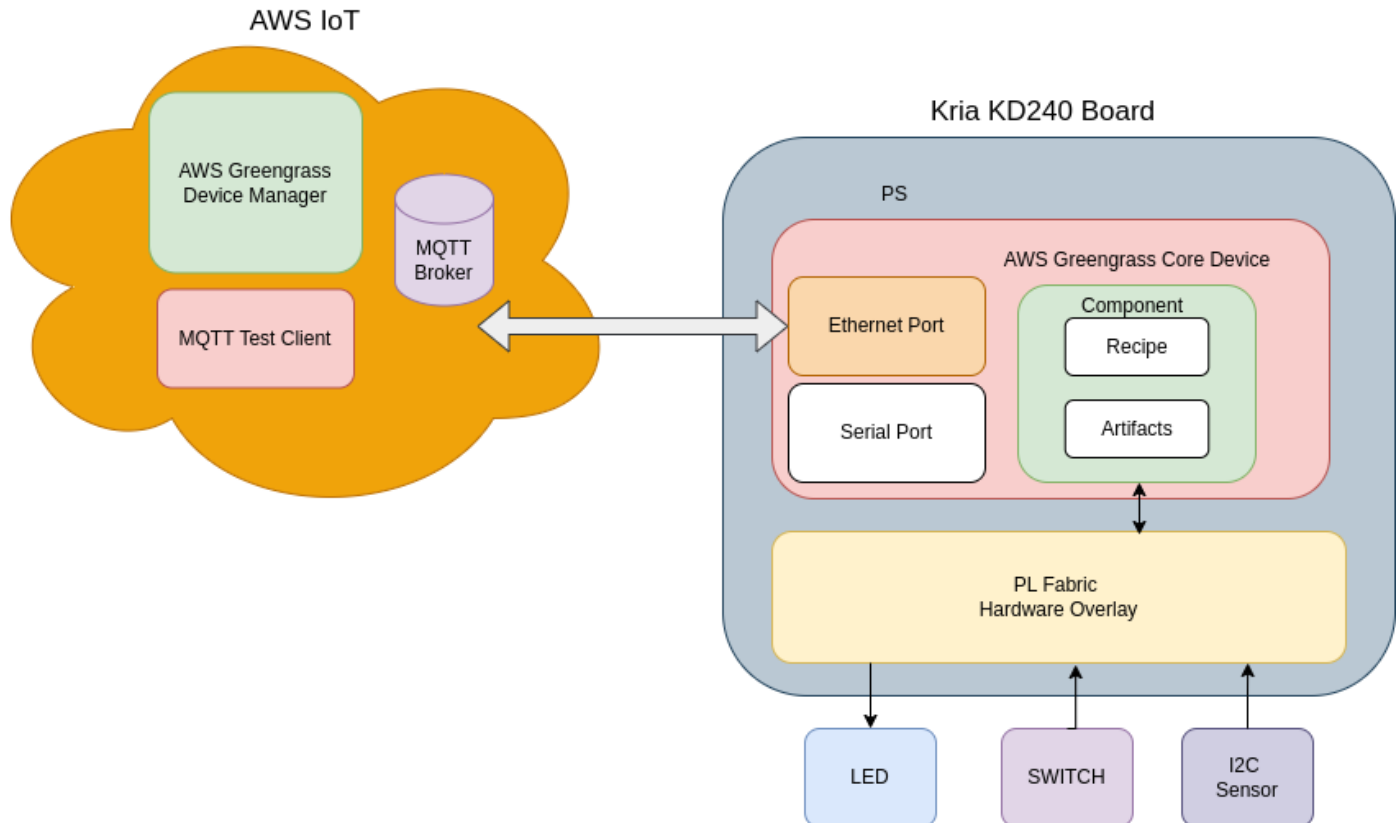


KD240 to AWS IoT Greengrass Architecture



This diagram shows the software and hardware architecture used in this tutorial. Kria KD240 board consists of 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 KD240 LED will be controlled through subscribed topic and also publish Switch pressed event to AWS IoT cloud.

Preparing Ubuntu 22.04 OS for KRIA KD240 board

Download the Ubuntu 22.04 image from the [download link](#)

Ubuntu Server 22.04

The version of optimised Ubuntu Server 22.04 is beta for now, the certified version is coming soon.

Works on:

- ✓ AMD Kria™ KD240 Drives 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](#)

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

Now boot the KD240 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 KD240 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 KD240 device console.

