Course Overview

Course Overview

Hello everyone! My name is Warner Chaves, and welcome to my course, Delivering Real-time Data with Azure and Power BI Streaming. I'm a Microsoft data platform MVP at Pythian. And in this course, we're going to build a real-time dashboard solution with Azure and Power BI. Some of the major topics that we are going to cover include ingesting data into event hubs, analyzing data on the fly with stream analytics, and adding real-time visuals to Power BI dashboards. By the end of this course, you'll know the main pieces of a real-time architecture in Azure and then know how to get started building one. Before beginning the course, you only need to be familiar with basic use of Azure and Power BI. From here, you should feel comfortable diving into more complex scenarios with more advanced courses at Pluralsight around event hubs, stream analytics, or Power BI. I hope you'll join me on this journey to deliver faster value and faster insights with Delivering Real-time Data with Azure and Power BI Streaming here at Pluralsight.

Real-time Data and Azure

What Is Real-time Data?

Hi! This is Warner Chaves with Pluralsight. Welcome to my course, Delivering Real-time Data with Azure and Power BI Streaming. This is module 1, Real-time Data and Azure. In this module, we're going to go over the basic concepts to understand what we mean by real-time data and why we should care about it. We're also going to look into an example architecture that enables a company to consume and analyze real-time data, and it is the architecture that we are going to use throughout this course. We are then going to look at the individual Azure components from a high-level point of view that we're going to use as part of our architecture. And then, finally, we are going to do an overview of the solution that you guys are going to be joining me in building during the course. Before diving in into details, let's answer the question, What is real-time? The definition of real-time data is that it is delivered to the end consumer at the same rate as it is generated. So ideally if we are generating data points every 5 seconds, that is the same refresh rate where the user will see their dashboard change. However, a lot of the time, clients are okay with higher delays, even if they're not consuming the data at the same rate as it is generated. What I find is that most use cases, a 1-minute or less delay is still considered real-time. So we have, for example, a set of devices. They are pushing data to some sort of system that will process it and might store it. And then it will push it forward into some sort of reporting application or visualization where users can manipulate it, analyze it, and possibly get alerted on differing conditions. So the entire process from the devices pushing into the streaming system, then the streaming system moving it forward to the visualization layer takes 1 minutes or less. So why should we care about real-time and enabling these new types of data solutions? The truth is that modern business competitiveness and user expectations demand faster time to insight. Users are not happy anymore with having to wait hours or days to get access to the data that is important to them. In a traditional environment, you would have batch style processing where we accumulate data over long periods of time and then we process it. In the modern environment, we will have streaming data that will produce real-time visualizations and give the users that data as fast as possible. In a traditional environment, this also meant that the data points that users will see could have delays of hours or sometimes even days. In this modern environment, the data delay is measured in seconds or minutes at the most. In a traditional environment, there was also a lot of requirement for infrastructure expertise. We would need to know about queueing systems. We would need to know about messaging systems. We would need to know about streaming systems. And then we would need to also know about hosting our own reporting tools. These modern cloud-based environments all come with cloud turnkey components where we don't have to manage any infrastructure at all. We just have to deploy the components and start using them. Finally, in a traditional environment, the reporting and visualization tools are gated by dedicated IT or BI teams, which sometimes become bottlenecks for the end users to get the visualizations and reports that they need as quickly as possible. Compared to this, dynamic self-service BI tools such as Power BI enable the users to consume the dataset and then easily change their visualizations, create dashboards, and so on without any sort of administration involvement.

Real-time Data Architecture

Let's check an example architecture for doing real-time data analysis with Azure and Power BI. This is the architecture that we will use during the course. Basically what we're going to do is demystify this black box that I mentioned before, the one that takes that data from all these devices and then pushes the data into a visualization layer. In this case, we're going to be using Microsoft Azure. Keep in mind we're going to look at one, but it's not the only one, reference architecture to do this with Azure and Power BI. There might be components that will not be necessary depending on your use case and your organization's requirements. However, this architecture that we are going to use will give you a good starting point regardless of your final requirements. First, we're going to start again with our set of devices that send data very frequently. Then those devices are going to push that data into what is called an event hub. From that event hub, that data gets moved into a service called Stream Analytics, which can run queries on the fly to that data. And then from Stream Analytics, Stream Analytics will push the analyzed or aggregated data into Power BI where it can be manipulated and visualized by the end users. Using this architecture, for example, we could enable also a batch processing layer to this platform. Event Hubs could save that data coming from the devices into files in blob storage. Once that data is in blob storage, then we could analyze it with a big data tool such as HDInsight. If we have structured data, then we could also use something like Azure's SQL Data Warehouse and the loading technology called PolyBase to load directly from blob storage into tables in SQL Data Warehouse. If it was necessary, we could even build online analytical models using Azure Analysis Services consuming the data from SQL Data Warehouse. And then, finally, we could consume those models back in Power BI. So as you can see, this particular architecture you can remove many pieces depending on your requirements. For this course, we are going to be focusing on the piece at the top of the diagram, which is the piece that does the real-time consumption, analysis, and visualization with Power BI.

Azure Components

Let's look at the individual components of our solution. First, we have Azure Event Hubs. The way to think about Azure Event Hubs is as a landing pad to Azure. It's where the devices send their data that's an entry point into an Azure datacenter. Azure Event Hubs consumes data in any form. It does not enforce any sort of format or schema on the data. It supports stream and batch data, stream by just pushing directly into Event Hubs and being consumed directly from Event Hubs, and also batch data by using a feature called Event Hubs capture that basically takes the output of Event Hubs and stores it as files. It is highly scalable and completely transparent to do so. You can just get a larger event hub with more capacity and pay more dollars for it of course. And, finally, it is a fully managed service. Again, there is no infrastructure to manage. You simply deploy your event hub, and then you can immediately start pushing data into it without having to worry about any sort of infrastructure. The next component is Azure Stream Analytics. And what you want to think of here is that it lets you analyze your data on the fly. So Azure Stream Analytics can use SQL and JavaScript to query a stream of data coming from different types of sources. And we will look at it in more detail later on in the course. It transforms the data inflight, so it consumes the data, runs some sort of transformation on it depending on your SQL and JavaScript, and then outputs this transformed data into another service that is going to consume it. It is highly scalable as well and transparent. So if you need more capacity, you can just scale up the service, and if you don't, you can just scale it down. And, finally, just like Event Hubs, it is fully managed. You don't have to do anything in regards to infrastructure. And the last piece that we're going to use is Microsoft Power BI, which is Microsoft's cloud service for rich, self-service reports and dashboards. Power BI is a good fit because it is fully integrated with Azure. It is very easy to pick up. It's very easy to get data loaded into the service, and then with a few clicks create some nice-looking visualizations. It is built for collaboration so that one person can create a report or dashboard and then easily share it with their entire organization. And, finally, just like all the other components that we're going to use in this course, it is fully managed by Microsoft. You simply use the Power BI service. You do not have to manage your own infrastructure. By having an architecture that lives completely in the cloud, we get several big advantages over having our own on-premises solution. We can deploy within minutes through the Azure portal or using the PowerShell. There is no infrastructure to manage, so we don't have to worry about operating systems. We don't have to worry about hypervisors, and we certainly don't have to worry about managing hardware. We can scale up or down as needed. For example, if we needed to consume more data, then we could scale up Event Hubs and Stream Analytics, and once that event has passed, we can scale down the services and then immediately start getting some savings. And, finally, because they are all built in Azure and in the case of Power BI made also by Microsoft, these services are all tightly integrated and don't require any sort of coding for integrations or API connections.

Course Solution Overview

Now let's check out the solution that we are going to be building throughout the course. We are going to do a machinery temperature monitoring dashboard. We are going to have a warehouse that is connected to Azure, any of the Azure datacenters where these services are located. And then we are going to have machines that stream their temperature readings multiple times per second to Azure. Once we have those readings, we are going to aggregate them and add reference information to them. With that information, we are then going to let the users visualize it and receive alerts on them in real-time. This same architecture and type of solution has many applications in different industries so that you can take what you learned through the course and apply it to your specific company. For example, you could use this to analyze user behavior using a website or using an application or playing a game. You could also use it to monitor a fleet of vehicles. This could be cars. This could be planes, ships, etc. And, finally, you could visualize sales and inventory movements in real-time to provide the fastest insights and the biggest competitive advantage to a retail company. As we go through the course, we are going to be building this solution piece by piece, and each module will be building on the previous one. So after this module, we are going to go into module 2, Consuming Data with Event Hubs. Once we have that data in Event Hubs, we will go into module 3, Analyzing Data On-the-fly with Stream Analytics. And once we have Stream Analytics analyzed in our data on-the-fly, we will go into module 4, Ingesting Streaming Data into Power BI. And once we have landed that data into Power BI, we will go into the final module for our course, module 5, Building Real-time Visualizations with Power BI. The tools that we're going to need for the course are Visual Studio 2017. I'm going to be using the community edition and really the requirement is to have a C# 4. 5 or above compiler. We're going to be using an Azure subscription of course because Event Hubs and Stream Analytics are Azure services. And we're also going to need a Power BI subscription for the Power BI part of the course. A free Power BI subscription is more than enough to follow along. A quick note on keeping up to date. Even though the fundamentals and the services used in the course are not going to change, in the cloud, small service details can change all the time. Refer to azure. com and powerbi. com for the latest information.

Module Summary

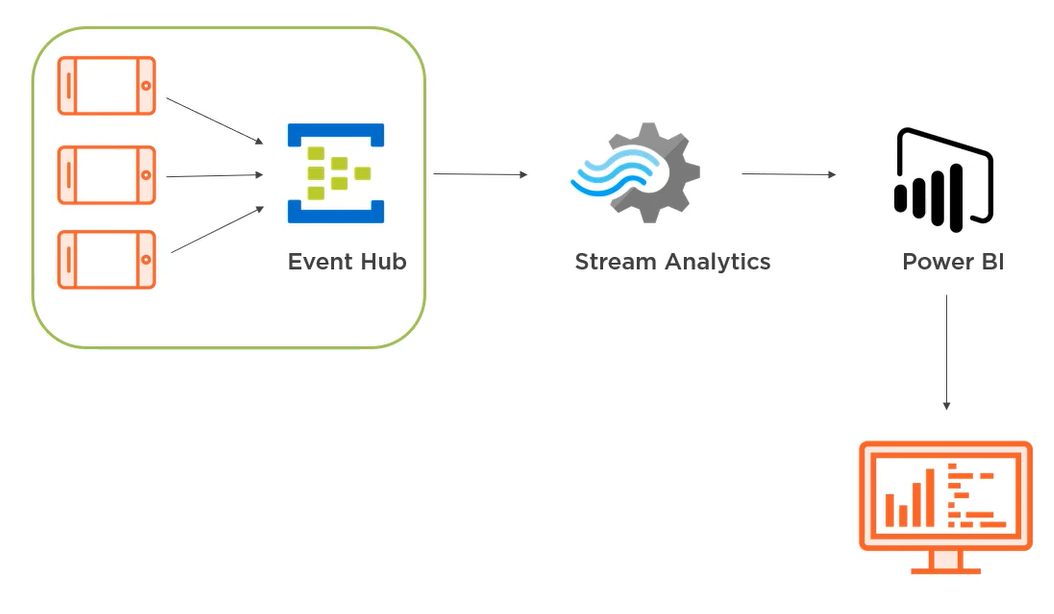
In summary, we checked how the demand for real-time data is increasing. And these types of solutions have some really big advantages over traditional batch-style solutions. In Azure, we can architect a solution that requires no infrastructure management, that is also very elastic and very flexible. As part of the course, we are going to be building a temperature monitoring dashboard. And this particular solution can be applied to all sorts of different industries like software application, website user tracking, monitoring fleets of vehicles, or retail activity monitoring. Join me on the next module as we start building our solution by consuming data through Event Hubs.

# Consuming Data Through Event Hubs

## The Use Case for Event Hubs

Hi! This is Warner Chaves with Pluralsight. Welcome to the next module in our course, Delivering Real-time Data with Azure and Power BI Streaming. This is Consuming Data through Event Hubs.

In this module, we're going to go over the Event Hub fundamentals that you need to understand to use the service. We're also going to go into what is necessary to create an Event Hub and manage it. And then we're going to look into the actual process of pushing data through Event Hub using C# as an example. As a reminder, Azure Event Hubs is the number one component in our solution. If you want a quick and easy way to remember what it does, just remember it is the landing pad to Azure. As a reminder, this is the diagram of the solution that we're going to build. We're taking devices that stream data to Event Hub. Then that data's going to get pushed to Stream Analytics, then from Stream Analytics to Power BI, and then, finally, to be visualized by the end user. In this module, we're going to focus on this piece of the diagram, the part that streams the data from devices into Event Hubs.

