**Course Overview and Schedule**

During this course, *IoT Data Analytics and Storage*, you will learn how to make the most of your live-stream and historical telemetry data produced by the IoT devices and sensors that support your business.

The first module begins with a general introduction to analytics and how it applies to Azure IoT. It next discusses concepts related to cold storage and how to set up Azure Data Lake for cold storage and analysis. Students begin working with an IoT scenario involving simulated wind farm data.

Module two picks up the wind farm scenario and uses it to illustrate concepts for warm storage. Students set up Azure Cosmos DB as an endpoint to receive data from Azure Stream Analytics jobs.

The third module examines the analytic capabilities of the Azure edge runtime. Students set up stream analytics, to run on a simulated edge device, and also examine how other Azure services – such as functions – can help process data on the edge.

In the final module, students take a deeper look at the stream analytics querying, routing and analysis capabilities. They work through labs that combine concepts from the first three modules and simulate changing device state, based on analytic information.

All of the information that you need in order to complete the course objectives is included in the course documents. The links to additional content that we've included are for those of you who are interested in digging deeper into the technologies.

**Students**

The target audience for this course is anyone interested in developing (or updating) data analytics skills for the IoT scenario. This course is well suited to existing data analysts who are new to IoT as well as people working in the IoT field who want to learn how to analyze their IoT data.

**Schedule**

*IoT Data Analytics and Storage* is a self-paced course that consists of four modules. All four of the modules are available when the course begins, and you can work your way through the modules at your own pace.

While each student will complete this course at a pace that suits their own requirements, we anticipate that an average student will be able to complete the lab assignments in about 10-15 hours. Please note that some of the labs are considerably longer than others due to the way that coding activities are distributed. You may find that the module 3 labs take longer to complete than the labs in the other modules.

**Module Structure**

This course is completely lab-based. There are no lectures or required reading sections. All of the learning content that you will need is embedded directly into the labs, right where and when you need it. Introductions to tools and technologies, references to additional content, video demonstrations, and code explanations are all built into the labs.

Some assessment questions will be presented during the labs. These questions will help you to prepare for the final assessment.

**Course Outline**

The *IoT Data Analytics and Storage* course includes four modules. Each of the four modules are described in the sections below.

**Module 1: IoT Analytics and Cold Storage**

* Lab 1: Configuring the Wind Farm Simulator
* Lab 2: Getting Started with Data Lake Storage and Analytics

**Module 2: Warm Storage**

* Lab 1: Getting Started with Warm Storage
* Lab 2: Implementing Business System Integration

**Module 3: Analytics on the Edge**

* Lab 1: Getting Started with IoT Edge
* Lab 2: Implementing Analytics on the Edge
* Lab 3: Deploying an Azure Function to the IoT Edge

**Module 4: Advanced Analytics**

* Lab 1: Constructing Analytics Queries
* Lab 2: Managing Analytics Topologies
* Lab 3: Device Management and Analytics

**Module Introduction - IoT Analytics and Cold Storage**

**Module Introduction - IoT Analytics and Cold Storage**

In this module, you will learn about

* Building an IoT architecture for analysis
* Building a fleet of virtual devices
* The role of cold storage in an IoT architecture

This module introduces you to the overall goals of building an IoT architecture. It then takes you through the process of setting up a fleet of virtual devices (wind farms) with the Azure Device Simulation Solution Accelerator. You then work with Azure Data Storage and Azure Data Lake Analytics to explore cold storage concepts and to perform big data analytics.

**Note**: Be sure to complete the lab configuration tasks at the end of Module 0.

During this module, you will complete the following hands-on labs:

* Lab 1: Configuring the Wind Farm Simulator
* Lab 2: L02-Getting Started with Data Lake Storage and Analytics

**Introduction to the Wind Farm Scenario**

In this scenario, you are a developer/architect for Contoso Wind Power. Contoso Wind Power (CWP) is a renewable energy company that was recently spun off from multi-national conglomerate Contoso Holdings. CWP owns and operates a number of wind farms in North America and Europe. In the United States, CWP has operating wind farms in California, Wyoming, Minnesota and Oklahoma.

As a result of the spin-off, CWP can no longer use the infrastructure and applications of its former parent company, so it is currently building new IT systems. The goal is to establish a cloud back-end that will enable CWP to integrate their worldwide operations.

You are working with a business analyst at Contoso to determine back-end data storage and analytics requirements. Each wind turbine is equipped with over 600 sensors that emit telemetry data 24 hours a day, and as a result, your wind turbines are producing massive amounts of data. Unfortunately, as of today this data is underutilized, with a limited number of sensors being monitored locally to ensure that threshold values are not exceeded.

After an initial investigation, you have agreed that the full range of data being produced by your turbines should be stored and analyzed, but you don’t have a clear idea of which sensor types will provide the most value to your business in the long term. Although the streaming data is not currently being captured or analyzed, temporal data is being sent to legacy-style csv files and can be imported from local repositories.

Note: We will simplify the sensor/telemetry scenario for the labs in this course.

You have created a plan to persist the data to cold storage in the cloud. That will allow you to do more in-depth analysis, and it will foster data exploration because – even with a huge data volume – you will be able to capture the raw data and run analysis jobs on it. Given the emergence of big data analytical tools, the opportunities are vast. You are also planning to import the legacy csv files into your cold storage, to provide even more historical data and context. You will work on these skills in this module – module 1 of the course.

Once you and your team have gained insights about the data and have a better idea of what it contains, you will save data to warm storage. Warm storage allows more immediate access to the streaming data. You still have the ability to do deeper analysis on warm storage data – it is typically more feasible to aggregate data and to perform richer queries on warm storage data than on streaming data. But with correct configuration, your data can be fresh enough to take timely action. You will learn about warm data technologies and strategies in module 2 of this course.

Lastly, you will leverage your streaming analysis tools to react to real-time telemetry data. You will build on the analysis you performed on your cold and hot data, in order to execute actions based on real-time data with Azure Stream Analytics and Edge Stream Analytics. Both module 3 and module 4 will cover topics related to real-time data. Module 3 will deal with analytics on the edge - analytics with devices capable of running analytics jobs. And finally module 4 will take you through deeper analytics topics.

**Run the Device Simulation Solution Accelerator**

Microsoft provides a flexible solution for creating simulated IoT data. It is part of the Microsoft Azure IoT Solution Accelerators located at https://www.azureiotsolutions.com/Accelerators. The solution accelerators are open source applications that cover many common IoT scenarios. Examining them all is outside the scope of this course, but they are worth studying on your own. The accelerators are production-grade, built with the idea that companies can use them as a starting point and adapt them into their own production environment.

**Note**: This course is focused on the analyzing data, rather than the mechanisms for producing that data, so we will start by setting up a system for generating simulated data. This will give you the opportunity to generate realistic quantities of IoT telemetry data, without having to purchase and configure a fleet of IoT hardware.

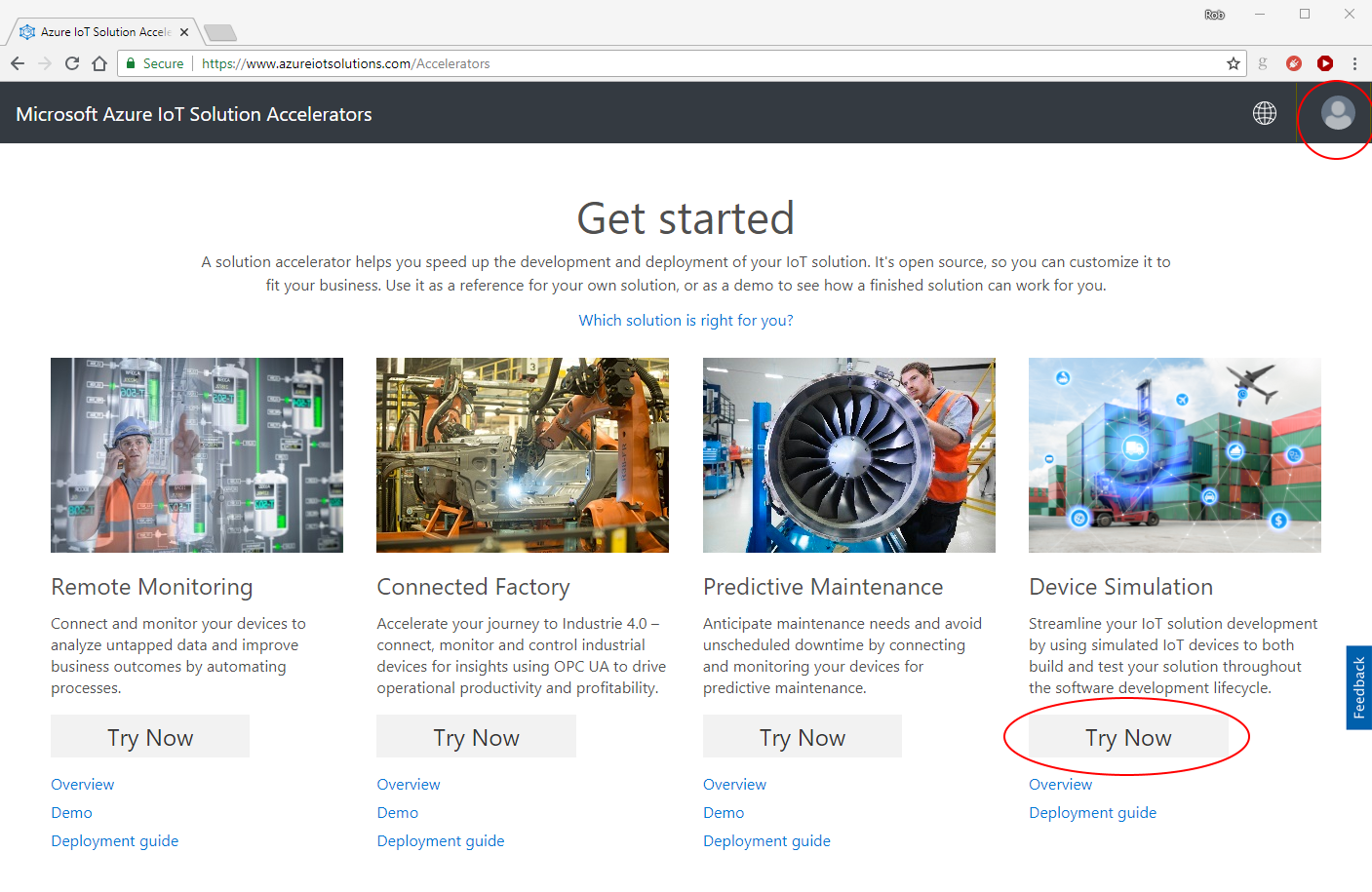
In this task, you will set up a solution accelerator and use it to generate simulated data for the wind farm scenario.

**Note**: You can come back to this task in subsequent modules if you need to generate more streaming data.

1. Navigate to the Solution Accelerator website at [Accelerators](https://www.azureiotsolutions.com/Accelerators).
2. In the upper-right corner of the web page, click the Account icon.
3. On the list, click **Sign in**

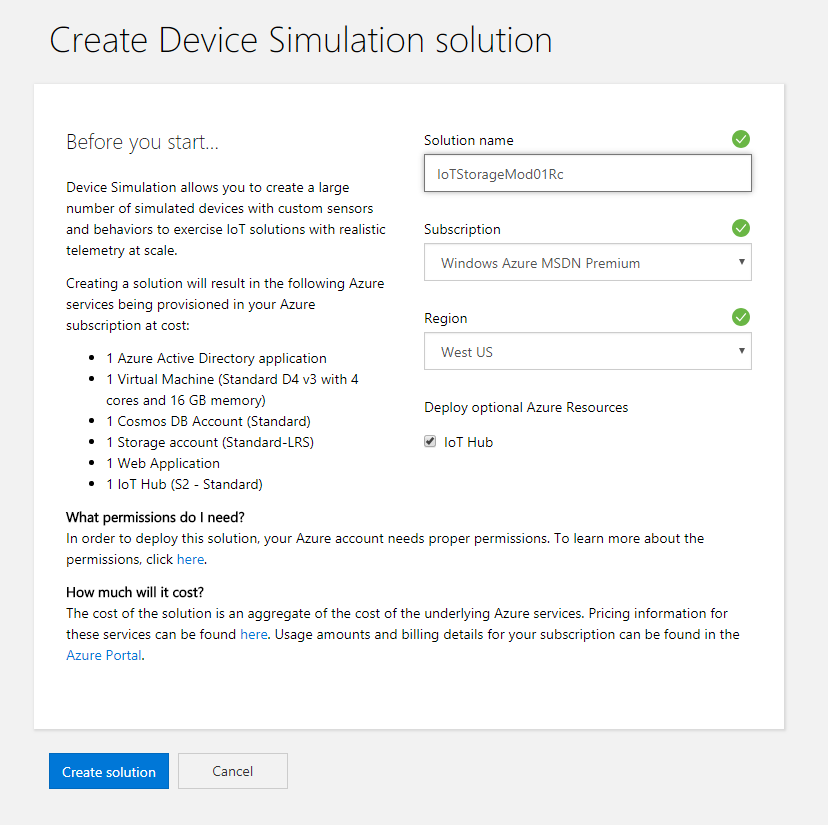
**Note**: You will need to use the same account for the accelerators that you use for your Azure subscription.

Once you are signed-in you should see your name appear next to Account.



1. On the right side of the page under Device Simulation, click **Try Now**

A page will be displayed that you can use to configure a solution accelerator.



1. Under **Solution name**, enter a unique name for your solution.

You need to provide a name that's globally unique. You will also want the name to be descriptive of your project and easy to remember. For this course we suggest using something like “DEV326xMod01” followed by your initials and date of birth.

For example: **DEV326xMod01CAH050961**

1. Under **Subscription**, use the list to select the Azure subscription that you are using for this course.
2. Under **Region**, select the Azure region nearest you.
3. To provision an IoT Hub for this task, click **IoT Hub**

We recommend selecting the **Iot Hub** checkbox, so that you provision an IoT hub just for this task. You are welcome to leave it unchecked if you want to use an IoT hub from previous work you have done.

**Note** The accelerator will create a large number of resources in your Azure subscription, and these resources do have an associated cost.

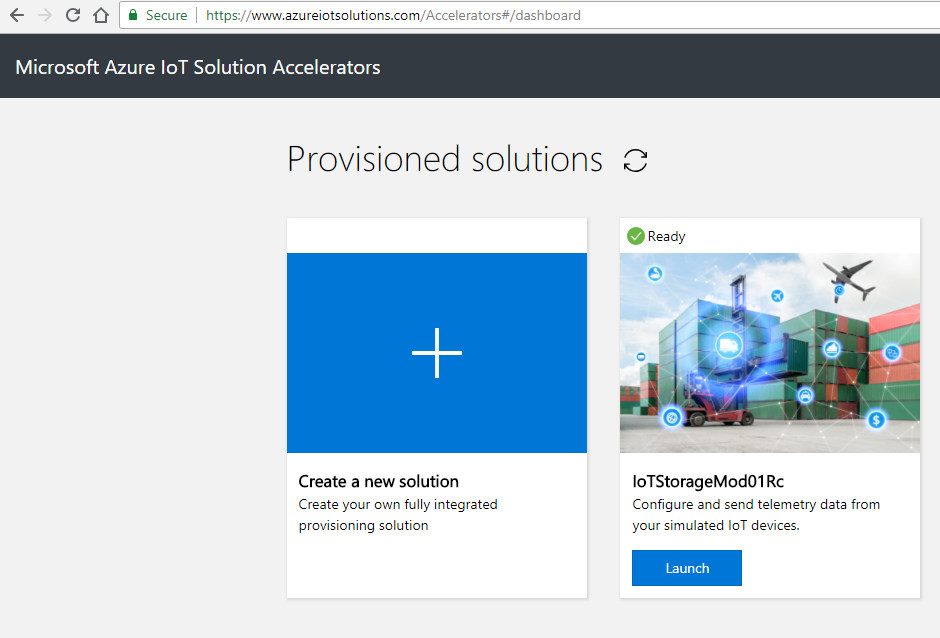
1. Click **Create solution** The accelerator dashboard will show a spinning graphic that says **Provisioning**

**Note** You can click **Details** while this is displayed to see which individual resources are being provisioned. When the accelerator is finished provisioning, you will be presented with a dashboard that allows you to navigate to your solution accelerator.

**Note**: It can take 10 minutes or more to provision the resources. You can use the time to study how other Microsoft Azure IoT Solution Accelerators are constructed, or you can just take a quick nap.

