Edge IoT with AWS and Yocto Linux

FIIF 29.4.2021
Kenneth Falck <kennu@nordcloud.com>

Nordcloud
an IBM Company

Introduction

About me.

Kenneth Falck <kennu@nordcloud.com>

Principal Architect at Nordcloud

Worked with

- IoT & Edge since 2016
- Serverless since 2014
- AWS Cloud since 2011
- Internet since 1995

Twitter: @kennu

About Nordcloud, an IBM Company

European Leader in Public Cloud.

Local presence in **10 countries**.

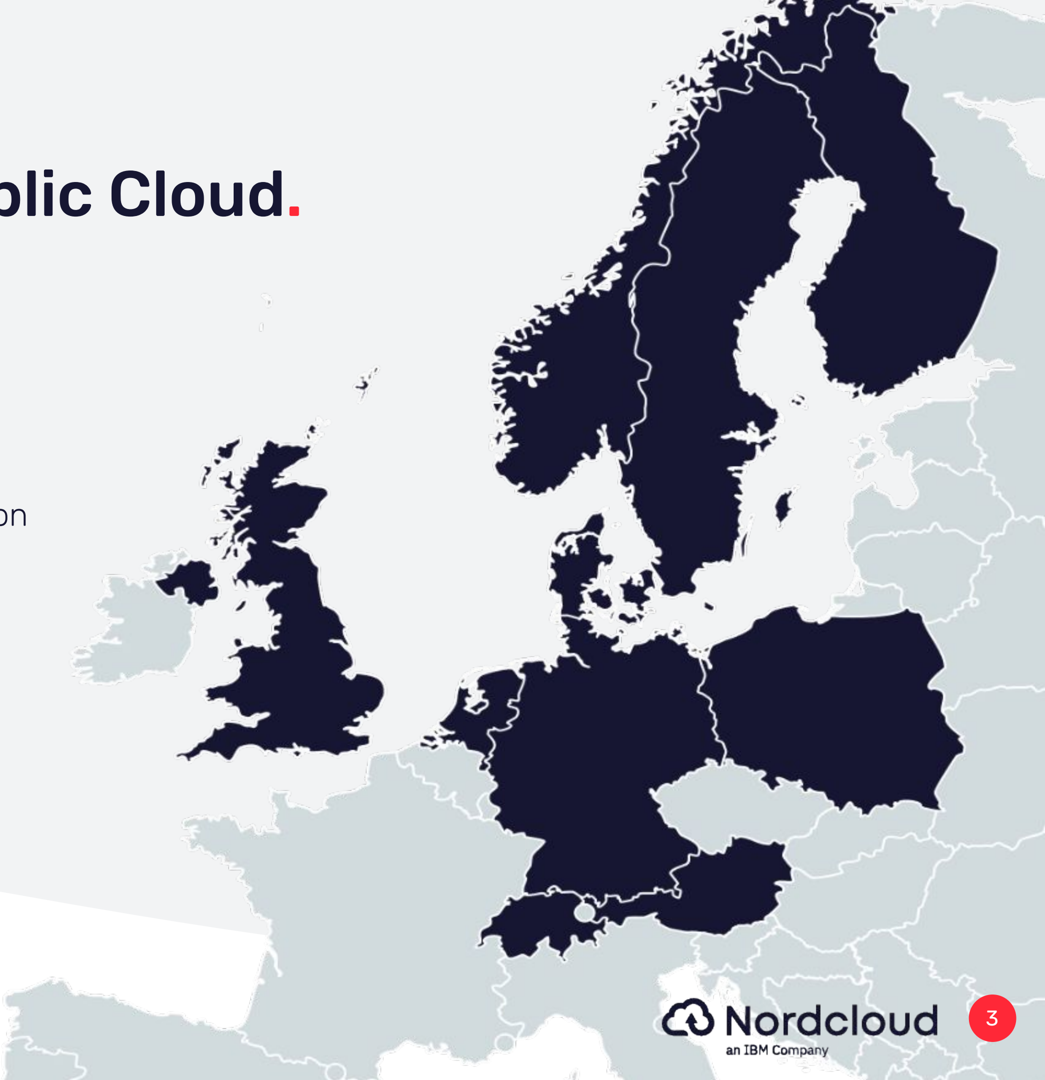
Nordcloud was born in the cloud 2011 and has grown to be the European leader in public cloud infrastructure solutions and cloud native application services. IBM announced its acquisition of Nordcloud on December 21, 2020.

