Wazee Group
Software Construction & Renovation

Taking the Pi for a Walk Through the AWS Greengrass

Jim Ladd

Wazee Group, Inc.

June 26, 2019

Contents

Introduction

The Raspberry Pi was originally intended to be an affordable, educational platform. First widely released in 2012, the Pi has wildly exceeded expectations with over 23 million units sold.

Back in 2012, my team at Wazee Group presented me with a Raspberry Pi as a gift for the winter holiday season. The Pi was a much-appreciated gift as it guided me back to my programming roots with single board computers. I was very impressed (and still am) with the features of the Pi and the extremely low cost. It truly must be the best value in the history of computers.

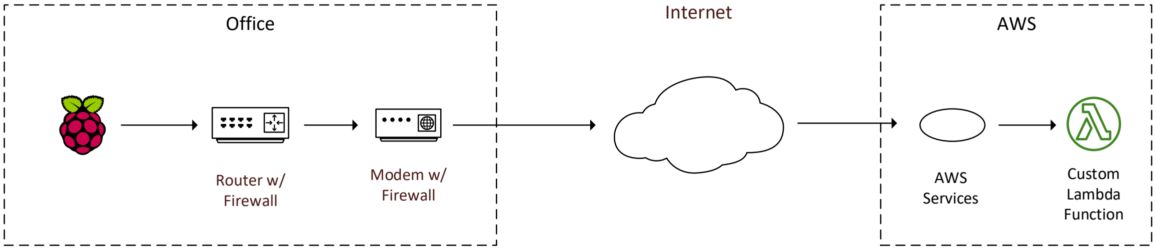
Not only has it become a stable in the computer education domain, the Pi is one of the most popular platforms for Internet of Things (IoT) solutions. A client of Wazee Group, Patrick Conroy, used the Pi to host a micro-service that automated the operations of a packaging machine. After thousands of hours of operation and millions of packages later, the Pi has performed flawlessly.

Unfortunately, not all of the Pi implementations that I have seen have been as impressive as Patrick's. Luckily, none of these are in mission critical situations. To assist others who are considering using the Pi as an IoT platform, I wanted to provide a consolidated step-by-step guide that configures a blank Raspberry Pi to send a message to a service within the AWS cloud in a secure, reliable, and widely acceptable fashion.

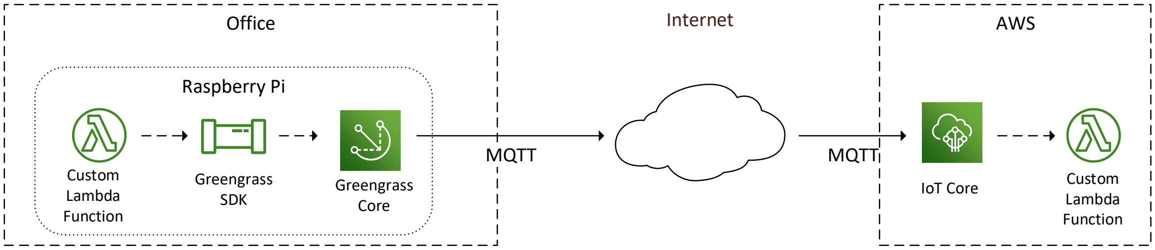
This guide intentionally relies heavily on the AWS IoT examples. I wanted to take the user to the first usable milestone in the AWS examples. The result of the guide will provide a seamless starting place for the additional AWS examples.

Overview

My mission was simple…I wanted to send a message from a Raspberry Pi to a Lambda function hosted in the AWS cloud. The environment is both typical and straightforward. The Pi would be sitting behind a router with all of the inbound ports closed. The router, in turn, would be behind a cable modem with all of the inbound ports closed. A diagram of the environment is shown below:



After researching the options provided by AWS, I decided to use their IoT Greengrass service. My solution could be defined in the following diagram. A Python script would be deployed as a Lambda function along with the Greengrass SDK. This function will publish a message through the SDK to the Greengrass Core software. The Core service will send the message to the AWS IoT Core service via the MQTT messaging framework. Once the message is received, a Lambda function inside the AWS cloud would be executed.



MQTT

One of the cornerstones of the Greengrass service is the MQTT messaging system. MQTT (i.e. MQ Telemetry Transport) is a publish/subscribe communication protocol that is widely used in the Internet of Things domain. MQTT was invented by Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom in 1999.

The reason that I like MQTT is that is allows secure, bi-directional communication based on connections made with the MQTT broker. This means that I do not have to alter my firewall settings or open ports on my modem and router.

In the Greengrass implementation, the MQTT is encapsulated within the service and remains largely unnoticed.

Next Steps

To realized the solution, this document covers the following steps:

1. Describe the hardware and present the associate costs.
2. Install the Raspbian operating system on the Pi.
3. Configure the Raspbian OS.
4. Secure the Pi for development and testing.
5. Configure Raspbian for the Greengrass software.
6. Install the Greengrass Core on the Pi.
7. Deploy and execute a Python Lambda function on the Pi.

Hardware

In 2012, my team at Wazee Group gave me a Raspberry Pi Model B as part of our winter holiday celebration. It was an amazing unit and I spent several hours working with it. However, for this effort, I wanted to upgrade to the latest Pi hardware. Instead of obtaining a turn key kit, I purchased the following components from both Amazon and a local Best Buy.

1. Raspberry Pi 3 Model B+ - $34.99

2. CanaKit - Power Adapter for Raspberry Pi 3 – Black - $10.82

3. STEADYGAMER - 32GB Raspberry Pi Preloaded (NOOBS) SD Card | 3B+ (Plus), 3B, 2, Zero Compatible with All Pi Models - $22.99

4. KuGi Raspberry Pi 3 Model B case PC Protective Case with 2x Heatsinks for Raspberry Pi 3 Model B+, Raspberry Pi 3 Model B, Pi 2 Model B & Pi Model B+ - $7.50

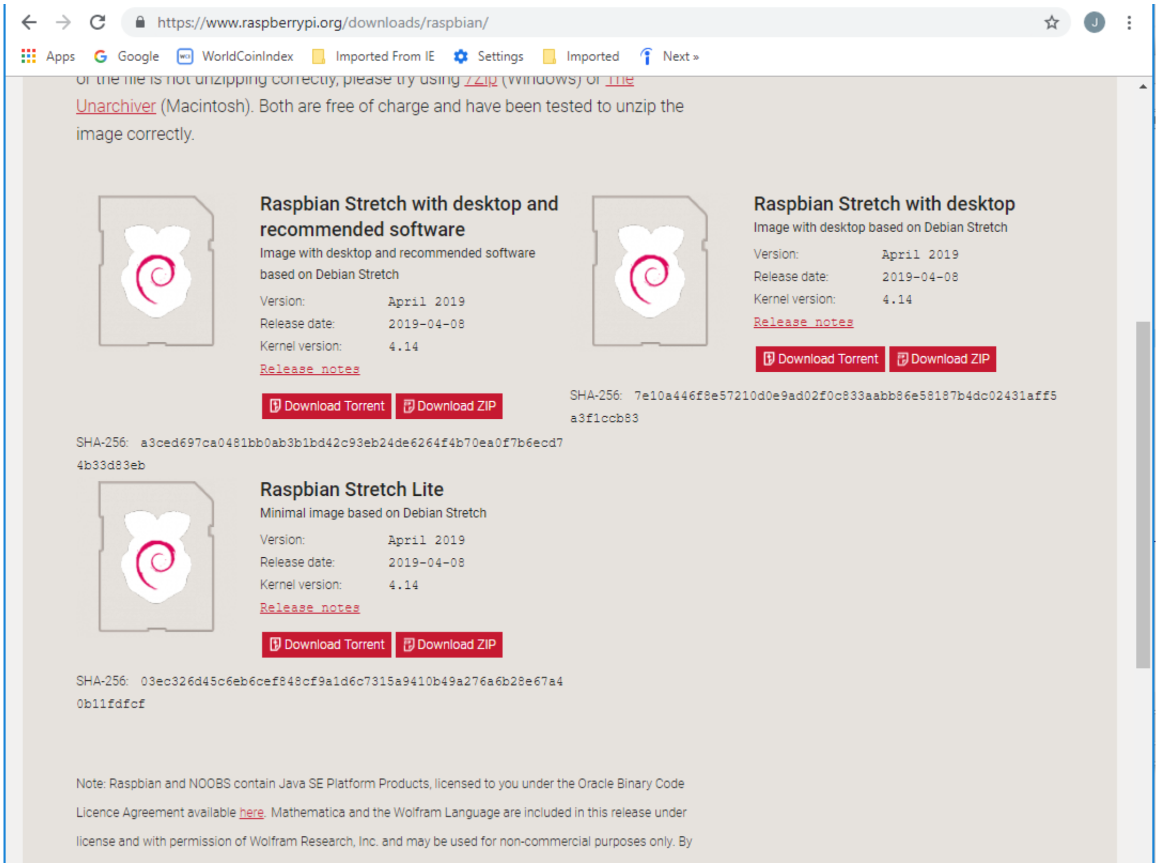
The total cost was $76.30. Not bad for an extremely impressive package. Here's a picture of my system.

Installing Raspbian on the Pi

After installing Pi board into the case but before connecting the power supply, I downloaded the latest release of Raspbian operating system from the following website:

https://www.raspberrypi.org/downloads/raspbian/

This link currently goes to the following page.



or the file is not unzipping correctly, please try using 7Zip (Windows) or The Unarchiver (Macintosh). Both are free of charge and have been tested to unzip the image correctly.

Raspbian Stretch with desktop and recommended software

Image with desktop and recommended software based on Debian Stretch

Version: April 2019
Release date: 2019-04-08
Kernel version: 4.14
Release notes

Download Torrent | Download ZIP

SHA-256: a3ced697ca0481bb0ab3b1bd42c93eb24de6264f4b70ea0f7b6ecd74b33d83eb

Raspbian Stretch with desktop

Image with desktop based on Debian Stretch

Version: April 2019
Release date: 2019-04-08
Kernel version: 4.14
Release notes

Download Torrent | Download ZIP

SHA-256: 7e10a446f8e57210d0e9ad02f0c833aabb86e58187b4dc02431aff5a3f1ccb83

Raspbian Stretch Lite

Minimal image based on Debian Stretch

Version: April 2019
Release date: 2019-04-08
Kernel version: 4.14
Release notes

Download Torrent | Download ZIP

SHA-256: 03ec326d45c6eb6cef848cf9a1d6c7315a9410b49a276a6b28e67a40b11fdfcf

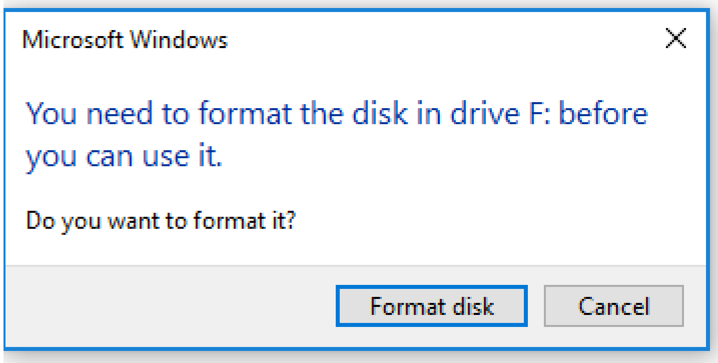
Note: Raspbian and NOOBS contain Java SE Platform Products, licensed to you under the Oracle Binary Code Licence Agreement available here. Mathematica and the Wolfram Language are included in this release under license and with permission of Wolfram Research, Inc. and may be used for non-commercial purposes only. By

I selected the ***Raspbian Stretch with desktop and recommended software*** option on this page. This option downloaded a 2 GB file named "2019-04-08-raspbian-stretch-full.zip"

Using the 7-Zip application on Windows, this file unzipped to a "2019-04-08-raspbian-stretch-full"directory with a single file named "2019-04-08-raspbian-stretch-full.img".

My laptop has a built in SD card reader so I downloaded the open source SD writing application called Etcher (https://www.balena.io/etcher/).

I placed the micro-card into the adapter and inserted the adapter into the reader. Windows showed the following message:

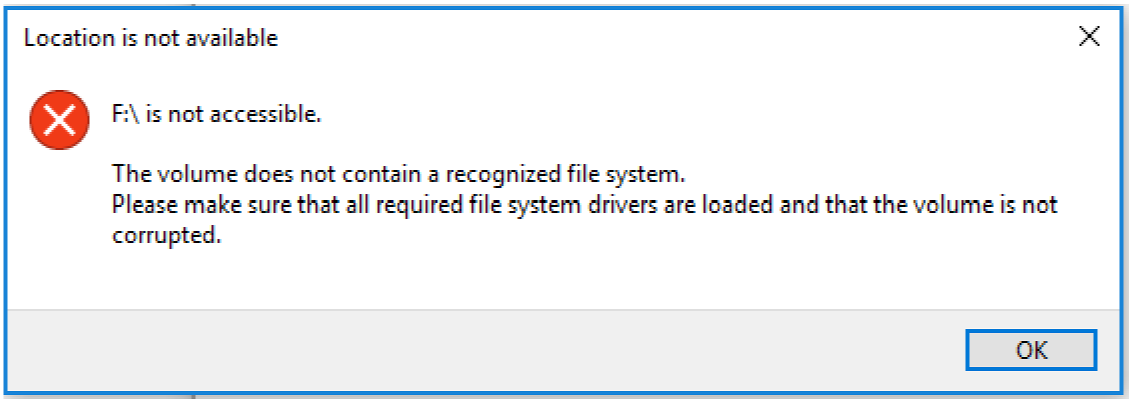


Microsoft Windows

You need to format the disk in drive F: before you can use it.

Do you want to format it?

Press ***Cancel***.



Location is not available

F:\ is not accessible.

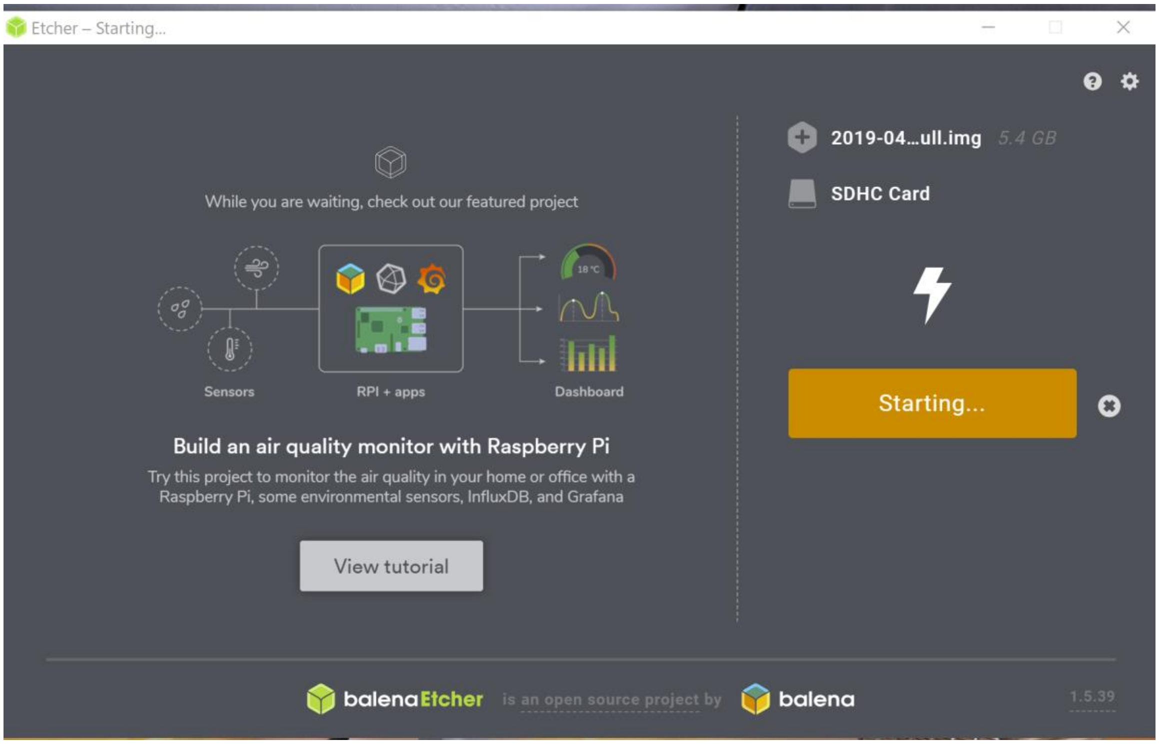
The volume does not contain a recognized file system.
Please make sure that all required file system drivers are loaded and that the volume is not corrupted.

Press the ***OK*** button.

Open the Etcher application.

Select the image (i.e. the "2019-04-08-raspbian-stretch-full.img" file).

Press the ***Flash!*** button.



Etcher – Starting...

While you are waiting, check out our featured project

Build an air quality monitor with Raspberry Pi

Try this project to monitor the air quality in your home or office with a Raspberry Pi, some environmental sensors, InfluxDB, and Grafana

2019-04...ull.img 5.4 GB

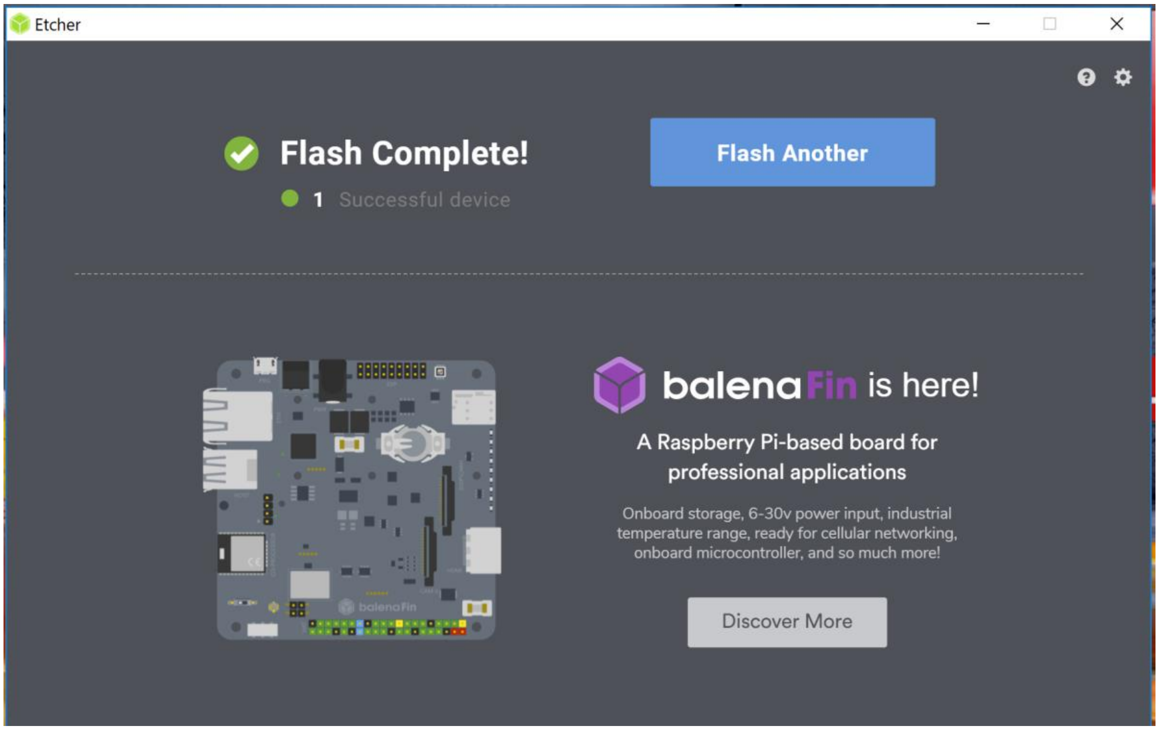
SDHC Card

Starting...

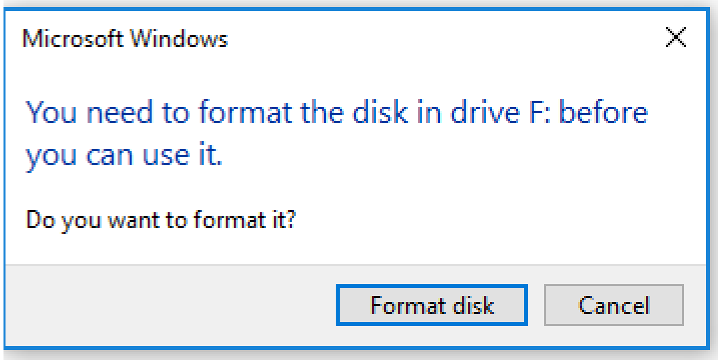
balenaEtcher is an open source project by balena 1.5.39

This takes a few minutes.

After flashing the image, Etcher will perform a validation check. This will also take a few minutes.


Etcher
Flash Complete!
1 Successful device
Flash Another
balenaFin is here!
A Raspberry Pi-based board for professional applications
Onboard storage, 6-30v power input, industrial temperature range, ready for cellular networking, onboard microcontroller, and so much more!
Discover More

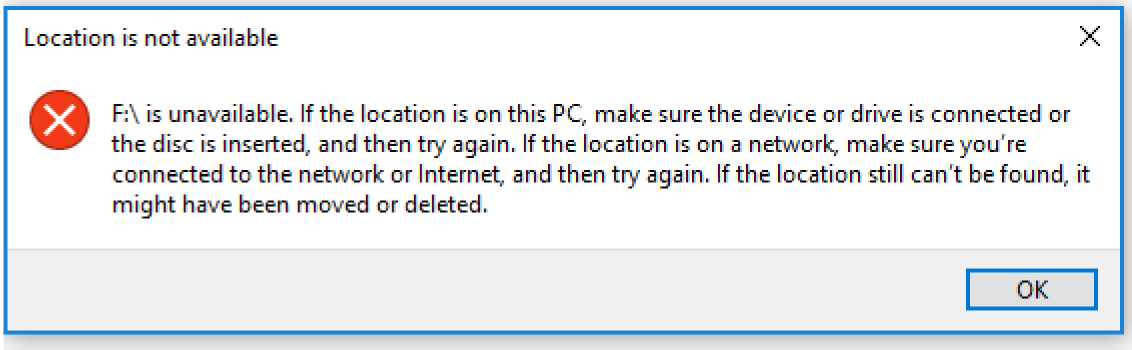
If Windows shows the following message, press the ***Cancel*** button.


