Azure Focused IOT

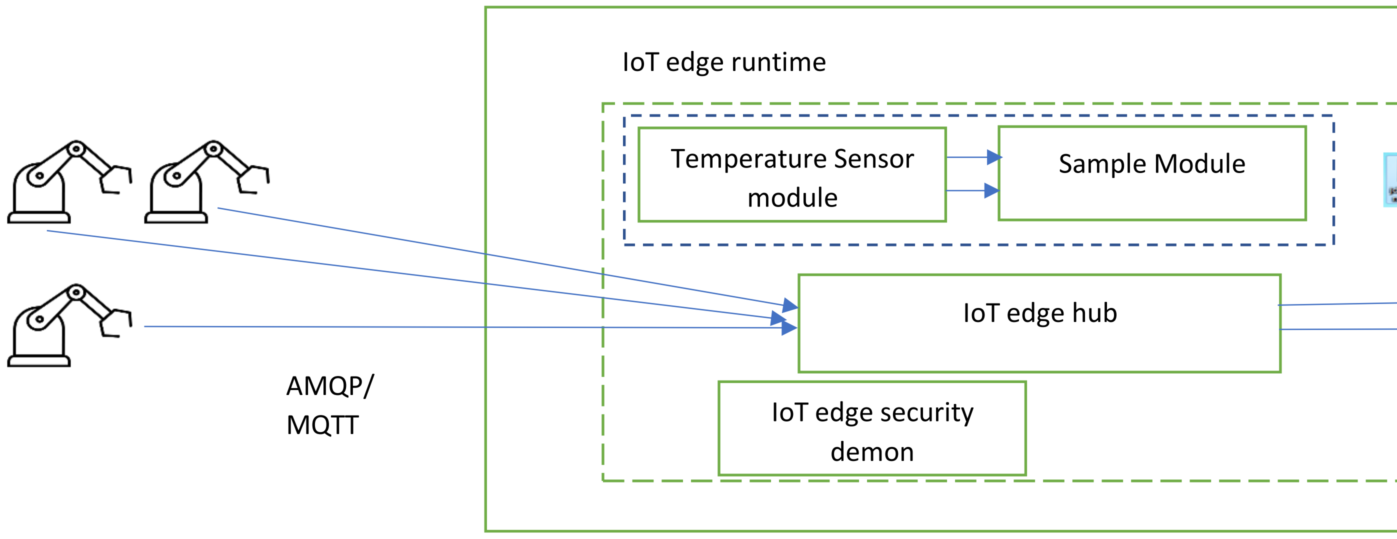
Table of contents:

Problem Statement:

Azure IOT edge devices offer a bunch of useful features like performing logic at the edge rather than in IOT hub reducing network latency, also acts like a IoT edge gateway to exchange telemetry between IoT hub and devices which does not support conventional IoT communication protocols like AMQP, MQTT or HTTP etc and lot of additional offerings.

In order to execute business logic at the IoT edge devices, we need to deploy the IoT edge runtime along with our custom runtime module in the form of containers and it can be deployed to IoT edge devices with the help of either VS code or Azure CLI. We will discuss how to **deploy custom container modules into Azure IoT edge enabled devices using Visual Studio Code and Java** in this tutorial.

Solution/Architecture:



Technical Details and Implementation of solution:

Prerequisites:

- Install Visual Studio Code 2017 or later versions
- Install Docker desktop
- Install JDK8.0 or later versions
- Maven build tool
- Azure subscription
- Docker hub or azure container registry repository

Environment set up:

First, we need to install a docker runtime engine in our local to build container images for IOT edge modules in our local environment and push it to either docker hub or azure container registry. Please download and install docker from below link,

Docker Desktop for Windows

After that, install VS code 2017 or later versions from following link:

Visual Studio Code

Create Azure IOT hub and edge device

Login into **https://portal.azure.com/** with your Azure subscription and go to **Create a Resource->IoT Hub->Create**

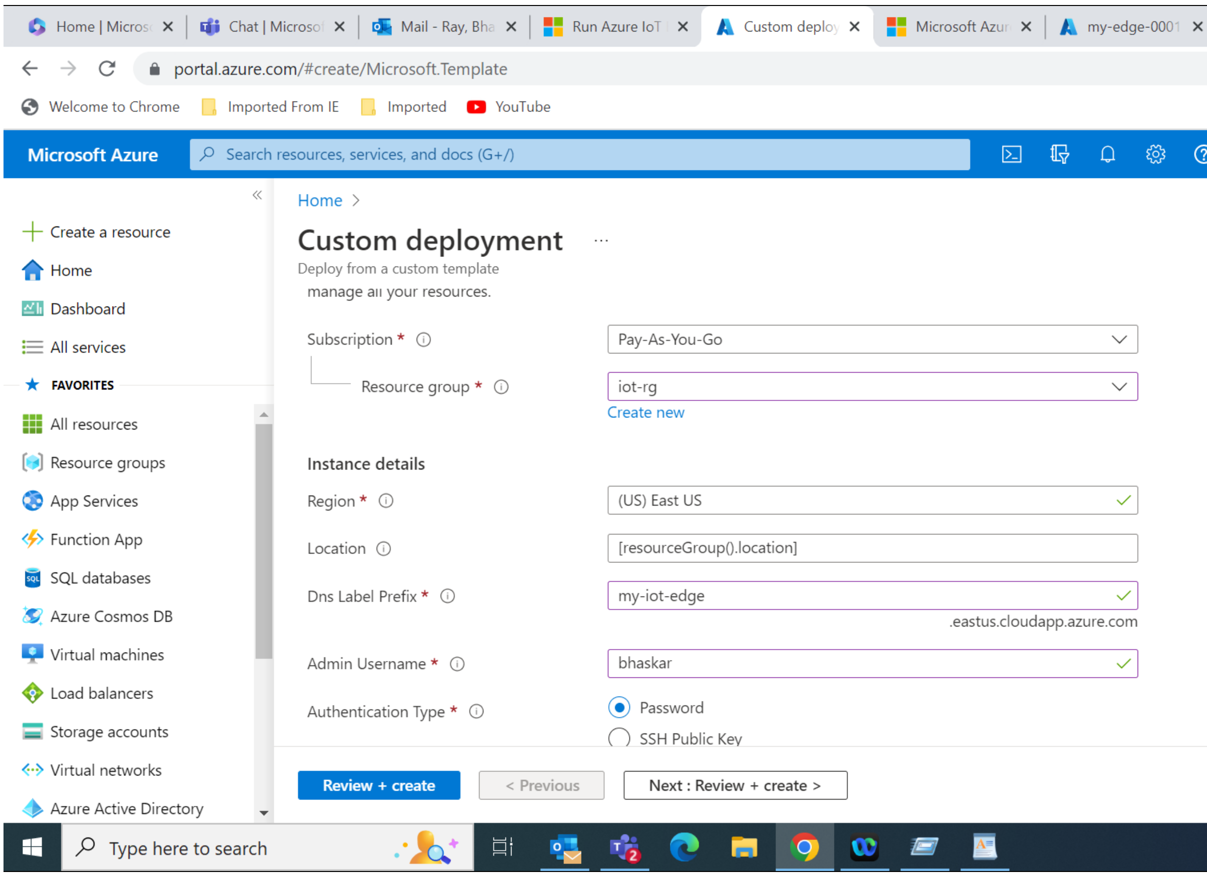
Then, create a IoT edge device, go to **IoT Edge->Add IoT Edge Device**.

Deploy a Linux VM into edge device

Now, in order to boot up the device as a IoT edge device, we need to install the IoT Edge runtime like IoT Edge agent and IoT Hub in it. We will deploy a Linux VM in the edge device and during boot up it will install the IoT edge runtime. We will use the Azure custom deployment template using the following link which will use ARM template to deploy the resource from a GitHub project.

https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2Fazure%2Fiotedge-vm-deploy%2F1.4%2FedgeDeploy.json

and fill up the required details to deploy the resource into Azure as per following screenshots (Give the device connection string as the IoT edge device primary connection string from the portal):


portal.azure.com/#create/Microsoft.Template

Microsoft Azure

Custom deployment

Deploy from a custom template

manage all your resources.