1. On your solution accelerators dashboard page, click **Launch**

**Note**: If your dashboard is not displayed automatically, navigate to <https://www.azureiotsolutions.com/Accelerators#/dashboard>, and then click **Launch**



1. To allow the solution accelerator to read your account profile, click **Accept**

You must grant permission to access your account (unless you have previously provisioned an accelerator).

You will be redirected to a Simulation setup page where you will configure your simulator.

1. On the Simulation setup page, Under Device model, in the **Select model** list, click **Custom**

Notice that you have the option to select one of several pre-configured device scenarios. Since we will be simulating wind turbines for Contoso Wind Energy, the Custom option allows us to simulate the sensor types that match our scenario.

You can now add sensors to your custom IoT device. For each sensor, you supply a name, a behavior (how the telemetry values change over time), min/max telemetry values, and a unit of incremental change for the value.

In the real world, a single wind turbine has hundreds of sensors and if you needed simulated data that accurately mimicked your actual configuration you might add a large percentage of them to your simulator. Luckily we will not be doing that for this course, and since wind turbine sensors fall into some major categories, we'll consolidate the sensors used in this simulation.

1. On the **Add Device Model** page, under **Device Model Name** enter **Wind Turbine**.
2. Under **Version**, enter one
3. Under **Telemetry Data**, use the fields provided to add the sensor values listed in the table below.

**Note**: When you finish entering a row of values for a sensor, to add another row, click **+ Add data point**

| **Sensor** | **Behavior** | **Min** | **Max** | **Unit** |
| --- | --- | --- | --- | --- |
| Bearings Temperature | Random | -30 | 65 | Celsius |
| Windings Temperature | Increment | -30 | 65 | Celsius |
| Tower sway | Random | 10 | 50 | g |
| Position sensor | Random | 2 | 30 | degrees |
| Gearbox fluid levels | Decrement | 1 | 1000 | ml |
| Blade strain gauge | Random | 25 | 5,000 | psi |
| Main shaft strain gauge | Random | 25 | 10,000 | psi |
| Shroud accelerometer | Random | 0 | 1,800 | rpm |
| Power Generation | Increment | 0 | 750 | kW |

When you are finished, your new **Wind Turbine** will show up in the list of device models.

1. Click **Simulations** on the left side of the window.
2. Click **Create Simulation**
3. In the **Simulation Setup** window, under **Name**, enter **Turbine Simulation**
4. Under **Simulation duration**, click **End in:**

We want the simulation to end after running for 10 minutes. That will produce about (50 devices x 9 sensors x 10 minutes x 6 readings per minute) = 27,000 data points to work with. You do not want to run this simulation indefinitely.

**Note**: Do not run this simulation with the Simulation duration set to *Run indefinitely*.

1. Under **MM**, type **10**
2. Under **Device Model**, click **+ Add a Device Type**
3. Under **Name**, choose your **Wind Turbine** from the list.
4. Under **Amount**, to provision 50 devices, type **50**

As noted, you can provision as many as 20,000 devices. For this module, we’ll start with a manageable number.

1. Notice that **Telemetry frequency** is specified in hours, minutes, and seconds.

Time frequency is a critical part of telemetry data. You always face the trade-off between granularity and volume. The more frequently data is sampled, the more accurate it is over time, but the quantities of data can get prohibitively large. When deciding on time frequency, analyze the underlying data. For instance, the frequency of readings for tectonic plate movement would not need to be as frequent as readings of molecular entropy. In the case of this wind turbine simulation, having the sensors report every 10 seconds is reasonable.

1. Under **SS**, to set the telemetry frequency to 10 seconds, type **10**
2. Under **Target IoT Hub**, ensure that **Use the pre-provisioned IoT Hub** is selected.

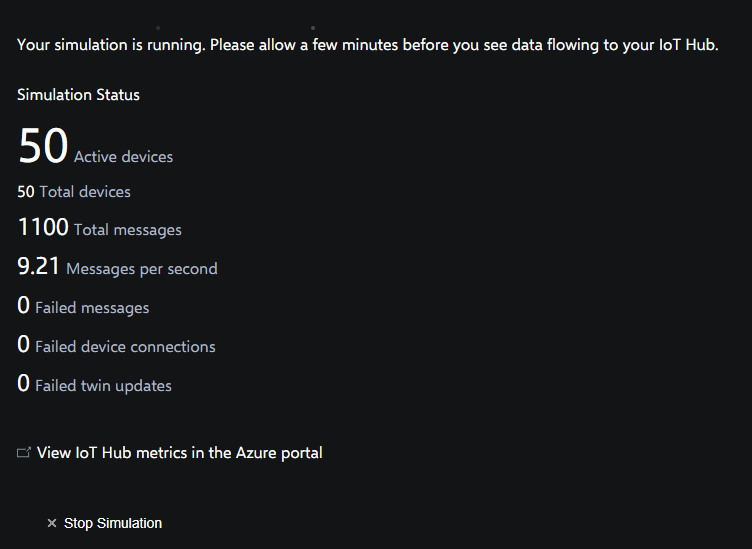
If you checked the **Iot Hub** checkbox when configuring your solution accelerator, the **Use the pre-provisioned IoT Hub** option should be selected by default. If you didn't check the **IoT Hub** checkbox, or if you want to use an existing IoT Hub, you can provide the connection string to the IoT Hub that you created previously.

1. To start the simulation, click **Start Simulation**
2. Create a record of the time when you started the simulation.

You will need to know the start-time when you are sampling or extracting the data later in the course.

1. Scroll down to view the progress of your simulation.

On your Simulation Setup page, you will start to see the simulated telemetry data flow from the device simulation to the IoT Hub in your Azure account.



1. Wait 3-4 minutes, and then click **View IoT Hub metrics in the Azure portal**

The link will open a new window that displays a view of your Azure portal. Specifically, the Metrics blade of the IoT Hub service.

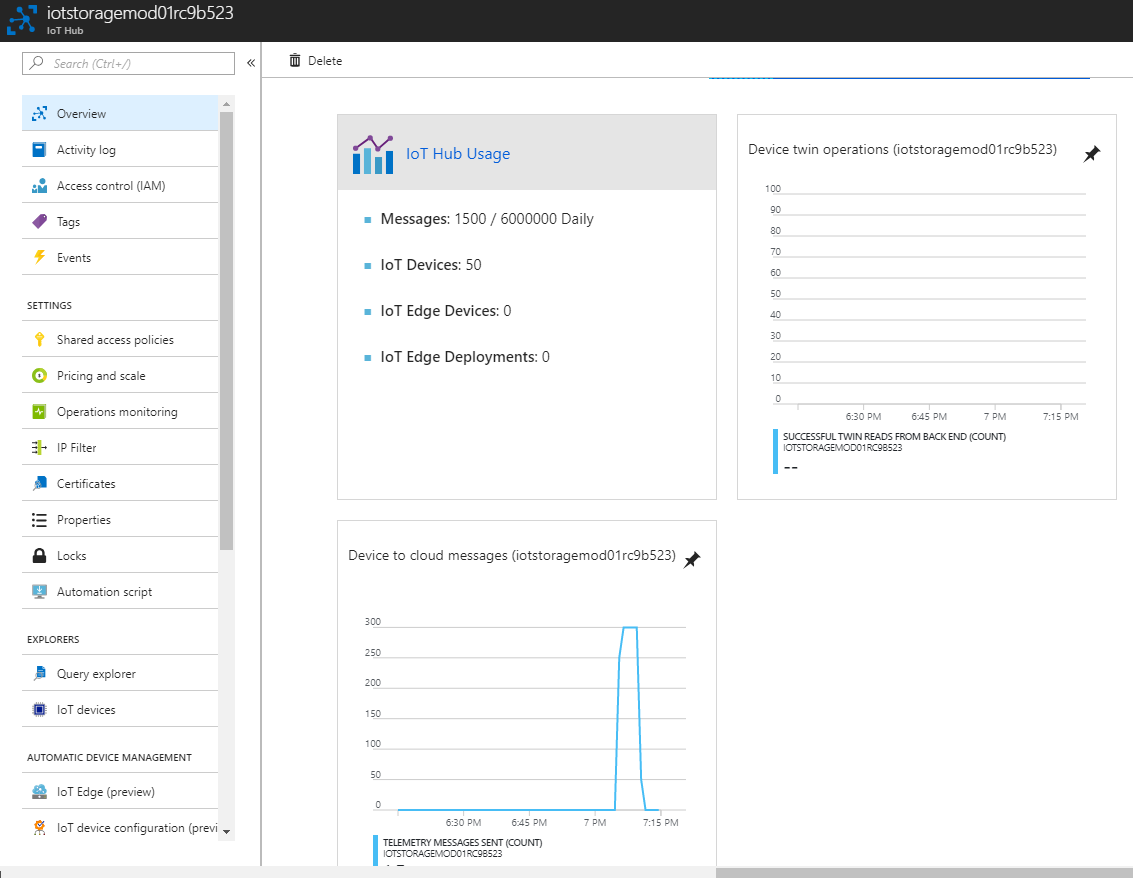
1. On the Metrics blade of your IoT Hub service, take a minute to review which metrics have been selected for display.

The Metrics blade gives you another view of the telemetry messages as they are ingested by your IoT Hub. Understanding how the IoT Hub fits into the IoT architecture is critical. The IoT hub provisions and manages devices. It also receives and stores IoT data. The hub then acts as a data source for the other components that store and analyze data.

1. Wait until the device simulation job has completed before continuing.

After about 10 minutes has passed, the number of “Connected Devices” will drop off, indicating that the simulation is finishing up.

1. On the menu to the left side of your IoT Hub blade, click **Overview**



1. On the left-side menu of your Azure portal, click **Resource groups**

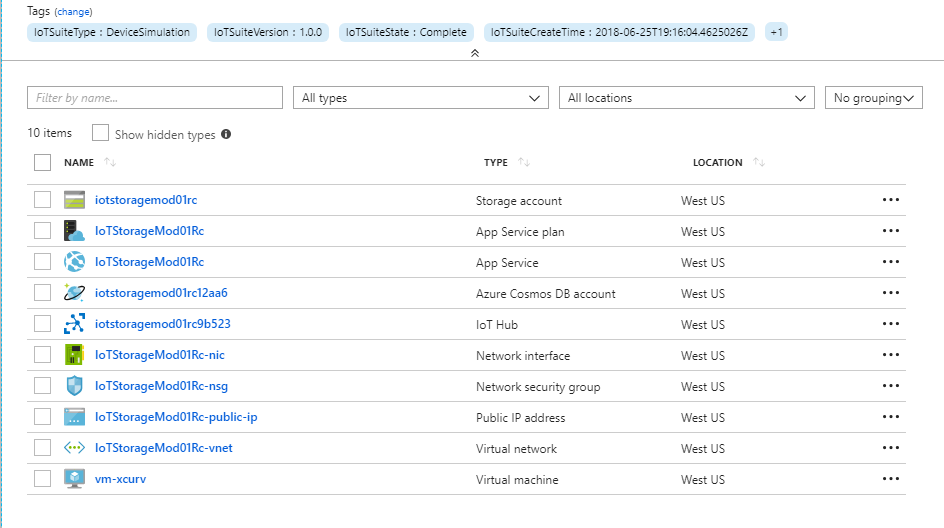
**Note**: If **Resource groups** is not listed under Favorites, click **All services**, and then, under GENERAL, click **Resource groups**

To gain an understanding of how the Solution Accelerator creates its telemetry messages, we will take a look at the resource group that it created.

1. On the Resources groups blade, click the resource group with name of your solution accelerator.

If you used the suggested naming convention it will start with **DEV312xMod04** and will include your initials and date of birth.

1. Notice that the Solution Accelerator provisions resources for network, application and storage. It also includes an IoT Hub, if you selected the default.



Your next task will be to use a Stream Analytics job to view a sample portion of our data.

**Run the Device Simulation Solution Accelerator**

Microsoft provides a flexible solution for creating simulated IoT data. It is part of the Microsoft Azure IoT Solution Accelerators located at https://www.azureiotsolutions.com/Accelerators. The solution accelerators are open source applications that cover many common IoT scenarios. Examining them all is outside the scope of this course, but they are worth studying on your own. The accelerators are production-grade, built with the idea that companies can use them as a starting point and adapt them into their own production environment.

**Note**: This course is focused on the analyzing data, rather than the mechanisms for producing that data, so we will start by setting up a system for generating simulated data. This will give you the opportunity to generate realistic quantities of IoT telemetry data, without having to purchase and configure a fleet of IoT hardware.

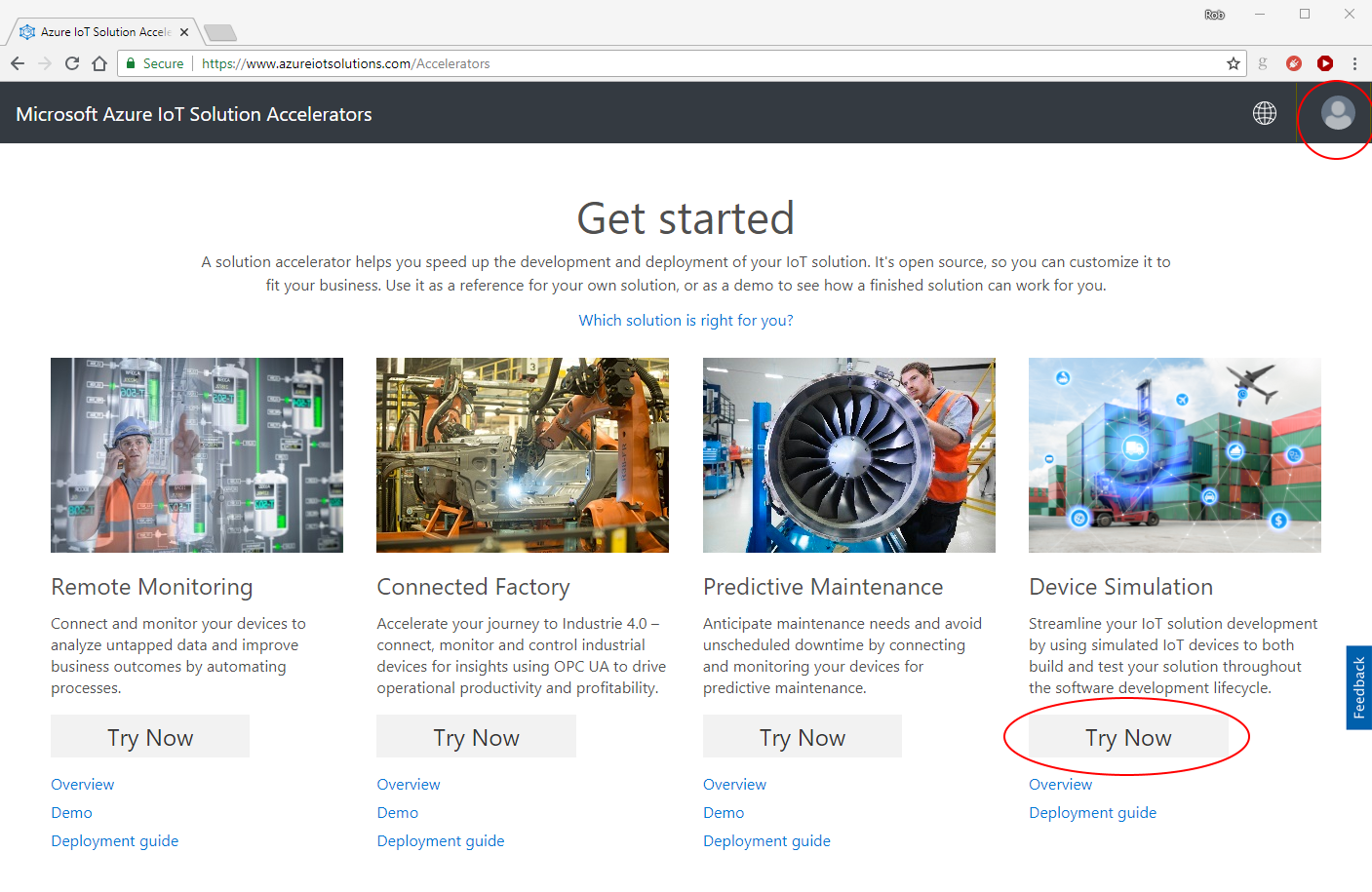
In this task, you will set up a solution accelerator and use it to generate simulated data for the wind farm scenario.

