null values, so that the systems can work together. For the time being delete this element as well. Also remove the last comma at the **rowDelimiter**.
* **linkedserviceName**: Type “LinkedServiceBlob”. This is the linked service that is going to be used to access to our blob.
* **frequency**: Type “***Hour***”. This element sets the frequency of data update for that dataset.
* **interval**: Type “***1***”. This element sets the number of times the dataset if going to be updated. In our case, we will have once in an hour updates.
* After the **interval** attribute put a **comma** and add the following line

**"waitonexternal": {}**

Since, our table is external, we need to add this line, so that ADF does not wait for files to show up.

Click on the “Deploy” icon at the top.

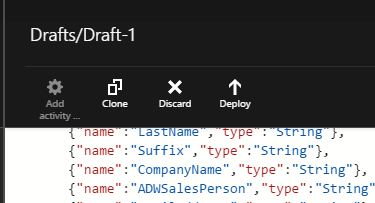


Figure 2‑9

We should have our first dataset, DimCustomer, defined. Now, we are going to create our destination dataset. However, this time around, we will use some shortcuts.

Click on DimCustomer from Datasets drop down list.

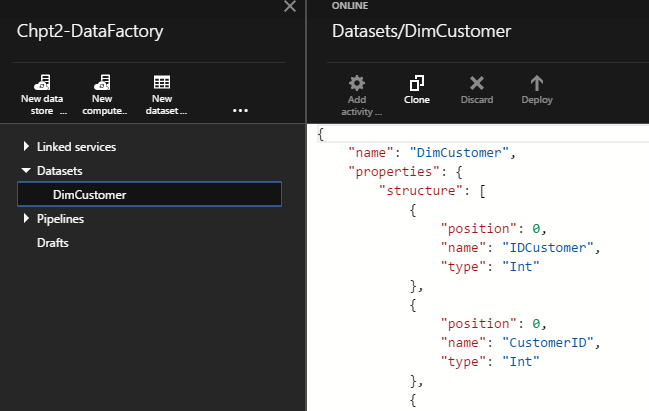


Figure 2‑10

Then hit “Clone” button at the top right bar. This action will create an exact replica of our DimCustomer object for us. Hence, we just need to modify a few attributes to make it work, instead of writing the JSON document from the beginning.

* Change the **Name** of the dataset to “**DimCustomerRaw**”. In ADF, the object names has to be unique.
* Change the **folderPath** attribute value to “**raw/**”.
* Put a **comma** after folderPath and add the following line:

**"fileName": "DimCustomer.csv",**

* At the availability section, delete the **comma** after the **internal** attribute and remove **"waitonexternal": {}** line.
* Hit “**Deploy**” icon at the top. This action should create our new object called “DimCustomerRaw”.

The last stage of our data factory is building the pipeline in between our datasets. This is where we defined our actions (transformations).

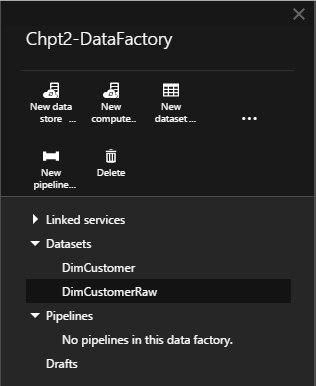


Figure 2‑11

* Click on the three dots “**…**” at the design pane. This is going to expand options and you should see “**New Pipeline**”. Click on it.
* Change the “**name”** of the pipeline to “**PipelineDimCustomer**”.
* Change the “**description**” of the pipeline to **“This pipeline migrates data from stage container to raw container.”** As the name implies, description field is to have basic documentation around the pipeline.
* Skip the “**activities**” for now, and set the “**start**” to yesterday’s date. Today is May 23rd, 2015, so we are going to set it to “6/22/2015”. However, the DateTime fields should be in ISO8601 format. Hence, we need to type it like “**2015-06-22T12:00:00Z**”.
* Set the “**end**” to be today’s date.
* After the “**end**” attribute put a comma and add the following lines.

