

Assignment 3

Name – Sati, Ankit

Date – 03/1/2022

Section - 001

Total in points (Maximum 100 points)–

Professors Comments –

Affirmation of Independent Effort – Ankit Sati

Question 1-

1. This questions wants us to walk through the edge computing that is being developed and take a look at it from the different perspectives listed below.
 - a. NEC - Network edge compute
 - b. MEC - Multi access edge compute
 - c. Help of Robotics
 - d. Impact on mission critical systems.
 - e. Implementation and use of private networks.

Main Goal – The primary goal is to deploy everything at the end for the users so that they can reduce the time of data on the networks. We try to provide the below services at the end of the network, which makes it faster/accurate with less chances of frame drops.

- Compute Power
- Cloud space
- High Availability
- Service Space

Main Objective - Everything for the end user need to remain the same from code creation to deployment/maintenance but the end user will not know where it is being used from. It can either be on the edge of the network or the azure platforms as it has been in the past.

NEC - Network edge compute

Network Edge Compute (NEC) is the network carrier equivalent, placing the edge computing platform within their network. Instead of needing to access applications and games running in the public cloud, software providers can bring their solutions physically closer to their end-users. At AT&T's Business Summit we gave an augmented reality demonstration, working with Taqtile, and showed how to perform maintenance on an aircraft landing gear.

Prime features

- Closer to end user
- One hop communications
- Compute power at the edge.
- Deployment of services at the end.

MEC - Multi access edge compute

Through the combination of local compute resources and private mobile connectivity (private LTE), we can enable many new scenarios. For instance, in the smart factory example used earlier customers are now able to run their robotic control logic, highly available and independent of connectivity to the public cloud. MEC helps ensure that operations and any associated critical

first-stage data processing remain up and production can continue uninterrupted. Advantage of near-infinite compute and storage, the cloud is ideal for large data-intensive and computational tasks, such as machine learning jobs for predictive maintenance analytics.

Prime Features

- Combination of local compute resources and private mobile connectivity
- Communication over private network.
- Complex logic like robotics can be accessed.
- **Near infinite compute storage.**

Prime advantages of EDGE COMPUTE

- Single hop computations
- Frames spend less time over the network.
- Reduce the chance of frame drops.
- Infinite compute storage at the end of network
- Complex logic can be deployed at the end of the network.
- Privatized network for mission critical projects.
- Very high speeds.
- High Availability of resources at the end.
- Cloud space with specific features as per demand of end user.

Examples of technologies used in field.

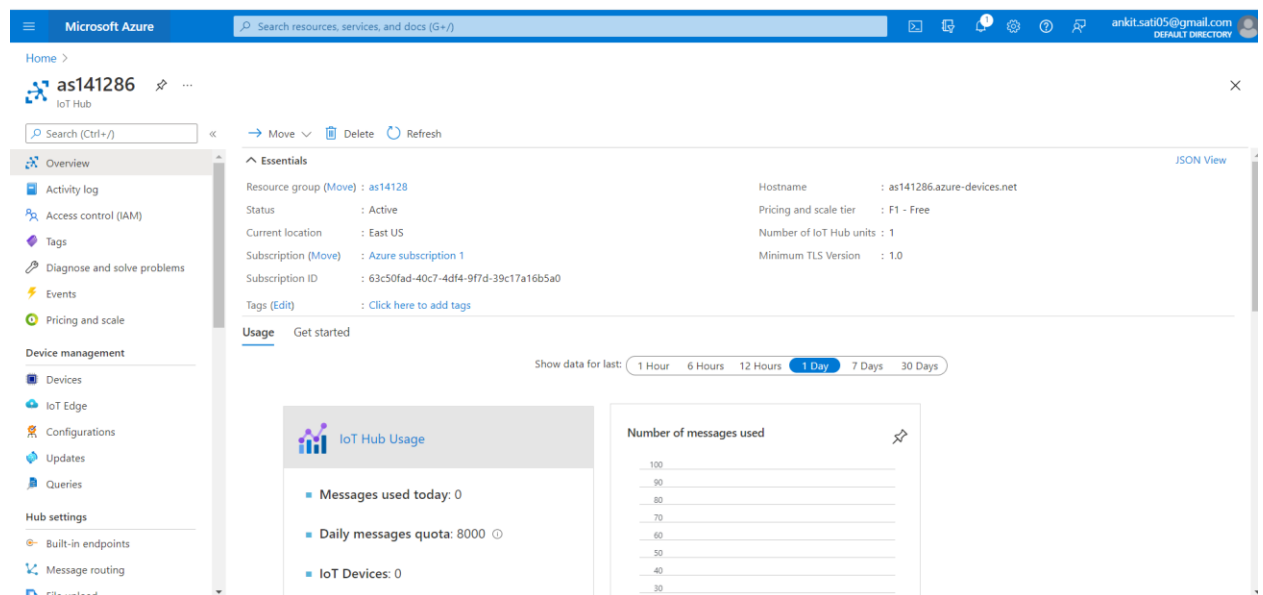
- Enterprise level – mission critical projects.
- Private LTE networks.
- Multiple cloud space
- Smart agriculture and services.
- Robotics in enterprise products.
- Product development.
- Resource deployment for projects.

Deploy your first IoT Edge module to a Windows device

Part 1 – Create your IOT HUB.

- Create an IoT Hub.
- Register an IoT Edge device to your IoT hub.
- Install and start the IoT Edge runtime on a virtual device.
- Remotely deploy a module to an IoT Edge device.

A resource group to manage all the resources you use in this quickstart. We use the example resource group name **IoTEdgeResources** throughout this quickstart.



Part 2 – Register an IoT Edge device (Screenshot attached)

Create a device identity for your IoT Edge device so that it can communicate with your IoT hub. The device identity lives in the cloud, and you use a unique device connection string to associate a physical device to a device identity. Since IoT Edge devices behave and can be managed differently than typical IoT devices, declare this identity to be for an IoT Edge device with the `--edge-enabled` flag.

hub name - as141286



resource group as14128

Connection Key - HostName=as141286.azure-devices.net;DeviceId=myEdgeDevice;SharedAccessKey=DUQ43decm2Rqt8D5u01ZEVDZ52gq7e1K8qdiUKwBs=


```
ing extensions without prompt.
unrecognized arguments: --hub-as141286


Examples from AI knowledge base:
https://aka.ms/cli_ref
Read more about the command in reference docs
ankit@Azure:~$ az iot hub device-identity create --device-id myEdgeDevice --edge-enabled --hub-name {hub_name}
Unable to find IoT Hub: {hub_name} in current subscription 63c50fad-40c7-4df4-9f7d-39c17a16b5a0.
ankit@Azure:~$ az iot hub device-identity create --device-id myEdgeDevice --edge-enabled --hub-name as141286
{
  "authentication": {
    "symmetricKey": {
      "primaryKey": "DUQ43decm2Rqt8D5u01ZEVDZ52gq7e1K8qdiUKwBs=",
      "secondaryKey": "8FKd5ktWZs41/Zq8S36UF1z79LLmJ9U/QbuA5qKx1A="
    },
    "type": "sas",
    "x509Thumbprint": {
      "primaryThumbprint": null,
      "secondaryThumbprint": null
    }
  },
  "capabilities": {
    "iotEdge": true
  },
  "cloudToDeviceMessageCount": 0,
  "connectionState": "Disconnected",
  "connectionStateUpdatedTime": "0001-01-01T00:00:00",
  "deviceId": "myEdgeDevice",
  "deviceScope": "ms-azure-iot-edge://myEdgeDevice-637725333601409849",
  "etag": "MjIzOTY3MzAy",
  "generationId": "637725333601409849",
  "lastActivityTime": "0001-01-01T00:00:00",
  "parentScopes": [],
  "status": "enabled",
  "statusReason": null,
  "statusUpdatedTime": "0001-01-01T00:00:00"
}
ankit@Azure:~$ az iot hub device-identity connection-string show --device-id myEdgeDevice --hub-name as141286
{
  "connectionString": "HostName=as141286.azure-devices.net;DeviceId=myEdgeDevice;SharedAccessKey=DUQ43decm2Rqt8D5u01ZEVDZ52gq7e1K8qdiUKwBs="
}
ankit@Azure:~$ ^C
```


Home > All resources > as141286 >


myEdgeDevice  


as141286


 Save


 Set modules


 Manage child devices


 Troubleshoot


 Device twin


 Manage keys


 Refresh


Device ID 


myEdgeDevice 


Primary Key 


..... 