**Note**: You can come back to this task in subsequent modules if you need to generate more streaming data.

1. Navigate to the Solution Accelerator website at [Accelerators](https://www.azureiotsolutions.com/Accelerators).
2. In the upper-right corner of the web page, click the Account icon.
3. On the list, click **Sign in**

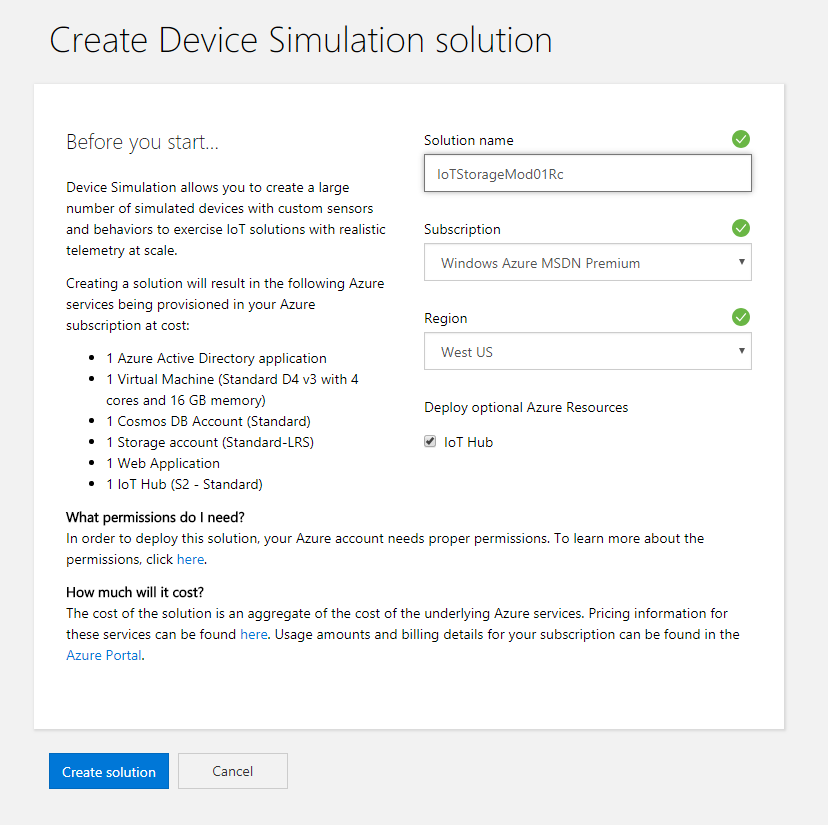
**Note**: You will need to use the same account for the accelerators that you use for your Azure subscription.

Once you are signed-in you should see your name appear next to Account.



1. On the right side of the page under Device Simulation, click **Try Now**

A page will be displayed that you can use to configure a solution accelerator.



1. Under **Solution name**, enter a unique name for your solution.

You need to provide a name that's globally unique. You will also want the name to be descriptive of your project and easy to remember. For this course we suggest using something like “DEV326xMod01” followed by your initials and date of birth.

For example: **DEV326xMod01CAH050961**

1. Under **Subscription**, use the list to select the Azure subscription that you are using for this course.
2. Under **Region**, select the Azure region nearest you.
3. To provision an IoT Hub for this task, click **IoT Hub**

We recommend selecting the **Iot Hub** checkbox, so that you provision an IoT hub just for this task. You are welcome to leave it unchecked if you want to use an IoT hub from previous work you have done.

**Note** The accelerator will create a large number of resources in your Azure subscription, and these resources do have an associated cost.

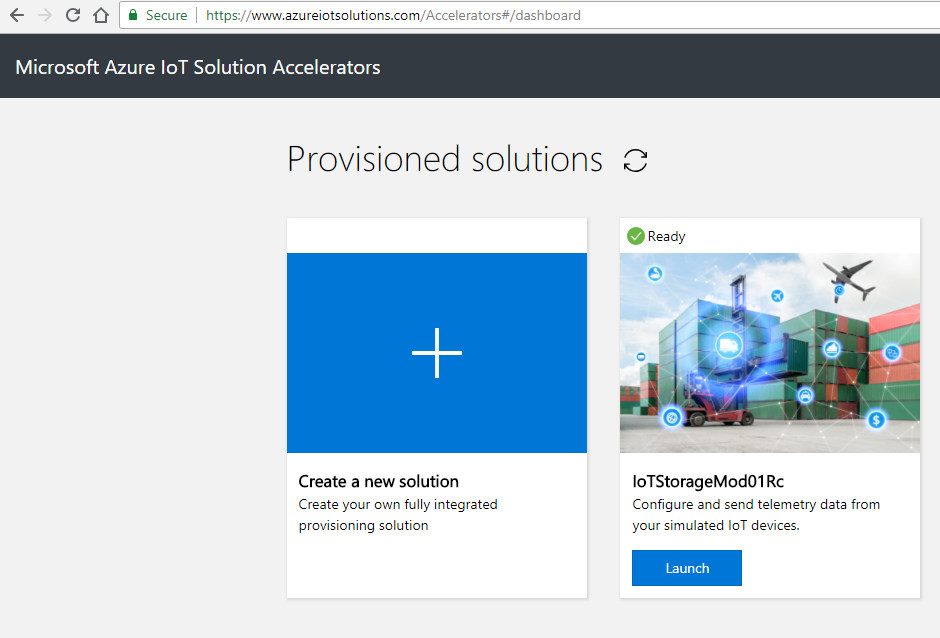
1. Click **Create solution** The accelerator dashboard will show a spinning graphic that says **Provisioning**

**Note** You can click **Details** while this is displayed to see which individual resources are being provisioned. When the accelerator is finished provisioning, you will be presented with a dashboard that allows you to navigate to your solution accelerator.

**Note**: It can take 10 minutes or more to provision the resources. You can use the time to study how other Microsoft Azure IoT Solution Accelerators are constructed, or you can just take a quick nap.

1. On your solution accelerators dashboard page, click **Launch**

**Note**: If your dashboard is not displayed automatically, navigate to <https://www.azureiotsolutions.com/Accelerators#/dashboard>, and then click **Launch**



1. To allow the solution accelerator to read your account profile, click **Accept**

You must grant permission to access your account (unless you have previously provisioned an accelerator).

You will be redirected to a Simulation setup page where you will configure your simulator.

1. On the Simulation setup page, Under Device model, in the **Select model** list, click **Custom**

Notice that you have the option to select one of several pre-configured device scenarios. Since we will be simulating wind turbines for Contoso Wind Energy, the Custom option allows us to simulate the sensor types that match our scenario.

You can now add sensors to your custom IoT device. For each sensor, you supply a name, a behavior (how the telemetry values change over time), min/max telemetry values, and a unit of incremental change for the value.

In the real world, a single wind turbine has hundreds of sensors and if you needed simulated data that accurately mimicked your actual configuration you might add a large percentage of them to your simulator. Luckily we will not be doing that for this course, and since wind turbine sensors fall into some major categories, we'll consolidate the sensors used in this simulation.

1. On the **Add Device Model** page, under **Device Model Name** enter **Wind Turbine**.
2. Under **Version**, enter one
3. Under **Telemetry Data**, use the fields provided to add the sensor values listed in the table below.

**Note**: When you finish entering a row of values for a sensor, to add another row, click **+ Add data point**

| **Sensor** | **Behavior** | **Min** | **Max** | **Unit** |
| --- | --- | --- | --- | --- |
| Bearings Temperature | Random | -30 | 65 | Celsius |
| Windings Temperature | Increment | -30 | 65 | Celsius |
| Tower sway | Random | 10 | 50 | g |
| Position sensor | Random | 2 | 30 | degrees |
| Gearbox fluid levels | Decrement | 1 | 1000 | ml |
| Blade strain gauge | Random | 25 | 5,000 | psi |
| Main shaft strain gauge | Random | 25 | 10,000 | psi |
| Shroud accelerometer | Random | 0 | 1,800 | rpm |
| Power Generation | Increment | 0 | 750 | kW |

When you are finished, your new **Wind Turbine** will show up in the list of device models.

1. Click **Simulations** on the left side of the window.
2. Click **Create Simulation**
3. In the **Simulation Setup** window, under **Name**, enter **Turbine Simulation**
4. Under **Simulation duration**, click **End in:**

We want the simulation to end after running for 10 minutes. That will produce about (50 devices x 9 sensors x 10 minutes x 6 readings per minute) = 27,000 data points to work with. You do not want to run this simulation indefinitely.

**Note**: Do not run this simulation with the Simulation duration set to *Run indefinitely*.

1. Under **MM**, type **10**
2. Under **Device Model**, click **+ Add a Device Type**
3. Under **Name**, choose your **Wind Turbine** from the list.
4. Under **Amount**, to provision 50 devices, type **50**

As noted, you can provision as many as 20,000 devices. For this module, we’ll start with a manageable number.

1. Notice that **Telemetry frequency** is specified in hours, minutes, and seconds.

Time frequency is a critical part of telemetry data. You always face the trade-off between granularity and volume. The more frequently data is sampled, the more accurate it is over time, but the quantities of data can get prohibitively large. When deciding on time frequency, analyze the underlying data. For instance, the frequency of readings for tectonic plate movement would not need to be as frequent as readings of molecular entropy. In the case of this wind turbine simulation, having the sensors report every 10 seconds is reasonable.

1. Under **SS**, to set the telemetry frequency to 10 seconds, type **10**
2. Under **Target IoT Hub**, ensure that **Use the pre-provisioned IoT Hub** is selected.

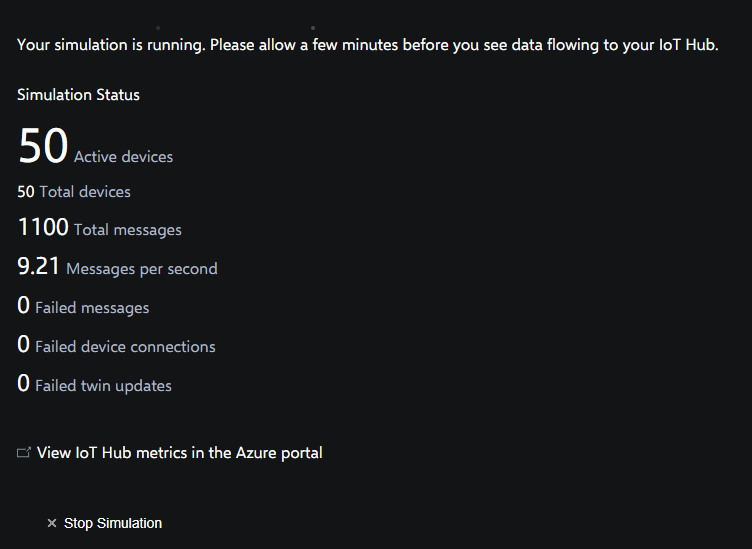
If you checked the **Iot Hub** checkbox when configuring your solution accelerator, the **Use the pre-provisioned IoT Hub** option should be selected by default. If you didn't check the **IoT Hub** checkbox, or if you want to use an existing IoT Hub, you can provide the connection string to the IoT Hub that you created previously.

1. To start the simulation, click **Start Simulation**
2. Create a record of the time when you started the simulation.

You will need to know the start-time when you are sampling or extracting the data later in the course.

1. Scroll down to view the progress of your simulation.

On your Simulation Setup page, you will start to see the simulated telemetry data flow from the device simulation to the IoT Hub in your Azure account.



1. Wait 3-4 minutes, and then click **View IoT Hub metrics in the Azure portal**

The link will open a new window that displays a view of your Azure portal. Specifically, the Metrics blade of the IoT Hub service.

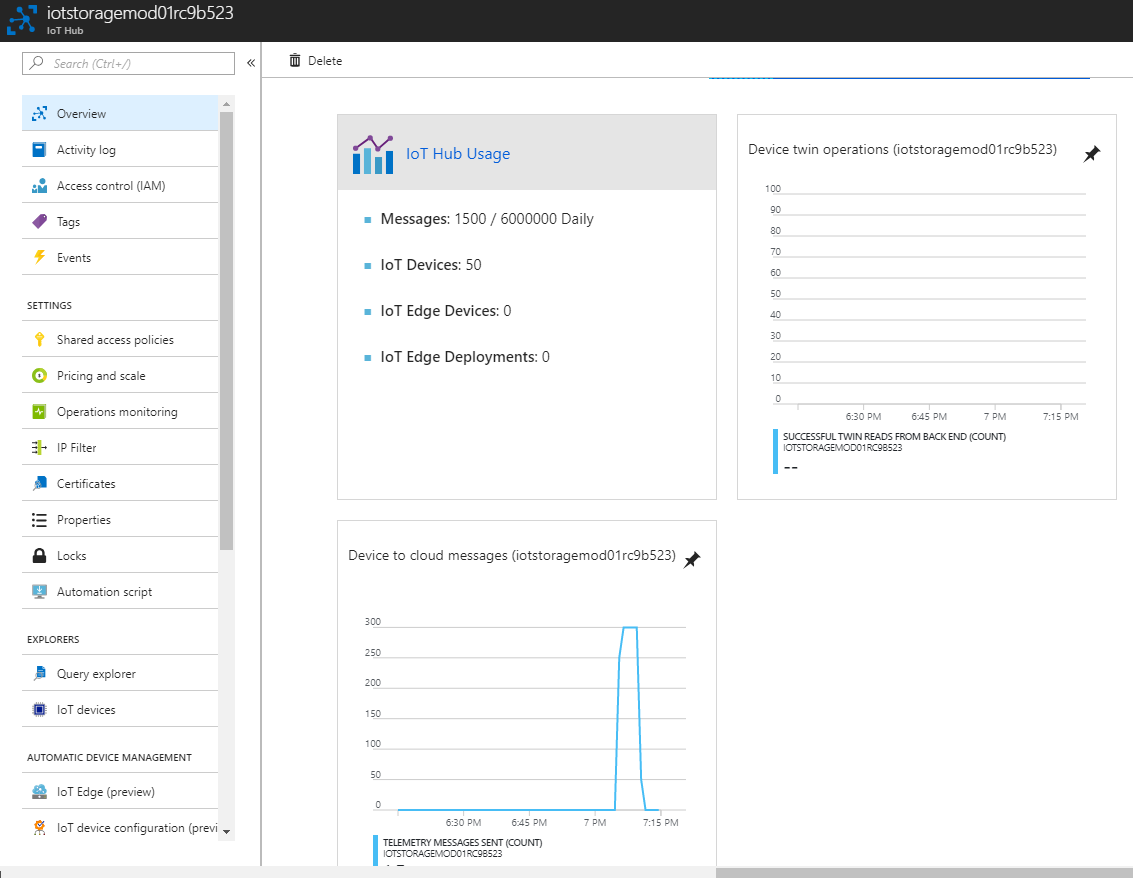
1. On the Metrics blade of your IoT Hub service, take a minute to review which metrics have been selected for display.

The Metrics blade gives you another view of the telemetry messages as they are ingested by your IoT Hub. Understanding how the IoT Hub fits into the IoT architecture is critical. The IoT hub provisions and manages devices. It also receives and stores IoT data. The hub then acts as a data source for the other components that store and analyze data.

1. Wait until the device simulation job has completed before continuing.

After about 10 minutes has passed, the number of “Connected Devices” will drop off, indicating that the simulation is finishing up.

1. On the menu to the left side of your IoT Hub blade, click **Overview**



1. On the left-side menu of your Azure portal, click **Resource groups**

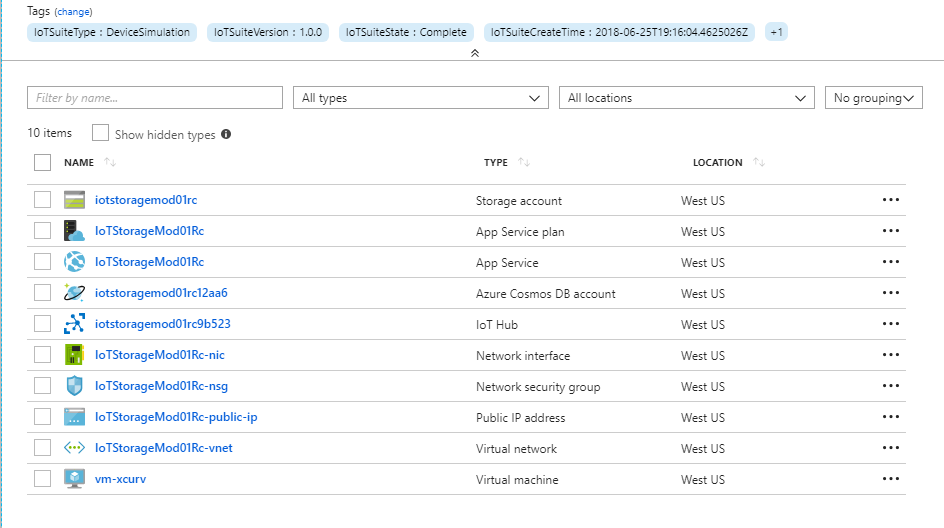
**Note**: If **Resource groups** is not listed under Favorites, click **All services**, and then, under GENERAL, click **Resource groups**

To gain an understanding of how the Solution Accelerator creates its telemetry messages, we will take a look at the resource group that it created.

