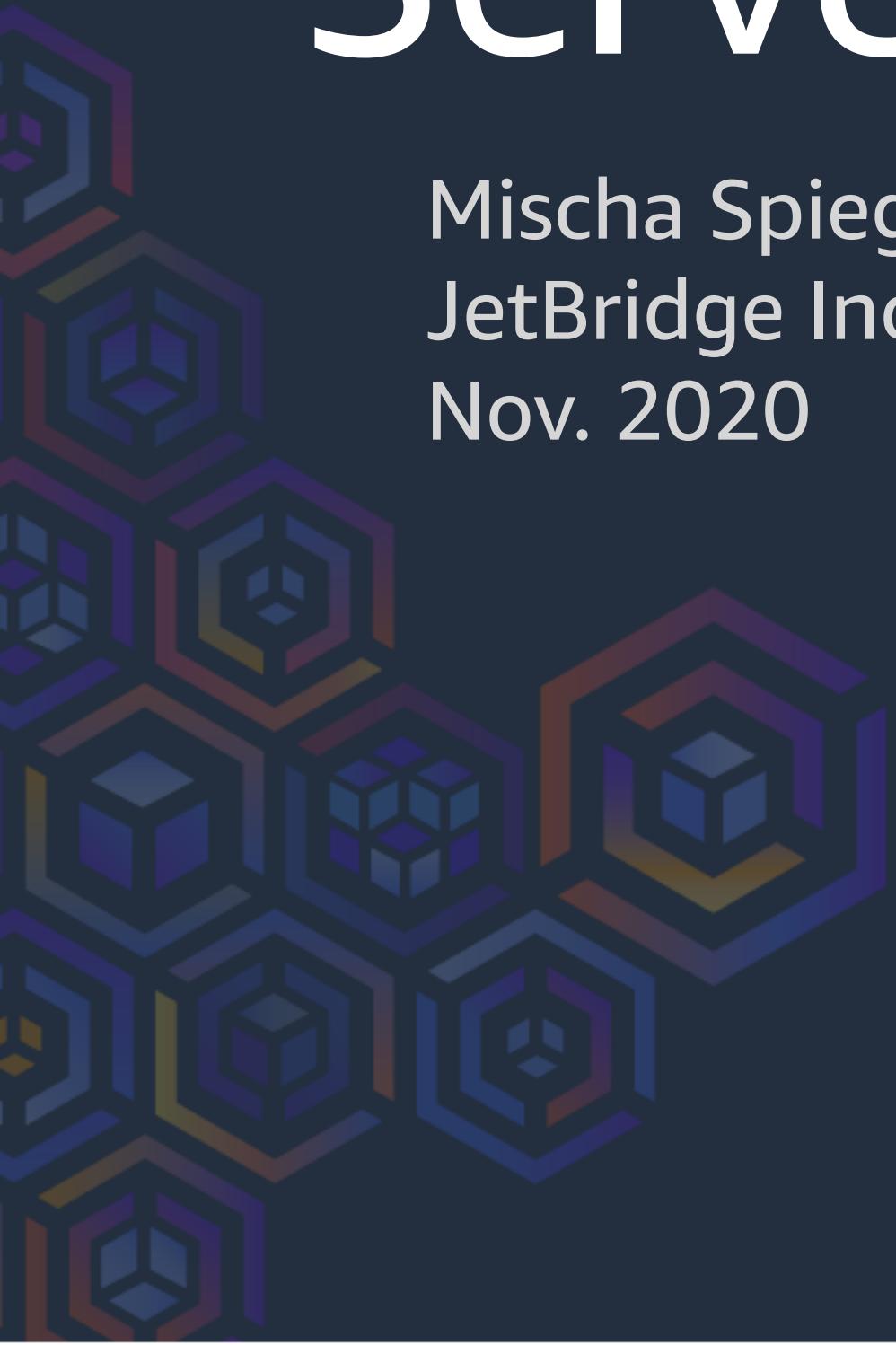




COMMUNITY DAY

Serverless AWS IoT

Mischa Spiegelmock
JetBridge Inc.
Nov. 2020



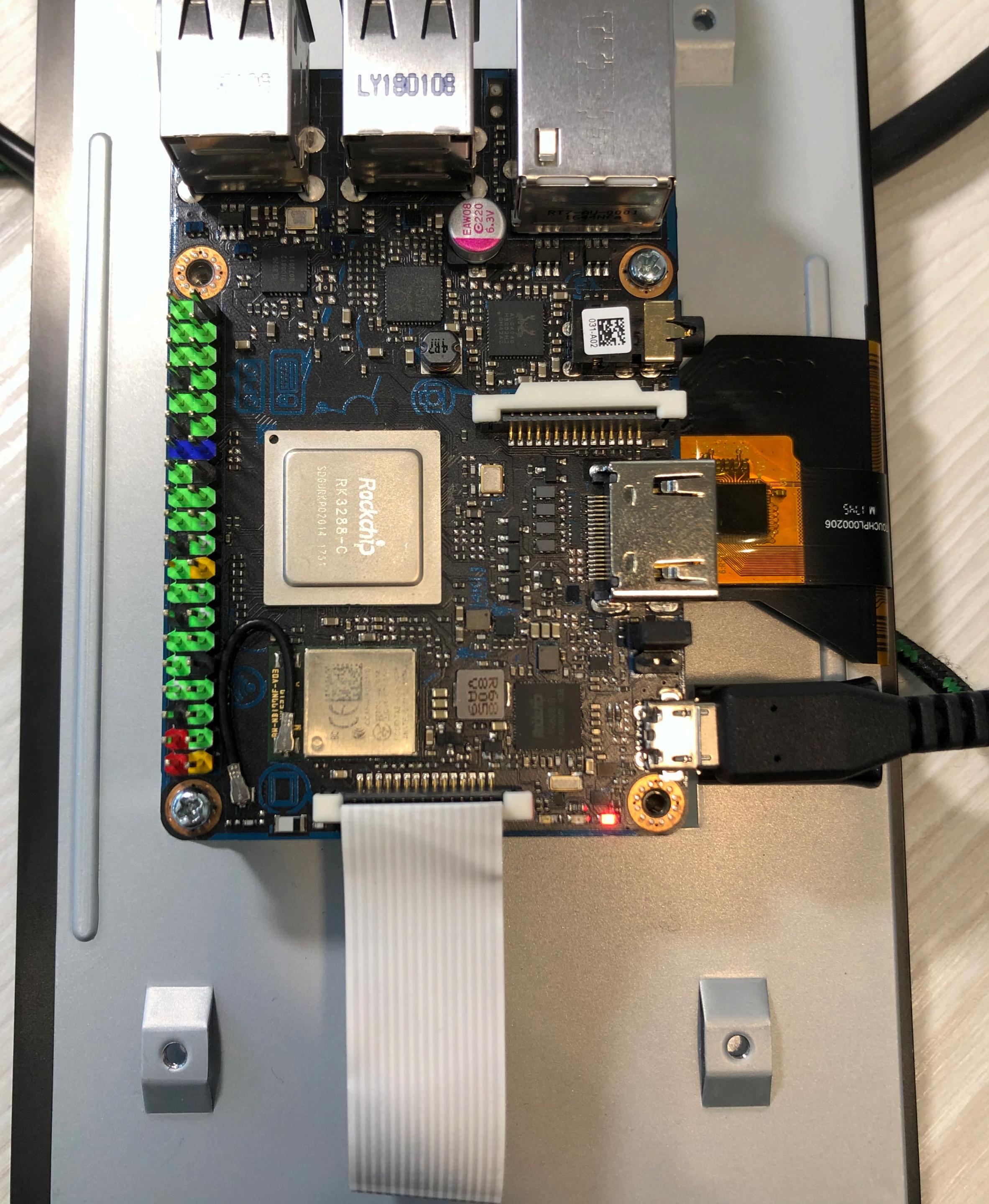


COMMUNITY DAY

About Me

- Berkeley, CA native
- Software Engineer
- AWS Solutions Architect (Assoc.)
- Design and engineer cloud-native applications
- Founder, JetBridge Inc.





```
root@tinkerboard:~# xinput list
↳ Virtual core pointer
↳ Virtual core XTEST pointer
↳ fts_ts
↳ Generic USB Audio
↳ DaKai 2.4G RX
↳ Virtual core keyboard
↳ Virtual core XTEST keyboard
↳ gpio-keys
↳ Generic USB Audio
↳ DaKai 2.4G RX
↳ DaKai 2.4G RX
root@tinkerboard:~#
root@tinkerboard:~# xrandr
Screen 0: minimum 320 x 200, current 800 x 480, maximum 8192 x 8192
DSI-1 connected primary 800x480+0+0 (normal left inverted right x axis y axis) 0
mm x 0mm
    800x480      58.76*
HDMI-1 disconnected (normal left inverted right x axis y axis)
root@tinkerboard:~#
```



Serverless + IoT?

