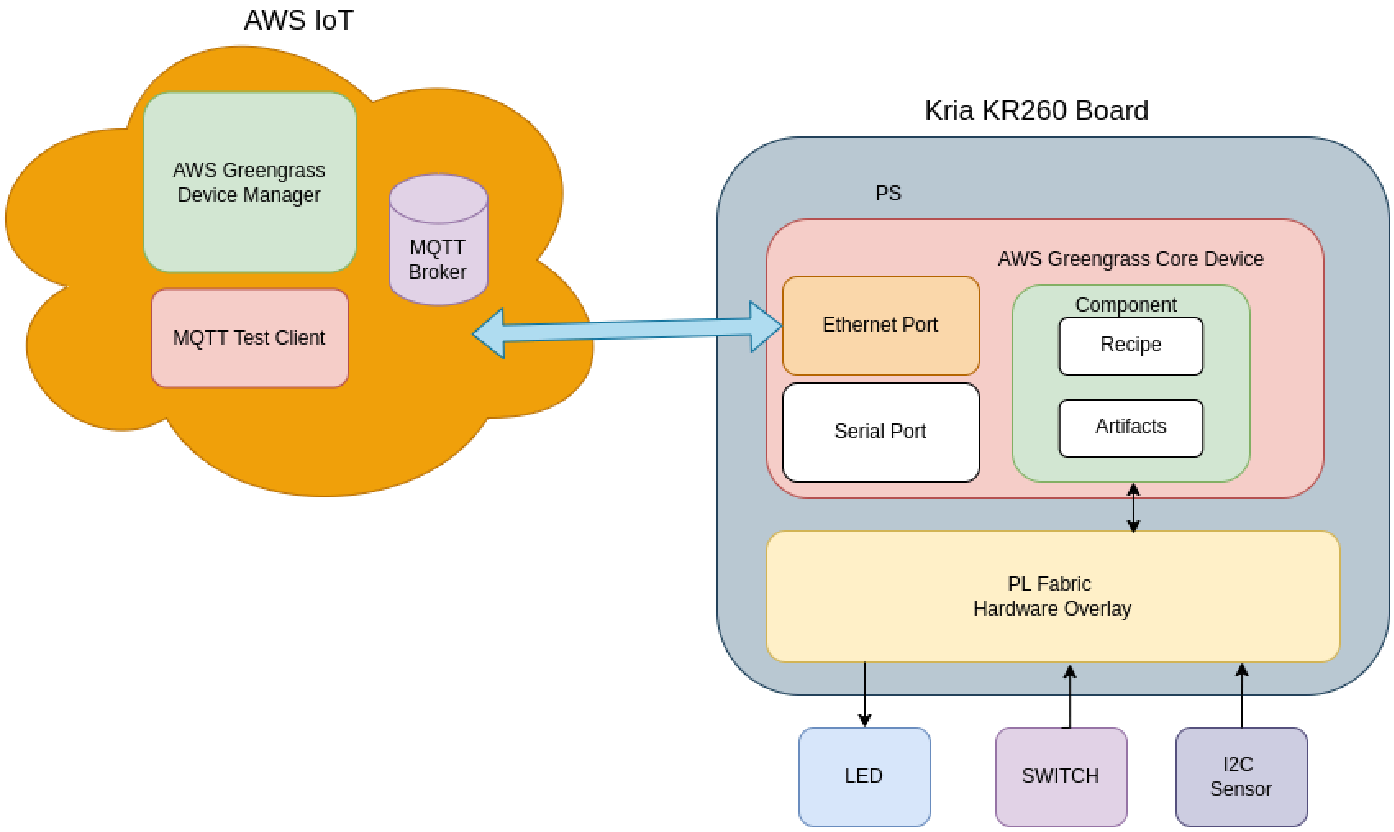
Connecting KR260 to AWS Greengrass IoT

2024.01.10/v0.2

Sanam Shakya

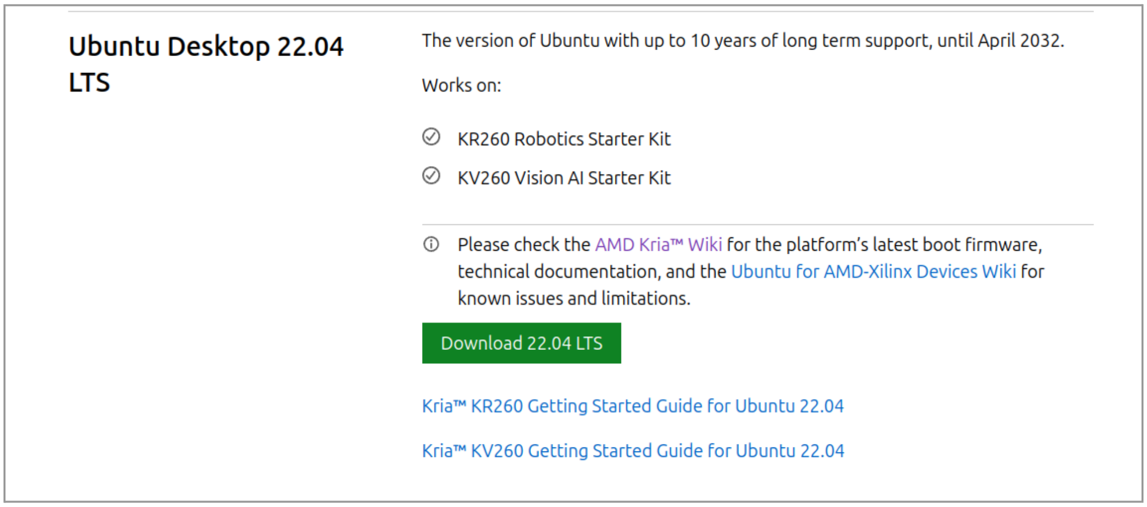
KR260 to AWS IoT Greengrass Architecture



This diagram shows the software and hardware architecture used in this tutorial. Kria KR260 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 KR260 LED will be controlled through subscribed topic and also publish Switch pressed event to AWS IoT cloud.

Preparing Ubuntu 22.04 OS for KRIA KR260 board

Download the Ubuntu 22.04 image from the download link



Ubuntu Desktop 22.04 LTS

The version of Ubuntu with up to 10 years of long term support, until April 2032.

Works on:

- KR260 Robotics Starter Kit
- KV260 Vision AI Starter Kit

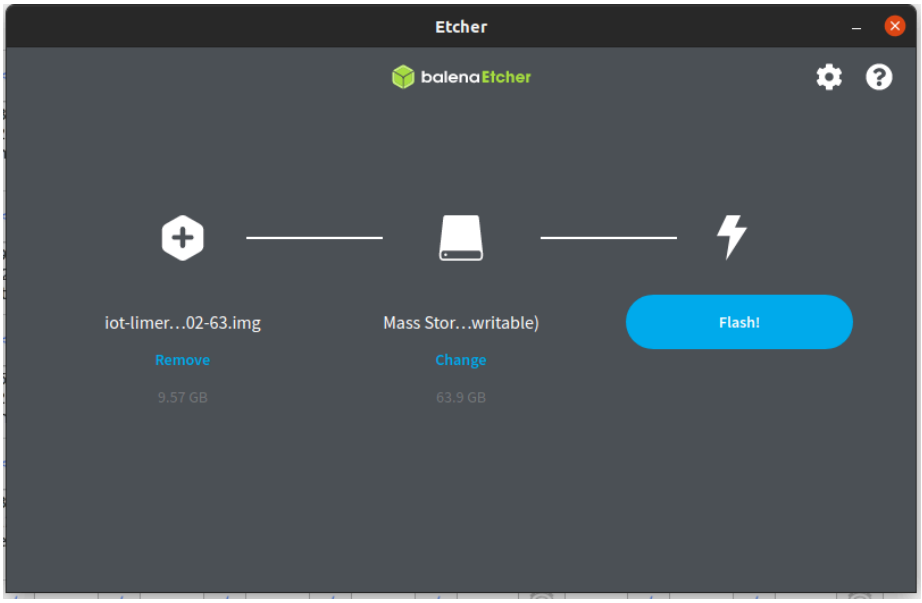
Please check the AMD Kria™ Wiki for the platform's latest boot firmware, technical documentation, and the Ubuntu for AMD-Xilinx Devices Wiki for known issues and limitations.

Download 22.04 LTS

Kria™ KR260 Getting Started Guide for Ubuntu 22.04

Kria™ KV260 Getting Started Guide for Ubuntu 22.04

Next, prepare the SD card with the above downloaded Ubuntu image using burning tools like Balena Etcher.



Now boot the KR260 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 KR260 connected to the 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 KR260 device console.

Installing hardware overlay

Get the KR260 firmware folder. It contains:

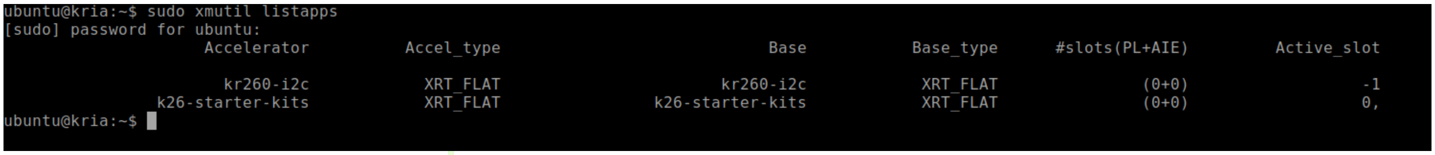
- kr260_i2c.bit.bin
- kr260_i2c.dtbo
- shell.json

Copy these file to the KR260 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 “kr260-i2c” at “/lib/firmware/xilinx” and copy the files in “kr260-i2c” folder.

```
cd /lib/firmware/xilinx
sudo mkdir kr260-i2c
sudo cp <kr260-firmware directory>/krc260_i2c* ./
sudo cp <kr260-firmware directory>/shell.json ./
```

Next, check the available fpga firmware using `xmutil listapps` command. `kr260-i2c` will be available in the list.

