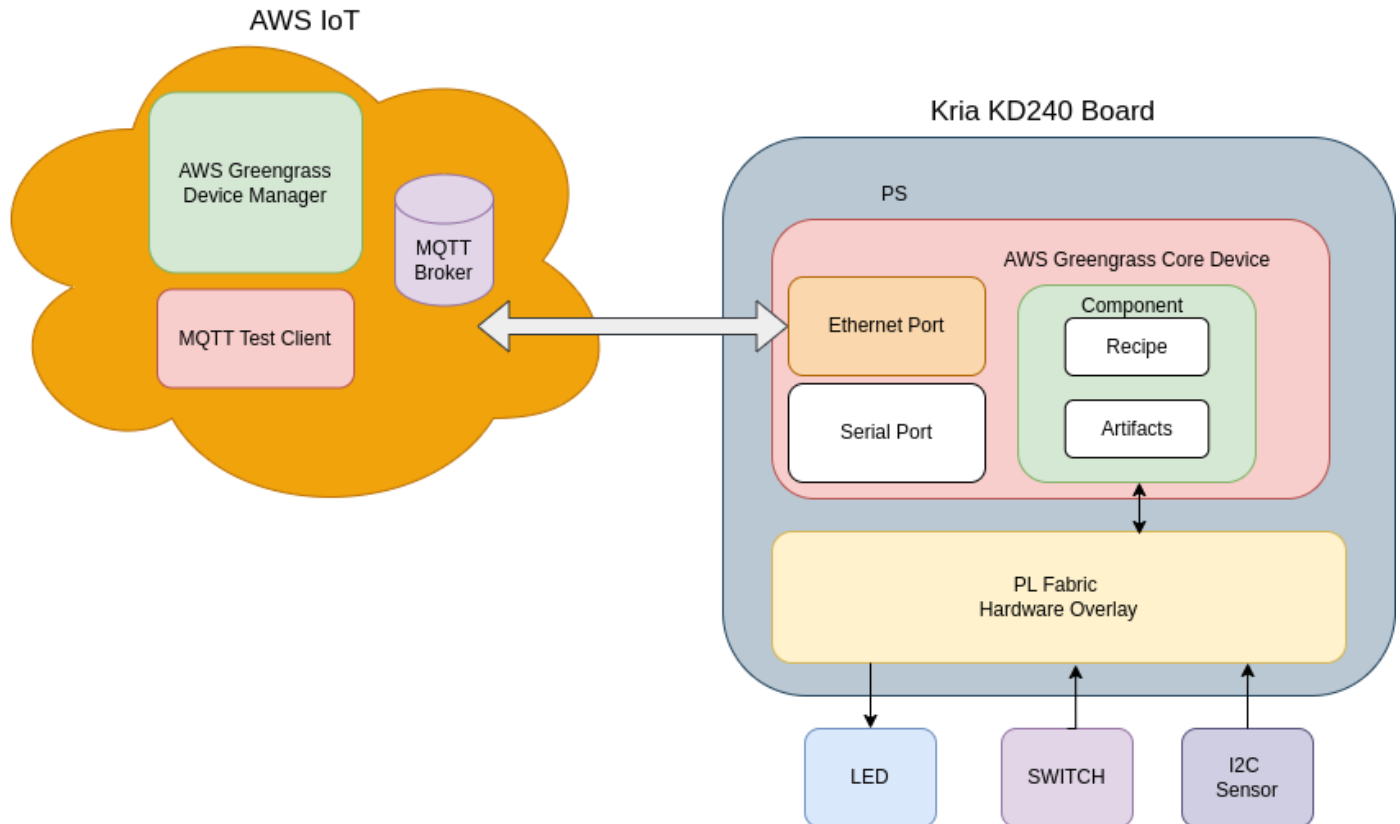


## KD240 to AWS IoT Greengrass Architecture



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.

## Creating Petalinux Project

### Create Petalinux from BSP

In host machine, create the petalinux project using provided BSP:

BSP feature:

- Petalinux version 2023.1
- Added meta-aws layer for installing required dependencies required for running greengrass core device in KD240
- Rootfs packages : packagegroup-buildessential, git, libgpiod, libgpiod-dev, libgpiod-tools
- Enabled FPGA-manager

Run following commands to create petalinux project after sourcing the Petalinux 2023.1 environment:

```
$ source <Path to Petalinux 2023.1>/settings.sh
$ petalinux-create -t project -s xilinx-kd240-starterkit-aws-iot.bsp -n kd240-aws-iot
$ cd kd240-aws-iot
```

Here “kd240-aws-iot” is the directory created by petalinux-create command, you can change the name according to your need. This directory is the petalinux-project base directory, which we will be using in further steps.

Next build the project:

```
$ petalinux-config --silentconfig
$ petalinux-build
```

Here is the console log after running the petalinux-build.

```
sanam@sanam-kg10n1: ~/kr260-p2022-2-default$ petalinux-build
[INFO] Sourcing buildtools
[INFO] Building project
[INFO] Sourcing build environment
[INFO] Generating workspace directory
INFO: Bitbake petalinux-image-minimal
NOTE: Started PSIServer with Suffix: /media/sanam/workspace2/sanam/kr260-p2022-2-default/build/cache/prserv.sqlite3, Address: 127.0.0.1:46633, PID: 558748
loading cache: 100% |
loaded 0 entries from dependency cache.
Parsing recipes: 100% |#####| ETA: 0:00:00
Parsing of 4532 .bb files complete (0 cached, 4532 parsed). 6625 targets, 626 skipped, 1 masked, 0 errors.
NOTE: resolving any missing task queue dependencies
Initialising tasks: 100% |#####| Time: 0:00:28
Checking state mirror object availability: 100% |#####| Time: 0:00:39
State summary: wanted 108 local, 28 network, 653 missed, 404 current, 2726 (62% match, 89% complete)
Removing 2 state objects for arch xilinx_k26_kr: 100% |#####| Time: 0:00:00
NOTE: Executing tasks
NOTE: Tasks Summary: Attempted 9498 tasks of which 9481 didn't need to be rerun and all succeeded.
INFO: Failed to copy built images to tftp dir: /tftpboot
[INFO] Successfully built project
sanam@sanam-kg10n1: ~/kr260-p2022-2-default$
```