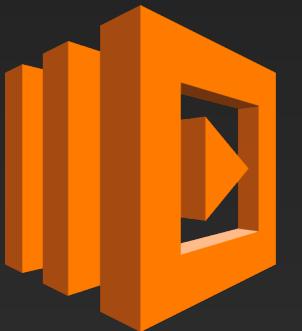
Technologies



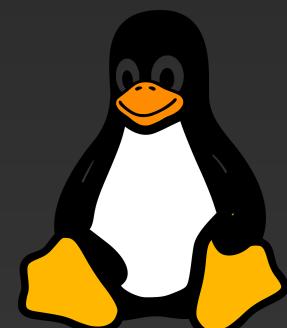
- **AWS IoT Core** - Thing registry, MQTT



- **AWS IoT Greengrass** - deploy + run code on device

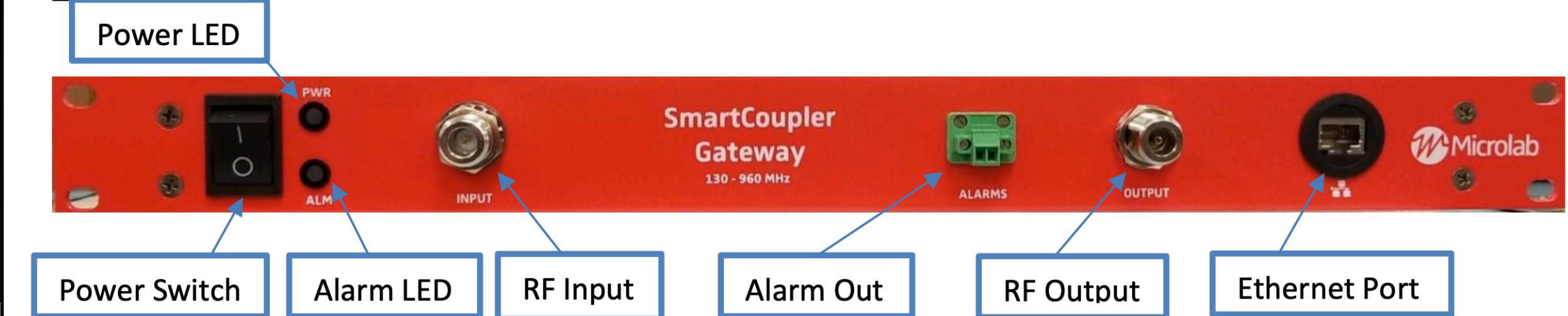
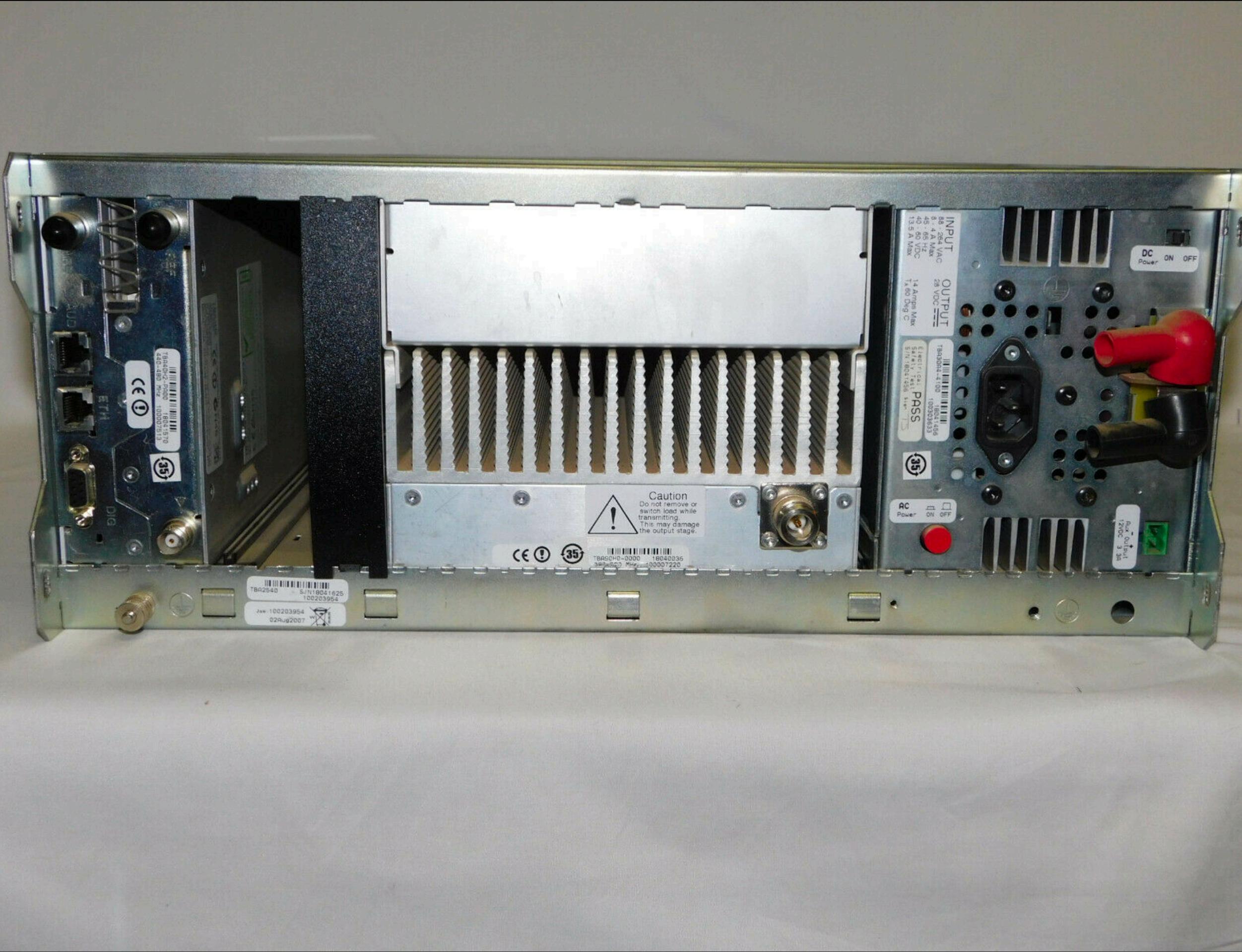