Secondary Key 


..... 


Primary Connection String 


..... 

Secondary Connection String 


..... 

IoT Edge Runtime Response 


NA 

Enable connection to IoT Hub 

☒ Enable ☐ Disable

Parent device 

No parent device



Modules

IoT Edge hub connections

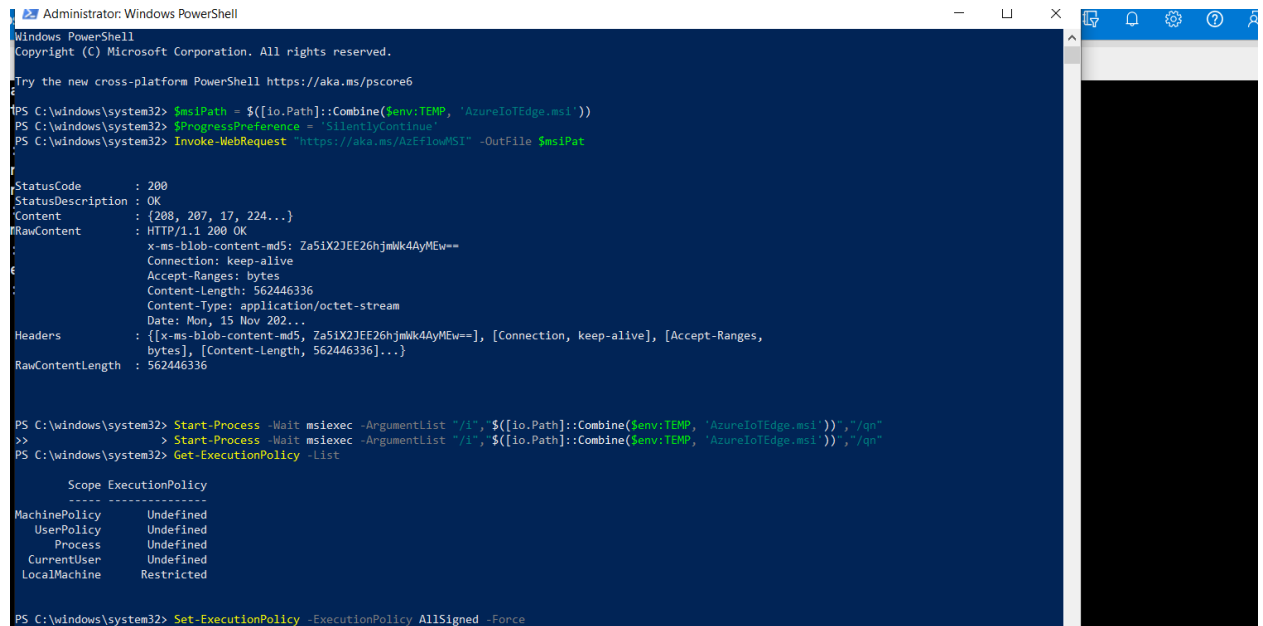
Deployments and Configurations

Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit Code
\$edgeAgent	Module Identity	NA	NA	NA	NA
\$edgeHub	Module Identity	NA	NA	NA	NA

Part 3- Install and start the IoT Edge runtime

This is one of the most critical steps while deploying the edge runtime. This basically a runtime E that will help us in checking and enabling the data once it is launched in space.

The issues that I was facing on the azure CLI were that I was not able to find and run the required hub name as it was not able to adapt the resources. Hence I soon switched to powershell to deploy my runtime.



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\windows\system32> $msiPath = $([io.Path]::Combine($env:TEMP, 'AzureIoTEdge.msi'))
PS C:\windows\system32> $ProgressPreference = 'SilentlyContinue'
PS C:\windows\system32> Invoke-WebRequest "https://aka.ms/AzEFLOWMSI" -OutFile $msiPath

StatusCode      : 200
StatusDescription : OK
Content          : {200, 207, 17, 224...}
RawContent       : HTTP/1.1 200 OK
                  x-ms-blob-content-md5: Za5IX2JEE26hjmK4AyMEw==
                  Connection: keep-alive
                  Accept-Ranges: bytes
                  Content-Length: 562446336
                  Content-Type: application/octet-stream
                  Date: Mon, 15 Nov 2022...
Headers          : {[x-ms-blob-content-md5, Za5IX2JEE26hjmK4AyMEw==], [Connection, keep-alive], [Accept-Ranges,
                  bytes], [Content-Length, 562446336]...}
RawContentLength : 562446336

PS C:\windows\system32> Start-Process -Wait msixec -ArgumentList "/I",$([io.Path]::Combine($env:TEMP, 'AzureIoTEdge.msi')),"/qn"
>>
> Start-Process -Wait msixec -ArgumentList "/I",$([io.Path]::Combine($env:TEMP, 'AzureIoTEdge.msi')),"/qn"
PS C:\windows\system32> Get-ExecutionPolicy -List

Scope ExecutionPolicy
-----
MachinePolicy Undefined
UserPolicy    Undefined
Process       Undefined
CurrentUser   Undefined
LocalMachine   Restricted

PS C:\windows\system32> Set-ExecutionPolicy -ExecutionPolicy AllSigned -Force
```

Part 4- Deploy a module

The module that you deploy in this section simulates a sensor and sends generated data. This module is a useful piece of code when you're getting started with IoT Edge because you can use the simulated data for development and testing. If you want to see exactly what this module does, you can view the simulated temperature sensor source code.

- When you create a new IoT Edge device, it will display the status code 417 -- The device's deployment configuration is not set in the Azure portal. This status is normal, and means that the device is ready to receive a module deployment.
- In IoT Edge Module Marketplace, search for and select the Simulated Temperature Sensor module.
- The module is added to the IoT Edge Modules section with the desired running status.

IoT Edge Module Marketplace



Marketplace



Simulated Temperature Sensor
Microsoft
IoT Edge module that simulates a temperature sensor

Select routes

- Routes to continue to the next step of the wizard.
- On the Routes tab, remove the default route, route, and then select Next: Review + create to continue to the next step of the wizard.
- The JSON file defines all of the modules that you deploy to your IoT Edge device. You'll see the SimulatedTemperatureSensor module and the two runtime modules, edgeAgent and edgeHub.
- \$edgeAgent, \$edgeHub, and SimulatedTemperatureSensor. If one or more of the modules has YES under SPECIFIED IN DEPLOYMENT but not under REPORTED BY DEVICE, your IoT Edge device is still starting them. Wait a few minutes, and then refresh the page.

```
{
  "modulesContent": {
    "$edgeAgent": {
      "properties.desired": {
        "modules": {
          "SimulatedTemperatureSensor": {
            "settings": {
              "image": "mcr.microsoft.com/azureiotedge-simulated-temperature-sensor:1.0",
              "createOptions": ""
            },
          },
        },
      },
    },
  },
}
```

```

        "type": "docker",
        "status": "running",
        "restartPolicy": "always",
        "version": "1.0"
    },
    },
    "runtime": {
        "settings": {
            "minDockerVersion": "v1.25"
        },
        "type": "docker"
    },
    "schemaVersion": "1.1",
    "systemModules": {
        "edgeAgent": {
            "settings": {
                "image": "mcr.microsoft.com/azureiotedge-agent:1.1",
                "createOptions": ""
            },
            "type": "docker"
        },
        "edgeHub": {
            "settings": {
                "image": "mcr.microsoft.com/azureiotedge-hub:1.1",
                "createOptions":
"{\"HostConfig\":{\"PortBindings\":{\"443/tcp\":[{\"HostPort\":\"443\"}],\"5671/tcp\":[{\"HostPort\":\"5671\"}],\"8883/tcp\":[{\"HostPort\":\"8883\"}]}}}"
            },
            "type": "docker",
            "status": "running",
            "restartPolicy": "always"
        }
    }
},
"$edgeHub": {
    "properties.desired": {
        "routes": {
            "route": "FROM /messages/* INTO $upstream",
            "SimulatedTemperatureSensorToIoTHub": "FROM
/messages/modules/SimulatedTemperatureSensor/* INTO $upstream"
        },
        "schemaVersion": "1.1",
        "storeAndForwardConfiguration": {
            "timeToLiveSecs": 7200
        }
    }
}

```

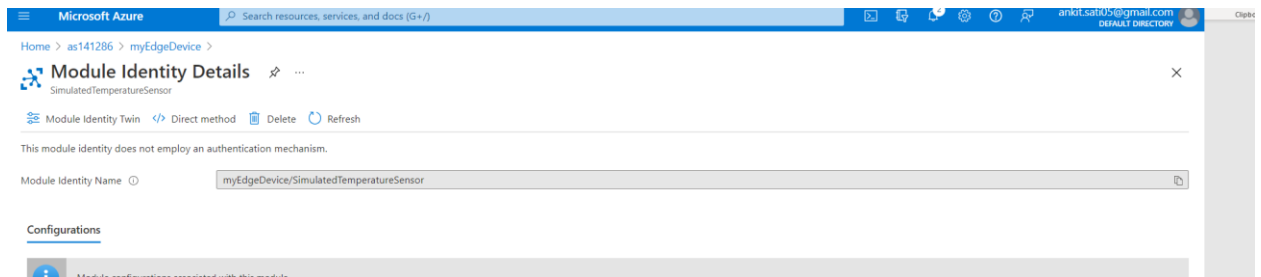


