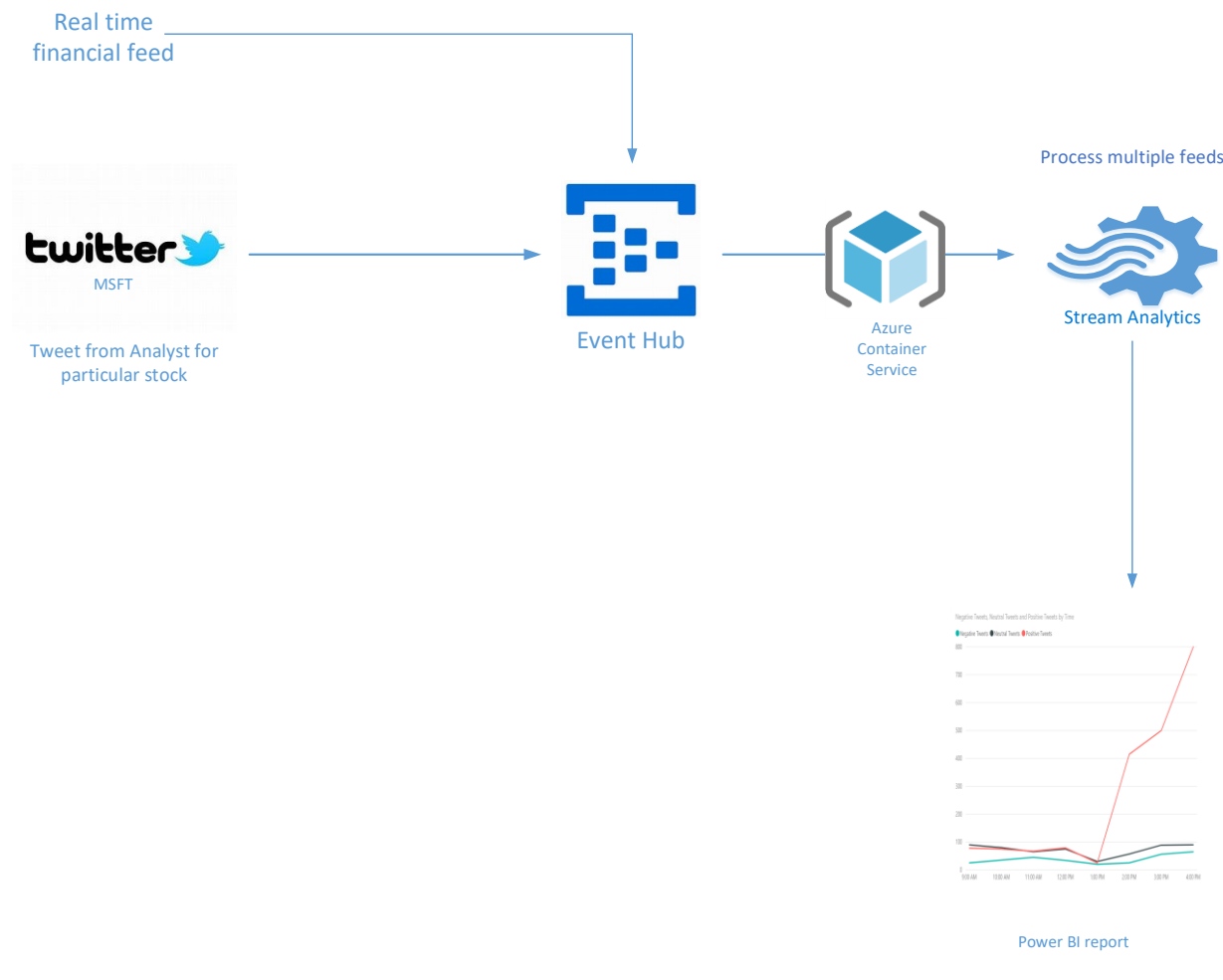# Sentiment Analysis using Azure Streaming Analytics

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.