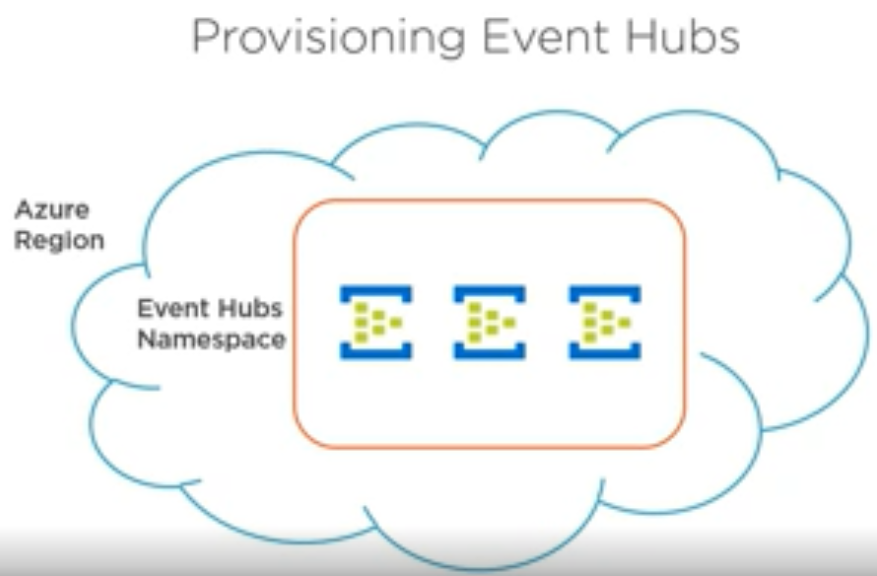


Since we are pushing data into Azure, there is a valid question here of why would we use an Event Hub, and why not something like, for example, a SQL database? Well, there are several advantages that Event Hubs have. For example, they have flexibility.

* The Event Hub does not need to define a schema. It does not need to define indexes. And it has no preconceived idea of the data that we're going to push into it. It receives the data in any format.
* Scalability is another big one. Event Hubs can easily scale to consume thousands of messages or megabytes of messages with the push of a button. That scalability is easy to achieve because the system is optimized simply to receive data.
* And, finally, cost. Because the system is basically built just to receive data and output it, it can be scaled to a higher throughput at a cost that is significantly lower than doing the same, for example, with Azure SQL database, because Azure SQL database also has to support things like computing queries, whereas Event Hub simply has to receive data and keep it moving.

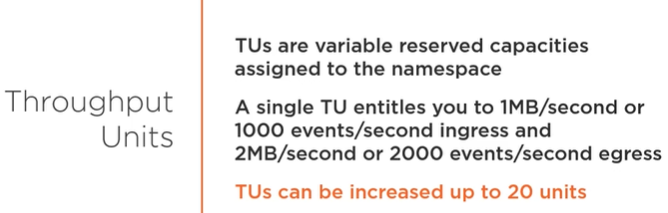
## Provisioning Event Hubs

Now that we have decided that we want Event Hubs in our solution, how do we get started working on it? Let's look at the process of provisioning Event Hubs. First, we must select an Azure region. Then we create an object called an Event Hubs namespace. Inside this namespace, then we can create multiple Event Hubs that we can send data to.

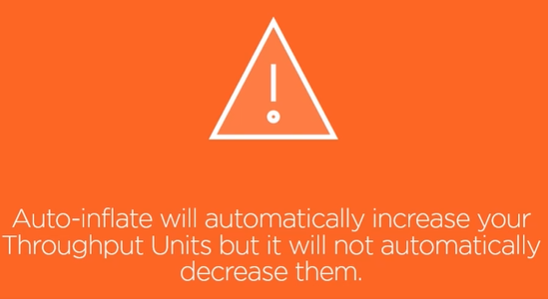


The performance of an Event Hub is determined by the concept of **throughput units**. Throughput units are variable reserved capacities assigned to the namespace. They are not assigned specifically to the Event Hub but to the namespace. That means that all the hubs under a namespace will share the throughput units.

A single throughput unit gives you a 1 mg/second or 1000 events/second in and 2 mg/second or 2000 events/second out. This is a single throughput unit. And starting from that, you can add more throughput units to increase your input and output capabilities. Throughput units can be increased up to 20 units. This is done through the portal. However, if you need to go past the capacity that you get from 20 units, you can also call Azure Support and see if they can give you even higher numbers.



Event Hubs also have a very handy feature called **auto-inflate**. Basically what it does is that when it hits the limits of the capacity of the Event Hub and you're still pushing data in through it, it can automatically increase your throughput units up to a certain limit that you set up. However, you have to be careful because auto-inflate will automatically increase your units, but it will not automatically decrease them. That means that it's up to you to go back to the portal, and then make sure that after the event has passed, you bring those throughput units back down.

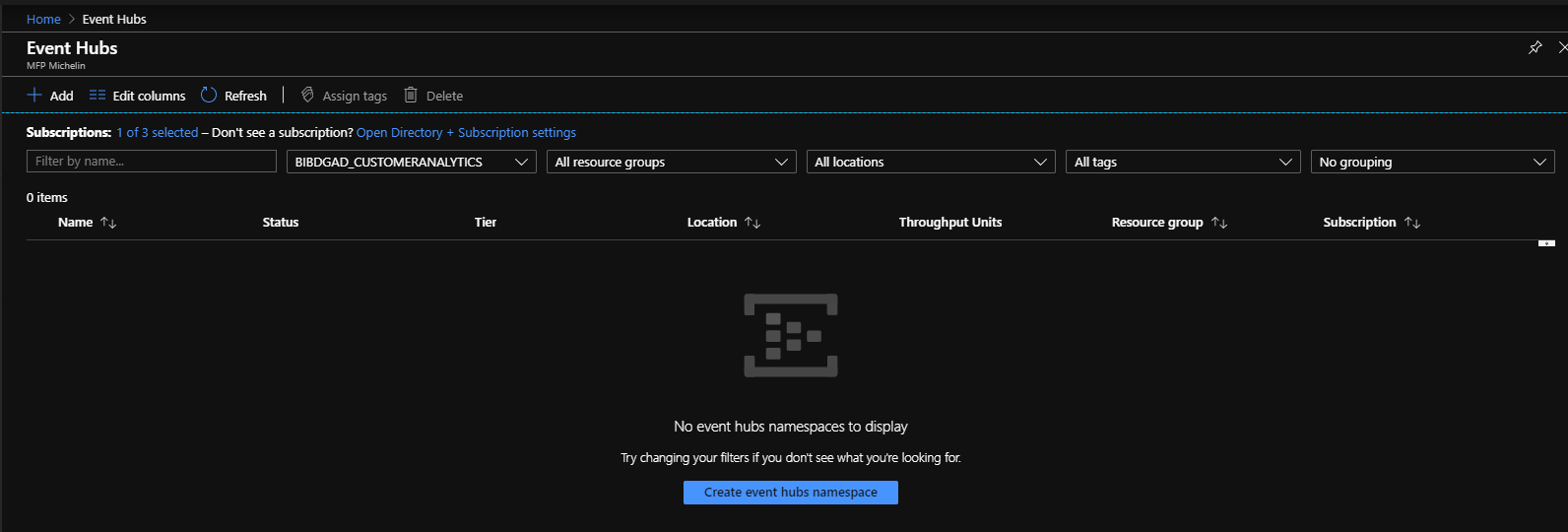


Let's check out the steps to create an Event Hub namespace. We have to pick a name for the namespace, and then we have to pick a region for it. Then we have to pick how many throughput units are going to be available for the entire namespace. And then, finally, we're going to select if we want to have **auto-inflate or not**. Once we have our namespace up and running, we can start creating Event Hubs. First, we have to pick a name for the Event Hub. Second, we have to pick a partition count, and I will go into detail into what that means in a second. Then, we have to set the message retention of the Event Hub. By default, the messages will be retained for one day, but you can set it up to a week. And, finally, we have to **enable event capture** if we desire it. Event capture will take the data from the Event Hubs and automatically move it into Azure blob storage for long-term retention.

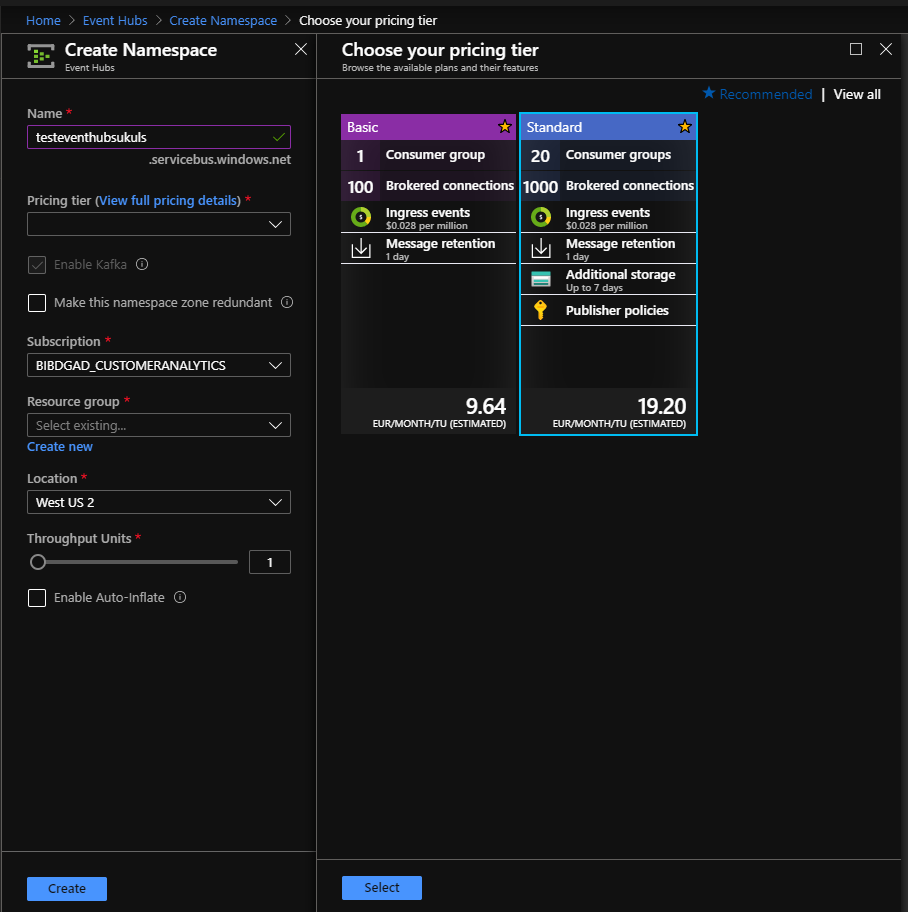
**Event Hub partitions** are a way to organize the data that is related to the number of concurrent hub readers that you're going to have. So if you have many different consumers of the Event Hub data, you want to have multiple partitions so that those readers can consume in parallel. For the purposes of the course, we are only going to have Stream Analytics consuming from the Event Hub, so we're going to stay with the default partition count of 2.

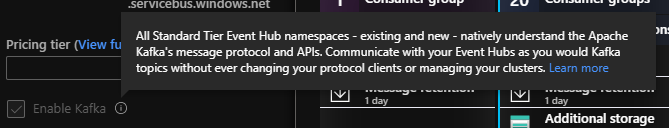
## Demo: Creating an Event Hub

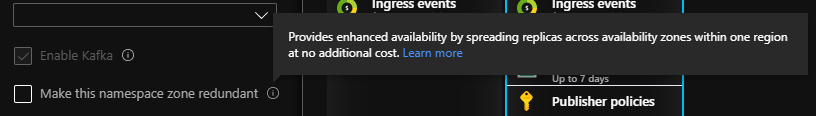
Let's check out a demo of creating an Event Hub through the Azure portal. I'm connecting now to the Azure portal, and the first thing we are going to do is to simply search for Event Hubs in the search space. Now we're going to get a result here that'll give us the link to the Event Hubs services blade. So I'm going to go down and click on it. Now this blade is going to give you the list of all the Event Hub namespaces that you have. I already have one created. I'm going to click on Add and create a new one so you guys can see the process from the very beginning.



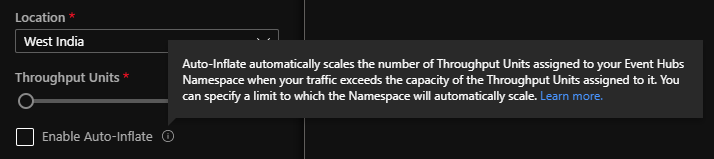
The first thing we need to do is to pick a globally unique name for our Event Hub namespace. In this case, I am just going to pick a name called coursedemo. We can see it's okay with that name. Then we have to pick a pricing tier. I'm going to click on it so we can see the two options. As we can see here, there's basic and standard. Standard gives you more capabilities for more consumers, more connections, message retention that is bigger, and additional storage. It is more expensive because of those reasons.