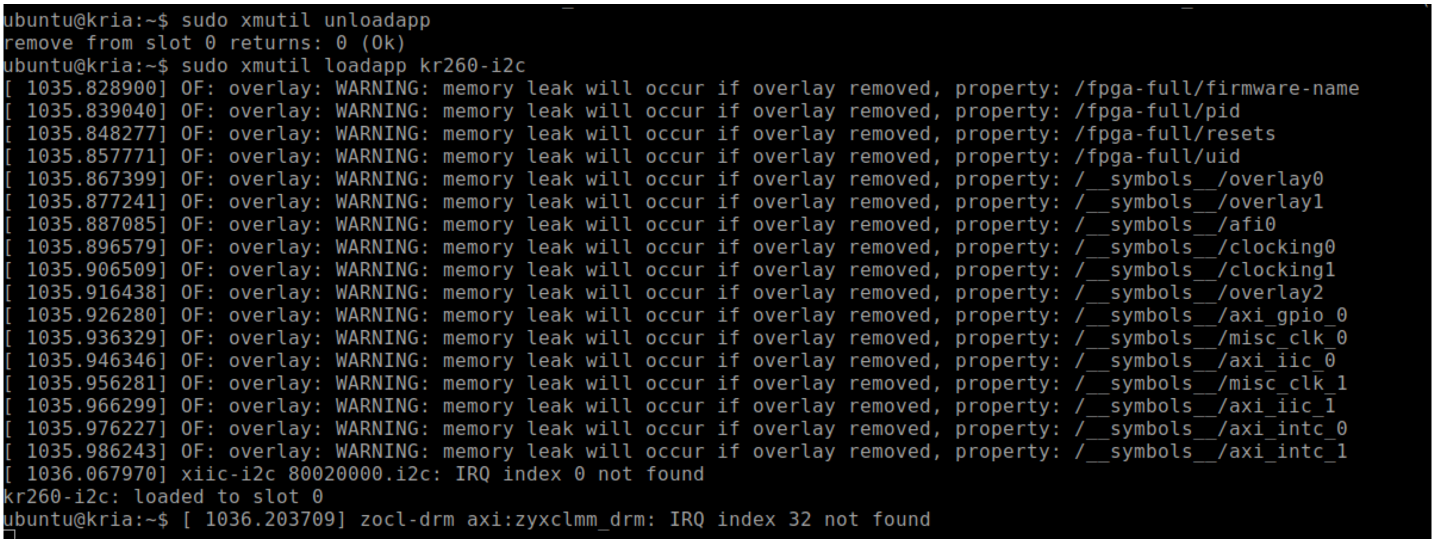


```
ubuntu@kria:~$ sudo xmutil listapps
[sudo] password for ubuntu:
                     Accelerator          Accel_type                            Base           Base_type      #slots(PL+AIE)            Active_slot

                       kr260-i2c            XRT_FLAT                       kr260-i2c            XRT_FLAT               (0+0)                     -1
                k26-starter-kits            XRT_FLAT                k26-starter-kits            XRT_FLAT               (0+0)                     0,
ubuntu@kria:~$ █
```

Next load the `kr260-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 kr260-i2c
```



```
ubuntu@kria:~$ sudo xmutil unloadapp
remove from slot 0 returns: 0 (Ok)
ubuntu@kria:~$ sudo xmutil loadapp kr260-i2c
[ 1035.828900] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
[ 1035.839040] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/pid
[ 1035.848277] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
[ 1035.857771] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/uid
[ 1035.867399] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay0
[ 1035.877241] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay1
[ 1035.887085] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/afi0
[ 1035.896579] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking0
[ 1035.906509] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking1
[ 1035.916438] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay2
[ 1035.926280] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_gpio_0
[ 1035.936329] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/misc_clk_0
[ 1035.946346] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_0
[ 1035.956281] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/misc_clk_1
[ 1035.966299] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_1
[ 1035.976227] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_0
[ 1035.986243] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_1
[ 1036.067970] xiic-i2c 80020000.i2c: IRQ index 0 not found
kr260-i2c: loaded to slot 0
ubuntu@kria:~$ [ 1036.203709] zocl-drm axi:zyxclmm_drm: IRQ index 32 not found
```

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

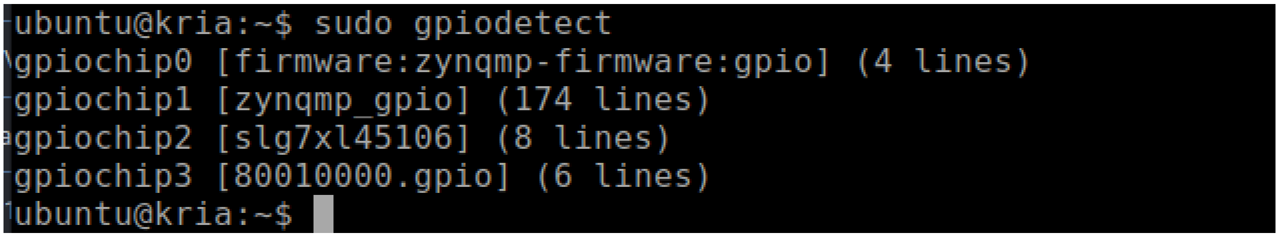
Installing gpiod packages

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-get install gpiod python3-libgpiod
```

Now we can check the available gpio using gpiod applications:

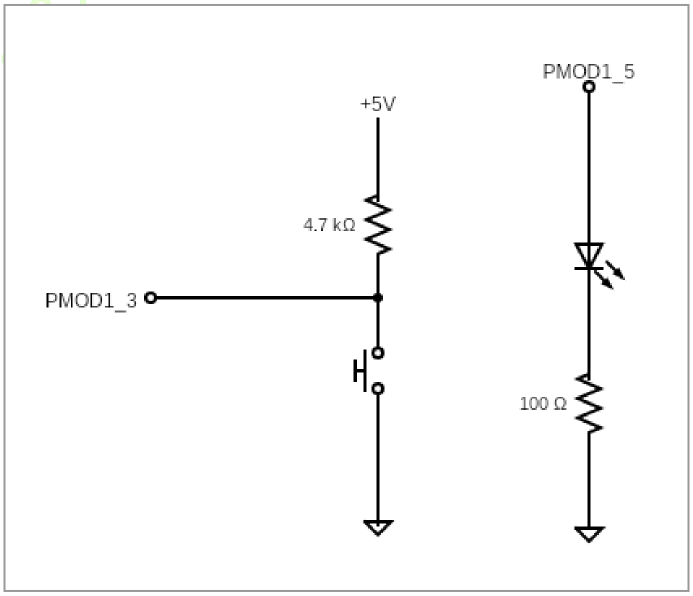
Using `gpiodetect` to get availabe gpio:



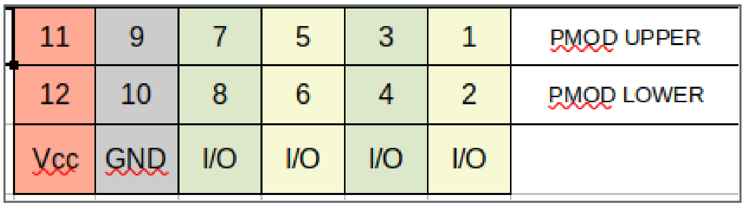
```
ubuntu@kria:~$ sudo gpiodetect
gpiochip0 [firmware:zynqmp-firmware:gpio] (4 lines)
gpiochip1 [zynqmp_gpio] (174 lines)
gpiochip2 [slg7xl45106] (8 lines)
gpiochip3 [80010000.gpio] (6 lines)
ubuntu@kria:~$
```

Here `gpiochip3` is the device corresponding to gpio in FPGA and it consists of 6 lines. Further these gpio lines are connected to PMOD 1 such that:

PMOD1-> 1 - gpiochip3 line 0
PMOD1-> 3 - gpiochip3 line 1
PMOD1-> 5 - gpiochip3 line 2