Installing hardware overlay

Get the KD240 firmware folder. It contains:

- kd240-gpio-i2c.bit.bin
- kd240-gpio-i2c.dtbo
- shell.json

Copy these file to the KD240 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 “kd240-gpio-i2c” at “/lib/firmware/xilinx” and copy the files in “kd240-gpio-i2c” folder.

```
cd /lib/firmware/xilinx
sudo mkdir kd240-gpio-i2c
Cd kd240-gpio-i2c
sudo cp <kd240-firmware directory>/kd240-gpio-i2c* ./
sudo cp <kd240-firmware directory>/shell.json ./
```

Next, check the available fpga firmware using `xmutl listapps` command. `kd240-gpio-i2c` will be available in the list.

```
ubuntu@kria:~$ sudo xmutl listapps
```

Accelerator	Accel_type	Base	Base_type	#slots(PL+AIE)	Active_slot
k24-starter-kits	XRT_FLAT	k24-starter-kits	XRT_FLAT	(0+0)	0,
kd240-gpio-i2c	XRT_FLAT	kd240-gpio-i2c	XRT_FLAT	(0+0)	-1

```
ubuntu@kria:~$
```

Next load the `kd240-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 xmutl unloadapp
sudo xmutl loadapp kd240-gpio-i2c
```

```
ubuntu@kria:~$ sudo xmutl loadapp kd240-gpio-i2c
[ 827.076900] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
[ 827.087054] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
[ 827.096939] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/afi0
[ 827.106454] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking0
[ 827.116398] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_0
[ 827.126422] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_1
[ 827.136450] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_gpio_0
[ 827.146477] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_0
kd240-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:

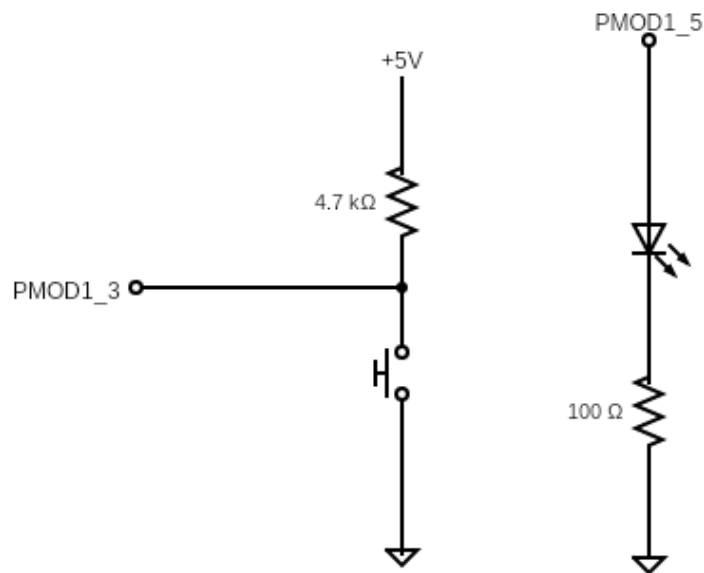
Using `gpiodetect` to get available gpio:

```
ubuntu@kria:~$ sudo gpiodetect
gpiochip0 [firmware:zynqmp-firmware:gpio] (4 lines)
gpiochip1 [zynqmp_gpio] (174 lines)
gpiochip2 [slg7xl45106] (8 lines)
gpiochip3 [800000000.gpio] (4 lines)
ubuntu@kria:~$
```

Here `gpiochip3` 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 - gpiochip3 line 0

PMOD1-> 7 - gpiochip3 line 1



Schematic for LED and Switch Connection

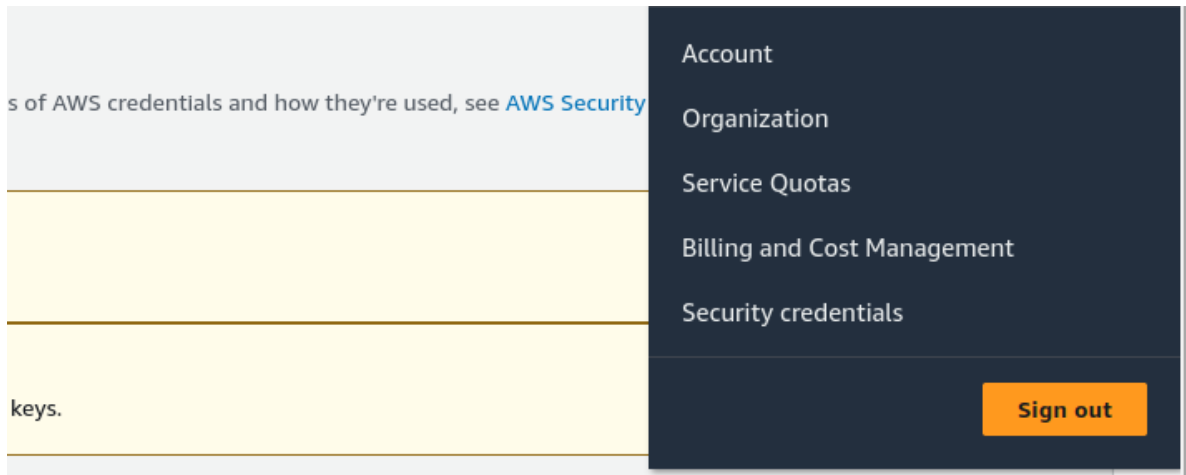
11	9	7	5	3	1	PMOD UPPER
12	10	8	6	4	2	PMOD LOWER
V _{cc}	GND	I/O	I/O	I/O	I/O	