1. On the Resources groups blade, click the resource group with name of your solution accelerator.

If you used the suggested naming convention it will start with **DEV312xMod04** and will include your initials and date of birth.

1. Notice that the Solution Accelerator provisions resources for network, application and storage. It also includes an IoT Hub, if you selected the default.



Your next task will be to use a Stream Analytics job to view a sample portion of our data.

## Create a Stream Analytics Job and View Your Data

# Create a Stream Analytics Job and View Your Data

Azure Stream Analytics is an event-processing engine that allows you to examine high volumes of data streaming from devices. Incoming data can be from devices, sensors, web sites, social media feeds, applications, and more. It also supports extracting information from data streams, identifying patterns, and relationships. You can then use these patterns to trigger other actions downstream, like alerts, feed information to a reporting tool, or store it for later use.

Following are some examples where Azure Stream Analytics can be used:

* Internet of Things(IoT) Sensor fusion and real-time analytics on device telemetry
* Web logs/clickstream analytics
* Geospatial analytics for fleet management and driverless vehicles
* Remote monitoring and predictive maintenance of hi-value assets
* Real-time analytics on Point of Sale data for inventory control and anomaly detection

The first example above is most pertinent to this course. Contoso Wind Power has wind farms that produce telemetry data.

In this task, you will create a Stream Analytics job and use it to process the streaming data that your IoT Device Simulation creates.

1. To open your Azure portal, navigate to the [portal.azure.com](http://portal.azure.com).

Ensure that you are signed in to the directory and account that you are using for this course.

1. In the top-left corner of your Azure portal, click **+ Create a resource**
2. In the Search box, type **Stream Analytics Job** and then press Enter.
3. In the list of filtered results, click **Stream Analytics job**
4. After reading through the text description, click **Create**
5. On the New Stream Analytics Job blade, under **Job name**, enter a name that follows the convention you established with your Solution Accelerator.

For example, if you specified **DEV312xMod04CAH070961** for your solution accelerator, enter a Job name of **DEV312xMod04CAH070961\_stream**

1. Under Subscription, select the subscription that you are using for this course.
2. Under Resource group, click **Use existing**
3. Select Resource group that was created by the solution accelerator.
4. Under Location, select a region location that is near you.
5. Under Streaming units, leave the default of 1.

Streaming units indicates how much computation power will be available to process your streaming job. You can leave it at the default of 1.

1. Click **Create**

After a short processing period, you will get a notification that your stream analytics job has been deployed successfully.

1. On the Notifications pane, to navigate to the stream analytics overview page, click **Go to resource**

You can also use your Resource group to open your Stream Analytics job.

1. On the Stream Analytics job blade, to begin defining an input, click **Inputs**

You can either click directly in the Inputs section of the Stream Analytics blade, or you can click **Inputs** from the navigation menu, which is located under the JOB TOPOLOGY section. If you use the menu, a new page will be loaded on the current blade. If you click the Inputs section of the Overview blade, a new Inputs blade will open. Either way, you will be able to begin specifying your input data source.

1. On the Inputs blade, click **+ Add stream input**, and then click **IoT Hub**

An IoT Hub input pane will open which you will use to configure your input data source.

1. On the IoT Hub input pane, under Input alias, enter **IoTHubTurbineData**

The Input alias is a friendly name used to reference the input. This is helpful because input sources can end up with rather templated or coded names, like HUB003WF013USACA. This is your chance to provide something more descriptive or recognizable.

1. Just below the Input alias field, ensure that **Select IoT Hub from your subscription** is selected.
2. Under Subscription, ensure that the correct Azure subscription is selected.

If you use multiple subscriptions this field may be populated by the most recently accessed subscription. In this case, the pre-populated value is likely to be the subscription that you want.

1. Under IoT Hub, ensure that the correct IoT Hub is selected.

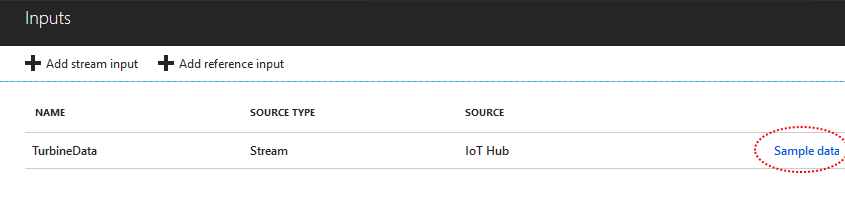
The IoT Hub that you specified in the accelerator setup may already be selected, but if not, select it from the **IoT Hub** list.

1. Leave the other fields at their pre-selected values:
   * Endpoint: Messaging
   * Share access policy name: iothubowner
   * Consumer group: $Default
   * Event serialization format: JSON
   * Encoding: UTF-8
   * Event compression type: None
2. At the bottom of the pane, click **Save**

It will take a moment for the Stream Analytics job to test the input. You should see a “Successful connection test” notification when the test is finished.

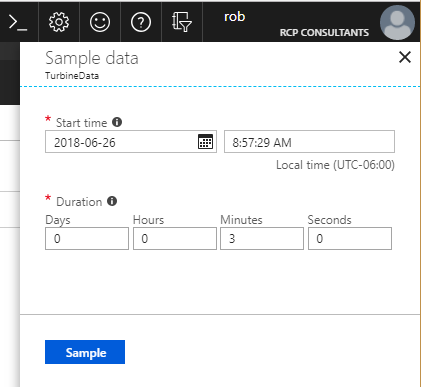
The Inputs blade should now be updated to show your new Input. **IoTHubTurbineData** should now be listed under NAME. To the right side of the screen, notice the link to **Sample data**

1. On the right side of the blade, click **Sample Data**



A Sample Data pane will open asking for the start time and duration.

1. On the Sample Data pane, use the Start time and Duration fields to specify a 3 minute block of time during the period when simulated data was being generated by the Data Simulation solution accelerator.



You should have recorded the start time earlier in this lab. You just need a sample to see how the data is formatted, so 3 minutes from the middle of the simulation period should be plenty.

1. At the bottom of the Sample Data pane, click **Sample**

It may take a few minutes to generate, so keep an eye on your notifications (the bell icon in the upper right of your Azure portal).

1. Open the Notifications pane.

After a brief period, you should see a notification appear with a message similar to **Sample input ‘IoTHubTurbineData’ succeeded**

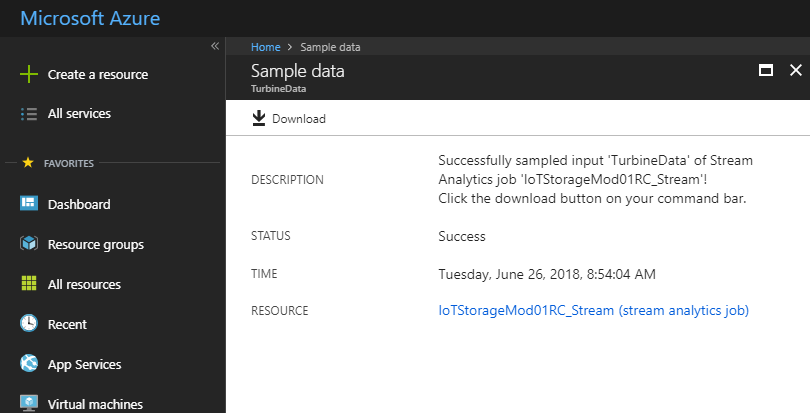
If you receive a notification telling you that “No events were found” during the specified period, check your time range. You can open your IoT Hub overview blade and check to see when your simulation data arrived at the hub (expand the time range if you ran the simulation more than 6 hours earlier).

**Note**: You will need a successful sample input before you can continue.

1. On the Notifications pane, click **Sample input ‘IoTHubTurbineData’ succeeded**

**Note**: There is a time limit on how long you can wait to download the data, so this is not the time to break for the day.

1. At the top of the Sample data blade, click **Download**



1. When prompted, save the sample data a local folder location.

The sample data is formatted as JSON file and has a record containing values for all the sensors that you set up in the Solution Accelerator. We will be using Visual Studio Code to view the data.

1. Open Visual Studio Code, and then open the sample data file that you just saved.
2. On the **File** menu, click **Save As**

The file is saved with a .txt file extension, or with no extension at all. We will change that file extension to JSON, which will help VSCode to format the data for us.

1. On the Save As dialog, open the **Save as type** list, click **JSON**, and then click **Save**

With the file extension changed to JSON, VSCode understands how to format the file for us.

1. To format the data file, use the keyboard shortcut appropriate to your platform:
   * On Windows: Shift + Alt + F
   * On Mac: Shift + Option + F
   * On Ubuntu: Ctrl + Shift + I
2. Your sample data file should now be formatted in a manner similar to the following:
3. [
4. {
5. "bearings temperature": -27.3733359469908,
6. "bearings temperature\_unit": "celcius",
7. "windings temperature": -29,
8. "windings temperature\_unit": "celcius",
9. "tower sway": 39.6212032761524,
10. "tower sway\_unit": "g",
11. "position sensor": 6.40746135656138,
12. "position sensor\_unit": "degrees",
13. "gearbox fluid levels": 999,
14. "gearbox fluid levels\_unit": "ml",
15. "blade strain gauge": 2164.26212725428,
16. "blade strain gauge\_unit": "psi",
17. "main shaft strain gauge": 2103.22657874936,
18. "main shaft strain gauge\_unit": "psi",
19. "shroud accelerometer": 1175.85299200185,
20. "shroud accelerometer\_unit": "rpm",
21. "power generation": 1,
22. "power generation\_unit": "kw",
23. "EventProcessedUtcTime": "2018-11-11T16:28:08.5710712Z",
24. "PartitionId": 2,
25. "EventEnqueuedUtcTime": "2018-11-10T17:54:31.4610000Z",
26. "IoTHub": {
27. "MessageId": null,
28. "CorrelationId": null,
29. "ConnectionDeviceId": "122f2c65-f8b9-4bb8-9314-fbc9fd1cf10e.9471b892-57c8-4260-9172-45533a79cc6f.9",
30. "ConnectionDeviceGenerationId": "636774692418824263",
31. "EnqueuedTime": "2018-11-10T17:54:31.4400000Z",
32. "StreamId": null
33. }
34. },
35. {
36. "bearings temperature": 22.4689256900311,
37. "bearings temperature\_unit": "celcius",
38. "windings temperature": -29,
39. "windings temperature\_unit": "celcius",
40. "tower sway": 17.8955025448909,
41. "tower sway\_unit": "g",
42. "position sensor": 5.58851467985125,
43. "position sensor\_unit": "degrees",
44. "gearbox fluid levels": 999,
45. "gearbox fluid levels\_unit": "ml",
46. "blade strain gauge": 2480.73703800595,
47. "blade strain gauge\_unit": "psi",
48. "main shaft strain gauge": 9711.08093376322,
49. "main shaft strain gauge\_unit": "psi",
50. "shroud accelerometer": 36.2303238530785,
51. "shroud accelerometer\_unit": "rpm",
52. "power generation": 1,
53. "power generation\_unit": "kw",
54. "EventProcessedUtcTime": "2018-11-11T16:28:06.4348585Z",
55. "PartitionId": 1,
56. "EventEnqueuedUtcTime": "2018-11-10T17:54:31.4960000Z",
57. "IoTHub": {
58. "MessageId": null,
59. "CorrelationId": null,
60. "ConnectionDeviceId": "122f2c65-f8b9-4bb8-9314-fbc9fd1cf10e.9471b892-57c8-4260-9172-45533a79cc6f.21",
61. "ConnectionDeviceGenerationId": "636774692418824263",
62. "EnqueuedTime": "2018-11-10T17:54:31.4680000Z",
63. "StreamId": null
64. }
65. }
66. ]
67. Take a minute or two to review the sample data.

Notice the additional properties below that the IoT hub and stream analytics job added.

* + There are time properties: EventProcessedUtcTime and EventEnqueuedUtcTime. These are important for real-time and streaming data. There are numerous situations where a sensor can generate an event, but it may arrive at the IoT hub after other events that were generated later. One of the decisions you make when processing data in real time is whether to use enqueued or processed order.
  + The IoTHub includes other metadata, such as a device ID and message and generation ID’s.

### Lab Summary

You now have several components of an IoT architecture:

* Simulated wind turbine IoT devices
* An IoT Hub with IoT data
* An Azure Stream Analytics job with the IoT Hub as an input data source

You will use these components throughout module 1.

It is also important to realize that the simulated data you can produce with the Solution Accelerator is useful, even when you do have physical devices. The Solution Accelerator allows you to set up a testing infrastructure, so that you can try out new configurations quickly and easily.

## Set up a Cold Storage Repository with Azure Data Lake Storage

For architectures that produce significant amounts of data, a common pattern is to split the data into “warm” and “cold” data stores. Traditionally, data stored in cold storage is historical. Cold storage database technology is usually cost effective, but there is a trade-off of slower performance. One option for cold storage with Azure IoT is Azure Blob Storage, which is a simple, inexpensive file storage database, with practically limitless capacity.

However, Contoso Wind Power would like to get more out of its cold storage than a long-term, historical storage repository. You have been tasked with gaining insights about aggregate data, and your data science team would like to perform deep analysis on the data. Additionally, the data team would like you to import historical data that was generated back when Contoso Holdings was still the parent company. With this in mind, you have investigated using Microsoft’s other main offering for cold storage: Azure Data Lake Storage. Azure Data Lake can store nearly unlimited amounts of data (like Blob Storage), but it has other features that make it ideal for data analytics.

In this lab, you will provision an Azure Data Lake instance and stream data to it using the device simulation that you set up in the previous lab. This workflow simulates how you would use Azure Data Lake for cold storage of live telemetry data. Next, you will run analytics operations on your data with Azure Data Lake Analytics.

For more information on Azure Data Lake Storage, begin with the following [Introduction to Azure Data Lake Storage](https://docs.microsoft.com/en-us/azure/storage/data-lake-storage/introduction).

In this task, you will create your Azure Data Lake Storage resource and use a Stream Analytics job to populate it with your simulated data.

1. To open your Azure portal, navigate to the [portal.azure.com](http://portal.azure.com).

Ensure that you are signed in to the directory and account that you are using for this course.

1. In the top-left corner of your Azure portal, click **+ Create a resource**
2. In the Search textbox, type **Data Lake Store** and then press Enter.
3. In the list of filtered results, click **Data Lake Store**

(Note – it may be named Data Lake Storage Gen1).

1. After reading through the text description, click **Create**
2. On the New Data Lake Storage blade, under Name, enter a name that mirrors the convention you established with your Solution Accelerator.

**Note**: The Name property is will be used for form a URL and must be globally unique. In addition, it is limited to lower case and numeric characters.

If you specified **DEV326xMod01CAH050961** for your solution accelerator, you could enter a Name of **dev326xmod01cah050961dls** for your Data Lake Store.