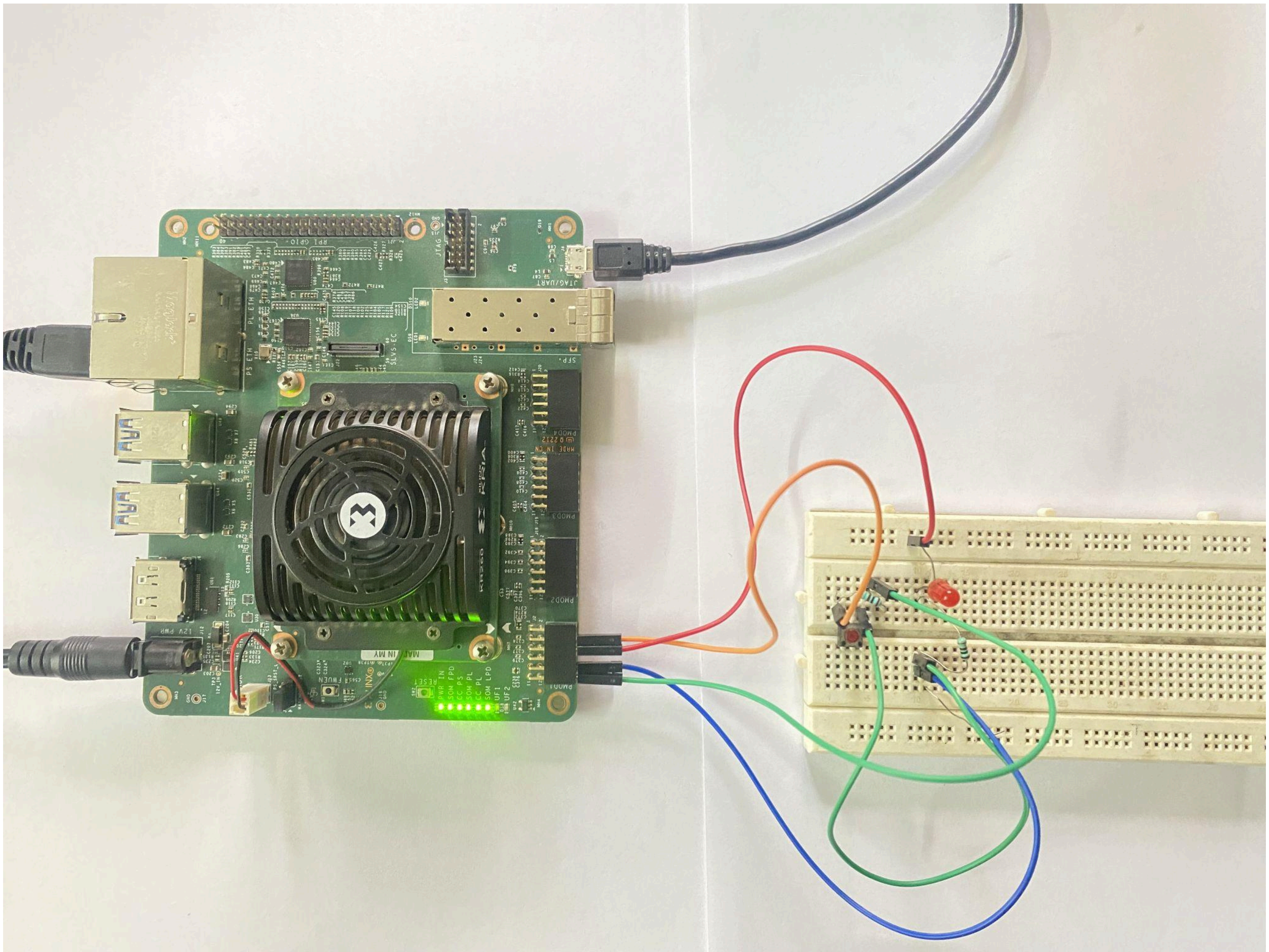


Schematic for LED and Switch Connection



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	

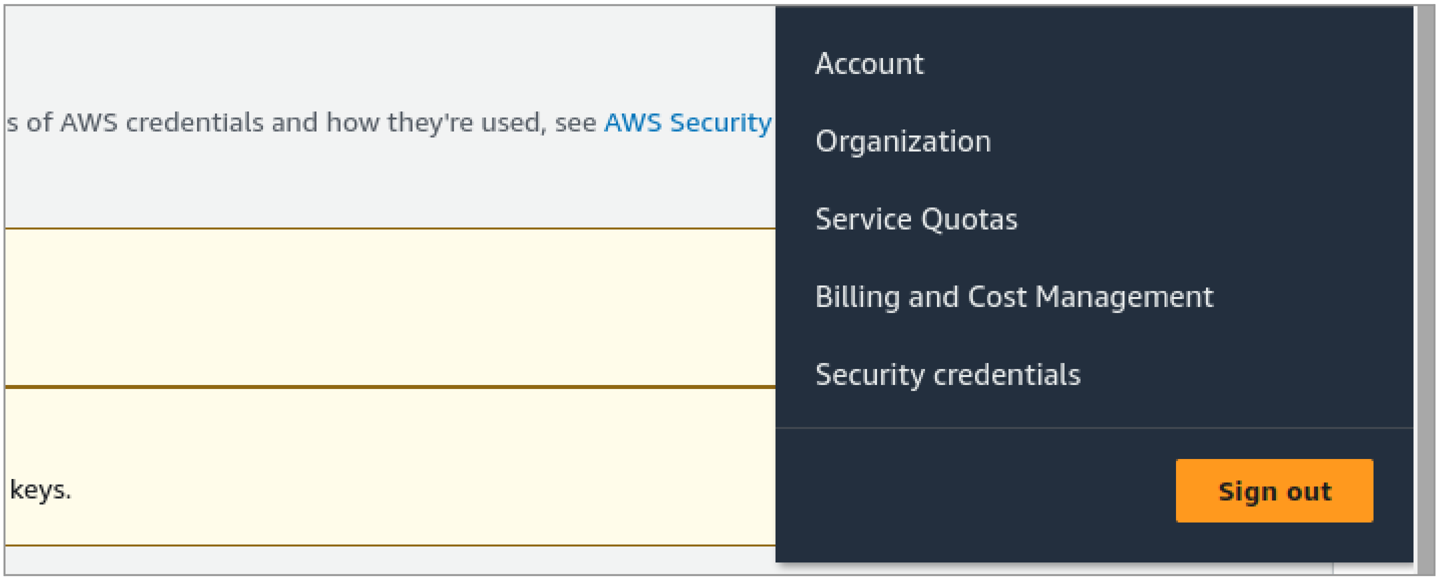
PMOD port numbering



KR260 Board Connected to switch and LED through PMOD1

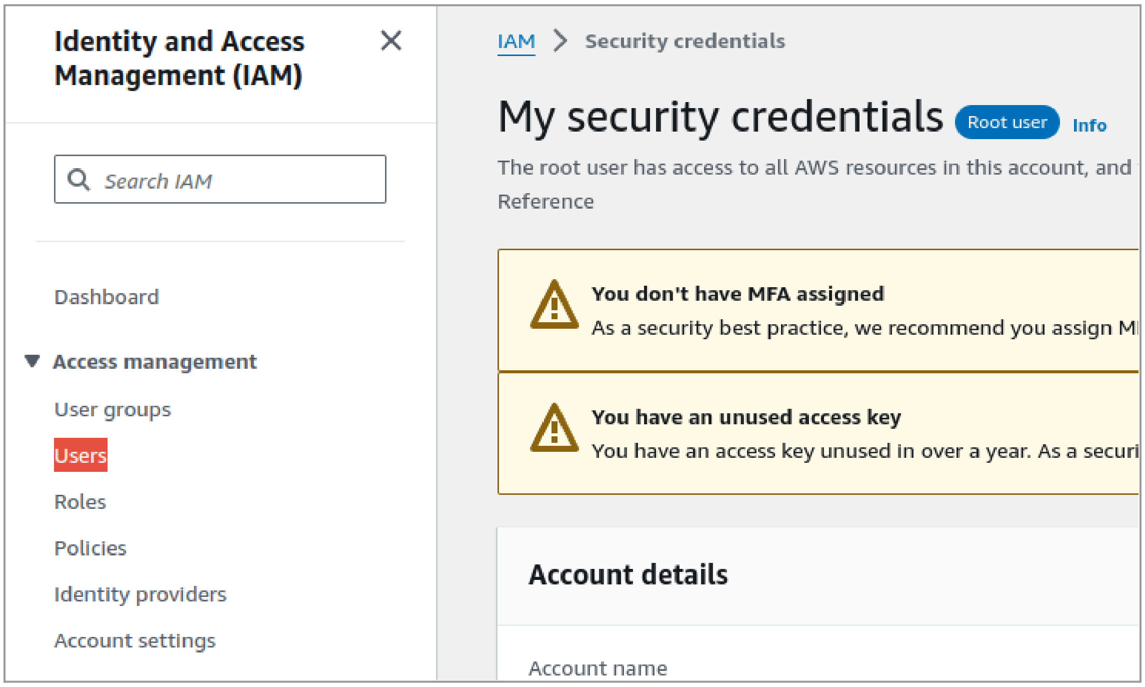
AWS IoT user creation

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


s of AWS credentials and how they're used, see AWS Security

keys.

- Next Go to User management page by clicking at the User link at IAM sidebar.
This will list the available users.



My security credentials

The root user has access to all AWS resources in this account, and
Reference

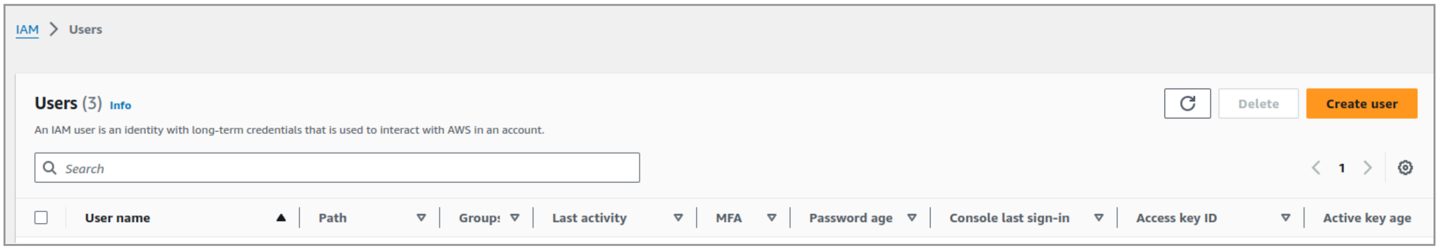
You don't have MFA assigned
As a security best practice, we recommend you assign M

You have an unused access key
You have an access key unused in over a year. As a securi

Account details

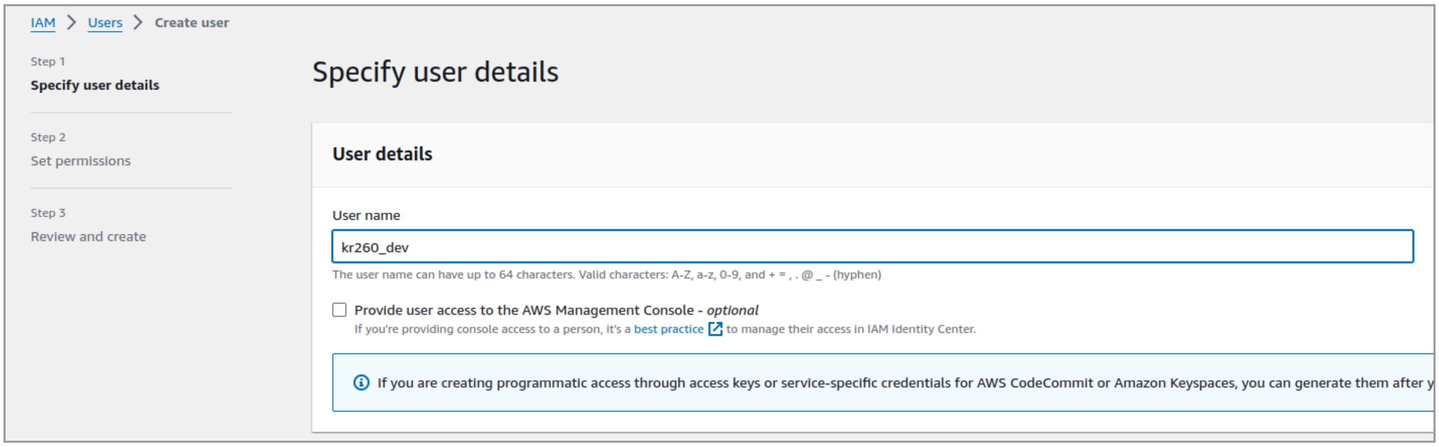
Account name

- Now create a new user for KR260 device by clicking the "Create User" button.



An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

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.



