Microsoft Data Science

Cortana Intelligence Suite – Azure Data Factory Lab

Lab Instructions

Welcome to the Cortana Intelligence Suite Workshop delivered by your Microsoft team. This experiment demonstrates how we can build a simple pipeline to analyze weblogs – we'll use the weblogs of the HDInsight on-demand cluster within the pipeline, sending that data out to Azure Blob storage where you can analyze it later.

ADF Process

1. Define Architecture: Set up objectives and flow
2. Create the Data Factory: Portal, PowerShell, VS
3. Create Linked Services: Connections to Data and Services
4. Create Datasets: Input and Output
5. Create Pipeline: Define Activities
6. Monitor and Manage: Portal or PowerShell, Alerts and Metrics

Business Goal: Transform and Analyze Web Logs each month

Design Process: Transform Raw Weblogs stored in a temporary location, using a Hive Query, storing the results in Blob Storage

Lab 1: Setup and file ingestion

1. Get your Azure Storage Account Name and Key - copy them to a notepad.exe file

2. Create a local directory on your system called **adfgetstarted** and copy the files **uploaddata.cmd**, **partitionweblogs.hql** and **input.log** to that local folder
3. Edit the **uploaddata.cmd** file, replace the <storageaccountname> item with your storage account name, and the <storagekey> item with your storage key. Save it and close it.
4. Open a command prompt, change to the **adfgetstarted** directory and run the **uploaddata.cmd** command. (if needed install AzCopy, http://aka.ms/downloadazcopy)

Lab 2: Create the ADF

1. Open Preview Portal: **https://ms.portal.azure.com/#**
2. After logging into the Azure Portal, do the following:
    a. Click **NEW** on the left menu.
    b. Click **Data + Analytics** in the **New** blade.
    a. Click **Data Factory** on the **Data + Analytics** blade.
2. In the New data factory blade, enter **<yourfirstandlastname>GetStartedDF** for the Name.
3. Select the Azure subscription where you want the data factory to be created.
4. Select existing resource group or create a new resource group. For the purpose of the tutorial, create a resource group named: **ADFGetStartedRG**.
5. Click **Create** on the New data factory blade.
6. You will see the data factory being created in the **Startboard** of the Azure Portal

Lab 3: Linked Services

1. Click **Author and deploy** on the **DATA FACTORY** blade for **GetStartedDF**. This launches the Data Factory Editor.
2. Click **New data store** and choose **Azure storage** - You should see the JSON script for creating an Azure Storage linked service in the editor.
3. Replace <accountname> with the name of your Azure storage account and <accountkey> with the access key of the Azure storage account.
4. Click **Deploy** on the command bar to deploy the storage linked service - After the linked service is deployed successfully, the Draft-1 window should disappear and you will see **AzureStorageLinkedService** in the tree view on the left.
5. In the **Data Factory Editor**, click **New compute** on the command bar and select **On-demand HDInsight cluster**.
6. Copy and paste the snippet below to the **Draft-1** window. The JSON snippet describes the properties that will be used to create the HDInsight cluster on-demand.

```
{
  "name": "HDInsightOnDemandLinkedService",
  "properties": {
    "type": "HDInsightOnDemand",
    "typeProperties": {
          "version": "3.2",
          "clusterSize": 1,
          "timeToLive": "00:30:00",
          "linkedServiceName": "AzureStorageLinkedService"
        }
      }
    }
```

----

7. Click **Deploy** on the command bar to deploy the linked service.
8. Confirm that you see both **AzureStorageLinkedService** and **HDInsightOnDemandLinkedService** in the tree view on the left.

Lab 4: Create Datasets

(Input Dataset)

1. In the **Data Factory Editor**, click **New dataset** on the command bar and select **Azure Blob storage**.
2. Copy and paste the snippet below to the **Draft-1** window. In the JSON snippet, you are creating a dataset called **AzureBlobInput** that represents input data for an activity in the pipeline. In addition, you specify that the input data is located in the blob container called **adfgetstarted** and the folder called **inputdata**.

```
{
  "name": "AzureBlobInput",
  "properties": {
    "type": "AzureBlob",
    "linkedServiceName": "AzureStorageLinkedService",
    "typeProperties": {
      "fileName": "input.log",
      "folderPath": "adfgetstarted/inputdata",
      "format": {
        "type": "TextFormat",
        "columnDelimiter": ","
      }
    },
    "availability": {
      "frequency": "Month",
      "interval": 1
    },
    "external": true,
    "policy": {}
  }
}
```

----

3. Click **Deploy** on the command bar to deploy the newly created input dataset. You should see the dataset in the tree view on the left.

(Output dataset)

Now, you will create the output dataset to represent the output data stored in the Azure Blob storage.