Microsoft Windows

You need to format the disk in drive F: before you can use it.

Do you want to format it?

If Windows shows the following message, press the ***OK*** button:



Location is not available

F:\ is unavailable. If the location is on this PC, make sure the device or drive is connected or the disc is inserted, and then try again. If the location is on a network, make sure you're connected to the network or Internet, and then try again. If the location still can't be found, it might have been moved or deleted.

Close the Etcher application.

Remove the SD adapter from the reader.

Remote the micro-card from the adapter.

Ensure the power supply is disconnected from the Raspberry.

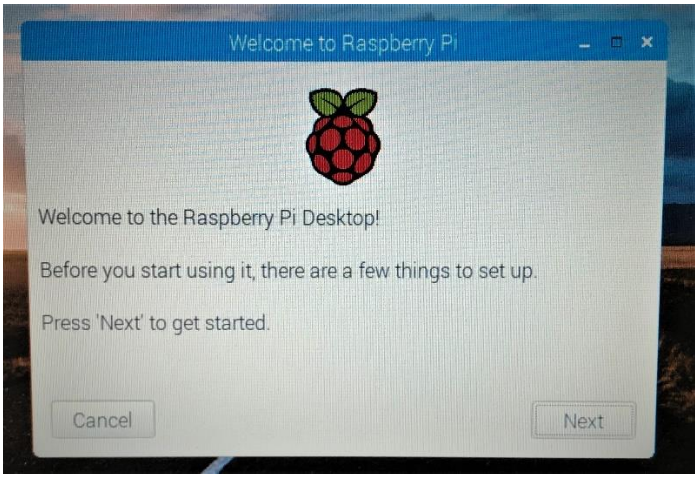
Insert the micro-card into the Pi.

Connect the video monitor, keyboard, and mouse to the Pi.

Connect the power supply to the Pi.

Initial Raspbian Configuration

After the Pi powers up, the following message should appear:

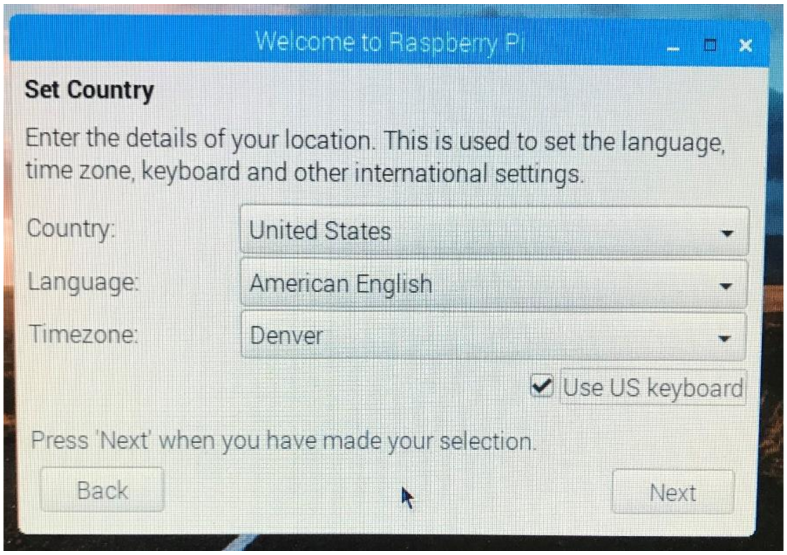


Welcome to the Raspberry Pi Desktop!

Before you start using it, there are a few things to set up.

Press 'Next' to get started.

Press the ***Next*** button. The following dialog should appear:



Set Country

Enter the details of your location. This is used to set the language, time zone, keyboard and other international settings.

Country: United States

Language: American English

Timezone: Denver

☑ Use US keyboard

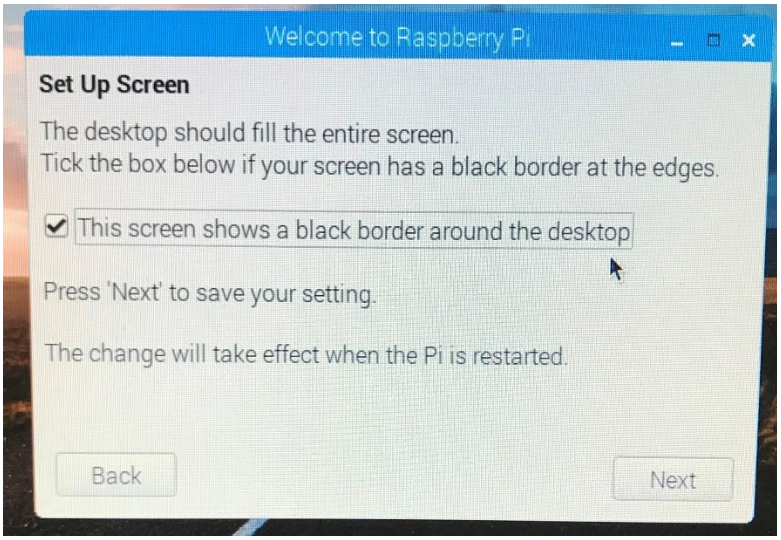
Press 'Next' when you have made your selection.

Set the values for your scenario. Press ***Next***.

The ***Change Password*** dialog will appear. Enter the new password for the default 'pi' user.

Confirm the password and press the ***Next*** button.

The ***Set Up Screen*** dialog will appear.



Set Up Screen

The desktop should fill the entire screen.
Tick the box below if your screen has a black border at the edges.

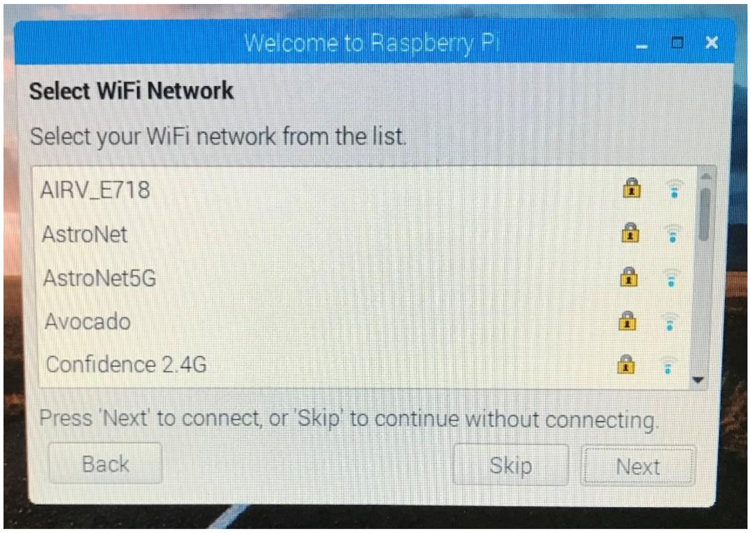
☑ This screen shows a black border around the desktop

Press 'Next' to save your setting.

The change will take effect when the Pi is restarted.

Press the ***Next*** button.

The ***Select WiFi Network*** dialog will appear. Select your network. You should select the same network that your computer uses.



Select WiFi Network

Select your WiFi network from the list.

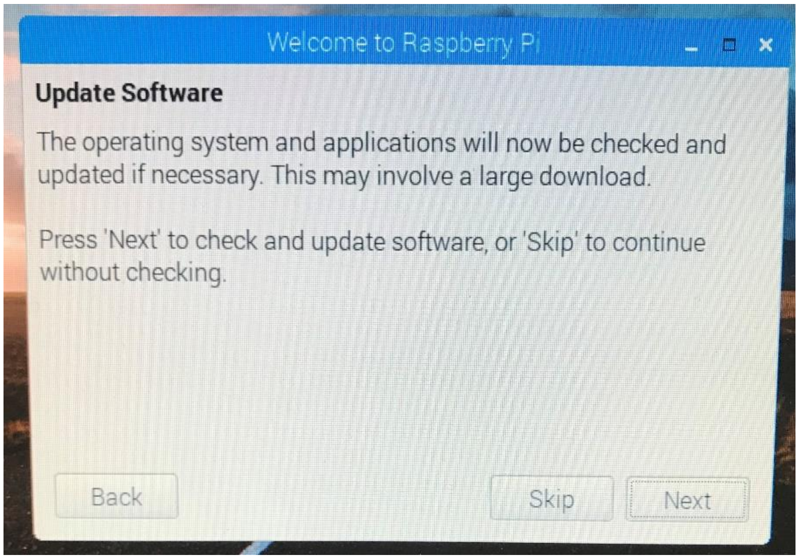
AIRV_E718
AstroNet
AstroNet5G
Avocado
Confidence 2.4G

Press 'Next' to connect, or 'Skip' to continue without connecting.

Press the ***Next*** button.

The ***Enter WiFi Password*** dialog will appear. Enter the value for your network and press the ***Next*** button.

The ***Update Software*** dialog will appear.



Update Software

The operating system and applications will now be checked and updated if necessary. This may involve a large download.

Press 'Next' to check and update software, or 'Skip' to continue without checking.

Press the ***Next*** button.

This can take several minutes.

When the update is finished, the ***Setup Complete*** dialog will appear. Click the ***Restart*** button.

No Pi for You!

This section describes a few techniques to increase the security of your Pi. It is not an all-inclusive set but represents reasonable precautions for a lab-based platform. For production releases, please research and evaluate additional safeguards.

Enable SSH

The first step is to enable the SSH interface. After the Pi has restarted from the previous section, the SSH interface can be enabled. This allows a user to remoted login to the Pi.

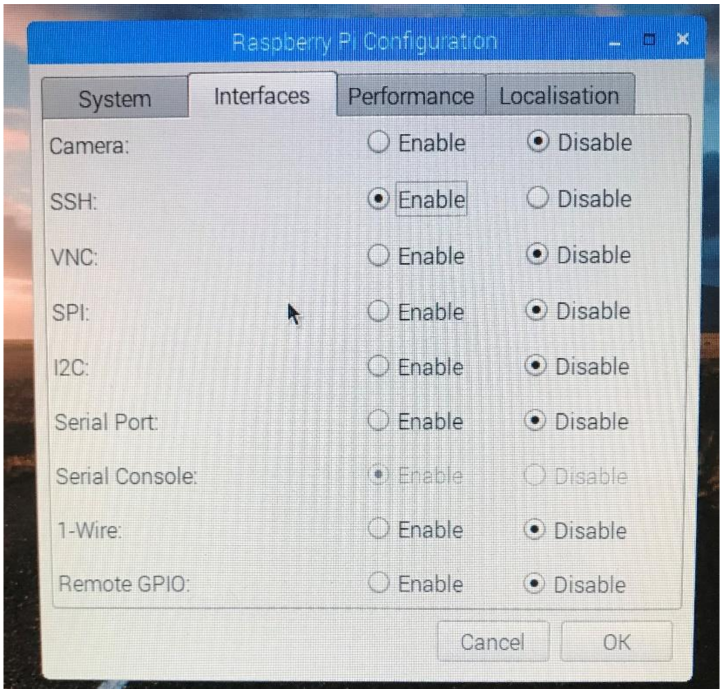
On the Pi desktop, click on the raspberry icon in the menu bar.

Click on the ***Preferences*** item.

Click on the ***Raspberry Pi Configuration*** item

The ***Raspberry Pi Configuration*** dialog will appear.

Click on the ***Interfaces*** tab.



Camera:	○ Enable	● Disable
SSH:	● Enable	○ Disable
VNC:	○ Enable	● Disable
SPI:	○ Enable	● Disable
I2C:	○ Enable	● Disable
Serial Port:	○ Enable	● Disable
Serial Console:	● Enable	○ Disable
1-Wire:	○ Enable	● Disable
Remote GPIO:	○ Enable	● Disable

Click on the ***Enable*** option for the SSH interface.

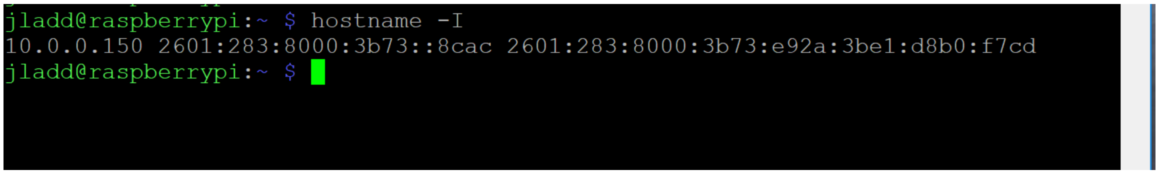
Ensure that all of the other interfaces have the ***Disable*** option select. The lone exception is the Serial Console which cannot be disabled.

Click the ***OK*** button.

To obtain the IP address of the Pi, open a terminal window and enter the following command:

```
hostname -I
```

For my platform, the information is:

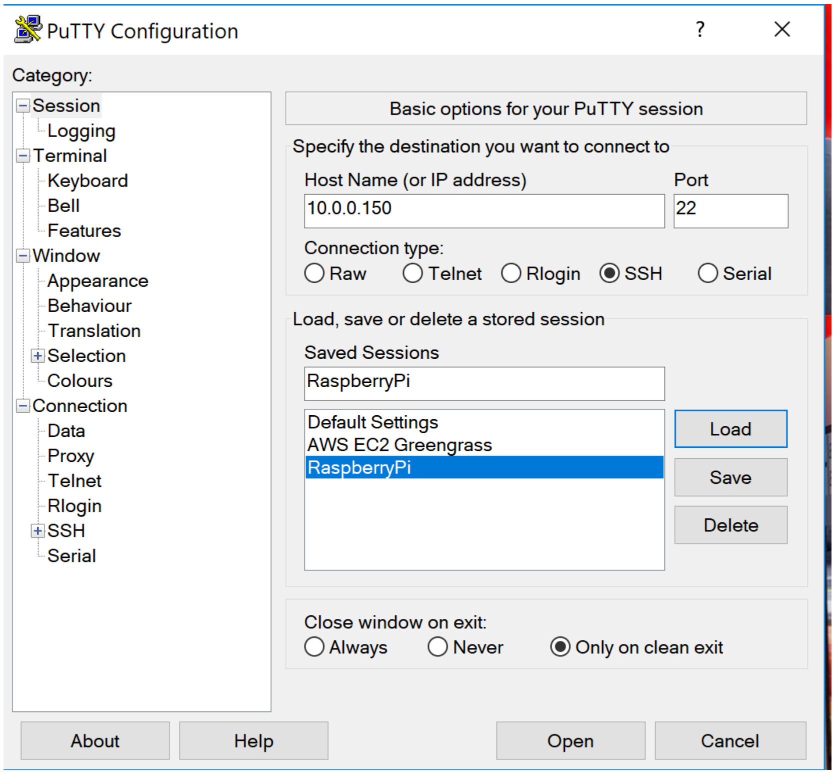


```
jladd@raspberrypi:~ $ hostname -I
10.0.0.150 2601:283:8000:3b73::8cac 2601:283:8000:3b73:e92a:3be1:d8b0:f7cd
jladd@raspberrypi:~ $
```

Download an SSH client application. I have used PuTTY for several years. Its available at the following link.

https://www.putty.org/

After installation, create a new session in PuTTY that will connect to the Pi:



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 10.0.0.150

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

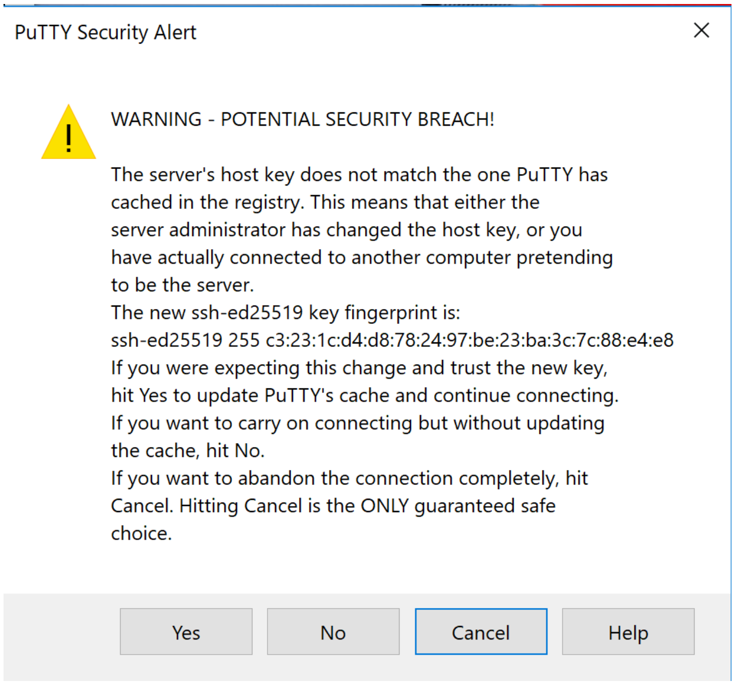
Saved Sessions: RaspberryPi

Default Settings
AWS EC2 Greengrass
RaspberryPi

Close window on exit: ○ Always ○ Never ● Only on clean exit

Enter your information and press the ***Open*** button.

You will probably receive the following message:



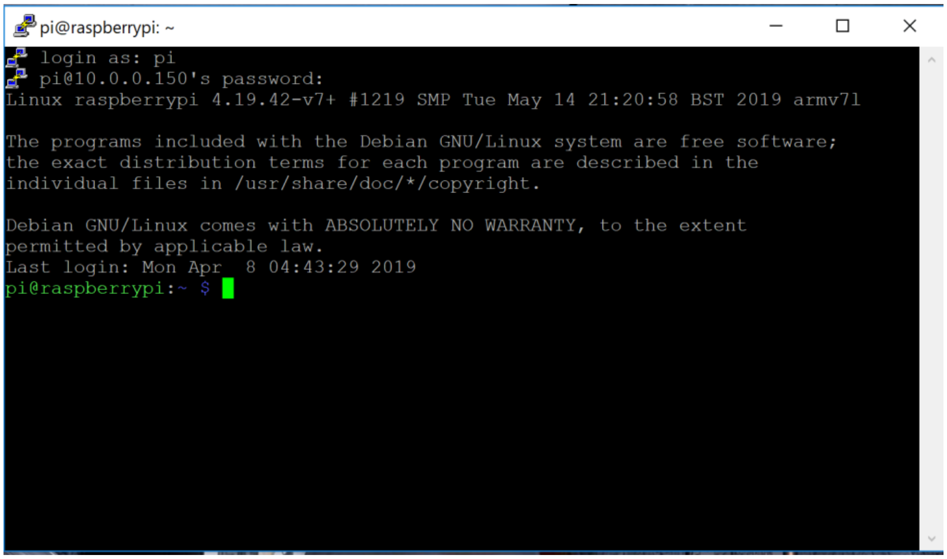
PuTTY Security Alert

WARNING - POTENTIAL SECURITY BREACH!

The server's host key does not match the one PuTTY has cached in the registry. This means that either the server administrator has changed the host key, or you have actually connected to another computer pretending to be the server.
The new ssh-ed25519 key fingerprint is:
ssh-ed25519 255 c3:23:1c:d4:d8:78:24:97:be:23:ba:3c:7c:88:e4:e8
If you were expecting this change and trust the new key, hit Yes to update PuTTY's cache and continue connecting.
If you want to carry on connecting but without updating the cache, hit No.
If you want to abandon the connection completely, hit Cancel. Hitting Cancel is the ONLY guaranteed safe choice.

Press the ***Yes*** button.

An SSH terminal windows will appear.