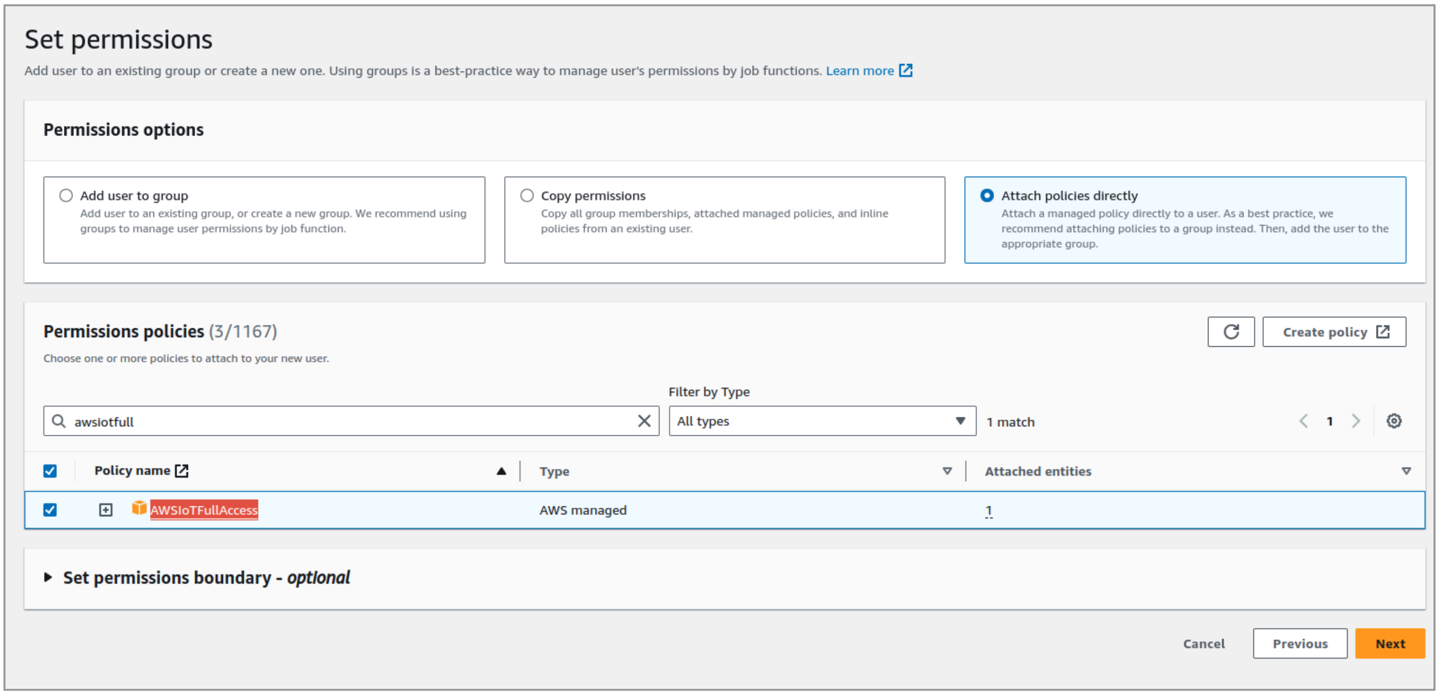
User name: kr260_dev

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

ⓘ If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after y

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)

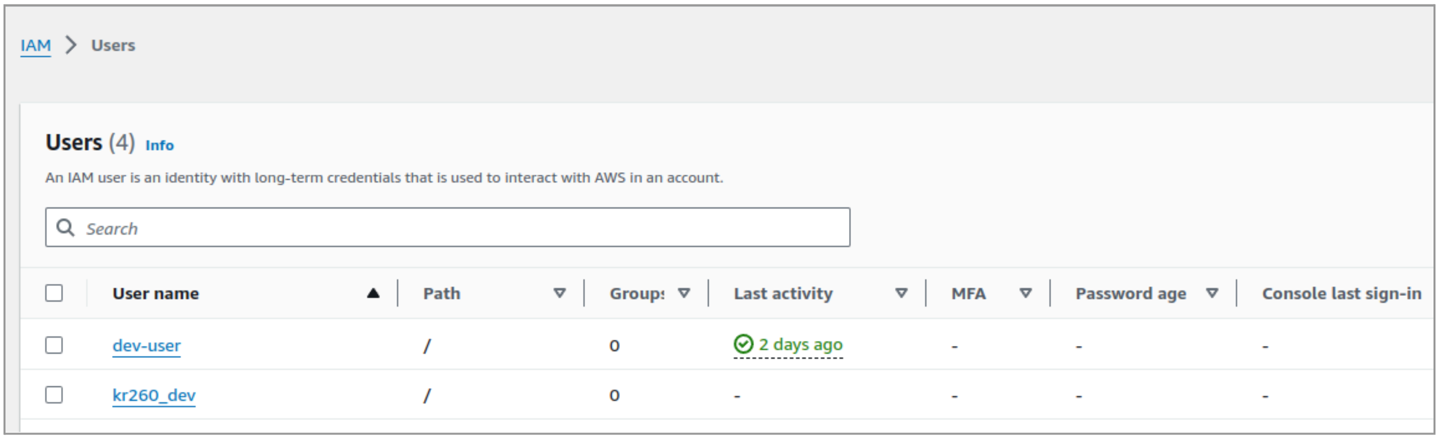
Choose one or more policies to attach to your new user.

Search: awsiotfull
Filter by Type: All types
1 match

☑	Policy name	Type	Attached entities
☑	AWSIoTFullAccess	AWS managed	1

▸ **Set permissions boundary - *optional***

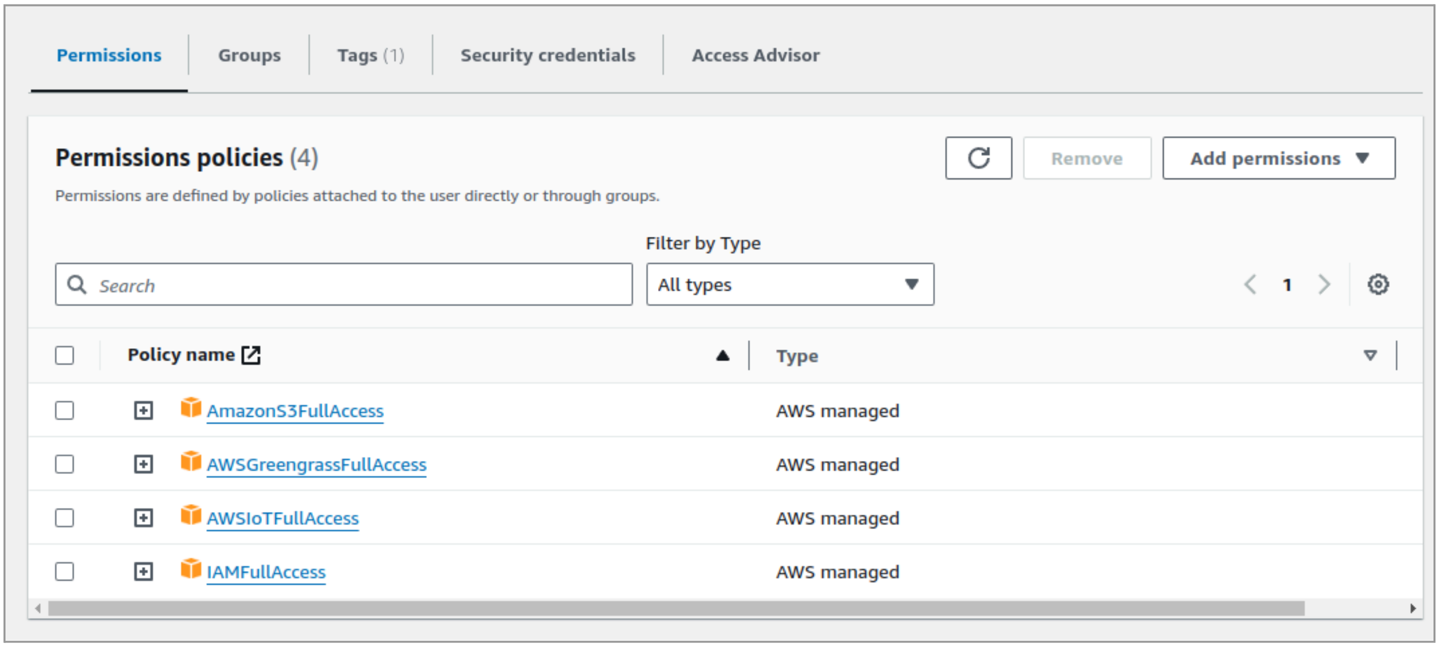
After finishing the above steps click "Create User" to finish the user creation.



Users (4) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

□	User name	Path	Group:	Last activity	MFA	Password age	Console last sign-in
□	dev-user	/	0	✓ 2 days ago	-	-	-
□	kr260_dev	/	0	-	-	-	-

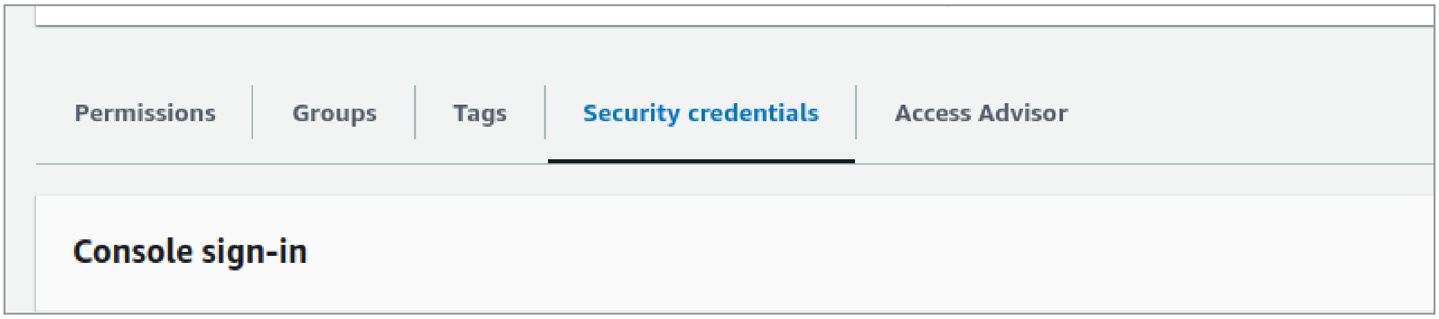


Permissions policies (4)

Permissions are defined by policies attached to the user directly or through groups.

