Orchestrating Big Data Solutions with Azure Data Factory

Lab 3 - Scheduling and Monitoring Azure Data Factory Pipelines

Overview

In this lab, you will use Azure Data Factory to schedule a data pipeline that copies online game data from files in Azure Blob Storage to a table in Azure SQL Database.

What You'll Need

To complete the labs, you will need the following:

- A web browser
- A Microsoft account
- A Microsoft Azure subscription
- A Windows, Linux, or Mac OS X computer
- The lab files for this course
- The Azure resources created in the previous labs

Important: If you have not completed Lab 1, or you have deleted the storage account, SQL database, and Azure Data Factory resources you created, complete Lab 1 now.

Exercise 1: Scheduling a Pipeline

In this exercise, you will create an Azure Data Factory pipeline to implement a daily copy operation for online game data.

Prepare the Database

The data your pipeline will copy contains details of points awarded to players in an online game. You will copy this data to a database table named **points**.

Note: The following steps assume that you are using the cross-platform SQL Server command line interface (mssql). You may use an alternative SQL Server client tool if you prefer.

1. Start your SQL Server client tool of choice, and connect to the **DataDB** database on your Azure SQL Database server (*server*.database.windows.net) using the server admin login credentials you specified when creating the Azure SQL database.

If you are using the cross-platform SQL Server command line interface, open a command line or console, and enter the following (case-sensitive) command, replacing *server* with your Azure SQL Database Server name, *login* with your server admin login name, and *password* with your server login password. Note that on Linux and Mac OS X operating systems, you may need to prefix any special characters (such as **$**) with a **** character (for example, if your password is *Pa$$w0rd*, enter *Pa\$\$w0rd*):

```
mssql -s server.database.windows.net -u login -p password -d DataDB -e
```

2. When connected to the database, enter the following Transact-SQL statement to create a table (note that when using *mssql*, commands must be entered on a single line):

```
CREATE TABLE points(id int IDENTITY, player varchar(20), time
varchar(5), points int);
```

3. Keep the SQL Server client tool open. You will return to it in a later procedure.

Verify Existing Linked Services

Azure Data Factory uses *linked services* to access, transform, and store data. In the case of your pipeline, you require linked services for the blob store account where the source data is stored, and the Azure SQL Database containing the table you want to load. You should have created these linked services in the previous lab.

1. In the Microsoft Azure portal, browse to the blade for your data factory, and click the **Author and deploy** tile.
2. In the pane on the left, expand **Linked Services** and note that linked services named **blob-store** and **sql-database** are already defined for the blob store and SQL database – these were created by the **Copy Data** wizard in the previous exercise.
3. Click the **blob-store** linked service to view its JSON definition, which should look like this:

```
{
 "name": "blob-store",
 "properties": {
  "hubName": "adf_name_hub",
  "type": "AzureStorage",
  "typeProperties": {
  "connectionString":
"DefaultEndpointsProtocol=https;AccountName=your_store;AccountKey=***"
  }
 }
}
```

4. Click the **sql-database** linked service to view its JSON definition, which should look like this:

```
{
 "name": "sql-database",
 "properties": {
  "hubName": "adf_name_hub",
  "type": "AzureSqlDatabase",
  "typeProperties": {
  "connectionString": "Data
Source=your_server.database.windows.net;Initial
Catalog=DataDB;Integrated Security=False;User
ID=SQLUser;Password=**********;Connect Timeout=30;Encrypt=True"
  }
```

```
    }
}
```

Upload Source Data

In this exercise, you will create a pipeline that copies data every day within a scheduled period. The source data will be published in a folder hierarchy that represents the year and month, and the file name will indicate the date. For the purposes of this lab, the pipeline period will be in the past (from January to June 2016); but the same principles apply to pipelines that operate over future timespans.

1. In the folder where you extracted the lab files, view the contents of the **gamedata\2016** folder, which contains subfolders named **00** to **06**, representing the months January to June in 2016. Each of the month-level folders, contains a text file for each day, named **01.csv**, **02.csv**, and so on, which contain records of points awarded to players in the online game.

   The folder hierarchy looks like this:

   - gamedata
     - 2016
       - 01
         - 01.csv
         - 02.csv
         - …
       - 02
         - 01.csv
         - 02.csv
         - …
       - 03
         - 01.csv
         - 02.csv
         - …
       - 04
         - 01.csv
         - 02.csv
         - …
       - 05
         - 01.csv
         - 02.csv
         - …
       - 06
         - 01.csv
         - 02.csv
         - …

2. Start Azure Storage Explorer, and if you are not already signed in, sign into your Azure subscription.
3. Expand your storage account and the **Blob Containers** folder, and then double-click the **adf-data** blob container you created in the previous lab.
4. In the **Upload** drop-down list, click **Folder**. Then upload the **gamedata** folder as a block blob to create a folder named **gamedata** in the root of the container.