We have the highest accreditations with all three cloud hyperscalers: AWS, Azure and GCP.

Our highly skilled organisation comprises of:

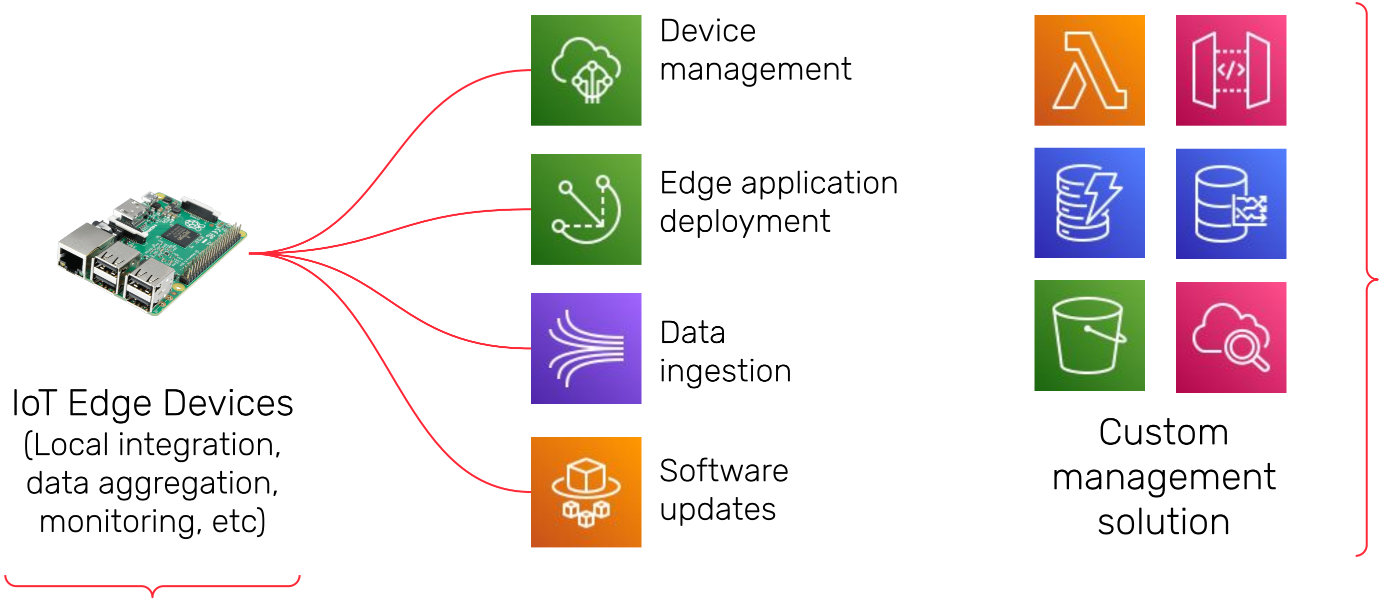
500+ cloud experts with

300+ certifications



Agenda

Overview of an IoT Edge System.



Device management

Edge application deployment

Data ingestion

Software updates

IoT Edge Devices (Local integration, data aggregation, monitoring, etc)

Scale in 1000s or 10000s of devices

Custom management solution

No servers! Use managed cloud services.

Agenda

Topics to discuss.