Next create the SD card image with the following commands:

```
petalinux-package --wic --images-dir images/linux/ --bootfiles
"ramdisk.cpio.gz,u-boot,boot.scr,Image,system.dtb,system-zynqmp-sck-kd-g-revA.dtb"
--disk-name "sda"
```

This will create the petalinux-sdimage.wic image at <petalinux project directory>/image/linux folder. Copy the created wic image to SD card using tools like Balena Etcher.

## Installing hardware overlay

After booting the previously used SD card into KD240.

Login to KD240 serial terminal using login name: petalinux  
For the first login one has to update the new password.

```
xilinx-kr260-starterkit-20222 login: petalinux
You are required to change your password immediately (administrator enforced).
New password:
Retype new password:
xilinx-kr260-starterkit-20222:~$
```

Next copy the KD240 firmwares to KD240 using network tools like scp or manually copying the firmware files at /home/petalinux directory of SD card.

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
sudo cp <kd240-firmware directory>/kd240_gpio_i2c* ./
sudo cp <kd240-firmware directory>/shell.json ./
```

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

```
xilinx-kd240-starterkit-20231:~$ sudo xmutil listapps
Password:
      Accelerator      Accel_type      Base      Base_type      #slots(PL+AIE)      Active_slot
      kd240-gpio-i2c      XRT_FLAT      kd240-gpio-i2c      XRT_FLAT      (0+0)      0,
```

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 xmutil unloadapp
sudo xmutil loadapp kd240-gpio-i2c
```

```
xilinx-kd240-starterkit-20231:~$ sudo xmutil loadapp kd240-gpio-i2c
Password:
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /_symbols_/afi0
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /_symbols_/clocking0
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /_symbols_/axi_intc_0
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /_symbols_/axi_intc_1
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /_symbols_/axi_gpio_0
Dec 25 03:53:18 xilinx-kd240-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /_symbols_/axi_iic_0
kd240-gpio-i2c: loaded to slot 0
xilinx-kd240-starterkit-20231:~$
```

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

## Installing gpiod python modules

GPIOD packages are required to access the GPIO channels. It also provides python binding for accessing GPIO in python programming. Install the gpiod python modules:

```
sudo pip3 install gpiod
```

Now we can check the available gpio using gpiod applications:

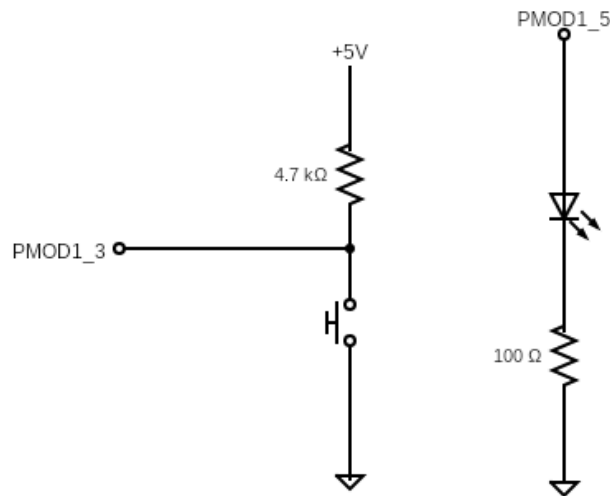
Using `gpiodetect` to get available gpio:

```
xilinx-kd240-starterkit-20231:~$ sudo gpiodetect
gpiochip0 [firmware:zynqmp-firmware:gpio] (4 lines)
gpiochip1 [zynqmp_gpio] (174 lines)
gpiochip2 [slg7xl45106] (8 lines)
gpiochip3 [80000000.gpio] (4 lines)
xilinx-kd240-starterkit-20231:~$
```

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	<u>PMOD UPPER</u>
12	10	8	6	4	2	<u>PMOD LOWER</u>
<u>V<sub>CC</sub></u>	<u>GND</u>	I/O	I/O	I/O	I/O	

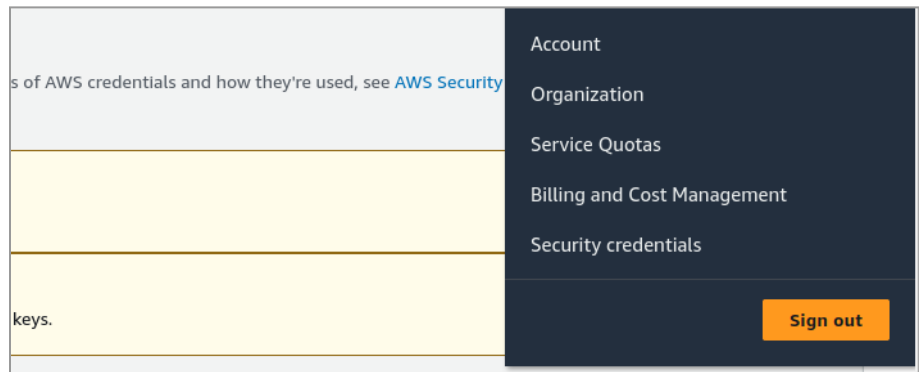
PMOD port numbering

We will be using these gpios while creating component of Greengrass AWS core device in KD240.