Create Datasets

Datasets define schema for data flows, including data to be extracted from a source and data to be loaded into a destination (or *sink*). You must create datasets to define the text file data source and the database table to be used in your pipeline.

1. In the author and deploy page, in the pane on the left, click **More** and then click **New dataset**, and click **Azure Blob storage**. This creates a new draft JSON document to define the dataset for the source text data.
2. Replace the default JSON with the following code, (which you can copy and paste from **daily-gamedata-csv.json** in the folder where you extracted the lab files):

```
{
  "name": "daily-gamedata-csv",
  "properties": {
    "structure": [
      {
        "name": "player",
        "type": "String"
      },
      {
        "name": "time",
        "type": "String"
      },
      {
        "name": "points",
        "type": "Int32"
      }
    ],
    "published": false,
    "type": "AzureBlob",
    "linkedServiceName": "blob-store",
    "typeProperties": {
      "folderPath": "adf-data/gamedata/{Year}/{Month}",
      "fileName": "{Day}.csv",
      "format": {
        "type": "TextFormat",
        "columnDelimiter": ",",
        "firstRowAsHeader": true
      },
      "partitionedBy": [
        {
          "name": "Year",
          "value": {
            "type": "DateTime",
            "date": "SliceStart",
            "format": "yyyy"
          }
        },
        {
          "name": "Month",
          "value": {
            "type": "DateTime",
            "date": "SliceStart",
            "format": "MM"
```

```
          }
        },
        {
          "name": "Day",
          "value": {
            "type": "DateTime",
            "date": "SliceStart",
            "format": "dd"
          }
        }
      ]
    },
    "availability": {
      "frequency": "Day",
      "interval": 1
    },
    "external": true,
    "policy": {}
  }
}
```

This JSON defines a schema for comma-delimited text containing a string **player** column, a string **time** column, and an integer **points** column. The data is available in the **adf-data** blob storage container, in a folder hierarchy that reflects the current year and month, in a file named after the current day. New data will be available every day.

3. Click **Deploy** to deploy the dataset definition to your Azure Data Factory.
4. In the author and deploy page, in the pane on the left, click **More** and then click **New dataset**, and click **Azure SQL**. This creates a new draft JSON document to define the dataset for your SQL database table destination.
5. Replace the default JSON with the following code (which you can copy and paste from **dbo-points.json** in the folder where you extracted the lab files):

```
{
  "name": "dbo-points",
  "properties": {
    "type": "AzureSqlTable",
    "linkedServiceName": "sql-database",
    "structure": [
      {
        "name": "player",
        "type": "String"
      },
      {
        "name": "time",
        "type": "String"
      },
      {
        "name": "points",
        "type": "Int32"
      }
    ],
    "typeProperties": {
      "tableName": "dbo.points"
```

```
    },
    "availability": {
      "frequency": "Day",
      "interval": 1
    }
  }
}
```

This JSON defines a schema for a table named **dbo.points** containing an **id** column of integer values, a player column of string values, and a **points** column of integer values in the database defined by the **sql-database** linked service. The dataset can be updated every fifteen minutes.

6. Click **Deploy** to deploy the dataset definition to your Azure Data Factory.
7. In the pane on the left, expand the **Datasets** folder and verify that the **daily-gamedata-csv** and **dbo-points** datasets are listed.

Create a Pipeline

Now that you have defined the linked services and datasets for your data flow, you can create a pipeline to encapsulate it. A pipeline defines a series of actions that use linked services to operate on datasets. In this case, your pipeline will consist of a single action to copy data from the **blob-store** linked service to the **sql-database** linked service.