Subscription * : Pay-As-You-Go

Resource group * : iot-rg

Create new

Instance details

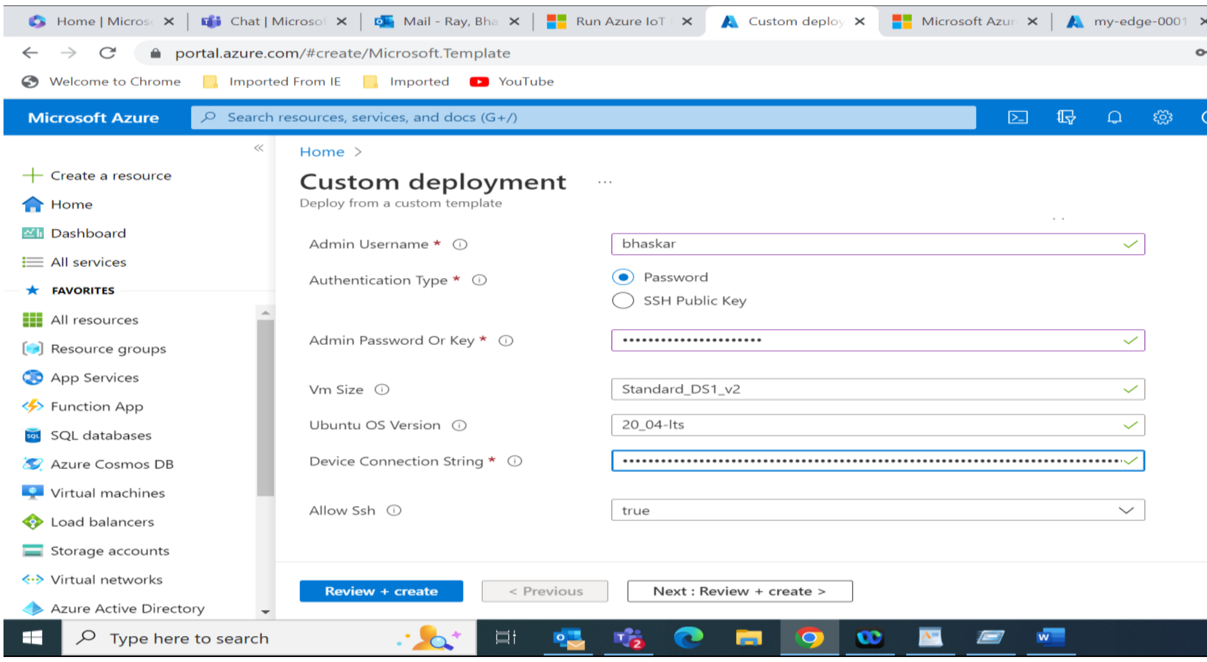
Region * : (US) East US

Location : [resourceGroup().location]

Dns Label Prefix * : my-iot-edge .eastus.cloudapp.azure.com

Admin Username * : bhaskar

Authentication Type * : [✓ Password] SSH Public Key


portal.azure.com/#create/Microsoft.Template

Microsoft Azure

Custom deployment

Deploy from a custom template

Admin Username * : bhaskar

Authentication Type * : [✓ Password] SSH Public Key

Admin Password Or Key * : ••••••••••••••••••••••

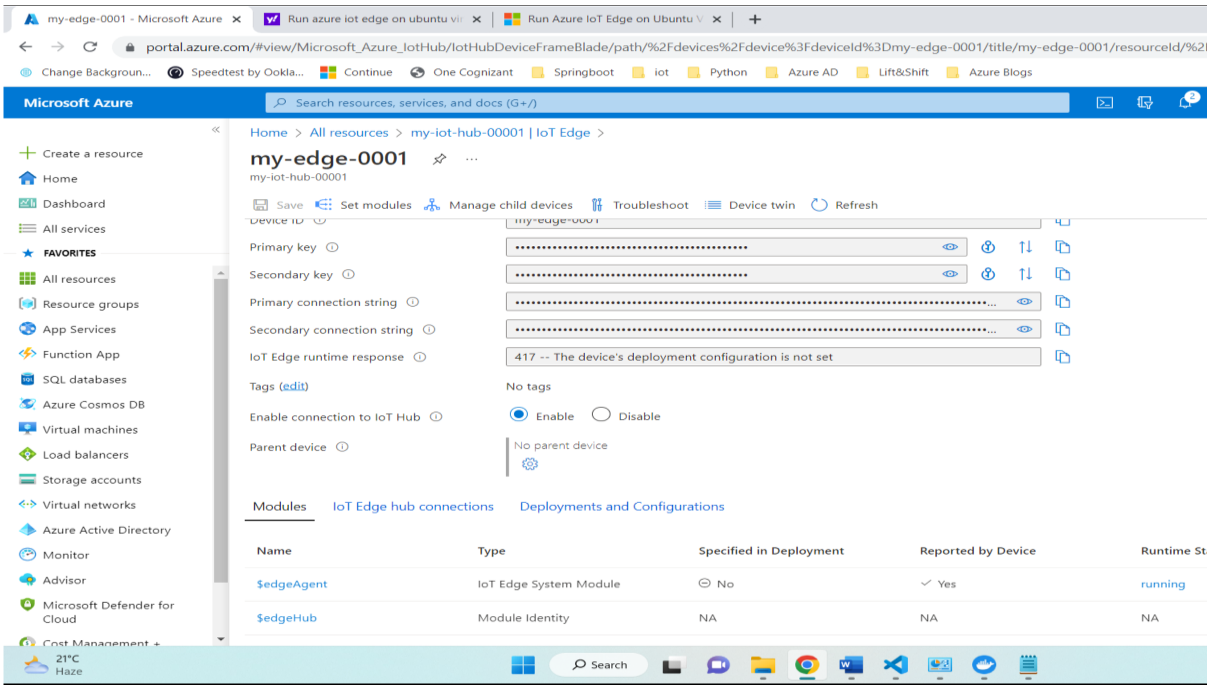
Vm Size : Standard_DS1_v2

Ubuntu OS Version : 20_04-lts

Device Connection String * : ••

Allow Ssh : true

Once the VM is deployed into Azure, you will see IoT edge agent is running in list of modules in IoT edge devices:


Home > All resources > my-iot-hub-00001 | IoT Edge >

my-edge-0001

my-iot-hub-00001

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

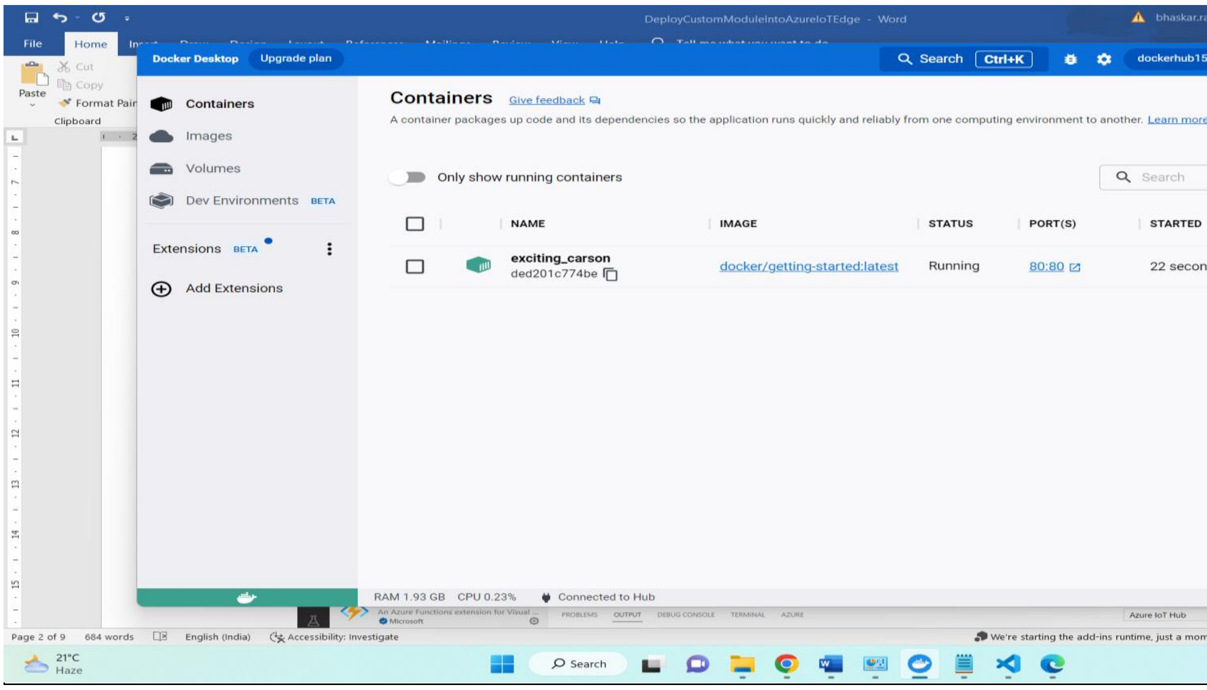
Device ID	my-edge-0001
Primary key	••
Secondary key	••
Primary connection string	••...
Secondary connection string	••...
IoT Edge runtime response	417 -- The device's deployment configuration is not set
Tags (edit)	No tags
Enable connection to IoT Hub	[✓ Enable] Disable
Parent device	No parent device

Modules | IoT Edge hub connections | Deployments and Configurations

Name	Type	Specified in Deployment	Reported by Device	Runtime St
$edgeAgent	IoT Edge System Module	No	✓ Yes	running
$edgeHub	Module Identity	NA	NA	NA

Create a IoT edge solution template project from VS code, build a container image of it and push to docker container registry

Now, launch docker engine from desktop and start the container and you should see docker container is running at localhost port 80 as per below screenshot:


Containers Give feedback

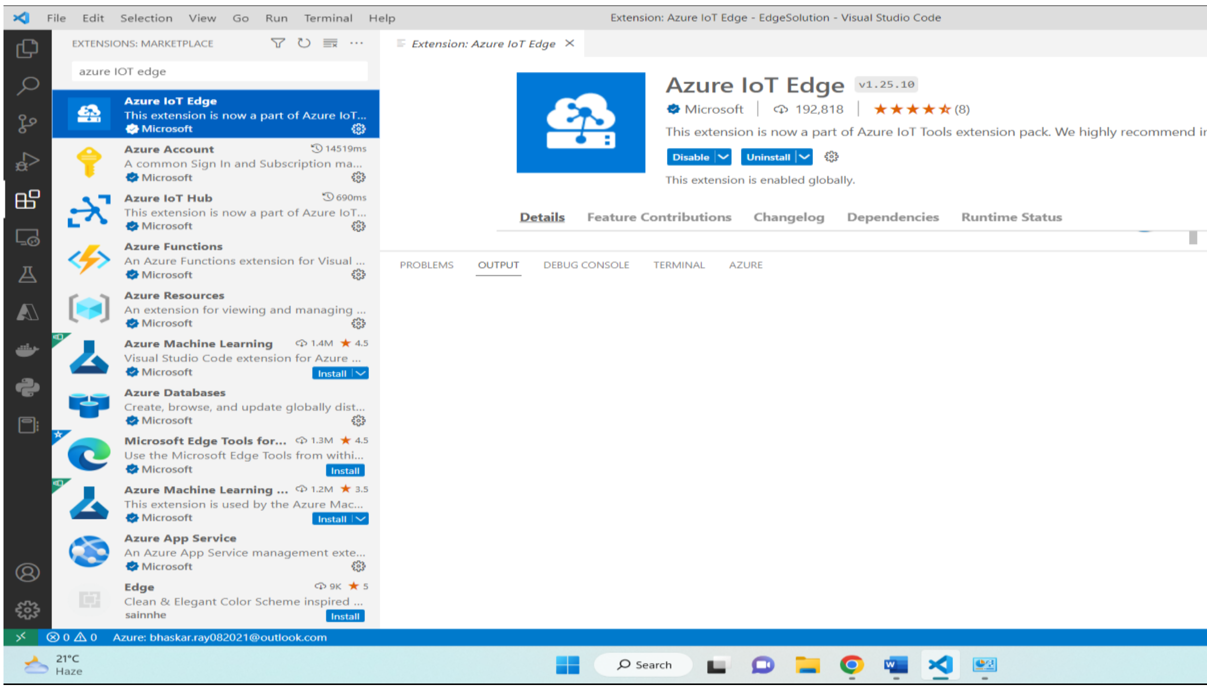
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only show running containers

NAME	IMAGE	STATUS	PORT(S)	STARTED
exciting_carson ded201c774be	docker/getting-started:latest	Running	80:80	22 secon

RAM 1.93 GB CPU 0.23% Connected to Hub

Then, Open VS Code and install Azure IoT Edge extension for building a IoT edge solution from a predefined VS code template available from Microsoft.



Azure IoT Edge
This extension is now a part of Azure IoT...
Microsoft

Azure Account 14519ms
A common Sign In and Subscription ma...
Microsoft

Azure IoT Hub 690ms
This extension is now a part of Azure IoT...
Microsoft

Azure Functions
An Azure Functions extension for Visual ...
Microsoft

Azure Resources
An extension for viewing and managing ...
Microsoft

Azure Machine Learning 1.4M ★ 4.5
Visual Studio Code extension for Azure ...
Microsoft Install

Azure Databases
Create, browse, and update globally dist...
Microsoft

Microsoft Edge Tools for... 1.3M ★ 4.5
Use the Microsoft Edge Tools from withi...
Microsoft Install

Azure Machine Learning ... 1.2M ★ 3.5
This extension is used by the Azure Mac...
Microsoft Install

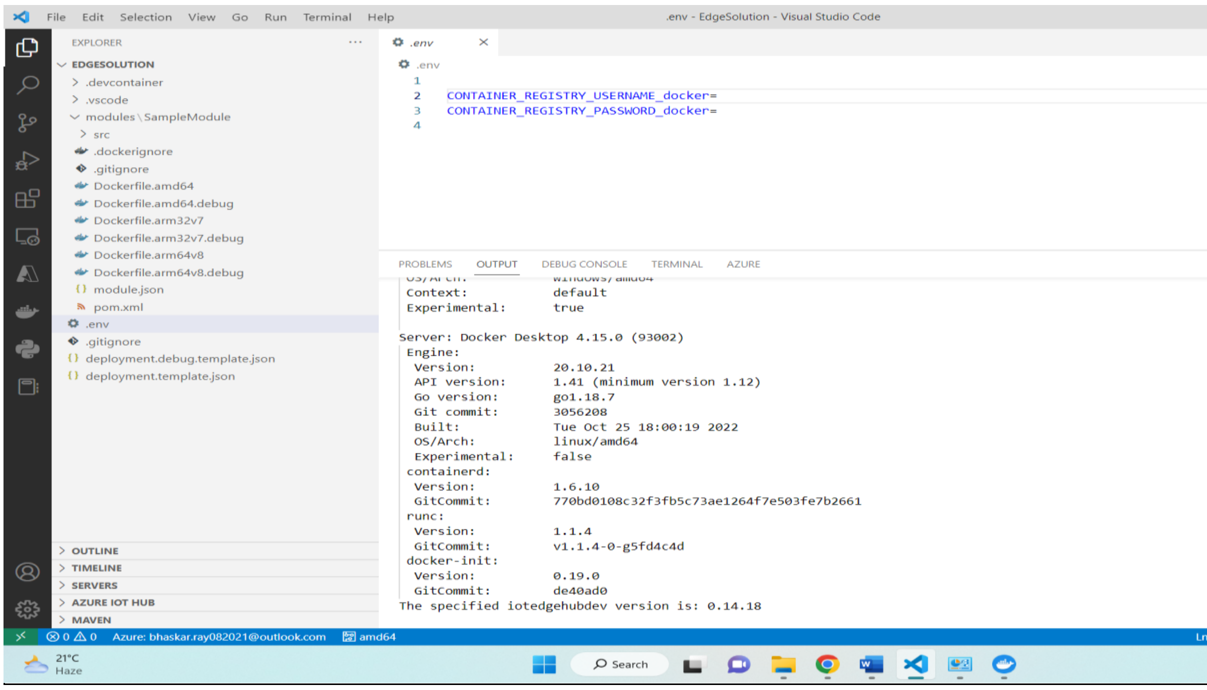
Azure App Service
An Azure App Service management exte...
Microsoft

Edge 9K ★ 5
Clean & Elegant Color Scheme inspired ...
sainnhe Install

This extension is now a part of Azure IoT Tools extension pack. We highly recommend i

This extension is enabled globally.

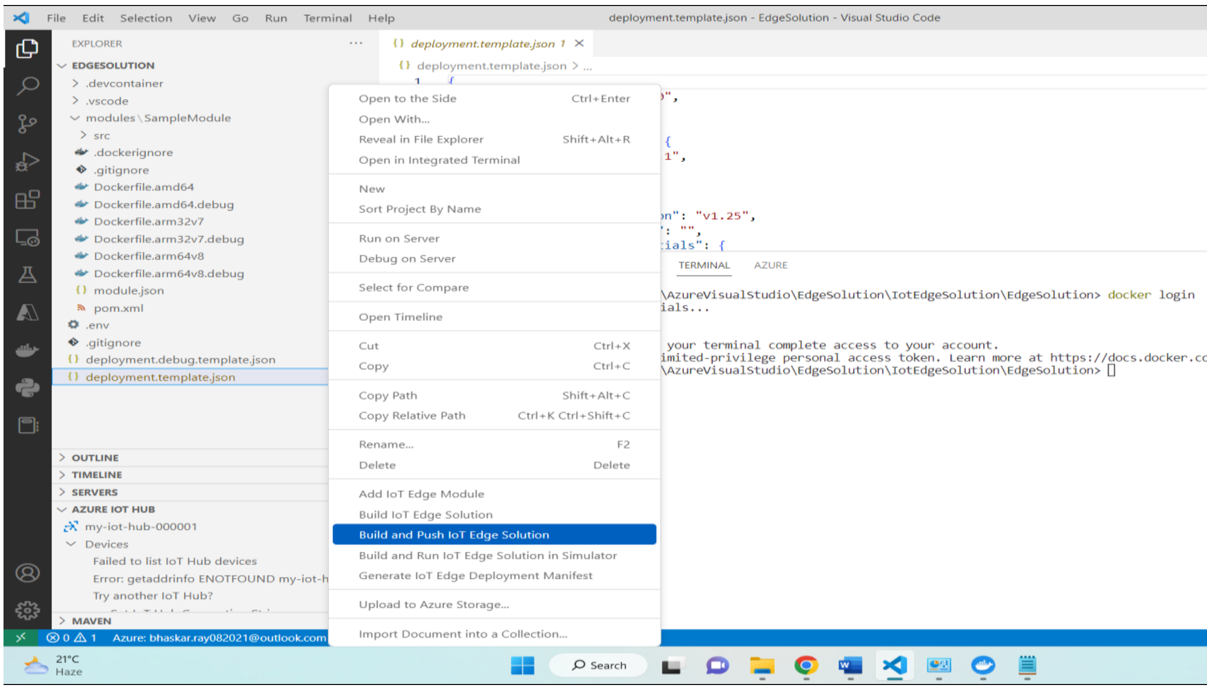
Go to View menu and open Command Palette, then type Azure IoT edge and select "**Azure IoT edge: new IoT edge solution**"->Select a path where you will keep the code in your local directory->name the project as **EdgeSolution**->Select any language as per your choice(in my case it's "**Java module use Azure IoT java SDK to build a module**")-> name the module as **SampleModule**->give the path of docker container registry, in my case I have given my docker hub account name in place of **~~docker.io~~/samplemodule**->package name **com.edgemodule** and press enter and it will create and build a new maven project and you will get a prompt to given your docker hub/ACR credentials in .env file as below:



```

CONTAINER_REGISTRY_USERNAME_docker=
CONTAINER_REGISTRY_PASSWORD_docker=