```

    }
  },
  "SimulatedTemperatureSensor": {
    "properties.desired": {
      "SendData": true,
      "SendInterval": 5
    }
  }
}
}
}

```

REVIEW AND CREATE



PART 5 – Viewing the device and the data

Finally we need to make sure that all of our data upto this point is up and running and we will push in the data and monitor it.

We need to push the Connect-EflowVm.

Modules					
IoT Edge hub connections					
Deployments and Configurations					
Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit Code
\$edgeAgent	Module Identity	NA	NA	NA	NA
\$edgeHub	Module Identity	NA	NA	NA	NA
SimulatedTemperatureSensor	Module Identity	NA	NA	NA	NA

Monitor the data

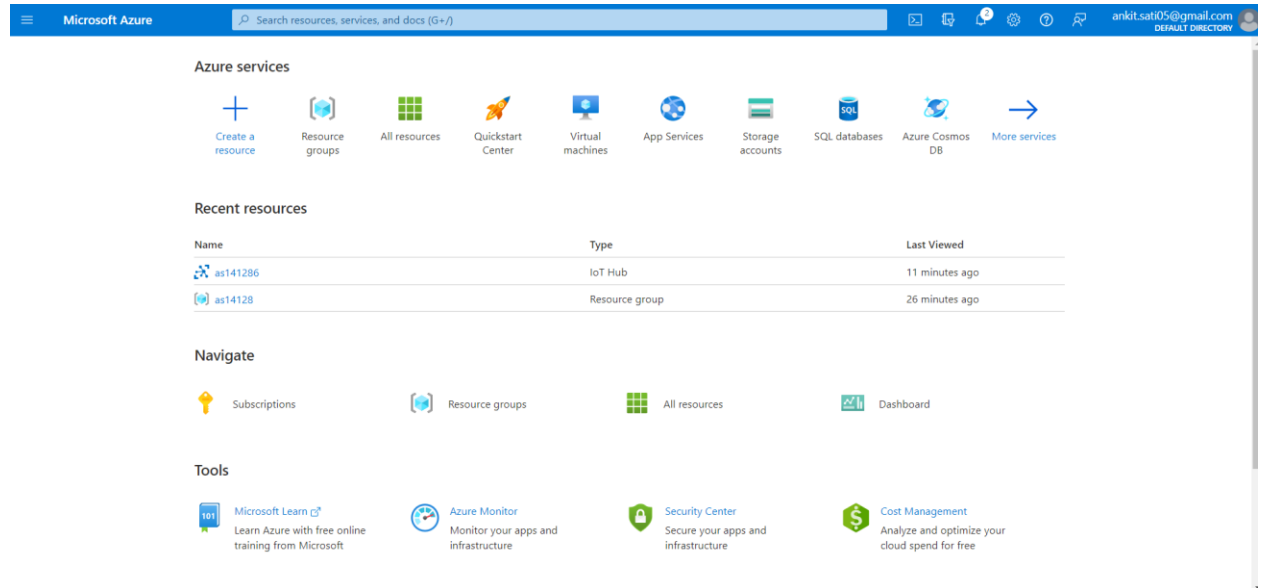
```

>> > Start-Process -Wait msixexec -ArgumentList "/1", "$([io.Path]::Combine($env:TEMP, "Azure
PS C:\windows\system32> Get-ExecutionPolicy -List
NAME                        STATUS      DESCRIPTION      CONFIG
SimulatedTemperatureSensor running     Up 3 hours       mcr.microsoft.com/azureiotedge-simulated-temperature-sensor:1.0
edgeAgent                  running     Up 3 hours       mcr.microsoft.com/azureiotedge-agent:1.0
edgeHub                    running     Up 3 hours       mcr.microsoft.com/azureiotedge-hub:1.0

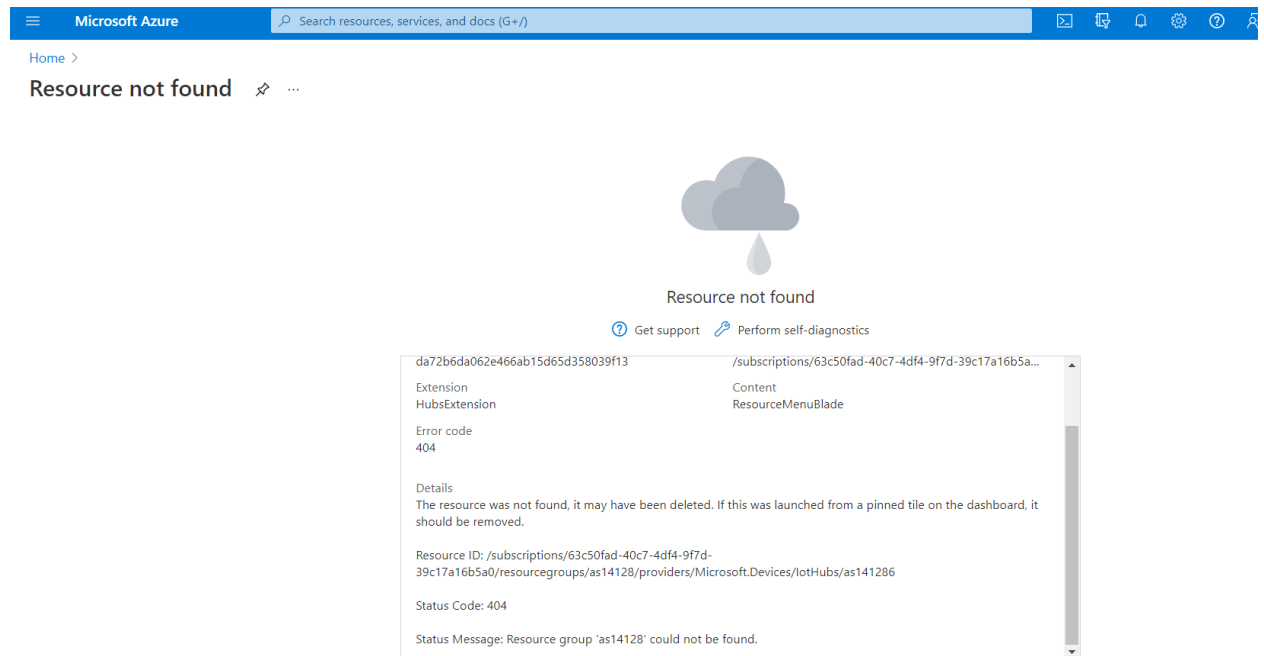
```

FINAL PART _ CLEANING UP THE RESOURCES

Final screenshot before deletion



Post Deletion



Activity log snippet

Notifications



[More events in the activity log →](#)

[Dismiss all](#)



Deleted resource group as14128



Deleted resource group as14128

2 minutes ago



Set Modules



Successfully updated IoT Edge settings for device myEdgeDevice.

13 minutes ago



Deployment succeeded



Deployment 'as141286-1114192541' to resource group 'as14128' was successful.

[Go to resource](#)

[Pin to dashboard](#)

53 minutes ago

Example 2 – AWS

Now we need to implement the somewhat similar functionalities on AWS.

The screenshot displays the AWS Management Console interface. At the top, there's a header with the AWS logo, a search bar, and user information. The main content area is titled 'AWS Management Console' and is divided into several sections. On the left, there's a sidebar with 'AWS services' and 'Build a solution'. The 'AWS services' section has links to 'Recently visited services' and 'All services'. The 'Build a solution' section has three cards: 'Launch a virtual machine', 'Build a web app', and 'Build using virtual servers'. On the right, there's a 'New AWS Console Home' section with a 'Switch now' button, a 'Stay connected to your AWS resources on-the-go' section with a link to the AWS Console Mobile App, and an 'Explore AWS' section with a link to the Machine Learning University. The footer contains 'Feedback', 'English (US)', and copyright information.

Launch Status

Your instances are now launching
The following instance launches have been initiated: i-045e181e24b79803b [View launch log](#)

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. [Find out how to connect to your instances.](#)

▼ Here are some helpful resources to get you started

- [How to connect to your Linux instance](#)
- [Learn about AWS Free Usage Tier](#)
- [Amazon EC2: User Guide](#)
- [Amazon EC2: Discussion Forum](#)

While your instances are launching you can also

- Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
- Create and attach additional EBS volumes (Additional charges may apply)
- Manage security groups

1. Step 2 and 3 – SSH into the instance that we have created.