- **AWS Lambda** - define functions



- **Linux-based devices**

System: Remote Monitoring

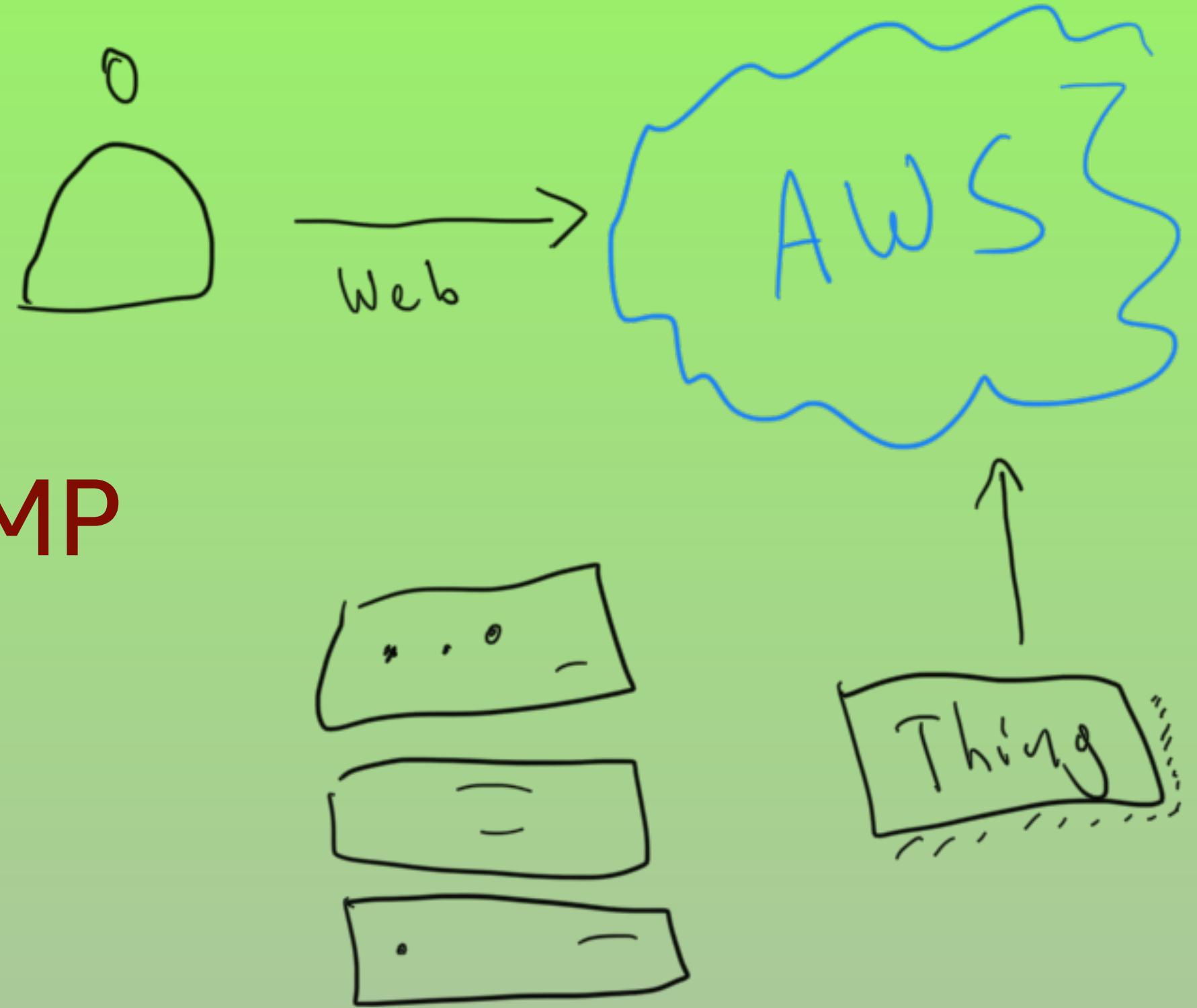


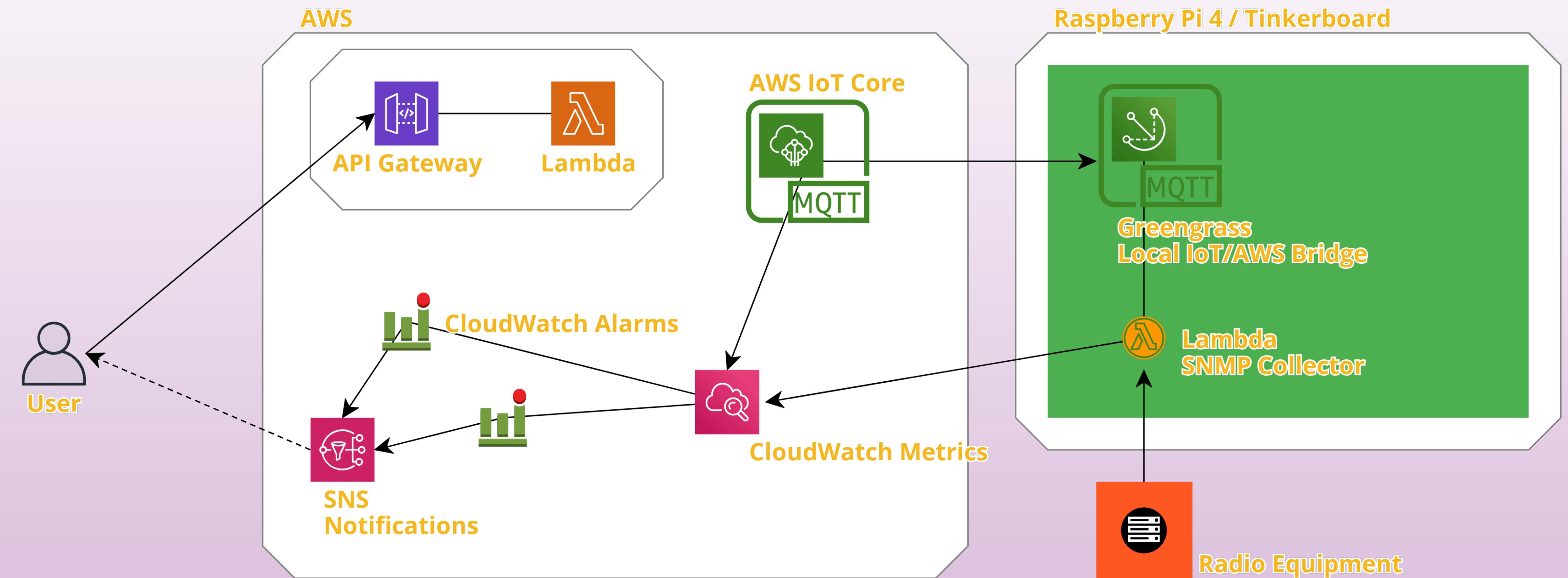
Rear (AC)



Deliverables

- **Monitor** radio hardware on-site via SNMP
 - (RFC1067 and friends)
- **Report** telemetry and faults to cloud
- **View** telemetry and faults in a web interface





Challenges

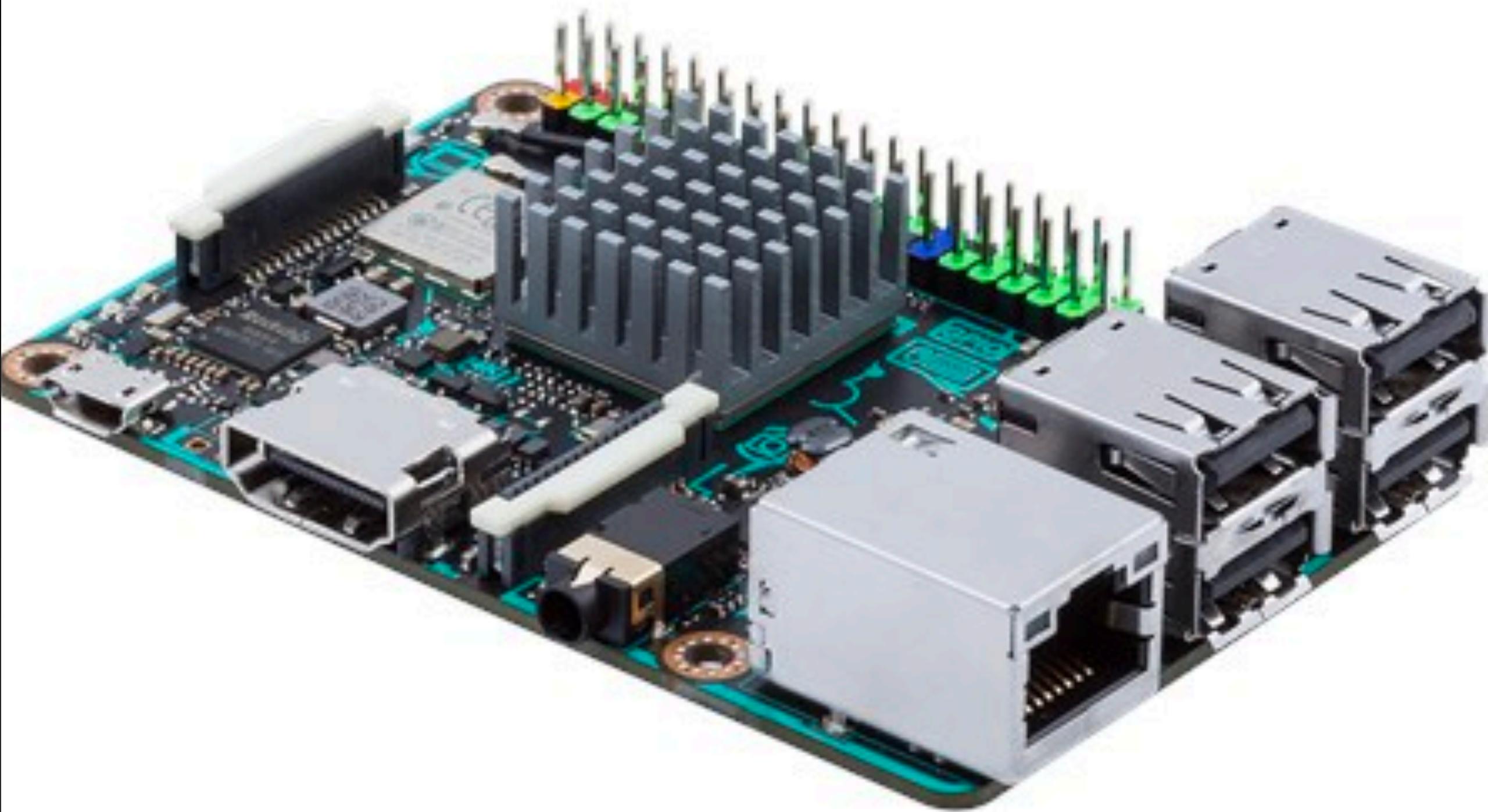
- Hardware
- OS image
- Provisioning
- Deployments / Updates
- Monitoring
- Fix stuff remotely
- GUI

```
23 function setup_greengrass {  
24     adduser --system --uid 1200 ggc_user  
25     groupadd --system --gid 1200 ggc_group  
26  
27     # initscript  
28     cat <<EOF >/etc/init.d/S02greengrass  
29 #!/bin/sh  
30 cd /greengrass/ggc/core  
31 ./greengrassd \$@  
32 EOF  
33 chmod 755 /etc/init.d/S02greengrass  
34  
35     # systemd unit file  
36     cat <<EOF >/etc/systemd/system/greengrass.service  
37 [Unit]  
38 Description=AWS Greengrass Service  
39 After=network.target  
40  
41 [Service]  
42 Type=simple  
43 ExecStart=/etc/init.d/S02greengrass start  
44 Restart=always  
45 RestartSec=120  
46 User=root  
47 PIDFile=/run/greengrassd.pid  
48  
49 [Install]  
50 WantedBy=multi-user.target  
51 EOF  
52     # start GG at startup  
53     systemctl enable greengrass.service  
54  
55     # install greengrass application  
56     tar Czxf / /tmp/overlay/greengrass-linux-armv7l-$GG_VERSION.tar.gz  
57  
58     # add root CA  
59     mkdir -p /greengrass/certs  
60     cp /tmp/overlay/root.ca.pem /greengrass/certs/  
61 }  
62  
63 function setup_ssh {  
64     systemctl enable ssh  
65  
66     mkdir -p /root/.ssh  
67     chmod 700 /root/.ssh
```