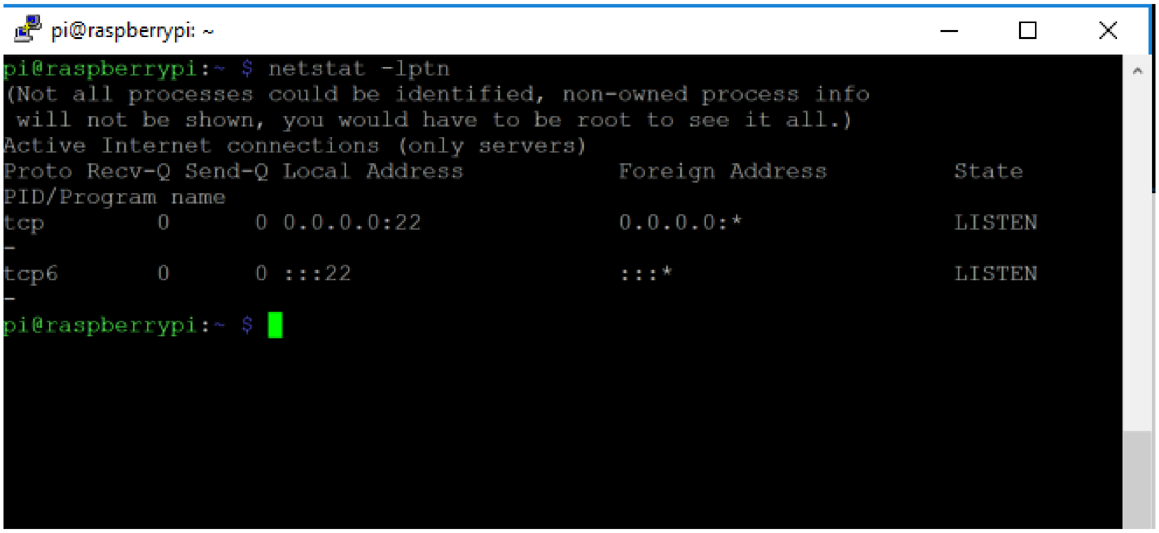
pi@raspberrypi: ~

```
login as: pi
pi@10.0.0.150's password:
Linux raspberrypi 4.19.42-v7+ #1219 SMP Tue May 14 21:20:58 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr  8 04:43:29 2019
pi@raspberrypi:~ $
```

Change the SSH Port

See what ports are being used:

```
netstat -lptn
```

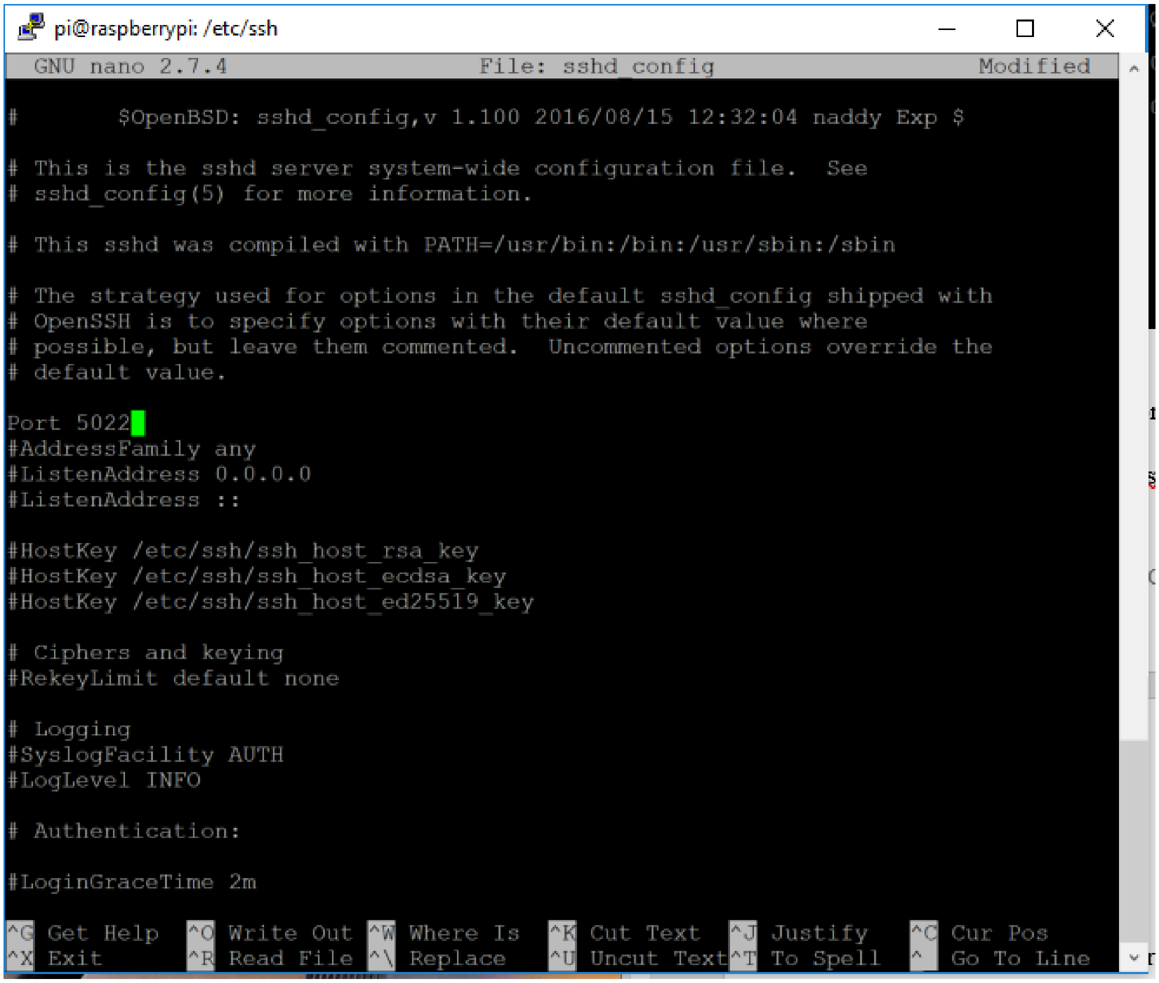


```
pi@raspberrypi:~ $ netstat -lptn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
-
tcp6       0      0 :::22                   :::*                    LISTEN
-
pi@raspberrypi:~ $
```

Change the SSH port:

```
cd /etc/ssh

sudo nano sshd_config
```

Uncomment the Port and change it to the new value. For my scenario, I used 5022.



pi@raspberrypi: /etc/ssh

```
  GNU nano 2.7.4              File: sshd_config                  Modified

#       $OpenBSD: sshd_config,v 1.100 2016/08/15 12:32:04 naddy Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Port 5022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Enter *Control-O* to write the changes to the file.

Enter *Control-X* to exit the nano editor.

The Pi must be restarted for the change to take effect.

Enter "`sudo reboot`" in the terminal window.

Once the Pi has rebooted, start a new session in the PuTTY application (with the new value for the port).

Remove the "pi" User from SSH Access

Most experts would recommend removing the default user, i.e. the "pi" account, from the system. However, I read that some of the applications and processes installed on Raspbian require the “pi” user. As a compromise, I kept the pi user account but remove its access via the SSH interface. Before removing this access, create a new user that will have access via SSH.

To add a new user, enter in a terminal window:

```
sudo adduser <username>
```

In my scenario the command is:

```
sudo adduser jladd
```

You will be prompted for the new password and then asked to confirm the password.

Add the new user to the sudo group:

```
sudo adduser <username> sudo
```

In my scenario the command is:

```
sudo adduser jladd sudo
```

Ensure the change took effect by checking the permissions:

```
sudo su
```

If everything took effect, remove the pi user from SSH logins by invoking the nano text editor:

```
sudo nano /etc/ssh/sshd_config
```

Navigate to the end of the file and add the following:

```
AllowUsers <username>
```

In my scenario the line is:

```
AllowUsers jladd
```

Enter *Control-O* to write the change to the file.

Enter *Control-X* to exit the nano editor.

Enter the following to reboot the Pi:

```
sudo reboot
```

If you try to login via SSH with the pi user, the access will be denied.

If you try to login via SSH with the new user, you should have access.

Using MFA with Google Authenticator

Another security step is to configure the Pi to use MFA with the Google smart phone app. First install the Authenticator application on your smart phone.

In a terminal window on the Pi, enter:

```
sudo apt-get install libpam-google-authenticator
```

then:

```
google-authenticator
```

Enter the following command to invoke the nano editor:

```
sudo nano /etc/pam.d/sshd
```

Add the following text at the beginning of the file:

```
auth required pam_google_authenticator.so
```

Enter *Control-O* to write the change to the file.

Enter *Control-X* to exit the nano editor.

Now edit the sshd file. Enter the following command to invoke the nano editor:

```
sudo nano /etc/pam.d/sshd
```

Locate the line containing *ChallengeResponseAuthentication*. This option is set to "no" by default, change it to "yes".

Enter `Control-O` to write the change to the file.

Enter `Control-X` to exit the nano editor.

Enter the following to reboot the Pi:

```
sudo reboot
```

Now when a person tries to login to the Pi via the SSH interface, they will be prompted for the Google authenticator value first and, if successful, they will be prompted for the password for the user account. This approach avoids the situation where a hacker is guessing the password before encountering the Google authenticator challenge.

Require User Credentials with the GUI

The last recommended security step is to require user credentials when using the GUI. First open the terminal window or login via SSH.

Edit the configuration file for the desktop manager:

```
sudo nano /etc/lightdm/lightdm.conf
```

Disable the auto login option. Find the following line and comment it out.

```
auto-login-user=pi
```

becomes

```
#auto-login-user=pi
```

Enable the option to hide the list of users. Find the following line and change it:

```
greet-hide-users=false
```

becomes

```
greet-hide-users=true
```

Save the file and reboot. When the GUI appears, it will have a login screen.

Configuring Raspbian for Greengrass

This section prepares the Pi for the AWS Greengrass code. First add a user and group for the Greengrass software. In a terminal window, enter the following commands:

```
sudo adduser --system ggc_user

sudo addgroup --system ggc_group
```

Enable hardlink and softlink protection at startup by editing a configuration file.

```
sudo nano /etc/sysctl.d/98-rpi.conf
```

Add the following lines to the end of the file:

```
fs.protected_hardlinks = 1

fs.protected_symlinks = 1
```

Enter *Control-O* to write the changes to the file.

Enter *Control-X* to exit the nano editor.

Enter the following to reboot the Pi:

```
sudo reboot
```

Enable and mount memory cgroups by first editing another configuration file:

```
sudo nano /boot/cmdline.txt
```

Append the following to the end of the line (NOT AS A NEW LINE)

```
cgroup_enable=memory cgroup_memory=1
```

Enter *Control-O* to write the changes to the file.

Enter *Control-X* to exit the nano editor.

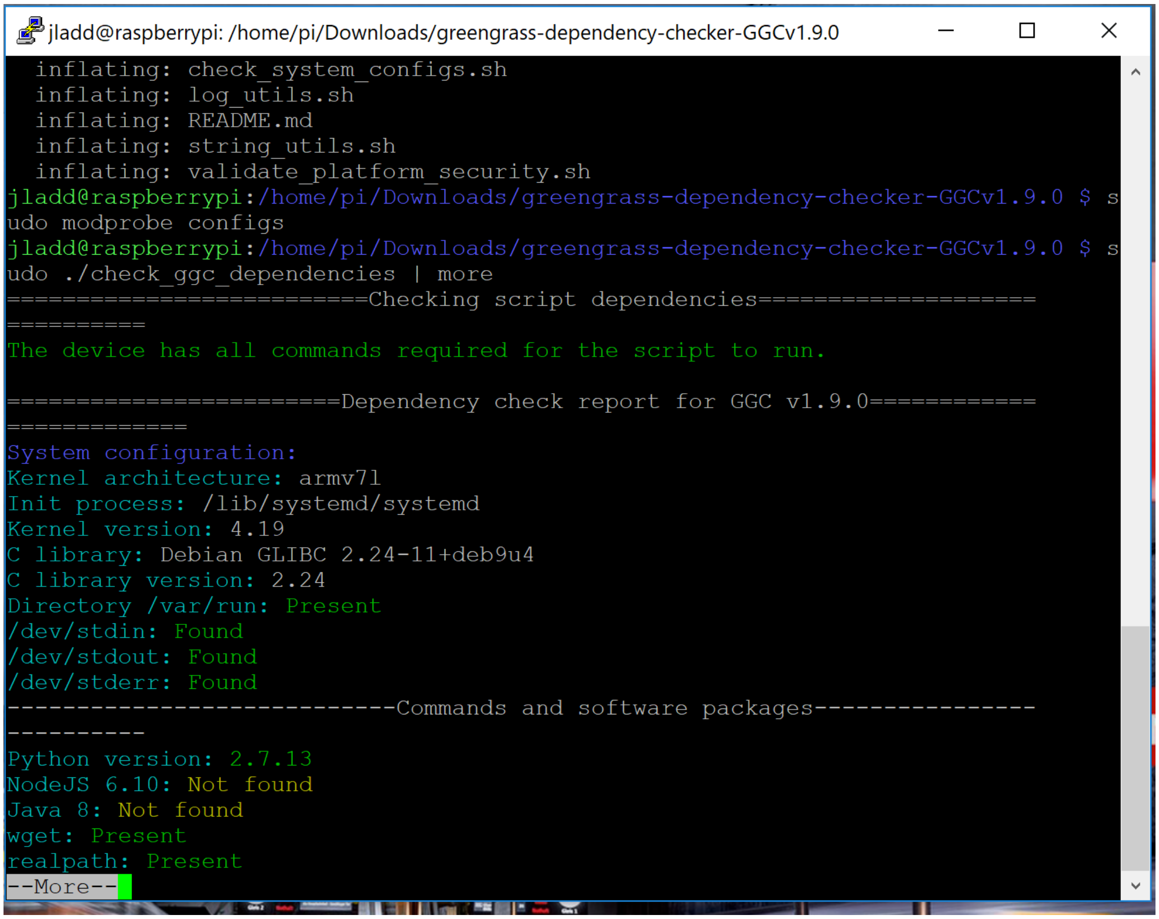
Enter the following to reboot the Pi:

```
sudo reboot
```

Time to check the dependencies for Greengrass. Download the validation program and execute it.

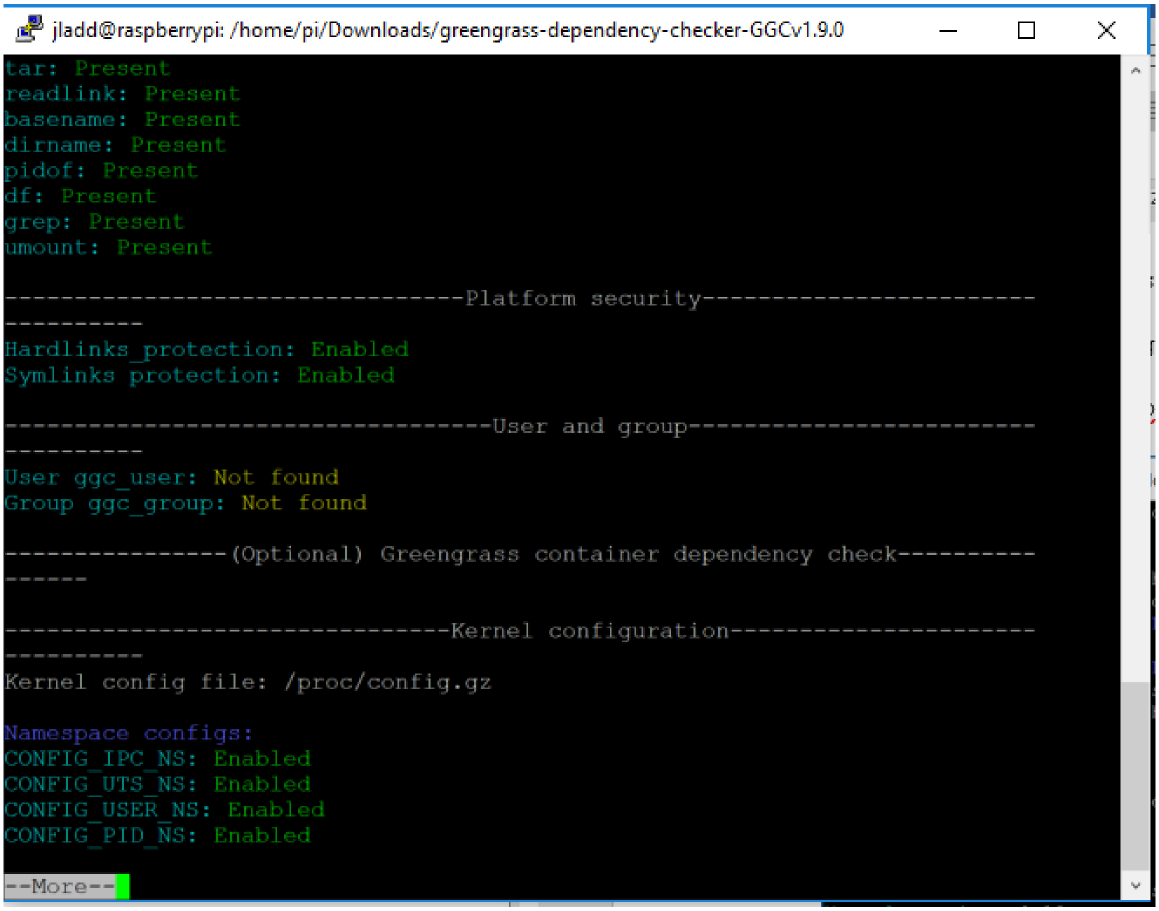
```
cd /home/pi/Downloads

sudo mkdir greengrass-dependency-checker-GGCv1.9.0

cd greengrass-dependency-checker-GGCv1.9.0
```

```
sudo wget https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-dependency-checker-GGCv1.9.0.zip

sudo unzip greengrass-dependency-checker-GGCv1.9.0.zip

sudo modprobe configs

sudo ./check_ggc_dependencies | more
```

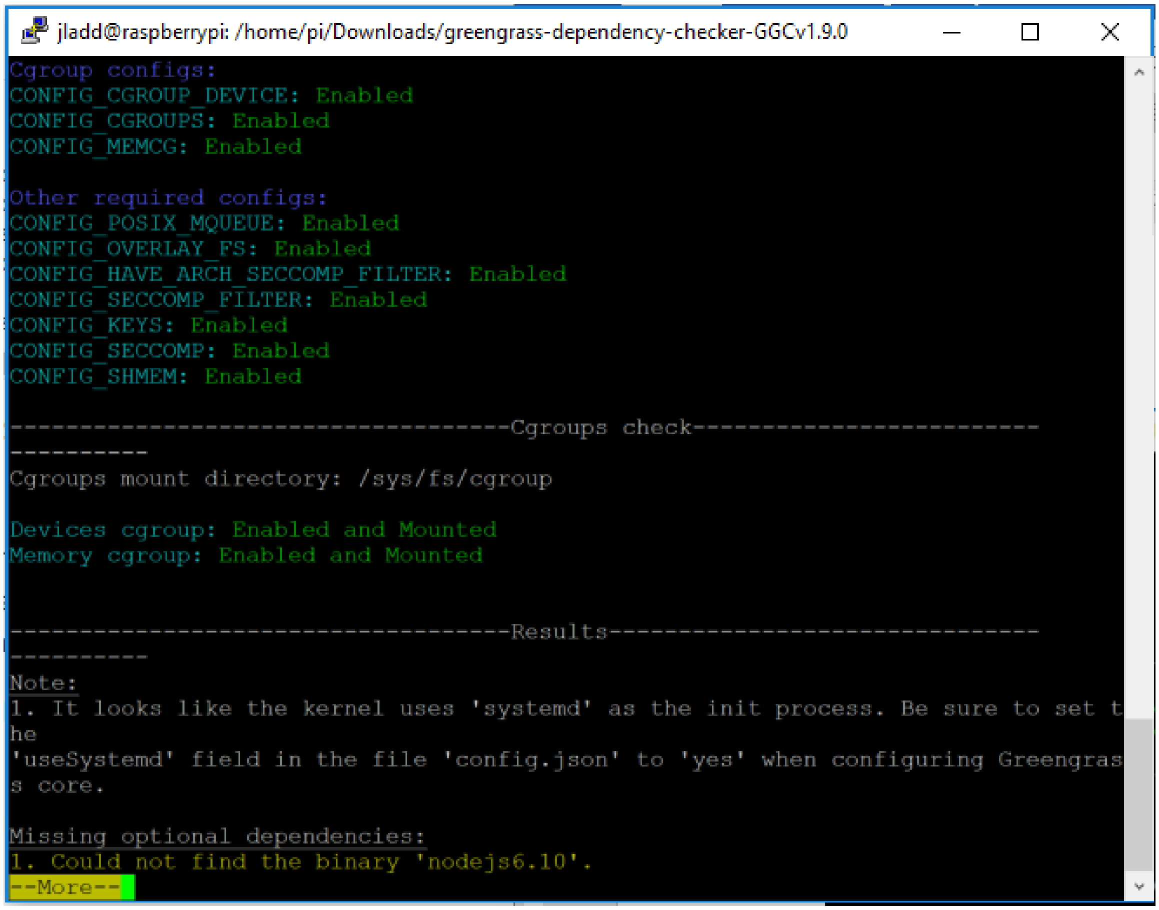


jladd@raspberrypi: /home/pi/Downloads/greengrass-dependency-checker-GGCv1.9.0

```
  inflating: check_system_configs.sh
  inflating: log_utils.sh
  inflating: README.md
  inflating: string_utils.sh
  inflating: validate_platform_security.sh
jladd@raspberrypi:/home/pi/Downloads/greengrass-dependency-checker-GGCv1.9.0 $ s
udo modprobe configs
jladd@raspberrypi:/home/pi/Downloads/greengrass-dependency-checker-GGCv1.9.0 $ s
udo ./check_ggc_dependencies | more
==========================Checking script dependencies=====================
==========
The device has all commands required for the script to run.

========================Dependency check report for GGC v1.9.0============
=============
System configuration:
Kernel architecture: armv7l
Init process: /lib/systemd/systemd
Kernel version: 4.19
C library: Debian GLIBC 2.24-11+deb9u4
C library version: 2.24
Directory /var/run: Present
/dev/stdin: Found
/dev/stdout: Found
/dev/stderr: Found
----------------------------Commands and software packages----------------
----------
Python version: 2.7.13
NodeJS 6.10: Not found
Java 8: Not found
wget: Present
realpath: Present
--More--
```


jladd@raspberrypi: /home/pi/Downloads/greengrass-dependency-checker-GGCv1.9.0

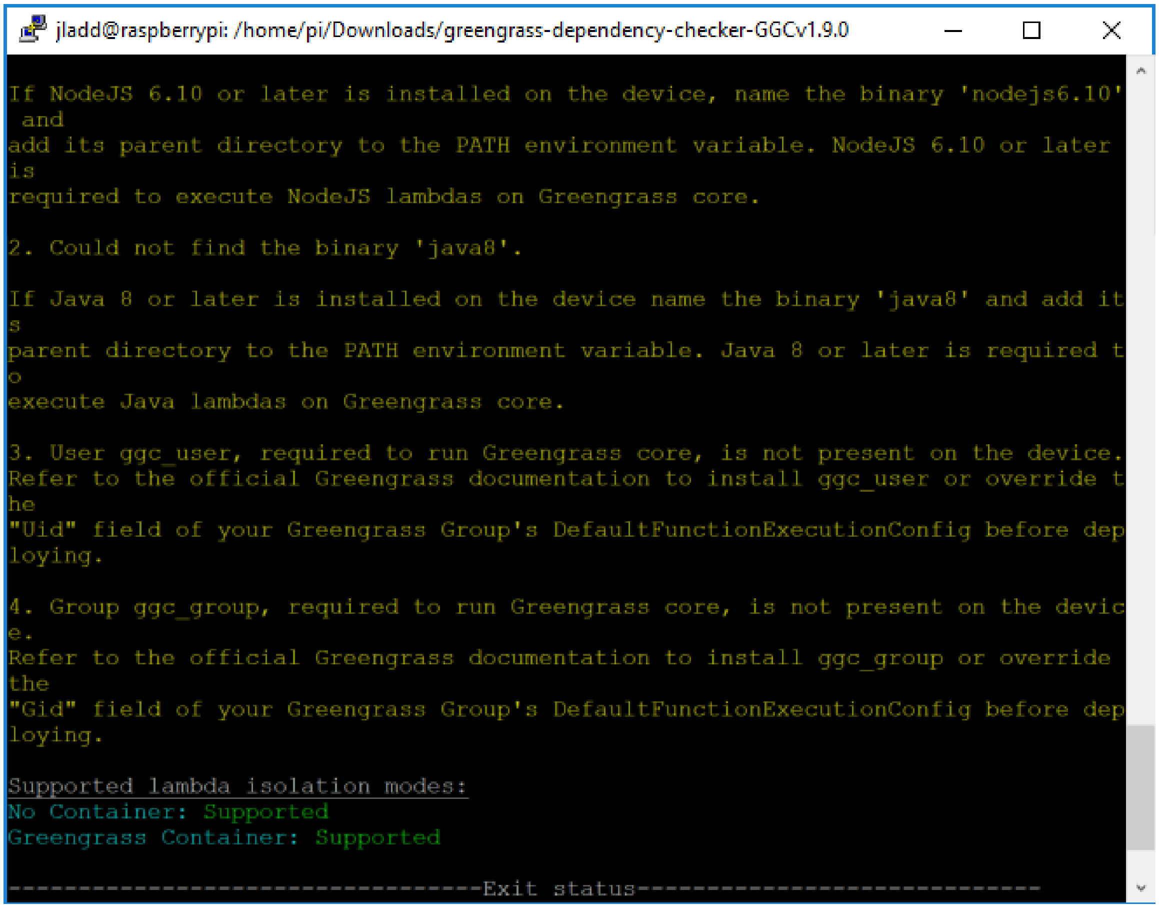
```
tar: Present
readlink: Present
basename: Present
dirname: Present
pidof: Present
df: Present
grep: Present
umount: Present

---------------------------------Platform security------------------------
----------
Hardlinks_protection: Enabled
Symlinks_protection: Enabled

-----------------------------------User and group-------------------------
----------
User ggc_user: Not found
Group ggc_group: Not found

----------------(Optional) Greengrass container dependency check----------
------

--------------------------------Kernel configuration----------------------
----------
Kernel config file: /proc/config.gz

Namespace configs:
CONFIG_IPC_NS: Enabled
CONFIG_UTS_NS: Enabled
CONFIG_USER_NS: Enabled
CONFIG_PID_NS: Enabled

--More--
```



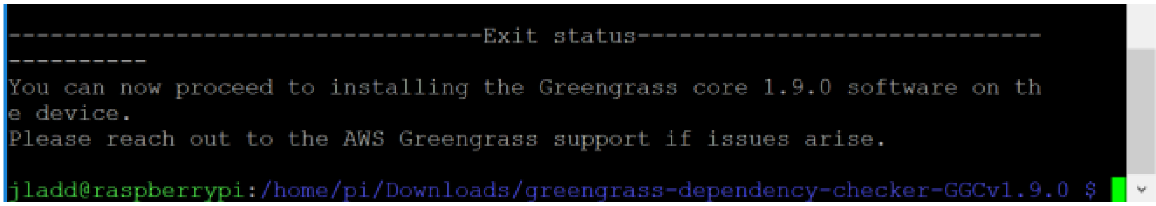
jladd@raspberrypi: /home/pi/Downloads/greengrass-dependency-checker-GGCv1.9.0

```
Cgroup configs:
CONFIG_CGROUP_DEVICE: Enabled
CONFIG_CGROUPS: Enabled
CONFIG_MEMCG: Enabled

Other required configs:
CONFIG_POSIX_MQUEUE: Enabled
CONFIG_OVERLAY_FS: Enabled
CONFIG_HAVE_ARCH_SECCOMP_FILTER: Enabled
CONFIG_SECCOMP_FILTER: Enabled
CONFIG_KEYS: Enabled
CONFIG_SECCOMP: Enabled
CONFIG_SHMEM: Enabled

------------------------------------Cgroups check------------------------
----------
Cgroups mount directory: /sys/fs/cgroup

Devices cgroup: Enabled and Mounted
Memory cgroup: Enabled and Mounted


------------------------------------Results------------------------------
----------
Note:
1. It looks like the kernel uses 'systemd' as the init process. Be sure to set t
he
'useSystemd' field in the file 'config.json' to 'yes' when configuring Greengras
s core.