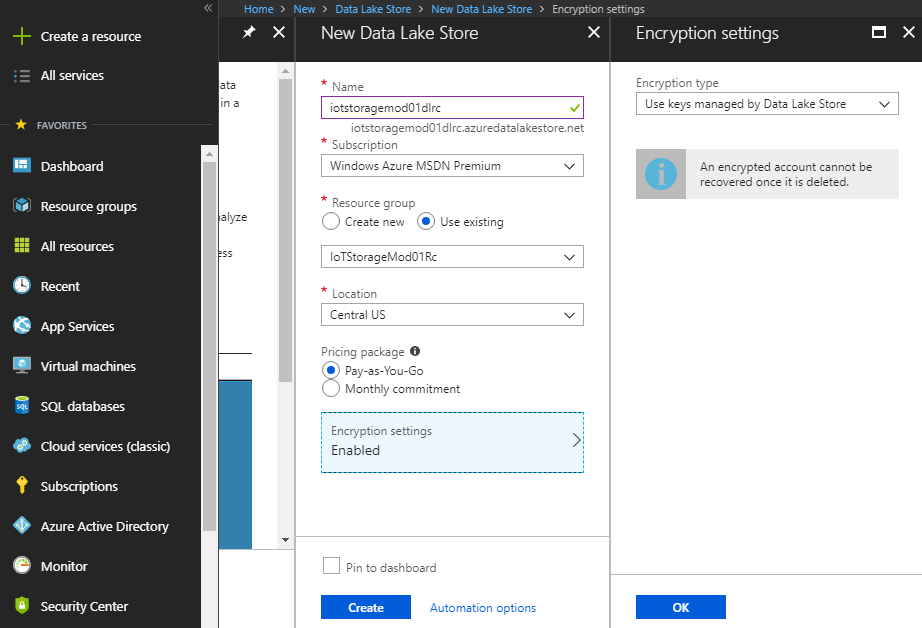
1. Under Subscription, select the subscription that you are using for this course.
2. Under Resource group, click **Use existing**, and then select Resource group that was created by the solution accelerator.
3. Under Location, select a region location that is near you.
4. Under Pricing package, ensure that **Pay-as-You-Go** is selected.

In a real-world cold storage architecture, you would probably reserve some storage capacity with a monthly commitment, but that’s not necessary here.

**Note**: For **Encryption settings**, leave default of setting of **Enabled** (which should specify “Use keys managed by Data Lake Store” under Encryption type).

1. At the bottom of the blade, click **Create**

That's all it takes to provision a data lake. It may take a few moments to complete, though, so keep an eye on your notifications.

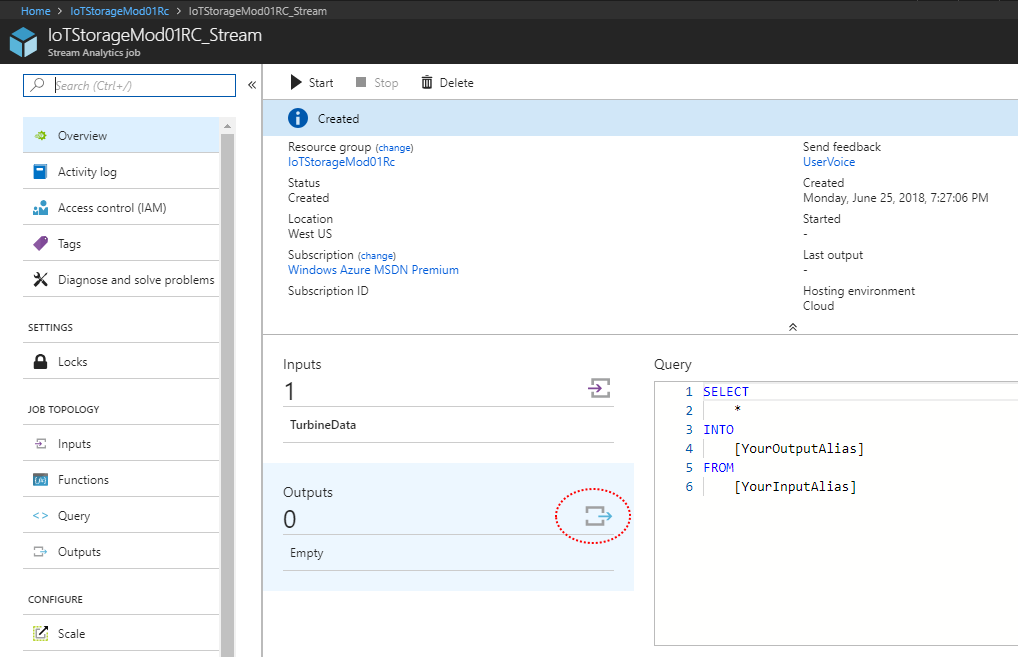


1. Once the deployment has complete, navigate to the Stream Analytics job that you created in the previous lab.

There are several ways to open your Stream Analytics job. One of the easiest is to use the search bar at the top of your Azure portal. Type in **stream** and a link to your stream analytics job should appear in the filtered list of resources.

1. On the Overview blade of your Stream Analytics job, click **Outputs**

If you click Outputs on the left-side menu, the current blade will be refreshed to show Outputs. If you click the Outputs section on the Overview blade (as highlighted below), Outputs will open a new blade.



1. In the upper left corner of the Outputs blade, click **+ Add**.

You will be given several endpoint options for output.

1. In the list of options, click **Data Lake Store**
2. On the Data Lake Store output pane, under Output alias, enter **DataLakeStoreTurbineData**
3. Just below the Output alias field, ensure that **Select Data Lake Store from your subscription** is selected.
4. Under Subscription, select the subscription that you are using for this course.
5. Notice the text field for “Path prefix pattern”.

This field bears some explaining. The data that you export to Azure Data Lake will be stored in files. The file path that you enter here will determine where the files are located within your data lake. You can specify one or more instances of the {date} and {time} variables in your file path, to make the file path dynamically reflect the current UTC time.

* + Example 1: mod01/logs/{date}
  + Example 2: mod01/logs/{date}/{time}

1. Under Path prefix pattern, enter **mod01/logs/{date}**
2. Notice that you have control of the date format.

In the first example above, if you use a data format of YYYY/MM/DD, files created on August 15, 2018 would reside in a folder path of mod01/logs/2018/08/15.

If we had chosen to add {time} to the path, an HH time format would be included in the path. For the second example above, if we ran the job at 10:15 am UTC, our output file would reside in mod01/logs/2018/08/15/10.

1. After specifying the Path prefix pattern and setting up the date format, accept the default settings for the remaining fields.

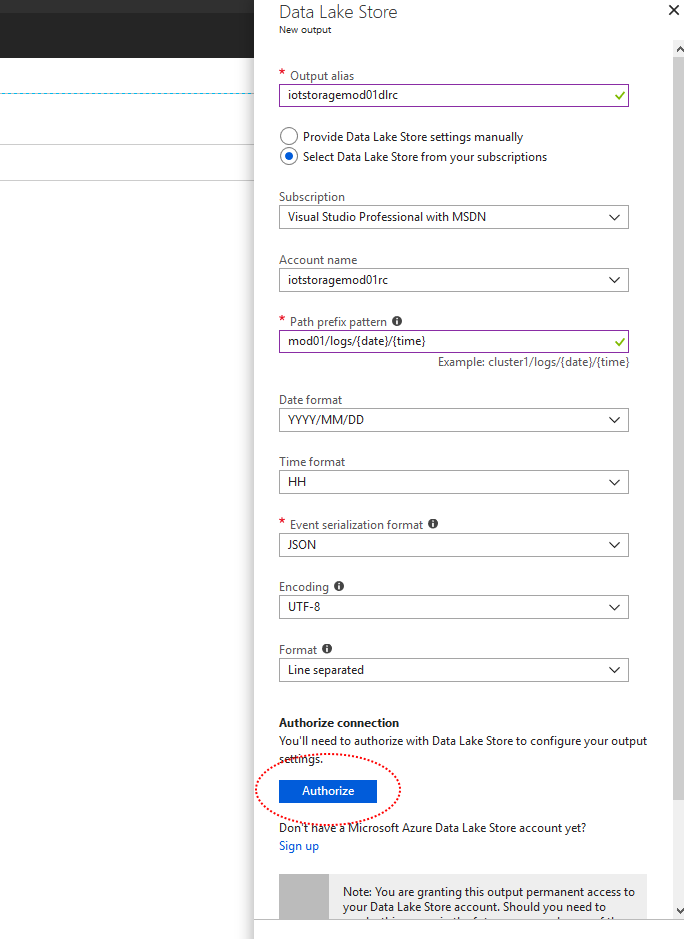
Event serialization format: JSON Encoding: UTF-8 Format: Line seperated

1. Under Authorize connection, notice that you need to authorize your Stream Analytics job to write to your data lake.

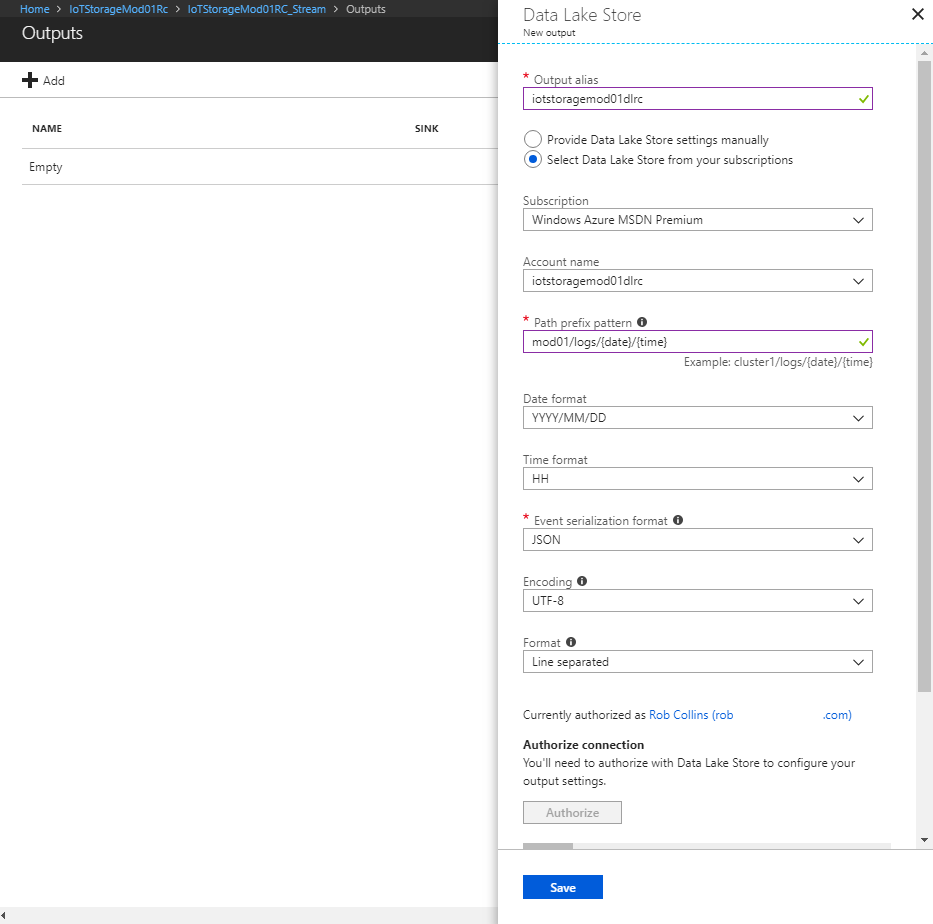
**Note**: If you have trouble with the Authorize steps below, close your Azure portal, log back in, and then repeat the steps to create the Output for your Stream Analytics job.

1. Click **Authorize**

The Data Lake Store output pane should refresh and display a message stating: Currently authorized as <Your Azure Identity>



1. At the bottom of the pane, click **Save**



1. Navigate back to the Overview blade for your Stream Analytics job.

Now that we have an input (the simulated data we generated with the solution accelerator) and an output (the Azure Data Lake Storage), we are going to take our first look at writing a query to transfer the data.

1. On the Overview blade, to the right of the Query section, click **Edit Query**

You will be taken to the Query Editor screen.

One of the biggest strengths of Azure Stream Analytics (ASA) is its use of the SQL language for data querying and transformation. SQL is a well-understood, popular language. If you have a familiarity with SQL, you are well on your way to working effectively with ASA. ASA’s version of SQL is nearly identical to the SQL you would use with SQL Server, with the addition of some extensions for time series data. We will take an increasingly closer look at ASA’s implementation of SQL as the course progresses, but for now, we will just use a basic query to get the entire dataset from the IoT hub into Azure Data Lake Storage.

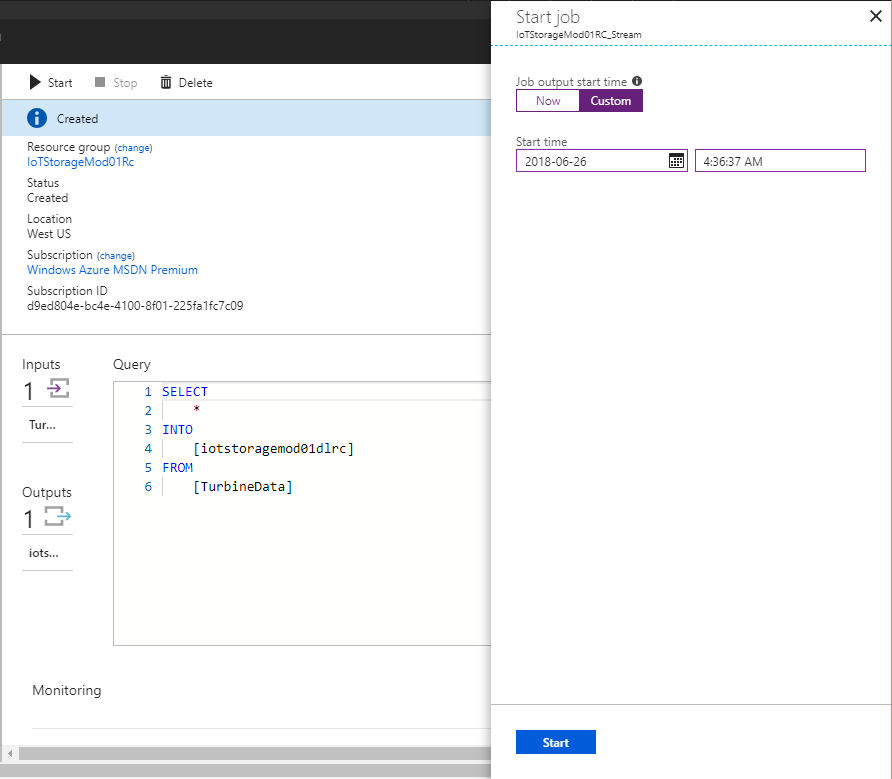
1. On the Query editor pane, under **FROM**, specify **IoTHubTurbineData**, and then, under **INTO**, specify **DataLakeStoreTurbineData**

This will give you the most basic query you can use to transfer data. As with other implementations of SQL, SELECT \* simply means select everything.

1. In the upper left corner, click the **Save**
2. To save the changes, click **Yes**, and then close the Query editor blade.
3. On the Overview blade of your Stream Analytics job, click **Start**

A Start job configuration pane will open that prompts you for the start time for the job. If you select “Now” for the "Job output start time", you will only get streaming events going forward. We want to capture the events that were created from the device simulator that we ran earlier.

1. On the Start job pane, click **Custom**, and then set a date/time just prior to the time when the device simulation started.
2. At the bottom of the Start job pane, click **Start**

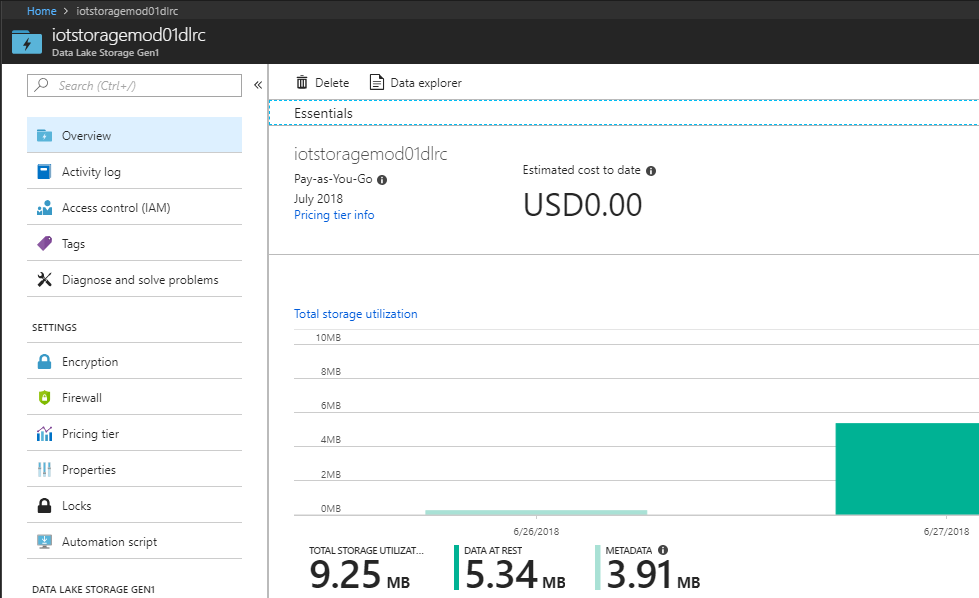


It will take a few minutes for the Stream Analytics job to start and to ingest all the simulated data. Keep an eye on the graphs – when the resource graph levels out, the data ingestion will be finished.