**"isPaused": false,**

**"hubName": "chpt2-datafactory\_hub"**

Your pipeline should look like the following figure.

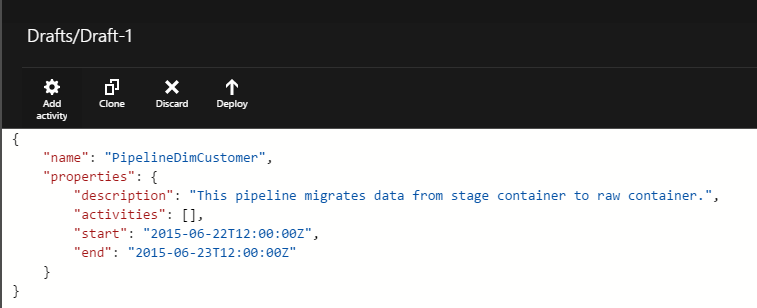


Figure 2‑12

* Pipelines are just containers to hold one or more activities. If a pipeline has more than one activity, it will execute them in sequential order from the top to the bottom. Now, it is time to add an activity into our pipeline. Click on the “**Add activity**” icon at the top left corner of figure 2-12.
* From the drop down list, choose “**Copy Activity**”. This action will add a section to your pipeline.
* Change the **name** of the activity from “CopyActivityTemplate” to “**CopyDimCustomer**”.
* Leave the **type** as “**CopyActivity**”. Soon, we will be working on some other activities.
* Under the **inputs**, change the **name** to “**DimCustomer**”. This field is a list in JSON. Hence, we could potentially add more than once dataset.
* Under the **outputs**, change the **name** to “**DimCustomerRaw**”.
* Under the **transformations**, under **source**, change the **type** from **SqlSource** to ”BlobSource”. Delete the comma at end and also delete the “sqlReaderQuery” attribute.
* Leave the “**policy**” section as is for now.
* Click on the “**Deploy**” icon at the top right bar.

This action should create the pipeline between our datasets. Now, close the “Author and deploy” blades. Now, we should have two datasets, one pipeline and one linked service defined in your data factory, figure 2-13.

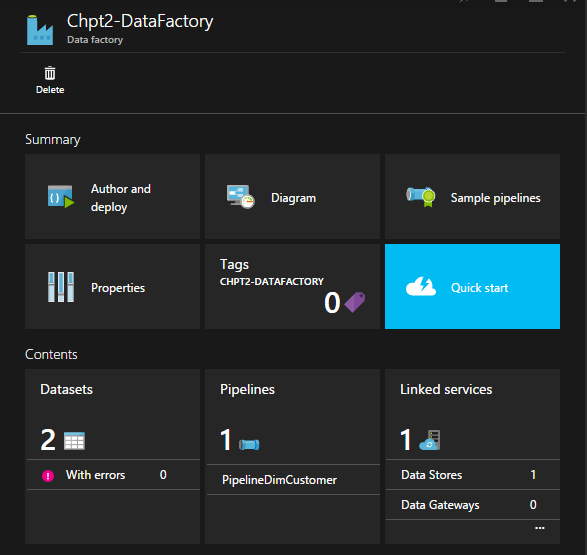


Figure 2‑1‑

* Click on the “Diagram” and double click on “DimCustomerRaw”.

