Implementing an Azure Databricks Environment in Microsoft Azure

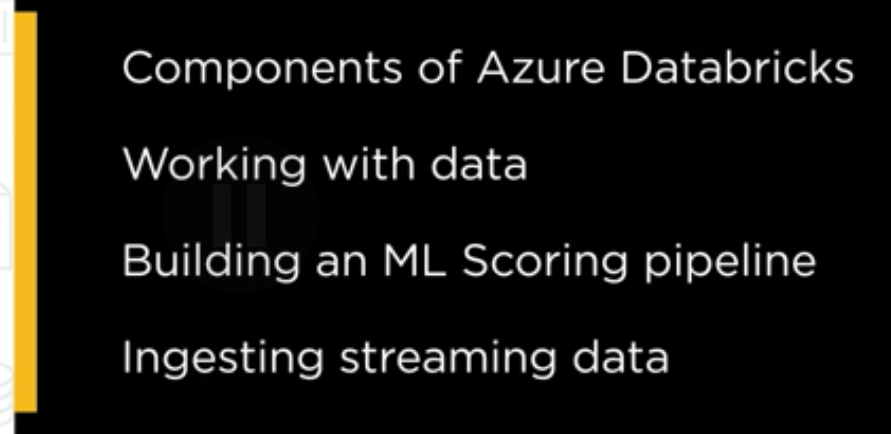
Contents

[Course Overview 1](#_Toc30331526)

[Implementing an Azure Databricks Environment 1](#_Toc30331527)

## Course Overview

[Autogenerated] Hi, everyone. My name is Michael Bender and welcome my course. Implementing Azure Databricks in Microsoft Azure. I'm an author. Evangelist Set quarrel site. Working with Big Data's a challenge as your dataBricks makes spinning up resource is to solve your big data analysis issues quick and easy. With this course, you'll gain the ability to implement azure Databricks for use by all of your data consumers like business users and data scientists. Some of the major topics that will cover include fundamental components of azure databricks , working with data and Databricks, notebooks, building an ML scoring pipeline and ingesting streaming data.



By the end of this course, you'll have the skills and knowledge of azure Databricks needed to implement data pipeline solutions for your data consumers before beginning the course, you should be familiar with building and employing azure data solutions like Azure SQL, Azure SQL, Data Warehouse and as your Data Lakes.

You should be ableto understand machine learning concepts and model building techniques and also know a common language like python Scala R for SQL for working with data.

From here, you should feel comfortable diving into azure data pipelines with courses on batch scoring, Azure Dev ops and Azure Automation. I hope you'll join me on this journey to learn azure Databricks with the implementing Azure dataBRicks in Microsoft Azure course at PLuralSight.

## Implementing an Azure Databricks Environment

Overview

Most data and analytics leaders realize that when it comes to embarking on new initiatives and the needs of intelligent applications, it's still really about the data first and foremost. Their teams need to figure out how to get a massive amount of data, often in real time, to their applications and processes in a way that supports an iterative process and generates a meaningful business result. And it needs to be collaborative and meet the needs of all types of consumers, from business users to data scientists. That's where Azure Databricks comes in.

In this course, Implementing an Azure Databricks Environment with Microsoft Azure, you'll learn about Databricks and how to implement it as a scalable, collaborative, analytics platform designed in the cloud. Along with this, you'll learn to use Azure Databricks with ETL, batch scoring, and streaming data scenarios.

I'm Michael Bender, and welcome to the first module, Implementing an Azure Databricks Environment on Pluralsight. I'm Michael Bender, and welcome to the first module, Implementing an Azure Databricks Environment. In this module, Implementing an Azure Databricks Environment, you'll be introduced to Azure Databricks, a collaborative, analytics platform in the cloud. You'll learn all about its components, including workspaces, clusters, and more.

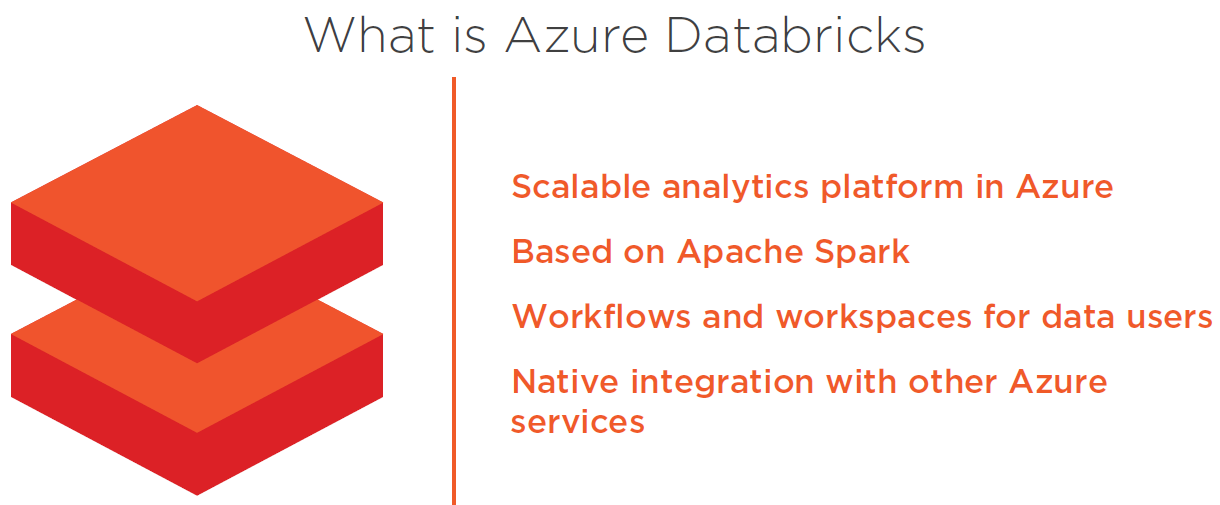
With plenty of demos, you'll see how to work with these important components with Azure Databricks. This will lay the foundation for you to begin using Azure Databricks. Upon completion of the module, you'll be ready to hit the ground running using Azure Databricks and ready to begin more complex usages like batch scoring and ETL in future modules.

So before we get started, I wanted to point out that this and every module has exercise files you can download from Pluralsight.com. This download will include a PDF of the slides, any code or file assets used during the courses for demos, and links to resources for more information about the topics that we covered. As with everything, we won't cover everything about Azure Databricks, so I'll provide you with a list of follow up resources for deeper learning. So to get started with the exercise files, you simply go to the course on Pluralsight.com, you find the Exercise Files tab, and then you click on the exercise files download. And that's it. Then you have the resources ready to go and work along with the demos. Now let's dive into Azure Databricks.

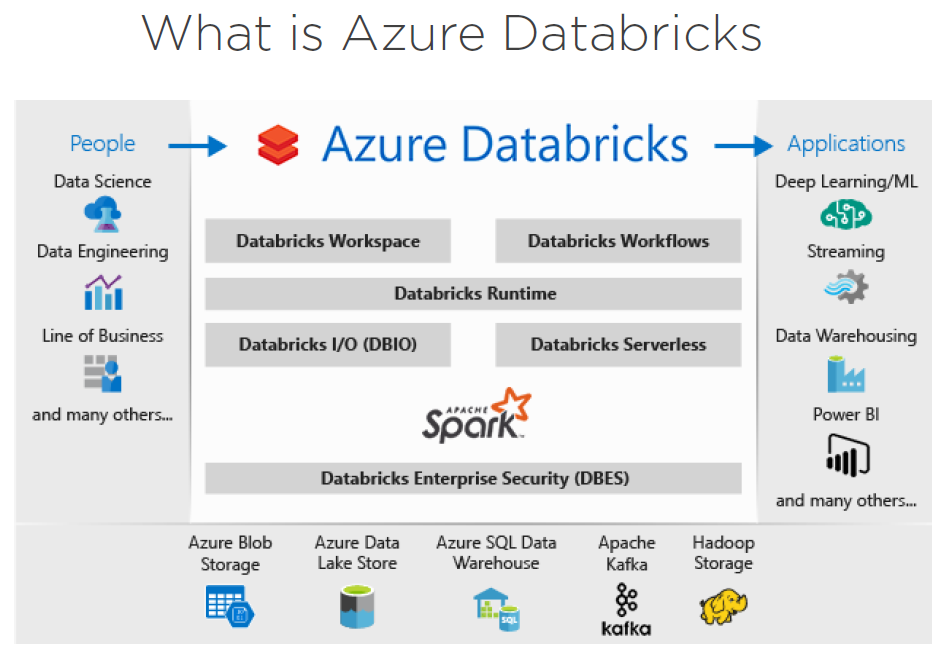
## Introduction to Azure Databricks

Azure Databricks is a scalable, collaborative, Apache Spark-based, analytics platform that's optimized for Microsoft Azure. Designed in collaboration with the founders of Apache Spark, Azure Databricks provides streamline workflows and an interactive workspace that enables collaboration between your data scientists, your data engineers, and your business analysts for faster innovation within your businesses.

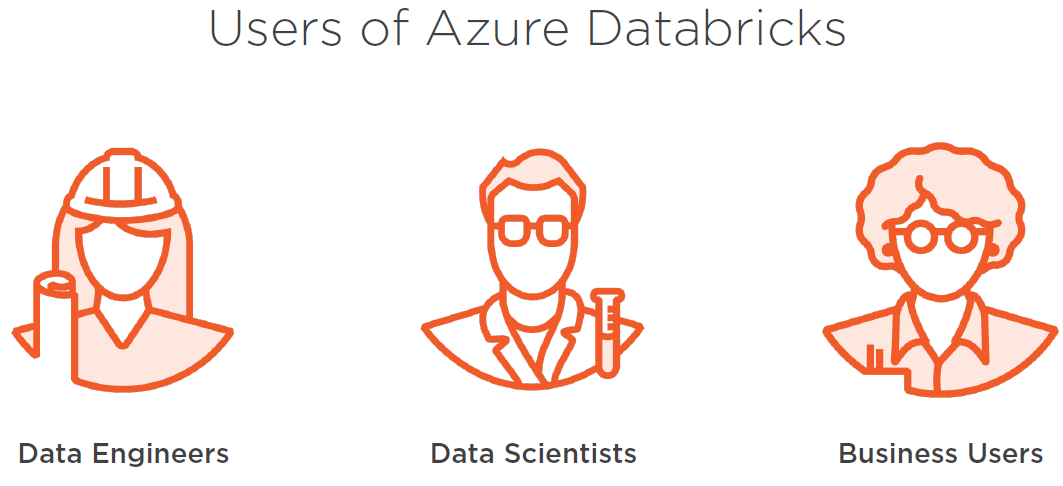
These workflows and workspaces provide the tooling for processing and modeling large amounts of data in the workspaces, a critical task in today's data pipeline scenarios. As an Azure service, users automatically benefit from native integration with other Azure services, such as Power BI, SQL Warehouse, Cosmos DB, and Active Directory.



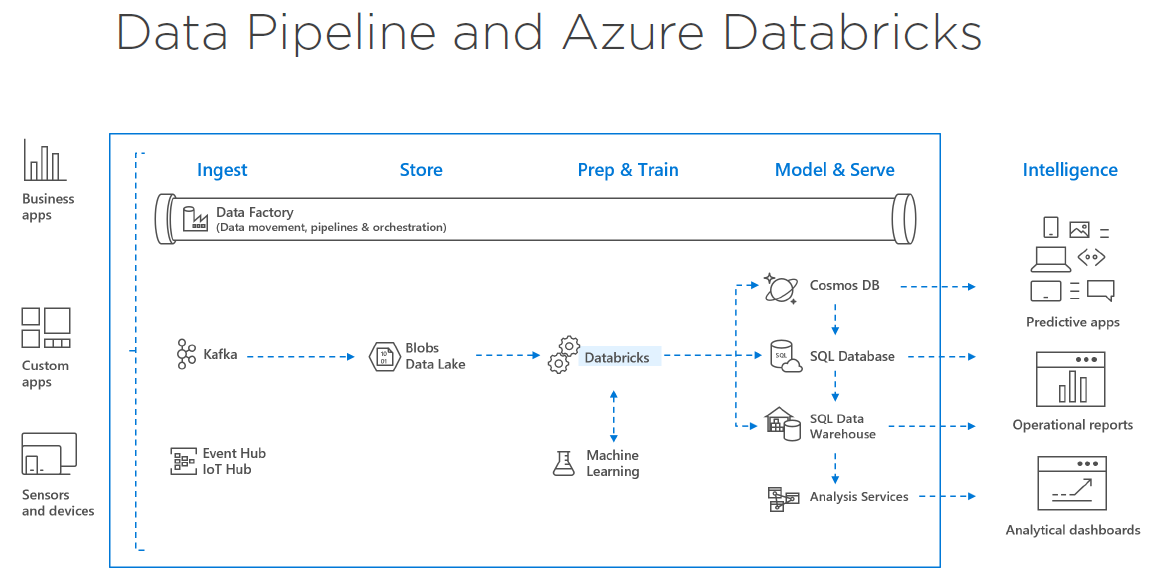
To give you a visual idea, Databricks works by providing a collaborative environment for your users to analyze, train, and prep data for different applications. The data can be sitting in various storage formats, like Azure Blob storage or a data lake, or it can be streaming data. Because it's based in the cloud, it's scalable and flexible to the needs of its users.



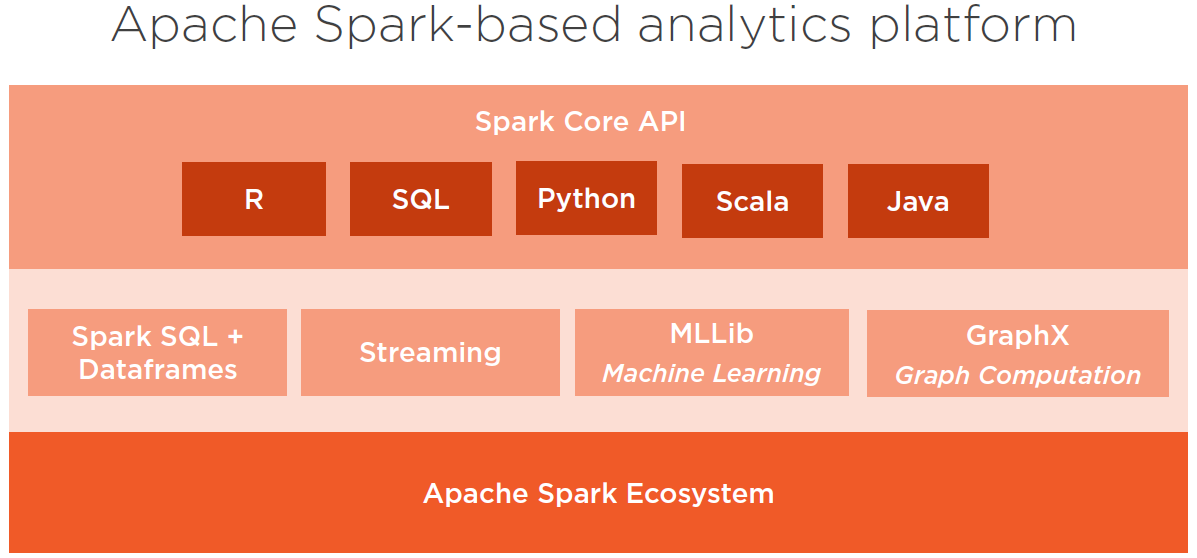
Speaking of users, Azure Databricks provides a single workspace for a variety of users.   
- It provides a place for data engineers to transform data and create and schedule batch and streaming ETL jobs. It makes it easy for them to build the infrastructure that's needed for all of your data transformation needs.   
- Data scientists can also use Azure Databricks to explore data, to create machine learning models, and perform other analytics tasks.   
- Even business analysts can write SQL queries and analyze and visualize data in Databricks notebooks. Whether you perform one or all of these roles, Azure Databricks provides a collaborative environment for your big data processing and analysis needs.