PMOD port numbering

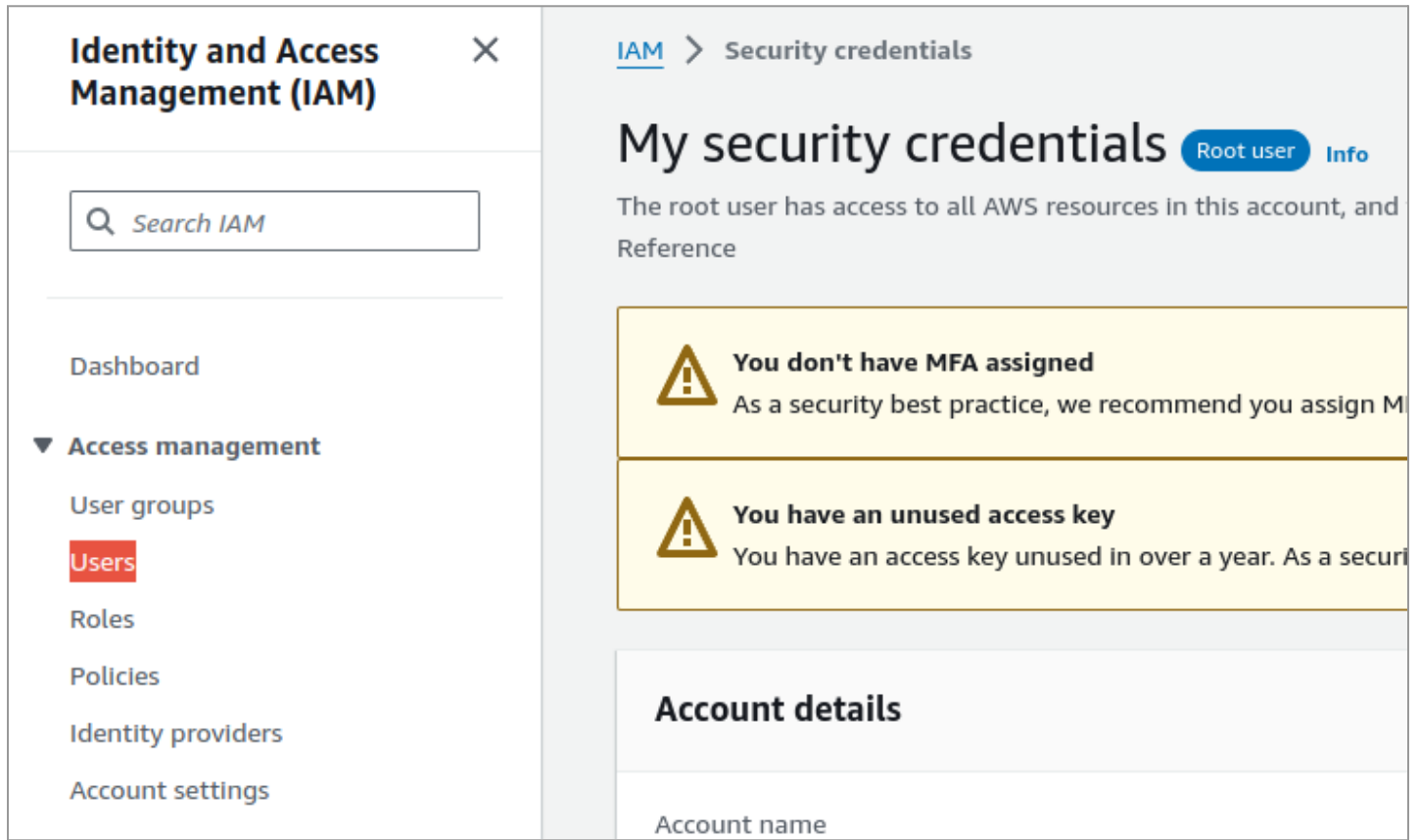
AWS IoT user creation

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.



Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

My security credentials Root user Info

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

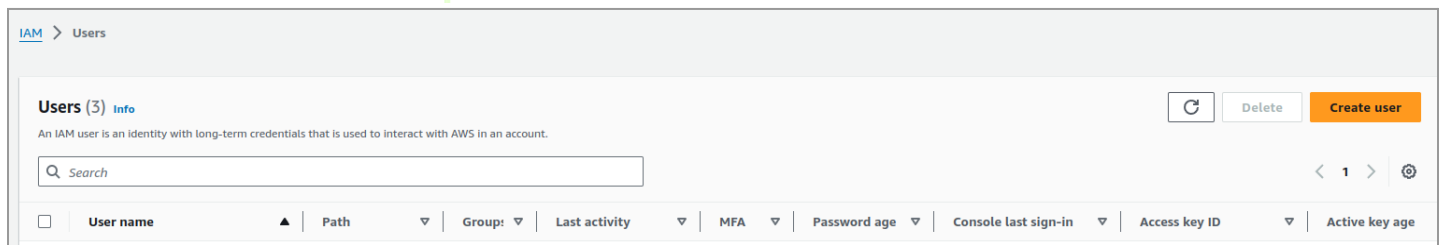
You don't have MFA assigned
As a security best practice, we recommend you assign MFA to the root user.

You have an unused access key
You have an access key unused in over a year. As a security best practice, we recommend you delete unused access keys.

Account details

Account name

- Now create a new user for KD240 device by clicking the "Create User" button.



Users (3) Info

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

Search

Refresh Delete Create user

<input type="checkbox"/>	User name	Path	Group	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age

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.

IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

User name

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 you create this IAM user. [Learn more](#)

Cancel **Next**

Next, update the Permissions options by attaching following policies by selecting "Attach policies directly" in Permission options:

- 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) [Refresh](#) [Create policy](#)

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

Filter by Type

All types

1 match

<input checked="" type="checkbox"/>	Policy name	Type	Attached entities
<input checked="" type="checkbox"/>	AWSIoTFullAccess	AWS managed	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
kd240-user

Console password type
None

Require password reset
No

Permissions summary

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 Users table.