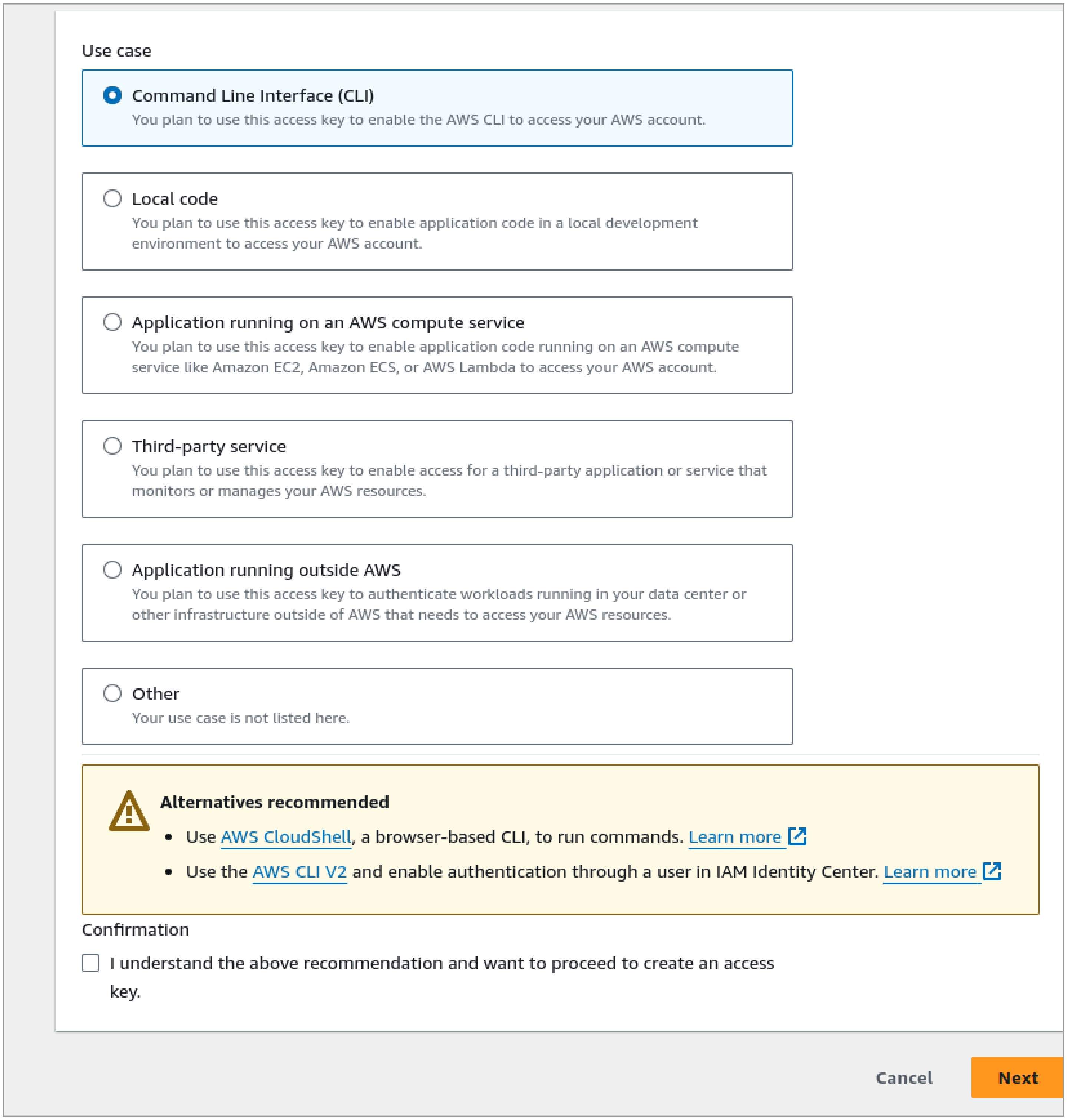
□	Policy name	Type
□	AmazonS3FullAccess	AWS managed
□	AWSGreengrassFullAccess	AWS managed
□	AWSIoTFullAccess	AWS managed
□	IAMFullAccess	AWS managed

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.



Console sign-in

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

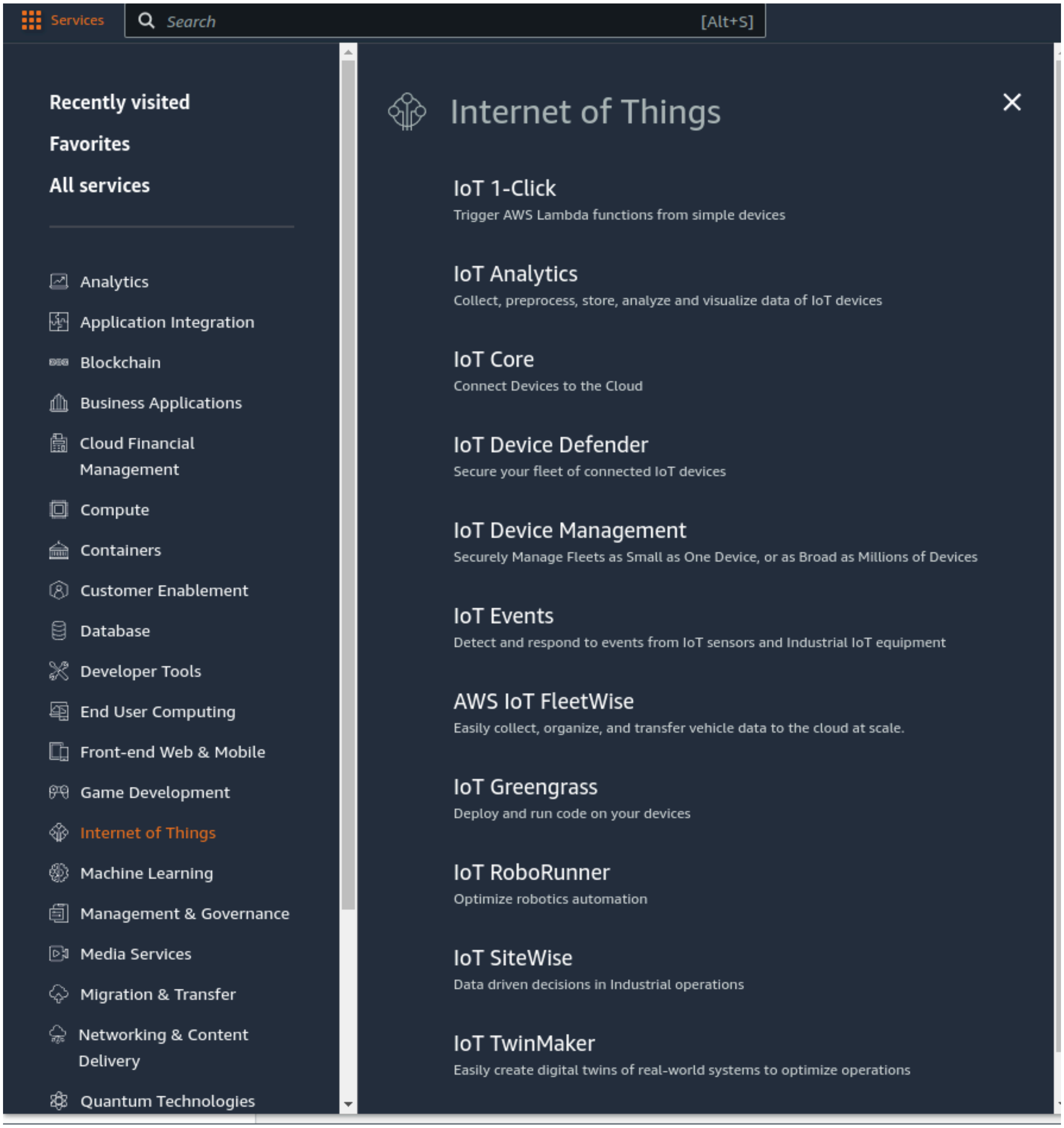
Confirmation

□ I understand the above recommendation and want to proceed to create an access key.

Next save the "Access Key" and "Secret Access Key" . We will need this later while using greengrass CLI in KR260 console or downloading the csv file.

Installing Greengrass CLI on KR260

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



Internet of Things

IoT 1-Click
Trigger AWS Lambda functions from simple devices

IoT Analytics
Collect, preprocess, store, analyze and visualize data of IoT devices

IoT Core
Connect Devices to the Cloud

IoT Device Defender
Secure your fleet of connected IoT devices

IoT Device Management
Securely Manage Fleets as Small as One Device, or as Broad as Millions of Devices

IoT Events
Detect and respond to events from IoT sensors and Industrial IoT equipment

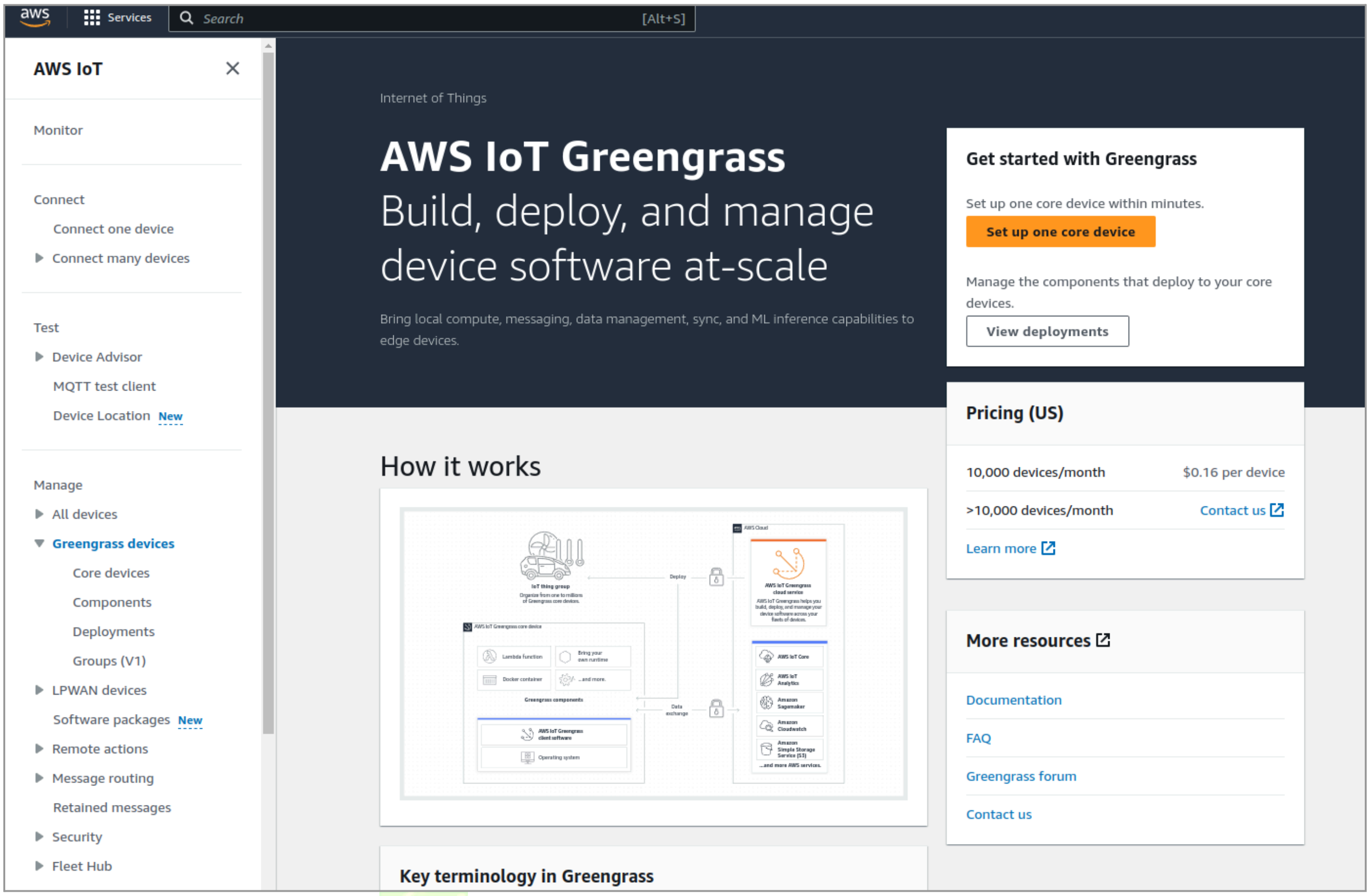
AWS IoT FleetWise
Easily collect, organize, and transfer vehicle data to the cloud at scale.

