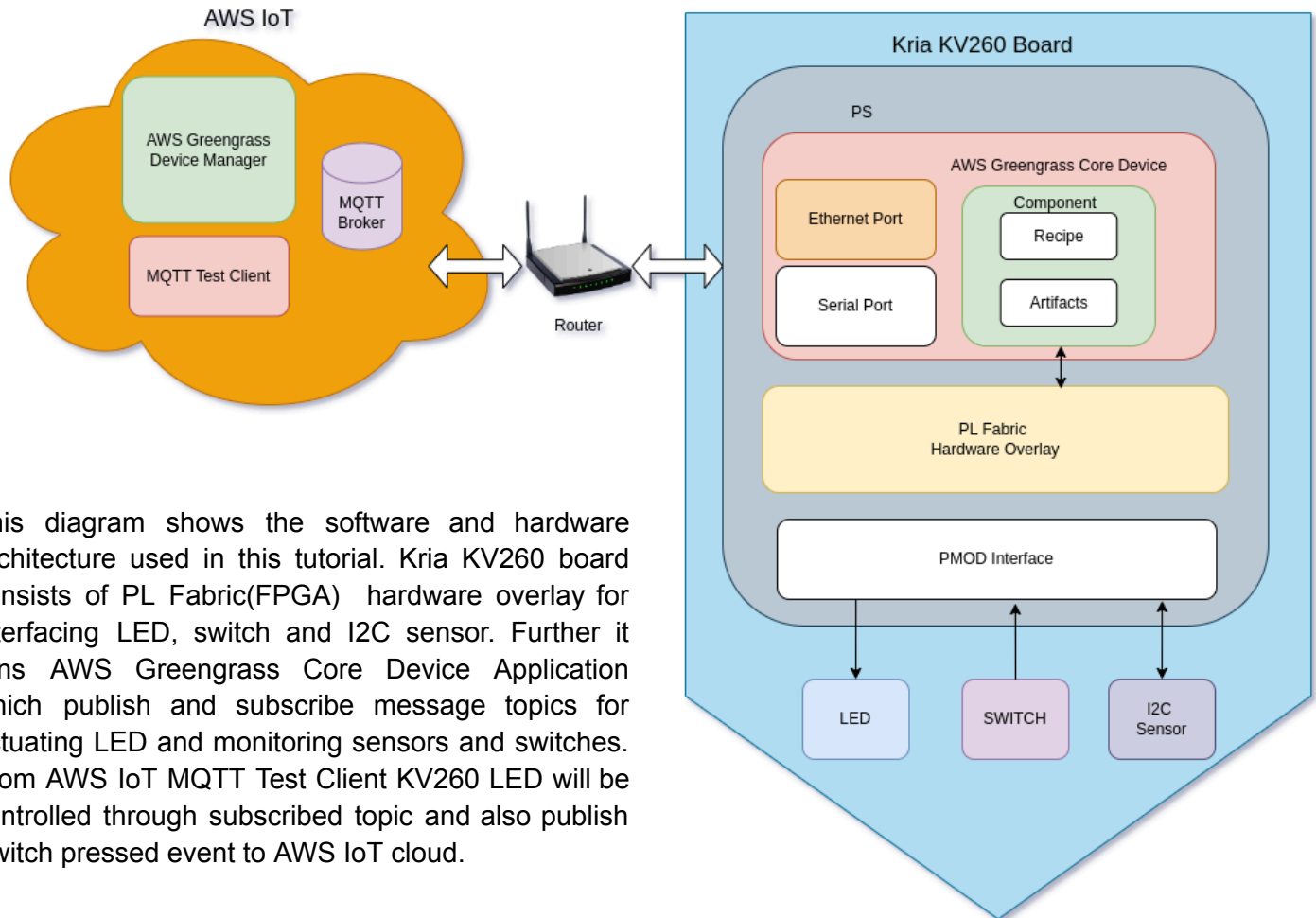# KV260 to AWS IoT Greengrass Architecture

