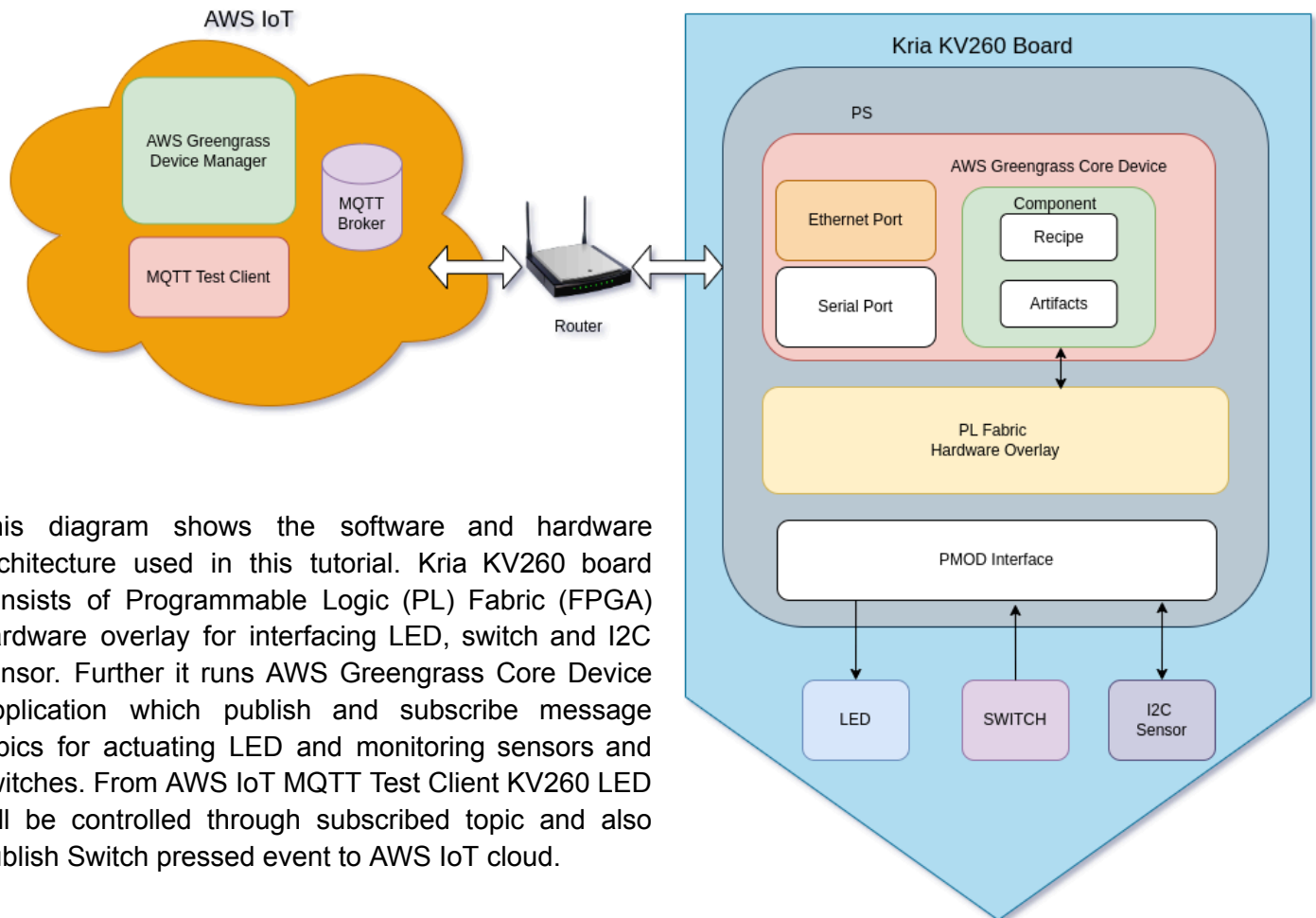# KV260 to AWS IoT Greengrass Architecture

This diagram shows the software and hardware architecture used in this tutorial. Kria KV260 board consists of Programmable Logic (PL) Fabric (FPGA) hardware overlay for interfacing LED, switch and I2C sensor. Further it runs AWS Greengrass Core Device Application which publish and subscribe message topics for actuating LED and monitoring sensors and switches. From AWS IoT MQTT Test Client KV260 LED will be controlled through subscribed topic and also publish Switch pressed event to AWS IoT cloud.
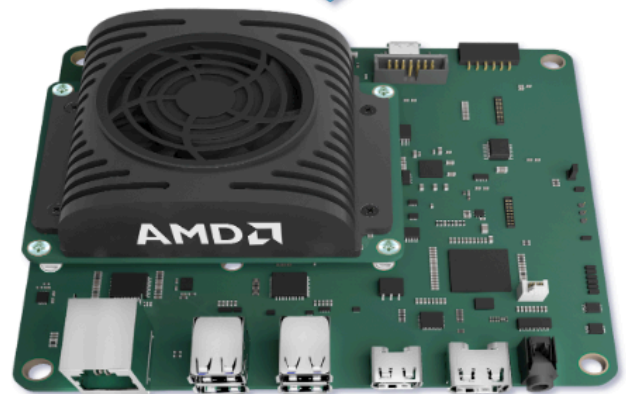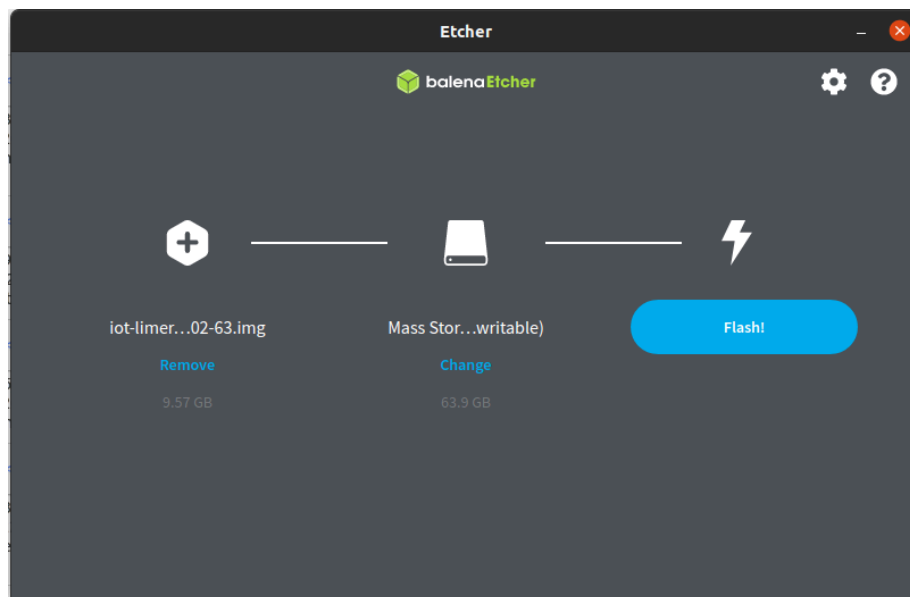
Preparing Ubuntu 22.04 OS for KRIA KV260 board