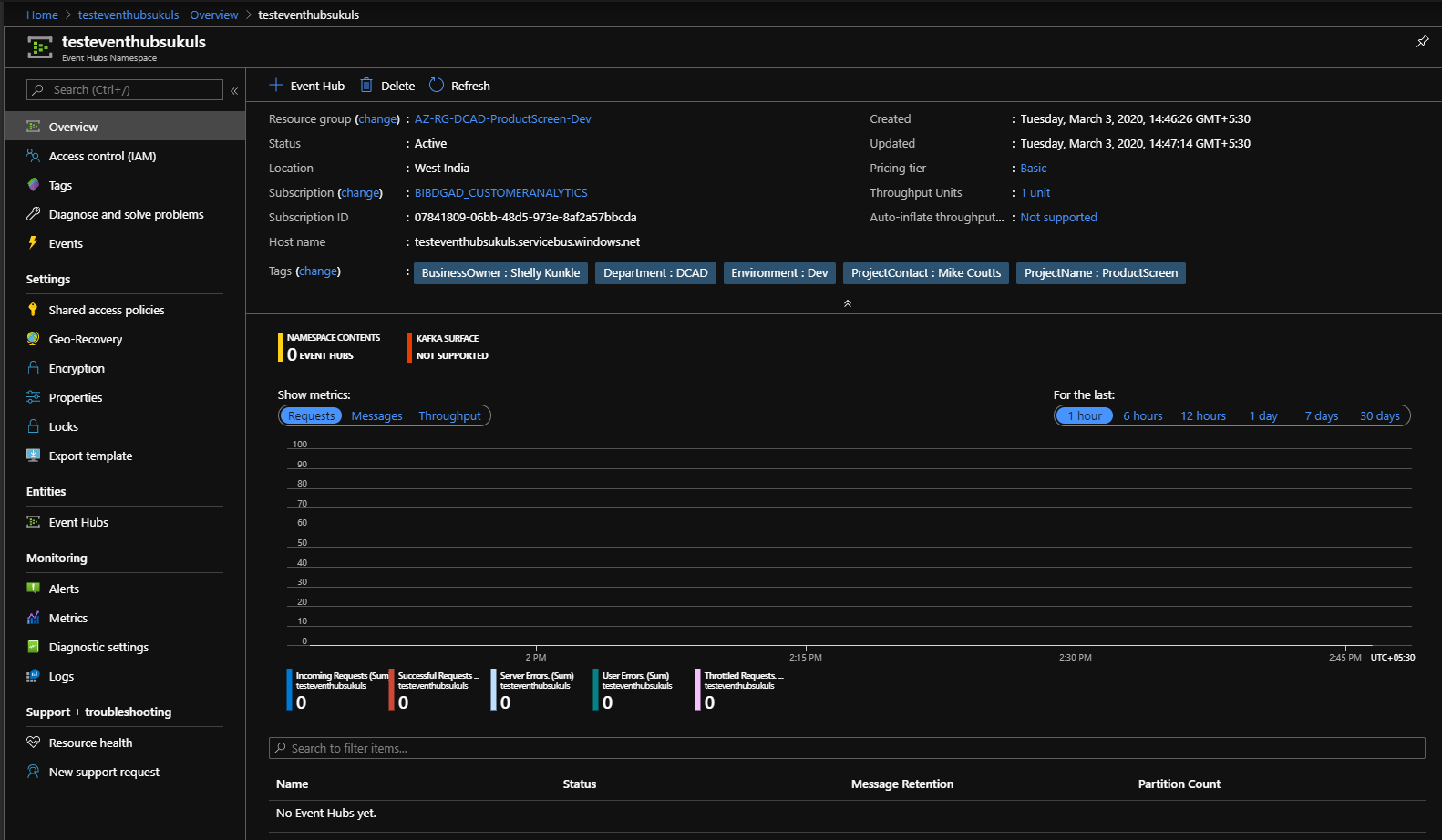




For the purposes of the course, we are okay with just selecting basic. And it is better because it will consume less resources from your subscription anyway. So we're just going to select basic. Then we need to select the subscription that we're going to be creating the Event Hub namespace in. I'm just going to leave mine like that. And then we need to create the resource group or use an existing one. I recommend to create a new resource group and give it a name like realtimecourse so that you know that all the resources that you created for our course are there in that resource group, and you can delete it once you're done with the course. Next, we need to select the location for our Event Hub namespace. It really depends on where you are located in the world. As you can see, there are many, many options of all the different regions where we can create our Event Hub namespace, so it is definitely up to where you are located so you can get low latency. In this case, I'm just going to leave it on East US. And, finally, we get the option to pick the throughput units that we want in our namespace. As I mentioned in the slides, it goes all the way from 1 to 20 as the max. For our course demo purposes, we're just going to leave it at 1, and then just click Create.

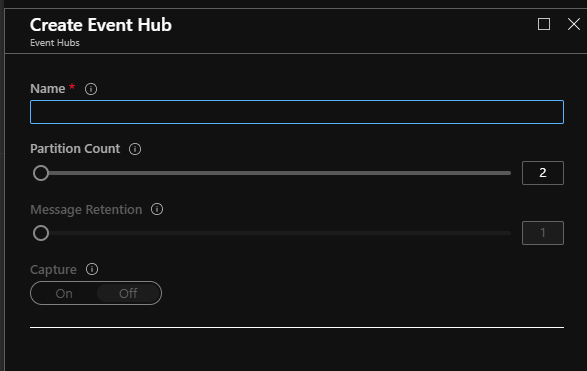


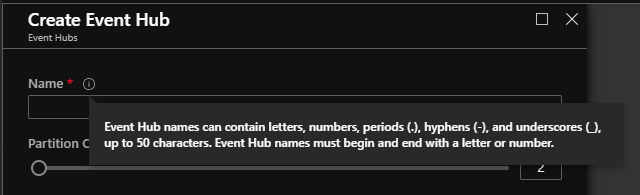
After a few minutes, we can see that our new namespace has been created, and then we can go into it. We can see here on the overview as well that we don't have any Event Hubs right now in our namespace. It is empty.

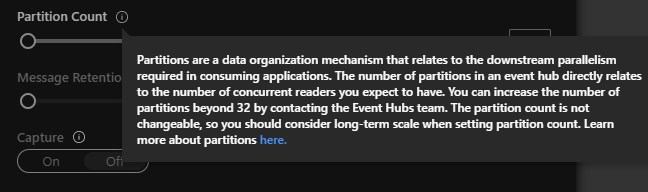


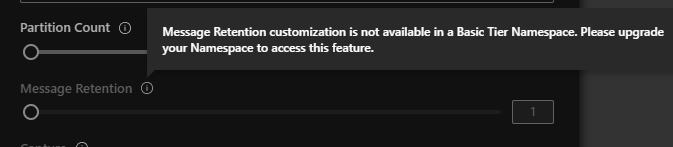
So let's go ahead and add an Event Hub. The first thing we need to do is to give it a name. In this case, it does not have to be globally unique because it's under the namespace. For our demo, I'm just going to call it FactoryFloorTemp. Then we have to select the partition course. As I mentioned on the slides, this is dependent on the amount of parallel readers that you're going to have. Since we're going to have only Stream Analytics as our reader, we're just going to do the minimum partition count of 2.

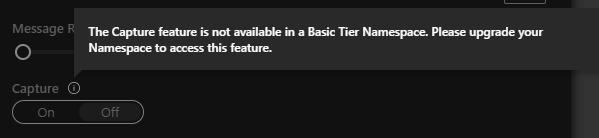
Next, if we were using an Event Hub in the standard tier, we would be able to increase our message retention. However, because we're just using basic, it is set to the default of 1. And then, finally, we can also here enable or disable the Event Hub capture, that feature that sends the Event Hub data into Azure blob storage. Because we are using Event Hub basic, we don't have the option of enabling that feature. If you want to use that feature, you have to be using Event Hub standard.

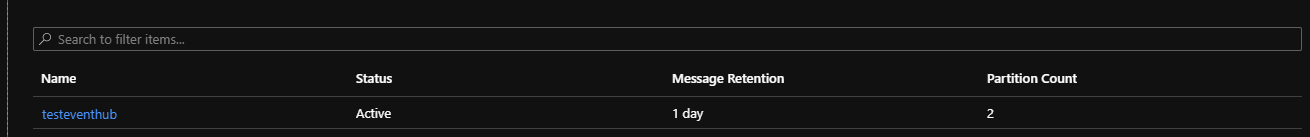




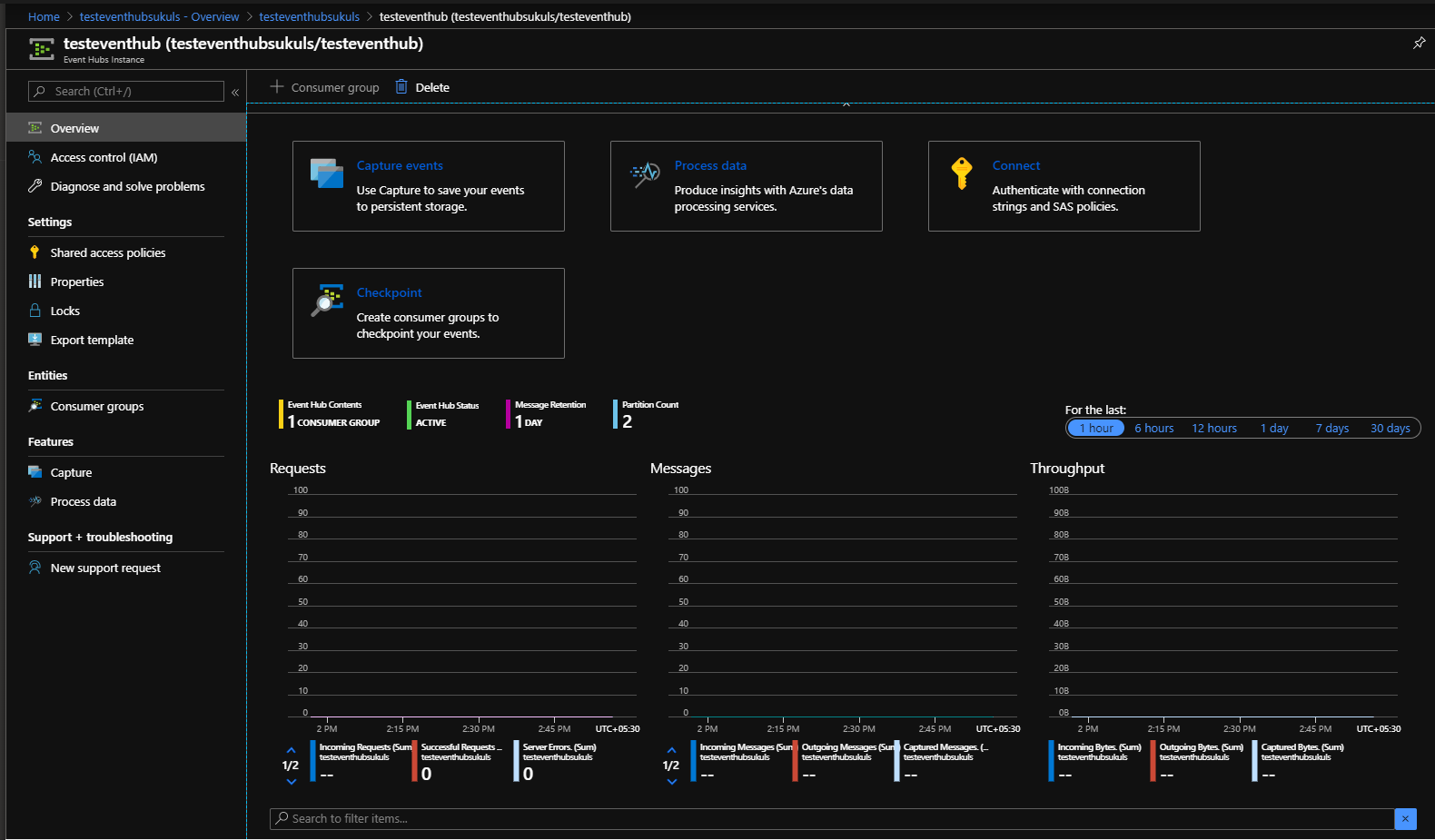




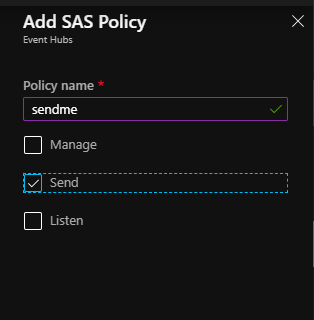
In any case, at this point I can just go ahead and click on Create. After a few seconds, now we can see that our Event Hub has been successfully created, and the namespace has one Event Hub under it. If we scroll down to the bottom, then we can see the different Event Hubs under this namespace, and we can click on them as well.