In the big picture data, Azure Databricks fits in as part of a logical processing pipeline when data transformation or analysis is needed by data users.



Starting on the left with your structured or unstructured data from various sources, data can be fed from a data factory or from streaming sources into raw storage, like Azure Blobs or Azure Data Lakes. Then Azure Databricks comes in at the train and prep phase to transform the extracted data based on the user's needs. Last, the transform data is made available for intelligent apps by way of various database structures like SQL Data Warehouse and Cosmos DB that are used to house the transformed data at the end.

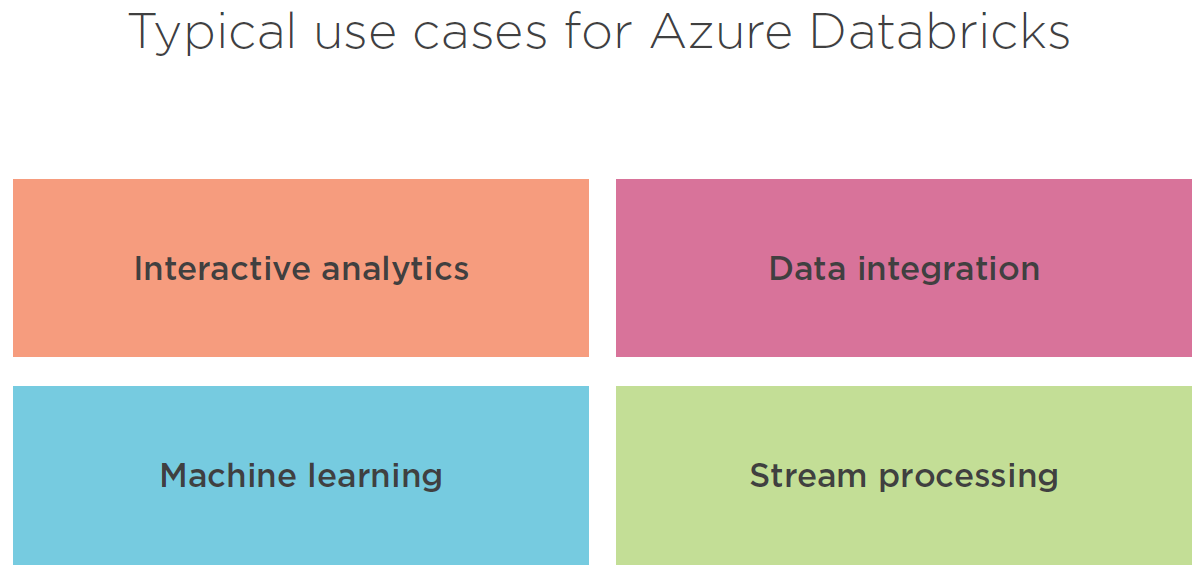


As I mentioned before, Azure Databricks is built on top of Apache Spark. Apache Spark is an open source distributed general purpose cluster computing framework. Spark provides an interface for programing the entire computing cluster, it's scalable, flexible, and fault tolerant.

As a first-class service in Azure, Azure Databricks comprises the complete set of open source Apache Spark cluster technologies and capabilities. So all of them are built in the box for you.

* First, Spark SQL is the Spark module for working with structured data within Azure Databricks. A data frame is a distributed collection of data organized into name columns. It's conceptually equivalent to a table in a relational database or a data frame in R or Python.
* Then we have streaming support, which provides real time data processing and analysis for analytical and interactive application. It integrates with things like HDFS, flume, and Apache Kafka.
* Next, we have the machine learning library, or MLLib. This consists of common learning algorithms and utilities for doing things like classification, regression, collaborative filtering, and many more things, all for working with machine learning within Azure Databricks and working with that in your data pipeline.
* Next, we have the GraphX. This provides graphs and graph computation for a broad scope of use cases from cognitive analytics to data exploration.
* And, last, we have the Spark Core API. This includes support for R, SQL, Python, Scala, and Java. So you can work with it using the language of your choice.

Now let's finish off with some of the use cases for Azure Databricks.

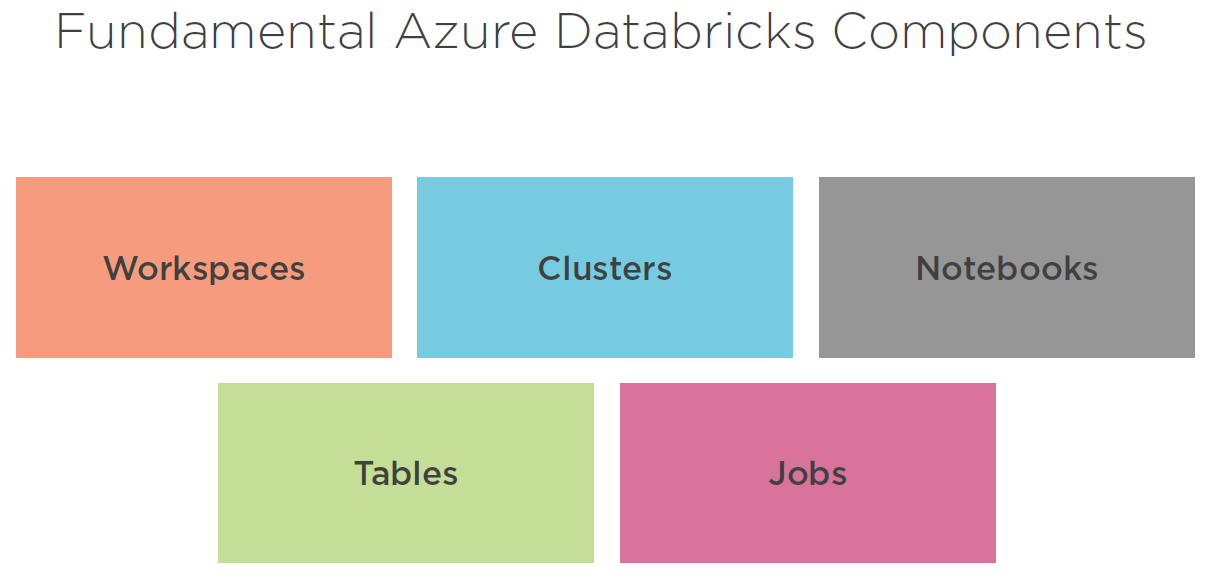


So when we take a look at Azure Databricks, we have a number of use cases. So in the past, pre-canned queries were used in data applications with modifications being few and far between. Today's data scientists and business analyst want to engage with their data in real time. They want to iterate as they're going along, being able to ask new questions, review the results, and see where those results lead. So scalable and flexible platforms, like Azure Databricks, allow for this type of interactive analysis. Because our data comes from many different sources and many different states, bringing it all together requires training and prep so that the data is useable on the other side by our intelligent applications. Extract, transform, and load, or ETL processes, using tools like Azure Databricks, help you to integrate your data and transform it for use by your applications.

The Apache Spark machine learning library in Azure Databricks lets data scientists focus on complex problems and models, not building infrastructure and tooling. And a big challenge for many organizations is dealing with sensors, IoT devices, social networks, and online transactions. Millions, if not billions, of different types of data that need to be monitored constantly and acted upon quickly. As a result, the need for large scale, real time stream processing is greater than ever. All of these use cases and more can find a home inside of Azure Databricks.

## Fundamentals of Azure Databricks

When working with Azure Databricks, you need to understand a few components and their relationship within Azure Databricks.

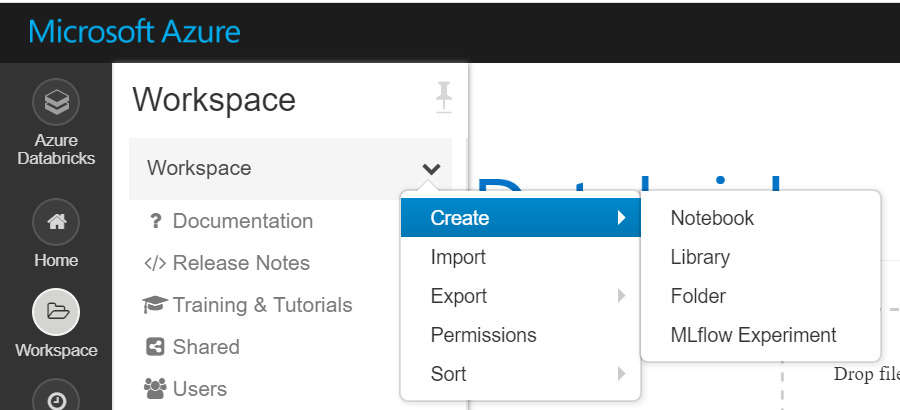


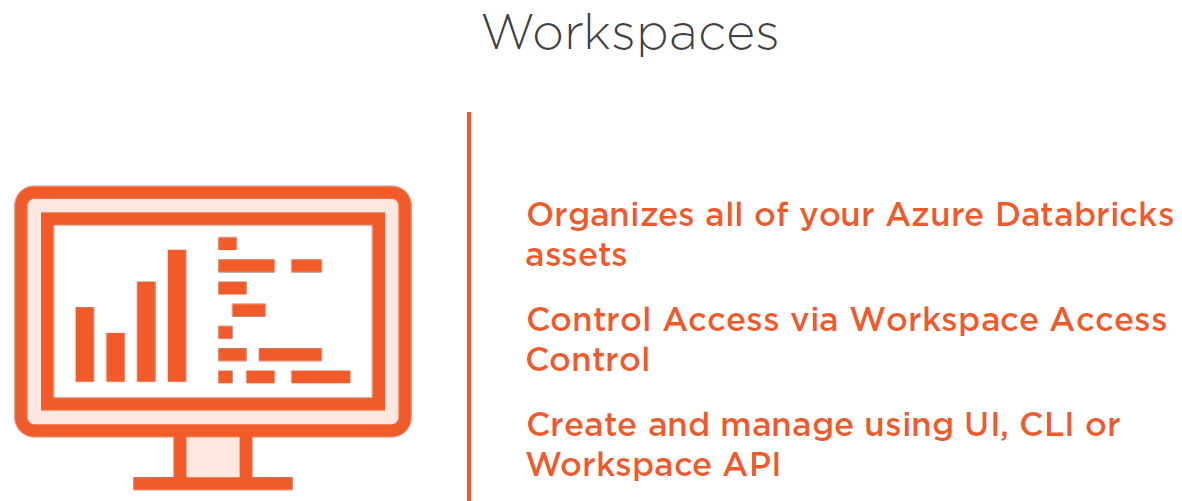
* First are the collaborative workspaces that contain all of your assets.
* Then you have Apache Sparks clusters. These do the heavy lifting of your analysis work, providing you with scalable cluster computing environment.
* Next, we have the notebooks that provide a collaborative space for training and preparing your data and creating your data pipelines.
* We have tables that provide data structures within your workspaces.
* And last are the jobs for scheduling data analysis jobs within your workspaces.

While these are not all of the components that you're going to come across in Azure Databricks, we'll focus on these to get you on the road to implementing Azure Databricks.

Now let's take a look at each of these in more depth, starting with workspaces and folders.

**A workspace** is an environment for assessing all of your Azure Databricks assets.   
It organizes notebooks, libraries, dashboards, and experiments into folders and provides access to data objects and computational resources.   
By default, the workspace and all the contents are available to all users that have access to a workspace.   
But each user has a private home folder that is not shared. You can control the view, edit, and run of objects in the workspace by enabling “workspace access control”.   
And you can create and manage workspaces using the UI, the CLI or the Workspace API.



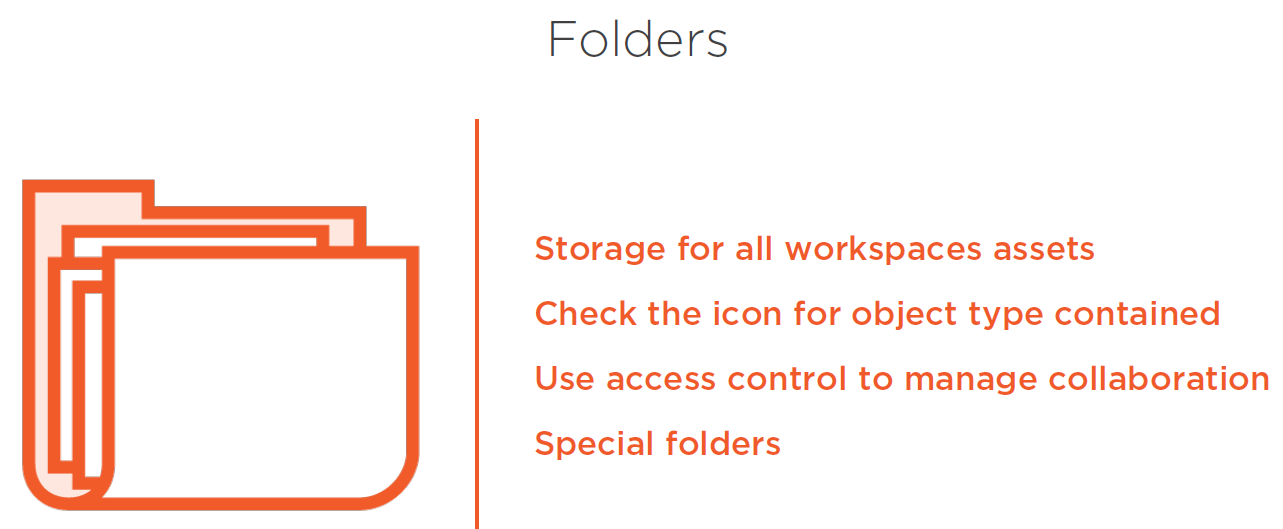


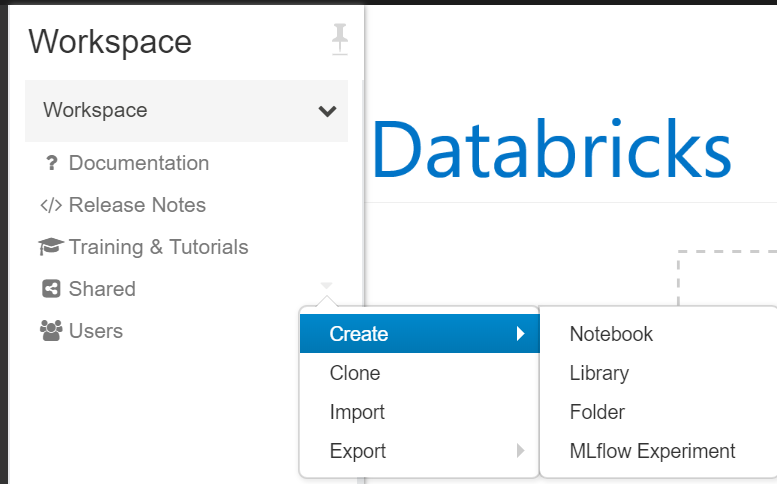
As mentioned, folders hold everything within the workspace. This is just like a folder on your desktop or local computer. Folders can contain notebooks, libraries, experiments, and other folders.   
Icons indicate the type of object that's contained within a folder.   
You can create folders for your use or collaboration with others by setting access control.   
And Azure Databricks workspace has three special folders that it's important for you to know about,

* workspace,
* shared, and
* users.

So the workspace is a special root folder for all of your organization's Azure Databricks assets.   
- Next, the shared folder is for sharing objects across your organization.   
- And then the users folder contains the folder for each registered user. Now let's jump into a demo to create your first workspace.

The users and shared folders are under the workspace folder.