Download the Ubuntu 22.04 image from the download link
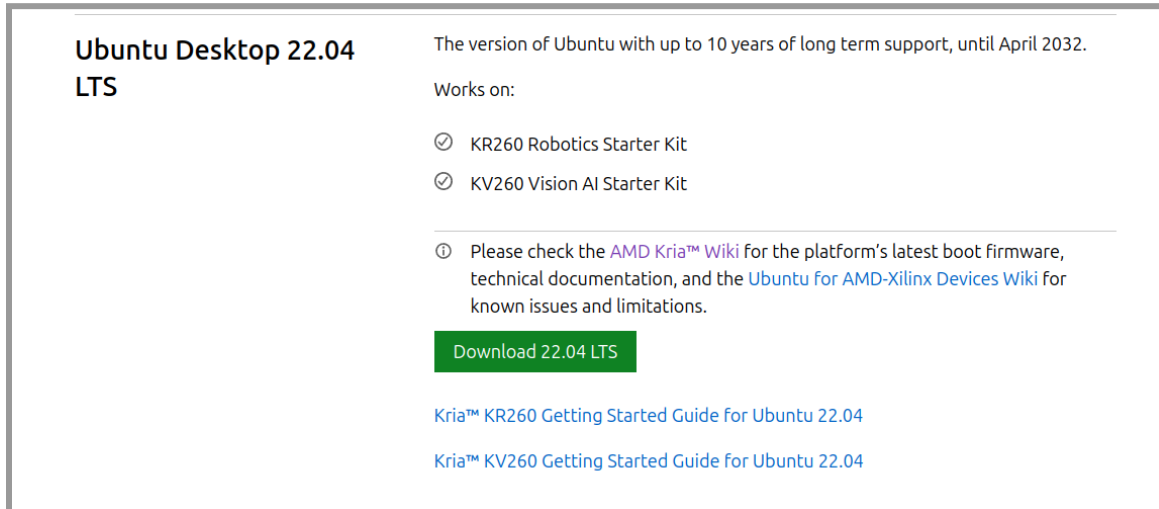
Next, prepare the SD card with the above downloaded Ubuntu image using burning tools like Balena Etcher.

Now boot the KV260 with the SD card with Ethernet and USB to Serial cable connected to board. We will be using Serial console for initial access and debugging and Ethernet network for accessing through SSH and connect KV260 to internet.

For initial login here are the Login Details:
Username : ubuntu
Password: ubuntu
This will ask to change the password. So update the password and login the system.
After successful login, one can access the KV260 device console.

Installing hardware overlay in KV260

Get the KV260 firmware folder. It contains:
- kv260-gpio-i2c.bit.bin
- kv260-gpio-i2ci2c.dtbo
- shell.json

Copy these file to the KV260 board. For firmware to be loaded using xmutil (FPGA manager), one has to copy these file at "/lib/firmware/xilinx".

For this create the folder at "kv260-gpio-i2c" at "/lib/firmware/xilinx" and copy the files in "kv260-gpio-i2c" folder.

```
cd /lib/firmware/xilinx
sudo mkdir kv260-gpio-i2c
sudo cp <kv260-firmware directory>/kv260-gpio-i2c* ./
sudo cp <kv260-firmware directory>/shell.json ./
```

Next, check the available fpga firmware using `xmutil listapps` command. `kv260-gpio-i2c` will be available in the list.

```
ubuntu@kria:~$ sudo xmutil listapps
                Accelerator          Accel_type                    Base      Base_type    #slots(PL+AIE)    Active_slot

           k26-starter-kits          XRT_FLAT        k26-starter-kits       XRT_FLAT           (0+0)              0,
             kv260-gpio-i2c          XRT_FLAT          kv260-gpio-i2c       XRT_FLAT           (0+0)             -1
ubuntu@kria:~$
```

Next load the `kv260-gpio-i2c` firmware, which contains necessary hardwares (gpio) and interfaces. In our Greengrass Demo we will be using these gpio to trigger the publishing data to AWS Greengrass IoT cloud server and also actuate GPIO on the message received from AWS cloud.

```
sudo xmutil unloadapp
sudo xmutil loadapp kv260-gpio-i2c
```

```
ubuntu@kria:~$ sudo xmutil loadapp kv260-gpio-i2c
[ 1027.134932] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
[ 1027.145049] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
[ 1027.154904] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/afi0
[ 1027.164406] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking0
[ 1027.174332] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking1
[ 1027.184267] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_0
[ 1027.194279] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_1
[ 1027.204304] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_gpio_0
[ 1027.214321] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_0
kv260-gpio-i2c: loaded to slot 0
ubuntu@kria:~$
```

Now to access GPIO in user application, we will be using `gpiod` library.