IoT Greengrass
Deploy and run code on your devices

IoT RoboRunner
Optimize robotics automation

IoT SiteWise
Data driven decisions in Industrial operations

IoT TwinMaker
Easily create digital twins of real-world systems to optimize operations



aws | Services | Search [Alt+S]

AWS IoT ✕

Monitor

Connect
- Connect one device
- ▶ Connect many devices

Test
- ▶ Device Advisor
- MQTT test client
- Device Location New

Manage
- ▶ All devices
- ▼ Greengrass devices
  - Core devices
  - Components
  - Deployments
  - Groups (V1)
- ▶ LPWAN devices
- Software packages New
- ▶ Remote actions
- ▶ Message routing
- Retained messages
- ▶ Security
- ▶ Fleet Hub

Internet of Things

AWS IoT Greengrass

Build, deploy, and manage device software at-scale

Bring local compute, messaging, data management, sync, and ML inference capabilities to edge devices.

Get started with Greengrass

Set up one core device within minutes.

[Set up one core device]

Manage the components that deploy to your core devices.

[View deployments]

Pricing (US)

10,000 devices/month	$0.16 per device
>10,000 devices/month	Contact us

Learn more

How it works

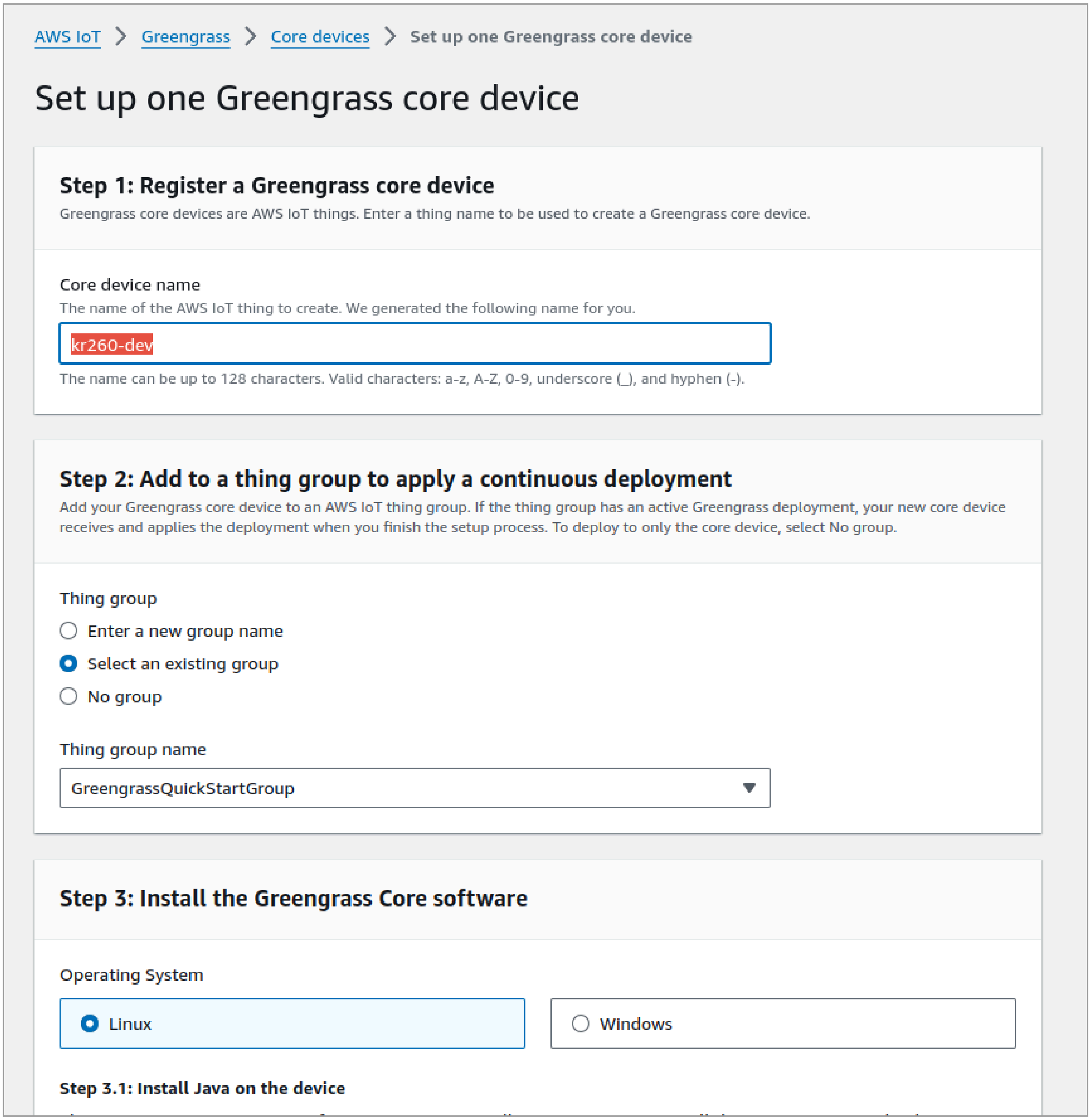
IoT thing group
Organize from one to millions of Greengrass core devices.
Deploy
Data exchange
AWS Cloud
AWS IoT Greengrass cloud service
AWS IoT Greengrass helps you build, deploy, and manage your device software across your fleets of devices.
AWS IoT Greengrass core device
Lambda function
Bring your own runtime
Docker container
...and more.
Greengrass components
AWS IoT Greengrass client software
Operating system
AWS IoT Core
AWS IoT Analytics
Amazon Sagemaker
Amazon Cloudwatch
Amazon Simple Storage Service (S3)
...and more AWS services.

More resources

- Documentation
- FAQ
- Greengrass forum
- Contact us

Key terminology in Greengrass

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 `kr260-dev"



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.

kr260-dev

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

GreengrassQuickStartGroup ▼

Step 3: Install the Greengrass Core software

Operating System

● Linux ○ Windows

Step 3.1: Install Java on the device

Now in KR260 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
```

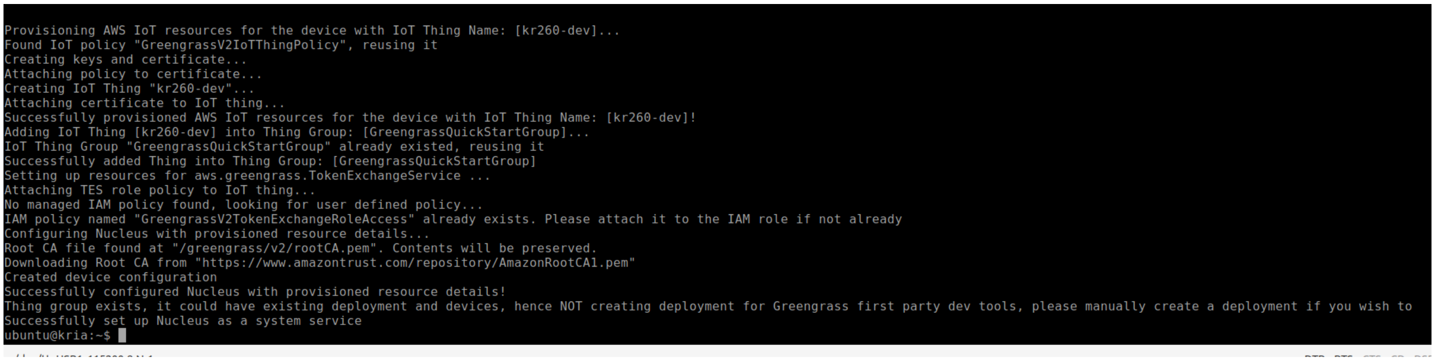
Download and install Greengrass core software.

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

Next install the Greengrass core device:

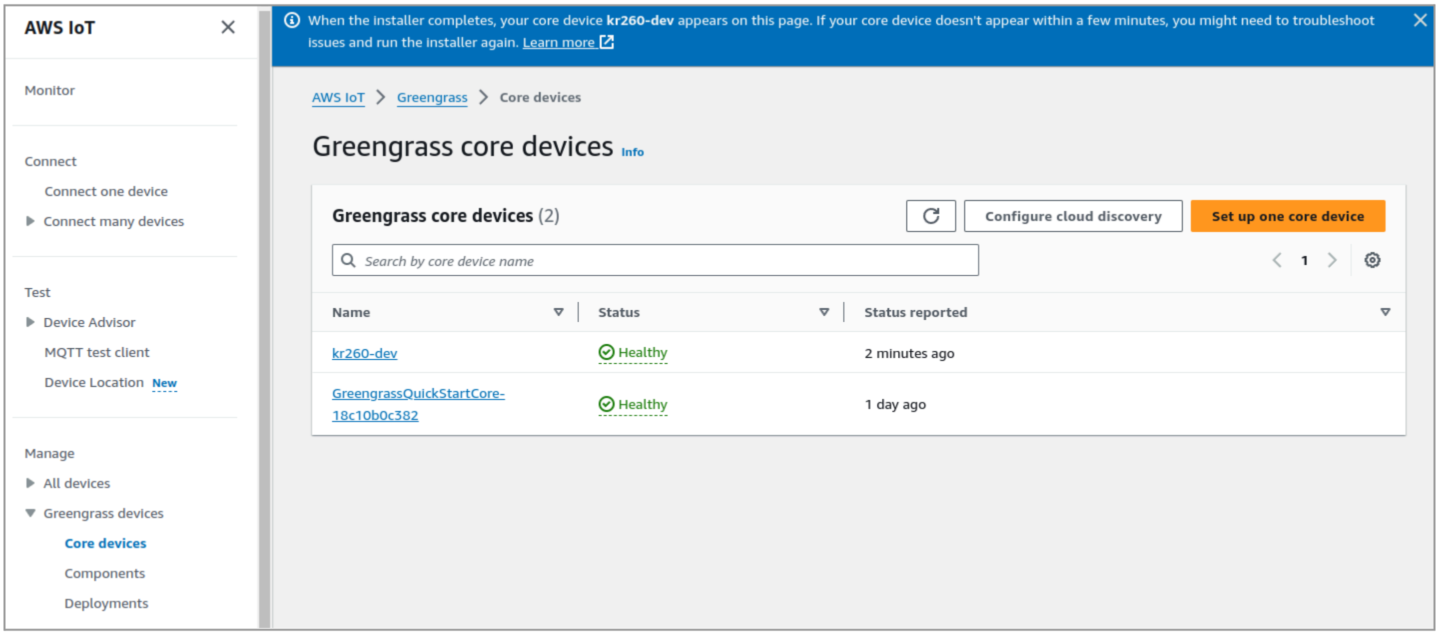
```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar --aws-region us-east-1 --thing-name kr260-dev --thing-group-name GreengrassQuickStartGroup --component-default-user ggc_user:ggc_group --provision true --setup-system-service true --deploy-dev-tools true
```

Here is the console log after running above command:



```
Provisioning AWS IoT resources for the device with IoT Thing Name: [kr260-dev]...
Found IoT policy "GreengrassV2IoTThingPolicy", reusing it
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "kr260-dev"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [kr260-dev]!
Adding IoT Thing [kr260-dev] into Thing Group: [GreengrassQuickStartGroup]...
IoT Thing Group "GreengrassQuickStartGroup" already existed, reusing it
Successfully added Thing into Thing Group: [GreengrassQuickStartGroup]
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...
Root CA file found at "/greengrass/v2/rootCA.pem". Contents will be preserved.
Downloading Root CA from "https://www.amazontrust.com/repository/AmazonRootCA1.pem"
Created device configuration
Successfully configured Nucleus with provisioned resource details!
Thing group exists, it could have existing deployment and devices, hence NOT creating deployment for Greengrass first party dev tools, please manually create a deployment if you wish to
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 `kr260-dev` in the list.



