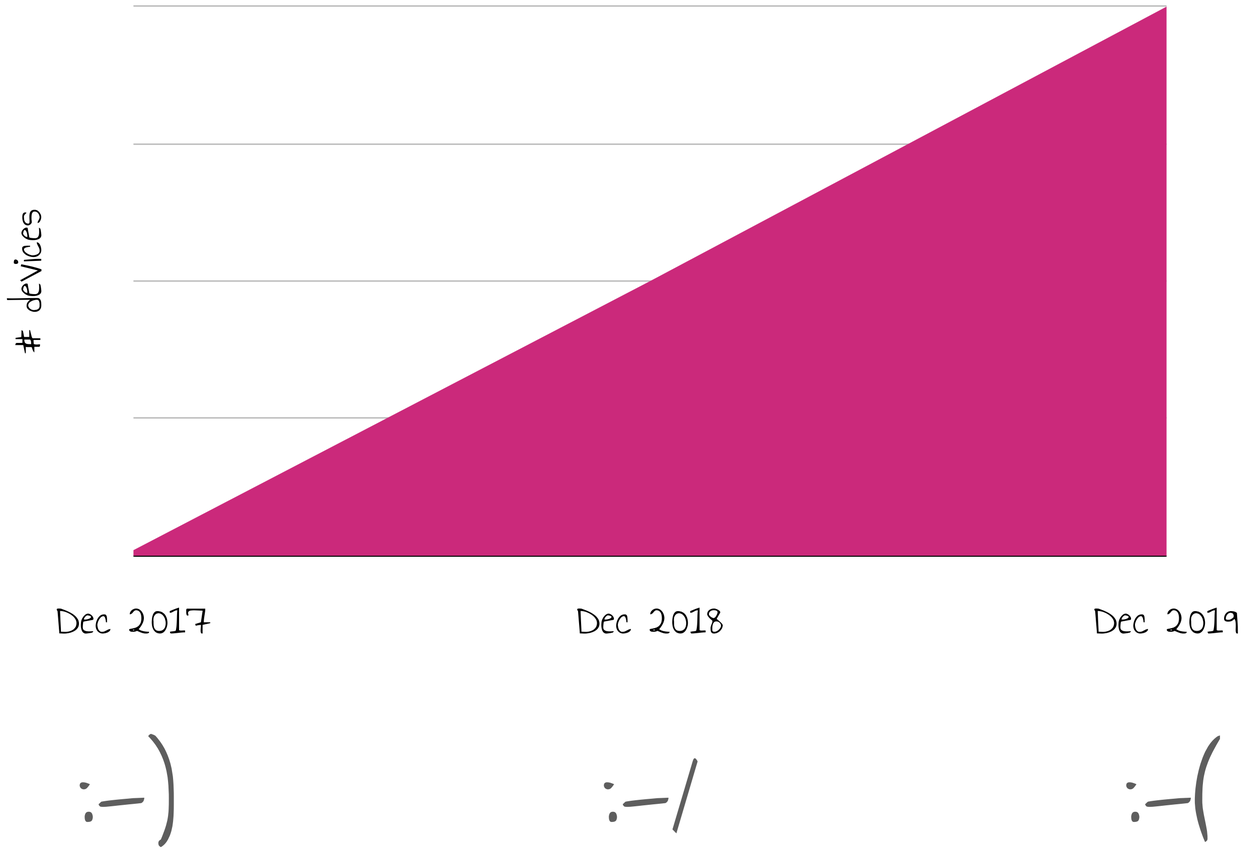
Embedded GNU/Linux Developer Meetup
Upparat

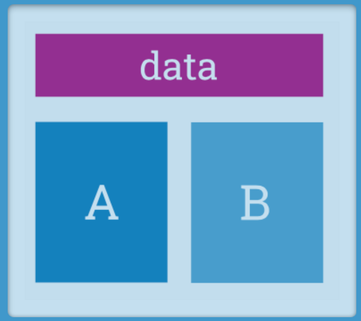
About me &
my work at CARU

Number of devices vs developer happiness to update devices



Challenges?