Installing gpiod packages

GPIOD packages are required to access the GPIO channels. It also provides python binding for accessing GPIO in python programming. Install the package using apt-get:

```
sudo apt update
sudo apt-get install gpiod python3-libgpiod
```

Now we can check the available gpio using gpiod applications:

Using `gpiodetect` to get availabe gpio:

```
ubuntu@kria:~$ sudo gpiodetect
gpiochip0 [firmware:zynqmp-firmware:gpio] (4 lines)
gpiochip1 [zynqmp_gpio] (174 lines)
gpiochip2 [80020000.gpio] (4 lines)
ubuntu@kria:~$ 
```

Here `gpiochip2` is the device corresponding to gpio in FPGA and it consists of 4 lines. Further these gpio lines are connected to PMOD 1 such that:
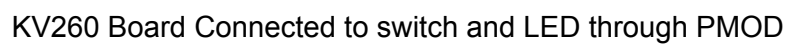PMOD1-> 5 - gpiochip2 line 0
PMOD1-> 7 - gpiochip2 line 1
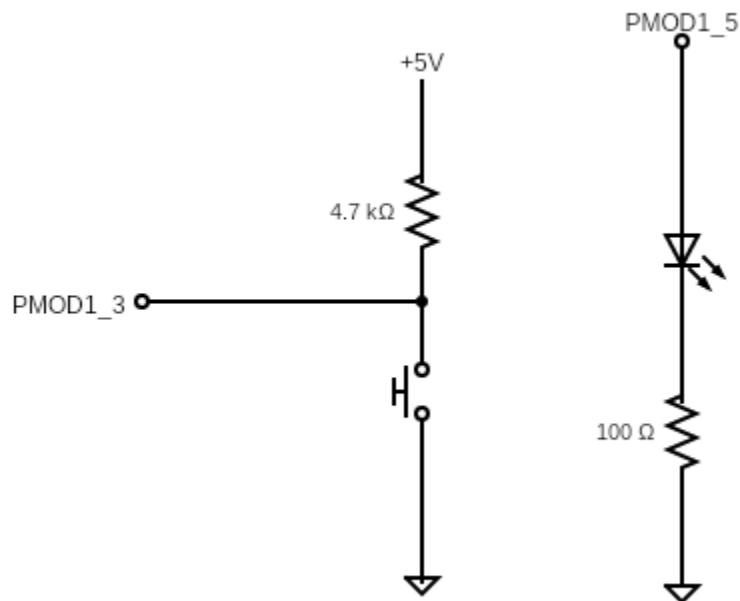
Schematic for LED and Switch Connection

