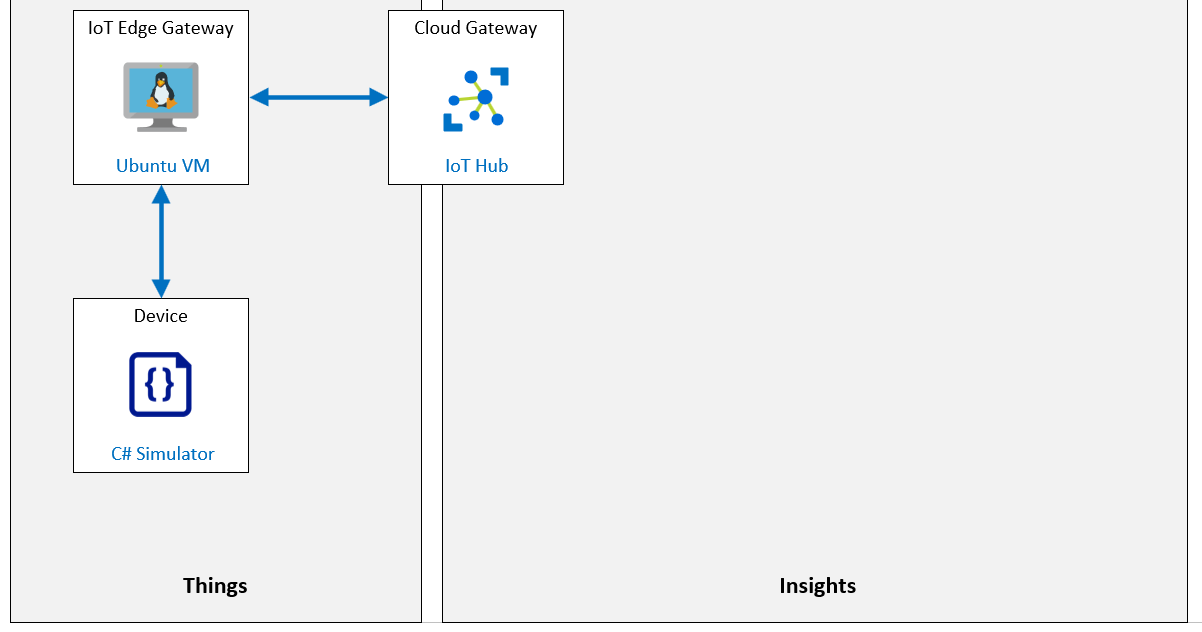
Setup an IoT Edge Gateway

There are **three patterns** for using an IoT Edge device as a gateway: **transparent**, **protocol translation**, and **identity translation**:

* **Transparent: devices and IoT Hub can communicate by using MQTT, AMQP, Http protocols through IoT Edge device without aware the existing of intermediate Device.**
* **Protocol translation: Devices do not support MQTT, AMQP, or HTTP can use gateway device to send data to IoT Hub on their behalf.**
* **Identity translation: Devices cannot to IoT Hub by using their Identity can connect to gateway device, instead. Their identities are accepted by gateway device.**



1. **Create Azure Resources:**

|  |  |
| --- | --- |
| Resource Group | rg-az220 |
| IoT Hub | iot-az220-testing-0001 |

1. **Deploy an Azure IoT Edge Enabled Linux VM as an IoT Edge Device**

**Resource group** dropdown, click **rg-az220vm**

**vm-az220-testing-gw0001-thien-0001**

**Image** field, select **Ubuntu Server 18.04 LTS - Gen1** image.

**VM Size**, click **Standard\_B1ms**

User/password.

1. **Generate and Configure IoT Edge Device CA Certificates**

#### **Task 1: Connect to the VM from Cloud Shell**

ssh [username@52.170.205.79](mailto:username@52.170.205.79)

#### **Task 2: Generate certificates**

To download and configure some Azure IoT Edge helper scripts using OpenSSL

git clone <https://github.com/Azure/iotedge.git>

To create a working directory named “lab12”

mkdir lab12

cd lab12

To copy helper scripts into the lab12 directory,

cp ../iotedge/tools/CACertificates/\*.cnf .

cp ../iotedge/tools/CACertificates/certGen.sh .

ls

To generate the root CA certificate and one intermediate certificate

./certGen.sh create\_root\_and\_intermediate

The script will have created several certificate and key files. You will be using the following root CA certificate file later in this lab:

# Root CA certificate

~/lab12/certs/azure-iot-test-only.root.ca.cert.pem

the IoT Edge device CA certificate and private key need to be generated.

