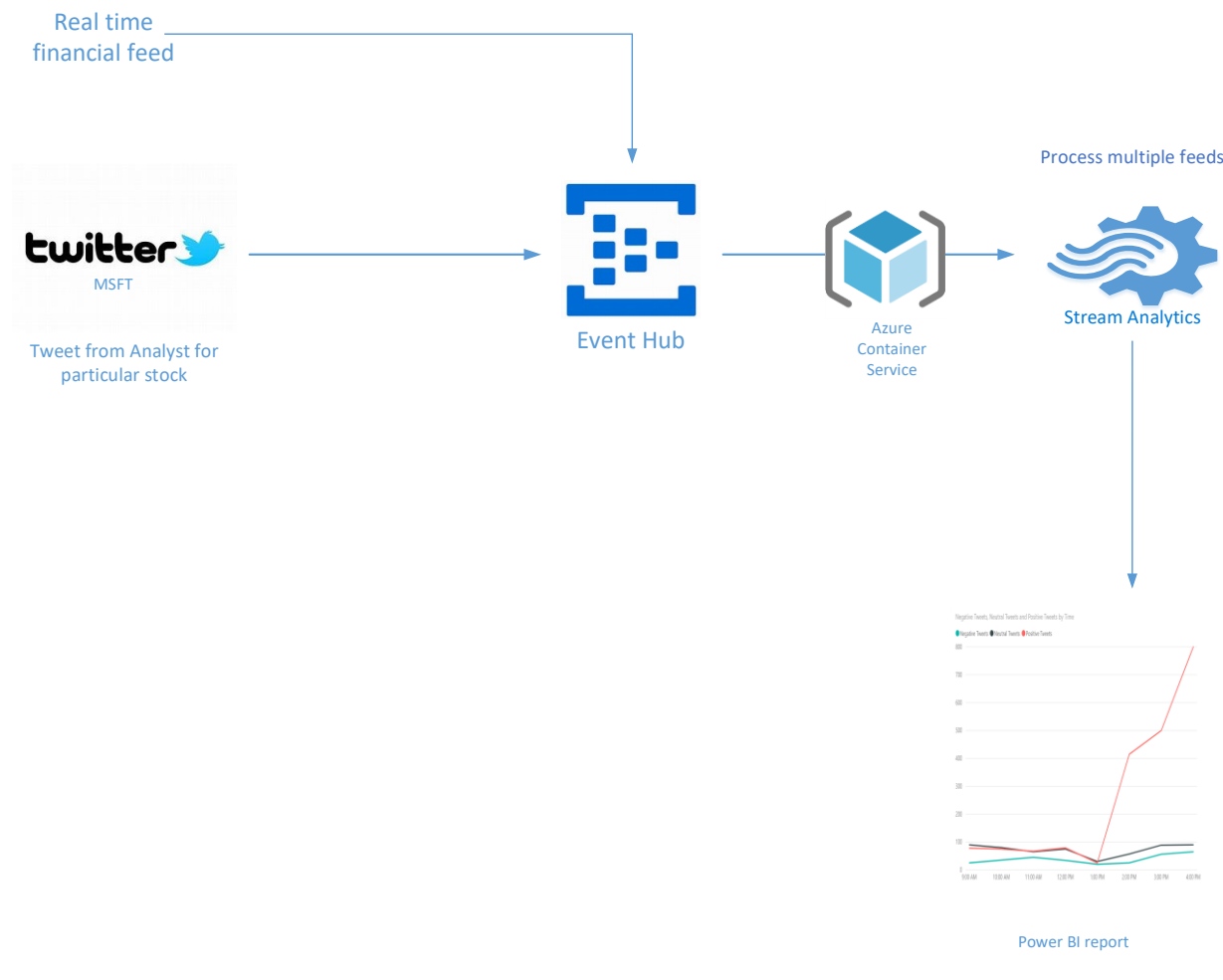


# Sentiment Analysis using Azure Streaming Analytics

## Enabling co-relation of pricing data and sentimental analysis with Azure streaming analytics service

Sentiment analysis offers added parameter for any company evaluation that may provide insight into what analyst and other influential people are thinking about stock future. Apart from many regular research information such as quarterly earnings, PE ratio etc., market sentiment can provide another datapoint.

As shown below, Azure streaming analytics and visualization product such as Power BI can provide real-time view on sentimental data and its relationship with price fluctuations.



Typical flow will include:

1. Receiving feed from multiple external sources
2. Processing feeds by fetching relevant information using Azure Streaming Analytics Query Language, a subset of T-SQL syntax
3. Presenting output to database or Power BI dashboard for real-time visualization

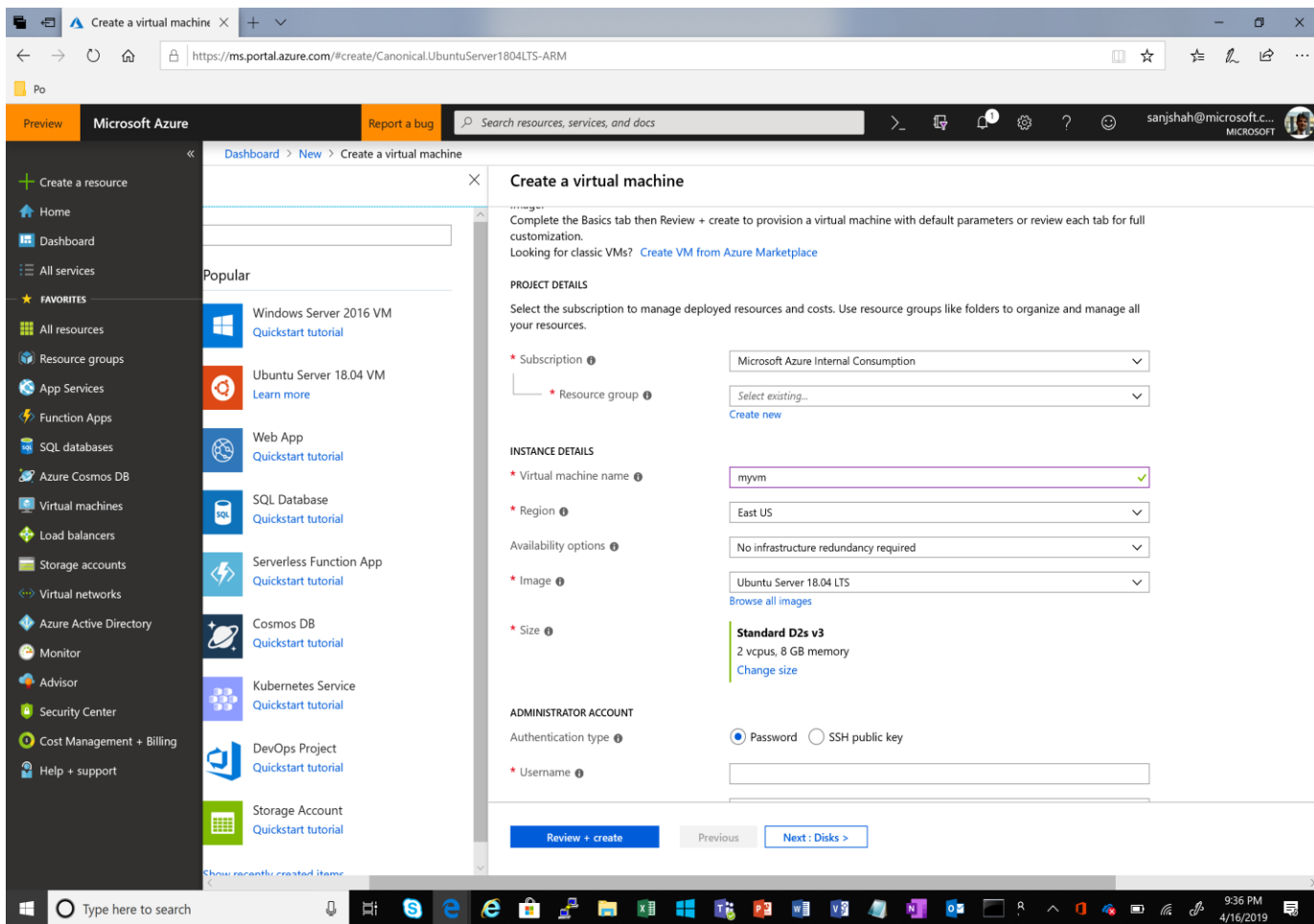
## Workshop to simulate above analysis: Pre-requisites

