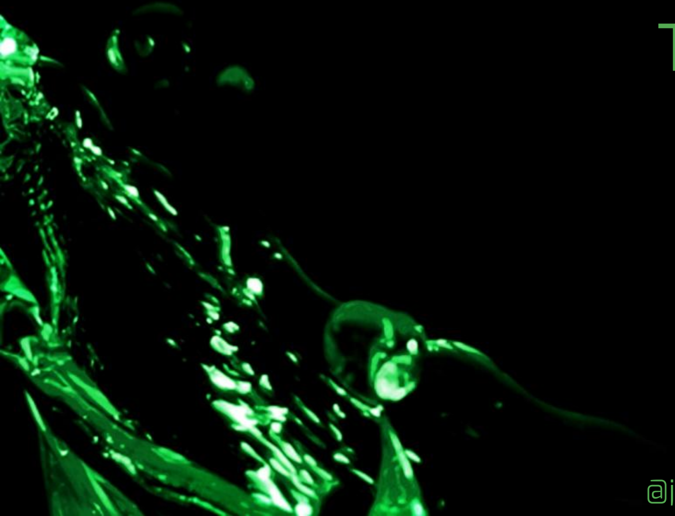
Update Strategies for the Edge

There's a better way.



JFrog

@jfrog | Copyright © 2019 JFrog. All Rights Reserved

Kat Cosgrove

IoT Engineer

Developer Advocate

Twitter: @Dixie3Flatline

Email: katc@jfrog.com

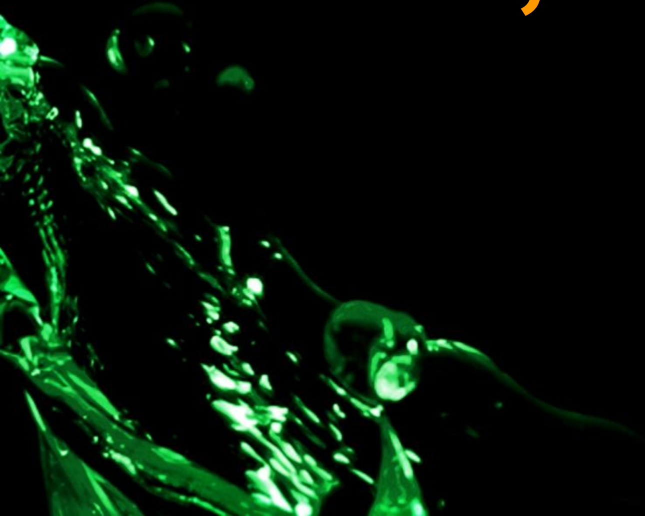
jfrog.com/shownotes

Copyright © 2019 JFrog. All Rights Reserved

How large is the Edge?

20,400,000,000

That's a lot of devices.



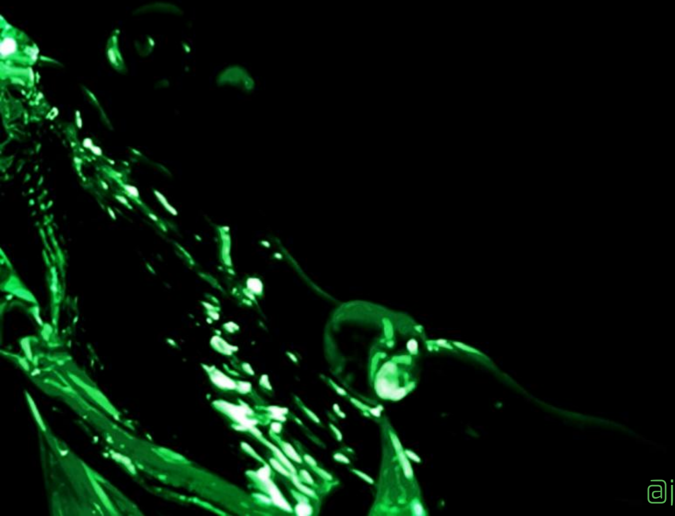
Updates Today

They don't update; device is effectively single-use

OR

It's time-consuming, complicated, or requires physical access

Why change?