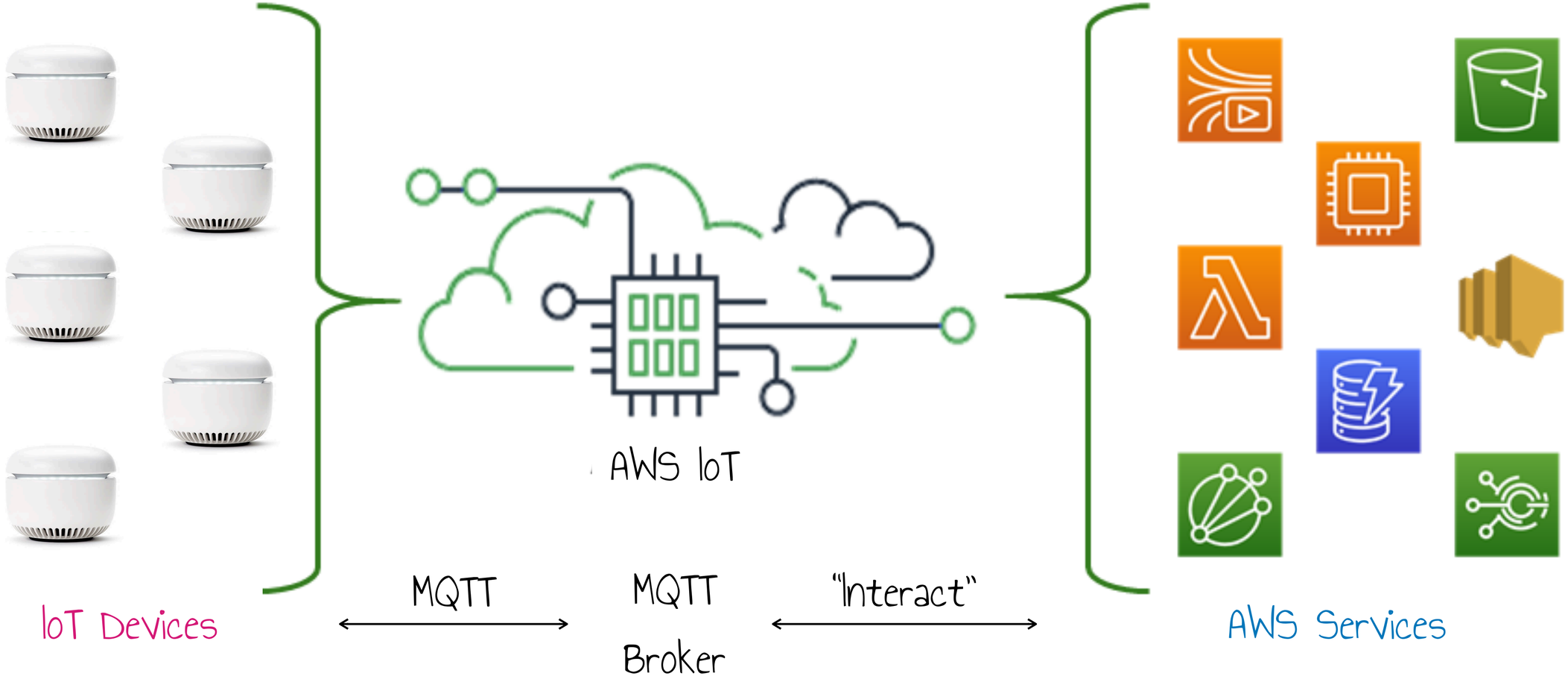
- Device Sided Update Solutions
  - SWUpdate
  - RAUC
  - ...



