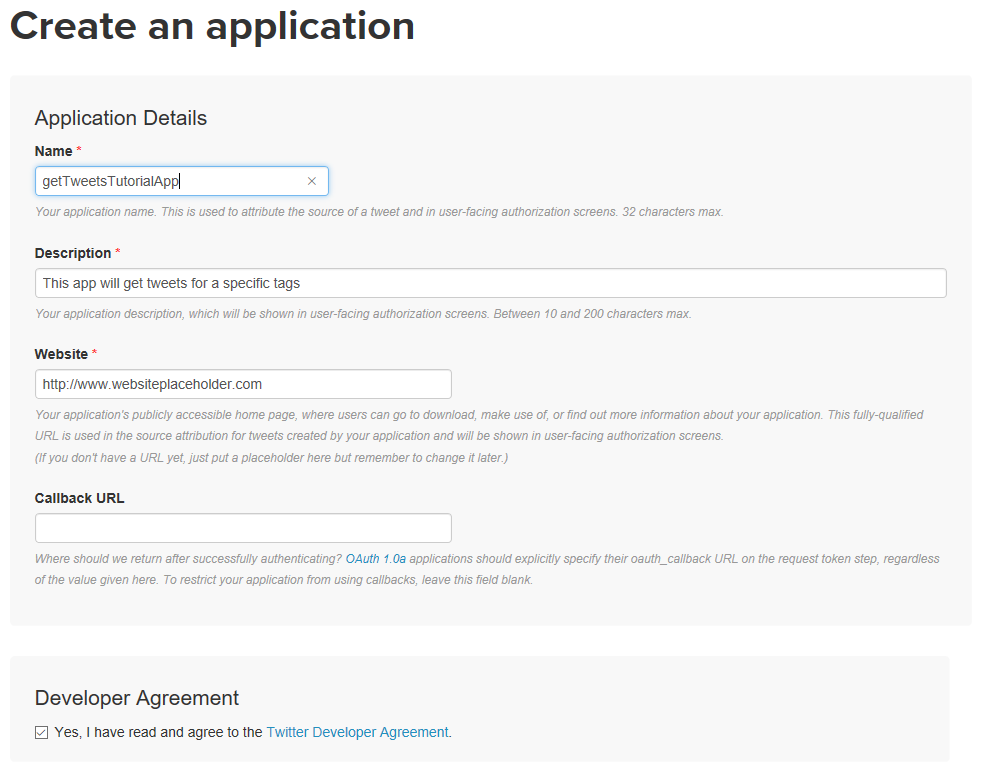
Let's setup Twitter application to retrieve credential for making API calls to get tweets for a specific tag.

1) Go to twitter <https://apps.twitter.com/>

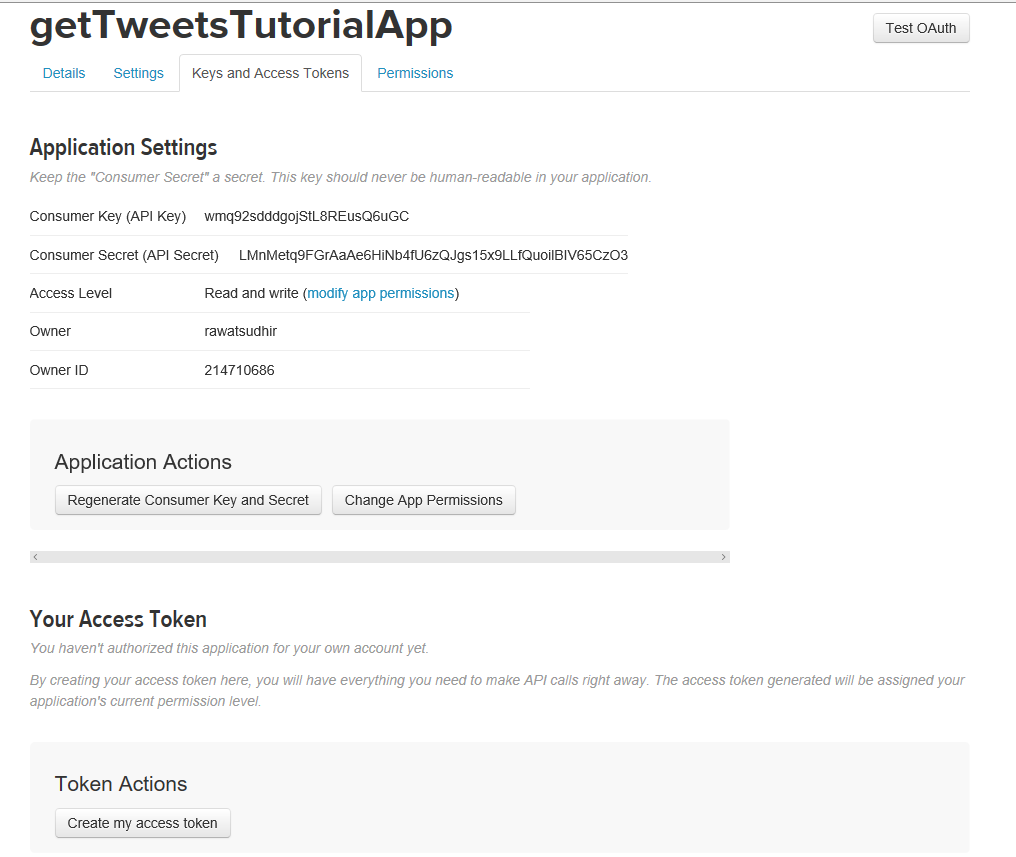
2) Click “Create New App”

3) Fill out information



4) Click “Create New Application”

5) Once application is created. Go to “Keys and Access Tokens”



6) Click “Create my access Token” and save it to use it future step.

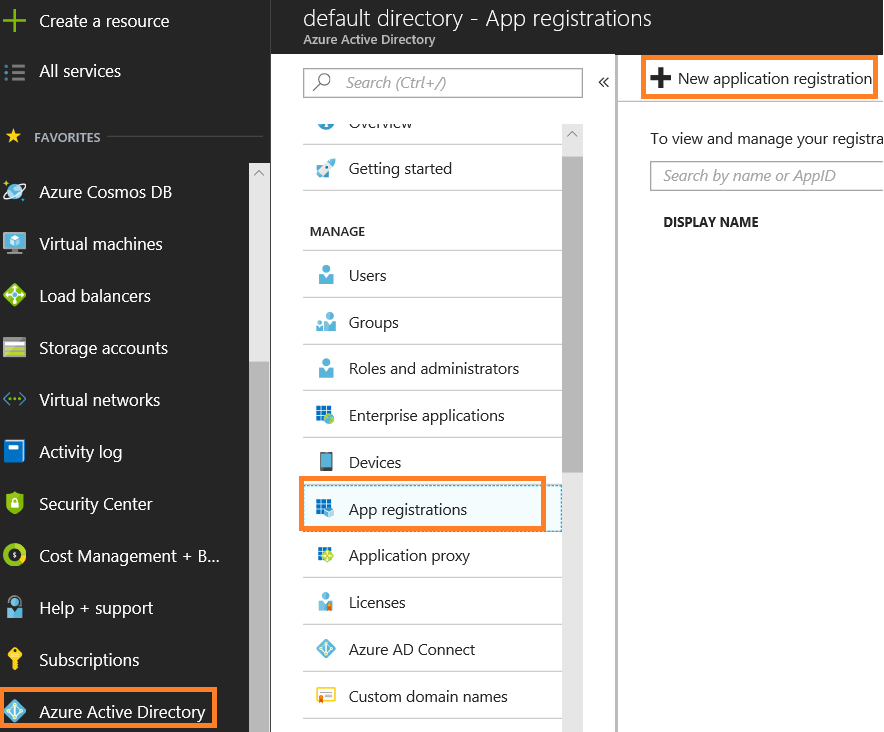
Let's setup Azure active directroy app to get token and access Azure Key Vault

1) Switch to <https://portal.azure.com>

2) Click Azure Active Directory from left side

3) Click App registrations

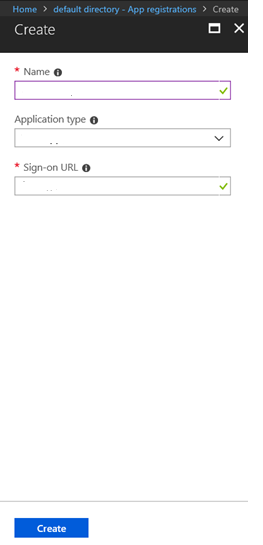
4) Click New application registration



5) Provide Name

6) Select Application type (Web app /api)