IoT Devices

Hardware architectures

Yocto Linux development

yocto PROJECT

Cloud Management

AWS IoT & device provisioning

Software updates

aws

MENDER

Edge Applications

Greengrass applications

Docker applications

IoT Devices

Hardware architectures.

Focus here

Microcontrollers

Need low-level programming

Linux-capable ARM

Low-power, low-cost edge applications

PCs

Large, expensive, warm, etc.

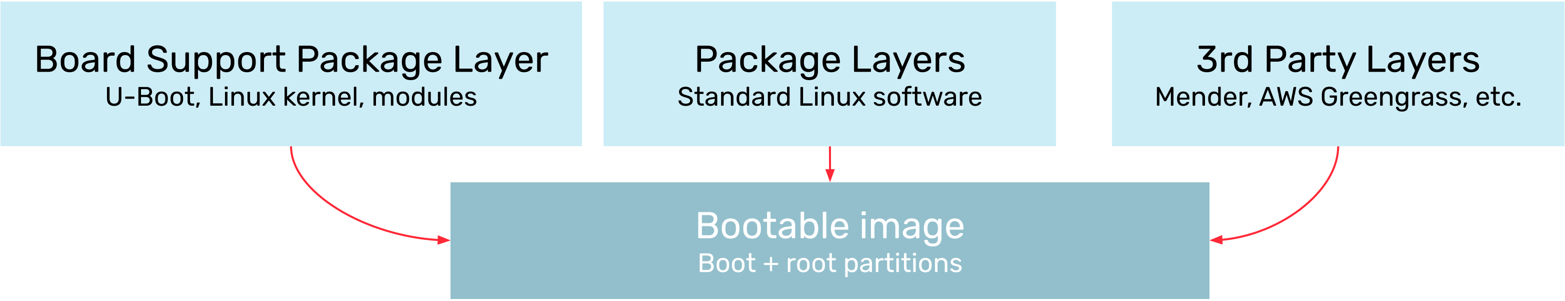
Tip: **AWS Partner Device Catalog** is a database of AWS IoT compatible devices: https://devices.amazonaws.com.

Yocto Linux.

Yocto Linux creates a **custom Linux distribution** tailored for our device.

Steep learning curve, but also the industry standard.

- Strange terminology (“layers”, “recipes”, “packages”, “bsp”, “meta”, etc.).
- Build process is hacky, based on shell scripts and custom syntax.
- **Officially supported by hardware manufacturers**, but often need older version.



Board Support Package Layer
U-Boot, Linux kernel, modules

Package Layers
Standard Linux software

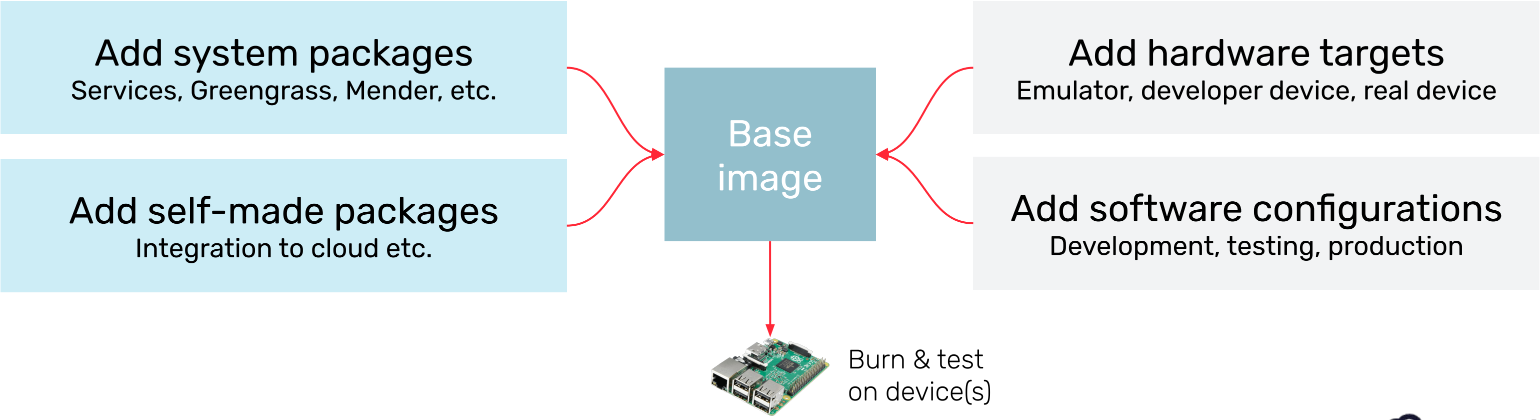
3rd Party Layers
Mender, AWS Greengrass, etc.

