# Introduction

## Why Azure Data Factory

Azure Data Factory is a service available on Microsoft Azure platform. This service allows you to do the following in the clouds:

* Manage data transfers from On-Premise and Azure cloud storage spaces.
  + Transfer data from a variety of sources (SQL Server, Oracle, file system, Azure blobs, etc.)
  + Monitor data transfers and easily recover from failure
* Coordinate Azure HDInsight, Stream Analytics computing resources.
* Simplify cloud based data ingestion like DocumentDB , Hadoop (HDInsight) and Azure Batch.

## Comparison with SSIS

### Two different paradigms

SSIS is an ETL (extract, transform and load) tool that comes with SQL Server Standard edition and higher. Its primary mission is to extract/transform and load on premise data. Since June 2015, an Azure feature pack is available for SQL Server 2012 and 2014. This feature pack allows SSIS to download and upload files from and to with Azure storage. It also allows us to create and drop HDInsight clusters as well as calling Pig or Hive script from there clusters.

SSIS is a service that has to be installed on an on premise Windows server and comes with SQL Server. When a new version of SQL Server is released, you have the option to upgrade your server to that version and get the new functionalities.

With a Microsoft Azure subscription, you can take of many service available. These include Azure Data Factory. Being a service means that you only pay for the time you use it. As opposed to on premise software, you don’t have to worry about updates. They’re applied automatically in Azure. Let’s say for example that they add a new feature or improve performance on specific modules, you don’t have to do anything. The new features will be available as soon as Microsoft make them available in Azure.

Azure data factory allows you manage data movements with existing ETL tools (e.g. SSIS) as well as Big Data technologies like Pig/Hive. It allows you to monitor and take advantage of data created or transformed via an ETL tool, bring it to the clouds and furtherly have it transformed by Hive or Pig. It can then be integrated with other cloud or on premise data down the road. Azure Data Factory by itself doesn’t have any transformation engine, it coordinate the data movement and by using other services or tools. It doesn’t have any transformation engine like SSIS have.

### Data Lake

With the proliferation of data sources especially with Internet of things (IOT, sensor data, communication devices, etc.), comes a new paradigm: the data lake. A data lake is the combination of structured, semi-structured or non-structured data that is loaded in a storage location that is not expensive. In Azure, we’re talking of an Azure Storage account. Chapter 2 - Components , describes what a storage account is.

As data warehouse developers, we are always solicited to load all kinds of data sources. The difference is that with a data warehouse, we have to know what the user wants to do with it because we have to add it to our data warehouse models.

With big data, things are different. We have to load data that would sometimes make no sense to have it in data warehouse since it cannot be modeled without doing lots of transformations. Also that same data might arrive in massive volume, in near real time or real time schedule that doesn’t go well with data warehouse batch load mode.

From all of the above concerns, it’s clear that we need another eat of integrating data; the data lake. A data lake has basically the following elements:

* A large and cheap storage that contains raw data, be it structured, semi structured or non-structured.
* A data warehouse in Azure SQL, on-premise SQL server or Oracle as well of SQL Server or Oracle in an Azure virtual machine. The DW can be an active part of the data lake. In some cases, it might contains most of the structured data in the data lake.

### To sum up

Azure data factory is also more cloud aware than SSIS. As you’ll see in this book, it’s built to manage and orchestrate big data transfers. SSIS on the other hand has almost everything we need to interact with on premise data sources and services.

In conclusion, SSIS and Azure Data Factory share common grounds but are aimed at two different platforms:

* On premise ETL for SSIS and some Azure components interactions with the SSIS Azure Feature pack.
* In the clouds ELT for Azure Data Factory with on premise interaction using a data management gateway\*.

\* We’ll talk about the data management gateway in chapter 4 – Pipeline activities.