./certGen.sh create\_edge\_device\_ca\_certificate "MyEdgeDeviceCA"

# Device CA certificate

~/lab12/certs/iot-edge-device-ca-MyEdgeDeviceCA-full-chain.cert.pem

# Device CA private key

~/lab12/private/iot-edge-device-ca-MyEdgeDeviceCA.key.pem

#### **Task 3: Add the Microsoft installation packages to the package manager**

To configure the VM to access the Microsoft installation packages

curl https://packages.microsoft.com/config/ubuntu/18.04/multiarch/prod.list > ./microsoft-prod.list

To add the downloaded package list to the package manager,

sudo cp ./microsoft-prod.list /etc/apt/sources.list.d/

To install the packages, the Microsoft GPG public key must be installed.

curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.gpg

sudo cp ./microsoft.gpg /etc/apt/trusted.gpg.d/

#### **Task 4: Install a container engine**

sudo apt-get update

sudo apt-get install moby-engine

#### **Task 5: Install IoT Edge**

sudo apt-get update

apt list -a iotedge

sudo apt-get install iotedge

sudo apt-get install iotedge=1.0.9-1 libiothsm-std=1.0.9-1

iotedge version

sudo apt-get remove iotedge

docker pull mcr.microsoft.com/azureiotedge-agent:1.0

docker pull mcr.microsoft.com/azureiotedge-hub:1.0

#### **Task 6: Configure IoT Edge**

sudo chmod a+w /etc/iotedge/config.yaml

sudo vi /etc/iotedge/config.yaml

To update the **certificates** code lines

certificates:

device\_ca\_cert: "/home/useradmin/lab12/certs/iot-edge-device-ca-MyEdgeDeviceCA-full-chain.cert.pem"

device\_ca\_pk: "/home/useradmin/lab12/private/iot-edge-device-ca-MyEdgeDeviceCA.key.pem"

trusted\_ca\_certs: "/home/useradmin/lab12/certs/azure-iot-test-only.root.ca.cert.pem"

The X.509 certificates configured in this section are used for the following purposes:

| Setting | Purpose |
| --- | --- |
| **device\_ca\_cert** | This is the Device CA Certificate for the IoT Edge Device. |
| **device\_ca\_pk** | This is the Device CA Private Key for the IoT Edge Device. |
| **trusted\_ca\_certs** | This is the Root CA Certificate. This certificate must contain all the trusted CA certificates required for Edge module communications. |

:wq!

systemctl restart iotedge

systemctl status iotedge

end the SSH session

exit

Next, you need to “download” the **MyEdgeDeviceCA** certificate from the **vm-az220-training-gw0001-{your-id}** virtual machine so that it can be used to configure the IoT Edge device enrollment within Azure IoT Hub Device Provisioning Service.

At the Cloud Shell command prompt, to download the **~/lab12** directory from the **vm-az220-training-gw0001-{your-id}** virtual machine to the **Cloud Shell** storage, enter the following commands:

mkdir lab12

scp -r -p useradmin@ 40.88.2.87:~/lab12 .

To verify that the files have been downloaded

cd lab12

ls

1. **Create an IoT Edge Device Identity in IoT Hub using Azure Portal**

**rg-az220>iot-az220-training-0001>IoT Edge**

**Add an IoT Edge device**

**Create a device**

**Device ID** field, enter **vm-az220-testing-gw0001-thien-0001**

**Auto-generate keys** box checked.

 copy the **Primary Connection String**.

notice that the list of **Modules** is limited to **$edgeAgent** and **$edgeHub**.