Bootable image
Boot + root partitions

Development & build process.

Start with a simple base image that supports our hardware.

- Add packages to implement functionality.
- Add build targets to support different environments.
- Note: This is the **application platform** – actual applications will deploy later on top of it.



Add system packages
Services, Greengrass, Mender, etc.

Add self-made packages
Integration to cloud etc.

Base image

Add hardware targets
Emulator, developer device, real device

Add software configurations
Development, testing, production

Burn & test
on device(s)

Cloud Management

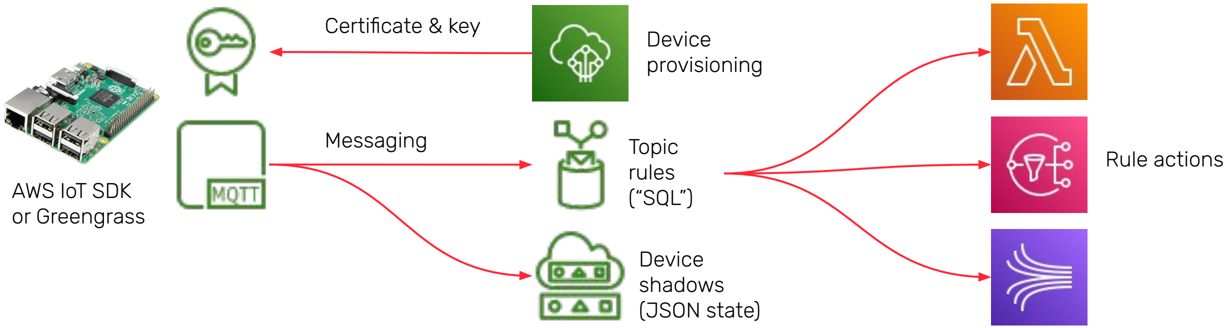
AWS IoT.

AWS IoT Core manages IoT devices and acts as message broker.

- Register each device as a “Thing” in built-in database.
- Authenticate devices that connect via MQTT.
- Process MQTT messages (which are organized under Topics).

AWS IoT is fully managed and serverless!

Pay per messages and connection minutes



Other AWS IoT services like Greengrass, IoT Analytics, IoT Events, etc. are built on IoT Core.