Hardware

IoT “Thing”

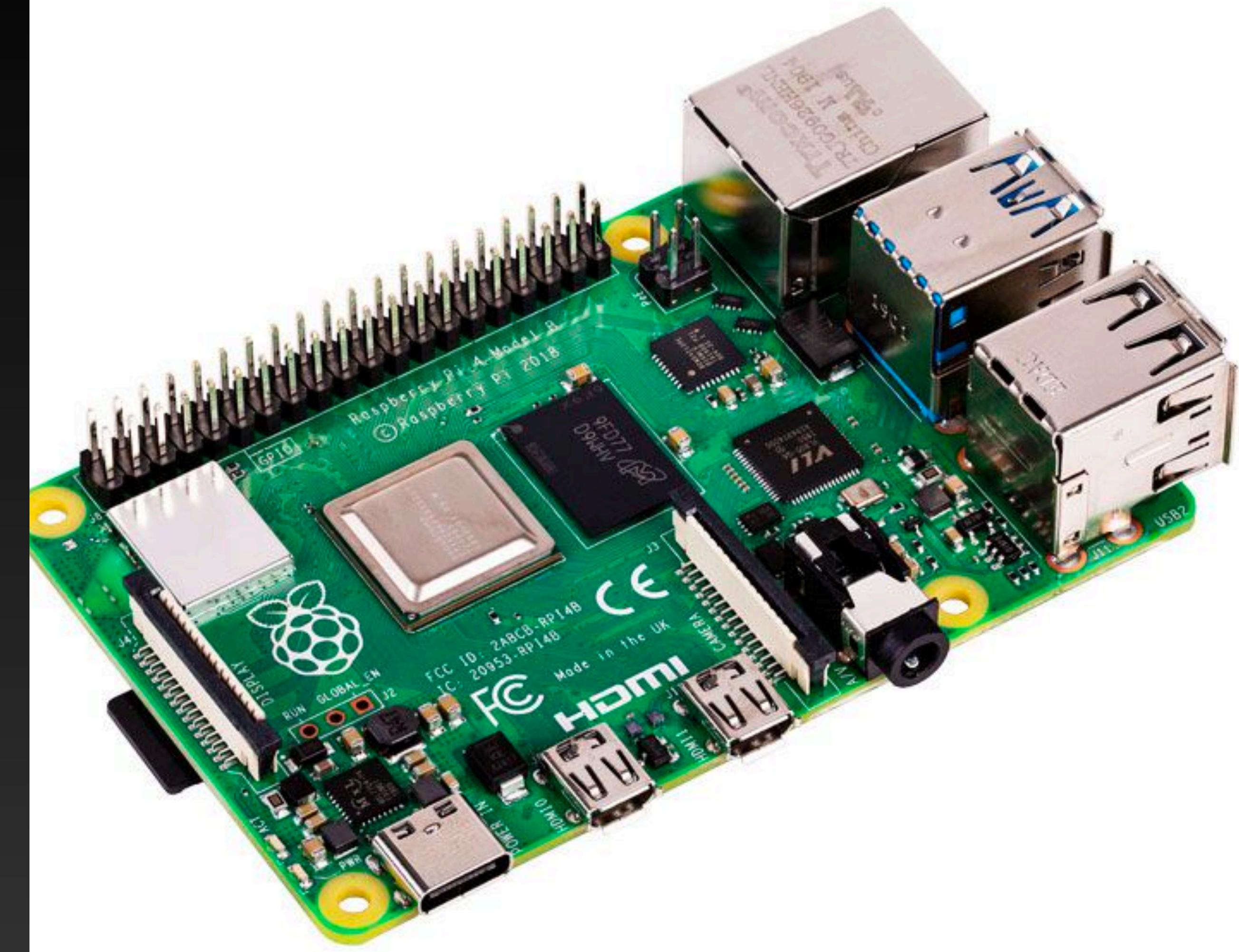
- Raspberry Pi
- Asus Tinkerboard S
- AWS Greengrass Client
- Debian OS image
- Provisioning



Raspberry Pi

Well-supported

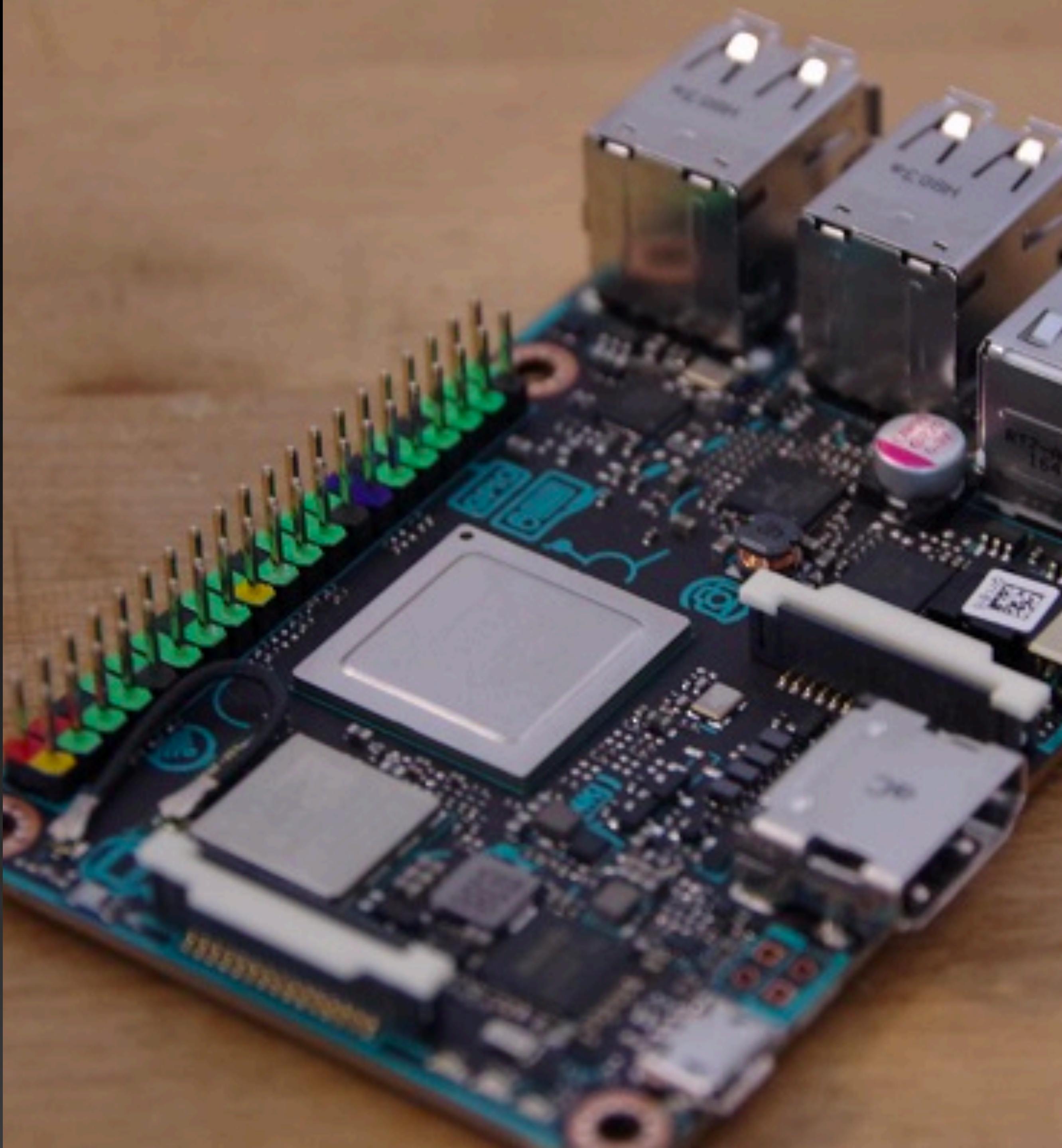
No internal flash



Asus Tinkerboard S

👍 Internal flash

👎 No pi-gen



Pi-gen
Armbian
meta-aws
Yocto
Buildroot

...