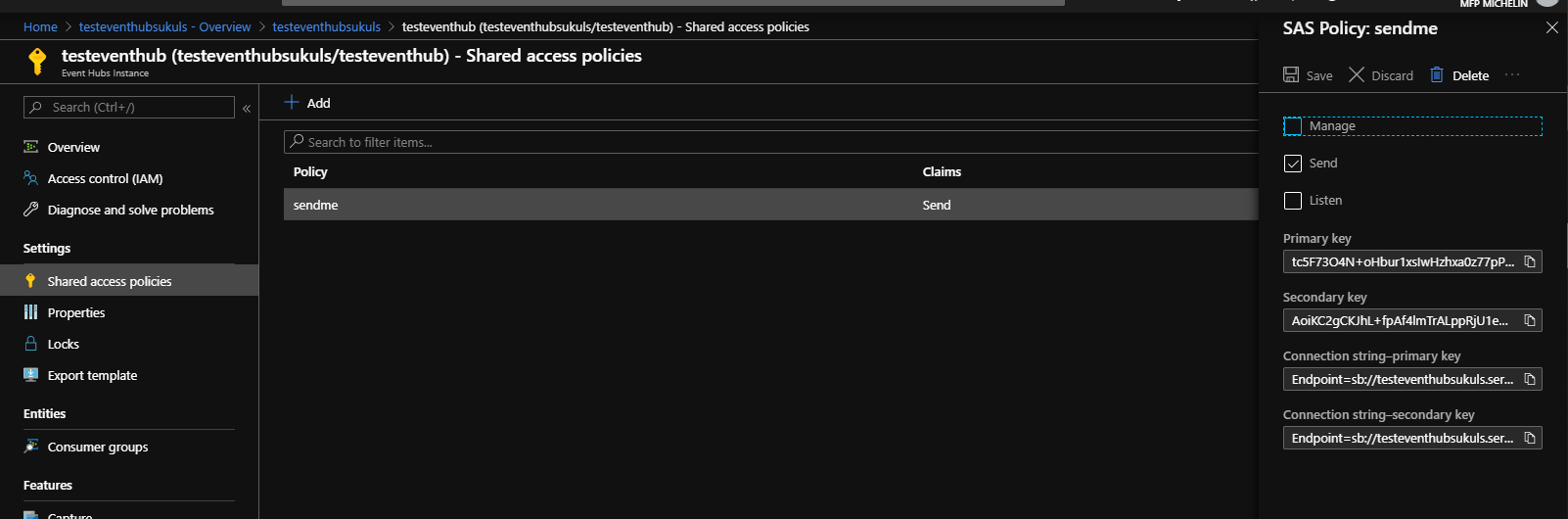
Once inside the Event Hub blade, then we can get the different metrics of the performance of this particular Event Hub. At this point, we're not using it, so everything is zeroed out.



Now to be able to send data into the Event Hub, we have to set the proper policy and get our credentials. This is done by going here into settings where it says Shared access policies. As we can see, we don't have any set up yet. So we're going to go click on Add, and then we're going to give it a name. In this case, I'm just going to call it Sender. And I'm going to allow these to send and click on Create.



Now that we have the policy, we can click on it. And here's where we get the information that we're going to have to use in our Visual Studio solution on our next demo. As we can see, we can have a primary and secondary key as part of the policy, and we have a connection string with both keys. This connection string is what we're going to be using in our. NET application. At this point, we have our namespace up and running, our Event Hub up and running, and now our Sender policy up and running. We are ready to start sending that machine temperature data.



## Starting the Course Solution

Now that we have an Event Hub namespace and an Event Hub created, let's start building our solution. We're going to be doing machinery temperature monitoring. We have a warehouse connected to Azure. And we're going to stream the machines temperature readings multiple times per second to Azure. For a solution, we're going to have six machines pushing their temperature readings twice per second to Azure. In terms of the implementation, what we have is a Windows 10 virtual machine where I'm going to be running a Visual Studio solution that is a C# program to simulate the multiple machines and how they change their temperature readings as we provide different commands to move the temperature up, down, stable, or bouncing. The code that we are going to use has dependencies in two packages. WindowAzure. ServiceBus is the package that contains the client for Event Hubs, and Newtonsoft. Json, which is the very popular. NET library for working with JSON.

## Demo: Pushing Data to Event Hubs

Let's check out the demo of pushing the machine temperature data to Event Hubs. I have the solution open now in Visual Studio 2017. I'm using community edition. You will be able to find the solution as part of the download package for this module. First, let's focus here on the components of the solution. You will see there are two projects inside the solution. One is called EventHubDemo, and the other one is called FactoryFloor. FactoryFloor is simply a C# library that contains the definitions for the machine class. It does not have any information or code related to Event Hubs. EventHubDemo is the one that generates the executable file that will send the messages, and it is the one that contains the Event Hub client. Now, to be able to configure the solution properly, you should go into EventHubDemo, then go to Properties, and then go here on Settings. There are two settings that you need to set, first, the EventHubConnectionString and then the EventDelayMs. Because we are writing twice per second, I have set the EventDelayMs into 500. Now for the EventHubConnectionString, we need to get that from the Azure portal. So we're going to go into the Azure portal, and I am located here in the policy blade of my factoryfloortemp Event Hub. I have the connection string here. I'm just going to click to copy. And once it's copied, I'm going to go back into Visual Studio and paste it in. Now I have both settings properly configured, and I can save this. Now, let's close the Solution Explorer and focus on the code for the Event Hub client. This client is very simple, and I hope it really showcases how easy it is to get up and running with writing data to the Event Hubs. It has only three different class members. One is the string that has the connectionString. The other one is the EventHubClient that will come inside the WindowsAzure. ServiceBus namespace. And then, finally, just a Boolean value to stop writing the Event Hub data. To construct the Event Hub, we simply get the connectionString value from the Settings file, and then we instantiate a new client with that connectionString. And that's it. Then we have a method called ObjectCollectionInfiniteWriteWithDelay because basically what we're going to do is an infinite loop where we write the values of the different machines that we create, and we just wait a set amount of time. We can see there that I have a while(! stop) loop, and we do two async calls, one that says WriteEventFromObjectCollection because we are passing in multiple machines, and then we just make the thread wait depending on what we set in the delayMs setting. For the next method, it's called WriteEventFromObjectCollection, and we simply pass the collection of objects that we want to write their data into Event Hub. We can see here I just create a list of tasks, and then for each item in the collection, I just do the task of writing that specific object. And then we await that async method until all these tasks are done. Now, finally, for the actual part that sends the data into Event Hub, it is very simple. We just pass in the individual object like we did in the last method, and we do an async call through the Event Hub client. SendAsync method. We just do a new EventData. We set the encoding to be UTF-8 and GetBytes. And what we're using is the ToString method in the object that we are passing in. As part of the machines class definition, that ToString object has been redefined to send the machine data as JSON. At this point, we can start our solution and begin pushing data into Event Hubs. I'm just going to click on Start. We can see the solution here has started, and we are going to use six machines as part of our demo. So I'm just going to type in 6. Now we have the machines going. And now we can use different commands. First, if I just type status, we're going to be able to see the temperatures that the machines have right now and the directions that they're moving in. So we can see m1 through 6, they have all started with the same temperature, and the direction of the temperature right now is stable. We can manipulate each machine individually, and this is going to become very handy later on once we are building the real-time visualizations. So, for example, if I wanted, say, m1 to increase, I can type that in. And if I go into the status now, we can see that m1 is increasing now to 66 degrees, and the direction says that it's increasing. For example, if I wanted to say m4, I want it to bounce, I can just say m4 bounce, and then we can check the status again. And we can see that m4 is now not 63, but it actually changed to 61 and the direction of the temperature is bouncing. So that means it's going to go up, it's going to go down randomly as it goes through sending the data to the Event Hub. Now if I go back into the Azure portal, I can also go back and check and see how my Event Hub is doing. You can go back into the overview and then come back into the metrics. And I should be able to see immediately that my requests are starting to go up because I have my program up and running. As time goes on, the messages are going to start to increase as well, and the throughput that is being recorded on the portal.

## Module Summary

In summary, in this module we saw that Event Hubs is a scalable, cost-effective data ingestion service, and it's very useful as a landing pad to Azure. We learned that Event Hubs allocate their capacity through a concept known as throughput units. They are assigned to a namespace. And under that namespace, you create the Event Hubs. Event Hubs provides client libraries for. NET, Java, Node. js, and C. In our case, we're using the. NET library. And, finally, at this point in our solution, we have machines simulated by the C# solution that are successfully pushing data to our Event Hub. Join me on the next module as we are going to take that Event Hub data, and we are going to be analyzing it on the fly with Stream Analytics.

Very Imp: <https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-samples>

Sample code available here.

Event hub has 2 options : Version 1 and Version 5

With python , we have two options : Sync and async

|  |
| --- |
| """  Example to show streaming sending events with different options to an Event Hub.  """  # pylint: disable=C0111  import time  import os  from azure.eventhub import EventHubProducerClient, EventData  CONNECTION\_STR = "Endpoint=sb://testeventhubsukuls.servicebus.windows.net/;SharedAccessKeyName=sendme;SharedAccessKey=tc5F73O4N+oHbur1xsIwHzhxa0z77pPNYv1IbuoMa+E=;EntityPath=testeventhub"  EVENTHUB\_NAME = "testeventhub"  start\_time = time.time()  producer = EventHubProducerClient.from\_connection\_string(  conn\_str=CONNECTION\_STR,  eventhub\_name=EVENTHUB\_NAME  )  to\_send\_message\_cnt = 500  bytes\_per\_message = 256  with producer:  event\_data\_batch = producer.create\_batch()  for i in range(to\_send\_message\_cnt):  event\_data = EventData( int(random() \* 100) \* bytes\_per\_message)  try:  event\_data\_batch.add(event\_data)  except ValueError:  producer.send\_batch(event\_data\_batch)  event\_data\_batch = producer.create\_batch()  event\_data\_batch.add(event\_data)  if len(event\_data\_batch) > 0:  producer.send\_batch(event\_data\_batch)  print("Send messages in {} seconds.".format(time.time() - start\_time)) |

# Analyzing Data On-the-fly with Stream Analytics

## Stream Analytics Fundamentals

Hi! This is Warner Chaves with Pluralsight. And welcome to the next module in our course for Delivering Real-time Data with Azure and Power BI Streaming. This is Analyzing Data on-the-fly with Stream Analytics. In this module, we're going to go over the basic concepts of Stream Analytics. We're going to go over the process of creating a Stream Analytics job through the Azure portal. Then we're going to look into the basics of creating Stream Analytics queries and testing them on the portal as well. Then, finally, we're going to continue building our solution as we process machine temperature coming in from the Event Hubs through Stream Analytics.

In this module, we're going to focus on the second component in our Azure architecture, that is Azure Stream Analytics. And, remember, this is a Platform as a Service offering for analyzing your data on the fly. Here is the architecture that we are going to be using as a reminder. We have the devices that are sending information through the Event Hub. Then it goes into Stream Analytics to be aggregated and analyzed. Then it's going to be ingested into Power BI, and then, finally, visualized by our end users. In this module, we are going to be focusing on this second step, the one that feeds the Event Hub data into Stream Analytics and then aggregates the data that we're going to push further into Power BI.

As you work through the course solution, and as you encounter real-life situations, you might find yourself when you don't always need a Stream Analytics job, and you can just pass in the streaming data directly to Power BI. So when do you actually need to go through the steps of working with Stream Analytics? Well, Stream Analytics has some good use cases when you're dealing with real-time data.

* First, if you need to do aggregation, for example, if you're receiving too many data points that you don't all want to push further into the visualization layer.
* Also, it is easy to transform the data if you want to do on-the-fly transforming before sending it to the data consumers, then Stream Analytics is also a very good choice.
* If you want to add reference data, then Stream Analytics is a good place to add it. And we're going to go further into what reference data is and how to do it further on in this module.
* And, finally, if you need to analyze the data by either using some JavaScript functions or some actual Azure machine learning, then Stream Analytics is a great place to do that while working with your data on the fly.