1. In the author and deploy page, in the pane on the left, click **More** and then click **New pipeline**. This creates a new draft JSON document to define the pipeline for your data flow.
2. Replace the default JSON with the following code, which you can copy and paste from **copy-daily-game-points.json** in the folder where you extracted the lab files. Note the **start** and **end** parameters, which specify the start and end of the time period in which the pipeline will run – these are expressed as times in the UTC time zone (which is the same as GMT).

```
{
  "name": "Copy Daily Game Points",
  "properties": {
    "activities": [
      {
        "type": "Copy",
        "typeProperties": {
          "source": {
            "type": "BlobSource",
            "recursive": false
          },
          "sink": {
            "type": "SqlSink",
            "writeBatchSize": 0,
            "writeBatchTimeout": "00:00:00"
          },
          "translator": {
            "type": "TabularTranslator",
            "columnMappings": "player:player,time:time,points:points"
          }
        },
        "inputs": [
          {
            "name": "daily-gamedata-csv"
```

```
        }
      ],
      "outputs": [
        {
          "name": "dbo-points"
        }
      ],
      "policy": {
        "timeout": "00:05:00",
        "concurrency": 5,
        "executionPriorityOrder": "OldestFirst",
        "retry": 1
      },
      "scheduler": {
        "frequency": "Day",
        "interval": 1
      },
      "name": "Copy points data"
    }
  ],
  "start": "2016-01-01T00:00:00Z",
  "end": "2016-06-30T23:59:59Z",
  "pipelineMode": "Scheduled"
 }
}
```

This JSON defines a pipeline that includes a **Copy** activity to copy the **daily-gamedata-csv** dataset in the blob store to the **dbo.points** table every day within start and end times. The **Copy** activity includes a policy that specifies a timeout of five minutes, and that failed activities should be retried once. Additionally, the policy specifies that datasets before the current time slice should be processed retrospectively on order of oldest first, using a concurrency level of five (so five overdue datasets will be processed at a time.)

3. Click **Deploy** to deploy the pipeline to your Azure Data Factory.

View Pipeline Status

Now that you have deployed your pipeline, you can use the Azure portal to monitor its status.

1. After the pipeline has been deployed, return to the blade for your Azure Data Factory, wait a few minutes for the **Pipelines** tile to indicate the addition of your pipeline (it may already indicate the pipelines you created in the previous lab).
2. Click the **Diagram** tile, and view the graphical representation of the **Copy Daily Game Points** pipeline. Note that the **daily-gamedata-csv** dataset may indicate an error – you will investigate this later.
3. Close the diagram, and on the blade for your Azure Data Factory, click the **Pipelines** tile.
4. Click the **Copy Daily Game Points** pipeline, and observe the state of the activities it contains – it may initially contain no activities, as the datasets are validated and the pipeline is prepared.
5. On the **Copy Daily Game Points** blade, click the **Consumed** tile to view the input datasets for this pipeline, and note that it consumes the **daily-gamedata-csv** dataset.
6. Click the **daily-gamedata-csv** dataset and note the status for each **slice** to be extracted from this dataset. You may need to wait as Azure Data Factory verifies the input files for each time-period to be processed. Note that this blade lists slices that have succeeded and slices that have failed.

7. On the **Copy Game Points** blade, click the **Produced** tile to view the output datasets for this pipeline, and note that it produces the **dbo-points** dataset.
8. Click the **dbo-points** dataset and note the status for each **slice** to be processed for this dataset.
9. Leave the pipeline running for around 15 minutes, and then re-check the status of the **Copy Daily Game Points** pipeline and the **daily-gamedata-csv** and **dbo-points** datasets.

Verify that Data is Being Copied

Now that your pipeline is running, you can verify that data is being copied to your SQL database.

1. In your SQL Server client tool, enter the following query:

   ```
   SELECT * FROM dbo.points;
   ```

2. Verify that the table now contains gamer points data, copied from the text files in your blob store.
3. Keep monitoring the pipeline, and re-running the query to verify that the data is being copied.
4. Close the SQL Server client tool:

   To exit *mssql*, enter the following command:

   ```
   .quit
   ```

Exercise 2: Monitoring and Troubleshooting a Pipeline

In this exercise, you will monitor the pipeline you created in the previous exercise.

View Activity Status

In addition to the blades in the Azure portal, Azure data Factory provides a **Monitor and Manage** tool for pipelines, that you can use to observe and troubleshoot pipeline activities.

1. In the Azure portal, in the blade for your Azure Data Factory, click the **Monitor and Manage** tile. This opens a new tab in your browser.
2. In the Monitor and Manage tool, view the pipeline diagram and the list of activity windows beneath it.
3. Above the diagram, click the start time and change it to 0:00 on January 1st 2016, and click **Done**. Then click the end time and change it to 0:00 on July 1st 2016, and click **Done**. Click **Apply** for the filter to take effect.
4. In the list of activity windows, click the **Status** column header, and in the list of possible status values, select **Ready**. This filters the list to show event windows where the output of the activity is ready for consumption.
5. Select any of the activity windows in the filtered list, and note that the **Activity Window Explorer** pane shows the time-slot for the selected activity window. Activity windows that are ready are shown as solid green squares. Note that Azure Data Factory has retrospectively started to run the activity windows that were scheduled to occur in the past.
6. Click the timeslot in the **Activity Window Explorer** pane, and note the status and other information that is displayed for the selected activity window.
7. In the list of activity windows, click the **Status** column header, and in the list of possible status values, select **Waiting**. This filters the list to show event windows that are waiting to be run.
8. Select any of the activity windows in the filtered list, and note that activity windows that are waiting to run are shown as solid orange squares.
9. Click the timeslot in the **Activity Window Explorer** pane, and note the status and other information that is displayed for the selected activity window. Your specified a concurrency level

of two, so the past-due slices are being processed two at a time, but many of the overdue activity windows are awaiting system resources before they can be run.