# 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.

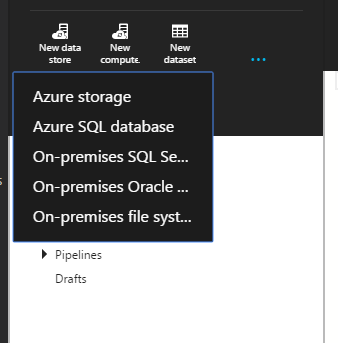


Figure ‑

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).*

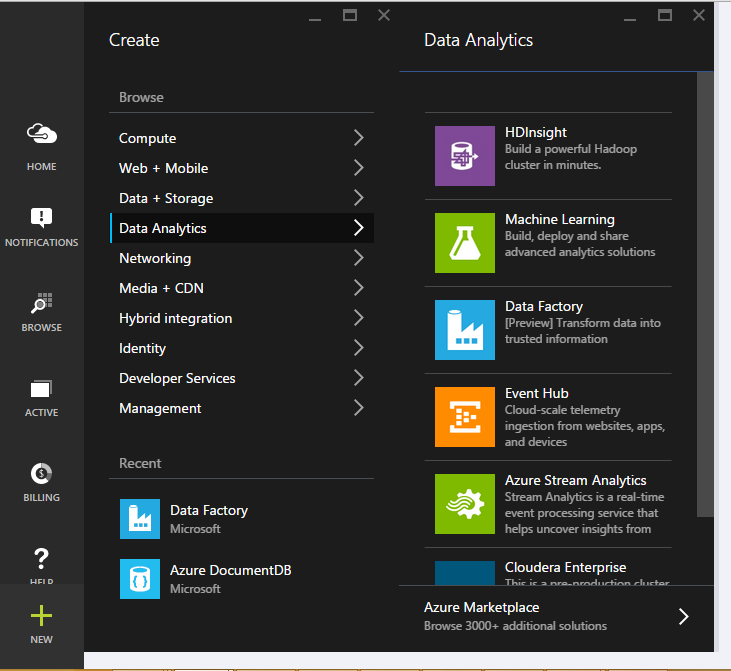


Figure 2‑‑

* ***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.)*

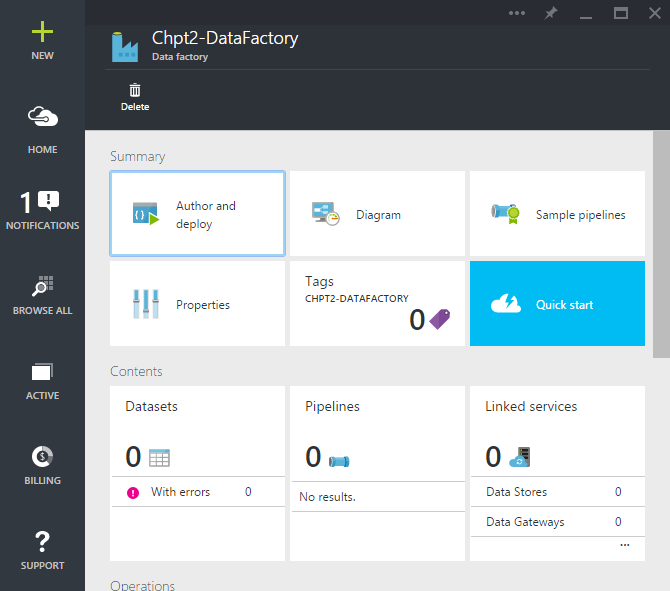


Figure 2‑3

* *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.*

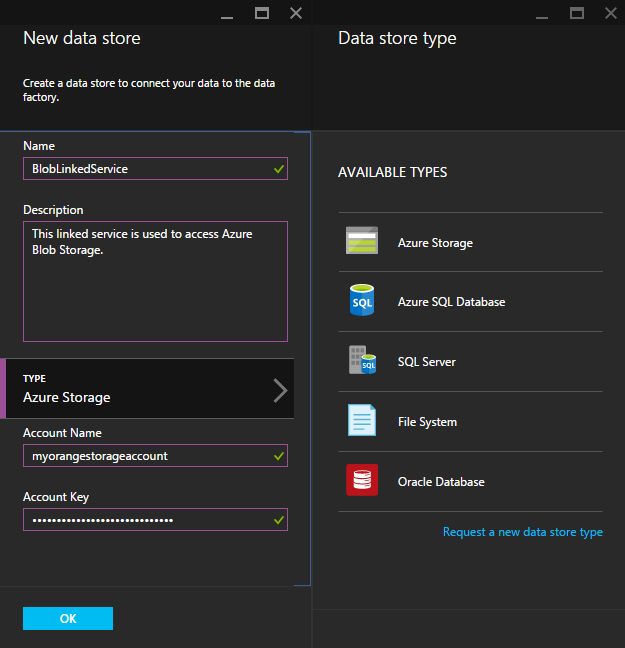
**

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.*

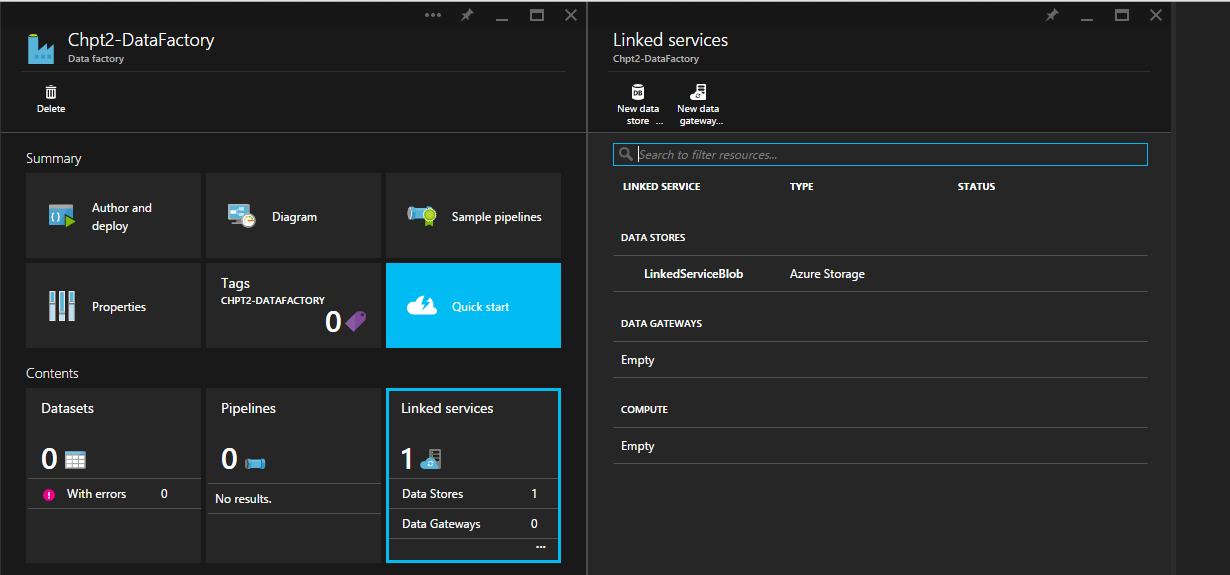


Figure 2-5

## 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/

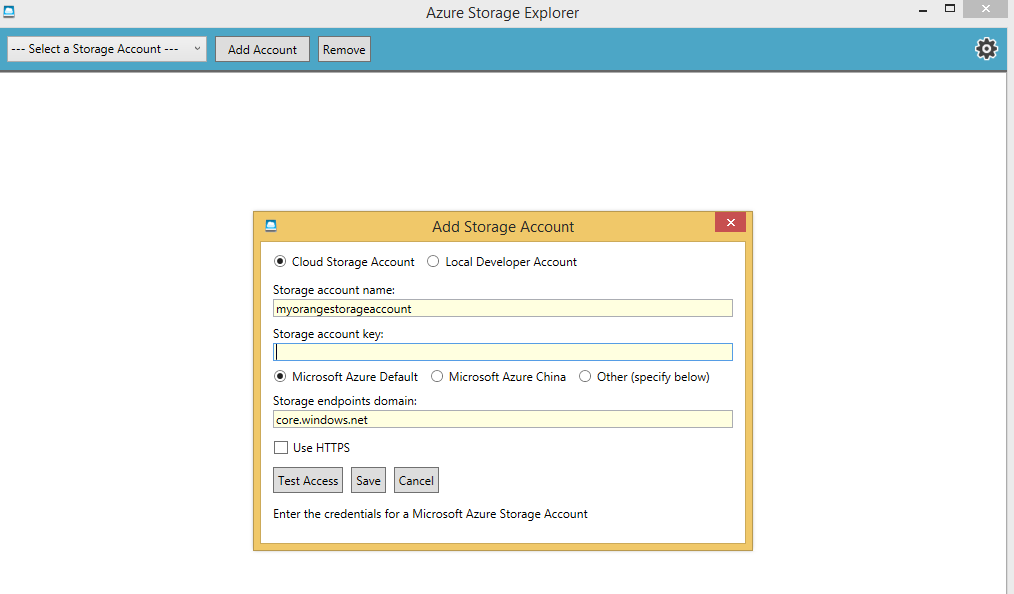
When you run the Azure Data Explorer, click on the “**Add Account**” icon at the top.

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.*

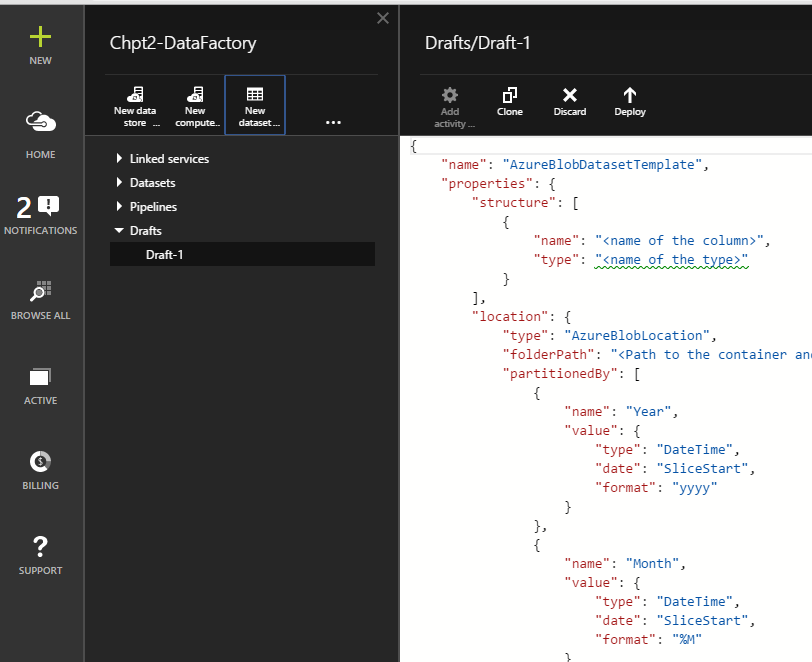


Figure 2‑8

* ***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.

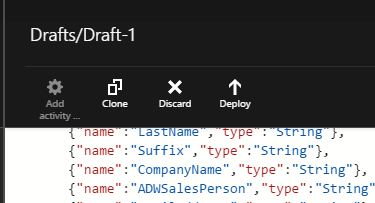


Figure 2‑9

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.

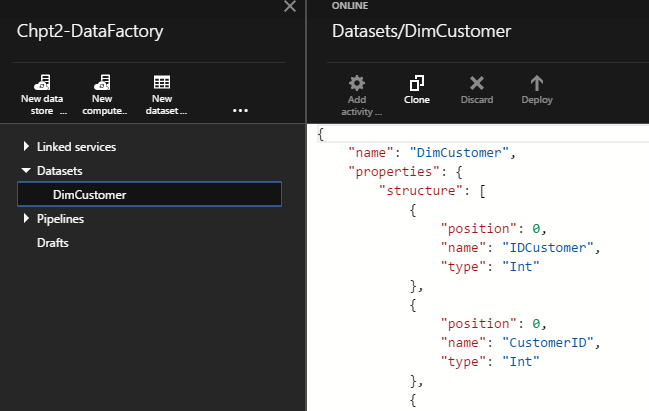


Figure 2‑10

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).

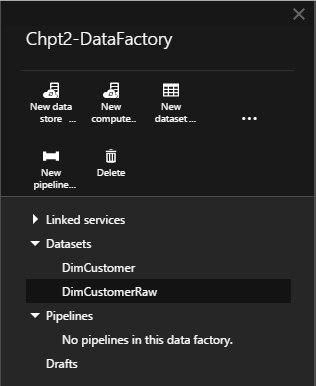


Figure 2‑11

* 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.

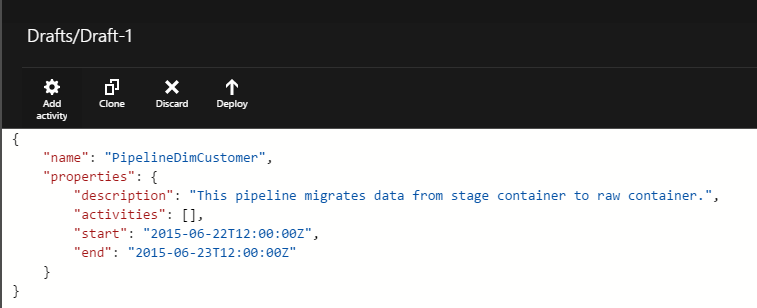


Figure 2‑12

* 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.