Missing optional dependencies:
1. Could not find the binary 'nodejs6.10'.
--More--
```

NOTE: You can ignore the warning message regarding Node.js and Java.


jladd@raspberrypi: /home/pi/Downloads/greengrass-dependency-checker-GGCv1.9.0

```
If NodeJS 6.10 or later is installed on the device, name the binary 'nodejs6.10'
 and
add its parent directory to the PATH environment variable. NodeJS 6.10 or later
is
required to execute NodeJS lambdas on Greengrass core.

2. Could not find the binary 'java8'.

If Java 8 or later is installed on the device name the binary 'java8' and add it
s
parent directory to the PATH environment variable. Java 8 or later is required t
o
execute Java lambdas on Greengrass core.

3. User ggc_user, required to run Greengrass core, is not present on the device.
Refer to the official Greengrass documentation to install ggc_user or override t
he
"Uid" field of your Greengrass Group's DefaultFunctionExecutionConfig before dep
loying.

4. Group ggc_group, required to run Greengrass core, is not present on the devic
e.
Refer to the official Greengrass documentation to install ggc_group or override
the
"Gid" field of your Greengrass Group's DefaultFunctionExecutionConfig before dep
loying.

Supported lambda isolation modes:
No Container: Supported
Greengrass Container: Supported

----------------------------------Exit status-----------------------------
```



```
----------------------------------Exit status-----------------------------
----------
You can now proceed to installing the Greengrass core 1.9.0 software on th
e device.
Please reach out to the AWS Greengrass support if issues arise.