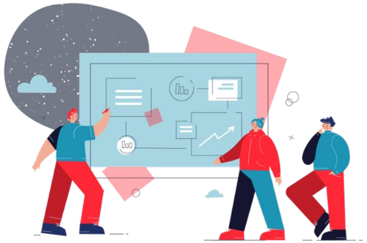
Cloud Management

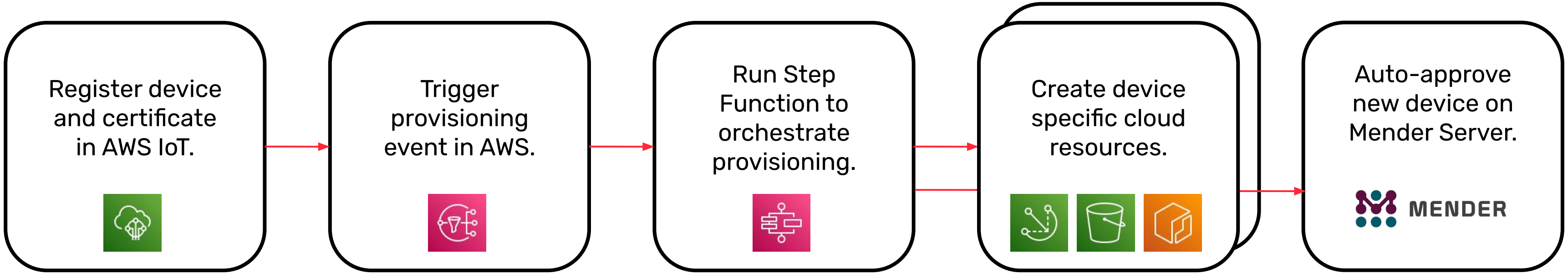
Device provisioning.

We need an **automated solution** to provision **a large number of devices**.

How to authenticate devices for auto-registration before we provision a certificate?

- Option A: Device image includes an **auto-registration code** that allows it to auto-register on first boot.
- Option B: **"Factory guy"** at the factory **creates certificates** with OpenSSL and Private CA (complicated).





Register device and certificate in AWS IoT.

Trigger provisioning event in AWS.

Run Step Function to orchestrate provisioning.

Create device specific cloud resources.

Auto-approve new device on Mender Server.

Cloud Management

Software updates.

Mender provides **reliable** and **efficient** software updates.

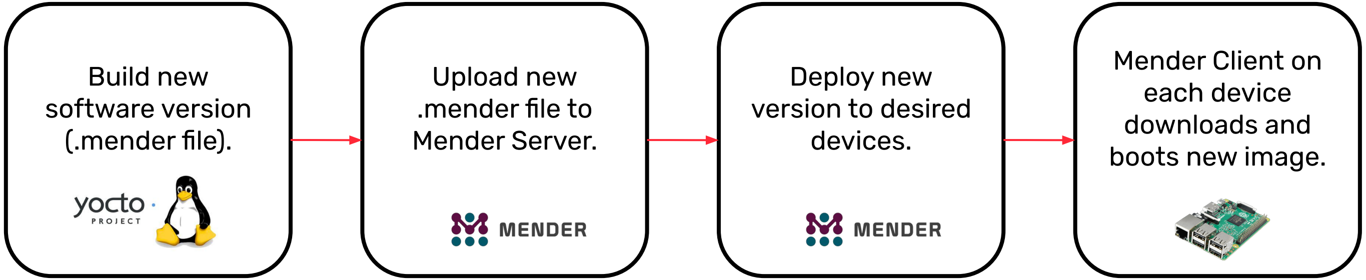
- Download new image directly to partition B (avoid disk full errors).
- Keep previous version on Partition A and boot new image from Partition B.
- Integrate with U-Boot, fallback to partition A on failure.

Application Updates

AWS Greengrass
(no reboot)

OS Updates

MENDER



Build new software version (.mender file).

Upload new .mender file to Mender Server.

Deploy new version to desired devices.

Mender Client on each device downloads and boots new image.

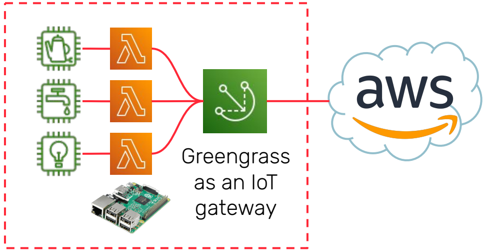
Tip: Mender is open source but offers **hosted cloud service** if you don't want to self-host. Paid version has incremental updates (smaller images).

