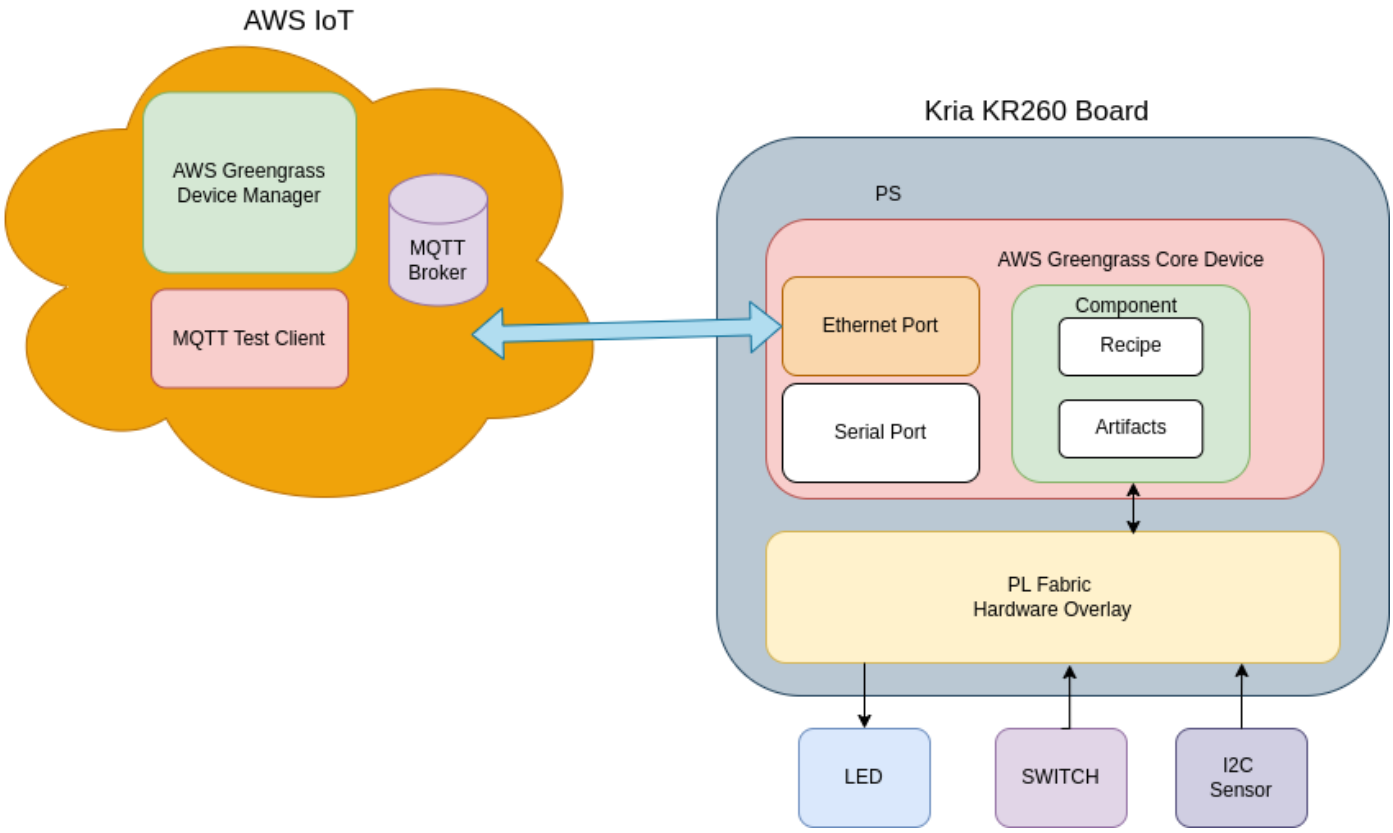
2023.12.20/v0.1 Sanam Shakya

# KR260 to AWS IoT Greengrass Architecture




This diagram shows the software and hardware architecture used in this tutorial. Kria KR260 board consists of 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.

# Creating Petalinux Project

## Create Petalinux from BSP

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

- Petalinux version 2022.2
- Added meta-aws layer for installing required dependencies required for running greengrass core device in KR260
- Rootfs packages : packagegroup-buildessential, git, libgpiod, libgpiod-dev, libgpiod-tools
- Enabled FPGA-manager

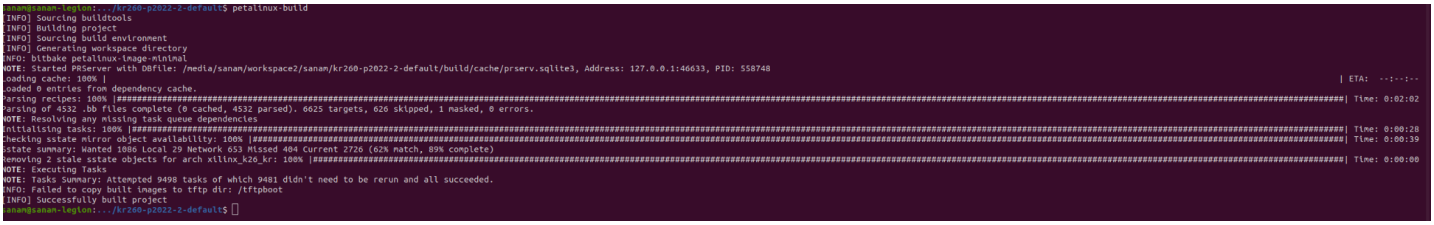
Run following commands to create petalinux project after sourcing the Petalinux 2022.2 environment:

```
source <Path to Petalinux 2022.2>/settings.sh
petalinux-create -t project -s xilinx-kr26-starterkit-aws-iot.bsp -n kr260-aws-iot
cd kr260-aws-iot
```

Here “kr260-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.




Next create the SD card image with the following commands:

```
petalinux-package --boot --u-boot --force
petalinux-package --wic --images-dir images/linux/ --bootfiles
"ramdisk.cpio.gz.u-boot,boot.scr,Image,system.dtb,system-zynqmp-sck-kr-g-revB.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 in KR260

After booting the previously used SD card into KR260.
Login to KR260 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 KR260 firmwares to KR260 using network tools like scp or manually copying the firmware files at /home/petalinux directory of SD card.

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.