Before you start, make sure you have the following:

- If you don't have an Azure subscription, create a [free account](#).
- Log in to the [Azure portal](#).
- You will need Power BI account. You can create free account at <https://powerbi.microsoft.com/en-us/>.
- Python scripts and SQL query mentioned in this workshop can be pulled from Git repository at <https://github.com/sanjshah2001/sentimentanalysis>.
- Quandl.com (Third Party) account subject to all terms and condition provided by the company

### 1. Create new Linux virtual machine and login using Console:

- a. Choose **Create a resource** in the upper left corner of the Azure portal.
- b. In the search box above the list of Azure Marketplace resources, search for and select **Ubuntu Server 18.04 VM** by Canonical, then choose **Create** under "resource manager". Do not choose classic.

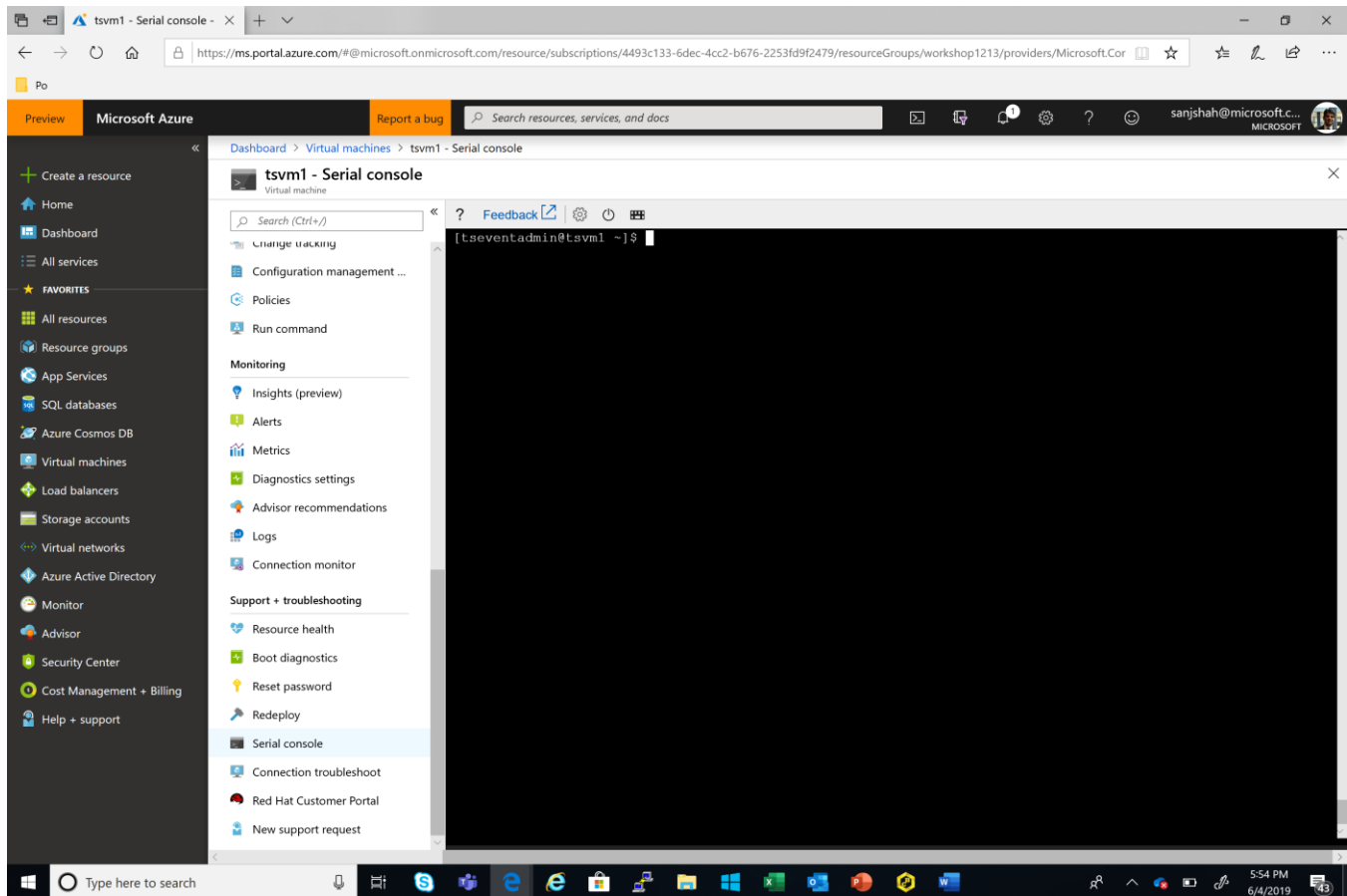


- c. In the **Basics** tab, under **Project details**, make sure the correct subscription is selected and then choose to **Create new** under **Resource group**. In the pop-up, type *myResourceGroup* for the name of the resource group and then choose **OK**.
- d. Under **Instance details**, type *myVM* for the **Virtual machine name** and choose *East US* for your **Region**. Leave the other defaults.
- e. Under **Administrator account**, select **Password**, type your username and password.
- f. Under **Inbound port rules** > **Public inbound ports**, choose **Allow selected ports** and then select **SSH(22)** and **HTTP(80)** from the drop-down.
- g. Leave the remaining defaults and then select the **Review + create** button at the bottom of the page.
- h. On the **Create a virtual machine** page, you can see the details about the VM you are about to create. When you are ready, select **Create**.

It will take a few minutes for your VM to be deployed.

## 2. Connect to virtual machine

a) Use serial console to connect to Virtual Machine:



## 3. Install Azure CLI using below command:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

(Alternatively, it can also be installed using procedure highlighted at

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt?view=azure-cli-latest>)

Perform login sequence by typing:

```
az login
```

(Follow the URL provided on command line and type in your browser. Enter code provided on the command line)

## 4. Create Azure container registry:

```
az acr create -n <YourRegistryUniqueName> -g MyResourceGroup --sku Standard
```

(Save ID that was created for creating cluster below)

5. Create AKS cluster using procedure outlined below. Run all commands from VM that you just created. Cluster creation will take time, so once you run the command to create cluster, move on to next step while it is being processed:

- a. <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-cluster>  
(Save **appld** and **password** while creating cluster to be used later-on)

## 6. Create storage account

- a. Click storage account on left panel -> Click +Add

The screenshot shows the 'Create storage account' wizard in the Azure portal. The left sidebar contains navigation links like 'Home', 'Dashboard', 'All services', and 'Storage accounts'. The main area is titled 'Create storage account' and has tabs for 'Basics', 'Advanced', 'Tags', and 'Review + create'. The 'Basics' tab is active, showing fields for 'Subscription' (Microsoft Azure Internal Consumption), 'Resource group' ((New) myresourcegroup), 'Storage account name' (analyticdata), 'Location' ((US) East US), 'Performance' (Standard), 'Account kind' (StorageV2 (general purpose v2)), 'Replication' (Read-access geo-redundant storage (RA-GRS)), and 'Access tier (default)' (Hot). The 'Review + create' button is at the bottom.