11	9	7	5	3	1	PMOD UPPER
12	10	8	6	4	2	PMOD LOWER
Vcc	GND	I/O	I/O	I/O	I/O	

PMOD port numbering

KV260 Board Connected to switch and LED through PMOD
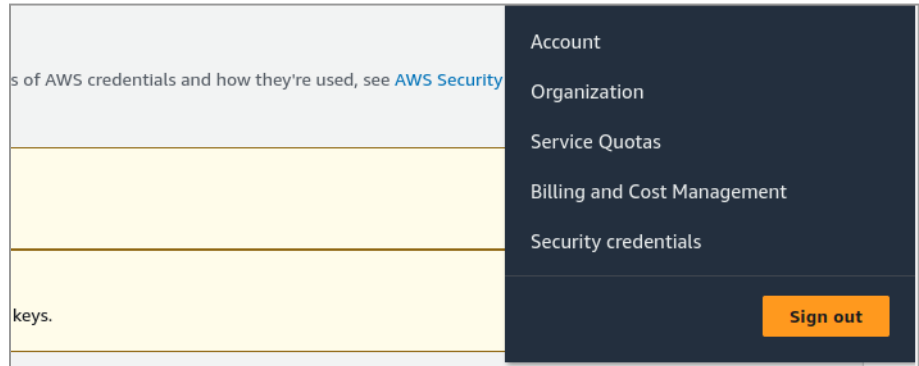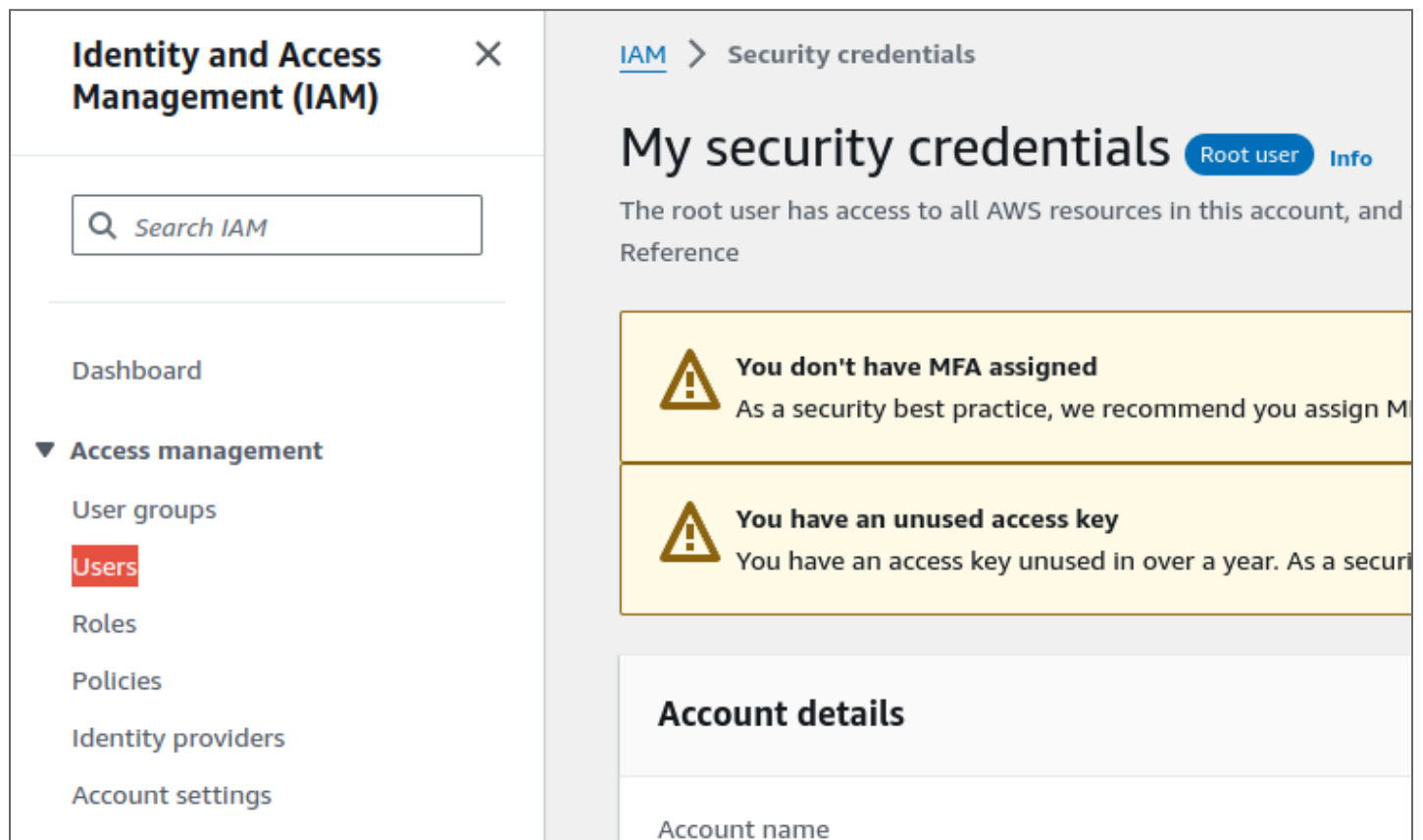
AWS IoT user creation

For and non human access to AWS services one has to create a user with required permissions.
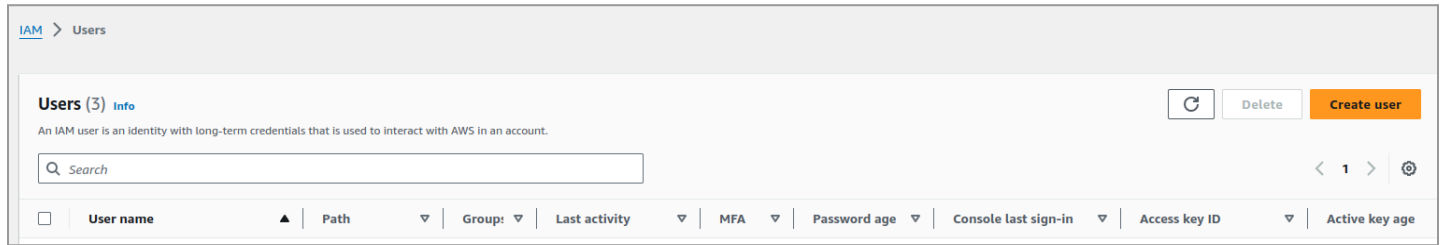- Login to AWS console
- Next go to `Security credentials` link available at root user drop down at top right corner of the AWS console

- Next Go to User management page by clicking at the User link at IAM sidebar.
This will list the available users.
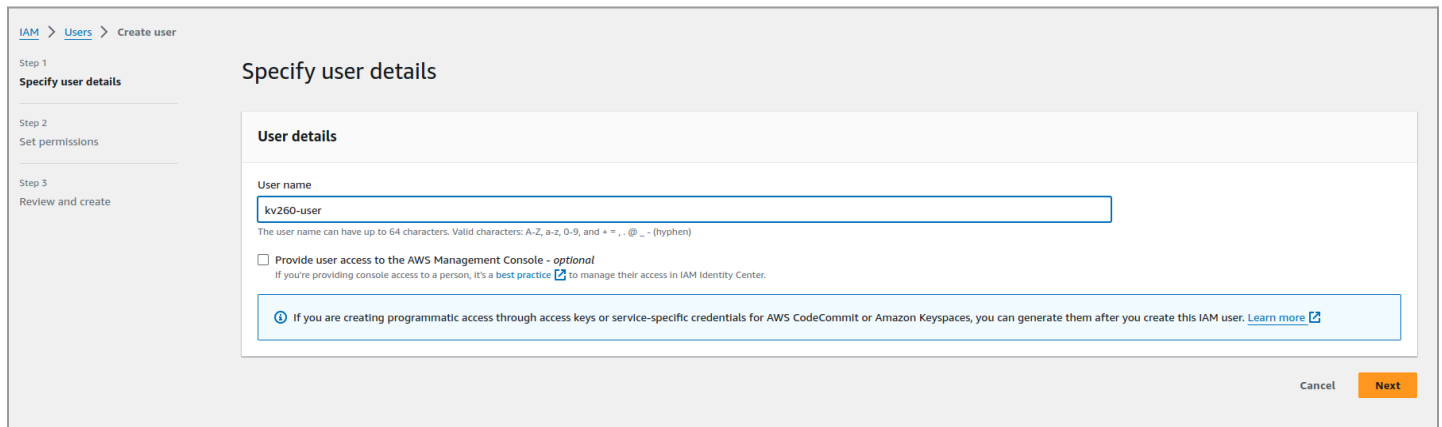
- Now create a new user for KV260 device by clicking the "Create User" button.