```

```
 Context:               default
 Experimental:          true

Server: Docker Desktop 4.15.0 (93002)
 Engine:
  Version:              20.10.21
  API version:          1.41 (minimum version 1.12)
  Go version:           go1.18.7
  Git commit:           3056208
  Built:                Tue Oct 25 18:00:19 2022
  OS/Arch:              linux/amd64
  Experimental:         false
 containerd:
  Version:              1.6.10
  GitCommit:            770bd0108c32f3fb5c73ae1264f7e503fe7b2661
 runc:
  Version:              1.1.4
  GitCommit:            v1.1.4-0-g5fd4c4d
 docker-init:
  Version:              0.19.0
  GitCommit:            de40ad0
The specified iotedgehubdev version is: 0.14.18
```

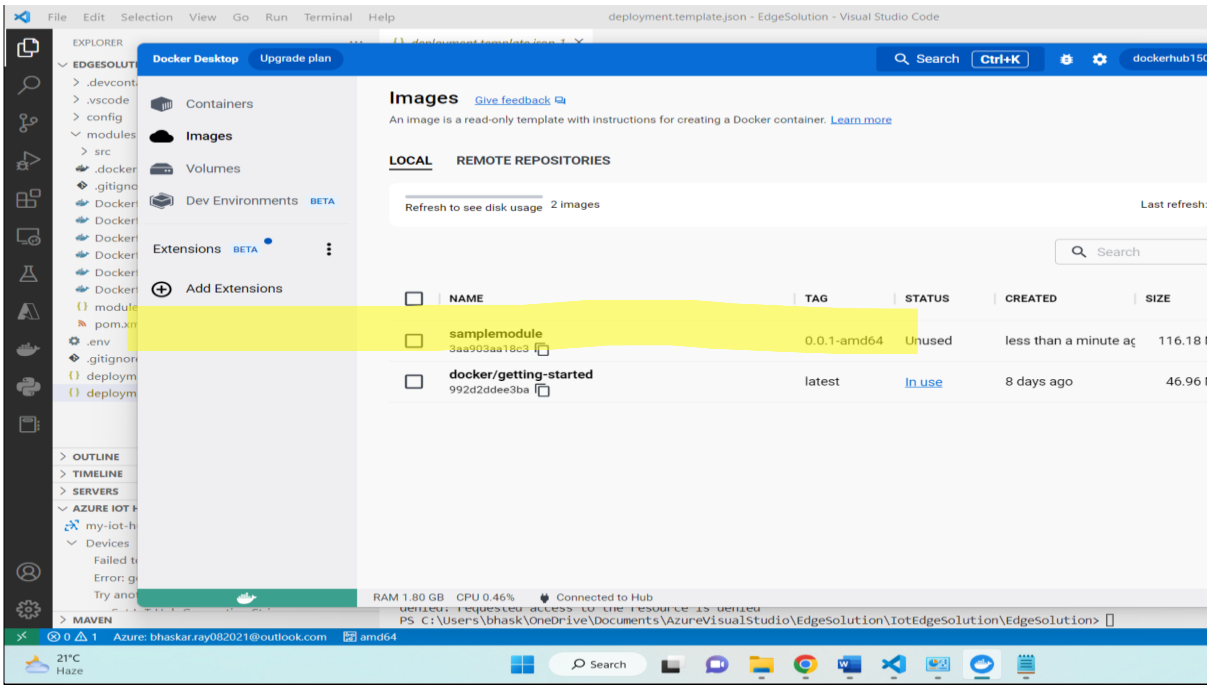
Now open Terminal from top Menu and type in “**docker login**”. Once you logged in successfully, right click on **deployment.template.json** in the project and click on **Build and push IoT edge solution**



```
\AzureVisualStudio\EdgeSolution\IotEdgeSolution\EdgeSolution> docker login
ials...

 your terminal complete access to your account.
imited-privilege personal access token. Learn more at https://docs.docker.c
\AzureVisualStudio\EdgeSolution\IotEdgeSolution\EdgeSolution> []
```

This will start creating docker images of the project and push it to docker hub.

Now, open the docker and validate if you see the pushed images in docker repository or not as per below screenshot:


Docker Desktop Upgrade plan
Search Ctrl+K
dockerhub15
Containers
Images
Volumes
Dev Environments BETA
Extensions BETA
Add Extensions
LOCAL REMOTE REPOSITORIES
Refresh to see disk usage
Last refresh
Search
RAM 1.80 GB CPU 0.46% Connected to Hub
Azure: bhaskar.ray082021@outlook.com amd64
21°C Haze

Images Give feedback

An image is a read-only template with instructions for creating a Docker container. Learn more

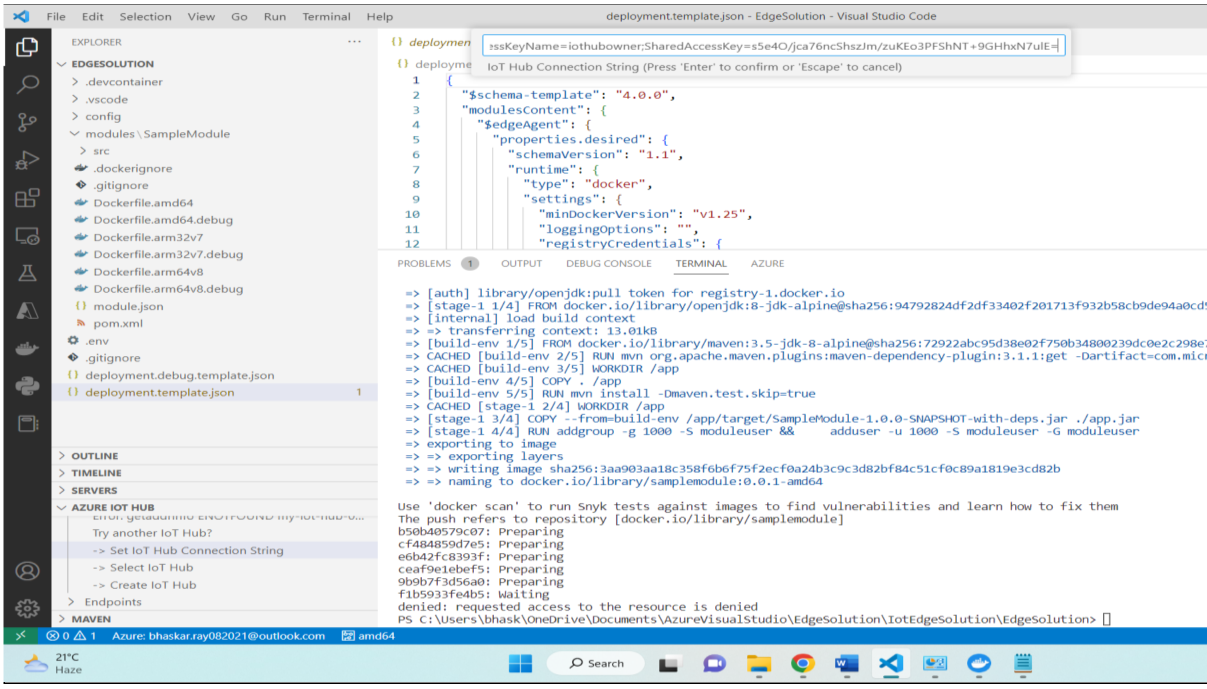
2 images

NAME	TAG	STATUS	CREATED	SIZE
samplemodule 3aa903aa18c3	0.0.1-amd64	Unused	less than a minute ag	116.18
docker/getting-started 992d2ddee3ba	latest	In use	8 days ago	46.96

```
denied: requested access to the resource is denied
PS C:\Users\bhask\OneDrive\Documents\AzureVisualStudio\EdgeSolution\IotEdgeSolution\EdgeSolution> []
```