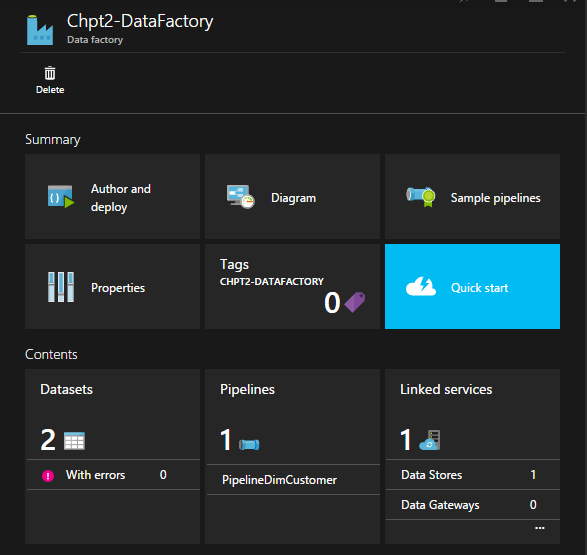


Figure 2‑1‑

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

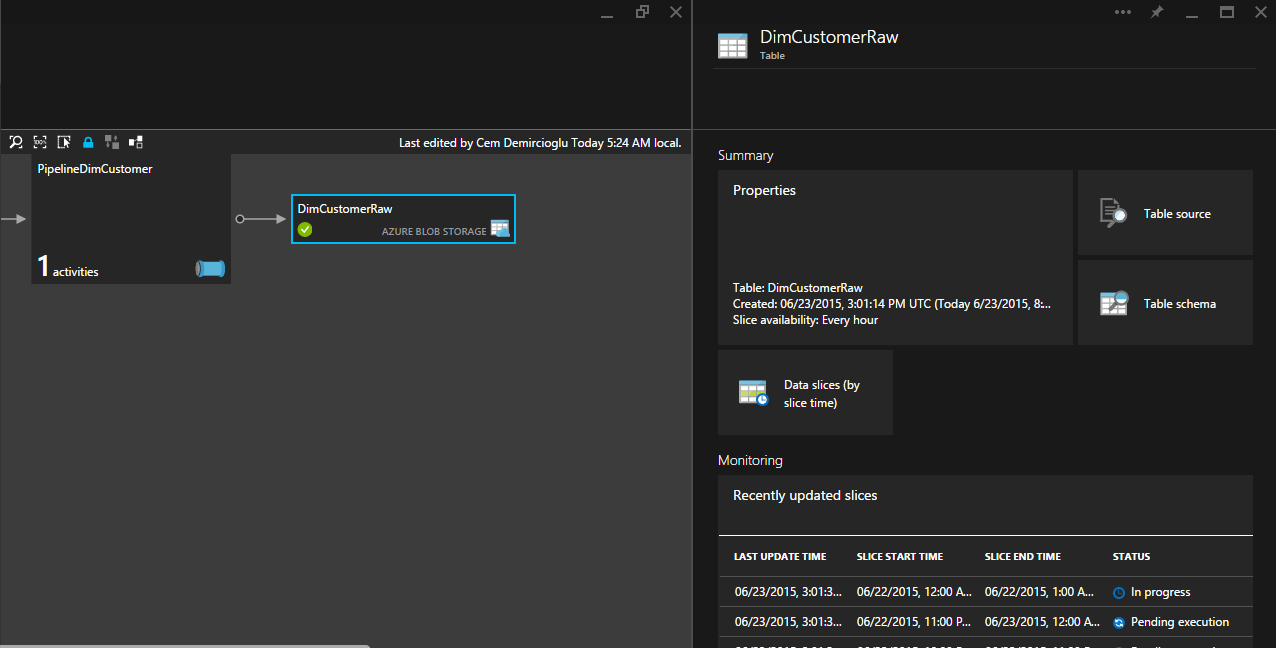


Figure 2‑1‑

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.

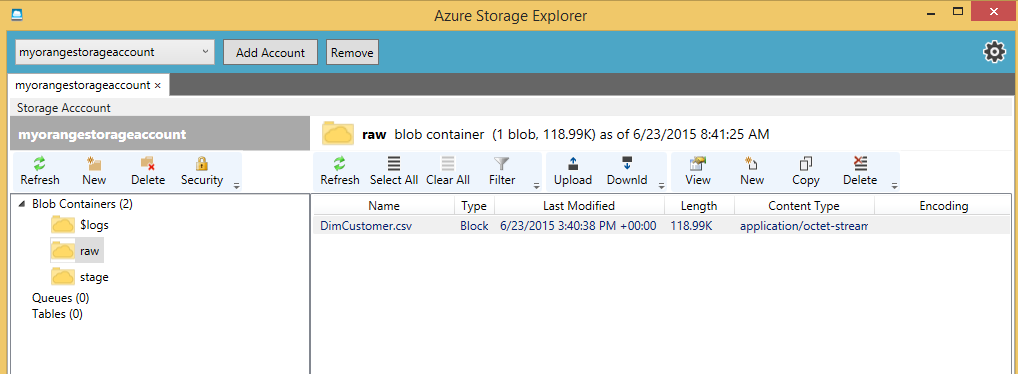


Figure 2‑1

This concludes our chapter.

# Getting ready

## Azure account setup

Since Azure Data Factory relies on Azure cloud, you’ll need am Azure account to create, manage and interact with your factory (ies).

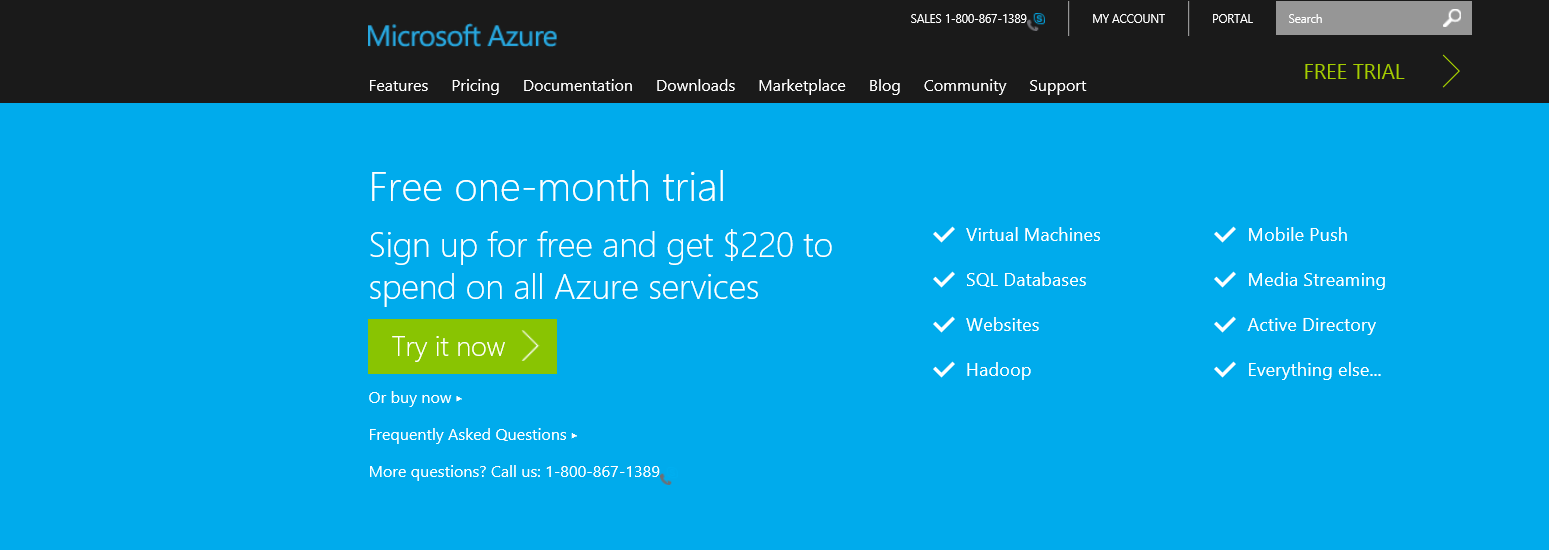
To do so you first go to the [Microsoft Azure web site](http://azure.microsoft.com/en-us/pricing/free-trial/?WT.mc_id=azurebg_US_sem_bing_BR_BRTop_Nontest_FreeTrial_azure&WT.srch=1) and sign in with your Microsoft account that is linked to Microsoft Azure. If you don’t have a Microsoft account, you have the option to get a one-month trial for free.

Figure ‑

If you select “Try it now”, you are redirected to the following scree. You now have two options: Sign-in with your Microsoft account or create a new one by clicking “Sign up now” in the screen below.

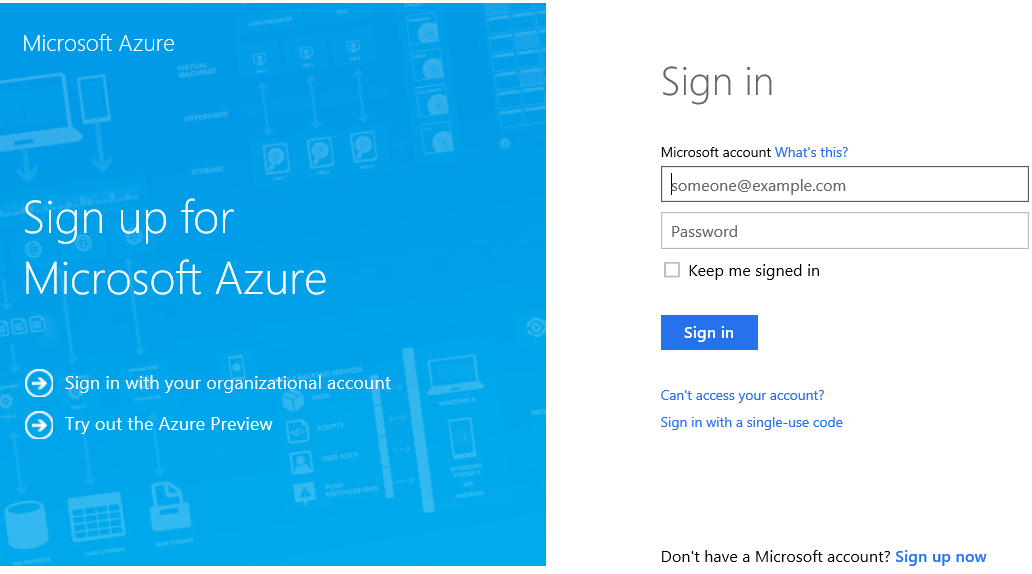


Figure ‑

Once you have setup your account, go the new [Azure portal](https://portal.azure.com/) and enter your credentials to log in. You should see a screen similar to the following one:

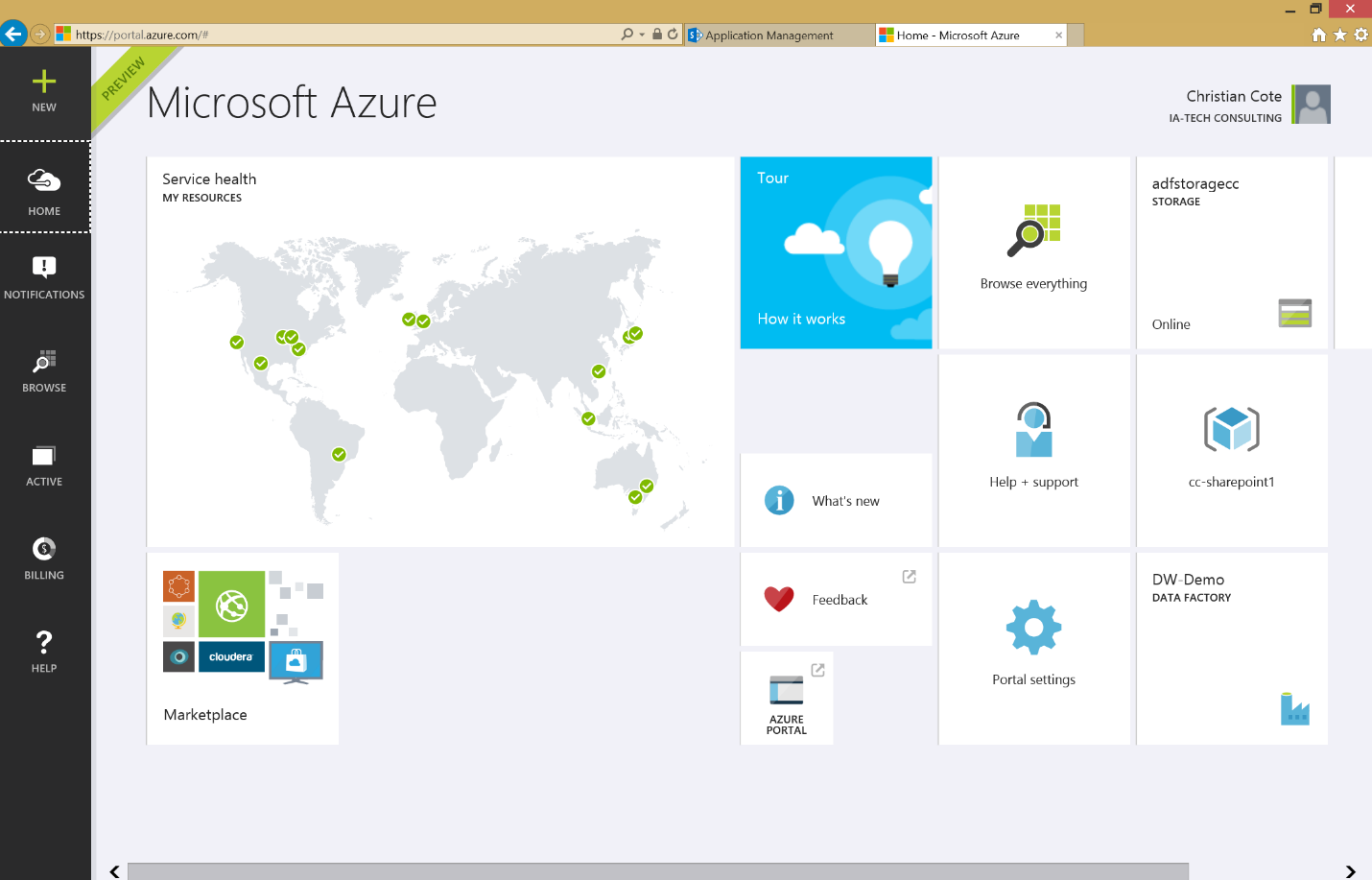




Figure ‑

In this book, we’ve changed the default theme to use the dark one. To do this, click on the portal settings (gear). The following screen appears:

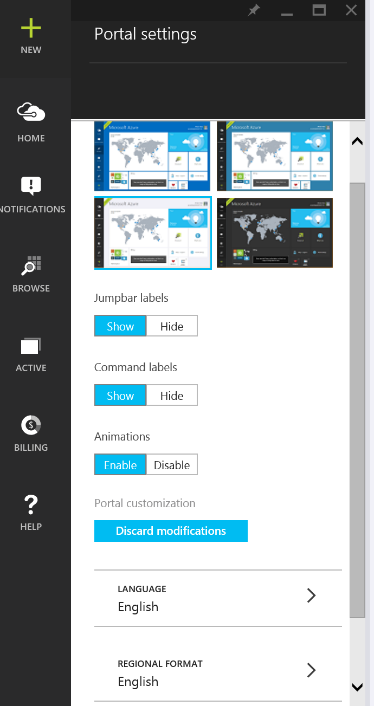


Figure ‑

This screen allows you to customize the portal as you like. But the only thing we want here is to use the dark theme. So we click on the dark theme to select it (Figure 3‑5)



Figure ‑

The background instantaneously turn black. Now, click on the X at the top of the Portal setting blade to close it as shown in Figure 3‑6 below.

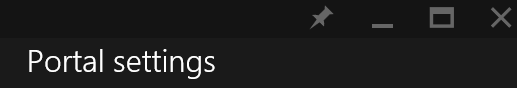


Figure ‑

## Creating a storage account

The next step is to configure the storage account. Almost every services in Azure are using storage. The counterpart of Azure storage on premise would be the hard drives or thumb drives on computers or servers. They contains files that are used by us or various services in Azure. Almost everything we do on Azure generate files. For example, when we execute a map reduce job on an HDInsight cluster, it generates a lot of log files that is the result of the job execution as well of the execution by itself.

### Create a storage from the portal

From the portal main page, click on the + sign, select “Data + Storage” and “Storage” at the right.

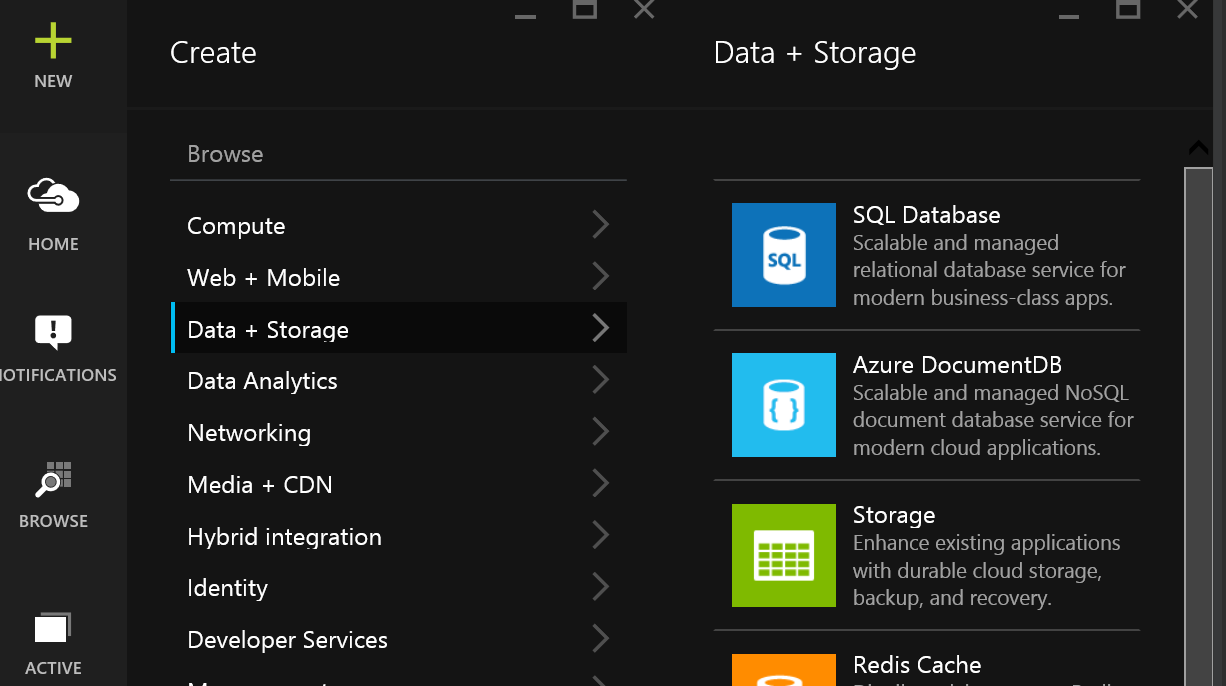


Figure ‑

The storage account blade appears.

#### Storage name

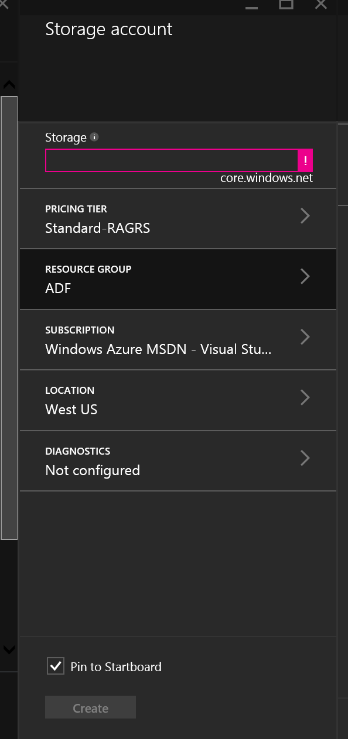


Figure ‑

The name of the storage has to be in lowercase, can only contains letters and digits and has to be unique. The reason for the uniqueness is that the storage has an address accessible via internet. For example, if I select MyAdfStorage, it will be known as being myadfstorage.core.windows.net and that’s how many tools or service will refer to it.

As you can see on Figure 3‑8 the name is validated as we enter it. In that case, the text box is magenta because we didn’t provided any name for the storage. Also if you hover you mouse over the exclamation mark at the right of the storage name text box (Figure 3‑9), you’ll get the validation error message that tells you what’s wrong with the name of your storage.



Figure ‑

Once the chosen name is valid, you’ll get a check green mark that tells you that the name is ok.

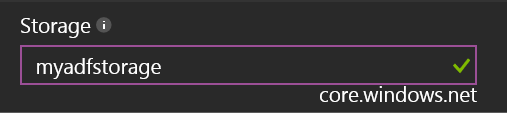


Figure ‑

#### Pricing tier

The pricing tier blade allows you to select the kind of storage you want to use. For the purpose of this book, we’ll use the basic (cheapest) one. But depending on your service level agreement (SLA) or requirements with your client or users, you might want to use a more expensive and performant storage.