This diagram shows the software and hardware architecture used in this tutorial. Kria KV260 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 KV260 LED will be controlled through subscribed topic and also publish Switch pressed event to AWS IoT cloud.
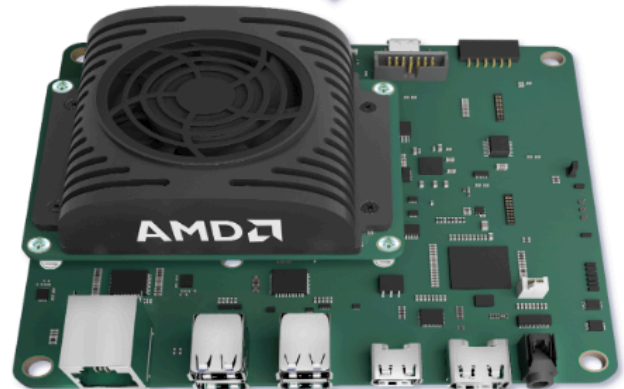
Creating Petalinux Project

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 KV260
- Rootfs packages : packagegroup-buildessential, git, libgpiod, libgpiod-dev, libgpiod-tools
- Enabled FPGA-manager

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

```
source <Path to Petalinux 2022.2>/settings.sh
petalinux-create -t project -s xilinx-kv260-starterkit-aws-iot.bsp -n kv260-aws-iot
cd kv260-aws-iot
```

Here "kv260-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:

```
package --wic --images-dir images/linux/ --bootfiles
"ramdisk.cpio.gz.u-boot,boot.scr,Image,system.dtb,system-zynqmp-sck-kv-g-revB.dtb"
--disk-name "mmcblk1"
```

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 KV260.
Login to KV260 serial terminal using login name: petalinux
For the first login one has to update the new password.

Next copy the KV260 firmwares to KV260 using network tools like scp or manually copying the firmware files at /home/petalinux directory of SD card.

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

Copy these file to the KV260 board. For firmware to be loaded using xmutil (FPGA manager), one has to copy these file at "/lib/firmware/xilinx".
For this, create the folder at "kv260-gpio-i2c" at "/lib/firmware/xilinx" and copy the files in "kv260-gpio-i2c" folder.

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

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

```
xilinx-kv260-starterkit-20231:~$ sudo xmutil listapps
Password:
                  Accelerator        Accel_type                    Base      Base_type      #slots(PL+AIE)      Active_slot

              k26-starter-kits        XRT_FLAT          k26-starter-kits       XRT_FLAT              (0+0)               0,
               kv260-gpio-i2c        XRT_FLAT            kv260-gpio-i2c        XRT_FLAT              (0+0)              -1
xilinx-kv260-starterkit-20231:~$
```

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

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

```
xilinx-kv260-starterkit-20231:~$ sudo xmutil loadapp kv260-gpio-i2c
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/afi0
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking0
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking1
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_0
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_1
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_gpio_0
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_0
Dec 27 23:06:57 xilinx-kv260-starterkit-20231 kernel: gpio gpiochip2: (80020000.gpio): not an immutable chip, please consider fixing it!
kv260-gpio-i2c: loaded to slot 0
xilinx-kv260-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 availabe gpio:

```
xilinx-kv260-starterkit-20231:~$ sudo gpiodetect
Password:
gpiochip0 [firmware:zynqmp-firmware:gpio] (4 lines)
gpiochip1 [zynqmp_gpio] (174 lines)
gpiochip2 [80020000.gpio] (4 lines)
xilinx-kv260-starterkit-20231:~$
```

Here `gpiochip2` 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-> 5 - gpiochip2 line 0
PMOD1-> 7 - gpiochip2 line 1
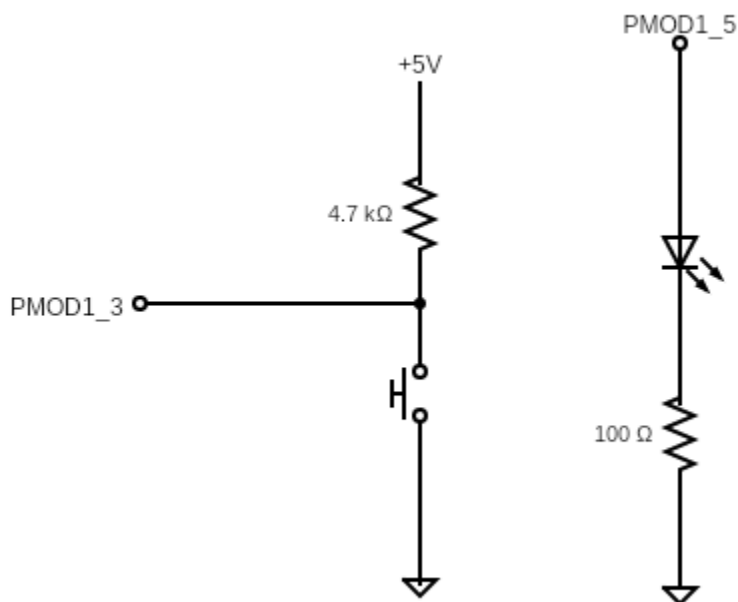
Schematic for LED and Switch Connection

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

PMOD port numbering

We will be using these gpios while creating component of Greengrass AWS core device in KV260.
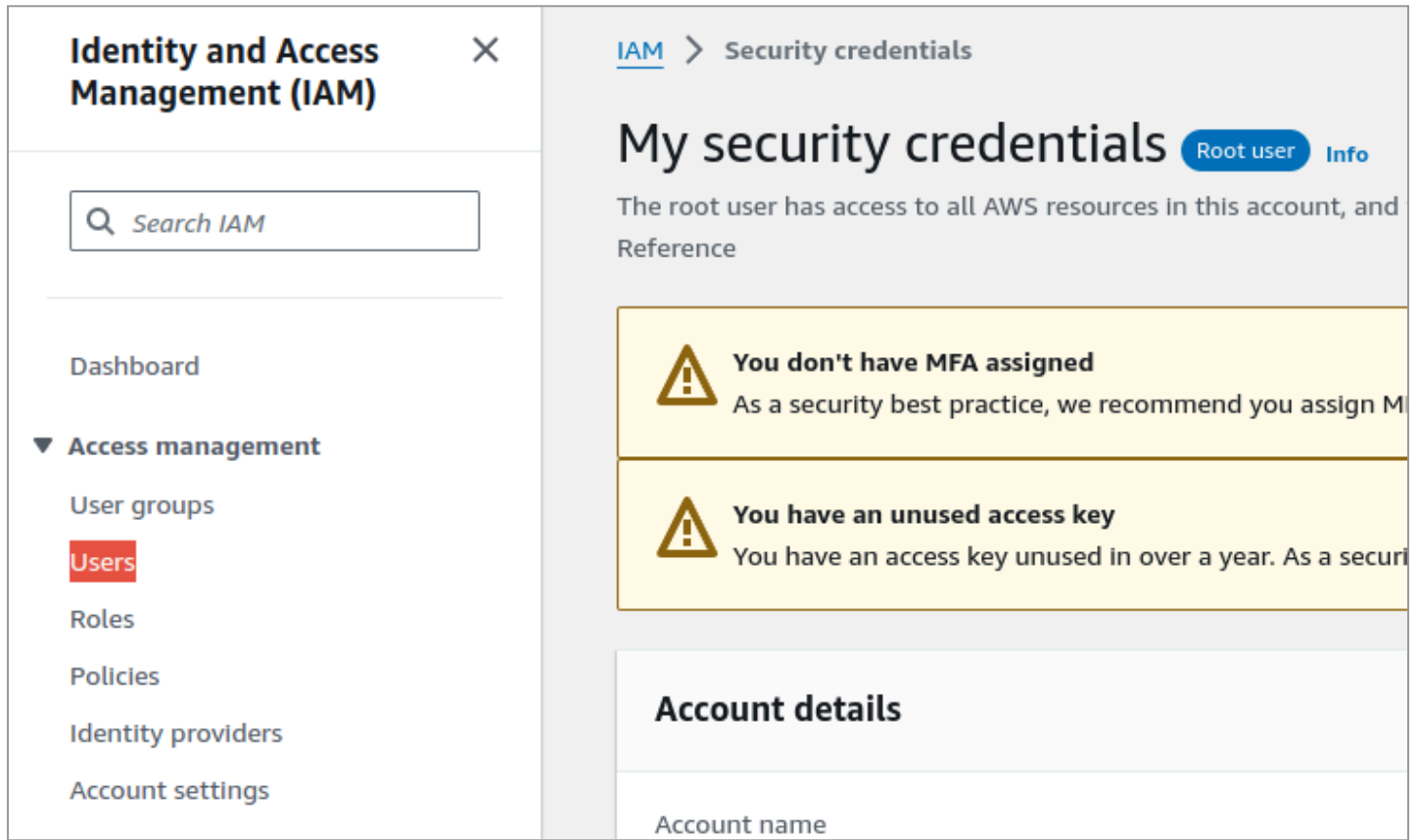
AWS IoT user creation