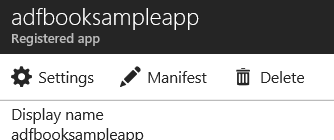
7) Provide Sign-on URL. Need not to be a site which exists (you can put <http://test1.adventureworks.com>)



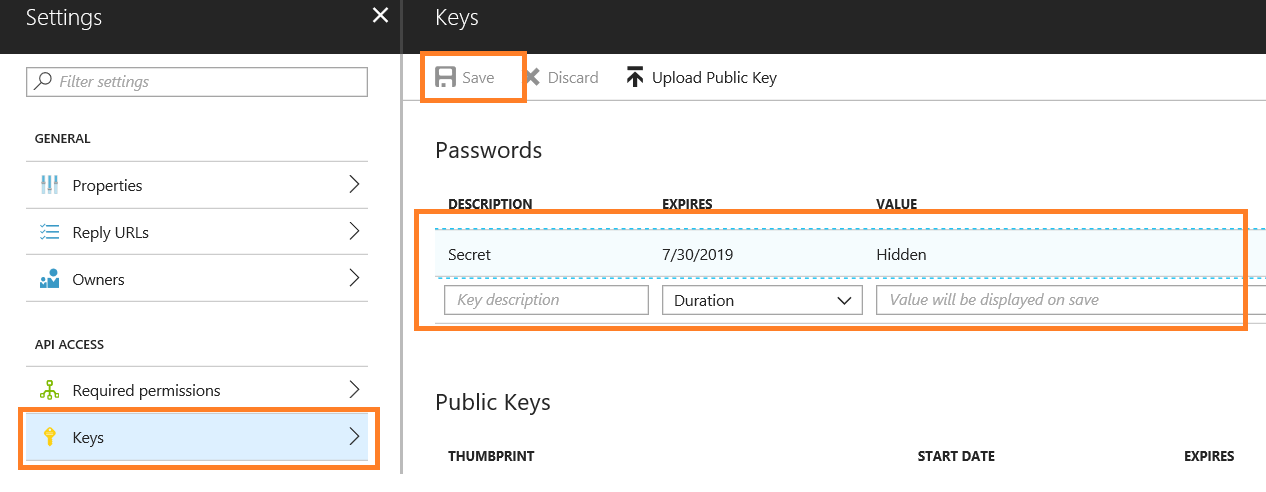
8) Click Create

9) Once App is registered, click on it

10) Click Settings

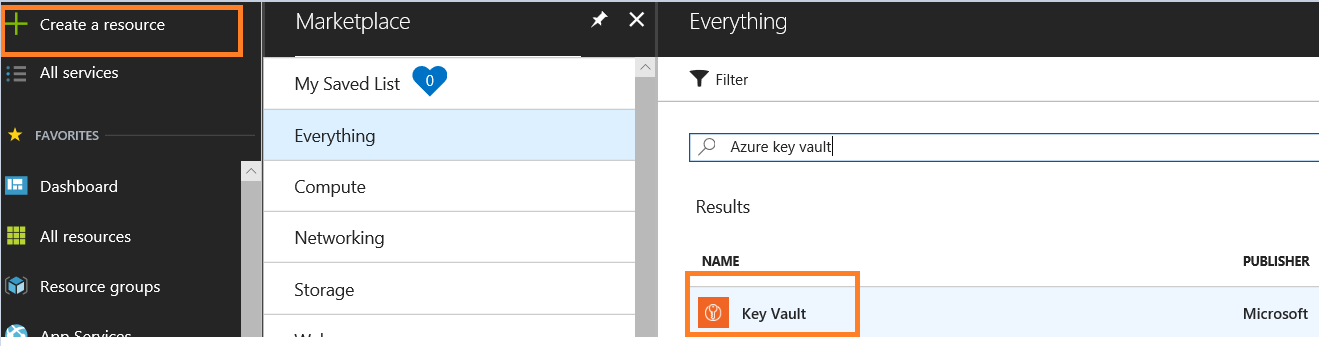


11) Click Keys, In Password section provide Description and Expires. Click Save and it will show up the password. Copy it in a notepad.



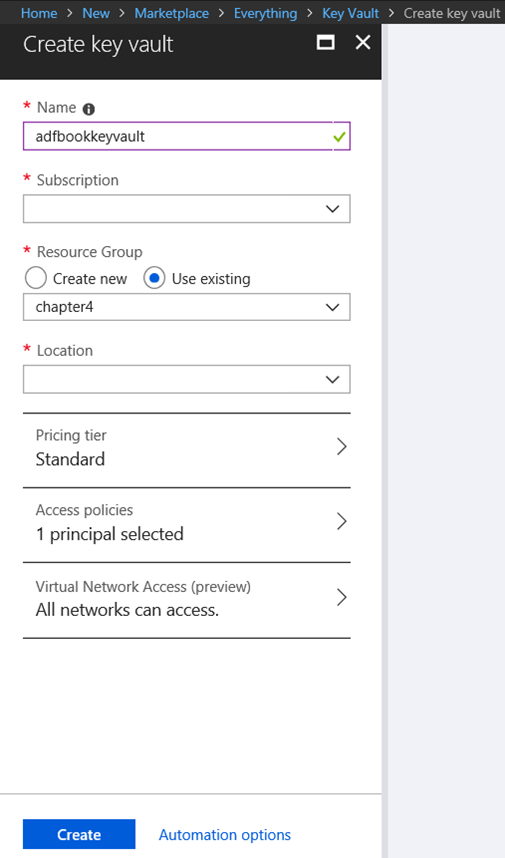
Let's setup Azure Key Vault

1) Click Create a resource and search for Azure Key Vault



2) Select Key Vault and click Create

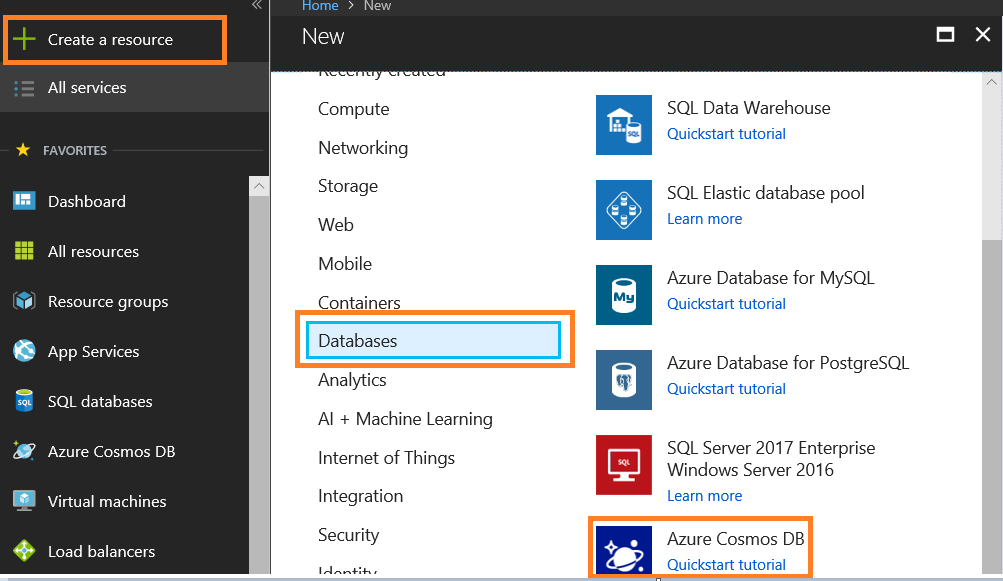
3) Enter Name, Susbcription, Resource Group, Location, let default selected for Pricing tier, Access Policies ad Virtual Network Access



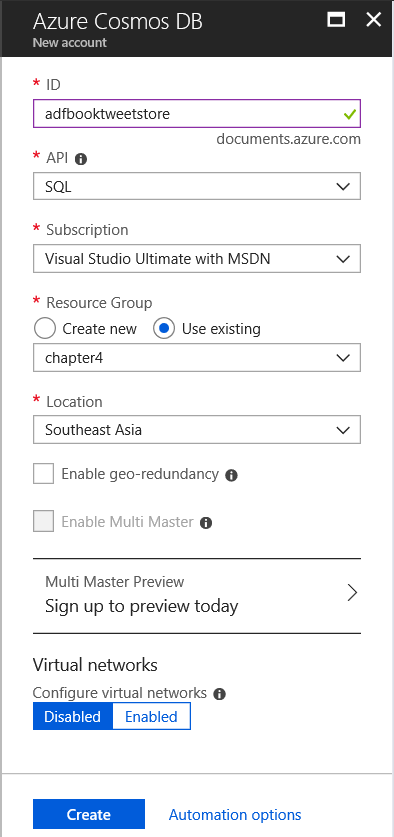
4) Click Create

Let's setup Azure cosmosDB account to store tweets.