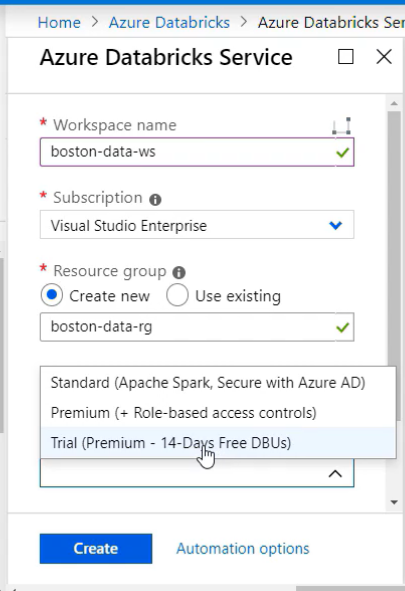




## Creating an Azure Databricks Workspace

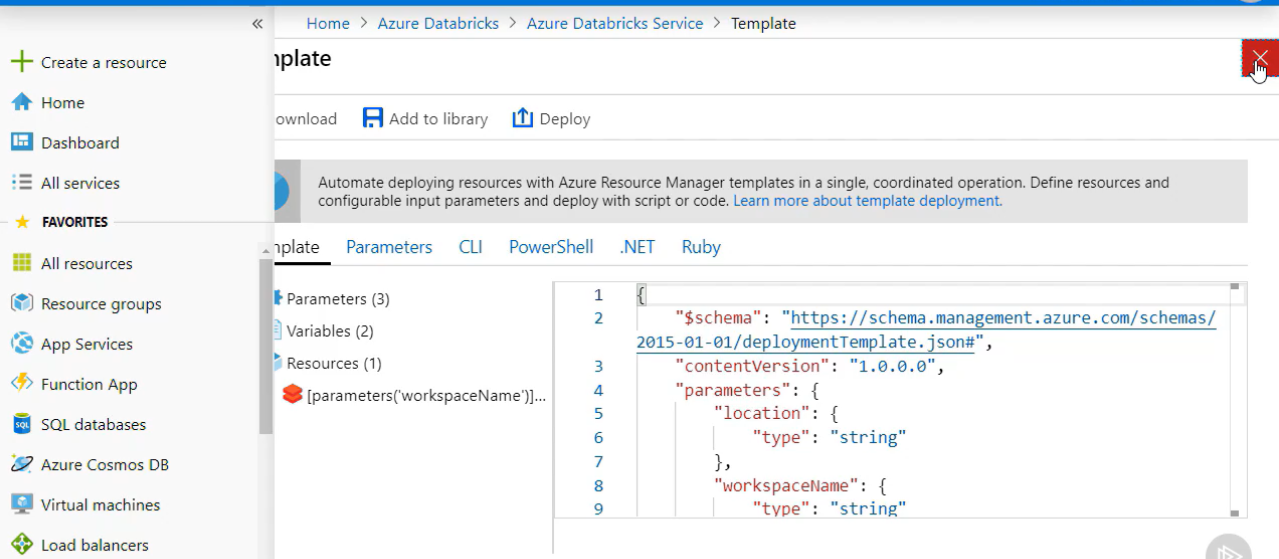
In this demo, you'll learn how to create an Azure Databricks workspace through the Azure portal. After creation, we'll explore the Azure Databricks workspace and get started with working with folders.

So let's jump into the Azure portal. Once we're inside the Azure portal, the easiest way to get to Azure Databricks is simply to go up to the search line and we'll type in databricks, and you'll see that Azure Databricks will come up, and that will bring us to the resource area for Azure Databricks. As you notice here, I've got a number of Databricks installations that I had worked with previously. If I wanted to work with one of those, I could click on them, go into Launch. I'm simply going to go in and add a new workspace into the Azure Databricks service. So we're going to call this one boston-data-ws. I'm going to leave that subscription.

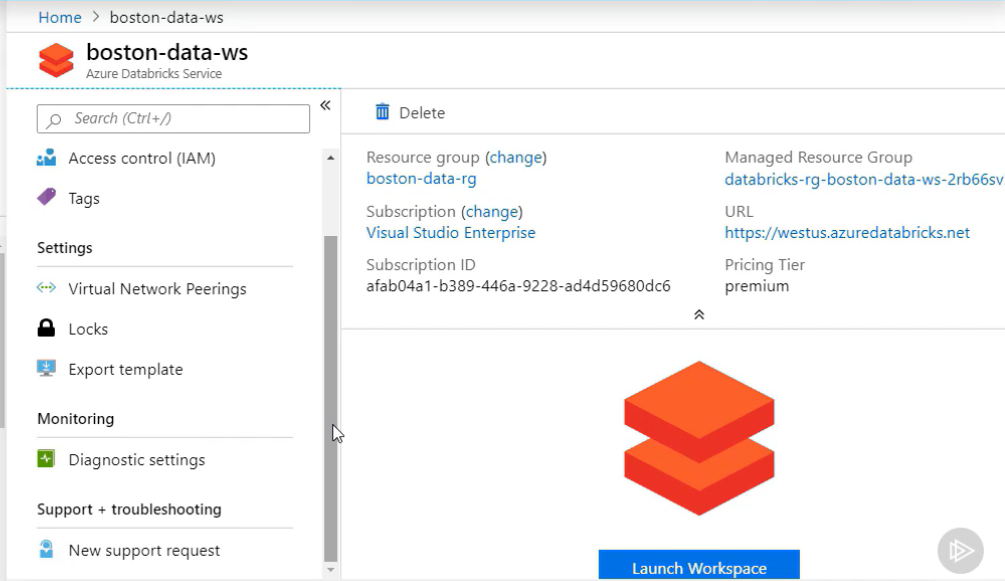


I'm going to create a new resource group for this and I'm going to call it boston-data- resource group, -rg. I'm going to leave the location. And then I got the opportunity for some different pricing tiers. So standard is going to allow you to secure a thing with Azure AD. Premium is going to give you the role-based access control. So this is going to be probably what you're going to want to use in most organizations. So I'm going to go ahead and click on the premium with role-based access control and then in preview you have the ability to deploy this into your own virtual network. We're just going to leave that with the default as no.

Just a quick note here, for some automation pieces here, so if I wanted to create an ARM template, an Azure Resource Manager template, in order to deploy this. I could simply take the JSON code here and use that for my deployment. I could also add it into the library or deploy straight from here.



So notice that when we go back here, validation is successful. So I'm going to go ahead and create our Azure Databricks workspace. This is going to take a bit of time, so when we come back, we should be ready to go into our Azure Databricks workspace. So now we see that our deployment has succeeded. So I'm going to click got to the resource, and that'll bring me into the resource



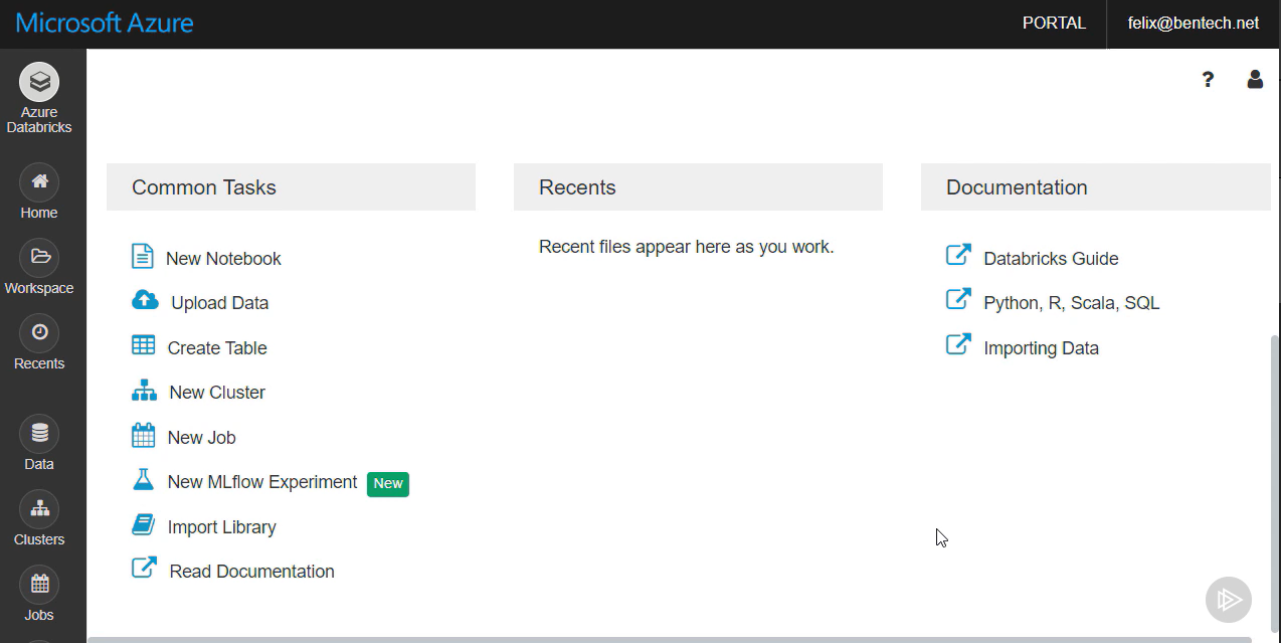
so I can manage different things like access control, tagging, I can export the template, diagnostic settings, support requests can all be done through here through the resource.

So if I go over to this area in here, it shows me what resource group it's in, the subscription, also the URL in which my workspace is located in, which is the westus.azuredatabricks .net.

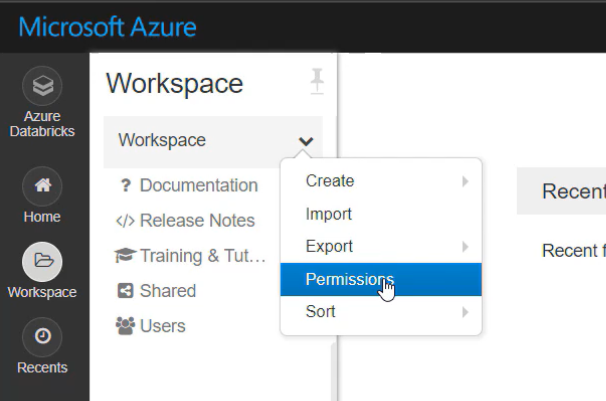
And if I go down here a little bit, you'll see that it has a number of links to documentation, getting started, how to do different things within Azure Databricks. So this is a great place for you to go in and check out how to do different things with Databricks.

So I'm going to go and launch the workspace, and this is going to launch another window, it's going to use single sign on to use the credentials that I was logged into the Azure portal to log into the Azure Databricks workspace that I just created. And again, this might take a few minutes for this to log in the first time, after subsequent times I've found that it usually takes a little bit less time to get you logged in.

Now here we are inside of our workspace in Azure Databricks. So from the landing page here, we see we have a number of different things we can do. It provides a lot of different information about how to get started with Azure Databricks. It has a common tasks area with all the common things that you would do within Databricks, also a documentation area.

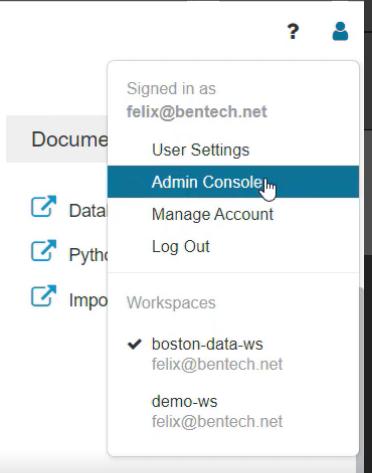


On the left side, you see you have callouts for workspace, recently used assets, data, clusters, jobs, different things that you can work with. So if I go and I click on my workspace here, it gives me my workspace dropdown. Notice I've got my shared and my users folder and then my workspace is actually that special workspace folder that we talked about previously. From here, if I do a left-click, it'll allow me go in and create different things. I can import and export. I can work with the permissions for this workspace.

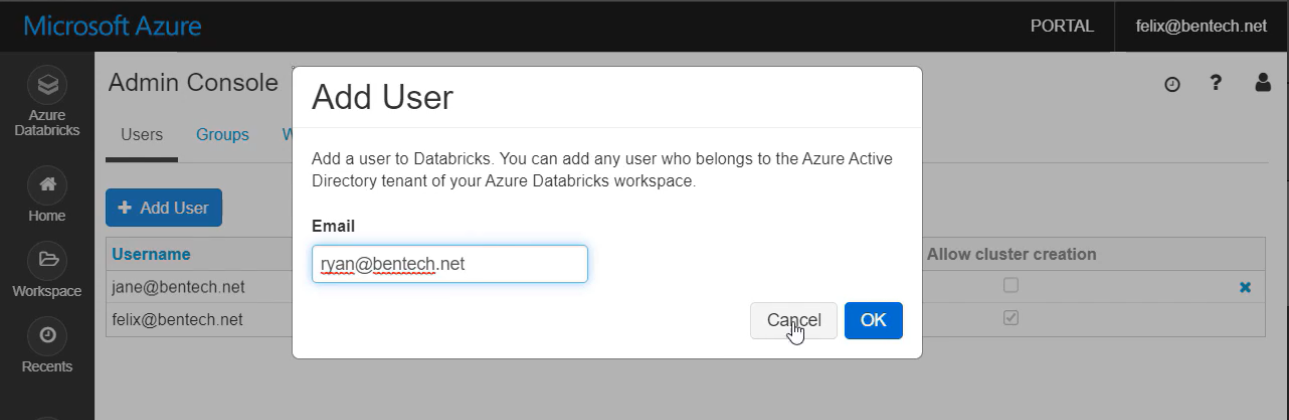


So if I want users to be able to do anything within my workspace, I need to give them permissions at this level.

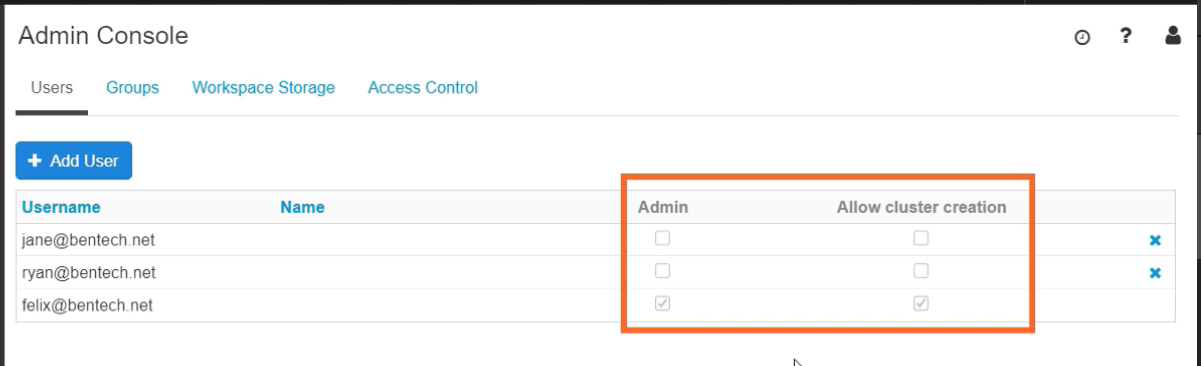
So I'm going to go over to my account icon here, and I'm going to go down to the Admin Console.



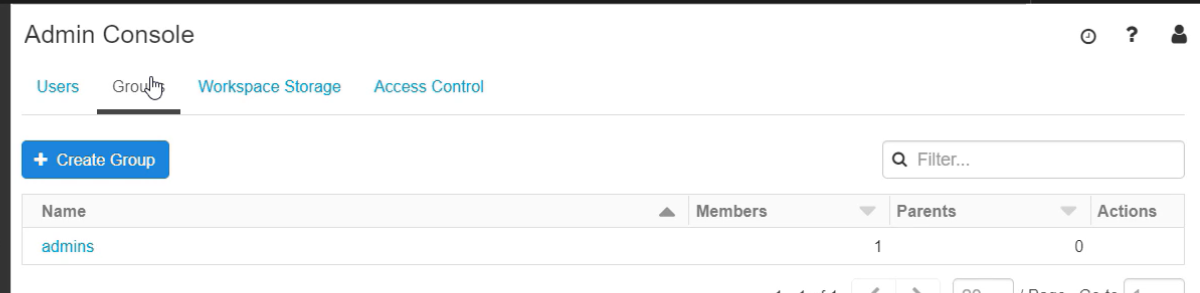
What this will allow me to do is this will allow me to add different users in here. So I'm going to add two users in here, jane@ bentech.net, and we're going add ryan@ bentech.net.



