Databricks Architectures

Saturday, August 29, 2020

10:47 AM

|  |
| --- |
| Machine generated alternative text: Real Time Analytics on Big Data Architecture  red)  C' i "streams a  Events  Components  Am re Data  Azure H DI r%ight  Pmp and tnin  Data Lake  Oat  0  Ann DB  Power B I  - time Apps  Data Flow  I.Easily ingest live streaming data for an application using Apache  Kafka cluster in Azure HDlnsight.  2.Bring together all your structured data using Azure Data Factory  to Azure Blob Storage.  3.Take advantage of Azure Databricks to clean, transform, and  analyze the streaming data, and combine it with structured data  from operational databases or data warehouses.  4.Use scalable machine learning/deep learning techniques, to  derive deeper insights from this data using Python, R or Scala, with  inbuilt notebook experiences in Azure Databricks.  5.Leverage native connectors between Azure Databricks and Azure  Synapse Analytics to access and move data at scale.  6.Build analytical dashboards and embedded reports on top of  Azure Data Warehouse to share insights within your organization  and use Azure Analysis Services to serve this data to thousands of  users.  7.Power users take advantage of the inbuilt capabilities of Azure  Databricks and Azure HDlnsight to perform root cause  determination and raw data analysis.  8.Take the insights from Azure Databricks to Cosmos DB to make  them accessible through real time apps.  •Azure Synapse Analytics is the fast, flexible and trusted cloud data warehouse that lets you scale, compute and store elastically and independently, with a massively parallel processing  architecture.  •Azure is a hybrid data integration service that allows you to create, schedule and orchestrate your ETL/ELT workflows.  •Azure Data Lake Storage: Massively scalable, secure data lake functionality built on Azure Blob Storage  •Azure Databricks is a fast, easy, and collaborative Apache Spark-based analytics platform.  •Azure HDlnsight is a fully managed, full spectrum open-source analytics service for popular open-source frameworks such as Hadoop, Spark, Hive, LLAP, Kafka, Storm, R & more.  •Azure Cosmos DB is a globally distributed, multi-model database service. Then learn how to replicate your data across any number of Azure regions and scale your throughput independent  from your storage.  •Azure Analysis Services is an enterprise grade analytics as a service that lets you govern, deploy, test, and deliver your Bl solution with confidence.  •Power Bl is a suite of business analytics tools that deliver insights throughout your organization. Connect to hundreds of data sources, simplify data prep, and drive ad hoc analysis. Produce  beautiful reports, then publish them for your organization to consume on the web and across mobile devices. |

|  |
| --- |
| Machine generated alternative text: Advanced Analytics Architecture  Logs, fies, and  media (unstructured)  Business/custom  apm (structured)  Components  Ingest  Azure  Data Factory  Stcre  Am re Data  Lake Storage  Prep and Train  Azure  Data)ricks  M Ockl and serve  re Syn ame  Analytics  Azure  Cosmt% DB  Azure AnalysS  power Bl  web  Data Flow  1. Bring together all your structured, unstructured and  semi-structured data (logs, files, and media) using Azure  Data Factory to Azure Data Lake Storage.  2. Use Azure Databricks to clean and transform the  structureless datasets and combine them with structured  data from operational databases or data warehouses.  3. Use scalable machine learning/deep learning techniques,  to derive deeper insights from this data using Python, R  or Scala, with inbuilt notebook experiences in Azure  Databricks.  4. Leverage native connectors between Azure Databricks  and Azure Synapse Analytics to access and move data at  scale.  5. Power users take advantage of the inbuilt capabilities of  Azure Databricks to perform root cause determination  and raw data analysis.  6. Query and report on data in Power B'.  7. Take the insights from Azure Databricks to Cosmos DB to  make them accessible through web and mobile apps.  •Azure Synapse Analytics is the fast, flexible and trusted cloud data warehouse that lets you scale, compute and store elastically and independently, with a massively parallel  processing architecture.  •Azure Data Factory is a hybrid data integration service that allows you to create, schedule and orchestrate your ETL/ELT workflows.  •Azure Blob storage is a Massively scalable object storage for any type of unstructured data-images, videos, audio, documents, and more-easily and cost-effectively.  •Azure Databricks is a fast, easy, and collaborative Apache Spark-based analytics platform.  •Azure Cosmos DB is a globally distributed, multi-model database service. Learn how to replicate your data across any number of Azure regions and scale your throughput  independent from your storage.  •Azure Analysis Services is an enterprise grade analytics as a service that lets you govern, deploy, test, and deliver your Bl solution with confidence.  •Power Bl is a suite of business analytics tools that deliver insights throughout your organization. Connect to hundreds of data sources, simplify data prep, and drive ad hoc  analysis. Produce beautiful reports, then publish them for your organization to consume on the web and across mobile devices. |

**PolyBase** enables your SQL Server instance to process Transact-SQL queries that read data from external data sources. SQL Server 2016 and higher can access external data in Hadoop and **Azure** Blob Storage. ... This allows you to combine data from external sources with high-value relational data in your database.

Databricks - Query SQL Server running in a Docker container

Sunday, July 14, 2019

1:45 PM

**Brief Overview:**

* + Create a VNET
  + If you don't already have Ubuntu for Windows and SQL Server Management studio on your workstation, Download and install:
    - Ubuntu for Windows
    - SQL Server Management Studio
  + Create a VM. Ensure that the **Ubuntu Server version is 16.04 LTS** and the VM size is Standard **B2ms**
  + Go to the VM's IP screen and change the public IP to Static and enter a name in the DNS name label
  + Go to the Network Security Group and add the following Inbound rules:
    - Source IP Any (or My IP address); port Any. Target IP Any and target port is 22. Protocol is **Any** and Action is **Allow**
    - Source IP Any; port Any. Target IP Any and target port is 1433. Protocol is **Any** and Action is **Allow**
  + Open an SSH session into your VM
  + Install SQL Server 2017 on it
  + Open up SQL Server Management Studio and connect to your newly created SQL Server
  + Open up a Query screen and create a database, a table and insert a few rows in the table
  + Go to your Azure portal and create a Databricks service.
  + When the service is created, click on the Databricks Workspace
  + When the Databricks Cluster screen opens up, create a cluster
  + Create a Python notebook.
  + Ensure that you can connect with the VM of the SQL server
  + Create a database connection and run a query against the SQL server database table and see the data that you had created in the SQL Server in the Azure Databricks cluster

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Azure Databricks Guidance

<https://docs.microsoft.com/en-us/azure/azure-databricks/vnet-injection-sql-server>

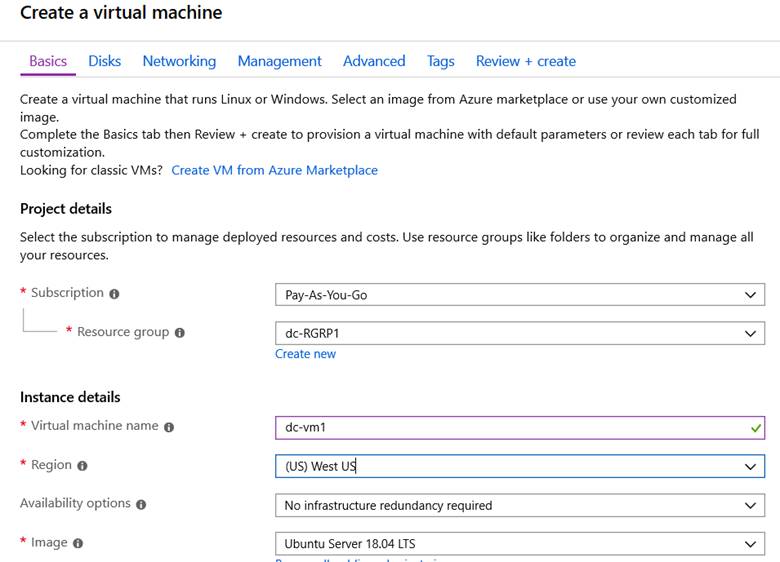
<https://docs.microsoft.com/en-us/azure/azure-databricks/quickstart-create-databricks-workspace-vnet-injection>

**Prerequisites**

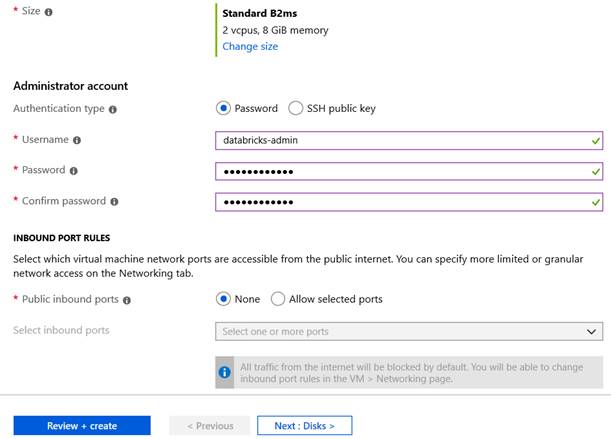
* + Create an [Azure Databricks workspace in a virtual network](https://docs.microsoft.com/en-us/azure/azure-databricks/quickstart-create-databricks-workspace-vnet-injection).
    - **Local created version and documented:** [**Databricks Workspace in a Virtual Network**](onenote:#Databricks%20Workspace%20in%20a%20Virtual%20Network&section-id={03256399-F65F-4D04-ABBB-CFAE428E2977}&page-id={7D6F090B-D088-4F91-9C9C-CF7C9EE4F9B2}&end&base-path=D:\Dave\DC%20-%20Notebook\Technology%20Notes.one)
  + Install [Ubuntu for Windows](https://www.microsoft.com/p/ubuntu/9nblggh4msv6?activetab=pivot:overviewtab).
  + Download [SQL Server Management Studio](https://docs.microsoft.com/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017).

**Create a Linux virtual machine**

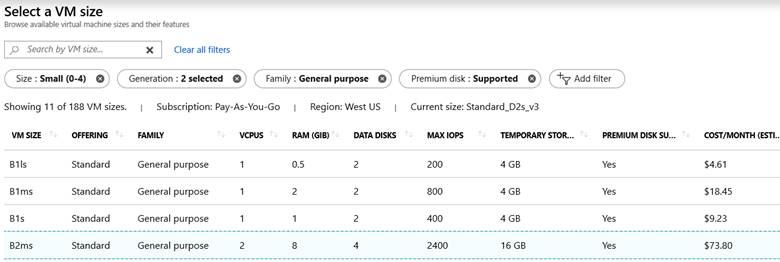
* + In the Azure portal, select the icon for **Virtual Machines**. Then, select **+ Add**.



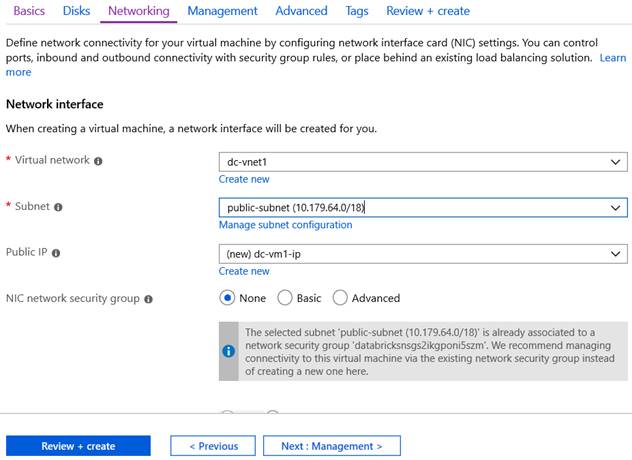
* + Ubuntu Server **must** be version 16.04 LTS
  + The size must be equal to the size required for the VM + SQL server, e.g., Standard B2ms (2 vcpus, 8 GiB memory)



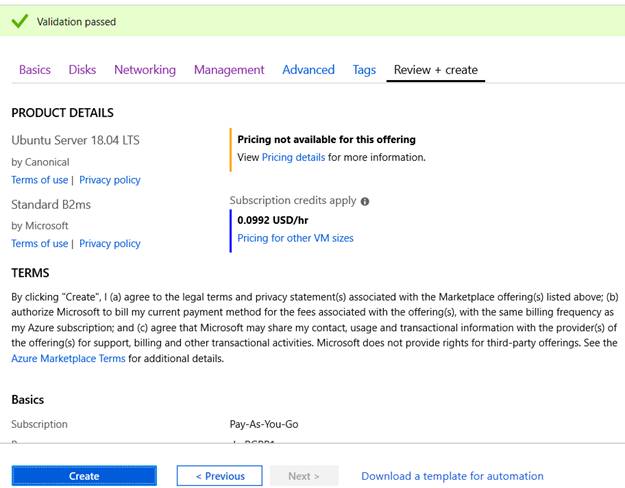
Select the right size. NOTE: smaller than required VM will create unintended results.



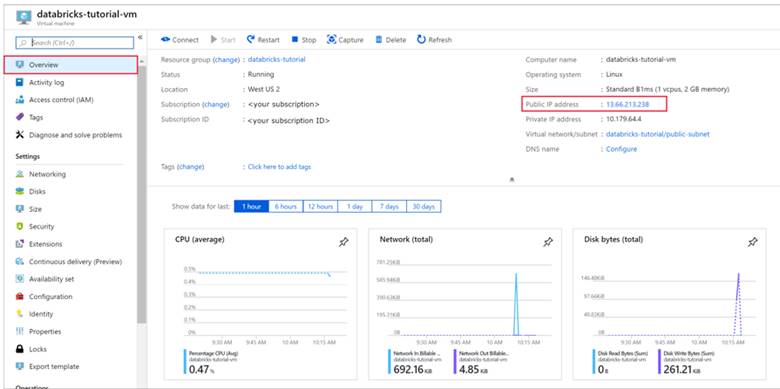
* + Make networking setting selections



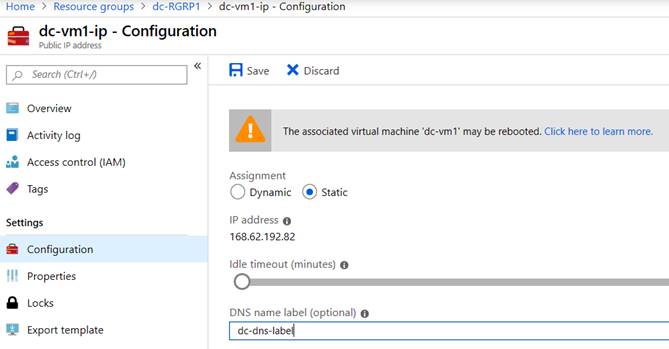
* + Click on Review + Create
  + If validation passes, click on create



When the deployment is complete, navigate to the virtual machine. Notice the Public IP address and Virtual network/subnet in the **Overview**. Select the **Public IP Address**



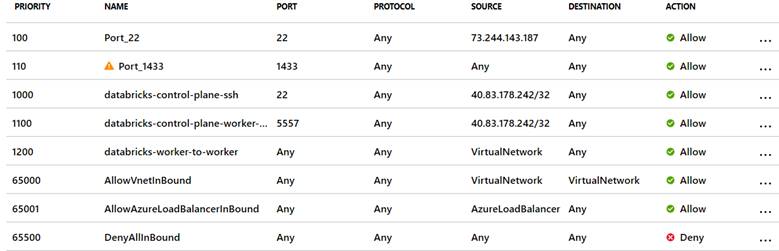
Change the **Assignment** to **Static** and enter a **DNS name label**. Select **Save**, and restart the virtual machine.



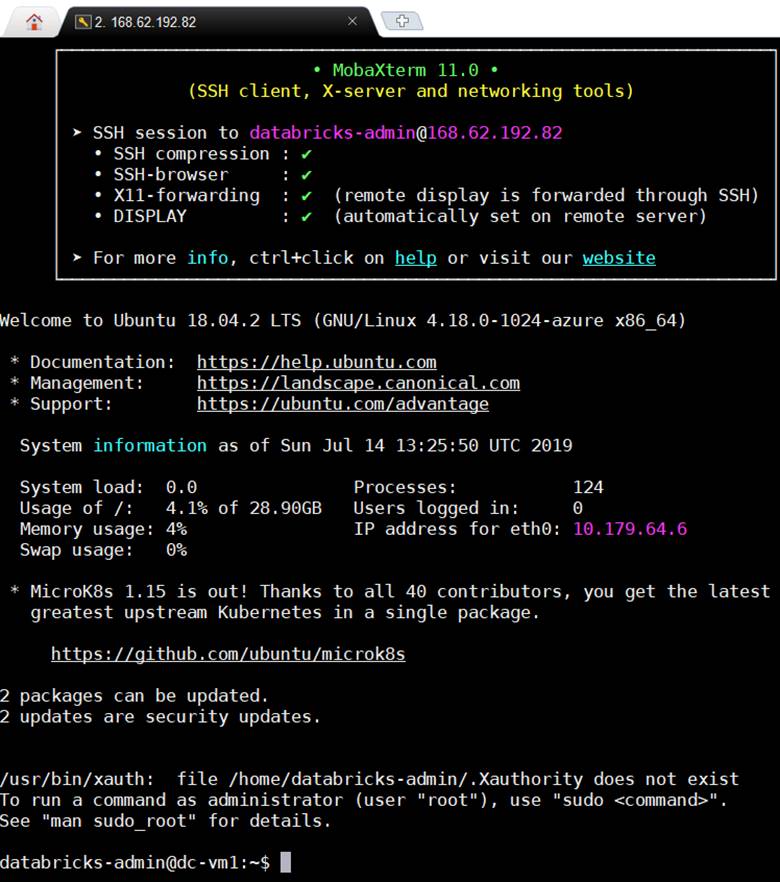
* + Select the **Networking** tab under **Settings**. Notice that the network security group that was created during the Azure Databricks deployment is associated with the virtual machine. Select **Add inbound port rule**.
  + Add a rule to open port 22 for SSH. Use the following settings:

|  |  |  |
| --- | --- | --- |
| **Setting** | **Suggested value** | **Description** |
| Source | IP Addresses | IP Addresses specifies that incoming traffic from a specific source IP Address will be allowed or denied by this rule. |
| Source IP addresses | <your public ip> | Enter the your public IP address. You can find your public IP address by visiting [bing.com](https://www.bing.com/) and searching for **"my IP"**. |
| Source port ranges | \* | Allow traffic from any port. |
| Destination | Any | IP Addresses specifies that outgoing traffic for a specific source IP Address will be allowed or denied by this rule. |
| Destination port ranges | 22 | Open port 22 for SSH. |
| Priority | 100 | Give the rule a priority. Rules are validated based on priority |
| Name | ssh-databricks-tutorial-vm | Give the rule a name. |

Your inbound security rules for port 22 and 1433 should look like:



* + Open up an SSH session using **MobaXterm 11.0** using the public ip address of your VM. Use the user id and password you had entered for the administrative account when you created the VM.
  + Once the SSH session to the VM is open it should look like:

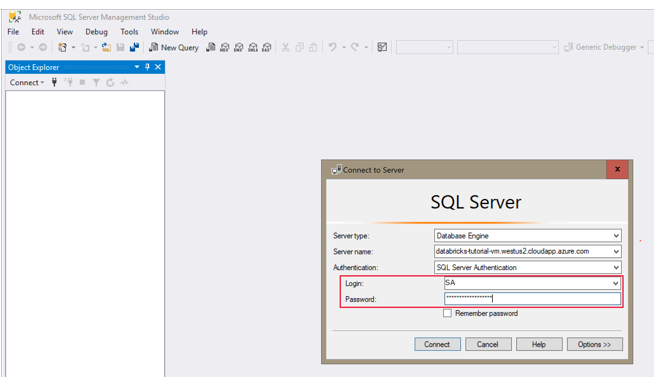


* + Execute **sudo apt-get install docker.io**. Note: if get get, not found, execute this command first: **sudo apt-get update** and then execute the install docker command
  + Execute **sudo docker –version**
  + Execute **sudo docker pull mcr.microsoft.com/mssql/server:2017-latest**
  + Execute **sudo docker images**
  + Execute **sudo docker run -e 'ACCEPT\_EULA=Y' -e 'SA\_PASSWORD=Password1234' -p 1433:1433 --name sql1  -d mcr.microsoft.com/mssql/server:2017-latest --restart=always mcr.microsoft.com/mssql/server:2017-latest**
  + **DO NOT EXECUTE THIS LINE**: ## Execute **sudo docker run --restart=always**
  + Execute **sudo docker ps -a**

The SQL server on your VM has been created and verified

**Create a SQL database**

* + Open SQL Server Management Studio and connect to the server using the server name and SQL Authentication. The sign in username is **SA** and the password is the password set in the Docker command. The password in the example command is Password1234.



Once successfully connected, select **New Query** and enter the following code snippet to create a database, a table, and insert some records in the table.

CREATE DATABASE MYDB;

GO

USE MYDB;

CREATE TABLE states(Name VARCHAR(20), Capitol VARCHAR(20));

INSERT INTO states VALUES ('Delaware','Dover');

INSERT INTO states VALUES ('South Carolina','Columbia');

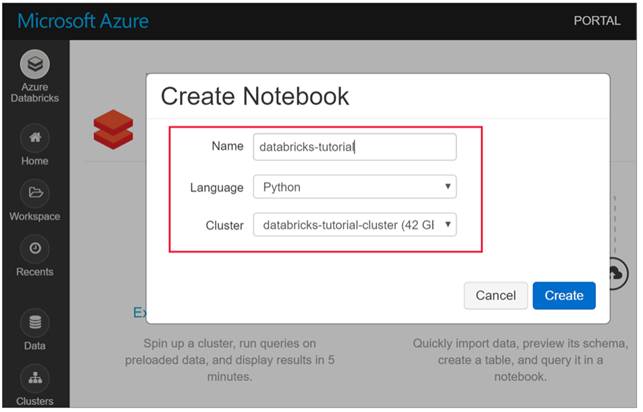
INSERT INTO states VALUES ('Texas','Austin');

SELECT \* FROM states

GO

**Query SQL Server from Azure Databricks**

* + Navigate to your Azure Databricks workspace and verify that you created a cluster as part of the prerequisites. Then, select **Create a Notebook**. Give the notebook a name, select *Python* as the language, and select the cluster you created.



**Once connected, execute the following set of commands**

jdbcHostname = "10.179.64.4"

jdbcDatabase = "MYDB"

userName = 'SA'

password = 'Password1234'

jdbcPort = 1433

jdbcUrl = "jdbc:sqlserver://{0}:{1};database={2};user={3};password={4}".format(jdbcHostname, jdbcPort, jdbcDatabase, userName, password)

df = spark.read.jdbc(url=jdbcUrl, table='states')

display(df)

**The output:**

List of states

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**To connect to Azure Microsoft SQL Database from Databricks or from SQL Server Management Studio, the SQL Server configuration must be STANDARD; Basic does not work!!!**

Databricks Workspace in a Virtual Network

Monday, July 15, 2019

8:11 PM

**3 Reasons to Choose Azure Databricks** From <<https://www.blue-granite.com/blog/3-reasons-to-choose-azure-databricks-for-data-science-and-big-data-workloads>>

**Reason #1: Speed**: It can run up to **100x faster than Hadoop MapReduce** when running in-memory, or up to **10x faster when running on-disk**. **Azure Databricks is even faster!**

**Reason #2: Security:** Azure Databricks integrates directly with Azure Active Directory (AAD) out of the box, with no custom configuration.

**Reason #3: Collaboration**: Azure Databricks provides a platform where data scientists and data engineers can easily share workspaces, clusters and jobs through a single interface. They can also commit their code and artifacts to popular source control tools, like GitHub.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

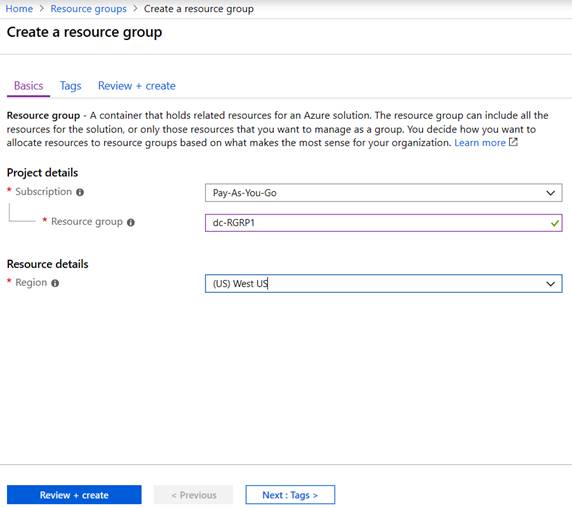
**Brief Overview:**

* + Create a VNET. it will create a default address space and a default subnet
  + Create a second address space with CIDR 16. But don't create any subnets in it.
  + Create a Databricks service
  + Create public subnet and a private subnet for it. Ensure they don't overlap with the default subnet address range or between themselves
  + When created, click on Databricks Workspace (hint: Red bricks)
  + Create a cluster
  + Click on **Spark UI** and then click on **Executors** and you will see newly created nodes in the address range you had created in the VNET

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

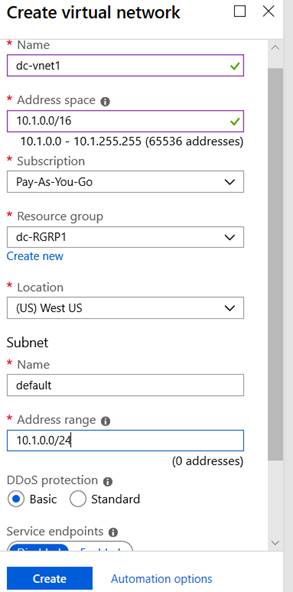
[Databricks workspace in a virtual network](https://docs.microsoft.com/en-us/azure/azure-databricks/quickstart-create-databricks-workspace-vnet-injection).

* + Create a Resource Group

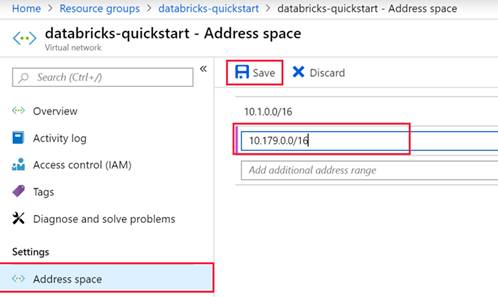


* + **Create a virtual network (VNET)**
    1. In the Azure portal, select **Create a resource** > **Networking** > **Virtual network**.
    2. Under **Create virtual network**, apply the following settings:

|  |  |  |
| --- | --- | --- |
| **Setting** | **Suggested value** | **Description** |
| Name | databricks-quickstart | Select a name for your virtual network. |
| Address space | 10.1.0.0/16 | The virtual network's address range in CIDR notation. |
| Subscription | <Your subscription> | Select the Azure subscription that you want to use. |
| Resource group | databricks-quickstart | Select **Create New** and enter a new resource group name for your account. |
| Location | <Select the region that is closest to your users> | Select a geographic location where you can host your virtual network. Use the location that's closest to your users. |
| Subnet name | default | Select a name for the default subnet in your virtual network. |
| Subnet Address range | 10.1.0.0/24 | The subnet's address range in CIDR notation. It must be contained by the address space of the virtual network. The address range of a subnet which is in use can't be edited. |



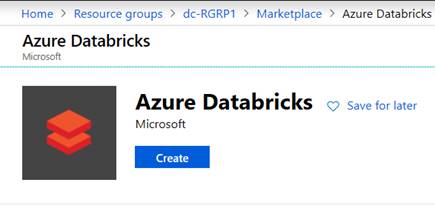
Upon deployment completion, go to your virtual network and select **Address space** under **Settings**. In the box that says *Add additional address range*, insert 10.179.0.0/16 and select **Save**.