It's beyond inconvenient

Edge computing is massive and growing

- Consumer
- Industrial
- Medical

Slow OTA updates are annoying

Wired updates are expensive and more annoying

Copyright © 2019 JFrog. All Rights Reserved

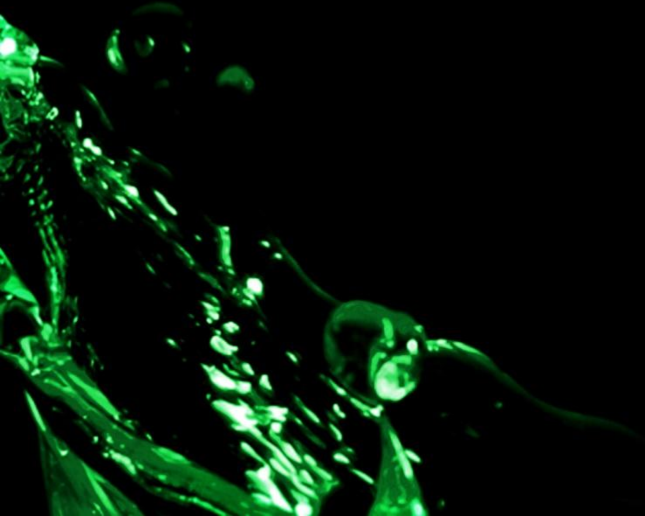
It's dangerous

Unpatched bugs can be a huge vulnerability

- Expose private data
- Harnessed for a botnet
- Used for cryptocurrency mining
- Safety implications for medical

Copyright © 2019 JFrog. All Rights Reserved

What’s slowing us down?



Not building for it.

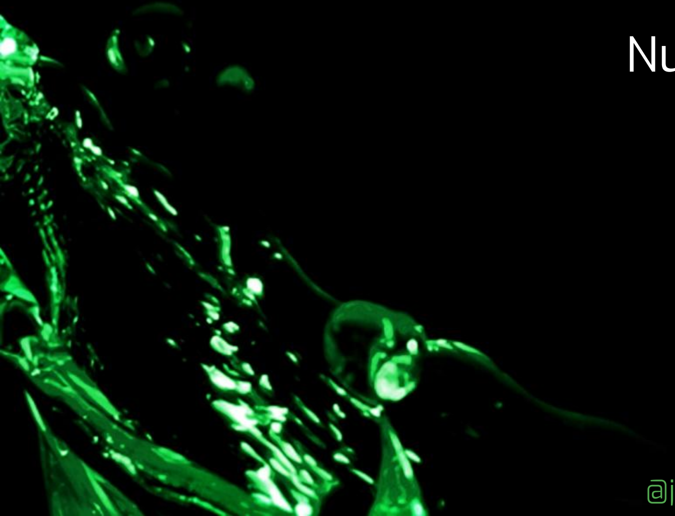
Many devices are not made to be updated.

- Designed to run one version until the end
- “Update strategy” here is flashing the device
- Bugs are inevitable

Copyright © 2019 JFrog. All Rights Reserved

Between 1 and 25

Number of bugs per 1000 LOC



Connectivity Concerns

We can’t rely on the device’s network

- Networks may be unstable
- Bandwidth may be low
- Network probably isn’t secure

Copyright © 2019 JFrog. All Rights Reserved

Hardware Variations

- It’s 20.4 billion devices
- Lots of specialized hardware
- Variations in memory, storage space, architecture

How do we design something that handles so much variety?

Think future-forward.

Updates are your friend. Embrace updates, not security nightmares.

@jfrog | Copyright © 2019 JFrog. All Rights Reserved

Get better with age.

Your product should not be getting worse from the moment it ships.

Copyright © 2019 JFrog. All Rights Reserved

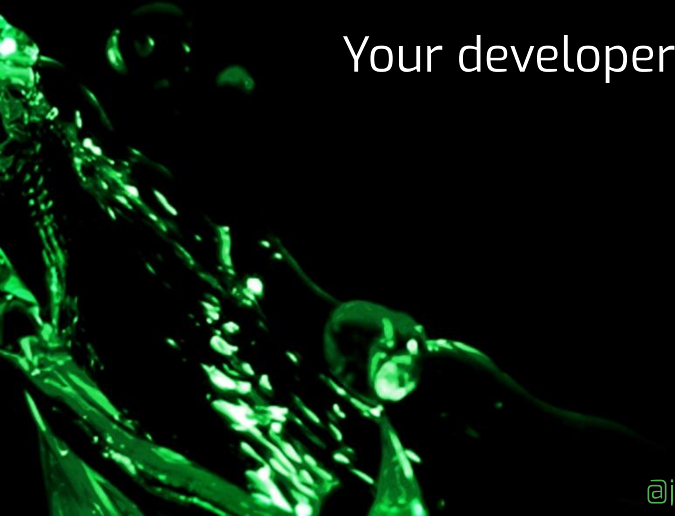
Build robust.