jladd@raspberrypi:/home/pi/Downloads/greengrass-dependency-checker-GGCv1.9.0 $
```

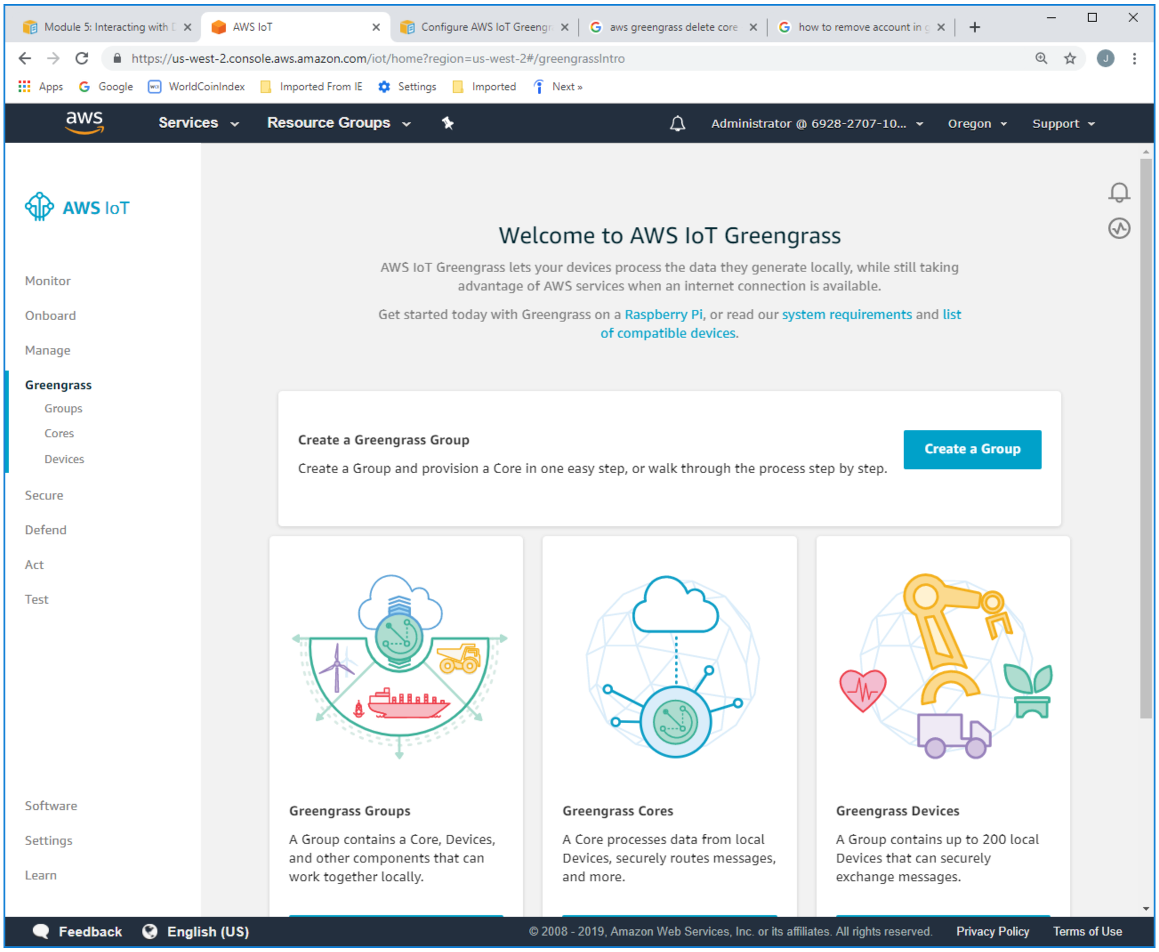
If everything looks okay, you are ready to install the Greengrass Core code in the next section.

Installing the Greengrass Core

This section describes the steps needed to install the Greengrass Core software on the Pi. The first step is the create a Greengrass Group.

Login to ***AWS Console***

Select ***IoT Greengrass*** (under the ***Internet of Things***). The following page should appear:



Welcome to AWS IoT Greengrass

AWS IoT Greengrass lets your devices process the data they generate locally, while still taking advantage of AWS services when an internet connection is available.

Get started today with Greengrass on a Raspberry Pi, or read our system requirements and list of compatible devices.

Create a Greengrass Group

Create a Group and provision a Core in one easy step, or walk through the process step by step.

Greengrass Groups

A Group contains a Core, Devices, and other components that can work together locally.

Greengrass Cores

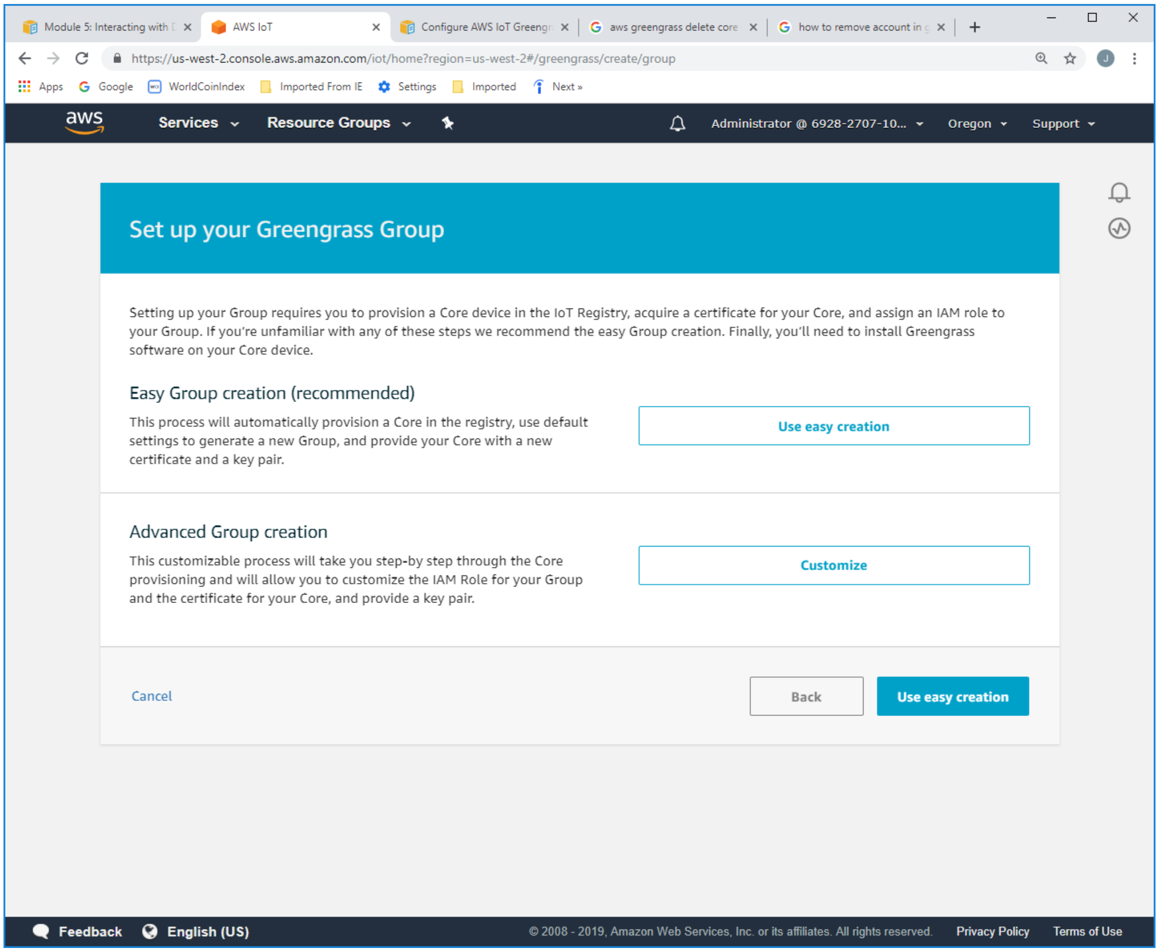
A Core processes data from local Devices, securely routes messages, and more.

Greengrass Devices

A Group contains up to 200 local Devices that can securely exchange messages.

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Click the ***Create a Group*** button.


Module 5: Interacting with
AWS IoT
Configure AWS IoT Greengr
aws greengrass delete core
how to remove account in
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/create/group
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Set up your Greengrass Group

Setting up your Group requires you to provision a Core device in the IoT Registry, acquire a certificate for your Core, and assign an IAM role to your Group. If you're unfamiliar with any of these steps we recommend the easy Group creation. Finally, you'll need to install Greengrass software on your Core device.

Easy Group creation (recommended)

This process will automatically provision a Core in the registry, use default settings to generate a new Group, and provide your Core with a new certificate and a key pair.

Use easy creation

Advanced Group creation

This customizable process will take you step-by step through the Core provisioning and will allow you to customize the IAM Role for your Group and the certificate for your Core, and provide a key pair.

Customize

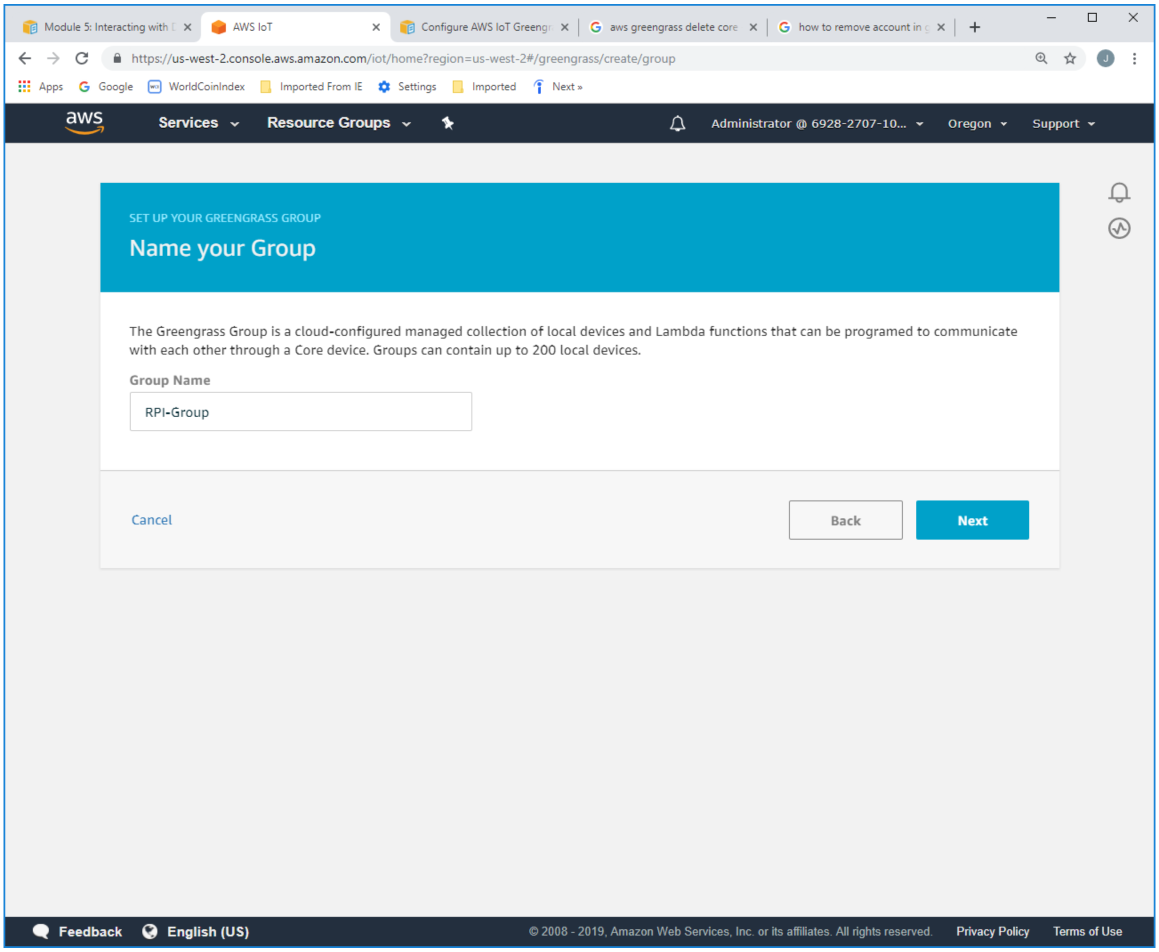
Cancel

Back

Use easy creation

Click the ***Use easy creation*** button.

On the next page, enter the name of your Greengrass group.


Module 5: Interacting with
AWS IoT
Configure AWS IoT Greengr
aws greengrass delete core
how to remove account in
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/create/group
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support

SET UP YOUR GREENGRASS GROUP

Name your Group

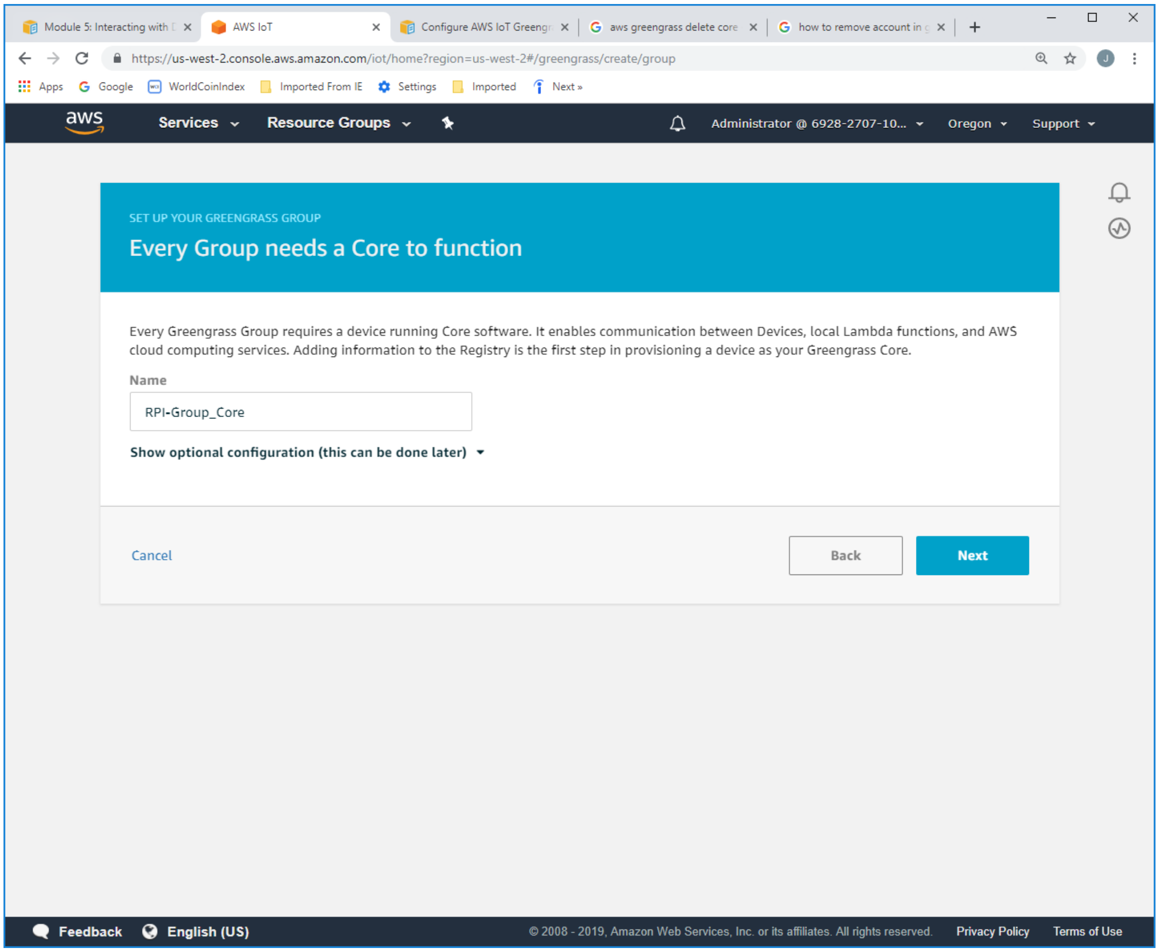
The Greengrass Group is a cloud-configured managed collection of local devices and Lambda functions that can be programed to communicate with each other through a Core device. Groups can contain up to 200 local devices.

Group Name

RPI-Group

Cancel
Back
Next
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Click the ***Next*** button. The following page should appear:


Module 5: Interacting with
AWS IoT
Configure AWS IoT Greengr
aws greengrass delete core
how to remove account in
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/create/group
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

SET UP YOUR GREENGRASS GROUP

Every Group needs a Core to function

Every Greengrass Group requires a device running Core software. It enables communication between Devices, local Lambda functions, and AWS cloud computing services. Adding information to the Registry is the first step in provisioning a device as your Greengrass Core.

Name: RPI-Group_Core

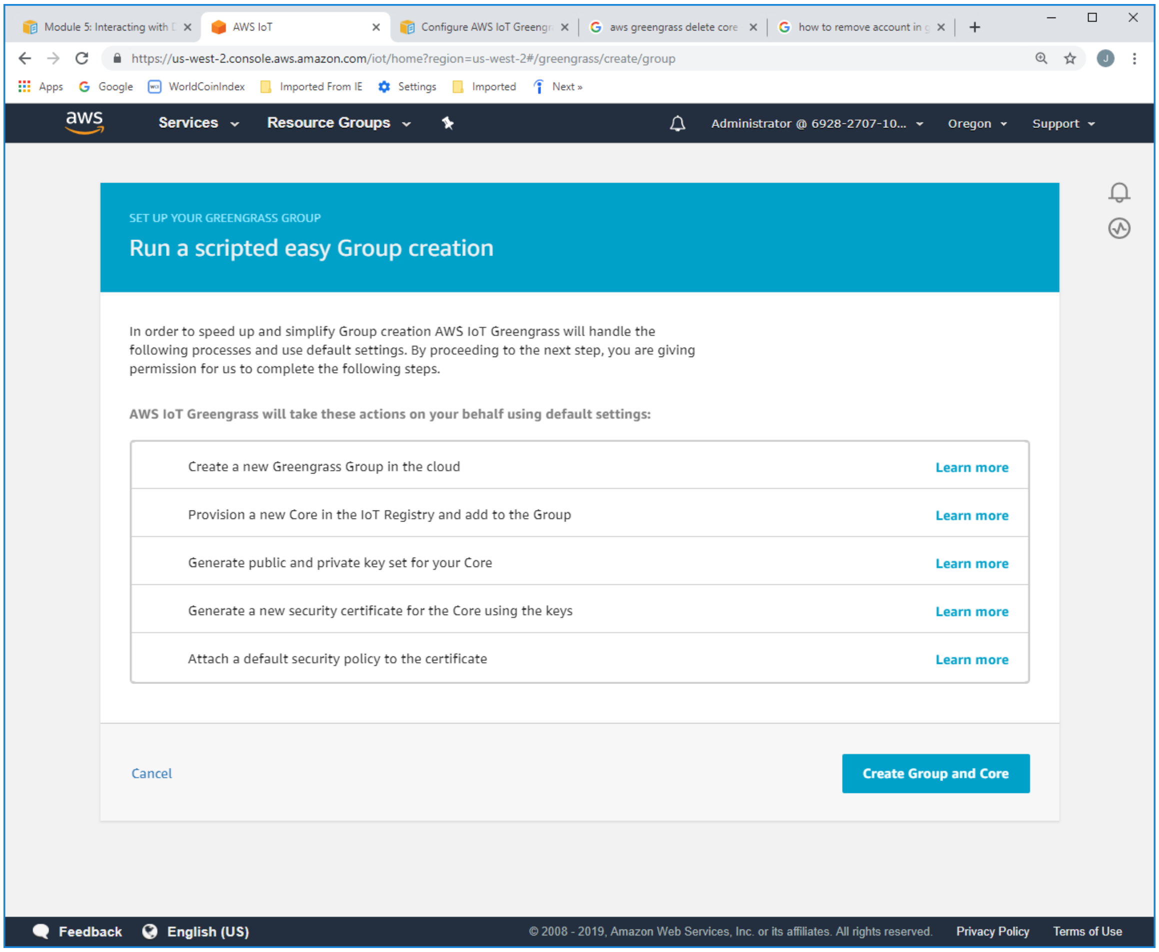
Show optional configuration (this can be done later)
Cancel
Back
Next

Every Greengrass group requires one device to run the Core software. Enter the name of the device to host the Core software for the new group.

Press the ***Next*** button.

The next page describes all of the provisioning steps that will be automatically executed for the new group and core. This is one of the benefits of using the Greengrass service.

Press the ***Create Group and Core*** button.



Module 5: Interacting with
AWS IoT
Configure AWS IoT Greengr
aws greengrass delete core
how to remove account in
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/create/group
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support

SET UP YOUR GREENGRASS GROUP

Run a scripted easy Group creation

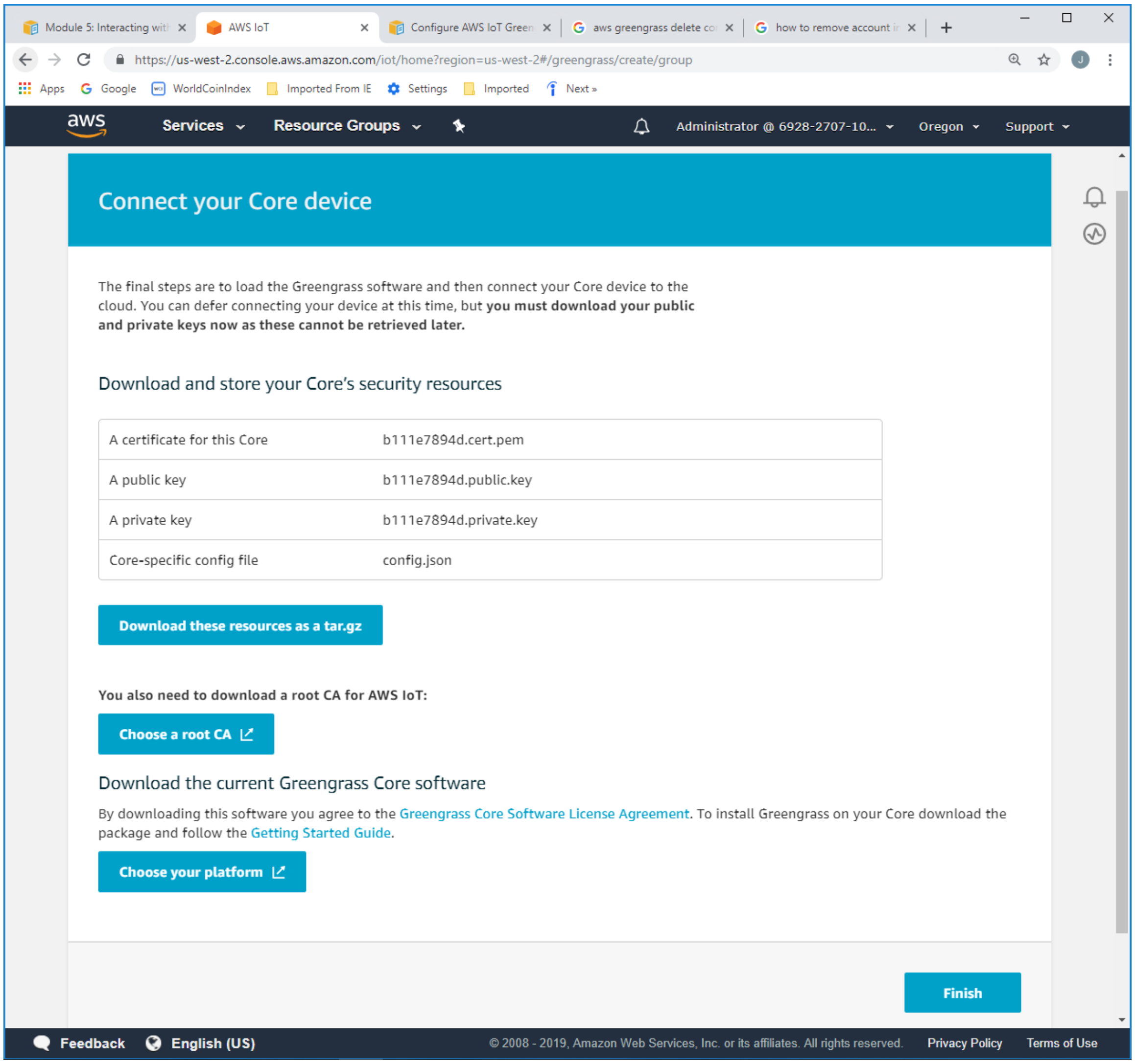
In order to speed up and simplify Group creation AWS IoT Greengrass will handle the following processes and use default settings. By proceeding to the next step, you are giving permission for us to complete the following steps.

AWS IoT Greengrass will take these actions on your behalf using default settings:

Create a new Greengrass Group in the cloud	Learn more
Provision a new Core in the IoT Registry and add to the Group	Learn more
Generate public and private key set for your Core	Learn more
Generate a new security certificate for the Core using the keys	Learn more
Attach a default security policy to the certificate	Learn more

Cancel
Create Group and Core
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Once the processing is completed, the following page will be shown:



Connect your Core device

The final steps are to load the Greengrass software and then connect your Core device to the cloud. You can defer connecting your device at this time, but **you must download your public and private keys now as these cannot be retrieved later.**

Download and store your Core's security resources

A certificate for this Core	b111e7894d.cert.pem
A public key	b111e7894d.public.key
A private key	b111e7894d.private.key
Core-specific config file	config.json

You also need to download a root CA for AWS IoT:

Download the current Greengrass Core software

By downloading this software you agree to the Greengrass Core Software License Agreement. To install Greengrass on your Core download the package and follow the Getting Started Guide.

Choose your platform
Finish
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

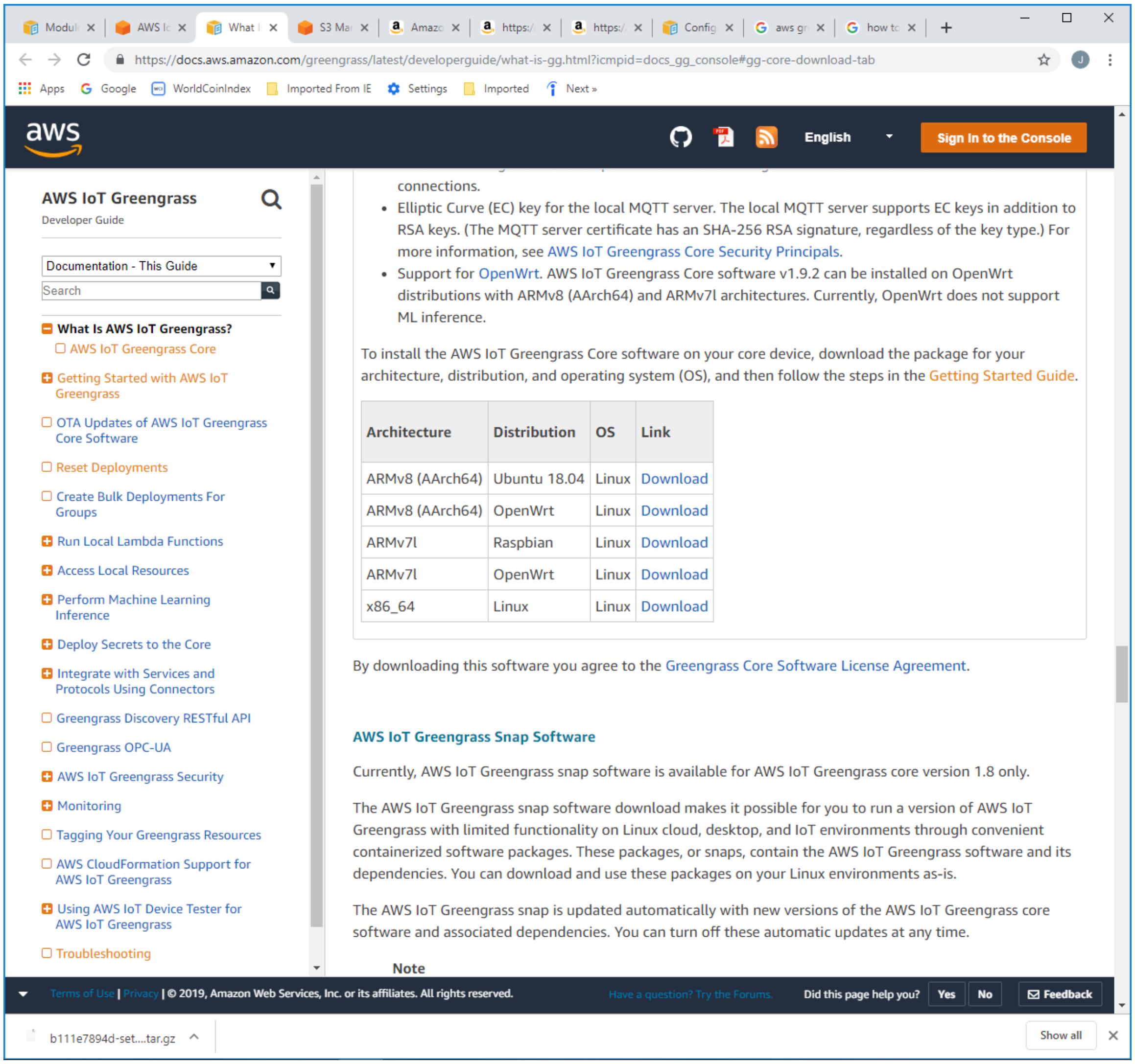
Click the ***Dowload these resources as a tar.gz*** button.

The file *b111e7894d-setup.tar.gz* was downloaded. This file will be used later on.

Don't worry about the ***Choose a root CA*** button. The CA file will be downloaded later on.

Click the ***Choose your platform*** button.

The following page will be show:


https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html?icmpid=docs_gg_console#gg-core-download-tab
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
English
Sign In to the Console
AWS IoT Greengrass
Developer Guide
Documentation - This Guide
Search
What Is AWS IoT Greengrass?
AWS IoT Greengrass Core
Getting Started with AWS IoT Greengrass
OTA Updates of AWS IoT Greengrass Core Software
Reset Deployments
Create Bulk Deployments For Groups
Run Local Lambda Functions
Access Local Resources
Perform Machine Learning Inference
Deploy Secrets to the Core
Integrate with Services and Protocols Using Connectors
Greengrass Discovery RESTful API
Greengrass OPC-UA
AWS IoT Greengrass Security
Monitoring
Tagging Your Greengrass Resources
AWS CloudFormation Support for AWS IoT Greengrass
Using AWS IoT Device Tester for AWS IoT Greengrass
Troubleshooting
Terms of Use | Privacy | © 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Have a question? Try the Forums.
Did this page help you?
Yes
No
Feedback
b111e7894d-set....tar.gz
Show all

connections.

- Elliptic Curve (EC) key for the local MQTT server. The local MQTT server supports EC keys in addition to RSA keys. (The MQTT server certificate has an SHA-256 RSA signature, regardless of the key type.) For more information, see AWS IoT Greengrass Core Security Principals.
- Support for OpenWrt. AWS IoT Greengrass Core software v1.9.2 can be installed on OpenWrt distributions with ARMv8 (AArch64) and ARMv7l architectures. Currently, OpenWrt does not support ML inference.

To install the AWS IoT Greengrass Core software on your core device, download the package for your architecture, distribution, and operating system (OS), and then follow the steps in the Getting Started Guide.

Architecture	Distribution	OS	Link
ARMv8 (AArch64)	Ubuntu 18.04	Linux	Download
ARMv8 (AArch64)	OpenWrt	Linux	Download
ARMv7l	Raspbian	Linux	Download
ARMv7l	OpenWrt	Linux	Download
x86_64	Linux	Linux	Download

By downloading this software you agree to the Greengrass Core Software License Agreement.

AWS IoT Greengrass Snap Software

Currently, AWS IoT Greengrass snap software is available for AWS IoT Greengrass core version 1.8 only.

The AWS IoT Greengrass snap software download makes it possible for you to run a version of AWS IoT Greengrass with limited functionality on Linux cloud, desktop, and IoT environments through convenient containerized software packages. These packages, or snaps, contain the AWS IoT Greengrass software and its dependencies. You can download and use these packages on your Linux environments as-is.

The AWS IoT Greengrass snap is updated automatically with new versions of the AWS IoT Greengrass core software and associated dependencies. You can turn off these automatic updates at any time.

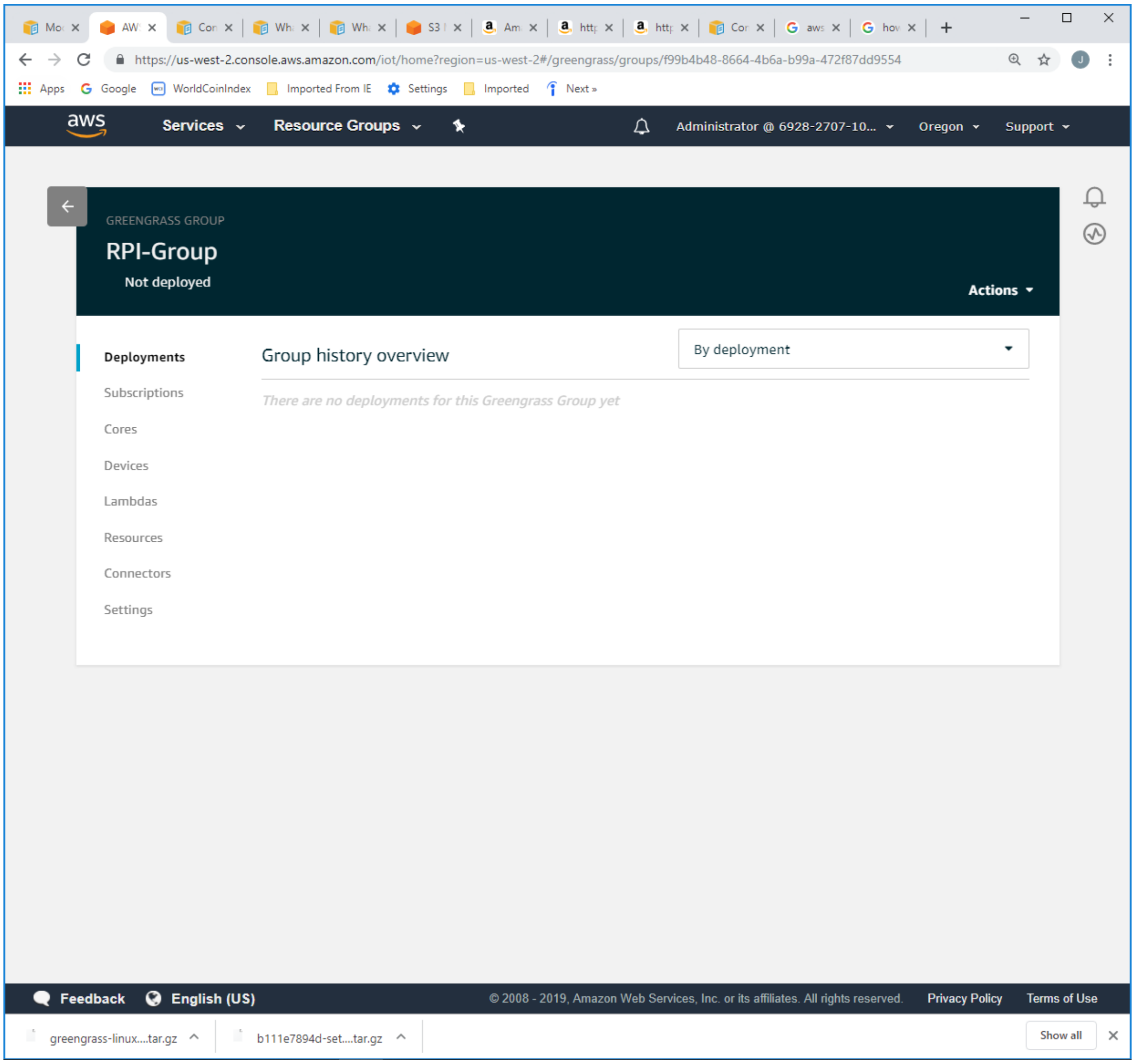
Note

Click the ***Download*** link for the Raspbian distribution.

The *greengrass-linux-armv7l-1.9.2.tar.gz* file is download.

Return to the ***Connect your Core device*** page and click on the ***Finish*** button.

The following page should appear:



https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/groups/f99b4b48-8664-4b6a-b99a-472f87dd9554

GREENGRASS GROUP

RPI-Group

Not deployed

Group history overview

By deployment

There are no deployments for this Greengrass Group yet

If you have been following this section closely, you downloaded two .gz files to your computer. We need to copy them to our Pi. You can use tools like WinSCP or PuTTY pscp. Another approach is to upload them from the computer to a S3 bucket and then download them to the Pi. This is approached that I used.

In the ***AWS Console***, click on the ***S3*** link under the ***Storage*** category.

Click on the ***Create bucket*** button.

Enter the bucket name.

Click the ***Next*** button.

Click the ***Next*** button.

Click the ***Next*** button.

Click the ***Create bucket*** button.

The new bucket should appear in the S3 buckets list.

Click on the newly created bucket.

Click on the ***Upload*** button.

Click on the ***Add files*** button.

Select the two .gz files.

Click on the ***Upload*** button.

The two files have been uploaded to the S3 bucket. Now let's download them to the Pi.

Login to the Pi console.

Open the web browser.

Login to the ***AWS Console***.

In the ***AWS Console***, click on the ***S3*** link under the ***Storage*** category.

Click on the newly created bucket.

Check the box on the first file.

Click on the ***Download*** button.

After the file has download, close the dialog box.

Check the box on the second file.

Click on the ***Download*** button.

After the file has download, close the dialog box.

Open a terminal window on the Pi.

The files were downloaded to the */home/pi/Downloads* directory.

Change to that directory.

```
cd /home/pi/Downloads
```

Decompress the software. The first will create a */greengrass* directory

```
sudo tar -xzvf <os file name> -C /
```

In my scenario the line is:

```
sudo tar -xzvf greengrass-linux-armv7l-1.9.2.tar.gz -C /
```

This will decompress the security resources in a */greengrass/config* folder.

```
sudo tar -xzvf <security file name> -C /greengrass
```

In my scenario the line is:

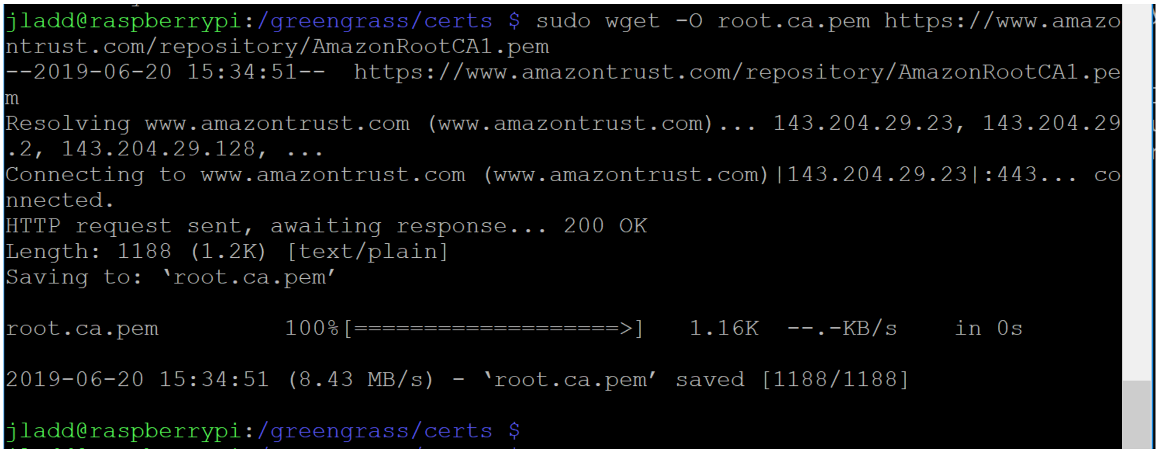
```
sudo tar -xzvf b111e7894d-setup.tar.gz -C /greengrass
```

Download the appropriate ATS root CA certificate.