click **Set Modules:** The **Set module on device** blade can be used to add additional modules to an IoT Edge Device

click **Routes**

* **NAME**: route
* **VALUE**: FROM /\* INTO $upstream

### **Exercise 5: Setup IoT Edge Gateway Hostname**

1. open your IoT Edge virtual machine, click **vm-az220-testing-gw0001-0001**
2. On the **vm-az220-training-gw0001-{your-id}** blade, in the upper section, locate the **DNS name** field.
3. the **DNS name label** field, enter **vm-az220-testing-gw0001-thien-0001**

vm-az220-testing-gw0001-cah191230.westus2.cloudapp.azure.com

vm-az220-testing-gw0001.westus.cloudapp.azure.com

sudo vi /etc/iotedge/config.yaml

**ssh to vitual machine.**

1. Scroll down within the file to locate the **Edge device hostname** section.

**Note**: Here are some tips for using **vi** when editing the **config.yaml** file:

* + Press **Esc** and enter **/** followed by a search string, then press enter to search
  + Pressing **n** will cycle through matches.
  + Press the **i** key to put the editor into Insert mode, then you will be able to make changes.
  + Press **Esc** to go stop Insert mode and return to Normal mode.
  + To Save and Quit, type **:x**, and press **Enter**.
  + Save the file, type **:w**, and press **Enter**.
  + To quit vi, type **:quit** and press **Enter**.

1. Set the **hostname** value to the value of the **Full DNS name** that you saved earlier.

This is the **Full DNS name** of the **vm-az220-training-gw0001-{your-id}** virtual machine.

**Note**: If you did not save the name, you can find it in the **Overview** pane of the virtual machine. You can even copy it from there to paste into the Cloud Shell window.

The resulting value will look similar to the following:

CodeCopy

hostname: "vm-az220-training-gw0001-{your-id}.eastus.cloudapp.azure.com"

The hostname setting configures the Edge Hub server hostname. Regardless of the case used for this setting, a lowercase value is used to configure the Edge Hub server. This is also the **hostname** that downstream **IoT devices** will need to **use** when connecting to the **IoT Edge Gateway** for the **encrypted communication** to work properly.

1. Leave **config.yaml** open in vi/vim (or the editor you are using)

### **Exercise 6: Connect IoT Edge Gateway Device to IoT Hub**

In this exercise, you will connect the IoT Edge Device to Azure IoT Hub.

1. Return to the **config.yaml** document in vi/vim:
2. Find the **Manual provisioning configuration using a connection string** of the file and uncomment the Manual provisioning configuration using a connection string section, if it isn’t already uncommented by removing the leading **’# ‘** (pound symbol and space) characters and replace <ADD DEVICE CONNECTION STRING HERE> with the Connection String you copied previously for your IoT Edge Device:

CodeCopy

# Manual provisioning configuration using a connection string

provisioning:

source: "manual"

device\_connection\_string: "<ADD DEVICE CONNECTION STRING HERE>"

dynamic\_reprovisioning: false

**Important**: YAML treats spaces as significant characters. In the lines entered above, this means that there should not be any leading spaces in front of **provisioning:** and that there should be two leading spaces in front of **source:**, **device\_connection\_string:**, and **dynamic\_reprovisioning:**

1. To save your changes and exit the editor, press **Esc** and type **:x** and then press **Enter**
2. To apply the changes, the IoT Edge daemon must be restarted with the following command:

CodeCopy

systemctl restart iotedge

1. To ensure the IoT Edge daemon is running, enter the following command:

CodeCopy

sudo systemctl status iotedge

This command will display many lines of content, of which the first 3 lines indicate if the service is running. For a running service, the output will be similar to:

CodeCopy

● iotedge.service - Azure IoT Edge daemon

Loaded: loaded (/lib/systemd/system/iotedge.service; enabled; vendor preset: enabled)

Active: active (running) since Fri 2021-03-19 18:06:16 UTC; 1min 0s ago

1. To verify the IoT Edge runtime has connected, run the following command:

CodeCopy

sudo iotedge check

This runs a number of checks and displays the results. For this lab, ignore the **Configuration checks** warnings/errors. The **Connectivity checks** should succeed and be similar to:

CodeCopy

Connectivity checks

-------------------

√ host can connect to and perform TLS handshake with IoT Hub AMQP port - OK

√ host can connect to and perform TLS handshake with IoT Hub HTTPS / WebSockets port - OK

√ host can connect to and perform TLS handshake with IoT Hub MQTT port - OK

√ container on the default network can connect to IoT Hub AMQP port - OK

√ container on the default network can connect to IoT Hub HTTPS / WebSockets port - OK

√ container on the default network can connect to IoT Hub MQTT port - OK

√ container on the IoT Edge module network can connect to IoT Hub AMQP port - OK

√ container on the IoT Edge module network can connect to IoT Hub HTTPS / WebSockets port - OK

√ container on the IoT Edge module network can connect to IoT Hub MQTT port - OK

If the connection fails, double-check the connection string value in **config.yaml**.