Deploy custom module into IoT edge:

Connect to Azure IoT hub from VS code as per below screenshot by providing IoT hub connection string from **IoT Hub-> Shared Access Keys->IoTHubOwner->Primary Connection String**



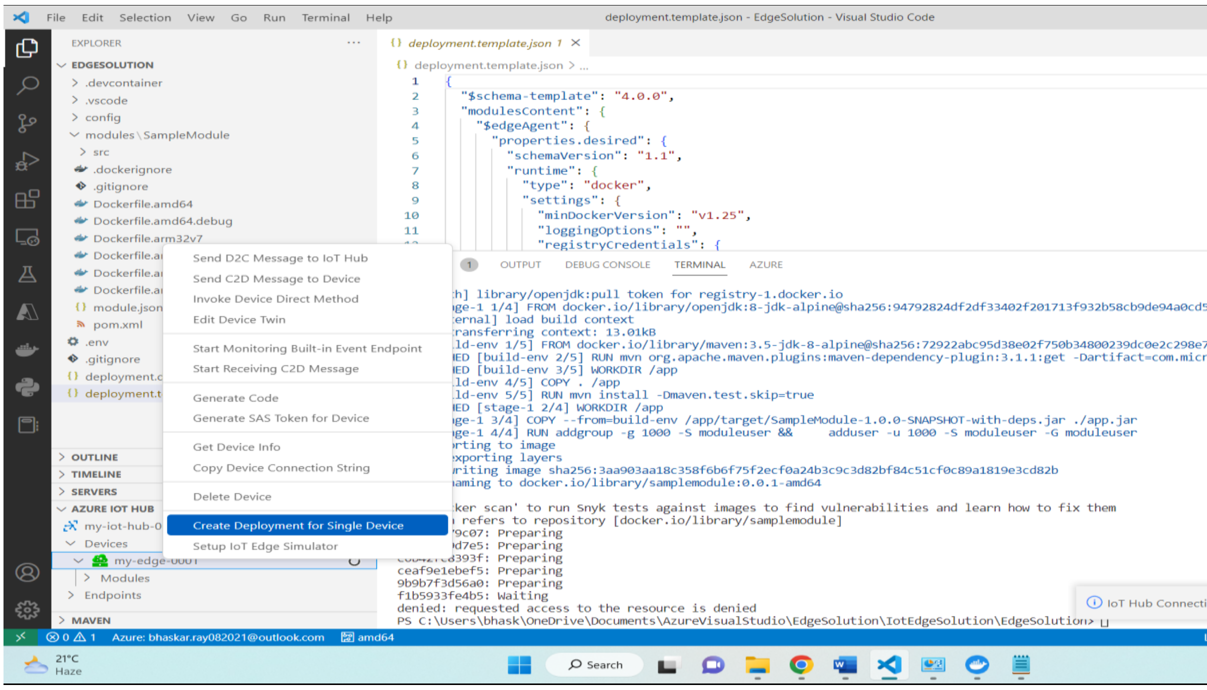
essKeyName=iothubowner;SharedAccessKey=s5e4O/jca76ncShszJm/zuKEo3PFShNT+9GHhxN7ulE=

IoT Hub Connection String (Press 'Enter' to confirm or 'Escape' to cancel)

```
{
  "$schema-template": "4.0.0",
  "modulesContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.1",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion": "v1.25",
            "loggingOptions": "",
            "registryCredentials": {
```

```
=> [auth] library/openjdk:pull token for registry-1.docker.io
=> [stage-1 1/4] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd
=> [internal] load build context
=> => transferring context: 13.01kB
=> [build-env 1/5] FROM docker.io/library/maven:3.5-jdk-8-alpine@sha256:72922abc95d38e02f750b34800239dc0e2c298e
=> CACHED [build-env 2/5] RUN mvn org.apache.maven.plugins:maven-dependency-plugin:3.1.1:get -Dartifact=com.mic
=> CACHED [build-env 3/5] WORKDIR /app
=> [build-env 4/5] COPY . /app
=> [build-env 5/5] RUN mvn install -Dmaven.test.skip=true
=> CACHED [stage-1 2/4] WORKDIR /app
=> [stage-1 3/4] COPY --from=build-env /app/target/SampleModule-1.0.0-SNAPSHOT-with-deps.jar ./app.jar
=> [stage-1 4/4] RUN addgroup -g 1000 -S moduleuser &&     adduser -u 1000 -S moduleuser -G moduleuser
=> exporting to image
=> => exporting layers
=> => writing image sha256:3aa903aa18c358f6b6f75f2ecf0a24b3c9c3d82bf84c51cf0c89a1819e3cd82b
=> => naming to docker.io/library/samplemodule:0.0.1-amd64

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
The push refers to repository [docker.io/library/samplemodule]
b50b40579c07: Preparing
cf484859d7e5: Preparing
e6b42fc8393f: Preparing
ceaf9e1ebef5: Preparing
9b9b7f3d56a0: Preparing
f1b5933fe4b5: Waiting
denied: requested access to the resource is denied
PS C:\Users\bhask\OneDrive\Documents\AzureVisualStudio\EdgeSolution\IotEdgeSolution\EdgeSolution> 
```

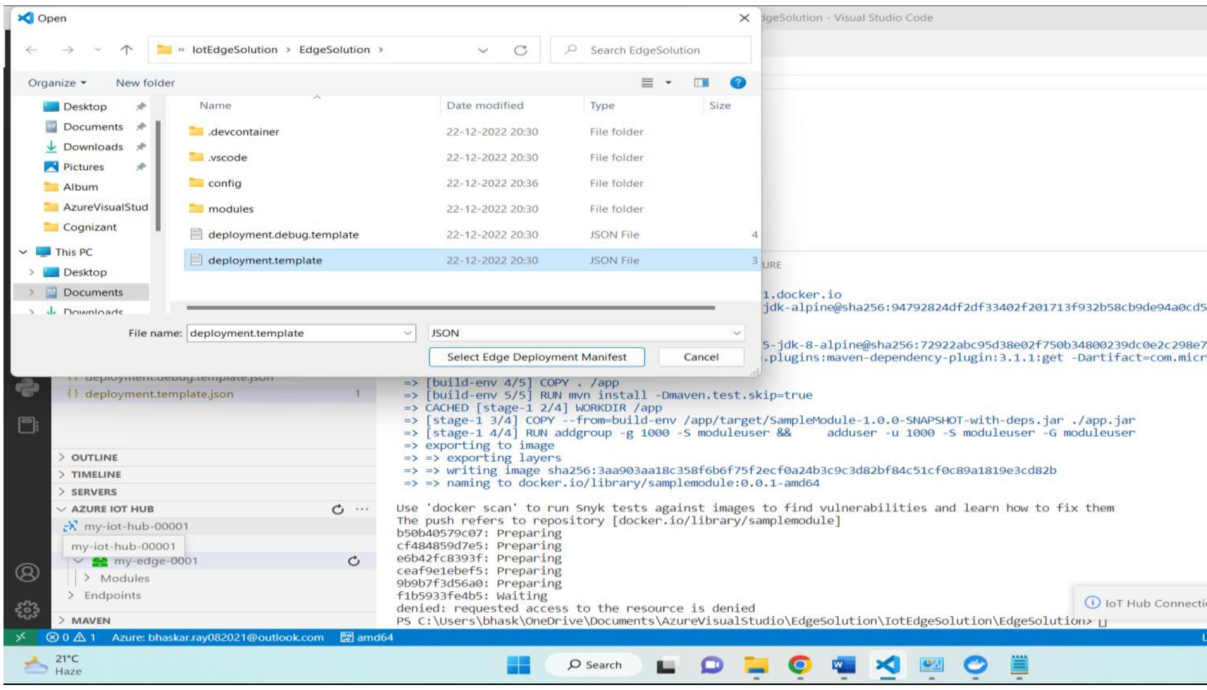
Now, you are connected to IoT hub, right click on IoT edge device and click on Create deployment for single device as we are deploying to a single edge device for now.



```
{
  "$schema-template": "4.0.0",
  "modulesContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.1",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion": "v1.25",
            "loggingOptions": "",
            "registryCredentials": {
```

```
h] library/openjdk:pull token for registry-1.docker.io
ge-1 1/4] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd
ernal] load build context
ransferring context: 13.01kB
ld-env 1/5] FROM docker.io/library/maven:3.5-jdk-8-alpine@sha256:72922abc95d38e02f750b34800239dc0e2c298e
ED [build-env 2/5] RUN mvn org.apache.maven.plugins:maven-dependency-plugin:3.1.1:get -Dartifact=com.mic
ED [build-env 3/5] WORKDIR /app
ld-env 4/5] COPY . /app
ld-env 5/5] RUN mvn install -Dmaven.test.skip=true
ED [stage-1 2/4] WORKDIR /app
ge-1 3/4] COPY --from=build-env /app/target/SampleModule-1.0.0-SNAPSHOT-with-deps.jar ./app.jar
ge-1 4/4] RUN addgroup -g 1000 -S moduleuser &&     adduser -u 1000 -S moduleuser -G moduleuser
rting to image
xporting layers
riting image sha256:3aa903aa18c358f6b6f75f2ecf0a24b3c9c3d82bf84c51cf0c89a1819e3cd82b
aming to docker.io/library/samplemodule:0.0.1-amd64

ker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
 refers to repository [docker.io/library/samplemodule]
9c07: Preparing
d7e5: Preparing
c8393f: Preparing
ceaf9e1ebef5: Preparing
9b9b7f3d56a0: Preparing
f1b5933fe4b5: Waiting
denied: requested access to the resource is denied
PS C:\Users\bhask\OneDrive\Documents\AzureVisualStudio\EdgeSolution\IotEdgeSolution\EdgeSolution> 
```