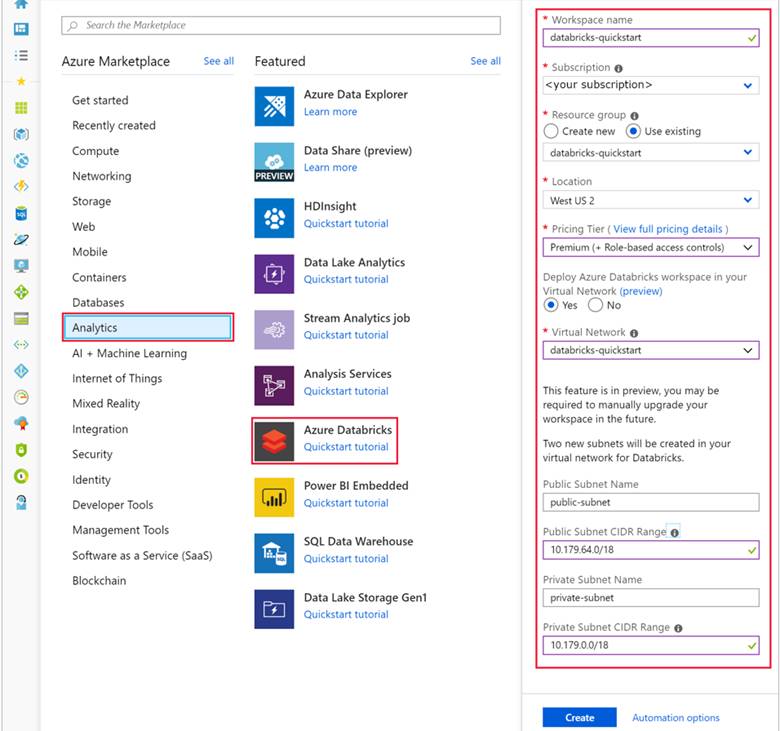


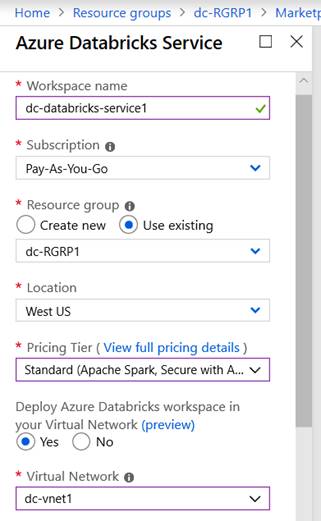
**Create an Azure Databricks workspace**

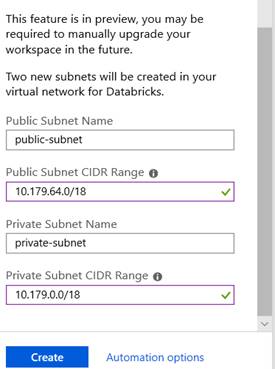
* + In the Azure portal, select **Create a resource** > **Analytics** > **Databricks**.
  + Under **Azure Databricks Service**, apply the following settings:

|  |  |  |
| --- | --- | --- |
| **Setting** | **Suggested value** | **Description** |
| Workspace name | databricks-quickstart | Select a name for your Azure Databricks workspace. |
| Subscription | <Your subscription> | Select the Azure subscription that you want to use. |
| Resource group | databricks-quickstart | Select the same resource group you used for the virtual network. |
| Location | <Select the region that is closest to your users> | Choose the same location as your virtual network. |
| Pricing Tier | Choose between Standard or Premium. | For more information on pricing tiers, see the [Databricks pricing page](https://azure.microsoft.com/pricing/details/databricks/). |
| Deploy Azure Databricks workspace in your Virtual Network | Yes | This setting allows you to deploy an Azure Databricks workspace in your virtual network. |
| Virtual Network | databricks-quickstart | Select the virtual network you created in the previous section. |
| Public Subnet Name | public-subnet | Use the default public subnet name. |
| Public Subnet CIDR Range | 10.179.64.0/18 | CIDR range for this subnet should be between /18 and /26. |
| Private Subnet Name | private-subnet | Use the default private subnet name. |
| Private Subnet CIDR Range | 10.179.0.0/18 | CIDR range for this subnet should be between /18 and /26. |

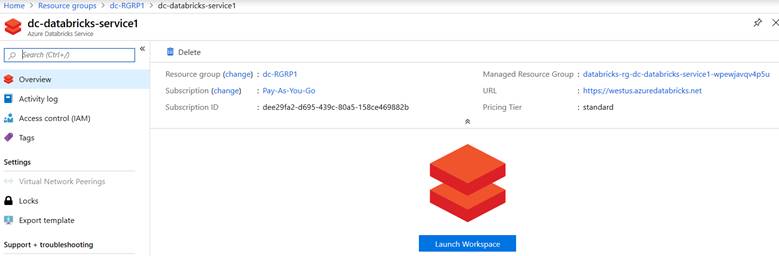




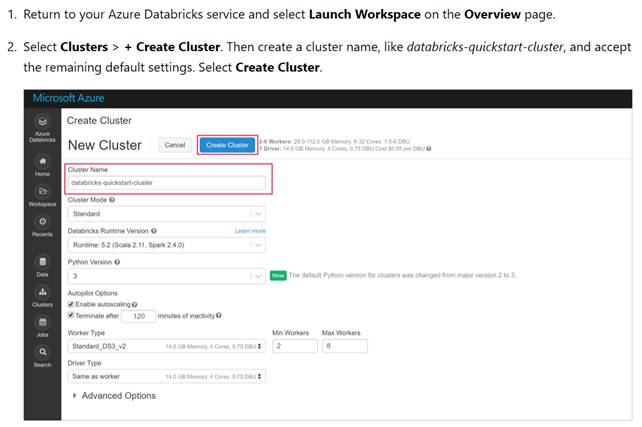


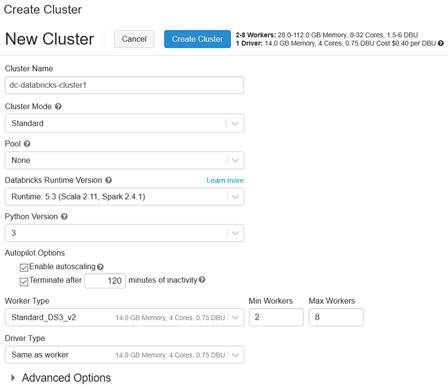


* + Launch Databricks workspace

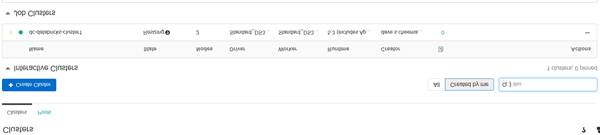


* + Create Databricks Cluster

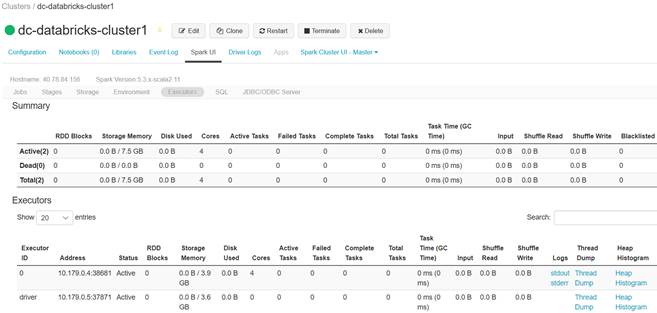




* + Click on Create Cluster



* + Verify results



Databricks - Delta Lake

Thursday, October 24, 2019

5:03 PM

[Delta Lake](https://delta.io/) is an [open source storage layer](https://github.com/delta-io/delta) that brings reliability to data lakes. Delta Lake provides **ACID transactions, scalable metadata handling, and unifies streaming and batch data processing**. Delta Lake runs on top of your existing data lake and is fully compatible with Apache Spark APIs.

Delta Lake on Azure Databricks allows you to configure Delta Lake based on your workload patterns and provides optimized layouts and indexes for fast interactive queries.

This is the documentation for Delta Lake on Azure Databricks.

* + [Introduction to Delta Lake](https://docs.azuredatabricks.net/delta/delta-intro.html)
    - [Quickstart](https://docs.azuredatabricks.net/delta/delta-intro.html#quickstart)
    - [Resources](https://docs.azuredatabricks.net/delta/delta-intro.html#resources)
  + [Introductory Notebooks](https://docs.azuredatabricks.net/delta/intro-notebooks.html)
    - [Delta Lake Quickstart Python notebook](https://docs.azuredatabricks.net/delta/intro-notebooks.html#deltaquickstart-python)
    - [Delta Lake Quickstart Scala notebook](https://docs.azuredatabricks.net/delta/intro-notebooks.html#deltaquickstart-scala)
    - [Delta Lake Quickstart SQL notebook](https://docs.azuredatabricks.net/delta/intro-notebooks.html#deltaquickstart-sql)
  + [Table Batch Reads and Writes](https://docs.azuredatabricks.net/delta/delta-batch.html)
    - [Create a table](https://docs.azuredatabricks.net/delta/delta-batch.html#create-a-table)
    - [Read a table](https://docs.azuredatabricks.net/delta/delta-batch.html#read-a-table)
    - [Write to a table](https://docs.azuredatabricks.net/delta/delta-batch.html#write-to-a-table)
    - [Schema validation](https://docs.azuredatabricks.net/delta/delta-batch.html#schema-validation)
    - [Update table schema](https://docs.azuredatabricks.net/delta/delta-batch.html#update-table-schema)
    - [Replace table schema](https://docs.azuredatabricks.net/delta/delta-batch.html#replace-table-schema)
    - [Views on tables](https://docs.azuredatabricks.net/delta/delta-batch.html#views-on-tables)
    - [Table properties](https://docs.azuredatabricks.net/delta/delta-batch.html#table-properties)
    - [Table metadata](https://docs.azuredatabricks.net/delta/delta-batch.html#table-metadata)
  + [Table Streaming Reads and Writes](https://docs.azuredatabricks.net/delta/delta-streaming.html)
    - [Delta table as a stream source](https://docs.azuredatabricks.net/delta/delta-streaming.html#delta-table-as-a-stream-source)
    - [Delta table as a sink](https://docs.azuredatabricks.net/delta/delta-streaming.html#delta-table-as-a-sink)
  + [Table Deletes, Updates, and Merges](https://docs.azuredatabricks.net/delta/delta-update.html)
    - [Delete from a table](https://docs.azuredatabricks.net/delta/delta-update.html#delete-from-a-table)
    - [Update a table](https://docs.azuredatabricks.net/delta/delta-update.html#update-a-table)
    - [Upsert into a table using Merge](https://docs.azuredatabricks.net/delta/delta-update.html#upsert-into-a-table-using-merge)
    - [Merge examples](https://docs.azuredatabricks.net/delta/delta-update.html#merge-examples)
  + [Table Utility Commands](https://docs.azuredatabricks.net/delta/delta-utility.html)
    - [Vacuum](https://docs.azuredatabricks.net/delta/delta-utility.html#vacuum)
    - [History](https://docs.azuredatabricks.net/delta/delta-utility.html#history)
    - [Convert to Delta](https://docs.azuredatabricks.net/delta/delta-utility.html#convert-to-delta)
  + [Delta Lake API Reference](https://docs.azuredatabricks.net/delta/delta-apidoc.html)
  + [Concurrency Control](https://docs.azuredatabricks.net/delta/concurrency-control.html)
    - [Optimistic concurrency control](https://docs.azuredatabricks.net/delta/concurrency-control.html#optimistic-concurrency-control)
  + [Migrate Workloads to Delta Lake on Azure Databricks](https://docs.azuredatabricks.net/delta/porting.html)
  + [Example](https://docs.azuredatabricks.net/delta/porting.html#example)
    - [Save as Delta table](https://docs.azuredatabricks.net/delta/porting.html#save-as-delta-table)
    - [Convert to Delta table](https://docs.azuredatabricks.net/delta/porting.html#convert-to-delta-table)
  + [Optimizations](https://docs.azuredatabricks.net/delta/optimizations/index.html)
    - [Optimizing Performance with File Management](https://docs.azuredatabricks.net/delta/optimizations/file-mgmt.html)
    - [Auto Optimize](https://docs.azuredatabricks.net/delta/optimizations/auto-optimize.html)
    - [Optimizing Performance with Caching](https://docs.azuredatabricks.net/delta/optimizations/delta-cache.html)
    - [Isolation Levels](https://docs.azuredatabricks.net/delta/optimizations/isolation-level.html)
    - [Replicating MySQL Tables to Delta Tables](https://docs.azuredatabricks.net/delta/optimizations/mysql-delta.html)
    - [Optimize Join Performance](https://docs.azuredatabricks.net/delta/join-performance/index.html)
    - [Optimized Data Transformation](https://docs.azuredatabricks.net/delta/data-transformation/index.html)
    - [Table Versioning](https://docs.azuredatabricks.net/delta/optimizations/versioning.html)
  + [Best Practices](https://docs.azuredatabricks.net/delta/best-practices.html)
    - [Provide data location hints](https://docs.azuredatabricks.net/delta/best-practices.html#provide-data-location-hints)
    - [Choose the right partition column](https://docs.azuredatabricks.net/delta/best-practices.html#choose-the-right-partition-column)
    - [Compact files](https://docs.azuredatabricks.net/delta/best-practices.html#compact-files)
  + [Frequently Asked Questions (FAQ)](https://docs.azuredatabricks.net/delta/delta-faq.html)
    - [What is Delta Lake?](https://docs.azuredatabricks.net/delta/delta-faq.html#what-is-delta)
    - [How is Delta Lake related to Apache Spark?](https://docs.azuredatabricks.net/delta/delta-faq.html#how-is-delta-related-to-as)
    - [What format does Delta Lake use to store data?](https://docs.azuredatabricks.net/delta/delta-faq.html#what-format-does-delta-use-to-store-data)
    - [How can I read and write data with Delta Lake?](https://docs.azuredatabricks.net/delta/delta-faq.html#how-can-i-read-and-write-data-with-delta)
    - [Where does Delta Lake store the data?](https://docs.azuredatabricks.net/delta/delta-faq.html#where-does-delta-store-the-data)
    - [Can I stream data directly into and from Delta tables?](https://docs.azuredatabricks.net/delta/delta-faq.html#can-i-stream-data-directly-into-and-from-delta-tables)
    - [Does Delta Lake support writes or reads using the Spark Streaming DStream API?](https://docs.azuredatabricks.net/delta/delta-faq.html#does-delta-support-writes-or-reads-using-the-spark-streaming-dstream-api)
    - [When I use Delta Lake, will I be able to port my code to other Spark platforms easily?](https://docs.azuredatabricks.net/delta/delta-faq.html#when-i-use-delta-will-i-be-able-to-port-my-code-to-other-spark-platforms-easily)
    - [How do Delta tables compare to Hive SerDe tables?](https://docs.azuredatabricks.net/delta/delta-faq.html#how-do-delta-tables-compare-to-hive-serde-tables)
    - [What DDL and DML features does Delta Lake not support?](https://docs.azuredatabricks.net/delta/delta-faq.html#what-ddl-and-dml-features-does-delta-not-support)
    - [Does Delta Lake support multi-table transactions?](https://docs.azuredatabricks.net/delta/delta-faq.html#does-delta-support-multi-table-transactions)
    - [How can I change the type of a column?](https://docs.azuredatabricks.net/delta/delta-faq.html#how-can-i-change-the-type-of-a-column)
    - [What does it mean that Delta Lake supports multi-cluster writes?](https://docs.azuredatabricks.net/delta/delta-faq.html#what-does-it-mean-that-delta-supports-multi-cluster-writes)
    - [Can I modify a Delta table from different workspaces?](https://docs.azuredatabricks.net/delta/delta-faq.html#can-i-modify-a-delta-table-from-different-workspaces)
    - [Can I access Delta tables outside of Databricks Runtime?](https://docs.azuredatabricks.net/delta/delta-faq.html#can-i-access-delta-tables-outside-of-dbr)
  + [Additional Resources](https://docs.azuredatabricks.net/delta/delta-resources.html)
    - [Blog posts](https://docs.azuredatabricks.net/delta/delta-resources.html#blog-posts)
    - [Talks](https://docs.azuredatabricks.net/delta/delta-resources.html#talks)
    - [Delta Lake transaction log specification](https://docs.azuredatabricks.net/delta/delta-resources.html#delta-transaction-log-specification)

Azure Databricks Sentiment analysis on streaming data

Monday, July 29, 2019

4:59 PM

<https://docs.microsoft.com/en-us/azure/azure-databricks/databricks-sentiment-analysis-cognitive-services>

**Architecture**

Machine generated alternative text:
Azure Databricks 
Producer 
Notebook 
Consumer 
Notebook 
Uses Spark 
Azure 
connector to 
Event Hubs 
Event Hubs 
Spark on Databricks 
A PACHE 
Spork 
Uses Spark 
to process 
Azure 
Cognitive Services 
Use for sentiment 
analysis 

**Ensure Databricks Runtime version is at least 5.5**

Machine generated alternative text:
Clusters / dc-databricks-cluster 
dc-databricks-cluster 
Configuration Notebooks (2) 
Libraries 
Cluster Mode O 
Standard 
Databricks Runtime Version 
5.5 (includes Apache Spark 2_4.3, Scala 211) 
Python Version 
3 
Autopilot Options 
Z] Enable autoscaling 
[7 Edit 
Event Log Spark UI 
Clone 
Driver Logs 
Terminate 
X Delete 
Metrics Apps Spark Cluster UI - Master 
Terminate after 
Worker Type 
Standard DS3 v2 
Driver Type 
Standard DS3 v2 
Advanced Options 
30 
minutes of inactivity 
14.0 Ga Memory, 4 Cores, 
14.0 Ga Memory, 4 Cores, 
Min Workers Max Workers 
0.75 DBL' 
0.75 DBL' 
2 
3 

**Install the following libraries:**

Machine generated alternative text:
Clusters / dcl-databricks-cluster 
O dc 1 -databricks-cluster 
(7 Edit 
Clone 
C Restart 
Terminate 
X Delete 
Configuration 
% Uninstall 
Name 
Notebooks (0) 
Libraries Event Log Spark UI 
% Install New 
Driver Logs 
Metrics 
Apps Spark Cluster UI - Master • 
2. 
Type 
Maven 
Maven 
Status 
Installed 
Installed 
Source 
com_microsoft.azure:azure-eventhubs-spark 

**To do this:**

click on Libraries --> Install New

click on Library Source: Maven

enter com.microsoft.azure:azure-eventhubs-spark\_2.11:2.3.12 in the **Coordinates** field and click Install

Again, enter org.twitter4j:twitter4j-core:4.0.7 in the Coordinates field and click Install

**ENSURE the Scala and Spark versions are compatible in the Databricks Runtime Version during Databricks cluster creation and the installed libraries. For example:**

Machine generated alternative text:
Databricks Runtime Version O 
Runtime 56 211 , spark 2.43) 
Learn 

**And** com.microsoft.azure:azure-eventhubs-spark\_2.11:2.3.12

Send tweets to Event Hubs (**SendTweetsToEventHub)**

|  |
| --- |
| //---------------------------------------------------------------------  //Twitter feed reader and Posting to EventHub  //----------------------------------------------------------------------    import scala.collection.JavaConverters.\_  import com.microsoft.azure.eventhubs.\_  import java.util.concurrent.\_  import scala.collection.immutable.\_  import scala.concurrent.Future  import scala.concurrent.ExecutionContext.Implicits.global    val namespaceName = "dc1-event-hubs1"  val eventHubName = "dc1-event-hub1"  val sasKeyName = "dc1-eh-policy1"  val sasKey = "9GAqLLwoSXgVO/MSHGdWEGQAR1C/d4Q/fgi/SEdfVI8="  val connStr = new ConnectionStringBuilder()  .setNamespaceName(namespaceName)  .setEventHubName(eventHubName)  .setSasKeyName(sasKeyName)  .setSasKey(sasKey)    val pool = Executors.newScheduledThreadPool(1)  val eventHubClient = EventHubClient.create(connStr.toString(), pool)    def sleep(time: Long): Unit = Thread.sleep(time)    def sendEvent(message: String, delay: Long) = {  sleep(delay)  val messageData = EventData.create(message.getBytes("UTF-8"))  eventHubClient.get().send(messageData)  System.out.println("Sent event: " + message + "\n")  }    import twitter4j.\_  import twitter4j.TwitterFactory  import twitter4j.Twitter  import twitter4j.conf.ConfigurationBuilder    // Twitter configuration!  val twitterConsumerKey = "5j6NRxTCMeYeX699HqcaQmnW1"  val twitterConsumerSecret = "OJGV4ZZJZ92q5s0mTDeXKvJEqs1i4JXicXm9onwQUEfZuk1aXB"  val twitterOauthAccessToken = "283119593-rBCTLgENNzPWjj9nB065jWgD9RZlqSSZzAUJ0CHz"  val twitterOauthTokenSecret = "116ErxfGGImxpa1liSkPRDgYVfr1VPRDeHlJq1aIRSBz3"    val cb = new ConfigurationBuilder()  cb.setDebugEnabled(true)  .setOAuthConsumerKey(twitterConsumerKey)  .setOAuthConsumerSecret(twitterConsumerSecret)  .setOAuthAccessToken(twitterOauthAccessToken)  .setOAuthAccessTokenSecret(twitterOauthTokenSecret)    val twitterFactory = new TwitterFactory(cb.build())  val twitter = twitterFactory.getInstance()    // Getting tweets with keywordFilter and sending them to the Event Hub in realtime!  val keywordFilter = {"#Trump"; "#Covid-19"; "#Coronavirus"; "#Biden"}    // Getting tweets with keyword "Azure" and sending them to the Event Hub in realtime!  val query = new Query(keywordFilter)  query.setCount(100)  query.lang("en")  var finished = false  while (!finished) {  val result = twitter.search(query)  val statuses = result.getTweets()  var lowestStatusId = Long.MaxValue  for (status <- statuses.asScala) {  if(!status.isRetweet()){  sendEvent(status.getText(), 500)  }  lowestStatusId = Math.min(status.getId(), lowestStatusId)  }  query.setMaxId(lowestStatusId - 1)  }    // Closing connection to the Event Hub  eventHubClient.get().close() |

Read tweets from Event Hubs (**AnalyzeTweetsFromEventHub**) - a separate notebook

|  |
| --- |
| import org.apache.spark.eventhubs.\_  import org.apache.spark.sql.types.\_  import org.apache.spark.sql.functions.\_    // Build connection string of the EventHub with the above information  //1. Endpoint of the event hub instance you created in EventHubs  //2. setEventHubName in the name of the event hub instance  val connectionString = ConnectionStringBuilder("Endpoint=sb://dc1-event-hubs1.servicebus.windows.net/;SharedAccessKeyName=dc1-eh-policy1;SharedAccessKey=9GAqLLwoSXgVO/MSHGdWEGQAR1C/d4Q/fgi/SEdfVI8=;EntityPath=dc1-event-hub1").setEventHubName("dc1-event-hub1").build    val customEventhubParameters =  EventHubsConf(connectionString)  .setMaxEventsPerTrigger(5)    val incomingStream = spark.readStream.format("eventhubs").options(customEventhubParameters.toMap).load()    incomingStream.printSchema |

|  |
| --- |
| //++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  //**Run sentiment analysis on tweets - Part I**  //++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  import java.io.\_  import java.net.\_  import java.util.\_    case class Language(documents: Array[LanguageDocuments], errors: Array[Any]) extends Serializable  case class LanguageDocuments(id: String, detectedLanguages: Array[DetectedLanguages]) extends Serializable  case class DetectedLanguages(name: String, iso6391Name: String, score: Double) extends Serializable    case class Sentiment(documents: Array[SentimentDocuments], errors: Array[Any]) extends Serializable  case class SentimentDocuments(id: String, score: Double) extends Serializable    case class RequestToTextApi(documents: Array[RequestToTextApiDocument]) extends Serializable  case class RequestToTextApiDocument(id: String, text: String, var language: String = "") extends Serializable    // COMMAND ----------    //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  //**Run sentiment analysis on tweets - Part II**  // Define an object that contains functions to call the Text Analysis API to run language detection and sentiment  // analysis. //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  import javax.net.ssl.HttpsURLConnection  import com.google.gson.Gson  import com.google.gson.GsonBuilder  import com.google.gson.JsonObject  import com.google.gson.JsonParser  import scala.util.parsing.json.\_    object SentimentDetector extends Serializable {    // Cognitive Services API connection settings  // 1. accessKey is the Access Key of the cognitive service instance you created.  // 2. host is the endpoint of the cognitive service instance you created.  val accessKey = "e4415b58187f46b3a134c60e37429991"  val host = "<https://dc1-text-analytics1.cognitiveservices.azure.com/>"  val languagesPath = "/text/analytics/v2.0/languages"  val sentimentPath = "/text/analytics/v2.0/sentiment"  val languagesUrl = new URL(host+languagesPath)  val sentimenUrl = new URL(host+sentimentPath)  val g = new Gson    def getConnection(path: URL): HttpsURLConnection = {  val connection = path.openConnection().asInstanceOf[HttpsURLConnection]  connection.setRequestMethod("POST")  connection.setRequestProperty("Content-Type", "text/json")  connection.setRequestProperty("Ocp-Apim-Subscription-Key", accessKey)  connection.setDoOutput(true)  return connection  }    def prettify (json\_text: String): String = {  val parser = new JsonParser()  val json = parser.parse(json\_text).getAsJsonObject()  val gson = new GsonBuilder().setPrettyPrinting().create()  return gson.toJson(json)  }    // Handles the call to Cognitive Services API.  def processUsingApi(request: RequestToTextApi, path: URL): String = {  val requestToJson = g.toJson(request)  val encoded\_text = requestToJson.getBytes("UTF-8")  val connection = getConnection(path)  val wr = new DataOutputStream(connection.getOutputStream())  wr.write(encoded\_text, 0, encoded\_text.length)  wr.flush()  wr.close()    val response = new StringBuilder()  val in = new BufferedReader(new InputStreamReader(connection.getInputStream()))  var line = in.readLine()  while (line != null) {  response.append(line)  line = in.readLine()  }  in.close()  return response.toString()  }    // Calls the language API for specified documents.  def getLanguage (inputDocs: RequestToTextApi): Option[Language] = {  try {  val response = processUsingApi(inputDocs, languagesUrl)  // In case we need to log the json response somewhere  val niceResponse = prettify(response)  // Deserializing the JSON response from the API into Scala types  val language = g.fromJson(niceResponse, classOf[Language])  if (language.documents(0).detectedLanguages(0).iso6391Name == "(Unknown)")  return None  return Some(language)  } catch {  case e: Exception => return None  }  }    // Calls the sentiment API for specified documents. Needs a language field to be set for each of them.  def getSentiment (inputDocs: RequestToTextApi): Option[Sentiment] = {  try {  val response = processUsingApi(inputDocs, sentimenUrl)  val niceResponse = prettify(response)  // Deserializing the JSON response from the API into Scala types  val sentiment = g.fromJson(niceResponse, classOf[Sentiment])  return Some(sentiment)  } catch {  case e: Exception => return None  }  }  }    // COMMAND ----------    //++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  //**Run sentiment analysis on tweets - Part III**  //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  // User Defined Function for processing content of messages to return their sentiment.  val toSentiment =  udf((textContent: String) =>  {  val inputObject = new RequestToTextApi(Array(new RequestToTextApiDocument(textContent, textContent)))  val detectedLanguage = SentimentDetector.getLanguage(inputObject)  detectedLanguage match {  case Some(language) =>  if(language.documents.size > 0) {  inputObject.documents(0).language = language.documents(0).detectedLanguages(0).iso6391Name  val sentimentDetected = SentimentDetector.getSentiment(inputObject)  sentimentDetected match {  case Some(sentiment) => {  if(sentiment.documents.size > 0) {  sentiment.documents(0).score.toString()  }  else {  "Error happened when getting sentiment: " + sentiment.errors(0).toString  }  }  case None => "Couldn't detect sentiment"  }  }  else {  "Error happened when getting language" + language.errors(0).toString  }  case None => "Couldn't detect language"  }  }  )    // COMMAND ----------    //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  //**Run sentiment analysis on tweets - Part IV**  //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  // Prepare a dataframe with Content and Sentiment columns  val streamingDataFrame = incomingStream.selectExpr("cast (body as string) AS Content").withColumn("Sentiment", toSentiment($"Content"))    // Display the streaming data with the sentiment  streamingDataFrame.writeStream.outputMode("append").format("console").option("truncate", false).start().awaitTermination() |



Azure Databricks with CosmosDB

Monday, July 15, 2019

7:15 PM

**Brief Overview:**

* In Azure portal, create a virtual network
* Create a Databricks service, click Databricks Workspace, create a Databricks cluster of the right Spark version
* Get the Spark connector from the internet (search for Maven repository) and install in the Databricks Workspace
* Download the test data, e.g., NOAA Centers for Environment information
* Create a Cosmos DB service endpoint -->
  + Go to the public-subnet of the Databricks Workspace and under Services, select Microsoft.AzureCosmosDB. Note: you'll find the public-subnet in the Subnets section of the VNET
* Create a Cosmos DB account. Fill out the details and select Core (SQL) for the API
* Be sure to select **Allow access from Azure Portal**
* Create the Cosmos DB.
* Select **Keys** under **Settings** and **copy the primary key**
* Populate some data in the CosmosDB database
  + Use Data Explorer and select New Collection to add a new database and collection to the Cosmos DB account
* Use Azure Cosmos DB Data Migration Tool to upload data into Cosmos DB.
* Enter the Source Information and Target information and enter the connection information, you had copied earlier. Upload Cosmos DB.
* Go to Databricks workspace and create a Databricks nodes cluster
* Load connecters, e.g., CosmosDB Spark connector or other libraries, if not already present
* Create a notebook, e.g., Python notebook
* Setup connection parameters and connect to the CosmosDB.
* Now you're ready to extract your data from CosmosDB into the Databricks workspace for further processing

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Implement Azure Databricks with a Cosmos DB endpoint** <https://docs.microsoft.com/en-us/azure/azure-databricks/service-endpoint-cosmosdb>

Prerequisites

Before you start, do the following:

* Create an [Azure Databricks workspace in a virtual network](https://docs.microsoft.com/en-us/azure/azure-databricks/quickstart-create-databricks-workspace-vnet-injection).
  + **Local created version and documented:** [**Databricks Workspace in a Virtual Network**](onenote:#Databricks%20Workspace%20in%20a%20Virtual%20Network&section-id={03256399-F65F-4D04-ABBB-CFAE428E2977}&page-id={7D6F090B-D088-4F91-9C9C-CF7C9EE4F9B2}&end&base-path=D:\Dave\DC%20-%20Notebook\Technology%20Notes.one)
* Download the [Spark connector](https://search.maven.org/remotecontent?filepath=com/microsoft/azure/azure-cosmosdb-spark_2.4.0_2.11/1.3.4/azure-cosmosdb-spark_2.4.0_2.11-1.3.4-uber.jar).
  + [**https://search.maven.org/artifact/com.microsoft.azure/azure-cosmosdb-spark\_2.4.0\_2.11/1.4.0/jar**](https://search.maven.org/artifact/com.microsoft.azure/azure-cosmosdb-spark_2.4.0_2.11/1.4.0/jar)
* Download sample data from the [NOAA National Centers for Environmental Information](https://www.ncdc.noaa.gov/stormevents/). Select a state or area and select **Search**. On the next page, accept the defaults and select **Search**. Then select **CSV Download** on the left side of the page to download the results.
* Download the [pre-compiled binary](https://aka.ms/csdmtool) of the Azure Cosmos DB Data Migration Tool.