```
xilinx-kr260-starterkit-20222:~$ 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,
                     kr260-i2c           XRT_FLAT                     kr260-i2c           XRT_FLAT             (0+0)                 -1
xilinx-kr260-starterkit-20222:~$
```

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
```

```
xilinx-kr260-starterkit-20222:~$ sudo xmutil loadapp kr260-i2c
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/pid
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/uid
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay0
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay1
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/afi0
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking0
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking1
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay2
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_gpio_0
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/misc_clk_0
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_0
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/misc_clk_1
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_1
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_0
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_1
Nov 19 09:35:53 xilinx-kr260-starterkit-20222 kernel: irq: type mismatch, failed to map hwirq-0 for interrupt-controller@80000000!
kr260-i2c: loaded to slot 0
xilinx-kr260-starterkit-20222:~$
```

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:

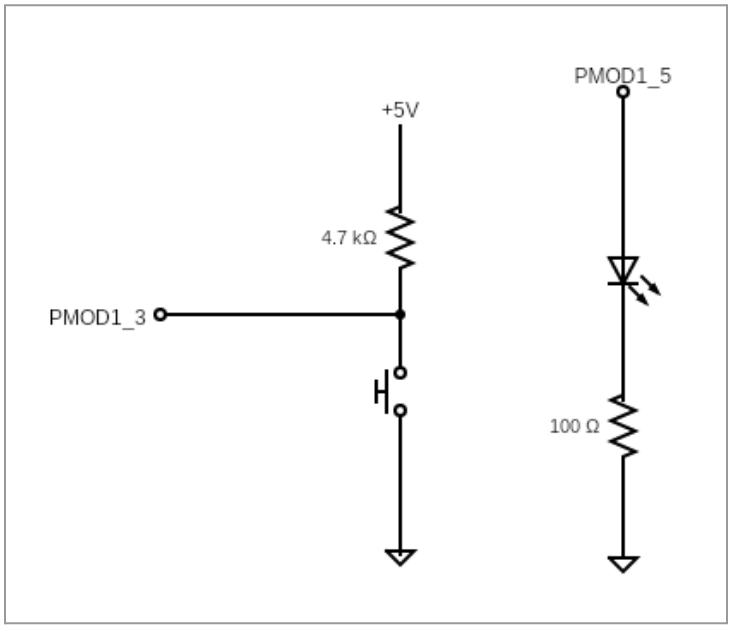
```
sudo pip3 install gpiod
```

Now we can check the available gpio using gpiod  applications:

Using `gpiodetect` to get availabe gpio:

```
xilinx-kr260-starterkit-20222:~$ sudo gpiodetect
gpiochip0 [firmware:zynqmp-firmware:gpio] (4 lines)
gpiochip1 [zynqmp_gpio] (174 lines)
gpiochip2 [slg7xl45106] (8 lines)
gpiochip3 [80010000.gpio] (6 lines)
xilinx-kr260-starterkit-20222:~$
```

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

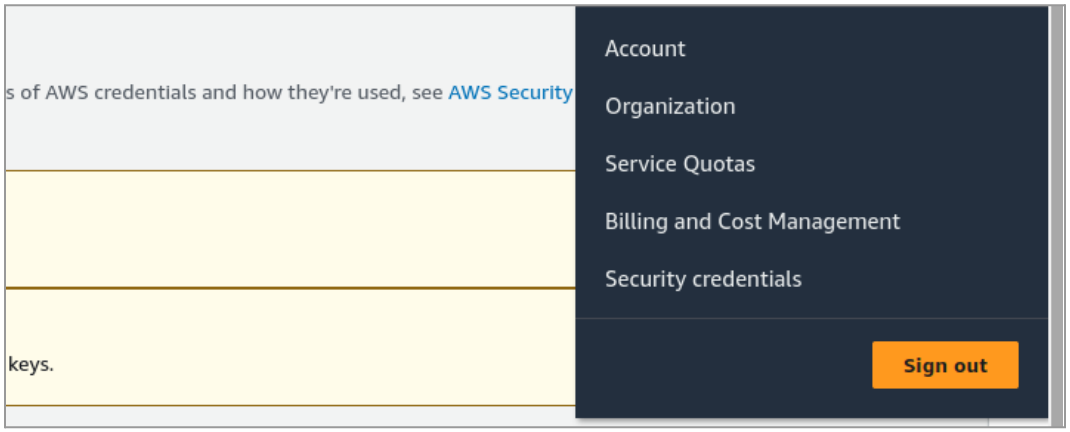
| | | | | | | |
|---|---|---|---|---|---|---|
| 11 | 9 | 7 | 5 | 3 | 1 | PMOD UPPER |
| 12 | 10 | 8 | 6 | 4 | 2 | PMOD LOWER |
| Vcc | GND | I/O | I/O | I/O | I/O | |

PMOD port numbering

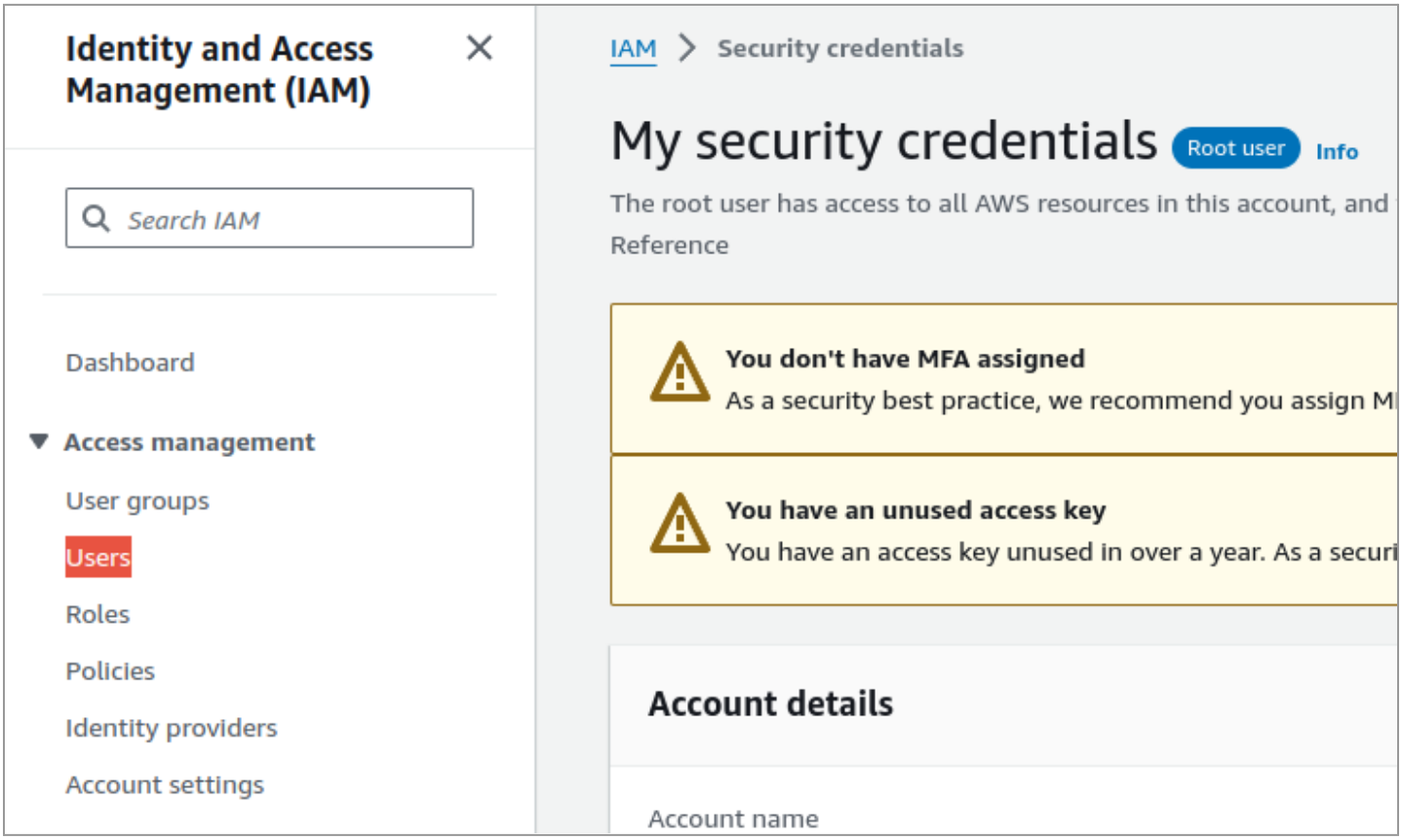
We will be using these gpios while creating component of Greengrass AWS core device in KR260.