```
ankit@LAPTOP-S2U1QMGB:/mnt/c/Users/ankit$ sudo apt-get install -y kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  kubectl
0 upgraded, 1 newly installed, 0 to remove and 112 not upgraded.
Need to get 8929 kB of archives.
After this operation, 46.6 MB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 kubectl amd64 1.23.3-00 [8929 kB]
Fetched 8929 kB in 1s (10.5 MB/s)
Selecting previously unselected package kubectl.
(Reading database ... 32226 files and directories currently installed.)
Preparing to unpack .../kubectl_1.23.3-00_amd64.deb ...
Unpacking kubectl (1.23.3-00) ...
Setting up kubectl (1.23.3-00) ...
ankit@LAPTOP-S2U1QMGB:/mnt/c/Users/ankit$
```

➔ Saving the required keys as mentioned in the assignment.

Key.pub identification

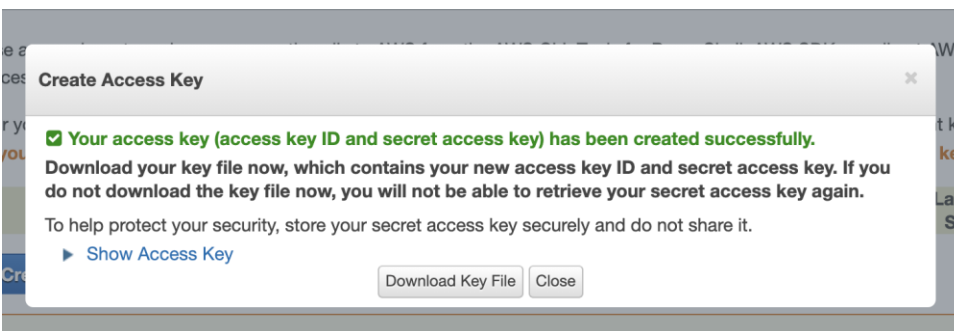
Type – RSA 4096

Key – Screenshot attached (Blurred the actual key for privacy)

```

ankit@LAPTOP-S2U1QMGB:~/tce-linux-amd64-v0.9.1$ ssh-keygen -t rsa -b 4096 -C "jc@archemyl.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ankit/.ssh/id_rsa): Key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in Key
Your public key has been saved in Key.pub
The key fingerprint is:
SHA256:JhV8DJE4Jv00XXwkwNcVU [REDACTED]
The key's randomart image is:
+---[RSA 4096]-----+
|  . o+=XBB00=|
|  . = o+. =o*Bo|
|  o +.= . =o.*|
|  .. . . 000|
|  . S      ..|
|  o      E   |
+---[SHA256]-----+
ankit@LAPTOP-S2U1QMGB:~/tce-linux-amd64-v0.9.1$

```



- ➔ The final build needs to be created on the GUI first.
- ➔ Post that we need to setup the EC2 instance and the S3 storage bucket
- ➔ Finally, we can **ssh** into the created instance.

- **Screenshot of the final build**

```

ankit@LAPTOP-S2U1QMGB:/mnt/c/Users/ankit$ sudo apt-get install -y kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  kubectl
0 upgraded, 1 newly installed, 0 to remove and 112 not upgraded.
Need to get 8929 kB of archives.
After this operation, 46.6 MB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 kubectl amd64 1.23.3-00 [8929 kB]
Fetched 8929 kB in 1s (10.5 MB/s)
Selecting previously unselected package kubectl.
(Reading database ... 32226 files and directories currently installed.)
Preparing to unpack ../kubectl_1.23.3-00_amd64.deb ...
Unpacking kubectl (1.23.3-00) ...
Setting up kubectl (1.23.3-00) ...
ankit@LAPTOP-S2U1QMGB:/mnt/c/Users/ankit$

```

The CSV file- <https://github.com/domoritz/random-csv>

Step 4 – Single GPU push

- ➔ Here I have used a regular CSV file instead of the IOT repository as it is of no cost to us.

- ➔ The CSV file has the data of the mobile and temperature models which are the same ones used in azure cloud.
- ➔ The reason for it not implementing is that the storage buckets are built different here so we need more than 1 GPU and multiple **nodes to emulate the IOT sensors.**

Provisioned read capacity units	1
Provisioned write capacity units	1
Last decrease time	-
Last increase time	-
Storage size (in bytes)	0 bytes
Item count	0
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:367622474624:table/simplilearn_users

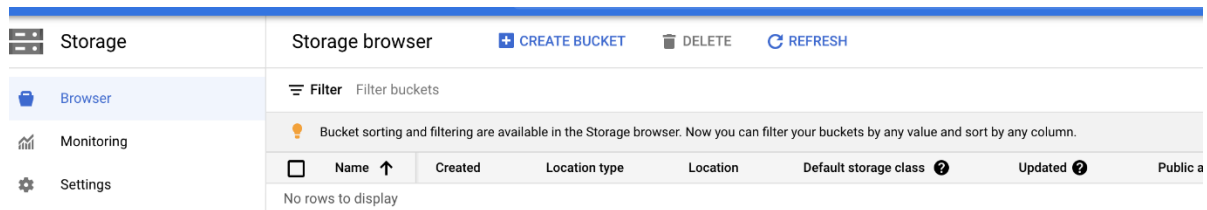
Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

After running the notebook, go to aws console and go to dynamodb from there. Click on tables and under that click on DataTable and click on items tab to view the contents of the table.

	PartitionKey	RowKey	date	description	url
<input type="checkbox"/>	experiment1	id1	12/1/2016	"here is a view of what is to be made"	https://cc-lab2.s3.
<input type="checkbox"/>	experiment1	id2	12/3/2016	"map reduce picture"	https://cc-lab2.s3.
<input type="checkbox"/>	experiment1	id5	12/6/2016	"bio samples"	https://cc-lab2.s3.
<input type="checkbox"/>	experiment2	id3	12/4/2016	"sample notebook"	https://cc-lab2.s3.
<input type="checkbox"/>	experiment3	id4	12/5/2016	"workers"	https://cc-lab2.s3.

GCP

Login to GCP account and go to Storage from the navigation panel.



Click on create bucket and give a valid name.

Once the bucket is created, it should be visible

Once done, go to terminal and run the following command:

```
docker run -i -t -p 8888:8888 dbgannon/tutorial
```

After this go to local host link and run the python notebook called gcloud.ipynb after making necessary edits related to bucket name and file paths.

