Cloud, Meet Edge

How to Deploy AWS IoT Greengrass Using Docker Containers and Ubuntu snap

Tatiana Cooke, Sr Product Manager
AWS IoT Greengrass

Alan Findley
Sr. VP Emerging Technologies
Prologis

Gary Bruns
VP Emerging Technologies
Prologis

Toban Zolman
VP Product
Rigado

3/26/2019

AWS IoT architecture

Data services

How do I extract value from my IoT data?

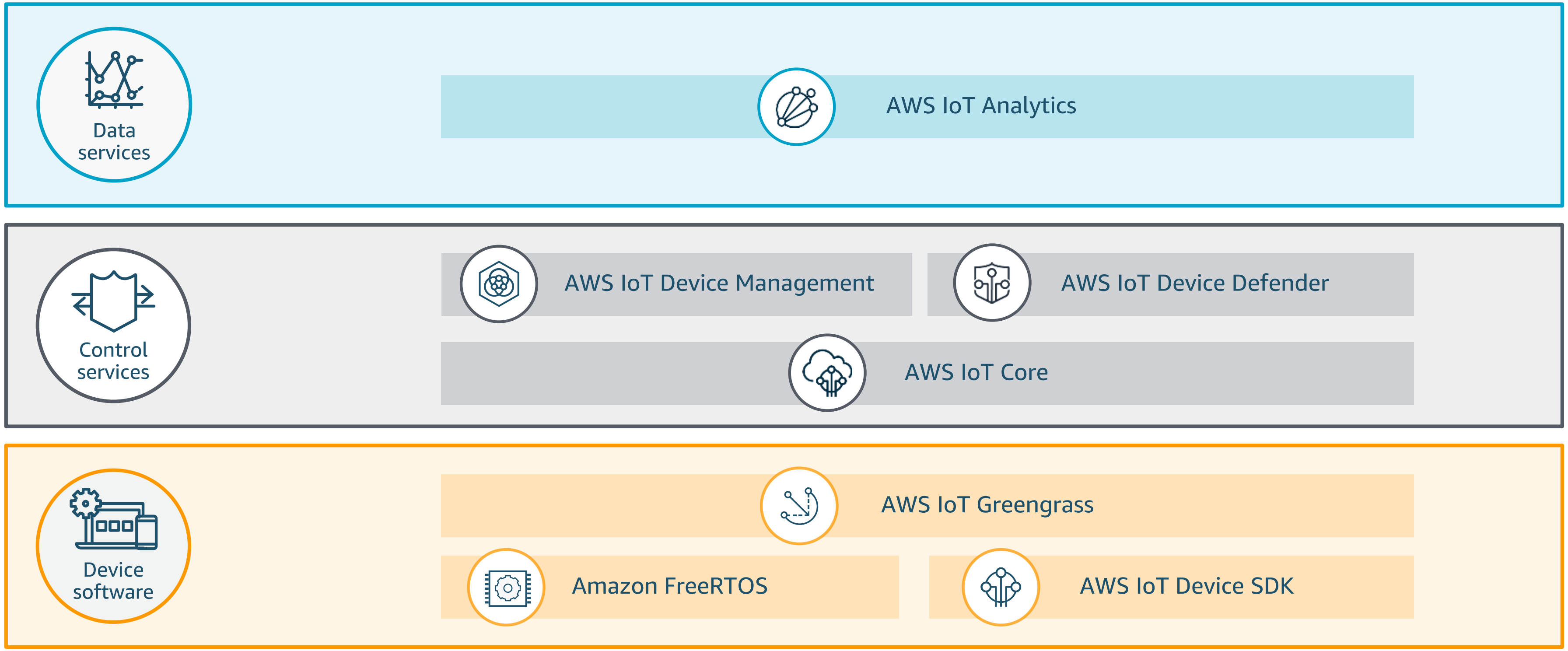
Control services

How can I control, manage, and secure my devices?

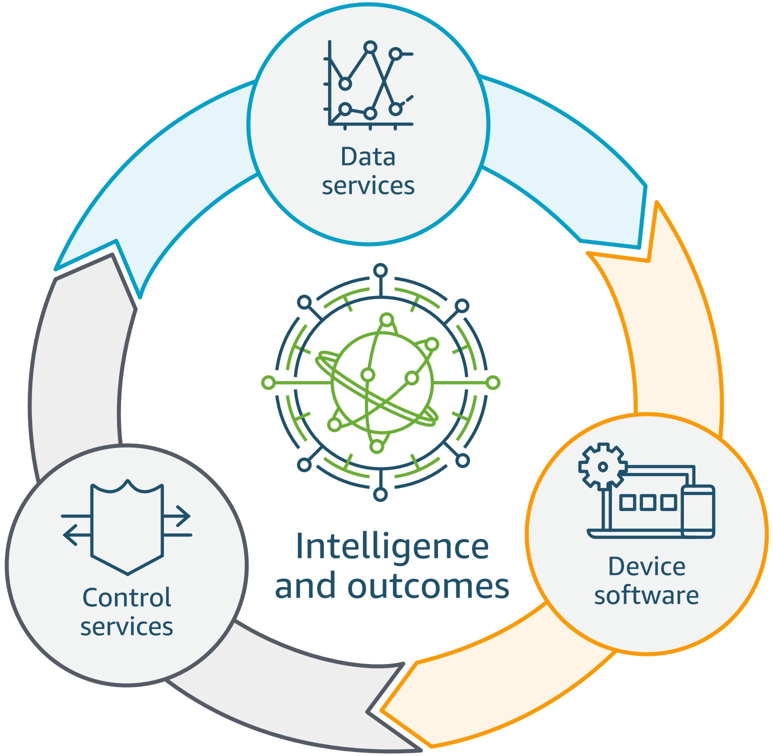
Device software

How can I connect my devices and operate at the edge?

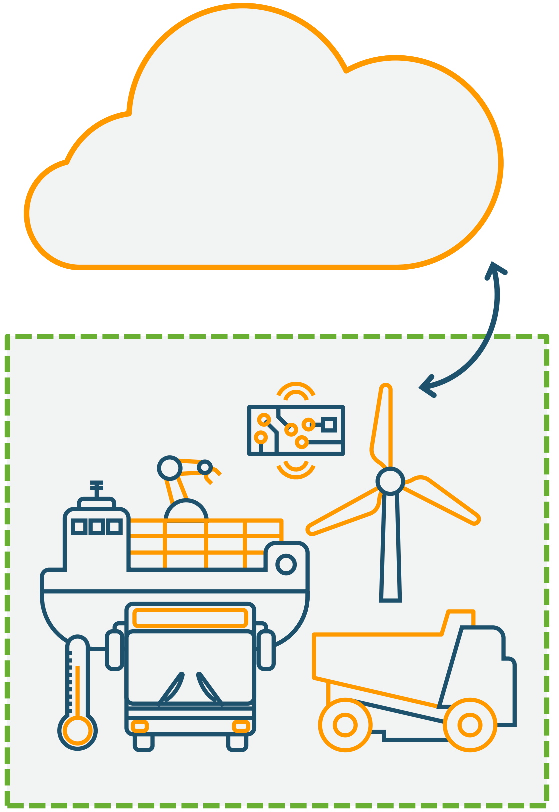
AWS IoT Architecture



IoT virtuous cycle



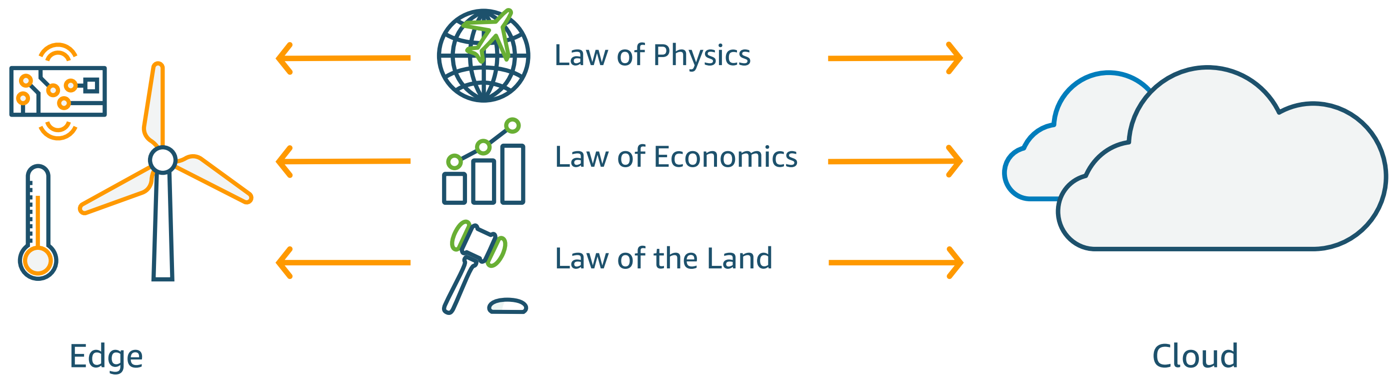
How can I extend AWS cloud capabilities to the edge?



Device software

AWS IoT Greengrass

AWS IoT Greengrass extends AWS IoT onto your devices, so that they can act locally on the data they generate, while still taking advantage of the cloud.



Device software

AWS IoT Greengrass

Local Messages and Triggers	Local Actions	Data and State Sync	Security	Local Resource Access	Machine Learning Inference	Connectors	Secrets Manager	Over the Air Updates
Enable device communication without a cloud connection	Simplify device programming with AWS Lambda	Operate devices offline & synchronize data when reconnected	Mutual authentication & authorization between cloud and devices	AWS Lambda functions can access & use local resources of a given device	Perform ML Inference locally	Extend edge devices with connections to external services	Deploy secrets to edge devices	Easily update AWS IoT Greengrass Core

Device software

Local Messages and Triggers

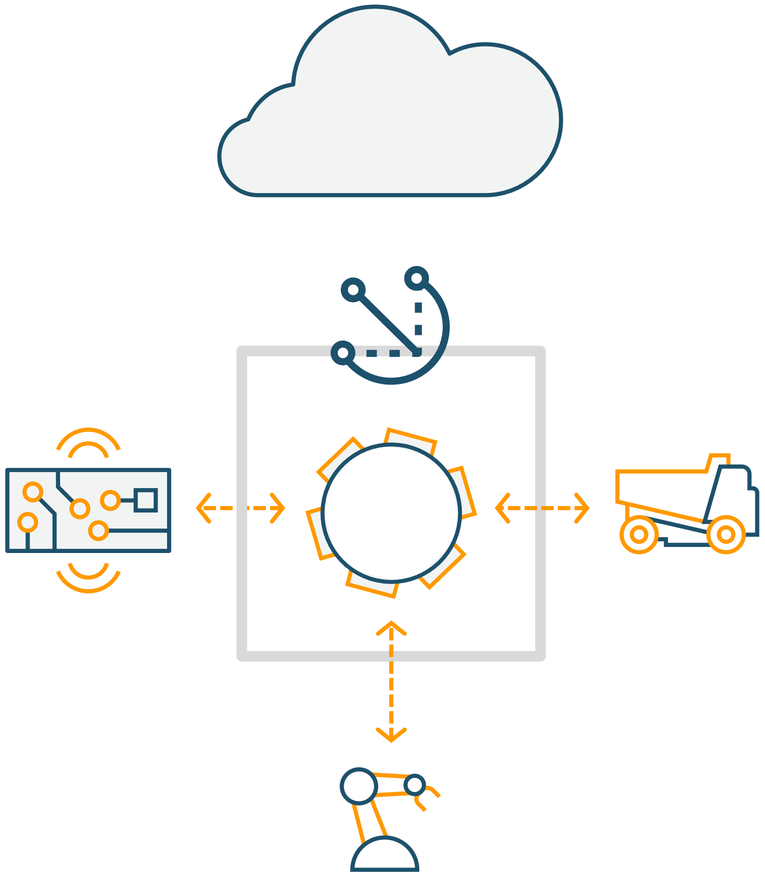
Enables messaging between devices on a local network so they can communicate without a cloud connection