# AWS IoT user creation

For an 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

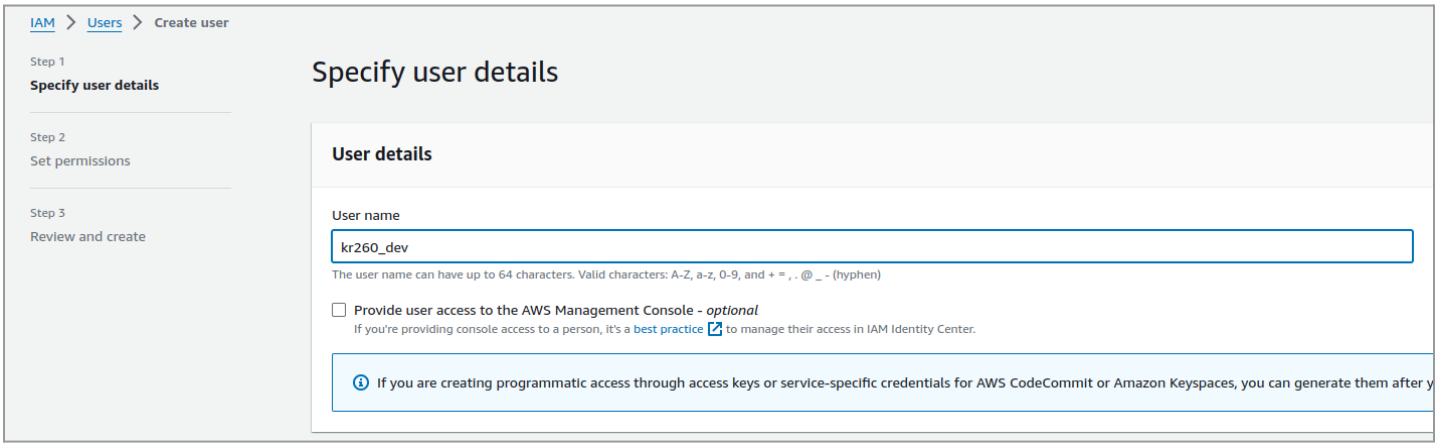



- Next Go to User management page by clicking at the User link at IAM sidebar.
This will list the available users.

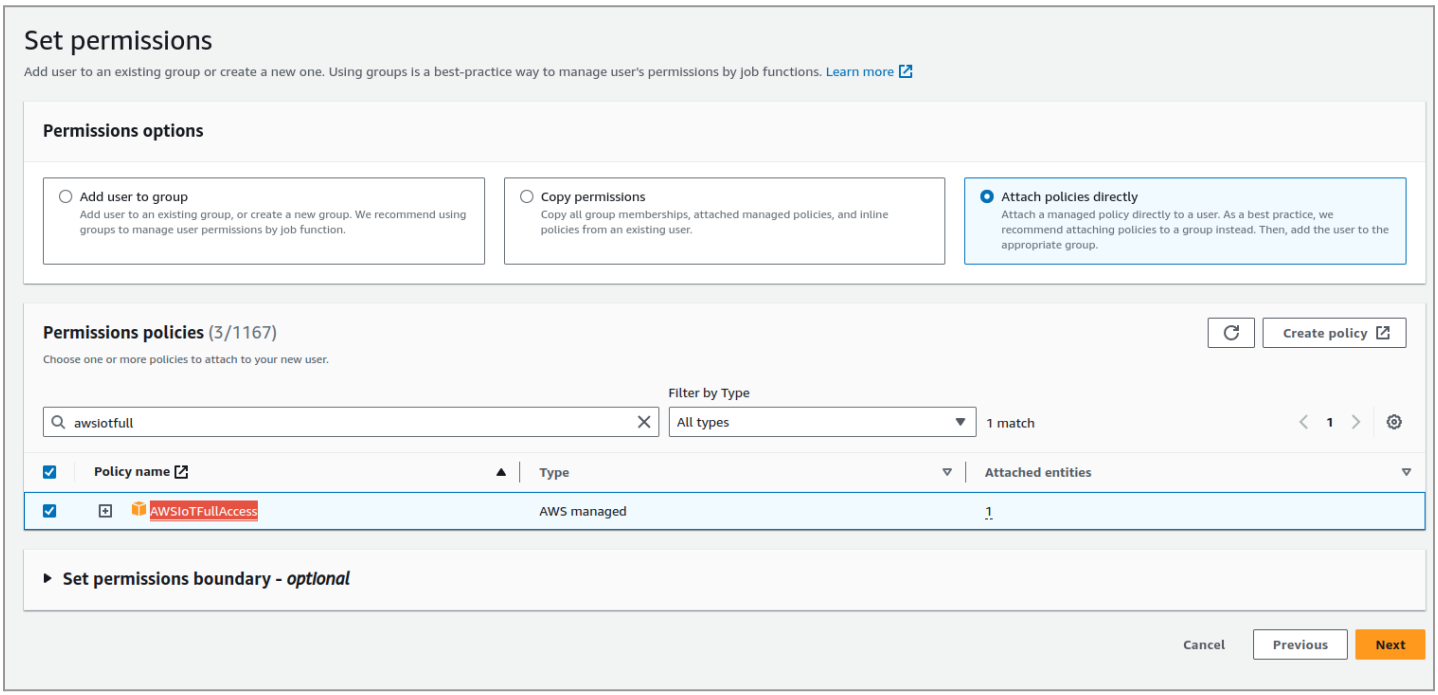



- Now create a new user for KR260 device by clicking the "Create User" button.

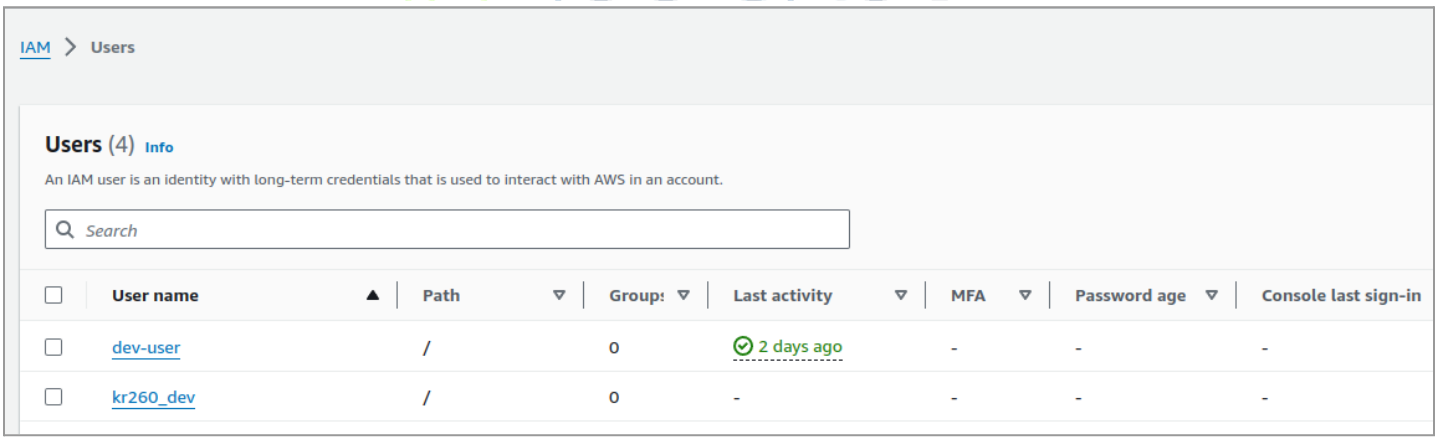

This will lead to step wise User creation forms. So fill the User details,
This will lead to step wise User creation forms.
So fill the User details, leave the console access unchecked as user does not have to access the AWS console through the web.