Brittle software means a brittle device, and that doesn’t inspire trust.

@jfrog | Copyright © 2019 JFrog. All Rights Reserved

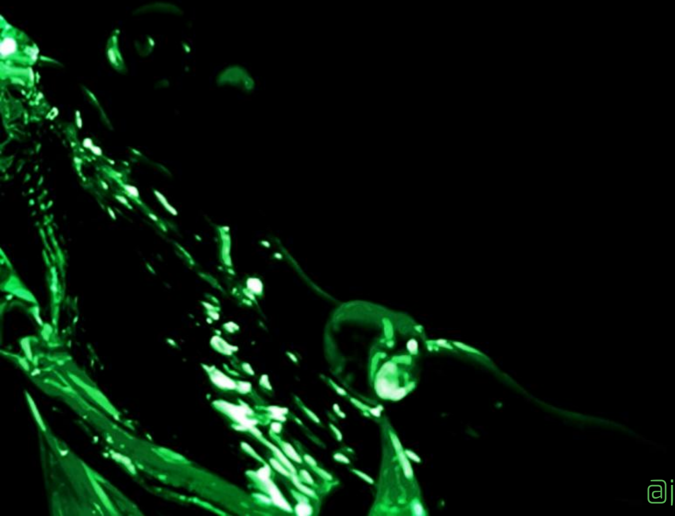
Modern DevOps tools.

Your developers will thank you and things will run more smoothly.



@jfrog | Copyright © 2019 JFrog. All Rights Reserved

The Proof of Concept



@jfrog | Copyright © 2019 JFrog. All Rights Reserved

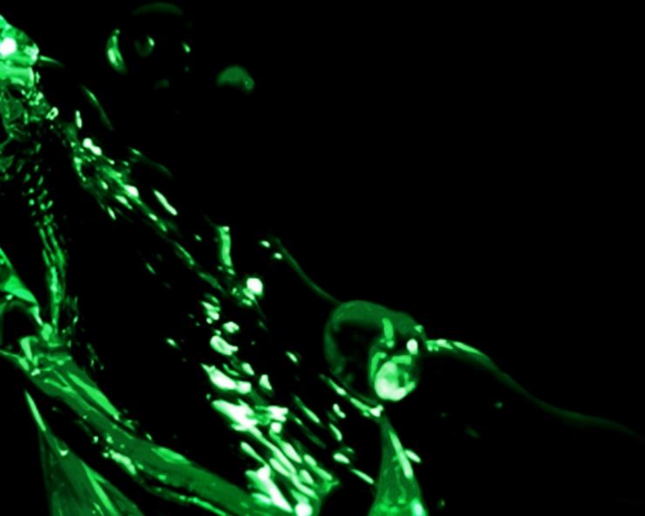
Cars Now

- Majority not designed for OTA updates
- OTA updates are still slow and inconvenient
- Little standardization
- Significant portion of recalls are due to software

Cars as Edge Devices

- Presented a range of solvable pain points in one device
- Tangible example for end users and manufacturers
- Device in question meant speed, reliability, and safety were equally important

Workflows and Tools



Two Distinct Workflows

Software Updates

- Without flashing firmware
- No interruption of user
- Takes only seconds
- Relies on K3S and Helm