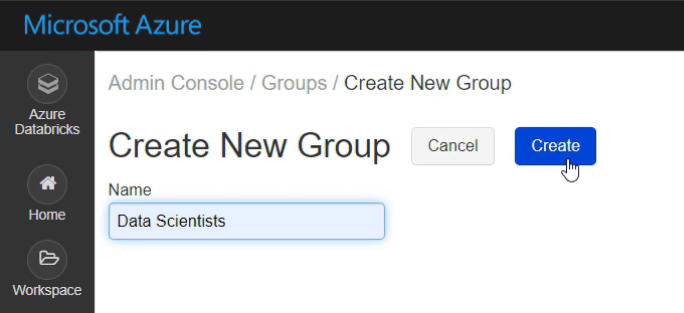
These are different users within our workspace. Notice in here, as well, this tells you whether they're an admin and whether they're allowed to do cluster creation.



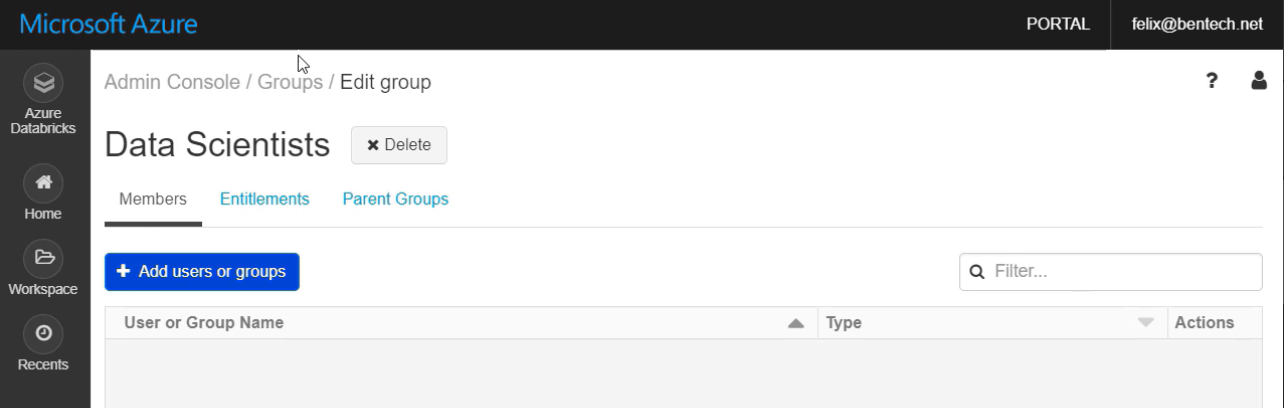
I can also go in here and I can create groups.



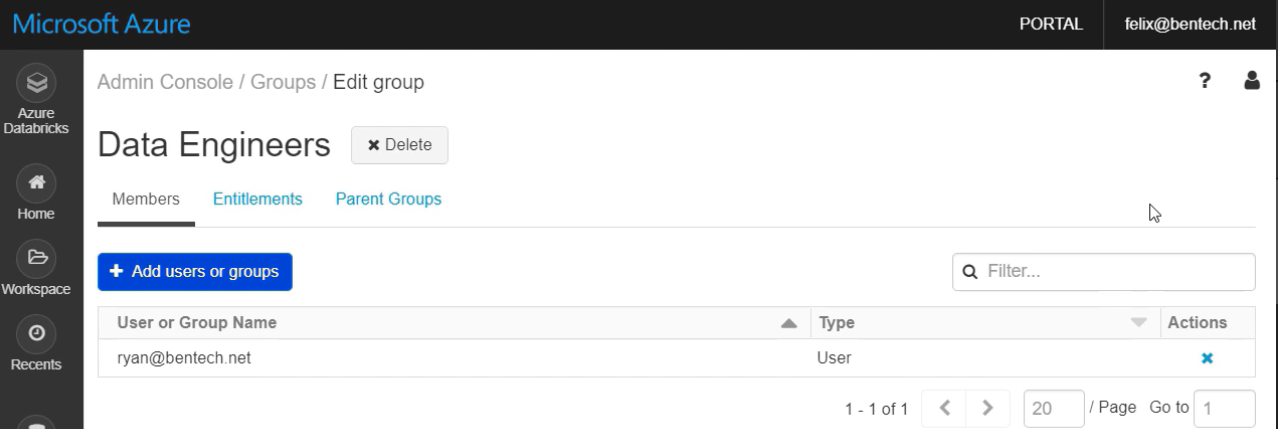
So you can use groups with your access control to give specific users access to specific things. So let's say I wanted to create a group for data scientists.



Go ahead and add that in here. And at this point, I could add users directly into this.

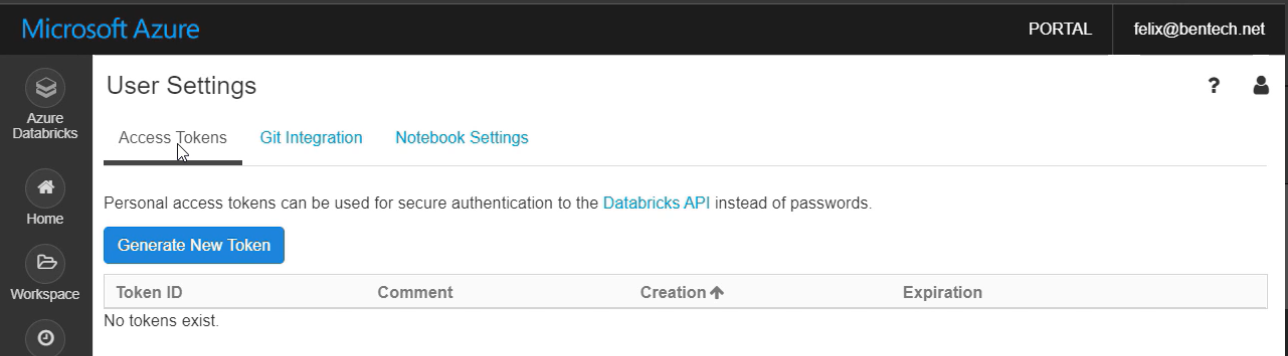


I'm not going to add a user in at this time, I'm going to go back, I'm going to create a data engineers. We'll go ahead and click and create that. And in the data engineers, we're going to go ahead and add Ryan in as a data engineer.



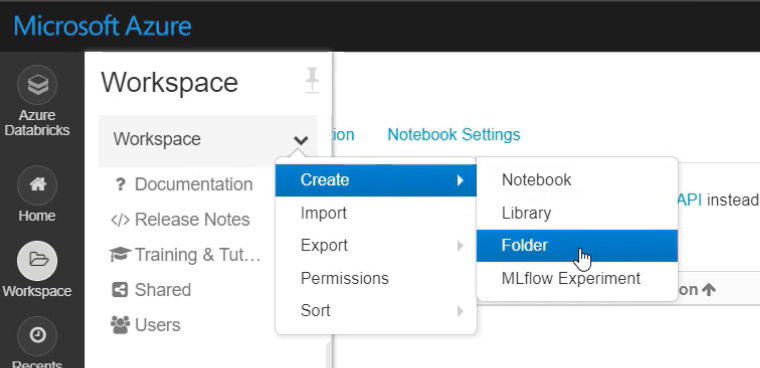
So now I could use the data engineers group in order to set permissions throughout all of the different components and assets within Azure Databricks.

I can also go up here and I can modify different user settings. So can we can work with Git Integration, Notebook Settings, and something we're going to work with in the CLI is generating a token for our user to be able to access this with the Databricks CLI.

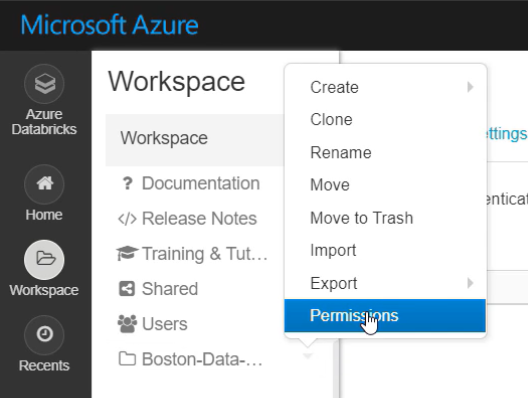


So that gives you a little tour of the Azure Databricks workspace.

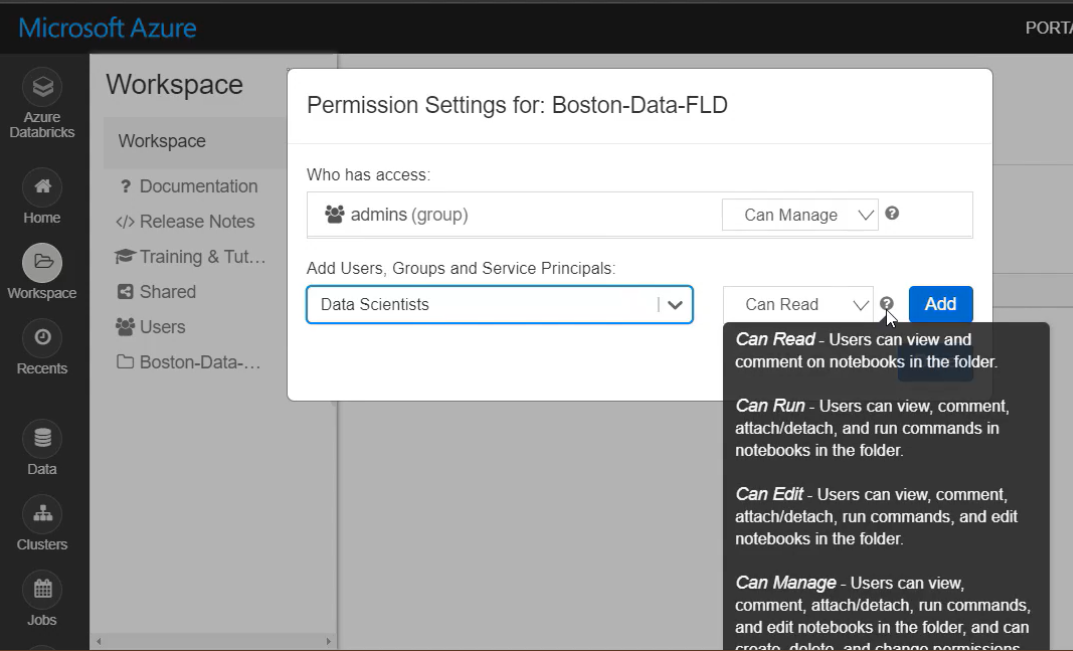
Now let's create a folder in our workspace. So click on Workspace, click the down arrow, and we're going to go create, folder.



We're going to call this Boston-Data-FLD. And now what I want to do is I want to modify the permissions here.



So if I go to the permissions, we see we have the admins group. From here, I can choose the Data Scientists, and from here we've got some options.



We can read, can run, can edit, can manage. So read is going to give you the ability to view and comment on the notebooks. Run is going to allow them to do everything with read, along with be able to run the notebooks if you need that. So the edit phase, that's going to allow you to do everything read and run, but also edit the notebook as well. And this is the type of permission that most data scientists are going to need. The can manage, that gets into permission sets and chances are you only want to have your data engineers with that and people that are managing the underlying infrastructure.

So once we have that set, we'll go click and add. We'll click Save Changes. And that's how you go through the process for creating a new folder that you can put assets into and setting the permissions.

## Getting Started with the Databricks CLI

For many people, working from the command line is easier, as well as provides better mechanisms for automation and repeatability. In this demo, you'll learn how the Databricks CLI can be used for managing your Azure Databricks implementations. I'll be using Azure Cloud Shell, a web-based CLI for accessing Azure. You can also set up the Databricks CLI locally, and a link to that set up is going to be in the course notes. So let's jump in and take a look at the Databricks CLI. So in order to get using Azure Cloud Shell and Databricks CLI, what I need to do is I need to generate a token for my user to be able to access my workspace using the Databricks CLI. So I'm going to put a comment. I'm just going to call it cli. I'm going to set this to 30 days. We're going to go ahead and generate that, and I'm going to go ahead and copy that code. This is the only time you're going to see this, so you need to use this right away in order to get working with it. Otherwise you'll have to come back in and create a new one. So I'm going to go over to my other tab here, and that's at shell.azure .com, so that's loaded me into my Cloud Shell. So once I'm connected in with my Azure Cloud Shell using bash, at this point what I want to do is I want to create a virtual environment for my Databricks CLI to live in. And basically that's just going to be a little environment. I'm going to go ahead and create, copy the code in here. I'm putting in the virtual environment -p to the path of where I want this to go to, and we're calling it databrickscli. So when we go ahead and run that, That'll take a couple moments to run. That'll create a virtual environment for us to work with so that we can install the Databricks CLI, it'll be running Python 2.7. So once that's all done, I want to enter that virtual environment, so I do that by using the source command, source databricks/bin/activate as location for activating that virtual environment. So you'll notice that the prompt changes here, so in parentheses it tells me that I'm in the databrickscli location. And at this point I want to install Databricks CLI into my virtual environment. So we're just going to do a pip install databricks-cli. And that'll go through the process for installing that onto my system. So once that's all installed on my system, the next thing what I need to do is I need to configure this virtual environment to go to the workspace that I just created. So we're going to put in databricks configure --token. After a couple moments, it's going to ask me for the Databricks host. If we go back to our workspace here, we see the URL, simply copy that over that. Go ahead and paste that in here. And then it's going to ask us for that token. So that token that you had copied previously, go ahead and paste that in, and there we go. So let's simply do a test here. We'll do a databricks -h to bring up the help for Databricks, and we see it brings up the help for all of the Databricks commands that we have available. If I put in databricks workspace -h, you see it brings the help for the Databricks workspace command. If I do a databricks workspace list, you'll see that it lists the workspaces available. Because I'm inside that workspace that I created, all I have available to me here are the users and the shared. And that's how you get started with the Databricks CLI and get that installed. We'll see more of this throughout the rest of the course.

## Azure Spark Clusters

Azure Databricks clusters provide a unified cluster computing platform for various use cases, such as running production ETL pipelines, streaming analytics, ad hoc analytics, and machine learning. Azure Databricks has two types of clusters, interactive and job. You use interactive clusters to analyze data collaboratively with interactive notebooks. You use job clusters to run fast and robust automated jobs that are scheduled.

With Azure Databricks, you can easily create and manage your clusters, again, using the UI, the CLI, or by invoking the cluster's API.