Once we have determined that we have a proper use case for Stream Analytics, how do we get started with it? The basic concept of Stream Analytics is the Stream analytics job. In order to create one, we need to pick a hosting environment. This is usually either the cloud or what is known as the Azure IoT gateway. For the scope of this course, we're just going to be hosting our Stream Analytics job in the cloud. Then we have to pick a region. Usually you want to pick a region that has very low latency between the source of the data and your Stream Analytics job. And then, finally, we have to pick streaming units. And we are going to go into detail as to what those are. Streaming units represent the computing resources of a job. Once again, it's very common in Azure to not have the actual hardware exposed to the past services, but instead use this type of units concept, same as how we had on the Event Hubs. Streaming units are a blended measure of CPU, memory, read and write rates. The number of events that you are pushing through a Stream Analytics job, the complexity of the query that you're building, or if you're using functions or actually doing callouts to machine learning all impact the amount of streaming units that you are going to consume. The nice thing about this is that it's very easy to monitor the usage of the streaming units, and it's also very easy to scale them up and scale them down. Flexibility once again is one of the main drivers to use the cloud. One big advantage of Stream Analytics is that unlike an Event Hub namespace, when the job is not running or you are not using it, you can stop it, and you're not going to be billed while the job is in the stopped status. Let's look at the components of a Stream Analytics job. First, you have to have a query that is going to be what tells Stream Analytics to do with the data. Then you're going to have some inputs that says the data is coming in to be analyzed by the Stream Analytics query. We can have here Event Hub data, for example, or let's say blob storage data as well. We also have another type of data called reference data. And we're going to look into that further down in this module. We also can call JavaScript functions as part of our Stream Analytics query. We can also do callouts to Azure machine learning APIs as part of our Stream Analytics query. Once we have the proper output that we want, then we can send it out into Power BI or, for example, Azure SQL database. Once in the services, these outputs can be manipulated by the end users, or they can be further down consumed by other services. There are three data formats that Stream Analytics understands as part of its querying engine, that is JSON, Avro, and CSV. In the case of our course solution, we are sending the machine temperature data as JSON. Let's check out a demo of creating a Stream

Demo: Creating a Stream Analytics Job

Analytics job through the Azure portal. Okay, I'm connected to the Azure portal, and we're going to do a search here for Stream Analytics. And this way, we can get our link here to the Stream Analytics jobs blade. I'm going to click on there. And I already have a Stream Analytics job created. However, I'm going to create a brand-new one so you can see the whole process end to end. So I'm going to click here on Add. Now we have to give the job a name. So I'm going to just call it FactoryTemperaturesCourse. And at this point, I'm going to use the same resource group that we used for the Event Hubs. This way, like I said before, all the resources in the course are in the same resource group. And I'm going to leave it in the East US location, and we can see here at the bottom is where I can set the streaming units. I can go all the way up to 120 streaming units. However, for the course purposes, it should be enough to just have 1 streaming unit for the job. Finally, we have here the option of the hosting environment, either the cloud or the IoT edge gateway. Like I mentioned for the purposes of the course, we are going to be hosting a Stream Analytics engine in the cloud. At this point, we can just click here on Create. After a few minutes, we can already see the results, and we have our streaming jobs created. I'm going to go ahead and refresh. We can see now I have the two different Stream Analytics jobs. So I'm going to go into it, FactoryTemperaturesCourse. And at this point we can see we have a Stream Analytics job, but it has no input, it has no output, and it just has an empty template query. So it's no good obviously for our analysis. So step number one, we are going to define the input. In this case, it's the Event Hub data that is coming from the machines. So we'll just click here on Inputs. And you can see here at the top, it says Add stream input or Add reference input. At this point what we want is to add a stream input. And out of the options that we have, obviously what we want is Event Hub. So we are going to call it FactoryEventHub. And here the portal has some nice integration, so we can actually select the Event Hub that we want to work with right here from this blade. So if we scroll down here, we can see it has already detected that we have our coursedemo, Event Hub namespace, and it has already detected also that we have this Event Hub called factoryfloortemp. Now here we can see where it says Event Hub policy name. This is the policy the Stream Analytics is going to use to connect to the Event Hub. Now if I open on the available policies, we have the RootManageSharedAccessKey policy, which is the administrator policy for our Event Hub, and the Sender policy for the Event Hub, which is the one that is being used by our. NET application to send the machine temperature. However, neither one of those is appropriate because Stream Analytics only wants to read from the Event Hub. So we are going to add that policy right now. I already have the Event Hub blade open here on the other tab. Once on this page, I scroll here to the bottom so we find our actual Event Hub, the factoryfloortemp one. Once we are on the blade for the specific Event Hub, we're going to click again on Shared access policies. And now there we can see we have the Sender policy, which has the permission to send. That is not what we want for our Stream Analytics job. So we're going to add a new shared access policy. I'm just simply going to call it Listener, and we're going to give it the permission to listen to our Event Hub. And just click on Create. Now we have it successfully created, and we can go back into the other blade where we have the other policies. Now if you open this right now, you will see that it is not populated. So we are going to have to refresh this to be able to get it added. So we'll just close this again. We're going to add a streaming input again. Here we're going to give it the same alias. All the other options remain the same except now we have the Listener policy. So we select the Listener policy. And here we also have to tell Stream Analytics the format of our data. If I open this up like we mention in the slides, it's JSON, Avro, or CSV for the case of our Event Hub, we are sending data for JSON. The encoding we can leave the same. And we are not using any compression. So we can go ahead and save this now. The input gets added to the job, and it also gets tested. And we can see the message where it says that the connection succeeded, and the Stream Analytics job can read from the Event Hub. Now, at this point, we can actually go here to the query, and, for example, we could do a simple SELECT \* FROM FactoryEventHub. And we can click here on test. Now to test, we actually have to give it some sample data. We can either give it the sample data from a file, or we can actually get it directly from the hub itself. So we're going to click OK here. Then we're going to click here where it says FactoryEventHub, and we're going to say Sample data from input. And we're just going to tell it to start right now. Now we get this message here that says there were no events found for FactoryEventHub, and that's because we don't have our application right now sending messages over to the Event Hub. So I'm going to minimize the browser, and I'm going to start our. NET application. I'm going to set six machines. And it's going to start. And I'm going to say status, and we can see the machines right now are sending data. So since we're sending the data now into the Event Hub, we are going to go back, and we're going to try to sample the input again. I'm going to click again Sample data from input. And we're going to see again here where it says the time and the amount of time that is going to read off of the input to be able to do some sampling. So we can leave the defaults again, and I'll just click on OK. And this time, Azure is going to be reading the data that we are sending out actively through our. NET application. We can see here that the sampling has succeeded because now we do have data coming into the Event Hub. And we can just test it here by running the SELECT \* FROM the FactoryEventHub. If we click here on Test now, and we can see that Azure Stream Analytics is understanding properly the data. And we have the temperature, we have the measurement, we have the direction of the temperature, and the machine names, and so on. So successfully we have connected our Event Hub to Stream Analytics, and Stream Analytics is able to sample data from it and is also able to query it now. So we are ready to dive deeper.

Stream Analytics Queries

Now that we have a Stream Analytics job created, and we just tested it with a very simple query, how do we go further querying the stream of data? Well, Stream Analytics has SQL-style queries with optional JavaScript functions. So these are two very popular languages that most developers are going to be familiar with. The streaming data is very easy to analyze and manipulate on Stream Analytics because it's based on time windows. You can tell it if you have an overlapping or non-overlapping window and how long the window of time you want it to be where you want to group the data to analyze through your on-the-fly query. Finally, all this time-based manipulation is easy to do because the Stream Analytics SQL has time-aware SQL constructs, and we're going to look into the basics of those next. At the top, we have a very simple SELECT query. In the FROM clause, we can see where it says FactoryEventHub TIMESTAMP BY temperatureReadTimestamp. The TIMESTAMP BY defines the application timestamp field, and further time logic done by Stream Analytics will be done based on that field. Here we have another example where we're actually doing some aggregations. We can see here at the bottom where we do a GROUP BY TUMBLINGWINDOW, and we give it a time value, in this case as s and 1, which means 1 second. This TUMBLINGWINDOW defines a non-overlapping window of time in the stream. That means while the Stream Analytics job is running, it is going to create windows of 1 second, analyze that data, and output that as a result. Stream Analytics also allows you to do joins on your queries. You can do joins of multiple inputs, or you can do joins of subqueries defined by CTEs, like in this example, to the main data coming in from the stream. When you do a join, you do have to specify a DATEDIFF condition. This is because Stream Analytics could potentially be looking at an endless stream of data, so we have to bound the join by a certain amount of time. In this particular case, we are evaluating the join on data that is separated between 0 and 1 seconds coming from FactoryEventHub and the LastInWindow CTE. Here we're only covering the basics for Stream Analytics so you get a good understanding of the service, and you can continue building the course solution. However, if you want to do a deeper dive into Stream Analytics, Pluralsight has you covered, and we have a course called Understanding Azure Stream Analytics by Alan Smith.

Demo: Querying a Stream

Now that we have created the Stream Analytics job, and we have some queries, let's continue building our solution. In our solution, we are going to stream the machine temperature readings multiple times per second to Azure, and we are going this already through the Event Hubs as we built it in the last module. Then what we want to do now in the Stream Analytics job is to aggregate the temperature readings, and we want to add reference information to it. As part of the aggregations, we're going to have the average temperature, maximum temperature, minimum temperature, the standard deviation of the temperature, and the last temperature direction in each time window. Let's check out a demo of querying Event Hub data with Stream Analytics. We're going to test some different queries over our streaming data. And, remember, you just have to go into the Stream Analytics jobs blade, and here on the left panel the option that says Query, like we did at the end of the previous demo. So now I have here this blank, and I'm going to be using some sample queries. This file, StreamAnalytics - sample queries, is going to be included in the download package for this module. Now the first query that I have here on the file is a Simple Select statement. I'm not going to rerun it. We did that in the last demo at the end. So let's check out Query #2. This is an aggregation over a time window. So I'm just going to copy and paste this. And I'm going to give it a little bit more room so we can see it. So here you can see at the very top, we're just doing multiple aggregations over the data that is coming in the stream. Remember, we have six machines, each one sending two data points per second. So just by aggregating over 1 second, we are cutting the amount of data in half. Here we can see the different aggregations that are available in Stream Analytics. For example, we checked the max(temperatureReadTimestamp) to see where this particular window ends. We see the average, standard, max, and minimum of the temperature, and the machineName of course so we know what machine is sending this specific data. Here we just do a very simple FROM FactoryEventHub, which is the name of the table. And look here where we have this TIMESTAMP BY construct so that Stream Analytics knows that our machines are sending the time values on this field called temperatureReadTimestamp. We are finally setting a GROUP BY TUMBLINGWINDOW of 1 second. So every data is being aggregated on windows of 1 second. So we're going to go here, just click on Test. And we're going to be able to see the results. So if I scroll down here, I'm going to open this column so you can see the time, we can see where we have the values for the different machines, and we are going to be seeing time windows of 1 second. For example, these top records are at the 39th second. And then we can see the next six records are at the 40th second. And then we can see, for example, the next six records again at the 41st second. So Stream Analytics is splitting our Event Hub data into those 1-second windows and then aggregating it GROUP BY the machine name. So we're going to go back, and we're going to check out another sample query. In this case, it's using a CTE. If you're familiar with SQL Server anti SQL, you're going to be very comfortable here working with Stream Analytics. It's exactly the same concept. So I'm going to scroll here at the top, and we can see I am defining a CTE called LastInWindow. And inside that CTE, I am grouping by the same TumblingWindow, and I'm using a function called TopOne over the temperaturereadtimestamp because I want to find the last event for each machine inside that particular window. And the reason for that is because I want to be able to say, What was the last direction that the temperature was moving in at the end of the time window? So we can see here if I go at the bottom on the SELECT statement, I'm selecting the machineName, the currentDirection, and the timestamp of that particular topEvent CTE. I'm just going to go ahead and test this as well. And I'm going to open up the results. And if I scroll down here, we can see now how for each one of those machines under the same time window, we have a current direction value taken from that TopOne function. And in this case, they're all saying Stable because that is the data that we are feeding it through the Event Hub application. Same thing, we can see that we are moving here in windows of 1 second. Perfect! So I'm going to open this up again, and we're going to see the last sample query. And this last sample query for this demo basically joins both the CTE concept and the fact that we have to do joins using DATEDIFF conditions. So I'm going to copy and paste this query. This one's a little bit bigger. And we're going to override this one here. So let's look at this step by step. So, first, I'm using the same LastInWindow CTE that we use in the previous demo to get the last event of the time window. Second, I am defining another CTE. This one is very similar to the first query we did in this demo and basically computes all the aggregations coming in from the stream. We can see here the max, average, same thing, all coming in to this GROUP BY TUMBLINGWINDOW. And then, finally, once we have those aggregates and the LastInWindow event, we are going to select fields from both of them, and at the end, we are going to be doing the join. Here we do the DATEDIFF join in a unit of seconds between aggregates and LastInWindow and BETWEEN 0 AND 1, so we want only to be looking at a window of 1 second to be doing these joins. And at this point, again, we can come here and test the output. We'll go back and open the results. And we can see here we have not only the aggregated data, but we also have the last direction column, which is the one that comes from the TopOne function and the first CTE, so they have been successfully joined to create this new output.