* Machine generated alternative text:
  o 
  dtui 
  dtui.exe.config 
  dtui.pdb 
  dtui 
  EULA 
  6/20/2019 7:32 PM 
  6/20/2019 7:32 PM 
  6/20/2019 7:32 PM 
  6/20/2019 7:32 PM 
  6/20/2019 7:32 PM 
  Application 
  XML Configuration... 
  Program Debug D... 
  XML Document 
  Rich Text Format 
  107 KB 
  7 KB 
  232 KB 
  10 KB 
  68 KB 

Create a Cosmos DB service endpoint

* Once you have deployed an Azure Databricks workspace to a virtual network, navigate to the virtual network in the [Azure portal](https://portal.azure.com/). Notice the public and private subnets that were created through the Databricks deployment.
* Select the *public-subnet* and create a Cosmos DB service endpoint. Then **Save**.

Machine generated alternative text:
public-subnet 
data 

Create a Cosmos DB account

1. Open the Azure portal. On the upper-left side of the screen, select **Create a resource > Databases > Azure Cosmos DB**.
2. Fill out the **Instance Details** on the **Basics** tab with the following settings:

|  |  |
| --- | --- |
| **Setting** | **Value** |
| Subscription | *your subscription* |
| Resource Group | *your resource group* |
| Account Name | db-vnet-service-endpoint |
| API | Core (SQL) |
| Location | West US |
| Geo-Redundancy | Disable |
| Multi-region Writes | Enable |

Machine generated alternative text:
Home 
New 
Azure Marketplace 
Sta rte•d 
Recen reated 
Mobile 
Containers 
see all 
see all 
x Create Azure COS mos DB Account 
Basics Network Tags Review create 
azure 02 is a fully managed globa"y multi-model database service, transparently replicating your data 
number of regions. an elastically "ale thrwghput of 
SQL In stm 
QuickstUt tutorial 
SQL Database 
Quickstet 
SQL Data rehouse 
Quickswt tutorial 
Maria DB 
Couchbase Enterprise Edition (Hourly 
pricing) (preview) 
MysQL 
QuickstUt tutorial 
Quickstet tutorial 
DB 
Quickstut tutorial 
SQL Sen-«2017 Enterp 
2016 
Redis Cache 
Quickstet tutorial 
PRO' LCT 
Sel«t to —ge 
c. 
• Rcount 
0 
Ae foldes to and —ge a" 
A' Machine Learning 
Internet Of Things 
Mixed Rea lity 
Secu rity 
'd entity 
Management Is 
Software as a Service (SaaS) 
Blockehain 

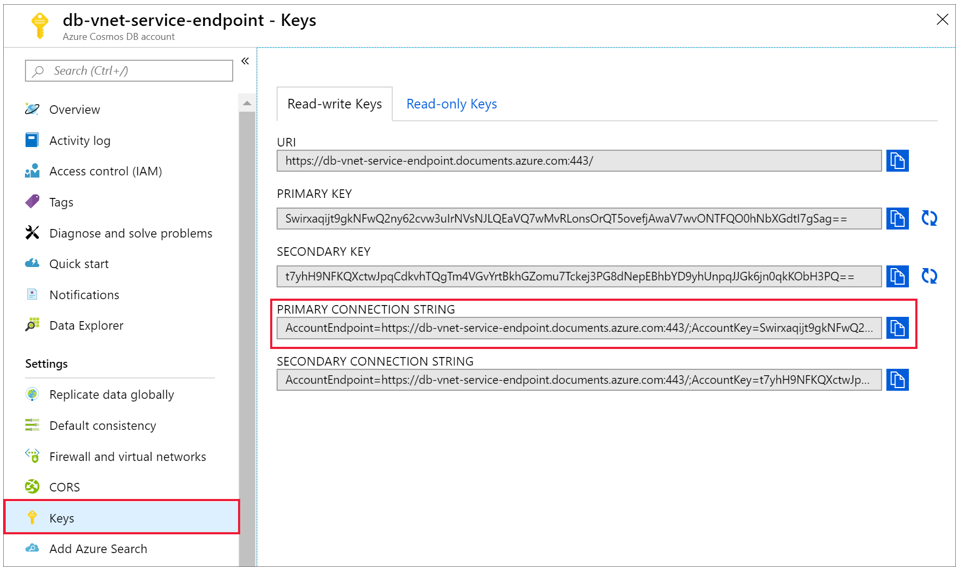
* Select the **Network** tab and configure your virtual network.
  + Choose the virtual network you created as a prerequisite, and then select *public-subnet*. Notice that *private-subnet* has the note *'Microsoft AzureCosmosDB' endpoint is missing'*. This is because you only enabled the Cosmos DB service endpoint on the *public-subnet*.

* Ensure you have **Allow access from Azure portal** enabled. This setting allows you to access your Cosmos DB account from the Azure portal. If this option is set to **Deny**, you will receive errors when attempting to access your account.

Machine generated alternative text:
Create Azure Cosmos DB Account 
Basics Network Tags 
CONFIGURE VIRTUAL NETWORKS 
Virtual Network O 
Subnet O 
CONFIGURE FIREWALL 
Review + create 
data bricks-vnet-injection-service-endpoint 
Create a new virtual network 
public-subnet 
Allow access from Azure Portal O 
Allow access from my IP (75.167.33.214) O 
Allow 
Allow 
Deny 
Deny 

Machine generated alternative text:
Review + create 
Previous 
Next: Tags 

* Select **Review + Create**, and then **Create** to create your Cosmos DB account inside the virtual network.
* Once your Cosmos DB account has been created, navigate to **Keys** under **Settings**. Copy the primary connection string and save it in a text editor for later use.



* Select **Data Explorer** and **New Collection** to add a new database and collection to your Cosmos DB account.

Machine generated alternative text:
Home > db-vnet-service-endpoint - Data Explorer 
db-vnet-service-endpoint - Data Explorer 
Azure Cosmos DB account 
p Search (Ctrl* D 
Overview 
Activity log 
Access control (IAM) 
Tags 
Diagnose and solve problems 
Quick start 
Notifications 
Data Explorer 
Settings 
Replicate data globally 
Default consistency 
Firewall and virtual networks 
CORS 
Keys 
Add Azure Search 
Add Azure Function 
Locks 
e 
New Collection 
SQL API 
Add Collection 
* Database id O 
@ Create new 
O 
Use existing 
Storm 
o 
Provision database throughput O 
* Collection Id O 
StormC011ection 
Where did 'fixed' collections go? O 
* Partition key O 
/event_id 
* Throughput goo - RI-vs) O 
1000 
Estimated spend (USD): SO.080 hourly / $1.92 daily. 
Unique keys O 
+ Add unique key 

Machine generated alternative text:
Export template 
Collections 
Browse 
Scale 
Settings 
Document Explorer 
Query Explorer 
Script Explorer 
o 

Upload data to Cosmos DB

1. Open the graphical interface version of the [data migration tool for Cosmos DB](https://aka.ms/csdmtool), **Dtui.exe**.

Machine generated alternative text:
Azure Cosmos DB Data Migration Tool 
x 
Welcome 
Welcome 
Source Information 
Target Information 
Advanced 
Summary 
Results 
o 
o 
Azure Cosmos DB Data Migration Tool 
IJse the Azure Cosmos DB data migration tool to import data to Cosmos DB from a variety of sources. The 
tool can not only help scope certain source data via queries (e.g. SQL Server, MongoDB), but can also 
transform tabular data (e.g. CSV file, SQL Server) into hierarchical structures. 
For help using the tool, please see the online documentation located here. 
Feel free to suggest and vote for new features by posting here. 



* On the **Source Information** tab, select **CSV File(s)** in the **Import from** dropdown. Then select **Add Files** and add the storm data CSV you downloaded as a prerequisite.

Machine generated alternative text:
ure Cosmos DB Data Migration Tool 
Source Information 
Welcome 
Source Information 
Target Information O 
Advanced 
Summary 
Results 
Specify source information 
1m rt from: 
CSV File(s) 
Add Files 
Add Folder Add IJRL(s) Add BLOB(s) Remove 
Nesting Separator O 
C] Trim quoted values 
[2] Treat unquoted NULL as string 
C] Use regional format settings 
C] Decompress data 

On the **Target Information** tab, input your connection string. The connection string format is AccountEndpoint=*https://dc-cosmos-db-acct.documents.azure.com:443/*;AccountKey=*Fx2QYfXimDJUyw1aycBQwoFmm3nYkYnVLpyV7wzqTzd5exaj3ibEfrCudtnWW94sOGKNNF2qakCezb50fJvGQQ==*;Database=*Strom*. The AccountEndpoint and AccountKey are included in the primary connection string you saved in the previous section. Append Database=*Storm* to the end of the connection string, and select **Verify**. Then, add the Collection name and partition key.

Machine generated alternative text:
Azure Cosmos DB Data Migration Tool 
Target Information 
Welcome 
Source Information 
Target Information 
Advanced 
Summary 
Results 
Specify target information 
Export to: 
Azure Cosmos DB - Sequential record import (partitioned collection) 
•NVsNJLQEaVQ7wMvRLonsOrQT50vefjAwaV7mONTFQOOhNbXGdt17gSag 
Collection 
StormCoIlection 
Partition Key O 
/event_id 
Collection Throughput 
1000 
Id Field 
v Advanced Options 

Select **Next** until you get to the Summary page. Then, select **Import**.

Machine generated alternative text:
Create a cluster and add library 
1. Navigate to your Azure Databricks service in the Azure portal and select Launch Workspace. 
w i njection 
rags 
ImpM 

* Create a new cluster. Choose a Cluster Name and accept the remaining default settings.

Machine generated alternative text:
Microsoft Azure 
Create Cluster 
New Cluster 
Cluster Name 
cosmos -d b-CO n 
Cluster O 
Sta rd 
Oatabricks Runtime Version O 
Runtime: 5.2 (Scala 2.11, Spark 2.40) 
a.s 8-32 
Create Cluster 
1 Driver: 140 4 075 0 
Learn more 
The default python version for clusters was changed from major version 2 to 3, 
Python Version O 
Autopilot Options 
•J Enable autoscalingo 
Termi nate after 120 
Worker Type 
Driver Type 
Same as worker 
Advanced Options 
minutes Of inactivity O 
Min 
l.voc;a • 
Max Workers 

* After your cluster is created, navigate to the cluster page and select the **Libraries** tab. Select **Install New** and upload the Spark connector jar file to install the library.

Machine generated alternative text:
Microsoft Azure 
Clusters I cosmos-db-connect' 
Install Library 
cosmos-db-co 
Configuration 
% Uninstall 
C] Name 
Libra Source 
Notebooks (O) 
upload DBFS 
Install New 
Libra Type 
Jar Python Egg 
cosmosdb- 
spark_2.4 
I .3.4•uber.jar 
Cancel upload 
pypl 
Maven 
CRAN 
Workspace 
Python Whl 
Cancel 
PORTAL 
Install 

You can verify that the library was installed on the **Libraries** tab.

Machine generated alternative text:
Microsoft Azure 
Clusters / cosmos-db-connection 
cosmos-db-connection 
[7 Edit Clone C Restart 
Terminate 
PORTAL 
X Delete 
Configuration Notebooks (O) 
Libraries 
Event Log Spark UI Driver Logs Spark Cluster UI - Master • 
% Uninstall 
Install New 
O 
o 
Name 
azure 
Type 
JAR 
Status 
Installed 
Source 

Query Cosmos DB from a Databricks notebook

1. Navigate to your Azure Databricks workspace and create a new python notebook.

Machine generated alternative text:
Microsoft ARI' re 
Azure Dat 
Explore the Quickstart Tutorial 
Create Notebook 
datab 
python 
Spin up a cluster. run on preloaded data. and display 
results in S minutes. 
Common Tasks 
New NO t ebOOk 
upload Data 
Create Table 
New Cluster 
New Job 
New MLfIow Experiment 
Can 
Ouicny data. "s scnc«na. create a 
query it in a notebook. 
Recents 
Qui dstart Nat ebook 
ane 
PORTAL 
Create a Blank Notebook 
Create a naterx»k start querying visuali 
rnceoling data. 
Documentation 
DatabriC*S Guide 
python , R. Scala. SQL 
Importing Data 

Run the following python code to set the Cosmos DB connection configuration. Change the **Endpoint**, **Masterkey**, **Database**, and **Collection** accordingly.

|  |
| --- |
| **connectionConfig** = {    "Endpoint" : "[*https://dc-cosmos-db-acct.documents.azure.com:443/*](https://dc-cosmos-db-acct.documents.azure.com/)",    "Masterkey" : "*Fx2QYfXimDJUyw1aycBQwoFmm3nYkYnVLpyV7wzqTzd5exaj3ibEfrCudtnWW94sOGKNNF2qakCezb50fJvGQQ==*",    "Database" : "*Storm*",    "preferredRegions" : "*West US*",    "Collection": "*StormCollection*",    "schema\_samplesize" : "1000",    "query\_pagesize" : "200000",    "query\_custom" : "SELECT \* FROM c"  }    users = spark.read.format("com.microsoft.azure.cosmosdb.spark").options(\*\*connectionConfig).load()  users.createOrReplaceTempView("storm")    %sql  select \* from storm |

Machine generated alternative text:
Instead of New Container, 
Choose New Container 

Azure Databricks - ETL, ADLS Gen2

Saturday, July 20, 2019

12:40 PM

**Brief Overview:**

**Architecture**

Machine generated alternative text:
Logs, files, and media 
(unstructured) 
Store 
Azure Data Lake 
Storage Gen2 
Prep & Train 
Azure 
(Spark) 
Model & Serve 
Azure SQL Data 
Warehouse 

* + Various data sources --> Azure Data Lake Storage Gen2 --> Databricks --> SQL Data Warehouse

* + Create a SQL data warehouse, use Gen2 DW500c for the storage
  + Create a server-level IP firewall rule - add client IP in start and end IP's
  + Connect to the server as server admin
  + Create a masterkey for the SQL Data Warehouse
  + Create a Blob Storage account
  + Create an Azure Data Lake Storage Gen2 account
  + Create a service principal
  + Register the application in Azure AD and assign it a role
  + Copy and save values for the tenant id, app id, and password

* + Create a container
  + Upload a block blob
  + Create an Azure Active Directory application in Azure AD. Choose Accounts in this organizational directory only (default directory)
  + Create a Redirect URI
  + Assign the application to the role of **Storage Blob Data Contributor**

* + Create a file system in the Azure Data Lake Storage Gen2

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

* + In the Databricks cluster, create a SCALA notebook

* + In the SCALA notebook, do the following set of steps:
    - Set default service principal credentials for any ADLS Gen 2 account accessed in the Spark session.
    - Append the account name to the setting to specify credentials for a specific ADLS Gen 2 account.
    - Ingest source (e.g., sample) data into the Azure Data Lake Storage Gen2 account. Using dbutils.fs.cp command, copy source data into the Azure Data Lake Storage Gen2 account.
    - Load the source (sample) json file as a data frame in Azure Databricks.
    - You can view the contents of the data frame using df.show() command
    - Extract only the columns you need, e.g., firstName**,** lastName**,** gender**,** location**,** and level from the dataframe
    - You can further transform the extracted data. For example, rename the column level to subscription\_type using specificColumnsDf.withColumnRenamed command
    - Load data into Azure SQL Data Warehouse
      * Use the Azure SQL Data Warehouse connector for Azure Databricks to directly upload a dataframe as a table in a SQL data warehouse. The SQL Data Warehouse connector uses Azure Blob storage as temporary storage to upload data between Azure Databricks and Azure SQL Data Warehouse.
      * Provide the configuration to access the Azure Storage (Blob) account from Azure Databricks
      * Specify a temporary folder to use while moving data between Azure Databricks and Azure SQL Data Warehouse
      * Store Azure Blob storage access keys in the configuration using *sc.hadoopConfiguration.set* command. This ensures that you won’t have to keep the keys in the notebook.
      * Provide the parameters to connect to the Azure SQL Data Warehouse instance.
      * Load the transformed dataframe, **renamedColumnsDF**, as a table in a SQL data warehouse, e.g., SampleTable
    - You can login to SQL Datawarehouse to see the results

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

<https://docs.microsoft.com/en-us/azure/azure-databricks/databricks-extract-load-sql-data-warehouse>

* + ETL (extract, transform, and load data) operation by using Azure Databricks. You extract data from Azure Data Lake Storage Gen2 into Azure Databricks, run transformations on the data in Azure Databricks, and load the transformed data into Azure SQL Data Warehouse  
    Prerequisites
    - Create an Azure SQL data warehouse, create a server-level firewall rule, and connect to the server as a server admin.
      * [Create and query an Azure SQL data warehouse in the Azure portal](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/create-data-warehouse-portal).
    - Create a database master key for the Azure SQL data warehouse. See [Create a database master key](https://docs.microsoft.com/sql/relational-databases/security/encryption/create-a-database-master-key).
    - Create an Azure Blob storage account, and a container within it. Also, retrieve the access key to access the storage account. See [Quickstart: Upload, download, and list blobs with the Azure portal](https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal).
    - Create an Azure Data Lake Storage Gen2 storage account. See [Quickstart: Create an Azure Data Lake Storage Gen2 storage account](https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-quickstart-create-account).
    - Create a service principal. See [How to: Use the portal to create an Azure AD application and service principal that can access resources](https://docs.microsoft.com/azure/active-directory/develop/howto-create-service-principal-portal).  
      There's a couple of specific things that you'll have to do as you perform the steps in that article.
    - When performing the steps in the [Assign the application to a role](https://docs.microsoft.com/azure/active-directory/develop/howto-create-service-principal-portal#assign-the-application-to-a-role) section of the article, make sure to assign the **Storage Blob Data Contributor** role to the service principal in the scope of the Data Lake Storage Gen2 account. If you assign the role to the parent resource group or subscription, you'll receive permissions-related errors until those role assignments propagate to the storage account.  
      If you'd prefer to use an access control list (ACL) to associate the service principal with a specific file or directory, reference [Access control in Azure Data Lake Storage Gen2](https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-access-control).
    - When performing the steps in the [Get values for signing in](https://docs.microsoft.com/azure/active-directory/develop/howto-create-service-principal-portal#get-values-for-signing-in) section of the article, paste the tenant ID, app ID, and password values into a text file. You'll need those soon.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Create a data warehouse

An Azure SQL data warehouse is created with a defined set of [compute resources](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/memory-and-concurrency-limits). The database is created within an [Azure resource group](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview) and in an [Azure SQL logical server](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-logical-servers).

Follow these steps to create a SQL data warehouse that contains the AdventureWorksDW sample data.

* + Click **Create a resource** in the upper left-hand corner of the Azure portal.
  + Select **Databases** from the **New** page, and select **SQL Data Warehouse** under **Featured** on the **New** page.

Machine generated alternative text:
SQL Data Warehouse 
Microsoft 
SQL Data Warehouse 
M icrosoft 
Create 

* + New server settings:

Machine generated alternative text:
New server 
Microsoft 
* Server name 
dcl -datawarehouse-server 
.database.windows.net 
* Server admin login 
sql-datawarehouse-admin 
* Password 
* Confirm password 
* Location 
(US) West US 
Allow Azure services to access server O 

* + SQL Data Warehouse Settings:

Machine generated alternative text:
SQL Data Warehouse 
Microsoft 
* Basics * Additional settings Tags Review + create 
Create a SQL data warehouse with your preferred configurations. Complete the Basics tab then go to Review + Create to 
provision with smart defaults, or visit each tab to customize. Learn more 
Project details 
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and 
manage all your resources. 
* Subscription O 
* Resource group O 
Data warehouse details 
Pay-As-You-Go 
(New) dc-RGRPI 
Create new 
Enter required settings for this data warehouse, including picking a logical server and configuring the performance level. 
* Data warehouse name 
* Server O 
Review + create 
dcl -datawarehouse 
(new) dcl-datawarehouse-server (LIS) West US) 
Create new 
Next : Additional settings > 

Set the Gen2 Performance level before you click Create.

Machine generated alternative text:
* Performance level O 
Gen2 
DW500c 
Select performance level 

Machine generated alternative text:
SQL Data Warehouse 
Microsoft 
SQL Data Warehouse 
by Microsoft 
Terms of use I Privacy policy 
terms 
Est. Cost Per Hour 
7.55 USD 
View pricing details 
By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) 
authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as 
my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of 
the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For 
additional details see Azure Marketplace Terms. 
Basics 
Subscription 
Resource group 
Region 
Data warehouse name 
Server 
Performance level 
Additional settings 
Use existing data 
Collation 
Create 
< Previous 
Pay-As-You-Go 
(new) dc-RGRPI 
westus 
dcl-datawarehouse 
(new) dcl -datawarehouse-server 
Gen2: DW500c 
Blank 
Cl AS 
Download a template for automation 

* + Click Create.
  + Create Firewall settings:

Machine generated alternative text:
From database overview page 
1. To set a server-level IP firewall rule from the database overview page, click Set server firewall on the 
toolbar as shown in the following image: The Firewall settings page for the SQL Database server opens. 
Microsoft Azure 
SQ' databases 
g (change) 
East 
Subscr i ption (change) 
Subscription 10 
Tags (change) 
Click to togs 
Resource utilization 
senaes, and docs 
mySampreDatabase 
Firewall 
name 
ewservet •20181208.0% ndOwS. 
pCOl 
No elastic pc.' 
ngs 
Shcøa database ccmection strings 
pricing tie. 
standard so: 10 0TUs 
Oldest restore point 
2018-11-28 urc 
Firewall settings 
o 
Comecticns the prnides to databases in 
CE«t IP 
73.42207 
7342247 
from the VNEVSubnet provdes access to databases in 
Add virtual 
Create new 
vmun— ADDnss— swscxm— S— 

* + Click **Add client IP** on the toolbar to add the IP address of the computer you are currently using and then click **Save**. A server-level IP firewall rule is created for your current IP address.



From server overview page

The overview page for your server opens, showing you the fully qualified server name (such as **mynewserver20170403.database.windows.net**) and provides options for further configuration.

* + To set a server-level rule from server overview page, click **Firewall** in the left-hand menu under Settings:
  + Click **Add client IP** on the toolbar to add the IP address of the computer you are currently using and then click **Save**. A server-level IP firewall rule is created for your current IP address.

Connect to the server as server admin

This section uses [SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms) (SSMS) to establish a connection to your Azure SQL server.

* + Open SQL Server Management Studio. *OR you may want to use Azure Data Studio that you need to install to your local machine.*
  + In the **Connect to Server** dialog box, enter the following information:

|  |  |  |
| --- | --- | --- |
| **Setting** | **Suggested value** | **Description** |
| Server type | Database engine | This value is required |
| Server name | The fully qualified server name | Here's an example: **mynewserver-20180430.database.windows.net**. |
| Authentication | SQL Server Authentication | SQL Authentication is the only authentication type that is configured in this tutorial. |
| Login | The server admin account | Account that you specified when you created the server. |
| Password | The password for your server admin account | Password that you specified when you created the server. |

Machine generated alternative text:
DC Microsoft SQL Server Management Studio 
Edit View Debug Tools Window Help 
New Query 
Object Explorer 
Connect • 
Connect to Server 
Server type 
Server 
Ready 
Quick Launch (Ctrl*Q) 
x 
x 
SQL Server 
711130 windows ret 
SQL Server Atherticabon 
Remember password 
Options 

* + Click **Connect**. The Object Explorer window opens in SSMS.
  + In Object Explorer, expand **Databases**. Then expand **mySampleDatabase** to view the objects in your new database.

To create a database master key

* + Choose a password for encrypting the copy of the master key that will be stored in the database.
  + In **Object Explorer**, connect to an instance of Database Engine.
  + On the Standard bar, click **New Query**.
  + Copy and paste the following example into the query window and click **Execute**.