OS Image

- Automated build (shell)
- Configure
 - Networking
 - Xorg
 - SSH
 - OS packages
- Greengrass



AWS IoT Core

The screenshot shows the AWS IoT Core interface for managing Things. On the left, a sidebar menu lists various services: Monitor, Activity, Onboard (Get started, Fleet provisioning templates), Manage (Things, Types, Thing groups, Billing groups, Jobs, Tunnels), Greengrass, Secure, and Defend. The main content area is titled "AWS IoT > Things" and displays a list of registered things. A search bar at the top allows filtering by name. The "Fleet Indexing" tab is currently selected, while "Info" is also available. The table lists two entries:

| Name | Type | Actions |
|------|---------|---------|
| 87ac | NO TYPE | ... |
| aadd | NO TYPE | ... |

IoT Core

- “Thing” registry
- Authentication
 - Certificate - mutual TLS
 - Authorization - IAM policy document
- MQTT - messaging
- Shadow - fancy configuration
- Tunnels - remote administration

AWS Greengrass

AWS IoT > Greengrass > Groups > b2473926-8090-4405-b8a5-91c213fefc6c

GREENGRASS GROUP
aadd
● Successfully completed

Actions ▾

Deployments Group history overview By deployment ▾

| Subscriptions | Deployed | Version | Status |
|---------------|------------------------------------|--------------------------------------|-----------------------------|
| Cores | October 14, 2020, 16:07:09 (UT...) | 81dda7e0-a0fd-43f3-a978-0dc29f1b2a42 | ● Successfully complet... ⋮ |
| Devices | October 14, 2020, 14:21:35 (UT...) | 517366ed-35db-4491-85ef-97ae36ccba0f | ● Successfully complet... ⋮ |
| Lambdas | October 14, 2020, 14:19:19 (UT...) | 5dfdf56e-b9ea-4d5b-880c-be5da66f2de8 | ● Successfully complet... ⋮ |
| Resources | October 14, 2020, 14:12:27 (UT...) | 9a17a243-cea7-4937-a2ee-09cb03b152a2 | ● Successfully complet... ⋮ |
| Connectors | | | |
| Tags | | | |
| Settings | October 14, 2020, 13:28:25 (UT...) | 9a568922-5eba-43b5-a23f-65d71af29acd | ● Successfully complet... ⋮ |

Intro

Groups

Cores

Devices

Secure

Defend

Act

Greengrass

“Run Lambdas On My Device”

- Client-side gateway (“core”)
- MQTT client/server
- Lambda runner
- Code deployment client
- Subscriptions - Thing ↔ AWS

Greengrass Configuration

- Config
 - Root CA (Amazon)
 - Private key
 - Certificate
- Core Thing
 - Thing ARN
 - AWS endpoint

```
{  
    "coreThing" : {  
        "caPath" : "root.ca.pem",  
        "certPath" : "hash.cert.pem",  
        "keyPath" : "hash.private.key",  
        "thingArn" : "arn:partition:iot:region:account-id:thing/core-thing-name",  
        "iotHost" : "host-prefix-ats.iot.region.amazonaws.com",  
        "ggHost" : "greengrass-ats.iot.region.amazonaws.com",  
        "keepAlive" : 600,  
        "ggDaemonPort": 8000,  
        "systemComponentAuthTimeout": 5000  
    },  
    "runtime" : {  
        "maxWorkItemCount" : 1024,  
        "cgroup" : {  
            "useSystemd" : "yes"  
        }  
    },  
    "managedRespawn" : false,  
    "crypto" : {  
        " principals" : {  
            "SecretsManager" : {  
                "privateKeyPath" : "file:///greengrass/certs/hash.private.key"  
            },  
            "IoTCertificate" : {  
                "privateKeyPath" : "file:///greengrass/certs/hash.private.key",  
                "certificatePath" : "file:///greengrass/certs/hash.cert.pem"  
            }  
        },  
        "caPath" : "file:///greengrass/certs/root.ca.pem"  
    }  
}
```

AWS IoT Authentication

X.509 MTLS

- Certificate provisioned:
 - Client Key
 - Client Certificate
 - Policy
 - Thing Attachment

AWS IoT > Certificates > 79cdeb5190aa688d76c21baee5bd705064d425b2bc8d8ccde4b9a3c5280447c2

CERTIFICATE
79cdeb5190aa688d76c21baee5bd705064d425b2bc8d8ccde4b9a3c5280447c2
ACTIVE

Actions ▾

| | |
|----------------|---|
| Details | Certificate ARN |
| Policies | A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. Learn more |
| Things | <code>arn:aws:iot:us-east-1:794441982760:cert/79cdeb5190aa688d76c21baee5bd705064d425b2bc8d8ccde4b9a3c5280447c2</code> |
| Non-compliance | |

Details

Issuer
OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US

Subject
CN=AWS IoT Certificate

Create date
October 16, 2020, 16:40:54 (UTC+0300)

Effective date
October 16, 2020, 16:38:54 (UTC+0300)

Expiration date
January 01, 2050, 01:59:59 (UTC+0200)

Policies

| | |
|---------------|-----|
| mm_fe80_305c0 | ... |
|---------------|-----|

Things

| | |
|------|-----|
| fe80 | ... |
|------|-----|

Provisioning

Connecting a Thing

- Create Certificates
- Create Thing
- Create Group Version
- Create Group
- Generate Greengrass config
- Create Deployment

```
certs = self.create_certs()
thing = self.create_thing()
self.attach_thing_policy(certs=certs)

# create initial Group version
group_initial_version = self.generate_group_version(
    thing_arn=thing["thingArn"], certificate_arn=certs["certificateArn"]
)

# create Group
group = self.create_group(initial_version=group_initial_version)

# set thing shadow
initial_shadow = {}
if self.remote_addr:
    # save IP of who requested the provisioning
    initial_shadow["provision_addr"] = self.remote_addr
self.update_thing_shadow(initial_shadow)

# link thing to group - for our use later
iot.update_thing(
    thingName=self.name,
    attributePayload={
        "attributes": {THING_GROUP_ID_ATTR: group["Id"]},
        "merge": True,
    },
)

# perform initial deployment
self.create_deployment(
    thing_name=thing["thingName"],
    group=group,
    group_version_id=group["LatestVersion"],
)

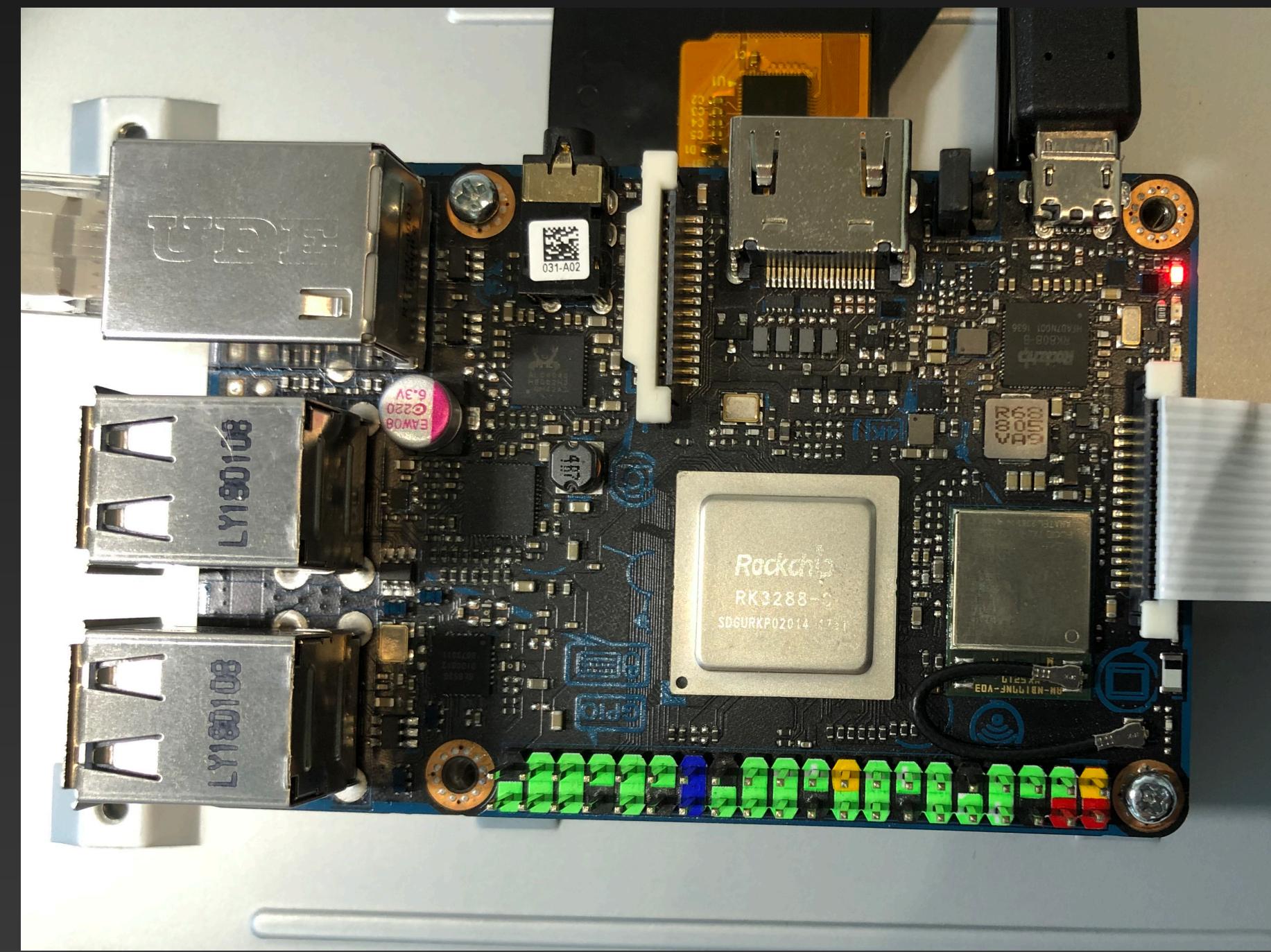
config_doc = self.gen_config_doc(thing=thing, group=group)
return {
    "thing": thing,
    "group": group,
    "certificatePem": certs["certificatePem"],
    "publicKey": certs["keyPair"]["PublicKey"],
    "privateKey": certs["keyPair"]["PrivateKey"],
    "ggConfig": config_doc,
}
```