Demo: Adding Reference Data

As mentioned before in this module, Stream Analytics supports the concept of reference data. This is data that enriches the incoming stream, but it's not part of it. It's coming from an outside source. This data must be a file or a set of files in blob storage. That is the service supported by Azure Stream Analytics for the reference data. This feature is very useful when the data origin is not aware of this reference data, and it has to be added through an intermediate step. To use it, you simply set it up inside your input, and you refer to it as another table in the Stream Analytics SQL. In the case of our course solution, we are going to augment the stream by adding information about which energy panel is connected to each machine. Potentially the machines would not be aware of this information. However, we are because we know how the warehouse is set up, and this is a perfect case of data that needs to be injected in an intermediate step. Let's check out a demo of adding reference data with Stream Analytics. So here is the reference data that I'm going to be using for the demo. The file is called machine\_energypanel. json, and it is going to be included as part of your download package. As you can see here, it's a very simple JSON file. It just has six records that correspond to each one of the machines and the energyPanel that they are connected to. So machines m1 to m3 are connected to the North panel, and machines m4 to m6 are connected to the South panel. So that's the actual reference data file. I'm going to close it down, and we're going to go back into the Azure portal. Now if you guys remember from the slides, the reference data has to be blob storage to be accessible for the Stream Analytics job. So I have here the storage account called warnerdata. And if I go here to Containers and see I already have a container called referencedata. And I have already uploaded the machine\_energypanel. json file. So as part of your setup for the course to follow along, make sure you create a storage account, a container called referencedata, and upload this same file to this container. So, once we have our reference data in place, I'm going to go back into the Stream Analytics job blade. And here we are going to go where it says Job Topology here on the side, and where it says Inputs. Once we are on the Inputs blade, we can see here at the top where we have that button that says Add reference input. So I'm going to click here, and we can add it from blob storage. Now I'm going to give it a name. In this case, I'm going to call it MachineEnergyPanel. Once I have the name here, I have to locate it in my storage accounts. So I'm just going to select it here, go into warnerdata, and here again pick reference data. And then just set the path, in this case it's machine\_energypanel. json. If we keep scrolling down, I am not going to have multiple files with different data formats, so I'm not going to be changing that. And the event format is already set in JSON. And then I just have to click on Save. Now we can see here our reference input has been added, and it has been successfully tested to connect to it from Stream Analytics. So now at this point, let's run a test query to make sure that it is working. So if I go here to Query, we can see here now I have both inputs, the FactoryEventHub, as well as the MachineEnergyPanel reference data. And I'm going to go back into the sample queries, and I'm going to select the Query #5. I'm just going to copy it and paste it into the query window. Now before I can run this test query, I need to do sampling for both inputs as well like we did before. So here I'm just going to go to FactoryEventHub and sample data from the input itself. Instead of 3 minutes, I'm just going to sample 30 seconds so it loads faster, and say OK. And we're going to wait for the sampling to be done for FactoryEventHub. In the meantime, I'm going to say that I want to sample data from a file for the reference data. Just click here. And I'm going to upload it from my machine. So I have here my JSON file. And just click on OK as well. Once both inputs have been sampled, then we can run the test query. Here we can see both inputs have been sampled now. So going back into our query, we can see pretty much the only difference here is a couple of things. First at line 34 at the bottom, we see INNER JOIN MachineEnergyPanel, which is the alias for that reference data, ON Aggregates, which is one of the CTEs in the query, on the machineName column. So we're just joining on that machineName so that we can get the reference data for the energy panel for each machine. And then in the SELECT list, we can see the last column that is selected is simply MachineEnergyPanel. energyPanel. So we are augmenting that streaming data with the information of the energy panels. I'll just click on Test, and we're going to see the results that we get here in the Azure portal. As we can see here in the results, if we scroll down, we can see machines m6, m4, m5 all have the energy panel now set to the South panel. Machines m1, m3, m2 here in the sample data as well all have it set correctly to the North energy panel as expected. This is the data that we are going to be pushing now into Power BI on the next module.

Module Summary

In this module, we learned that Stream Analytics allows for augmenting, aggregating, and analyzing data on the fly all while taking advantage of the power and flexibility of the cloud. Stream Analytics uses the concept of streaming units to represent the compute resources for the job. And you can scale them up or down as needed. Jobs are defined in a SQL query language, which makes it very easy to get up and running if you are a SQL developer. And, finally, at this point in our solution, we have the Stream Analytics query that we will use to push data forward to Power BI as we go into the next module. Join me on the next module as we will start to ingest streaming data into Power BI.

Ingesting Streaming Data into Power BI

Why Use Power BI?

Hi! This is Warner Chaves at Pluralsight. Welcome to the next module in our course for Delivering Real-time Data with Azure and Power BI. This is Ingesting Streaming Data into Power BI. In this module, we are going to go over the basics of Power BI streaming capabilities. We are going to look at the different types of datasets that can support real-time functionality. We are also going to look at how you can create a real-time dataset manually on PowerBI. com. Then we're going to look at the integration built in to the Azure portal that allows you to push Stream Analytics data directly to Power BI. In this module, we are going to be focusing on that third component in our real-time data architecture. Now we are moving into the visualization layer, and this is Power BI. And, remember, this is Microsoft's web service for rich, self-service BI reports and dashboards. This is the architecture that we've been working with throughout the course. We have the devices generating data to Event Hubs analyzed through Stream Analytics, and to be visualized in Power BI. In this particular module, we are going to be focusing on the step that brings the data from Stream Analytics into Power BI. So considering you're building a solution that is going to do real-time data visualization, why should you select Power BI as your tool of choice? There are several reasons why Power BI is a great choice if you are building your real-time solution with Azure. First of all is because there is really tight integration. Because both Power BI and Azure are controlled by Microsoft, they make it really easy for you to use one with the other. Second, Power BI has really good native real-time data support. As we are going to be looking at in these last few modules of the course, it is really easy to pick up real-time data, put it into Power BI, and then visualize it with all types of graphic capabilities. And, finally, the visualization experience is very rich into Power BI. It supports a very easy-to-use drag-and-drop interface, and you can achieve powerful results in a short amount of time. In the course, we are not going to look into Power BI fundamentals, and we are only going to look at enabling the real-time visualizations. For more Power BI learning material, please refer to the Pluralsight course library where you can find courses for all the skill levels.

Power BI Real-time Datasets

So we have decided that we are going to use Power BI as part of our architecture. So how do we get started with real-time data? First, Power BI has several different types of datasets that support real-time data consumptions. We have a streaming dataset, which is extremely fast. And we also have a push dataset, which is also fast but is not as fast as the streaming dataset. However, as is usually the case with technology, one has some trade-offs over the other. A streaming dataset supports very fast dashboard tiles. However, that is where it's limited. You can only place them in dashboards, not inside reports. It has limited graphic options because these are built-in streaming tiles created by Microsoft. Under the covers, what streaming dataset does is that it uses a Redis cache. This is a common in-memory cache used for many different types of applications. Because of its very fast in-memory nature, Power BI only keeps approximately 1 hour of data inside the streaming dataset. A push dataset, on the other hand, supports regular report creation. It is not just limited to adding graphics on the dashboard. All graphics are supported as well. You are not just limited by the few graphics that Microsoft has created as streaming tiles. You can add all sorts of graphics including custom visualizations, which we will do in the next module. The downside of the push dataset is that the refresh rate is slower. It's about 5 seconds or more compared to the streaming dataset where it's usually at 1 second or faster refresh. Under the covers, a push dataset uses an Azure SQL database. That's why it is fast but is not as fast as the streaming dataset. And, finally, this Azure SQL database keeps a revolving history of 200, 000 records. Power BI also makes it really easy to pretty much just get the best of both worlds. You can create a dataset called a push-streaming dataset, which will have the Redis cache under the covers for the streaming dataset, which means you'll be able to use streaming tiles, and it will also have the push dataset that you can use for reports, custom visuals, and uses the Azure SQL database under the covers. So how do you select between the two modes--push or streaming? Well, you can create either one through the REST API. However, you can also create both with the push-streaming dataset. So you don't have to choose one or the other necessarily. You can always just go with the push-streaming dataset and have both at the same time. Stream Analytics, the integration that it has built in to the Azure portal, actually creates a push-streaming first-in/first-out dataset. That means that once the push dataset has reached the size of 200, 000 records, it starts to get rid of the old ones. And, finally, my recommendation is to either do a push-streaming or a push dataset. If you are not going to use streaming tiles at all, you can just do a push dataset. Otherwise, just doing streaming is really not good because of the limited graphics options.

Demo: Creating a Real-time Dataset on PowerBI.com

The architecture presented in the course is just an example that really shows how to do large volume analytics data real-time into Power BI. However, if you have a low amount of data, and you don't need Event Hubs or Stream Analytics, you can actually send your data directly to Power BI endpoints through the Power BI REST API. Keep in mind when you create a Power BI real-time dataset, it is supported through the REST API or the website, but there is no interface to do it with the full application of Power BI desktop. Let's check out a demo of creating a real-time dataset through the Power BI website. I am logged in right now to powerbi. com. Remember, you don't need a paid subscription to follow along with the course. A free powerbi. com account works just fine. So here we are in the home page, which is usually the recently viewed page. Now, to find where to create the real-time dataset, go here to where it says My Workspace. Now when we are in the My Workspace view, you will see this button in the top right-hand corner that says Create. So we're going to go in and click on this, and we see the different options. And at the bottom, you can see where it says streaming dataset. So I'm going to click there. And we have three options. We have the option of creating an API streaming dataset directly to powerbi. com. We have the option of Azure Stream Analytics integration, which we will do in another demo. And we also get this option of PubNub. PubNub is a third-party service provider for real-time data, and it is really outside of the scope of this course. In this case, we are going to be doing the manual API creation of the streaming dataset. So it's already selected, and I just need to click on Next. Once here, we're just going to give it a name. So I'm going to call it RealtimeAPI. And then because we are creating it manually through powerbi. com, we need to name the columns here on the interface and their data type. So if we wanted to do something similar to what we have on Stream Analytics, let's say we could have machineName and the type would be Text, and then we would have something like averageTemperature and the type would not be a Text here. We can see the different types. It's either a Text, a Number, or a DateTime. So let's say we have a number, and then we would have something else like the temperatureTimestamp. And this one would be a DateTime. And we will not define any other values. Now you can see here at the bottom where it gives you a preview of what your JSON data should look like to push into this streaming dataset. So we have the machineName, the averageTemperature, and the temperatureTimestamp. Now, if you see here we also have this little radio button that says Historic data analysis, Off or On. What happens here is if you leave it as Off, then it will be created as a streaming dataset. Remember, this is the really fast Redis cache-based dataset that only keeps one hour of data. However, if we click here and turn this Historic data analysis on, it will actually create a push streaming dataset. So we'll have both the in-memory Redis cache that lasts for one hour, and also we will have the regular push dataset that has 200, 000 records that you can continuously keep pushing data into. At this point, this looks good, and we can click on Create. Now as you can see here, Power BI now gives us a push URL. This is an HTTPS endpoint where you can push your streaming data directly. It gives us here the format. If you're going to use cURL, you could just do a Post request using this format that it shows you here on the website, or if you're going to use PowerShell, you can also use this small snippet here as a sample of how you could push data into the service. We will just define the endpoint through that push URL. Then our payload, which is the JSON file with the information that we want to move, and then, finally, we will just be invoking a Post RestMethod. Now, because you are skipping Event Hubs and Stream Analytics, you would have to write some sort of application to write directly into these Power BI datasets. If we click on Done, we can see it works. And if we go to Datasets, we can see now we have our RealtimeAPI here.

Demo: Creating a Real-time Dataset with Stream Analytics

Now that we have covered the fundamentals of Power BI real-time datasets, let's continue building our course solution. Remember, we are doing machine temperature monitoring. We have aggregated the temperature readings, and we have added reference information to them through Stream Analytics. Now we want users to visualize and receive alerts on this data in real-time. Now in this particular module, we are not going to go into the visualization part yet. We are only going to focus on the ingestion of the data into Power BI. To make it really easy to work with Stream Analytics and Power BI, the Azure portal already has a built-in integration. You just have to have a Stream Analytics job, and it'll automatically push the data into Power BI. This particular integration will create a push-streaming dataset. So you will have that in-memory Redis cache for streaming tiles, and you will also have the longer history of 200, 000 records on an under-the-covers Azure SQL database. Let's check out a demo of creating a streaming dataset through the Stream Analytics portal. I'm back in the Azure portal here. We are on the Query option of the FactoryTemperaturesCourse Stream Analytics job. This is the Query #5 that we were working on in the previous module. If you haven't done so already, make sure that you save this query here in this blade. Now after you have saved the query, we are going to go into the outputs here on the left-hand panel and click here on Outputs. Now, once we are in the Outputs page, we are going to click here on Add. And here at the bottom, we have the option of Power BI. So we're going to click there. Then we just need to give an alias to the output, easy to just pick PowerBI. We have to give it the option there to authorize connection to load workspaces so we can actually get that information from Power BI. Then we're going to give it a name for the dataset. I'm just going to give it a name here, RealTimeDataset. And the table name that is going to be used inside that dataset for this particular output. I'm just going to call it RealtimeTable. Once we have that metadata filled in, we need to authorize here that connection on the Azure portal so that it can enable moving the data from the Stream Analytics job into the Power BI dataset. Just click here on Authorize. You're going to have to log in with your Power BI account. I already have here my credentials filled in. I'll just click on Sign in. And as you can see here, it will actually tell you if you're authorized or not right away. That is the last step, and you can just click here on Save. At this point, it will add the output to the Stream Analytics job, and it will also test the output by trying out your credentials and giving you a message if it succeeds to connect into Power BI. At this point, we can see it is successful. And now the last step here is to actually start our Stream Analytics job. So we will go back here to Overview. Now before I actually start the job, remember you need to have your Event Hubs populating with data. So I'm going to minimize here, and I'm going to run the Event Hub Demo application to start pushing data into our Event Hub. So I'm just going to use the same amount of machines, six machines, and then I'm just going to type here status so we verify all machines started with a temperature reading of 11 degrees. And they are pushing data now to Event Hubs. So I'm going to go back here into the Azure portal and just click on Start for the Stream Analytics job. Now we have an option here to start the job right now so it will start to read the Event Hub to the point in time that we are right now. You can actually select a custom time where you want it to start, or you can select when last stopped if you were previously working with the Stream Analytics job. Now for this case, I'm just going to click to start right now. I'm just going to click here on Start. And as you can see, the Azure portal will take the request and get the streaming job started. If you want to follow along to see what the streaming job is doing, you can also click here on the Activity log and see what are the latest operations that the Stream Analytics job is doing. And after a couple of minutes, we get the notification that the job has been successfully started. Now at this point, we can also switch into Power BI, and here on the Datasets page, I am just going to refresh it. And as we can see here at the bottom now, we have a new dataset called RealTimeDataset. We can click here on the API info, and we can verify we are capturing all the different columns from the Stream Analytics output. Now if we want to verify further that we are indeed getting data from Stream Analytics, we can go here to the RealTimeDataset on Create Report. Let's just click there. And we're going to insert a new table. I'm just going to select here the machinename, the windowend, and let's say the averagetemperature for example. Now the actual defaults here, the text looks really small, so I'm going to make it bigger. I'm just going to go here into the column headers, and I'm going to make those a little bit bigger. And I'm also going to go into the values, and I'm going to make those just as big as well. I'm going to go all the way up to 14. And I can open up the table here so we can see all the values. And as we can see here, if we look through the dataset, we can see that indeed Power BI is consuming the data from all the machines. And we are going to be able to build more visuals as we go along on the next module.