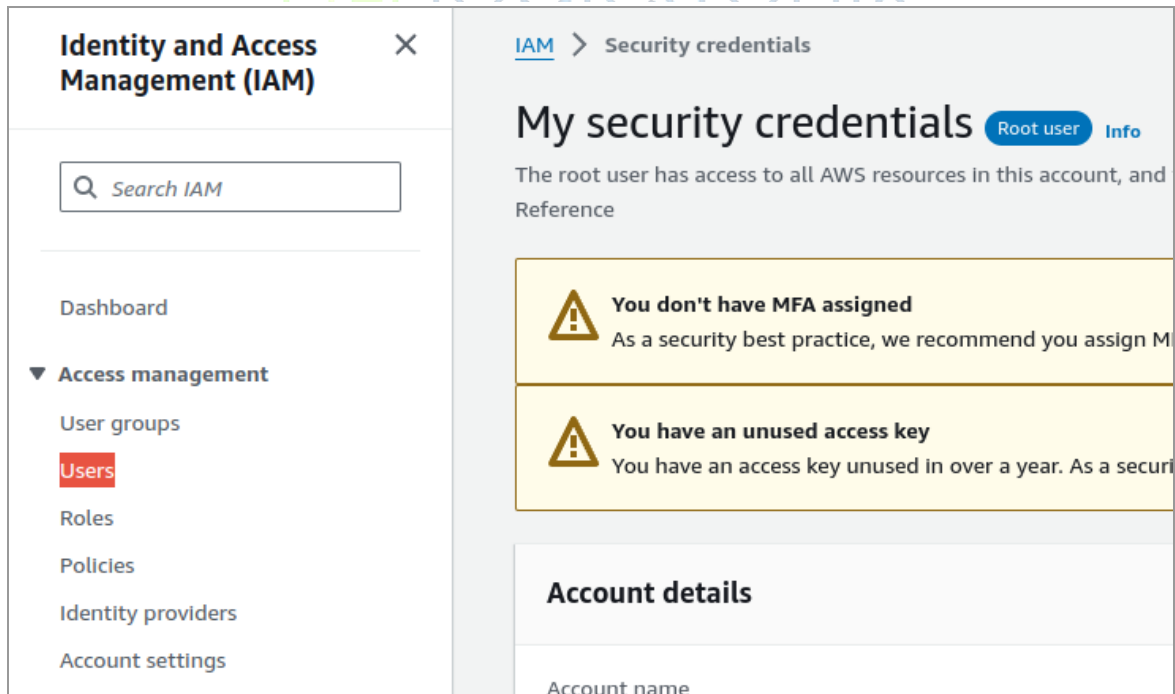
## AWS IoT user creation

For a non human access to AWS services one has to create a user with required permissions. Follow following steps to create the user for IoT end devices.

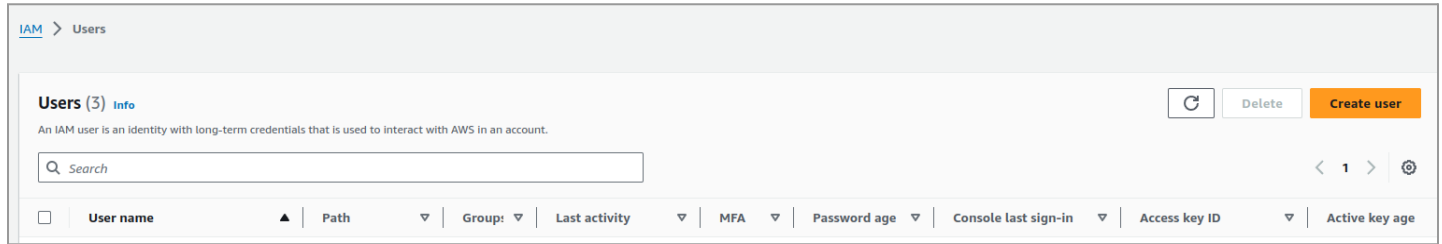
- Login to AWS console
- Next go to `Security credentials` link available at root user drop down at top right corner of the AWS console



- 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 KD240 device by clicking the "Create User" button.