In this demo, you'll learn how to create and manage Spark clusters, both through the UI and via Databricks CLI. So let's jump into our workspace and start working with clusters. Back in our Azure Databricks workspace, let's walk through the process for adding a new cluster into here. So we can do this a couple different ways, I can go down to the Common Tasks, I can click on New Cluster, or I can go over to the Clusters icon and that'll bring me into the same area. So at this point, I want to click on Create. I'm going to give this a name, I'm going to call it boston-data-cl01, and then I can have a couple of modes, high concurrency or standard. High concurrency is going to be optimized for SQL, Python, and R workloads. It's going to be for when you have multiple users, and it was previously known as serverless. We're going to go with the standard one that's recommended for standard single user clusters. If I wanted to create a pool or had created a pool of ready to go clusters, I could choose from that. I choose my Databricks runtime. You'll notice from the list, it shows about 6 here, there's 18 others, it'll give you different runtimes that have GPU, ones that are designed for machine learning, different versions of Scala, Spark, a number of different things for you to choose one. I'm going to take the basic runtime 5.4 that was selected for me. I'm going to take version 3 of Python, you can do 2, but that is deprecated. I want to leave enable autoscaling because that's going to automatically scale the minimum and maximum number of nodes based on loads. So this is designed to really take the need for determining infrastructure out of the hands of your data engineers and really allow the cloud to be able to determine that based on your jobs and your needs. And then I can terminate this after a certain number of minutes of inactivity, because you're going to be charged for the amount of time when your clusters are running, so you only want them running when you're actually doing workloads. So in this case, I'm going to bring this one down to 60 minutes and then it's going to choose the worker type, and I've got a number of different virtual machine types that I can choose with here. I'm going to go with the basic. I'm going to leave it with 2 minimum workers, 8 max workers. Notice that I have a little warning here, it's going to say I might not have enough CPUs. The reason I'm showing you this is this is commonly going to come up in most accounts that you have inside of Azure. So what you need to do is you need to go over to this article and you can go and open on an online request here and this article will take you through that process for requesting more resources in Azure. But we're just going to go with what we have. We don't have a lot of load running. And then the type of driver that we want, we're going to go with that as being as the same as a worker. We've got a number of advanced options. Choose our spark config, we can add tags, logging if we desire, and then the initialization scripts if we have any of those going to our database file system, or DBFS, location if we had an initialization script. I'm going to go ahead and create that cluster and that'll take a couple moments for that cluster to be created. So you notice that this states as pending. I've got a number of actions that I can perform from here. I could terminate, I could restart, I could also clone this, and I could create a new cluster based off of that. So that's really quick and easy for creating a cluster through the UI. Now let's take a look at creating a cluster through Databricks CLI. So over in Cloud Shell, the first thing I need to do is I need to get back into that virtual environment. So I'm going to go ahead and copy in my command, source databricks, and notice it goes into the proper prompt. I'm going to take a look at a JSON file that we're going to use to build out our cluster. And I'm going to open that up in Visual Studio Code or Code, which is based on the Monaco Editor inside of Azure Cloud Shell. So I simply go to the path for my JSON file. You could do this locally if you're running VS Code locally. Notice it has the JSON file with all of the settings, just like if I would set them in the UI, it allows me to do it through Code. So if I wanted to move this up to 8, if I wanted to modify the name or the Spark version, any of the settings that we have, I'll go ahead and change the minutes to 60. So that gives us some changes for creating our cluster. So now we're going to put in the command that's going to allow us to build that cluster out. So I'm going to be using databricks clusters create using the --json-file parameter, going to that JSON file that we just took a look at. When I hit Enter, it'll take a few moments, and that'll build the cluster for us. And that's real easy and simple, how you go through the process for building out a cluster inside of Databricks using Databricks CLI. So now if I want to see the clusters that are available, I can put in databricks clusters list. And then it will list, we see we have one running, and we see that second one we just created pending. So great tool, allow you to be able to quickly and easily create stuff, be able to see what's going on, and to be able to work with your clusters.

Notebooks

A notebook is a web-based interface to a document that contains runnable code, visualization, and narrative text that are all organized in cells in the notebook. When you use a notebook, you're primarily developing and running these cells as you iterate through your results. Notebooks are the most commonly used platform for interfacing with Azure Databricks. Often production ready notebooks will be run as a job once tweaking of code is complete. Notebooks will be the foundation for building pipelines within Azure Databricks, allowing you to gather, clean and prep, and load data all the way through. Just like other Azure Databricks assets, you can manage the notebooks using the UI, CLI, and by invoking the workspace API, and Databricks supports several notebook external formats based on Python, Scala, markdown, and HTML. So let's take a look at some demos with notebooks. In this demo, you'll dive into notebooks in Azure Databricks. You'll learn to create a notebook, add various elements, like code and text, into the notebook. We'll also cover sharing, import and export, and permissions as well. So let's dive into notebooks. So what I'm going to do is I'm going to import a notebook, so you can import notebooks from a number of different formats, and this is a great way to take notebooks that other people have used. In this case, it's going to give us the ability to use a notebook that I've already created that has a number of cells created that we can walk through. So here we are inside of a preconfigured notebook to show you some of the different cells that you can build inside of a notebook. You'll notice that this is a Python notebook because it has Python next to the title. The first thing we want to do since we didn't attach a cluster at the time of the creation of this and because we imported it so it doesn't have a cluster, what we want to do is we want to find a cluster and connect it to that. So I'm going to go ahead and add that to the boston-data-cl01 cluster, which it looks like it's about starting up. Then we'll take a look at the cells here. So notice I just have some cells here, if I double-click, notice that we're using %md and that clarifies what type of cell this is. So if I want to use other code besides Python in here, I need to put percent and then what I'm using, so %scala, %sql, that'll allow you to be able to use those. Percent HTML will work as well. So I'm using basic markdown to create a title here. And you see, we're using basic markdown here to simply create a link going out. Then we get into our actual code blocks in the cells. We see we have one for setting our storage, so it's using Python code to go and set our access information, and then it's going to read that information in from blob storage. It's going to create a temporary data frame based on that data, and it's going to display out the last 10 rows. So if I go to the top here, I can run each one of these individually. Notice it's showing a message that it's sending to the cluster. So depending on what it's doing, it'll take a little bit of time or work. And notice that it completed there. Now if I want to run everything here, I can simply go up and run all of these. And notice it's showing you that it's going to each of the different commands. So if I wanted to go to the command that it's working on, it'll show me which of those commands that it's working on. So once that completes, you notice that it's running a sparks job there, it will display for me the top 10 rows from this query. So now we see, there's our query of the information based on what we got. If we want to see that in a different format, if I want to see that in, say, a pie chart, I go to plot options, we're going to leave it at source and longitude, you see it shows a nice value for that. And then I could change this around to different options, but the pie works out best for the one that we're looking at here. So let's say you're a data scientist and you're like, you know, I need more information here. I can simply go right into the code here and I simply click run the cell, and it will run that cell and run that job based on the changes that I made into there. Since all of the information above was the same, I don't need to make that change. So you see that change a little bit the percentages. And so that's the basics of working with a notebook. And the next thing we're going to take a look at is working with some tables.

Azure Databricks Tables

An Azure Databricks table is a collection of structured data. Tables are equivalent to Apache Spark data frames. In Spark, a data frame is a distributed collection of data organized into name columns. It's conceptually the equivalent to a table in a relational database, or a data frame in R or Python, but it comes with much richer optimizations under the hood. This means you can cache, filter, and perform any operations supported by data frames on your tables. You can query your tables with Spark APIs and Spark SQLs. You can also create, read, write, create unions between tables, and other operations that are common within your data structures. Often the tables are used with a cluster as a temporary data structure during the data transformation process before the data is loaded into the final database location. So let's take a look at working with tables. So in this demo, you're going to learn how to work with tables in Azure Databricks. And I'm going to do that by importing data into an Azure Databricks table and then performing some basic queries against the table. So first thing I'm going to do here is I'm going to import into my folder here, I'm going to bring in a notebook, and what this notebook is going to do is it's going to go through some basic table commands. So what this is going to allow me to do, and I'm using some publicly available datasets from Azuredatabricks.NET, I'm going to go ahead and attach this to the cluster that I was using before. So it's going to display some datasets that I have available, it's going to then allow me to read the review files, and these are simply for if you want to take a look at the datasets that we're working with. Next we're going to get into the process of creating a Spark table and what this table is going to contain, it's going to contain information about diamonds and then it's going to bring that in from a CSV file. So notice because I am using a Python notebook, I need to call out that I'm going to put in SQL code here. And I'm simply using a drop table to drop a table if it exists, and I'm creating a new table called diamonds using the CSV and the path to that CSV file in the blob storage. And then I'm just simply doing a SELECT so we can see all of the information in there. So we'll go ahead and run that. It's going to send that to the cluster, it's going to run, wait for that, run the command. Once that's completed, once the job's completed, then I'll have all of my information available. So notice now it shows me, there's my table. And if I simply wanted to do a simple query against that, notice I'm using SQL SELECT statement, I'm looking for the diamond cut and the average price based on the diamond cut there. And I can simply run the cell and that will bring back the results of the different cuts of diamonds and the average price for those. Now if we want to actually see the data inside our workspace, I can click over on Data. Notice now I have a diamonds table here, and it shows me all of the different column names, the types of data, and if there's any comments, and then it also shows me the sample data. So that's quick and easy, showing you how to use notebooks, how to work with the different cells, and then also how to work with tables.

Apache Spark Jobs

In Azure Databricks a job is a way of running a notebook or working with a Java archive or jar immediately or on a scheduled basis. You can create and run jobs using the UI, CLI, or invoking the job's API, just like you can with the other components inside of Azure Databricks. Similarly, you can monitor job run results in the UI, using the CLI and the API as well. One thing to note is that the number of jobs in a workspace is limited to 1000, and the number of actively concurrent runs a workspace is created is limited to 1500. This should meet the needs of most organizations, but it's something you want to keep an eye on if you're performing lots of run work within a workspace. Now let's see how we can use Azure Databricks. In this demo, you'll learn how to work with interactive and scheduled jobs using various tools like the UI and the Databricks CLI. Creating a job in the Azure Databricks workspace is actually pretty easy. We simply just go down to Jobs and I click on Jobs and I can click Create Job. So what I'm going to do is I'm going to create a job and we're just going to call it Notebook-job, and I'm going to do it based on one of the notebooks that I had created. So we're going to do it based on that Boston-Data-Notebook. And notice it'll allow you to be able to edit the parameters as well. So if you're running something that has parameters that need to go in, you can add in those key values or you can add in JSON to go into there. We're not going to make any edits to those. I want to run this on a schedule. So I want this to run every day and I want it to run at 2:00 am Pacific time. And then I'll go ahead and confirm that and that'll set the time for me to run. If we take a look at the cluster, it's going to pick up the standard to run with. I can edit this as well to add in the type of Databricks runtime, the Python version, the worker type, all things, when we created clusters before we did these. Notice up here that it gives you the ability to do a new cluster or existing cluster. The recommendation that comes from Apache Spark is to use a new cluster for your production level jobs or the jobs that are important to complete. Where existing jobs come in, those are going to be most of the time the background jobs that run for things like your dashboards that you need to run something at a regular interval. So in this case, we're going to simply just use a new cluster here. And I'll click Confirm there. And then notice here I can do a Run Now if I want to run this now, and we'll go ahead and kick that off and that'll run the job and then this will come back and tell me the status and I can see all of the information about my job from here. So now let's say we want to do that through the Databricks CLI. I'm over in Azure Cloud Shell. I'm into my virtual environment that has my Databricks CLI loaded into it. So the first thing that I'm going to do here is I want to see what commands are available to run with Databricks CLI with the job CLI. So if I go ahead and put in databricks jobs -help to call the help command, it shows at the bottom, I have a number of different commands available for me. Create, delete, get, list, reset, run-now, so a number of commands that we can use. So if I want to see what jobs are available, I'm going to use the list command. so we'll go ahead and put in databricks jobs list and it lists all of the jobs. So we see we have that notebook job that we had created through the portal, and that's id number 4. So if I want the specifics about that job, I can use the get command. So we'll go ahead and put our command in here, databricks jobs get, we'll put the job-id of 4 in, and it'll bring back all of the JSON information. So what's nice about this is you can actually take this JSON, you could put it into a file of your own and tweak it and use that to be able to create new jobs through the CLI calling that JSON file. And that's what we're going to do right now is we're going to take a look a pre-created file that I had that's part of the exercise downloads, and we're going to open that up in code. So you see, here's what a JSON file looks like that's used to build out a job. So it's got a name and everything that we would set up through the GUI. So we'll go ahead and close that out. Now we're going to go ahead and we're going to use the create command to be able to set this up. We'll do databricks jobs create --json-file, calling the JSON file that I have. We'll go ahead and run that, and we see that that gives us a job id of 7. So if I want to run that now, I can simply use the run-now command, and we'll go ahead and put that in, and with an id of 7, and that'll run the job for us. And so that's a little bit of how to work with the job CLI and Databricks CLI to manage your jobs and also how to manage your jobs through the UI in Azure Databricks.

Summary

And there we go, your introduction to Azure Databricks. You should have learned how easy it is to spin up Azure Databricks, an Apache Spark-based cluster, computing, and analytics platform in Azure. And to begin with their components like clusters, notebooks, jobs, and more as you work to wrangle your data needs. As you move forward, remember to use the auto-scaling option to let Azure Databricks dynamically scale your clusters based on the needs of your jobs. Notebooks are a key tool for you and your data users and consumers to train and prep data in your data pipeline. Use your favorite language, R, Scala, Python, whatever you need to get the process moving. And jobs are a great --- If you need further learning resources or information about the topics covered in this module, check out both sets of documentation on Azure Databricks at docs.microsoft .com and docs.azuredatabricks .net. They're great resources from the developers of the platform and will allow you to go deeper on any topics related to Azure Databricks. You'll also find that curated list of specific links I think you'll find helpful in the module exercise files on Pluralsight.com. Look for the read me file for more information. And if you have any questions about the course, please join the discussion on the discussion board for the course at Pluralsight.com. And now that you have the fundamentals of Azure Databricks, it's time to move on to the next module, Performing ETL, (Extract, Transform, Load) Operations with Azure Databricks where you'll learn how Azure Databricks works into and works with your ETL operations. Thanks for watching, and I'll see you in the next module.

Performing ETL (Extract, Transform, Load) Operations with Azure Databricks

Overview

A common problem that organizations face is how to work with all of the different sources and formats of data that they have coming in every single day. Often where the data meets to reside for an application is not the same as where it's collected from, and along with data being in different formats, it often needs refining and shaping before it is stored for use by consuming applications. This process of moving and refining data in the data pipeline is commonly known as extract, transform, and load, or ETL. And that's what we'll cover in this module, performing ETL, extract, transform, load operations, with Azure Databricks. In this module, you'll learn the basics of ETL. Then we'll take a look at how we use those within Azure Databricks. This will be a hands on walk through of extracting data from Azure Data Lake Storage using Azure Databricks to ingest and transform the data, and then finish off loading the data into an Azure SQL Data Warehouse. Before we get started, I wanted to point out that this, and every module, has exercise files you can download from Pluralsight.com. This download will include a PDF copy of the slides, any code or file assets used during the course for demos, and links to resources for more information about the topics covered. Specific to this module, you'll find an Azure Databricks notebook with all of the code needed for you to follow along and perform the demo that I'm going to be going through. So you simply click on Exercise Files and download the exercise files to your local system.

Basics of Extract, Transform, and Load (ETL) Process

Extract, transform, and load, or ETL, is a data pipeline used to collect data from various sources. Transform that data according to business rules and load it into a destination data store. This makes way for your intelligent applications to access the transform data they need in the format that they need it. In the ETL model, data is stored raw in different types of storage. This can be Azure Blob Storage, Data Lake, Hadoop, just to name a few different types of storage. Since you don't want to change your raw data and it's often more data than needed for the end consumer applications, you're going to extract it from its stores for the transform process. The process of extracting comes out of notebook calls requesting the data that they need. This allows the process to be scheduled and interactive, and because Azure Databricks lives in Azure, you have easy access to all of your storage native and securely. Once the data is extracted, transformation can begin. Transformation in Azure Databricks involves processing your raw data into predictions and insights. A transformation activity executes in the Azure Databricks Apache clusters driven by Azure Databricks notebooks and jobs, often data is temporarily stored in tables for use during the ETL process. I like to think of this part as where you refine your data down to what you need for your intelligent applications and analysis dashboard. The data transformation that takes place usually involves various operations, such as filtering and sorting, joining data together, cleaning, removing duplicates, and making sure that your data is valid. And you can use your language of choice, Python, R, Scala, SQL, whatever you need to use, you can use for this transformation process. Once the data is transformed, it needs to be placed into a database service for use by the consuming applications. Options include Azure SQL Data Warehouse, Azure SQL, Cosmos DB, and more. You choose the database service that meets your needs, as many different options are available. From here, consumers can use the data as they need it for those applications in dashboards like Power BI. Because the data has been transformed based on the needs of the consuming applications, the databases will have all the data you need and little that you don't, and what if you don't get all of the data you need? Then you just rerun the process by tweaking your notebooks and jobs to extract, transform, and load what you do need. Pretty simple and straightforward.