Real time financial feed

Process multiple feeds

MSFT

Tweet from Analyst for particular stock

Event Hub

Azure Container Service

Stream Analytics

Power BI report

Power BI

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: Prerequisites

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 SSH:**

    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.
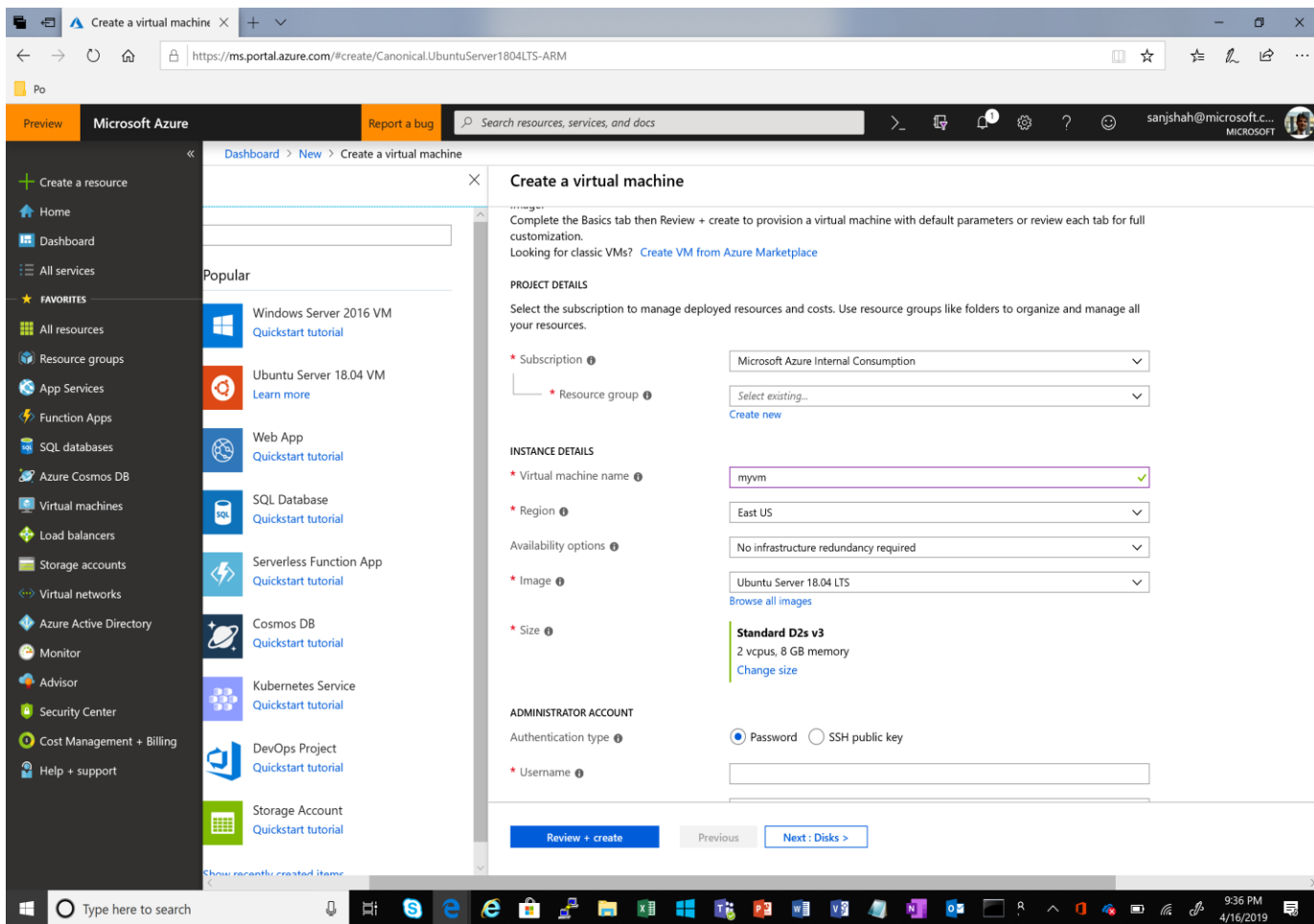
c.   In the **Basics** tab, under **Project details**, make sure the correct subscription is selected and then choose to **Create new** under **Resource group**. In the pop-up, type *myResourceGroup* for the name of the resource group and then choose **OK**.

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) Select the **Connect** button on the overview page for your VM.
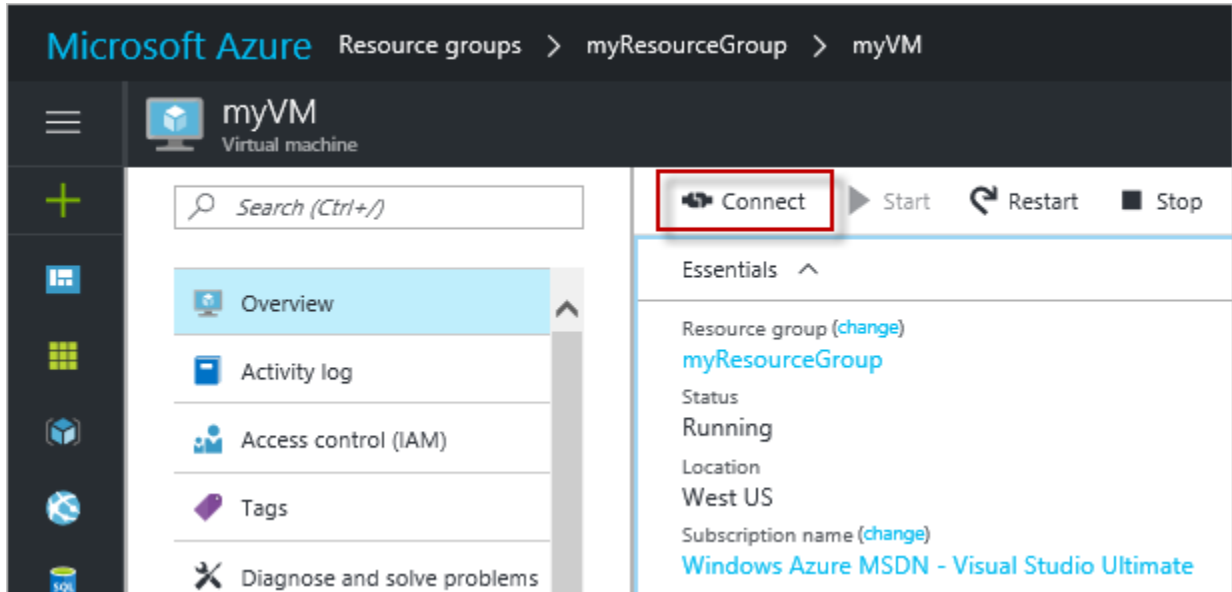
   b) In the **Connect to virtual machine** page, keep the default options to connect by IP address over port 22. In **Login using VM local account** a connection command is shown. Click the button to copy the command. The following example shows what the SSH connection command looks like:

   bash Copy

   ```
   ssh <username>@<IP ADDRESS>
   ```

   c) Paste the SSH connection command into the shell to create an SSH session.

3. **Install Azure CLI 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 finservreg -g MyResourceGroup --sku Standard
   (Save ID that was created for creating cluster below)

5. **Create AKS cluster using procedure outlined at:**
   a. **https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-cluster**
      (Save appId and password to be to create cluster)
      (Go to next step while cluster is being created)

6. **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:

   a. Log in to the Azure portal.

   b. Select **Create a resource** > **Internet of Things** > **Event Hubs**.
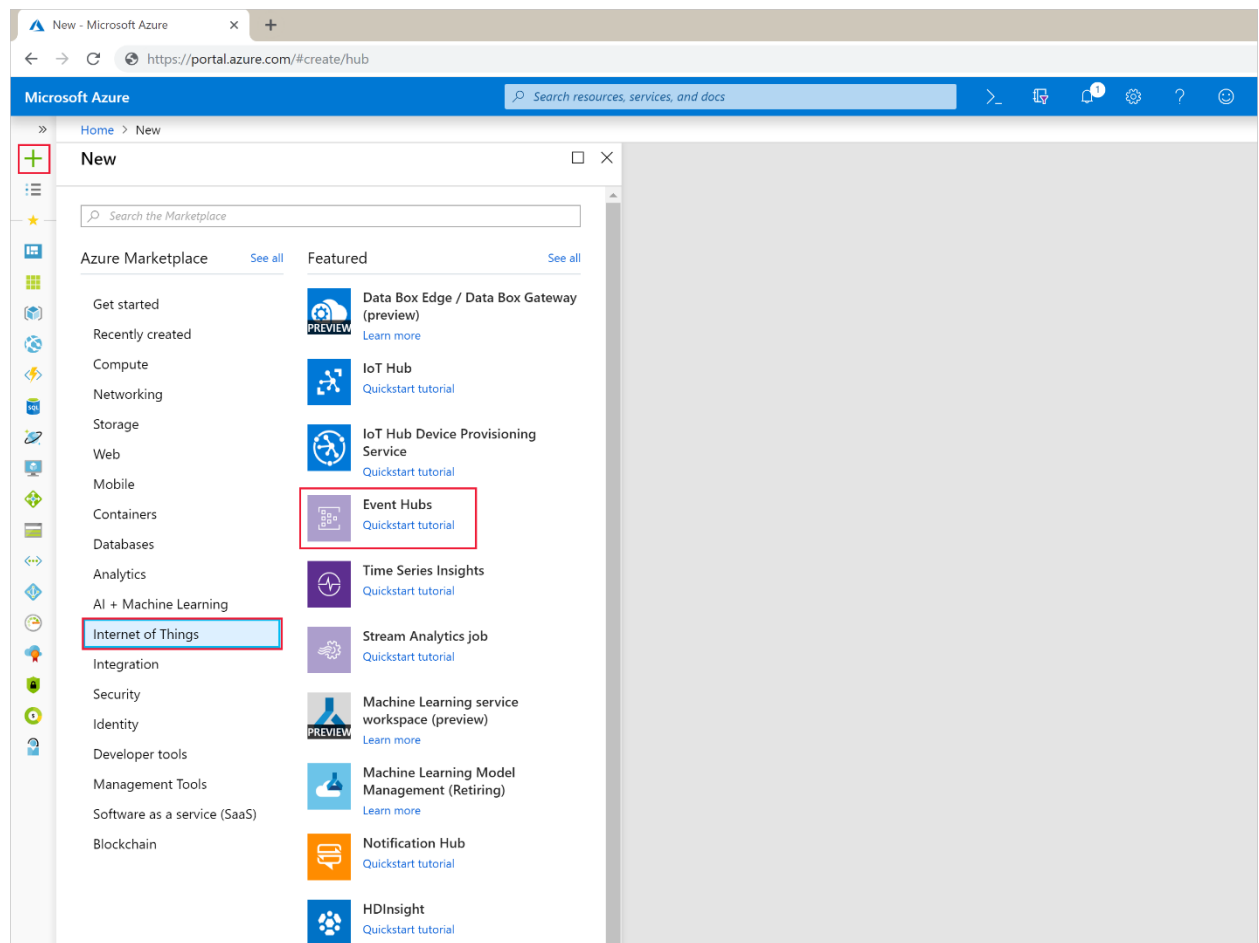
c. Fill out the **Create Namespace** pane with the following values:

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 **Create New** 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**.

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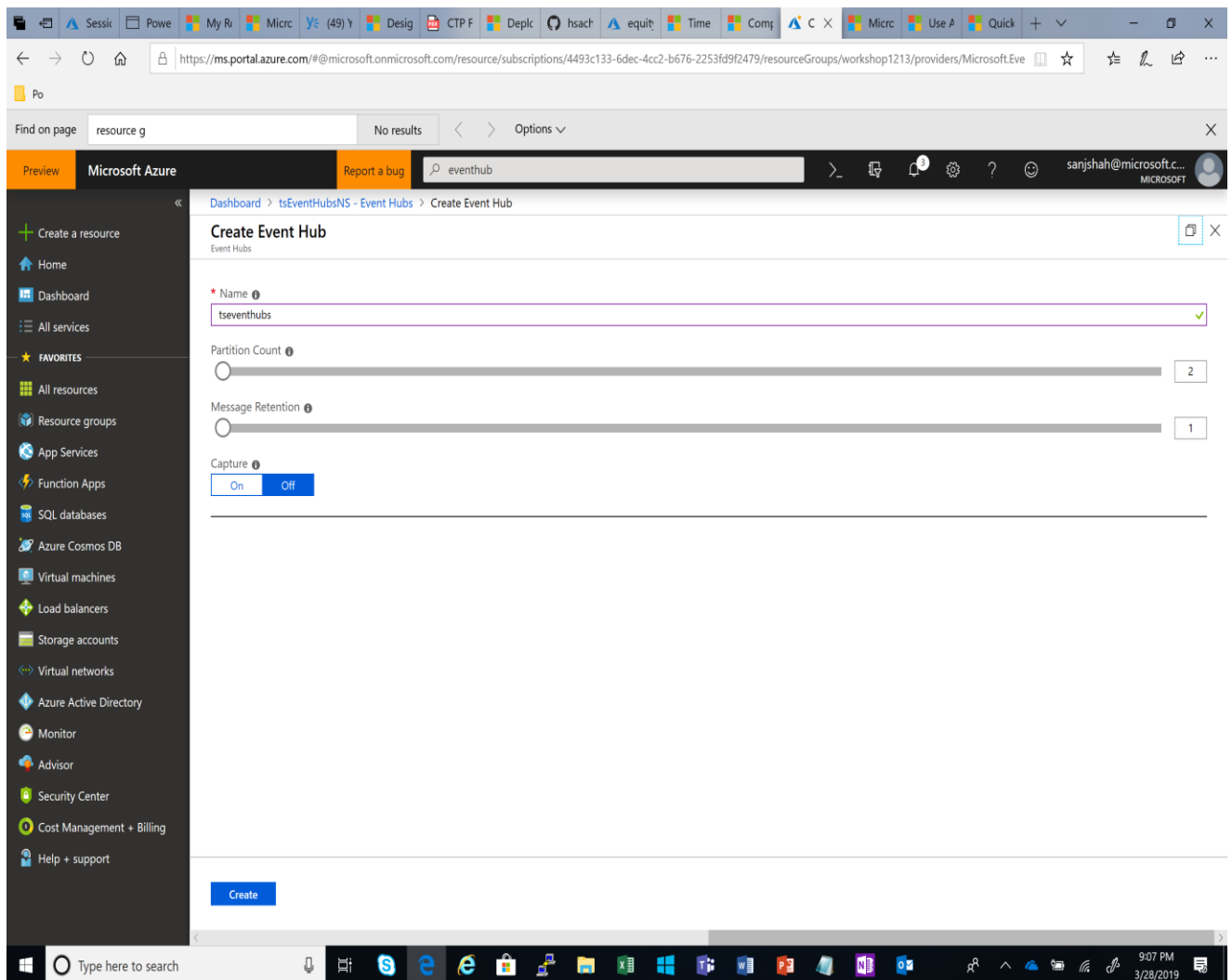
g. Create one more Event Hub for sentiment data. Select **+Event Hub** and enter the **Name** as **sentieventhub** or a different name of your choice. Use the default options on the remaining settings and select **Create**. Then wait for the deployment to succeed.