4. In the **Data Factory Editor**, click **New dataset** on the command bar and select **Azure Blob storage**.
5. Copy and paste the snippet below to the **Draft-1** window. In the JSON snippet, you are creating a dataset called **AzureBlobOutput**, and specifying the structure of the data that will be produced by the Hive script. In addition, you specify that the results are stored in the blob container called **adfgetstarted** and the folder called **partitioneddata**. The availability section specifies that the output dataset is produced on a monthly basis.

```
{
  "name": "AzureBlobOutput",
  "properties": {
    "type": "AzureBlob",
    "linkedServiceName": "AzureStorageLinkedService",
    "typeProperties": {
      "folderPath": "adfgetstarted/partitioneddata",
      "format": {
        "type": "TextFormat",
        "columnDelimiter": ","
      }
    },
    "availability": {
      "frequency": "Month",
      "interval": 1
    }
  }
}
```

----

6. Click **Deploy** on the command bar to deploy the newly created dataset.
7. Verify that the dataset is created successfully.

Lab 5:

1. In the **Data Factory Editor**, click Elipsis **(…) More commands** and then click **New pipeline**.
2. Copy and paste the snippet below to the **Draft-1** window. Important: Replace <storageaccountname> with the name of your storage account in the JSON.

```
{
   "name": "MyFirstPipeline",
   "properties": {
      "description": "My first Azure Data Factory pipeline",
      "activities": [
         {
            "type": "HDInsightHive",
            "typeProperties": {
               "scriptPath": "adfgetstarted/script/partitionweblogs.hql",
               "scriptLinkedService": "AzureStorageLinkedService",
               "defines": {
                  "inputtable":
"wasb://adfgetstarted@<storageaccountname>.blob.core.windows.net/inputdata",
                  "partitionedtable":
"wasb://adfgetstarted@<storageaccountname>.blob.core.windows.net/partitioneddata"
               }
            },
            "inputs": [
               {
                  "name": "AzureBlobInput"
               }
            ],
            "outputs": [
               {
                  "name": "AzureBlobOutput"
               }
            ],
            "policy": {
               "concurrency": 1,
               "retry": 3
            },
            "scheduler": {
               "frequency": "Month",
                    "interval": 1
                  },
                  "name": "RunSampleHiveActivity",
                  "linkedServiceName": "HDInsightOnDemandLinkedService"
                }
              ],
              "start": "2016-04-01T00:00:00Z",
              "end": "2016-04-02T00:00:00Z",
              "isPaused": false
            }
          }
```

----

3. Confirm the following:
    a. **input.log** file exists in the **inputdata** folder of the **adfgetstarted** container in the Azure blob storage
    b. **partitionweblogs.hql** file exists in the script folder of the **adfgetstarted** container in the Azure blob storage. Please complete the prerequisite steps in the Tutorial Overview if you don't see these files.
    c. Confirm that you replaced storageaccountname with the name of your storage account in the pipeline JSON.
4. Click **Deploy** on the command bar to deploy the pipeline. Since the start and end times are set in the past and *isPaused* is set to *false*, the pipeline (activity in the pipeline) runs immediately after you deploy.
5. Confirm that you see the pipeline in the tree view.

Lab 6: Monitor and Manage

1. Click **X** to close Data Factory Editor blades and to navigate back to the **Data Factory** blade, and click on **Diagram**.
2. In the **Diagram View**, you will see an overview of the pipelines, and datasets used in this tutorial.
3. To view all activities in the pipeline, right-click on pipeline in the diagram and click **Open Pipeline**.
4. Confirm that you see the **Hive activity** in the pipeline. To navigate back to the previous view, click **GetStartedDF** in the breadcrumb menu at the top.
5. In the **Diagram View**, double-click on the dataset **AzureBlobInput**. Confirm that the slice is in **Ready** state. It may take a couple of minutes for the slice to show up in Ready state. If it does not happen after you wait for some time, see if you have the input file (**input.log**) placed in the right container (**adfgetstarted**) and folder (**inputdata**).
6. Click **X** to close **AzureBlobInput** blade.
7. In the **Diagram View**, double-click on the dataset **AzureBlobOutput**. You will see that the slice that is currently being processed.
8. When processing is done, you will see the slice in **Ready** state.

Note:

Creation of an on-demand HDInsight cluster usually takes some time (approximately 20 minutes or more).

8. When the slice is in Ready state, check the **partitioneddata** folder in the **adfgetstarted** container in your blob storage for the output data.

Important:

The input file gets deleted when the slice is processed successfully. Therefore, if you want to rerun the slice or do the tutorial again, upload the input file (**input.log**) to the **inputdata** folder of the **adfgetstarted** container.