Demo: Modifying a Real-time Dataset

Now let's cover the scenario where after you have deployed your real-time dataset, you need to modify it. Where do you do this? You have two options. You can either modify it in Stream Analytics, or you can modify it in Power BI. If the modification is a new column calculation or if it needs to flow to multiple outputs, let's say you're not just outputting the Power BI but also to a service bus, then you're going to have to do that change in Stream Analytics. On the other hand, if the modification is to add a Power BI measure to the dataset, then you can do it through Power BI desktop. Keep in mind, you cannot add calculated columns through Power BI desktop, only measures. So if you do need a calculate column, you're going to have to add it on Stream Analytics. As part of our solution, we need to do one modification to our Stream Analytics job. We are going to add an extra column that will have the text version of the temperature read timestamp that the machines are sending. In the next module, we will see why this modification is necessary. For now, we're just going to focus on the mechanics of making changes. So let's check out a demo of modifying a real-time dataset. We are back into the Azure portal, and we are going to modify our real-time dataset by doing a modification to the Stream Analytics job. Now we have our job actively running right now. So as you can see here, we are in the Query page, and I actually cannot modify anything here on the query. That is because the job is running. So you want to go back into the Overview, and here we are going to stop our job. Once the job has actually stopped, we are going to be able to modify that query. So the job has stopped, and if we go back here now into Query, we can see now this query textbox is now editable. So I'm going to take this one, it's the one that we have before, and I'm going to delete it. And I'm going to replace it with this query. This is Query #6, and it's going to be in a file called StreamAnalytics - modified query that is going to be part of a download package for this module. And this is the final query that we are going to have as our last version here on Stream Analytics. I'm just going to copy/paste it right now. And I'm going to paste it here. Now, the only difference between Query #5 and Query #6 is this line here, line 21, that says I'm going to do a cast of the WindowEnd timestamp as an nvarchar(max), meaning I am converting it into text. Now because this is a calculated column, Power BI does not allow me to add calculated columns in a real-time dataset, so the only choice that I have is to add it here in the Stream Analytics job. Now you might be asking yourself, Well, why are we converting the temperature read timestamp into text? Well, the answer for that is going to be in the next module, and you'll see why we had to do this. For now, simply we replace the query, and then we can save it again. Let's click on Save. Let's click on Yes. We see here that the settings have been saved. We're just going to go back into the Overview and start the job again. I'm just going to start it right now. And now the streaming job is going to start again. And because we added an extra column as part of the SELECT, we have actually modified the output of our Stream Analytics job, and it's going to have an added column. And this change is going to trickle down into Power BI as well. Once the Stream Analytics job has started running again, I'm going to switch back now into Power BI, and I'm going to refresh here so we get the latest information refreshed into powerbi. com. And I'm going to go here on the API info for the real-time dataset. And it gives us, like I said, the raw view here of the JSON. Now if you guys look here at the very end of the JSON description, we can see how we have that new field, windowendtext. So powerbi. com has picked up the new field that is being pushed from Stream Analytics, and we can use it to start working with it.

Module Summary

In this module, we learned that Power BI supports self-service reporting with real-time data, and it's tightly integrated with Azure. There are streaming, push, and push-streaming datasets. And you need to understand the differences between them to be able to choose the correct one for your solution. You can also create a real-time dataset through the Power BI site like we showed in the demos or through the Azure portal Stream Analytics integration. And we also did a demo for that. At this point in our solution, as well, we have the final Stream Analytics job outputting data into our Power BI dataset. Join me on the final module in our course as we take that data that has moved through Event Hubs and Stream Analytics and is ingested into Power BI, and we finally build real-time visualizations with Power BI.

Building Real-time Visualizations with Power BI

Power BI Real-time Visuals

Hi! This is Warner Chaves with Pluralsight. And welcome to the final module in our real-time data delivery with Azure and Power BI course. This is Building Real-time Visualizations with Power BI. In this module, we are going to look into the details of working with Power BI real-time visuals. That means we're going to look into working with streaming tiles and dashboards. We're also going to look into using report visuals to do real-time visualizations. And, finally, we are going to look at how you can quickly answer and respond to changes in your data using Power BI alerts. So in this module, once again, we are going to be focused on Microsoft Power BI, except this time we're not just going to talk about ingesting data, we're also going to talk about visualizing it. So this is our diagram again for the architecture that we've been building throughout the course. We are now finally in the last step where we're going to take all that data that we've already processed, and we're going to finally create some real-time visualizations. Let's look at the different options that Power BI offers in terms of real-time visuals. We spoke before about the two different types of real-time datasets. First, we have the streaming dataset, and also we have the push dataset. And we mentioned how they can be combined into the push-streaming dataset. In the same way, Power BI offers two different types of real-time visuals. We have the streaming tiles on a dashboard, and we also have report visuals pinned to a dashboard. The streaming tiles are backed by the streaming dataset, and the report visuals are backed by the push dataset. Streaming tiles support very fast dashboard refresh. However, they have a big downside in that they offer limited graphics options. And in the demo, we're going to look at what those options are. These tiles are also not accessible from inside a report. They can only be placed and analyzed and looked at in a dashboard. On the other hand, we have report visuals, which can be tweaked to work and update in real-time. Any of the different graphics that Power BI already supplies can be used to create this type of visual. On top of that, you can even use custom visuals, and they will also refresh and change based on your data. The downside of report visuals for real-time is that the refresh rate is slower. It's about 5 seconds to 10 seconds depending on how fast your data is coming in. And also the report visuals in order to manifest all the update capabilities must be pinned to a dashboard. In the end, the best real-time dashboard that you can build will be a mix of both types of visuals because you're going to want to have some streaming tiles for some really fast refresh, and at the same time, you're going to want to have some real-time report visuals to provide a lot of richness and custom visuals.

Demo: Power BI Streaming Tiles

Let's continue building our solution. Remember, we are doing machinery temperature monitoring. And we have aggregated the temperature readings, and we added reference information to them. In the last module, we also landed the data on Power BI. Now we're at that final step where we want to build the visualizations. In this case, we are going to start building our machine temperature dashboard by adding a streaming tile that will show the real-time changes in machine temperature. So let's check out a demo of adding a streaming tile to a Power BI dashboard. I'm back into my Power BI workspace page, and the first thing we need to do obviously before we start building is to make sure that our data flow is happening. So I already have the Event Hub solution running as you guys can see here. I have six machines active as we've been doing throughout the course. Now if you forget about the commands that you can use here in the Event Hub demo app, simply type help, and you'll get a quick refresh. Now if I go to status, I can see here right now all of the six machines are in a temperature reading of 40 degrees, and they are stable at the moment. Now if I also go back into Azure, you can see I have opened my FactoryTemperaturesCourse blade, and at this point in time, the job is already running. So we have the data flowing and being analyzed on the fly with Stream Analytics. So that means that we can go back into Power BI and start working with it. So we are going to add a streaming tile. The first step here is to create a new dashboard. I'm just going to click here on Create and Dashboard, and I'm just going to call it---this is going to be our Course Dashboard. I'll just click on Create. Now we have the dashboard, it's completely blank. We want to go here at the top where it says Add tile. Now here it gives you some options. You want to scroll here down to the bottom, and we see this category that says Real-Time Data, and it says Custom Streaming Data. So we're going to click right here and then on Next. Now, we already have some datasets that Power BI detects that are actually streaming. So in this case, we want to select the one that we created at the end of the last module. It's this one, RealTimeDataset, and click on Next. Now at this point, we have the different types of visuals. I'm going to open it here right now. Now this is what I mean when I mentioned in the slides that the streaming tiles are limited by the different types of visuals that are offered. We only have these five visuals here. We have card, line chart, clustered bar chart, clustered column chart, or a gauge. Now depending on your type of data, some of these will not apply at all and will not be really useful. In our case, for example, a card would not make any sense because all the machine data is coming through the same stream from multiple machines. So the card would just be mixing up all the different values. On the other hand, a line chart is okay because we are going to be able to tell apart the different lines from the different machines. So let's check that out. I'm picking line chart here. Then on the axis is where we want to put our time series. So if I open this, you can see Power BI already detected that there's only one DateTime value in the dataset. It's windowend, so it's the end of the window, slides by Stream Analytics. Then if we look at the legend, what we want to add here is the name of the machines. So we're going to pick machinename. And then, finally, what we want to see here are the values. Now the easiest way to see the current value of the machine is to simply pick the averagetemperature because, remember, Stream Analytics is taking the multiple values that are sent by the machines per second and then averaging them out. So I'm going to pick averagetemperature. And you guys will be able to see here it gives me an option of the time window that I want to display. Now it can be either seconds or minutes. I'm going to select here that I want to see the last 30 seconds of the stream. At this point in time, it looks good, and I'm going to click on Next. Now I'm going to give it a title here. I'm just going to say Machine Temperatures. And if we scroll down here, you can actually set a link if you want so that when you click on the tile, it takes you to a specific URL. In this case, we just want to display it, so I'm just going to click here on Apply. And we can see now we have our streaming tile showing us the data live. Now it might seem like we only have one line, but actually what's going on is that all the machines right now have the temperature set to 40. Now I'm going to go back into the application. Remember, we had this temperature reading that started at 40 degrees. Now let's start to do changes and see how they reflect very fast on the streaming tile. For example, I'm going to move this here to the side so we can see it really quick as it goes to change. I'm going to enter a new command here. We're going to say, for example, we want m1 to increase. As we can see here, the change starts to happen really quickly. I did not edit the video. That's the amount of time that it took to start happening. Now let's say I want m1 to stabilize. So I'm just going to type in m1 stable, and we're going to see how it starts to refresh very fast as well once it stabilizes. As you can see here, now it's a flat line because m1 stabilized. Now let's do, for example, something like m2 decrease. Now we'll see how quickly again that streaming tile starts to change, and the scale of the streaming tile starts to change as m2 starts to decrease and move lower and lower. Once we want to see the change stop, then we can go back and say, of course, m2 stable as well. Now we can do the same with another one of the machines. For example, we could say something like m6 bounce. Now what bounce is going to do is that it's going to move it up and down randomly, not just in one trajectory like we did with m1 only increasing or m2 only decreasing. Bounce will start to move up and down as it goes along randomly. Now it's really nice to see here as well almost instantly, we can see the line there for m6 start to change up and down as it goes along bouncing. Now if we are okay with the way it is, we can just say m6 stable as well.

Power BI Real-time Report Visuals

Now that we have seen how simple it is to add a streaming tile to a dashboard, let's talk about real-time report visuals. Real-time report visuals have different behaviors depending on where they are. As long as the visual is inside the report, it has to be manually refreshed to get the latest data. However, once you pin that report visual into a dashboard, auto-refresh will kick in. And then it will start to automatically change as the data changes. These are the steps into creating an auto-refreshing report visual. First, you want to show a small recent window of time because this is all about real-time visualization. So usually we want to see 1 minute or less of the last slice of data. Next, on the report editing, we're going to add a top N filter based on the timestamp field. For example, since we are consuming data at 1 data point per second, if we add a top 10 filter, then we're going to be seeing the last 10 seconds of data. Next, we're going to add a text representation of the timestamp. This goes back to what we did in the last demo to change the dataset in our previous module. This is a current issue with Power BI working with push datasets where top N Filters are not working on the direct DateTime data. However, it's really easy to do this work around, and I'll show you exactly how on the demo. And then, finally, once you have that report visual properly configured, you just spin it to the dashboard, and auto-refresh will kick in.