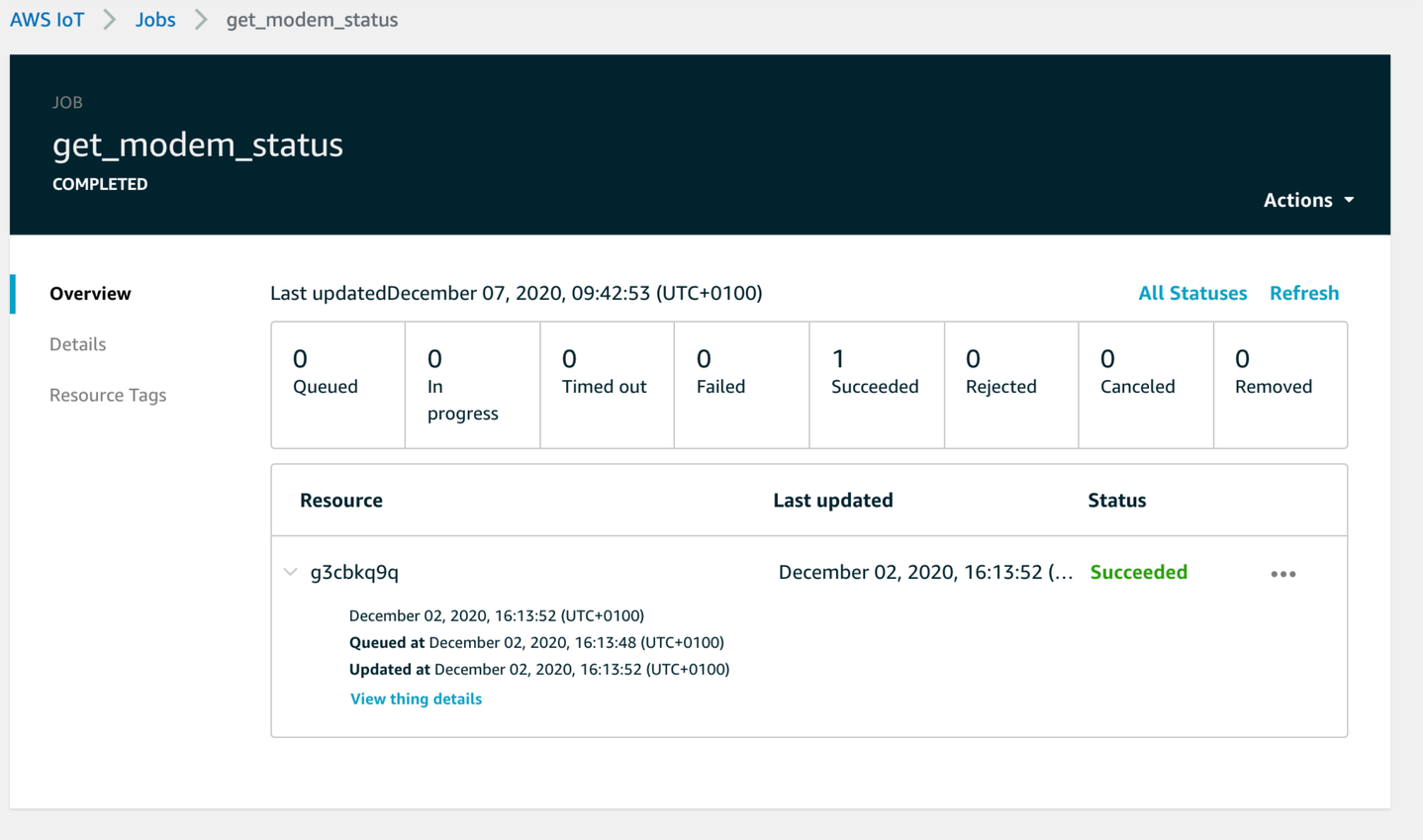
- "Platform Solutions"
  - Mender
  - Balena
  - Hawkbit
  - (Bosch IoT Rollouts)
  - ...

But what if you already use AWS IoT anyhow?

"AWS IoT is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices"



AWS IoT Jobs?


AWS IoT > Jobs > get_modem_status
JOB
get_modem_status
COMPLETED
Actions
Overview
Details
Resource Tags
Last updatedDecember 07, 2020, 09:42:53 (UTC+0100)
All Statuses
Refresh
0 Queued
0 In progress
0 Timed out
0 Failed
1 Succeeded
0 Rejected
0 Canceled
0 Removed
Resource
Last updated
Status
g3cbkq9q
December 02, 2020, 16:13:52 (...
Succeeded
December 02, 2020, 16:13:52 (UTC+0100)
Queued at December 02, 2020, 16:13:48 (UTC+0100)
Updated at December 02, 2020, 16:13:52 (UTC+0100)
View thing details

"AWS IoT jobs can be used to define a set of remote operations that are sent to and executed on one or more devices connected to AWS IoT."