1. On the Stream Analytics job blade, click **Stop**

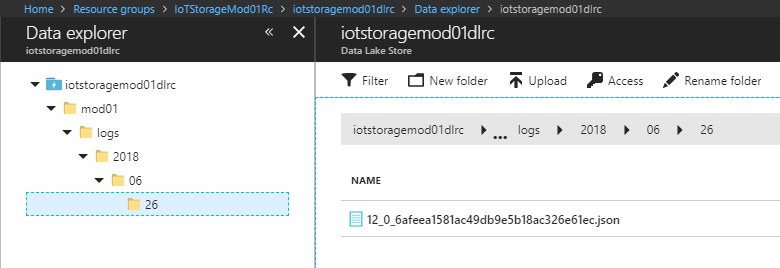
You don't want the ASA job to run indefinitely.

1. Close the Stream Analytics job blade, and then open the Data Lake Storage blade.
2. At the top of the blade, click **Data explorer**



1. Expand the folder structure that you set for your ASA job.

In the deepest folder, you will find a JSON file containing your data.

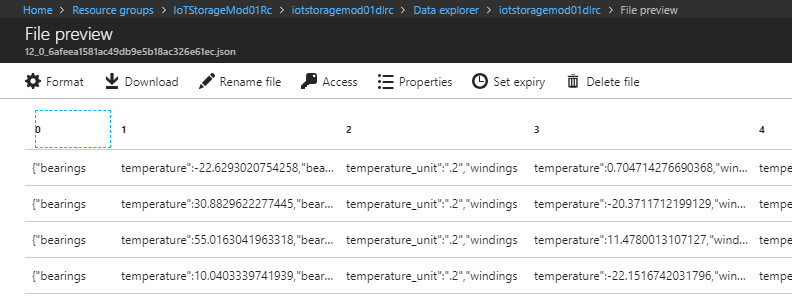


1. To preview your data, click the JSON file.

A new File preview blade will open.

1. Take a minute to review the data.

Notice that the file contains the same data as the input data from the IoT hub. If you download the file, you should find that all the data is there. This is the result of writing a SELECT \* query – all the data is transferred as-is.



### Summary

That’s it for Azure Data Lake Storage as cold storage for IoT data.

* You went through the steps to set up a Data Lake Store – a massively scalable storage offering.
* You set the data lake as the output endpoint for an Azure Stream Analytics job
* You wrote an ASA query to copy the data from your IoT Hub to the data lake

You could stop here for your cold path stream analytics and cold storage architecture. But the real strength of using an Azure Data Lake is all the other transformation and analytics features that it provides. We will take a look at those features in the next task\

**Azure Data Lake Analytics - Usage and purpose**

In this lesson, you will learn about

* Using Azure Data Lake Storage for IoT cold storage
* Other options for cold storage
* Using Azure Data Lake Analytics for analyzing cold path data

Data Lake Storage is effective at ingesting data from disparate sources and saving it in its original format. That enables an analysis workflow where you save data with full fidelity, then extract and transform it with big data analysis tools. Instead of the traditional Extract-Transform-Load workflow, you can use a Load-Transform-Extract workflow. All the source data will be available for analysis – it won't be truncated with the initial transform step.

## Batch Analyze wind farm data with Azure Data Lake Analytics

In the previous task, you streamed data into Azure Data Lake Storage for cold storage. In this task, your objective is to leverage the capabilities of Azure Data Lake Analytics against historical wind farm data. Your team wants to discover insights and patterns in the data that it can use to help Contoso Wind Power operate more efficiently.

The architecture for Azure Data Lake Analytics service was built from the ground up for cloud scale and performance. It takes away the complexities normally associated with big data processing. It can transform, combine, aggregate and otherwise process data on a massive scale.

Azure Data Lake Storage works well with numerous other analytics tools on the market, but Azure Data Lake Analytics is purpose-built to tightly integrate with it.

1. On your Azure portal, click **+ Create a resource**.
2. On the New blade, in the Search box, type **data lake** and then press Enter.
3. In the list of filtered results, click **Data Lake Analytics**, and then click **Create**.
4. On the New Data Lake Analytics account blade, under Name, enter a unique name.

For example: **dev326xmod1cah050961dlaa**.

1. Use the existing resource group that you have been using for previous tasks.
2. Choose a region near you, and then, under Data Lake Store, click **Configure required settings**
3. On the Select Data Lake Store blade, select the Data Lake Store that you created previously.
4. Under Pricing package, ensure that **Pay-as-You-Go** is selected, and then click **Create**

The account will take a few moments to provision.

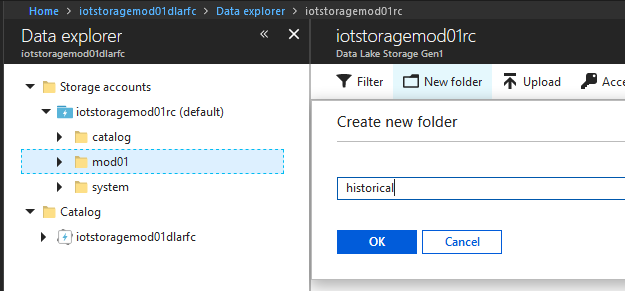
1. Once the provisioning process has completed, open your new Data Lake Analytics resource.

**Note**: You may need to refresh your dashboard before you see it listed in your resources.

1. On the Data Lake Analytics Overview blade, take a minute to review the options listed along the top menu.

You can create new jobs, you can work with sample scripts and you can navigate to the Data explorer in your storage account, you can configure security, etc.

1. At the top of the blade, click **Data explorer**
2. On the Data explorer blade, under Storage accounts, click **mod01**
3. On the Data Lake Storage blade, on the top menu, click **New folder**
4. In the folder name box, type **historical** and then click **OK**



1. To open the historical folder, click **historical**

This is where you will upload a source file for historical data analysis.

1. Save the following CSV file to your local machine.

[TurbineData2017.csv](https://prod-edxapp.edx-cdn.org/assets/courseware/v1/96c136b4bab9dbbdcfa779c63220301c/asset-v1:Microsoft+DEV326x+1T2019+type@asset+block/TurbineData2017.csv)

**Note** To open a Save As dialog, right-click the file name above and then click **Save As**. Save the file to a location that is easy to access.

This file is based on real-world wind farm data that ENGIE has made available for study [here](https://opendata-renewables.engie.com/pages/home/).

This file (and the original source) will give you a sense of what real-world turbine data looks like.

| **Wind\_turbine\_name** | **Date\_time** | **Ws\_avg** | **Ws\_std** | **Ot\_avg** | **Ds\_avg** | **Va\_avg** |
| --- | --- | --- | --- | --- | --- | --- |
| R80721 | 2017-02-07T21:40:00-07:00 | 1.9 | 0.30000001 | 5.3000002 | 88.18 | 25.18 |
| R80721 | 2017-02-07T23:40:00-07:00 | 0.2 | 0.52999997 | 4.8899999 | 23.01 | 44.310001 |
| R80721 | 2017-02-08T00:00:00-07:00 | 0.18000001 | 0.44 | 4.8000002 | 38.110001 | 39.939999 |
| R80721 | 2017-01-25T14:30:00-07:00 | 4.9099998 | 0.43000001 | -6.4200001 | 1209.53 | -1.9 |
| R80721 | 2017-01-25T16:40:00-07:00 | 4.1999998 | 0.31999999 | -7.3000002 | 1006.33 | 8.5299997 |
| R80721 | 2017-01-25T18:40:00-07:00 | 4.8499999 | 0.38999999 | -7.1599998 | 1205.52 | 11 |
| R80721 | 2017-01-25T20:20:00-07:00 | 5.1100001 | 0.28999999 | -6.98 | 1158.66 | -11.97 |
| R80721 | 2017-01-26T05:50:00-07:00 | 5.04 | 0.47999999 | 1.79 | 1181.7 | 3.8900001 |

The file also represents a real-world business scenario you are likely to encounter – the need to load and analyze historical data that may or may not be in the same format as your current data.

For the purposes of this course, we will say that this data represents historical telemetry readings that we need to make available to Contoso Wind Power’s business analysts.

1. On the Data Lake Storage blade, on the top menu, click **Upload**
2. On the Upload files blade, open the saved .csv file, and then click **Add selected files**

The STATUS will change to “Uploading” and the upload progress will be displayed. The STATUS will change to Completed when the upload has finished.

1. Close the **Upload files** blade.

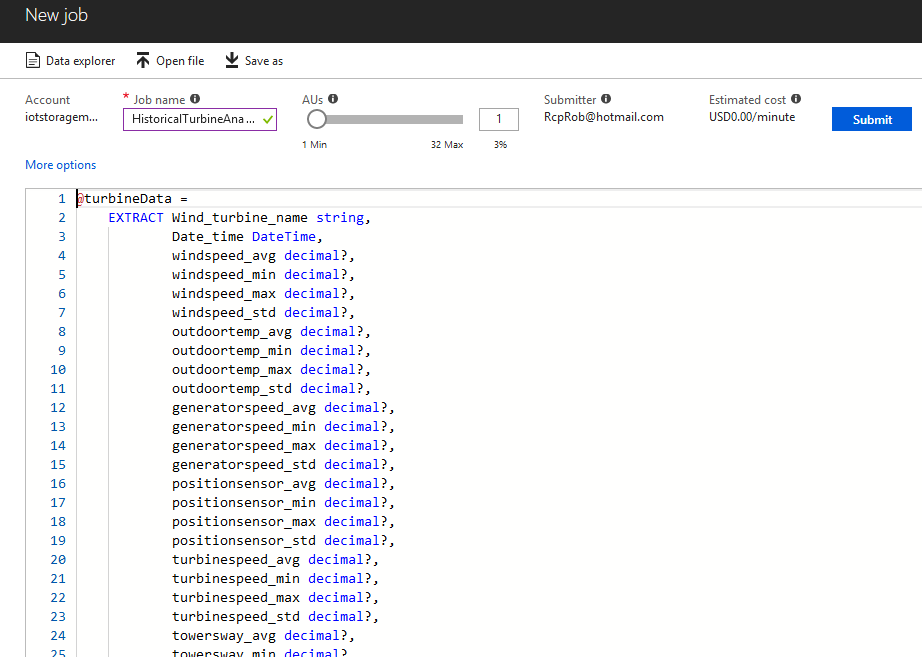
You should now see the TurbineData2017.csv file listed in your folder named **historical**.

1. With the historical folder open, click **New folder**, and then name the folder **analysis**

This is where the output files will be written.

1. Navigate back to the Azure Data Analytics Overview blade, and then click **+ New job**
2. On the New job blade, in the Job name textbox, enter **HistoricalTurbineAnalysis**

The large area under the More options label will be used for U-SQL code, which will extract the information from the TurbineData2017.csv file.



1. In the text box for U-SQL code, enter the following code:
2. @turbineData =
3. EXTRACT Wind\_turbine\_name string,
4. Date\_time DateTime,
5. windspeed\_avg decimal?,
6. windspeed\_min decimal?,
7. windspeed\_max decimal?,
8. windspeed\_std decimal?,
9. outdoortemp\_avg decimal?,
10. outdoortemp\_min decimal?,
11. outdoortemp\_max decimal?,
12. outdoortemp\_std decimal?,
13. generatorspeed\_avg decimal?,
14. generatorspeed\_min decimal?,
15. generatorspeed\_max decimal?,
16. generatorspeed\_std decimal?,
17. positionsensor\_avg decimal?,
18. positionsensor\_min decimal?,
19. positionsensor\_max decimal?,
20. positionsensor\_std decimal?,
21. turbinespeed\_avg decimal?,
22. turbinespeed\_min decimal?,
23. turbinespeed\_max decimal?,
24. turbinespeed\_std decimal?,
25. towersway\_avg decimal?,
26. towersway\_min decimal?,
27. towersway\_max decimal?,
28. towersway\_std decimal?,
29. powergeneration\_avg decimal?,
30. powergeneration\_min decimal?,
31. powergeneration\_max decimal?,
32. powergeneration\_std decimal?,
33. bearingtemp\_avg decimal?,
34. bearingtemp\_min decimal?,
35. bearingtemp\_max decimal?,
36. bearingtemp\_std decimal?,
37. windingstemp\_avg decimal?,
38. windingstemp\_min decimal?,
39. windingstemp\_max decimal?,
40. windingstemp\_std decimal?
41. FROM "/mod01/historical/TurbineData2017.csv"
42. USING Extractors.Csv(skipFirstNRows:1);

Notice that the column names in the U-SQL EXTRACT statement do not match the header values in the TurbineData2017.csv file. For instance, the Ws\_avg column from the file is queried as windspeed\_avg column in the U-SQL query.

This is an important concept to grasp. Azure Data Lake Analytics can coerce and transform unstructured data into structured data. Notice, for instance, that query uses a string data type for the Wind\_turbine\_name and a nullable decimal data type (decimal?) for the outdoortemp\_av property. Those data types are not defined in the source csv file – they are converted by the analytics query.

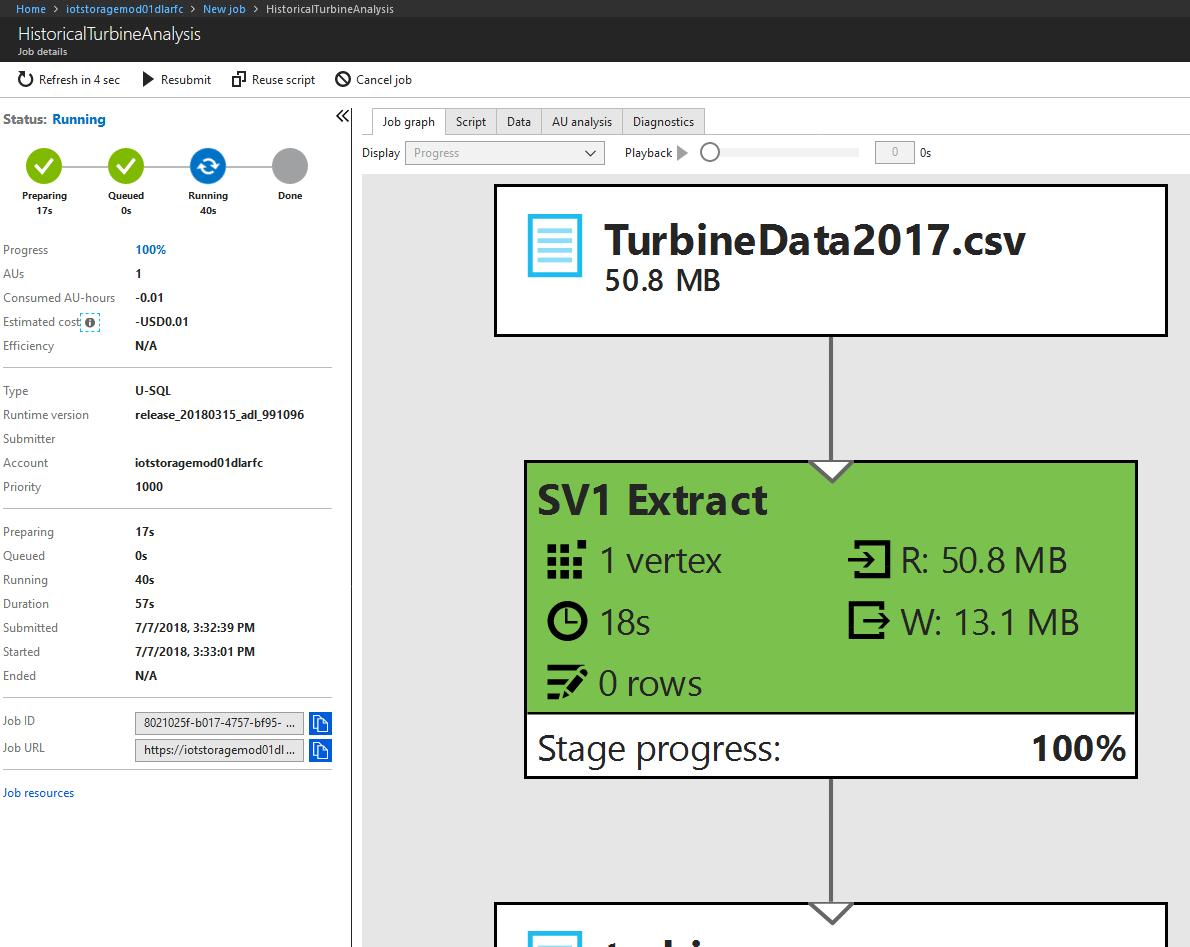
So now you have the entire dataset available as input. Next you need transform the data and serialize it somewhere.