Demo: Real-time Custom Report Visual

For our dashboard, we're also going to be working with Power BI custom visuals as that can really add an extra level of professionalism to your dashboards. And it is really easy to do so. In this case, for example, we're going to have a map of the floor of our warehouse. And each one of those boxes is going to be a machine. We're going to use a custom visual called a Synoptic Panel to represent the machines in the factory floor, and the map is going to change colors as the temperature changes. So let's check out that demo and create a custom real-time report visual for our machine temperature. So step #1 is going to be to create that custom visual of that factory floor map. And as I mentioned before, we are going to be using the Synoptic Panel for Power BI. You can find the Synoptic Designer here, and the URL is https://synoptic. design. Now step #1 is to simply drag the image that we want to use for our custom visual. I am going to add it as part of the download package. It's this FactorFloor. png. So I'm just going to drag it over, and then it's really simple. The Synoptic Designer already has a magic wand tool, so I just have to select area, and then give it the proper value that it should be expecting from the dataset. Now in our case, the machines are called as m1, m2, and so on. Now I'm just going to rename it here as well, m1, and we're going to continue the same way with the rest of the map. So m2 goes here, m3 will go here. Now I'm going to switch over to the hand tool so I can scroll down, back to the magic wand, then select m4. Then I'm going to select here m5. And then, finally, m6. Now at this point, the map is ready, and I just have to click here where it says Export to Power BI. Now once you have the preview here, just right-click on it and do a Save picture as. As you can see, it will save it as a file called synoptic. svg. And then we're going to be lowering that file into Power BI. So at this point, we are good to go on the Synoptic Designer. Let's go back into Power BI. I'm going to go here into the datasets. And then I'm just going to click here on our RealTimeDataset that we created in the last module and click on Create report. Now to add the custom visuals, we just go here to the three dots in the Visualizations panel and click on Import from marketplace. And here I'm just going to type synoptic, and we should be able to find it right away and just click on Add. And just like that, Power BI has added it. We just click on it, and we're going to make it bigger so it's easy to work with. So now the trick here to load the map is that we first need to put some data into the visualization. What we want to do here is to first select the averagetemperature, and we're going to use that as our measure to display in the map. Now something very, very important is that if you click here, you'll see that Power BI wants to aggregate the average temperature as a sum. Now it makes no sense to display the sum of the averages. What we want to do is actually to do the average of the average as well. So I'm just going to select here average of the average. Now as a category, we're going to select the machinename, and then as the subcategory, we're going to select here where it says lastdirection. Now, finally, we're going to go here where it says Local maps, and we can load that SVG file that we generated on the designer. Click on Open, and we can see right away Power BI has loaded the visualization. Now what we want to do is set some formatting. So I'm going to go back here to the format. First, for example, I want to turn the labels on. I'm going to open this. And I don't want to only see the data column. I also want to see the column and the value. So I have this now. And then I also want to make this bigger because, otherwise, we can't really see well. And I also want to decrease here the decimals as it has way too many. I just want to see, let's say, three decimals. And then let's go back up. Now the other thing that I want to do is for the colors to change as the temperature changes. So I'm going to go here where it says States. Now for the cooler temperatures, I want to select cooler colors, and for the higher temperatures, I want to see warmer colors. So let's say we're going to start at -1, and I'm working with Celsius here. And I'm going to select a light blue color. Then let's say we're working with 10 degrees, I'm going to select here the green. Let's say I'm moving up to 30, and I'm going to select here the yellow. Now let's say I'm going to move up to 50, so I'm going to select here more like an orange color. And then, finally, I'm going to set it to 70, which is really hot in Celsius, and I'm going to select here the more red color. And as we can see actually right away with data that is loaded right now, we can see how we have the values that are below 30 right now are of this light green color, and the ones that are above 30 but below 50 are actually in that yellow color. Next is the top N filter. Remember, this top N filter is so that we only see the latest values being taken into account for the visualization. So let's go back into the data and scroll down to the visual level filters. Now, because of a limitation in push dataset handling for Power BI, what we want to add is a visual level filter on that text representation of the timestamp. This is really critical and is the reason why we added the text as the output of Stream Analytics in the last module. So I'm going to take this field, windowendtext, and drag it over to the filters. What we want to do here is to set a top N filter, and we only want to see, let's say, the last 10 seconds. So we're going to do the top 10. And now where it says here By value, Drag data fields here, you want to drag the actual timestamp column. So in this case, it's windowend. So I'm going to put windowend here. And, of course, we don't want to see the earliest. We want to see the latest because it's real-time that we want to see being refreshed. So we pick here latest, and then very important as well, you have to click where it says Apply filter. So now we have the filter, and look it actually refreshes them again to show us what the status is right now. While you are in the report view, though, this is going to be only manual refresh. And, of course, for real-time visualization and monitoring, we want it to be changing in real-time. So, finally, we go here, and we have to pin the visual to the dashboard. So we're going to save the report. I'm just going to call it Course Report and click on Save. Now Power BI already gives us the suggestion to add it to the course dashboard. So that's what I'm going to do. Just click on Pin, and then click on Go to dashboard. As we can see here, now we have a mix. On the machine temperature side on the line chart, we have a streaming tile. And then on the other side, we have this report visual that is actually tracking the same data. However, the refresh rate on the report visual is going to be slower than the streaming tile. But it will still respond in real-time. So let's check it out. For example, if we go and see right now status, we can see how the different temperatures are, and they are correct as how they're shown in the visualization. So let's say, for example, we want m1 to increase. I'm going to type m1 increase, and we'll see it how it starts to change there in the streaming visual, and then we'll also see how it starts to change in the actual report visual as well. As you can see here now, it changed and says increasing 6, 8. 5. As it goes along, it's going to start changing. Now also keep in mind we set the different colors. We can see it already switch now to that green because it's turning a little bit hotter. Now when it crosses 30, it's actually going to change again into that yellow color. So let's see it's going to do it very soon, and we can see now it crossed 30, and it's doing it as well. Now let's see how long it takes before it crosses 50. As it crosses 50, it is going to be changed as well. Now it's going to go into the orange, for example. Forty-nine, and then as it crosses, it goes over into 50. And now we have that orange color. Now let's say we want to change it here, we'll just say m1 should stabilize. And we'll see it levels out in the streaming tile, and it will also stop changing in the report visual. If we want to do it and test it out even more, we can see the opposite applies as well. For example, m6, we can set it to decrease. And as m6 starts to drop in value, we can see how the streaming tile picks it up first, and then a few seconds after, the report visual on the map is going to start changing it. See now it says Decreasing. It changed down to 25. And it's starting to change the color as well as it starts to go down. So as you can see, the report visuals in this way also respond in real-time. It's just the refresh rate is a little bit slower. However, it's still very nice and very powerful, especially what you can build with this Synoptic Panel.

Demo: Power BI Real-time Alerts

So now we have streaming tiles and real-time report visuals on the dashboard. How do we react to real-time data changes? Power BI offers the capability of alerts. It can notify you when a certain data point reaches a certain threshold value that you configure. These alerts are set on tiles that are pinned from report visuals. So they can only be placed while these report visuals are on dashboards. There are also some limitations as to what the graphics can be. They have to be either a gauge, a KPI, or a card. The alerts get triggered when a value goes above or below a threshold that you select. So you can specify alerts when they go either too high or too low. In response to an alert, a notification and an email can be sent. This notification can be either a push notification on a mobile device, or it can also be a notification in powerbi. com's Notification Center. In our case for our solution, we are going to have a machine temperature alert because we want to alert operators when the temperature of one of the machines in an energy panel gets too high. Now you want to apply this same type of logic to whatever scenario you're building. This is the great thing about this architecture. You can just take it to whatever solution you're building either on gaming, other industries, sales, etc. So let's check out a demo and configure a Power BI alert from a real-time visual. I'm back in my Power BI workspace, and we are going to add a report visual and hook up an alert to it. So let's go to Reports and click here on Course Report. And so far what we did is in the last demo we added that custom map. So we're going to click here on Edit report. I'm going to add another page. And then here we're going to set the new visual that we're going to use for alerting. Now, remember, only some specific visual types can support alerting, that is, a gauge, a card, or a KPI. So I'm going to select a gauge, and I'm going to make it bigger. And then I'm going to select the maxtemperature as the value that I want in the gauge. Now as you can see, we get a really high value, and that is because Power BI is actually aggregating the max temperatures as a sum. That is not what we want, so we want to change it into a maximum. Also, I'm going to focus on a specific energy panel. For example, we could do that by adding energypanel here as a filter and selecting, for example, only from the north panel. At this point, if we wanted to, we could also set, for example, a minimum value or a maximum value for the gauge if we had them as fields. However, we don't have that in our dataset. You guys feel free if you want to modify the demos or the solution, go ahead, and you could do something like this. Now in our case, I just want to go here, and I'm going to change the title of our visualization so we actually know what we are looking at. We are looking at the Max Temperature of the Energy Panel North. And we're going to make that title a lot bigger because we can't really see it with that font size. So there we go. We have the Max Temperature Energy Panel North. Now, again, we need to add that filter so that we get only the last 10 seconds like we did with the map visual. So remember the steps. We take the text timestamp as the filter. We go in and select Top N. We select here 10 seconds. And then we take the real timestamp as the data field. And then we change from earliest to latest. Finally, we click on the Apply filter. At this point, we can actually go in and pin the visual and test it out in the dashboard. So I click here on Pin visual, Course Dashboard, click on Pin, and let's check it out. So now we are here in the dashboard. Now let's scroll down here to the gauge. Now if we go to the gauge and click on the three dots, we can find the option here where it says Manage alerts. So I'm going to add a new alert rule tied to the gauge. We're going to set that it is active, and what we're going to have to set is for the Max of maxtemperature. So, for example, at any point in time, if it goes above, let's say, for example, 75 degrees, I want to receive a notification. Now you have some options here, for example, to send it at most once every 24 hours or at most once an hour. I'm just going to leave it here at the default since it's only for a demo. You need to decide what's best for your situation. And if you want, it can send you an email as well and not just send you a notification. That is up to you. In this case, I'm just going to do this, Save and Close. So now I have an alert here tied. I will get it once this Max Temperature Energy Panel North, so any of the machines that are hooked up to the north energy panel, that is m1, m2, and m3, cross the value of 75. So let's do that. I have here the Event Hub application. And we can see here the current status. So m1, for example, is the one that we are looking at right now. You can see m1 right now is reading for 65. 089. So if we go, for example, and just say m1 increase, let me make this here. So I'm just going to type m1 increase, and we should start seeing it change in the visual and then also start seeing it in the alert once it changes. We can see the gauge almost immediately, not as fast as the streaming tile, but it still picked it up pretty fast. Now it's increasing. It's going to 72, 73. It's going to cross that alert threshold very fast. Now it has actually crossed it, and we'll see how long it takes for Power BI to actually generate the notification. In the meantime, I'm going to set m1 as stable. And after a few seconds, we can see a notification has appeared both on the dashboard graphic, we can see that little bell there, and also on the Notification Center here of Power BI. So I can go on it and click on it, and it tells me the alert that has been triggered and what was the violation for it. We can see here 75. 5, which is above the threshold of 75. And if we want, we can go to the tile directly where there is the value that has triggered the alert. I also get an email for it. Let's check it out. And here is the alert email that I got from Power BI. As we can see under Subject, we get the name of the alert, Alert for Max Temperature in Energy Panel North, and then we can see here in the description on the text, it gives us the accurate value that has crossed the threshold. So in this case, it says Alert for Max Temperature Energy Panel North on Course Dashboard is 75. 52142, which is above the threshold of 75. As soon as it crossed that value of 75 in that gauge visual, the email was triggered and sent to my mailbox.

Course Summary

In this module, we learned that Power BI supports real-time dashboards using the two different types of visuals, the streaming tiles or pinned report visuals. We learned how streaming tiles are added through the dashboard experience. We also learned how you can take any report visual, configure it properly, and make it exhibit real-time behavior. And, finally, we saw how easy it is to go on a dashboard and monitor alert values of your real-time data with Power BI. So at this point, let's look at the solution that we built throughout the course. We took the machine data and pushed it into Event Hubs. Then, we took that data, and we aggregated it and added reference information to it through Stream Analytics. Then, we ingested that data into Power BI real-time dataset. And, finally, we created a real-time dashboard that uses both a streaming tile and a custom visual that shows a real-time map of the changes in the machine temperature. And, finally, we set an alert so our operators can know if the machines are running too hot. With this information that you have acquired, I hope you can let your imagination fly and build your own real-time dashboard solution with Azure and Power BI. Until next time, thanks for watching!