```
cd /greengrass/certs/
```

(The following text should be treated as a single line command)

```
sudo wget -O root.ca.pem
      https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

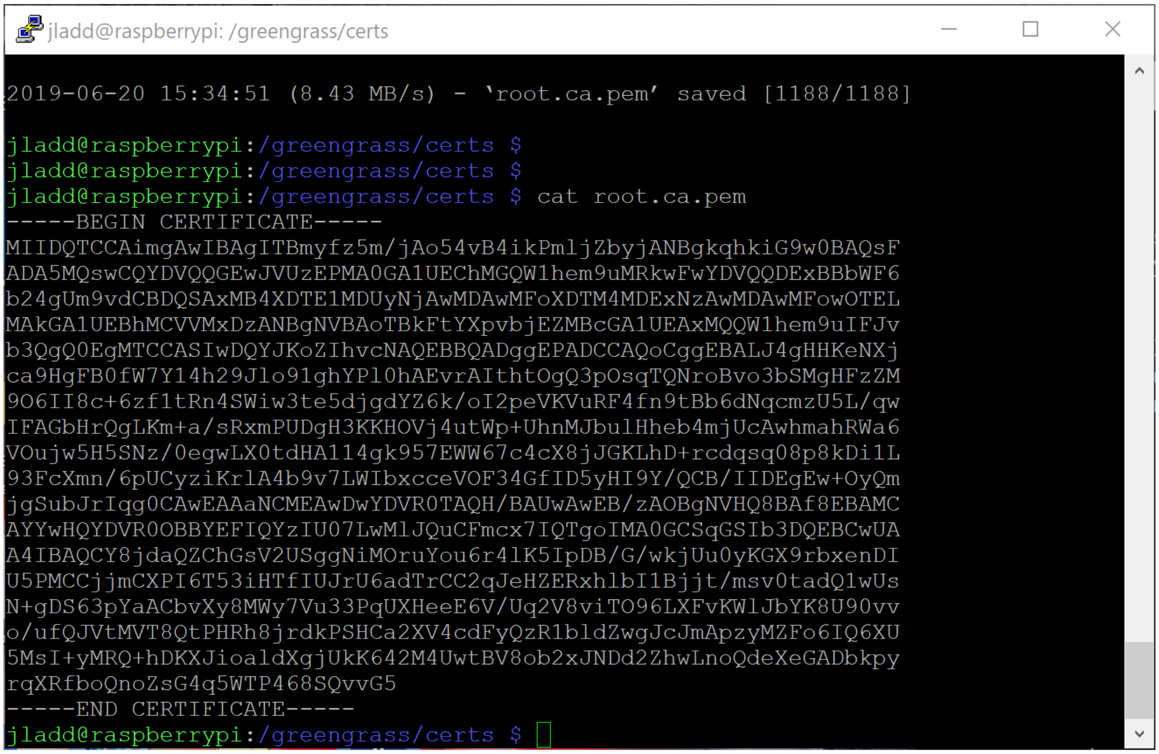


```
jladd@raspberrypi:/greengrass/certs $ sudo wget -O root.ca.pem https://www.amazo
ntrust.com/repository/AmazonRootCA1.pem
--2019-06-20 15:34:51--  https://www.amazontrust.com/repository/AmazonRootCA1.pe
m
Resolving www.amazontrust.com (www.amazontrust.com)... 143.204.29.23, 143.204.29
.2, 143.204.29.128, ...
Connecting to www.amazontrust.com (www.amazontrust.com)|143.204.29.23|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 1188 (1.2K) [text/plain]
Saving to: ‘root.ca.pem’

root.ca.pem         100%[===================>]   1.16K  --.-KB/s    in 0s

2019-06-20 15:34:51 (8.43 MB/s) - ‘root.ca.pem’ saved [1188/1188]

jladd@raspberrypi:/greengrass/certs $
```

Execute the following command to confirm:

```
cat root.ca.pem
```

The results from the command should appear as:

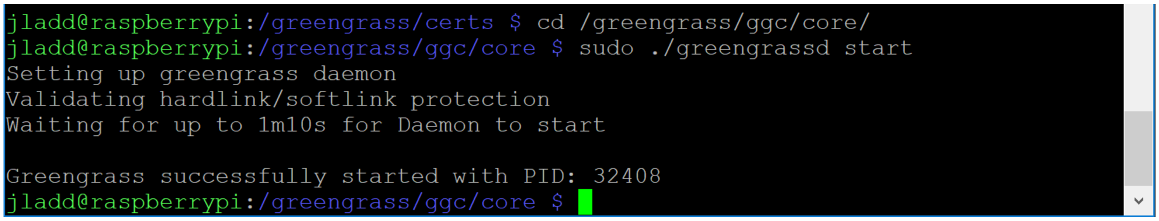


jladd@raspberrypi: /greengrass/certs

```
2019-06-20 15:34:51 (8.43 MB/s) - ‘root.ca.pem’ saved [1188/1188]

jladd@raspberrypi:/greengrass/certs $
jladd@raspberrypi:/greengrass/certs $
jladd@raspberrypi:/greengrass/certs $ cat root.ca.pem
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
jladd@raspberrypi:/greengrass/certs $
```

Start the Greengrass server.

```
cd /greengrass/ggc/core/

sudo ./greengrassd start
```



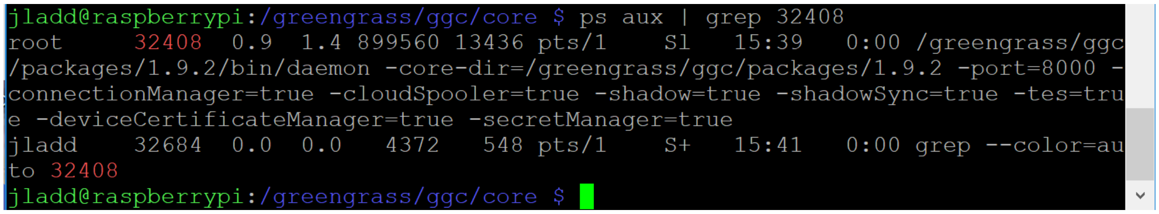
```
jladd@raspberrypi:/greengrass/certs $ cd /greengrass/ggc/core/
jladd@raspberrypi:/greengrass/ggc/core $ sudo ./greengrassd start
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 1m10s for Daemon to start

Greengrass successfully started with PID: 32408
jladd@raspberrypi:/greengrass/ggc/core $
```

To confirm that the software is running, execute the following command:

```
ps aux | grep PID-number
```

In my scenario the line is:

```
ps aux | grep 32408
```



```
jladd@raspberrypi:/greengrass/ggc/core $ ps aux | grep 32408
root      32408  0.9  1.4 899560 13436 pts/1    Sl   15:39   0:00 /greengrass/ggc
/packages/1.9.2/bin/daemon -core-dir=/greengrass/ggc/packages/1.9.2 -port=8000 -
connectionManager=true -cloudSpooler=true -shadow=true -shadowSync=true -tes=tru
e -deviceCertificateManager=true -secretManager=true
jladd     32684  0.0  0.0   4372   548 pts/1    S+   15:41   0:00 grep --color=au
to 32408
jladd@raspberrypi:/greengrass/ggc/core $ 
```

Deploying a Lambda Function to the Pi

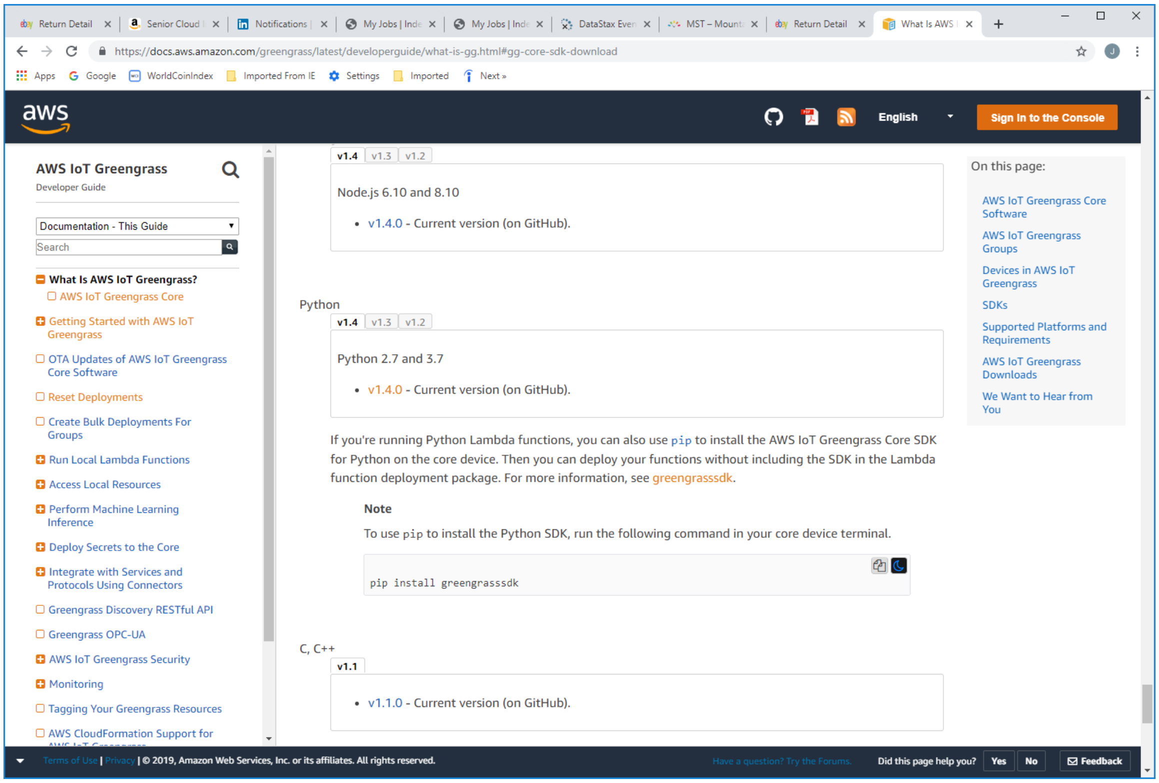
This section deploys a Python Lambda function to the Pi. The Greengrass SDK and a Python script file will be combined into a zip file and then deployed to the Pi via the Greengrass console.

First, let’s download the SDK to your computer.

Navigate to the download page located at the following link:

https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html#gg-core-sdk-download

Navigate to the section of the page that looks like the following:


Return Detail
Senior Cloud
Notifications
My Jobs | Inde
My Jobs | Inde
DataStax Even
MST – Mount
Return Detail
What Is AWS
https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html#gg-core-sdk-download
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
English
Sign In to the Console
AWS IoT Greengrass
Developer Guide
Documentation - This Guide
Search
What Is AWS IoT Greengrass?
AWS IoT Greengrass Core
Getting Started with AWS IoT Greengrass
OTA Updates of AWS IoT Greengrass Core Software
Reset Deployments
Create Bulk Deployments For Groups
Run Local Lambda Functions
Access Local Resources
Perform Machine Learning Inference
Deploy Secrets to the Core
Integrate with Services and Protocols Using Connectors
Greengrass Discovery RESTful API
Greengrass OPC-UA
AWS IoT Greengrass Security
Monitoring
Tagging Your Greengrass Resources
AWS CloudFormation Support for AWS IoT Greengrass
On this page:
AWS IoT Greengrass Core Software
AWS IoT Greengrass Groups
Devices in AWS IoT Greengrass
SDKs
Supported Platforms and Requirements
AWS IoT Greengrass Downloads
We Want to Hear from You
Terms of Use | Privacy | © 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Have a question? Try the Forums.
Did this page help you?
Yes
No
Feedback

v1.4 | v1.3 | v1.2

Node.js 6.10 and 8.10

- v1.4.0 - Current version (on GitHub).

Python

v1.4 | v1.3 | v1.2

Python 2.7 and 3.7

- v1.4.0 - Current version (on GitHub).

If you're running Python Lambda functions, you can also use `pip` to install the AWS IoT Greengrass Core SDK for Python on the core device. Then you can deploy your functions without including the SDK in the Lambda function deployment package. For more information, see greengrasssdk.

Note

To use `pip` to install the Python SDK, run the following command in your core device terminal.

```
pip install greengrasssdk
```

C, C++

v1.1

- v1.1.0 - Current version (on GitHub).

Click on the *v1.4.0* link in the *Python* section.

On the GitHub page, download the zip file.

The *aws-greengrass-core-sdk-python-master.zip* file is downloaded.

Extract all of the files in the zip file.

The top-level folder is *aws-greengrass-core-sdk-python-master* with subfolders of *examples* and *greengrasssdk*.

The first example file is that we will use is the */examples/greengrassHelloWorld.py.*

Here is the code:

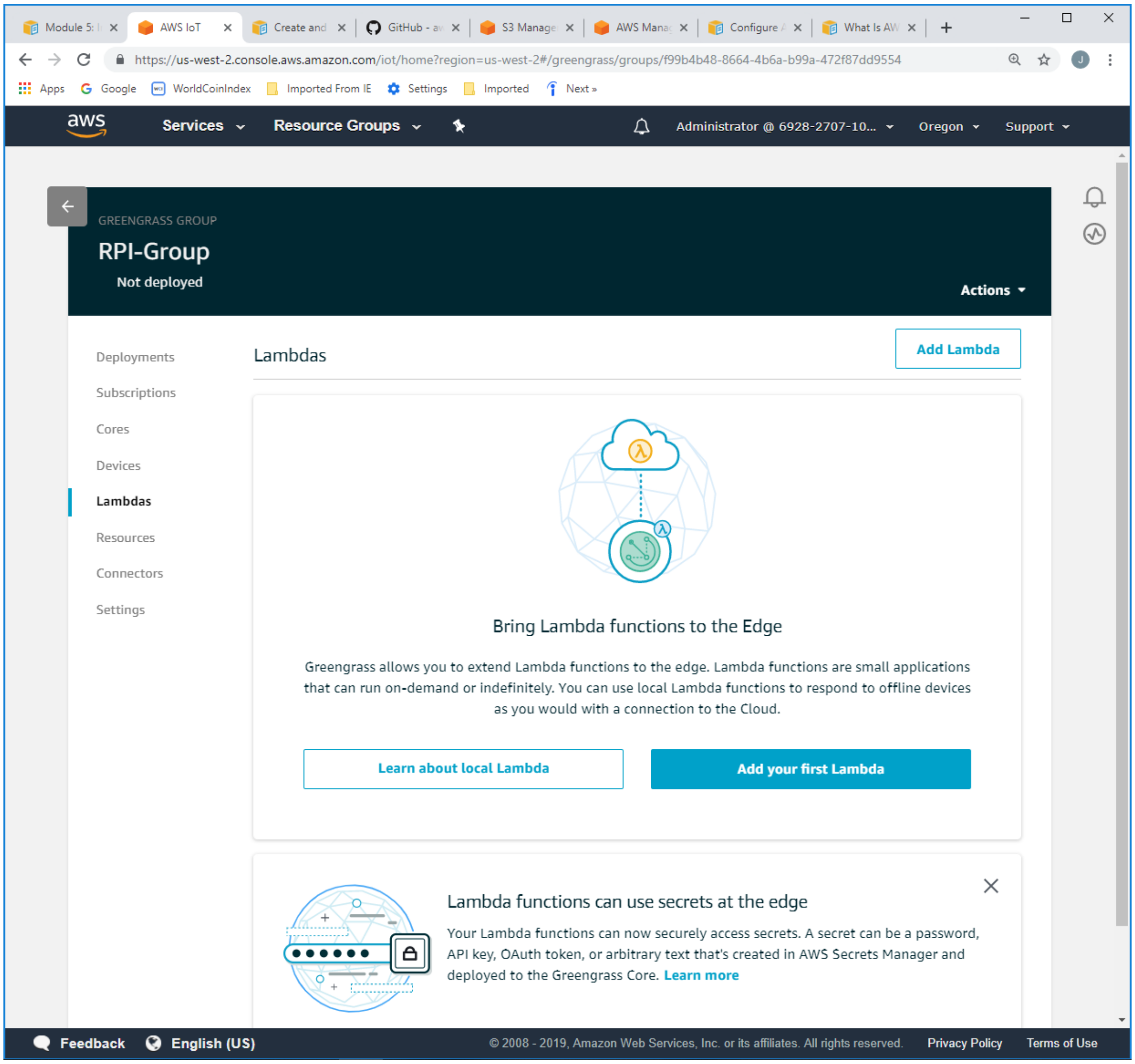
```
#
# Copyright 2010-2017 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#

# greengrassHelloWorld.py
# Demonstrates a simple publish to a topic using Greengrass core sdk
# This lambda function will retrieve underlying platform information and send
# a hello world message along with the platform information to the topic
# 'hello/world'. The function will sleep for five seconds, then repeat.
# Since the function is long-lived it will run forever when deployed to a
# Greengrass core.  The handler will NOT be invoked in our example since
# the we are executing an infinite loop.

import greengrasssdk
import platform
from threading import Timer

# Creating a greengrass core sdk client
client = greengrasssdk.client('iot-data')

# Retrieving platform information to send from Greengrass Core
my_platform = platform.platform()

# When deployed to a Greengrass core, this code will be executed immediately
# as a long-lived lambda function.  The code will enter the infinite while
# loop below.
# If you execute a 'test' on the Lambda Console, this test will fail by
# hitting the execution timeout of three seconds.  This is expected as
# this function never returns a result.

def greengrass_hello_world_run():
    if not my_platform:
        client.publish(
            topic='hello/world',
            payload='Hello world! Sent from Greengrass Core.')
    else:
        client.publish(
            topic='hello/world',
            payload='Hello world! Sent from '
                    'Greengrass Core running on platform: {}'
                    .format(my_platform))

    # Asynchronously schedule this function to be run again in 5 seconds
    Timer(5, greengrass_hello_world_run).start()

# Start executing the function above
greengrass_hello_world_run()
```

```
# This is a dummy handler and will not be invoked
# Instead the code above will be executed in an infinite loop for our example
def function_handler(event, context):
    return
```

Create a zip file (`hello_world_python_lambda.zip`) that includes the */examples/greengrassHelloWorld.py* file and the *greengrasssdk* folder.

Navigate back to your Greengrass group (i.e. in my example this is the **RPI-Group**) page in the ***AWS Console***.

Click on the ***Lambdas*** link.



GREENGRASS GROUP

RPI-Group

Not deployed

Lambdas

Bring Lambda functions to the Edge

Greengrass allows you to extend Lambda functions to the edge. Lambda functions are small applications that can run on-demand or indefinitely. You can use local Lambda functions to respond to offline devices as you would with a connection to the Cloud.

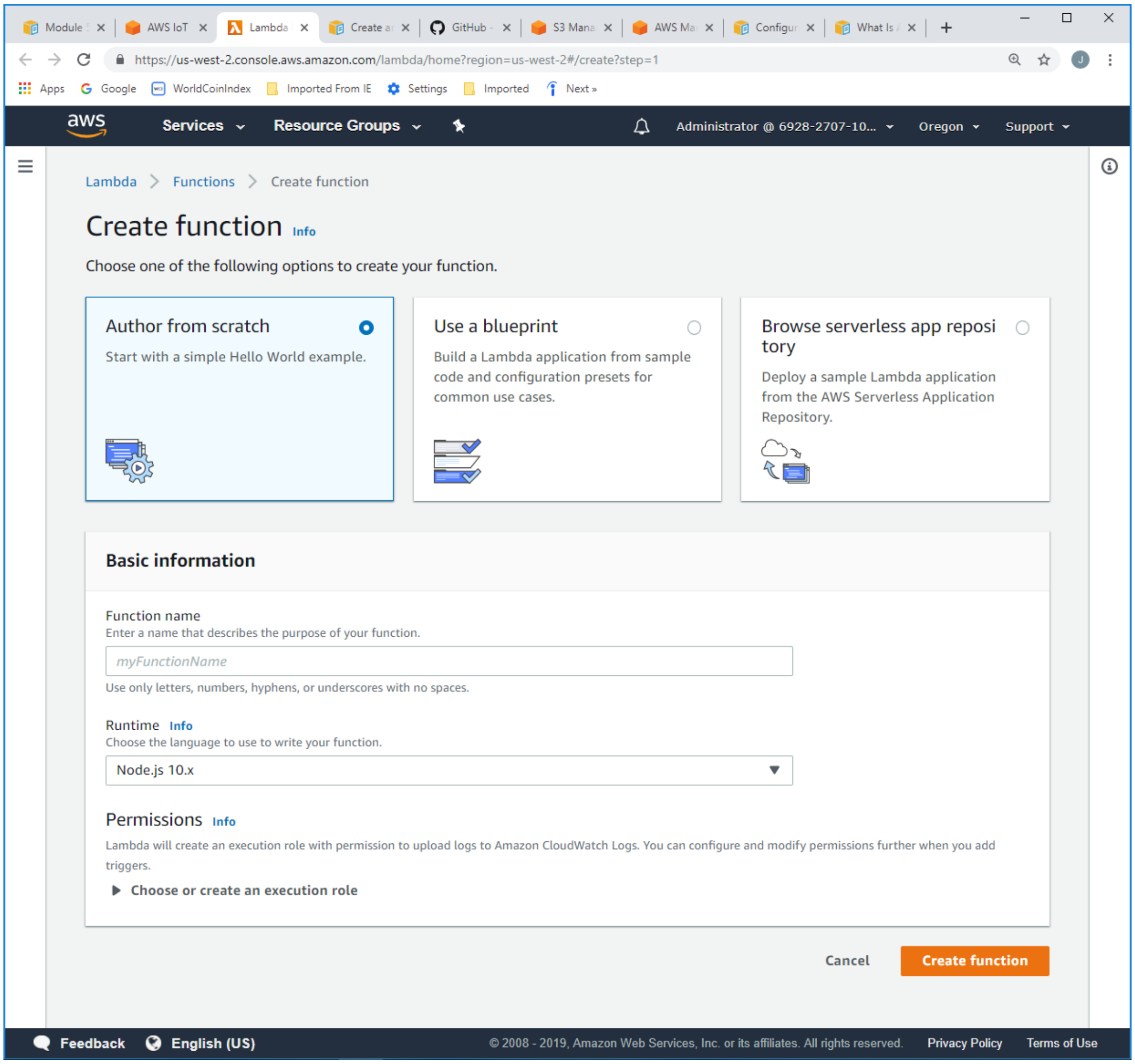
Lambda functions can use secrets at the edge

Your Lambda functions can now securely access secrets. A secret can be a password, API key, OAuth token, or arbitrary text that's created in AWS Secrets Manager and deployed to the Greengrass Core. Learn more

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Click on the ***Add Lambda*** button.