```

In [22]: client = storage.Client()

In [23]: from gcloud import datastore
         clientds = datastore.Client()

In [29]: import csv

In [53]: bucket = client.bucket('cc-lab2')
         key = clientds.key('book-table')

In [55]: with open('/datadir/experiments.csv', 'rt') as csvfile:
         csvf = csv.reader(csvfile, delimiter=',', quotechar='"')
         for item in csvf:
             print(item)
             blob = bucket.blob(item[3])
             data = open("/datadir/"+item[3], 'rb')
             blob.upload_from_file(data)
             blob.make_public()
             url = "https://console.cloud.google.com/storage/browser/cc-lab2/"+item[3]
             entity = datastore.Entity(key=key)
             entity['experiment-name'] = item[0]
             entity['experiment-id'] = item[1]
             entity['date'] = item[2]
             entity['description'] = item[4]
             entity['url'] = url

```

Once the above steps are executed, go to GCP again and go to storage to see the uploaded objects.

Buckets > cc-lab2

[UPLOAD FILES](#)
[UPLOAD FOLDER](#)
[CREATE FOLDER](#)
[MANAGE HOLDS](#)
[DOWNLOAD](#)
[DELETE](#)

Filter by name prefix only Filter objects and folders

<input type="checkbox"/>	Name	Size	Type	Created time	Storage class	Last modified	Public access	Encryption	Retention
<input type="checkbox"/>	a	54.5 KB	application/octet-stream	18 Feb 2021, ...	Standard	18 Feb 202...	Public to Internet	Copy URL	Google-managed key
<input type="checkbox"/>	b	68 KB	application/octet-stream	18 Feb 2021, ...	Standard	18 Feb 202...	Public to Internet	Copy URL	Google-managed key
<input type="checkbox"/>	c	86.2 KB	application/octet-stream	18 Feb 2021, ...	Standard	18 Feb 202...	Public to Internet	Copy URL	Google-managed key
<input type="checkbox"/>	d	41.8 KB	application/octet-stream	18 Feb 2021, ...	Standard	18 Feb 202...	Public to Internet	Copy URL	Google-managed key
<input type="checkbox"/>	e	114.9 KB	application/octet-stream	18 Feb 2021, ...	Standard	18 Feb 202...	Public to Internet	Copy URL	Google-managed key

Check the datastore to have a look at created table

c. RabbitMQ

Now we will setup and push the same data in the RabbitMQ environment. RabbitMQ is an open-source message-broker software that originally implemented the Advanced Message Queuing Protocol and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol, MQ Telemetry Transport, and other protocols.

Installation and Setup part

➔ Before setting up RabbitMQ we need to download the ERLangPackages.

RabbitMQ nodes are often managed, inspected and operated using [CLI Tools in PowerShell](#).

On Windows, CLI tools have a `.bat` suffix compared to other platforms. For example, `rabbitmqctl` on Windows is invoked as `rabbitmqctl.bat`.

In order for these tools to work they must be able to [authenticate with RabbitMQ nodes](#) using a shared secret file called the Erlang cookie.

The main [CLI tools guide](#) covers most topics related to command line tool usage.

In order to explore what commands various RabbitMQ CLI tools provide, use the `help` command:

```
# Lists commands provided by rabbitmqctl.bat
rabbitmqctl.bat help

# Lists commands provided by rabbitmq-diagnostics.bat
rabbitmq-diagnostics.bat help

# ...you guessed it!
rabbitmq-plugins.bat help
```

To learn about a specific command, pass its name as an argument to `help`:

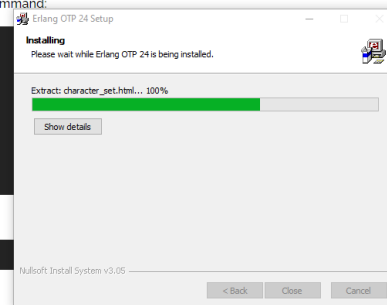
```
rabbitmqctl.bat help add_user
```

Cookie File Location

On Windows, the [cookie file location](#) depends on whether the `HOME` and `HOME\PATH` environment variables are set.

If RabbitMQ is installed using a non-administrative account, a [shared secret](#) file used by nodes and CLI tools will not be placed into a correct location, leading to [authentication failures](#) when `rabbitmqctl.bat` and other CLI tools are used.

One of these options can be used to mitigate:



➔ After this we need to download and setup the RabbitMQ over our local machine.

tools to work they must be able to [authenticate with RabbitMQ nodes](#) using a shared secret file called the

guide

what c

provided

provided

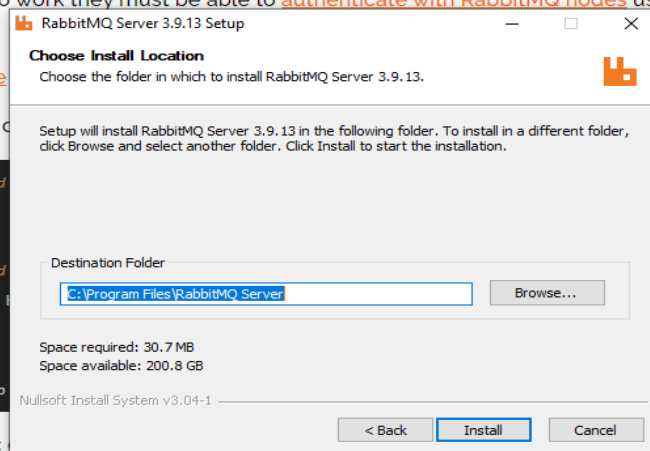
s.bat l

t help

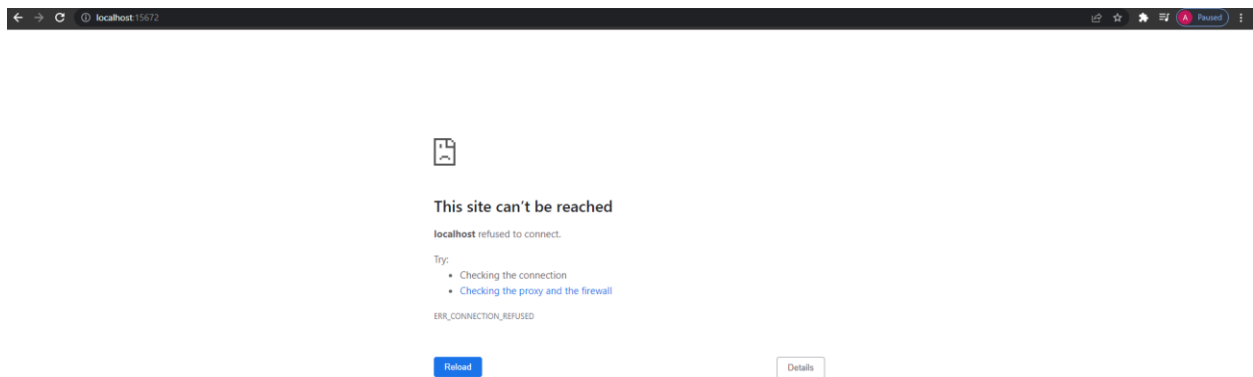
specific

o add_user

ation



➔ Once this is setup we need to try and open the local host - <http://localhost:15672/#/>



- ➔ As we can see that the local host is not setup.
- ➔ We need to open the management plugin to make it work,