7. 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, **tseventhub**. Select **Shared access policies** under **Settings**, and then select **+ Add**.

b. Name the policy **MyPolicy** and ensure **Manage** is checked. Then select **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.

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 <tseventhub>

8. **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.

   a. Navigate to the event hub you created in the previous step, **tseventhub**. Select **Shared access policies** under **Settings**, and then select **+ Add**.

   b. Name the policy **SentiPolicy** and ensure **Manage** is checked. Then select **Create**.

   c. Copy **Connection string–primary key**. Select the blue **copy** button next to the connection string.

   d. You will require:

      <**Your event hub namespace**> OR <myEventHubsNS>
       <**Your shared access policy name**> OR <SentiPolicy>
      <**Generated key**>
      <**Your event hub name** OR <sentieventhub>

9. **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.

   a. To create a Stream Analytics job, navigate to the [Azure portal](#).

   b. Select **Create a resource** > **Internet of Things** > **Stream Analytics job**.

   c. Fill out the **New Stream Analytics job** pane with the following values:

Setting	Suggested value	Description
Job name	**tsAnalytics**	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 **Use existing** 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 understanding and adjusting streaming units article.

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

10. **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.

11. 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.

a. From the Azure portal open **All resources** pane, and the **tsAnalytics** Stream Analytics job.

b. In the **Job Topology** section of the Stream Analytics job pane, select the **Outputs** option.

c. Select **+ Add** > **Power BI**. Then fill the form with the following details and select **Authorize**:

Setting	Suggested value
Output alias	sentimentanalysis
Dataset name	sentimentdb
Table name	sentiment

d. When you select **Authorize**, a pop-up window opens, and you are asked to provide credentials to authenticate to your Power BI account. Once the authorization is successful, **Save** the settings.

12. **Define a 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.

a. 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.

b. 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.

13. **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.

14. **Go back to VM and check if job to create cluster is finished**

15. **Get code from Git repository:**
    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`**`>`

16. **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 finservreg
    f. sudo docker tag timeseries finservreg.azurecr.io/timeseries
    g. sudo docker push finservreg.azurecr.io/timeseries

17. **Install the Kubernetes CLI**
    a. sudo az aks install-cli
    b. PATH=$PATH:/usr/local/bin; export PATH

18. **Connect to cluster using kubectl**
    a. sudo az aks get-credentials --resource-group myResourceGroup --name myAKSCluster

19. **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)

20. Login to gettsdata container by typing:
   a. sudo kubectl exec -it finservpod --container gettsdata -- /bin/bash

21. 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

22. Run script to push data to Azure event hub:
   (This will send pricing and sentiment data to Event Hub)

   a. /app/pushtsdata.py -i senti_data
   b. /app/pushtsdata.py -i price_data

23. Visualize output using Power BI

   Once the job succeeds, navigate to [Power BI](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"
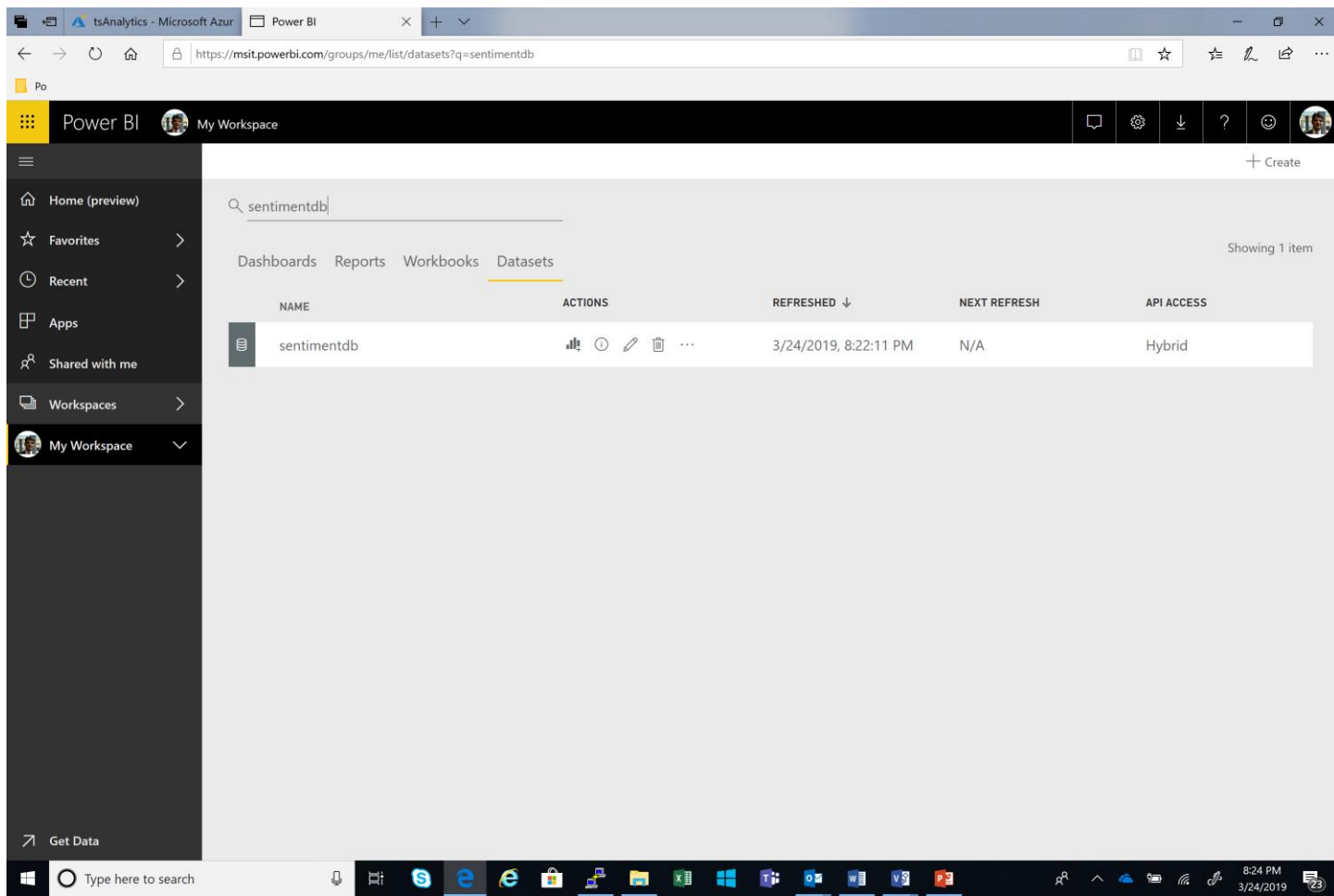
a. Click on graph icon under actions for your dataset which will open a new dashboard