"One of us"

Create Job

"bling"

AWS IoT

S3

Job Document

```
{
  "cmd": "do-bling"
}
```

$aws/things/hal9000/jobs/notify {QUEUED & IN_PROGRESS jobs}

$aws/things/hal9000/jobs/bling/get

$aws/things/hal9000/jobs/bling/get/accepted {JobDocument}

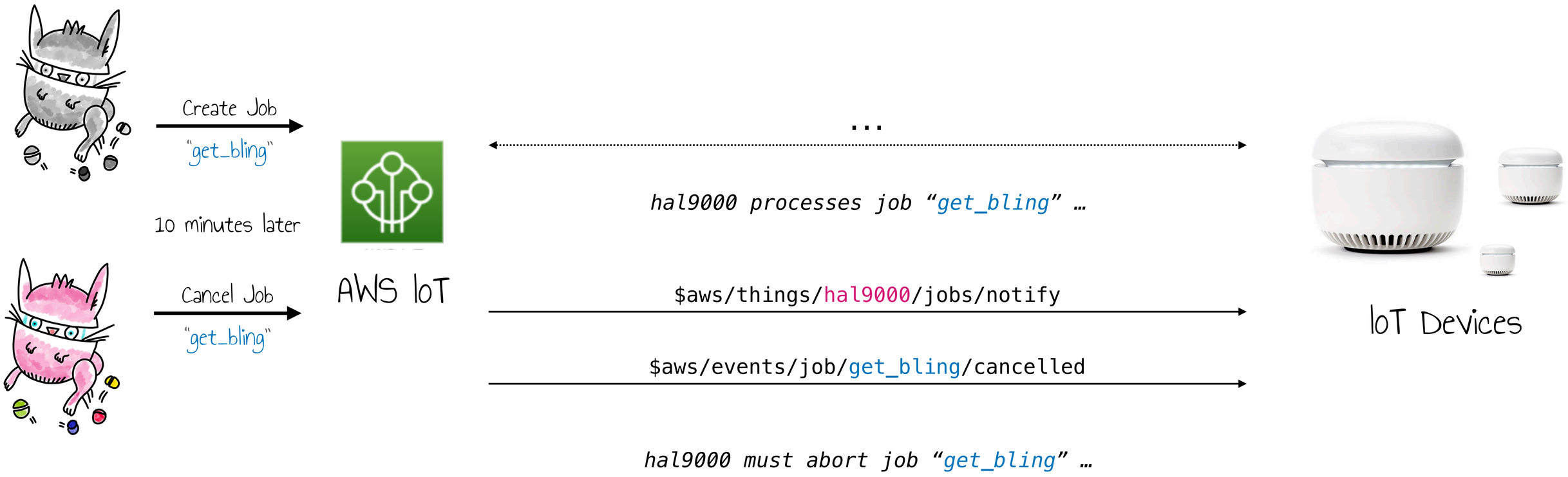
hal9000 processes job "bling" …

$aws/things/hal9000/jobs/bling/update {"status":"SUCCEEDED"}

HAL9000

IoT Devices

AWS IoT Jobs can be cancelled / deleted / timed-out



AWS IoT Jobs can have pre-signed URLs

Create Job

"get_bling"

AWS IoT

...

HAL9000

IoT Devices

hal9000 processes job "get_bling" …

HTTP GET {jobDocument.url}

S3

fancy_bling.zip

```
{
  "cmd": "get-new-bling",
  "url": "${aws:iot:s3-presigned-url:https://xxx.s3.amazonaws.com/fancy_bling.zip}"
}
```

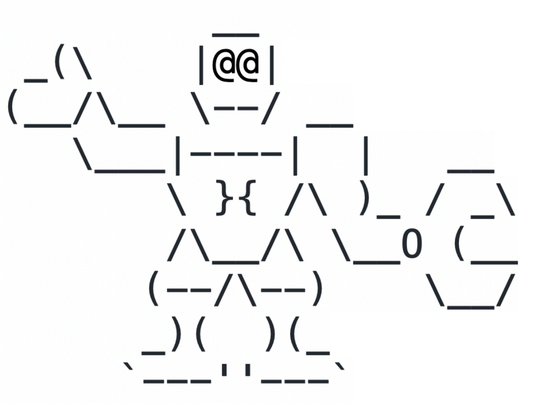
Upparat?!