Extends the AWS IoT MQTT pub/sub messaging paradigm locally to the edge

Allows AWS Lambda functions written in the cloud and deployed locally on the AWS IoT Greengrass Core to trigger and respond to events

Enables offline command and control operations from the AWS IoT Greengrass Core and other devices that use the AWS IoT Device SDK

For example, the AWS IoT Greengrass core can detect low moisture in the soil and in response, trigger an action to spray more water in smart greenhouse, without a connection to the cloud



aws

Local Actions

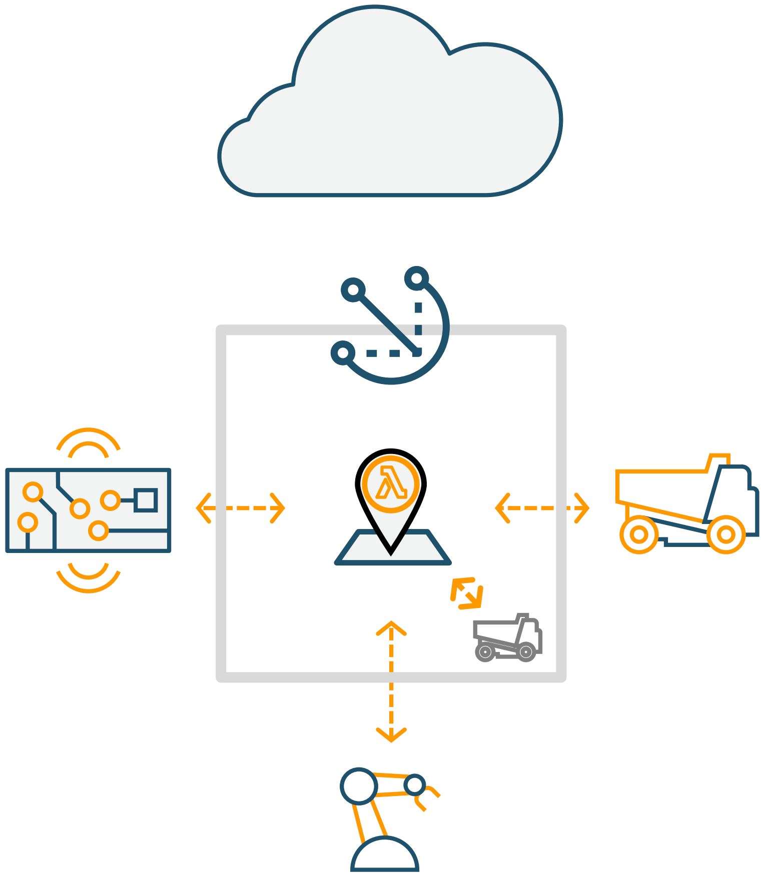
Simplify embedded software development with local AWS Lambda functions

Write event-driven AWS Lambda functions in the cloud and deploy them to devices

Run AWS Lambda functions written in Python 2.7, Node.js or Java

Invoke AWS Lambda functions with messaging and shadow updates

Offline actions and triggers for example, detecting low moisture in the soil and then triggering controls to spray more water inside a smart greenhouse



Data & State Sync

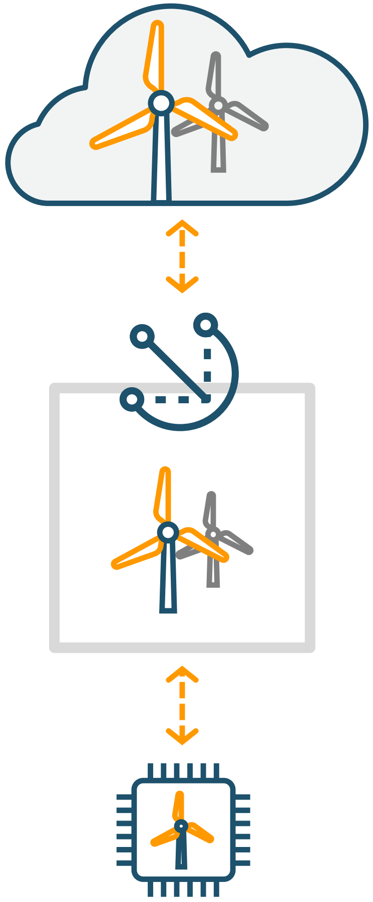
Operate devices during intermittent connectivity and synchronize data with the cloud when reconnected

Enables you to define a shadow state for a device as a JSON document in any logical manner—a single wind turbine, a windfarm, or a resource grid

Allows shadow states to be local or synced to the cloud

AWS Lambda functions running on the AWS IoT Greengrass Core can update shadow states through MQTT messages

For example, the AWS IoT Greengrass Core can update a tractor's shadow with continuous information on harvest quality and a snapshot of the data can be synced to the cloud at the end of the day



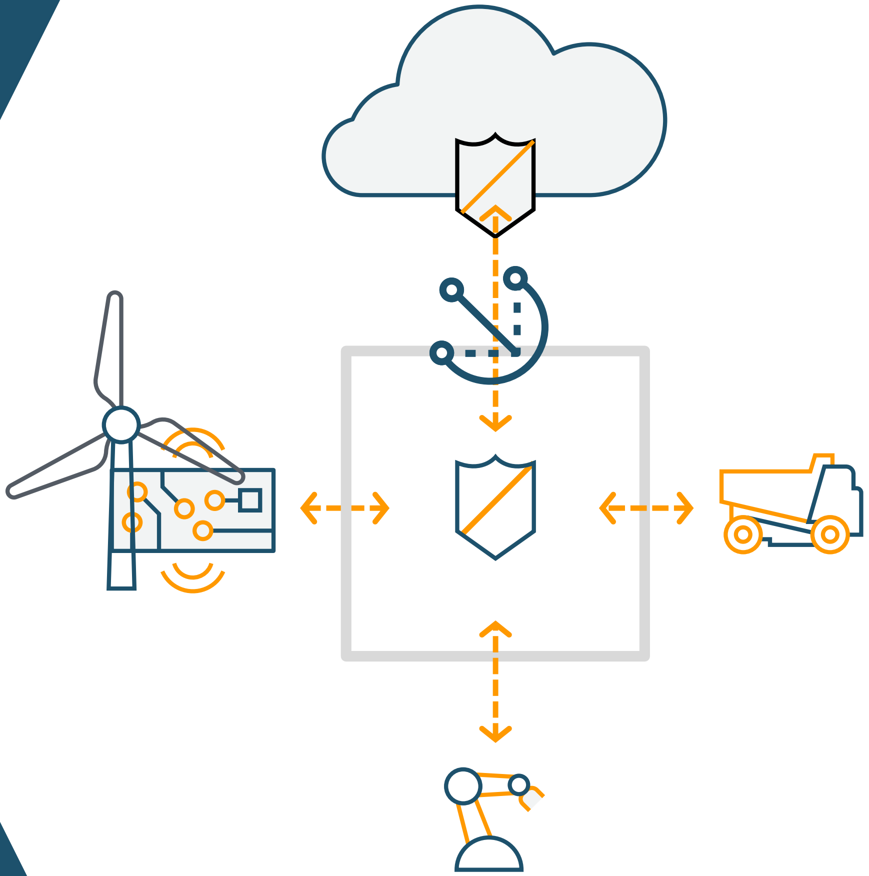
Security

Authenticates and encrypts device data for local and cloud communications

Supports TLS mutual authentication, both locally and with the cloud

Certificates on your devices can be associated to SigV4 credentials in the cloud

Establish hardware-based root of trust for encrypting secrets used in local AWS Lambda functions and for storing private device keys



Local Resource Access

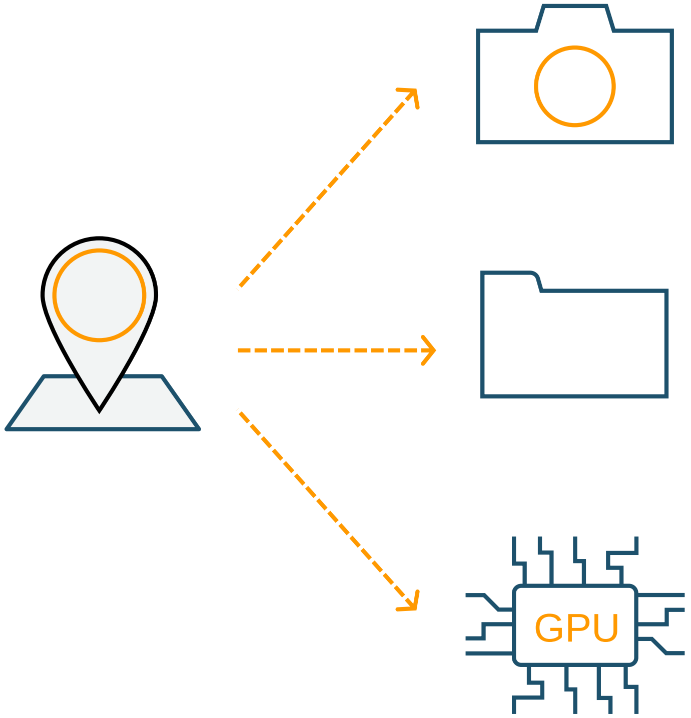
AWS Lambda functions can access & use local resources of a given device

Allows Lambdas to access local resources on a device

GPIO can be accessed to process sensor and actuator data

Lambdas can take advantage of the local file system on your operating system

Lambdas can use GPUs for hardware acceleration for machine learning



Machine Learning Inference

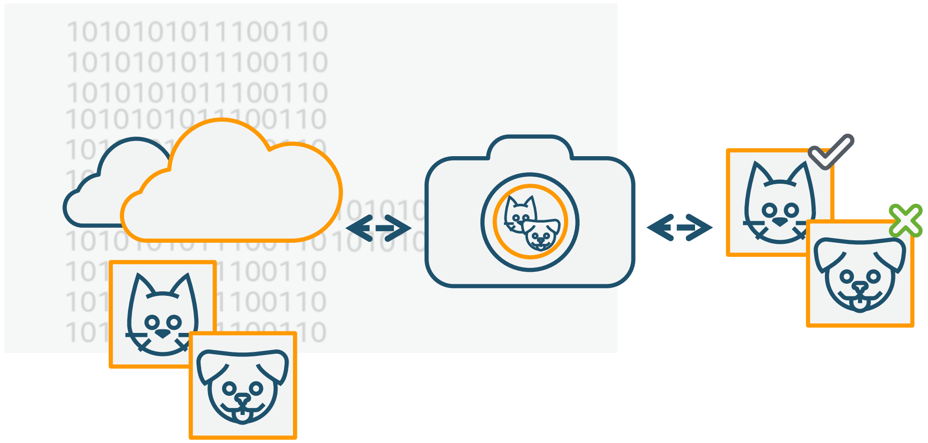
Perform ML Inference locally without data transfer costs or increased latency