And then select deployment template as per below screenshot:



Name	Date modified	Type	Size
.devcontainer	22-12-2022 20:30	File folder	
.vscode	22-12-2022 20:30	File folder	
config	22-12-2022 20:36	File folder	
modules	22-12-2022 20:30	File folder	
deployment.debug.template	22-12-2022 20:30	JSON File	4
deployment.template	22-12-2022 20:30	JSON File	3

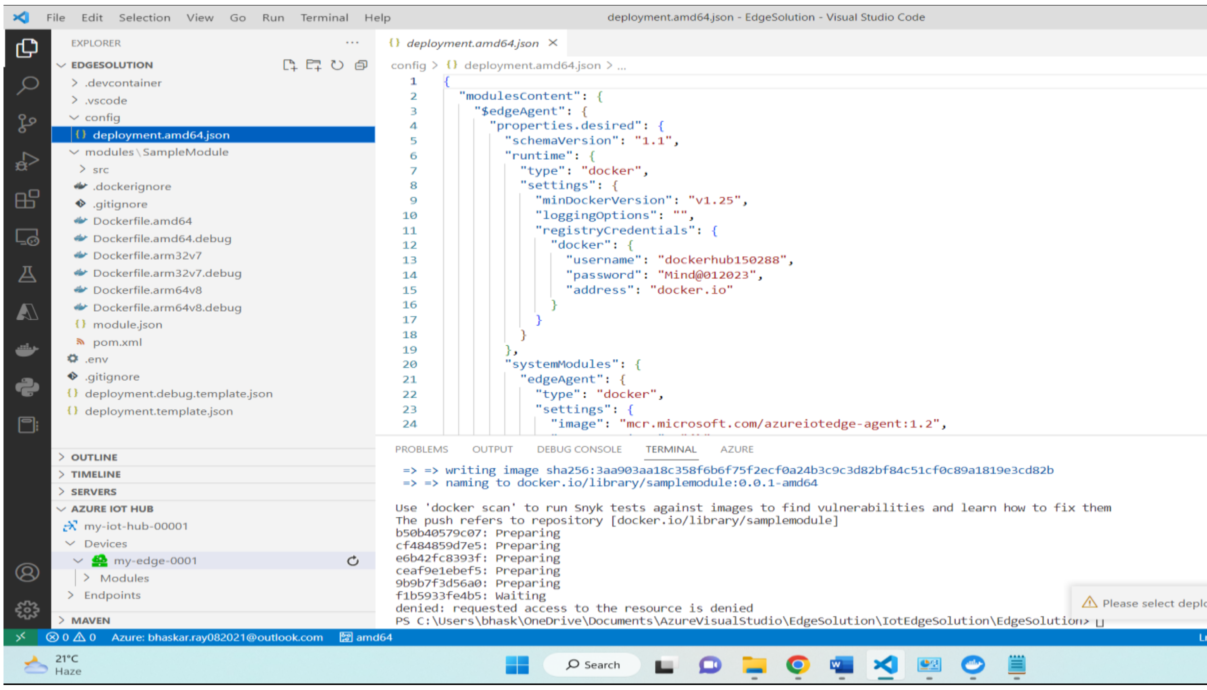
File name: deployment.template

JSON

Select Edge Deployment Manifest | Cancel

```
1.docker.io
jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd5

5-jdk-8-alpine@sha256:72922abc95d38e02f750b34800239dc0e2c298e7
.plugins:maven-dependency-plugin:3.1.1:get -Dartifact=com.micr
 => [build-env 4/5] COPY . /app
 => [build-env 5/5] RUN mvn install -Dmaven.test.skip=true
 => CACHED [stage-1 2/4] WORKDIR /app
 => [stage-1 3/4] COPY --from=build-env /app/target/SampleModule-1.0.0-SNAPSHOT-with-deps.jar ./app.jar
 => [stage-1 4/4] RUN addgroup -g 1000 -S moduleuser &&     adduser -u 1000 -S moduleuser -G moduleuser
 => exporting to image
 => => exporting layers
 => => writing image sha256:3aa903aa18c358f6b6f75f2ecf0a24b3c9c3d82bf84c51cf0c89a1819e3cd82b
 => => naming to docker.io/library/samplemodule:0.0.1-amd64

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
The push refers to repository [docker.io/library/samplemodule]
b50b40579c07: Preparing
cf484859d7e5: Preparing
e6b42fc8393f: Preparing
ceaf9e1ebef5: Preparing
9b9b7f3d56a0: Preparing
f1b5933fe4b5: Waiting
denied: requested access to the resource is denied
PS C:\Users\bhask\OneDrive\Documents\AzureVisualStudio\EdgeSolution\IotEdgeSolution\EdgeSolution>
```

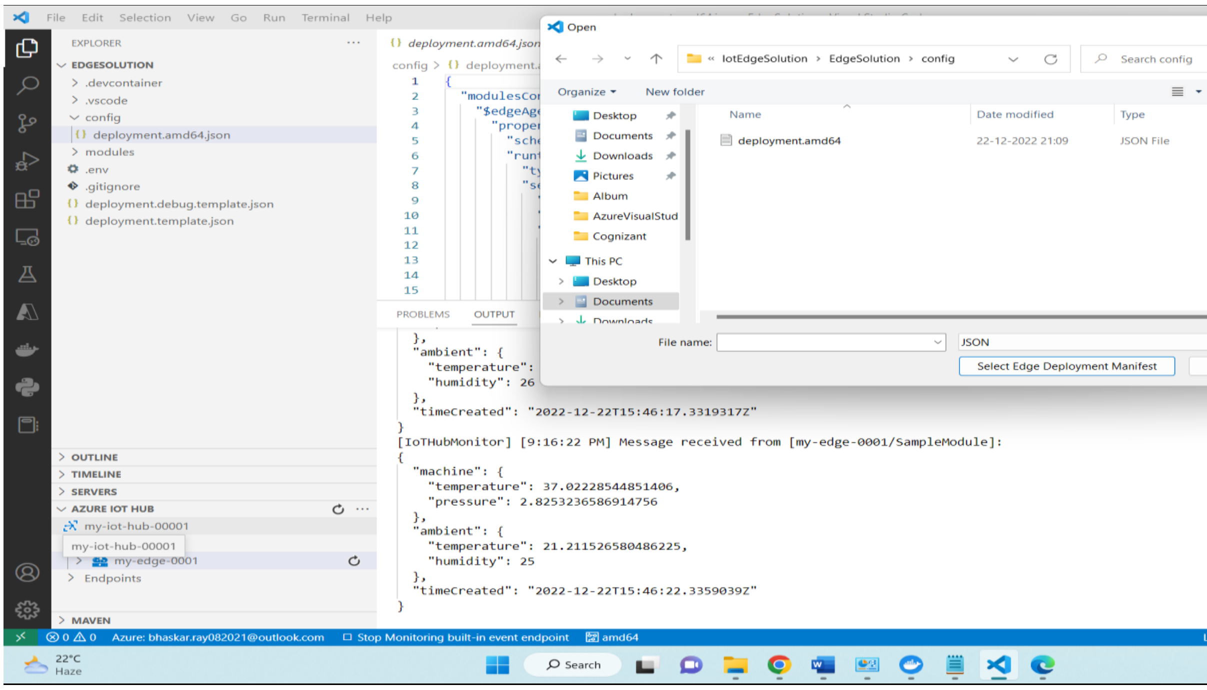
It will create a deployment manifest under config folder as below:



```
{
  "modulesContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.1",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion": "v1.25",
            "loggingOptions": "",
            "registryCredentials": {
              "docker": {
                "username": "dockerhub150288",
                "password": "Mind@012023",
                "address": "docker.io"
              }
            }
          }
        },
        "systemModules": {
          "edgeAgent": {
            "type": "docker",
            "settings": {
              "image": "mcr.microsoft.com/azureiotedge-agent:1.2",
```