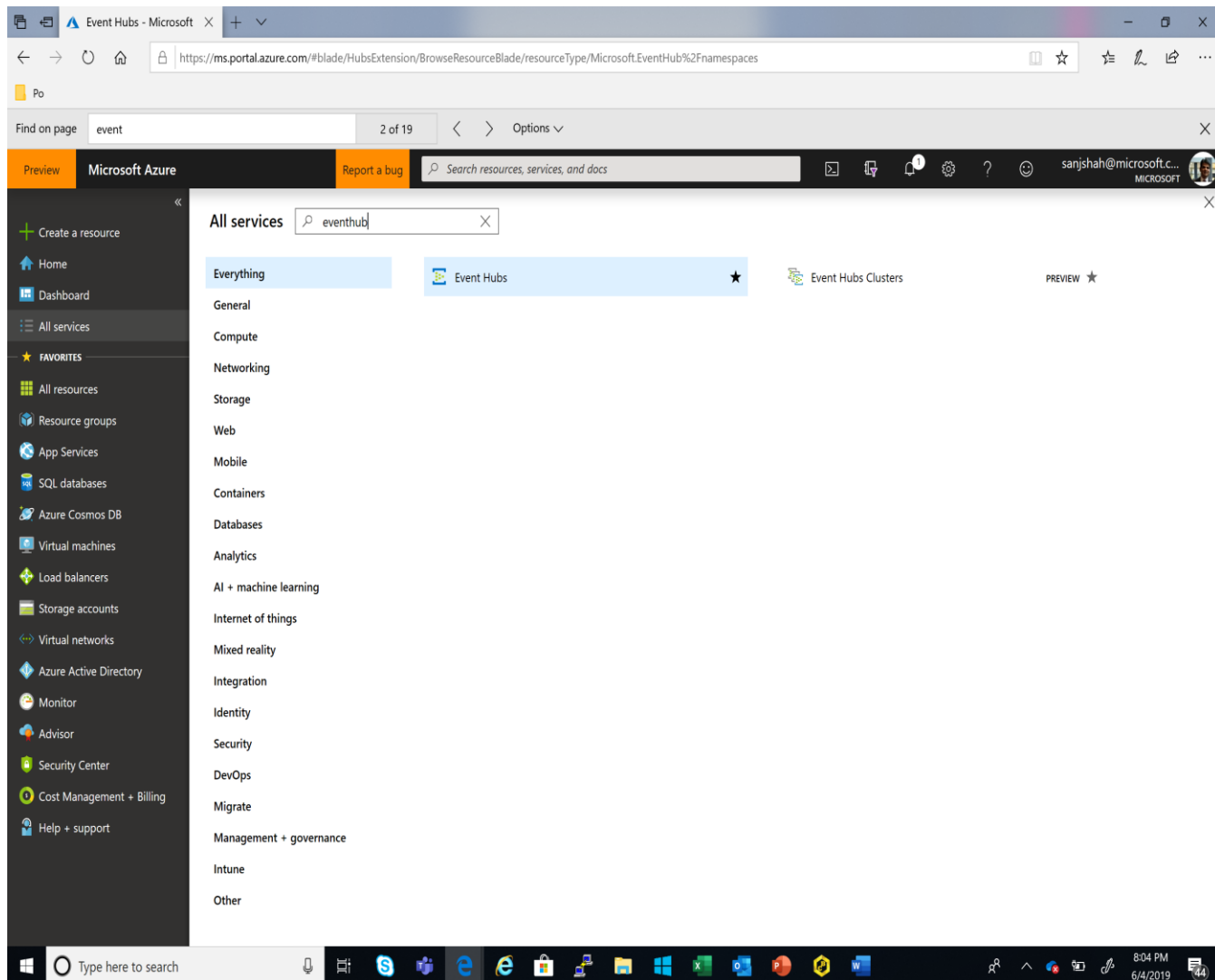
- b. Select existing resource group that you created above "myResourceGroup"
- c. Storage account name : <YourUniqueName>
- d. Location: East Us
- e. Keep rest of the items default and click "review and create"
- f. It will say "validation passed"
- g. Click create

## 7. Create an Azure Event Hub

Before Stream Analytics can analyze the stock market data stream, the data needs to be sent to Azure. In this tutorial, you will send data to Azure by using [Azure Event Hubs](#).

Use the following steps to create an Event Hub and send call data to that Event Hub:

- Log in to the [Azure portal](#).
- Select **All Services > Eventhub (Search) -> Select Event Hubs**.



- c. Fill out the **Create Namespace** pane with the following values:

Setting	Suggested value	Description
Name	myEventHubsNS	A unique name to identify the event hub namespace.
Subscription	<Your subscription>	Select an Azure subscription where you want to create the event hub.
Resource group	myResourceGroup	Select <b>Create New</b> and enter a new resource-group name for your account.
Location	East US	Location where the event hub namespace can be deployed.

- d. Use default options on the remaining settings and select **Create**.

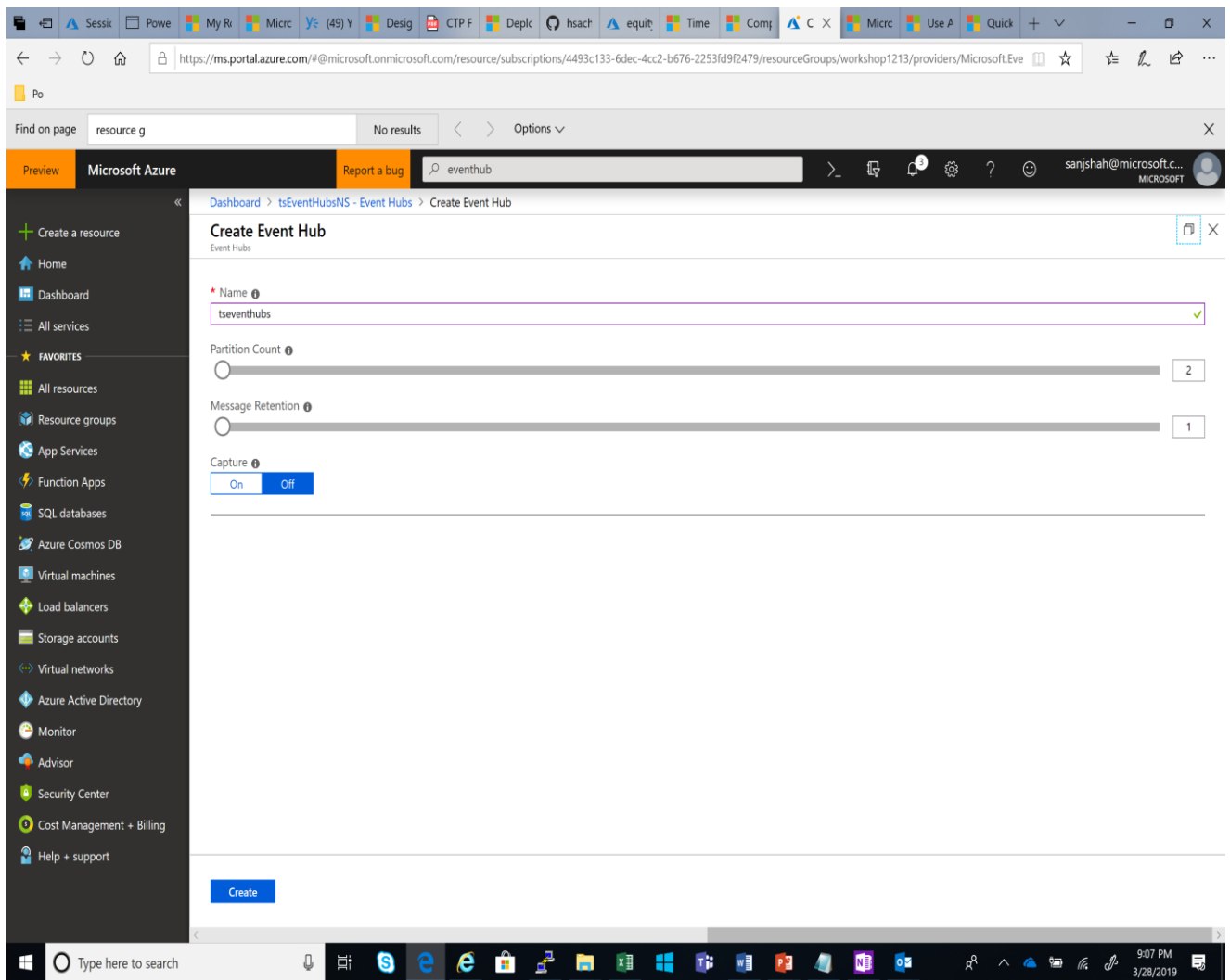
The screenshot shows the 'Create Namespace' page in the Microsoft Azure portal. The breadcrumb navigation at the top reads 'Home > New > Create Namespace'. The page title is 'Create Namespace' with 'Event Hubs' below it. The form contains the following fields and options:

- Name:** A text box containing 'myEventHubsNS' with a green checkmark icon to its right. Below the text box is the suffix '.servicebus.windows.net'.
- Pricing tier:** A dropdown menu showing 'Standard (20 Consumer groups, 1000 Brok...' with a link '(View full pricing details)' to its left.
- Enable Kafka:** An unchecked checkbox with an information icon to its right.
- Make this namespace zone redundant:** An unchecked checkbox with an information icon to its right.
- Subscription:** A dropdown menu.
- Resource group:** A dropdown menu showing '(New) MyASADemoRG' with a link 'Create new' below it.
- Location:** A dropdown menu showing 'West US 2'.
- Throughput Units:** A slider control with a value of '1' displayed in a box to its right.
- Enable Auto-Inflate:** A checked checkbox with an information icon to its right.
- Auto-Inflate Maximum Throughput Units:** A slider control with a value of '20' displayed in a box to its right.

A blue 'Create' button is located at the bottom of the form.

- e. When the namespace has finished deploying, go to **All resources** and find *myEventHubsNS* in the list of Azure resources. Select *myEventHubsNS* to open it.
- f. Create event hub for pricing data. Select **+Event Hub** and enter the **Name** as **tseventhhub** or a different name of your choice. Use the default options on the remaining settings and select **Create**. Then wait for the deployment to succeed.





- g. Create one more Event Hub for sentiment data. Select **+Event Hub** and enter the **Name** as **sentieventhub** or a different name of your choice. Use the default options on the remaining settings and select **Create**. Then wait for the deployment to succeed.

## 8. Grant access to the event hub for pricing data and get a connection string

Before an application can send data to Azure Event Hubs, the event hub must have a policy that allows appropriate access. The access policy produces a connection string that includes authorization information.

- a. Navigate to the event hub you created in the previous step, **tsevenhubs**. Select **Shared access policies** under **Settings**, and then select **+ Add**.
- b. Name the policy **MyPolicy** and ensure **Manage** is checked. Then select **Create**.

Home > Resource groups > MyASADemoRG > myEventHubsNS - Event Hubs > myeventhub - Shared access policies

myeventhub - Shared access policies

Event Hubs Instance

Search (Ctrl+*/*)

«

+

Add

Overview

Access control (IAM)

Diagnose and solve problems

Settings

Shared access policies

Properties

Locks

Automation script

Entities

Consumer groups

Features

Capture

Support + troubleshooting

New support request

Search to filter items...

POLICY

CLAIMS

no policies have been set up yet.

Add SAS Policy

Event Hubs

\* Policy name

MyPolicy ✓

☒ Manage

☒ Send

☒ Listen

Create

- c. Once the policy is created, select to open the policy, and find the **Connection string–primary key**. Select the blue **copy** button next to the connection string.

The screenshot shows the Azure portal interface for managing Event Hubs. On the left, the 'Shared access policies' table lists the 'MyPolicy' policy with claims for 'Manage, Send, Listen'. On the right, the 'SAS Policy: MyPolicy' pane displays the policy details. The 'Connection string–primary key' is highlighted with a red box, showing the full connection string: 'Endpoint=sb://myeventhubsns.servicebus.windows.net/;SharedAccessKeyName=MyPolicy;SharedAccessKey=Z9oG...'. A blue copy icon is visible next to the connection string.

- d. Paste the connection string into a text editor. You need this connection string in the next section.

The connection string looks as follows:

```
Endpoint=sb://<Your event hub namespace>. servicebus.windows.net/;SharedAccessKeyName=<Your
shared access policy name>;SharedAccessKey=<Generated key>;EntityPath=<Your event hub name>
```

Notice that the connection string contains multiple key-value pairs separated with semicolons: Endpoint, SharedAccessKeyName, SharedAccessKey, and EntityPath.

You will require:

<**Your event hub namespace**> OR <myEventHubsNS>  
 <**Your shared access policy name**> OR <MyPolicy>  
 <**Generated key**>  
 <**Your event hub name**> OR <tseventhuhub>

## 9. Grant access to the event hub for sentiment data and get a connection string

Create another policy for sentiment data using similar procedure outlined in last step.

- Navigate to the event hub you created in the previous step, **tseventhub**. Select **Shared access policies** under **Settings**, and then select + **Add**.
- Name the policy **SentiPolicy** and ensure **Manage** is checked. Then select **Create**.
- Copy **Connection string–primary key**. Select the blue **copy** button next to the connection string.
- You will require:

<**Your event hub namespace**> OR <myEventHubsNS>

<**Your shared access policy name**> OR <SentiPolicy>

<**Generated key**>

<**Your event hub name**> OR <sentieventhub>

## 10. Create a Stream Analytics job

Now that you have a stream of call events, you can create a Stream Analytics job that reads data from the event hub.

- To create a Stream Analytics job, navigate to the [Azure portal](#).
- Select **Create a resource** > **Internet of Things** > **Stream Analytics job**.
- Fill out the **New Stream Analytics job** pane with the following values:

Setting	Suggested value	Description
Job name	<b>tsAnalytics</b>	A unique name to identify the event hub namespace.
Subscription	<Your subscription>	Select an Azure subscription where you want to create the job.
Resource group	myResourceGroup	Select <b>Use existing</b> and enter a new resource-group name for your account.

Setting	Suggested value	Description
Location	East US	Location where the job can be deployed. It's recommended to place the job and the event hub in the same region for best performance and so that you don't pay to transfer data between regions.
Hosting environment	Cloud	Stream Analytics jobs can be deployed to cloud or edge. Cloud allows you to deploy to Azure Cloud, and Edge allows you to deploy to an IoT edge device.
Streaming units	1	Streaming units represent the computing resources that are required to execute a job. By default, this value is set to 1. To learn about scaling streaming units, see <a href="#">understanding and adjusting streaming units</a> article.

- d. Use default options on the remaining settings, select **Create** and wait for the deployment to succeed.

## 11. Configure job input

The next step is to define an input source for the job to read data using the event hub you created in the previous section.

- a. From the Azure portal, open the **All resources** pane, and find the **tsAnalytics** Stream Analytics job.
- b. In the **Job Topology** section of the Stream Analytics job pane, select the **Inputs** option.
- c. Select **+ Add stream input** and **Event hub**. Fill out the pane with the following values:

(This input stream is for pricing data)

Setting	Suggested value	Description
Input alias	tsinput	Provide a friendly name to identify your input. Input alias can contain alphanumeric characters, hyphens, and underscores only and must be 3-63 characters long.

Subscription	<Your subscription>	Select the Azure subscription where you created the event hub. The event hub can be in same or a different subscription as the Stream Analytics job.
Event hub namespace	myEventHubsNS	Select the event hub namespace you created in the previous section. All the event hub namespaces available in your current subscription are listed in the dropdown.
Event Hub name	tseventhub	Select the event hub you created in the previous section. All the event hubs available in your current subscription are listed in the dropdown.
Event Hub policy name	Mypolicy	Select the event hub shared access policy you created in the previous section. All the event hubs policies available in your current subscription are listed in the dropdown.