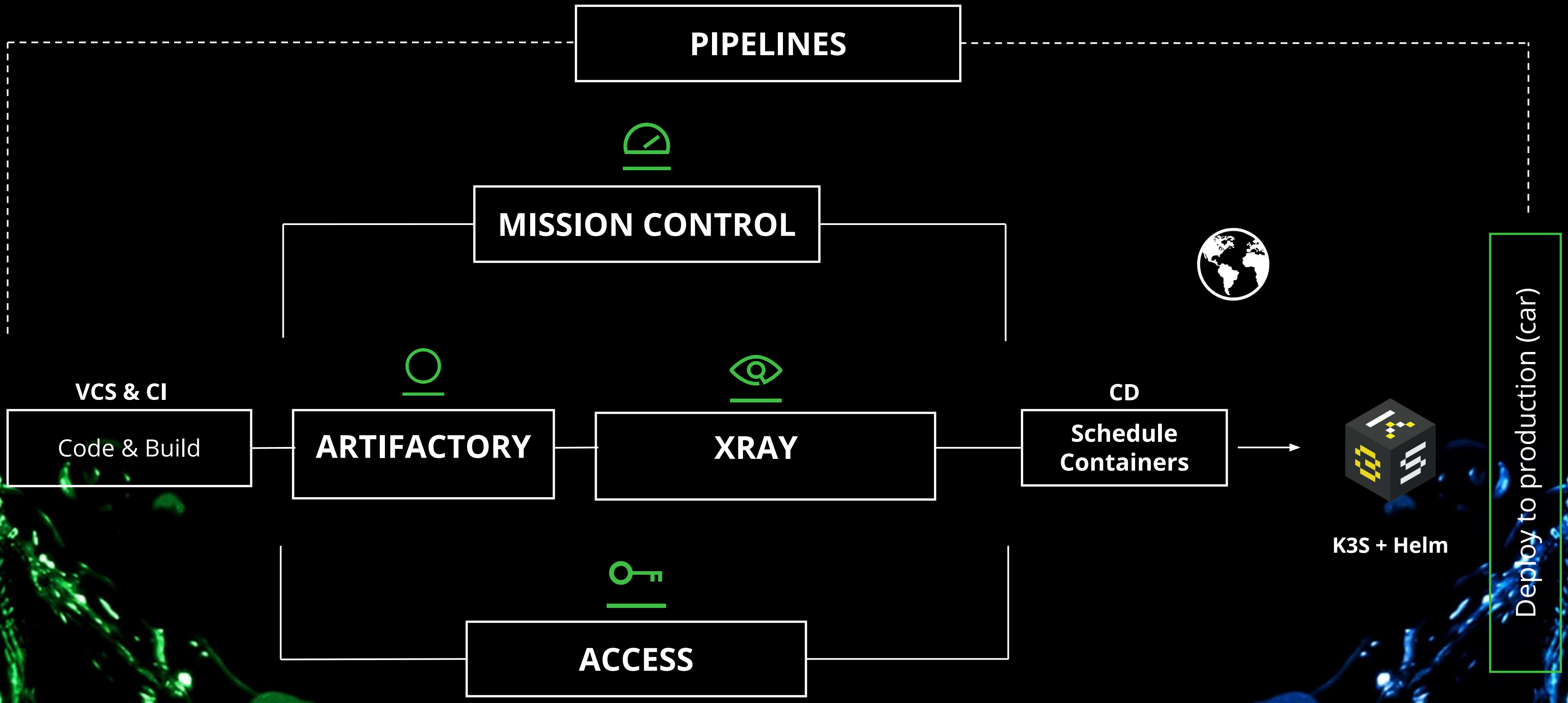
Firmware Updates

- More difficult update
- Takes only minutes
- Rollback if there is a failure
- Relies on Mender, Yocto, and Artifactory

Copyright © 2019 JFrog. All Rights Reserved

Software Workflow

Copyright © 2019 JFrog. All Rights Reserved



PIPELINES

MISSION CONTROL

VCS & CI

Code & Build

ARTIFACTORY

XRAY

CD

Schedule Containers

K3S + Helm

Deploy to production (car)