-- Creates a database master key for the "AdventureWorks2012" database.

-- The key is encrypted using the password "23987hxJ#KL95234nl0zBe."

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Toaster2010!';

GO

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

[**Upload, download, and list blobs with the Azure portal**](https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal)

Prerequisites

* + Create a blob storage account.
  + For help creating the account, see [Create a storage account](https://docs.microsoft.com/en-us/azure/storage/common/storage-quickstart-create-account).

Machine generated alternative text:
Create storage account 
your resources. 
* Subscription 
* Resource group 
Instance details 
Pay-As-You-Go 
dc-RGRPI 
Create new 
The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using 
the classic deployment model instead. Choose classic deployment model 
* Storage account name O 
* Location 
Performance O 
Account kind O 
Replication O 
Access tier (default) O 
Review + create 
dclstorageacct 
(LIS) West US 
O 
• Standard 
Premium 
StorageV2 (general purpose v2) 
Read-access geo-redundant storage (RA-GRS) 
Cool • Hot 
< Previous 
Next : Advanced > 

Create a container

To create a container in the Azure portal, follow these steps:

* + Navigate to your new storage account in the Azure portal.
  + In the left menu for the storage account, scroll to the **Blob service** section, then select **Blobs**.
  + Select the **+ Container** button.
  + Type a name for your new container. The container name must be lowercase, must start with a letter or number, and can include only letters, numbers, and the dash (-) character.
  + Set the level of **public access** to the container. The default level is **Private (no anonymous access)**.
  + Select **OK** to create the container.

Machine generated alternative text:
storeageexample - Blobs 
Blob service 
Blobs 
Custom domain 
Soft de ete 
Azure CDN 
Add Azure Search 
Lifecycle Management (previ... 
x 
+ Container 
Refresh 
New container 
Name 
container-example 
Public access level O 
Private (no anonymous access) 
Delete 

Upload a block blob

Block blobs consist of blocks of data assembled to make a blob. Most scenarios using Blob storage employ block blobs. Block blobs are ideal for storing text and binary data in the cloud, like files, images, and videos. This quickstart shows how to work with block blobs.

To upload a block blob to your new container in the Azure portal, follow these steps:

* + In the Azure portal, navigate to the container you created in the previous section.
  + Select the container to show a list of blobs it contains. Since this container is new, it won't yet contain any blobs.
  + Select the **Upload** button to upload a blob to the container.
  + Browse your local file system to find a file to upload as a block blob, and select **Upload**.

Machine generated alternative text:
Upload blob 
container-example/ 
Files O 
Authentication type O 
OAuth (preview) 
SAS 
Overwrite if files already exist 
v Advanced 
Upload 
x 

You can easily download a blob when necessary

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

* + **Create a service principal**. [Use the portal to create an Azure AD application and service principal that can access resources](https://docs.microsoft.com/azure/active-directory/develop/howto-create-service-principal-portal).

Create Application Registration in the Azure Active Directory

* + Select **Azure Active Directory**.
  + Select **App registrations**.
  + Select **New registration**.
  + Name the application. Select a supported account type, which determines who can use the application. Under **Redirect URI**, select **Web** for the type of application you want to create. Enter the URI where the access token is sent to. You can't create credentials for a [Native application](https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/application-proxy-configure-native-client-application). You can't use that type for an automated application. After setting the values, select **Register**.

Machine generated alternative text:
1) Create an Active directory Service Principle (App Registration). Get the application ID, Key and directory ID (tenant ID) for the active directory. 
M i æft • mp«tiø 

Machine generated alternative text:
2) For the service principle grant "delegated Permissions" to Azure Data Lake 

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Steps below are critical**

Create a file system in the Azure Data Lake Storage Gen2 account

Machine generated alternative text:
3) Create a storage account as "StorageV2 (general purpose v2). Also known as Azure Data Lake Store Gen2, however it is created under Storage Accounts. 

Machine generated alternative text:
4) In the storage account for IAM, add the service principle as 'l Reader", "Contributor" and "Storage Account Contributor" roles. Below wpcADD1 is my service principle. 

Machine generated alternative text:
5) In the storage account create a "File system" 
• Vile 

Machine generated alternative text:
Home > Resource groups > dcl-rg > dc1adlsgen2account I Containers > 
dcl-adlsgen-containerl I Access Control (IAM) 
Container 
P Search (Ctrl+/) 
Overview 
Access Control (IAM) 
Settings 
Access policy 
Properties 
Metadata 
+ Add Download role assignments 
Edit columns Refresh 
X Remove 
Check access Role assignments Roles Deny assignments Classic administrators 
Number of role assignments for this subscription 
3 
Name Q) 
Search by name or email 
2 items (2 Service Principals) 
Name 
Contribu tor 
Type G) 
2000 
Type 
Role G) 
2 selected 
Microsoft-AzureML-Support-App-785d4e App 
Storage Blob Data Contributor 
DcETLGen2 
App 
C) Got feedback? 
Scope G) 
All scopes 
Role 
Contributor Q) 
Storage 810b Data Contributor Q) 
Group by G) 
Role 
Scope 
Subscription (Inherited) 
This resource 

Machine generated alternative text:
7) In Azure Storage Explorer you can sign in and drill down to the file system (blob container) that you created in the portal. Right mouse click for manage access 
x 

Machine generated alternative text:
8) For Manage Access click the Add button and enter the Object ID and set the Read\Write\Execute Access and default permissions. Click Save. 
Manage Access 

Create an Azure Databricks Cluster

Machine generated alternative text:
Clusters / dcl-db-clusterl 
dcl-db-clusterl 
Cluster Name 
dcl -db-clusterll 
Cluster Mode O 
Standard 
Pool O 
None 
Databricks Runtime Version O 
Runtime: TO (Scala 2.12, spark 3.0 0) 
Cancel 
2-8 workers: 28.0-1120 GB Memory, 8-32 cores, 1.5-6 DBL' 
Confirm and Resize 
1 Driver: 14.0 Ga Memory, 4 cores, 0.75 Dau O 
Learn more 
This Runtime version supports only Python 3 
New 
Autopilot Options 
O Enable autoscaling O 
O Terminate after 
30 
Worker Type O 
Standard DS3 v2 
Driver Type 
Standard DS3 v2 
Advanced Options 
minutes of inactivity O 
14.0 Memory, 4 cores, 0.75 v 
14.0 Memory, 4 cores, 0.75 v 
Min Workers 
2 
Max Workers 
8 

Create an Azure Databricks service

In this section, you create an Azure Databricks service by using the Azure portal.

* + In the Azure portal, select **Create a resource** > **Analytics** > **Azure Databricks**.
    - Create an [Azure Databricks workspace in a virtual network](https://docs.microsoft.com/en-us/azure/azure-databricks/quickstart-create-databricks-workspace-vnet-injection).
      * **Local created version and documented:** [**Databricks Workspace in a Virtual Network**](onenote:#Databricks%20Workspace%20in%20a%20Virtual%20Network&section-id={03256399-F65F-4D04-ABBB-CFAE428E2977}&page-id={7D6F090B-D088-4F91-9C9C-CF7C9EE4F9B2}&end&base-path=D:\Dave\DC%20-%20Notebook\Technology%20Notes.one)
      * **Be sure to use Spark Runtime:** 6.1 (includes Apache Spark 2.4.4, Scala 2.11)

In this section, you create a notebook in Azure Databricks workspace and then run code snippets to configure the storage account

* + In the [Azure portal](https://portal.azure.com/), go to the Azure Databricks service that you created, and select **Launch Workspace**.
  + On the left, select **Workspace**. From the **Workspace** drop-down, select **Create** > **Notebook**.

Machine generated alternative text:
Microsoft Azure 
workspace 
? Documentation 
Release Notes 
Datuu:ks 
Honk 
Workspace 
Create 
Import 
Export 
Notebook 
Library 
Folder 
Training & Tutorials 
Permissions 
e snared 
Users 

In the **Create Notebook** dialog box, enter a name for the notebook. Select **Scala** as the language, and then select the Spark cluster that you created earlier.

Machine generated alternative text:
In the Create Notebook dialog box, enter a name for the notebook. Seli 
select the Spark cluster that you created earlier. 
Create Notebook 
Name 
Language 
Cluster 
samplenotebook 
Scala 
mysparkcluster (42 GB, Running 
cancel 

* + Select **Create**.

* + Start a Databricks cluster and create a SCALA notebook.

* + Use the Scala Notebook code (working) below:

|  |
| --- |
| // Databricks notebook source  val appID = "dbe739d3-f17f-4130-a7d9-45cea942b84d"  val secret = "o~1\_v-gySAG4fD\_7P32Oek5we4NEWZYn1i"  val tenantID = "6bdf9621-fbe5-401d-9c2e-8c9ed7555112"    spark.conf.set("fs.azure.account.auth.type", "OAuth")  spark.conf.set("fs.azure.account.oauth.provider.type", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")  spark.conf.set("fs.azure.account.oauth2.client.id", appID)  spark.conf.set("fs.azure.account.oauth2.client.secret", secret)  spark.conf.set("fs.azure.account.oauth2.client.endpoint", "<https://login.microsoftonline.com/>" + tenantID + "/oauth2/token")  spark.conf.set("fs.azure.createRemoteFileSystemDuringInitialization", "true")    // COMMAND ----------    //Ensure Service Principal of the ADLS Gen2 Storage Account has **Storage Blob Data Contributor** role added.    val storageAccountName = "dc1adlsgen2account"  val fileSystemName = "dc1-adlsgen-container1"    spark.conf.set("fs.azure.account.auth.type." + storageAccountName + ".dfs.core.windows.net", "OAuth")  spark.conf.set("fs.azure.account.oauth.provider.type." + storageAccountName + ".dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")  spark.conf.set("fs.azure.account.oauth2.client.id." + storageAccountName + ".dfs.core.windows.net", "" + appID + "")  spark.conf.set("fs.azure.account.oauth2.client.secret." + storageAccountName + ".dfs.core.windows.net", "" + secret + "")  spark.conf.set("fs.azure.account.oauth2.client.endpoint." + storageAccountName + ".dfs.core.windows.net", "<https://login.microsoftonline.com/>" + tenantID + "/oauth2/token")  spark.conf.set("fs.azure.createRemoteFileSystemDuringInitialization", "true")  dbutils.fs.ls("abfss://" + fileSystemName + "@" + storageAccountName + ".dfs.core.windows.net/")  spark.conf.set("fs.azure.createRemoteFileSystemDuringInitialization", "false")    // COMMAND ----------    // MAGIC %sh wget -P /tmp <https://raw.githubusercontent.com/Azure/usql/master/Examples/Samples/Data/json/radiowebsite/small_radio_json.json>    // COMMAND ----------    dbutils.fs.cp("file:///tmp/small\_radio\_json.json", "abfss://" + fileSystemName + "@" + storageAccountName + ".dfs.core.windows.net/")    // COMMAND ----------    val df = spark.read.json("abfss://" + fileSystemName + "@" + storageAccountName + ".dfs.core.windows.net/small\_radio\_json.json")    // COMMAND ----------    df.show()    // COMMAND ----------    val specificColumnsDf = df.select("firstname", "lastname", "gender", "location", "level")  specificColumnsDf.show()    // COMMAND ----------    val renamedColumnsDF = specificColumnsDf.withColumnRenamed("level", "subscription\_type")  renamedColumnsDF.show()    // COMMAND ----------    val blobStorage = storageAccountName + ".blob.core.windows.net"  val blobContainer = fileSystemName  val blobAccessKey = "dvOWdeA+c0Y+68RZ27qbHkhnKytFdhm3CCCQcZCoiDDCR1jiYjdNpkwkE9PZRX7p3dIBONPsFrYAai2VL9QEKw=="    // COMMAND ----------    val tempDir = "wasbs://" + blobContainer + "@" + blobStorage +"/tempDirs"    // COMMAND ----------    val acntInfo = "fs.azure.account.key."+ blobStorage  sc.hadoopConfiguration.set(acntInfo, blobAccessKey)    // COMMAND ----------    //Azure Synapse related settings  val dwDatabase = "dc1-sqldw-pool1"  val dwServer = "dc1-sqldw-server1.database.windows.net"  val dwUser = "dc-admin"  val dwPass = "Toaster2010!"  val dwJdbcPort = "1433"  val dwJdbcExtraOptions = "encrypt=true;trustServerCertificate=true;hostNameInCertificate=\*.database.windows.net;loginTimeout=30;"  val sqlDwUrl = "jdbc:sqlserver://" + dwServer + ":" + dwJdbcPort + ";database=" + dwDatabase + ";user=" + dwUser+";password=" + dwPass + ";$dwJdbcExtraOptions"  val sqlDwUrlSmall = "jdbc:sqlserver://" + dwServer + ":" + dwJdbcPort + ";database=" + dwDatabase + ";user=" + dwUser+";password=" + dwPass    // COMMAND ----------    spark.conf.set(  "spark.sql.parquet.writeLegacyFormat",  "true")    renamedColumnsDF.write.format("com.databricks.spark.sqldw").option("url", sqlDwUrlSmall).option("dbtable", "SampleTable") .option( "forward\_spark\_azure\_storage\_credentials","True").option("tempdir", tempDir).mode("overwrite").save()    // COMMAND ---------- |

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

* + Go to your **SQL Server Management Studio**
  + Login to it
  + Explore your database, e.g., dc1-datawarehouse
  + You'll find a table, SampleData
  + Run a Select query against it and you'll find your data

Azure Databricks - Stream data into Databricks using Event Hubs

Sunday, July 21, 2019

2:32 PM

**Brief Overview:**

**Architecture:**

Machine generated alternative text:
Azure Databricks 
Producer 
Notebook 
Uses Spark 
connector to 
Event Hubs 
Event Hubs 
Uses Spark 
to process 
Spark on Databricks 
APACHE 
Spork 

* Create a VNET, Databricks service and Event Hubs
* Connect a data ingestion system with Azure Databricks to stream data into an Apache Spark cluster
  + Set up data ingestion system using Azure Event Hubs and then connect it to Azure Databricks to process the incoming messages.
  + Use Twitter APIs to ingest tweets into Event Hubs
  + In Azure Databricks, run analytics for further analysis of data

* Create an Azure Event Hubs
* Copy connection string to access the Event Hubs namespace, e.g., Endpoint=sb://<namespace>.servicebus.windows.net/;SharedAccessKeyName=<key name>;SharedAccessKey=<key value>.
* Get shared access policy name and policy key for Event Hubs

* Create Databricks service, workspace, cluster and Scala notebook
* Write two Scala programs - Data feed, e.g., Twitter feed sender and Twitter feed receiver

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

<https://docs.microsoft.com/en-us/azure/azure-databricks/databricks-stream-from-eventhubs>

You connect a data ingestion system with Azure Databricks to stream data into an Apache Spark cluster in near real-time. You set up data ingestion system using Azure Event Hubs and then connect it to Azure Databricks to process the messages coming through. To access a stream of data, you use Twitter APIs to ingest tweets into Event Hubs. Once you have the data in Azure Databricks, you can run analytical jobs to further analyze the data.

Prerequisites

* An Azure Event Hubs namespace.
* An Event Hub within the namespace.
* Connection string to access the Event Hubs namespace. The connection string should have a format similar to *Endpoint=sb://<namespace>.servicebus.windows.net/;SharedAccessKeyName=<key name>;SharedAccessKey=<key value>*.
* Shared access policy name and policy key for Event Hubs.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Twitter Keys:

Consumer API keys

API key: YVVIpEQgWP8CBzPKcoTiFdEKE

API secret key: dvBYpkkTuzykDf7Vl11O0hAOEcu5tzySMcR7FauIfALvq3LRq8

Access token: 283119593-Tcc5CJQsy11yTqvleRNP0WD2h88RMTIvK65XSWlg

Access token secret: phYbkGR1Ic43OAlwMsVwFPtnFUBsPEbrrkutQZFW5iJsb

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

[Create an Azure Event Hubs namespace and event hub](https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-create).

Cluster settings - **Be sure to set the Databricks Runtime version to at 6.2 or later.**

Machine generated alternative text:
Databricks Runtime Version 
6.2 (includes Apache Spark 2_4.4, Scala 211) 

Machine generated alternative text:
Clusters / dcl-db-clusterl 
dcl-db-clusterl 
Cluster Name 
dcl -db-clusterll 
Cluster Mode O 
Standard 
Pool O 
None 
Databricks Runtime Version O 
Cancel 
2-3 workers: 28.0-42.0 Ga Memory, 8-12 cores, 1.5-225 OSU 
Confirm and Resize 
1 Driver: 14.0 Memory, 4 cores, 0.75 DSC' O 
Learn more 
Runtime 55 CTS (Scala spark 2.43) 
Python Version O 
2 
Autopilot Options 
O Enable autoscaling O 
erminate after 
30 
Worker Type O 
Standard DS3 v2 
Driver Type 
Standard DS3 v2 
Advanced Options 
minutes of inactivity O 
14.0 Gd Memory, 4 Cores, 
14.0 Gd Memory, 4 Cores, 
Min Workers 
2 
Max Workers 
3 
0.75 OSU 
0.75 OSU 

**Library Packages:**

Machine generated alternative text:
Clusters / dc4-databricks-cluster 
O dc4-databricks-cluster 
(7 Edit 
Start 
Driver Logs 
Clone 
Metrics 
X Delete 
Apps 
Configuration 
% Uninstall 
Name 
Notebooks (0) 
Libraries Event Log Spark UI 
Install New 
Spark Cluster UI - Master 
2. 
Type 
Maven 
Maven 
Status 
Source 
com_microsoft.azure:azure-eventhubs-spark 
org _ twitter4j: 

**VERY IMPORTANT:**

Machine generated alternative text:
dcl -dbc 
Edit 
Clone 
Restalt 
Terminate 
Driver Logs 
X Delete 
Metrics 
Configuration 
% Uninstall 
Name 
Notebooks (1) 
Install New 
Libraries Event Log Spark UI 
Apps 
Spark Cluster UI - Master 
2. 
Type 
Maven 
Maven 
Status 
Installed 
Installed 
Source 
com_microsoft.azure:azure-eventhubs-spark 
org _ 7 

**To do this:**

click on Libraries --> Install New

click on Library Source: Maven

enter com.microsoft.azure:azure-eventhubs-spark\_2.11:2.3.12 in the **Coordinates** field and click Install

Again, enter org.twitter4j:twitter4j-core:4.0.7 in the Coordinates field and click Install

**ENSURE the Scala and Spark versions are compatible in the Databricks Runtime Version during Databricks cluster creation and the installed libraries. For example:**

Machine generated alternative text:
Databricks Runtime Version O 
Runtime 56 211 , spark 2.43) 
Learn 

**And** com.microsoft.azure:azure-eventhubs-spark\_2.11:2.3.12

**Tweet Sender:**

|  |
| --- |
| import scala.collection.JavaConverters.\_  import com.microsoft.azure.eventhubs.\_  import java.util.concurrent.\_  import scala.collection.immutable.\_  import scala.concurrent.Future  import scala.concurrent.ExecutionContext.Implicits.global    val namespaceName = "dc1-event-hubs1"  val eventHubName = "dc1-event-hub1"  val sasKeyName = "dc1-eh-policy"  val sasKey = "dKDjmcvfmbmuJc+IyCXHgECLnX/xBeOLPHwz51G9fEg="  val connStr = new ConnectionStringBuilder()  .setNamespaceName(namespaceName)  .setEventHubName(eventHubName)  .setSasKeyName(sasKeyName)  .setSasKey(sasKey)    val pool = Executors.newScheduledThreadPool(1)  val eventHubClient = EventHubClient.create(connStr.toString(), pool)    def sleep(time: Long): Unit = Thread.sleep(time)    def sendEvent(message: String, delay: Long) = {  sleep(delay)  val messageData = EventData.create(message.getBytes("UTF-8"))  eventHubClient.get().send(messageData)  System.out.println("Sent event: " + message + "\n")  }    // Add your own keywords to the search query.  val dataSource = "twitter"    if (dataSource == "twitter") {    import twitter4j.\_  import twitter4j.TwitterFactory  import twitter4j.Twitter  import twitter4j.conf.ConfigurationBuilder    // Twitter configuration!  // Replace values below with you  val twitterConsumerKey = "5j6NRxTCMeYeX699HqcaQmnW1"  val twitterConsumerSecret = "OJGV4ZZJZ92q5s0mTDeXKvJEqs1i4JXicXm9onwQUEfZuk1aXB"  val twitterOauthAccessToken = "283119593-rBCTLgENNzPWjj9nB065jWgD9RZlqSSZzAUJ0CHz"  val twitterOauthTokenSecret = "116ErxfGGImxpa1liSkPRDgYVfr1VPRDeHlJq1aIRSBz3"    val cb = new ConfigurationBuilder()  cb.setDebugEnabled(true)  .setOAuthConsumerKey(twitterConsumerKey)  .setOAuthConsumerSecret(twitterConsumerSecret)  .setOAuthAccessToken(twitterOauthAccessToken)  .setOAuthAccessTokenSecret(twitterOauthTokenSecret)    val twitterFactory = new TwitterFactory(cb.build())  val twitter = twitterFactory.getInstance()    // Getting tweets with keywordFilter and sending them to the Event Hub in realtime!  val keywordFilter = {"#Trump"; "#covid-19"; "#Coronavirus"; "#Biden"}    val query = new Query(keywordFilter)  query.setCount(500)  query.lang("en")  var finished = false  while (!finished) {  val result = twitter.search(query)  val statuses = result.getTweets()  var lowestStatusId = Long.MaxValue  for (status <- statuses.asScala) {  if(!status.isRetweet()){  sendEvent(status.getText(), 500)  }  lowestStatusId = Math.min(status.getId(), lowestStatusId)  }  query.setMaxId(lowestStatusId - 1)  }  }    // Closing connection to the Event Hub  eventHubClient.get().close() |

**Tweet Reader:**

|  |  |
| --- | --- |
| /**/---------------------------------------------------------------------------**  **//Message reader from EventHub**  **//----------------------------------------------------------------------------**  import org.apache.spark.eventhubs.\_  import com.microsoft.azure.eventhubs.\_  import org.apache.spark.sql.types.\_  import org.apache.spark.sql.functions.\_    // Build connection string with the above information  val namespaceName = "dc1-event-hubs1"  val eventHubName = "dc1-event-hub1"  val sasKeyName = "dc1-eh-policy"  val sasKey = "dKDjmcvfmbmuJc+IyCXHgECLnX/xBeOLPHwz51G9fEg="  val connStr = new com.microsoft.azure.eventhubs.ConnectionStringBuilder()  .setNamespaceName(namespaceName)  .setEventHubName(eventHubName)  .setSasKeyName(sasKeyName)  .setSasKey(sasKey)    val customEventhubParameters =  EventHubsConf(connStr.toString())  .setMaxEventsPerTrigger(5)    val incomingStream = spark.readStream.format("eventhubs").options(customEventhubParameters.toMap).load()    incomingStream.printSchema    // Sending the incoming stream into the console.  // Data comes in batches!  incomingStream.writeStream.outputMode("append").format("console").option("truncate", false).start().awaitTermination() |  |

Databricks Excel workbook processing

Saturday, September 12, 2020

8:48 PM

// Databricks notebook source

val containerName = "input"

val storageAccountName = "dc1storageacct1"

val key = "HVI8udZTh/FFzvnZFGTqPc+bvxYHXQXm39SUBvTOZVjTH4PPtjMTjOT/soBAUpe3vULQajvEhklEY0pndP3SSA=="

val url = "wasbs://" + containerName + "@" + storageAccountName + ".blob.core.windows.net/"

val config = "fs.azure.account.key." + storageAccountName + ".blob.core.windows.net"

dbutils.fs.mount(

source = url,

mountPoint="/mnt/myblob",

extraConfigs = Map(config -> key))

// COMMAND ----------

display(dbutils.fs.ls("/mnt/myblob"))

// COMMAND ----------

val df = spark.read

.format("com.crealytics.spark.excel")

.option("header", true)

.load("/mnt/myblob/data.xlsx")

// COMMAND ----------

display(df)

// COMMAND ----------

import com.crealytics.spark.excel.\_

val df = spark.read

.format("com.crealytics.spark.excel")

.option("header", true)

.load("/mnt/myblob/data.xlsx")

display(df)

// COMMAND ----------

val df = spark.read

.format("com.crealytics.spark.excel")

.option("dataAddress", "planes!A1")

.option("header", true)

.load("/mnt/myblob/data.xlsx")

display(df)

// COMMAND ----------

val df = spark.read

.format("com.crealytics.spark.excel")

.option("dataAddress", "Table1[#All]")

.option("header", true)

.load("/mnt/myblob/data.xlsx")

display(df)

// COMMAND ----------

val sheetNames = WorkbookReader(

Map("path" -> "/mnt/myblob/data.xlsx"),

spark.sparkContext.hadoopConfiguration)

.sheetNames

// COMMAND ----------

sheetNames.foreach {item =>

var data = spark.read

.format("com.crealytics.spark.excel")

.option("dataAddress", (item + "!A1"))

.option("header", true)

.load("/mnt/myblob/data.xlsx")

data.repartition(1)

.write

.format("com.databricks.spark.csv")

.mode("overwrite")

.option("header", true)

.save("/mnt/myblob/databricks-output/" + item + ".csv")

}

// COMMAND ----------

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

val movieSchema = StructType(Array(

StructField("Profitability", DoubleType, nullable = false),

StructField("Film", StringType, nullable = false),

StructField("Genre", StringType, nullable = false)

))

val df = spark.read

.format("com.crealytics.spark.excel")

.option("dataAddress", ("movies!A1"))

.option("header", true)

.schema(movieSchema)

.load("/mnt/myblob/data.xlsx")

display(df)

// COMMAND ----------

val df = spark.read

.format("com.crealytics.spark.excel")

.option("dataAddress", ("planes!A1"))

.option("header", true)

.load("/mnt/myblob/data.xlsx")

// COMMAND ----------

import org.apache.spark.sql.functions.\_

val aggregated = df

.groupBy("manufacturer")

.count()

display(aggregated)

// COMMAND ----------

aggregated

.write

.format("com.crealytics.spark.excel")

.option("header", true)

.option("dataAddress", ("Aggregated!A1"))

.mode("overwrite")

.save("/mnt/myblob/output.xlsx")

// COMMAND ----------

df

.write

.format("com.crealytics.spark.excel")

.option("header", true)

.option("dataAddress", ("Original!A1"))

.mode("append")

.save("/mnt/myblob/output.xlsx")