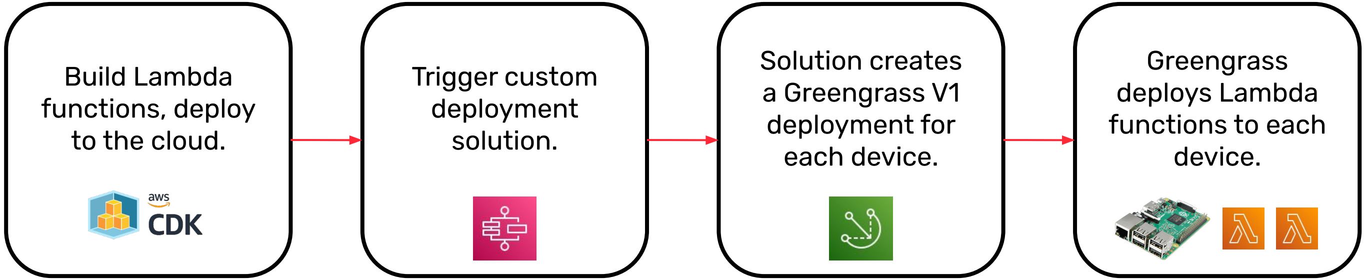
Greengrass edge applications.

Greengrass (V1) is a **MQTT message broker** that deploys and runs **Lambda functions** locally on devices**.**

Our edge applications are Lambda functions (Python or TypeScript modules).

- Functions communicate with MQTT messages.
- Functions access local hardware to integrate with sensors and actuators.
- Greengrass provides built-in functions (Modbus, GPIO, etc).





Build Lambda functions, deploy to the cloud.

Trigger custom deployment solution.

Solution creates a Greengrass V1 deployment for each device.

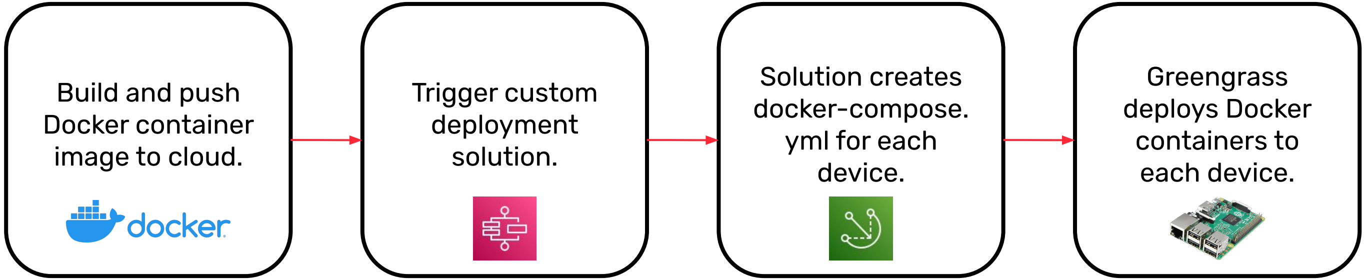
Greengrass deploys Lambda functions to each device.

Tip: **Greengrass V2** was released recently with a new, simplified component model.

Docker edge applications.

Sometimes you want to run **Docker containers** instead of Greengrass Lambda functions.

- Greengrass (V1) can deploy **docker-compose.yml** files to devices and start the containers.
- Greengrass configures permissions so Docker can pull images from AWS ECR.



Build and push Docker container image to cloud.

Trigger custom deployment solution.