b. 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:
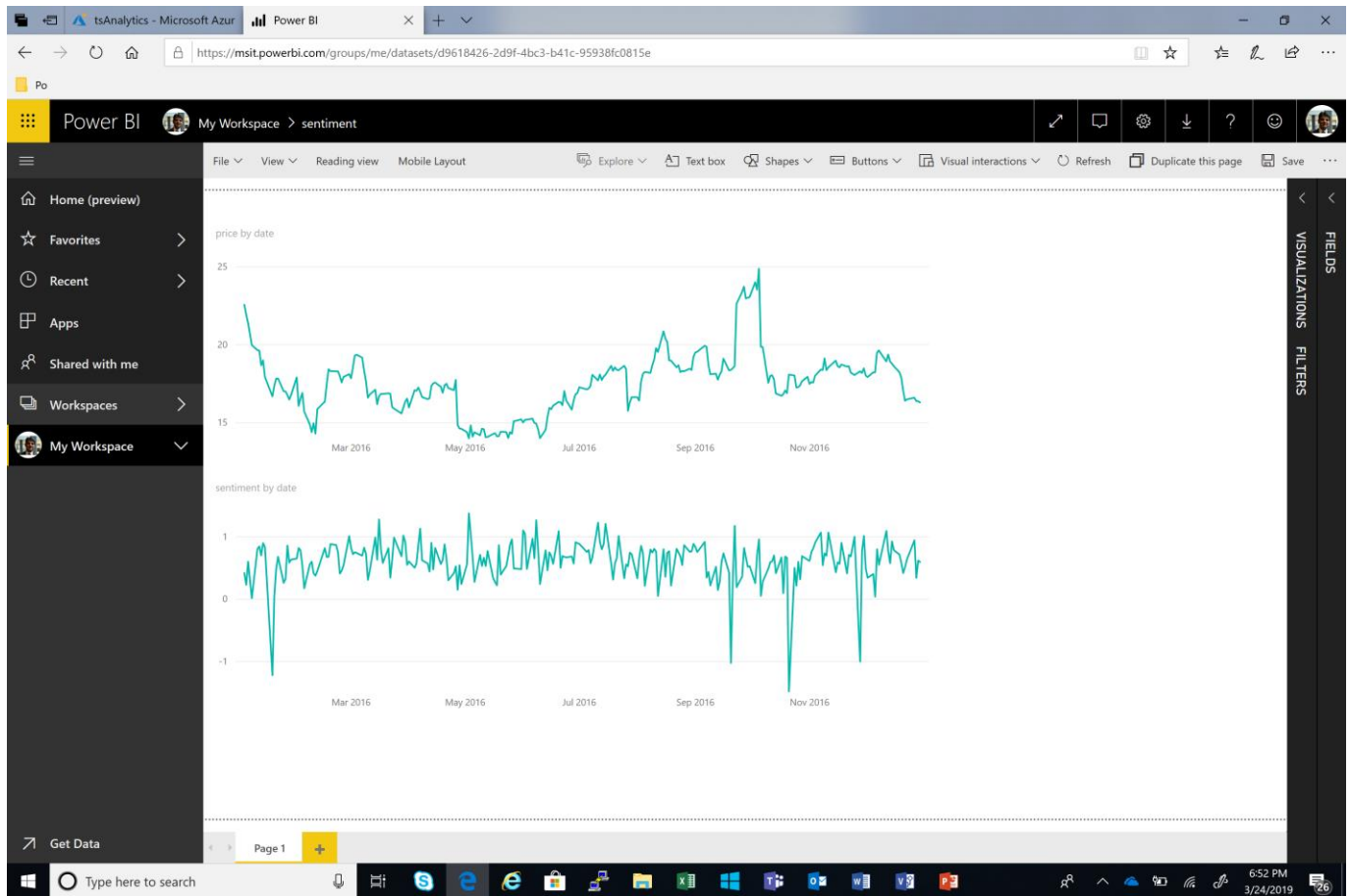
DevOPS Automation:

1. **Get repository:**

   a. Login to Github

   b. Clone below repository to your account:

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

2. **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

3. **Create Build Pipeline:**

   a. Click on pipeline -> New pipeline

   b. Select GitHub (YAML) (Not enterprise server option)

   c. Select repository that you cloned earlier

   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

4. 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 form 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:
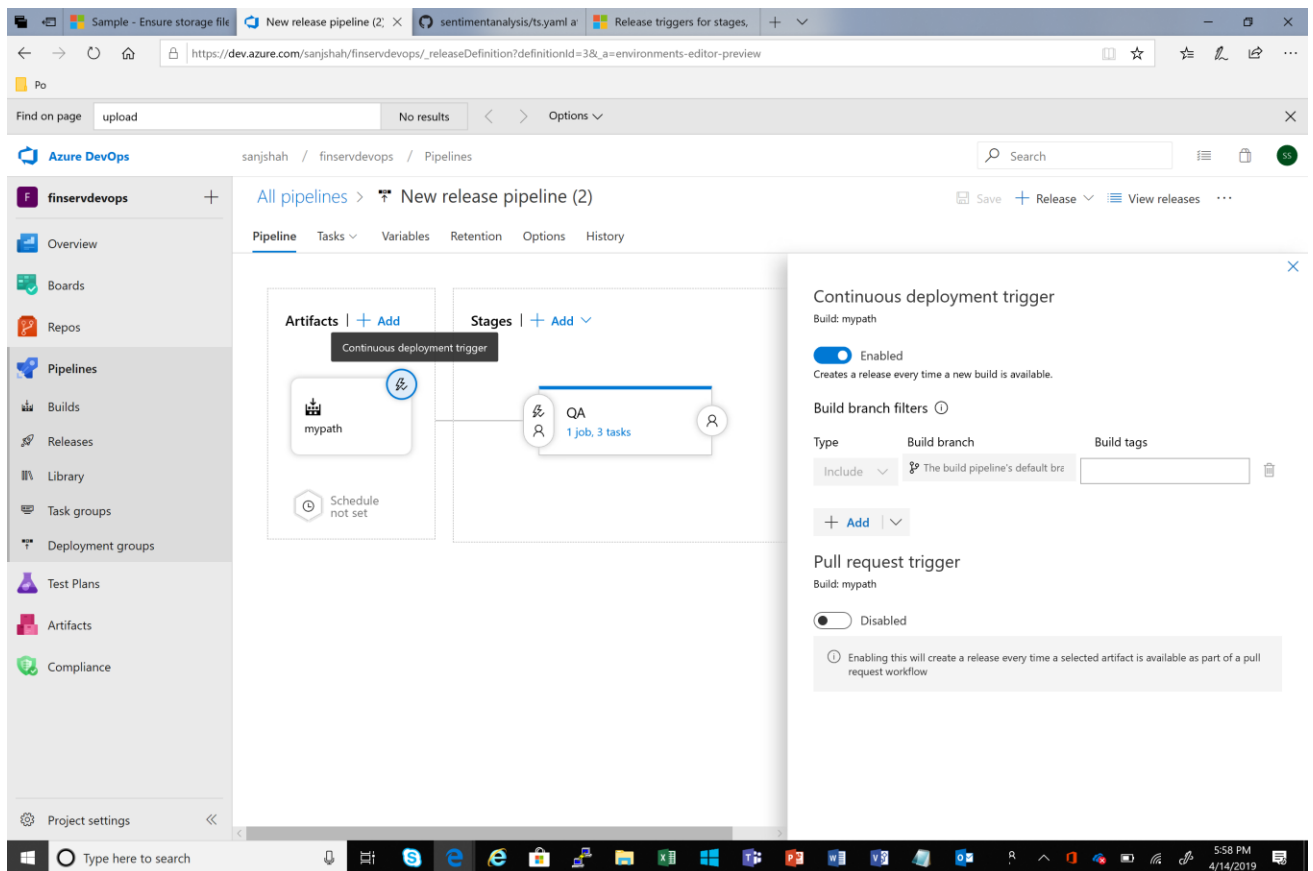
m. Enable option where is says "Create a release every time new build is available"

n. Give any tag name and Save

o. Click on "Task" from the menu on top

p. Click on Kubectl 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: finservreg.azurecr.io/timeseries