Install Greengrass

- .tar.gz
- Debian APT
- Docker
- Snap

```
[ OK ] Stopped target Network.
[ OK ] Stopping WPA supplicant.
[ OK ] Stopping Ralink network interface.
[ OK ] Stopping Iffup for eth0...
[ OK ] Stopped WPA supplicant.
[ OK ] Stopping D-Bus System Message Bus.
[ OK ] Stopped D-Bus System Message Bus.
[ OK ] Stopped target Basic System.
[ OK ] Stopped target sockets.
[ OK ] Closed syslog socket.
[ OK ] Stopped target Paths.
[ OK ] Stopped target Services.
[ OK ] Removed slice User and Session.
[ OK ] Closed D-Bus System Message Bus.
[ OK ] Stopped target System Initialization.
[ OK ] Stopping Restore / save the current system state.
[ OK ] Stopping Update UTMP about System.
[ OK ] Stopping Armbian memory support.
[ OK ] Stopping Load/Save Random Seed.
[ OK ] Stopped target Local Encrypted.
[ OK ] Stopped Forward Password Request.
[ OK ] Restore / save the current system state.
[ OK ] Stopped Load/Save Random Seed.
[ OK ] Stopped Load/Save Random Seed.
[ OK ] Stopped Update UTMP about System.
[ OK ] Stopped target Local Encrypted.
```

Now What?



Greengrass Deployments

- CreateDeployment
 - GroupVersionId
- CreateGroupVersion

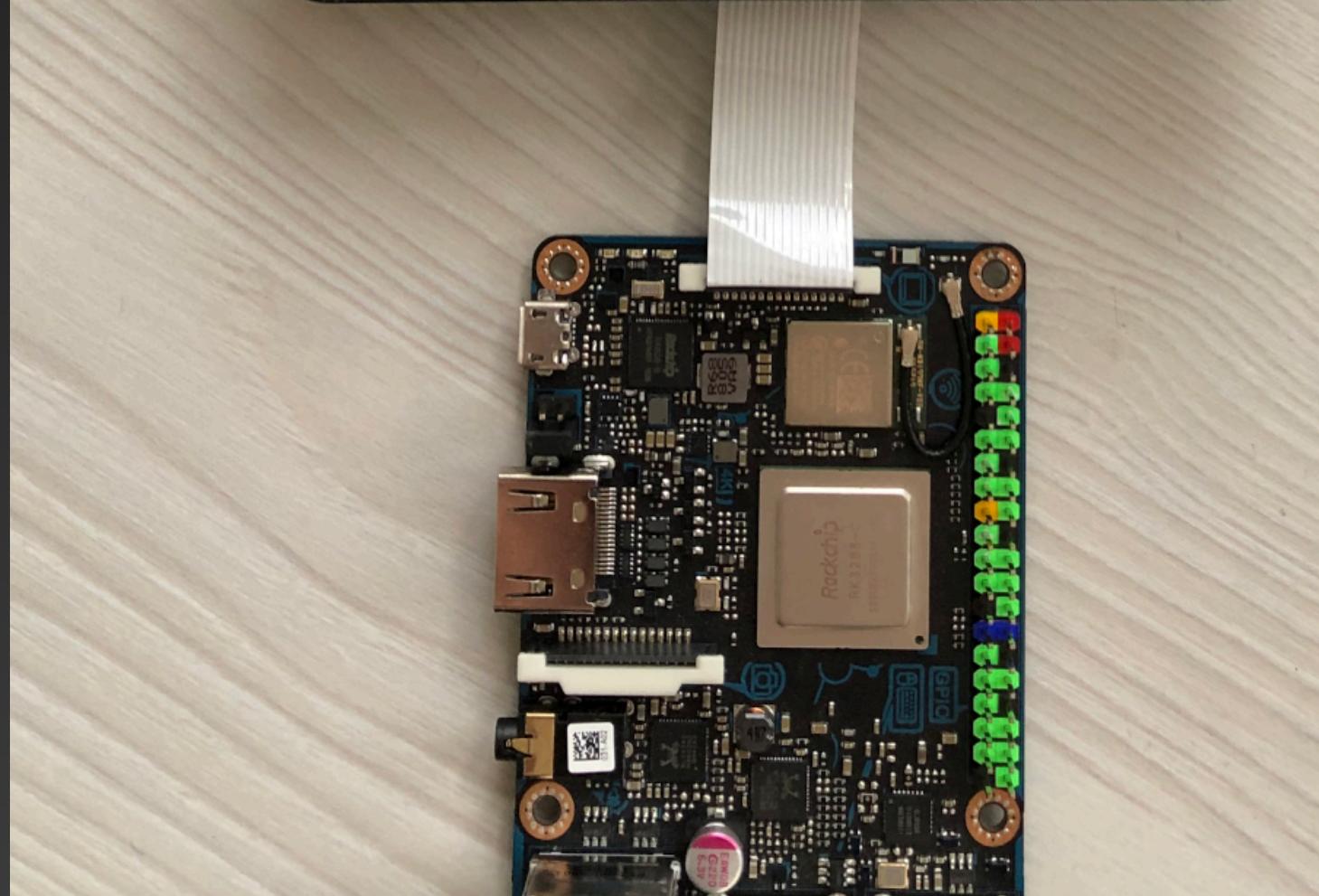
```
{  
    "GroupId": "string",  
    "CoreDefinitionVersionArn": "string",  
    "DeviceDefinitionVersionArn": "string",  
    "FunctionDefinitionVersionArn": "string",  
    "SubscriptionDefinitionVersionArn": "string",  
    "LoggerDefinitionVersionArn": "string",  
    "ResourceDefinitionVersionArn": "string",  
    "ConnectorDefinitionVersionArn": "string",  
    "AmznClientToken": "string"  
}
```

| GREENGRASS GROUP | | | |
|---|--|------------------------|---------|
| aadd | | | |
| Successfully completed | | | |
| Deployments | | Group history overview | |
| Subscriptions | | Deployed | Version |
| Cores | | Status | |
| Devices | | Successfully complet.. | |
| Lambdas | | Successfully complet.. | |
| Resources | | Successfully complet.. | |
| Connectors | | Successfully complet.. | |
| Tags | | Successfully complet.. | |
| Settings | | Successfully complet.. | |
| October 15, 2020, 23:07:45 (UT... 373608c3-113c-4fa7-b8cd-5129d3ad33d4 | | Successfully complet.. | |
| October 15, 2020, 22:57:05 (UT... 8298ba00-4b03-4308-a678-66f824485bb6 | | Successfully complet.. | |
| October 15, 2020, 17:08:10 (UT... ac9d3ee2-e89d-44cb-95fb-2361764dcdf9 | | Successfully complet.. | |
| October 14, 2020, 16:07:09 (UT... 81dda7e0-a0fd-43f3-a978-0dc29f1b2a42 | | Successfully complet.. | |
| October 14, 2020, 14:21:35 (UT... 517366ed-35db-4491-85ef-97ae36ccba0f | | Successfully complet.. | |
| October 14, 2020, 14:19:19 (UT... 5fdfdf56e-b9ea-4d5b-880c-be5da66f2de8 | | Successfully complet.. | |
| October 14, 2020, 14:12:27 (UT... 9a17a243-cea7-4937-a2ee-09cb03b152a2 | | Successfully complet.. | |
| October 14, 2020, 13:28:25 (UT... 9a568922-5eba-43b5-a23f-65d71af29acd | | Successfully complet.. | |
| October 14, 2020, 13:02:45 (UT... ecc9cd88-3f26-4340-8fa6-b137b305d063 | | Successfully complet.. | |
| October 14, 2020, 12:46:49 (UT... d9f50e24-b0f9-403b-a81b-a7ebbb960565 | | Successfully complet.. | |
| October 14, 2020, 12:15:28 (UT... 3727293a-7ed4-48f3-a801-3438750536bd | | Successfully complet.. | |
| October 14, 2020, 11:36:39 (UT... 867ae738-3f30-4762-a10c-589bd9aae3d5 | | Successfully complet.. | |
| October 14, 2020, 11:22:35 (UT... 5c7ceeff-7622-4515-ab50-e8bd023d68fb | | Successfully complet.. | |
| September 25, 2020, 00:13:24 (...) 6bb65957-c7ff-41bf-bed4-6a714254b610 | | Successfully complet.. | |

Function Definition

Lambdas on Things

- Put Lambdas on your Thing
- Special features:
 - UNIX UserID / GroupID
 - Run as root
 - Run in docker
 - Run forever



Function Definition

Lambdas on Things

- Put Lambdas on your Thing
- Special features:
 - UNIX UserID / GroupID
 - Run as root
 - Run in docker
 - Run forever

[View function in AWS Lambda](#)

Version 57 [Remove version](#)

Run as [Info](#)

Use group default (currently: 1/10)

Another user ID/group ID

UID (number)

0

GID (number)

0

Containerization [Info](#)

Use group default (currently: No container)

Greengrass container (always)

No container (always)

Timeout

10

Second ▾

Lambda lifecycle

On-demand function

Make this function long-lived and keep it running indefinitely

Input payload data type

JSON

Binary

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code.