```
Administrator: RabbitMQ Command Prompt (sbin dir)
C:\Program Files\RabbitMQ Server\rabbitmq_server-3.9.13\sbin>rabbitmq-plugins enable rabbitmq_management
Enabling plugins on node rabbit@LAPTOP-S2U1QMGB:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@LAPTOP-S2U1QMGB...
The following plugins have been enabled:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
started 3 plugins.
C:\Program Files\RabbitMQ Server\rabbitmq_server-3.9.13\sbin>
```

➔ After this we need to open the localhost to check the status of **RabbitMQ**.

RabbitMQ 3.9.13 Erlang 24.2.2

Overview Connections Channels Exchanges Queues Admin

Overview

Totals

Queued messages [last minute](#)

Currently idle

Message rates [last minute](#)

Currently idle

Global counts

Connections: 0 Channels: 0 Exchanges: 7 Queues: 0 Consumers: 0

Nodes

Name	File descriptors	Socket descriptors	Erlang processes	Memory	Disk space	Uptime	Info	Reset stats
rabbit@LAPTOP-SZU1QMG8	0	0	353	68 MiB	201 GiB	4m 11s	basic disc 1 rse	This node All nodes

Churn statistics

Ports and contexts

Export definitions

Import definitions

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

➔ Once this is done, we need to run a simple num.py file.

➔ Before that we will create a queue and open an channel.

Queues

All queues (1)

Pagination

Page 1 of 1 - Filter: ☐ Regex

Displaying 1 item, page size up to: 100

Overview

Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
Cloud-Test	classic	Args	idle	0	0	0			

Add a new queue

Type:

Name:

Durability:

Auto delete:

Arguments:

Add Message TTL | Auto expire | Overflow behaviour | Single active consumer | Dead letter exchange | Dead letter routing key | Max length | Max length bytes | Maximum priority | Lazy mode | Master locator

Add queue

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

➔ After this we need to create an exchange so that the **run.py** file uploaded.

Name	Type	Features	Message rate in	Message rate out	+/-
(AMQP default)	direct	D			
amq.direct	direct	D			
amq.fanout	fanout	D			
amq.headers	headers	D			
amq.match	headers	D			
amq.rabbitmq.trace	topic	D I			
amq.topic	topic	D			
ankit-Exchange	direct	D			

Add a new exchange

Name:
Type: direct
Durability: Durable
Auto delete: No
Internal: No
Arguments: = String
Add Alternate exchange

Add exchange

➔ Now we need to bind the file to the queue so that we can monitor the activity on the same. – **Binding the queue**

RabbitMQ 3.9.13 Erlang 24.2.2

Refreshed 2022-03-03 22:12:22 Refresh every 5 seconds
Virtual host All
Cluster rabbit@LAPTOP-S2U1QMGB
User guest Log out

Overview
Connections
Channels
Exchanges
Queues
Admin

Exchange: ankit-Exchange

Overview

Message rates last minute
Currently idle
Details

Type direct
Features durable: true
Policy

Bindings

This exchange
↓
... no bindings ...

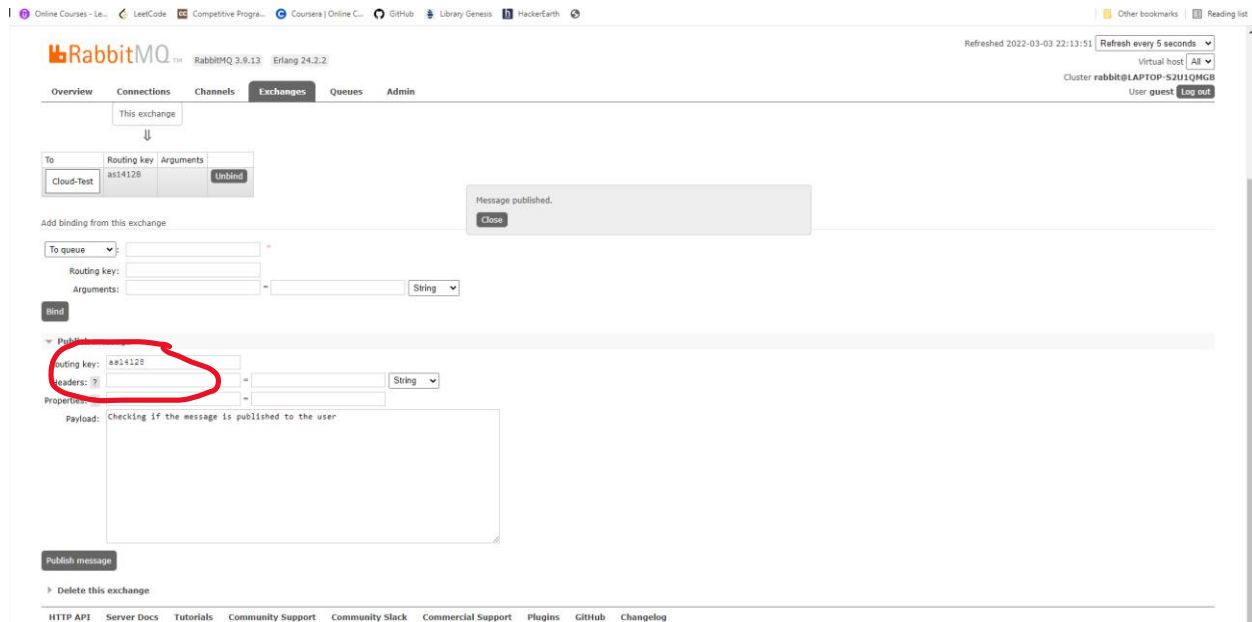
Add binding from this exchange

To queue Cloud-Test
Routing key aa14128
Arguments
String

Bind

Publish message
Delete this exchange

➔ Publishing the file and printing the message.



The screenshot shows the RabbitMQ Admin interface for the 'anix-Exchange'. The 'Publish' section is active, and a message has been successfully published, as indicated by the 'Message published.' notification. The 'Routing key' is set to 'anix-123', which is circled in red. The 'Payload' field contains the text 'Checking if the message is published to the user'.

Overview | Connections | Channels | **Exchanges** | Queues | Admin

This exchange

To: Cloud-Test | Routing key: anix-123 | Arguments: | Unbind

Message published. [Close]

Add binding from this exchange

To queue: | Routing key: | Arguments: | String

Bind

▼ Publish

Routing key: anix-123 | Arguments: | String

Properties

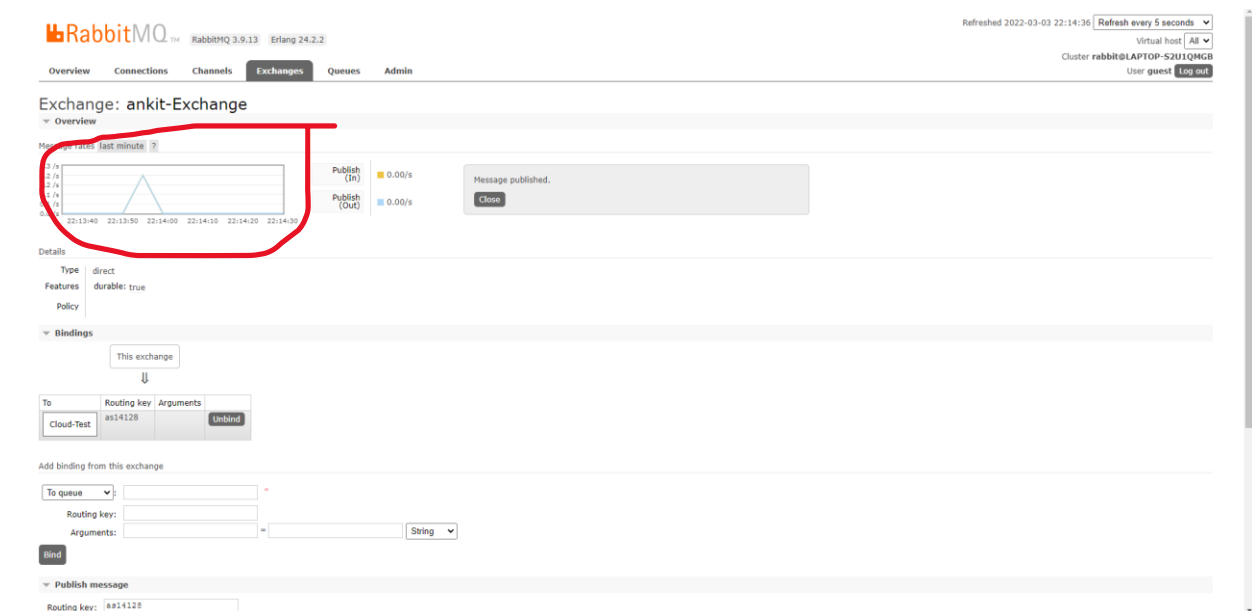