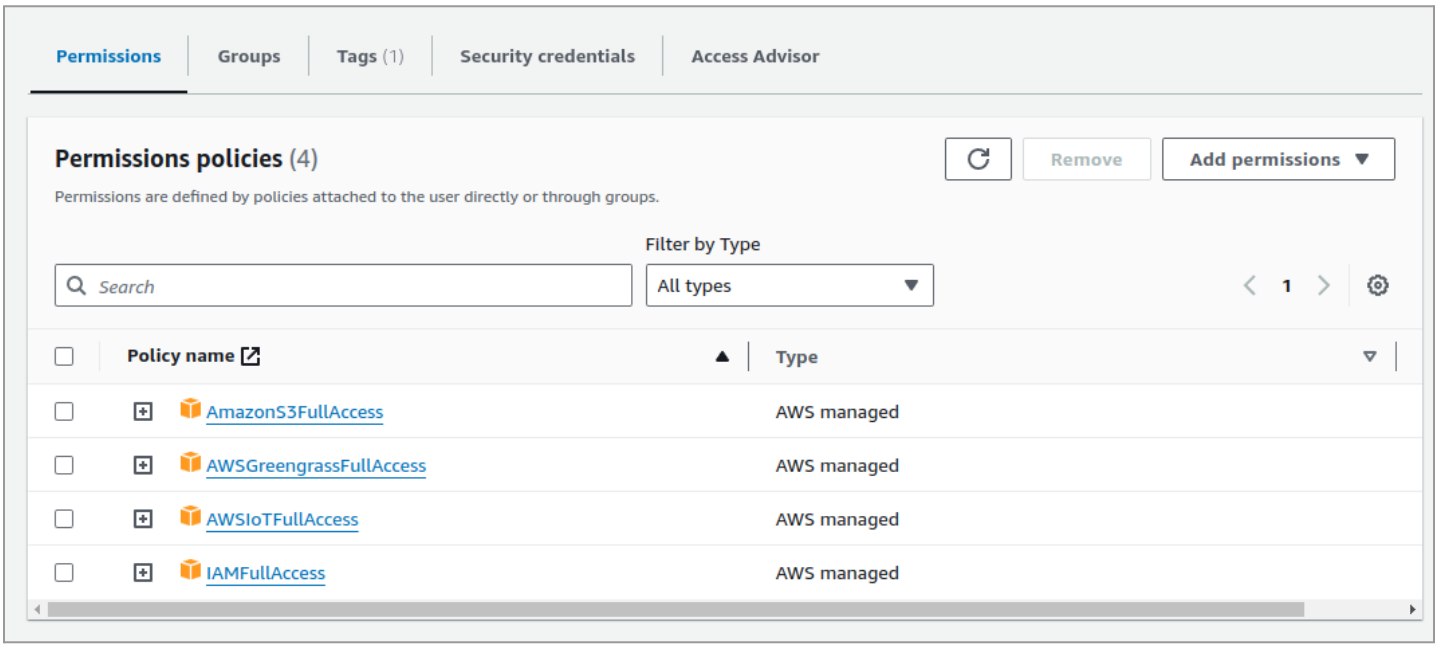



Next, update the Permissions options by attaching following policies:
- AWSGreengrassFullAccess
- IAMFullAccess
- AWSIoTFullAccess
- AmazonS3FullAccess



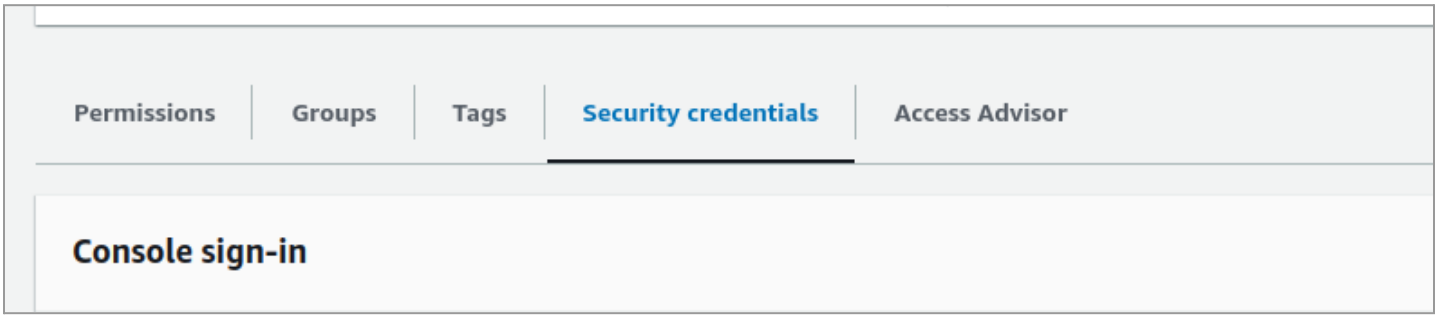

After finishing the above steps click "Create User" to finish the user creation.



