# Prerequisites (5 Minutes)

# Create an IoT Hub

1. In cloudshell, run: pip3 install azure-iot-device
2. In cloudshell, run: git clone <https://github.com/microsoft/2023iotlevelup.git>
3. In cloudshell, run: cd 2023iotlevelup/MQTT

# Hands on lab with IoT Hub and MQTT (20 minutes)

1. Create a device in the IoT Hub and get the connection string to use in step 3

Text

Description automatically generated

1. In cloudshell, run: export conn\_str="*{yourconnectionstringhere}*”
2. In cloudshell, run: python3 ./lab1.py
3. In the portal, 1:) read the twin, 2:) set a twin, 3:) call a direct method method1 and 4:) send a message to device

Graphical user interface, text

Description automatically generated

1. In cloudshell, run: code ./lab1.py to read the code Control + Q to exit code
2. In cloudshell, run: code ./lab2.py to read the code for MQTT Control + Q to exit code
   1. Optional, run: python3 ./lab2.py to see the MQTT code in action

# Hands on lab with IoT Hub, MQTT and x509 certs (20 minutes)

1. In cloudshell, run: mkdir certs;cd certs
2. In cloudshell, run: openssl genrsa -out rootCA.key 4096 to create a Root key
3. In cloudshell, run: openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem to create and self-sign the Root Certificate
4. In cloudshell, run: openssl genrsa -out device1.key 2048 to generate the device certificate key
5. In cloudshell, run: openssl req -new -sha256 -key device1.key -subj "/C=US/ST=WA/O=Microsoft/CN=device1" -out device1.csr to create a certificate signing request (CSR) for your device with id “device1”
6. In cloudshell, run: openssl req -in device1.csr -noout -text to verify the CSR content
7. In cloudshell, run: openssl x509 -req -in device1.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out device1.pem -days 500 -sha256 to generate the device certificate
8. In cloudshell, run: openssl x509 -in device1.pem -text -noout to verify the cert content
9. In Azure portal upload your public Root certificate to IoT Hub and hit “Save”:

Graphical user interface, application

Description automatically generated

1. Create a device in IoT Hub called “device1”, select “X.509 CA Signed” authentication type and hit “Save”:

Graphical user interface, application, Teams

Description automatically generated

1. In cloudshell, run: export HOSTNAME="{*youriothubhostname}.azure-devices.net*"
2. In cloudshell, run: export DEVICE\_ID="*device1*"
3. In cloudshell, run: export X509\_CERT\_FILE="*device1.pem*"
4. In cloudshell, run: export X509\_KEY\_FILE="*device1.key*"
5. In cloudshell, run: python3 ../lab3\_X509.py

# Optional Labs with or without TPM

1. [Client Certificate Authentication on Windows with and without the TPM (Trusted Platform Module) – Kevin Saye (wordpress.com)](https://kevinsaye.wordpress.com/2019/08/21/client-certificate-authentication-on-windows-with-and-without-the-tpm-trusted-platform-module/)
2. [arlotito/iot-edge-1.2-tpm: How to to provision an IoT Edge 1.2 via DPS, using identity certs issued by a CA via EST and keys stored in an HSM (TPM/PKCS11) (github.com)](https://github.com/arlotito/iot-edge-1.2-tpm)
3. [arlotito/iotedge-tpm2cloud: Zero-touch provisioning of Azure IoT Edge devices with x509, TPM, PKCS#11, EST (github.com)](https://github.com/arlotito/iotedge-tpm2cloud)