Payload: Checking if the message is published to the user

Publish message

► Delete this exchange

HTTP API | Server Docs | Tutorials | Community Support | Community Slack | Commercial Support | Plugins | GitHub | Changelog

➔ Checking the activity on the client side.



The screenshot shows the RabbitMQ Admin interface for the 'anix-Exchange'. The 'Overview' section is active, and a graph shows the message rates over time. The 'Message published.' notification is also visible. The 'Exchange: anix-Exchange' title is circled in red.

Overview | Connections | Channels | **Exchanges** | Queues | Admin

Exchange: anix-Exchange

▼ Overview

Message rates (last minute)

2 /s
1 /s
0 /s

22:13:40 22:13:50 22:14:00 22:14:10 22:14:20 22:14:30

Publish (In) 0.00/s

Publish (Out) 0.00/s

Message published. [Close]

Details

Type: direct

Features: durable: true

Policy

▼ Bindings

This exchange

To: Cloud-Test | Routing key: anix-123 | Arguments: | Unbind

Add binding from this exchange

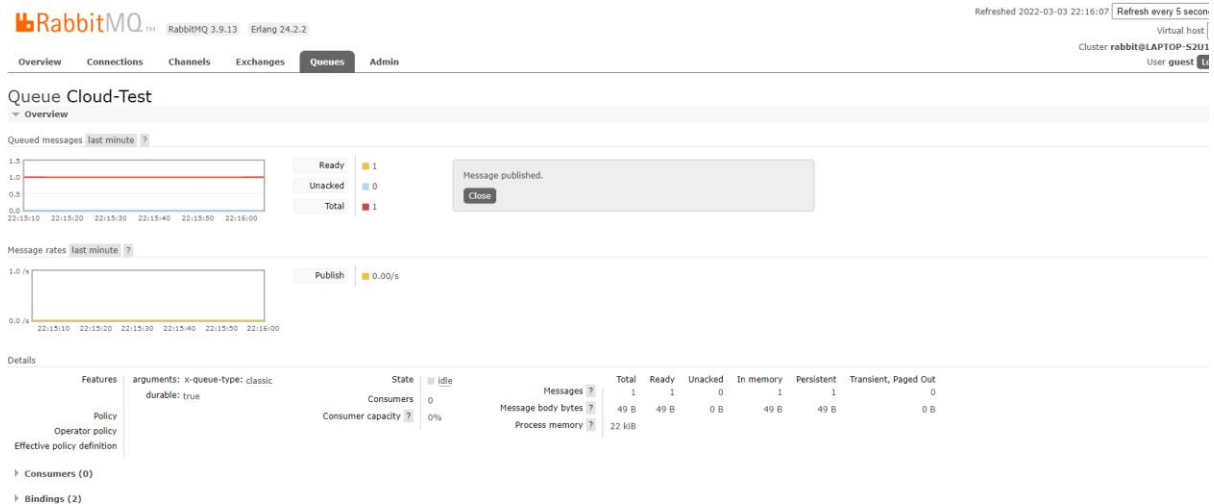
To queue: | Routing key: | Arguments: | String

Bind

▼ Publish message

Routing key: anix-123

- ➔ Checking the activity after pushing the data in a local cloud.
- ➔ CLOUD QUEUE TEST.



PART 3 and Extra credits.

Tutorial 1 - Monitor IoT Edge devices

Step 1 - Create a Log Analytics workspace

Workspace ID : b2050b98-4b47-4ee2-8a19-7c6d7b6f65b1

Delete

Essentials

Resource group (Move) : newgroup

Status : Active

Location : East US

Subscription (Move) : Azure subscription 1

Subscription ID : 63c50fad-40c7-4df4-9f7d-39c17a16b5a0

Tags (Edit) : Click here to add tags

Workspace Name : as14128

Workspace ID : b2050b98-4b47-4ee2-8a19-7c6d7b6f65b1

Pricing tier : Pay-as-you-go

Access control mode : Use resource or workspace permissions

Operational issues : OK

Get started with Log Analytics

Step 2 - Create a Log Analytics workspace

ID - b7f18d8e-271b-442d-8714-ee5d59e0159e

Resource ID

Security settings

Identity

Support + troubleshooting

Resource health

System-assigned User-assigned

A system-assigned managed identity enables Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. Once enabled, all necessary permissions can be granted via Azure role-based-access-control. The lifecycle of this type of managed identity is tied to the lifecycle of this resource. Additionally, each resource (e.g. Virtual Machine) can only have one system-assigned managed identity. [Learn more about Managed identities.](#)

Save Discard Refresh Feedback

Status

☒ On

☐ Off

i This resource is registered with Azure Active Directory. You can control its access to services like Azure Resource Manager, Azure Key Vault, etc. [Learn more](#)

Object ID

b7f18d8e-271b-442d-8714-ee5d59e0159e

Permissions

Azure role assignments

Step 3 - Deploy the metrics collector module

Dashboard > as14128 >

as14128 ...

Save Set modules Manage child devices Troubleshoot Device twin Manage keys Refresh

Device ID as14128

Primary Key

Secondary Key

Primary Connection String

Secondary Connection String

IoT Edge Runtime Response NA

Enable connection to IoT Hub ☒ Enable ☐ Disable

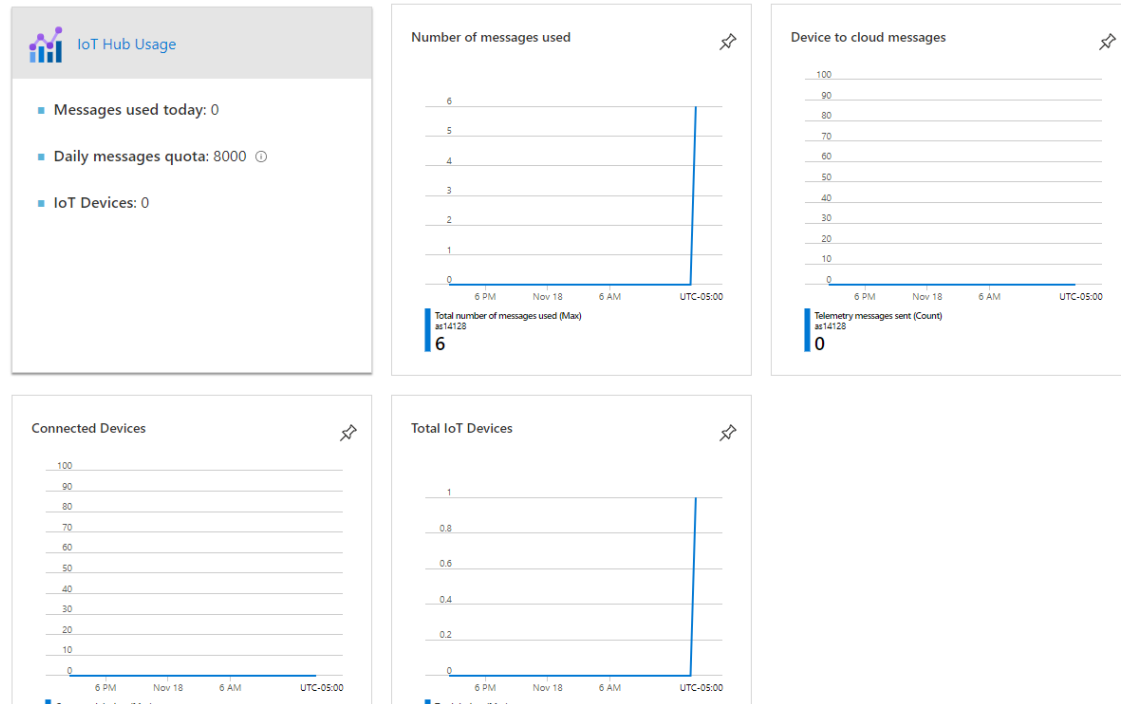
Parent device No parent device

Modules IoT Edge hub connections Deployments and Configurations

Client ID Status

There are no entries for this device.

Step 4 - Explore the fleet view and health snapshot workbooks

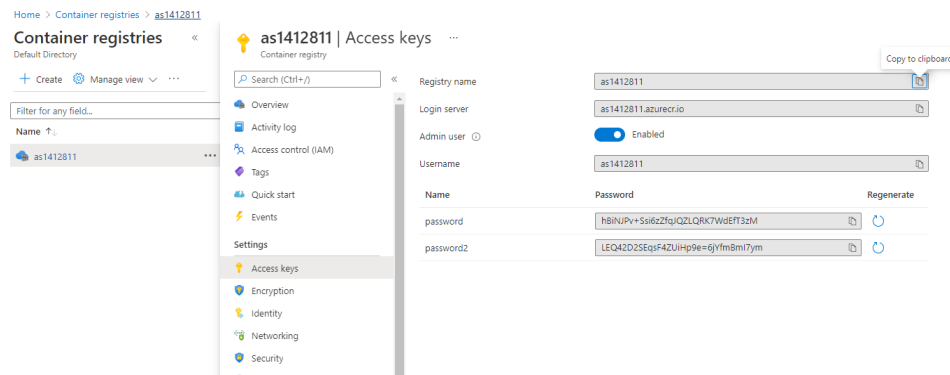


Tutorial 2 - Develop IoT Edge modules using Windows containers

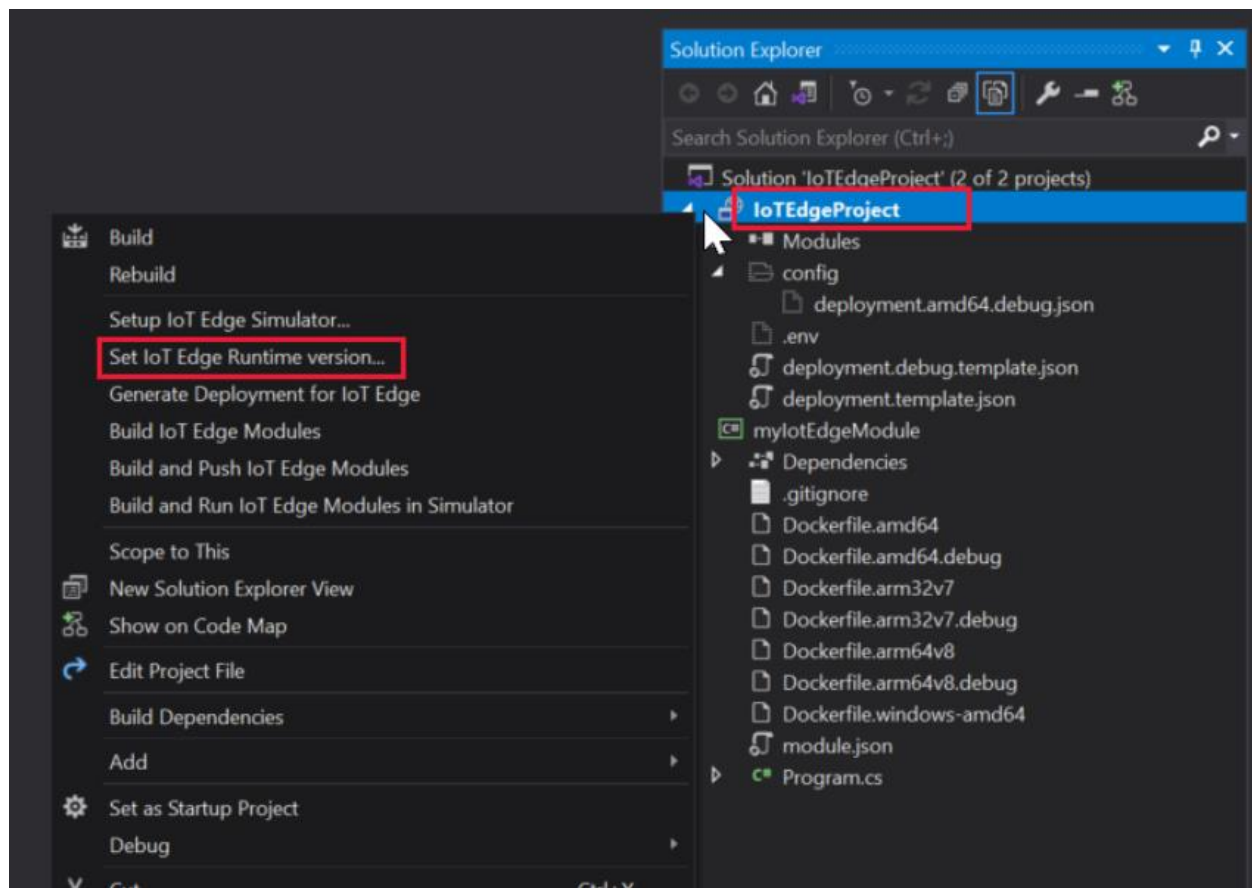
Step 1 - Set up VS

Installing environment and repos.

Step -2 – Create a container registry



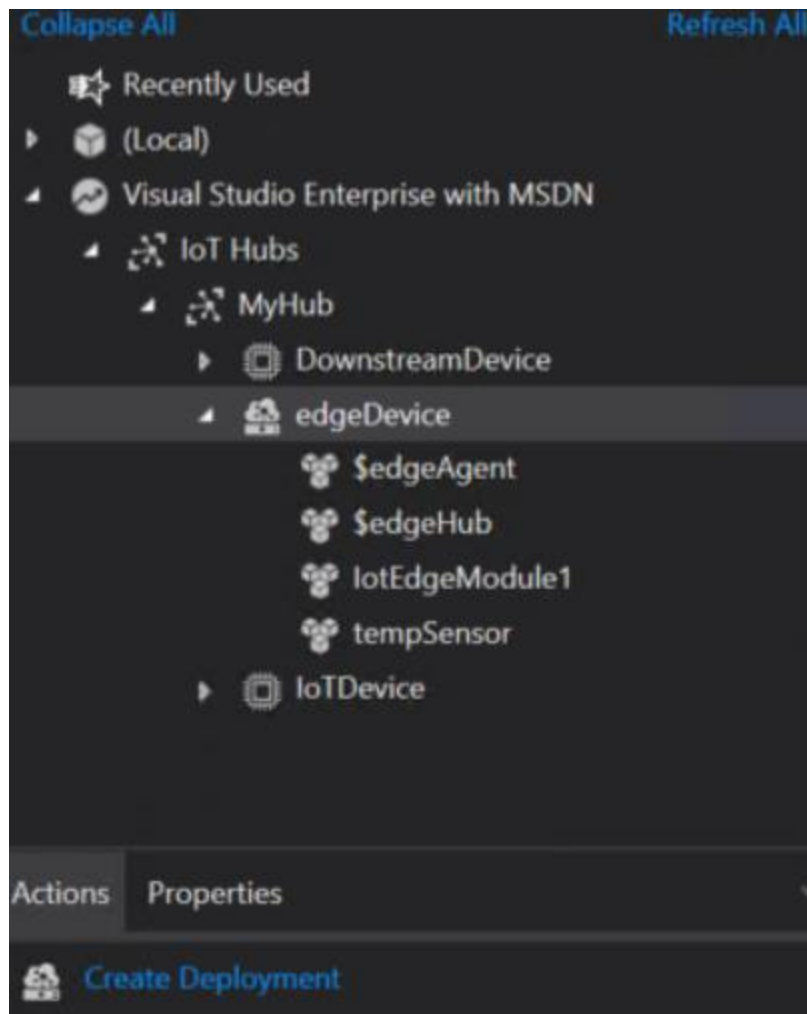
Step 3 – Edge Runtime



Step 4 - Provide your registry credentials to the IoT Edge agent