Key

ALARM_SNS_TOPIC

Value

arn:aws:sns:us-east-1:7944419

Greengrass

Other goodies

- OTA updates
- Machine learning inference
- Data stream management (→ Kinesis, IoT Analytics, S3)
- Secrets (with Secrets Manager)
- Connectors (MQTT-driven functionality)

Monitoring



Logging

- Direct to CloudWatch:
 - Lambdas
 - Greengrass System

Configure Group logging

Greengrass Groups are capable of sending configurable logs for Greengrass system and user lambdas to Cloudwatch.

[View logs in CloudWatch](#)

User lambda logs

What level of logs should be sent?

Debug logs

[Remove log type](#)

Greengrass system logs

What level of logs should be sent?

Informational logs (recommended)

[Remove log type](#)

Add another type of log

[Add another log type](#)

[Cancel](#)

[Save](#)

Logging

- CloudWatch Logs subscriptions

```
logger: # handler for logs generated by monitor functions + greengrass runtime
handler: handler.logger_entry
memorySize: 128
timeout: 3
events:
  # these will need to be commented out at first deploy
  # there need to be logs that already exist for these subscriptions to be created
  - cloudwatchLog: "/aws/greengrass/GreengrassSystem/runtime"
  - cloudwatchLog: "/aws/greengrass/GreengrassSystem/GGConnManager"
  - cloudwatchLog: "/aws/greengrass/Lambda/#{AWS::Region}/#{AWS::AccountId}/${{self:custom.stackName}}-monitor"
  - cloudwatchLog: "/aws/greengrass/Lambda/#{AWS::Region}/#{AWS::AccountId}/${{self:custom.stackName}}-tunnel"
  - cloudwatchLog: "/aws/greengrass/Lambda/#{AWS::Region}/#{AWS::AccountId}/${{self:custom.stackName}}-reconfigure"
  - cloudwatchLog: "/aws/greengrass/Lambda/#{AWS::Region}/#{AWS::AccountId}/${{self:custom.stackName}}-system"
```



metromon APP 19:12

[INFO]-ipc_client.py:202,Got work item with invocation id [e0a63a44-b59f-42d2-7b9d-ed6ab5e560cb]
[INFO]-handler_system.py:21,Rebooting! Hope it comes back up!

[INFO]-lambda_runtime.py:364,Caught signal 15. Stopping runtime.

[INFO]-ipc_client.py:223,Posting work result for invocation id [e0a63a44-b59f-42d2-7b9d-ed6ab5e560cb] to
http://localhost:8000/2016-11-01/functions/arn:aws:lambda:us-east-1:79444 ·prod-system:58/work
[INFO]-ipc_client.py:232,Posted work result for invocation id [e0a63a44-b59f-42d2-7b9d-ed6ab5e560cb]
[INFO]-ipc_client.py:192,Getting work for function [arn:aws:lambda:us-east-1:79444 ·prod-system:58] from
http://localhost:8000/2016-11-01/functions/arn:aws:lambda:us-east-1:794441 ·prod-system:58/work
[INFO]-lambda_runtime.py:364,Caught signal 15. Stopping runtime.

[INFO]-server.py:72,IPC connection closed code = 1006 (connection closed abnormally [internal]), no reason
[INFO]-Stopping...
[INFO]-Stopping...
[INFO]-Stopping...
[INFO]-Stopping...
[INFO]-Stopping...
[WARN]-monitor.py:216,Collector thread finished

Client connected to MQTT gateway.

Client: aadd

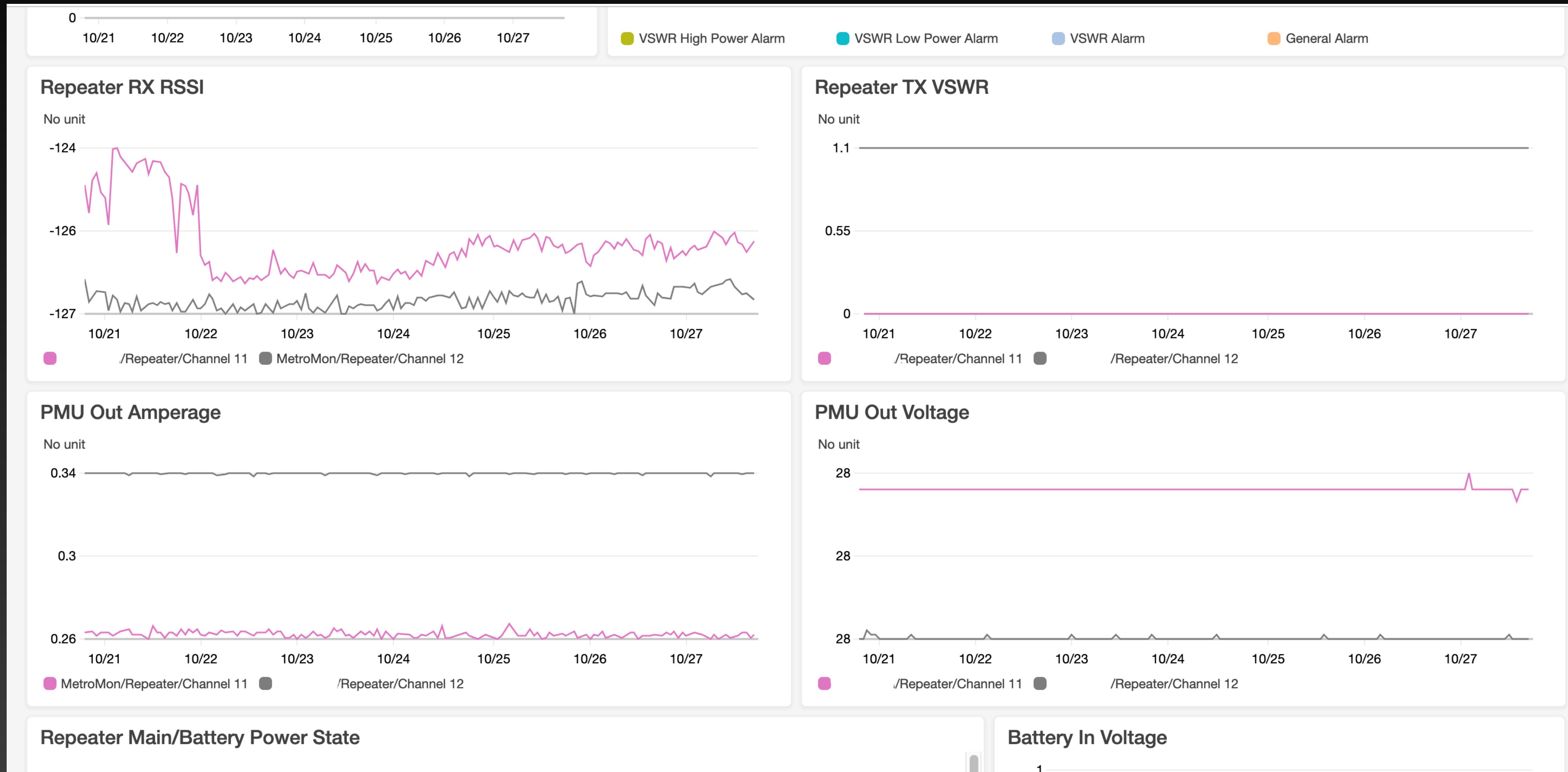
IP: 100.2.