AWS IoT Jobs + $DEVICE_UPDATE_PROCESS

= "Update Apparat" aka Upparat

AWS IoT

interacts



interacts

RAUC

$DEVICE_UPDATE_PROCESS

runs on

Upparat ...

- runs on the device
- takes care of AWS IoT job handling
- takes care of reliably downloading the firmware
- provides "hooks" to adapt to your use case
- is written in Python :)

Upparat: Hooks

```
# mandatory
version = /etc/upparat/hooks/version.sh
install = /etc/upparat/hooks/install.sh

# optional
download = /etc/upparat/hooks/download.sh
restart = /etc/upparat/hooks/restart.sh
ready = /etc/upparat/hooks/ready.py
```

Upparat: Version Hook

```bash
#!/usr/bin/env bash
# Gets the system version

# $1: time elapsed since first call
# $2: retry count
# $3: meta from job document

# I.e. 1.6.2-RELEASE
cat /etc/caru-version
```

if not matching:
install update

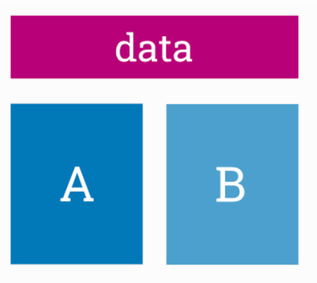
Upparat Job Document

```json
{
  "action": "update",
  "version": "1.6.4-RELEASE",
  "file": “${aws:iot:s3-presigned-url:…}”
}
```

Upparat: Download Hook (Optional)

```
#!/usr/bin/env bash
# Start download if critical section lock isn't present
#
# $1: timestamp from first call
# $2: retry count
# $3: meta from job document

if test -f /tmp/cortex.critical.lock; then
    exit 3 # retry later
fi
```

Upparat: Install Hook



Example Setup
RAUC w/ 2 partitions

```bash
#!/usr/bin/env bash

# Installs the downloaded rauc bundle
#
# $1: time elapsed since first call
# $2: retry count
# $3: meta from job document
# $4: file location

if test -f /tmp/cortex.critical.lock; then
    # Cortex is running a critical section -> retry later
    exit 3
else
   rauc install $4
fi
```

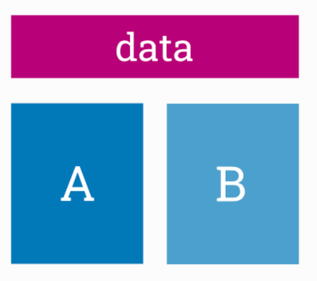
Upparat: Restart Hook (Optional)

```bash
#!/usr/bin/env bash

# Restarts the system if critical section lock isn't present
# $1: time elapsed since first call
# $2: retry count
# $3: meta from job document
# $4: force job ("True" or "False")

# swallow SIGTERM which we will receive after emiting reboot because
# we don't want to fail this hook because it could lead to a failed job.
trap "" SIGTERM

if [ ! -f /tmp/cortex.critical.lock ] || [ "$4" == "True" ]; then
     sudo reboot
     sleep 30
     # something is very wrong if we are still here.
     exit 1
else
     exit 3
fi
```

Upparat: Ready Hook (Optional)

```python
#!/usr/bin/env python3
import json
import subprocess
import sys

MAX_RETRY_ATTEMPTS = 3

def main(args):
    rauc_status = subprocess.run(
        ["rauc", "status", "--output-format=json"],
        stdout=subprocess.PIPE,
        universal_newlines=True,
    )

    rauc_status_details = json.loads(rauc_status.stdout)
    booted_slot = rauc_status_details["booted"]

    # Get remaining attempts
    barebox_state = subprocess.run(
        ["sudo", "barebox-state", “-d"],
        stdout=subprocess.PIPE, universal_newlines=True
    )

    remaining_attempts_key = "bootstate.{}.remaining_attempts".format(booted_slot)
    remaining_attempts = 0
    for line in barebox_state.stdout.splitlines():
        key, value = line.split("=")
        if key == remaining_attempts_key:
            remaining_attempts = int(value)
            break

    # Signal retry if slot not (yet) marked as good
    if remaining_attempts != MAX_RETRY_ATTEMPTS:
        sys.exit(3)

if __name__ == "__main__":
    main(sys.argv[1:])
```

```
# waits for this:
rauc status mark-good booted
```