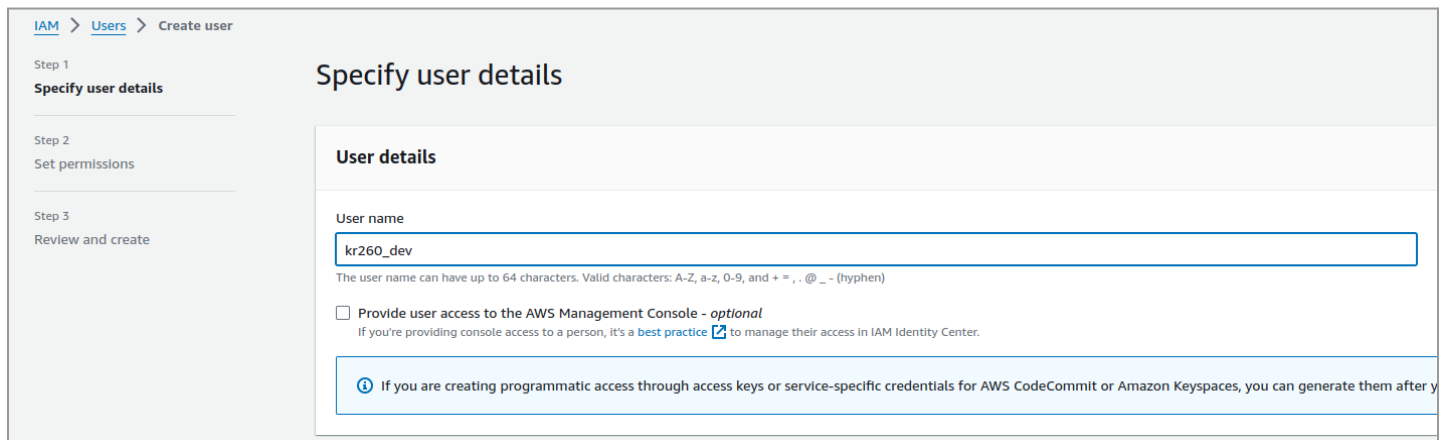


The screenshot shows the AWS IAM console 'Users' page. At the top, there's a breadcrumb 'IAM > Users'. Below it, a header shows 'Users (3)' with an 'Info' link. A description states: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' To the right are buttons for 'Refresh', 'Delete', and 'Create user'. A search bar is present. Below the search bar is a table with columns: 'User name', 'Path', 'Group', 'Last activity', 'MFA', 'Password age', 'Console last sign-in', 'Access key ID', and 'Active key age'. The table is currently empty.

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.



The screenshot shows the 'Specify user details' form in the AWS IAM console. The breadcrumb is 'IAM > Users > Create user'. On the left, there's a sidebar with 'Step 1: Specify user details', 'Step 2: Set permissions', and 'Step 3: Review and create'. The main section is titled 'Specify user details' and contains a 'User details' section. It has a 'User name' field with the value 'kr260\_dev'. Below it, a note says: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)'. There is an unchecked checkbox labeled 'Provide user access to the AWS Management Console - optional' with a note: 'If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.' At the bottom, there's a blue box with an information icon and text: '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.

Filter by Type
All types
1 match

<input checked="" type="checkbox"/>	Policy name	Type	Attached entities
<input checked="" type="checkbox"/>	<b>AWSIoTFullAccess</b>	AWS managed	1

► Set permissions boundary - optional

Cancel
Previous
Next

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

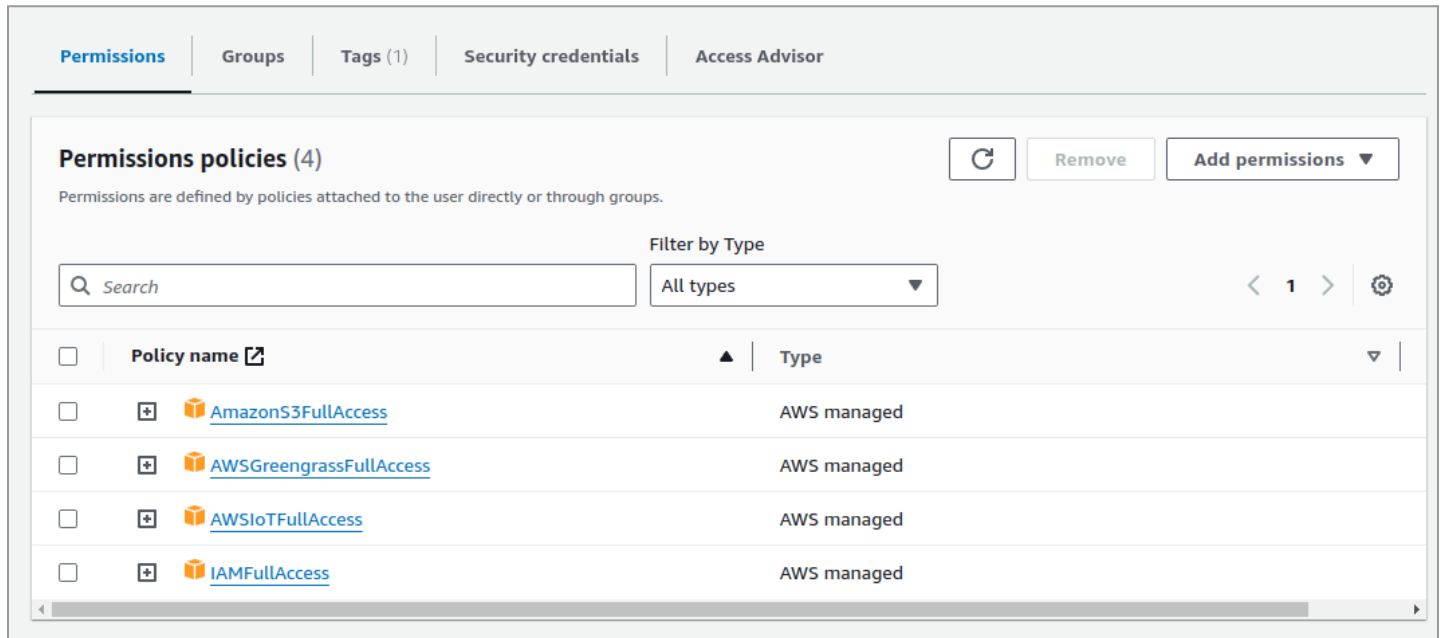
IAM > Users

### Users (4) Info

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

<input type="checkbox"/>	User name	Path	Group	Last activity	MFA	Password age	Console last sign-in
<input type="checkbox"/>	<a href="#">dev-user</a>	/	0	✓ 2 days ago	-	-	-
<input type="checkbox"/>	<a href="#">kr260_dev</a>	/	0	-	-	-	-