ACCESS

@jfrog | Copyright © 2019 JFrog. All Rights Reserved

JFrog ARTIFACTORY

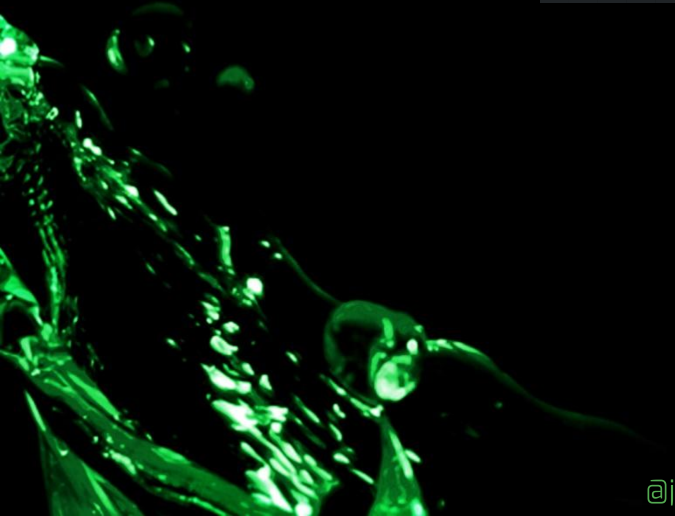
JFrog XRAY

JFrog Xray

- Vulnerability scanning tool
- All major package types supported
- Continuously scans your artifacts
- Risk Based Security’s VulnDB

K3S

Kubernetes, but 5 less



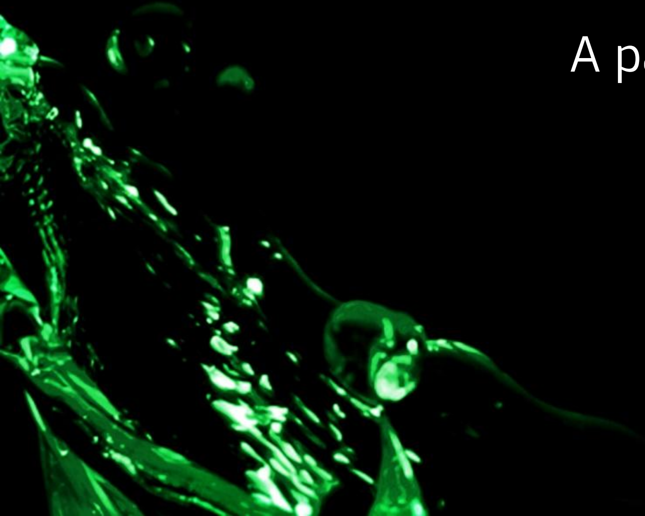
K3S

- Lightweight Kubernetes, designed for Edge devices
- Uses only 512mb of RAM
- 40mb binary
- Very minimal OS requirements