1. In the text box for U-SQL code, to establish an output file, enter the following code:
2. DECLARE @outputFile0 string = "/mod01/historical/analysis/turbine\_power.csv";
3. Consider this first use case scenario:

Let’s say that a business analyst would like a report on some of the key historical readings for the wind turbines – windspeed, temperature and power generation that she can use for analysis. But she doesn’t want the entire historical file with all of its columns.

1. To create a query that simply narrows down the columns that are captured, enter the following code:
2. @rs0 =
3. SELECT Wind\_turbine\_name,
4. Date\_time,
5. windspeed\_avg,
6. outdoortemp\_avg,
7. powergeneration\_avg
8. FROM @turbineData;
9. Finally, to write an OUTPUT statement that will flush the queried data to the turbine\_power.csv file, enter the following code:
10. OUTPUT @rs0
11. TO @outputFile0
12. ORDER BY Wind\_turbine\_name,
13. Date\_time
14. USING Outputters.Csv(outputHeader : true);
15. Click **Submit**

This will kick off the process of preparing, queuing and running the job. Watch the steps proceed until done.



1. At the bottom of the Job Graph tab, notice the box indicating the turbine\_power.csv file.
2. To open a File preview blade displaying the data, click **turbine\_power.csv**.
3. Compare this data to the query that you wrote.

| **Wind\_turbine\_name** | **Date\_time** | **windspeed\_avg** | **outdoortemp\_avg** | **powergeneration\_avg** |
| --- | --- | --- | --- | --- |
| R80711 | 2016-12-31T16:00:00.0000000-07:00 | 4.5599999 | 3.8299999 | 76.489998 |
| R80711 | 2016-12-31T16:10:00.0000000-07:00 | 4.9899998 | 3.8199999 | 82.330002 |
| R80711 | 2016-12-31T16:20:00.0000000-07:00 | 4.8400002 | 3.8499999 | 66.480003 |
| R80711 | 2016-12-31T16:30:00.0000000-07:00 | 4.6999998 | 3.8299999 | 83.230003 |
| R80711 | 2016-12-31T16:40:00.0000000-07:00 | 4.8699999 | 3.78 | 85.25 |
| R80711 | 2016-12-31T16:50:00.0000000-07:00 | 4.98 | 3.5699999 | 85.010002 |

1. **Note**: You can also download the data to see the full file. As you may have expected, it has all the rows of the original file, just fewer columns, based on the query.
2. Close the File preview blade, and then close the HistoricalTurbineAnalysis Job details blade.

You should find yourself back at the blade containing the U-SQL code that you entered earlier.

1. Consider a second use case scenario:

A business analyst who would like to analyze how efficient individual wind turbines are at running their generators, compared with wind speed.

**Note** We will reuse the existing EXTRACT, FROM, and USING statements that we entered above.

1. At the bottom of the code, to create a query that aggregates the historical generator speed vs. wind speed, enter the following:
2. DECLARE @outputFile1 string = "/mod01/historical/analysis/turbine\_speed.csv";
3. @rs1 =
4. SELECT Wind\_turbine\_name,
5. AVG(generatorspeed\_avg) AS AvgGeneratorSpeed,
6. AVG(windspeed\_avg) AS AvgWindSpeed
7. FROM @turbineData
8. GROUP BY Wind\_turbine\_name;
9. OUTPUT @rs1
10. TO @outputFile1
11. USING Outputters.Csv(outputHeader:true);

Notice that the SELECT statement is identical to how you would write it if you were querying a comparable database table in SQL Server. U-SQL is its own language, but like many of Microsoft’s analytical tools, it benefits from its similarity with the ubiquitous SQL language. The GROUP BY clause, the AVG operator, and column aliasing – all the same.

1. Click **Submit**

The updated job will now output two files, rather than one. Let it run to completion.

1. On the Job Graph tab, click **turbine\_speed.csv**

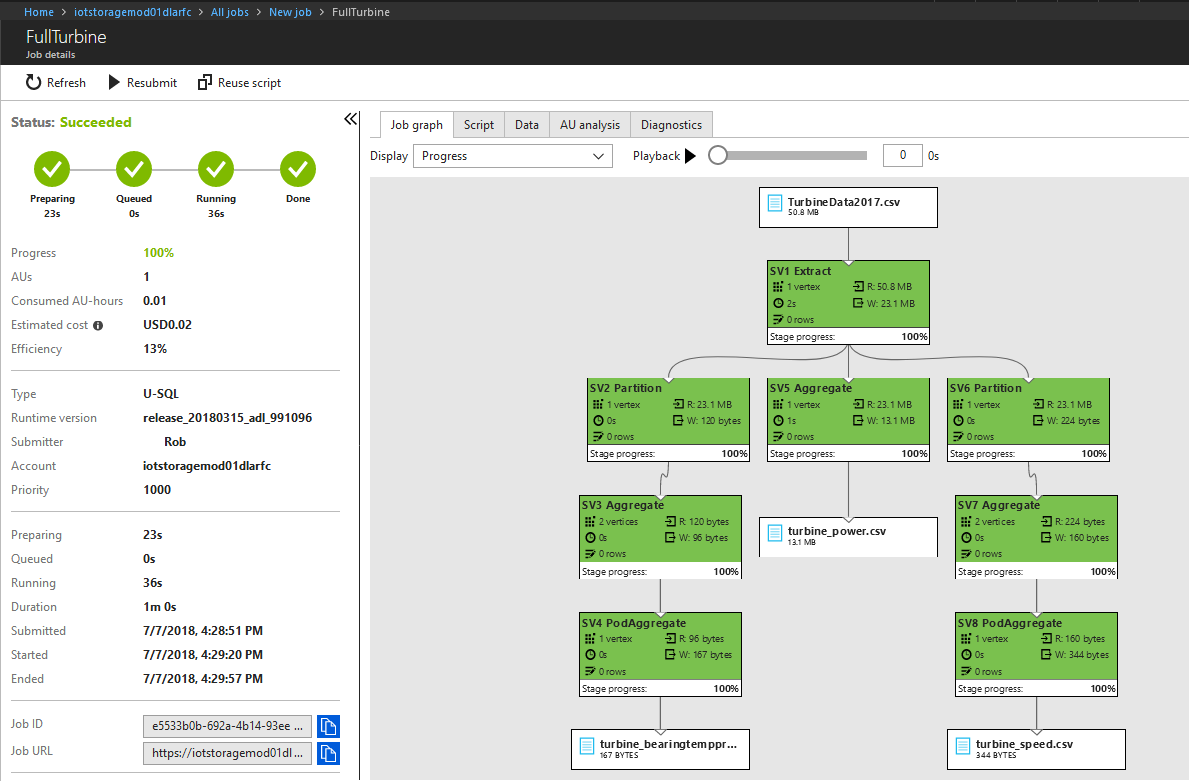
You will see aggregated data that looks similar to the following:

| **Wind\_turbine\_name** | **AvgGeneratorSpeed** | **AvgWindSpeed** |
| --- | --- | --- |
| R80711 | 1155.3409896154276769526942414 | 5.7141909477811205823908481438 |
| R80721 | 1057.1523036635778248430126705 | 5.2307390477307557052693923634 |
| R80736 | 1061.8179266976232980332829047 | 5.3056885600686058360352014822 |
| R80790 | 1103.8741829208970218472369671 | 5.4804645058541762976386397677 |

1. Navigate back to the blade containing your U-SQL code.
2. Consider the final use case scenario:

There have been costly incidents where turbines have had to shut down when their bearings overheated. A business analyst would like to identify how often bearings have reached dangerous temperatures.

1. At the bottom of the code, to create a query to find how often bearings temperatures have risen more than 2 standard deviations beyond its normal temperature, and the average temperature when that happened, enter the following code:
2. DECLARE @outputFile2 string = "/mod01/historical/analysis/turbine\_bearingtempprobs.csv";
3. @rs2 =
4. SELECT Wind\_turbine\_name,
5. COUNT(bearingtemp\_std) AS bearingTempAnomalies,
6. AVG(bearingtemp\_avg) AS bearingTempAvg
7. FROM @turbineData
8. GROUP BY Wind\_turbine\_name
9. HAVING bearingtemp\_std > 2.0M;
10. OUTPUT @rs2
11. TO @outputFile2
12. USING Outputters.Csv(outputHeader:true);
13. Click **Submit**, and then let the job run.



1. On the Job Graph tab, to view the resulting file, click **turbine\_bearingtempprobs.csv**

You should see output similar to the following:

| **Wind\_turbine\_name** | **bearingTempAnomalies** | **bearingTempAvg** |
| --- | --- | --- |
| R80711 | 9 | 49.984444444444444444444444444 |
| R80736 | 2 | 30.655 |
| R80790 | 34 | 30.684999958823529411764705882 |

1. Notice the following:

Turbine R80790 has a greater tendency to produce bearing temperature anomalies than the other turbines. It also appears that anything over 30 degrees is a temperature to watch for.

As we will see in future modules, having historical data as a reference for operational data is valuable. For instance, based on the observation above, we could add an alert to our stream data systems that goes off when the bearings temperature rises beyond 30 degrees.

### Summary

In this module, you created an Azure Data Lake Analytics account and integrated it with an Azure Data Lake Store account. You leveraged real-world wind farm data to learn about writing U-SQL queries and running batch analysis jobs.

You also got an idea of why Azure Data Lake can be a good choice for cold storage in an IoT application architecture. Azure Data Lake has integrations to many different types of analytics tools. It also has a deep and rich integration to its own Azure Data Lake Analytics service, which allows analysis of workloads o

## Batch Analyze wind farm data with Azure Data Lake Analytics

In the previous task, you streamed data into Azure Data Lake Storage for cold storage. In this task, your objective is to leverage the capabilities of Azure Data Lake Analytics against historical wind farm data. Your team wants to discover insights and patterns in the data that it can use to help Contoso Wind Power operate more efficiently.

The architecture for Azure Data Lake Analytics service was built from the ground up for cloud scale and performance. It takes away the complexities normally associated with big data processing. It can transform, combine, aggregate and otherwise process data on a massive scale.

Azure Data Lake Storage works well with numerous other analytics tools on the market, but Azure Data Lake Analytics is purpose-built to tightly integrate with it.

1. On your Azure portal, click **+ Create a resource**.
2. On the New blade, in the Search box, type **data lake** and then press Enter.
3. In the list of filtered results, click **Data Lake Analytics**, and then click **Create**.
4. On the New Data Lake Analytics account blade, under Name, enter a unique name.

For example: **dev326xmod1cah050961dlaa**.

1. Use the existing resource group that you have been using for previous tasks.
2. Choose a region near you, and then, under Data Lake Store, click **Configure required settings**
3. On the Select Data Lake Store blade, select the Data Lake Store that you created previously.
4. Under Pricing package, ensure that **Pay-as-You-Go** is selected, and then click **Create**

The account will take a few moments to provision.

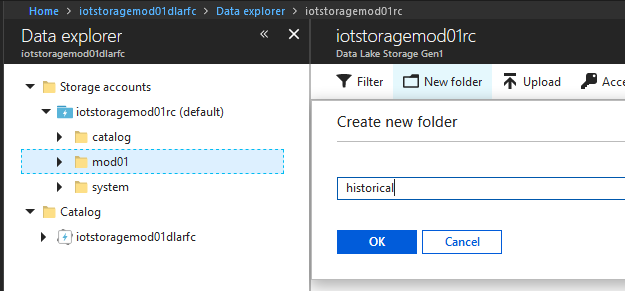
1. Once the provisioning process has completed, open your new Data Lake Analytics resource.

**Note**: You may need to refresh your dashboard before you see it listed in your resources.

1. On the Data Lake Analytics Overview blade, take a minute to review the options listed along the top menu.

You can create new jobs, you can work with sample scripts and you can navigate to the Data explorer in your storage account, you can configure security, etc.

1. At the top of the blade, click **Data explorer**
2. On the Data explorer blade, under Storage accounts, click **mod01**
3. On the Data Lake Storage blade, on the top menu, click **New folder**
4. In the folder name box, type **historical** and then click **OK**



1. To open the historical folder, click **historical**

This is where you will upload a source file for historical data analysis.

1. Save the following CSV file to your local machine.

[TurbineData2017.csv](https://prod-edxapp.edx-cdn.org/assets/courseware/v1/96c136b4bab9dbbdcfa779c63220301c/asset-v1:Microsoft+DEV326x+1T2019+type@asset+block/TurbineData2017.csv)

**Note** To open a Save As dialog, right-click the file name above and then click **Save As**. Save the file to a location that is easy to access.

This file is based on real-world wind farm data that ENGIE has made available for study [here](https://opendata-renewables.engie.com/pages/home/).

This file (and the original source) will give you a sense of what real-world turbine data looks like.

| **Wind\_turbine\_name** | **Date\_time** | **Ws\_avg** | **Ws\_std** | **Ot\_avg** | **Ds\_avg** | **Va\_avg** |
| --- | --- | --- | --- | --- | --- | --- |
| R80721 | 2017-02-07T21:40:00-07:00 | 1.9 | 0.30000001 | 5.3000002 | 88.18 | 25.18 |
| R80721 | 2017-02-07T23:40:00-07:00 | 0.2 | 0.52999997 | 4.8899999 | 23.01 | 44.310001 |
| R80721 | 2017-02-08T00:00:00-07:00 | 0.18000001 | 0.44 | 4.8000002 | 38.110001 | 39.939999 |
| R80721 | 2017-01-25T14:30:00-07:00 | 4.9099998 | 0.43000001 | -6.4200001 | 1209.53 | -1.9 |
| R80721 | 2017-01-25T16:40:00-07:00 | 4.1999998 | 0.31999999 | -7.3000002 | 1006.33 | 8.5299997 |
| R80721 | 2017-01-25T18:40:00-07:00 | 4.8499999 | 0.38999999 | -7.1599998 | 1205.52 | 11 |
| R80721 | 2017-01-25T20:20:00-07:00 | 5.1100001 | 0.28999999 | -6.98 | 1158.66 | -11.97 |
| R80721 | 2017-01-26T05:50:00-07:00 | 5.04 | 0.47999999 | 1.79 | 1181.7 | 3.8900001 |

The file also represents a real-world business scenario you are likely to encounter – the need to load and analyze historical data that may or may not be in the same format as your current data.

For the purposes of this course, we will say that this data represents historical telemetry readings that we need to make available to Contoso Wind Power’s business analysts.

1. On the Data Lake Storage blade, on the top menu, click **Upload**
2. On the Upload files blade, open the saved .csv file, and then click **Add selected files**

The STATUS will change to “Uploading” and the upload progress will be displayed. The STATUS will change to Completed when the upload has finished.

1. Close the **Upload files** blade.

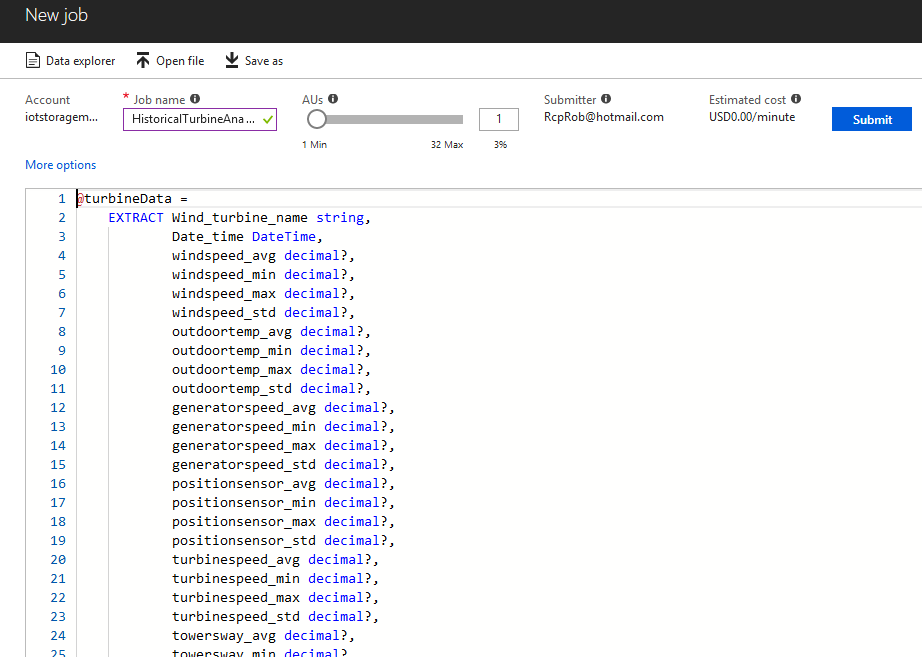
You should now see the TurbineData2017.csv file listed in your folder named **historical**.