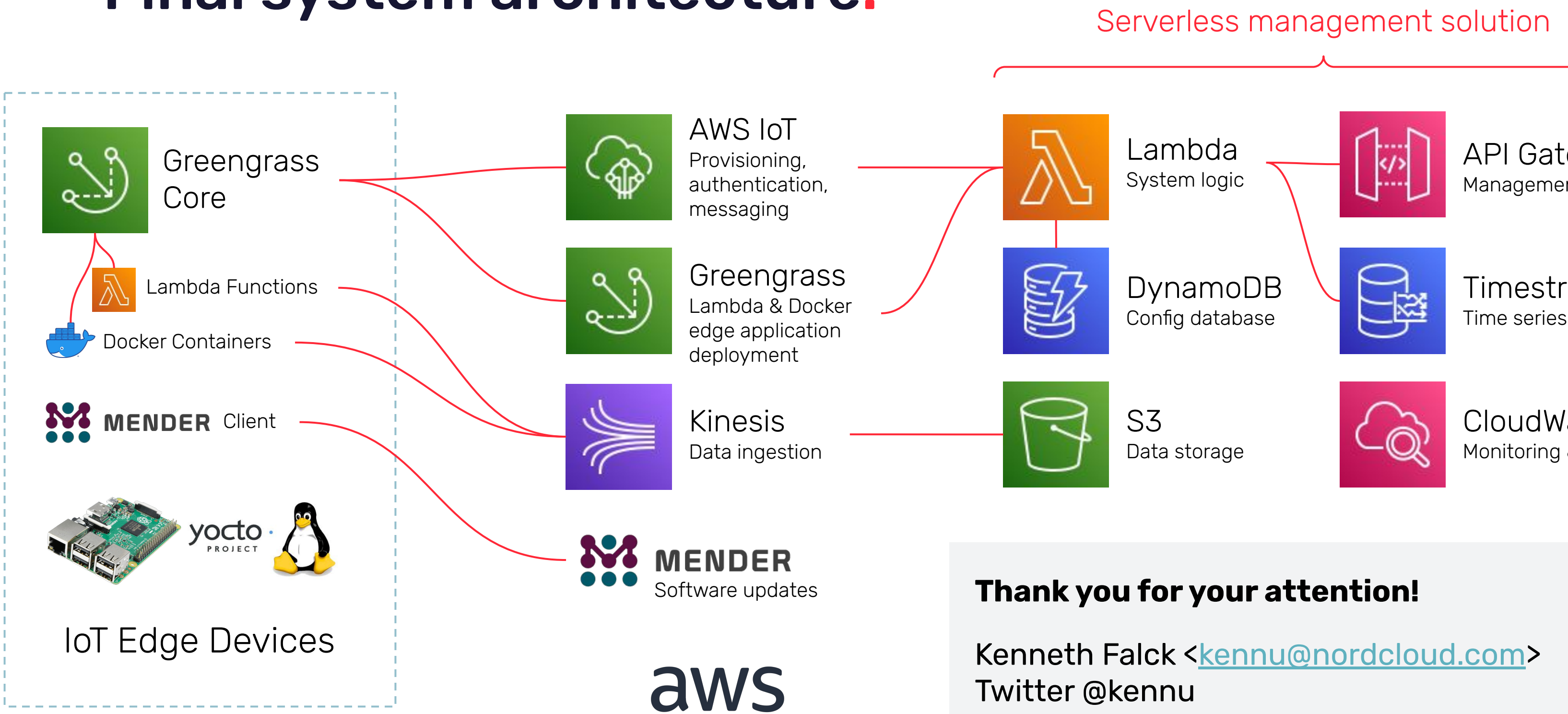
Solution creates docker-compose.yml for each device.

Greengrass deploys Docker containers to each device.

Tip: **Greengrass V2** has a more unified and simplified model for deploying Docker containers. All components are basically equal (functions, containers, shell scripts, any executable).

Summary

Final system architecture.



Serverless management solution

Greengrass Core

Lambda Functions

Docker Containers

MENDER Client

IoT Edge Devices

AWS IoT
Provisioning, authentication, messaging

Greengrass
Lambda & Docker edge application deployment

Kinesis
Data ingestion

MENDER
Software updates

Lambda
System logic

DynamoDB
Config database

S3
Data storage

API Gateway
Management API

Timestream
Time series database

CloudWatch
Monitoring & logging

Thank you for your attention!

Kenneth Falck <kennu@nordcloud.com>
Twitter @kennu