Databricks scripts (notebooks) version control in GitHub

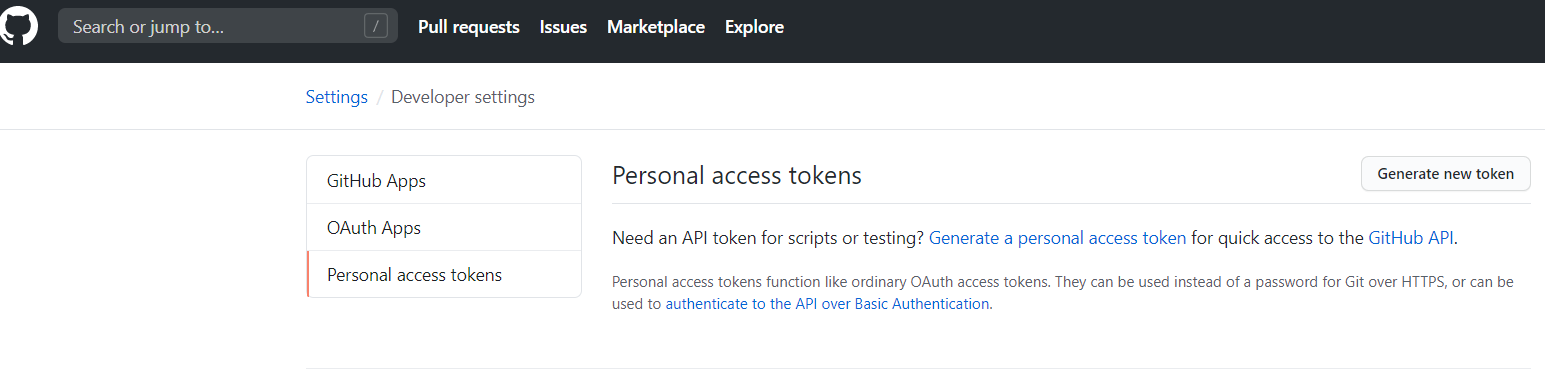
Saturday, September 26, 2020

12:49 PM

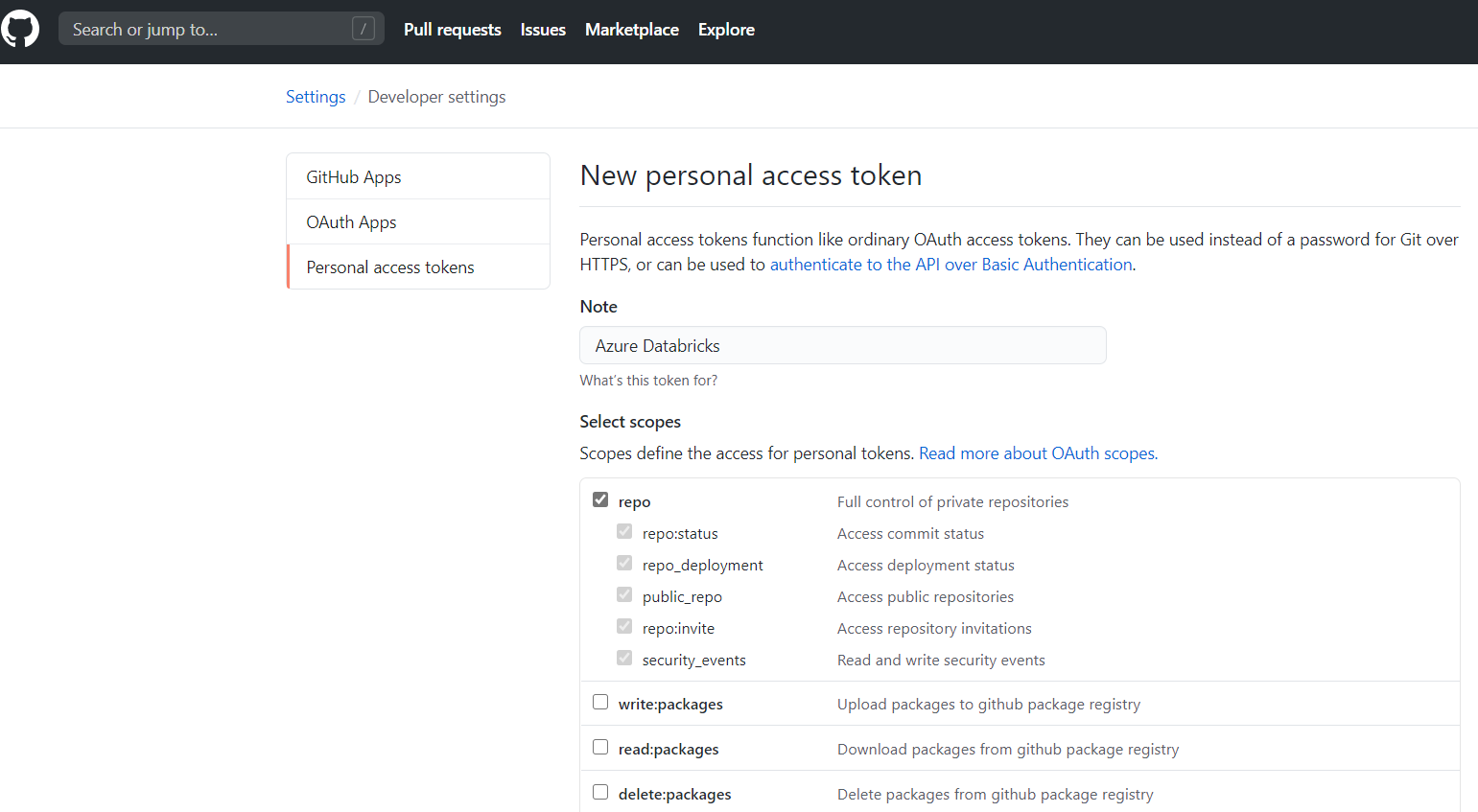
* 1. Create Databricks script using a Python/Scala/etc. notebook

Machine generated alternative text:
Microsoft Azure 
PORTAL 
Azure 
example (Python) 
dcl-db-clusterl 
Edit 
View: Standard • 
a Permissions 
@ Run All 
Clear • 
Schedule 
Comments 
o ? dcl-dbwsl 
Experiment Revision history 
Databricks 
Cmd 1 
1 pri 
Home 
Shi ft+Enter to run 
Workspace 
world! " 
shortcuts 

* 1. Login to GitHub using your credentials
  2. Create Access token for the Databricks
     1. Machine generated alternative text:
        Pull requests 
        DaveCheema 
        •h Personal settings 
        Profile 
        Account 
        Account security 
        Security log 
        Security & analysis 
        Emails 
        Notifications 
        Billing 
        SSH and GPC keys 
        Issues 
        Marketplace Explore 
        Public profile 
        Name 
        Your name may appear around GitHub where you contribute or are mentioned. You can 
        remove it at any time. 
        Public email 
        Select a verified email to display 
        You can manage verified email addresses in your email settings. 
        Bio 
        Tell us a little bit about yourself 
        You can @mention other users and organizations to link to them. 
        Profile picture 
        Edit 
        Signed in as 
        DaveCheema 
        Set status 
        Your profile 
        Your repositories 
        Your projects 
        Your stars 
        Your gists 
        Upgrade 
        Feature preview 
        Help 
        Settings 
        Sign out 
     2. Click on your user icon on the top left corner
     3. Click on Settings
     4. Machine generated alternative text:
        DaveCheema 
        •h Personal settings 
        Profile 
        Account 
        Account security 
        Security log 
        Security & analysis 
        Emails 
        Notifications 
        Billing 
        SSH and GPC keys 
        Repositories 
        Organizations 
        Saved replies 
        Applications 
        Developer settings 
        Moderation settings 
        Blocked users 
        Public profile 
        Name 
        Profile picture 
        Your name may appear around GitHub where you contribute or are mentioned. You can 
        remove it at any time. 
        Public email 
        Select a verified email to display 
        You can manage verified email addresses in your email settings. 
        Edit 
        Bio 
        Tell us a little bit about yourself 
        You can @mention other users and organizations to link to them. 
        URL 
        Twitter username 
        Company 
        You can @mention your company's GitHub organization to link it. 
        Location 
     5. Click on Developer settings



* 1. Click on Personal access tokens, click on Generate new token



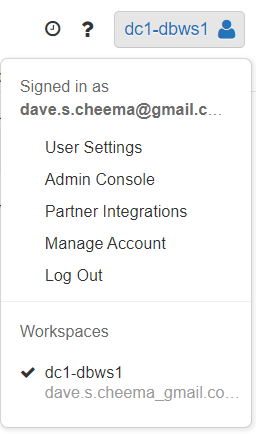
Click on **repo**

Machine generated alternative text:
gist 
notifications 
Cl user 
D read:user 
user:email 
user:follow 
delete_repo 
Cl write:discussion 
D read:discussion 
D admin:enterprise 
D manage_billing:enterprise 
D read:enterprise 
workflow 
Cl admin:gpg_key 
write:gpg_key 
D read:gpg_key 
Generate token 
Create gists 
Access notifications 
Update all user data 
Read all user profile data 
Access user email addresses (read-only) 
Follow and unfollow users 
Delete repositories 
Read and write team discussions 
Read team discussions 
Full control of enterprises 
Read and write enterprise billing data 
Read enterprise profile data 
Update github action workflows 
Full control of public user gpg keys (Developer Preview) 
Write public user gpg keys 
Read public user gpg keys 
Cancel 

Click on Generate token

Copy the token to a scratch pad

* 1. Go to Databricks workspace that you were working in
  2. Click on your user avatar (in the upper left corner) and click in User settings

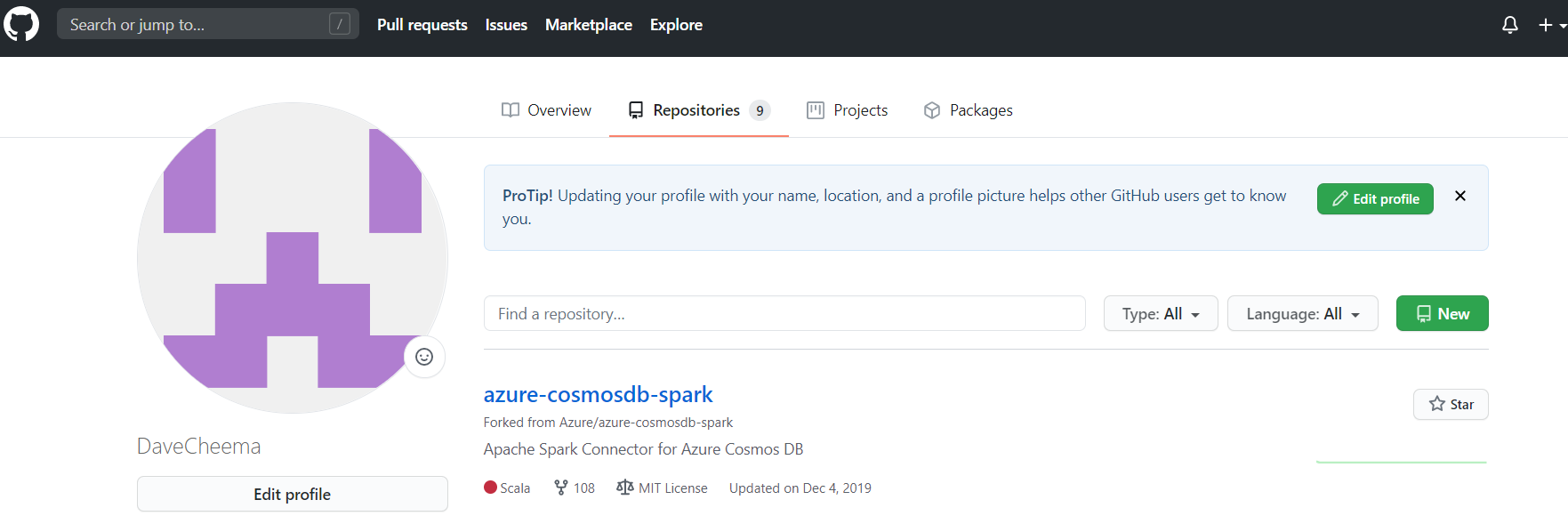




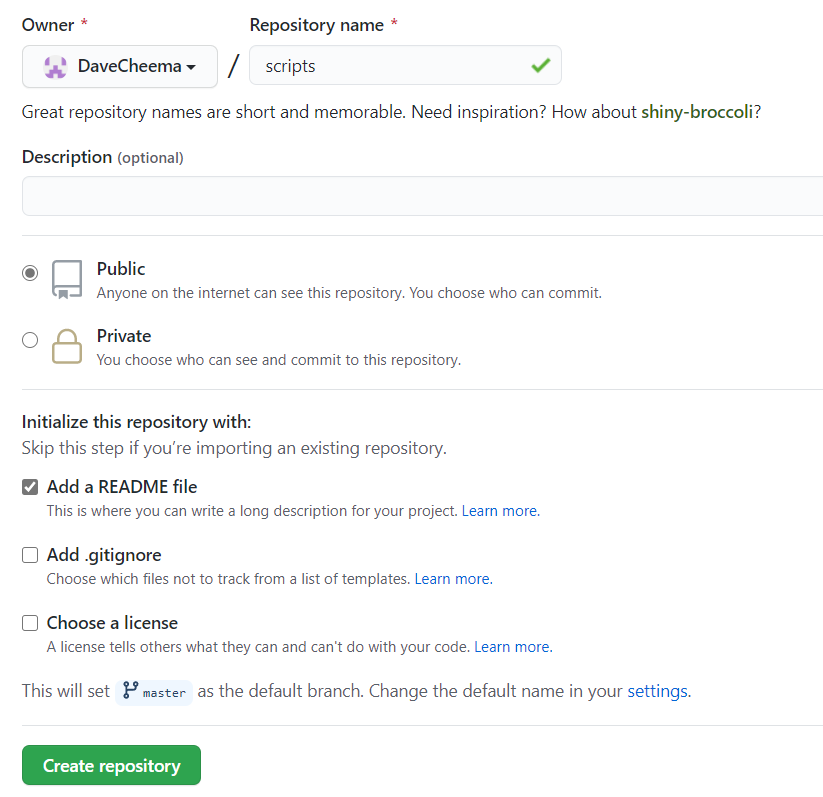
Click on Git Integration, paste the Token or app password that you had saved from Github

Click on Save

* 1. Go back to Github again
  2. Click on your user avatar and click on Your repositories

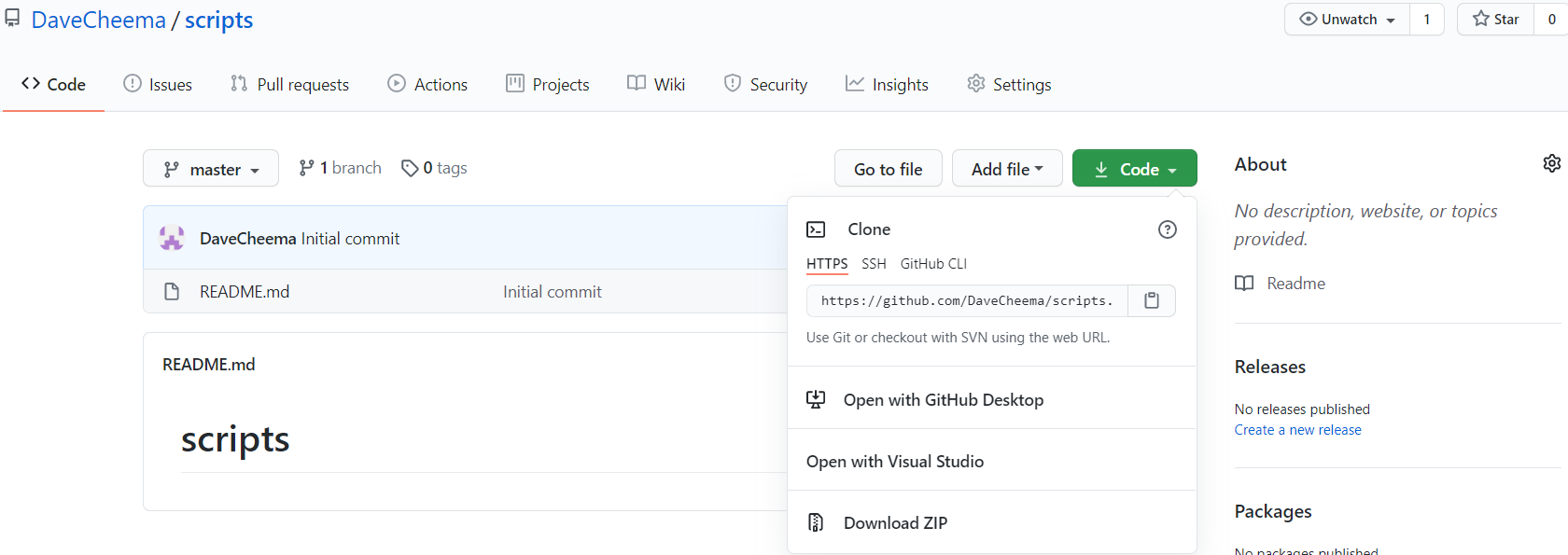


Click on New

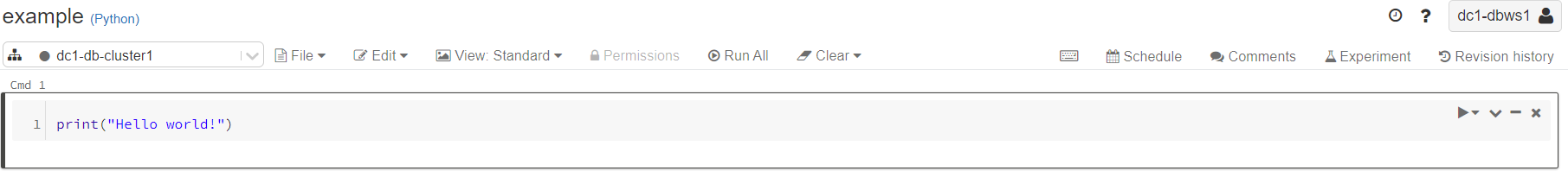


Enter Repository name, check on **Add a README file** (very important!) and click on Create repository

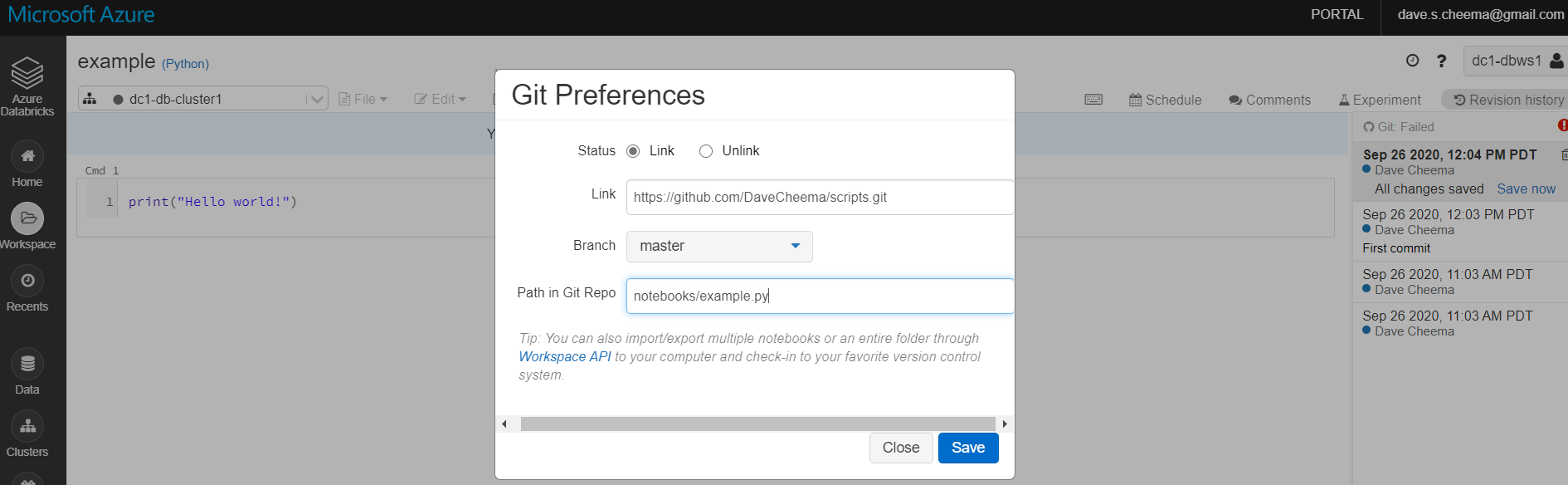
Click on Code and copy the HTTPS URL



Go back to Databricks workspace, in the notebook you were working on.

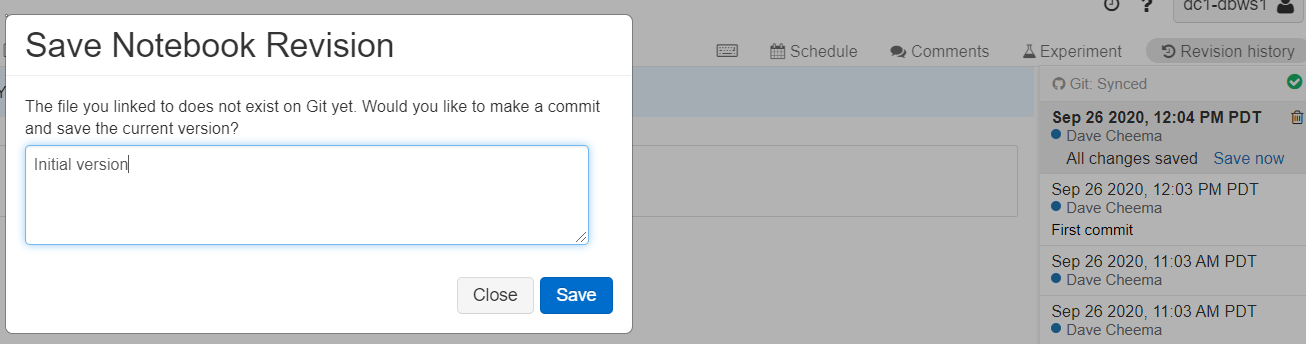


Click on Revision history

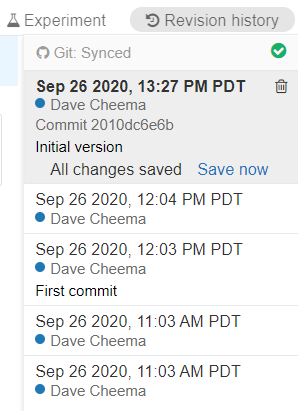


Click on Git unlinked

On Git Preferences page, click on Link, enter the url of the Git repository and the Path in Git Repo, e.g., notebooks/example.py and click on Save



Enter comments about the version and click Save. And you will get Git Synced



Now you're all set to version control your Databricks notebook using GitHub!

Create and execute Databricks jobs

Tuesday, September 29, 2020

10:04 AM

Three way to trigger a Databricks job:

* 1. Interactive - execute as you're developing notebooks
  2. Trigger a Databricks job from ADF v2. you can even pass parameters
  3. Create a Databricks job in the Databricks workspace. In this page, we'll discuss how to create and run a Databricks job in the Databricks workspace environment

Jobs

August 13, 2020

A job is a way of running a notebook or JAR either immediately or on a scheduled basis. The other way to run a notebook is interactively in the [notebook UI](https://docs.databricks.com/notebooks/index.html).

You can create and run jobs using the UI, the CLI, and by invoking the Jobs API. You can monitor job run results in the UI, using the CLI, by querying the API, and through email alerts. This article focuses on performing job tasks using the UI. For the other methods, see [Jobs CLI](https://docs.databricks.com/dev-tools/cli/jobs-cli.html) and [Jobs API](https://docs.databricks.com/dev-tools/api/latest/jobs.html).

Important

* 1. The number of jobs is limited to 1000.
  2. The number of jobs a workspace can create in an hour is limited to 5000 (includes “run now” and “runs submit”). This limit also affects jobs created by the REST API and notebook workflows.
  3. A workspace is limited to 150 concurrent (running) job runs.
  4. A workspace is limited to 1000 active (running and pending) job runs.

View jobs

Click the Jobs icon

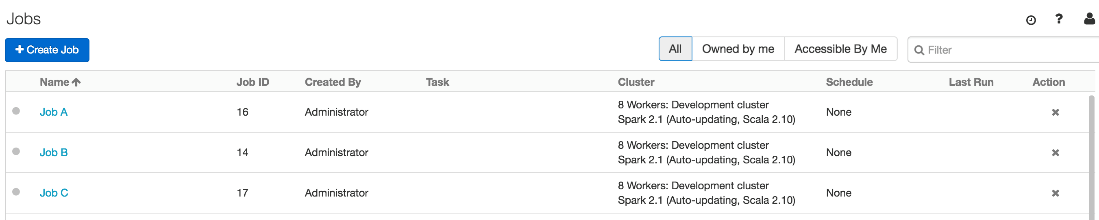
Jobs Menu Icon

 in the sidebar. The Jobs list displays. The Jobs page lists all defined jobs, the cluster definition, the schedule if any, and the result of the last run.

In the Jobs list, you can filter jobs:

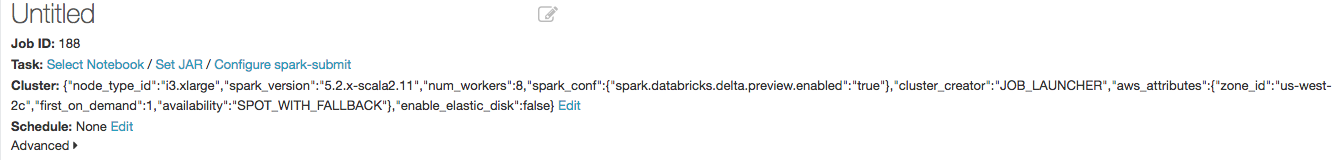
* 1. Using key words.
  2. Selecting only jobs you own or jobs you have access to. Access to this filter depends on [Jobs access control](https://docs.databricks.com/security/access-control/jobs-acl.html) being enabled.

You can also click any column header to sort the list of jobs (either descending or ascending) by that column. By default, the page is sorted on job names in ascending order.



Create a job

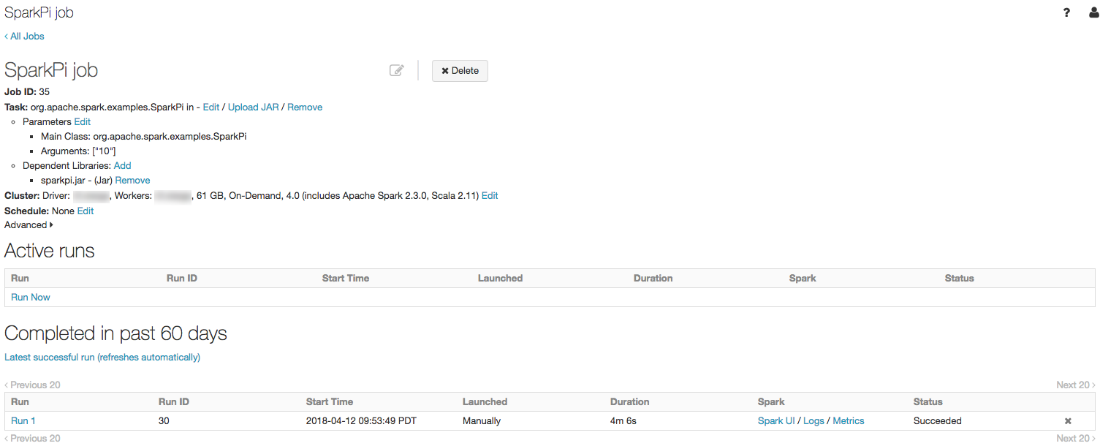
* 1. Click + Create Job. The job detail page displays.