```
"registryCredentials": {
  "<registry name>": {
    "username": "$CONTAINER_REGISTRY_USERNAME_<registry name>",
    "password": "$CONTAINER_REGISTRY_PASSWORD_<registry name>",
    "address": "<registry name>.azurecr.io"
  }
}
```

Step 5 - Changes on Device



STEP 6 –

Clean up resource

- Resources – Done
- Modules – Done
- Devices done

Part 4 – Custom code

Step 1 - Set up Java

Installing environment and repos.

Step 2 – Code applet




```
4 import java.util.Map;
5
6 import javax.json.Json;
7 import javax.json.JsonObject;
8 import javax.json.JsonReader;
9
10 import com.microsoft.azure.sdk.iot.device.DeviceTwin.Pair;
11 import com.microsoft.azure.sdk.iot.device.DeviceTwin.Property;
12 import com.microsoft.azure.sdk.iot.device.DeviceTwin.TwinPropertyCallback;
13
14 private static final String TEMP_THRESHOLD = "TemperatureThreshold";
15 private static AtomicLong tempThreshold = new AtomicLong(25);
16
17 protected static class MessageCallbackMqtt implements MessageCallback {
18     private int counter = 0;
19     @Override
20     public IoTHubMessageResult execute(Message msg, Object context) {
21         this.counter += 1;
22
23         String msgString = new String(msg.getBytes(), Message.DEFAULT_IOTHUB_MESSAGE_CHARSET);
24         System.out.println(
25             String.format("Received message %d: %s",
26                 this.counter, msgString));
27         if (context instanceof ModuleClient) {
28             try (JsonReader jsonReader = Json.createReader(new StringReader(msgString))) {
29                 final JsonObject msgObject = jsonReader.readObject();
30                 double temperature = msgObject.getJsonObject("machine").getJsonNumber("temperature").doubleValue();
31                 long threshold = App.tempThreshold.get();
32                 if (temperature >= threshold) {
33                     ModuleClient client = (ModuleClient) context;
34                     System.out.println(
35                         String.format("Temperature above threshold %d. Sending message: %s",
36                             threshold, msgString));
37                     client.sendEventAsync(msg, eventCallback, msg, App.OUTPUT_NAME);
38                 }
39             }
40         }
41     }
42 }
```

Step 3 – Push the custom modules







Step 4 - Deploy modules to device

All resources
Azure subscription 1

Refresh

 as14128	IoT Hub
 as14128	Log Analytics works...
 cs210032001a644cba6	Storage account

Azure getting started made easy!



Launch an app of your choice on Azure in a few quick steps

Create DevOps Starter

Quickstarts + tutorials

Step 5 - Edit the module twin

```
import com.microsoft.azure.sdk.iot.device.DeviceTwin.TwinPropertyCallback;

private static final String TEMP_THRESHOLD = "TemperatureThreshold";
private static AtomicLong tempThreshold = new AtomicLong(5);

protected static class MessageCallbackMqtt implements MessageCallback {
    private int counter = 0;
```

Step 6 – Delete all IOT edge modules.

- Resources – Done
- Modules – Done
- Devices done