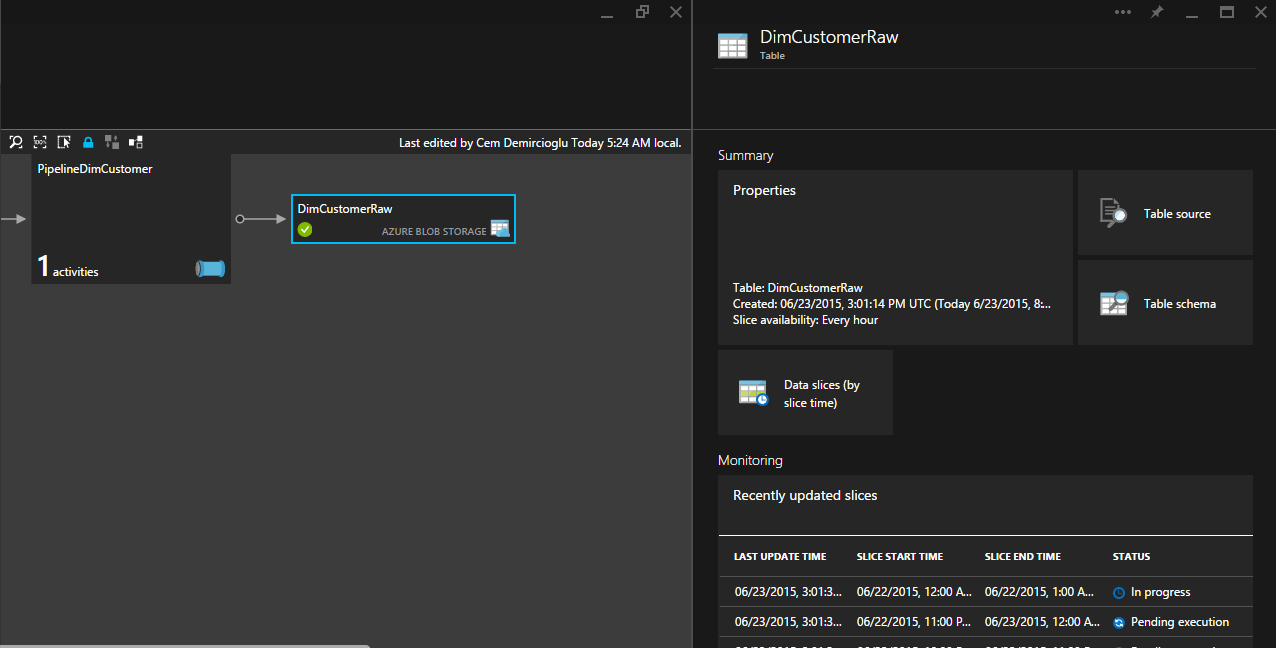


Figure 2‑1‑

You should see that the data is being processed or in progress. Once one of the slicers status changes to “Ready”, take a look at the “raw” container using Azure Explorer. Our DimCustomer file should be there, waiting for us.

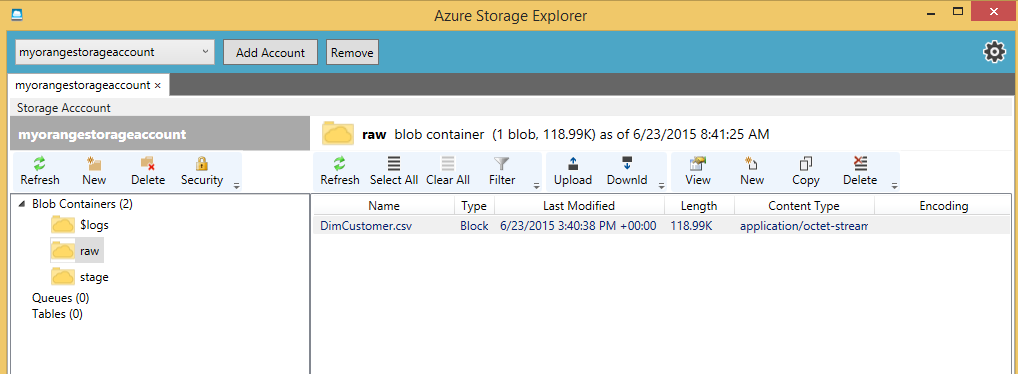


Figure 2‑1

This concludes our chapter.