Scenario: Working with Audience Information

Now let's take a look at ETL in action. In our scenario, we're dealing with radial list or information that's been gathered into a JSON file. This data needs to be placed into an Azure storage account using Azure Data Lake, so it can be used on our ETL process. Once the data is available, Azure Databricks notebooks are created in Azure Databricks, which interact with the data store to extract the listener information. Notebooks will then perform action to refine the data down to the desired state for use by the consuming applications. Once the data is transformed, it will be placed into an Azure SQL Data Warehouse for final use by the consuming applications. In order to get this scenario up and running, there are a number of prerequisites that need to be in place for the ETL process. Basically, what we need to have in place are the things that you would have in your environment when you bring Azure Databricks in. Things like your data stores where your initial raw data is held and where it's going to be held once it goes through the transformation process. You have things like service principles. The demo that we're going to be working on is based on a tutorial at the URL that you see below, if you want to review it later, or if you want to perform it on your own. Also, I've included the links to all of the prerequisite documentation in the module exercise files. And one last thing that you're going to see when we get into the demo, I provide an Azure Databricks notebook that includes all of the commands that are done in the tutorial, so you can simply import this notebook into your Databricks workspace and begin working with the demo.

Demo - Ingesting and Extracting Data in Azure Databricks

So in this demo, we're going to be performing the ETL operation in Azure Databricks for that audience information. So taking it from the point of it being a JSON file, putting it into our Data Lake Storage, extracting it, going through a transformation process on that, and putting it in its final destination inside an Azure SQL Warehouse. So let's get to our demo. Before we dive into our demo, I wanted to go out to docs.microsoft .com and take a look at the tutorial that we're going to be following. So this is a demo that will walk you through the entire ETL process using Azure Databricks. So you can get to it from the URL here or else the shortened URL, bit.ly /ETLdemo that you see on the screen. So as we scroll down here, what we're going to see is you're going to see the layout for audience data, we're going to be storing it into an Azure Data Lake Storage Gen2. We're going to then be prepping and training it with Azure Databricks, and then we're going to model and serve it with our Azure SQL Data Warehouse. As we talked about, there's going to be a number of prerequisites that you'd normally find in your environment, such as the Azure SQL Data Warehouse and the service principles and the blob storage. So we've got some links here that you can go out to if you don't have those or if you want to follow along with the demo. So then we go down and we'll see that the information that you're going to need going into the demo. And this is basically going to include all of the information for all of the existing prerequisite resources that we have for the environment. So once you're ready to get going, we'd simply step into the Azure portal. So if I go over here to the Azure portal, you see I have a resource group created here called audience-data-rg. And you'll see, inside here I've already created an Azure Databricks workspace called audienceData-ws. We've got a couple of storage accounts for our blob storage and also our Data Lakes Storage. We have an Azure SQL Server instance and then we have our audience data warehouse created as well. So to get started, I'm just simply going to open up my Azure Databricks workspace. We'll go ahead and launch that, and that's going to go through the single sign on process that we've seen before and go into our workspace, and then we'll be able to get ready to get going with our demonstration. Once I'm in my Azure Databricks workspace, the first thing I want to do is I want to go and I want to create a cluster that's going to be used by the notebook. So I'll go in and create a cluster and we'll call this audience-data-cl01, and we'll just leave all of the defaults and we'll just simply create that cluster. And next, what we're going to do is we're going to go into our workspace and we want to add in a couple of notebooks that are going to be used. So I've created a couple notebooks that are part of the exercise files that you can bring in and use for a demo. So we'll go ahead and add those in. So I'm going to go ahead and import that. I'm going to browse to the location where my files are located at. First I'm going to bring in the audience-notebook, and we'll click Import, and then it'll go through the process for importing the audience-notebook, which opens up. And now I'm going to go in and add in another notebook. So we'll go to, again we'll browse the same location. In this case, I'm taking the clean-audience-notebook, now that that's imported. So the clean-audience-notebook is one that you can basically start off with. Notice here the important note, you need to have the prerequisites. So the big difference between the clean-audience-notebook is that I've taken the code that is in the tutorial out at docs.microsoft .com, and then I've put it into a notebook for you, and if you'll notice on lines 3 through 7 here that it's creating values for storage account, appID, password, fileSystem, and tenantID. And then you'll see that it doesn't actually have the real information for those, it simply has placeholders. So you'll want to make sure to replace all of the placeholders. And you'll see, like you see on line 2, I always have a remark for where you're going to be replacing values in your notebook. So we're going to go ahead and go back to our audience notebook. So as you know, with a notebook, in order to be able to use the notebook in Azure Databricks, it needs to be attached to a cluster that we have. I'm going to go ahead and attach this to the cluster that I created. This is where we're going to get to work with the process for being able to use a notebook in Azure Databricks to do our ETL process. Again, remember we need to have those prerequisites in place before all of this will work. So the first thing what we're going to do is we're going to mount a file system in Azure Data Lake Storage Gen2 account. And what we're going to do is we're going to use an account configuration, because this is going to provide us the storage name that we need later going on. So there's two different configurations you can use from the tutorial, the account configuration that's going to work best for you. So now if we go down into that code cell, what we see in the code cell is we see that we have values that are being created. And you notice this is different than the clean version that I had shown you previously. This includes all of the information that I've copied over from the prerequisite creation process. So these are the pieces of information, in lines 4 through 8, that you're going to add in from your environment. And then the rest of the code, what it's basically doing, is it's setting up the configuration with that account and then on line 16 we're actually using dbutils to make the connection and mount that and make it available to our Spark clusters to work with. So I'm going to go ahead and run this so that it loads all of these values and makes those connections for us. Now that that's completed running, we can move on to our next demo, which is going to be starting the process of ETL, and we're going to start ingesting data.

Demo - Transforming Data in Azure Databricks

So in this part of the notebook, we're going to go through and ingest the sample data into our Azure Data Lakes Storage Gen2 account to create that raw storage that's going to be used by the ETL process. So this is going to involve downloading a JSON file from GitHub and putting it into a temporary directory, and then we're going to copy that JSON file into our Azure Data Lakes Storage. So the first command we have is we're using a shell command and we're using wget to bring that JSON file from GitHub into a temporary storage location. So we'll go ahead and run that. So the next command is we're going to copy that temporary JSON file to the Azure Data Lakes Storage location, and we're using dbutils to complete that process. So now we can begin the process for extracting the data. So to extract the data, simply creating a temporary data frame in the Spark cluster using that JSON file, reading that JSON file into that data frame. We'll go ahead and run that. Once that's complete, then we're going to go through the process for verifying the temporary data in the data frame. So I'll go and simply run that cell. Once that's completed, you see it's brought over all of the information, so you can see that it's got a header row and then a bunch of information for us to be able to view. So now we have that extracted and at this point we can begin the transformation process. So in the transform data process, we're going to take that raw sample data and we're going to work with it because really what we only want is we want specific information for it because our end application only needs first name, last name, gender, location, and level for the application to work. So the first bit of code, I'm going to create a temporary data frame, and it's going to show the contents of the data frame as well. And what I'm doing is I'm creating a temporary data frame and I'm selecting out just first name, last name, gender, location, and level from that raw data that I had created previously. We'll go ahead and run that. Once it completes, you see now we simply have those five pieces of information as opposed to all of the raw data that we had in there. And then next, just showing a little sample, I just changed things around a little bit, so I'm creating another data frame. Notice I have v2 at the end, and I'm simply selecting out different pieces of information. In this case, it's artist location, it's song, and user id. So when I run that, (pause) so when I run that, you see it'll bring me a different set of data. So you can simply use select in here to go through and pick out what information you want to work with. So now the next thing what I want to do is I want to do another transformation. So, level really doesn't make sense for my application because what level really is is the type of subscription. So my consuming application is looking for subscription\_type when the application is working. So in this case, what I'm simply doing down here is I'm simply creating another temporary data frame and I'm modifying the header and I'm simply changing it from level to subscription level. I'm sorry, changing it from level to subscription type. We go ahead and run this. And you'll see in the header row, subscription\_type is now the name of that header, and so that's going to be the name for those pieces of information that our application can use. And that's transforming the data. So the next thing we're going to do is we're going to look at loading the data.

Demo - Loading Data in Azure Databricks

Now we're going to kick off our final process of loading the transformed data into the SQL Data Warehouse. So the first thing what we need to do is we need to create values for the temporary blob storage account that's going to be used by Azure SQL Data Warehouse. So the Azure SQL Data Warehouse connector uses temporary blob storage for the process of moving this data. So we put in here our information that we would get from our Azure SQL Data Warehouse asset that we've created inside of Azure. So go ahead and run this. And then next we're going to specify a temporary blob storage location that's going to be used by that process. So, again, run that and that's going to specify that temporary location for us. And then the next piece is we're going to create a value that we're going to store the Azure Blob storage access keys in the configuration so that we can keep them out of plain text, and we'll run that as well. So the next piece what we're going to do is we're going to be setting the values that are needed in order to work with the SQL Data Warehouse. So this, again, is going to be values that you would bring over from your SQL Data Warehouse that you created. We're going to go ahead and run to load those in there. Now we're going to load the transformed data frame that we had called rename columns df, as a table into that SQL Data Warehouse, and we're going to call that sample table. So the first one sets how we're going to set that information when it's going in and then the, on line 6, this is where we're going to write into that SQL Data Warehouse that data frame. So we'll go ahead and run this. One set is completed. We're now going to just simply verify that the table data is actually in the SQL Data Warehouse. So we're simply going to load that sample table into a data frame so that we can display it inside of our notebook. Go ahead and run that. And once it's completed, there you see that we have our table. So it's all of that same data that we had transformed, and that's the process of using ETL within Azure Databricks.

Summary

And that's the ETL process in Azure Databricks. Now you're ready to start transforming your data through the Databrick pipeline for use by your intelligent applications. Again, notebooks provide the basis for all the work here. Through code in the notebooks, you're going to be able to complete all of the pieces from extracting the data, transforming and refining it, and loading it into its final storage place. Once you've worked out the bugs, the notebooks can be run interactively or scheduled based on the needs of the consuming application. And remember, if you don't get the results you need in the end, simply modify your queries and rerun all the notebooks. Super easy to do and it fits into the modern workflow of data scientists and business analysts. If you need further learning or information about the topics covered in this module, check out both sets of documentation on Azure Databricks at docs.microsoft .com and docs.azuredatabricks .net. Both are great resources and will allow you to go deeper on any of the topics related to Azure Databricks. You'll also find curated lists of specific links I think you'll find helpful in the module exercise files on Pluralsight.com. Look for the read me located within the exercise file. And you can always ask me questions on the discussion board on Pluralsight.com. Next up is Batch Scoring of Apache ML Models with Azure Databricks. Here you'll learn how to use machine learning models with Azure Databricks. Thanks for watching, and I'll see you in the next module.

Batch Scoring of Apache Spark ML Models with Azure Databricks

Overview

As organizations create more diverse and more user focused data products and services, there is a growing need for machine learning, which can be used to develop personalizations, recommendations, and predict insights. Azure Databricks and the Apache Spark machine learning library allows data scientists to focus on their data problems and models instead of solving the complexity surrounding distributed data, such as infrastructure and complex configuration. And it allows data engineers to easily build these scalable environments for their users. I'm Michael Bender and welcome to Batch Scoring of Apache Spark ML Models with Azure Databricks. We'll kick off this module discussing machine learning and the Azure Spark Machine Learning library. I'm sure you're up to speed on machine learning, but it's always good to refresh. Then I'll move on to batch scoring and implementing it within Azure Databricks using notebooks. This will all be tied together using a hands-on demo on batch scoring on Databricks. This is going to be based out of a tutorial at docs.com and you'll have an associated GitHub repository with all of the notebooks you'll need for the demo. This will show you how to build a scalable predictive maintenance solution for batch scoring using Apache Spark as a model and scheduling that using Azure Databricks. We'll cover the data ingestion, ML model training, model application for predicting outcomes, and job scheduling. When you're done, you'll have the resources to work with batch scoring in your own environment.

Machine Learning Intro

Before jumping into batch scoring with Azure Databricks, let's just make sure we're on the same page, so we'll start with some foundational stuff to set the stage. As you know, machine learning, per Wikipedia, is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, instead relying on patterns and inferences to make predictions. Machine learning works by taking an existing data set and processing it to discover patterns. These patterns are used to analyze new data and make predictions based on the observed patterns. Machine learning usually relies on specific representation of data, a set of features that are understandable for a computer. The machine learning algorithms build a mathematical model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning libraries provide a repository of utilities and algorithms that help data scientists and engineers as they build out their models. In Azure Databricks, the Apache Spark Machine Learning library is at your disposal. And as you'll see, notebooks provide the framework for implementing machine learning in Azure Databricks through code cells using common ML language like Python and Scala, the process of analyzing, training, and modeling is made easier for data scientists and engineers. To do its work, machine learning relies on models. A machine learning model is a file that's been trained to recognize certain patterns. You train a model over a set of training data, providing it an algorithm that it can use to reason over and learn from the data. Once you've trained the model, you can use it to reason over data that hasn't been seen before that's often referred to as the scoring data. And then predictions about the data are made. A good example is running a train model over a set of data from machine components in a factory to predict component failures. This allows companies to perform proactive maintenance and replacement. In Azure Databricks, notebooks will be used to train the model on existing data and to run the model on future data ingestions. Within Azure Databricks, users have accessed the Apache Spark Machine Learning library, or MLLib for short. This library is Spark's machine learning library. It's goal is to make practical machine learning scalable and easy. At a high level, it provides tools in five different areas. We have ML algorithms that contain common learning algorithms, such as classification, regression, clustering, and collaborative filtering. We have featurization, which focuses on extracting, transforming, and selecting features from data sets. Pipelines provide tools for constructing, evaluating, and tuning machine learning pipelines. Persistence tools for saving and loading algorithms, models, and pipelines. And utility tools, like linear algebra and statistics. As you begin doing your ML work in Azure Databricks, you'll import tooling from the library as your notebooks are run. In this example using Python, the import command is used to access the correlation API for use with data sets and data frames, and the lineal g utility, which is the MLLib utility for linear algebra. Then they're used in the code to output a data frame that contains the correlation matrix of the column of vectors. For more work with the library, be sure to check out the Apache Spark ML library guide using the link below. You'll find plenty of examples and instructions on how to use all of the resources in the library.

Apache Spark Machine Learning Libray

Batch scoring involves a scheduled application of an ML model on a data set to provide predictions. This can be done using a Spark job in Azure Databricks. While batch scoring may involve the use of real time or stream data, the process of scoring is scheduled. You'll often see this in data applications, like determining when a machine component will fail, based on an ML model, so that you can do preemptive maintenance versus waiting for the failure. In Azure Databricks, the best way to implement batch scoring is by using notebooks to build a batch scoring pipeline. A batch scoring pipeline allows you to produce and run ML models against data in a controlled and scheduled manner. Multiple notebooks are used to create the pipeline and complete each step of the process. With notebooks in Azure Databricks, data engineers and data scientists can create batch scoring pipelines by implementing ingestion, feature engineering, model building, and model scorings. Depending on the scenario, these notebooks can be run manually or they can be scheduled using the batch scheduler. Now you're ready to see this in action.