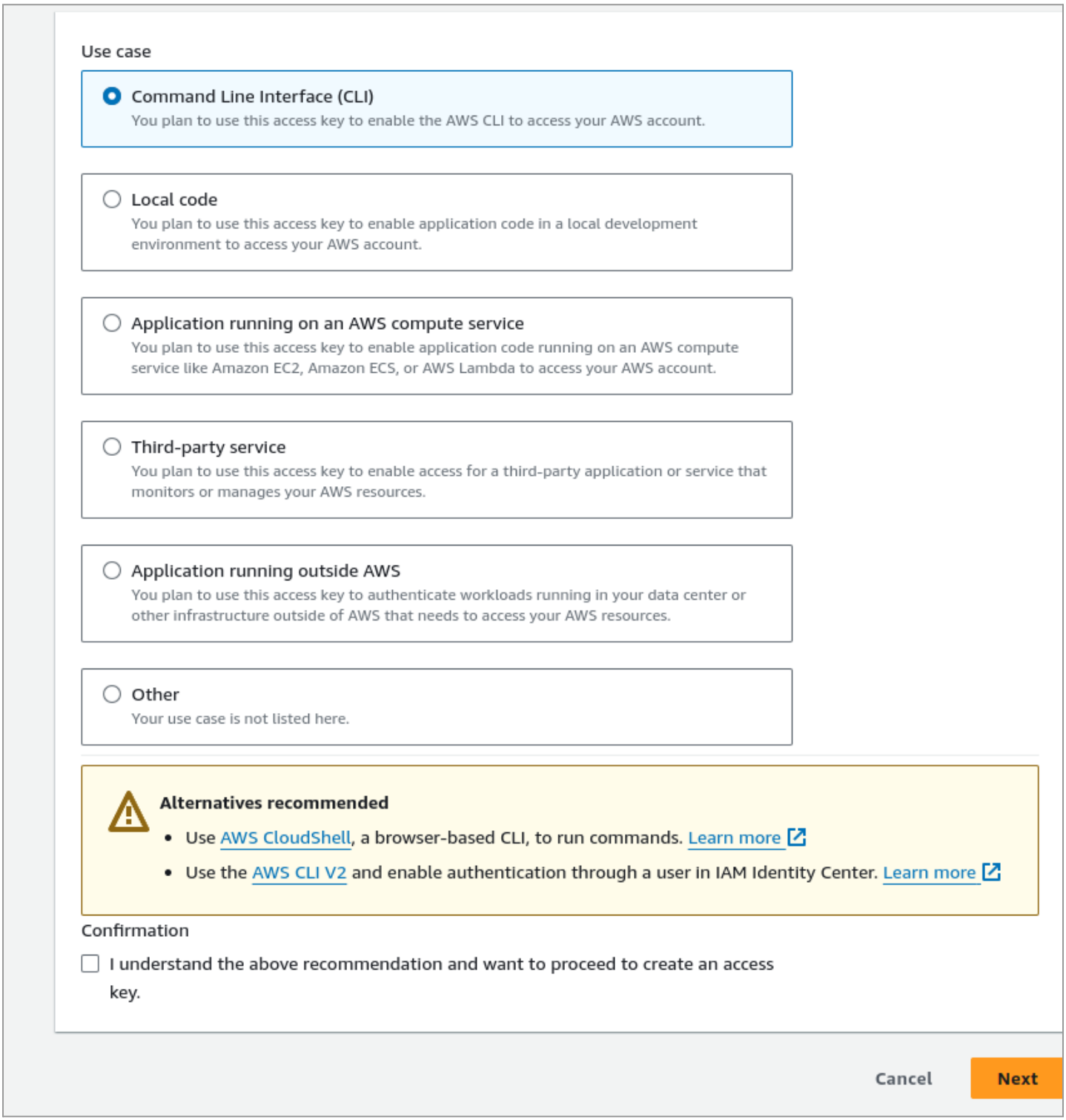



Next get the access token and access key for the user. For this open the user details by clicking on the user link in the above table.
And go to "Security credentials" for creating the Access Key for the user.





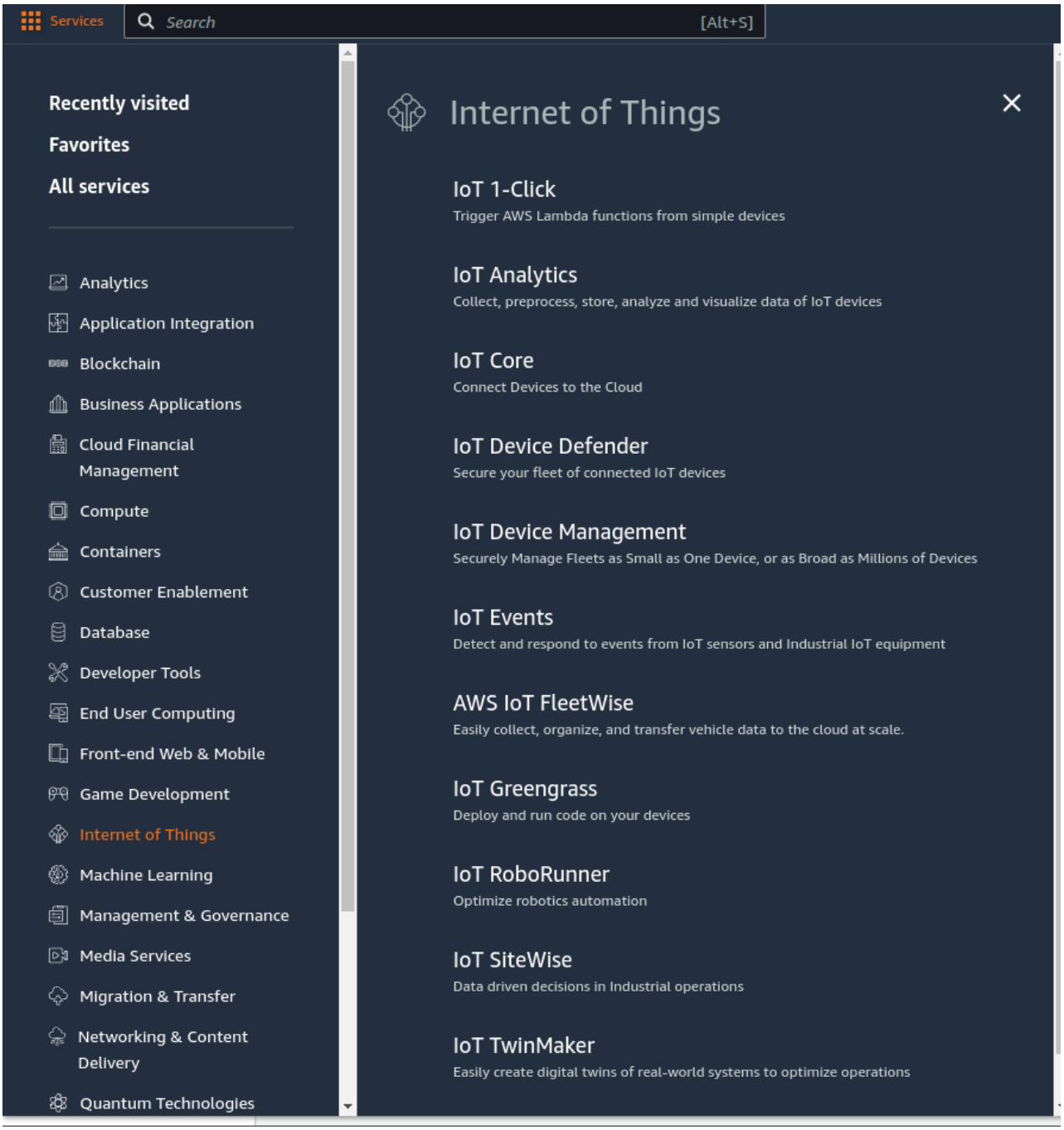

Select access key for command line based access control for user.