- d. Use default options on the remaining settings and select **Save**.
- e. Select + **Add stream input** and **Event hub**. Fill out the pane with the following values:  
(This input stream is for sentiment data)

Setting	Value	Description
Input alias	sentiinput	Provide a friendly name to identify your input. Input alias can contain alphanumeric characters, hyphens, and underscores only and must be 3-63 characters long.
Subscription	<Your subscription>	Select the Azure subscription where you created the event hub. The event hub can be in same or a different subscription as the Stream Analytics job.
Event hub namespace	myEventHubsNS	Select the event hub namespace you created in the previous section. All the event hub namespaces available in your current subscription are listed in the dropdown.
Event Hub name	sentieventhub	Select the event hub you created in the previous section. All the event hubs available in your current subscription are listed in the dropdown.

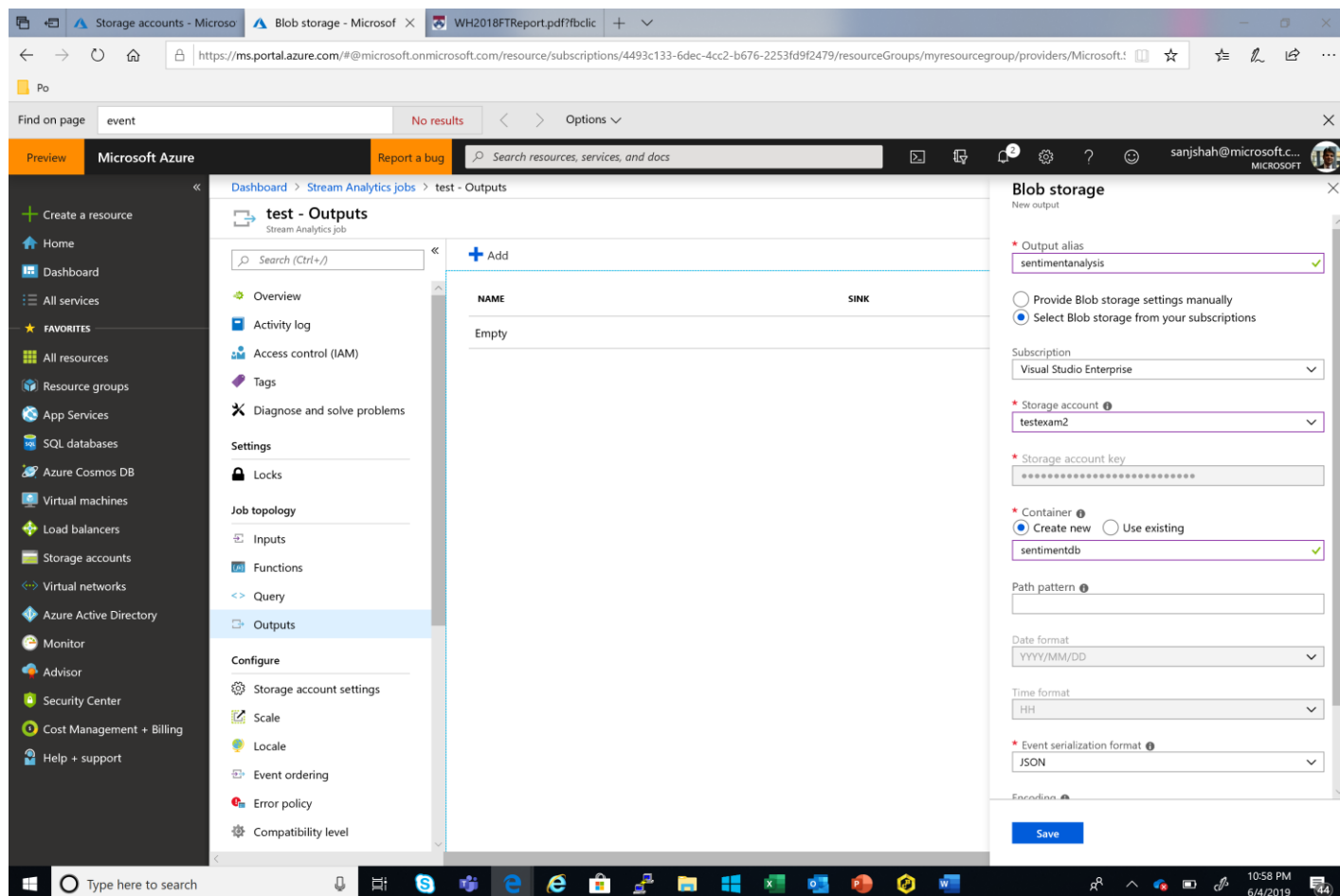
Event Hub policy name	SentiPolicy	Select the event hub shared access policy you created in the previous section. All the event hubs policies available in your current subscription are listed in the dropdown.
--------------------------	-------------	---

## 12. Configure job output

The last step is to define an output sink for the job where it can write the transformed data. In this tutorial, you output and visualize data with Power BI.

- From the Azure portal open **All resources** pane, and the **tsAnalytics** Stream Analytics job.
- In the **Job Topology** section of the Stream Analytics job pane, select the **Outputs** option.
- Select **+ Add > Blob Storage**. Then fill the form with the following details and select **Authorize**:

Setting	Suggested value
Output alias	sentimentanalysis
Storage account	<Pick from dropdown menu>
Container (Create new)	sentimentdb
Rest of the fields	<Default>



### 13. Query to analyze input data

The next step is to create a transformation that analyzes data in real time. You define the transformation query by using [Stream Analytics Query Language](#).

From the Azure portal open the **All resources** pane and navigate to the **tsAnalytics** Stream Analytics job you created earlier.

- In the **Job Topology** section of the Stream Analytics job pane, select the **Query** option. The query window lists the inputs and outputs that are configured for the job, and lets you create a query to transform the input stream.
- Replace the existing query in the editor with the following query

```
SELECT
CAST(T1.Price AS FLOAT(4)) AS Price,
CAST(T2.Date AS datetime) AS Date,
CAST(T2.Sentiment AS FLOAT(4)) AS Sentiment
INTO sentimentanalysis
FROM tsinput T1
JOIN sentiinput T2
ON T1.PDate=T2.Date
```



```
AND DATEDIFF( minute , T1 , T2 ) BETWEEN 0 AND 15
```

- c. **Save** the query.

#### 14. Start the job

- a. To start the job, navigate to the **Overview** pane of your job and select **Start**.
- b. Select **Now** for job output start time and select **Start**. You can view the job status in the notification bar.

#### 15. Go back to VM and check if job to create cluster is finished

#### 16. Get code from Git repository, run following commands from VM:

- a. **git clone** <https://github.com/sanjshah2001/sentimentanalysis.git>
- b. **cd sentimentanalysis**
- c. **Edit gettsdata.py script using vim or your favorite editor and update quandl API key with your own key in the script.**

```
quandl.ApiConfig.api_key = <YOUR KEY>
```

- d. **Update script "pushtsdata.py" with your Event Hub credentials that you created earlier for both sections under price\_data and senti\_data respectively:**

```
ADDRESS = sb://<Your event hub namespace>.servicebus.windows.net/<Your event hub name>
```

```
USER = <Your shared access policy name>
```

```
KEY = <generated key>
```