**Permissions policies (4)**

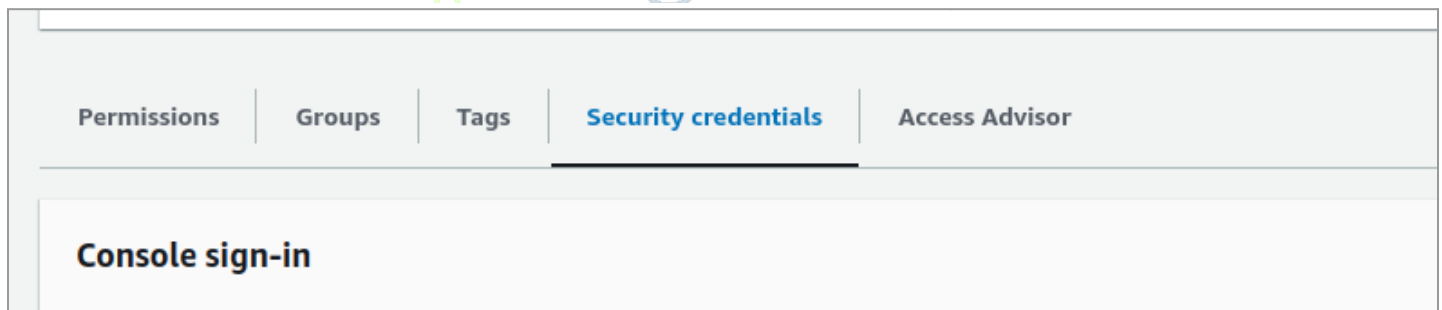
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

Policy name	Type
<a href="#">AmazonS3FullAccess</a>	AWS managed
<a href="#">AWSGreengrassFullAccess</a>	AWS managed
<a href="#">AWSIoTFullAccess</a>	AWS managed
<a href="#">IAMFullAccess</a>	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.