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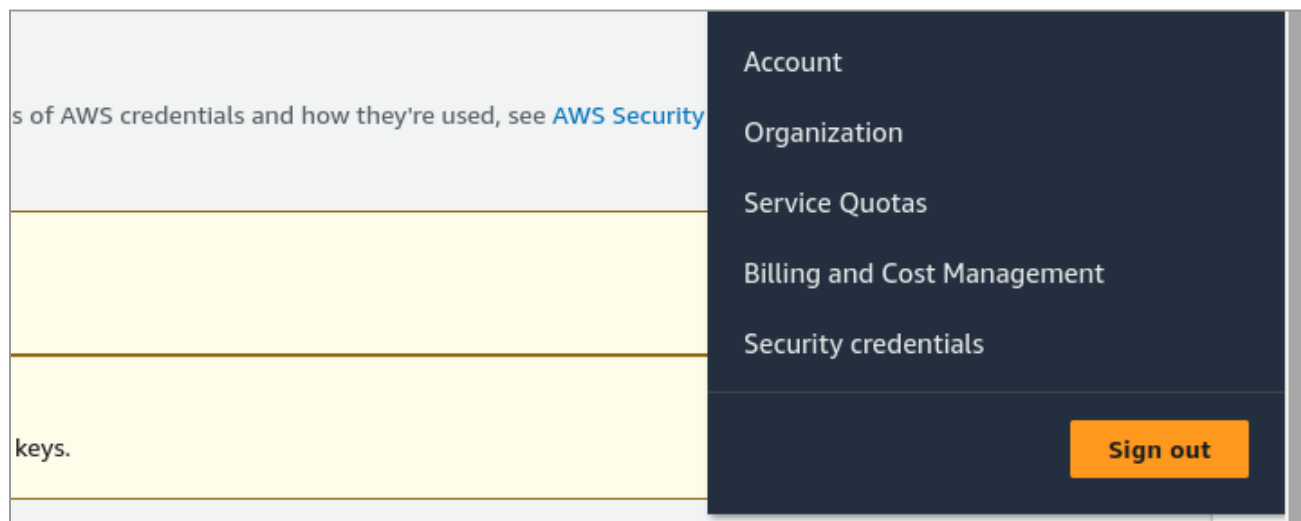
- Next Go to User management page by clicking at the User link at IAM sidebar.
This will list the available users.
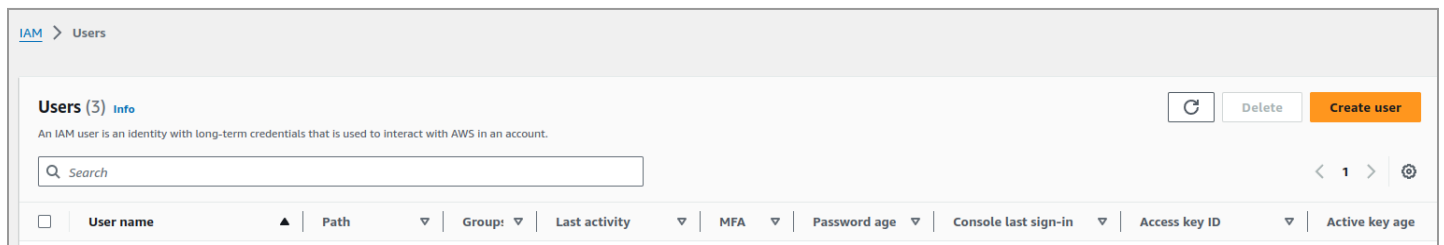
- Now create a new user for KV260 device by clicking the "Create User" button.

This will lead to step wise User creation forms. So fill the User details,

This will lead to step wise User creation forms.

So fill the User details, leave the console access unchecked as user does not have to access the AWS console through web.

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

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.