Train models in the cloud using Amazon SageMaker or another service using EC2

ML Inference works with Apache MXNet and TensorFlow

Transfer your trained models onto your device and also send data back to the cloud to improve model accuracy

Integration with Amazon SageMaker reduces model runtime footprint 100x and improves inference performance 2x



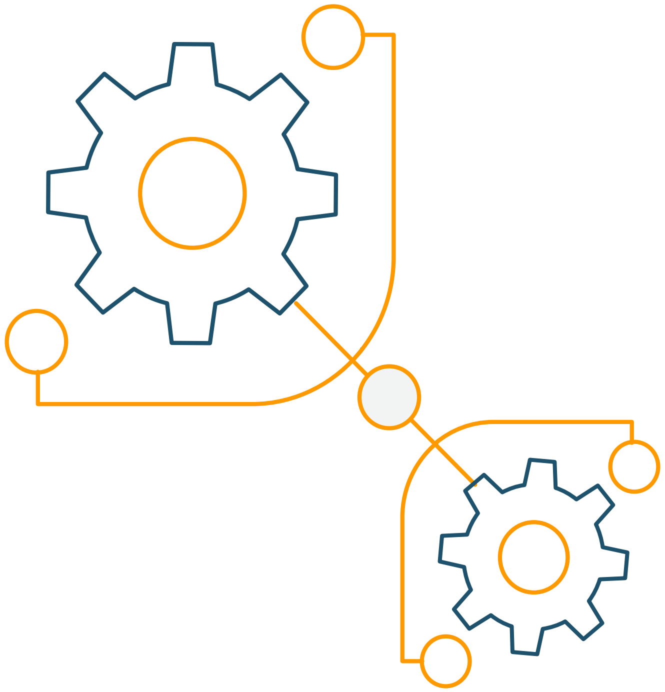
AWS IoT Greengrass Connectors

Quickly connect edge devices to third-party services, on-premises software, and AWS services

Pre-built functions enable easy connections with AWS Cloud services such as AWS Kinesis Firehose, Amazon CloudWatch, and Amazon Simple Notification Service (SNS)

Pre-built integrations with Twilio, ServiceNow, and other software as a service applications

Use connectors as building blocks and integrate into complex applications



AWS IoT Greengrass Secrets Manager

Deploy secrets to edge devices

Store, access, rotate, and manage secrets—device credentials, keys, endpoints, and configurations

Securely manage secrets in the cloud and deploy locally on edge devices

Manage secrets on devices through AWS Secrets Manager in the cloud

Over the Air Updates

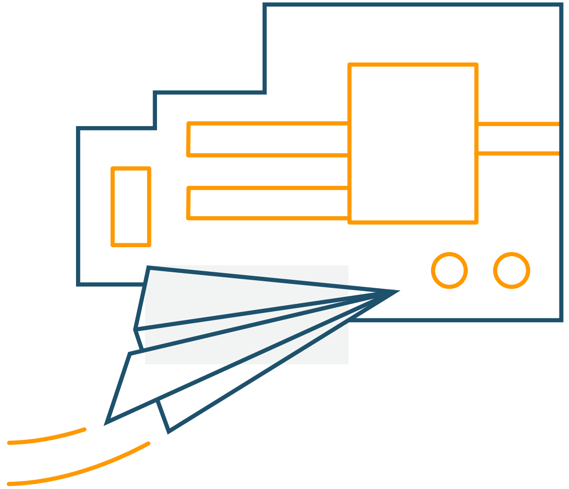
Easily update AWS IoT Greengrass devices and deploy security updates, bug fixes, & features

Remotely update an AWS IoT Greengrass Core device with the latest AWS IoT Greengrass software, security updates, bug fixes, and new features

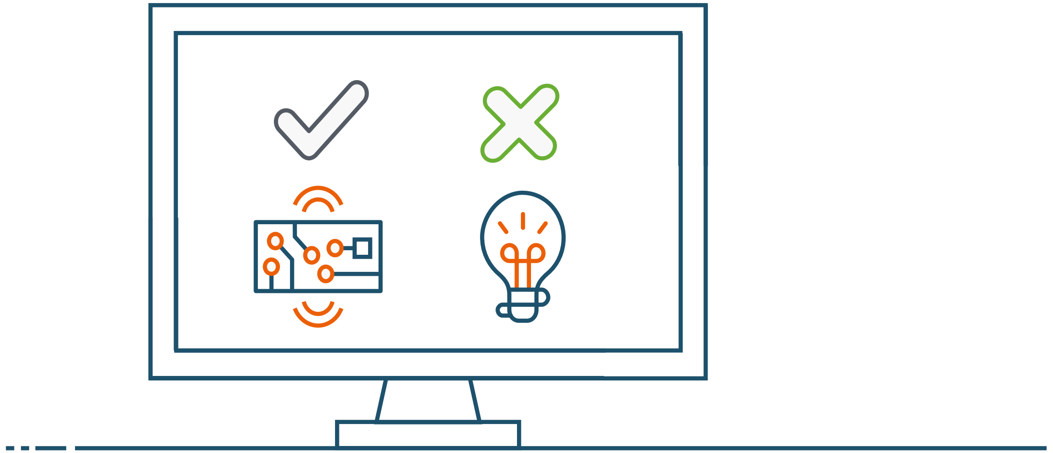
Enables bulk updates of many AWS IoT Greengrass Core devices at once

Updates are fail-safe: any breaking changes will trigger an automatic revert

Status of updates can be tracked from the AWS IoT console



How can I ensure my devices will work with AWS IoT services?



Device software

AWS IoT Device Tester

AWS IoT Device Tester is a test automation tool that lets you test Amazon FreeRTOS or AWS IoT Greengrass on your choice of devices.

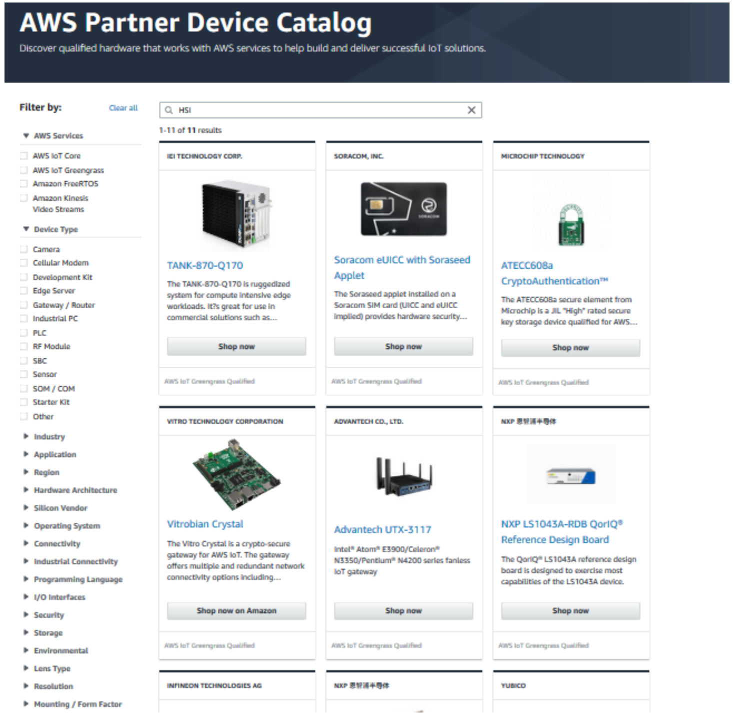
AWS IoT Device Tester for Amazon FreeRTOS
Tests if the Amazon FreeRTOS cloud connectivity, OTA, and security libraries function correctly on top of microcontroller board device drivers

AWS IoT Device Tester for AWS IoT Greengrass
Tests if the combination of device's CPU architecture, Linux kernel configuration, and drivers work with AWS IoT Greengrass

Download AWS IoT Device Tester

from AWS IoT Greengrass and Amazon FreeRTOS product pages

Greengrass hardware devices searchable in the AWS Partner Device Catalog



AWS Partner Device Catalog

Discover qualified hardware that works with AWS services to help build and deliver successful IoT solutions.

Filter by: Clear all

▼ AWS Services
- AWS IoT Core
- AWS IoT Greengrass
- Amazon FreeRTOS
- Amazon Kinesis Video Streams

▼ Device Type
- Camera
- Cellular Modem
- Development Kit
- Edge Server
- Gateway / Router
- Industrial PC
- PLC
- RF Module
- SBC
- Sensor
- SOM / COM
- Starter Kit
- Other

▶ Industry
▶ Application
▶ Region
▶ Hardware Architecture
▶ Silicon Vendor
▶ Operating System
▶ Connectivity
▶ Industrial Connectivity
▶ Programming Language
▶ I/O Interfaces
▶ Security
▶ Storage
▶ Environmental
▶ Lens Type
▶ Resolution
▶ Mounting / Form Factor

HSI

1-11 of **11** results

IEI TECHNOLOGY CORP.
TANK-870-Q170
The TANK-870-Q170 is ruggedized system for compute intensive edge workloads. It?s great for use in commercial solutions such as...
Shop now
AWS IoT Greengrass Qualified

SORACOM, INC.
Soracom eUICC with Soraseed Applet
The Soraseed applet installed on a Soracom SIM card (UICC and eUICC implied) provides hardware security....
Shop now
AWS IoT Greengrass Qualified

MICROCHIP TECHNOLOGY
ATECC608a CryptoAuthentication™
The ATECC608a secure element from Microchip is a JIL "High" rated secure key storage device qualified for AWS....
Shop now
AWS IoT Greengrass Qualified

VITRO TECHNOLOGY CORPORATION
Vitrobian Crystal
The Vitro Crystal is a crypto-secure gateway for AWS IoT. The gateway offers multiple and redundant network connectivity options including....
Shop now on Amazon
AWS IoT Greengrass Qualified

ADVANTECH CO., LTD.
Advantech UTX-3117
Intel® Atom® E3900/Celeron® N3350/Pentium® N4200 series fanless IoT gateway
Shop now
AWS IoT Greengrass Qualified

NXP 恩智浦半导体
NXP LS1043A-RDB QorIQ® Reference Design Board
The QorIQ® LS1043A reference design board is designed to exercise most capabilities of the LS1043A device.
Shop now
AWS IoT Greengrass Qualified

INFINEON TECHNOLOGIES AG

NXP 恩智浦半导体

YUBICO

Search the device catalog for the most current options: https://devices.amazonaws.com/search?page=1&sv=gg

Greengrass runs on a variety of arch/OS systems

Supported platforms:

Architecture: ARMv7l; OS: Linux; Distribution: Raspbian Stretch, 2018-06-29

Architecture: x86_64; OS: Linux; Distribution: Amazon Linux (amzn-ami-hvm-2016.09.1.20170119-x86_64-ebs), Ubuntu 14.04 – 16.04

Architecture: ARMv8 (AArch64); OS: Linux; Distribution: Arch Linux

Running on a different arch/OS?