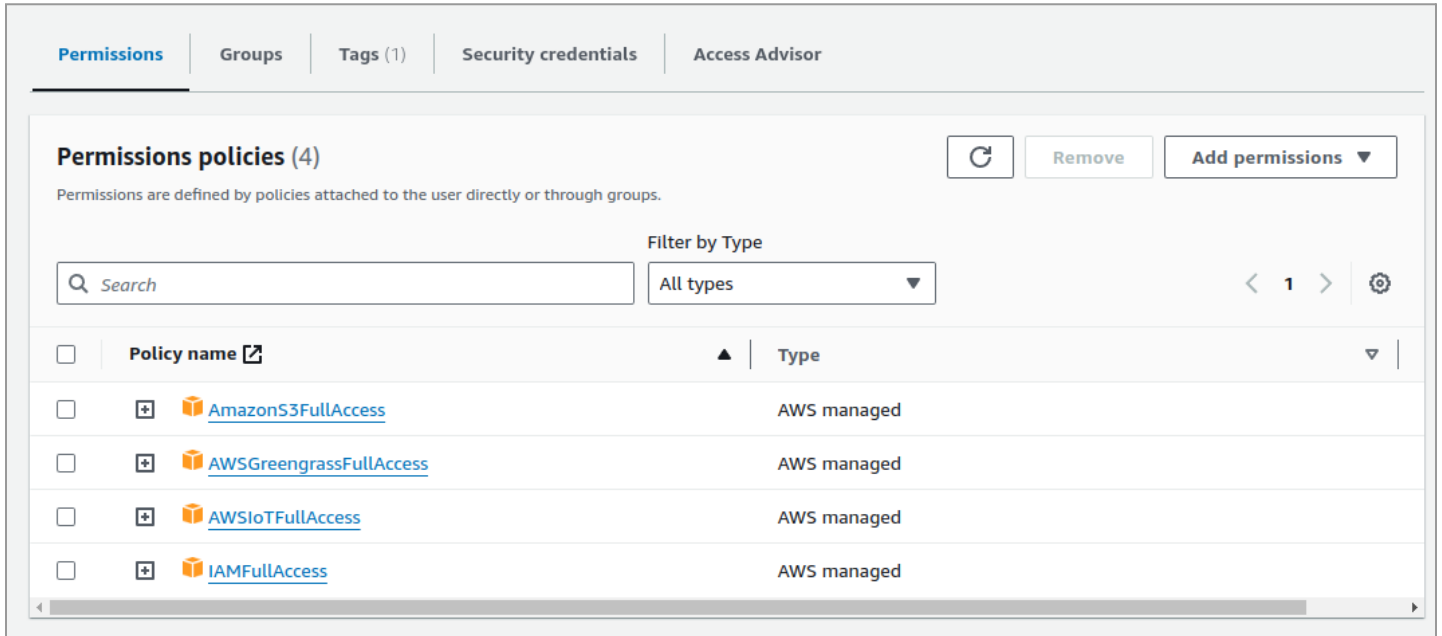
IAM > Users

Users (1/5) Info





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

Search

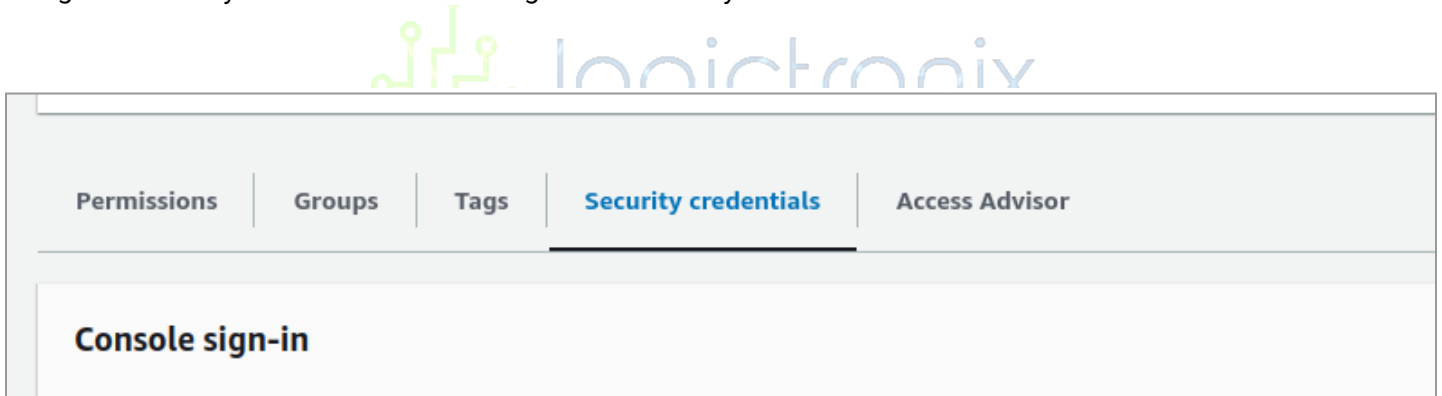
	User name	Path	Group	Last activity	MFA	Password age	Console last sign-in	Access key ID
<input type="checkbox"/>	dev-user	/	0	✓ Yesterday	-	-	-	Active - AKIAVPPB7PM...
<input checked="" type="checkbox"/>	kd240-user	/	0	-	-	-	-	-
<input type="checkbox"/>	kr260_dev	/	0	✓ 26 days ago	-	-	-	Active - AKIAVPPB7PM...
<input type="checkbox"/>	laxml	/	1	⚠ 557 days ago	-	⚠ 658 days	⚠ June 17, 2022, 16:24 (UT	Active - AKIAVPPB7PM...
<input type="checkbox"/>	monika	/	1	⚠ 520 days ago	-	⚠ 658 days	⚠ July 24, 2022, 13:51 (UTC	Active - AKIAVPPB7PM...



The screenshot shows the AWS IAM console interface for a user. The top navigation bar includes tabs for Permissions, Groups, Tags (1), Security credentials, and Access Advisor. The 'Permissions' tab is active, displaying 'Permissions policies (4)'. Below this, a message states: 'Permissions are defined by policies attached to the user directly or through groups.' There are buttons for 'Refresh', 'Remove', and 'Add permissions'. A search bar and a 'Filter by Type' dropdown (set to 'All types') are present. A table lists the attached policies:

<input type="checkbox"/>	Policy name ↗	Type
<input type="checkbox"/>	 AmazonS3FullAccess	AWS managed
<input type="checkbox"/>	 AWSGreengrassFullAccess	AWS managed
<input type="checkbox"/>	 AWSIoTFullAccess	AWS managed
<input type="checkbox"/>	 IAMFullAccess	AWS managed

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



The screenshot shows the AWS IAM console interface with the 'Security credentials' tab selected. The top navigation bar includes tabs for Permissions, Groups, Tags, Security credentials, and Access Advisor. The 'Security credentials' tab is active, displaying the 'Console sign-in' section.

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.