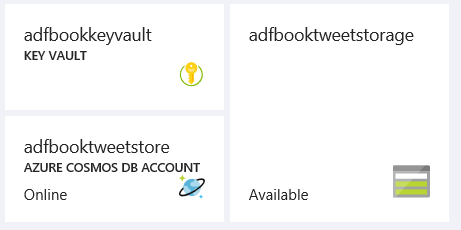
1) Click Create a resource, Databases and Azure Cosmos DB



2) Provide ID, API, Subscription (your subscription), Resource Group (select exisiting or new), Location, let other configuration be default and click Create.

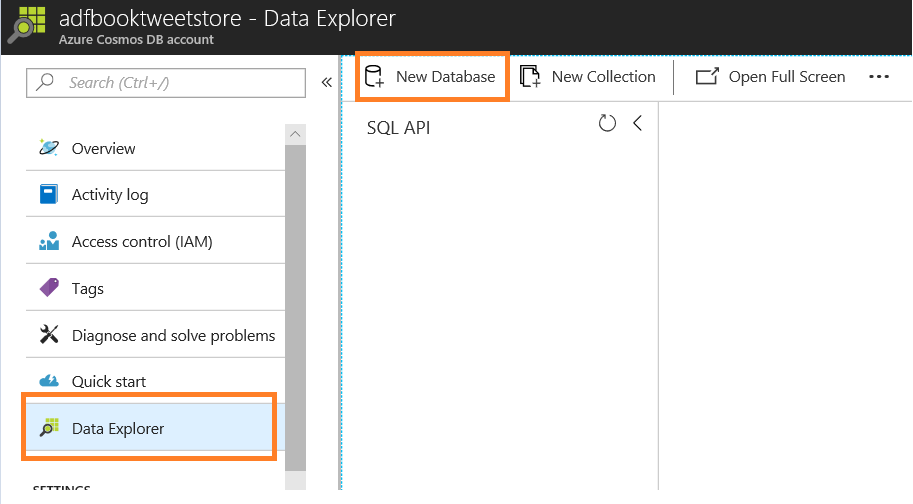


3) At the end, you will see three services added in your azure subscription

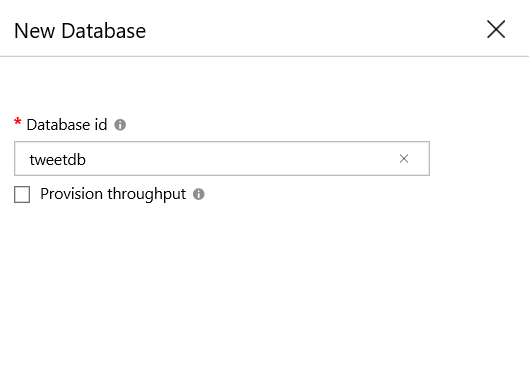


4) Click AZURE COSMOSDB ACCOUNT (created in earlier step)

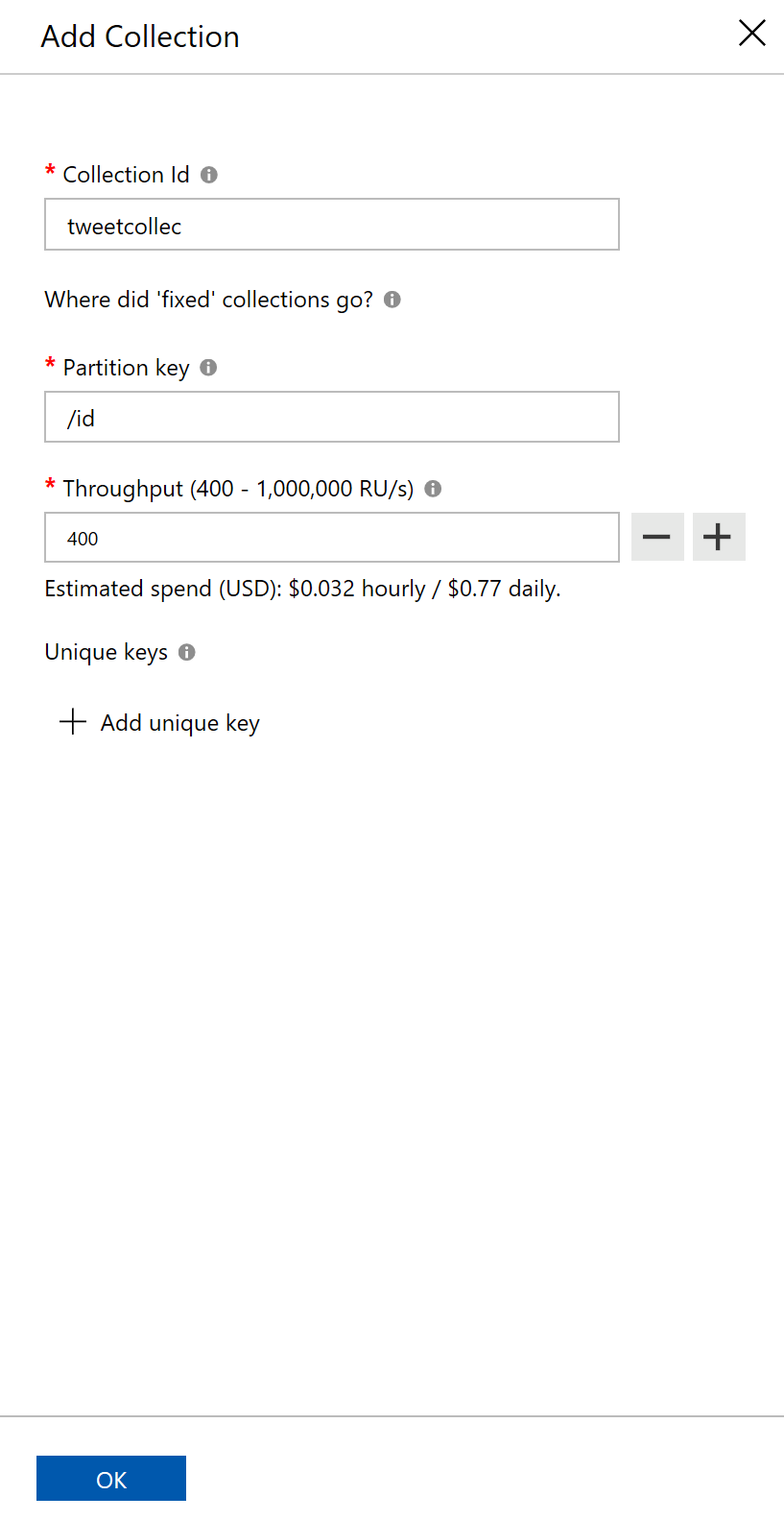
5) Click Data Explorer and New Database



6) Provide Database id and click OK



7) Click New Collection



8) Select Use existing and choose database id created in previous step

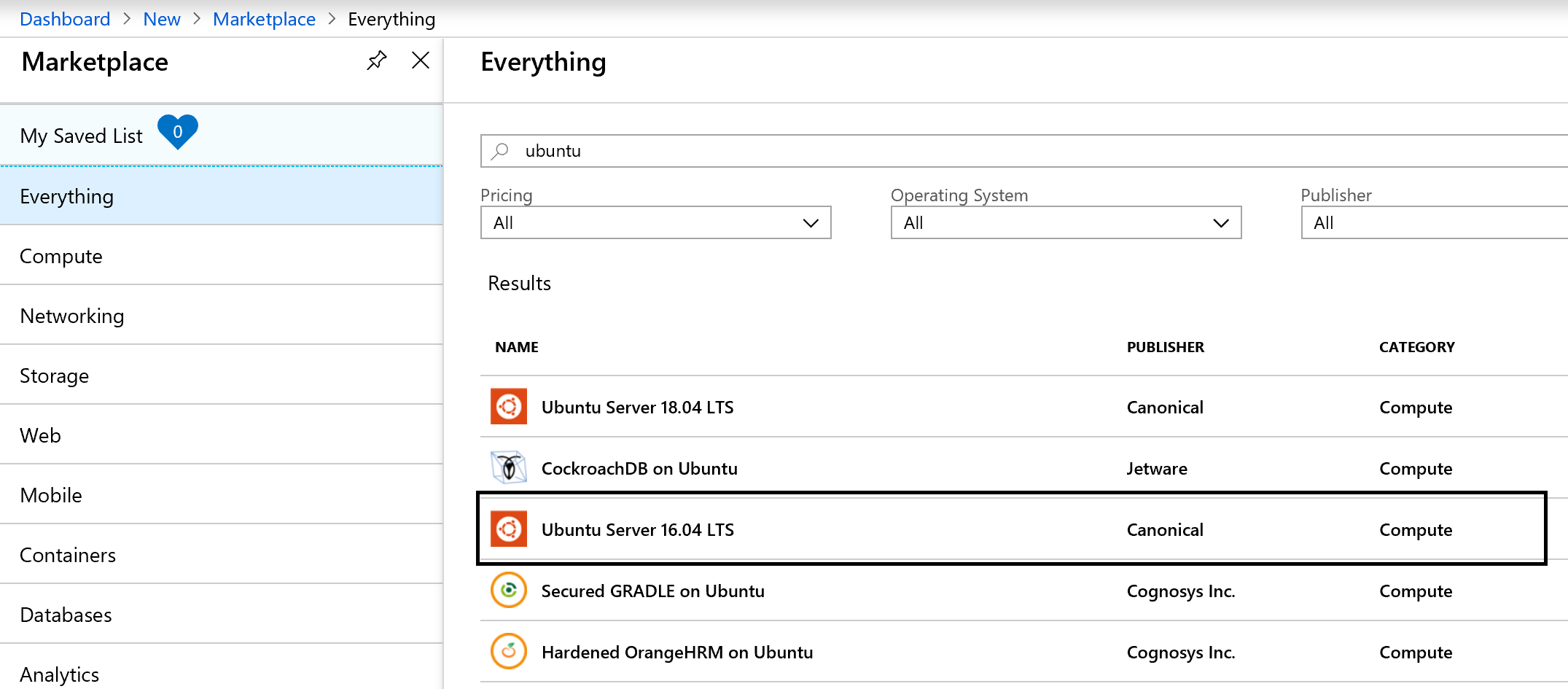
9) Provide Collection id (any name)