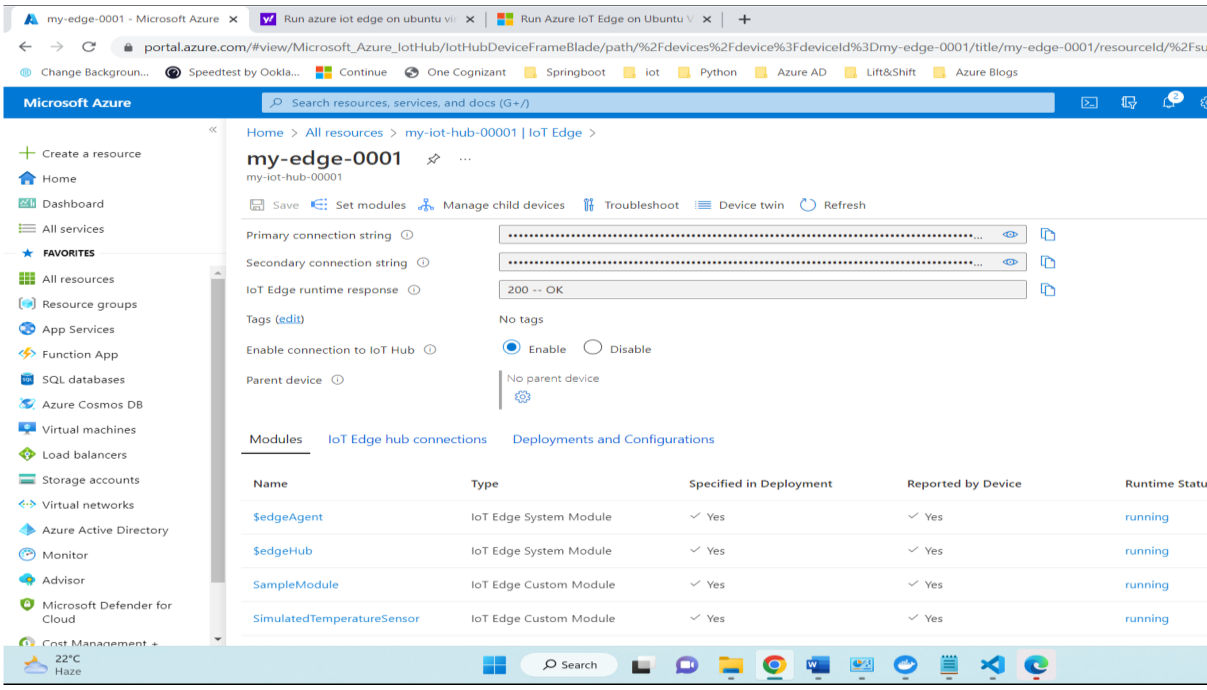
```
 => => writing image sha256:3aa903aa18c358f6b6f75f2ecf0a24b3c9c3d82bf84c51cf0c89a1819e3cd82b
 => => naming to docker.io/library/samplemodule:0.0.1-amd64

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
The push refers to repository [docker.io/library/samplemodule]
b50b40579c07: Preparing
cf484859d7e5: Preparing
e6b42fc8393f: Preparing
ceaf9e1ebef5: Preparing
9b9b7f3d56a0: Preparing
f1b5933fe4b5: Waiting
denied: requested access to the resource is denied
PS C:\Users\bhask\OneDrive\Documents\AzureVisualStudio\EdgeSolution\IotEdgeSolution\EdgeSolution>
```

After that, right click again on azure IoT edge device and click on Create deployment for single device and then choose the deployment manifest file under Config folder as per below screenshot:



```
  },
  "ambient": {
    "temperature":
    "humidity": 26
  },
  "timeCreated": "2022-12-22T15:46:17.3319317Z"
}
[IoTHubMonitor] [9:16:22 PM] Message received from [my-edge-0001/SampleModule]:
{
  "machine": {
    "temperature": 37.02228544851406,
    "pressure": 2.8253236586914756
  },
  "ambient": {
    "temperature": 21.211526580486225,
    "humidity": 25
  },
  "timeCreated": "2022-12-22T15:46:22.3359039Z"
}
```

It will deploy the custom **samplemodule** to edge device successfully and then you will be able to see all four modules are running in edge device as per following screenshot:


my-edge-0001 - Microsoft Azure
Run azure iot edge on ubuntu vi
Run Azure IoT Edge on Ubuntu V
portal.azure.com/#view/Microsoft_Azure_IotHub/IotHubDeviceFrameBlade/path/%2Fdevices%2Fdevice%3FdeviceId%3Dmy-edge-0001/title/my-edge-0001/resourceId/%2Fsu
Microsoft Azure
Search resources, services, and docs (G+/)
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Azure Active Directory
Monitor
Advisor
Microsoft Defender for Cloud
Cost Management +
Home > All resources > my-iot-hub-00001 | IoT Edge >
Save
Set modules
Manage child devices
Troubleshoot
Device twin
Refresh
22°C Haze
Search

my-edge-0001

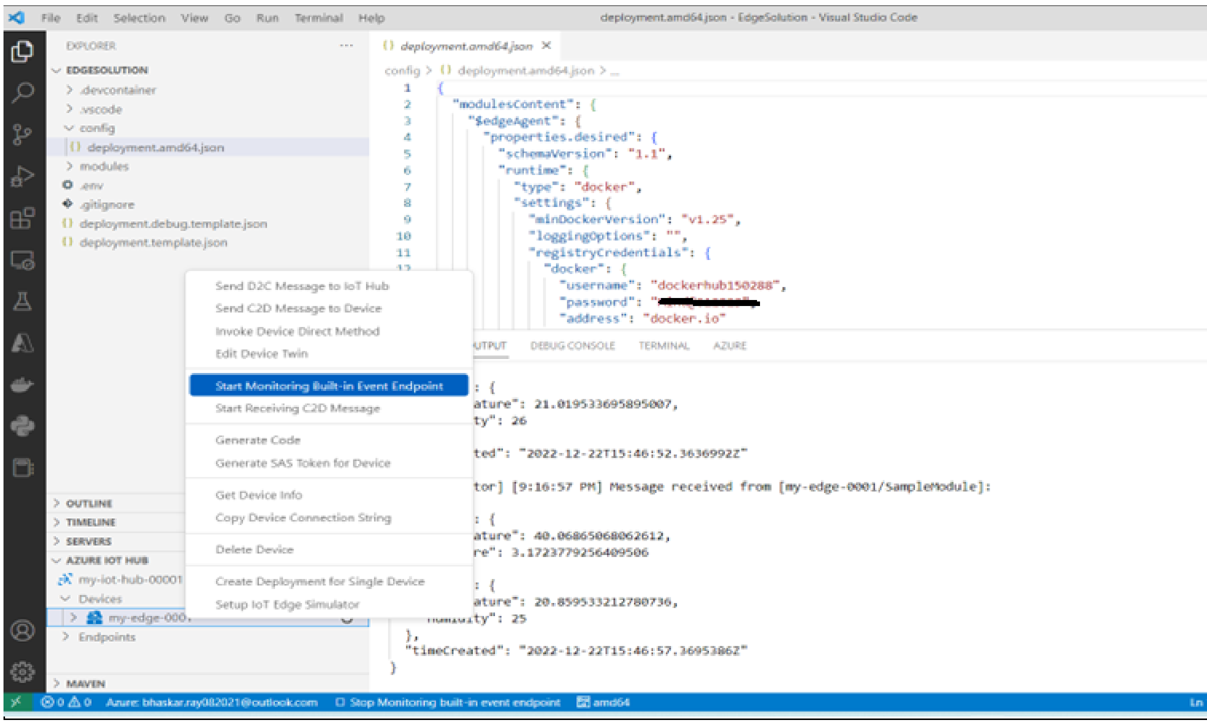
my-iot-hub-00001

Primary connection string	••
Secondary connection string	••
IoT Edge runtime response	200 -- OK
Tags (edit)	No tags
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 Statu
$edgeAgent	IoT Edge System Module	✓ Yes	✓ Yes	running
$edgeHub	IoT Edge System Module	✓ Yes	✓ Yes	running
SampleModule	IoT Edge Custom Module	✓ Yes	✓ Yes	running
SimulatedTemperatureSensor	IoT Edge Custom Module	✓ Yes	✓ Yes	running