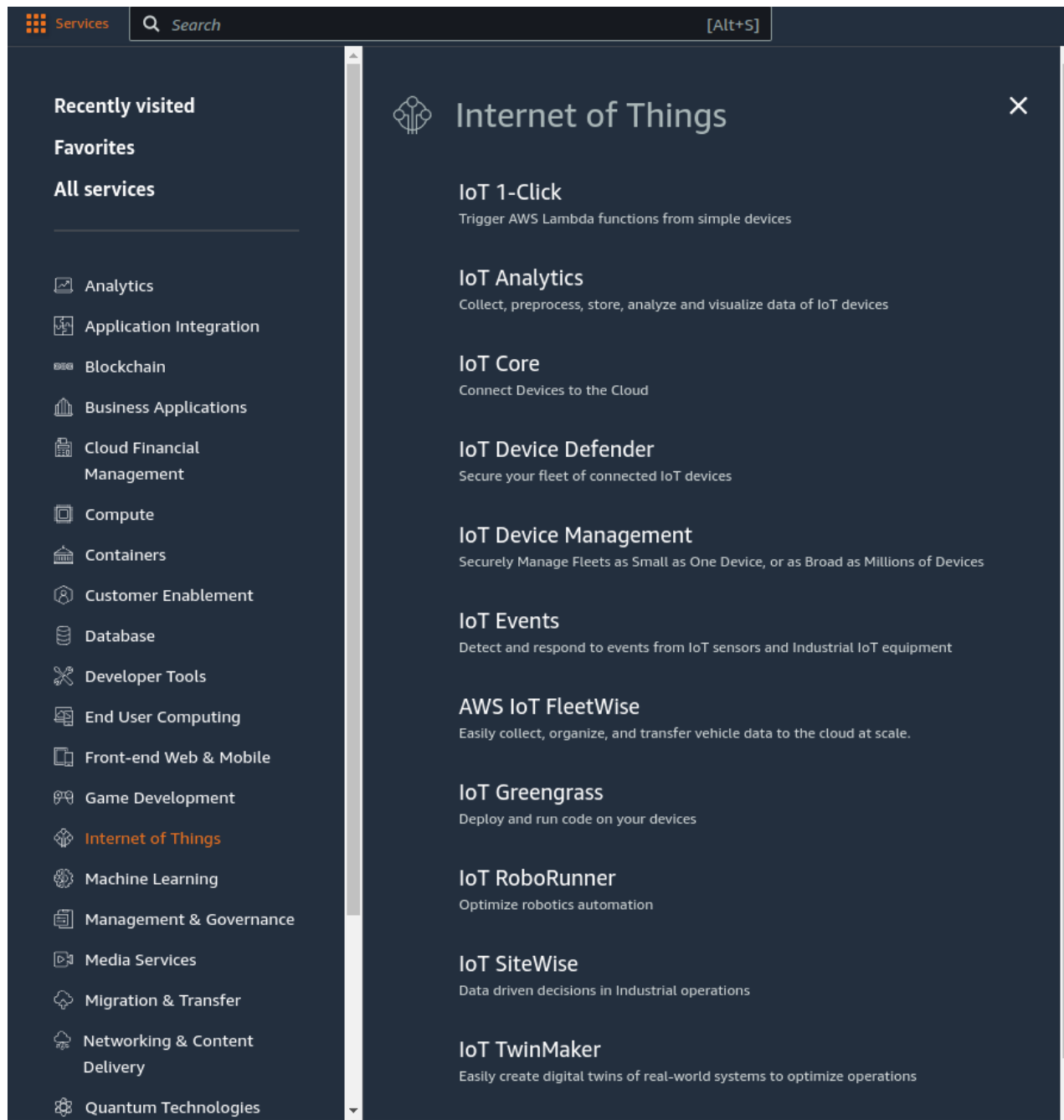
Cancel

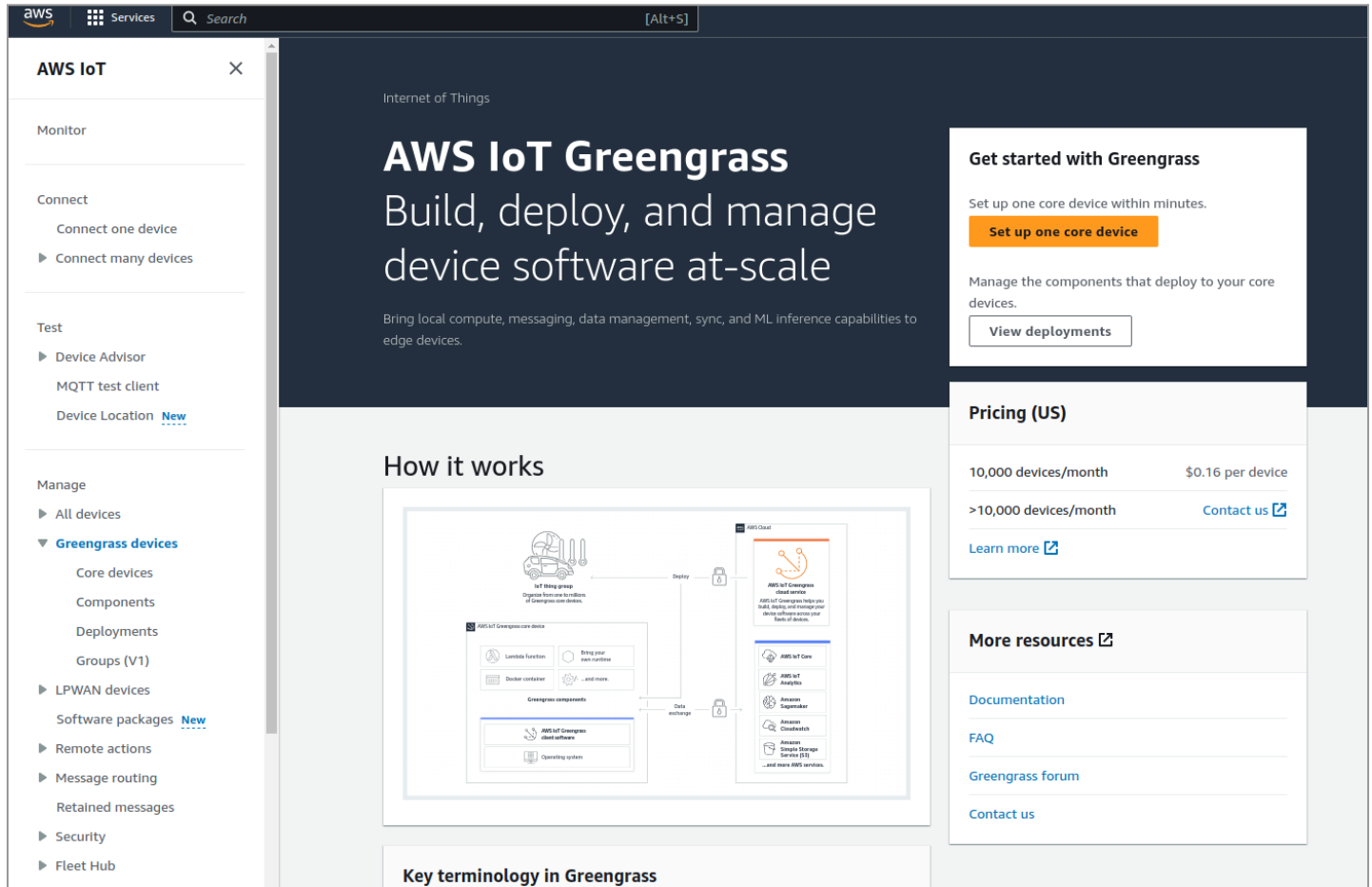
Next

Next save the "Access Key" and "Secret Access Key". We will need this later while using greengrass CLI in KD240 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





The screenshot shows the AWS IoT Greengrass console. On the left is a navigation menu with sections: Monitor, Connect (with sub-items 'Connect one device' and 'Connect many devices'), Test (with sub-items 'Device Advisor', 'MQTT test client', and 'Device Location'), and Manage (with sub-items 'All devices', 'Greengrass devices' (expanded to show 'Core devices', 'Components', 'Deployments', 'Groups (V1)'), 'LPWAN devices', 'Software packages', 'Remote actions', 'Message routing', 'Retained messages', 'Security', and 'Fleet Hub'). The main content area has a header 'Internet of Things' and 'AWS IoT Greengrass' with the tagline 'Build, deploy, and manage device software at-scale'. Below this is a 'How it works' diagram showing an 'IoT thing group' connecting to an 'AWS IoT Greengrass core device' via 'Deploy' and 'Data exchange'. The core device is shown with components like 'Lambda function', 'Bring your own container', 'Module container', and 'AWS IoT Greengrass client software'. To the right, there's a 'Get started with Greengrass' section with a 'Set up one core device' button and a 'View deployments' button. Below that is a 'Pricing (US)' table:

Pricing (US)	
10,000 devices/month	\$0.16 per device
>10,000 devices/month	Contact us

At the bottom right, there's a 'More resources' section with links to 'Documentation', 'FAQ', 'Greengrass forum', and 'Contact us'.

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 `kd240-ubuntu-dev1”

And set Group for KD240 group device.

[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.

kd240-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.

KD240UbuntuGroup

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

Step 3: Install the Greengrass Core software

Operating System

☒ Linux

☐ Windows

Step 3.1: Install Java on the device

The AWS IoT Greengrass Core software runs on Java. Follow instructions to install the Java runtime on the device. [Learn](#)

Now in KD240 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
sudo apt install unzip
```

Download and install Greengrass core software as instructed in AWS setup step 3.

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

Step 3.3: Run the installer

AWS IoT Greengrass provides an installer that you can use to set up a Greengrass core device in a few minutes. The installer runs on the device and does the following:

1. Provisions the Greengrass core device as an AWS IoT thing with a device certificate and default permissions. [Learn more](#)
2. Creates a system user and group, ggc_user and ggc_group, that the software uses to run components on the device.
3. Connects the device to AWS IoT.
4. Installs and runs the latest AWS IoT Greengrass Core software as a system service.

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
```