When the installer completes, your core device **kr260-dev** appears on this page. If your core device doesn't appear within a few minutes, you might need to troubleshoot issues and run the installer again. Learn more

AWS IoT > Greengrass > Core devices

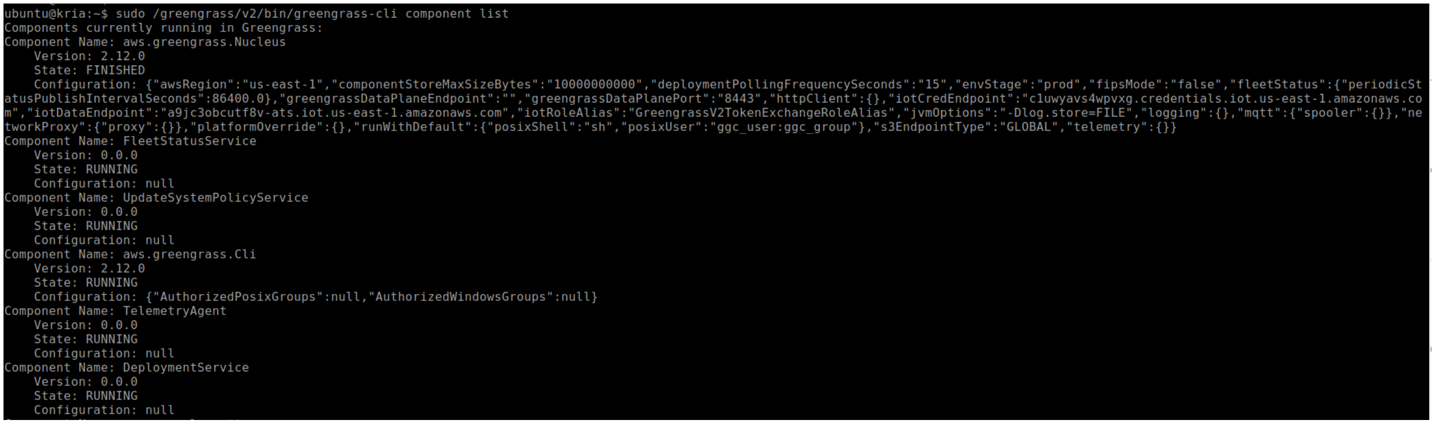
Greengrass core devices Info

Greengrass core devices (2)

Name	Status	Status reported
kr260-dev	Healthy	2 minutes ago
GreengrassQuickStartCore-18c10b0c382	Healthy	1 day ago

In KR260 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: 2.12.0
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequencySeconds":"15","envStage":"prod","fipsMode":"false","fleetStatus":{"periodicSt
atusPublishIntervalSeconds":86400.0},"greengrassDataPlaneEndpoint":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.co
m","iotDataEndpoint":"a9jc3obcutf8v-ats.iot.us-east-1.amazonaws.com","iotRoleAlias":"GreengrassV2TokenExchangeRoleAlias","jvmOptions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"ne
tworkProxy":{"proxy":{}},"platformOverride":{},"runWithDefault":{"posixShell":"sh","posixUser":"ggc_user:ggc_group"},"s3EndpointType":"GLOBAL","telemetry":{}}
Component Name: FleetStatusService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: UpdateSystemPolicyService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Cli
    Version: 2.12.0
    State: RUNNING
    Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
```

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 KR260 home directory.
It contains:
artifacts

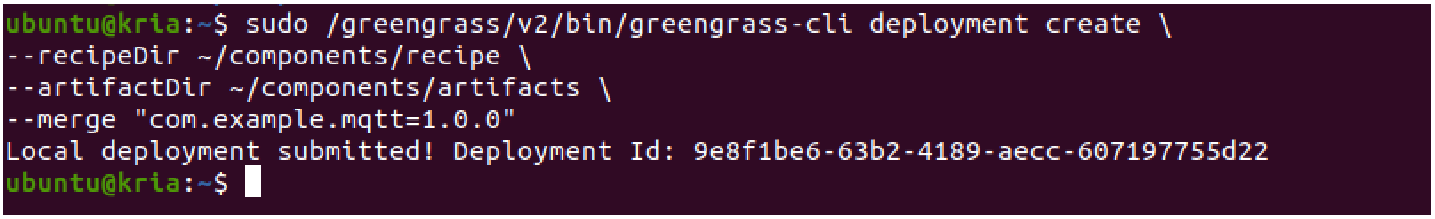
- com.example.mqtt
  - 1.0.0
    - mqtt.py (This python code published the data on button press and actuates gpio on receiving the data in subscribed topic)

recipe

- com.example.mqtt-1.0.0.json

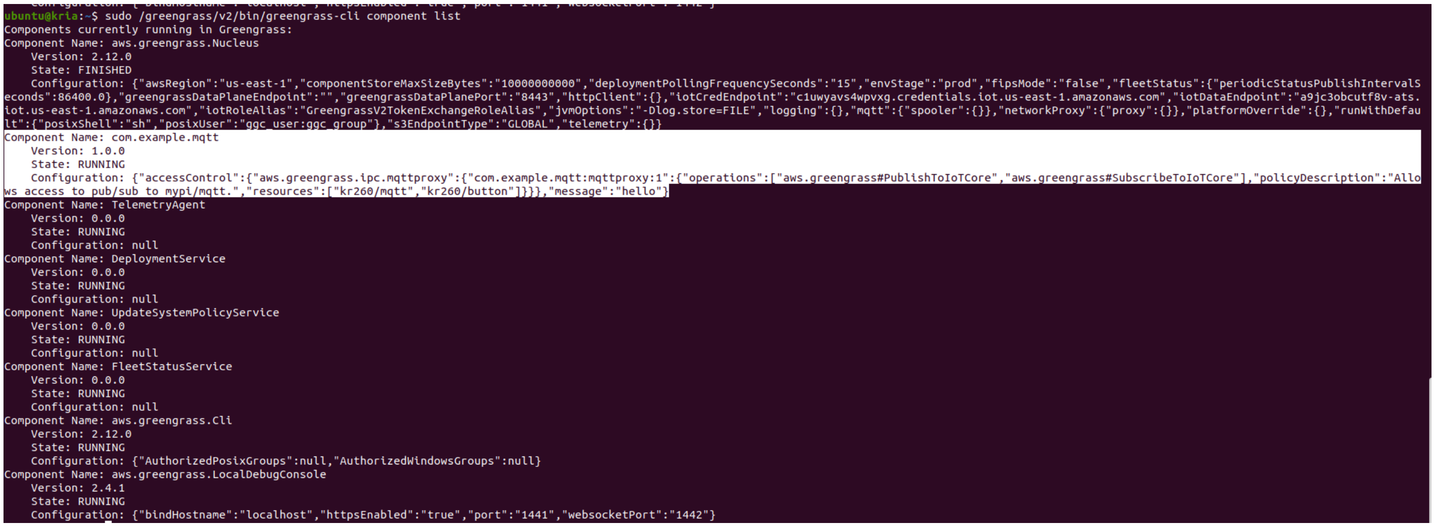
To install the above component run the following in the KR260 terminal:

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



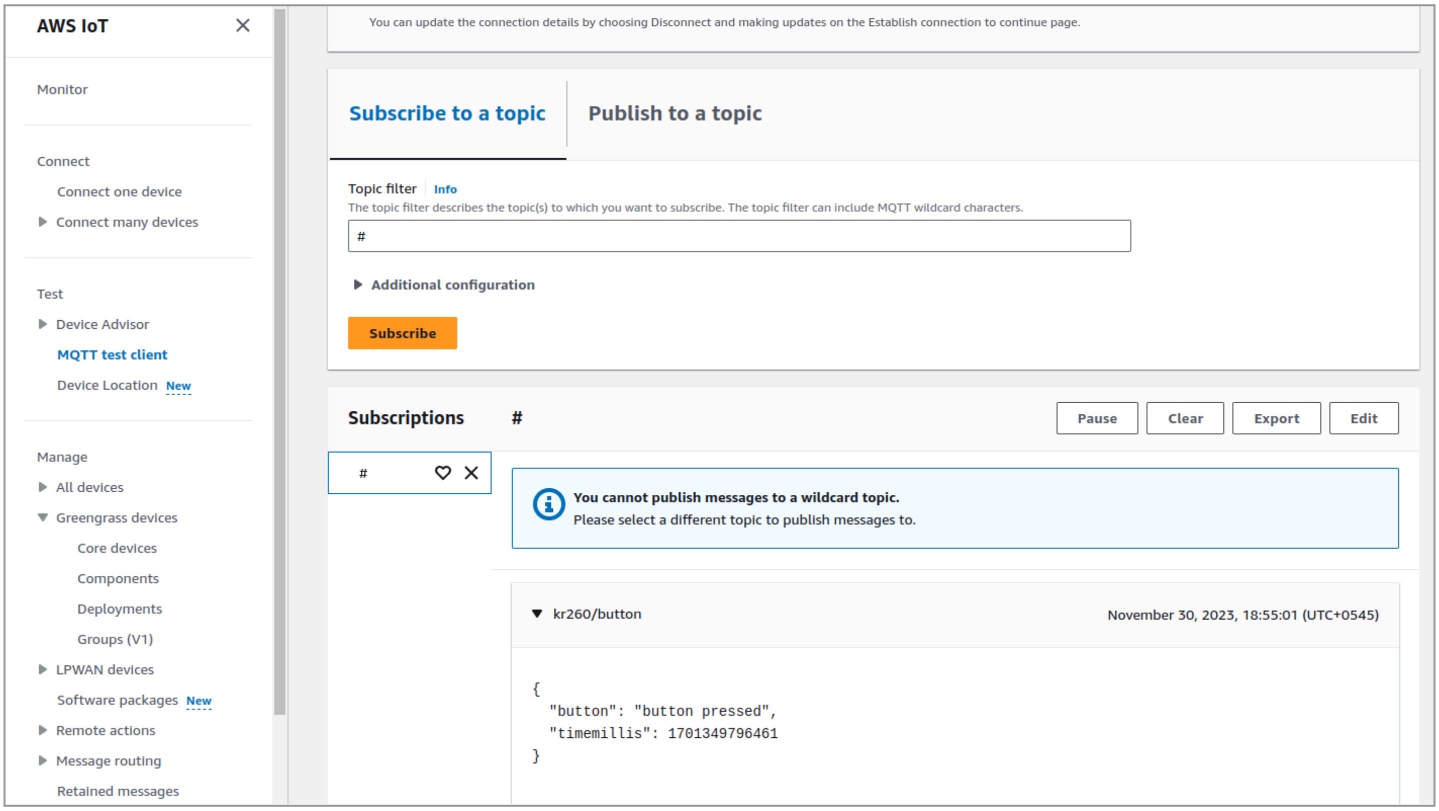
```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.mqtt=1.0.0"
Local deployment submitted! Deployment Id: 9e8f1be6-63b2-4189-aecc-607197755d22
ubuntu@kria:~$ 
```

Now check the installed component is in “running state”