* 1. Enter a name in the text field with the placeholder text Untitled.
  2. Specify the task type: click Select Notebook, Set JAR, or Configure spark-submit.
  3. Notebook
  4. Select a notebook and click OK.
  5. Next to Parameters, click Edit. Specify key-value pairs or a JSON string representing key-value pairs. Such parameters set the value of [widgets](https://docs.databricks.com/notebooks/widgets.html).
  6. JAR: Upload a JAR, specify the main class and arguments, and click OK. To learn more about JAR jobs, see [JAR job tips](https://docs.databricks.com/jobs.html#jar-jobs).
  7. spark-submit: Specify the main class, path to the library JAR, arguments, and click Confirm. To learn more about spark-submit, see the [Apache Spark documentation](https://spark.apache.org/docs/latest/submitting-applications.html).  
     Note  
     The following Databricks features are not available for spark-submit jobs:
  8. Cluster autoscaling. To learn more about autoscaling, see [Cluster autoscaling](https://docs.databricks.com/clusters/configure.html#autoscaling).
  9. [Databricks Utilities](https://docs.databricks.com/dev-tools/databricks-utils.html). If you want to use Databricks Utilities, use JAR jobs instead.
  10. In the Dependent Libraries field, optionally click Add and specify dependent libraries. Dependent libraries are automatically attached to the cluster on launch. Follow the recommendations in [Library dependencies](https://docs.databricks.com/jobs.html#library-dependencies) for specifying dependencies.  
      Important  
      If you have configured a library to [automatically install on all clusters](https://docs.databricks.com/libraries/cluster-libraries.html#install-workspace-libraries) or in the next step you select an existing terminated cluster that has libraries installed, the job execution does not wait for library installation to complete. If a job requires a certain library, you should attach the library to the job in the Dependent Libraries field.
  11. In the Cluster field, click Edit and specify the cluster on which to run the job. In the Cluster Type drop-down, choose New Job Cluster or Existing All-Purpose Cluster.  
      Note  
      Keep the following in mind when you choose a cluster type:
  12. For production-level jobs or jobs that are important to complete, we recommend that you select New Job Cluster.
  13. You can run spark-submit jobs only on new clusters.
  14. When you run a job on a new cluster, the job is treated as a data engineering (job) workload subject to the job workload pricing. When you run a job on an existing cluster, the job is treated as a data analytics (all-purpose) workload subject to all-purpose workload pricing.
  15. If you select a terminated existing cluster and the job owner has Can Restart [permission](https://docs.databricks.com/security/access-control/cluster-acl.html), Databricks starts the cluster when the job is scheduled to run.
  16. Existing clusters work best for tasks such as updating [dashboards](https://docs.databricks.com/notebooks/dashboards.html) at regular intervals.
  17. New Job Cluster - complete the [cluster configuration](https://docs.databricks.com/clusters/configure.html#cluster-configurations).
  18. In the cluster configuration, select a runtime version. For help with selecting a runtime version, see [Databricks Runtime](https://docs.databricks.com/release-notes/runtime/index.html#runtime-release-notes) and [Databricks Light](https://docs.databricks.com/runtime/light.html#light).
  19. To decrease new cluster start time, select a [pool](https://docs.databricks.com/clusters/instance-pools/cluster-instance-pool.html#cluster-instance-pool) in the cluster configuration.
  20. Existing All-Purpose Cluster - in the drop-down, select the existing cluster.
  21. In the Schedule field, optionally click Edit and schedule the job. See [Run a job](https://docs.databricks.com/jobs.html#run-a-job).
  22. Optionally click Advanced and specify advanced job options. See [Advanced job options](https://docs.databricks.com/jobs.html#advanced-job-options).

View job details

On the Jobs page, click a job name in the Name column. The job details page shows configuration parameters, active runs (running and pending), and completed runs.



Databricks maintains a history of your job runs for up to 60 days. If you need to preserve job runs, we recommend that you export job run results before they expire. For more information, see [Export job run results](https://docs.databricks.com/jobs.html#export-job-runs).

In the job runs page, you can view the standard error, standard output, log4j output for a job run by clicking the Logs link in the Spark column.

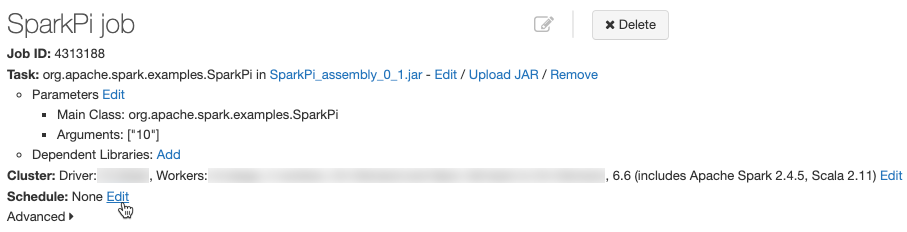
Run a job

You can run a job on a schedule or immediately.

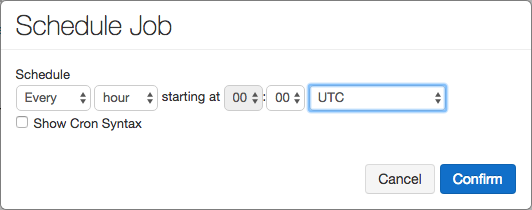
Schedule a job

To define a schedule for the job:

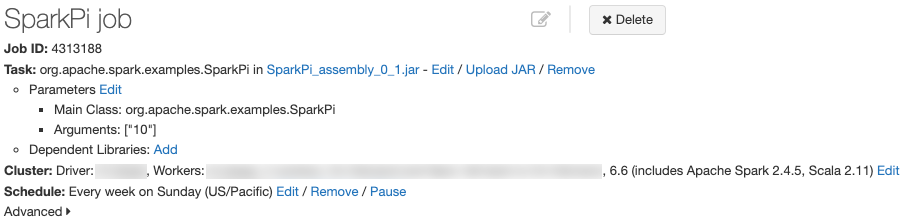
* 1. Click Edit next to Schedule.



The Schedule Job dialog displays.

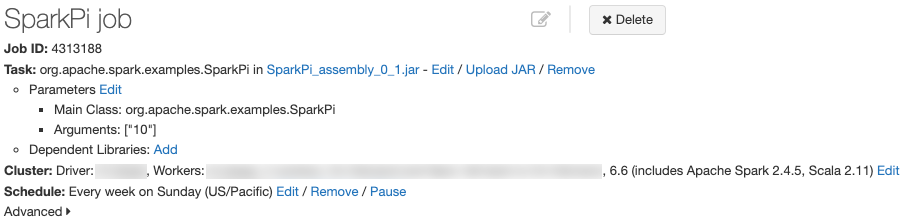


* 1. Specify the schedule granularity, starting time, and time zone. Optionally select the Show Cron Syntax checkbox to display and edit the schedule in [Quartz Cron Syntax](http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html).  
     Note
  2. Databricks enforces a minimum interval of 10 seconds between subsequent runs triggered by the schedule of a job regardless of the seconds configuration in the cron expression.
  3. You can choose a time zone that observes daylight saving time or a UTC time. If you select a zone that observes daylight saving time, an hourly job will be skipped or may appear to not fire for an hour or two [when daylight saving time begins or ends](https://www.quartz-scheduler.org/documentation/2.3.1-SNAPSHOT/faq.html#questions-about-jobs). If you want jobs to run at every hour (absolute time), choose a UTC time.
  4. The job scheduler, like the Spark batch interface, is not intended for low latency jobs. Due to network or cloud issues, job runs may occasionally be delayed up to several minutes. In these situations, scheduled jobs will run immediately upon service availability.
  5. Click Confirm.

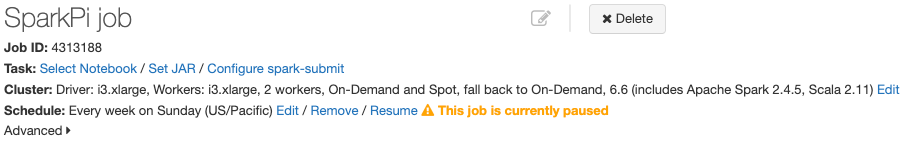


Pause and resume a job schedule

To pause a job, click the Pause button next to the job schedule:

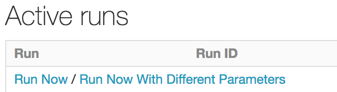


To resume a paused job schedule, click the Resume button:



Run a job immediately

To run the job immediately, in the Active runs table click Run Now.



Tip

Click Run Now to do a test run of your notebook or JAR when you’ve finished configuring your job. If your notebook fails, you can edit it and the job will automatically run the new version of the notebook.

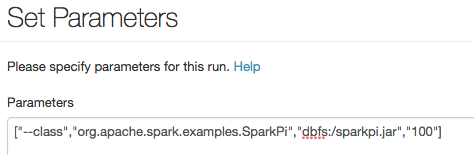
Run a job with different parameters

You can use Run Now with Different Parameters to re-run a job specifying different parameters or different values for existing parameters.

* 1. In the Active runs table, click Run Now with Different Parameters. The dialog varies depending on whether you are running a notebook job or a spark-submit job.
  2. Notebook - A UI that lets you set key-value pairs or a JSON object displays. You can use this dialog to set the values of [widgets](https://docs.databricks.com/notebooks/widgets.html):



* 1. spark-submit - A dialog containing the list of parameters displays. For example, you could run the SparkPi estimator described in [Create a job](https://docs.databricks.com/jobs.html#job-create) with 100 instead of the default 10 partitions:



* 1. Specify the parameters. The provided parameters are merged with the default parameters for the triggered run. If you delete keys, the default parameters are used.
  2. Click Run.

Notebook job tips

Total notebook cell output (the combined output of all notebook cells) is subject to a 20MB size limit. Additionally, individual cell output is subject to an 8MB size limit. If total cell output exceeds 20MB in size, or if the output of an individual cell is larger than 8MB, the run will be canceled and marked as failed. If you need help finding cells that are near or beyond the limit, run the notebook against an all-purpose cluster and use this [notebook autosave technique](https://kb.databricks.com/notebooks/notebook-autosave.html).

JAR job tips

There are some caveats you need to be aware of when you run a JAR job.

Output size limits

Note

Available in Databricks Runtime 6.3 and above.

Job output, such as log output emitted to stdout, is subject to a 20MB size limit. If the total output has a larger size, the run will be canceled and marked as failed.

To avoid encountering this limit, you can prevent stdout from being returned from the driver to Databricks by setting the spark.databricks.driver.disableScalaOutput Spark configuration to true. By default the flag value is false. The flag controls cell output for Scala JAR jobs and Scala notebooks. If the flag is enabled, Spark does not return job execution results to the client. The flag does not affect the data that is written in the cluster’s log files. Setting this flag is recommended only for job clusters for JAR jobs, because it will disable notebook results.

Use the shared SparkContext

Because Databricks is a managed service, some code changes may be necessary to ensure that your Apache Spark jobs run correctly. JAR job programs must use the shared SparkContext API to get the SparkContext. Because Databricks initializes the SparkContext, programs that invoke new SparkContext() will fail. To get the SparkContext, use only the shared SparkContext created by Databricks:

Scala

Copy

**val** goodSparkContext **=** SparkContext.getOrCreate()  
**val** goodSparkSession **=** SparkSession.builder().getOrCreate()

In addition, there are several methods you should avoid when using the shared SparkContext.

* 1. Do not call SparkContext.stop().
  2. Do not call System.exit(0) or sc.stop() at the end of your Main program. This can cause undefined behavior.

Use try-finally blocks for job clean up

Consider a JAR that consists of two parts:

* 1. jobBody() which contains the main part of the job
  2. jobCleanup() which has to be executed after jobBody(), irrespective of whether that function succeded or returned an exception

As an example, jobBody() may create tables, and you can use jobCleanup() to drop these tables.

The safe way to ensure that the clean up method is called is to put a try-finally block in the code:

Scala

Copy

**try** {  
 jobBody()  
} **finally** {  
 jobCleanup()  
}

You should *should not* try to clean up using sys.addShutdownHook(jobCleanup) or

Scala

Copy

**val** cleanupThread **= new** Thread { **override def** run **=** jobCleanup() }  
Runtime.getRuntime.addShutdownHook(cleanupThread)

Due to the way the lifetime of Spark containers is managed in Databricks, the shutdown hooks are not run reliably.

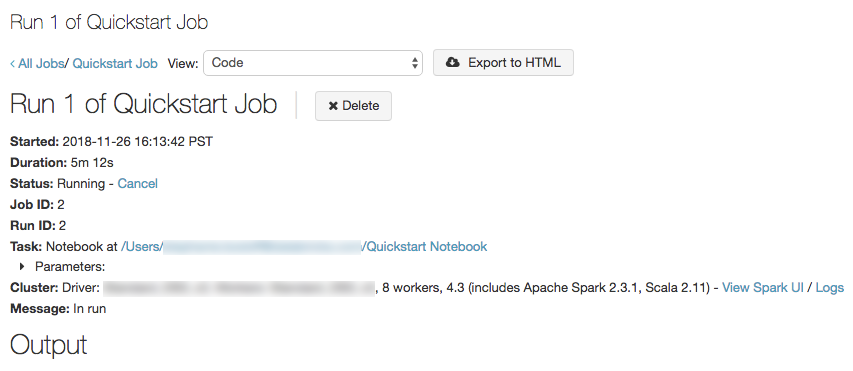
Configure JAR job parameters

JAR jobs are parameterized with an array of strings.

* 1. In the UI, you input the parameters in the Arguments text box which are split into an array by applying POSIX shell parsing rules. For more information, reference the [shlex documentation](https://docs.python.org/2/library/shlex.html#shlex.split).
  2. In the API, you input the parameters as a standard JSON array. For more information, reference [SparkJarTask](https://docs.databricks.com/dev-tools/api/latest/jobs.html#jobssparkjartask). To access these parameters, inspect the String array passed into your main function.

View job run details

A job run details page contains job output and links to logs:

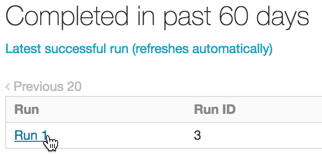


You can view job run details from the Jobs page and the Clusters page.

* 1. Click the Jobs icon

Jobs Menu Icon

. In the Run column of the Completed in past 60 days table, click the run number link.



* 1. Click the Clusters icon

Clusters Icon

. In a job row in the Job Clusters table, click the Job Run link.

Job run from Clusters

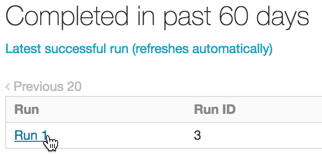
Export job run results

You can export notebook run results and job run logs for all job types.

Export notebook run results

You can persist job runs by exporting their results. For notebook job runs, you can [export](https://docs.databricks.com/notebooks/notebooks-manage.html#export-notebook) a rendered notebook which can be later be [imported](https://docs.databricks.com/notebooks/notebooks-manage.html#import-notebook) into your Databricks workspace.

* 1. In the job detail page, click a job run name in the Run column.



* 1. Click Export to HTML.



Export job run logs

You can also export the logs for your job run. To automate this process, you can set up your job so that it automatically delivers logs to DBFS or S3 through the Job API. For more information, see the [NewCluster](https://docs.databricks.com/dev-tools/api/latest/jobs.html#jobsclusterspecnewcluster) and [ClusterLogConf](https://docs.databricks.com/dev-tools/api/latest/clusters.html#clusterclusterlogconf) fields in the Job [Create](https://docs.databricks.com/dev-tools/api/latest/jobs.html#jobsjobsservicecreatejob) API call.

Edit a job

To edit a job, click the job name link in the Jobs list.

Delete a job

To delete a job, click the x in the Action column in the Jobs list.

Library dependencies

The Spark driver has certain library dependencies that cannot be overridden. These libraries take priority over any of your own libraries that conflict with them.

To get the full list of the driver library dependencies, run the following command inside a notebook attached to a cluster of the same Spark version (or the cluster with the driver you want to examine).

Bash

Copy

%sh  
ls /databricks/jars

Manage library dependencies

A good rule of thumb when dealing with library dependencies while creating JARs for jobs is to list Spark and Hadoop as provided dependencies. On Maven, add Spark and/or Hadoop as provided dependencies as shown in the following example.

XML

Copy

**<dependency>**  
 **<groupId>**org.apache.spark**</groupId>**  
 **<artifactId>**spark-core\_2.11**</artifactId>**  
 **<version>**2.3.0**</version>**  
 **<scope>**provided**</scope>**  
**</dependency>**  
**<dependency>**  
 **<groupId>**org.apache.hadoop**</groupId>**  
 **<artifactId>**hadoop-core**</artifactId>**  
 **<version>**1.2.1**</version>**  
 **<scope>**provided**</scope>**  
**</dependency>**

In sbt, add Spark and Hadoop as provided dependencies as shown in the following example.

Scala

Copy

libraryDependencies += "org.apache.spark" %% "spark-core" % "2.3.0" % "provided"  
libraryDependencies += "org.apache.hadoop" %% "hadoop-core" % "1.2.1" % "provided"

Tip

Specify the correct Scala version for your dependencies based on the version you are running.

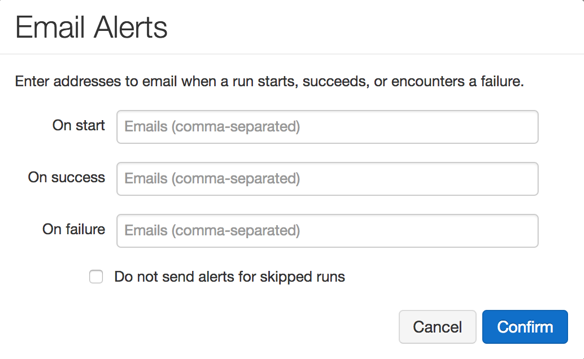
Advanced job options

Maximum concurrent runs

The maximum number of runs that can be run in parallel. On starting a new run, Databricks skips the run if the job has already reached its maximum number of active runs. Set this value higher than the default of 1 if you want to be able to perform multiple runs of the same job concurrently. This is useful for example if you trigger your job on a frequent schedule and want to allow consecutive runs to overlap with each other, or if you want to trigger multiple runs that differ by their input parameters.

Alerts

Email alerts sent in case of job failure, success, or timeout. You can set alerts up for job start, job success, and job failure (including skipped jobs), providing multiple comma-separated email addresses for each alert type. You can also opt out of alerts for skipped job runs.



Integrate these email alerts with your favorite notification tools, including:

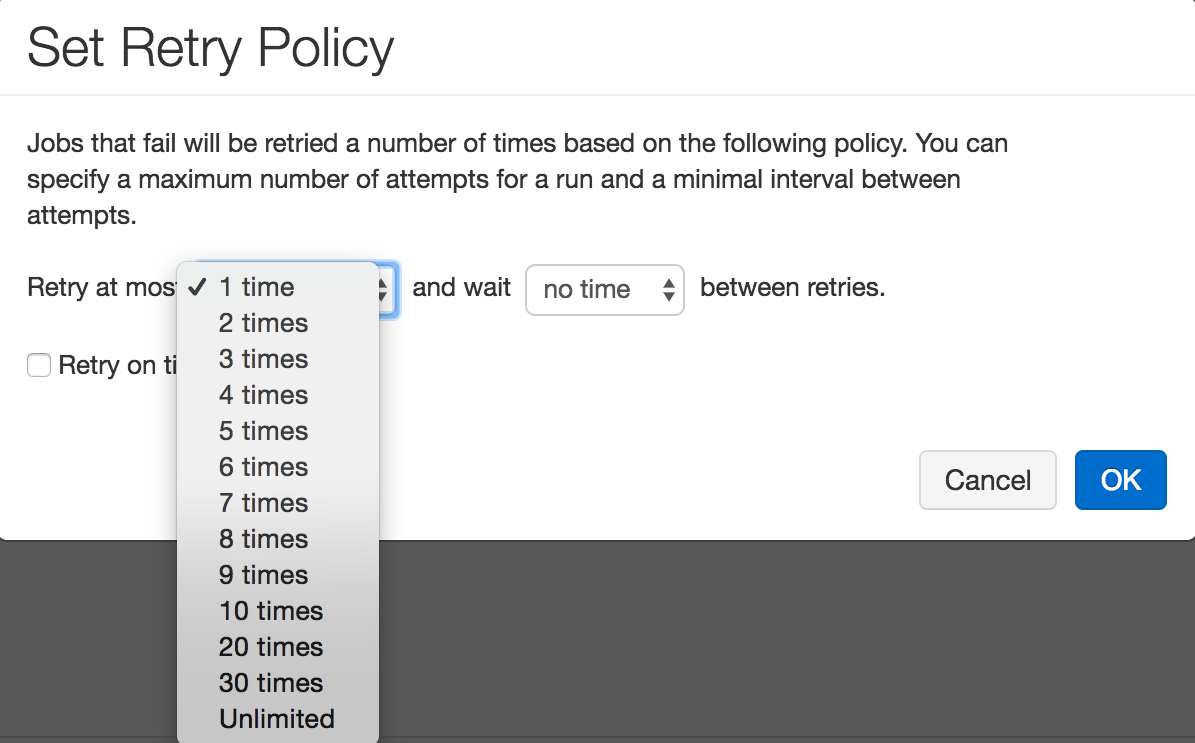
* 1. [PagerDuty](https://www.pagerduty.com/docs/guides/email-integration-guide)
  2. [Slack](https://get.slack.help/hc/articles/206819278-Send-emails-to-Slack)
  3. [Amazon SES and SNS](https://docs.aws.amazon.com/ses/latest/DeveloperGuide/receiving-email-setting-up.html)

Timeout

The maximum completion time for a job. If the job does not complete in this time, Databricks sets its status to “Timed Out”.

Retries

Policy that determines when and how many times failed runs are retried.



Note

If you configure both Timeout and Retries, the timeout applies to each retry.

Control access to jobs

Job access control enable job owners and administrators to grant fine grained permissions on their jobs. With job access controls, job owners can choose which other users or groups can view results of the job. Owners can also choose who can manage runs of their job (that is, invoke Run Now and Cancel.)

See [Jobs access control](https://docs.databricks.com/security/access-control/jobs-acl.html) for details.

From <[*https://docs.databricks.com/jobs.html*](https://docs.databricks.com/jobs.html)>

Azure Databricks GitHub Integration

Tuesday, September 7, 2021

10:11 AM

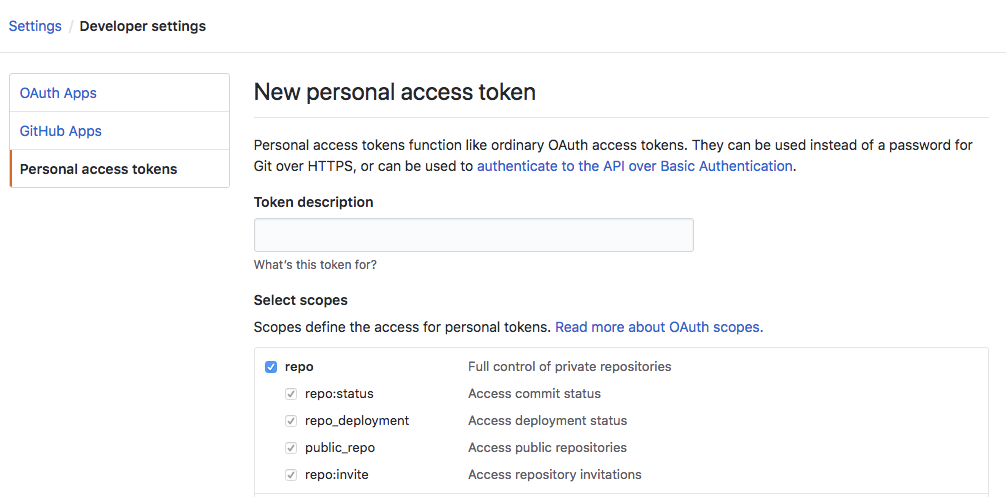
Configure version control

To configure version control, you create access credentials in your version control provider, then add those credentials to Azure Databricks.

Get an access token

In GitHub, follow these steps to create a personal access token that allows access to your repositories:

1. In the **upper-right corner** of any page, click **your profile photo**, then click Settings.
2. Click **Developer settings**.
3. Click the **Personal access tokens** tab.
4. Click the **Generate new token** button.
5. Enter a token description.
6. Select the repo permission, and click the**Generate token** button.



1. Copy the token to your clipboard. You **enter this token in Azure Databricks** in the next step.

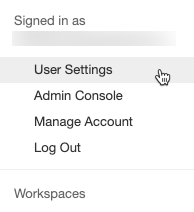
See the [GitHub documentation](https://help.github.com/articles/creating-an-access-token-for-command-line-use/) to learn more about how to create personal access tokens.

Save your access token to Azure Databricks

1. In Azure Databricks, click the User icon

Account Icon

 at the top right of your screen and click User Settings.



1. Click the **Git Integration** tab.
2. If you have previously entered credentials, click the Change settings button.
3. In the **Git provider** drop-down, select **GitHub**.
4. Paste your token into the Token field.
5. Enter your GitHub username or email into the Git provider username or email field and click Save.

Work with notebook revisions

You work with notebook revisions in the history panel. Open the history panel by clicking Revision history at the top right of the notebook.

Revision history

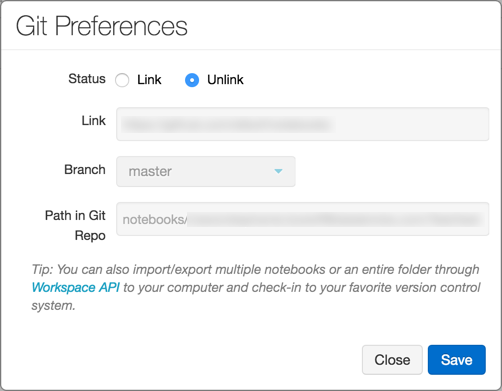
**Note**: You cannot modify a notebook while the history panel is open.

Link a notebook to GitHub

1. Click Revision history at the top right of the notebook. The Git status bar displays Git: Not linked.

Git status bar

1. Click Git: Not linked.  
   The Git Preferences dialog displays. The first time you open your notebook, the Status is Unlink, because the notebook is not in GitHub.

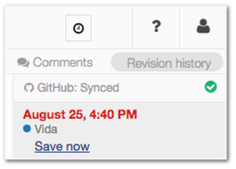


1. In the Status field, **click Link**.
2. In the **Link field**, **paste the URL of the GitHub repository**.
3. Click the Branch drop-down and select a branch or type the name of a new branch.
4. In the **Path in Git Repo field**, **specify where in the repository to store your file**.  
   Python notebooks have the suggested default file extension .py. If you use .ipynb, your notebook will save in iPython notebook format. If the file already exists on GitHub, you can directly copy and paste the URL of the file.
5. Click **Save** to finish linking your notebook. If this file did not previously exist, a prompt with the option Save this file to your GitHub repo displays.
6. Type a message and click Save.

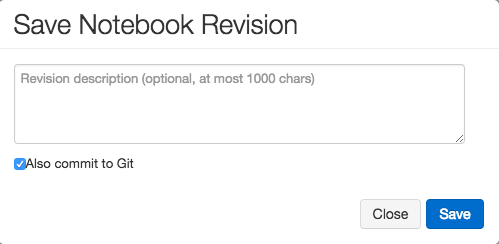
Save a notebook to GitHub

While the **changes** that you make to your notebook are saved automatically to the Azure Databricks revision history, **changes do not automatically persist to GitHub**.

1. Click Revision history at the top right of the notebook to open the history Panel.



1. Click**Save Now** to save your notebook to GitHub. The Save Notebook Revision dialog displays.
2. Optionally, enter a message to describe your change.
3. Make sure that Also **commit to Git is selected**.

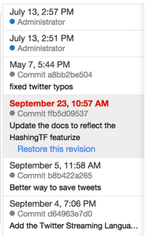


1. Click Save.

Revert or update a notebook to a version from GitHub

Once you link a notebook, Azure Databricks syncs your history with Git every time you re-open the history panel. Versions that sync to Git have commit hashes as part of the entry.

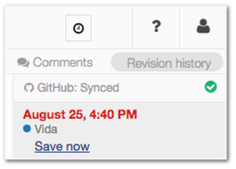
1. Click Revision history at the top right of the notebook to open the history Panel.



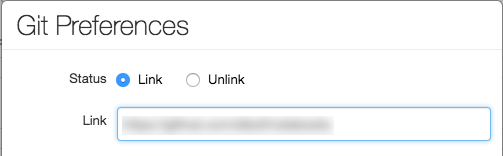
1. Choose an entry in the history panel. Azure Databricks displays that version.
2. Click Restore this version.
3. Click Confirm to confirm that you want to restore that version.