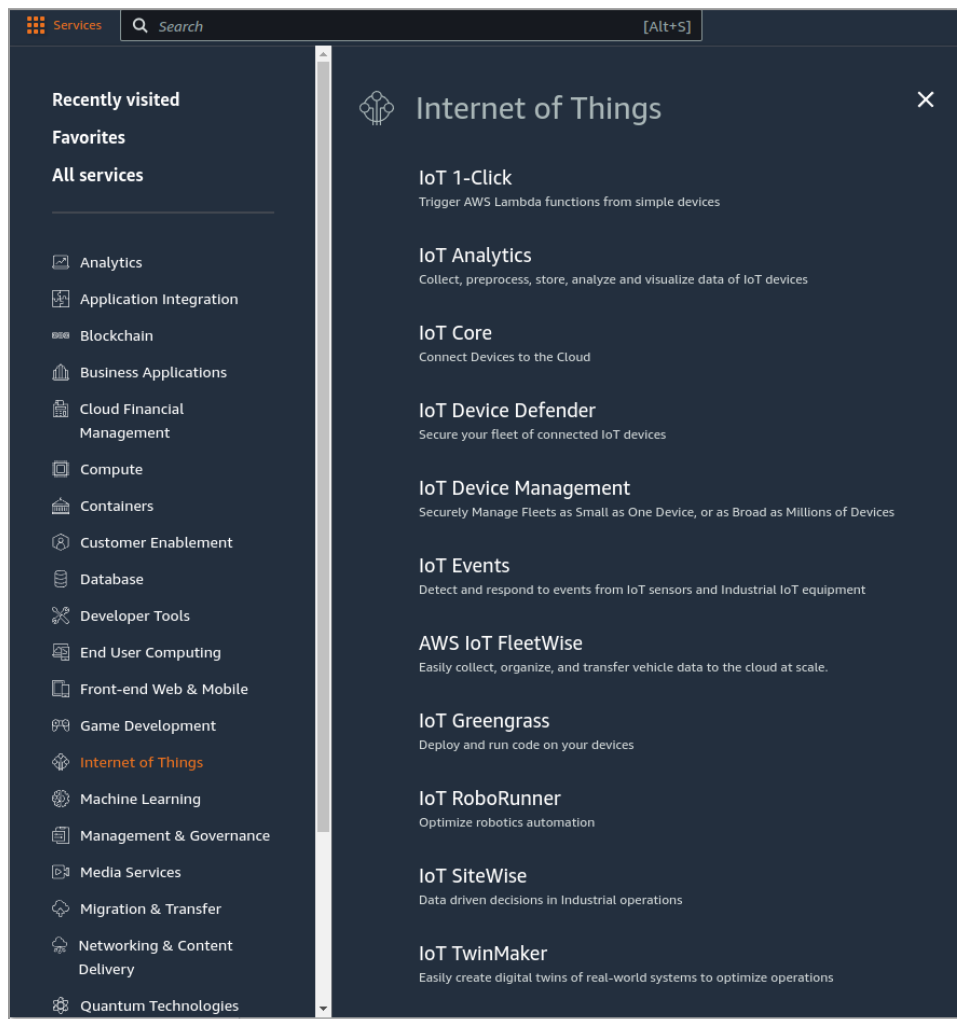
Cancel

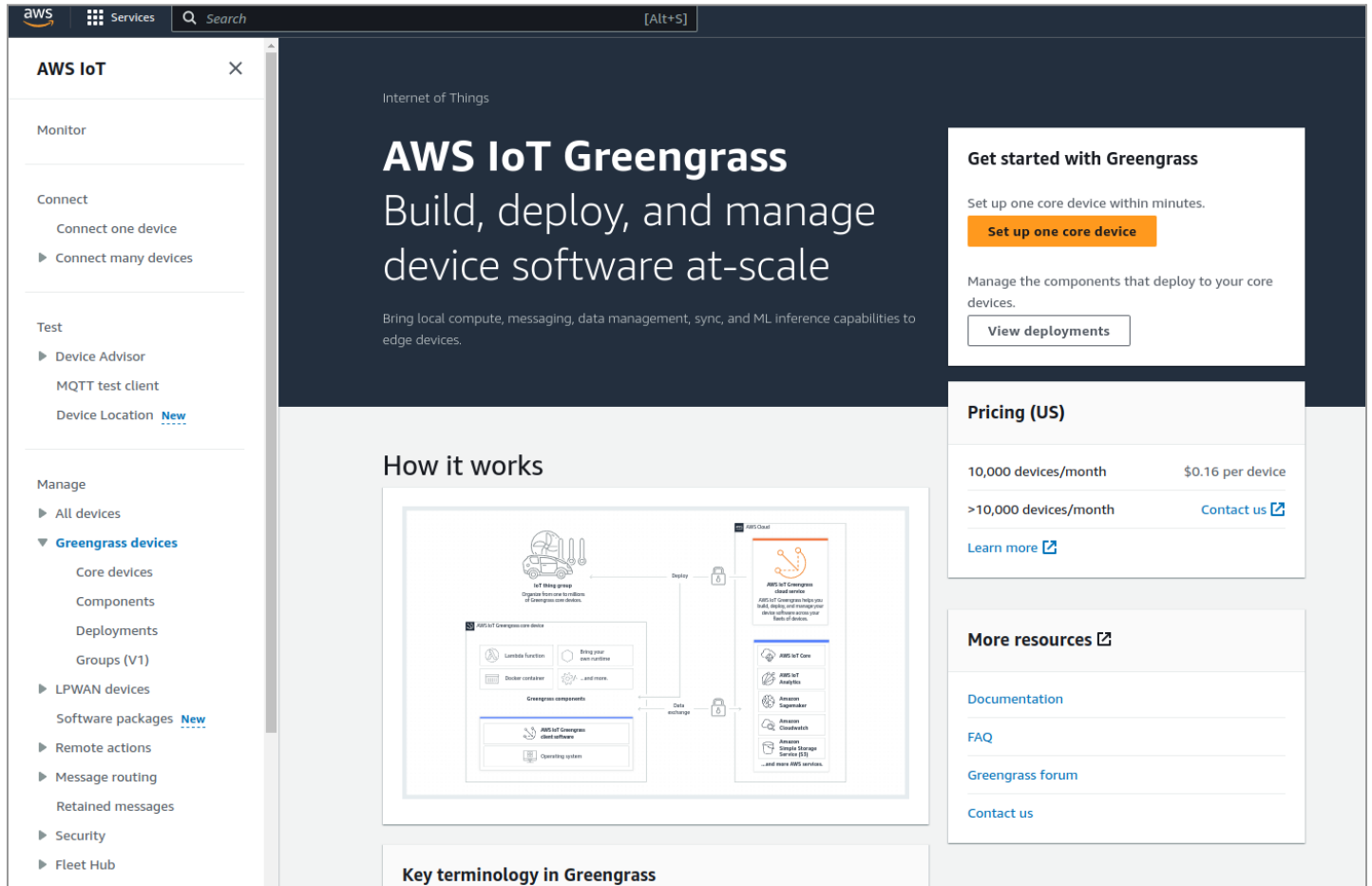
Next

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 in KD240 board

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 interface. On the left is a navigation sidebar 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 is titled 'Internet of Things' and 'AWS IoT Greengrass'. It features the headline 'Build, deploy, and manage device software at-scale' and a sub-headline 'Bring local compute, messaging, data management, sync, and ML inference capabilities to edge devices.' Below this is a 'How it works' diagram showing the flow from 'IoT things group' to 'AWS IoT Greengrass core device' and then to 'AWS IoT Greengrass cloud service'. The diagram also shows 'Greengrass components' (Lambda function, Docker container, and AWS IoT Greengrass client software) and 'Data exchange' between the core device and the cloud service.

On the right side, there are three panels: 'Get started with Greengrass' with a 'Set up one core device' button and a 'View deployments' button; 'Pricing (US)' with a table showing '10,000 devices/month' at '\$0.16 per device' and '>10,000 devices/month' with a 'Contact us' link; and 'More resources' 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 can change the Core device name like `kd240-dev2”

[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.

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

**⚠** An AWS IoT thing with this name already exists. You can create a core device from the existing thing, or enter a different name to create a new thing.

### 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.

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

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>
```

Download and install Greengrass core software by running the script available in AWS greengrass core device setup page:

## 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 by running the script available in AWS greengrass core device setup page:

## 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.

 Command copied

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar --aws-region us-east-1 --thing-name kd240-dev1 --thing-group-name KD240DevGroup --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:


```
Provisioning AWS IoT resources for the device with IoT Thing Name: [kd240-dev1]...
Found IoT policy "GreengrassV2IoTThingPolicy", reusing it
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "kd240-dev1"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [kd240-dev1]!
Adding IoT Thing [kd240-dev1] into Thing Group: [KD240DevGroup]...
Successfully added Thing into Thing Group: [KD240DevGroup]
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
Unable to set up Nucleus as a system service
xilinx-kd240-starterkit-20231:~$
```


Now in Greengrass set up page, one can view the Greengrass core devices and find above `kd240-dev2` in the list.





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

### Greengrass core devices Info

Greengrass core devices (4)



< 1 > 

Name	Status	Status reported
<a href="#">kr260-dev1</a>	 Healthy	10 days ago
<a href="#">kr260-peta-dev1</a>	 Healthy	10 days ago
<a href="#">kd240-dev2</a>	 Healthy	4 minutes ago
<a href="#">kr260-ubuntu-dev1</a>	 Healthy	7 days ago

In KD240 terminal one can get the device components by using `greengrass-cli`:

```
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":"1000000000","deploymentPollingFrequencySeconds":"15","envStage":"prod","fipsMode":"false","fleetStatus":{"periodicStatusPublishIntervalSeconds":86400.0},"greengrassDataPlaneEndpoint":"","greengrassDataPlanePort":8443,"httpClient":{"},"iotCredEndpoint":"cliwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.com","iotDataEndpoint":"a9jcs3obcutf8v-ats.iot.us-east-1.amazonaws.com","iotRoleAlias":"GreengrassV2TokenExchangeRoleAlias","jvmOptions":"","log.store=FILE","logging":{"},"mqtt":{"spooler":{"},"networkProxy":{"proxy":{"},"platformOverride":{"},"runWithDefault":{"posixUser":"ggc_user:ggc_group"},"s3EndpointType":"GLOBAL","telemetry":{"}}
Component Name: UpdateSystemPolicyService
  Version: 0.0.0
  State: RUNNING
  Configuration: null
Component Name: FleetStatusService
  Version: null
  State: RUNNING
  Configuration: null
Component Name: aws.greengrass.Cli
  Version: 2.12.1
  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
xilinx-kd240-starterkit-20231:/greengrass/v2/bin$
```

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

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"
```

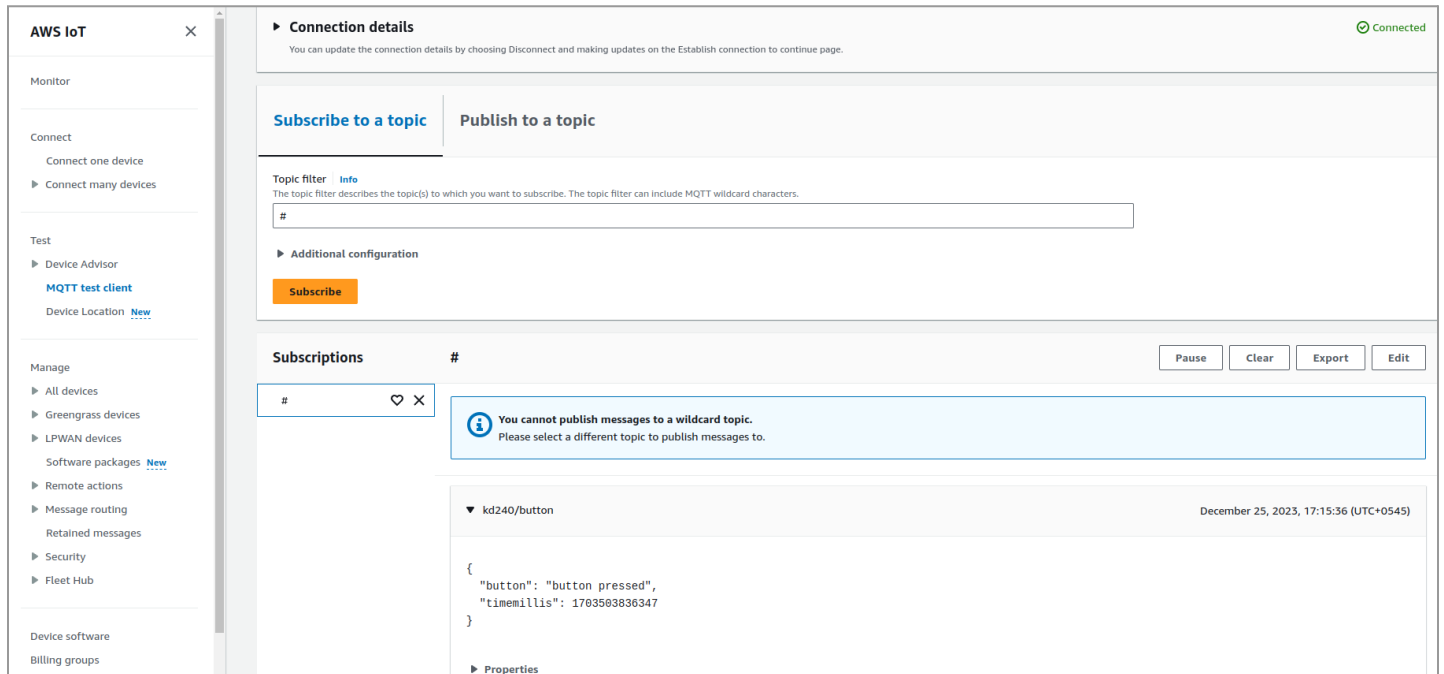
```
xilinx-kd240-starterkit-20231:~$ sudo /greengrass/v2/bin/greengrass-cli deployment create \  
> --recipeDir ~/components/recipe \  
> --artifactDir ~/components/artifacts \  
> --merge "com.example.mqtt=1.0.0"  
Password:  
Local deployment submitted! Deployment Id: 01fd9c40-bc9b-4650-9e07-55d89b305eb0  
xilinx-kd240-starterkit-20231:~$
```