1. Wait a few moments.
2. To list all the **IoT Edge Modules** currently running on the IoT Edge Device, enter the following command:

ShellCopy

iotedge list

After a short time, this command will show the edgeAgent and edgeHub modules are running. The output will look similar to the following:

CodeCopy

root@vm-az220-training-gw0001-{your-id}:~# iotedge list

NAME STATUS DESCRIPTION CONFIG

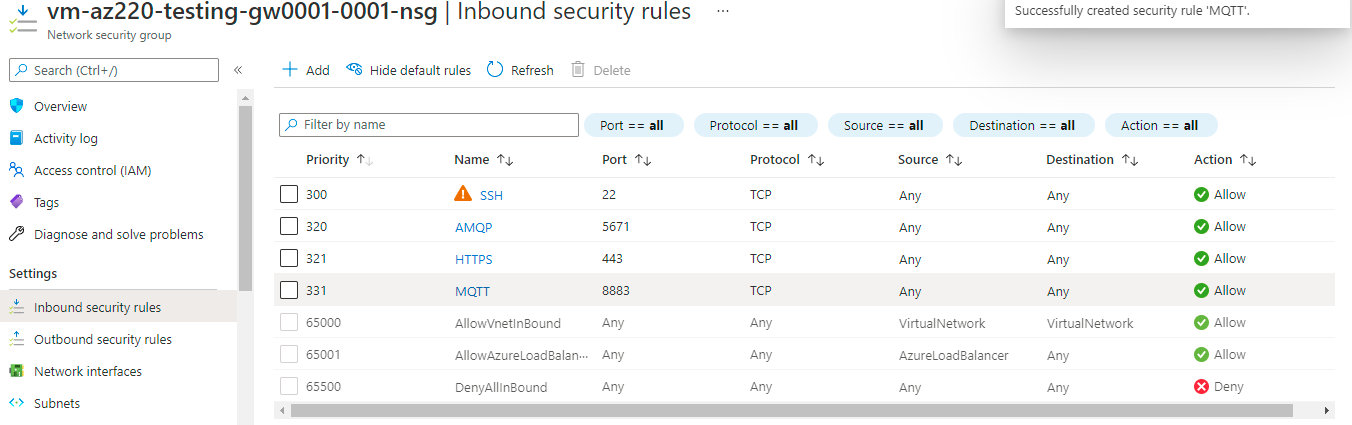
edgeHub running Up 15 seconds mcr.microsoft.com/azureiotedge-hub:1.0

edgeAgent running Up 18 seconds mcr.microsoft.com/azureiotedge-agent:1.0

If an error is reported, then you’ll need to double check that the configurations are set correctly. For troubleshooting, the **iotedge check –verbose** command can be run to see if there are any errors.

1. Close your Cloud Shell.

### **Exercise 7: Open IoT Edge Gateway Device Ports for Communication**



For the Azure IoT Edge Gateway to function, at least one of the IoT Edge hub’s supported protocols must be open for inbound traffic from downstream devices. The supported protocols are MQTT, AMQP, and HTTPS.

The IoT communication protocols supported by Azure IoT Edge have the following port mappings:

| Protocol | Port Number |
| --- | --- |
| MQTT | 8883 |
| AMQP | 5671 |
| HTTPS MQTT + WS (Websocket) AMQP + WS (Websocket) | 443 |

The IoT communication protocol chosen for your devices will need to have the corresponding port opened for the firewall that secures the IoT Edge Gateway device. In the case of this lab, an **Azure Network Security Group (NSG)** is used to **secure the IoT Edge Gateway**, so Inbound security rules for the NSG will be opened on these ports.

In a production scenario, you will want to open only the minimum number of ports for your devices to communicate. If you are using MQTT, then only open port 8883 for inbound communications. Opening additional ports will introduce addition security attack vectors that attackers could take exploit. It is a security best practice to only open the minimum number of ports necessary for your solution.