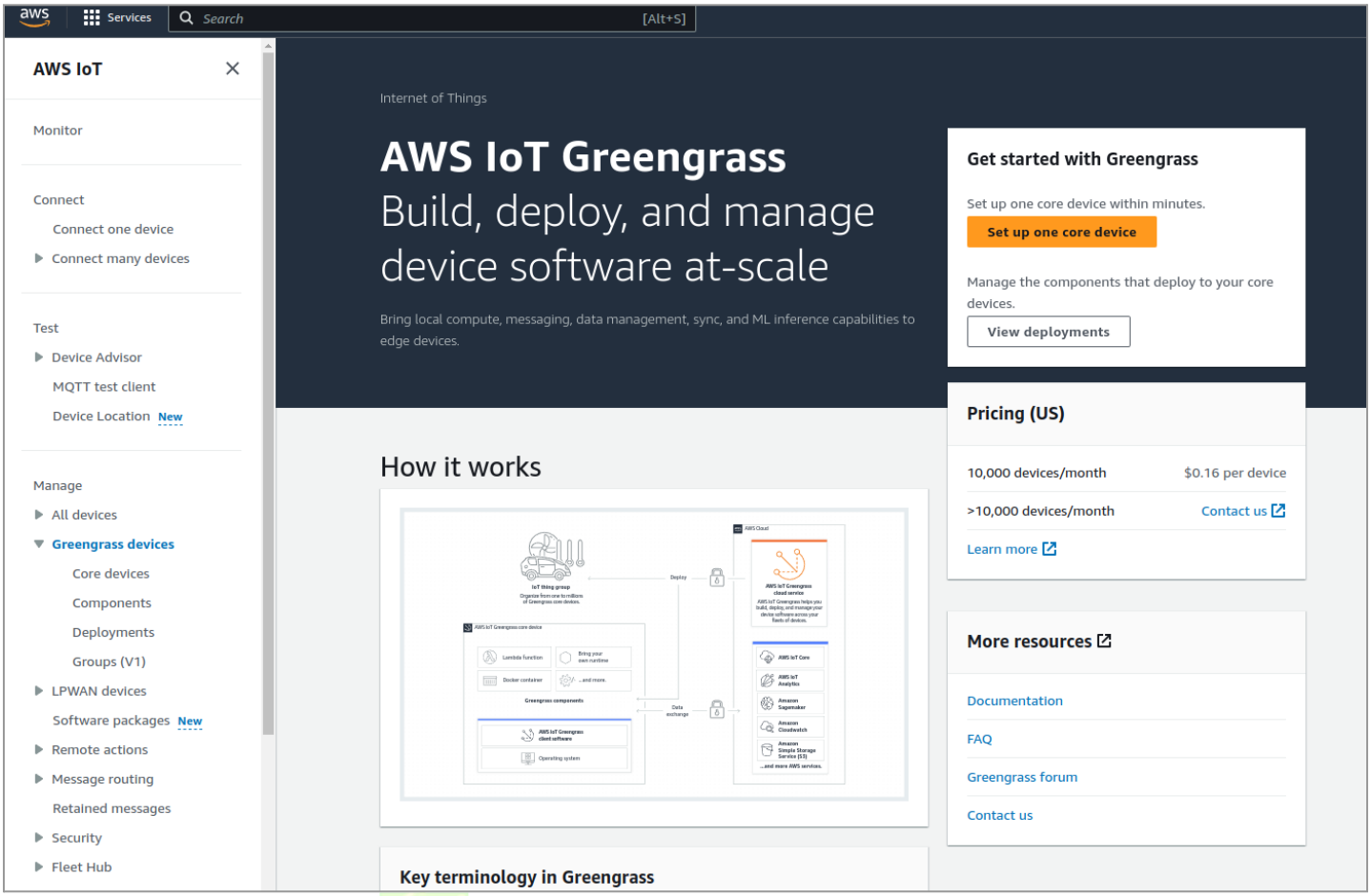



Next save the "Access Key" and "Secreat Access Key" . We will need this later while using greengrass CLI in KR260 console or downloading the csv file.

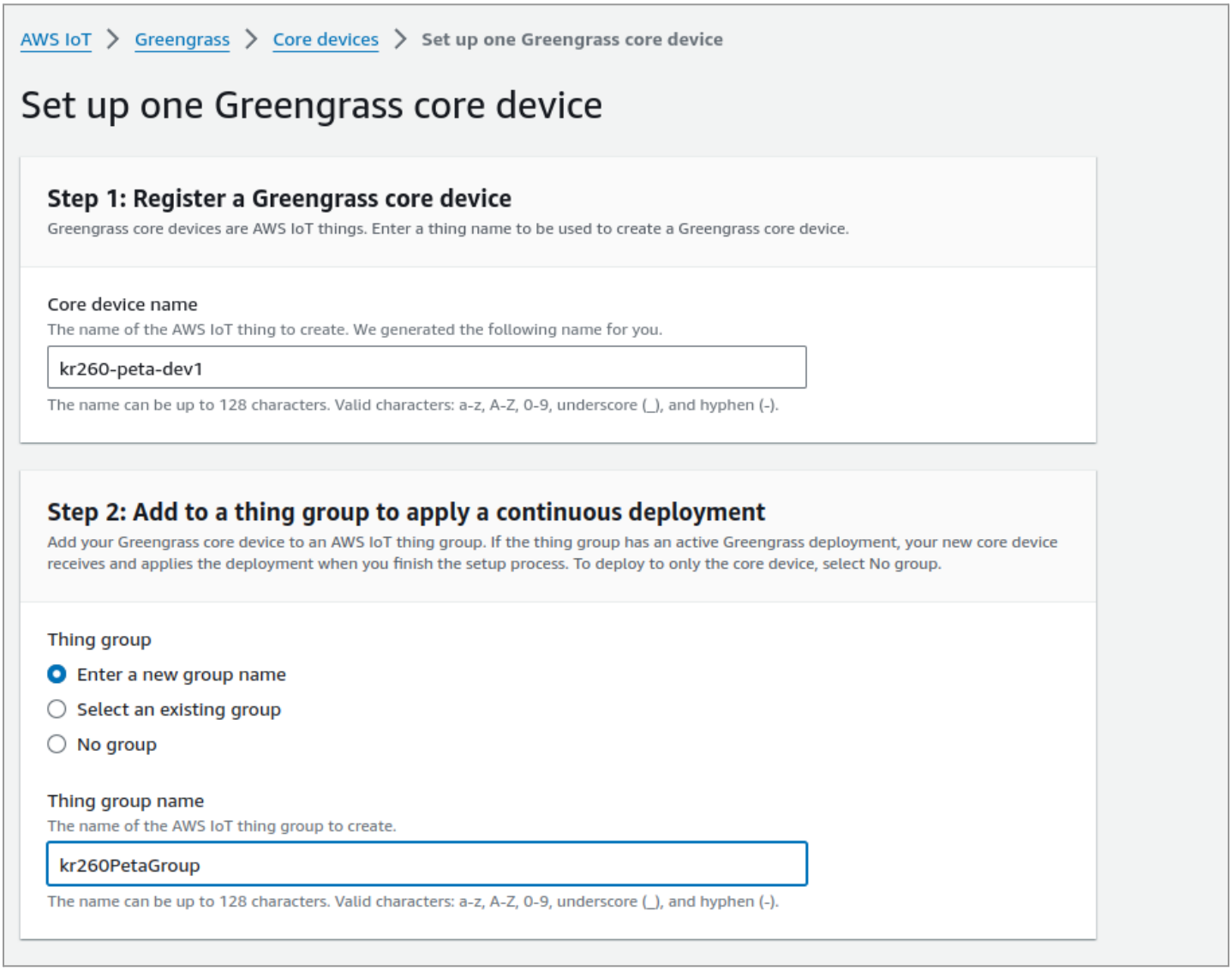
# Installing Greengrass CLI in KR260 board