Copyright © 2019 JFrog. All Rights Reserved

HELM

A package manager for Kubernetes



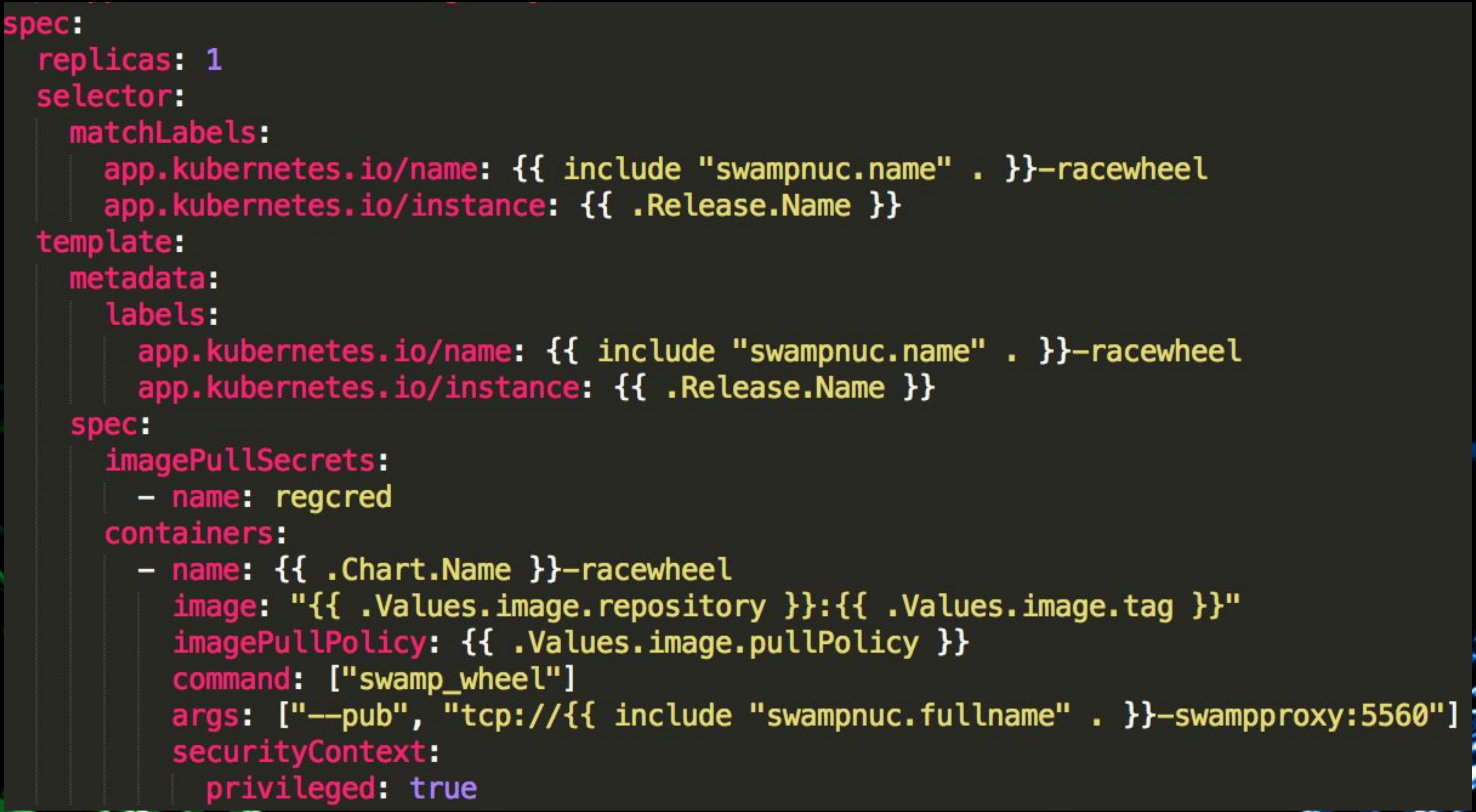
Helm

“Charts” describe complex applications

- Easily repeatable installation
- Final authority on application
- Easy to version
- Supports rollbacks

Copyright © 2019 JFrog. All Rights Reserved

Helm Charts



```
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: {{ include "swampnuc.name" . }}-racewheel
      app.kubernetes.io/instance: {{ .Release.Name }}
  template:
    metadata:
      labels:
        app.kubernetes.io/name: {{ include "swampnuc.name" . }}-racewheel
        app.kubernetes.io/instance: {{ .Release.Name }}
    spec:
      imagePullSecrets:
        - name: regcred
      containers:
        - name: {{ .Chart.Name }}-racewheel
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          command: ["swamp_wheel"]
          args: ["--pub", "tcp://{{ include "swampnuc.fullname" . }}-swampproxy:5560"]
          securityContext:
            privileged: true
```