Unlink a notebook

1. Click Revision history at the top right of the notebook to open the history Panel.
2. The Git status bar displays Git: Synced.



1. Click Git: Synced.



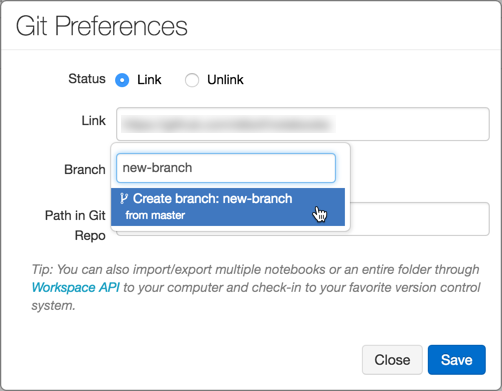
1. In the Git Preferences dialog, click Unlink.
2. Click Save.
3. Click Confirm to confirm that you want to unlink the notebook from version control.

Branch support

You can work on any branch of your repository and create new branches inside Azure Databricks.

Create a branch

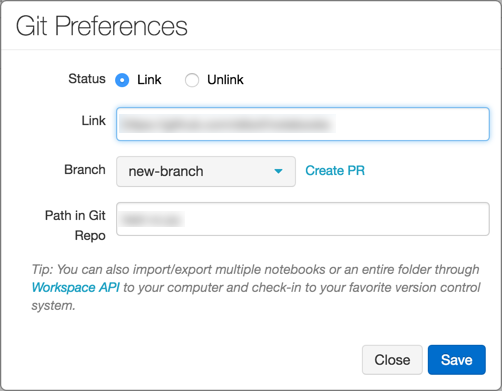
1. Click Revision history at the top right of the notebook to open the history Panel.
2. Click the Git status bar to open the GitHub panel.
3. Click the Branch dropdown.
4. Enter a branch name.



1. Select the Create Branch option at the bottom of the dropdown. The parent branch is indicated. You always branch from your current selected branch.

Create a pull request

1. Click Revision history at the top right of the notebook to open the history Panel.
2. Click the Git status bar to open the GitHub panel.



1. Click Create PR. GitHub opens to a pull request page for the branch.

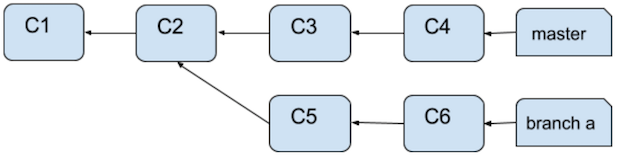
Rebase a branch

You can also rebase your branch inside Azure Databricks. The Rebase link displays if new commits are available in the parent branch. Only rebasing on top of the default branch of the parent repository is supported.

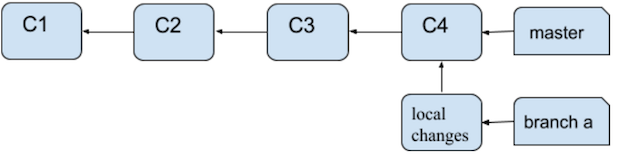
Rebase

For example, assume that you are working on databricks/reference-apps. You fork it into your own account (for example, brkyvz) and start working on a branch called my-branch. If a new update is pushed to databricks:master, then the Rebase button displays, and you will be able to pull the changes into your branch brkyvz:my-branch.

Rebasing works a little differently in Azure Databricks. Assume the following branch structure:



After a rebase, the branch structure will look like:



What’s different here is that Commits C5 and C6 will not apply on top of C4. They will appear as local changes in your notebook. Any merge conflict will show up as follows:



You can then commit to GitHub once again using the Save Now button.

What happens if someone branched off from my branch that I just rebased?

If your branch (for example, branch-a) was the base for another branch (branch-b), and you rebase, you need not worry! Once a user also rebases branch-b, everything will work out. The best practice in this situation is to use separate branches for separate notebooks.

Best practices for code reviews

Azure Databricks supports Git branching.

* You can link a notebook to any branch in a repository. Azure Databricks recommends using a separate branch for each notebook.
* During development, you can link a notebook to a fork of a repository or to a non-default branch in the main repository. To integrate your changes upstream, you can use the Create PR link in the Git Preferences dialog in Azure Databricks to create a GitHub pull request. The Create PR link displays only if you’re not working on the default branch of the parent repository.

GitHub Enterprise

This feature is in [Private Preview](https://docs.microsoft.com/en-us/azure/databricks/release-notes/release-types). To try it, reach out to your Azure Databricks contact.

You can also use the [Workspace API](https://docs.microsoft.com/en-us/azure/databricks/dev-tools/api/latest/workspace) to programmatically create notebooks and manage the code base in GitHub Enterprise Server.

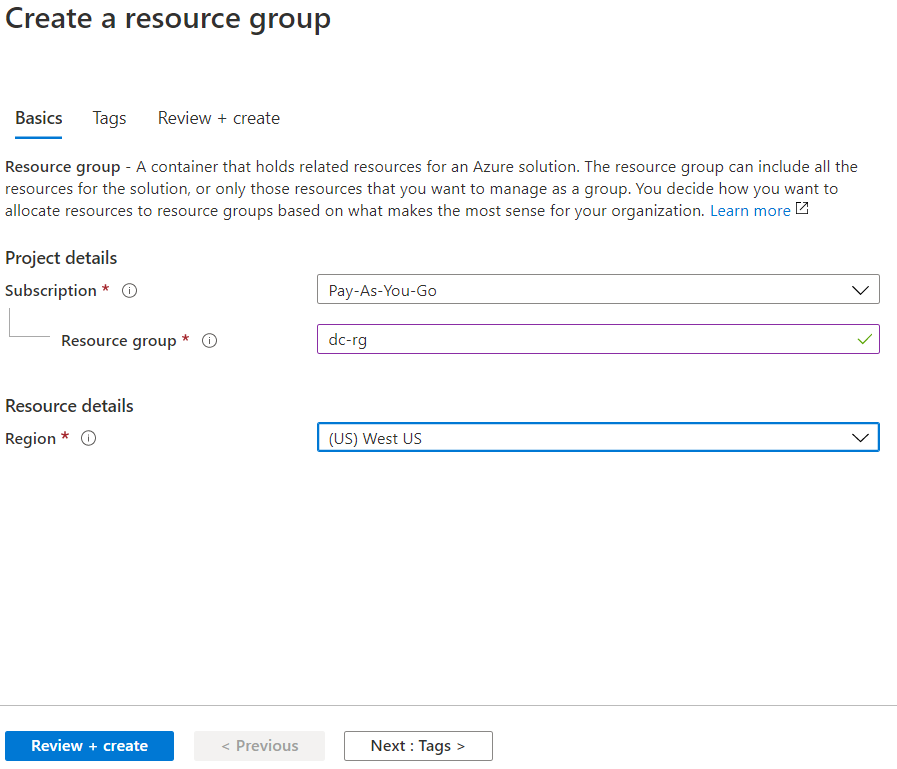
Reference [*https://docs.microsoft.com/en-us/azure/databricks/notebooks/github-version-control*](https://docs.microsoft.com/en-us/azure/databricks/notebooks/github-version-control)

ADF v2.0 Call Databricks with Parameters

Monday, September 21, 2020

6:47 PM

* 1. Create a resource group in Azure portal



Enter Resource group name, select Region and click Review + Create

Go to the Resource group details

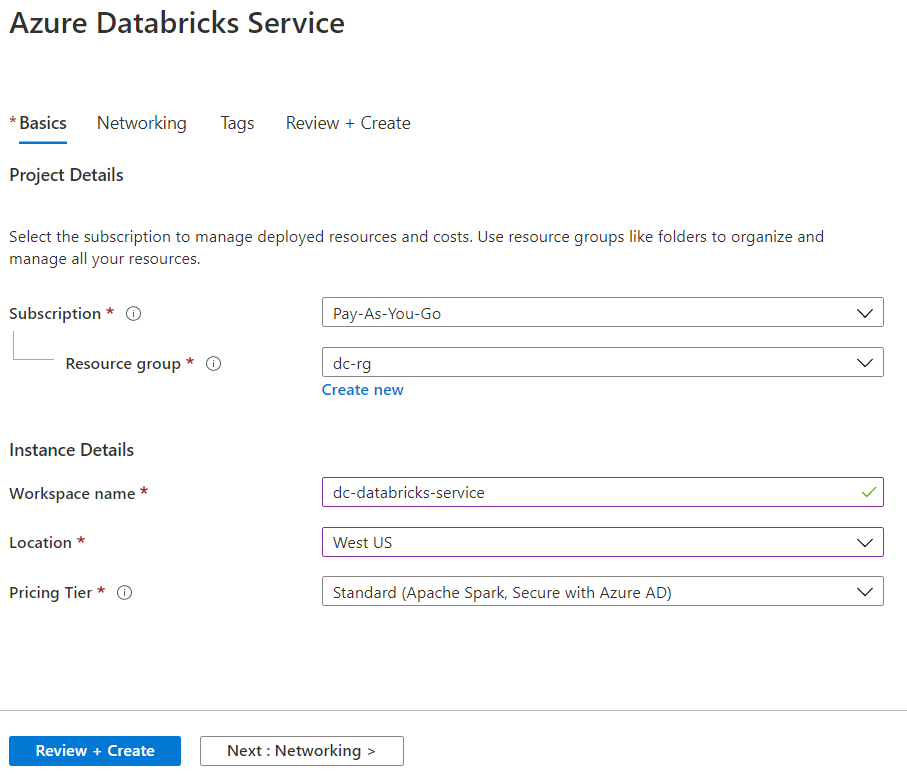
Machine generated alternative text:
-F Add 
Edit columns 
Essentials 
Subscription (change) 
Pay-As-You-Go 
Subscription ID 
Delete resource group 
Refresh 
= (all) X 
Export to CSV Open query 
Deployments 
No deployments 
+7 Add filter 
No grouping 
Type 
Assign tags 
Move v 
dee29fa2-d695-439c-80a5-158ce469882b 
Tags (change) 
Click here to add tags 
Filter by name... 
Showing O to O of O records. 
Name 
= (all) X 
Type 
Show hidden types 
Location 
List view 
Location 

Click on + Add

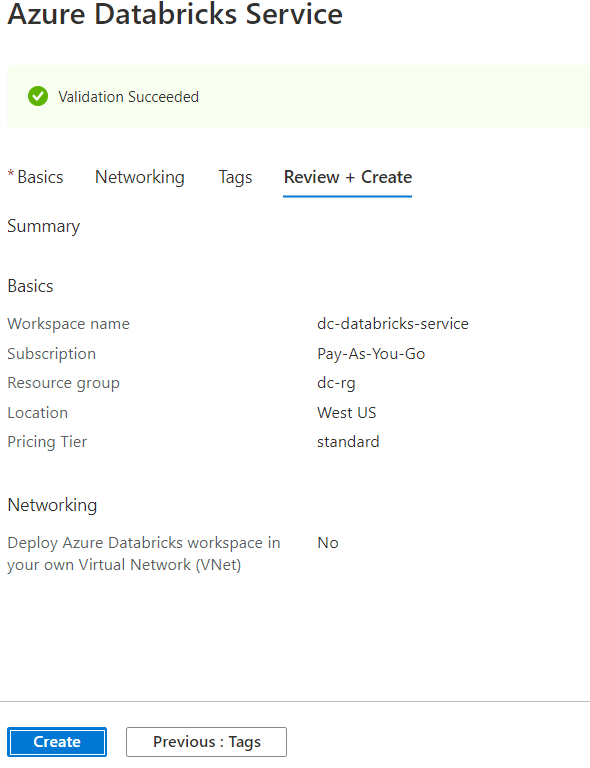
Search for Azure Databricks. When found, click on it

Machine generated alternative text:
Home > Resource groups > dc-rg > New > 
Azure Databricks 
Microsoft 
Overview 
Azure Databricks 
Microsoft 
Create 
C) Save for later 
Plans 
Usage Information + Support 

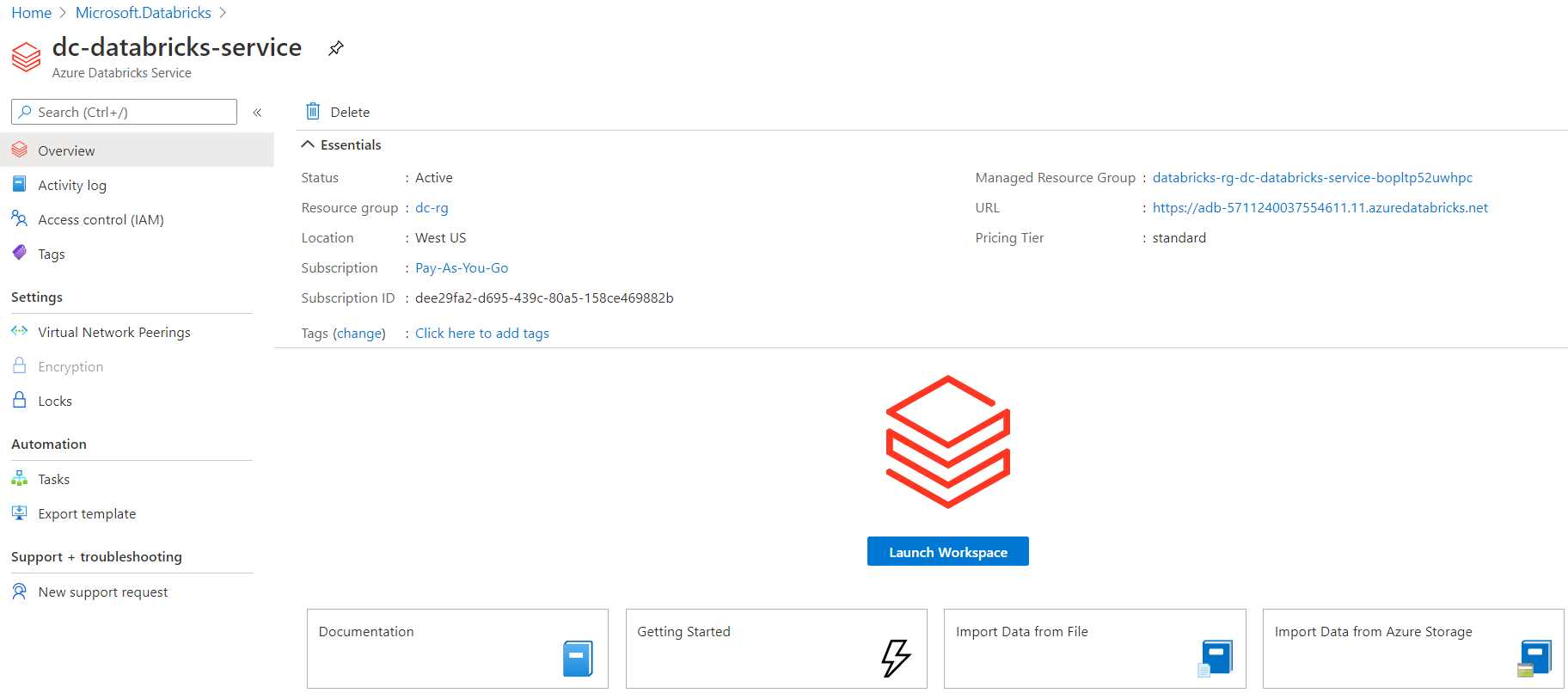
Create



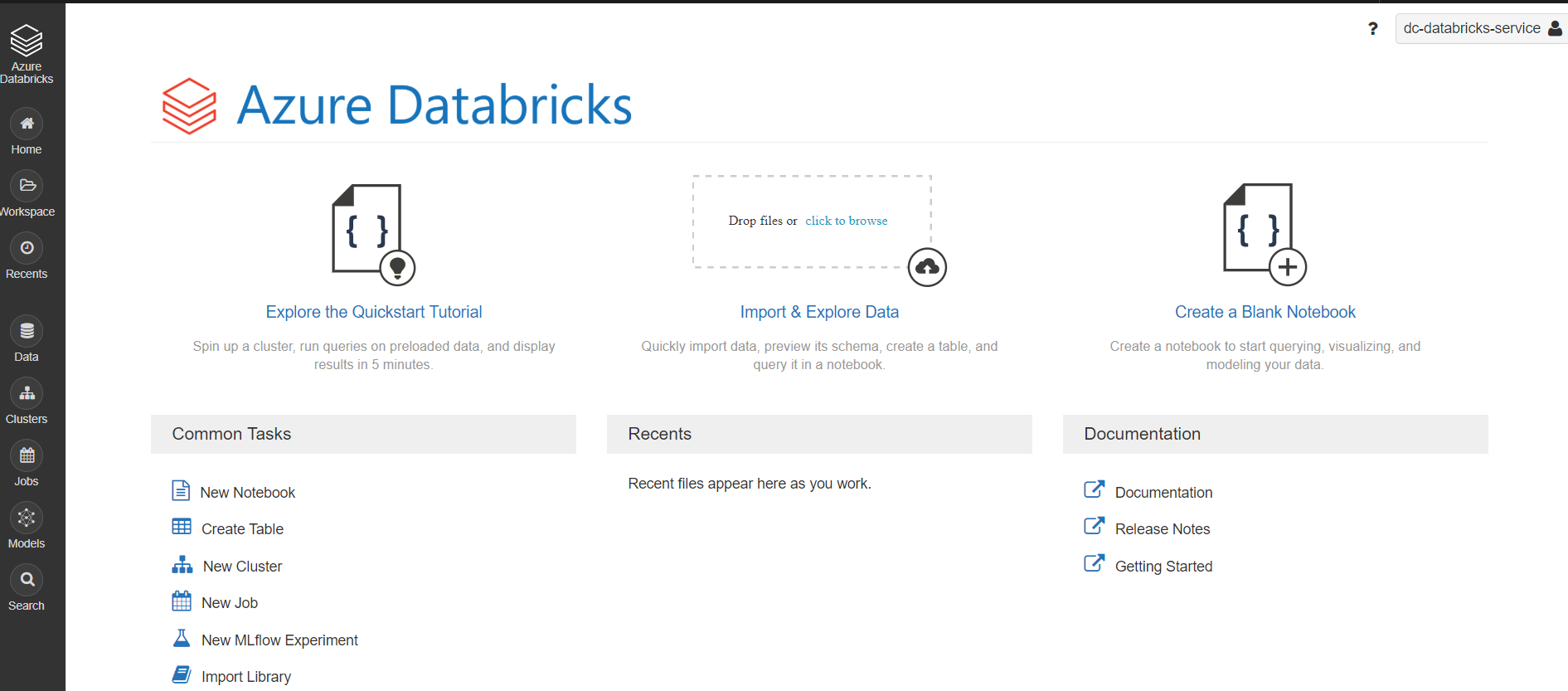
Select the Resource group, Workspace name, Location (should be the same as the resource group) and click on Review + Create



Click on Create



Click on Launch Workspace (in the middle of right panel)



Click on Clusters on the left side menu

Machine generated alternative text:
Clusters 
All-Purpose Clusters 
+ Create Cluster 
Name 
Job Clusters 
Pools 
State 
Nodes 
Runtime 
Driver 
All 
Worker 
Created by me 
Creator 
Accessible by me 
Actions 
Q 
Filter... 

Click on + Create Cluster

Machine generated alternative text:
Create Cluster 
New Cluster 
Cluster Name 
dc-databricks-cluster 
Cluster Mode O 
Standard 
Pool O 
None 
Cancel 
1-2 workers: 14.0-28.0 GB Memory, 4-8 cores, 0.75-1.5 OSU 
Create Cluster 
1 Driver: 14.0 GB Memory, 4 cores, 0.75 DSL' O 
Learn more 
Databricks Runtime Version O 
Runtime: 56 (Scala 2.117 spark 2.45) 
This Runtime version supports only Python 3 
New 
Autopilot Options 
O Enable autoscaling O 
erminate after 
30 
Worker Type O 
Standard DS3 v2 
Driver Type 
Same as worker 
Advanced Options 
minutes of inactivity O 
14.0 Gd Memory, 4 Cores, 
14.0 Gd Memory, 4 Cores, 
Min Workers 
Max Workers 
2 
0.75 OSU 
0.75 OSU 

Enter Cluster Name, select Databricks Runtime Version, enter Terminate after \_\_\_\_ minutes of inactivity

Select Worker type, Min Workers and Max Workers

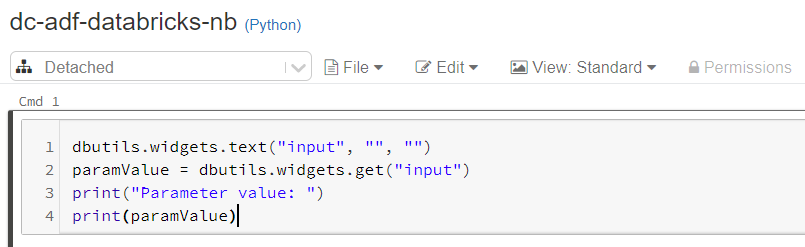
Click on Create Cluster (at the top)

After the cluster is created (takes about 20 minutes), click on Workspace --> Create --> Notebook

Machine generated alternative text:
Create Notebook 
Name 
Default Language 
Cluster 
dc-adf-databricks-nb 
Python 
dc-databncks-cluster 
Cancel 
Create 

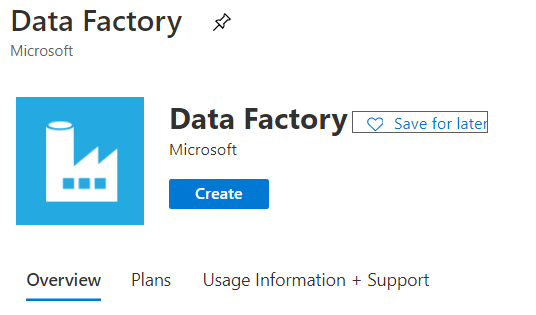
Enter Name, select Default Language, Cluster and click on Create

The script below shows how to receive and interpret Parameters

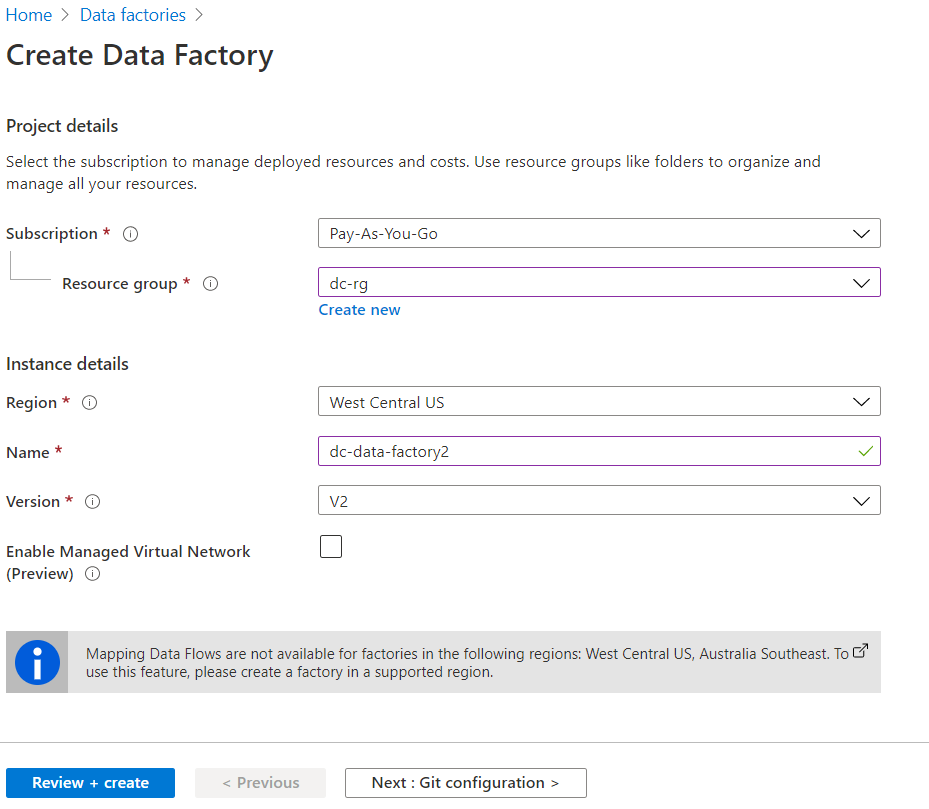


Now go back to Azure portal and create an instance of Azure Data factory v2.0

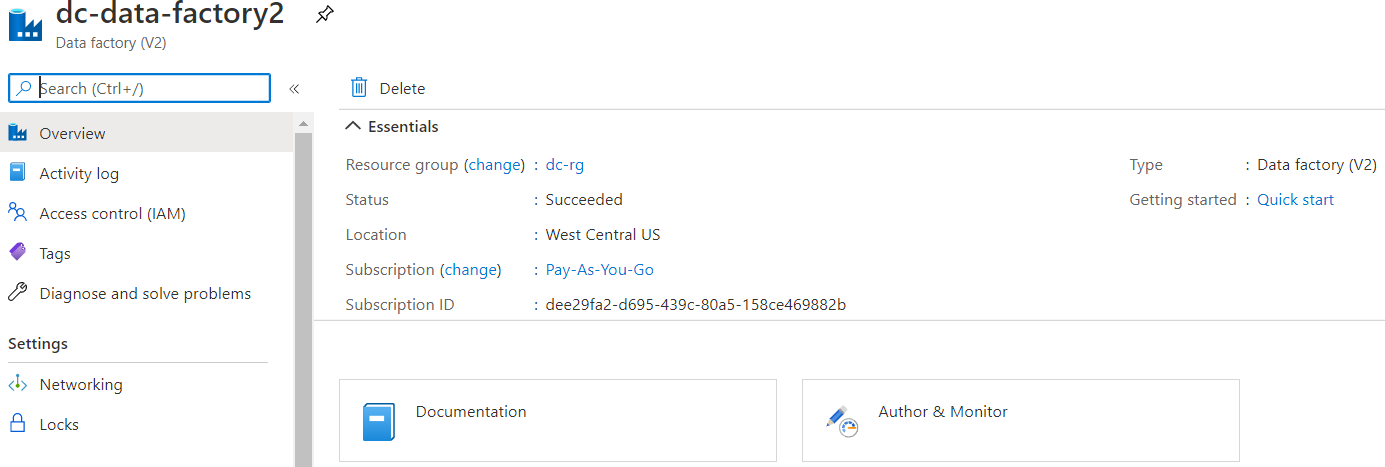
Search for Data factory and click on it



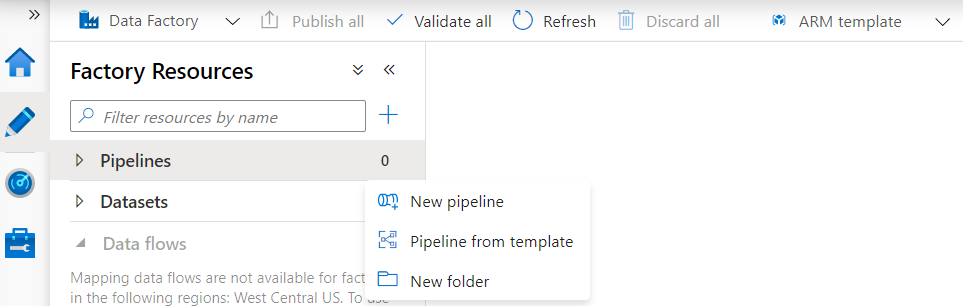
Click on Create



Click on Review + create

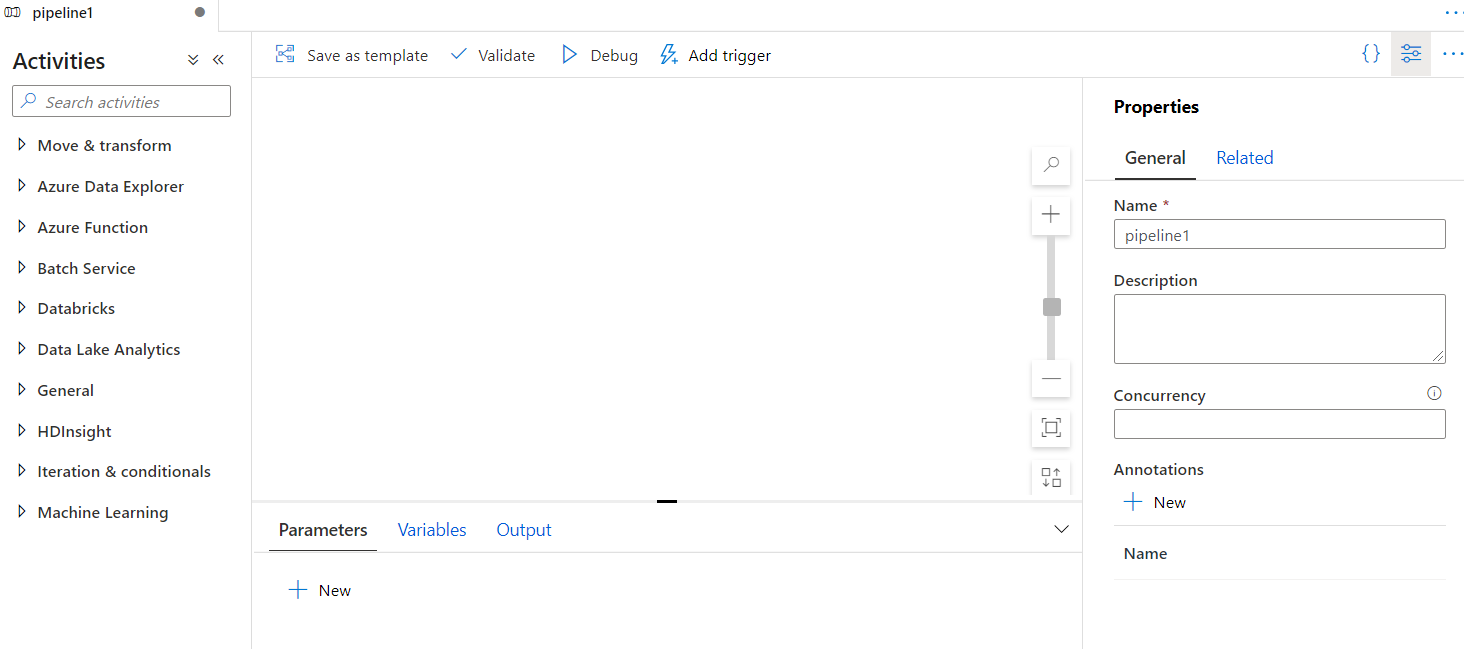


Click on Author & Monitor (in the center of right panel)