This will lead to step wise User creation forms. So fill the User details,
This will lead to step wise User creation forms.
So fill the User details, leave the console access unchecked as user does not have to access the AWS console through web.

Next, update the Permissions options by attaching following policies:
- AWSGreengrassFullAccess
- IAMFullAccess
- AWSIoTFullAccess
- AmazonS3FullAccess

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more [↗]

Permissions options

○ **Add user to group**
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

○ **Copy permissions**
Copy all group memberships, attached managed policies, and inline policies from an existing user.

● **Attach policies directly**
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (3/1167)

Choose one or more policies to attach to your new user.

[Create policy ↗]

	Policy name [↗]	▲	Type	▼	Attached entities	▼
☑	⊞ 📦 AWSIoTFullAccess		AWS managed		1	

Filter by Type: All types ▼   1 match   < 1 >

▶ **Set permissions boundary** - *optional*

Cancel   Previous   Next

After finishing the above steps click "Create User" to finish the user creation.

IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name	Console password type	Require password reset
kv260-user	None	No

Permissions summary

< 1 >

Name [↗]	▲	Type	▼	Used as	▼
AmazonS3FullAccess		AWS managed		Permissions policy	
AWSGreengrassFullAccess		AWS managed		Permissions policy	
AWSIoTFullAccess		AWS managed		Permissions policy	
IAMFullAccess		AWS managed		Permissions policy	

Tags - *optional*

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag]

You can add up to 50 more tags.

Cancel   Previous   Create user

Next get the access token and access key for the user. For this open the user details by clicking on the user link in the above table.

And go to "Security credentials" for creating the Access Key for the user.

Select access key for command line based access control for user.

Use case

◉ **Command Line Interface (CLI)**
You plan to use this access key to enable the AWS CLI to access your AWS account.

○ **Local code**
You plan to use this access key to enable application code in a local development environment to access your AWS account.

○ **Application running on an AWS compute service**
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

○ **Third-party service**
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

○ **Application running outside AWS**
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

○ **Other**
Your use case is not listed here.

⚠ **Alternatives recommended**
- Use AWS CloudShell, a browser-based CLI, to run commands. Learn more ↗
- Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center. Learn more ↗

Confirmation

☐ I understand the above recommendation and want to proceed to create an access key.
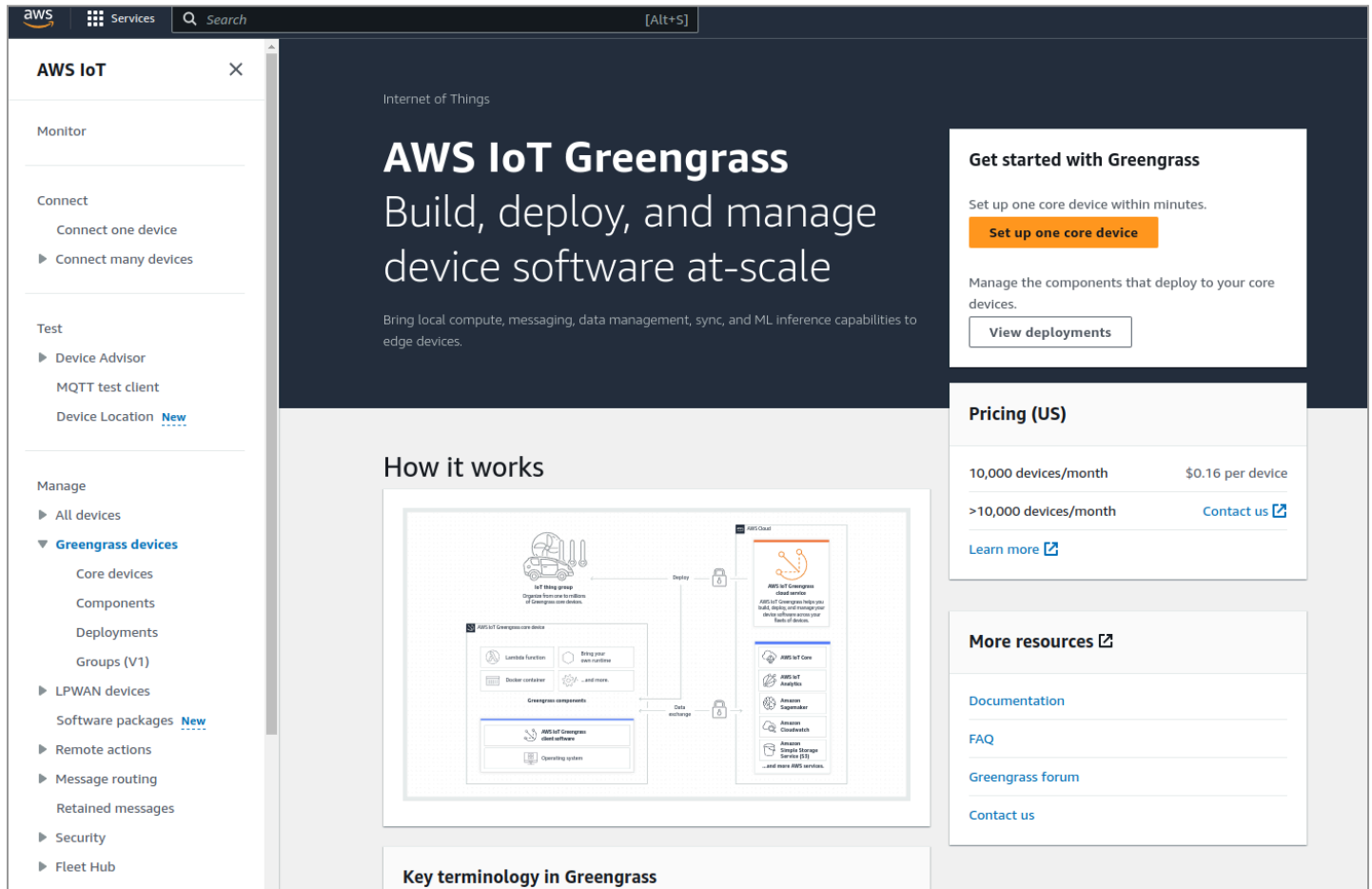
Cancel   **Next**