10. Click the **Status** column header again, and change the filter to show only **In progress** activity windows.
11. Select any of the activity windows in the filtered list, and note that activity windows that are waiting to run are shown as partially green squares. After Azure Data Factory has run all of the overdue activity windows, it will start to run the each future activity window at its scheduled time.
12. Remove the filter you have applied to the activity window list.
13. Select any of the activities, and scroll back and forward through the **Activity Window Explorer** pane to get a sense of the overall status of the pipeline. You can click the **Refresh this panel** button to update the status displayed.
14. Note that the activity window for January 13th shows a status of **Waiting** (a solid orange square), and the activity window for January 15th shows a status of **Failed** (a solid red square). You will troubleshoot this in the next procedure.

Troubleshoot Failed Slices

The Monitor and Manage tool can be particularly useful if you need to troubleshoot a failed activity slice.

1. In the list of activity windows, click the **Status** column header, and in the list of possible status values, select **Waiting: Dataset dependencies**. This filters the list to show event windows where a slice is awaiting a dataset.
2. Select the first failed activity window in the filtered list, which should be for January 13th, and note that the **Activity Window Explorer** pane shows the activity window as a solid orange square.
3. Click the highlighted activity window in the **Activity Window Explorer** pane, and note the status and other information that is displayed.
4. Expand **Upstream Dependencies**, and under **daily-gamedata-csv**, note the validation error that is displayed (you can widen the **Activity Window Explorer** pane or hold the mouse pointer over the error message to see it in a tooltip). The validation has failed because the source file could not be found.
5. Modify the filter on the **Status** column to select only activity windows with a status of **Failed**. This filters the list to show event windows where an activity window has failed.
6. Select the first failed activity window in the filtered list, which should be for January 15th, and note that the **Activity Window Explorer** pane shows the activity window as a solid orange square.
7. Click the highlighted activity window in the **Activity Window Explorer** pane, and note the status and other information that is displayed.
8. Under **Failed Execution**, click **More** to see the full error message for the failure. The failure is caused by an invalid value ("*ERROR*") in the **points** field.
9. In Azure Storage Explorer, browse to the **gamedata/2016/01** folder hierarchy, and verify that the **13.csv** data file is not present, then open the **15.csv** file and verify that it contains invalid data.

Resolve Processing Issues

Now that you know what the problems are, you can fix them and retry the failed slices.

1. Use Azure Storage Explorer to upload the **13.csv** file from the folder where you extracted the lab files to the **gamedata/2016/01** folder in your Azure blob store.

2. Use Azure Storage Explorer to delete the existing **15.csv** file in the **gamedata/2016/01** Azure blob storage folder (which contains invalid data). Then replace it with the **15.csv** file from the folder where you extracted the lab files (which contains valid data).
3. In the Monitor and Manage tool, in the **Activity Windows** pane, remove the filter you have applied, and in the **Activity Window Explorer** pane, select the orange square for the January 13th activity window (which is awaiting a dataset dependency).
4. In the **Activity Windows Explorer** pane, under **Activity Window**, in the **Rerun the selected activities** button drop-down list, click **Rerun**.
5. Select the red square for the January 15th activity window, which failed due to invalid data.
6. In the **Activity Windows Explorer** pane, under **Activity Window**, in the **Rerun the selected activities** button drop-down list, click **Rerun**.
7. Observe the **Activity Windows Explorer** pane as the activities run, clicking the **Refresh this panel** button occasionally to see the updated status – the squares will turn partially green, and then solid green (this may take some time).

Pause the Pipeline

You can use the Monitor and Manage tool to pause a pipeline. This can be useful if you need to investigate issues and you want to suspend the pipeline until you have resolved them.

1. In the Monitor and Manager tool, in the diagram pane, select the **Copy Daily Game Points** pipeline.
2. Above the diagram pane, click the **Pause and terminate activities** (☐) button. Then click **OK** when prompted.