Click on the ***Create new Lambda*** button.



Create function

Choose one of the following options to create your function.

● Author from scratch
Start with a simple Hello World example.

○ Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

○ Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
myFunctionName
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.
Node.js 10.x

Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

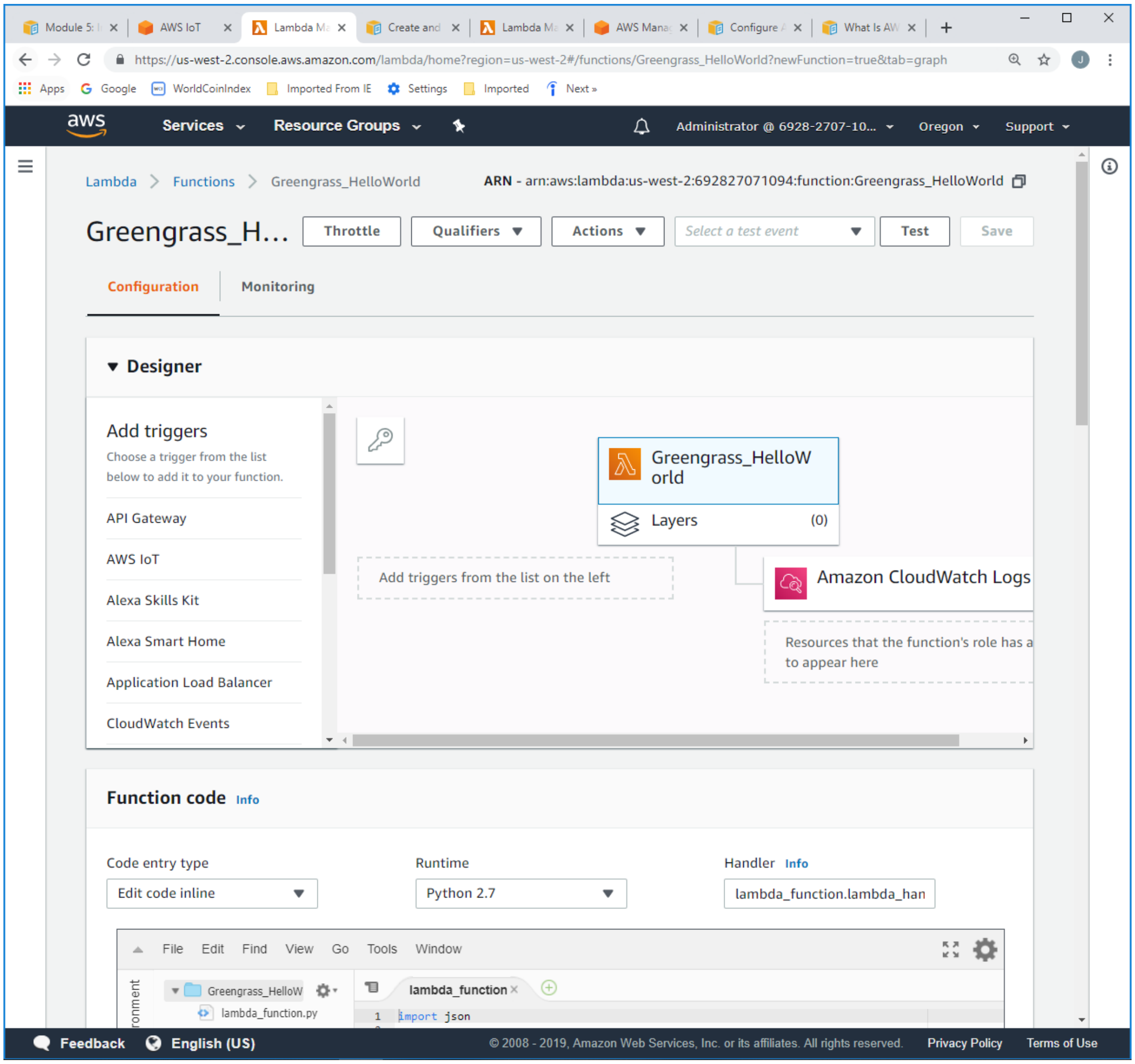
▶ Choose or create an execution role

Select the ***Author from scratch*** option.

In the *Function Name* field, enter "Greengrass_HelloWorld"

In the *Runtime* field, select **Python 2.7**

Click on the ***Create function*** button.



https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#/functions/Greengrass_HelloWorld?newFunction=true&tab=graph

ARN - arn:aws:lambda:us-west-2:692827071094:function:Greengrass_HelloWorld

Code entry type: Edit code inline

Runtime: Python 2.7

Handler: lambda_function.lambda_han

```
import json
```

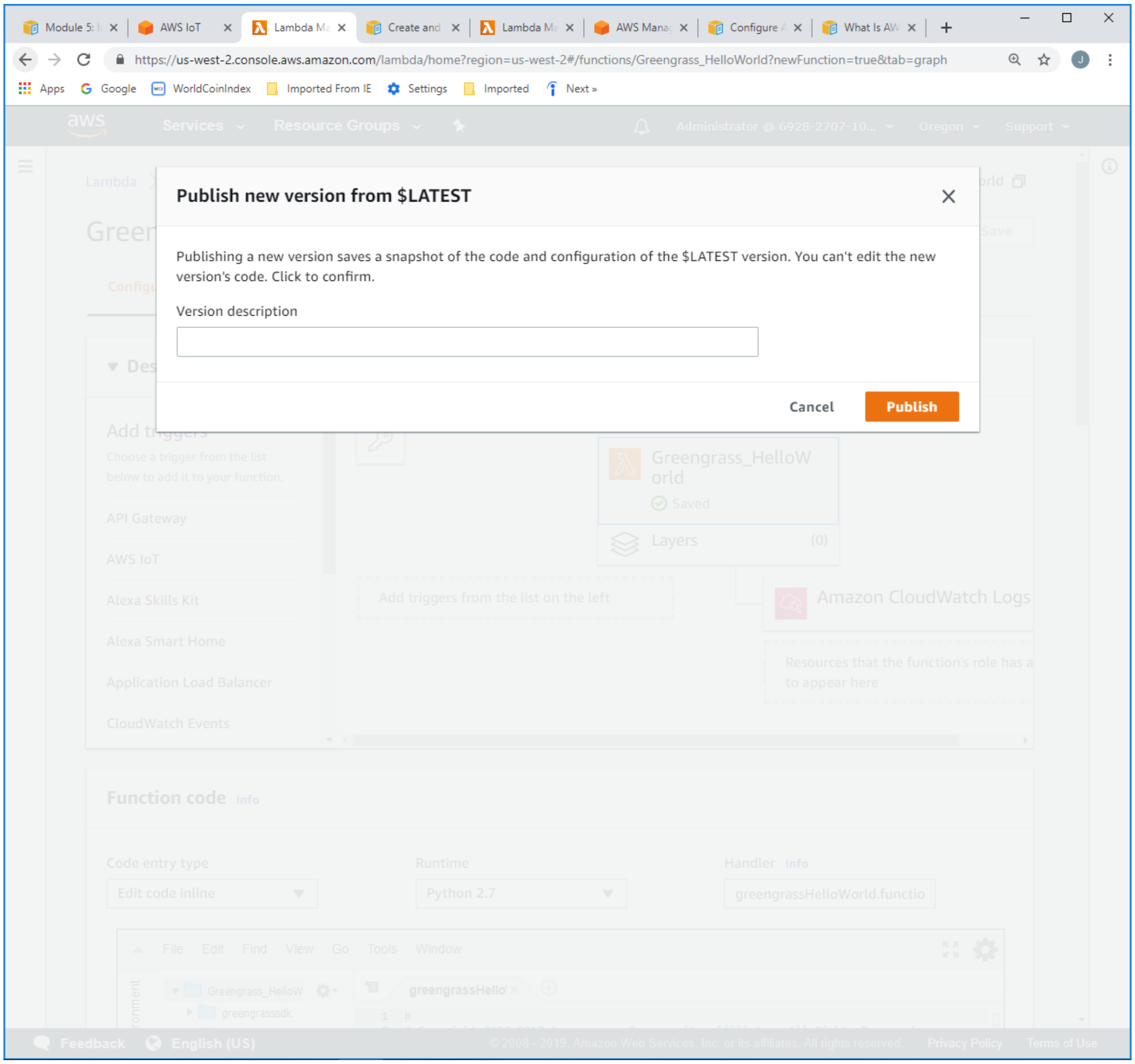
In the *Function code* section, select the ***Upload a .zip file*** option in the *Code entry type* field.

In the *Handler* field, enter “`greengrassHelloWorld.function_handler`”

Click on the ***Upload*** button and upload the *hello_world_python_lambda.zip* file.

Click on the ***Save*** button.

In the *Actions* field at the top of the page, select the ***Publish new version*** option.



Publish new version from $LATEST

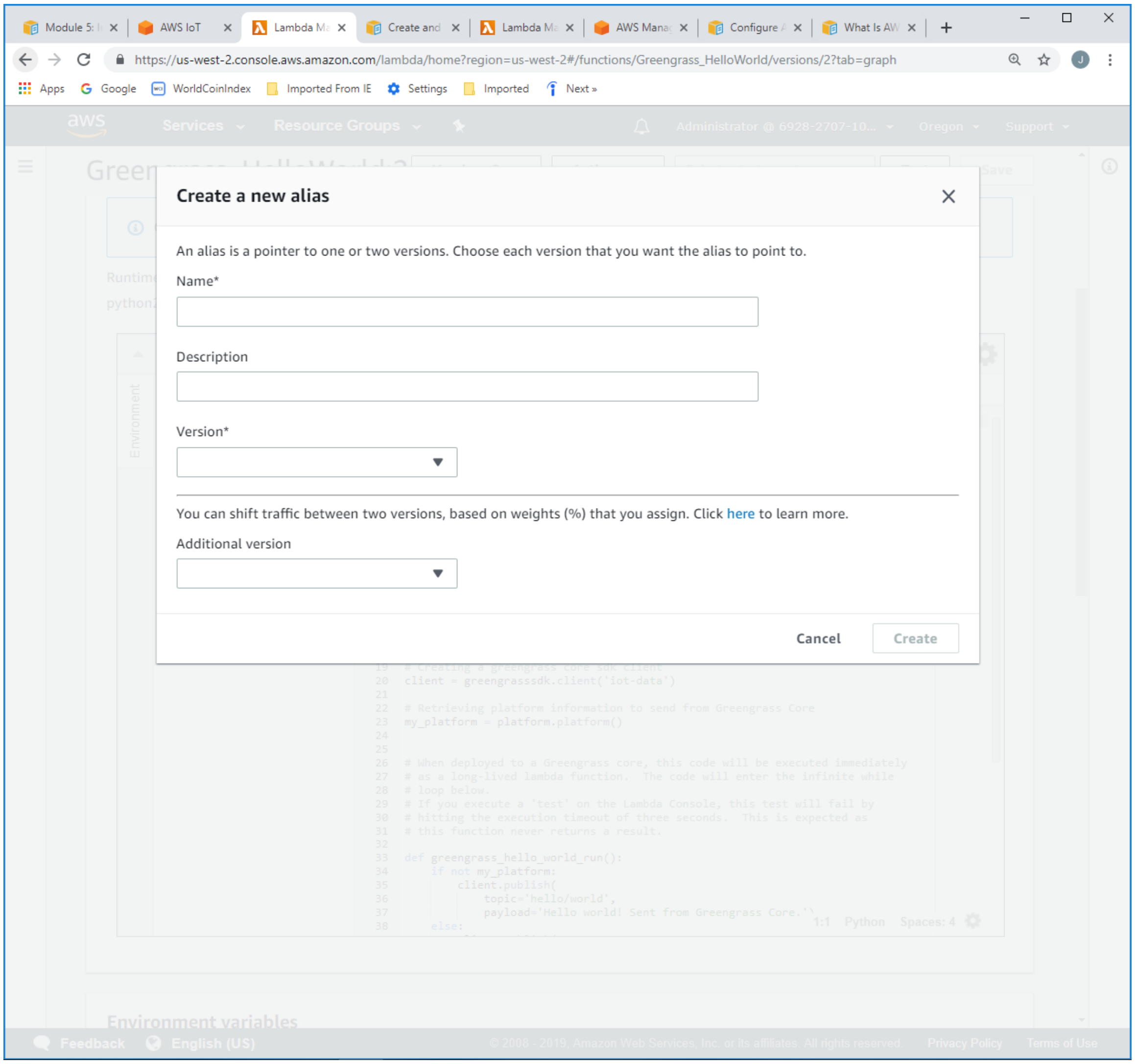
Publishing a new version saves a snapshot of the code and configuration of the $LATEST version. You can't edit the new version's code. Click to confirm.

Version description

In the *Version description* field, enter “First version”.

Click on the ***Publish*** button.

In the *Actions* field at the top of the page, select the ***Create alias*** option.


https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#/functions/Greengrass_HelloWorld/versions/2?tab=graph
Cancel
Create

Create a new alias

An alias is a pointer to one or two versions. Choose each version that you want the alias to point to.

Name*

Description

Version*

You can shift traffic between two versions, based on weights (%) that you assign. Click here to learn more.

Additional version

In the *Name* field, enter “GG_HelloWorld”.

In the *Version* field, enter “1”.

Click the ***Create*** button.

Navigate to the ***AWS Console***.

Select the ***IoT Greengrass*** link.

On the left side of the page, select ***Groups*** in the Greengrass section.

Click on the group that was created earlier.

Click on the ***Lambdas*** link.

Click on the ***Add Lambda*** button.

Click on the ***Use existing Lambda*** button.

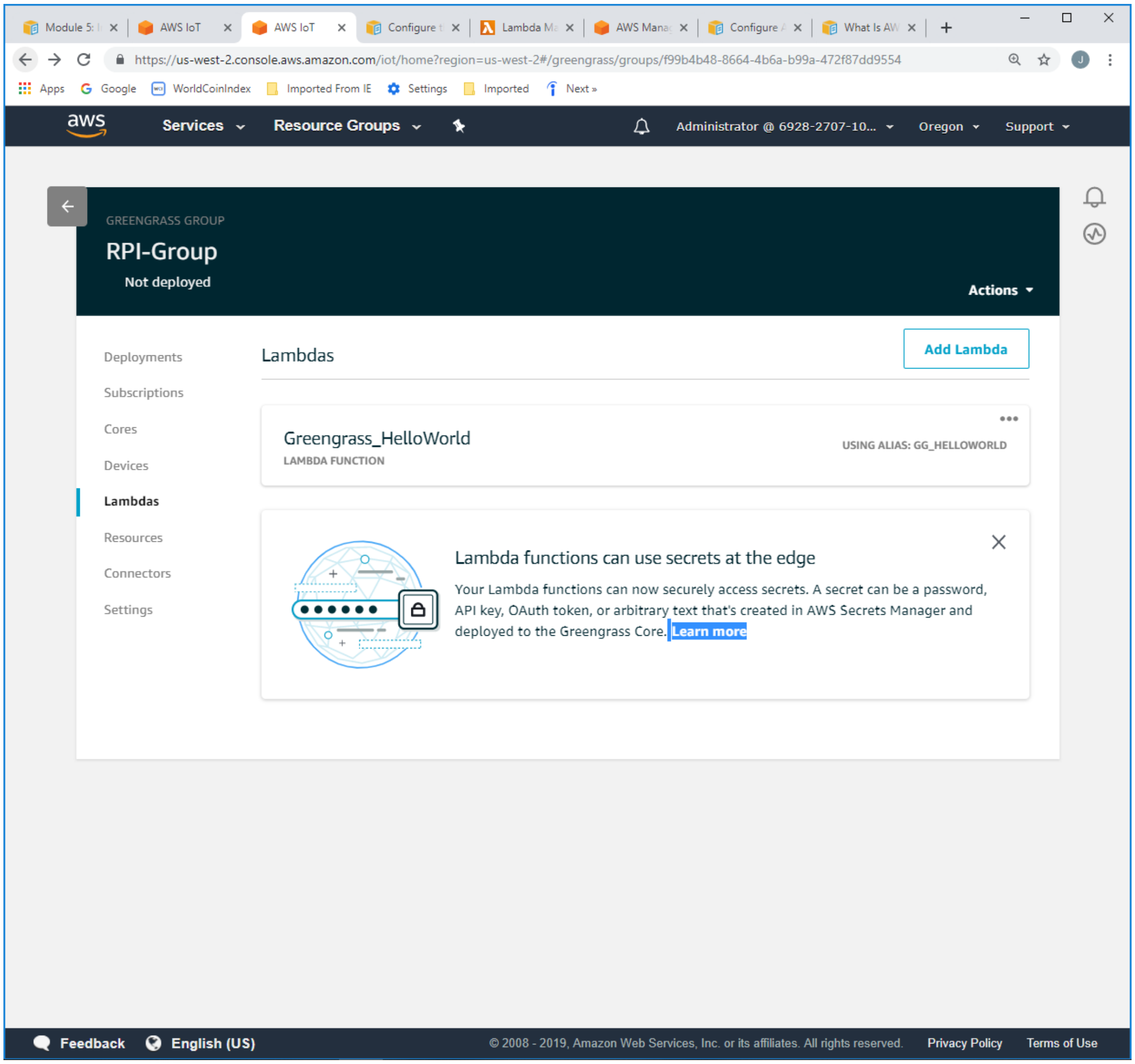
Select the ***Greengrass_HelloWorld*** item.

Click on the ***Next*** button.

Select the ***Alias:GG_HelloWorld*** item.

Click on the ***Finish*** button.

The list of Lambdas for the Group should appear.



GREENGRASS GROUP

RPI-Group

Not deployed

Lambdas

Greengrass_HelloWorld

LAMBDA FUNCTION

USING ALIAS: GG_HELLOWORLD

Lambda functions can use secrets at the edge

Your Lambda functions can now securely access secrets. A secret can be a password, API key, OAuth token, or arbitrary text that's created in AWS Secrets Manager and deployed to the Greengrass Core. Learn more

Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Edit the *Greengrass_HelloWorld* function.

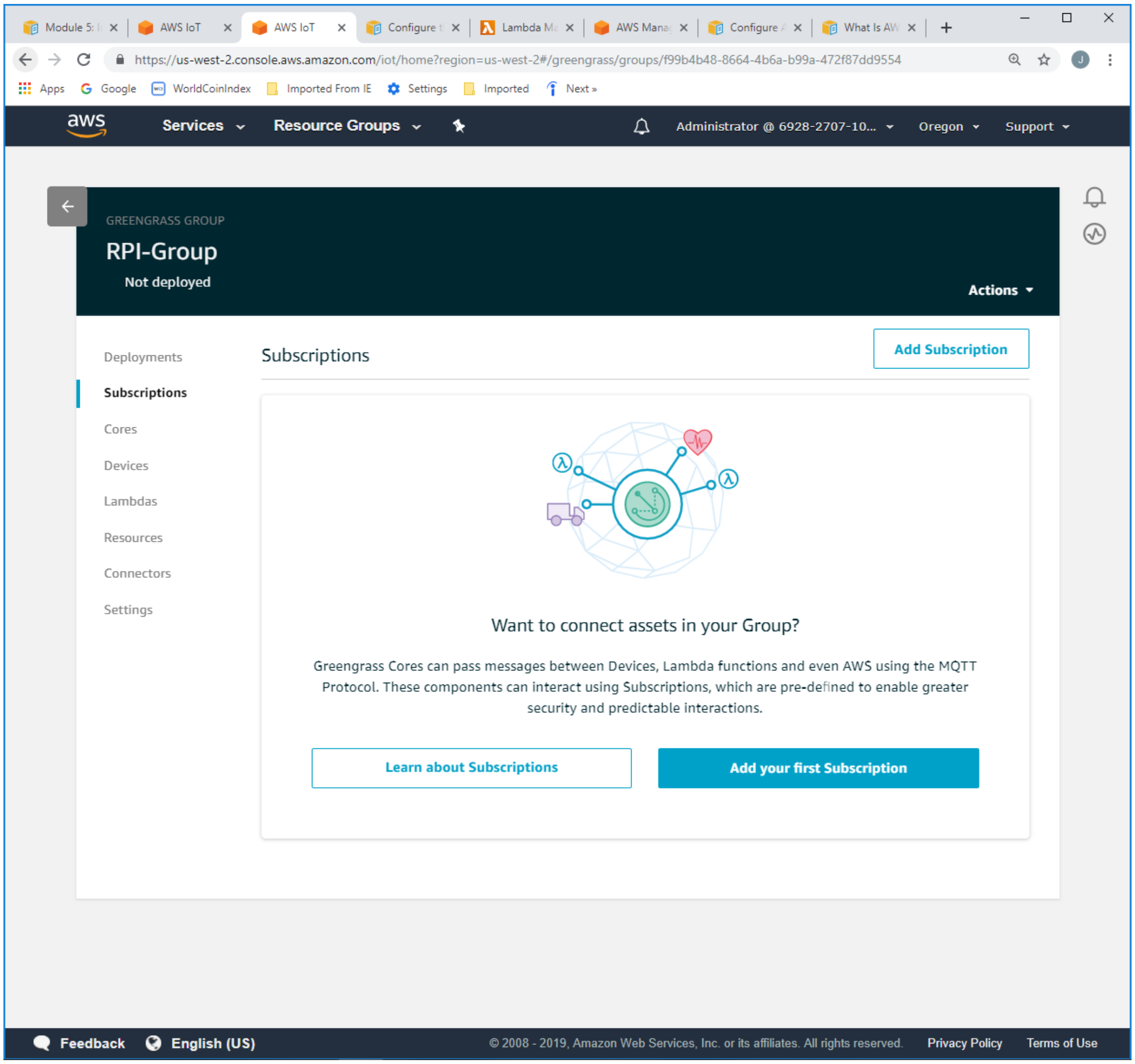
Set the *Timeout* field to 25 seconds.

Set the *Lambda lifecycle* to ***Make this function long-lived and keep it running indefinitely***.

Click the ***Update*** button.

Now create a subscription within the group.