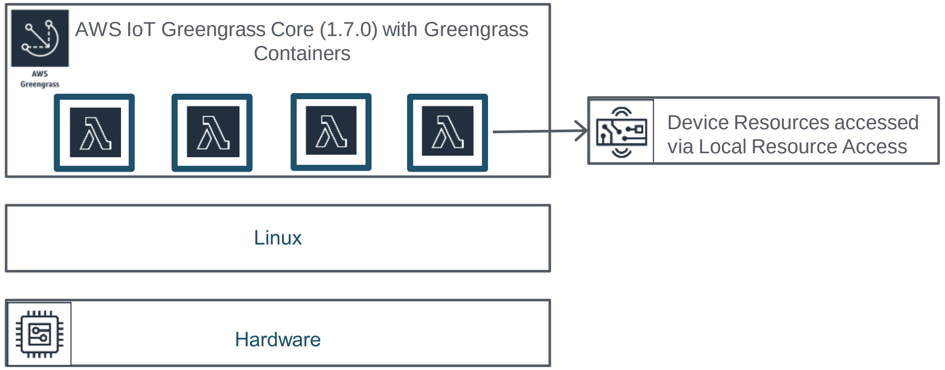
Greengrass can now run in other container environments

- Docker
- Ubuntu snap

New modes provide flexibility in configuring Greengrass

- Containerized: AWS IoT Greengrass with per-Lambda container isolation
- No Containers: Run Greengrass as an OS process. AWS Lambdas and Greengrass Group have no container
- Hybrid: Hybrid mix of isolated Lambdas and Lambdas as OS processes

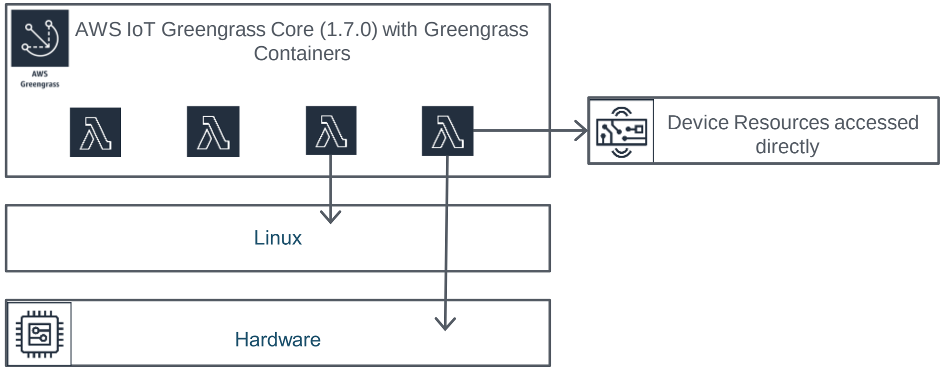
AWS IoT Greengrass with per-Lambda isolation


AWS Greengrass
AWS IoT Greengrass Core (1.7.0) with Greengrass Containers
Device Resources accessed via Local Resource Access
Linux
Hardware

New modes provide flexibility in configuring Greengrass

- Containerized: AWS IoT Greengrass with per-Lambda container isolation
- No Containers: Run Greengrass as an OS process. AWS Lambdas and Greengrass Group have no container
- Hybrid: Hybrid mix of isolated Lambdas and Lambdas as OS processes

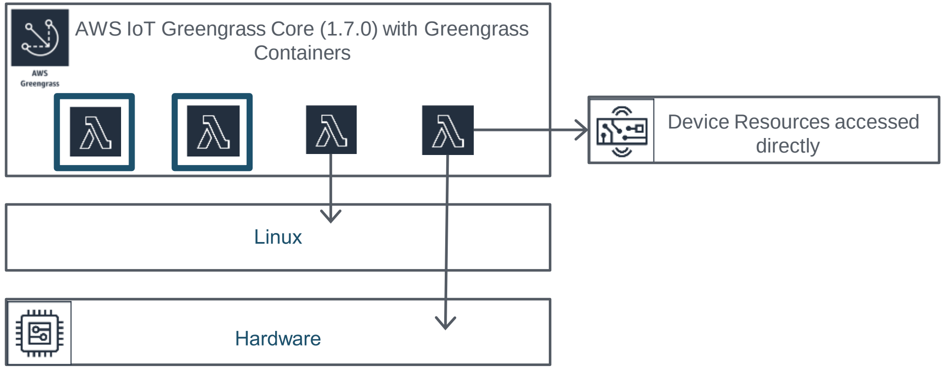
AWS IoT Greengrass without Lambda isolation


AWS IoT Greengrass Core (1.7.0) with Greengrass Containers
AWS Greengrass
Device Resources accessed directly
Linux
Hardware

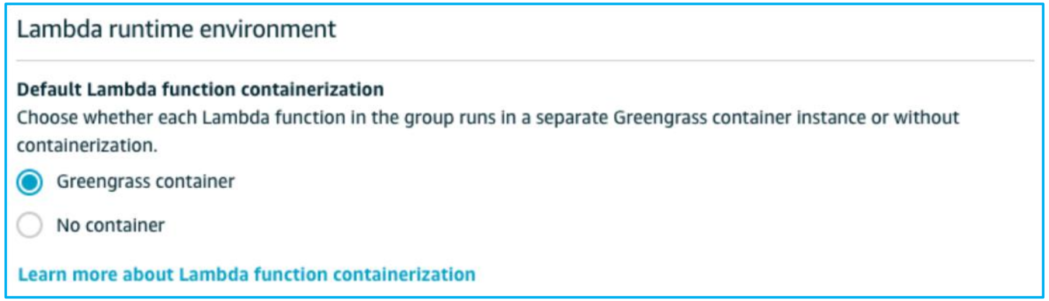
New modes provide flexibility in configuring Greengrass

- Containerized: AWS IoT Greengrass with per-Lambda container isolation
- No Container: Run Greengrass as an OS process. AWS Lambdas and Greengrass Group have no container
- Hybrid: Hybrid mix of isolated Lambdas and Lambdas as OS processes

AWS IoT Greengrass mix of isolation configurations


AWS Greengrass
AWS IoT Greengrass Core (1.7.0) with Greengrass Containers
Device Resources accessed directly
Linux
Hardware

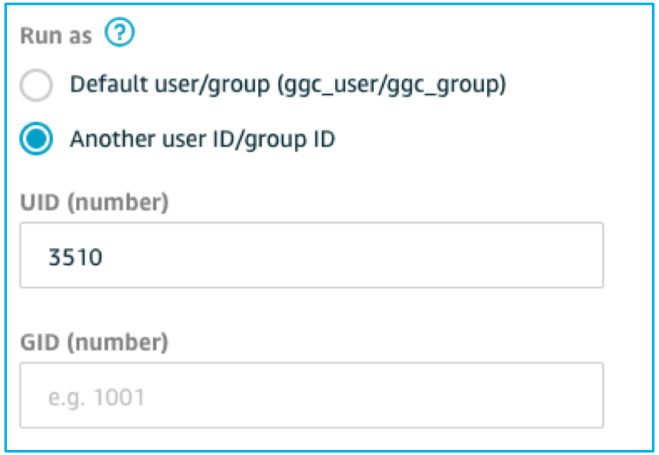
Greengrass has Group and Lambda-level isolation and permission settings


Lambda runtime environment

Default Lambda function containerization
Choose whether each Lambda function in the group runs in a separate Greengrass container instance or without containerization.

[✓ Greengrass container]
No container

Learn more about Lambda function containerization


Run as

Default user/group (ggc_user/ggc_group)
[✓ Another user ID/group ID]

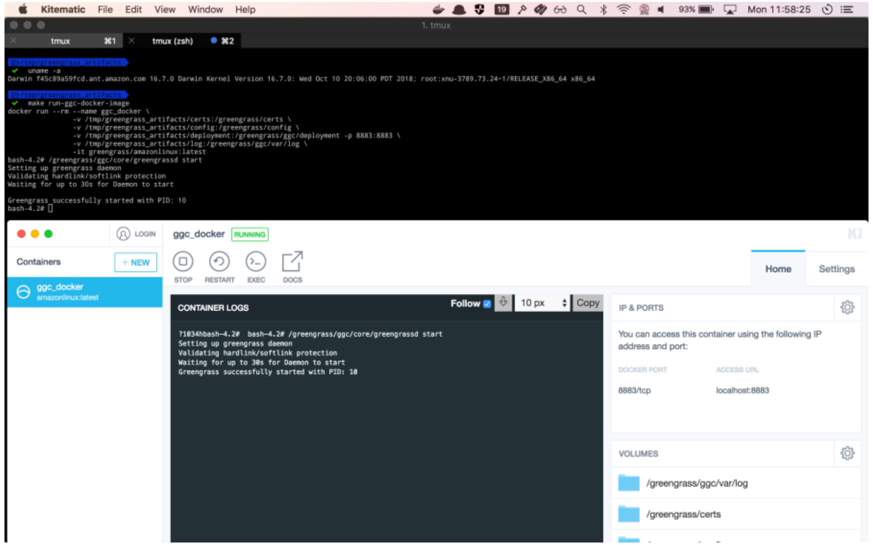
UID (number)

3510

GID (number)

e.g. 1001

Run Greengrass in Docker on Mac OS X or Windows 10



```
/tmp/greengrass_artifacts
✔ uname -a
Darwin f45c89a59fcd.ant.amazon.com 16.7.0 Darwin Kernel Version 16.7.0: Wed Oct 10 20:06:00 PDT 2018; root:xnu-3789.73.24-1/RELEASE_X86_64 x86_64

/tmp/greengrass_artifacts
✔ make run-ggc-docker-image
docker run --rm --name ggc_docker \
                -v /tmp/greengrass_artifacts/certs:/greengrass/certs \
                -v /tmp/greengrass_artifacts/config:/greengrass/config \
                -v /tmp/greengrass_artifacts/deployment:/greengrass/ggc/deployment -p 8883:8883 \
                -v /tmp/greengrass_artifacts/log:/greengrass/ggc/var/log \
                -it greengrass/amazonlinux:latest
bash-4.2# /greengrass/ggc/core/greengrassd start
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 30s for Daemon to start

Greengrass successfully started with PID: 10
bash-4.2# 
```

```
?1034hbash-4.2#  bash-4.2# /greengrass/ggc/core/greengrassd start
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 30s for Daemon to start
Greengrass successfully started with PID: 10
```

IP & PORTS

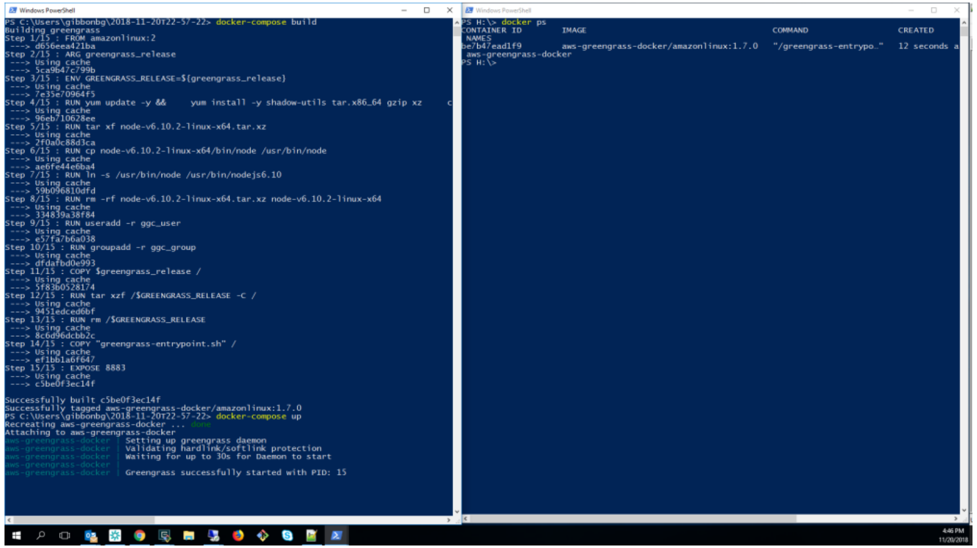
You can access this container using the following IP address and port:

DOCKER PORT	ACCESS URL
8883/tcp	localhost:8883

VOLUMES

/greengrass/ggc/var/log

/greengrass/certs



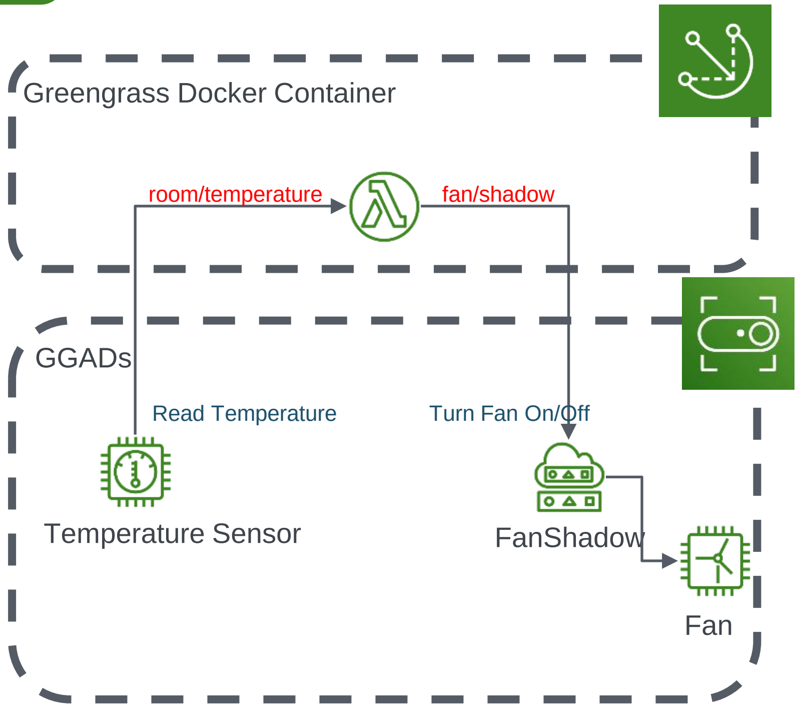
```
PS C:\Users\gibbonbg\2018-11-20T22-57-22> docker-compose build
Building greengrass
Step 1/15 : FROM amazonlinux:2
 ---> d656eea421ba
Step 2/15 : ARG greengrass_release
 ---> Using cache
 ---> 5ca9b47c799b
Step 3/15 : ENV GREENGRASS_RELEASE=${greengrass_release}
 ---> Using cache
 ---> 7e35e70964f5
Step 4/15 : RUN yum update -y &&     yum install -y shadow-utils tar.x86_64 gzip xz     c
 ---> Using cache
 ---> 96eb710628ee
Step 5/15 : RUN tar xf node-v6.10.2-linux-x64.tar.xz
 ---> Using cache
 ---> 2f0a0c88d3ca
Step 6/15 : RUN cp node-v6.10.2-linux-x64/bin/node /usr/bin/node
 ---> Using cache
 ---> ae6fe44e6ba4
Step 7/15 : RUN ln -s /usr/bin/node /usr/bin/nodejs6.10
 ---> Using cache
 ---> 59b096810dfd
Step 8/15 : RUN rm -rf node-v6.10.2-linux-x64.tar.xz node-v6.10.2-linux-x64
 ---> Using cache
 ---> 334839a38f84
Step 9/15 : RUN useradd -r ggc_user
 ---> Using cache
 ---> e57fa7b6a038
Step 10/15 : RUN groupadd -r ggc_group
 ---> Using cache
 ---> dfdafbd0e993
Step 11/15 : COPY $greengrass_release /
 ---> Using cache
 ---> 5f83b0528174
Step 12/15 : RUN tar xzf /$GREENGRASS_RELEASE -C /
 ---> Using cache
 ---> 9451edced6bf
Step 13/15 : RUN rm /$GREENGRASS_RELEASE
 ---> Using cache
 ---> 8c6d96dcbb2c
Step 14/15 : COPY "greengrass-entrypoint.sh" /
 ---> Using cache
 ---> ef1bb1a6f647
Step 15/15 : EXPOSE 8883
 ---> Using cache
 ---> c5be0f3ec14f

Successfully built c5be0f3ec14f
Successfully tagged aws-greengrass-docker/amazonlinux:1.7.0
PS C:\Users\gibbonbg\2018-11-20T22-57-22> docker-compose up
Recreating aws-greengrass-docker ... done
Attaching to aws-greengrass-docker
aws-greengrass-docker | Setting up greengrass daemon
aws-greengrass-docker | Validating hardlink/softlink protection
aws-greengrass-docker | Waiting for up to 30s for Daemon to start
aws-greengrass-docker |
aws-greengrass-docker | Greengrass successfully started with PID: 15
```

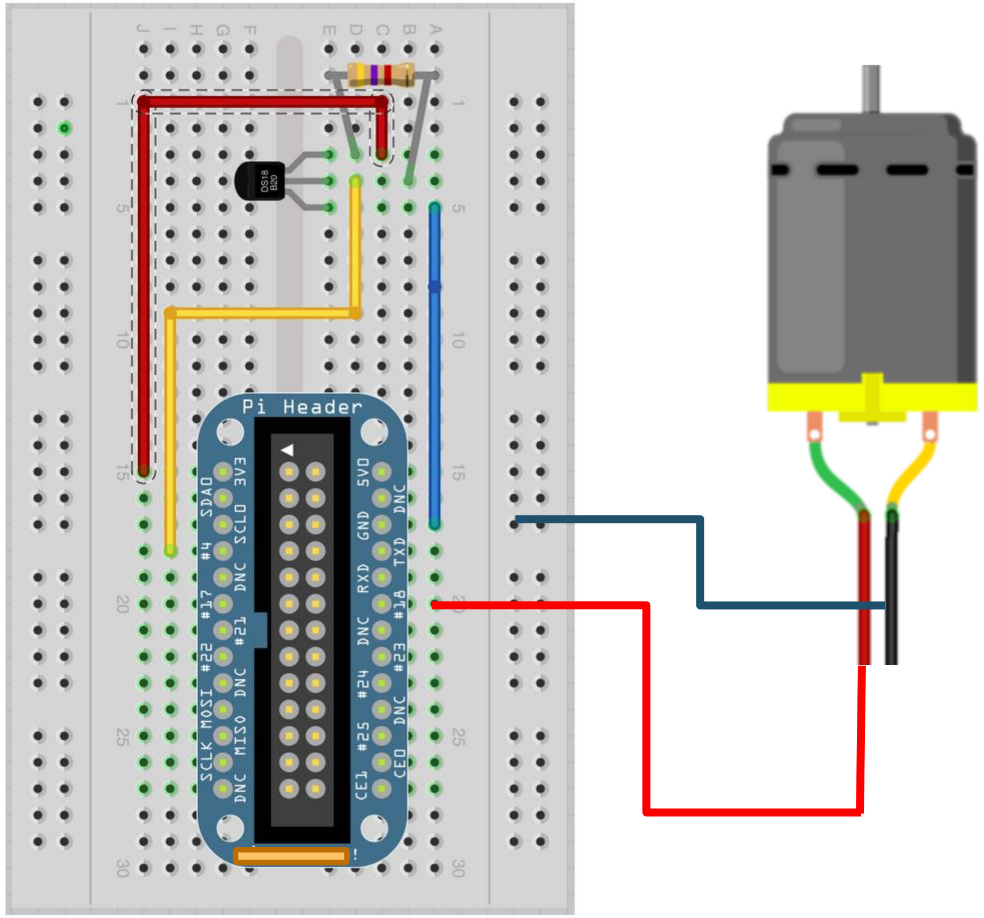
```
PS H:\> docker ps
CONTAINER ID        IMAGE                                    COMMAND                  CREATED
 NAMES
be7b47ead1f9        aws-greengrass-docker/amazonlinux:1.7.0   "/greengrass-entrypo…"   12 seconds a
 aws-greengrass-docker
PS H:\>
```

Creating IoT Architecture from your Windows dev environment: Smart fan demo

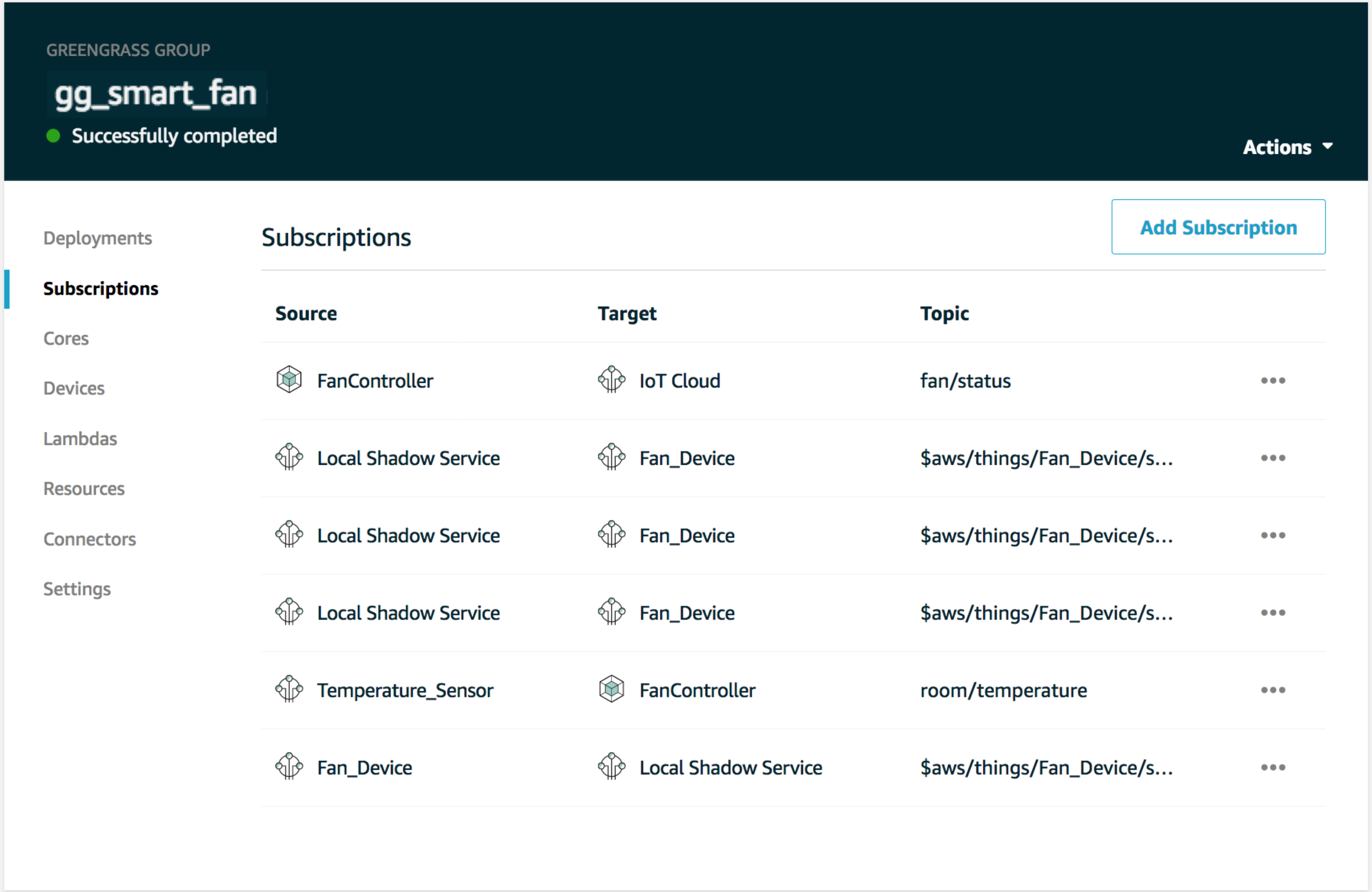
Designed by Ankit Gupta, Development Engineer
IoT Device Services