Timestamp: 2020-10-26 17:12:45.780000

Logging To Slack

New

CloudWatch Metrics



Secure Tunnel

```
from os import path, getenv
from subprocess import Popen
import logging

log = logging.getLogger(__name__)

def start_tunnel_entry(event, context):
    """Entry point for initiating secure tunnels.

https://docs.aws.amazon.com/iot/latest/developerguide/secure-tunneling.html
    """
    token = event["clientAccessToken"]
    services = event["services"]
    region = event["region"]

    thing_name = getenv("AWS_IOT_THING_NAME", "UNKNOWN")

    log.info(f"Starting tunnel for thing {thing_name} {services} in {region}")

    # start up localproxy in background
    env = dict(AWSIOT_TUNNEL_ACCESS_TOKEN=token)
    args = [get_exe_path(), "-d", "localhost:22", "-r", region]
    Popen(args, env=env)

def get_exe_path() -> str:
    """Get absolute path to localproxy executable."""
    filepath = path.abspath(__file__)
    return path.join(path.dirname(filepath), "tunnel", "localproxy")
```

Subscriptions

CREATE A SUBSCRIPTION

Confirm and save your Subscription

Your Subscription is complete and your objects are connected in this Group. You can now save, and then deploy your new Group definition to have this change take effect.



IoT Cloud

SERVICE

hello/world/counter/trigger



Greengrass_HelloWorld_Counter

LAMBDA

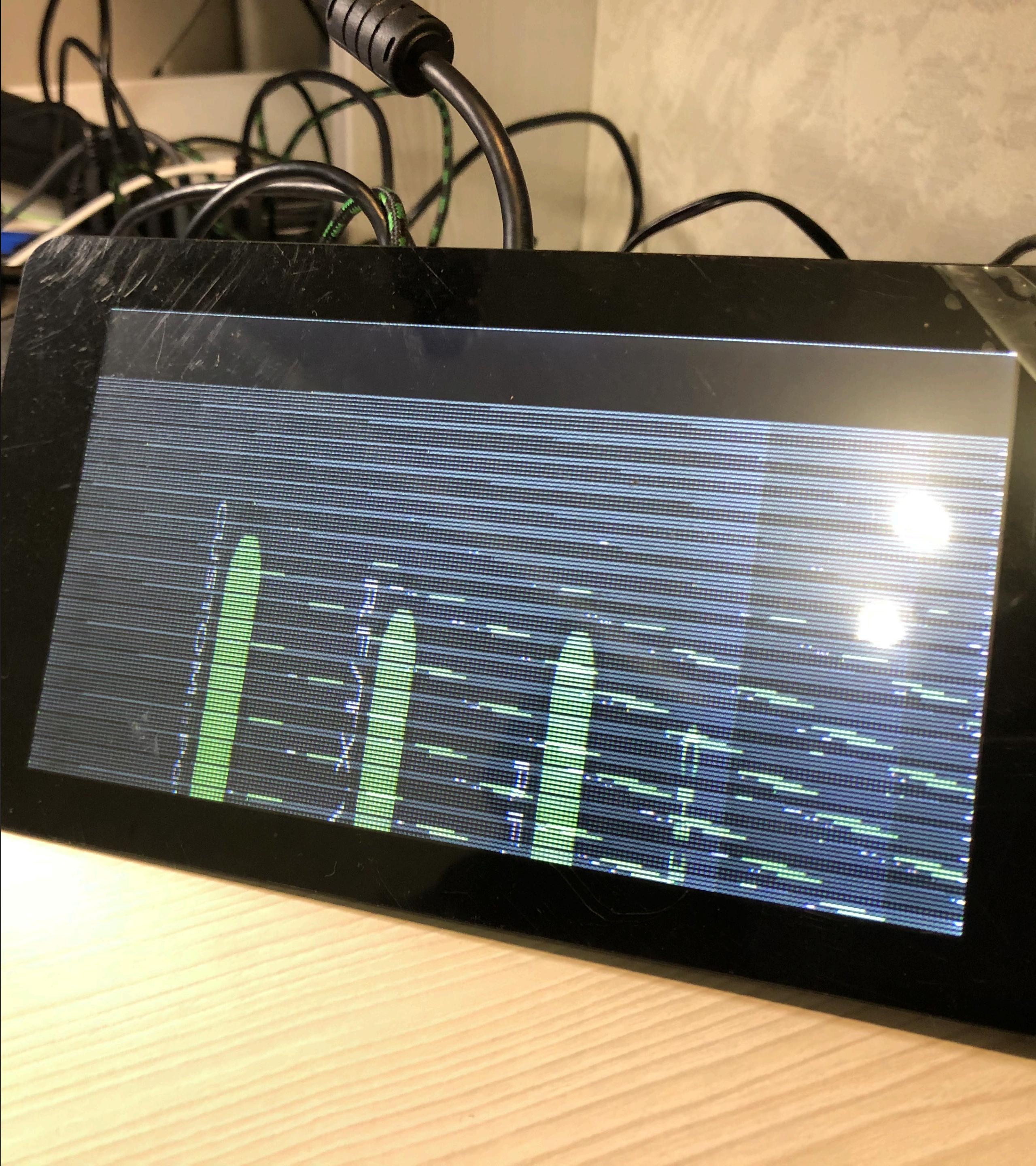
Back

Finish



Xorg Lambdas With GUIs?

- Sort of!



Xorg

Lambdas With GUIs?

- Sort of!
- GUI wrapper
 - Import lambda code on DM start
- Long-lived DM restart lambda

```
def restart_display_manager():
    restart_gui = ["/etc/init.d/lxdm", "restart"]
    subprocess.run(restart_gui)
```



```
from pathlib import Path

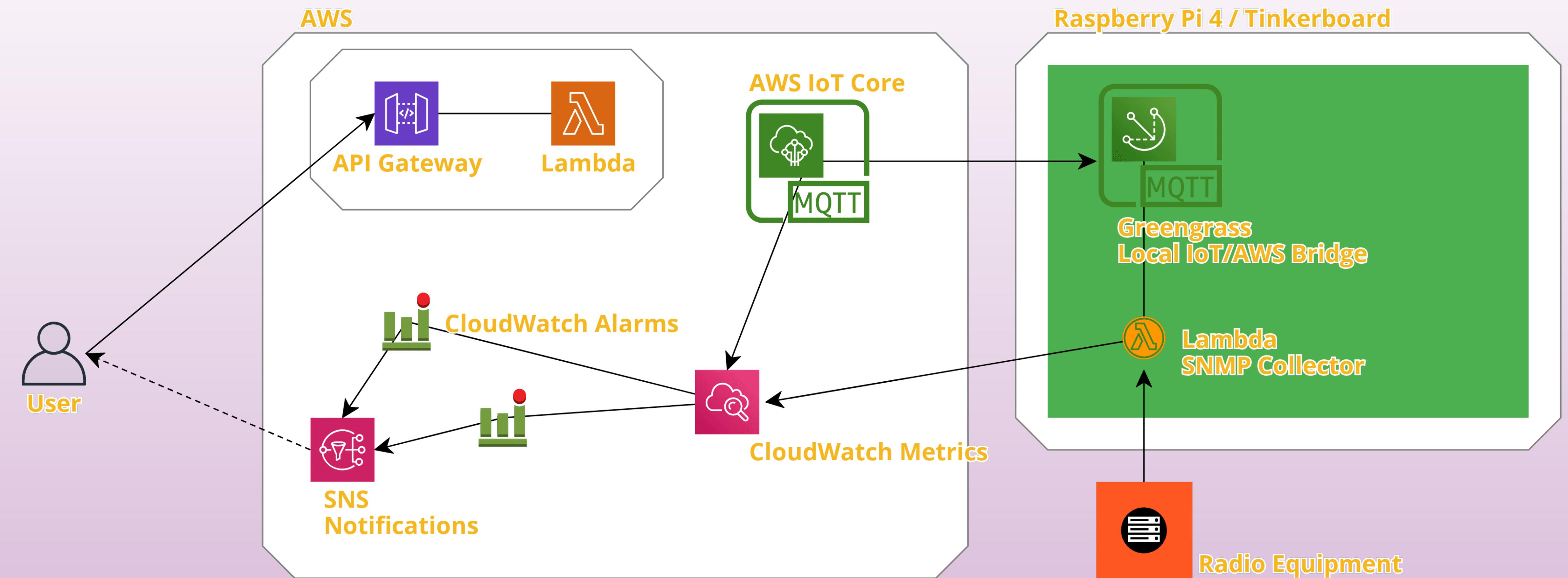
def find_lambda_system_module() -> Optional[str]:
    """Return import path to GUI code, if it exists."""
    # this is where our system code would be deployed to
    files = Path("/greengrass/ggc/deployment/lambda").glob(
        "arn.aws.lambda.*.*.function.myapp-*--system.*")
    matches = list(files)
    log.info(f"Found deployments: {matches}")
    if not matches:
        log.info("Didn't find system function deployed")
        return None

    if len(matches) > 1:
        log.info("Warning: found multiple deployments of system code")

    return str(matches.pop().absolute())
```



```
def display_lambda_gui(system_module_path: str):  
    sys.path.append(system_module_path)  
    try:  
        from device_gui.main import main as lambda_gui_main  
  
        os.environ["MYAPP_DISPLAY_WWRAPPED"] = "true"  
        lambda_gui_main()  
    except Exception as ex:  
        log.exception(ex)  
        display_default_gui(traceback.format_exc())
```





COMMUNITY DAY

Thanks!

Mischa Spiegelmock
Founder, JetBridge Inc.
me@mish.dev