Next save the "Access Key" and "Secreat Access Key" . We will need this later while using greengrass CLI in KV260 console or downloading the csv file.

Installing Greengrass CLI

Steps and scripts for installing greengrass device is provided by AWS Greengrass dashboard in AWS web console. So first access the AWS Greengrass IoT page, go to AWS Services -> Internet of Things -> IoT Greengrass link
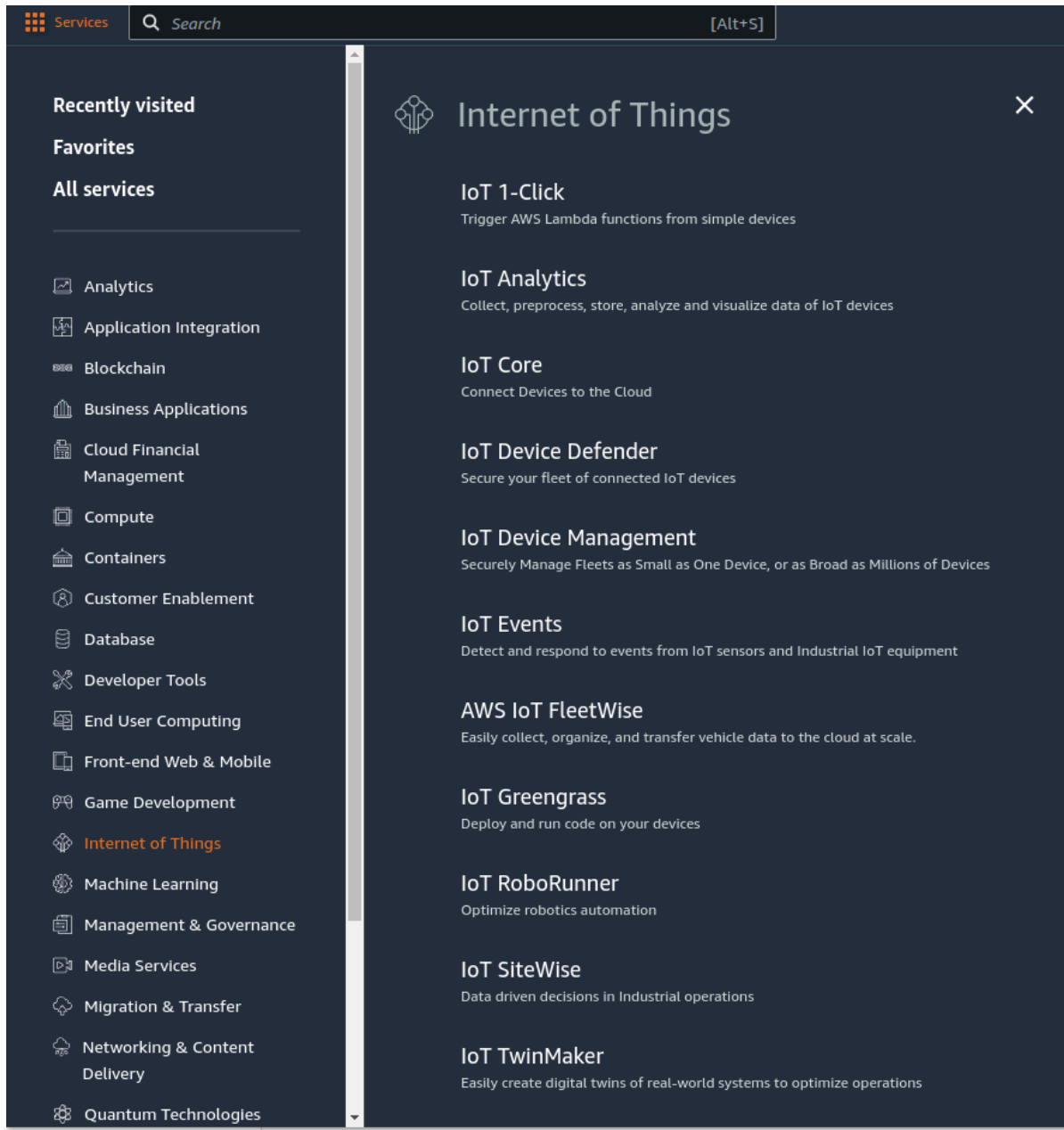
Now click on "Set up one core device" button

This will open the Greengrass core device setup page:

Here you change the Core device name like `kv260-ubuntu-dev1"

AWS IoT > Greengrass > Core devices > Set up one Greengrass core device

Set up one Greengrass core device

Step 1: Register a Greengrass core device

Greengrass core devices are AWS IoT things. Enter a thing name to be used to create a Greengrass core device.