#### 17. Build and push docker image:

- a. az login (In case you logged out)
- b. Install docker:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt install docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo docker --version
```

- c. `sudo docker build -t timeseries .`
- d. `sudo docker run -it --name ts -d timeseries`

- e. `sudo az acr login --name <YourRegistryUniqueName>`
- f. `sudo docker tag timeseries <YourRegistryUniqueName>.azurecr.io/timeseries`
- g. `sudo docker push <YourRegistryUniqueName>.azurecr.io/timeseries`

## **18. Install the Kubernetes CLI**

- a. `sudo az aks install-cli`
- b. `PATH=$PATH:/usr/local/bin; export PATH`

## **19. Connect to cluster using kubectl**

- a. `sudo az aks get-credentials --resource-group myResourceGroup --name myAKSCluster`

## **20. Update ts.yaml file on line 11 and 15 with name of your own registry name:**

Image: `<YourRegistryUniqueName>.azurecr.io/timeseries`

## **21. Create POD using kubectl**

- a. `sudo kubectl create -f ts.yaml`  
(This will create Docker POD called tmseries and two containers called gettsdata and pushtsdata)
- b. `sudo kubectl describe pods` (To see if PODs are created)

## **22. Login to gettsdata container by typing:**

- a. `sudo kubectl exec -it finservpod --container gettsdata -- /bin/bash`

## **23. Run script to fetch data from Quandl (Nasdaq):**

- a. `chmod 755 *.py`
- b. `/app/gettsdata.py`  
(This will run the script with default values. Use default for this lab).

(You can use customized values using below arguments for any future use:  
`/app/gettsdata.py -i <TickerSymbol> -r <RANGE> -m <MONTH> -d <DAY> -y <YEAR>`  
ensure that data provider has relevant information for the Ticker Symbol and date range)

Upon running it will create two output files:

price\_data – Collection of pricing data for the time range provided

senti\_data – Collection of sentimental data for the time range provided

## 24.Run script to push data to Azure storage Blob:

(This will send pricing and sentiment data to Storage Blob)

- /app/pushtsdata.py -i senti\_data
- /app/pushtsdata.py -i price\_data

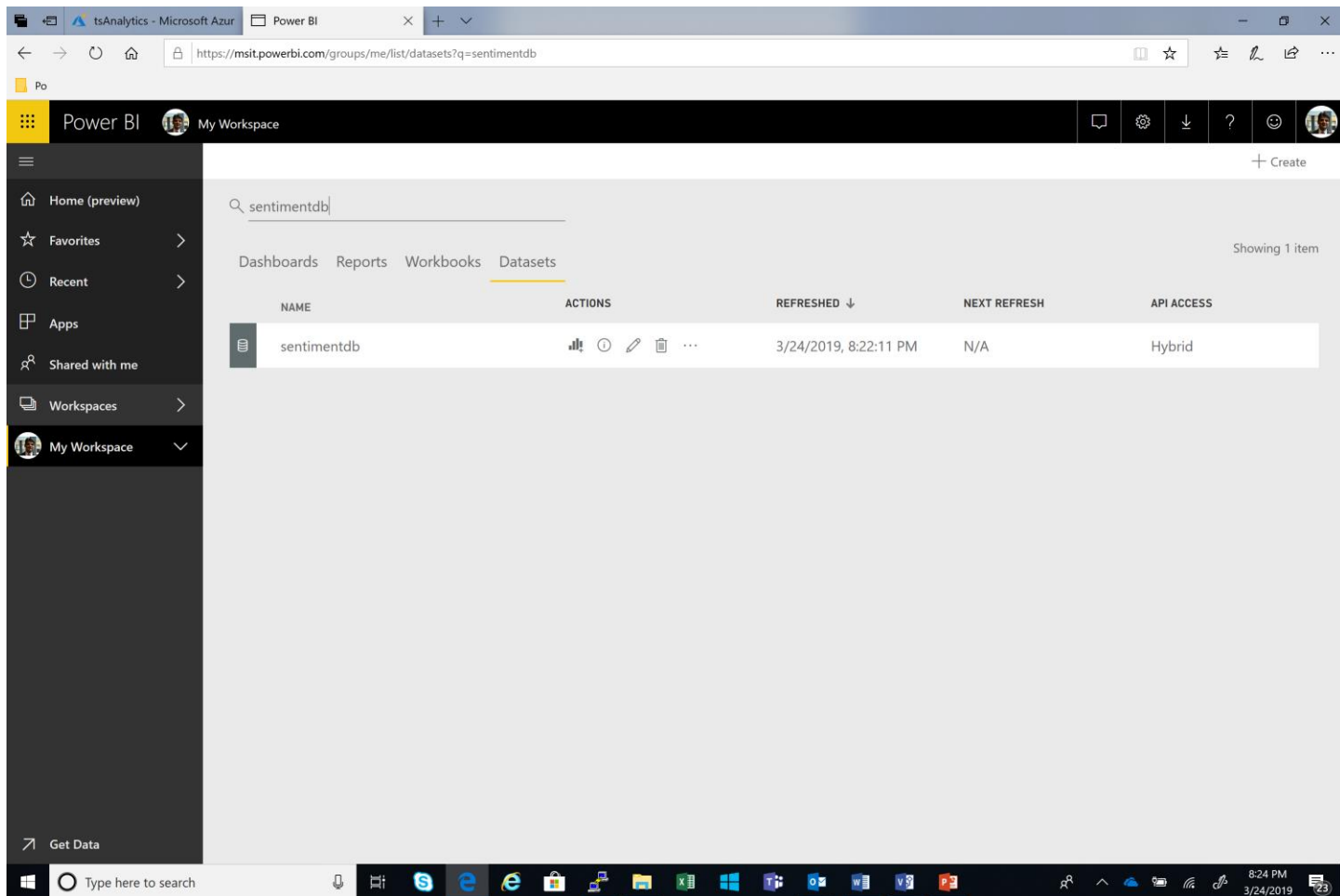
## 25.View data within storage container:

Go to storage container you created earlier and check for files senti\_data and price\_data

## 26.Visualize output using Power BI (Skip this step for non-PowerBI users)

Once the job succeeds, navigate to [Power BI](https://powerbi.microsoft.com) (powerbi.microsoft.com) and sign in with your work or school account. If the Stream Analytics job query is outputting results, the *sentimentdb* dataset you created exists under the **Datasets** tab.

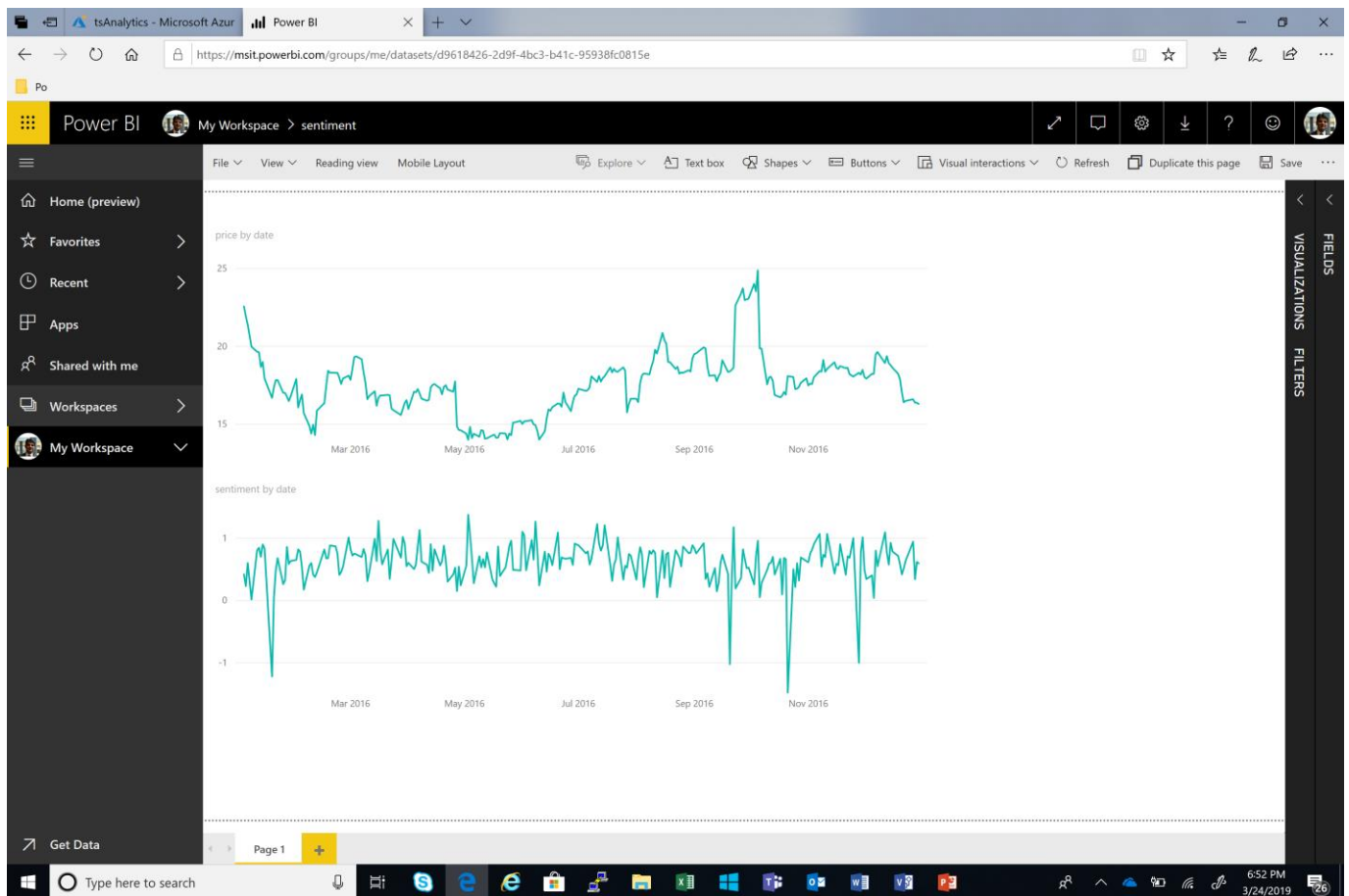
Go to Datasets under "My Workspace"



- Click on graph icon under actions for your dataset which will open a new dashboard
- Select date and price under FIELDS window on right and then hit the line graph in middle pane

- c. Select date and sentiment under FIELDS window on right and then hit the line graph in middle pane
- d. View should come up comparing stock market price and its sentiment value

Sample Power BI output that shows relation between price and sentiment data. It can be refreshed as new data comes in:



## DevOPS Automation:

### 1. Get repository:

- a. Login to Github.com on Web portal
- b. **Clone** below repository to your account:

<https://github.com/sanjshah2001/sentimentanalysis.git>

### 2. Update ts.yaml file:

- a. Click on ts.yaml
- b. Click Edit (Pencil Button)
- c. Change line 11 and 15 with your own registry name:  
Image: <YourRegistryUniqueName>.azure.io/timeseries

### 3. Create new DevOPS project

- a. Login to dev.azure.com
- b. Click on Create New Project with project name finservdevops
- c. Select visibility as public