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





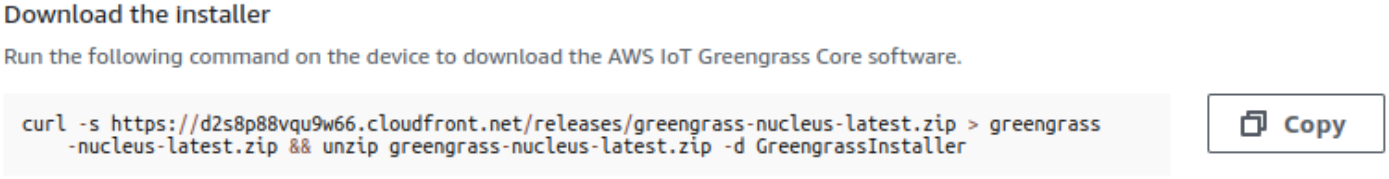

Now click on “Set up one core device” button
This will open the Greengrass core device setup page:
Here you can change the Core device name like `kr260-dev”




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>
```

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



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.

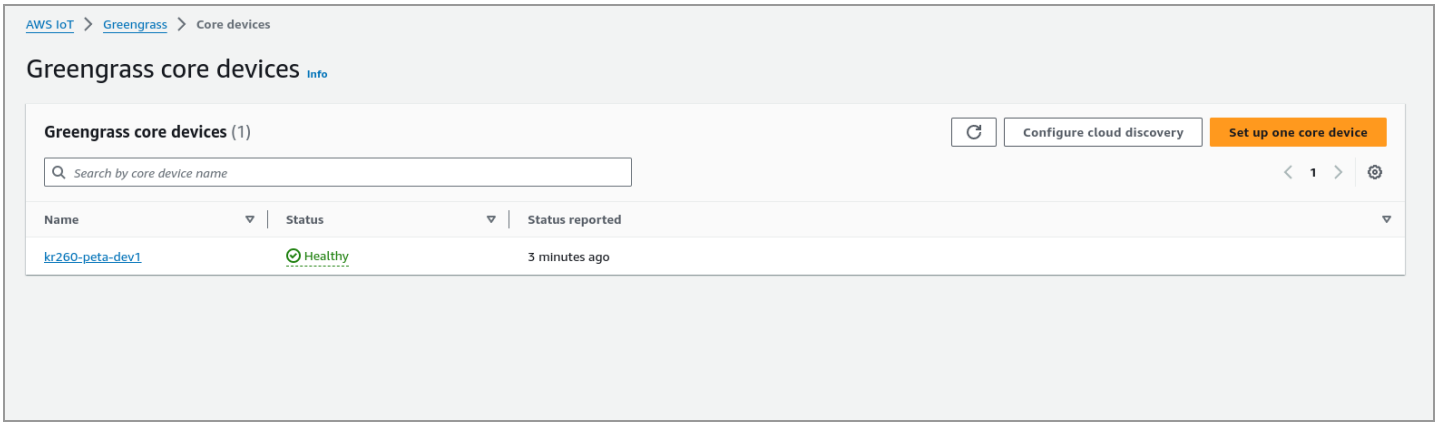
```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar
    --aws-region us-east-1 --thing-name kr260-peta-dev1 --thing-group-name kr260PetaGroup --component
    -default-user ggc_user:ggc_group --provision true --setup-system-service true --deploy-dev-tools
    true
```

Copy

Here is the console log after running above command:

```
xilinx-kr260-starterkit-20222:~$ sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengras
s.jar --aws-region us-east-1 --thing-name kr260-peta-dev1 --thing-group-name kr260PetaGroup --component-default-user ggc_user:g
gc_group --provision true --setup-system-service true --deploy-dev-tools true
Password:
Provisioning AWS IoT resources for the device with IoT Thing Name: [kr260-peta-dev1]...
Found IoT policy "GreengrassV2IoTThingPolicy", reusing it
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "kr260-peta-dev1"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [kr260-peta-dev1]!
Adding IoT Thing [kr260-peta-dev1] into Thing Group: [kr260PetaGroup]...
Successfully added Thing into Thing Group: [kr260PetaGroup]
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
xilinx-kr260-starterkit-20222:~$
```

Now in Greengrass set up page, one can view the Greengrass core devices and find above `kr260-dev` in the list.




In KR260 terminal one can get the device components by using `greengrass-cli`:

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

```
xilinx-kr260-starterkit-20222:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Password:
AWS libcrypto resolve: searching process and loaded modules
AWS libcrypto resolve: found static aws-lc HMAC symbols
AWS libcrypto resolve: found static aws-lc libcrypto 1.1.1 EVP_MD symbols
Dec 08, 2023 12:24:38 AM software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection$1 onConnectionSetup
INFO: Socket connection /greengrass/v2/ipc.socket:8033 to server result [AWS_ERROR_SUCCESS]
Dec 08, 2023 12:24:39 AM software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection$1 onProtocolMessage
INFO: Connection established with event stream RPC server
Components currently running in Greengrass:
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: UpdateSystemPolicyService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Nucleus
    Version: 2.4.0
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequ
8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.com","iotDataEndpo
":{},"mqtt":{"spooler":{}},"networkProxy":{"proxy":{}},"platformOverride":{},"runWithDefault":{"posixUser":"gg
Component Name: aws.greengrass.Cli
    Version: 2.4.0
    State: RUNNING
    Configuration: {"AuthorizedPosixGroups":null}
Component Name: FleetStatusService
    Version: null
    State: RUNNING
    Configuration: null
xilinx-kr260-starterkit-20222:~$
```

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:

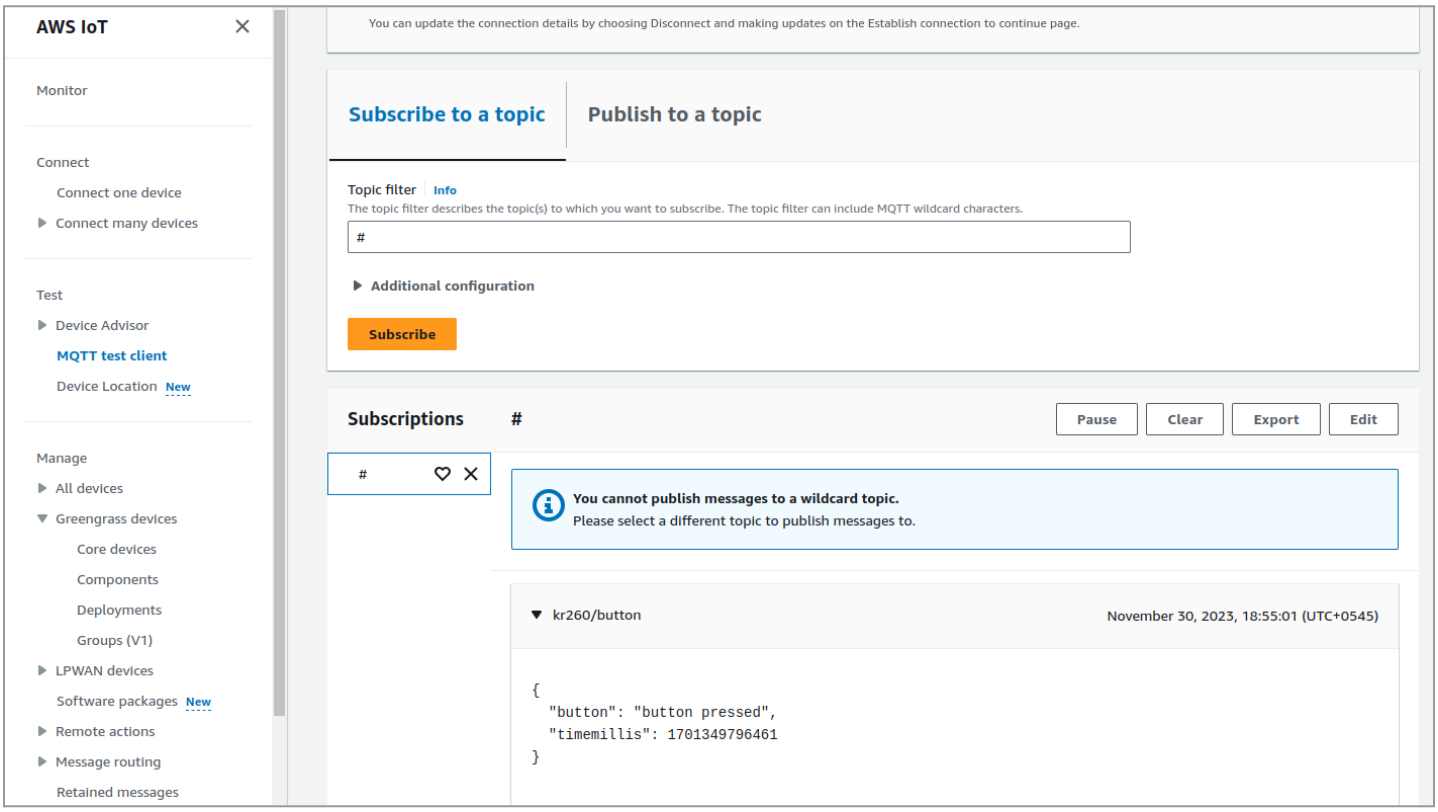
```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.mqtt=1.0.0"
```