Implementing a Batch Scoring Pipeline

So here are the two resources that you can refer to as you're going over this demo again to get more information about them. First is on docs.microsoft .com is the batch scoring of Spark machine learning models on Azure Databricks. So this document will give you all the information about the reference architecture for this demo, and then if you go to the GitHub website, that's what we're going to be using, that'll bring all of the resources over that you're going to use for this. So it gives you a lot of the same instructions on how to get this all set up, much of the same things that I'm going to do through the demo here. So let's jump into our setup. We're going to start off in azure CLI and we're in our databrickscli. At this point, I'm going to do an ls to see what's available and I'm going to change directories and go into my cloud drive because this is where we're going to put our GitHub repository that we're going to copy over. The next command is using git clone, and this is going to clone that GitHub repository. And again, all of these instructions are in the demo and they'll be in the exercise file for you. Once these are all copied over, what I'm going to do is change directories down and I'm going to go into the BatchSparkScoringPredictiveMaintenance directory that's going to contain all of the notebooks. And at this point, I'm going to cop over all of the notebook from my clone into my Azure Databricks workspace, and I'm going to use the Databricks workspace import directory command to do that. And so as you can see, it's going through the process and it's importing those. Once those are all imported, then we're going to go into the Azure Databricks portal, we'll go into workspaces, go into notebooks, and we see all of the notebooks are listed there for us. What I need to do for each notebook is I need to attach each of the notebooks to the cluster we have. So I'm attaching that to the batch scoring cluster. And I have to do that for all of these. So I'm going to go through the process off screen for you, I'm going to go attach all of these, and I'm going to actually run all of the notebooks so we can focus on what's in each one of the notebooks in the batch scoring pipeline. So we'll see you in the next demo on ingesting data.

Predictive Maintenance Batch Scoring

In our demo, I'll be working with a tutorial on docs.com called batch scoring of Spark machine learning models on Azure Databricks. You'll see the short link below and it'll be in the exercise files for the course. The scenario involves building a batch scoring pipeline on Azure Databricks to predict machine component failures within an organization. Using the scoring information from this, organizations can be proactive with maintenance and replacement of equipment based on the data provided by an ML model. This pipeline will be built in Azure Databricks using notebooks. In the first part of the pipeline, a notebook initiates the ingestion and storage of the raw data. The data ingested in stored in a collection of Databricks data sets, so it's available to all components in your Databricks workspace. In production environments, data is often stored in Databricks accessible storage, like Azure Blob Storage or Azure SQL, just to name a few of the many sources. In our scenario, this data covers 1000 different machines with multiple components and sensors. Information like machine features, telemetry data, maintenance history, failure, and error history are included in the data set. Once ingested, a feature engineering notebook creates a training data set from the ingested data. During this step, the notebook transforms the raw data set into a curated training data set ready to be used for model creation. Next, a model building notebook takes the curated data and trains the machine learning model against the Apache Spark Machine Learning library. This model can be used against new data as it comes in to be scored. And the model is stored inside of your workspace in the Databricks file systems so that it can be easily accessed. During the scoring phase of the pipeline, a notebook executes the feature engineering notebook to create the scoring data set from the ingested data, and then it executes the scoring notebook. The scoring notebook uses the train model that we created during the model building phase to generate predictions against the scoring data set. Once completed, the predictions are stored in the result store, a new data set on the Databricks data store without the workspace. And now this is available for use in whatever data applications consumers need.

Demo: Performing Batch Scoring in Azure Databricks Setup

In this demo, I'll walk you through setting up our Azure Databricks environment and getting it ready to implement a batch scoring pipeline with notebooks This will walk you through cloning the demo GitHub repo, adding notebooks from the repository into Azure Databricks through Databricks CLI, and attaching the notebooks to the Databricks cluster in the Azure Databricks workspace. I have instructions in the exercise files that you can follow, and you can check out the instructions at bit.ly /ghbatch, the links down below. First, we want to log into Azure Cloud Shell and access the Databricks CLI. We'll do this by activating a virtual environment. We created one of these in a previous lab and we'll reuse that. If you need to go over the step for setting this up again, those will be included in the exercise file. For cloning the repo of the demo environment, I'll change the directories to the shells clouddrive. Then I'll run git clone to create the clone in my clouddrive. Once the directory is cloned, then we'll copy the notebooks into Azure Databricks and our workspace using the Databricks workspace command we see here. The URL is going to be felix@ bentech.net and it's going to go into the Users/felix@ bentech.net /notebooks folder inside there. Now let's jump into the Azure Databricks portal for our workspace and then attach the notebooks. So I'll click on the Workspace, I'll click on Users, I'll go into felix's

Demo: Setting up a Batch Scoring Environment

and his notebooks, and there we see all of our notebooks have been added in. At this point, each of the notebooks needs to be attached to our cluster in our workspace. So I'll click on the first notebook and we'll go and attach the notebook to the cluster. So save us some time, I'll go ahead and attach all 10 notebooks to the cluster, while I'm at it I'll run all of the notebooks in sequence so we can explore all of the notebooks during the upcoming demo. So I'll see you in the next demo as we ingest maintenance data into Azure Databricks. This demo has 10 notebooks, so let's go over the notebooks that will be doing the work in our pipeline with. First, we have the data ingestion notebook. This will ingest all of the data we'll use for the demo from five comma separated value files, put those into storage in Azure Databricks. And next will be the training pipeline. This will perform the training and model building through the running of the two notebooks, feature engineering and model building. The feature engineering creates the training data set in Azure Databricks based off a subset of the ingested data, and model building will build the ML model and store it in the Azure Databricks file system for later use. And last, we have the scoring pipeline that'll be used to create the scoring data set using different parameters with the feature notebook, and then it will run the model that we created against that data set. In addition to these notebooks, you'll see optional notebooks in the GitHub repository that can be used after each of the steps in the pipeline to visualize and test what happened. You also find in the notebook there's a ton of information for you to really dive into what's happening in each step of the process. So let's dive into our demo on batch scoring in Azure Databricks.

Demo: Ingesting Data into Azure Databricks

In this demo, you'll see how the data ingestion process works with notebooks in Azure Databricks. With the first data ingestion notebook, the first step of the batch scoring pipeline is executed. Raw data is downloaded and stored in Sparks data frames within your Azure Databricks workspace. First, the notebook will import the utilities needed from the libraries in order to work with the data. You'll see this in every notebook. Just like setting up your Databricks CLI environment, this step is needed to set up the notebook environment to do its work. Then, variables for the URLs and file names are created. And last, for each of the five data sets that we talked about, the CSV file is going to be downloaded from the repository in GitHub. It's read into a pandas data frame, converted to pyspark to ensure it's usable in Spark, and then it's written into storage within the cluster. When we go under Data, you'll see the data tables for machine, maintenance, error, telemetry, and failure, all listed, all from our CSV files. Now that the raw data is accessible in our cluster, feature engineering can begin in the next demo.

Demo: Implementing Feature Engineering

This demo covers the training pipeline notebooks where raw data is transformed into a training data set and the machine learning model is trained and built. It's broken into three notebooks. We have the training pipeline, we have the 2a feature engineering, and we have the 2b model building. So when we take a look at our first notebook, the training pipeline This is used to set the default training file, whether to use Decision Tree or Random Forest for modeling, and specify Databricks parameters used to create the data set. You can modify these through the widget you see here or the code cell, and then it executes each of the notebooks using the defined parameters, using dbutils. Now let's take a look at 2a\_feature\_engineering. This notebook executes the second step of a data science process, feature engineering. This is where we manipulate and transform the raw data sets into a training data set for constructing a machine learning model or for building a data set for the machine learning model to consume. This training data set will be used by the next step modeling to create the ML model in our example. The notebook starts by loading all the libraries that are required and specifying the data location details. Then the notebook loads the raw data from DBFS and combines all of the data into a single data set of features, or variables, that will be used to create a model for determining machine health over a period of time. At this point, your data scientists or engineers would step in with code needed to transform the raw data to a single combined data set. As we scroll through the notebook, you'll see a number of processes for completing feature engineer. Last, the training data set will be stored in DBFS and available within your Azure Databricks workspace as training data. Let's take a look at it underneath the portal in our data. So here we are over in the Data portal. You see, it's been added to that list. We had the five data sets from our first process, now we have the training data set that's going to be used for the modeling process in our next demo.

Demo: Build ML Models

With our next demo, we're going to step into model building. So with this notebook, it constructs a machine learning model from the training data set we just created. This model will be stored in the Databricks workspace for use by batch scoring processes in the future. Again, the required libraries are imported and the training data set is loaded during the notebook process. In the next steps, the predictive maintenance model will be created by preparing the data. Again, you're going to have your data scientists and engineers, they'd be going in here and laying out the code that would be used to build this model. In our scenario, you can choose between a Decision Tree or a Random Forest classification method. Other parameters can be tweaked in the model building as you explore the model. This is a phase where users will rerun the notebook with tweaks until they get the model performing as they wish. The final step of this process is writing the model in a parquet file in DBFS for later use. With the model complete, you move into the final step of model scoring where the model will be used against another set of data and predictions will be made. That's our next demo.

Demo: Modeling Data in Azure Databricks

In this demo, we're going to go through the scoring pipeline. Like the training pipeline phase, a single notebook, the scoring pipeline, is used to provide details on the model building process and then kick off the process with two notebooks. So it's going to use the feature engineering notebook, which we've used before, and then a new model scoring notebook. The feature engineering notebook will convert the raw unseen data into a scoring data set. And the model scoring notebook will run the newly created model against that data set. In our demo, the scoring data set is a subset of the original raw data set using a data range that was not part of the training data. This simulates unseen data. Let's run the notebook and then we'll explore the model building process. Just like previously, you're going to load the required models needed for the notebook, and then we're going to set down some parameters for the customized run for creating that scoring data set that includes start and end date. This data set, called the scoring table, will be stored as a temporary data set inside of Azure Databricks. Feature engineering is then run, this time it's transforming the scoring data set for model consumption. Next step, the model is loaded from DBFS. The scoring data set is scored using that model and the results are persisted, also known as saved, in a data set inside the cluster. Now when we look in Data, you'll see two tables we just created, results output, created in this notebook, and scoring data, which was created by the feature engineering notebook in the previous step of the process. Now let's take a look at the exploration notebook that lets you visualize the data in a notebook. In the model evaluation notebook, you see a table is displayed laying out the component failures and the predicted values laid out in our model. From here, the results data could be placed into more permanent storage, like a data warehouse, for consumption by reporting or analysis tool. And that's how a scoring pipeline works in Azure Databricks using notebooks. In our last demo, we're going to show you how you go through the process for creating a job and setting this up as a batch scoring job. So see you in the next demo.

Demo: Add Batch Scoring Job for Scoring Pipeline

In our last demo, let's create a batch scoring job so the scoring process can be run on a scheduled basis. As you know, this process is a pipeline of operations, completed through the use of multiple Databricks notebooks. All you need to do is to point the scoring pipeline at a source of data you wish to work with and let it do its job. To create our job, we're going to customize a JSON template that is part of the demo GitHub repo. This JSON file will build the Spark job that instructs Azure Databricks to run the scoring pipeline with a specific set of parameters. Since creating a job in the portal is pretty straightforward, let's do it through the CLI with Azure Cloud Shell. Inside of Azure Cloud Shell, I'm going to jump into the Databricks CLI by activating the virtual environment. Then I want to change directories to my clouddrive where my GitHub cloned repo resides. Once important there, the first piece of data we need is the cluster id, so run databricks clusters list for the current clusters available in the workspace. This is needed when editing the JSON template. Next, open the JSON template in code. So we're going to make edits to this. So I'm going to change the clusterid in here and the username to match my environment. As you can see here, it has all of the information needed in order for Databricks to be able to run this job and execute the scoring pipeline. Once I'm done with all of this, this JSON file would create a job that would need to be run manually. So what if you want to schedule it to run on a regular basis? Maybe you want it run hourly every day of the week. Not a problem. Just add a schedule parameter to the code. In this example, the job will run daily, every hour, starting on the half hour. So 0730 to 1830. So you'll notice the schedule parameter here. So I go ahead and add this in. Then I'm going to go and save and close this. Now my JSON template is all ready to go. Now I want to create the job using the JSON file. So I'm going to use the databricks jobs create command, putting in the --json-file parameter with the path to my template. Once this completes, you want to take note of the job id because next, if we want to actually run this now, we could enter databricks jobs run-now and --job-id with the id number to run that. And we'll go ahead and do that. Let's jump back into the portal and click on the job. So when we click on the job, we'll see much of the information that was specified in the JSON file. Say we wanted to run this with different parameters, you could go in and modify the JSON to use a different date range. So as you can see here, I can go in and change the date. Super easy to adjust as you need to. And that's how you build a batch scoring pipeline built in Azure Databricks and schedule batch scoring for your ML models.

Summary

And there you have it, batch scoring in Azure Databricks. With Azure Databricks-- With Azue Databricks creating a batch scoring pipeline using notebooks is straightforward, from data ingestion to training and modeling to scoring new data. The only challenge may be keeping all of the notebooks straight. One of your big keys to make sure that you're successful with this is make sure to leverage the Apache Spark Machine Learning library. It has tons of tools and algorithms for your data scientists and data engineers to do their job. But one thing I did show you was a handy way to batch schedule the scoring process so it's quick and easy to run your scoring pipeline manually or you can run it on a scheduled basis if that data needs to be refreshed in a regular basis against your models. And that's it for batch scoring in Azure Databricks. If you need further learning or information about the topics that we covered in this module, check out both sets of documents on Azure Databricks at docs.microsoft .com and docs.azuredatabricks .net. Both are great resources and will allow you to go deeper on any of the topics related to Azure Databricks. You'll also find that curated list of links that I think you'll find helpful for each module in the exercise files on Pluralsight.com. Just look for the read me file for more information. And next up is our last module, Streaming HDIinsight-- And next up is our last module, Streaming HDInsight Kafka Data into Azure Databricks, where you'll learn how to work with-- where you'll learn how to work with streaming data from HDInsight and Apache Kafka in Azure Databricks. Thanks for watching and I'll see you in the next module.

Streaming HDInsight Kafka Data into Azure Databricks

Overview

A big challenge for many organizations is dealing with sensors, IoT, social networks, and online transactions that generate data that needs to be monitored constantly and acted upon quickly. As a result, the need for large-scale, real-time stream processing is greater than ever. In Azure, Apache Kafka and HDInsight provide this much needed service. So if you deal with this type of data, you've come to the right place to learn how to stream data into Azure Databricks with HDInsight and Kafka. I'm Michael Bender, and welcome to Streaming HDInsight Kafka Data into Azure Databricks. In this module, you'll learn how to use HDInsight in Apache Kafka to stream data into Azure Databricks. A common streaming scenario is taking a large set of changing data from many different sources and then performing a data analysis task against it. And that's a perfect scenario for using Apache Kafka running on HDInsight and Azure Databricks. In this demo, you'll see Apache Kafka on HDInsight used to stream data into Azure Databricks. I'll build each of the respective clusters, peer the virtual networks, author event producers to Kafka, and consume events from Kafka using Azure Databricks notebooks. So let's dive in.

Apache Kafka on Azure HDInsight

Apache Kafka is a popular open source event ingestion platform for storing and aggregating events from multiple sources and providing a single source for distributing collected data to data consumers. It's a single platform meeting the needs of event producers and consumers. In Azure, HDInsight is the platform for hosting Kafka. Azure HDInsight is a managed, open source, analytics service. You can use many open source frameworks such as Hadoop, Apache Spark, Apache Kafka, and more with HDInsight. So when we start taking a look at streaming with Kafka, the general setup for streaming with Kafka is quite simple. Producers send records to the clusters where they're stored as topics. A topic is just a write-ahead log where the producers append records. Within the clusters, topics are assigned to the partitions that can be replicated for fault tolerance across the clusters. In the case of Azure, Kafka topics reside in HDInsight clusters. These records, based on a key value pair, are available for consumers to use. Consumers will subscribe to the topics they need and receive changes in real time as event producers send more data into Kafka. Both sides work at their own pace with Kafka acting as the intermediary. All communication by producers and consumers happens with Kafka brokers running in the cluster, lots of flexibility with Kafka on HDInsight being the center of this event ingestion platform. Now let's get to work and start streaming. For our demo, I'll use data from a public dataset of taxi trip data to populate our data streaming. This includes location, fare, and many other pieces of data gathered on taxi trips in New York City in 2016. The idea is to store the data in a Kafka topic and then consuming a subset of the data into Azure Databricks. This is going to involve a number of steps. We're going to install the required components for HDInsight. We're going to build and configure HDInsight with Kafka. We're going to build an Azure Databricks workspace and cluster so that we're ready for streaming. We're going to implement virtual network peering between the two clusters. And then we'll kick off event production and consumption streams using Databricks notebook. The demo is similar to a demo on docs.com about structured streaming, except we've chosen to include virtual network peering and Azure Databricks to the mix. The link is below and in the exercise file for your reference. Now let's dive in and start building our first cluster.