✓ Command copied

Copy

Next install the Greengrass core device as instructed in AWS setup step 3.:

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar
./GreengrassInstaller/lib/Greengrass.jar --aws-region us-east-1 --thing-name
kd240-ubuntu-dev1 --thing-group-name KD240UbuntuGroup --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 kd240-ubuntu-dev1 --thing-group-name KD240UbuntuGroup
--component-default-user ggc_user:ggc_group --provision true --setup-system-service true --deploy-
dev-tools true
```

✓ Command copied

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: [kd240-ubuntu-dev1]...
Found IoT policy "GreengrassV2IoTThingPolicy", reusing it
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "kd240-ubuntu-dev1"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [kd240-ubuntu-dev1]!
Adding IoT Thing [kd240-ubuntu-dev1] into Thing Group: [KD240UbuntuGroup]...
Successfully added Thing into Thing Group: [KD240UbuntuGroup]
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 `kd240-ubuntu-dev` in the list.

[AWS IoT](#) > [Greengrass](#) > Core devices

Greengrass core devices [Info](#)

Greengrass core devices (5)

Name	Status	Status reported
kr260-dev1	✓ Healthy	12 days ago
kr260-peta-dev1	✓ Healthy	12 days ago
kd240-ubuntu-dev1	✓ Healthy	1 minute ago
kr260-ubuntu-dev1	✓ Healthy	9 days ago
kd240-dev2	✗ Unhealthy	22 hours ago

In KD240 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":"10000000000","deploymentP
t":"","greengrassDataPlanePort":"8443","httpClient":{"},"iotCredEndpoint":"cluwyavs4wpvzg.credential
ions":"-Dlog.store=FILE","logging":{"},"mqtt":{"spooler":{"}}},"networkProxy":{"proxy":{"}},"platformOv
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: null
  State: RUNNING
  Configuration: null
Component Name: TelemetryAgent
  Version: 0.0.0
  State: RUNNING
  Configuration: null
Component Name: aws.greengrass.Cli
  Version: 2.12.1
  State: RUNNING
  Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
ubuntu@kria:~$ █
```

Next 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 KD240 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

Before installing the component need install 'python3-pip'

```
sudo apt install python3-pip
```

To install the above component run the following in the KD240 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: 3e4cad17-9a79-4e76-bc20-58a3b5b0093f  
ubuntu@kria:~$
```

Now check the installed component is in "running state":

```
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: 4835f786-9e25-4250-8b1c-c3acd3bc3de9
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":"10000000000","deploymentPollingInterval":10,"greengrassDataPlanePort":"8443","httpClient":{"url":"https://greengrass.amazonaws.com","method":"GET"},"iotCredEndpoint":"cluwyavs4wpvxg.credentials.io","logStore":"-Dlog.store=FILE","logging":{"logLevel":"INFO"},"mqtt":{"spooler":{"enabled":true},"topic":"greengrass"},"networkProxy":{"proxy":{"url":"https://greengrass.amazonaws.com","method":"GET"},"platformOverride":{"url":"https://greengrass.amazonaws.com","method":"GET"},"platformOverride":{"url":"https://greengrass.amazonaws.com","method":"GET"}}}}
Component Name: UpdateSystemPolicyService
  Version: 0.0.0
  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
Component Name: FleetStatusService
  Version: 0.0.0
  State: RUNNING
  Configuration: null
Component Name: com.example.mqtt
  Version: 1.0.0
  State: BROKEN
  Configuration: {"accessControl":{"aws.greengrass.ipc.mqttproxy":{"com.example.mqtt:mqttproxy:1":{"operations":["kd240/mqtt","kd240/button"]}}},"message":"hello"}
Component Name: aws.greengrass.Cli
  Version: 2.12.1
  State: RUNNING
  Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":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

#

kd240/mqtt

♥ ×

#

♥ ×

ⓘ You cannot publish messages to a wildcard topic.

Please select a different topic to publish messages to.

▼ kd240/button

December 27, 2023, 14:04:43 (UTC+0545)

```
{
  "button": "button pressed",
  "timestamp": 1703665181137
}
```

► Properties

Pause

Clear

Export

Edit

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

Now to control the LED, publish the message to “kd240/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.

kd240/mqtt

Message payload

{
 "ledon": true
}

► Additional configuration

Publish

Subscriptions

#

#

♥ ×

ⓘ You cannot publish messages to a wildcard topic.

Please select a different topic to publish messages to.

▼ kd240/mqtt

December 27, 2023, 14:06:22 (UTC+0545)

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

Pause

Clear

Export

Edit

Created by www.LogicTronix.com | info@logictronix.com

Sanam Shakya | 19

Now to switch off the LED send "false" message in the "kd240/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.

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.

▼ kd240/mqtt

December 27, 2023, 14:06:59 (UTC+0545)

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

► Properties