Upparat: Retries hooks on exit code 3

```
retry_interval = 60
max_retries = 60


# if a hook returns exit code 3
# Upparat will retry it every 60s
# for up to 60 times.
```

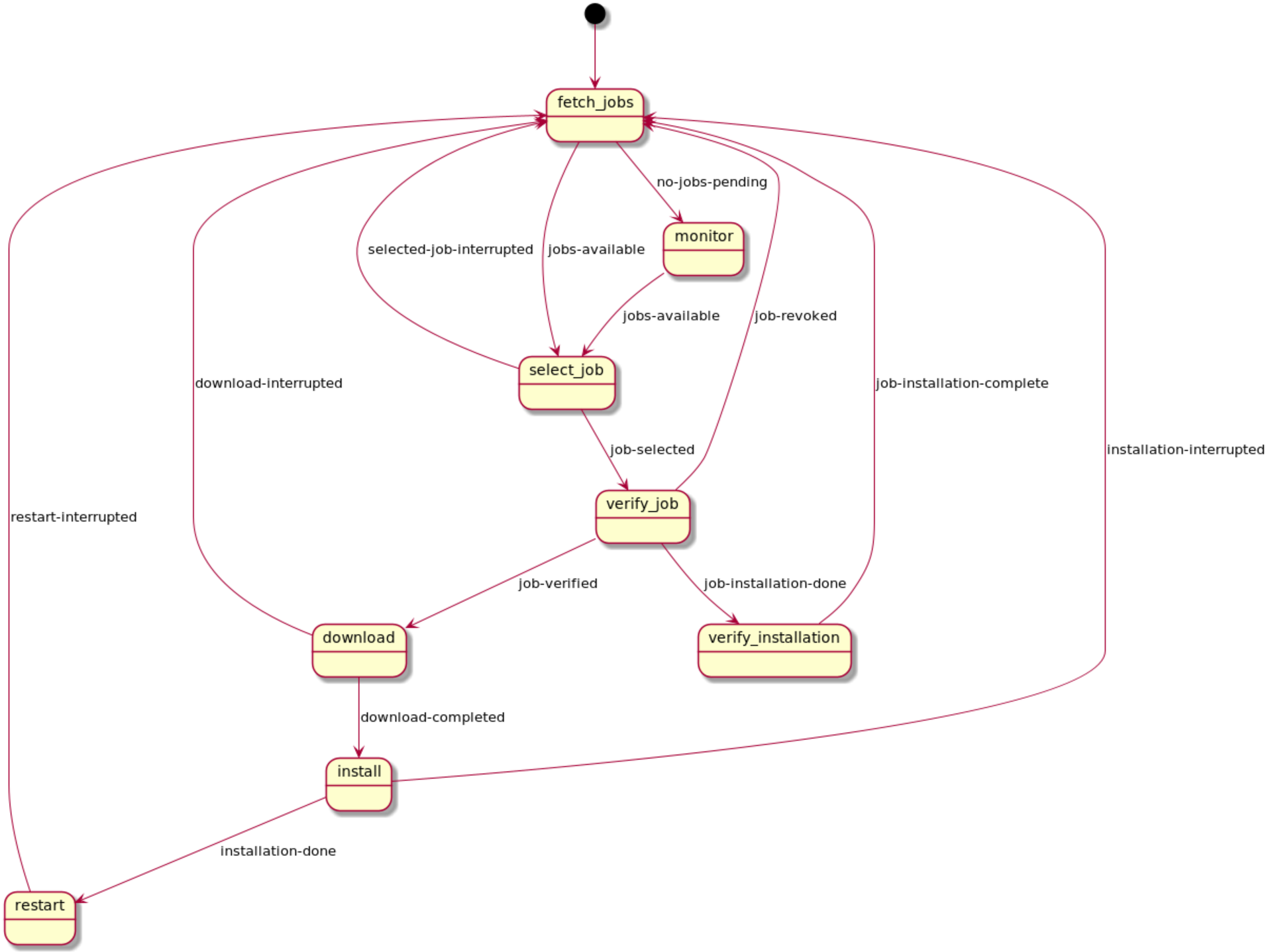
Upparat: Download Handling

"Robust" ;-)