10) Provide Throughput

11) Click OK

Let’s add Ubuntu VM which contains all libraries mentioned later which is required to run application.

1. Create a Ubuntu VM



1. Once VM is setup, SSH on the VM, install python and below libraries

sudo apt-get install python-pip

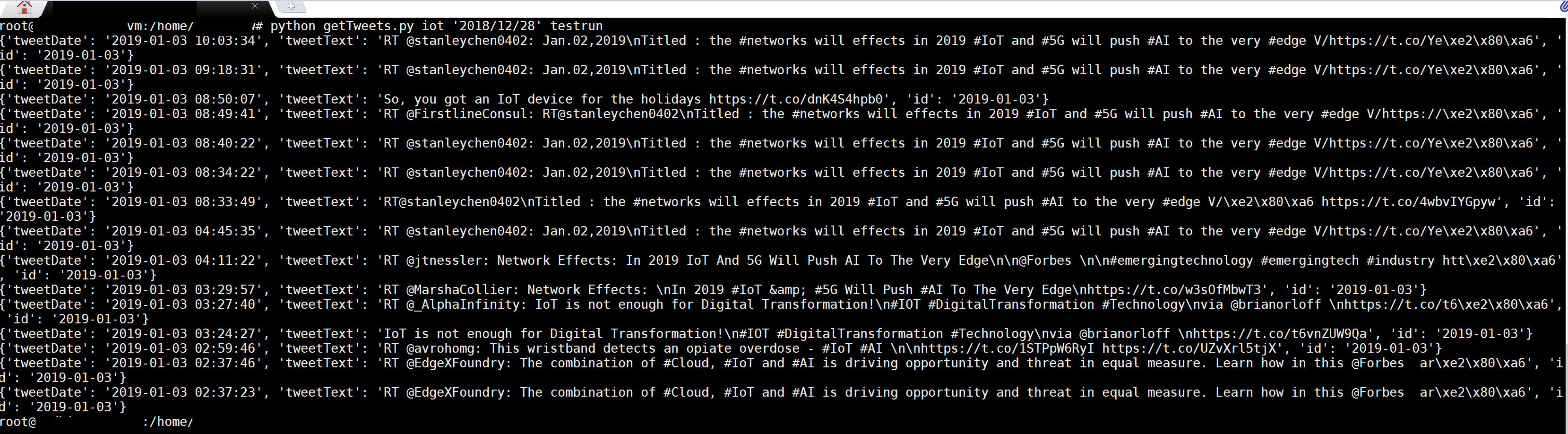
sudo su

pip install tweepy

pip install pydocumentdb --Install client library

pip install azure-keyvault --Install client library

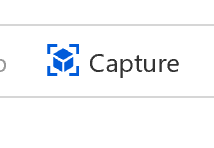
From the repository, copy getTweets.py change the secerts and execute the script to make sure there is no issue with library installation.