```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: aws.greengrass.Nucleus
    Version: 2.12.0
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequencySeconds":"15","envStage":"prod","fipsMode":"false","fleetStatus":{"periodicStatusPublishIntervalS
econds":86400.0},"greengrassDataPlaneEndpoint":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.com","iotDataEndpoint":"a9jc3obcutf8v-ats.
iot.us-east-1.amazonaws.com","iotRoleAlias":"GreengrassV2TokenExchangeRoleAlias","jvmOptions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"networkProxy":{"proxy":{}},"platformOverride":{},"runWithDefau
lt":{"posixShell":"sh","posixUser":"ggc_user:ggc_group"},"s3EndpointType":"GLOBAL","telemetry":{}}
Component Name: com.example.mqtt
    Version: 1.0.0
    State: RUNNING
    Configuration: {"accessControl":{"aws.greengrass.ipc.mqttproxy":{"com.example.mqtt:mqttproxy:1":{"operations":["aws.greengrass#PublishToIoTCore","aws.greengrass#SubscribeToIoTCore"],"policyDescription":"Allo
ws access to pub/sub to mypi/mqtt.","resources":["kr260/mqtt","kr260/button"]}}},"message":"hello"}
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: UpdateSystemPolicyService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: FleetStatusService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Cli
    Version: 2.12.0
    State: RUNNING
    Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: aws.greengrass.LocalDebugConsole
    Version: 2.4.1
    State: RUNNING
    Configuration: {"bindHostname":"localhost","httpsEnabled":"true","port":"1441","websocketPort":"1442"}
```

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



You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

Topic filter Info

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

#

Subscriptions #

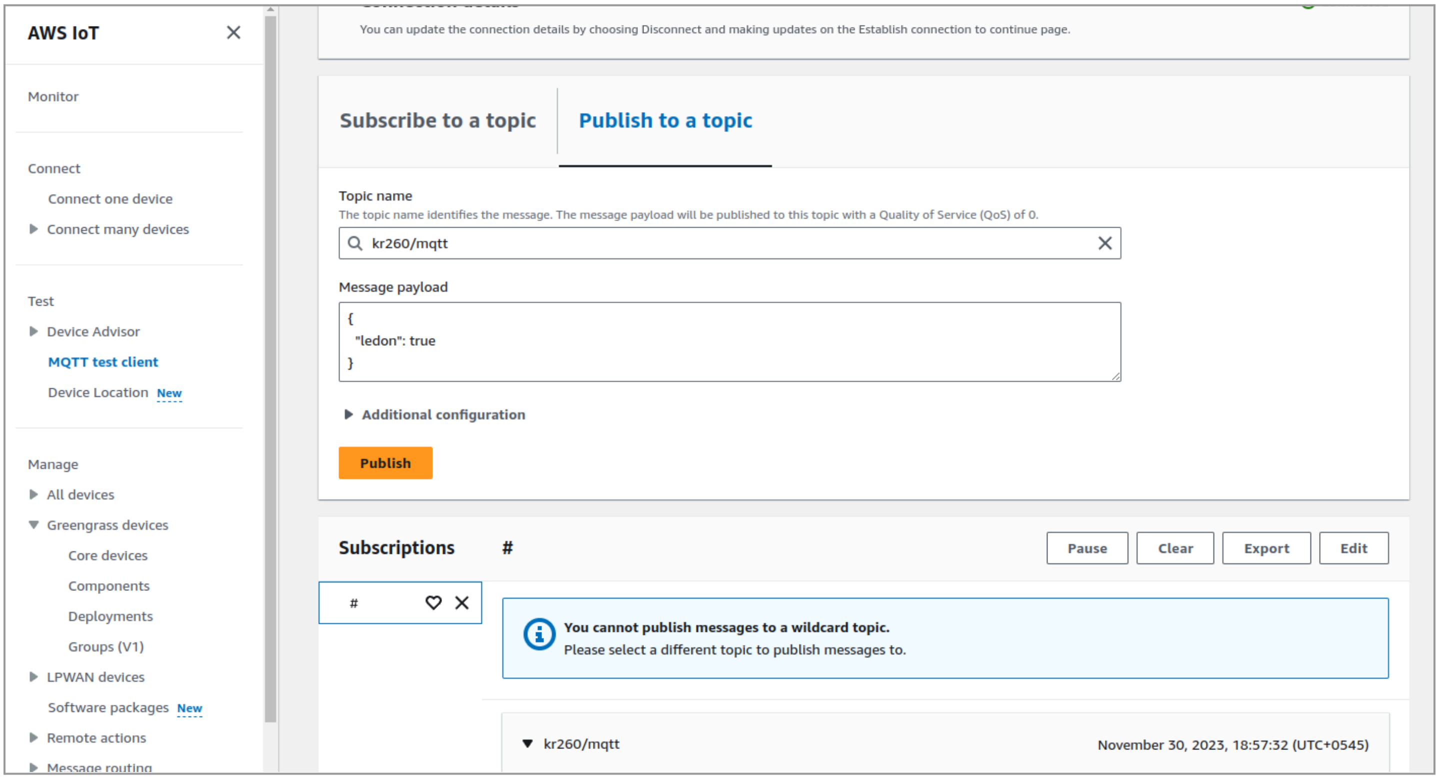
You cannot publish messages to a wildcard topic.
Please select a different topic to publish messages to.

kr260/button November 30, 2023, 18:55:01 (UTC+0545)

```
{
  "button": "button pressed",
  "timemillis": 1701349796461
}
```

You can see the “button pressed” message once the button is pressed.

Now to control the LED, publish the message to “kr260/mqtt” topic. Here is the screenshot of the message which switch on the LED.



You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

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.
kr260/mqtt

Message payload

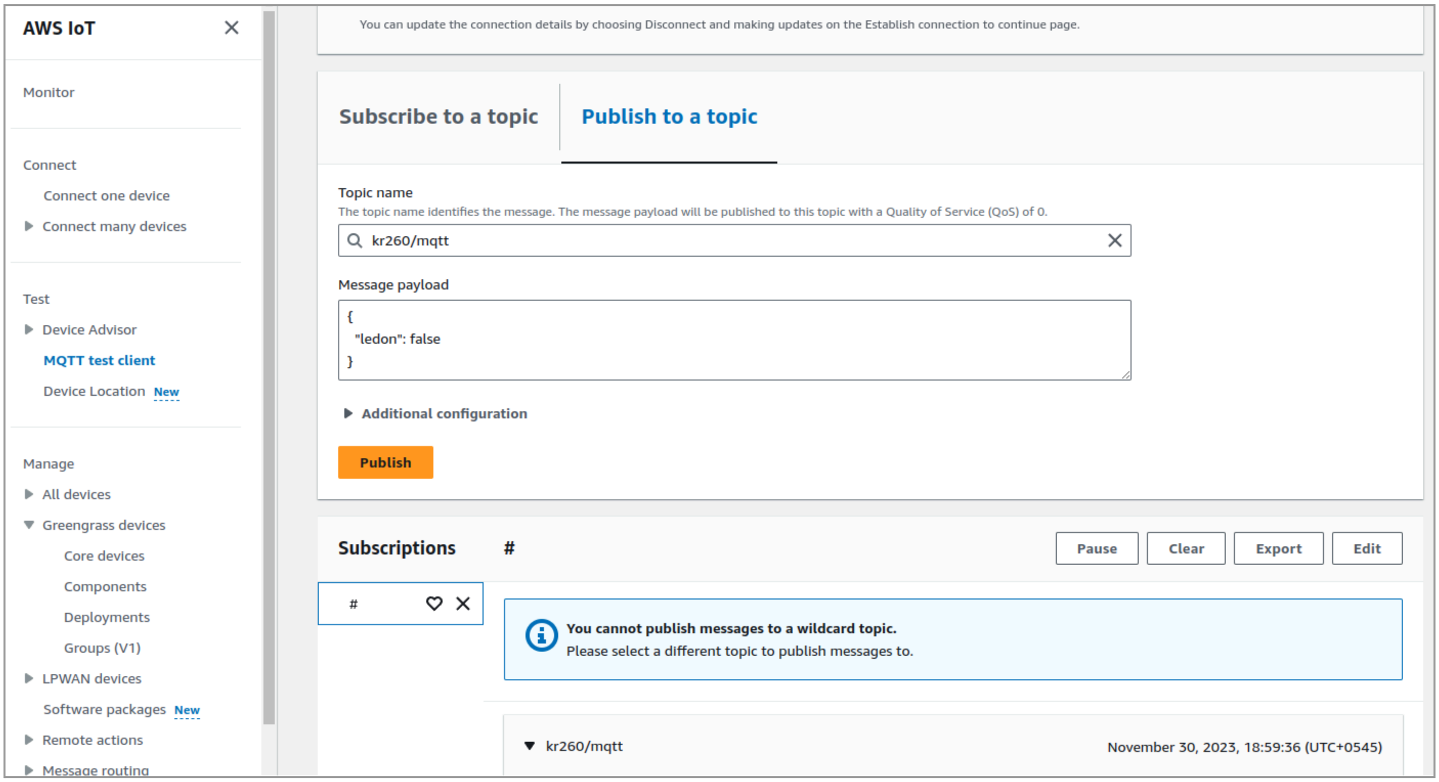
```
{
 "ledon": true
}
```

Subscriptions #

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

kr260/mqtt November 30, 2023, 18:57:32 (UTC+0545)

Now to switch off the LED send “false” message in the “kr260/mqtt” topic.



You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

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.
kr260/mqtt

Message payload

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

Subscriptions #

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

kr260/mqtt November 30, 2023, 18:59:36 (UTC+0545)