```
xilinx-kr260-starterkit-20222:~$ sudo /greengrass/v2/bin/greengrass-cli deployment create --recipeDir ~/components/r
ecipe --artifactDir ~/components/artifacts --merge "com.example.mqtt=1.0.0"
AWS libcrypto resolve: searching process and loaded modules
AWS libcrypto resolve: found static aws-lc HMAC symbols
AWS libcrypto resolve: found static aws-lc libcrypto 1.1.1 EVP_MD symbols
Dec 08, 2023 1:23:18 AM software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection$1 onConnectionSetup
INFO: Socket connection /greengrass/v2/ipc.socket:8033 to server result [AWS_ERROR_SUCCESS]
Dec 08, 2023 1:23:18 AM software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection$1 onProtocolMessage
INFO: Connection established with event stream RPC server
Local deployment submitted! Deployment Id: 89867086-fe40-45fa-b994-3d4d65036ae5
```

Now check the installed component is in “running state”

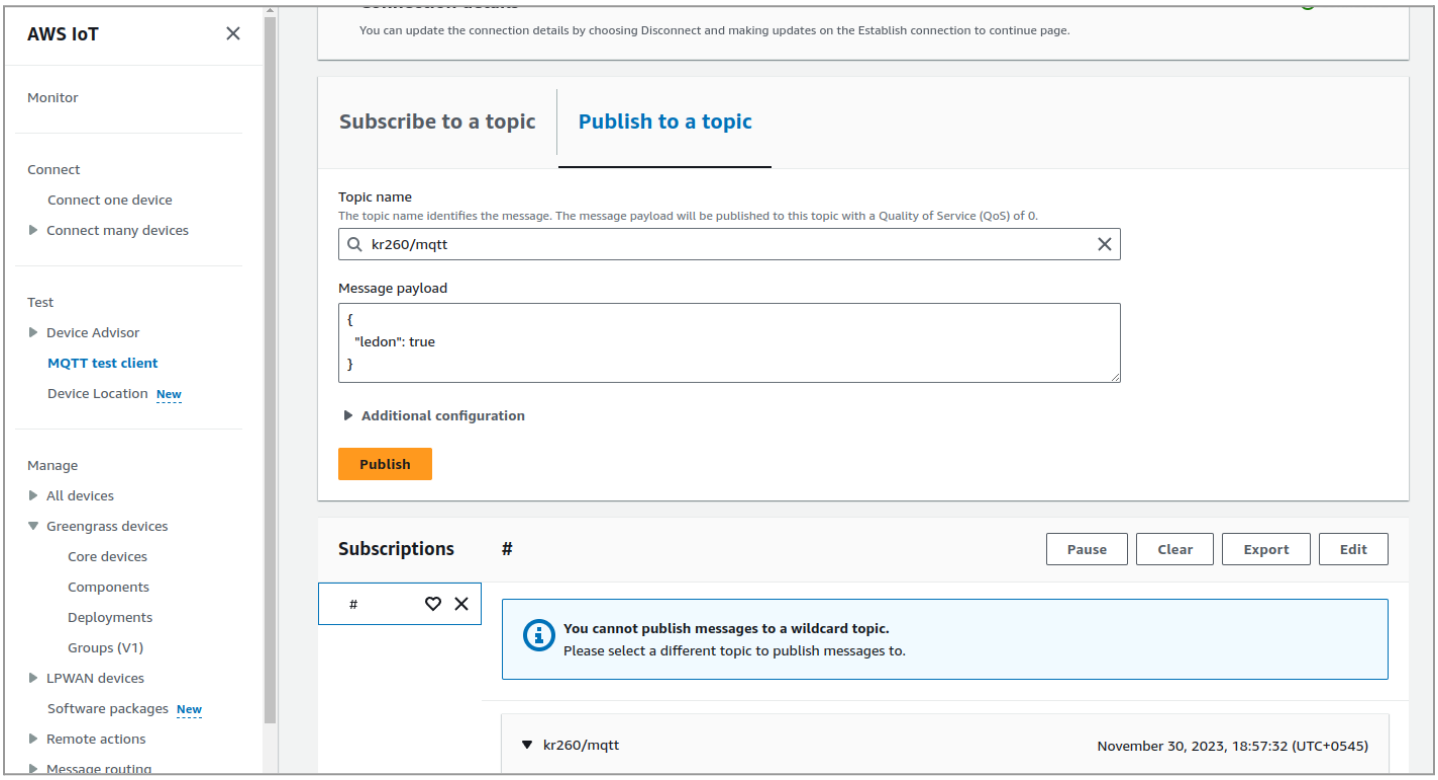
```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: aws.greengrass.Nucleus
    Version: 2.12.0
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequencySeconds":"15","envStage":"prod","fipsMode":"false","fleetStatus":{"periodicStatusPublishIntervalSeconds":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":{},"runWithDefault":{"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":"Allows 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 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.




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