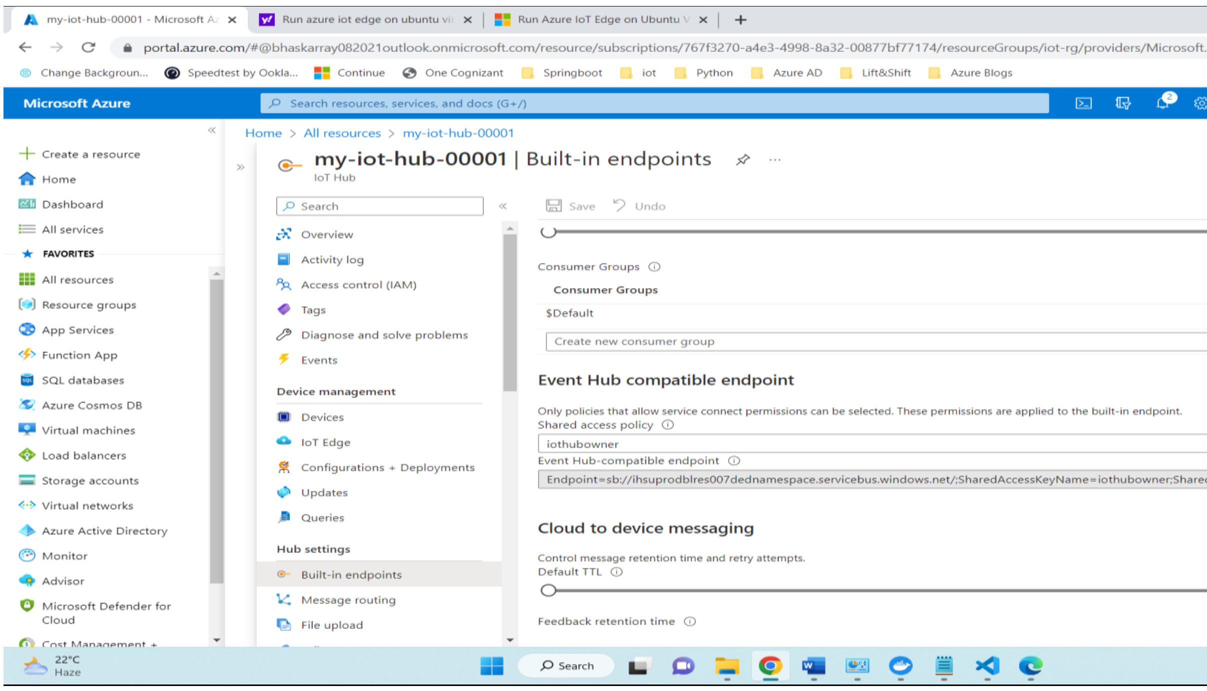
Now, as per the route defined in the deployment manifest, the temperature sensor module will start sending messages into the **samplemodule** and **samplemodule** will start sending sensor data into IoT hub. In order to monitor the telemetry data coming into IoT hub, we should right click on edge device and click on **start monitoring Built-in Event endpoint** as below:


File Edit Selection View Go Run Terminal Help
deployment.amd64.json - EdgeSolution - Visual Studio Code
EXPLORER
EDGESOLUTION
.devcontainer
.vscode
config
deployment.amd64.json
modules
.env
.gitignore
deployment.debug.template.json
deployment.template.json
OUTLINE
TIMELINE
SERVERS
AZURE IOT HUB
my-iot-hub-00001
Devices
my-edge-000
Endpoints
MAVEN
Send D2C Message to IoT Hub
Send C2D Message to Device
Invoke Device Direct Method
Edit Device Twin
Start Monitoring Built-in Event Endpoint
Start Receiving C2D Message
Generate Code
Generate SAS Token for Device
Get Device Info
Copy Device Connection String
Delete Device
Create Deployment for Single Device
Setup IoT Edge Simulator
deployment.amd64.json
config > deployment.amd64.json > ...
UTPUT
DEBUG CONSOLE
TERMINAL
AZURE
Azure: bhaskar.ray082021@outlook.com
Stop Monitoring built-in event endpoint
amd64
Ln

```
{
  "modulesContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.1",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion": "v1.25",
            "loggingOptions": "",
            "registryCredentials": {
              "docker": {
                "username": "dockerhub150288",
                "password": "[redacted]
                "address": "docker.io"
```

```
: {
ature": 21.019533695895007,
ty": 26

ted": "2022-12-22T15:46:52.3636992Z"

tor] [9:16:57 PM] Message received from [my-edge-0001/SampleModule]:

: {
ature": 40.06865068062612,
re": 3.1723779256409506

: {
ature": 20.859533212780736,
  humidity": 25
},
"timeCreated": "2022-12-22T15:46:57.3695386Z"
}
```

Then, we should give the IoT hub built in Event hub endpoint which can be found **IotHub->Built-in Endpoint** as per below screenshot:



my-iot-hub-00001 | Built-in endpoints
IoT Hub

Consumer Groups

Consumer Groups
$Default

Event Hub compatible endpoint

Only policies that allow service connect permissions can be selected. These permissions are applied to the built-in endpoint.

Shared access policy: iothubowner

Event Hub-compatible endpoint: Endpoint=sb://ihsuprodblres007dednamespace.servicebus.windows.net/;SharedAccessKeyName=iothubowner;Share

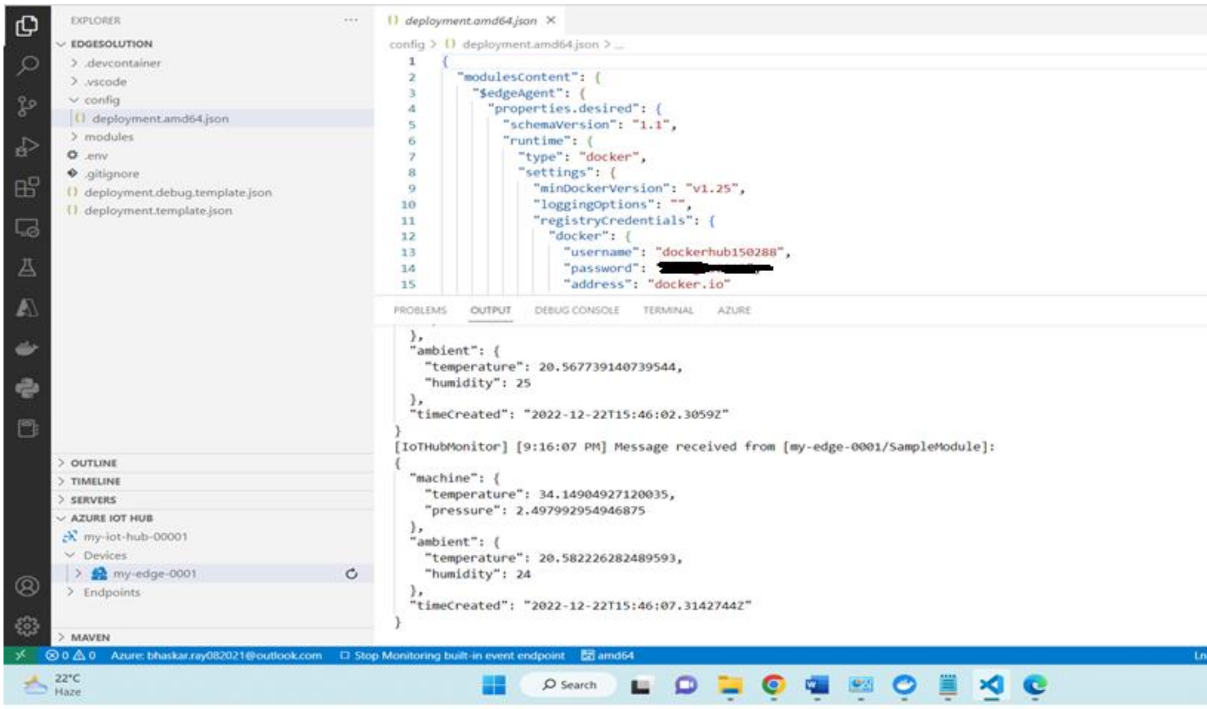
Cloud to device messaging

Control message retention time and retry attempts.

Default TTL

Feedback retention time

We are all set and will be able to view the sensor data that is consumed by Azure IoT hub now as per below screenshot:



deployment.amd64.json

```
{
  "modulesContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.1",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion": "v1.25",
            "loggingOptions": "",
            "registryCredentials": {
              "docker": {
                "username": "dockerhub150288",
                "password": [redacted]
                "address": "docker.io"
```

OUTPUT

```
  },
  "ambient": {
    "temperature": 20.567739140739544,
    "humidity": 25
  },
  "timeCreated": "2022-12-22T15:46:02.3059Z"
}
[IoTHubMonitor] [9:16:07 PM] Message received from [my-edge-0001/SampleModule]:
{
  "machine": {
    "temperature": 34.14904927120035,
    "pressure": 2.497992954946875
  },
  "ambient": {
    "temperature": 20.582226282489593,
    "humidity": 24
  },
  "timeCreated": "2022-12-22T15:46:07.3142744Z"
}
```

Challenges in implementing the solution:

Once we create the Azure IoT edge device in IoT hub, we will need to deploy the IoT edge runtime module inside the device via Linux Virtual machine.
Otherwise, the device will remain in offline state, and we will get error deploying our custom module in the device. So, before the deployment, we have
to boot up the device with IoT edge runtime installed in it either through ARM template custom deployment or Azure CLI etc or if it is a windows VM, we can RDP
to remotely connect to the VM and install the runtime agents through power shell.

Business Benefit:

Using VS code Azure IoT extension, the time to market will be greatly reduced to develop and deploy any custom IoT edge modules that can
written in any language as per our choice in C#, Java, Node.js or python etc. It can be utilized as a low code solution as well.

Code base:

The working code from VS code has been attached below in a zip format:

EdgeSolution.zip