| Permissions | Groups | Tags | **Security credentials** | Access Advisor |

Console sign-in
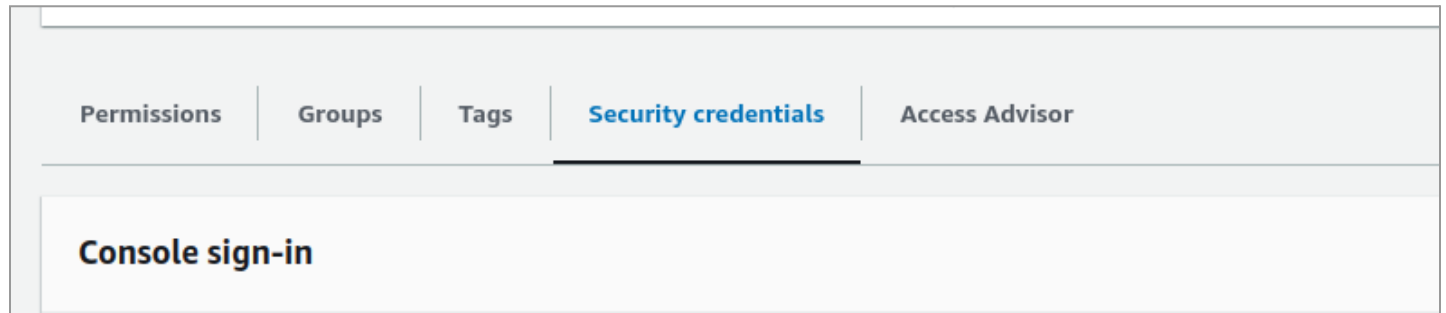
Select access key for command line based access control for user.

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.
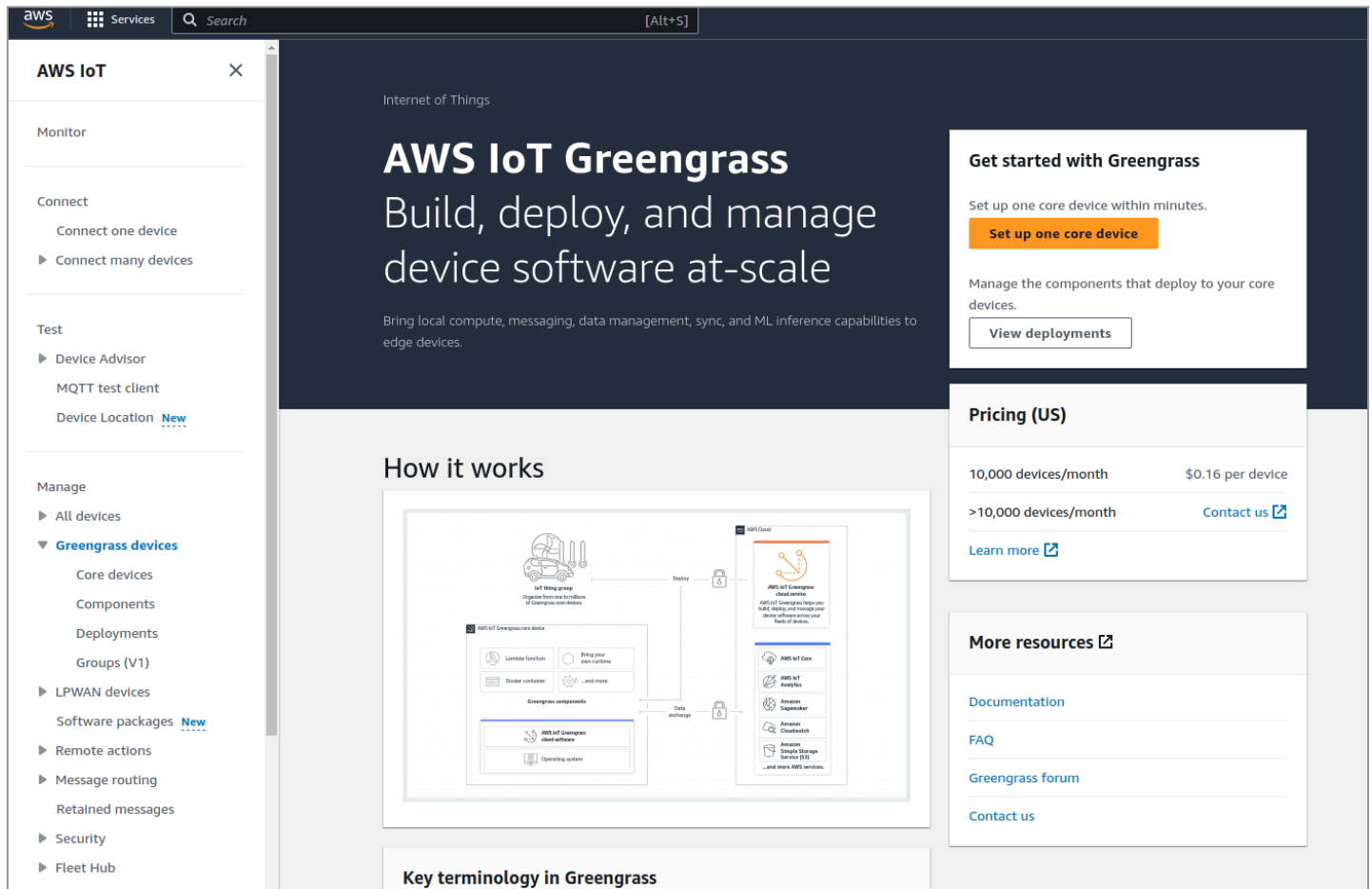
Cancel   **Next**

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

Installing Greengrass CLI in KV260 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 `kv260-peta-dev1"

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

Set up one Greengrass core device

Step 1: Register a Greengrass core device

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

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