Once application works successfully, run below command

sudo waagent -deprovision+user

Goto Azure portal and click Capture to create a image

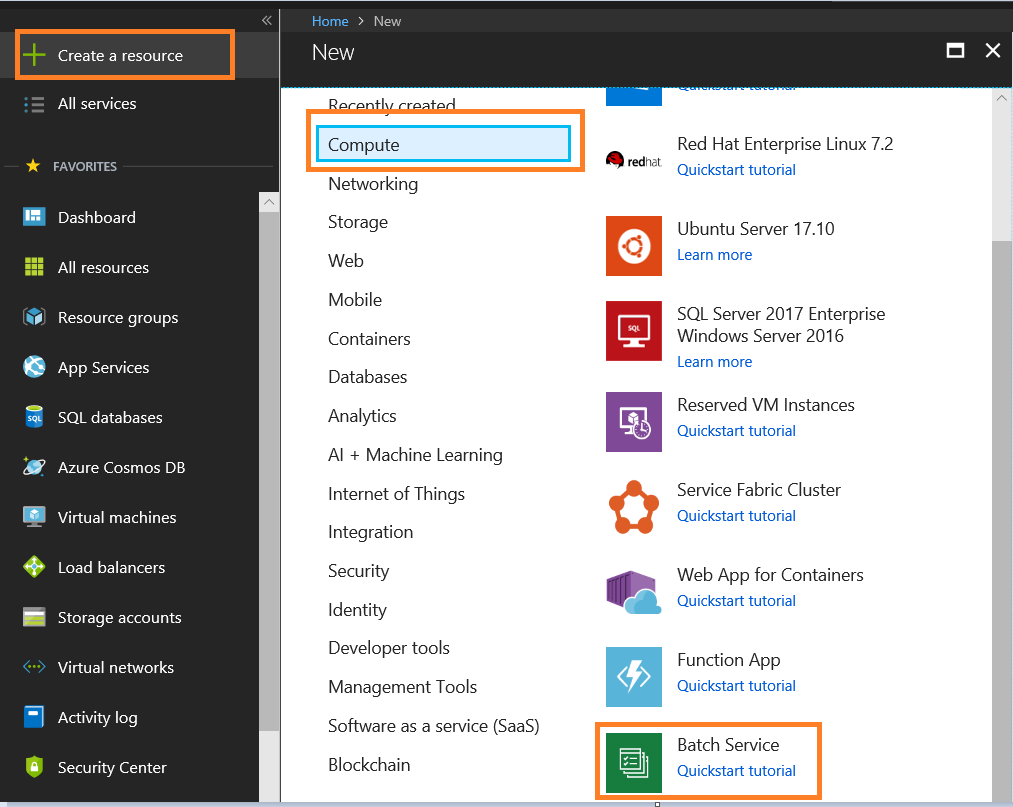


Let's setup Azure Batch service

1) Click Create a resource

2) Click Compute

3) Batch Service



4) Provide Account name

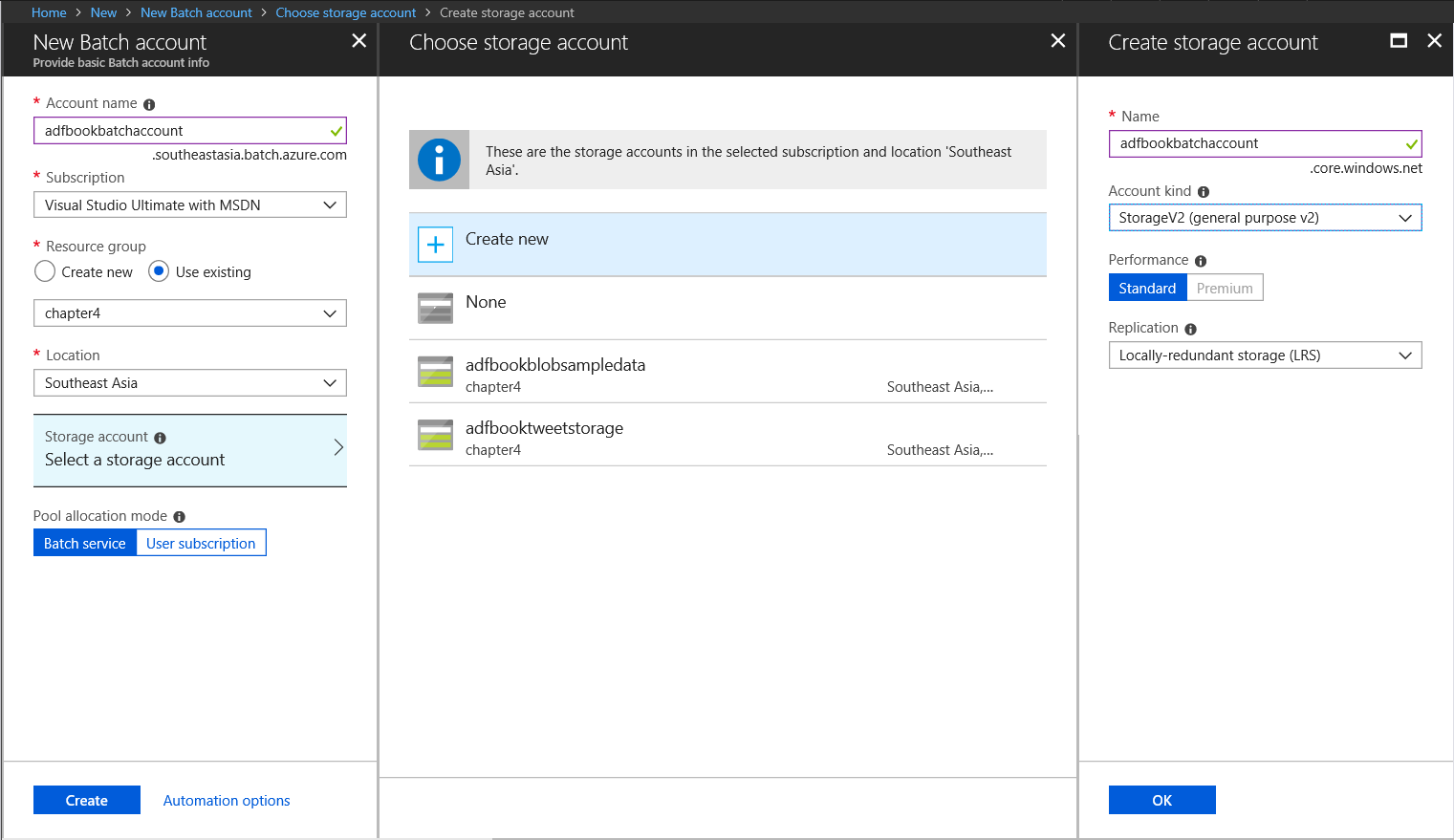
5) Select Subscription

6) Select Resource group or create new

6) Select Location

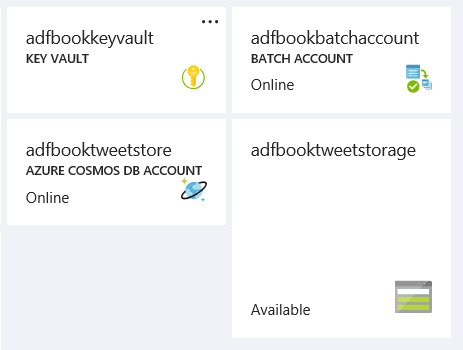
7) Storage account, create new or choose existing one

8) Select Batch service as Poll allocation mode



9) Click Create

10) Once batch services is setup, you will see following services on azure dashboad (if you choose to pin to dashboard)