- d. Click on advance
- e. Make sure version control is Git and Work item process is Basic

### 4. Create Build Pipeline:

- a. Click on pipeline -> New pipeline
- b. Select GitHub (YAML) (Not enterprise server option)
- c. Select repository that you cloned earlier (Your own repository)
- d. Screen will show azure-pipeline.yml file (DO NOT RUN IT YET)
- e. Go to Project settings page at the bottom left
- f. Create service endpoint as per procedure listed at <https://docs.microsoft.com/en-us/azure/devops/pipelines/library/connect-to-azure?view=azure-devops>

Connection Name: finservrg

Scope Level: Subscription

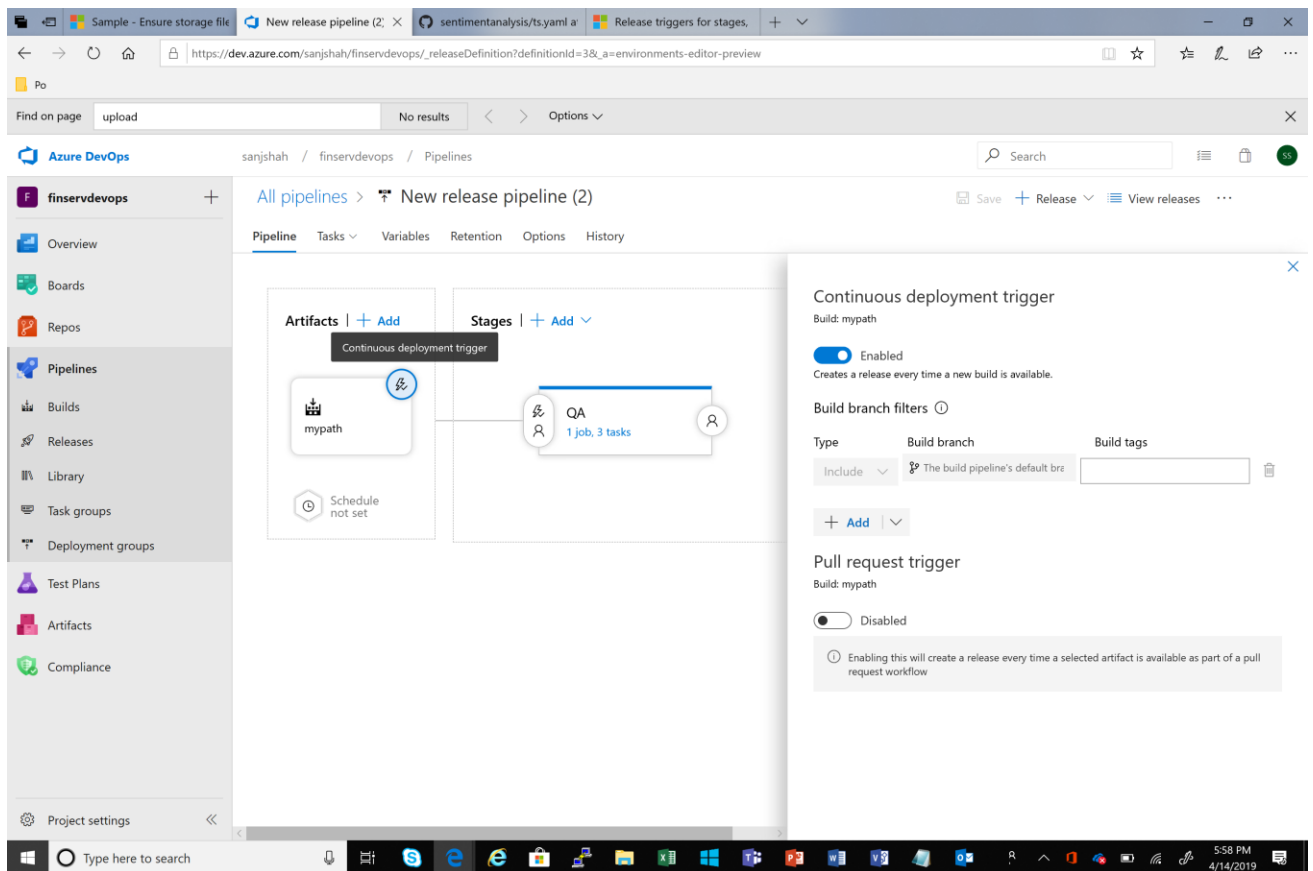
Subscription: Choose from drop down that you used in last exercise

Resource Group: myResourceGroup (Or name you gave in last exercise)

- g. Go to pipeline on left menu and select "Queue" on top right corner
- h. Select branch as your local branch and press Queue
- i. Click on "Update ts.yaml" to view progress of the build job
- j. As shown, this job will:
  - i. Build customized image
  - ii. Tag image
  - iii. Push image
  - iv. Run image
  - v. Copy source files on to containers
  - vi. Push artifacts to the release cycle

## **5. Create release pipeline:**

- a. Click on release option on left menu
- b. Select "New Pipeline"
- c. Select template "Deploy to a Kubernetes cluster" and Apply
- d. Select "QA" as stage name and then click save. Choose default options on pop-up window
- e. Next to Artifact, click "Add"
- f. Click "Build" as source type
- g. Choose project name from drop down that you created during build cycle
- h. Select source repository from drop down as build.pipeline
- i. Keep "Default" as latest version
- j. Under Source Alias enter "finpath"
- k. Click save
- l. Click small icon on top of Artifacts box which will open up continuous deployment trigger:



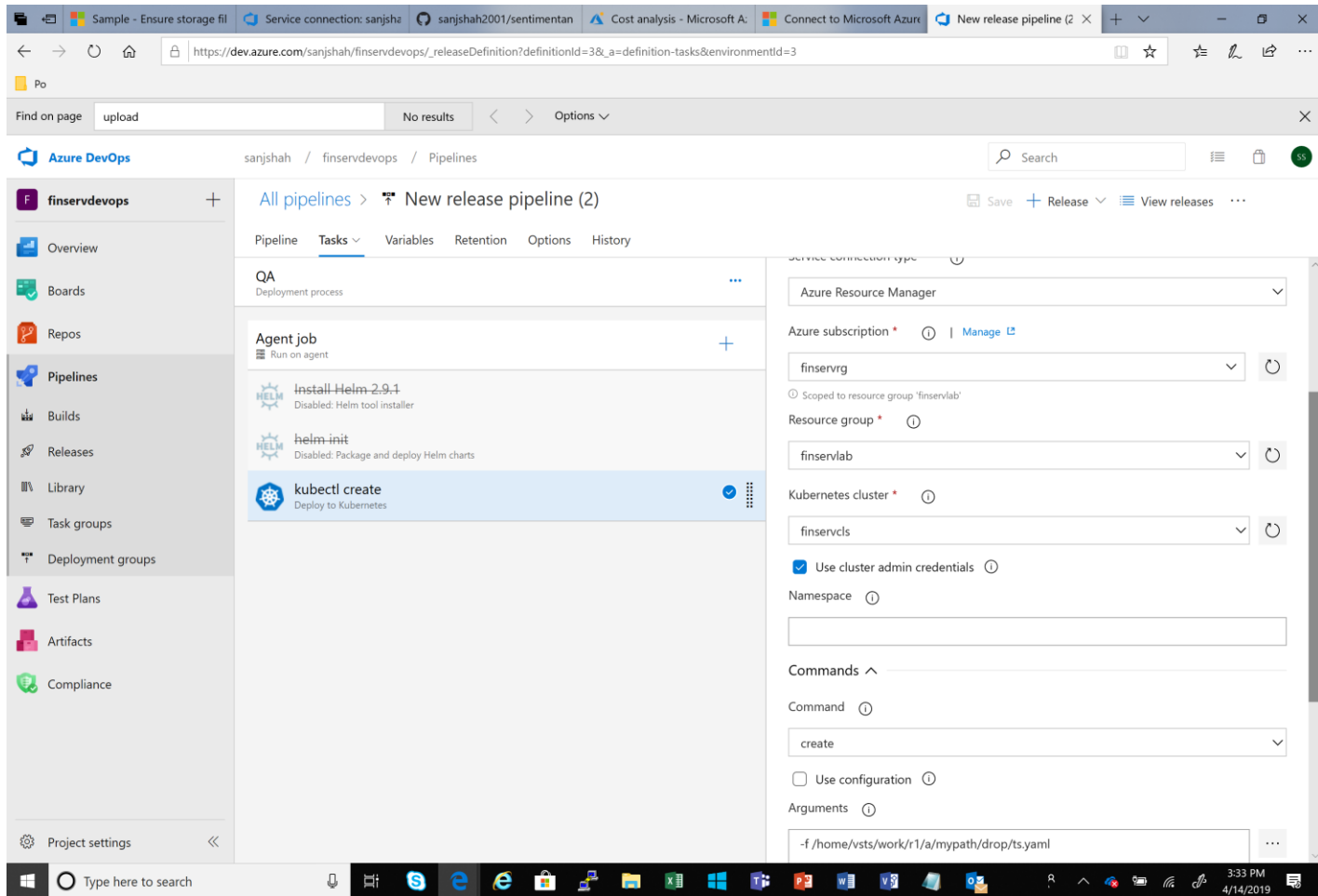
- m. Enable option where it says "Create a release every time new build is available"
- n. Give any tag name and Save
- o. Click on "Task" from the menu on top
- p. Click on Kubernetes and window will open on right
- q. Under Kubernetes Service connection, select "Manage"  
(select "Kubernetes" from the dropdown for New Service Connection)
- r. This will open another tab to create new service connection
- s. Click on Add new service connection
- t. Select Azure subscription
- u. Give Connection Name "KubeConnection"
- v. Select your subscription
- w. Drop down menu will automatically find your Kubernetes cluster that you created in previous exercise
- x. Select "default" as namespace

y. Click OK

z. Click on Variable task on top and add following variable:

- i. Name: imageRepoName                      Value:  
    <YourRegistryUniqueName>.azurecr.io/timeseries

aa. Go back to previous window and refresh Kubernetes service connection option. Drop down will show new connection you created. Select that one



bb. Under command enter "create"

cc. Under argument enter "-f D:\a\r1\a\finpath\drop\ts.yaml"

dd. Click save

## 6. Test new release

- a. Go to your GIT repository
- b. Click on ts.yaml
- c. Click Edit (Pencil Button)



- d. Edit file ts.yaml and change POD name from finservpod to finservpodv2 under metadata
- e. Save and commit changes
- f. Go back to Azure DevOPS build pipeline, a new build would have already triggered
- g. Once finished, check release pipeline, which is automatically triggered upon successful finish of build job
- h. Once release job is finished successfully, login to your VM created in previous exercise
- i. `sudo kubectl describe pods` (To ensure new POD is created)
- j. If successful release, a new POD called "finservpodv2" is created:

Confirm by logging in to VM you created earlier by typing:

```
kubectl exec -it finservpodv2 --container gettsdata -- /bin/bash
```


## 7. Security and Compliance Testing (Bonus Lab)

- a. Challenge: Modify the python script to export the data files (price\_data and senti\_data) directly to storage accounts and ensure that file and storage account encryption is enabled for secure data storage. For secure data storage enforcement, you should use Azure Policy to enforce this on your resourcegroup. Below are some helpful links to do so:
  - i. <https://docs.microsoft.com/en-us/azure/governance/policy/samples/ensure-https-storage-account>
  - ii. <https://docs.microsoft.com/en-us/azure/governance/policy/samples/ensure-storage-file-encryption>
  - iii. <https://docs.microsoft.com/en-us/azure/governance/policy/samples/require-storage-account-encryption>
  - iv. <https://docs.microsoft.com/en-us/azure/governance/policy/concepts/definition-structure>
- b. If you have an existing storage account within Azure for this lab, once you apply the policies to the resourcegroup, you can take action on them within Azure Security Center:
  - i. Example – below shows a storage account that does not meet the secure data transfer policy:

Dashboard > Security Center - Data & storage > finservlabdiag470

### finservlabdiag470

Storage account security health

Resource health  
 finservlabdiag470

Total recommendations  
**2**

Recommendations summary  





High	1	<div></div>
Medium	0	
Low	1	<div></div>

^ Storage account information


Resource Name	finservlabdiag470
Resource Group	finservlab
Subscription	MS Internal (gcheng)

^ Recommendation list

Recommendations (2) Passed assessments (1) Unavailable assessments (0)

DESCRIPTION	STATUS
 Require secure transfer to storage account	 High
 Restrict access to storage accounts with firewall and virtual network configurations (Preview)	 Low

- ii. If you drill down to non-compliant items, it provides remedy actions:  
Require secure transfer to storage account

 You have limited permissions on some of your subscriptions. Click here to load data on these subscriptions as well - This may take some time. →

^ Description

Secure transfer is an option that forces your storage account to accept requests only from secure connections (HTTPS). Use of HTTPS ensures authentication, confidentiality, and session-hijacking.

^ General Information

User impact	Low
Implementation cost	Low

^ Threats

- Data exfiltration
- Data spillage
- Threat resistance

^ Remediation steps

To enable secure transfer required:

- In your storage account, go to the 'Configuration' page.
- Enable 'Secure transfer required'.

Take action

- c. If you are interesting in exploring Azure Policy further, there is an additional hands on lab here:  
<https://handsonlabs.microsoft.com/handsonlabs/SelfPacedLabs?storyId=story://content-private/content/sp-azuregovernance/1-azpolicy/a-policy>