1. With the historical folder open, click **New folder**, and then name the folder **analysis**

This is where the output files will be written.

1. Navigate back to the Azure Data Analytics Overview blade, and then click **+ New job**
2. On the New job blade, in the Job name textbox, enter **HistoricalTurbineAnalysis**

The large area under the More options label will be used for U-SQL code, which will extract the information from the TurbineData2017.csv file.



1. In the text box for U-SQL code, enter the following code:
2. @turbineData =
3. EXTRACT Wind\_turbine\_name string,
4. Date\_time DateTime,
5. windspeed\_avg decimal?,
6. windspeed\_min decimal?,
7. windspeed\_max decimal?,
8. windspeed\_std decimal?,
9. outdoortemp\_avg decimal?,
10. outdoortemp\_min decimal?,
11. outdoortemp\_max decimal?,
12. outdoortemp\_std decimal?,
13. generatorspeed\_avg decimal?,
14. generatorspeed\_min decimal?,
15. generatorspeed\_max decimal?,
16. generatorspeed\_std decimal?,
17. positionsensor\_avg decimal?,
18. positionsensor\_min decimal?,
19. positionsensor\_max decimal?,
20. positionsensor\_std decimal?,
21. turbinespeed\_avg decimal?,
22. turbinespeed\_min decimal?,
23. turbinespeed\_max decimal?,
24. turbinespeed\_std decimal?,
25. towersway\_avg decimal?,
26. towersway\_min decimal?,
27. towersway\_max decimal?,
28. towersway\_std decimal?,
29. powergeneration\_avg decimal?,
30. powergeneration\_min decimal?,
31. powergeneration\_max decimal?,
32. powergeneration\_std decimal?,
33. bearingtemp\_avg decimal?,
34. bearingtemp\_min decimal?,
35. bearingtemp\_max decimal?,
36. bearingtemp\_std decimal?,
37. windingstemp\_avg decimal?,
38. windingstemp\_min decimal?,
39. windingstemp\_max decimal?,
40. windingstemp\_std decimal?
41. FROM "/mod01/historical/TurbineData2017.csv"
42. USING Extractors.Csv(skipFirstNRows:1);

Notice that the column names in the U-SQL EXTRACT statement do not match the header values in the TurbineData2017.csv file. For instance, the Ws\_avg column from the file is queried as windspeed\_avg column in the U-SQL query.

This is an important concept to grasp. Azure Data Lake Analytics can coerce and transform unstructured data into structured data. Notice, for instance, that query uses a string data type for the Wind\_turbine\_name and a nullable decimal data type (decimal?) for the outdoortemp\_av property. Those data types are not defined in the source csv file – they are converted by the analytics query.

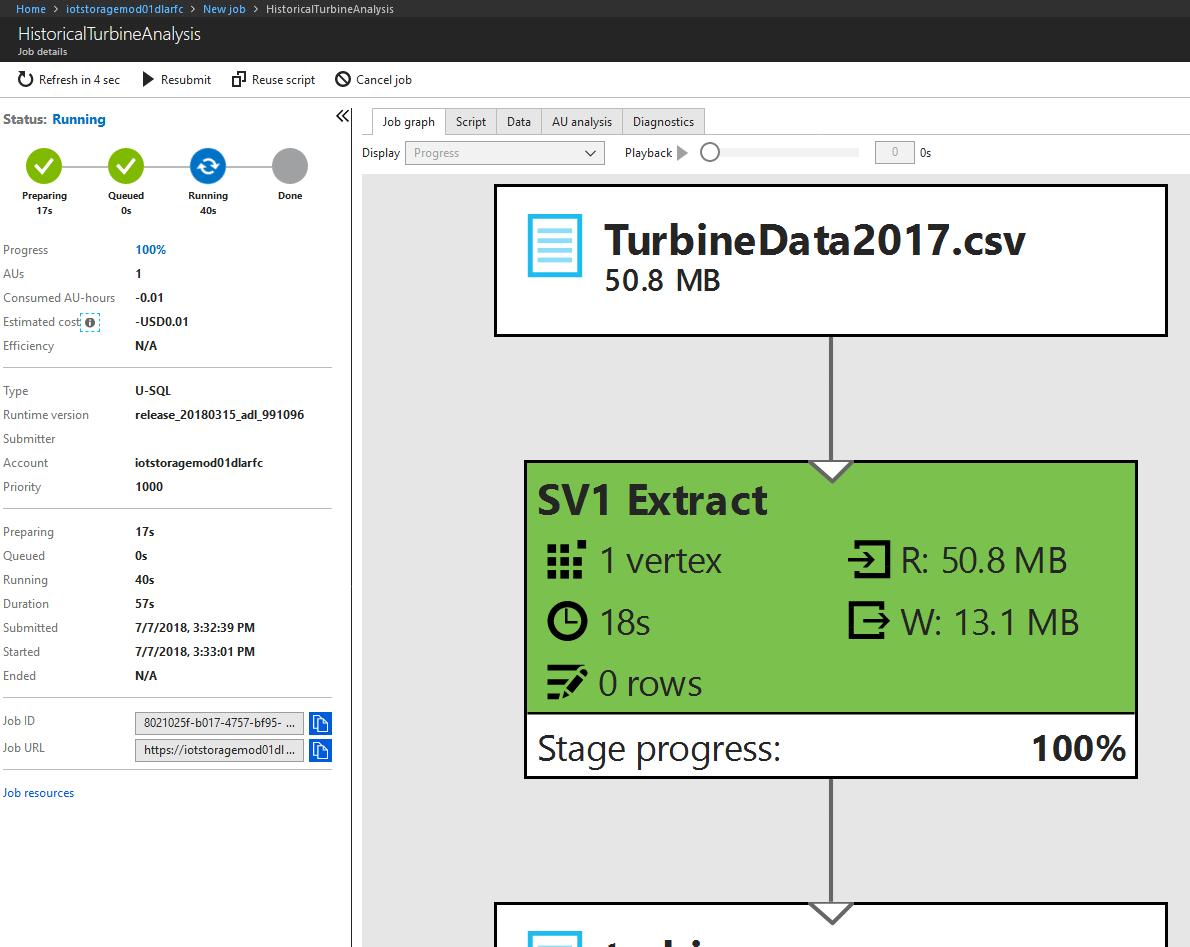
So now you have the entire dataset available as input. Next you need transform the data and serialize it somewhere.

1. In the text box for U-SQL code, to establish an output file, enter the following code:
2. DECLARE @outputFile0 string = "/mod01/historical/analysis/turbine\_power.csv";
3. Consider this first use case scenario:

Let’s say that a business analyst would like a report on some of the key historical readings for the wind turbines – windspeed, temperature and power generation that she can use for analysis. But she doesn’t want the entire historical file with all of its columns.

1. To create a query that simply narrows down the columns that are captured, enter the following code:
2. @rs0 =
3. SELECT Wind\_turbine\_name,
4. Date\_time,
5. windspeed\_avg,
6. outdoortemp\_avg,
7. powergeneration\_avg
8. FROM @turbineData;
9. Finally, to write an OUTPUT statement that will flush the queried data to the turbine\_power.csv file, enter the following code:
10. OUTPUT @rs0
11. TO @outputFile0
12. ORDER BY Wind\_turbine\_name,
13. Date\_time
14. USING Outputters.Csv(outputHeader : true);
15. Click **Submit**

This will kick off the process of preparing, queuing and running the job. Watch the steps proceed until done.



1. At the bottom of the Job Graph tab, notice the box indicating the turbine\_power.csv file.
2. To open a File preview blade displaying the data, click **turbine\_power.csv**.
3. Compare this data to the query that you wrote.

| **Wind\_turbine\_name** | **Date\_time** | **windspeed\_avg** | **outdoortemp\_avg** | **powergeneration\_avg** |
| --- | --- | --- | --- | --- |
| R80711 | 2016-12-31T16:00:00.0000000-07:00 | 4.5599999 | 3.8299999 | 76.489998 |
| R80711 | 2016-12-31T16:10:00.0000000-07:00 | 4.9899998 | 3.8199999 | 82.330002 |
| R80711 | 2016-12-31T16:20:00.0000000-07:00 | 4.8400002 | 3.8499999 | 66.480003 |
| R80711 | 2016-12-31T16:30:00.0000000-07:00 | 4.6999998 | 3.8299999 | 83.230003 |
| R80711 | 2016-12-31T16:40:00.0000000-07:00 | 4.8699999 | 3.78 | 85.25 |
| R80711 | 2016-12-31T16:50:00.0000000-07:00 | 4.98 | 3.5699999 | 85.010002 |

1. **Note**: You can also download the data to see the full file. As you may have expected, it has all the rows of the original file, just fewer columns, based on the query.
2. Close the File preview blade, and then close the HistoricalTurbineAnalysis Job details blade.

You should find yourself back at the blade containing the U-SQL code that you entered earlier.

1. Consider a second use case scenario:

A business analyst who would like to analyze how efficient individual wind turbines are at running their generators, compared with wind speed.

**Note** We will reuse the existing EXTRACT, FROM, and USING statements that we entered above.

1. At the bottom of the code, to create a query that aggregates the historical generator speed vs. wind speed, enter the following:
2. DECLARE @outputFile1 string = "/mod01/historical/analysis/turbine\_speed.csv";
3. @rs1 =
4. SELECT Wind\_turbine\_name,
5. AVG(generatorspeed\_avg) AS AvgGeneratorSpeed,
6. AVG(windspeed\_avg) AS AvgWindSpeed
7. FROM @turbineData
8. GROUP BY Wind\_turbine\_name;
9. OUTPUT @rs1
10. TO @outputFile1
11. USING Outputters.Csv(outputHeader:true);

Notice that the SELECT statement is identical to how you would write it if you were querying a comparable database table in SQL Server. U-SQL is its own language, but like many of Microsoft’s analytical tools, it benefits from its similarity with the ubiquitous SQL language. The GROUP BY clause, the AVG operator, and column aliasing – all the same.

1. Click **Submit**

The updated job will now output two files, rather than one. Let it run to completion.

1. On the Job Graph tab, click **turbine\_speed.csv**

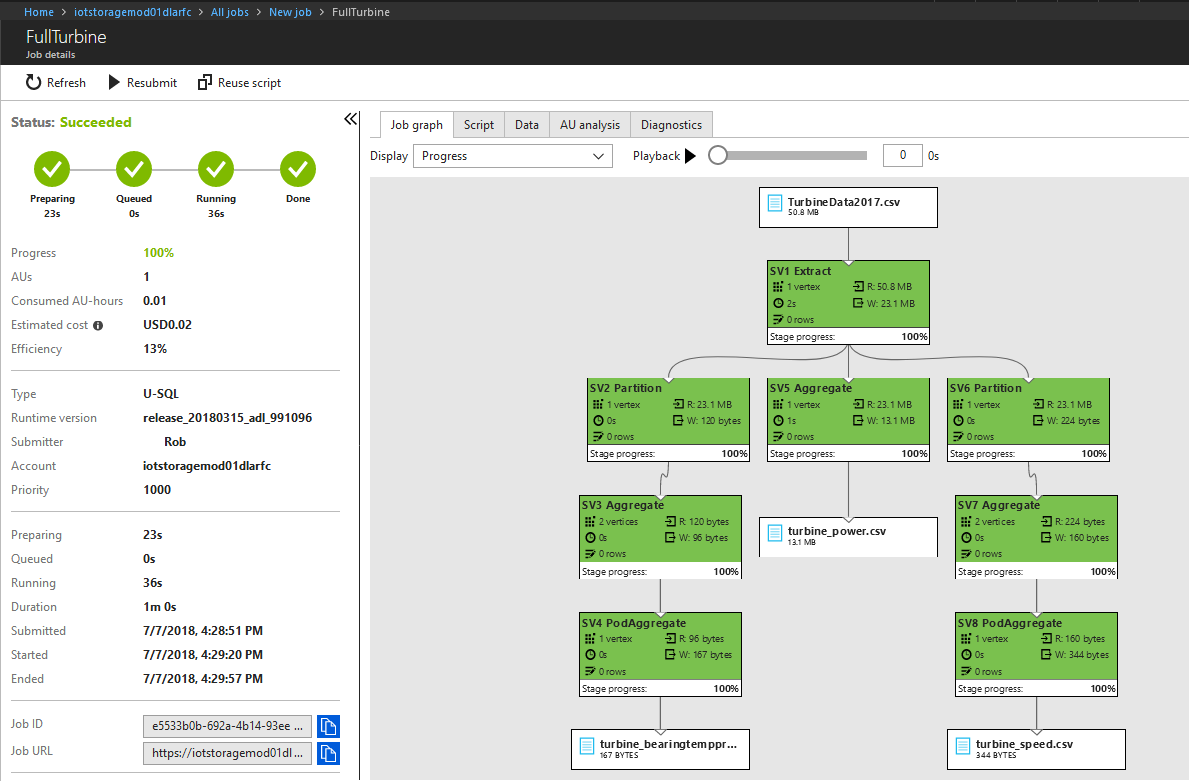
You will see aggregated data that looks similar to the following:

| **Wind\_turbine\_name** | **AvgGeneratorSpeed** | **AvgWindSpeed** |
| --- | --- | --- |
| R80711 | 1155.3409896154276769526942414 | 5.7141909477811205823908481438 |
| R80721 | 1057.1523036635778248430126705 | 5.2307390477307557052693923634 |
| R80736 | 1061.8179266976232980332829047 | 5.3056885600686058360352014822 |
| R80790 | 1103.8741829208970218472369671 | 5.4804645058541762976386397677 |

1. Navigate back to the blade containing your U-SQL code.
2. Consider the final use case scenario:

There have been costly incidents where turbines have had to shut down when their bearings overheated. A business analyst would like to identify how often bearings have reached dangerous temperatures.

1. At the bottom of the code, to create a query to find how often bearings temperatures have risen more than 2 standard deviations beyond its normal temperature, and the average temperature when that happened, enter the following code:
2. DECLARE @outputFile2 string = "/mod01/historical/analysis/turbine\_bearingtempprobs.csv";
3. @rs2 =
4. SELECT Wind\_turbine\_name,
5. COUNT(bearingtemp\_std) AS bearingTempAnomalies,
6. AVG(bearingtemp\_avg) AS bearingTempAvg
7. FROM @turbineData
8. GROUP BY Wind\_turbine\_name
9. HAVING bearingtemp\_std > 2.0M;
10. OUTPUT @rs2
11. TO @outputFile2
12. USING Outputters.Csv(outputHeader:true);
13. Click **Submit**, and then let the job run.



1. On the Job Graph tab, to view the resulting file, click **turbine\_bearingtempprobs.csv**

You should see output similar to the following:

| **Wind\_turbine\_name** | **bearingTempAnomalies** | **bearingTempAvg** |
| --- | --- | --- |
| R80711 | 9 | 49.984444444444444444444444444 |
| R80736 | 2 | 30.655 |
| R80790 | 34 | 30.684999958823529411764705882 |

1. Notice the following:

Turbine R80790 has a greater tendency to produce bearing temperature anomalies than the other turbines. It also appears that anything over 30 degrees is a temperature to watch for.

As we will see in future modules, having historical data as a reference for operational data is valuable. For instance, based on the observation above, we could add an alert to our stream data systems that goes off when the bearings temperature rises beyond 30 degrees.

### Summary

In this module, you created an Azure Data Lake Analytics account and integrated it with an Azure Data Lake Store account. You leveraged real-world wind farm data to learn about writing U-SQL queries and running batch analysis jobs.

You also got an idea of why Azure Data Lake can be a good choice for cold storage in an IoT application architecture. Azure Data Lake has integrations to many different types of analytics tools. It also has a deep and rich integration to its own Azure Data Lake Analytics service, which allows analysis of workloads of any size.