Now check the installed component is in "running state"



```
Local deployment submitted: Deployment ID: 01f03c40-bc30-4030-9c07-330030303000
xilinx-kd240-starterkit-20231:~$ 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","deploymentPol
ort":"8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.com
":{},"networkProxy":{"proxy":{},"platformOverride":{},"runWithDefault":{"posixShell":"sh","posixUse
Component Name: UpdateSystemPolicyService
  Version: 0.0.0
  State: RUNNING
  Configuration: null
Component Name: FleetStatusService
  Version: null
  State: RUNNING
  Configuration: null
Component Name: aws.greengrass.Cli
  Version: 2.12.1
  State: RUNNING
  Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: com.example.mqtt
  Version: 1.0.0
  State: RUNNING
  Configuration: {"accessControl":{"aws.greengrass.ipc.mqttproxy":{"com.example.mqtt:mqttproxy:1":{"
tton"]}}},"message":"hello"}
Component Name: TelemetryAgent
  Version: 0.0.0
  State: RUNNING
  Configuration: null
Component Name: DeploymentService
  Version: 0.0.0
  State: RUNNING
  Configuration: null
xilinx-kd240-starterkit-20231:~$ █
```

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



**AWS IoT** ×

Monitor

Connect

- Connect one device
- Connect many devices

Test

- Device Advisor
- MQTT test client**
- Device Location New

Manage

- All devices
- Greengrass devices
- LPWAN devices
- Software packages New
- Remote actions
- Message routing
- Retained messages
- Security
- Fleet Hub

Device software

Billing groups

**Connection details** Connected

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

**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.

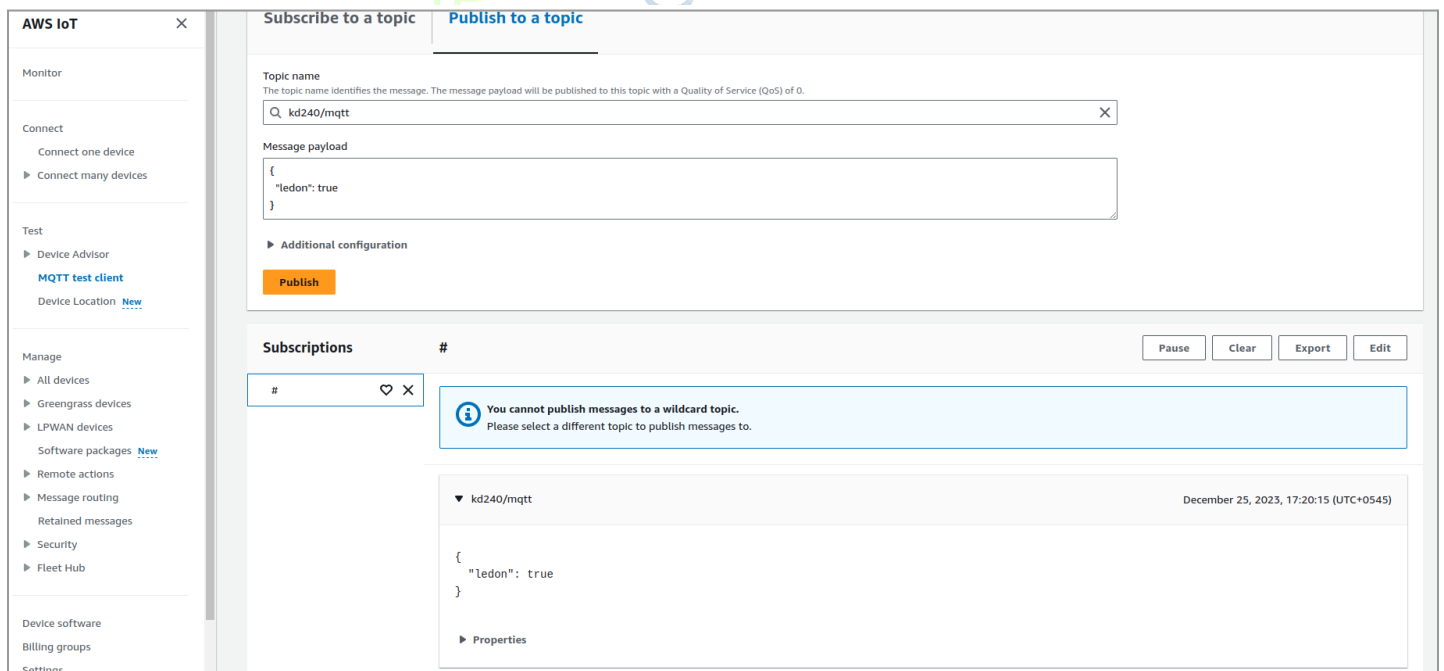
▼ kd240/button December 25, 2023, 17:15:36 (UTC+0545)

```
{
  "button": "button pressed",
  "tinemillis": 1703503836347
}
```

**Properties**

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.



**AWS IoT** ×

Monitor

Connect

- Connect one device
- Connect many devices

Test

- Device Advisor
- MQTT test client**
- Device Location New

Manage

- All devices
- Greengrass devices
- LPWAN devices
- Software packages New
- Remote actions
- Message routing
- Retained messages
- Security
- Fleet Hub

Device software

Billing groups

Settings

**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.

Q kd240/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.

▼ kd240/mqtt December 25, 2023, 17:20:15 (UTC+0545)

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

**Properties**

Now to switch off the LED send “false” message in the “kd240/mqtt” topic.

**AWS IoT** ×

Monitor

Connect

Connect one device

▶ Connect many devices

Test

▶ Device Advisor

MQTT test client

Device Location New

Manage

▶ All devices

▶ Greengrass devices

▶ LPWAN devices

Software packages New

▶ Remote actions

▶ Message routing

Retained messages

▶ Security

▶ Fleet Hub

Device software

Billing groups

Settings

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.

Q kd240/mqtt ×

Message payload

{  
  "ledon": false  
}

▶ Additional configuration

Publish

Subscriptions

#

Pause

Clear

Export

Edit

# ♥ ×

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

▼ kd240/mqtt

December 25, 2023, 17:19:37 (UTC+0545)

{  
  "ledon": false  
}

▶ Properties