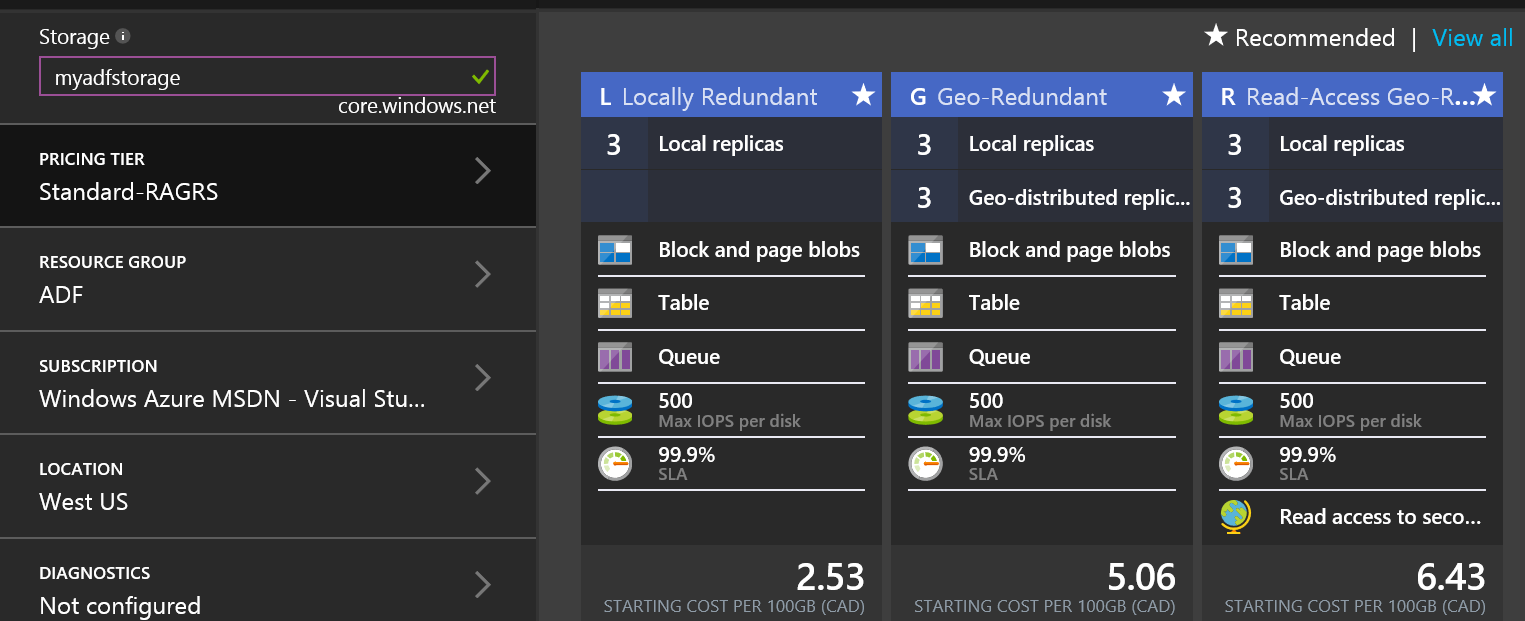


Figure ‑

If you click on “View all” as shown at the top right of Figure 3‑11, you’ll get more options in term of storage.

By default, all storage account types keep 3 copies of the files that you create or upload. This would allow file recovery if one file is damaged on a server for some reason. Some storage account options offer the possibility to geographically replicate and distribute your files at different data centers around the world. This gives more security and access to secondary storage via geographically distributed location.

With other storage options, you can have faster disk operations but at a higher cost. Prices varies upon time and are for each 100GB of monthly storage at time of writing.

#### Resource group

A resource group allows you to centralize resources into a common group in Azure. Think about them as folders in Windows/ A folder can have different types of files and / or folders. For example, let’s pretend you create a resource group named MyResourceGroup and you use it for factories, storage, HDInsight clusters and machines. When you decide to delete the resource group, all contained Azure resources contained into it will be deleted too.

You can access your resource groups in the Azure portal by clicking on “Browse” as shown on Figure 3‑12.

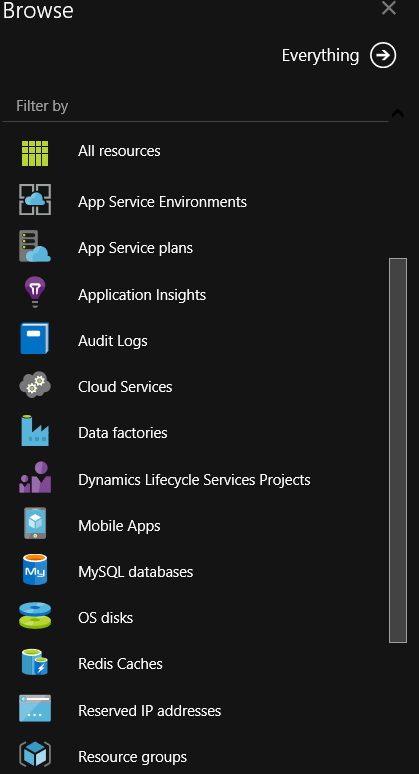


Figure ‑

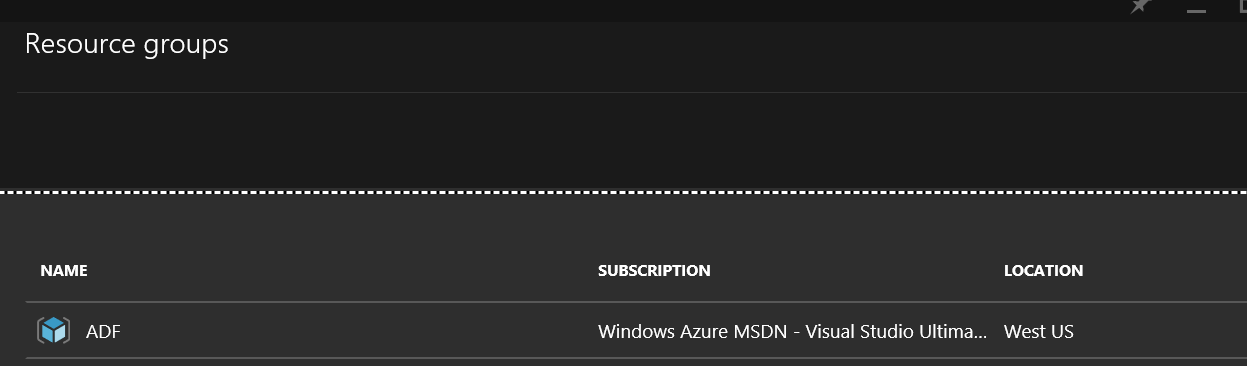
The Resource groups Azure Portal blade now appears.

Figure ‑

We can’t create resource groups individually, they have to be created when we create individual resources like storages, factories, etc. Every time theses individual resources are created, we have the option to specify a resource group.

#### Subscription

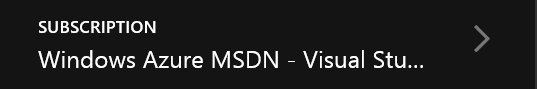


Figure ‑

The subscription blade allows you to select the Azure subscription you are using (and which will be invoiced) to interact with the storage. As we have talked about in Pricing tier section, depending on the type of storage you’ll be using and, as we’ll talk about in the Location section, storage is not free. You receive a monthly invoice for it as part of your Azure subscription.

#### Location

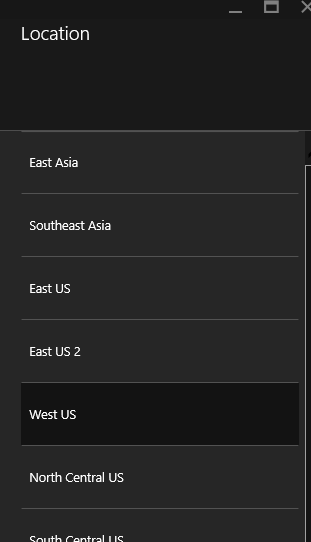


Figure ‑

When you create your storage resource, you have the option to create in different data centers in the world. The location that you choose has implications only if you plan to transfer data from one storage resource zone to another. And it depends if the data arrives in the zone (inbound) or goes out of it (outbound). For inbound traffic, there are no charges. But you will have to pay for outbound data or file movements.

Here are the various zones available at time of writing:

•Zone 1: US West, US East, US North Central, US South Central, US East 2, US Central, Europe West, Europe North

•Zone 2: Asia Pacific East, Asia Pacific Southeast, Japan East, Japan West, Australia East, Australia Southeast

•Zone 3: Brazil South

Notice that there are no charges for data transfer between locations that are in the same zone. For example, if you use Azure Data Factory to copy data from a blob storage that is located fin East-US to a SQL database located in Asia, depending on how much data we are transferring, charges may be applied. At time of writing, all monthly transfers below 5 gigabytes are free. You can refer to Microsoft Azure Data Transfers Pricing Details ( [http://azure.microsoft.com/en-us/pricing/details/data-transfers/#](http://azure.microsoft.com/en-us/pricing/details/data-transfers/)) if you want to see all available options related to data transfers between zones.

## Sample case used throughout this book

The case we’ll use in this book is based on AdventureWorks LT database. This database was available on [codeplex](http://msftdbprodsamples.codeplex.com/releases/view/55330) for all recent SQL Server edition up to SQL Server 2012. On the top of this database, we created a small data warehouse called AdventureWorksLTDW. This book’s support site will provide the necessary SQL Server data files for both databases.

### AdventureWorksLT database



Figure ‑

Figure ‑

### AdventureWorks LT DW

This database is a data warehouse that have been built on top of the operational database.