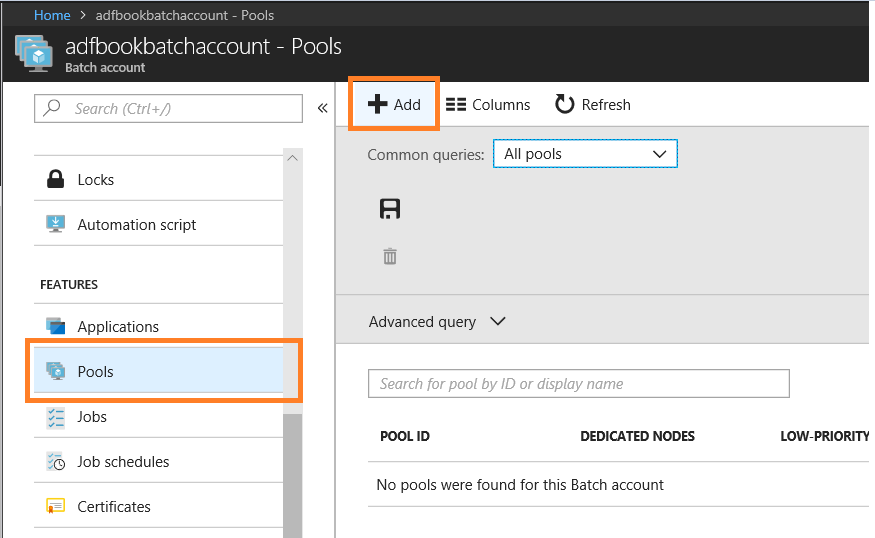


Let’s Add Pools

1) Click Azure batch service

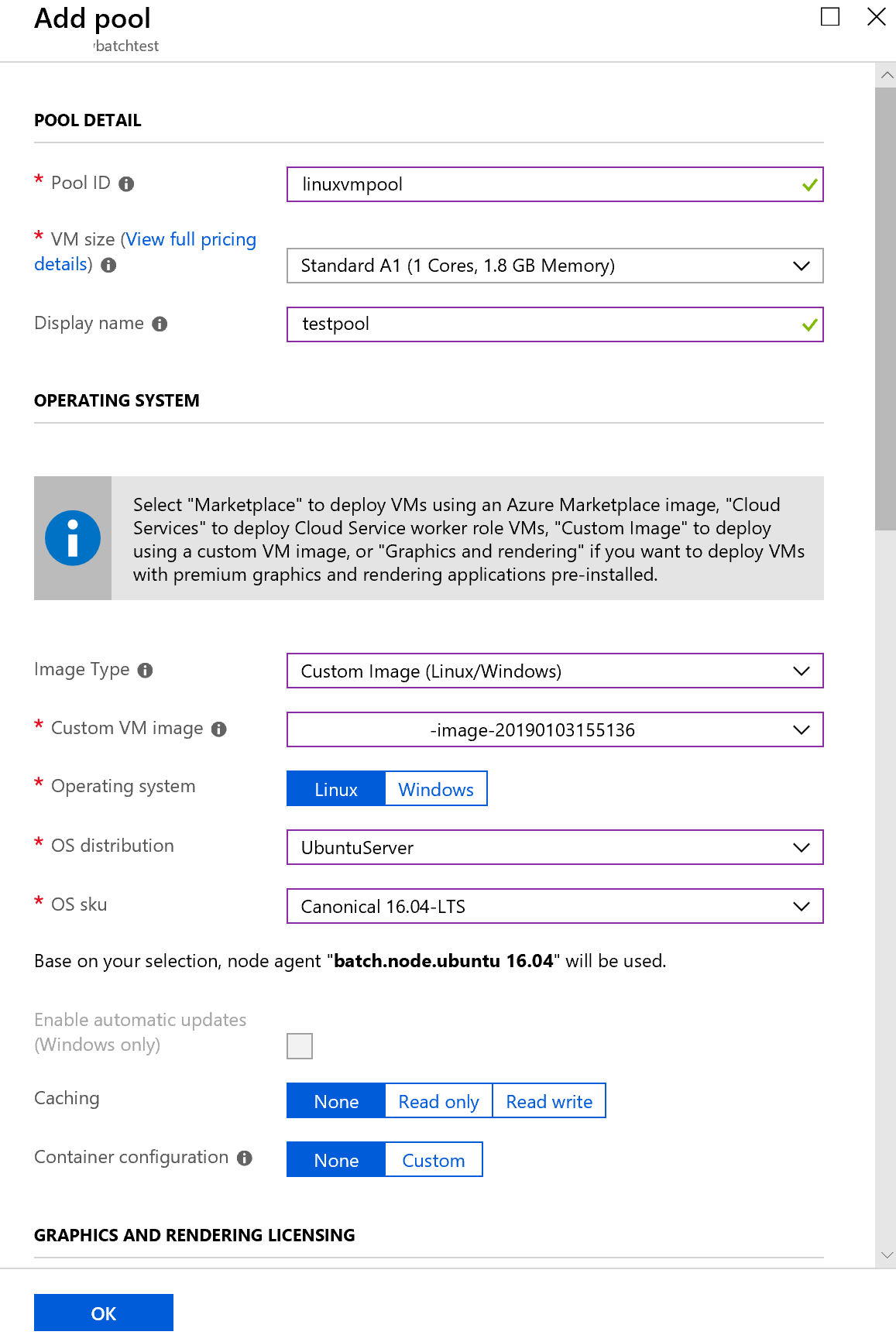
2) Click Pools

3) Click Add



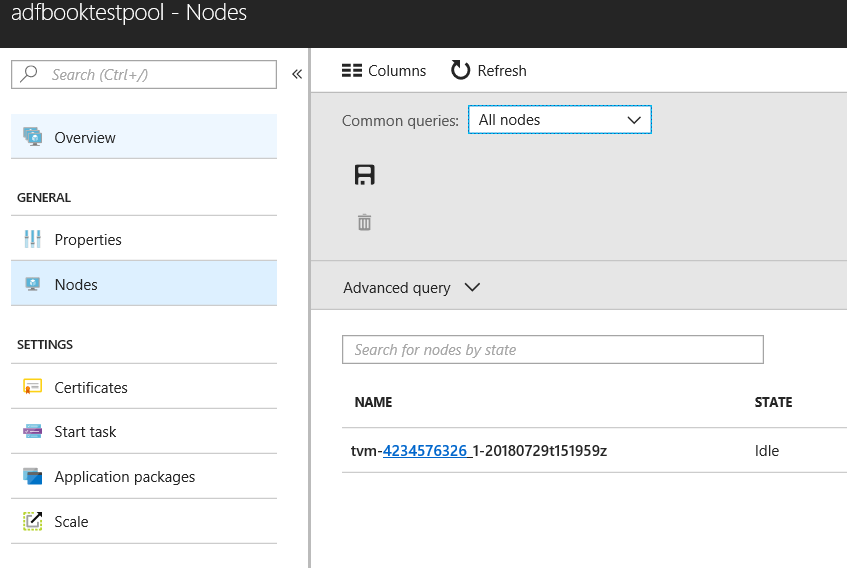
4) Add following information

|  |  |
| --- | --- |
| Property | Value |
| Pool ID | Name of the pool |
| Node pricing tier | Standard A1 (1 Cores, 1.8 GB). This demo is not compute extensive job hence basic compute works however you can go for higher compute |
| Display name | Description (optional) |
| Image Type | Custom Image (Linux/Windows) |
| Custom VM Image | Select custom image created earlier |
| Operating System | Linux |
| OS Distribution | UbuntuServer |
| OS version | Canonical 16.04-LTS |
| Caching | None |
| Container configuration | None |
| Metered licenses for rendering | Don't change. Not doing any rendering for this demo |
|  |  |
| Mode | Fixed. This service allows you choose Auto scale option which allow service to increase/decrease compute based on formula. This help organzation not to worry about scaling out and scaling in |
| Taget dedicated nodes | Set it to 1 |
| Low priority nodes | 0. Not using in this demo. This option reduces compute cost. Low priority nodes take advantages of surplus capacity in Azure. You use low priority nodes where job consumes less time or for batch processing. The tradeoff of using such option is VMs may not be available for allocation or preempted at any time, depending on available capacity. |
| Resize Timeout | 15 minutes. How long process wait for resizing |
| Start task | Disabled. Specify task which needs to run first when vm added to the pool |
| Max tasks per node | 1. You can specifiy what is the maximum number of tasks can be run on VM. Be cautious about VM size you choosen. |
| User accounts | Default |
| Task scheduling policy | Pack. Defines how tasks get distributed between Vms in the pool |
| Inter-node communication | No |
| Application Package | 0. Incase your application required packages to run it successfully. |
| Certificates | 0 |
| Pool endpoint configuration | Default |
| Network configuration | Default. Not required for this demo |
| Subnet | Default. Not required for this demo |



5) Click OK

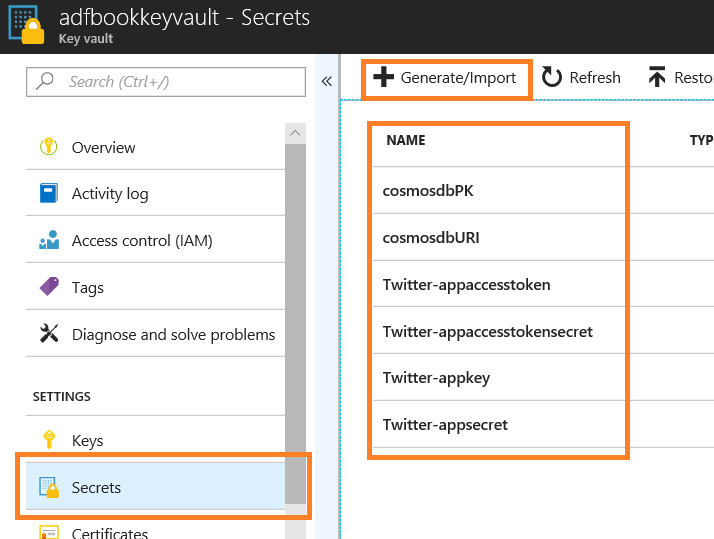
6) Once pool is created, click Pool (created in previous step) and Nodes to make sure VM is created



Let's store credential on Azure Key Valut and give access to Azure AD app

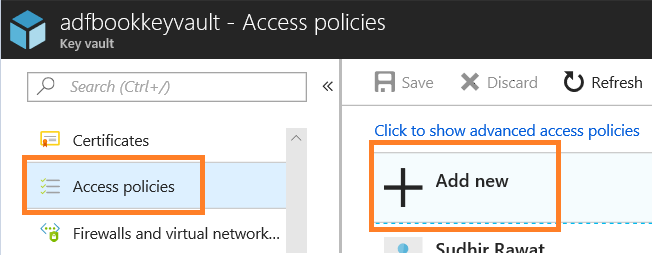
1)Switch to Azure Key Vault

2) Add all secrets like Azure cosmodb details and Twwitter API details on Azure Key Vault. Swtich to respective services to capture the keys



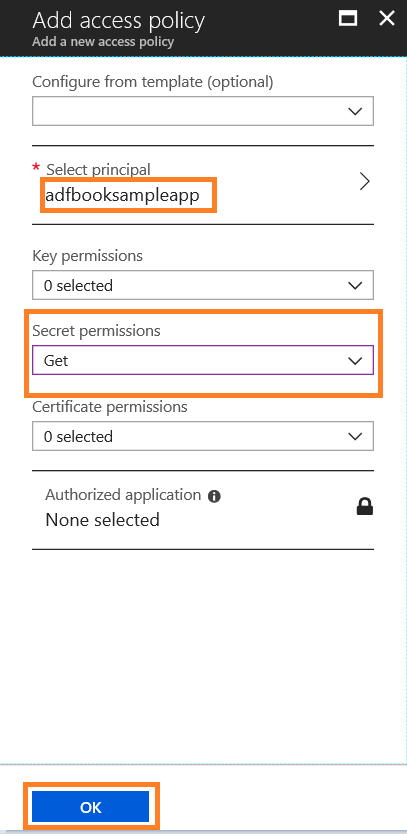
3) In Azure Key Vault, click Access policies

4) Click Add New



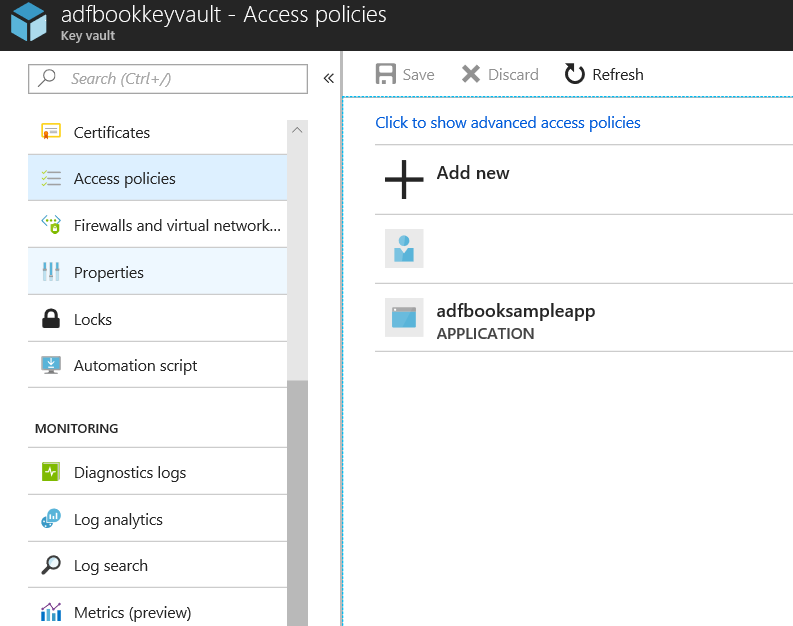
5) Select Prinicipal. This is the application registered in Azure Active Directory

6) Select Get as Secret permission



7) Click OK

8) Click Save



Now environment is set.