Demo: Building a HDInsight Kafka Cluster

In this demo, an Apache Kafka cluster is built with Azure HDInsight. Before we get started building the Kafka cluster, I need to add some resources into Azure. First, I'm going to add the resource group that we home for all of the resources built out, including the HDInsight cluster and its resource components. So, I'm going to go into the Azure portal, I'm going to click on the Resource groups icon, and I'm going to choose Add to add in a new resource group. I'm going to add in the name, streaming-data-rg as the resource group, and I'm going to click Create. After a few moments, we see that that resource group is created, so I'm going to go in to the resource group. I'm going to create the virtual network that's going to be used by our HDInsight cluster. I'm going to click Add. From the Marketplace, I'm going to choose Virtual Network, choose Create, and we'll go in and we'll create our virtual network. I'm going to call our virtual network hdinsight-kafka-vnet. I'm going to give it a 10.0 .0 /16 IP address. I'm going to leave the subscription. It's going to be the resource group. Since I created it in there, it's going to put that in there. I'm going to leave it at Central US. We'll leave the default name, and then we'll give it an address in the /24. That will set us up with 256 addresses. We'll leave the rest of the basics here. I'll click Create, and now our deployment is in process. So once that's completed, then we'll be ready to go through the process for actually installing HDInsight with Kafka. So what I'm going to do to do that is I'm going to go into my resource group. We'll do a quick Refresh. We see that our virtual network is there. At this point, I'm going to click Add, and I'm going to look for HDInsight. Click on HDInsight, click Create to start off the creation wizard. So first, I want to put in the cluster name. We'll see it's going through and it's going to create a name for us on the azurehdinsight.net network, and it will give us a name there. So we'll leave the subscription. We see that that is good to go, so that name's going to be good for us. At this point, I want to choose the type of cluster. From the drop-down list, you see we have a number of different types of clusters. I'm going to choose Kafka. We're going to leave the version there. Click Select, and at this point I'm going to put in the Cluster login password. There's our password, and notice it's got an SSH login as well. We'll use the same password for SSH. Of course in a production environment, you would change the cluster login name to something, and hey, why don't we go ahead and do that? We're going to go ahead and put this in as streamadmin so it's a little bit more secure. We'll leave the sshuser. We've got our streaming data resource group. And we're going to move this over to Central US so we keep all of our resources there. We'll click Next. Once that resolves, we're going to choose the virtual network we just created. We're going to leave all of the defaults here. We'll click Next. For the storage, we're going to go ahead and create some storage here. So I'll do Create new, and we're going to create a new storage group. And we're going to call that hdinsightkafkasa. We see that that's okay. And we'll click Next. For the applications, we're not adding any optional applications, so we'll click Next there. We're going to change the number of nodes down to three since we don't need a lot for this. And if you go into each one of these nodes, you'll see that it gives you the ability to manage the pricing tiers, the different component. When you look at your environment and you're determining how to push this out and how quickly you want to compute the work and how powerful you want it, you would go through and change all the worker that had the zookeeper nodes for this. We're going to go ahead and leave that. And notice it'll show me the price. This is in hours, so as soon as this starts up, it's going to charge you $239 US, so you want to keep this in mind as you're playing around with this because this can get really costly. Because this just includes what's created for the HDInsight cluster, it doesn't include any other resources that are going to get built out along with the process like the Azure Databricks cluster as well. So we'll click Next. We're not going to add any script actions. And then it's going to go through and validate that everything looks good. And I'm going to go ahead and create. And now it's going to go through the creation process for creating all this. This'll probably take about 15 to 20 minutes. So once this is completed, we'll go in and we'll do some post configurations to make Kafka and HDInsight ready to work with Azure Databricks. And that's what we'll see in our next demo.

Demo: Configuring Kafka for IP Advertising

In this demo, the HDInsight Kafka clusters configured to advertise using IP addresses instead of host names, so that'll allow us to be able to connect into the Kafka cluster and topics using an IP address. To get started with this, the easiest way is to use the URL to the website. And to get to the URL, you notice we're in here in our resource group. I'm going to click on the HDInsight cluster, and you'll see under the settings that it has a URL. I'll simply launch the URL. And that's going to connect me to the azurehdinsight.net service. I have the password stored already for this, so I put in this streaming admin and the password that I had used when creating Kafka. Once we're inside the portal, I'm going to go ahead and click on the Kafka and Configs so I can begin changing the configuration to allow us to advertise using IP addresses. First thing I need to change is the Kafka-env template. And what we're going to do is we're going to do down to the bottom here, and I'm going to insert a piece of code. And you'll find this code in the exercise file. And this'll tell Kafka to use IP addresses for its configuration. And the next thing what I need to do is change the listeners. I'm going to change that from localhost to 10.0 .0 .0. We'll go ahead and save these, put in updates for the save configuration, choose OK. And now that my changes are made, what I need to do is I'm going to need to do a restart to get Kafka using these new components. I want to do this in Maintenance Mode. So I'm going to turn on Maintenance Mode here. I'm going to click OK, click OK. And at this point, I can go ahead and restart all of the affected. I'll Confirm Restart All, and now it's going to go through the process for doing all of the restarts. Once that's completed, we click OK, and we see here's all of our brokers. The next piece that we need out of this is we need to have the IP addresses for the brokers and the zookeepers. That's going to be how we're going to be able to actually produce events into Kafka and then consume those out of that by subscribing to the topic that we create. What you need to do is I'm going to go in here, and I'm going to simply grab each of the IP addresses for the brokers. Now, I'm going to go into the zookeepers, and I'm going to do the exact same thing. And I'm simply going to grab each of the IP addresses. And now that I have all of those set, I can go into our next demo, which is going to be creating a topic inside of Kafka on HDInsight. We'll see you in the next demo.

Demo: Create a Kafka topic

With our HDInsight Kafka installation ready to go, it's time to create a new topic in Kafka. This will be used to store the retrieved taxi information that we have and make that available to be consumed by Azure Databricks. In the Azure portal, we're in our streaming data resource group, I'm going to go ahead and click on the Kafka cluster, and from here we're going to be using SSH. So I'm going to go down to the SSH+ cluster login. I'm going to choose the hostname from the drop-down, and I'm going to click Copy to copy the SSH command that's going to be used for me to connect in. The easiest way for me to connect in is simply using Azure Cloud Shell, so I'll click on the Cloud Shell icon. And that'll open up Cloud Shell in the Azure portal. Once I'm in the Azure portal, I'll put in my command to log in with SSH, and it's going to ask me, yes, I'm sure I want to continue. I'm going to put in my password that I had used previously, and there we are connected in to our HDInsight Kafka cluster. At this point, I'm going to use the export command to set the topic equal to taxidata. And then I'm going to use the export command as well to set KAFKAZOOKEEPERS to the IP addresses of the zookeepers. Those are the ones that I copied over from the portal. [00:13: 28.000 ] Next, we're going to enter a command in that's using kafka- topics.sh, a shell script used for creating topics within Kafka. It's going to call both the topic and the KAFKAZOOKEEPERS, and that will create our topic for us. And there we go. We see our taxidata topics created. It's ready to start having events produced into it, so at this point, now we're going to switch over to using Azure Databricks to create our cluster and then set up our virtual network peering. We'll see you in the next demo.

Demo: Building and Configuring an Azure Databricks Cluster

Now it's time to build out the Azure Databricks workspace and cluster, a process you should be familiar with. Starting in the portal, I'm going to search for Azure Databricks. We're going to go in and click in on that. I'm going to choose Add, and we're going to go through the process for creating our Databricks service like we've done before. We'll give it a name. We'll leave the subscription. I'm going to put it into that existing user group, the streaming-data-rg we just created where we put the Kafka HDInsight cluster. We'll choose our pricing tier, we're going to go with the Trial. And then we'll go ahead and Create. Once that's all set, then we need to go in and actually make sure that our workspace is running and we're actually going to create a cluster. Now that our workspace is created, let's go ahead and go into that workspace. So we'll go ahead and launch the workspace. We're going to sign in, doing our typical single sign-on into Azure Databricks. Now that we're logged in to the Azure Databricks workspace, it's time to create a cluster. Go ahead and click New Cluster. We'll give our cluster a name, and we're going to leave all of the other defaults for the creation of the cluster. We'll go ahead and click Create. And in a couple moments, our cluster will be created. As you can see, our cluster is created. And it is in the process of getting started up. It will be ready for us when we come back later to start producing events into Kafka and then consuming the events from the Kafka topics. Let's dive into the next topic, which is going to be virtual network peering.

Virtual Network Peering

Oftentimes, resources in Azure are deployed to different virtual networks based on the needs of the services. These networks are isolated by design. For those occasions when you need two virtual networks to be connected and the resources need to communicate, Azure virtual network peering is here to help. With peering, services in different virtual networks, or VNets, communicate with each other via high-bandwidth, low-latency Azure fiber backbones. In our streaming scenario, setting up virtual network peering between Kafka HDInsight and Azure Databricks is going to establish this private communication network and allow streaming to happen. This is done by configuring each side of the peered network. While Azure does support virtual network peering across regions, known as global VNet peering, you'll see peering set up within a single Azure region, the Central US region for our demo. Now let's see how we set up virtual network peering. For our demo, peering will be established between the Azure Databricks VNet and the HD Kafka VNet. So we'll start off in the Azure portal inside our resource group. We're going to go down to our Azure Databricks resource. We'll go down to Virtual Network Peerings. We'll click add Add Peering so we can add our peering into this. We're going to go ahead and call this peer spark-peer. And then the virtual network, notice that it has the kafka-vnet. And we're going to leave that so that Allow virtual network access is enabled. All of the other options we are simply going to leave as they are. We'll go ahead and click and Add that in. Once that's completed, we need to go over to the other side and complete the rest of the peering. We're going to do that by going over back into our resource group. I'm going to go up to our virtual network. And I'm going to go to Peerings, and I'm going to click Add here. On this side, I'm going to call this kafka-peer. I'm going to choose from the Virtual network. We're going to choose the workers-vnet. This is going to be the virtual network that was created when we created the Azure Databricks cluster. Depending on how many demos and how many clusters you have with Azure Databricks, you might have multiple here. You want to look for the name of the cluster. We see sparkcluster-ws in the full name of this worker net. We'll choose that. Notice it grayed out. It shows spark-peer, so we know that we're connected properly. We're going to leave the defaults here. We'll click OK. We'll click Refresh. We'll see that we are connected. Just to verify this is all good to go, I'm going to simply go back over to the sparkcluster. I'm going to go into my Virtual Network Peerings, and we'll do a quick refresh. After we do a refresh, we see that those are both connected. Now, virtual network peering is all set for us. At this point, we're ready to start talking about connecting Azure Databricks to Kafka and producing events and consuming events.

Azure Databricks and Streaming Data

Once connected to Kafka on HDInsight via peered VNet, Azure Databricks, using connectors built into the spark platform, make the connections needed to establish the streaming process. Like most things these days, all of this is done in code. And the best place to do that in Azure Databricks is using notebooks. For basic streaming, a notebook for producing events will be created. This establishes the flow of data from outside sources into Kafka and placed within topics for consumption. Another notebook will be created to consume the topic desired from Kafka brokers within the cluster and bring the data into Azure Databricks. From there, data scientists and data engineers can do whatever they need with the data. It can be added into data frames, visualized or moved to other storage services for end user consumption; lots of choices for you.

Producing Events and Consuming Data with Azure Databricks Notebooks

Now it's time to start working with the streaming data. In this demo, you'll use two notebooks, Taxi Trip Data - Produce and Taxi Trip Data - Consume, to handle the event production to and consumption from the taxi dataset in Kafka. Starting back in the Azure Databricks workspace, we'll click on the workspace and go into Shared because what we need to do is we need to import the notebooks. We'll choose Import, Notebook, and then import both of the notebooks. The two notebooks to be used are available in the exercise files. Once the notebooks are imported, we'll dive in. Let's take a look at the Taxi Trip Data - Produce notebook. Basically, this notebook will download the taxi data in JSON into a temporary data frame in Azure Databricks and then write the data in the data frame into the Kafka topic. The notebook starts by importing the needed libraries and creating a temporary data frame in Azure Databricks for use by the production process. When run, you'll see all of the pieces of data that will become part of the topic in Kafka. As you see here, we see all of the fields and data. Next, the Kafka brokers that will be used for writing the data and the destination topic are defined. This is where the IP addresses for the brokers from the Kafka portal come into play. You'll want to make sure that you have the right IP addresses. Last, the data from the data frame is written into Kafka. The Vendor ID field is specified as a key value, which is going to be used by Kafka when it partitions the data going into the topic. In this scenario, the data source is static, so it's written using a batch query method. The fields from the source are written into the Kafka message, also known as the value field, with a JSON string value. In case of a constant data stream like IoT devices, you would modify the code to stream the data into topic and continue running the notebook as needed. Now that the events have been produced and authored into the topic, it's time to begin consuming events. Now let's take a look at the Taxi Trip Data - Consume notebook. This is where the topic is subscribed to and the records are fed into Azure Databricks from the Kafka topic. Again, the notebook starts off with establishing the Kafka broker configuration and setting the topic. Make sure these match what you used in the producer notebook. Once the tools are imported, a schema is defined for the data. This will define the fields stored in the Kafka message. This can be used to refine the data to a specific dataset by removing fields from the schema. Next, notebook subscribes to the Kafka topic. In this instance, spark.read is used to create a batch query, so this action only occurs once. The data selected based on the schema defined and written into DBFS using a parquet format. It's now available in Azure Databricks based on the schema. Let's verify the data was written by viewing the files in DBFS. Using %fs ls and the directory path, the parquet files are displayed. Last, let's view the trip value and its stored fields by using sqlContext.read .parquet and the directory. This will load the data into a data frame and then display the data frame as you see here. Now let's say you only want a specific set of data to be consumed, and that data is to be streamed. You'll run through a similar set of commands, including loading the broker configuration and setting the topic again. Then the schema is modified to include only the fields to be in the final write as you see here. We've only listed a subset of the fields that we want to come in. To stream the Kafka topic, spark.readStream is used. With a stream, the notebook will continue streaming data in while the code is running. Like previously, data is selected based on the schema and written to a file. Notice the awaitTermination is set to a value of 30, 000 ms, so this is not going to run indefinitely. You probably want to set this so that it doesn't keep running on you. Let's review the results in DBFS. There we see the parquet files. And when the parquet files are read into a data frame and displayed, you see the value contains only the fields defined in the schema. Now you would be able to do whatever you would need to do with that data in Azure Databricks going out to all of your data consumers. And that's producing and consuming events from Kafka into Azure Databricks.

Summary

Streaming data comes in lots of different forms, and it's an important data type in many organizations. An effective streaming setup using structured streaming in Kafka allows for a flexible and scalable way of managing this data. In Azure, Kafka runs on HDInsight. In order to work with all these different data sources, the data needs a place for event producers and consumers to meet. That's where Kafka and topics come in. Topics provide a table-like key value pair format for ingestion of data from event producers and distribution of data to event consumers, all managed by the Kafka brokers. In Azure Databricks, it's easy to produce events and consume them all through a single component, notebooks. Notebooks for each type of process build the streaming data pipeline for you and allow you to control the production and consumption processes. And that wraps up Streaming Data with HDInsight Kafka and Azure Databricks. Next up on your data engineered learning path is a great course by Tim Warner, Building Batch Data Processing Solutions in Microsoft Azure. In this course, Tim covers more scenarios involving batch data processing with services like Hadoop, Hive, and a few familiar services like Azure Databricks. Thanks for watching, and I hope you learned a lot about Azure Databricks.