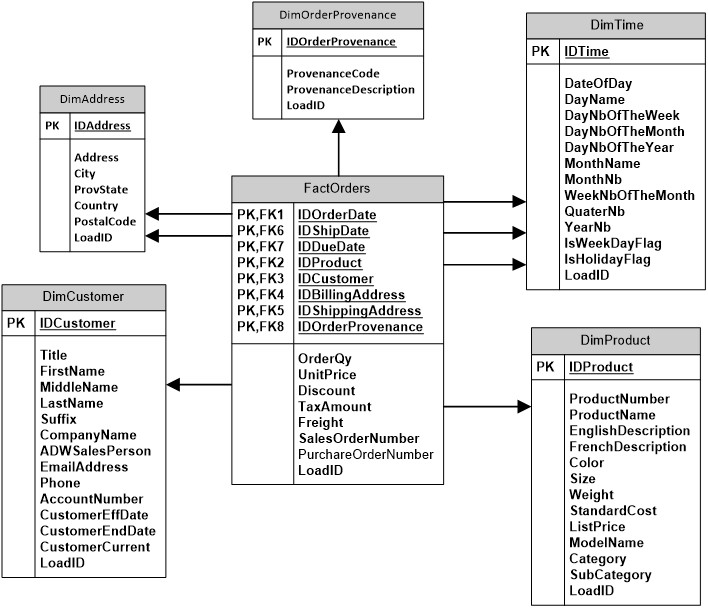


Figure ‑

The following list describes the main differences between AdventureWorksLT and the data warehouse.

* The tables “SalesOrderHeader” and “SalesOrderDetails” have been merged into a single fact table: FactOrders.
* We now have a single point of reference for all dates: DimDate. This dimension is referred three times in “FactOrders”. We call this type of links role playing dimension. The DimDate dimension can be used to constraint the fact table using the order, ship or due date of the order. This link provides more information’s on role playing dimensions.
* Product categories and sub categories have been denormalized in the “DimProduct” dimension table. This allow to easily navigate the fact table using the following hierarchy: Category🡪Sub category🡪products.
* We chose to keep only English and French product description. We denormalized them in the product dimension. This way, we can easily display the product in both languages.
* Customer’s addresses have their own dimension: DimAddress. Both billing and shipping addresses are linked to the fact table via the role playing dimension pattern. Since customer’s usually have several criteria that are related to sales that might change over the time, we modeled it as a [SCD type 2 dimension](https://en.wikipedia.org/wiki/Slowly_changing_dimension). This way, we’re able to compare sales for a customer before he/she was married and after.

## , Azure PowerShell

PowerShell is very useful when it comes to Azure. It allows Azure assets to be deployed from the command line and therefore automated. Once your Azure assets are scripted in PowerShell, they can be copied over to your source control software (e.g. Team Foundation Server (TFS), Subversion (SVN), Git, etc.).

### Installation

To setup Azure PowerShell, you need to download it from the following web site:

<https://azure.microsoft.com/en-us/documentation/articles/powershell-install-configure/>

The following screen appears:

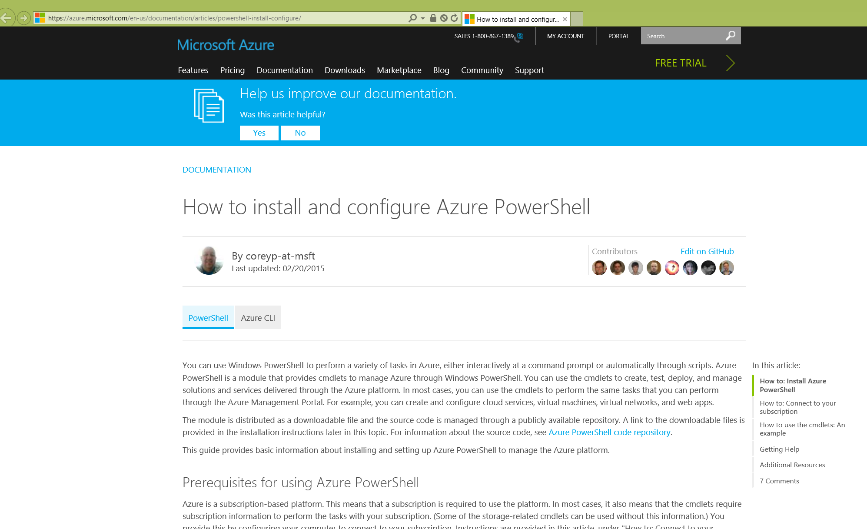


Figure ‑

In the upper section of the page you’ll see a section called: “How to: Install Azure PowerShell”:

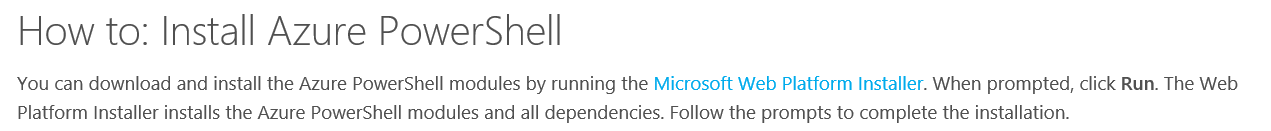


Figure ‑

Click on “Microsoft Web Platform Installer” to download and run the installer:



Figure ‑

The installer starts and the following screen appears when it’s ready to proceed at the installation:

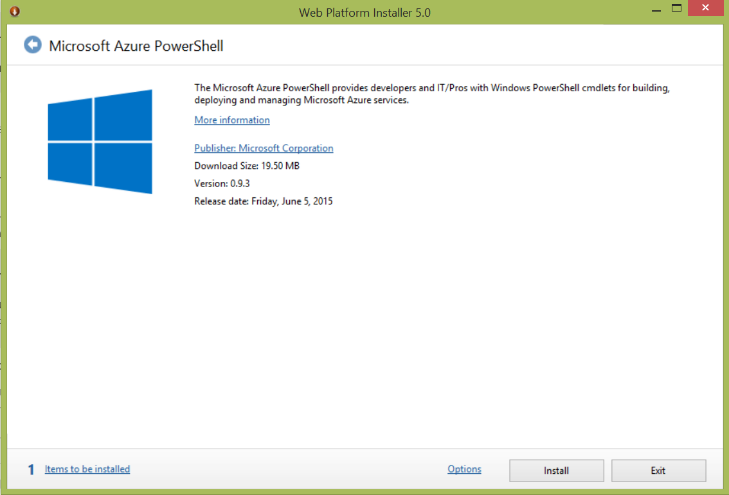


Figure ‑

You’ll notice that there’s at least one item to install as indicated at the bottom left of the screen:

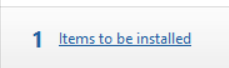


Figure ‑

You can click on the text “Items to be installed” to see what’s gonna be installed within this installation:

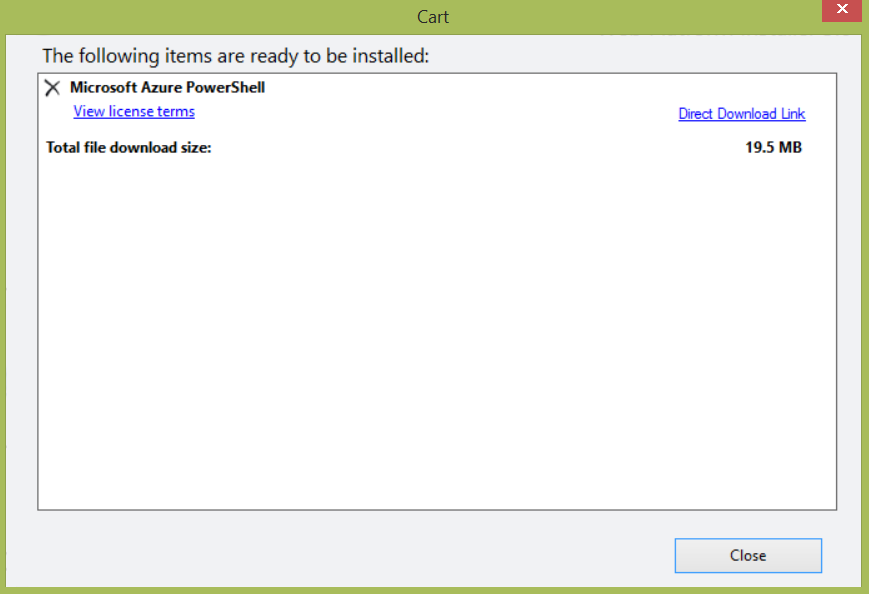


Figure ‑

Click close and click install in the main installer screen (Figure 3‑20). You’ll be directed to a prerequisites window telling you that you accept installation and licence terms:

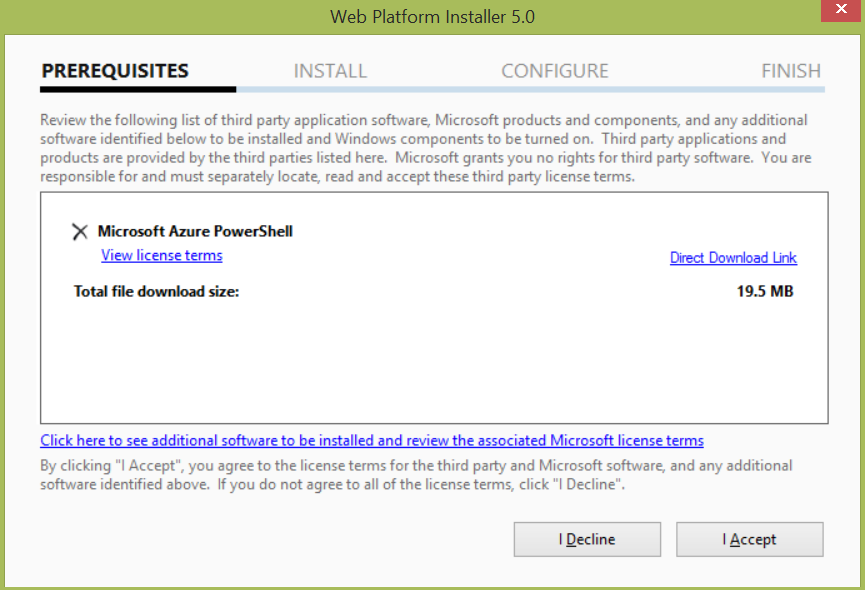


Figure ‑

Click I Accept to begin installation process:

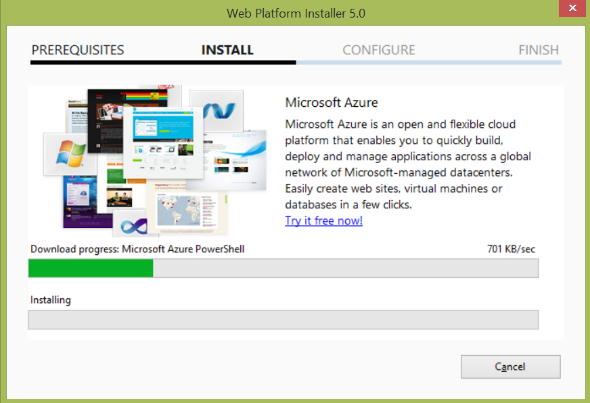


Figure ‑

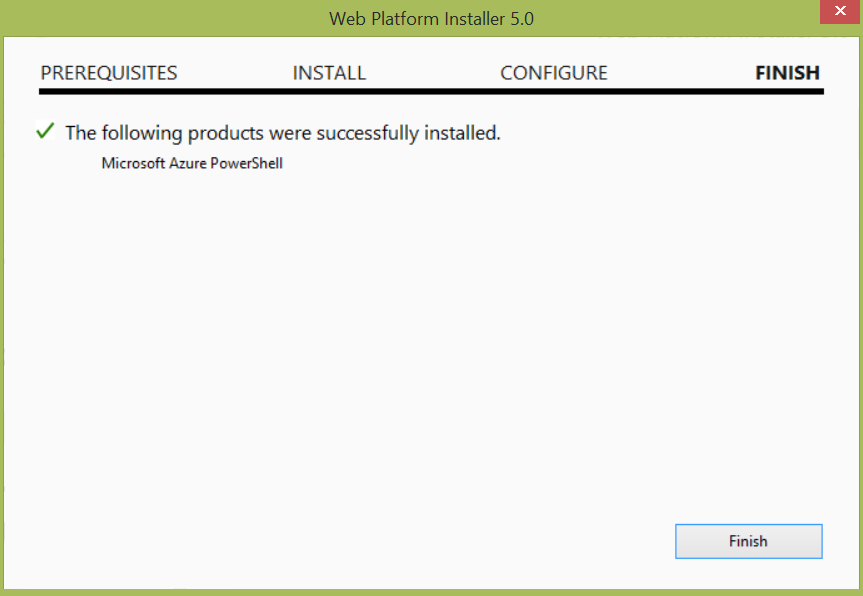
Once the installation finishes successfully, the following screen appears, click Finish to get rid of the screen. Installation of Azure PowerShell has now completed.  
  


Figure ‑

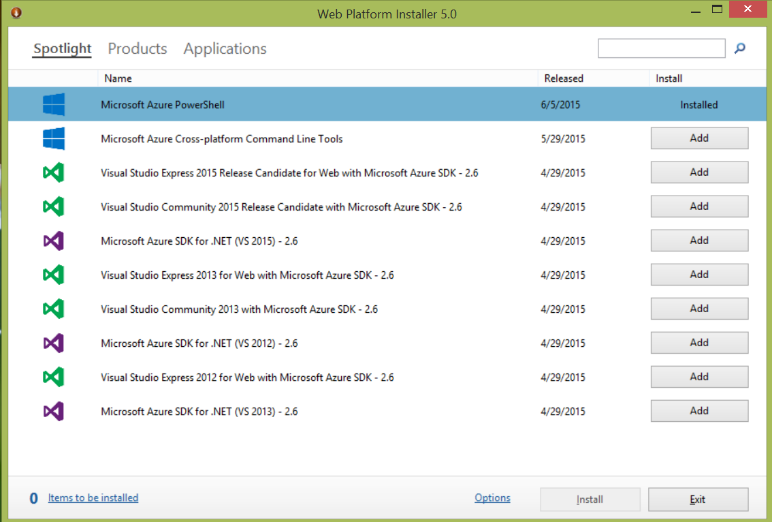
The installer allows you to install supplementary modules, click on the Exit button to close the screen:  
  


Figure ‑

### Verify Azure PowerShell installation

#### Windows 7

#### Windows 8.1

From the start screen, enter Azure PowerShell in the search text box and click Microsoft Azure PowerShell in the search results

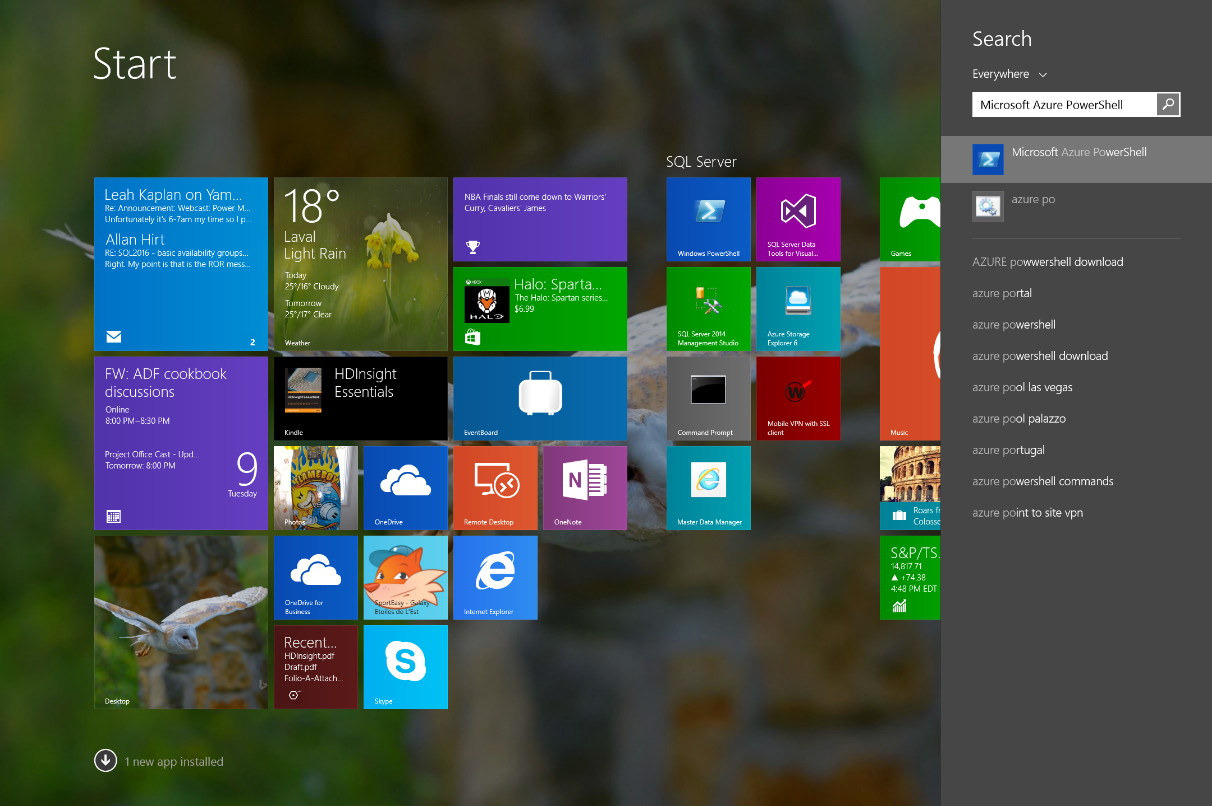


Figure ‑

### Windows 10

The following screen appears confirming that Azure PowerShell is correctly installed and works properly:

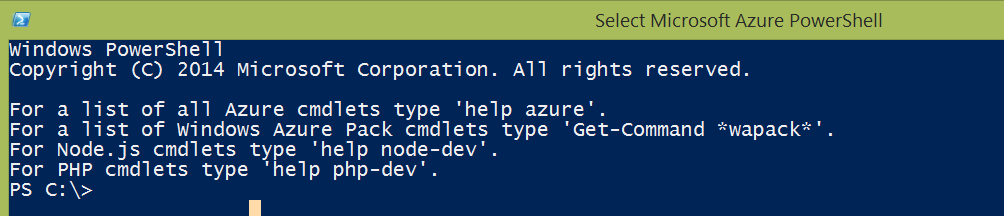


Figure ‑

To have a faster access to Azure PowerShell, you can also pin the program to your task bar. To do so, from the search result in Windows 8.1 or the all apps list in Windows 10, right-click on Microsoft Azure PowerShell and select the “Pin to taskbar” option from the drop down menu.