Let's look onto the python code. Upload the code to Azure blob storage

import tweepy

import sys

import pydocumentdb.documents as documents

import pydocumentdb.document\_client as document\_client

import pydocumentdb.errors as errors

from azure.keyvault import KeyVaultClient, KeyVaultAuthentication

from azure.common.credentials import ServicePrincipalCredentials

class IDisposable:

""" A context manager to automatically close an object with a close method

in a with statement. """

def \_\_init\_\_(self, obj):

self.obj = obj

def \_\_enter\_\_(self):

return self.obj # bound to target

def \_\_exit\_\_(self, exception\_type, exception\_val, trace):

# extra cleanup in here

self = None

credentials = None

def auth\_callback(server, resource, scope):

credentials = ServicePrincipalCredentials(

client\_id = 'XXXXXXXXXXXXXXX', #Azure AD APP Application ID

secret = 'XXXXXXXXXXXXXXXXXX', #Secret

tenant = 'XXXXXXXXXXXXXXXXXXXXXXXXXX', #Azure AD Directory ID

resource = "https://vault.azure.net"

)

token = credentials.token

return token['token\_type'], token['access\_token']

def insertintoCosmosDB(cdbhost, cdbmasterkey, tweetDate, tweetText):

tweetmessage = {'tweetDate': str(tweetDate),'id' : str(tweetDate).split()[0], 'tweetText': tweetText}

\_database\_link = 'dbs/tweetdb'

\_collection\_link = \_database\_link + '/colls/tweetcollec'

print (tweetmessage)

client = document\_client.DocumentClient(cdbhost, {'masterKey': cdbmasterkey})

#with IDisposable(document\_client.DocumentClient(cdbhost, {'masterKey': cdbmasterkey} )) as client:

try:

client.CreateDocument(\_collection\_link, tweetmessage)

except errors.DocumentDBError as e:

if e.status\_code == 409:

pass

else:

raise errors.HTTPFailure(e.status\_code)

def main():

# Twitter application key

client = KeyVaultClient(KeyVaultAuthentication(auth\_callback))

\_appkey = client.get\_secret("https://XXXXXXX.vault.azure.net/", "TwitterAPIKey", "XXXXXXXXXXXXXXXXXXX") # KeyVault URL, Secret, Version

\_appsecret= client.get\_secret("https://XXXXXXXX.vault.azure.net/", "TwitterAPISecretKey", "XXXXXXXXXXXXXXXX") # KeyVault URL, Secret, Version

\_appaccesstoken = client.get\_secret("https://XXXXXXXX.vault.azure.net/", "TwitterAccessToken", "XXXXXXXXXXXXXXXXXXXXXX") # KeyVault URL, Secret, Version

\_appaccesstokensecret = client.get\_secret("https://XXXXXXXXX.vault.azure.net/", "TwitterAccessTokenSecret", "XXXXXXXXXXXXXXXXX") # KeyVault URL, Secret, Version

\_tweetTag= sys.argv[1] # like Azure

\_tweetReadSince= sys.argv[2] #date from when you want to read tweets like '2018/12/28'

\_RandomId = sys.argv[3] #Azure Data Factory Pipeline ID 'testrun'

# CosmosDB Credential

\_cdbhost = client.get\_secret("https://XXXXXXXXXXX.vault.azure.net/", "cosmosdbURI", "XXXXXXXXXXXXXXXXXXXXXXXX") # KeyVault URL, Secret, Version

\_cdbmasterkey = client.get\_secret("https://XXXXXXXXXXXXXX.vault.azure.net/", "cosmosdbPK", "XXXXXXXXXXXXXXXX") # KeyVault URL, Secret, Version

#hashtag, tweetreadsince, filename includes pipeline id,

auth = tweepy.OAuthHandler(\_appkey.value, \_appsecret.value)

auth.set\_access\_token(\_appaccesstoken.value, \_appaccesstokensecret.value)

tweetapi = tweepy.API(auth,wait\_on\_rate\_limit=True)

for tweet in tweepy.Cursor(tweetapi.search,q=\_tweetTag,lang="en", since=\_tweetReadSince).items(15):

try:

if tweet.text.encode('utf-8') != '' :

insertintoCosmosDB (\_cdbhost.value, \_cdbmasterkey.value, tweet.created\_at,tweet.text.encode('utf-8'))

except errors.DocumentDBError as e:

if e.status\_code == 409:

pass

else:

raise errors.HTTPFailure(e.status\_code)

print("Error while fetching and storing tweets!!!")

break

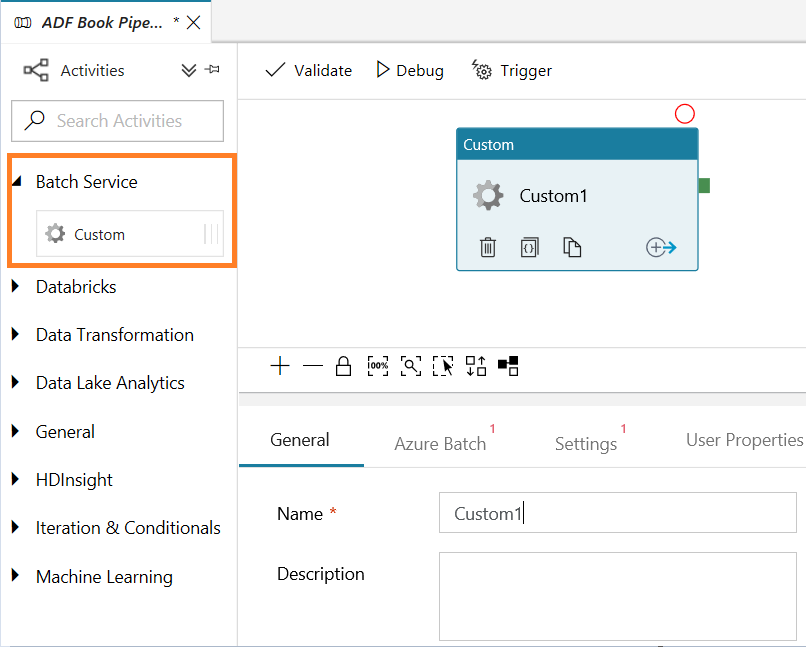
if \_\_name\_\_ == "\_\_main\_\_":

main()

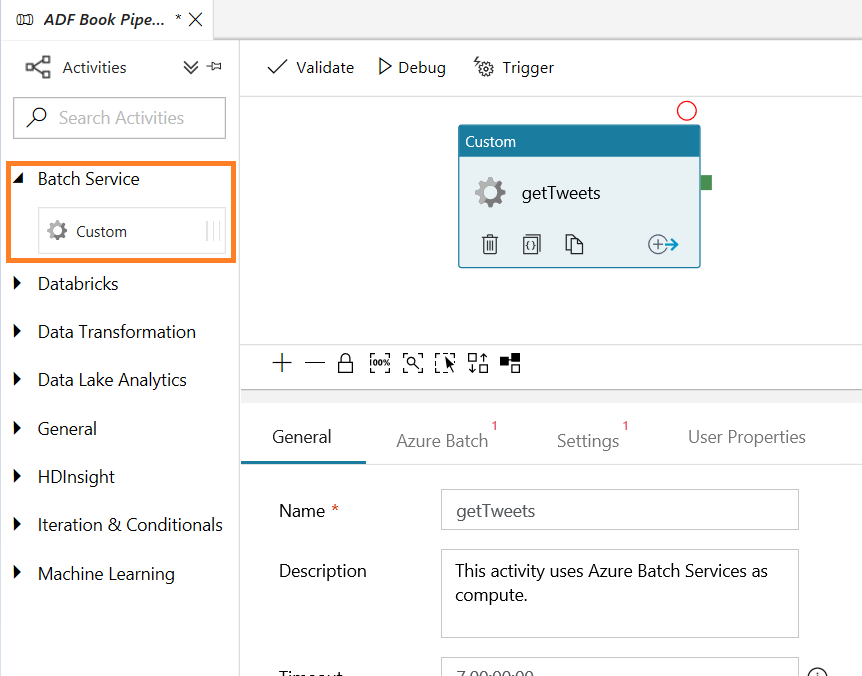
Let's setup Azure Data Factory

1) Switch to Azure Data Factory "Author & Monitor" UI

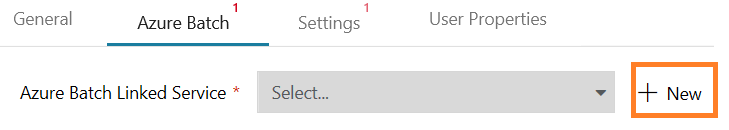
2) Drag and drop Custom Acvitity in designer