- Resumable (Chunked, HTTP Range Headers)
- Exponential backoff with jitter on errors
- Handles expired pre-signed URLs

Upparat: Minimal Configuration

```
[broker]
host = akzywbhaxlhqa-ats.iot.eu-central-1.amazonaws.com
port = 443

thing_name = hal9000

certfile = /etc/upparat/certfile
keyfile = /etc/upparat/keyfile
cafile = /etc/upparat/cafile

[hooks]
version = /etc/upparat/hooks/version.sh
install = /etc/upparat/hooks/install.sh
```


fetch_jobs
no-jobs-pending
monitor
selected-job-interrupted
jobs-available
jobs-available
job-revoked
select_job
download-interrupted
job-installation-complete
job-selected
installation-interrupted
verify_job
restart-interrupted
job-verified
job-installation-done
download
verify_installation
download-completed
install
installation-done
restart

Demo

```
                 __     /
  _(\        |@@| /
 (__/\__   \--/ __
    \___|----|  |   __
        \ }{ /\ )_ / _\
        /\__/\ \__O (__
       (--/\--)    \__/
       _)(  )(_
      `---''---`
```

Getting Started

```
pip install upparat
```

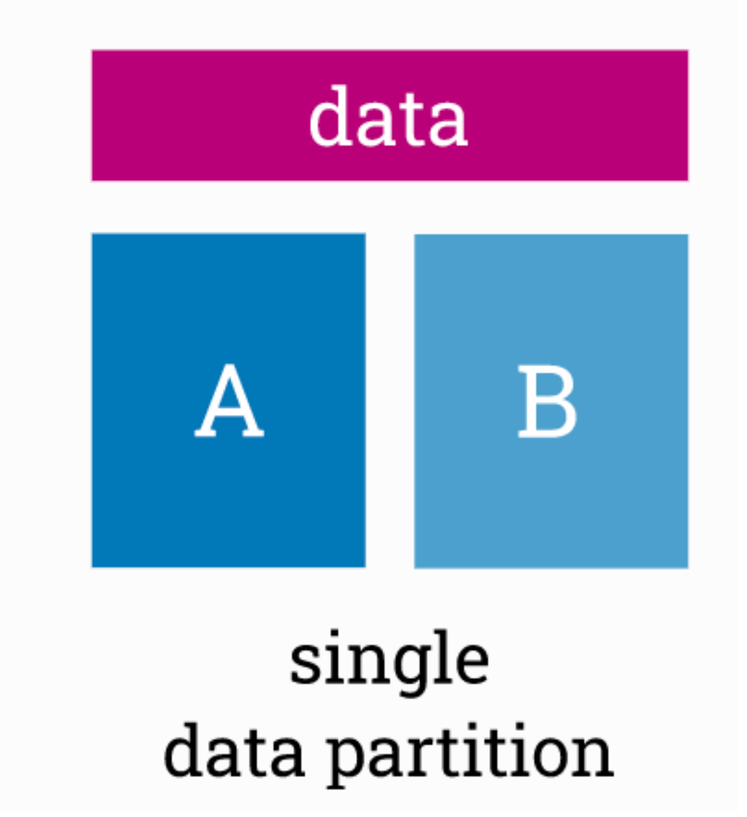
github.com/caruhome/upparat

```
                    Q&A!
                   /
          __      /
 _(\     |@@|    /
(__/\__   \--/   __
   \___|----|   |   __
       \ }{ /\ )_ / _\
       /\__/\ \__O (__
      (--/\--)    \__/
      _)(  )(_
     `---''---`
```

Few questions from my side:

- How does your update process look like?
- What do you like or dislike about it?

- Thanks for coming! :-)


data
A
B
single
data partition

```
            __
 _(\       |@@|
(__/\__  \--/ __
   \___|----|   |   __
       \ }{ /\ )_ / _\
       /\__/\ \__O (__
      (--/\--)    \__/
      _)(  )(_
     `---''---`
```