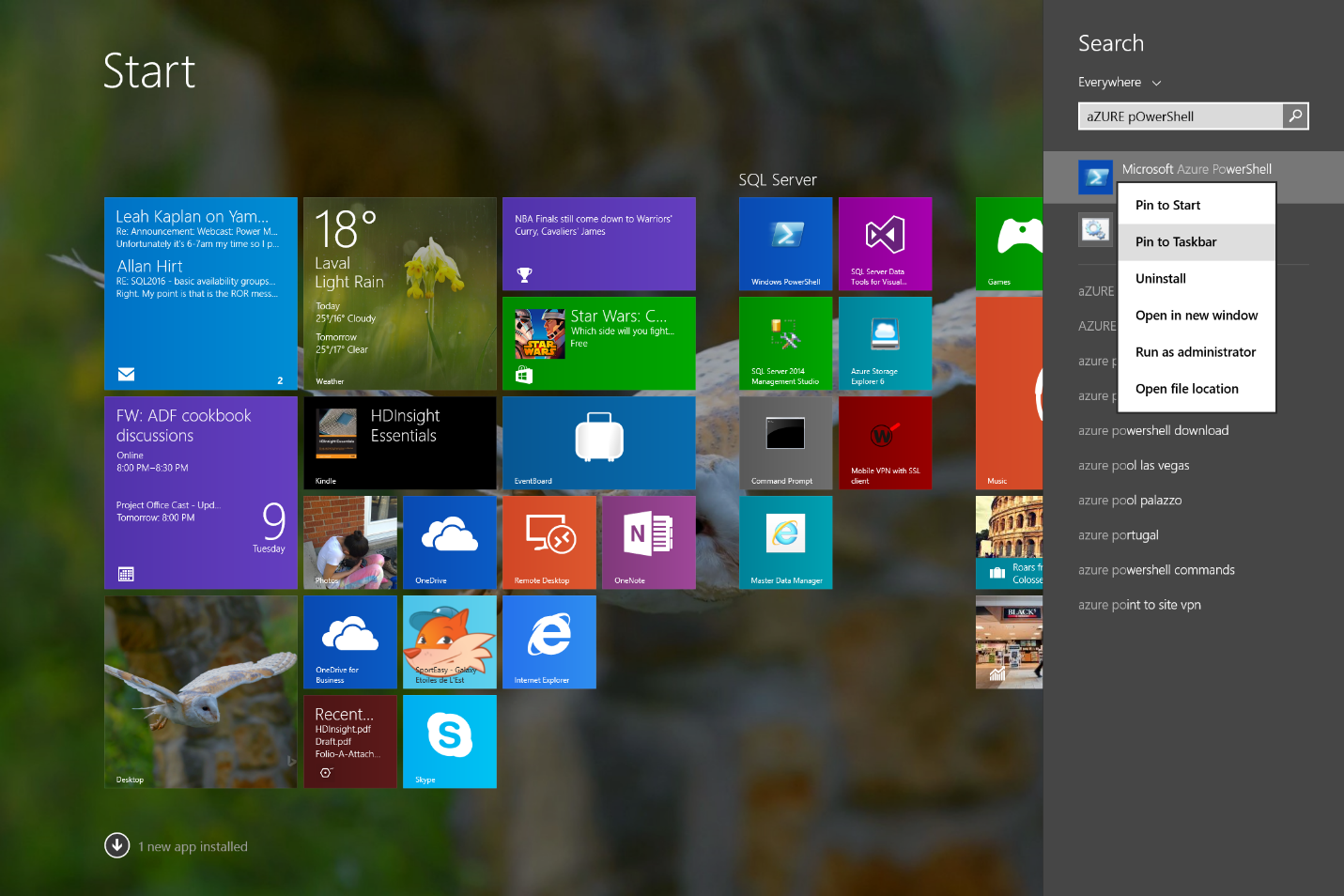


Figure ‑

Now, go to the desktop and you should see the following icon on your taskbar:



Figure ‑

Now, when you right-click on the icon in the taskbar, you can see these options:

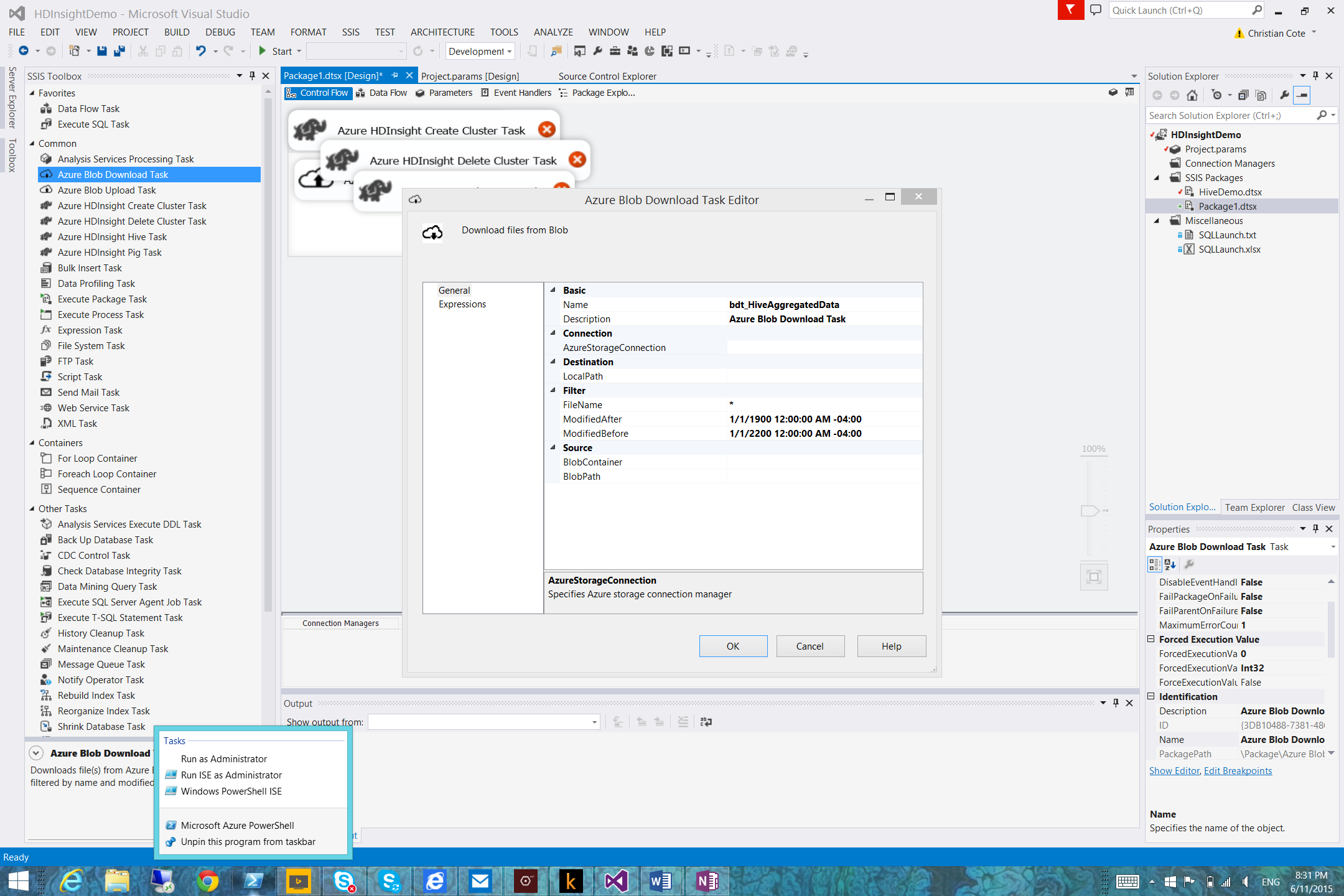


Figure ‑

You now have the option to open the following programs:

|  |  |
| --- | --- |
| Menu choice | Description |
| Run as Administrator | Opens regular Azure PowerShell like the one referred by Figure 3‑28 using local administrator privileges. |
| Run ISE as Administrator | Opens Windows PowerShell Integrated Scripting Environment (ISE) using local administrator privileges.(Figure 3‑32) |
| Windows PowerShell ISE | Opens Windows PowerShell Integrated Scripting Environment (ISE) using current user credentials. (Figure 3‑32) |

Here’s a screen capture of the Windows PowerShell Integrated Scripting Environment (ISE)



Figure ‑

This version of PowerShell has several capabilities compared to the “plain” version you’ve seen before (Figure 3‑28) such as multiline edition, debugging and possibility to run part of a script. You can learn more about PowerShell ISE here: <https://technet.microsoft.com/en-us/library/dd315244.aspx>.

# Deployment and operationalization

Once the factory is complete and running, we have to make sure of the following:

* We have the source code (JSON) files backed up safely, ideally in source control management tool.
* We’re able to extract and preserve in some ways historical execution logs.
* We’re able to trigger alerts when events occurred in our factory, more specifically, errors.

This chapter will cover the above concerns and gives you some ideas that can be extended and used in your own factories.

## Backup the factory

The Azure Data Factory interface in the portal doesn’t have the capability to download an entire factory structure. More specifically, we would like to extract and save the JSON code and keep a copy somewhere in case something wrong happens to our factory.

## Execution logs

One of the core things we want to achieve is to capture execution logs to be able to assess what went well and what did not. If something went wrong, we’d like to have as much details about it to be able to correct the issue. Keeping execution logs also help troubleshooting load processes. For example, we might notice that a pipeline activity took let’s say 5 minutes last few times and all of a sudden, it finishes after 10-15 minutes. Having the logs history, we know that the process takes longer than before and we’re able to investigate on that specific activity to understand why it takes more time now and correct the situation if necessary.

### Sample database framework to store execution logs

### PowerShell cmdlets to retrieve execution logs

## Deploying the factory using PowerShell

Creating a factory using the Azure Portal the easiest way. But there’s nothing available at the moment to make changes to our factory offline. For example, we might want to store our factory in source control software like Team Foundation in Azure like we would do with Visual Studio. Making changes offline has several benefits like adding slices in a data set and deploying them later or adding a Linked Service ahead of time of its availability.

For example, we might want to prepare our factory to integrate new data feeds from a Hive table but developers are still working on hive QL scripts. Adding the activity to the factory immediately would lead to failure and we might receive alerts that are not relevant for the moment.

### Modifying and deploying factory objects

PowerShell (as well as .Net as we’ll talk in the next section) has all cmdlets available to deploy a complete factory with all necessary objects.