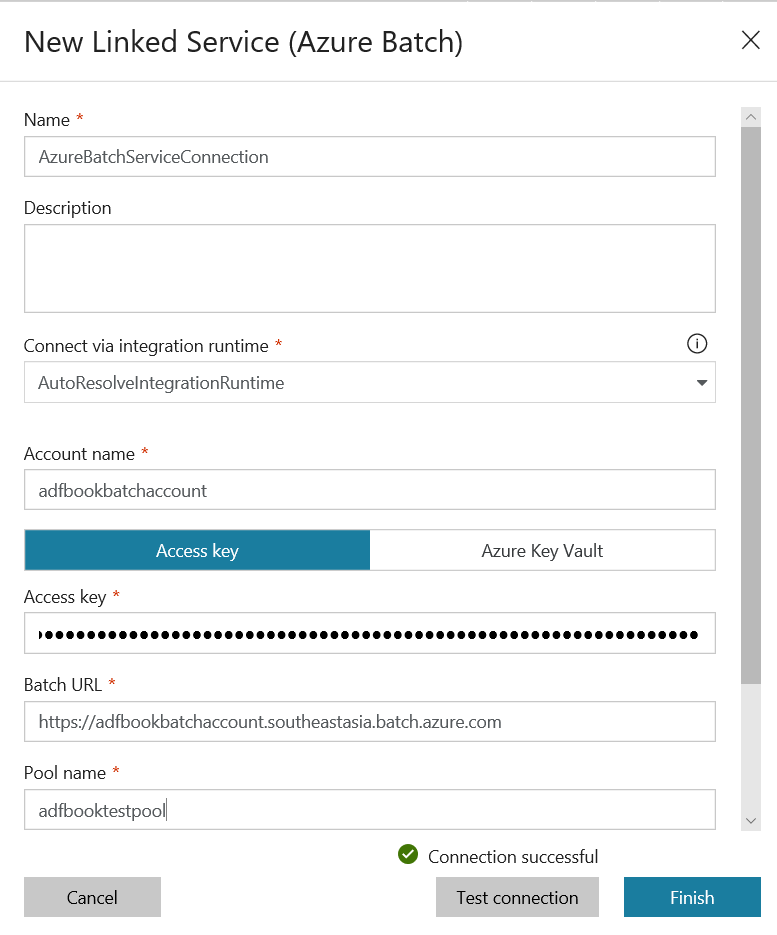
3) Provide Name and Description to the activity



4) In Azure Batch tab, click New



5) Provide Azure Bacth account details. Retrieve all information from Azure Batch account services

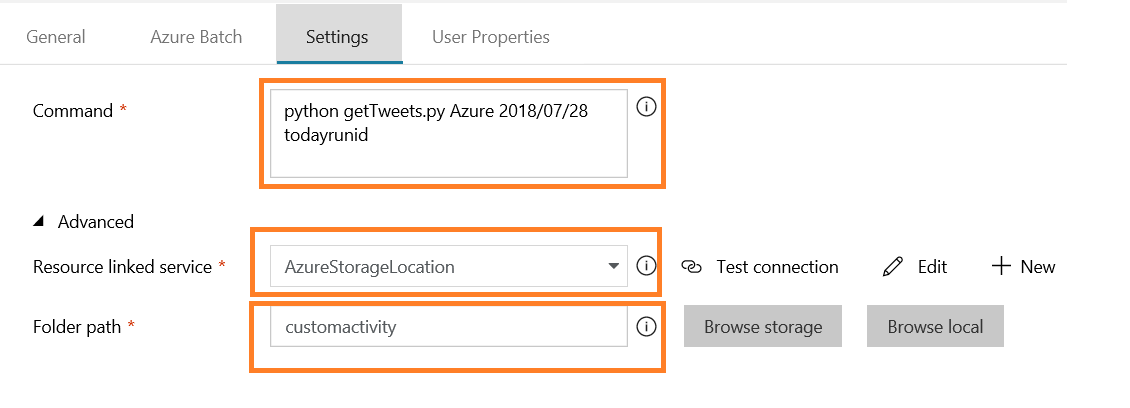


6) Click Finish

7) Click Settings

8) Provide command in Command Text Area

|  |  |
| --- | --- |
| Parameter | Description |
| getTweets.py | getTweets is name of the python program to execute |
| Azure | Get tweets for given Hash Tag |
| 2018/12/28 | Read since |
| todayrunid | Any text value to be passed incase of testing or debugging purpose |



9) Select Resourcec linked service. Azure storage location where python code is uploaded

10) Select Folder path. Folder location where python code is uploaded

11) Publish All

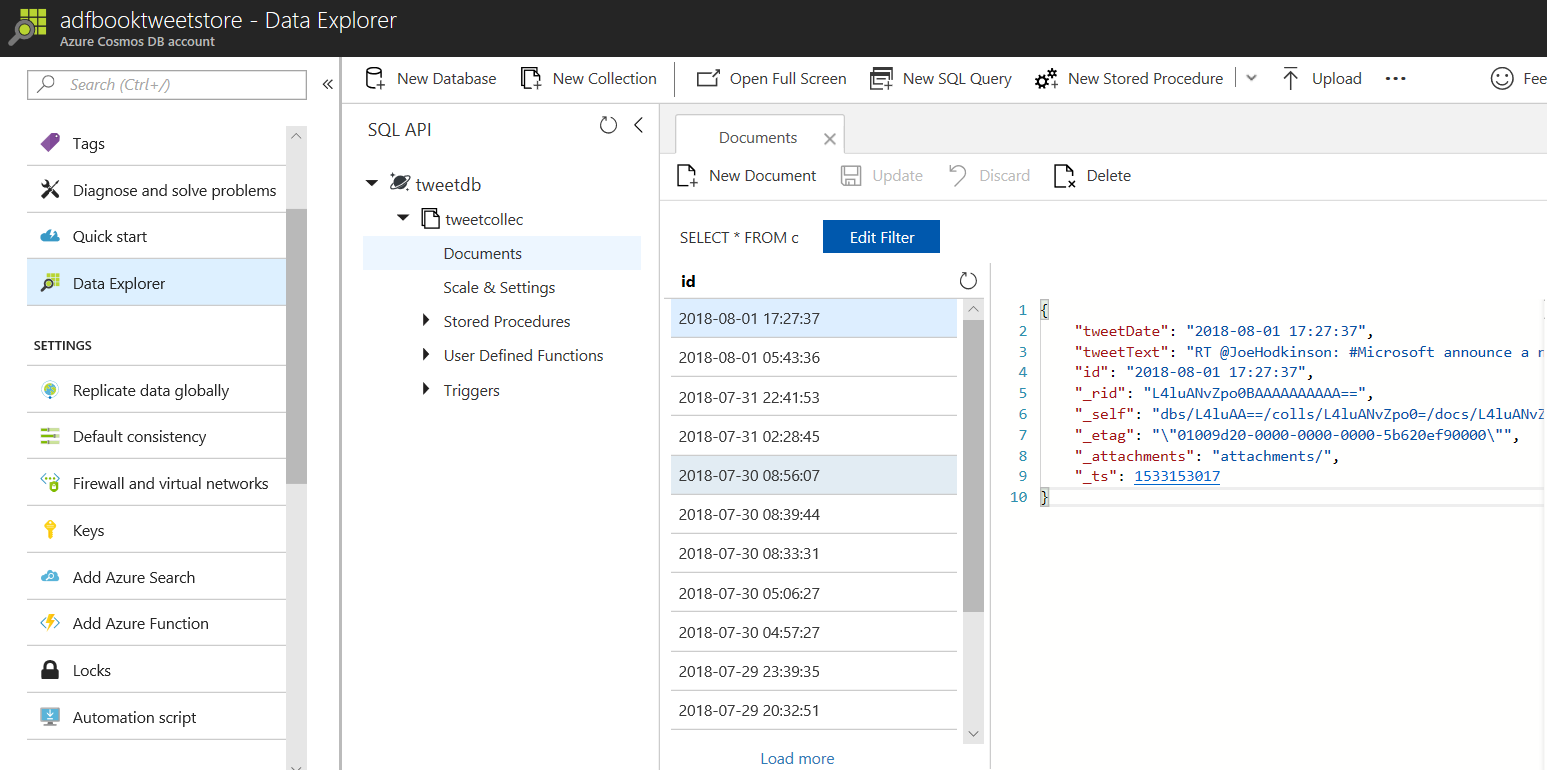
12) Click Trigger and Trigger Now

13) Click Finish

14) Click monitoring page and wait till pipeline get executed successfully



Finally, after successful completion the tweets gets stored on Azure CosmoDB



If you encountered any error do look at Azure batch services logs for the specific job to get insight on the type of error, it encountered.