    (Use name of registry that you created earlier if it's different then finservreg)

aa. Go back to previous window and refresh Kubernetes service connection option. Drop down will show new connection you created. Select that one
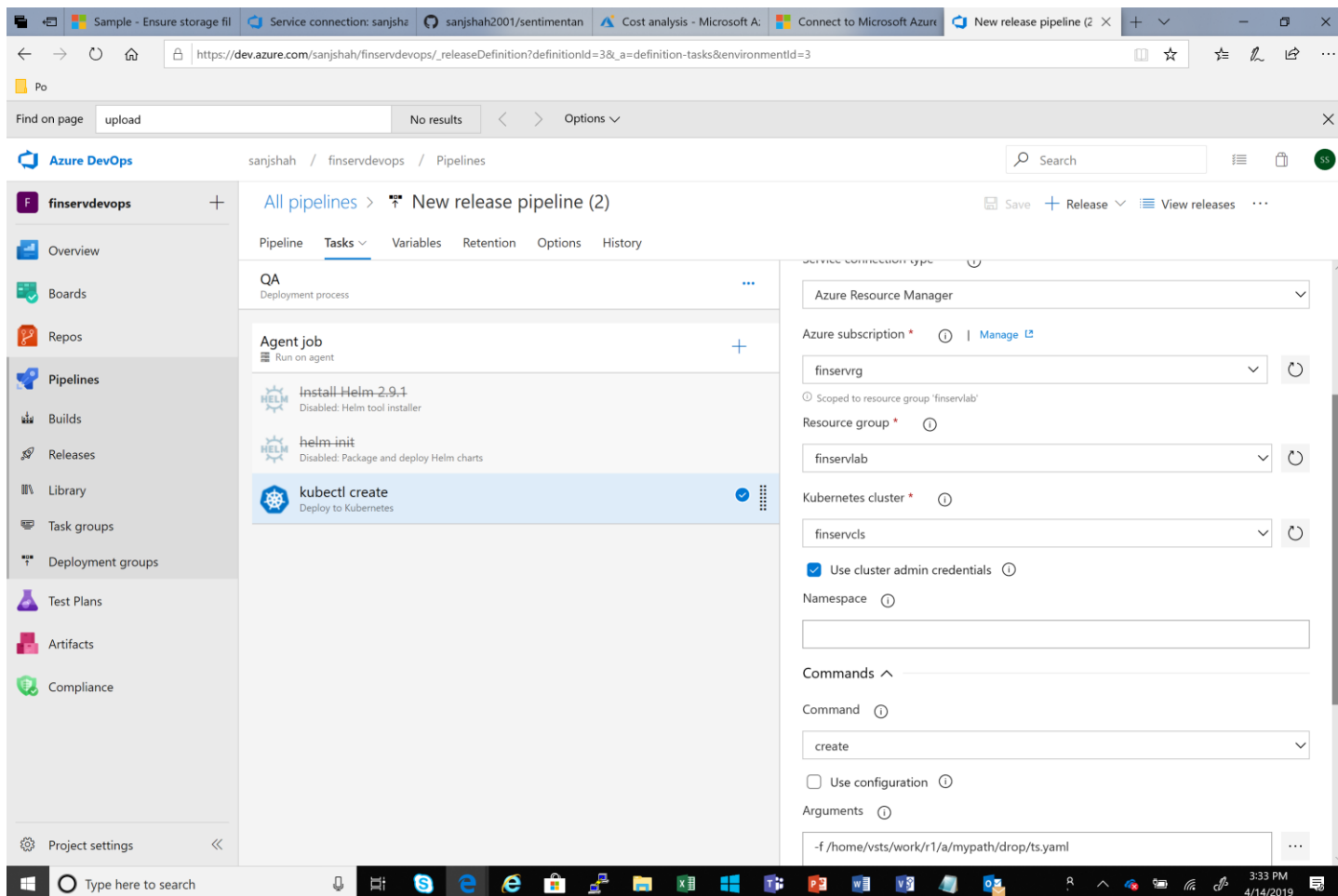
bb. Under command enter "create"

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

dd. Click save

5. Test new release

a. Go to your GIT repository

b. Edit file ts.yaml and change POD name from finservpod to finservpodv2 under metadata

c. Save and commit changes

d. Go back to Azure DevOPS build pipeline, a new build would have already triggered

e. Once finished, check release pipeline, which is automatically triggered upon successful finish of build job

f. Once release job is finished successfully, login to your VM created in previous exercise

g. 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