In this exercise, you will **configure** the **Network Security Group (NSG)** that secures access to the **Azure IoT Edge Gateway** from the Internet. The necessary ports for MQTT, AMQP, and HTTPS communications need to be opened so the downstream IoT device(s) can communicate with the gateway.

1. If necessary, log in to your Azure portal using your Azure account credentials.
2. On your Azure dashboard, locate the **rg-az220vm** resource group tile.

Notice that your resource group tile includes a link to the associated Network security group.

1. On the **rg-az220vm** resource group tile, click **vm-az220-training-gw0001-{your-id}-nsg**.
2. On the **Network security group** blade, on the left-side menu under **Settings**, click **Inbound security rules**.
3. At the top of the **Inbound security rules** pane, click **Add**.
4. On the **Add inbound security rule** pane, under **Destination port ranges**, change the value to **8883**
5. Under **Protocol**, click **TCP**.
6. Under **Name**, change the value to **MQTT**
7. Leave all other settings at the default, and then click **Add**.

This will define an inbound security rule that will allow communication for the MQTT protocol to the IoT Edge Gateway.

1. After the MQTT rule is added, to open ports for the **AMQP** and **HTTPS** communication protocols, add two more rules with the following values:

| Destination port ranges | Protocol | Name |
| --- | --- | --- |
| 5671 | TCP | AMQP |
| 443 | TCP | HTTPS |

1. **Note**: You may need to use the **Refresh** button in the toolbar at the top of the pane to see the new rules appear.
2. With these three ports open on the Network Security Group (NSG), the downstream devices will be able to connect to the IoT Edge Gateway using either MQTT, AMQP, or HTTPS protocols.

### **Exercise 8: Create Downstream Device Identity in IoT Hub**

In this exercise, you will create a new IoT device identity in Azure IoT Hub for the downstream IoT device. This device identity will be configured so that the Azure IoT Edge Gateway is a parent device for this downstream device.

1. If necessary, log in to your Azure portal using your Azure account credentials.
2. On your Azure dashboard, to open your IoT Hub, click **iot-az220-training-{your-id}**.
3. On the **iot-az220-training-{your-id}** blade, on the left-side menu under **Explorers**, click **IoT devices**.

This pane of the IoT Hub blade allows you to manage the IoT Devices connected to the IoT Hub.

1. At the top of the pane, to begin configuring a new IoT device, click **+ New**.
2. On the **Create a device** blade, under **Device ID**, enter **sensor-th-0072**

This is the device identity used for authentication and access control.

1. Under **Authentication type**, ensure that **Symmetric key** is selected.
2. Under **Auto-generate keys**, leave the box checked.

This will have IoT Hub automatically generate the Symmetric keys for authenticating the device.

1. Under **Parent device**, click **Set a parent device**.

You will be configuring this downstream device to communicate with IoT Hub through the IoT Edge Gateway device that you created earlier in this lab.

1. On the **Set an Edge device as a parent device** blade, under **Device ID**, click **vm-az220-testing-gw0001-0001**, and then click **OK**.
2. On the **Create a device** blade, to create the IoT Device identity for the downstream device, click **Save**.
3. On the **IoT devices** pane, at the top of the pane, click **Refresh**.
4. Under **Device ID**, click **sensor-th-0072**.

This will open the details view for this device.

1. On the IoT Device summary pane, to the right of the **Primary Connection String** field, click **Copy**.
2. Save the connection string for later reference.

Be sure to note that this connection string is for the sensor-th-0072 child device.

### **Exercise 9: Connect Downstream Device to IoT Edge Gateway**

1. **Cloud Shell**.

download lab12/certs/azure-iot-test-only.root.ca.cert.pem

1. In the EXPLORER pane, click **Program.cs**.

string trustedCACertPath = "C:\\Users\\Administrator\\Desktop\\Azure 220 OK\\TestCode\\Cert\\azure-iot-test-only.root.ca.cert.pem";

private readonly static string connectionString = " HostName=iot-az220-testing-0001.azure-devices.net;DeviceId=sensor-th-0072;SharedAccessKey=0XpOh3T7yEaKqmGRW1BzWTlbC5Hpaez6Ho+7kQMYwgc=;GatewayHostName=vm-az220-testing-gw0001-thien-0001.eastus.cloudapp.azure.com";

dotnet run

### **Exercise 10: Verify Event Flow**

az iot hub monitor-events -n iot-az220-testing-0001