Core device name
The name of the AWS IoT thing to create. We generated the following name for you.

    kv260-ubuntu-dev1

The name can be up to 128 characters. Valid characters: a-z, A-Z, 0-9, underscore (_), and hyphen (-).

Step 2: Add to a thing group to apply a continuous deployment

Add your Greengrass core device to an AWS IoT thing group. If the thing group has an active Greengrass deployment, your new core device receives and applies the deployment when you finish the setup process. To deploy to only the core device, select No group.

Thing group
- ● Enter a new group name
- ○ Select an existing group
- ○ No group

Thing group name
The name of the AWS IoT thing group to create.

    kv260UbuntuGroup

The name can be up to 128 characters. Valid characters: a-z, A-Z, 0-9, underscore (_), and hyphen (-).

Now in KV260 terminal console run following commands and scripts:

```
export AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
export AWS_SECRET_ACCESS_KEY=<AWS_SECRET_ACCESS_KEY>
```

Greengrass CLI depends on Java. So to install the dependency run the following:

```
sudo apt install default-jre
sudo apt install default-jdk
```

Download and install Greengrass core software as instructed in

---

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip >
greengrass-nucleus-latest.zip && unzip greengrass-nucleus-latest.zip -d
GreengrassInstaller
```

Download the installer

Run the following command on the device to download the AWS IoT Greengrass Core software.

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip > greengrass
    -nucleus-latest.zip && unzip greengrass-nucleus-latest.zip -d GreengrassInstaller
```
⧉ Copy

Next install the Greengrass core device:

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar
./GreengrassInstaller/lib/Greengrass.jar --aws-region us-east-1 --thing-name
kv260-ubuntu-dev1 --thing-group-name kv260UbuntuGroup --component-default-user
ggc_user:ggc_group --provision true --setup-system-service true --deploy-dev-tools true
```

Run the installer

The AWS IoT Greengrass Core software is a JAR file that installs the software when you run it for the first time. Run the following command on the device.

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar
    --aws-region us-east-1 --thing-name kv260-ubuntu-dev1 --thing-group-name kv260UbuntuGroup
    --component-default-user ggc_user:ggc_group --provision true --setup-system-service true --deploy
    -dev-tools true
```
⧉ Copy

Here is the console log after running above command:

```
Creating group ggc_group
ggc_group created
Added ggc_user to ggc_group
Provisioning AWS IoT resources for the device with IoT Thing Name: [kv260-ubuntu-dev1]...
Found IoT policy "GreengrassV2IoTThingPolicy", reusing it
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "kv260-ubuntu-dev1"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [kv260-ubuntu-dev1]!
Adding IoT Thing [kv260-ubuntu-dev1] into Thing Group: [kv260UbuntuGroup]...
Successfully added Thing into Thing Group: [kv260UbuntuGroup]
Setting up resources for aws.greengrass.TokenExchangeService ...
Attaching TES role policy to IoT thing...
No managed IAM policy found, looking for user defined policy...
IAM policy named "GreengrassV2TokenExchangeRoleAccess" already exists. Please attach it to the IAM role if not already
Configuring Nucleus with provisioned resource details...
Downloading Root CA from "https://www.amazontrust.com/repository/AmazonRootCA1.pem"
Created device configuration
Successfully configured Nucleus with provisioned resource details!
Creating a deployment for Greengrass first party components to the thing group
Configured Nucleus to deploy aws.greengrass.Cli component
Successfully set up Nucleus as a system service
ubuntu@kria:~$
```

Now in Greengrass set up page, one can view the Greengrass core devices and find above `kv260-ubuntu-dev` in the list.

In KV260 terminal one can get the device components by using `greengrass-cli`:

```
sudo /greengrass/v2/bin/greengrass-cli component list
```

```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: aws.greengrass.Nucleus
    Version: 2.12.1
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"100000000
t":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4w
ions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"networkProxy":{"proxy":
Component Name: UpdateSystemPolicyService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Cli
    Version: 2.12.1
    State: RUNNING
    Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: FleetStatusService
    Version: null
    State: RUNNING
    Configuration: null
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
ubuntu@kria:~$
```

We will be adding component to publish and subscribe the topic to the AWS cloud Broker.

Installing the component

Get the `components` folder and copy in the KV260 home directory.
It contains:
Artifacts
- com.example.mqtt
    - 1.0.0
        - mqtt.py (This python code published the data on button press and actuates gpio on receiving the data in subscribed topic)

Recipe
- com.example.mqtt-1.0.0.json

To install the above component run the following in the KV260 terminal:

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.mqtt=1.0.0"
```

```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.mqtt=1.0.0"
Local deployment submitted! Deployment Id: 51255b6b-795d-43f7-a84a-d0b7a3d561fd
ubuntu@kria:~$
```
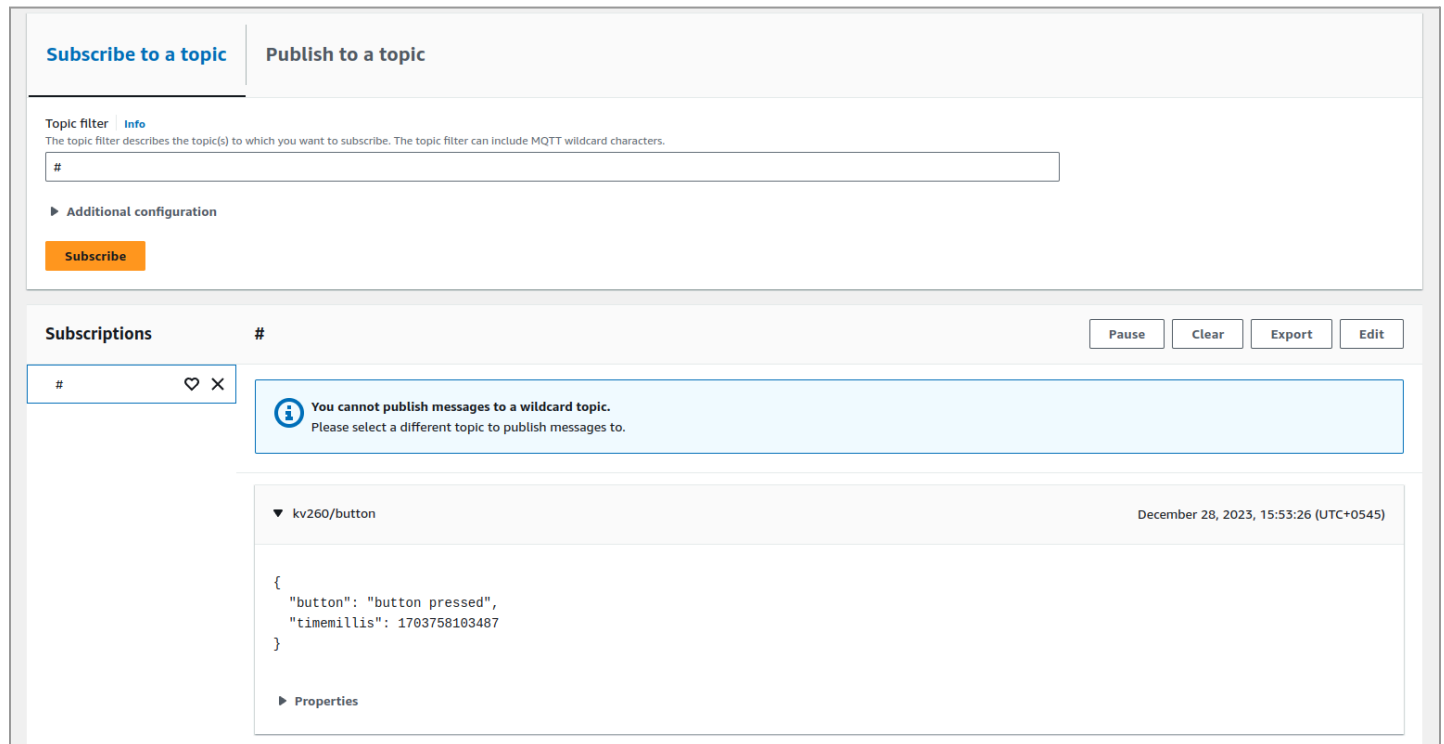
Now check the installed component is in "running state"

```
sudo /greengrass/v2/bin/greengrass-cli component list
```

```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: com.example.mqtt
    Version: 1.0.0
    State: RUNNING
    Configuration: {"accessControl":{"aws.greengrass.ipc.mqttproxy":{"com.example.mqtt:mqttproxy:1":{"operation
ces":["kv260/mqtt","kv260/button"]}}},"message":"hello"}
Component Name: aws.greengrass.Nucleus
    Version: 2.12.1
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFreque
t":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4wpvxg.credentials.iot.us-eas
ions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"networkProxy":{"proxy":{}},"platformOverride":{},"
Component Name: UpdateSystemPolicyService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Cli
    Version: 2.12.1
    State: RUNNING
    Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: FleetStatusService
    Version: null
    State: RUNNING
    Configuration: null
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
ubuntu@kria:~$
```

Now in aws IoT console, open "MQTT test client" and subscribe to "#"

Subscribe to a topic   **Publish to a topic**

Topic filter   Info
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

```
#
```

▶ **Additional configuration**

Subscribe

Subscriptions   #   Pause Clear Export Edit

♡ ✕

ⓘ **You cannot publish messages to a wildcard topic.**
   Please select a different topic to publish messages to.

▼ kv260/button   December 28, 2023, 15:53:26 (UTC+0545)

```
{
    "button": "button pressed",
    "timemillis": 1703758103487
}
```

▶ **Properties**

You can see the "button pressed" message once the button is pressed.

Now to control the LED, publish the message to "kv260/mqtt" topic. Here is the screenshot of the message which switch on the LED.

Subscribe to a topic	**Publish to a topic**

Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

🔍 kv260/mqtt ✕

Message payload
```
{
  "ledon": true
}
```

▶ Additional configuration

Publish

Subscriptions   #   Pause Clear Export Edit

♡ ✕

ⓘ **You cannot publish messages to a wildcard topic.**
Please select a different topic to publish messages to.

▼ kv260/mqtt   December 28, 2023, 15:54:38 (UTC+0545)

```
{
  "ledon": true
}
```

Now to switch off the LED send "false" message in the "kv260/mqtt" topic.

Subscribe to a topic	**Publish to a topic**

Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

🔍 kv260/mqtt ✕

Message payload
```
{
  "ledon": false
}
```

▶ Additional configuration

Publish

Subscriptions   #   Pause Clear Export Edit

♡ ✕

ⓘ **You cannot publish messages to a wildcard topic.**
Please select a different topic to publish messages to.

▼ kv260/mqtt   December 28, 2023, 15:55:08 (UTC+0545)

```
{
  "ledon": false
}
```

▶ Properties