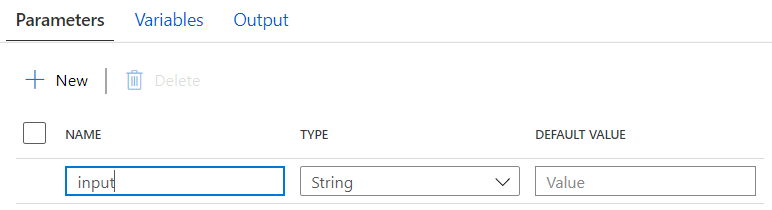


Click on Author --> Pipelines --> New pipeline

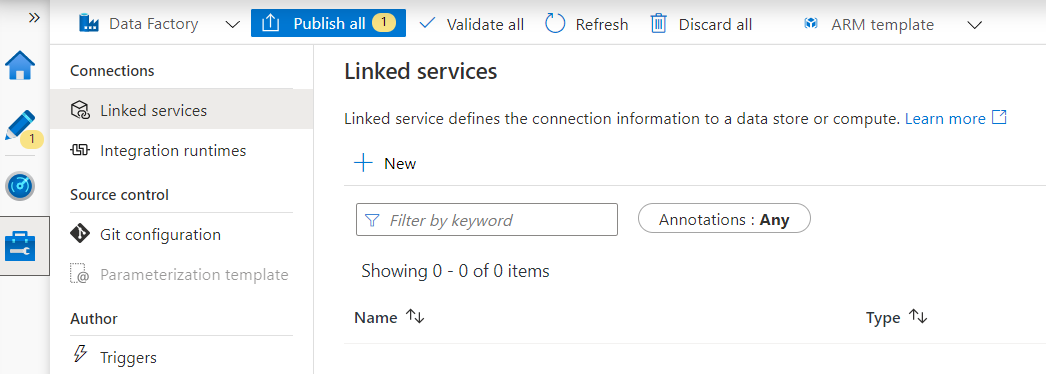
Note: in the exercise, we'll be sending simple parameter named input, of type string from ADF to Databricks. To do this, look at the bottom half of the right side panel



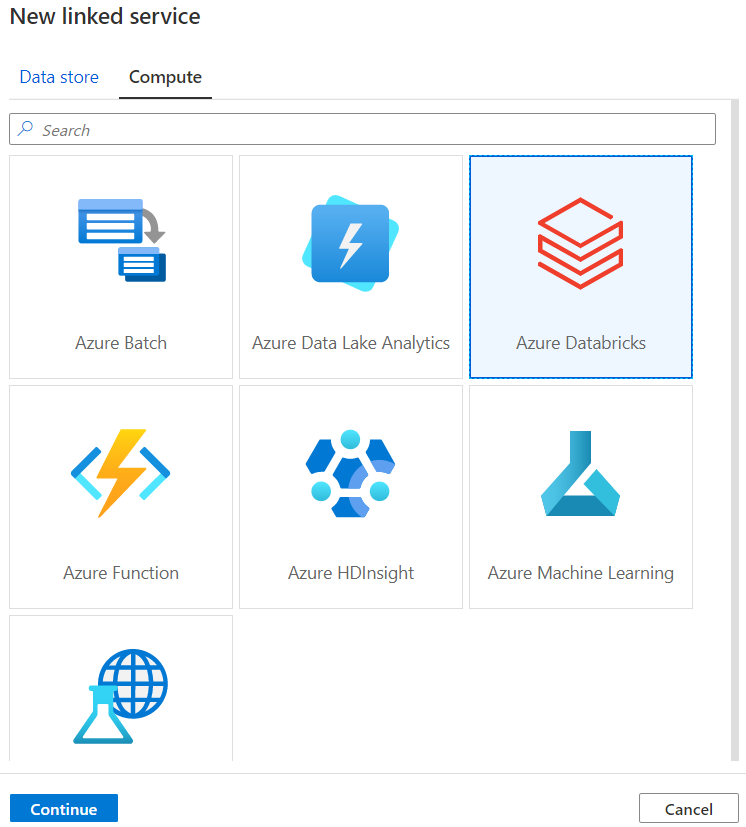
Under Parameters, click on + New



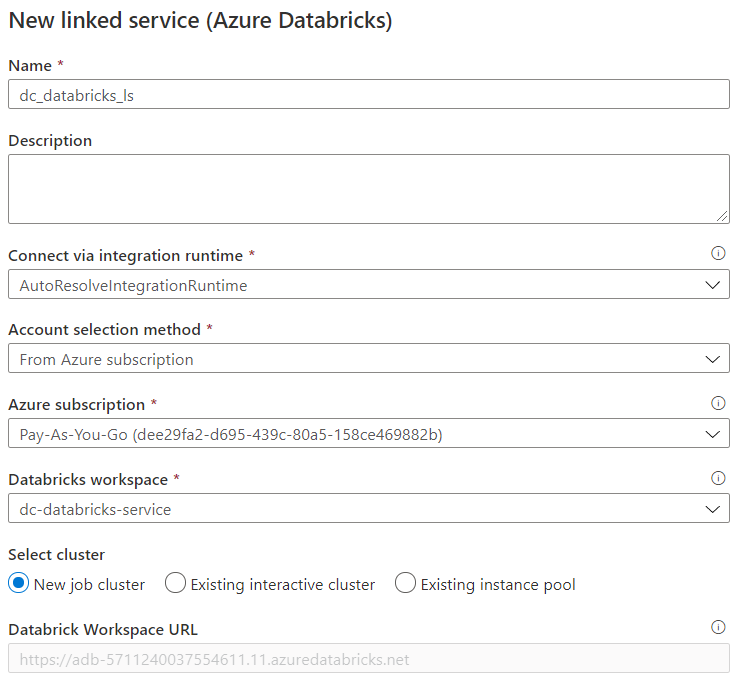
Now click on Manage menu item, on the left side

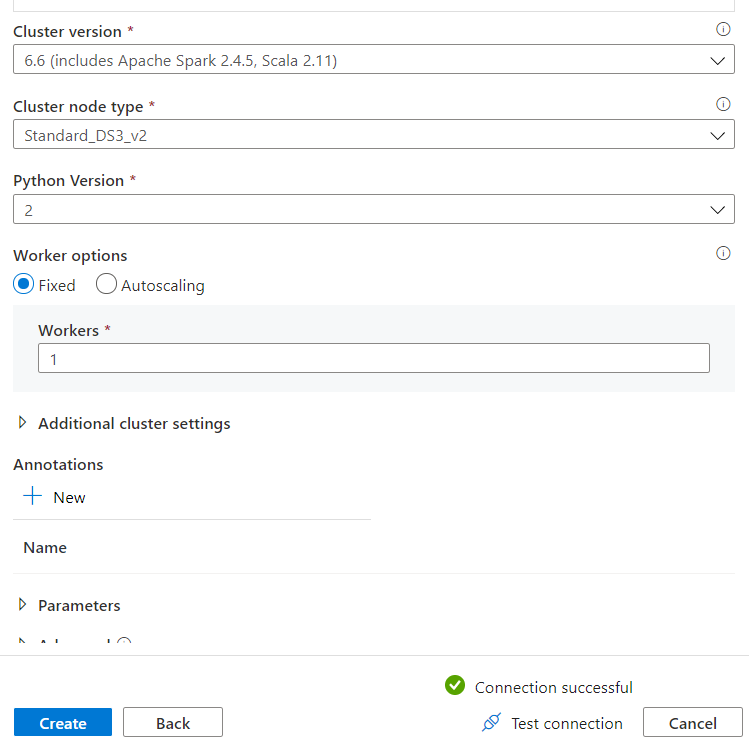


Click on + New under Linked services



Click on Azure Databricks and click on Continue

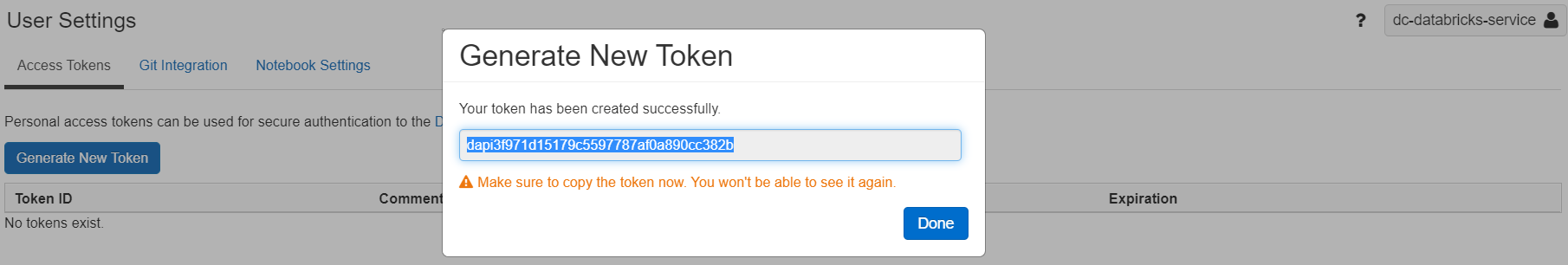




Enter Name and the rest of the parameter values

You will get Access token from the Databricks session user settings

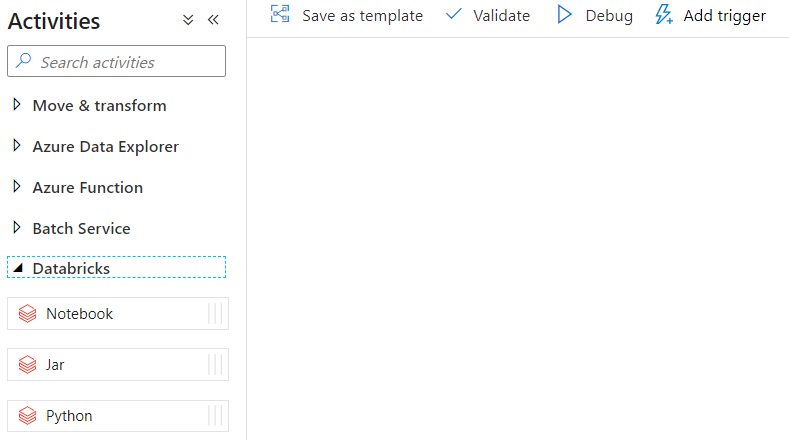
Click on user icon, at the top left corner, user settings, click on generate token, enter description and click on Generate



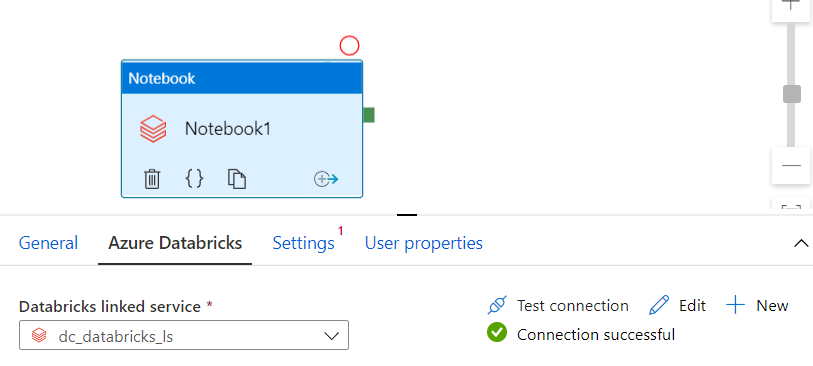
Save the token and enter it in the Access token field in the Linked service panel

Click on Create

Click on Author menu item and select your pipeline, e.g., pipeline1

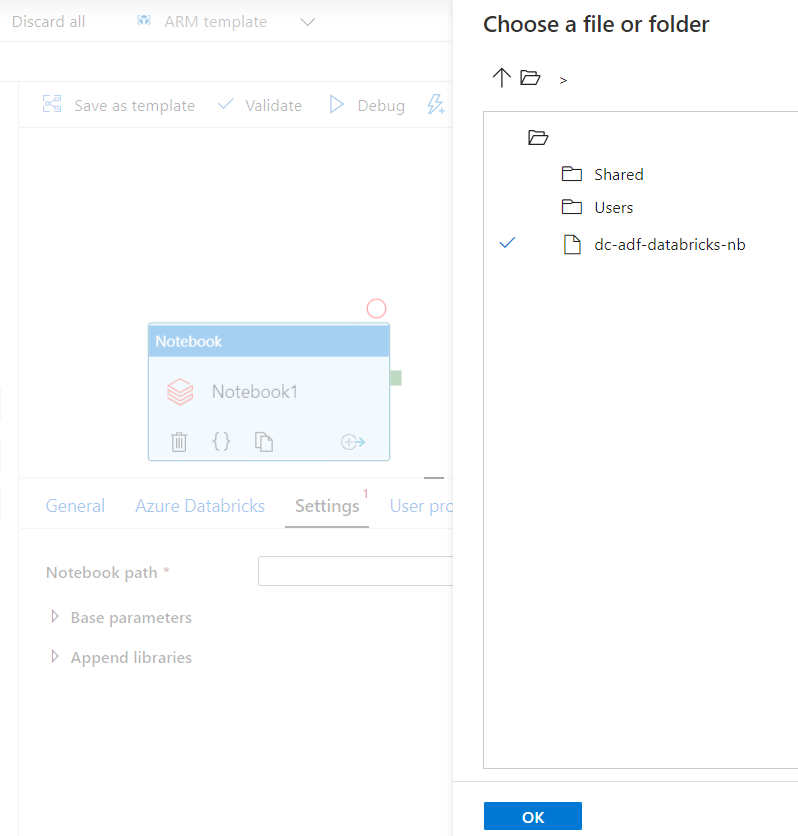


Under Activities, click on Databricks. Select and drag the Notebook icon and drop it in the pipeline panel

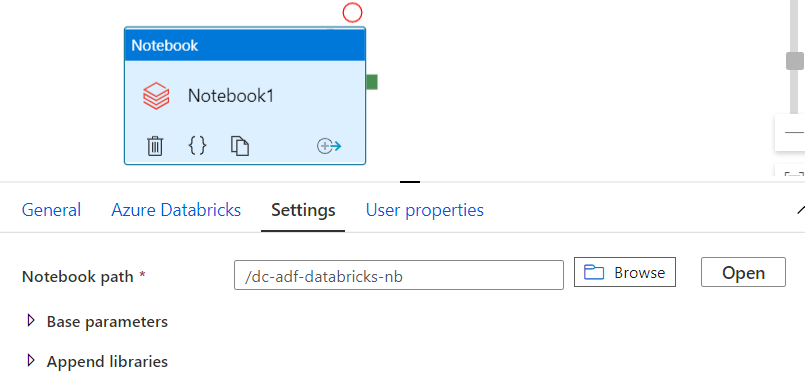


Click Azure Databricks and select Databricks linked service, Test connection

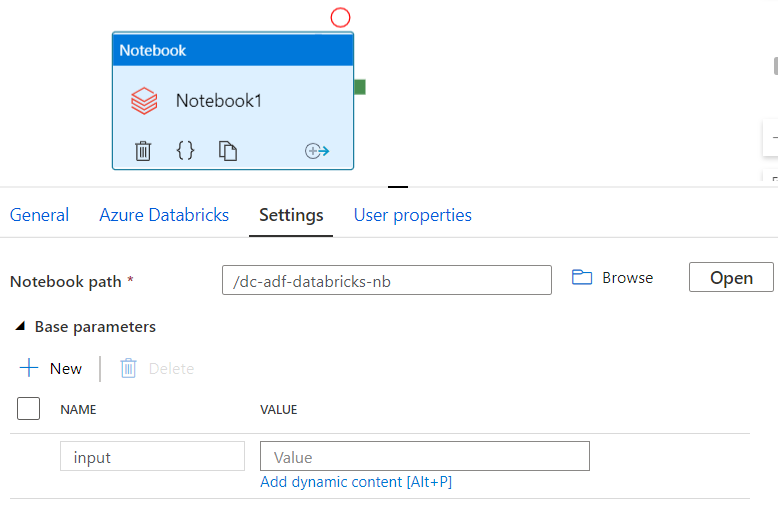
Click on Settings



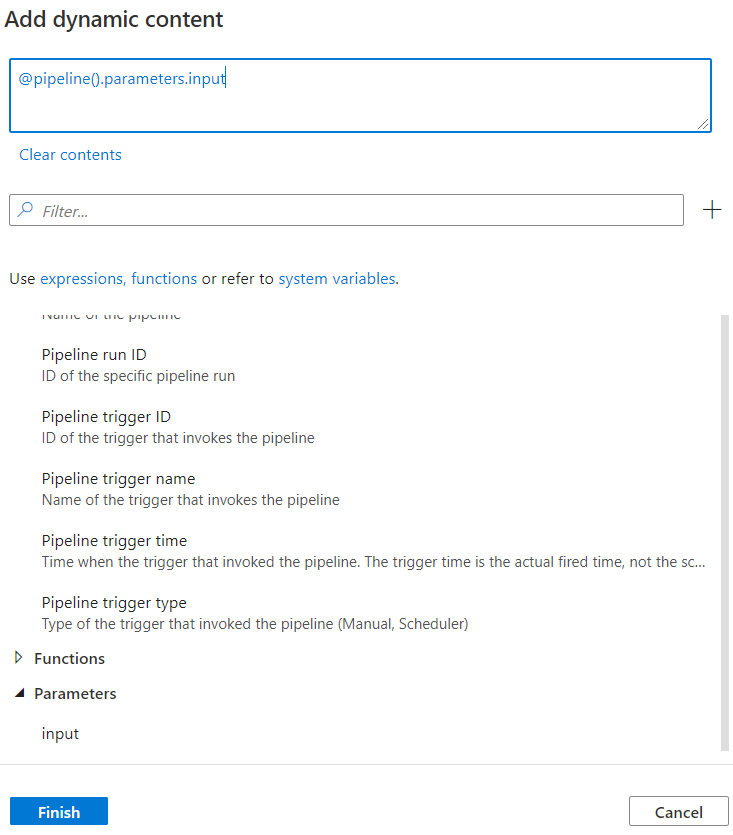
Click on the folder icon, follow it till the notebook is found. Click on Notebook and click on OK



Click on Base parameters

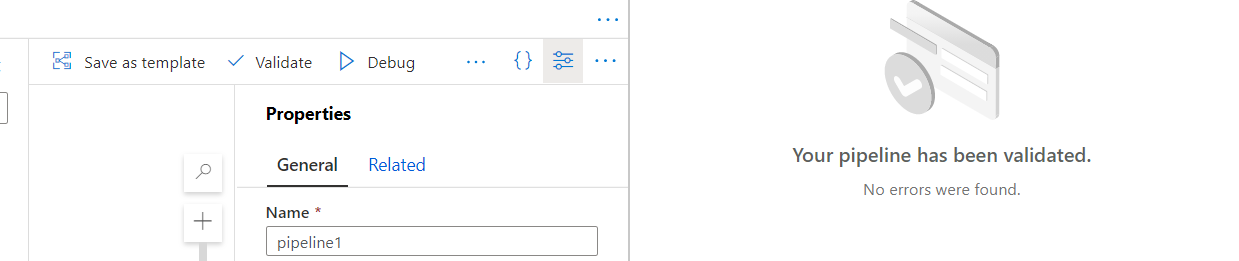


Enter input in the Name field and tab into the Value field. You will see Add dynamic content [Alt+P]. Click on it

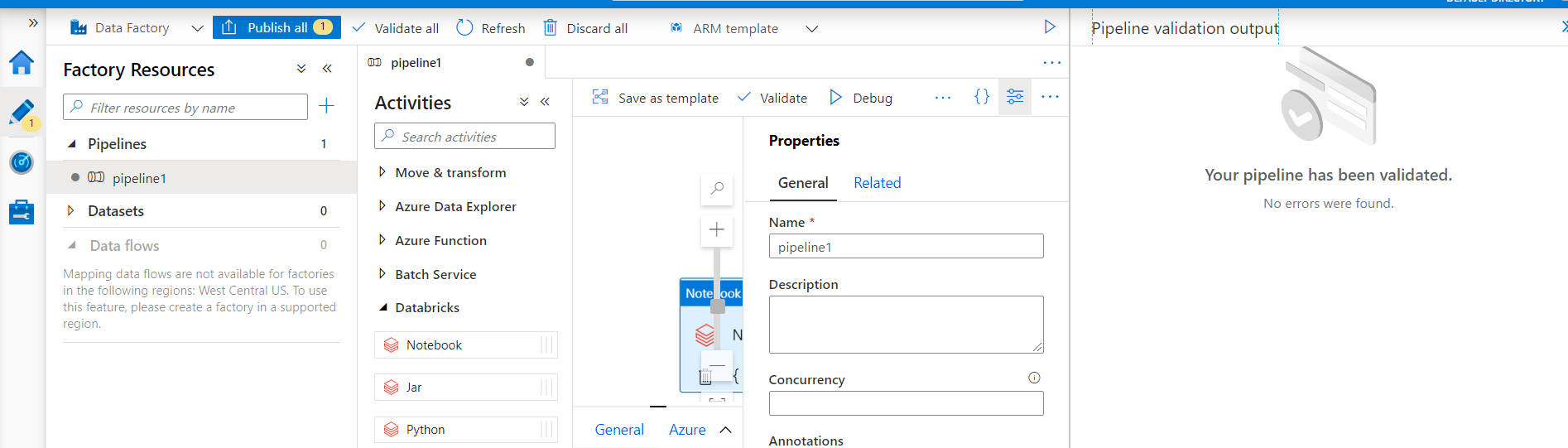


Go to Parameters section and click on input and you will the value in the contents field

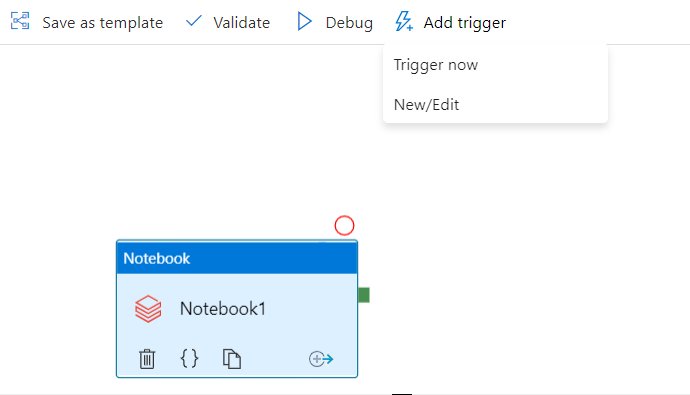
Click on Finish



Click on Validate menu item



Click on Publish and click on Publish in the detail panel



Click Add trigger --> click on Trigger now

Machine generated alternative text:
Pipeline run 
A Trigger pipeline now using last published configuration. 
Parameters 
TYPE 
string 
VALUE 
Toaster is great! 
input 
OK 
Cancel 

Enter some value in the input parameter and click on OK

To see the progress of the job, click on Monitor menu item on the left side and you will be the status if your job running. You click on Refresh icon.

Machine generated alternative text:
;fi6i Dashboards 
Runs 
Pipeline runs 
Trigger runs 
Runtimes & sessions 
Integration runtimes 
Data flow debug 
Pipeline runs 
Triggered Debug 
Y Search by run ID or name 
Showing 1 
- 1 items 
Piper 
Ine name 
pipelinel 
Rerun 
S Cancel 
Local time : 
Run start 
x/ CD Refresh 
Last 24 hours 
Edit columns 
Status : All 
Run end 
List 
Gantt 
Runs : Latest runs 
Duration 
Add filter 
Triggered by 
Manual trigger 
Status 
In progress 
Parameters 
Copy filters 
Annotations 
9/21/20, PM 

When don, you should see it as shown below:

Machine generated alternative text:
Pipeline runs 
Triggered Debug 
Y Search by run ID or name 
Showing 1 
- 1 items 
Piper 
Ine name 
pipelinel 
Rerun 
6) Cancel 
Local time : 
Run start 
x/ CD Refresh 
Last 24 hours 
Edit columns 
Status : All 
Run end 
9/21/20, PM 
List 
Gantt 
Runs : Latest runs 
Duration 
Add filter 
Triggered by 
Manual trigger 
Status 
e Succeeded 
Parameters 
Copy filters 
Annotations 
9/21/20, PM 

You can try various options

Machine generated alternative text:
Select cluster 
@ New job cluster 
Cluster version * 
6.6.x-scaIa2.1 1 
Cluster node type 
Standard DS3 v2 
Python Version 
2 
Worker options 
C) Existing interactive cluster 
C) Existing instance pool 
o 
o 
o 
@ Fixed C) Autoscaling 
Workers * 

For example, you can try different options in Select cluster field. You can try Worker options and number of Workers nodes.