Device Lab Fan System



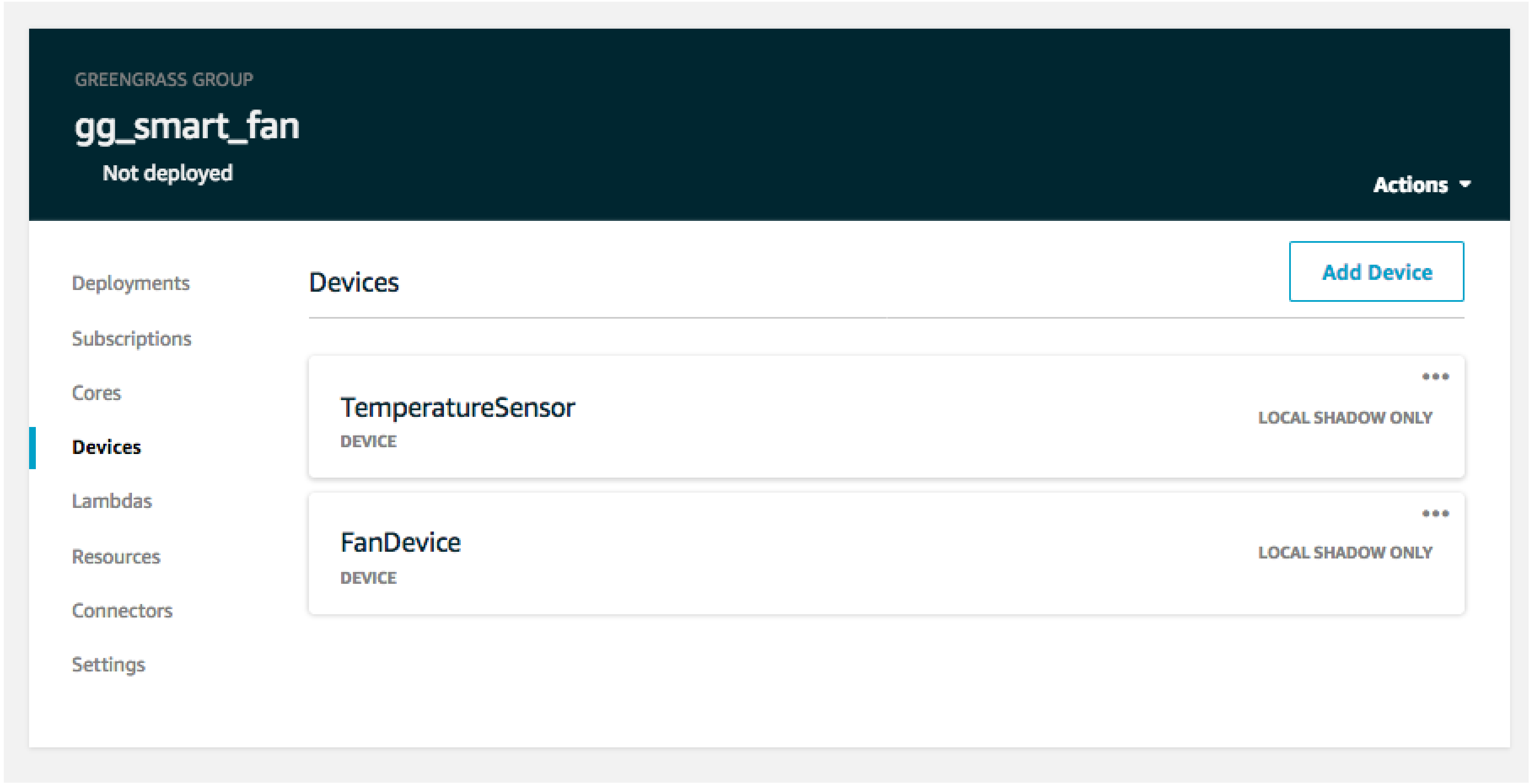


Pi Header
3V3
SDA0
SCL0
#4
DNC
#17
#21
#22
DNC
MOSI
MISO
SCLK
DNC
5V0
DNC
GND
TXD
RXD
#18
DNC
#23
#24
DNC
#25
CE0
CE1
DS18B20



Subscriptions

Source	Target	Topic
FanController	IoT Cloud	fan/status
Local Shadow Service	Fan_Device	$aws/things/Fan_Device/s...
Local Shadow Service	Fan_Device	$aws/things/Fan_Device/s...
Local Shadow Service	Fan_Device	$aws/things/Fan_Device/s...
Temperature_Sensor	FanController	room/temperature
Fan_Device	Local Shadow Service	$aws/things/Fan_Device/s...