Click on the ***Subscriptions*** link.


https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/groups/f99b4b48-8664-4b6a-b99a-472f87dd9554
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support
GREENGRASS GROUP
RPI-Group
Not deployed
Actions
Deployments
Subscriptions
Cores
Devices
Lambdas
Resources
Connectors
Settings
Subscriptions
Add Subscription

Want to connect assets in your Group?

Greengrass Cores can pass messages between Devices, Lambda functions and even AWS using the MQTT Protocol. These components can interact using Subscriptions, which are pre-defined to enable greater security and predictable interactions.

Learn about Subscriptions
Add your first Subscription
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

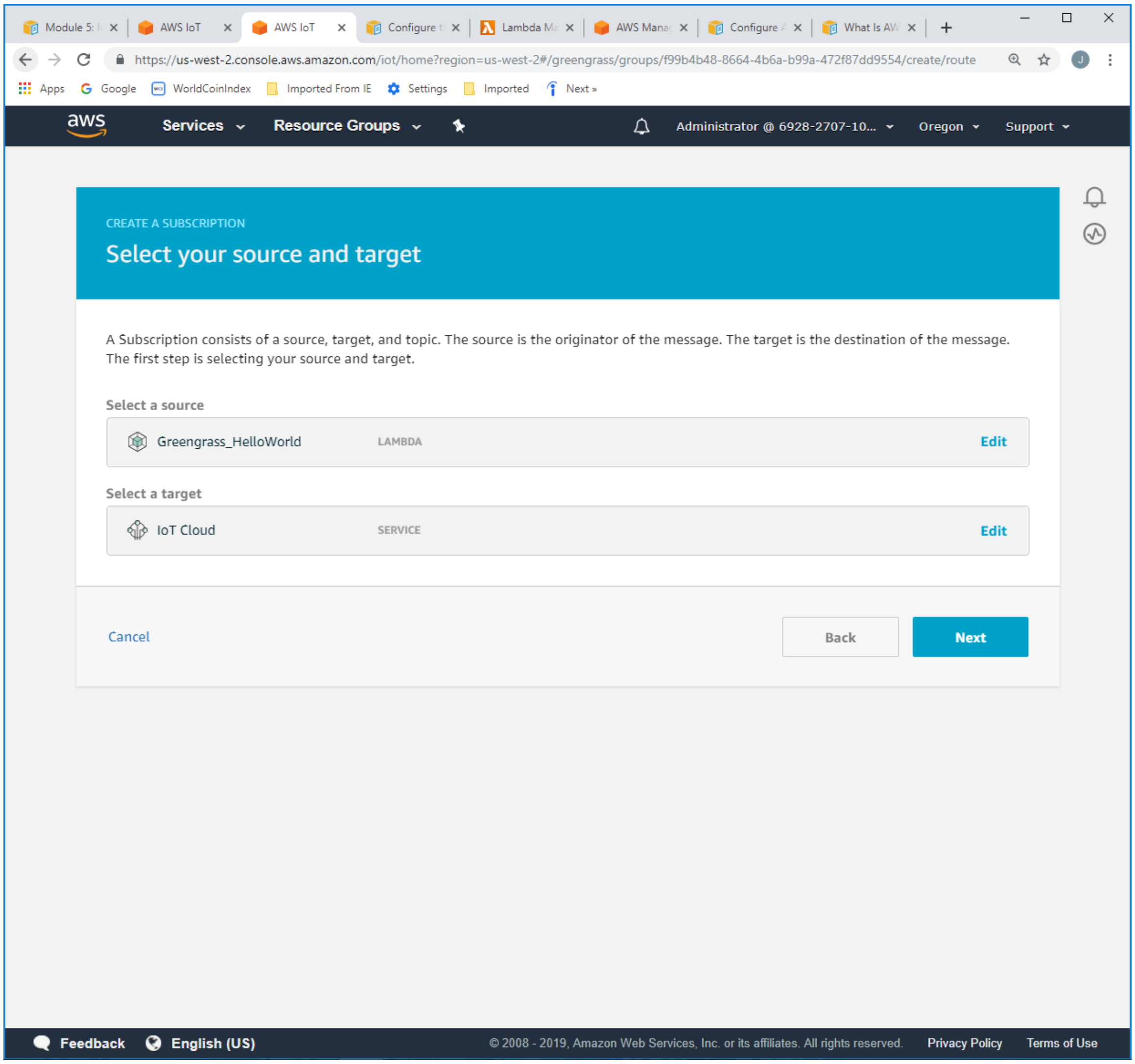
Click on the ***Add Subscription*** button.

In the *Select a source* section, click on the ***Lambdas*** link.

Select the *Greengrass_HelloWorld* item.

In the *Select a target* section, click on the ***Services*** link.

Select the ***IoT Cloud*** item.


Module 5:
AWS IoT
AWS IoT
Configure
Lambda Ma
AWS Mana
Configure
What Is AW
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/groups/f99b4b48-8664-4b6a-b99a-472f87dd9554/create/route
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

CREATE A SUBSCRIPTION

Select your source and target

A Subscription consists of a source, target, and topic. The source is the originator of the message. The target is the destination of the message. The first step is selecting your source and target.

Select a source

Greengrass_HelloWorld LAMBDA Edit

Select a target

IoT Cloud SERVICE Edit

Cancel Back Next

Click on the ***Next*** button.

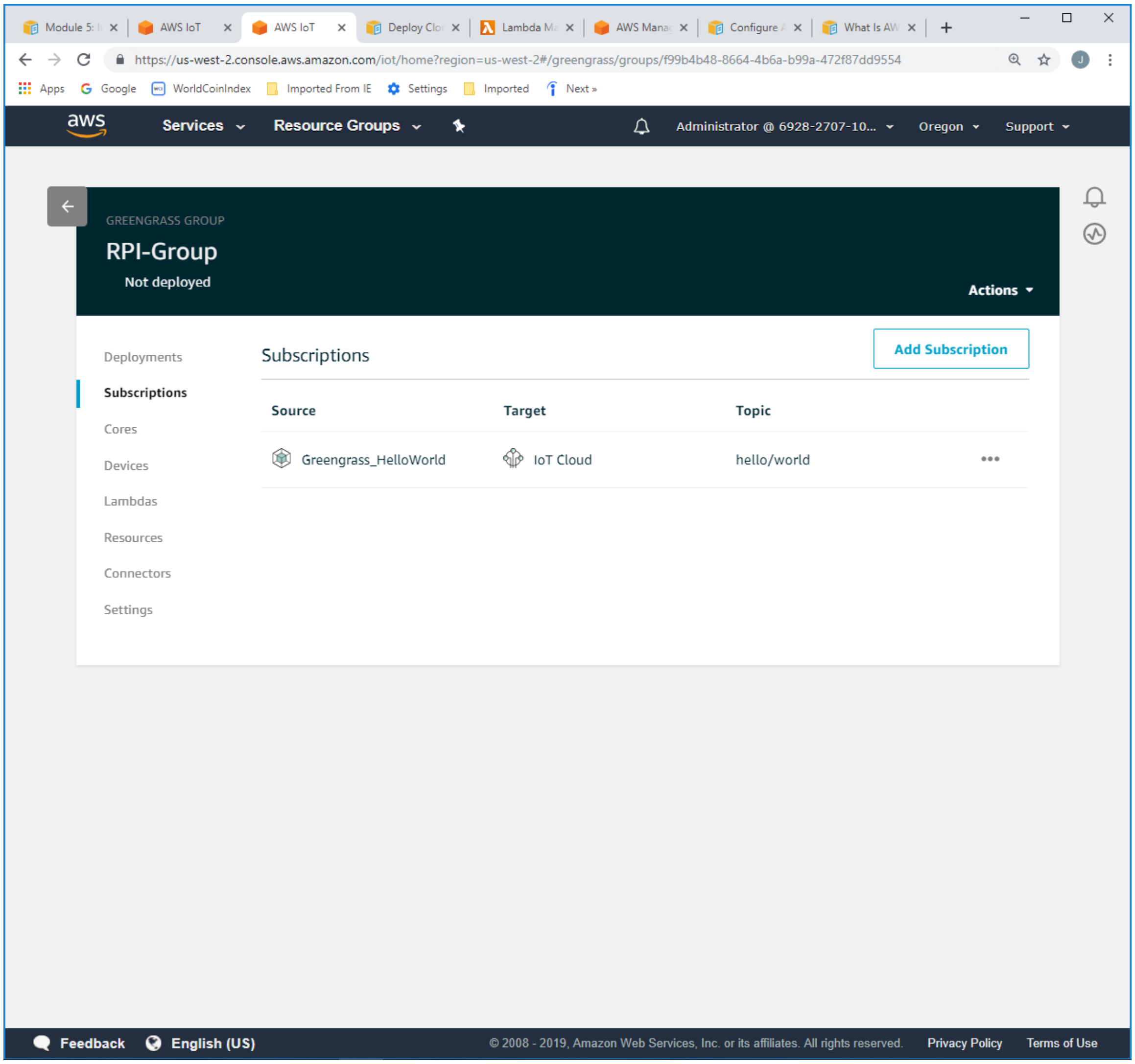
In the *Topic filter* field, enter “hello/world”

Click on the ***Next*** button.

Click on the ***Finish*** button.

Deploy the Lambda function and subscription configuration to the Pi.

Navigate back to the Group page.



GREENGRASS GROUP

RPI-Group

Not deployed

Actions ▾

Subscriptions

Source	Target	Topic
Greengrass_HelloWorld	IoT Cloud	hello/world

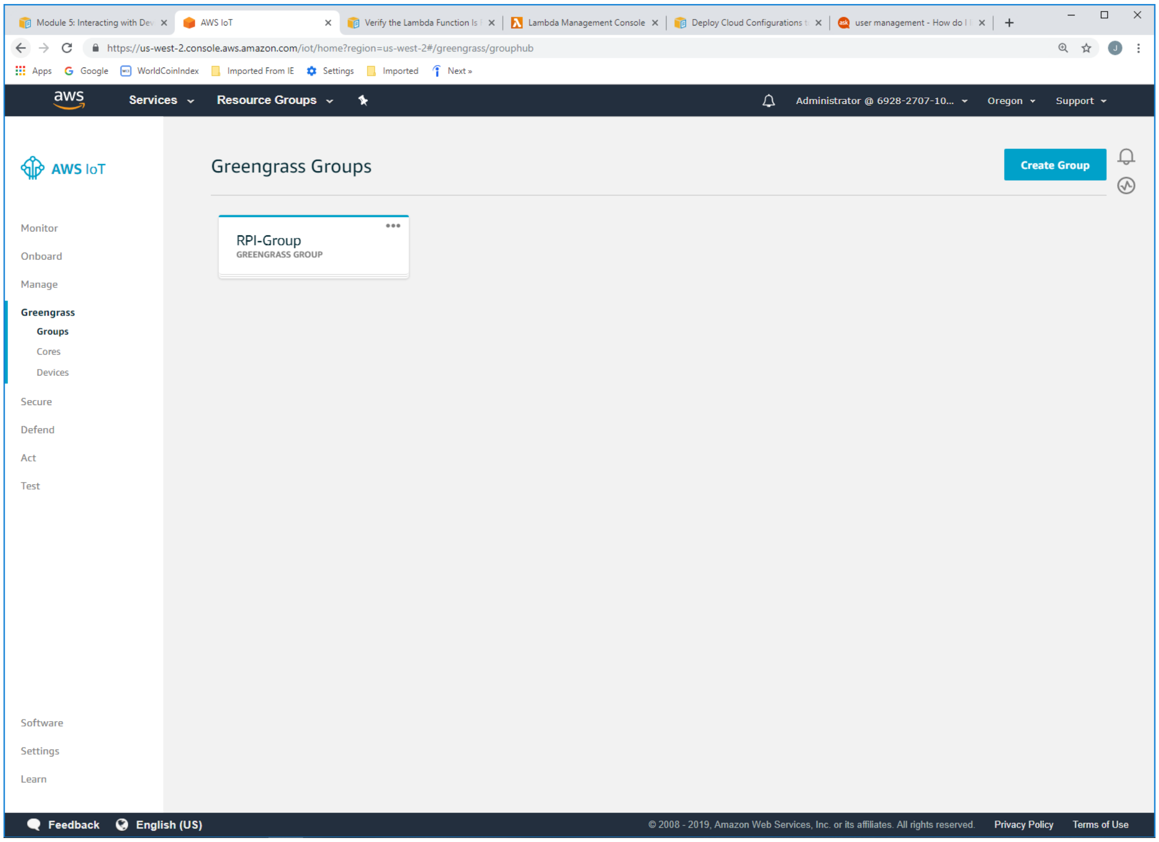
In the *Actions* field, select the ***Deploy*** option.

Click on the ***Automatic detection*** button.

After a few seconds, the status for the deployment will change to ***Successfully completed***.

Now let’s test the lambda function.

Navigate to the main AWS IoT page.


Module 5: Interacting with Dev
AWS IoT
Verify the Lambda Function Is
Lambda Management Console
Deploy Cloud Configurations t
user management - How do I
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/grouphub
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support
AWS IoT
Monitor
Onboard
Manage
Greengrass
Groups
Cores
Devices
Secure
Defend
Act
Test
Software
Settings
Learn
Greengrass Groups
Create Group
RPI-Group
GREENGRASS GROUP
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

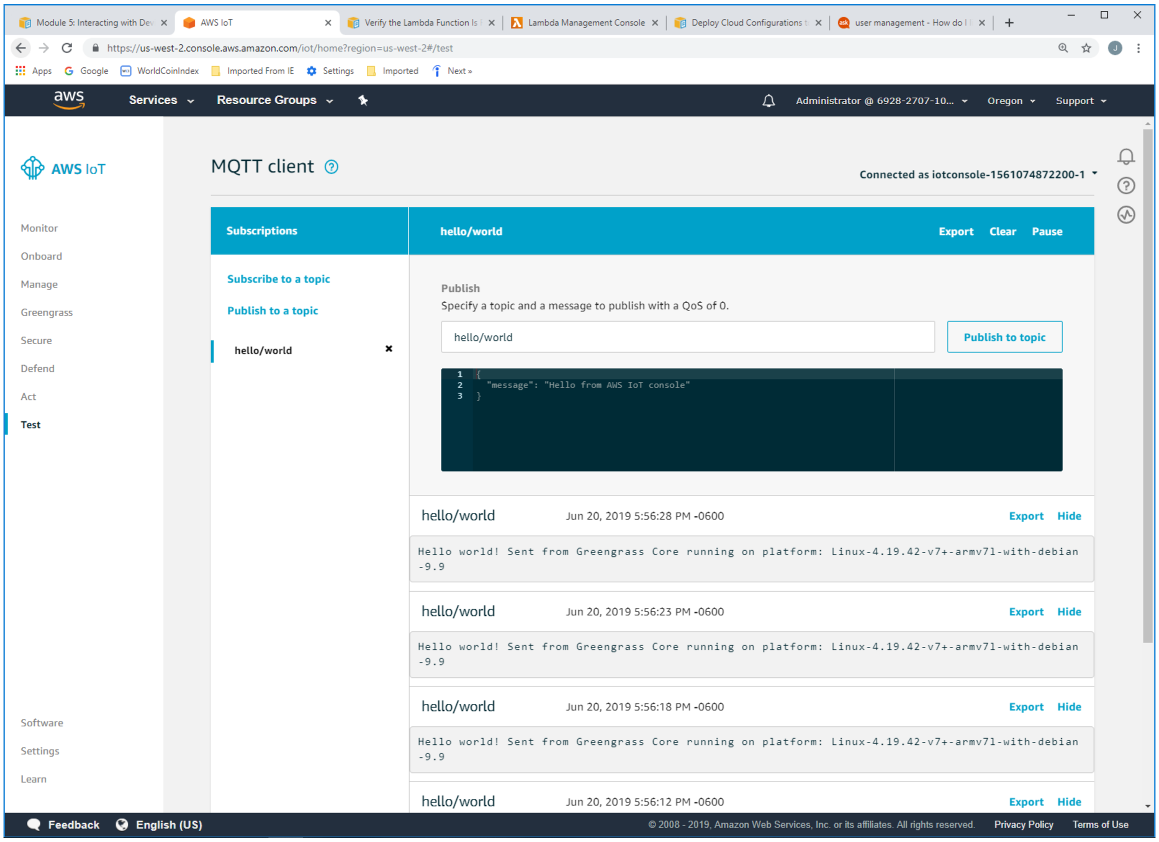
Click on the ***Test*** link.

In the *Subscription topic* field, enter "hello/world"

Set the *MQTT payload display* to ***Display payloads as strings***

Click on the ***Subscribe to topic*** button.

You will see the messages sent from the Pi (see the screenshot below).


Module 5: Interacting with Dev
AWS IoT
Verify the Lambda Function Is
Lambda Management Console
Deploy Cloud Configurations
user management - How do I
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/test
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support
AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Defend
Act
Test
Software
Settings
Learn
MQTT client
Connected as iotconsole-1561074872200-1
Subscriptions
Subscribe to a topic
Publish to a topic
hello/world
Export
Clear
Pause
Publish to topic
Hide
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Publish

Specify a topic and a message to publish with a QoS of 0.

hello/world

```
{
  "message": "Hello from AWS IoT console"
}
```

hello/world Jun 20, 2019 5:56:28 PM -0600

```
Hello world! Sent from Greengrass Core running on platform: Linux-4.19.42-v7+-armv7l-with-debian
-9.9
```

hello/world Jun 20, 2019 5:56:23 PM -0600

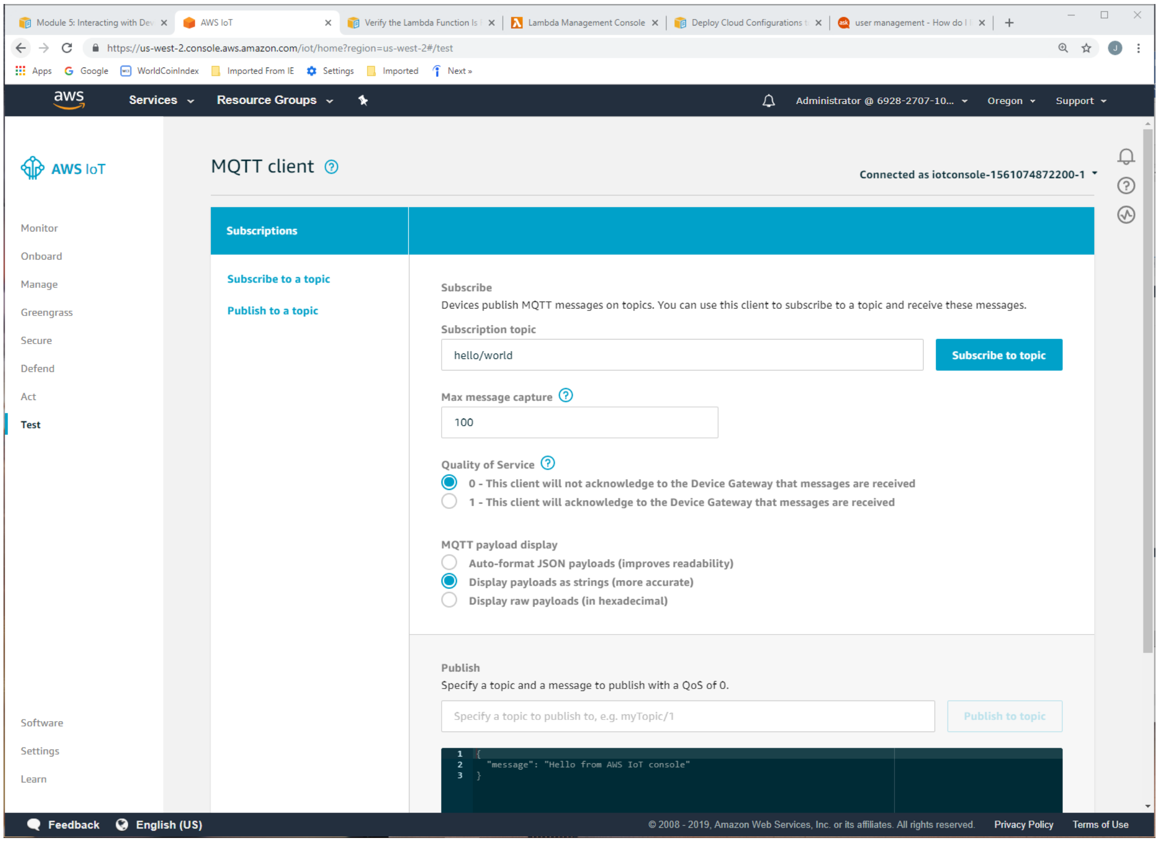
```
Hello world! Sent from Greengrass Core running on platform: Linux-4.19.42-v7+-armv7l-with-debian
-9.9
```

hello/world Jun 20, 2019 5:56:18 PM -0600

```
Hello world! Sent from Greengrass Core running on platform: Linux-4.19.42-v7+-armv7l-with-debian
-9.9
```

hello/world Jun 20, 2019 5:56:12 PM -0600

Terminate the subscription by click on the ***x*** by the ***hello/world*** label.


Module 5: Interacting with De
AWS IoT
Verify the Lambda Function Is
Lambda Management Console
Deploy Cloud Configurations
user management - How do I
https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/test
Apps
Google
WorldCoinIndex
Imported From IE
Settings
Imported
Next »
aws
Services
Resource Groups
Administrator @ 6928-2707-10...
Oregon
Support
AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Defend
Act
Test
Software
Settings
Learn
Subscriptions
Subscribe to a topic
Publish to a topic
Subscribe to topic
Publish to topic
Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

MQTT client

Connected as iotconsole-1561074872200-1

Subscribe

Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic: hello/world

Max message capture: 100

Quality of Service

● 0 - This client will not acknowledge to the Device Gateway that messages are received

○ 1 - This client will acknowledge to the Device Gateway that messages are received

MQTT payload display

○ Auto-format JSON payloads (improves readability)

● Display payloads as strings (more accurate)

○ Display raw payloads (in hexadecimal)

Publish

Specify a topic and a message to publish with a QoS of 0.

Specify a topic to publish to, e.g. myTopic/1

```
{
  "message": "Hello from AWS IoT console"
}
```

Conclusion

By following the steps in this document, you were able to configure a Pi to send a message to the AWS IoT service. This is really the end of the beginning. You can continue with the examples and tutorials on the AWS website to gain further knowledge regarding the AWS Greengrass and IoT services.