```
kv260-peta-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.

```
kv260PetaGroup
```

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

Now in KV260 terminal console run following commands and scripts:

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

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.

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar
    --aws-region us-east-1 --thing-name kv260-peta-dev1 --thing-group-name kv260PetaGroup --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-kv260-starterkit-20231:~$ sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar --aws-region us-east-1 --thing-name kv260-peta-dev1
c_group --provision true --setup-system-service true --deploy-dev-tools true
Provisioning AWS IoT resources for the device with IoT Thing Name: [kv260-peta-dev1]...
Found IoT policy "GreengrassV2IoTThingPolicy", reusing it
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "kv260-peta-dev1"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [kv260-peta-dev1]!
Adding IoT Thing [kv260-peta-dev1] into Thing Group: [kv260PetaGroup]...
IoT Thing Group "kv260PetaGroup" already existed, reusing it
Successfully added Thing into Thing Group: [kv260PetaGroup]
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!
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
xilinx-kv260-starterkit-20231:~$ ▮
```

Now in Greengrass set up page, one can view the Greengrass core devices and find above `kv260-peta-dev1` in the list.

AWS IoT > Greengrass > Core devices

Greengrass core devices Info

Greengrass core devices (6)			↻ Configure cloud discovery Set up one core device
🔍 Search by core device name			‹ 1 › ⚙

Name ▽	Status ▽	Status reported	▽
kr260-dev1	⊘ Healthy	13 days ago	
kr260-peta-dev1	⊘ Healthy	13 days ago	
kd240-ubuntu-dev1	⊘ Healthy	21 hours ago	
kv260-peta-dev1	⊘ Healthy	1 minute ago	
kr260-ubuntu-dev1	⊘ Healthy	10 days ago	
kd240-dev2	⊗ Unhealthy	2 days ago	

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

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

```
xilinx-kv260-starterkit-20231:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Nucleus
    Version: 2.12.1
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFreq
t":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4wpvxg.credentials.iot.us-e
ions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"networkProxy":{"proxy":{}},"platformOverride":{}
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}
xilinx-kv260-starterkit-20231:~$
```

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

Installing the component

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

recipe
- com.example.mqtt-1.0.0.json

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

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