GREENGRASS GROUP

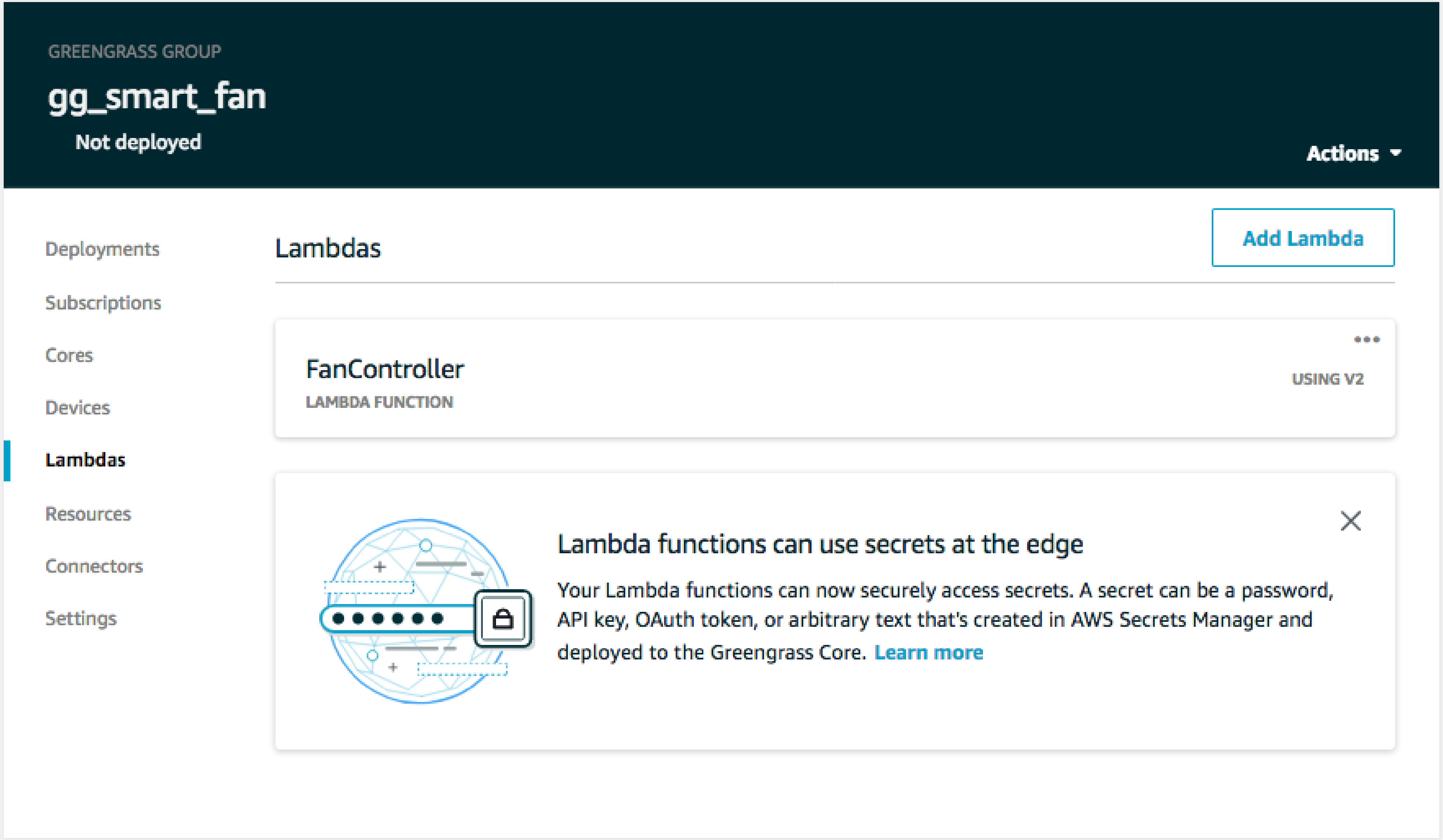
gg_smart_fan

Not deployed

Devices

TemperatureSensor
DEVICE
LOCAL SHADOW ONLY

FanDevice
DEVICE
LOCAL SHADOW ONLY



GREENGRASS GROUP

gg_smart_fan

Not deployed

Actions ▾

Deployments

Subscriptions

Cores

Devices

Lambdas

Resources

Connectors

Settings

Lambdas

Add Lambda

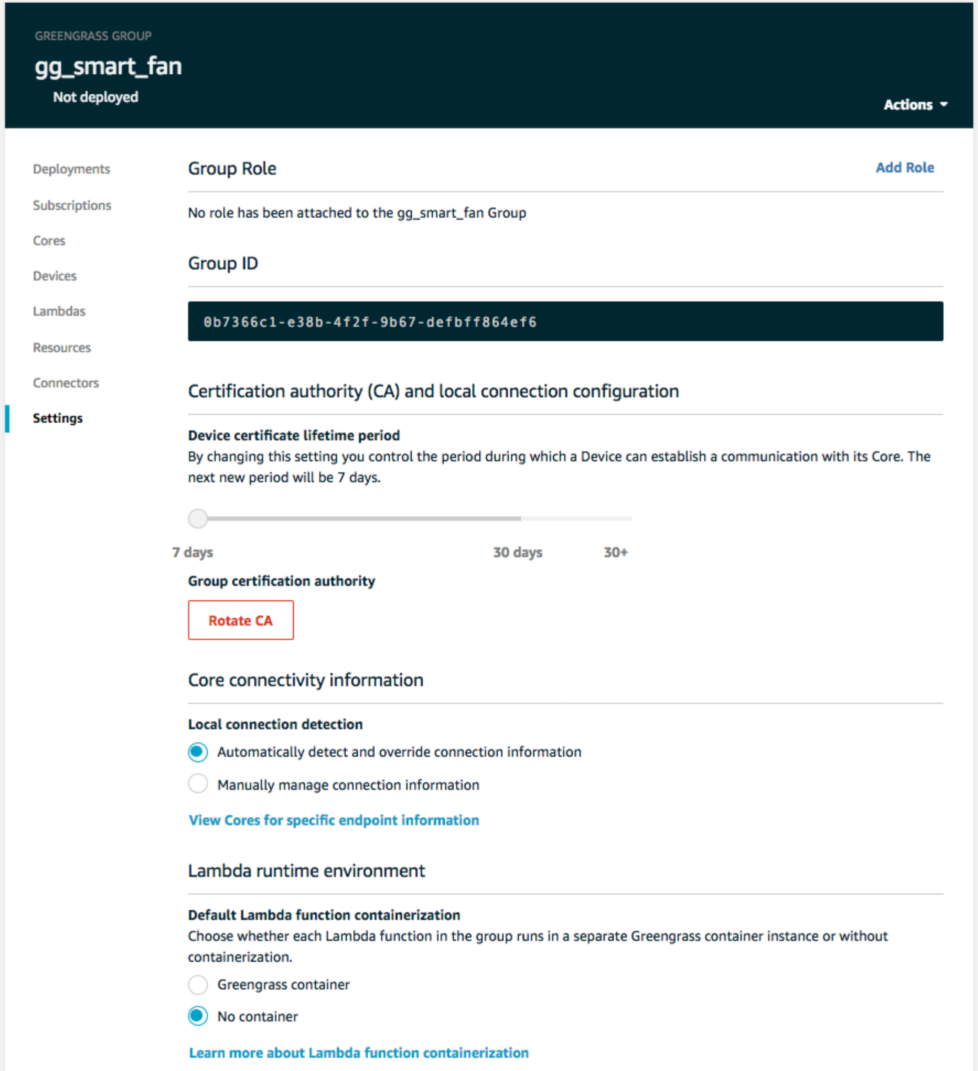
FanController

LAMBDA FUNCTION

USING V2

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



gg_smart_fan

Not deployed

Group Role

No role has been attached to the gg_smart_fan Group

Group ID

```
0b7366c1-e38b-4f2f-9b67-defbff864ef6
```

Certification authority (CA) and local connection configuration

Device certificate lifetime period

By changing this setting you control the period during which a Device can establish a communication with its Core. The next new period will be 7 days.

7 days — 30 days — 30+

Group certification authority

Core connectivity information

Local connection detection

[✓ Automatically detect and override connection information]

Manually manage connection information

View Cores for specific endpoint information

Lambda runtime environment

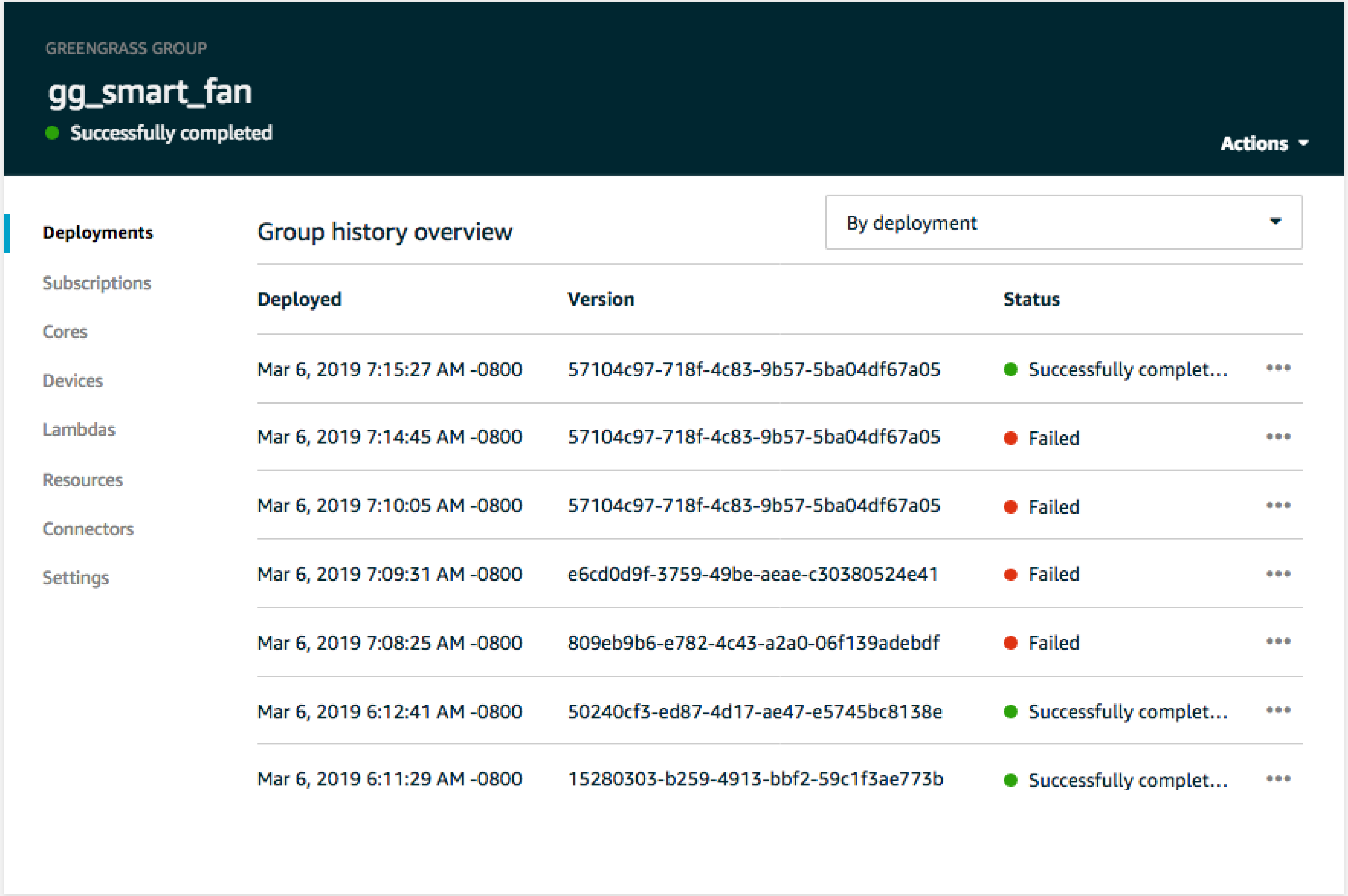
Default Lambda function containerization

Choose whether each Lambda function in the group runs in a separate Greengrass container instance or without containerization.

Greengrass container

[✓ No container]

Learn more about Lambda function containerization


GREENGRASS GROUP

gg_smart_fan

● Successfully completed

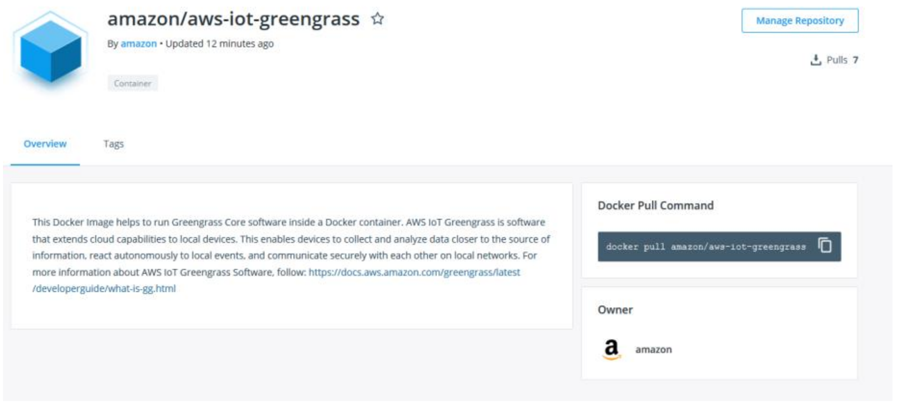
Group history overview

Deployed	Version	Status
Mar 6, 2019 7:15:27 AM -0800	57104c97-718f-4c83-9b57-5ba04df67a05	🟢 Successfully complet...
Mar 6, 2019 7:14:45 AM -0800	57104c97-718f-4c83-9b57-5ba04df67a05	🔴 Failed
Mar 6, 2019 7:10:05 AM -0800	57104c97-718f-4c83-9b57-5ba04df67a05	🔴 Failed
Mar 6, 2019 7:09:31 AM -0800	e6cd0d9f-3759-49be-aeae-c30380524e41	🔴 Failed
Mar 6, 2019 7:08:25 AM -0800	809eb9b6-e782-4c43-a2a0-06f139adebdf	🔴 Failed
Mar 6, 2019 6:12:41 AM -0800	50240cf3-ed87-4d17-ae47-e5745bc8138e	🟢 Successfully complet...
Mar 6, 2019 6:11:29 AM -0800	15280303-b259-4913-bbf2-59c1f3ae773b	🟢 Successfully complet...

Ready to get started with Docker?

Check out our Docker Image on ECR or Dockerhub

Dockerfile available via Amazon CloudFront



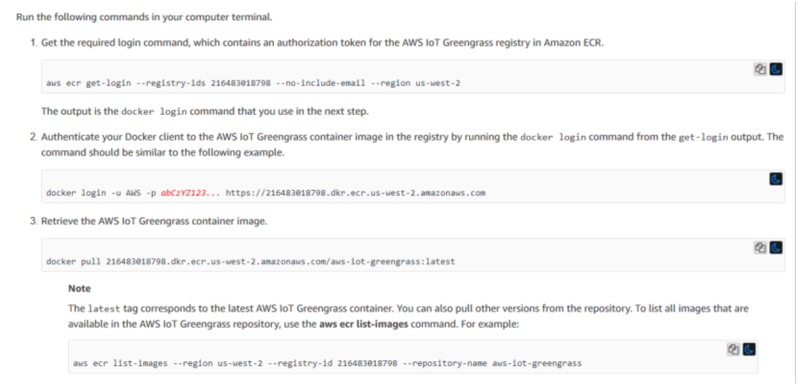
This Docker Image helps to run Greengrass Core software inside a Docker container. AWS IoT Greengrass is software that extends cloud capabilities to local devices. This enables devices to collect and analyze data closer to the source of information, react autonomously to local events, and communicate securely with each other on local networks. For more information about AWS IoT Greengrass Software, follow: https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html

Docker Pull Command

```
docker pull amazon/aws-iot-greengrass
```

Owner

amazon



Run the following commands in your computer terminal.

1. Get the required login command, which contains an authorization token for the AWS IoT Greengrass registry in Amazon ECR.

```
aws ecr get-login --registry-ids 216483018798 --no-include-email --region us-west-2
```

   The output is the `docker login` command that you use in the next step.

2. Authenticate your Docker client to the AWS IoT Greengrass container image in the registry by running the `docker login` command from the `get-login` output. The command should be similar to the following example.

```
docker login -u AWS -p abCzYZ123... https://216483018798.dkr.ecr.us-west-2.amazonaws.com
```

3. Retrieve the AWS IoT Greengrass container image.

```
docker pull 216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
```

Note

The `latest` tag corresponds to the latest AWS IoT Greengrass container. You can also pull other versions from the repository. To list all images that are available in the AWS IoT Greengrass repository, use the **aws ecr list-images** command. For example:

```
aws ecr list-images --region us-west-2 --registry-id 216483018798 --repository-name aws-iot-greengrass
```

Customer Story: Prologis & the Warehouse of the Future

Alan Findley
Sr. VP Emerging Technologies
Prologis

Gary Bruns
VP Emerging Technologies
Prologis

PROLOGIS®
Ahead of what's next

Prologis is the world's leading owner, operator and developer of logistics real estate

$1.5 TRILLION

is the economic value of goods flowing through our distribution centers each year, representing:

2.8%

of GDP for the 19 countries where we do business, and

2.0%

of the World's GDP

1.0MILLION

employees under Prologis' roofs

1983

Founded

$87B

Assets under management

100 GLOBAL

Most sustainable corporations

768 MSF

on four continents

aws

Prologis global customer network

PROLOGIS® LABS

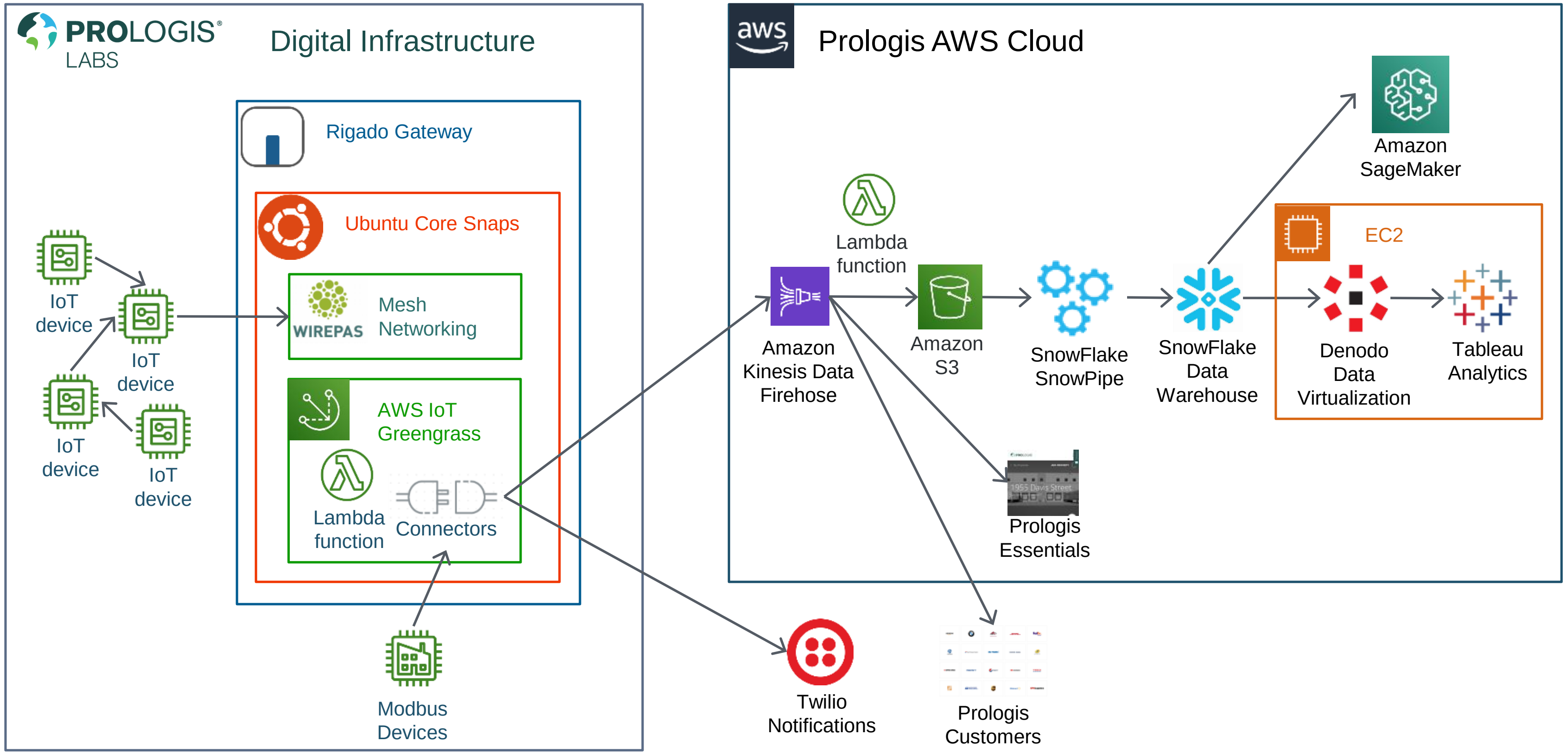
Prologis Labs incubates new business concepts and revenue streams by:

- Developing and testing technologies, products, and business models
- Sharing validated learnings from experiments
- Delivering promising opportunities to our customers, investors and company

One area of focus for Prologis Labs is the development of a global Digital Infrastructure supporting IoT sensors, asset tracking, robotics, autonomous vehicles, material handling, and more.

The partnerships with Rigado and Amazon are critical to the formation of the Prologis Digital Infrastructure.

Prologis Architecture Overview



Rigado Asset Tracking Demo

Toban Zolman
VP Product
Rigado

About Rigado

Rigado provides IoT Data Solutions for Smart Building Environments

COMPANY

Creating Bluetooth Low Energy (BLE) solutions with IoT innovators since 2010

SOLUTIONS

Edge Connectivity & Computing for asset tracking, sensing & monitoring using BLE

CUSTOMERS

More than 300 global customers with over 5 million devices in 30,000 locations

PROLOGIS, HILTI, Radius Networks, ONSET, NordicTrack, TCI LED, FAGERHULT, Steelcase, TANDEM DIABETES CARE, DECO digital

Cascade Edge-as-a-Service Solution

A secure & managed edge gateway solution for all your smart building applications

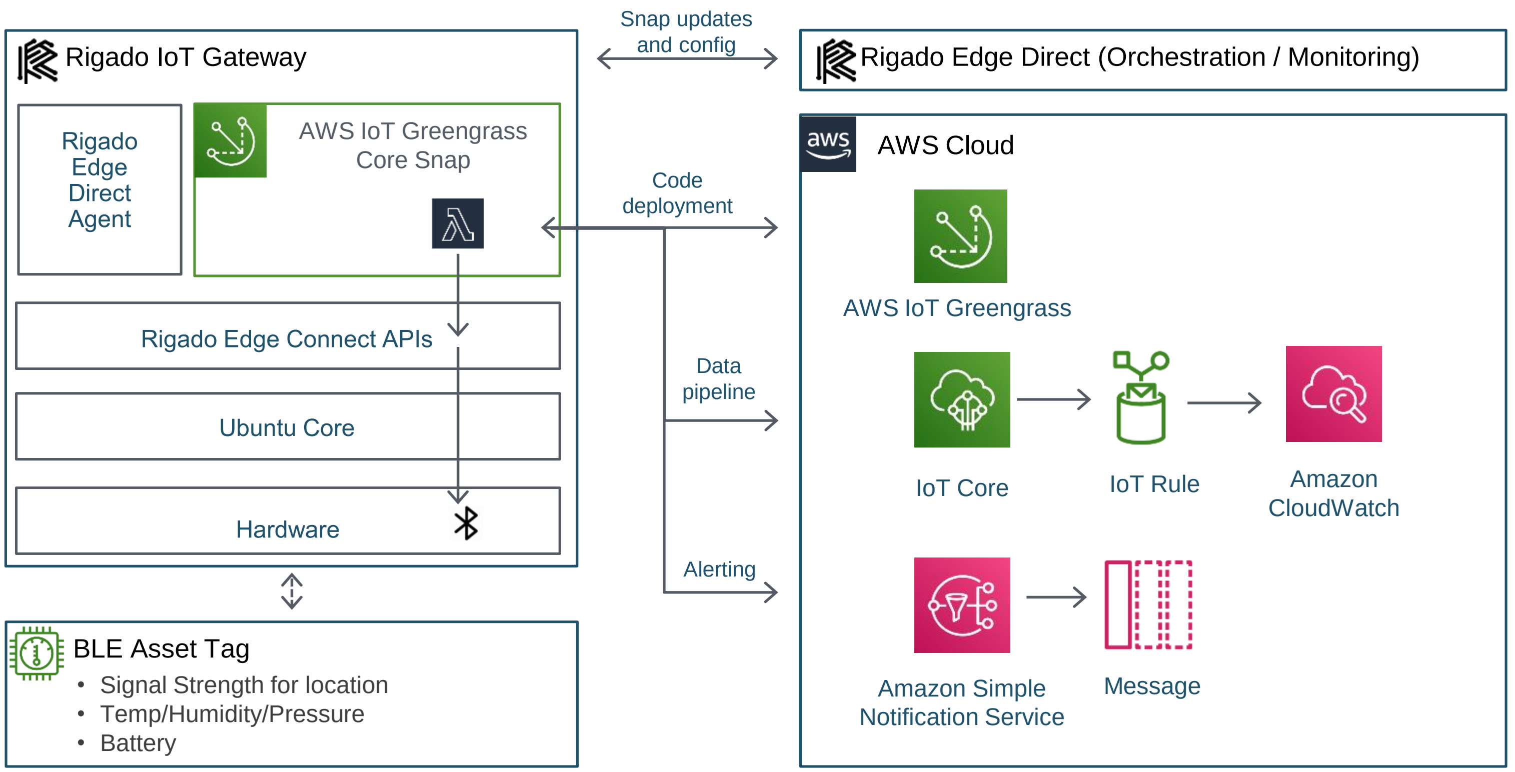
Offered as a simple monthly subscription

Connect Sensors & Devices
Environmental and Comfort Sensors
Asset Tracking and Location Tags
Smart Lighting and Building Controls

Deliver Data to Your Cloud
Integrate easily via MQTT & HTTP
Send directly to AWS
Run AWS IoT Greengrass

Scale & Manage Securely
Run multiple edge apps on gateways
Monitor and update apps as needed
Apply security updates from Rigado

Rigado Asset Tracking Architecture Overview



Get started:

Docker:

- Docker Image available via ECR or DockerHub

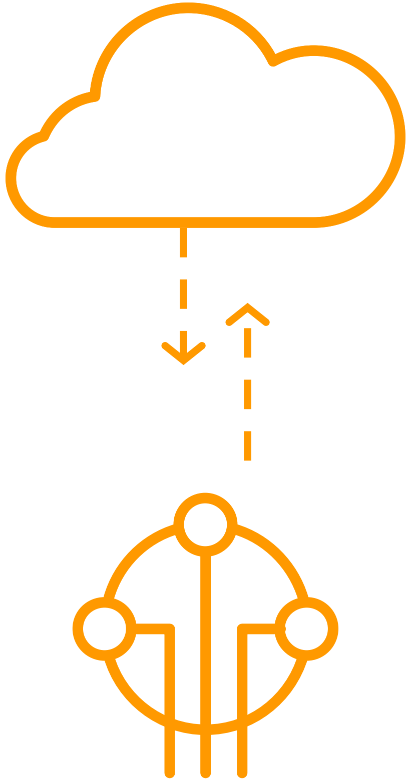
```
aws ecr get-login --registry-ids 216483018798 --no-include-email --region us-west-2
```

  https://hub.docker.com/r/amazon/aws-iot-greengrass

- Dockerfile available via CloudFront
  https://d1onfpft10uf5o.cloudfront.net/greengrass-core/downloads/1.8.0/aws-greengrass-docker-1.8.0.tar.gz

Snap:

- AWS IoT Greengrass snap will launch to GA on 4/1 at
  https://dashboard.snapcraft.io/snaps/aws-iot-greengrass/



Thank you!