```
xilinx-kv260-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: 3a5d208f-ce9a-412c-ae67-dc0838abf412
xilinx-kv260-starterkit-20231:~$
```
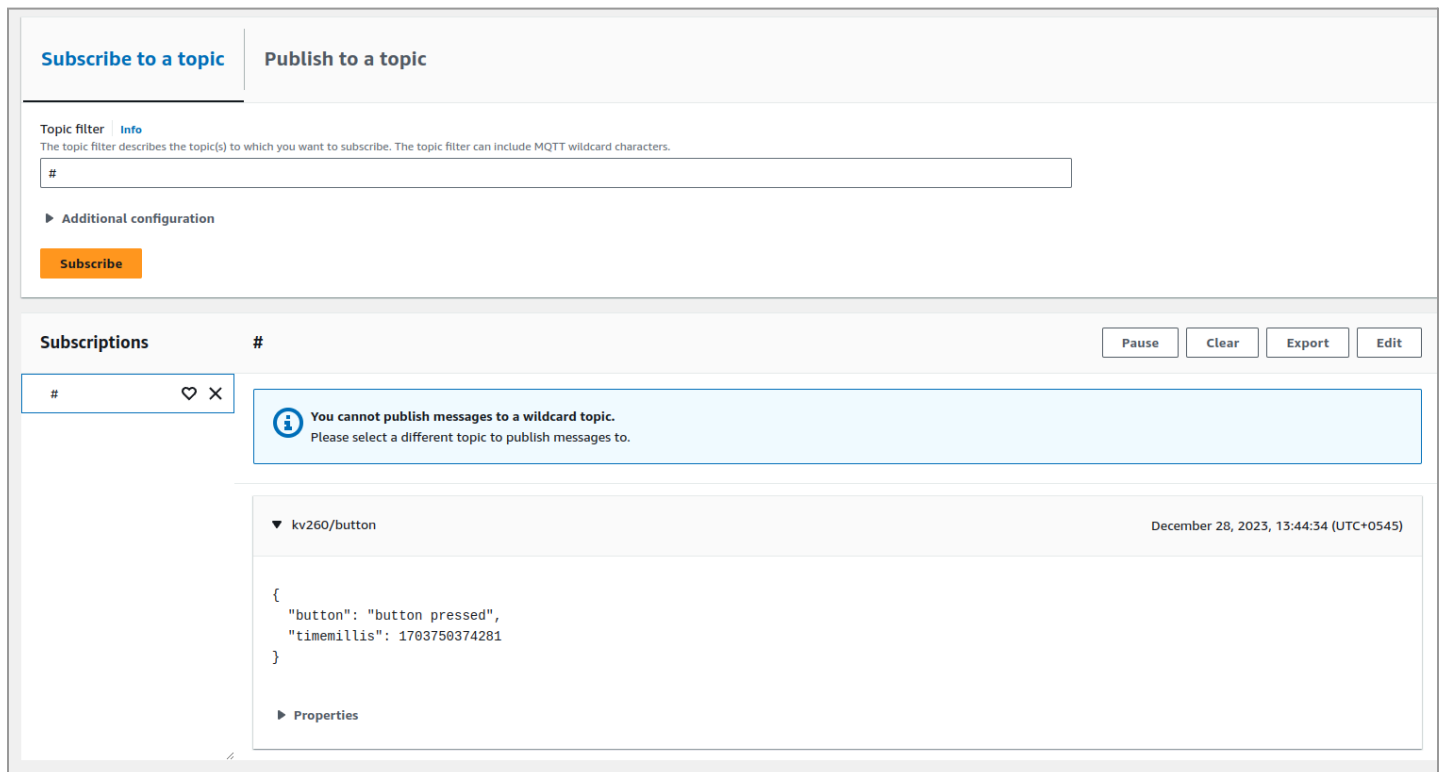
Now check the installed component is in "running state"

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

```
xilinx-kv260-starterkit-20231:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Nucleus
    Version: 2.12.1
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequencySeconds":"15","envStage":"pr
t":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"cluwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.com","iotDataEndp
ions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"networkProxy":{"proxy":{}},"platformOverride":{},"runWithDefault":{"posixShell":"
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#PublishToIo
ces":["kv260/mqtt","kv260/button"]}}},"message":"hello"}
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}
xilinx-kv260-starterkit-20231:~$
```

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

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

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

```
#
```

▶ Additional configuration

Subscribe

Subscriptions   **#**   Pause Clear Export Edit

```
#          ♡ ✕
```

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

▼ kv260/button   December 28, 2023, 13:44:34 (UTC+0545)

```
{
    "button": "button pressed",
    "timemillis": 1703750374281
}
```

▶ Properties

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

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

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

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

🔍 kv260/mqtt   ✕

Message payload

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

▶ Additional configuration

Publish

Subscriptions   #   Pause Clear Export Edit

♡ ✕

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

▼ kv260/mqtt   December 28, 2023, 13:46:08 (UTC+0545)

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

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

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

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

🔍 kv260/mqtt   ✕

Message payload

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

▶ Additional configuration

Publish

Subscriptions   #   Pause Clear Export Edit

♡ ✕

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

▼ kv260/mqtt   December 28, 2023, 13:45:38 (UTC+0545)

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