@jfrog | Copyright © 2019 JFrog. All Rights Reserved

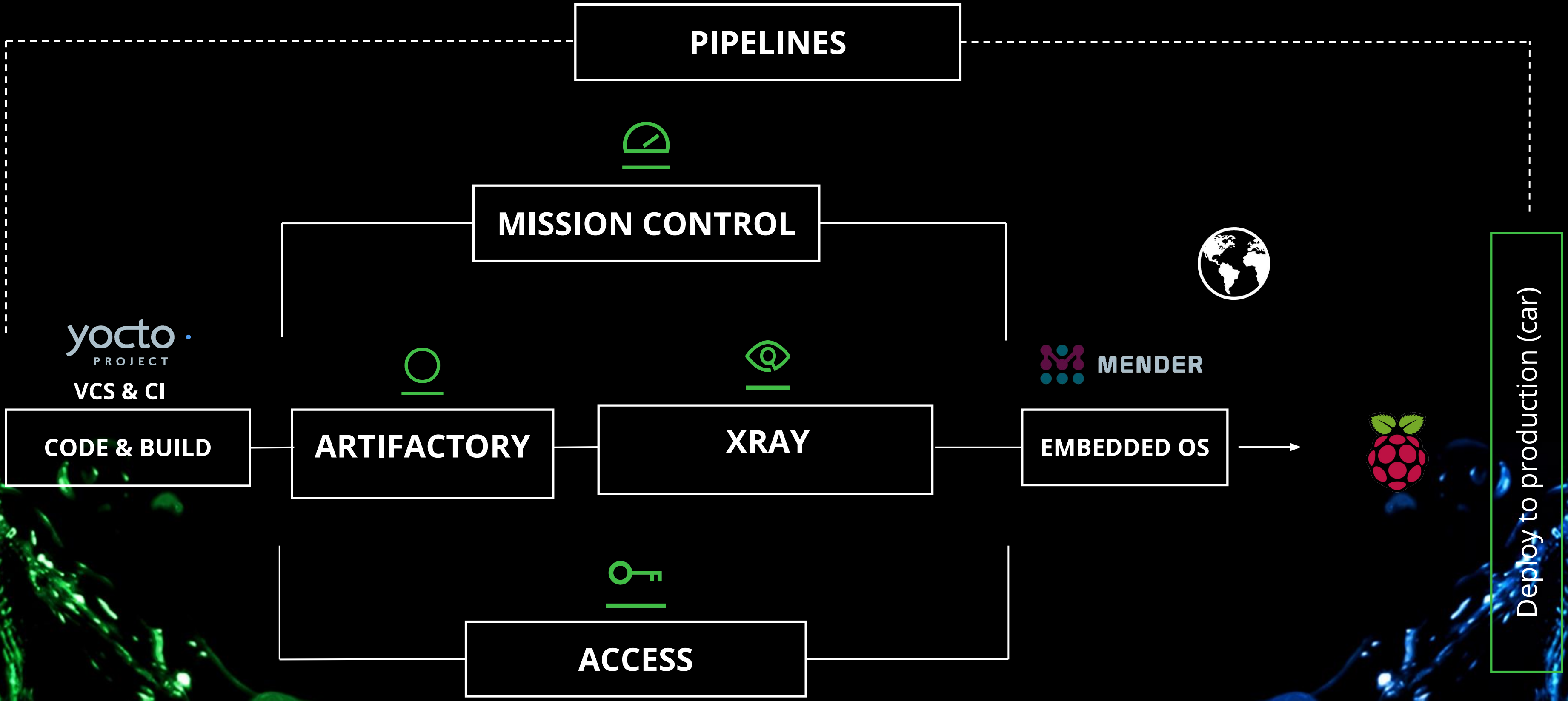
The Result - Software

Application updates are quick and efficient

- Average of 35 seconds from dev to car
- No interruption for the user
- Can happen while device is in use
- Could happen silently, depends on device purpose

Copyright © 2019 JFrog. All Rights Reserved

Firmware Workflow



@jfrog | Copyright © 2019 JFrog. All Rights Reserved

MENDER

OTA updates for embedded Linux devices

Copyright © 2019 JFrog. All Rights Reserved

Mender Overview

Ticks several of the boxes we're looking for:

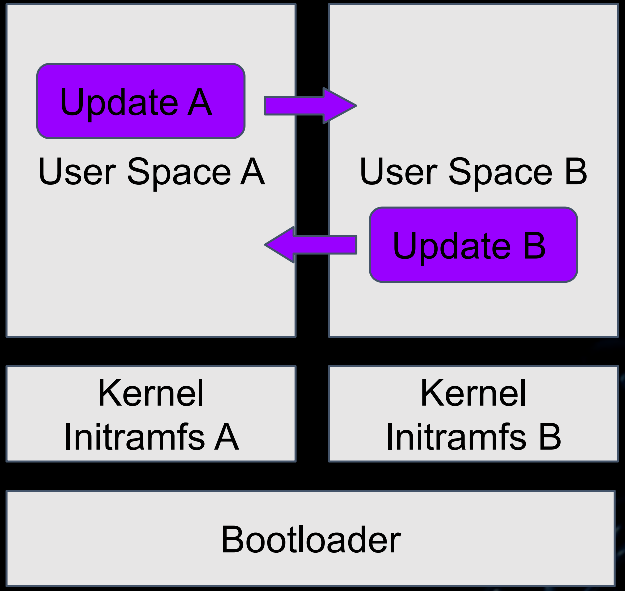
- Updates are signed and verified
- Supports automatic rollbacks
- Several distinct installation strategies
- Dual A/B strategy

Mender - A/B

Two partitions are on the device

- Bootloader aware of “active”
- Update streams to “inactive”
- Automatically revert to previous partition on failure

Now let’s handle the size of our builds.



yocto PROJECT

Custom Linux distributions for any hardware architecture

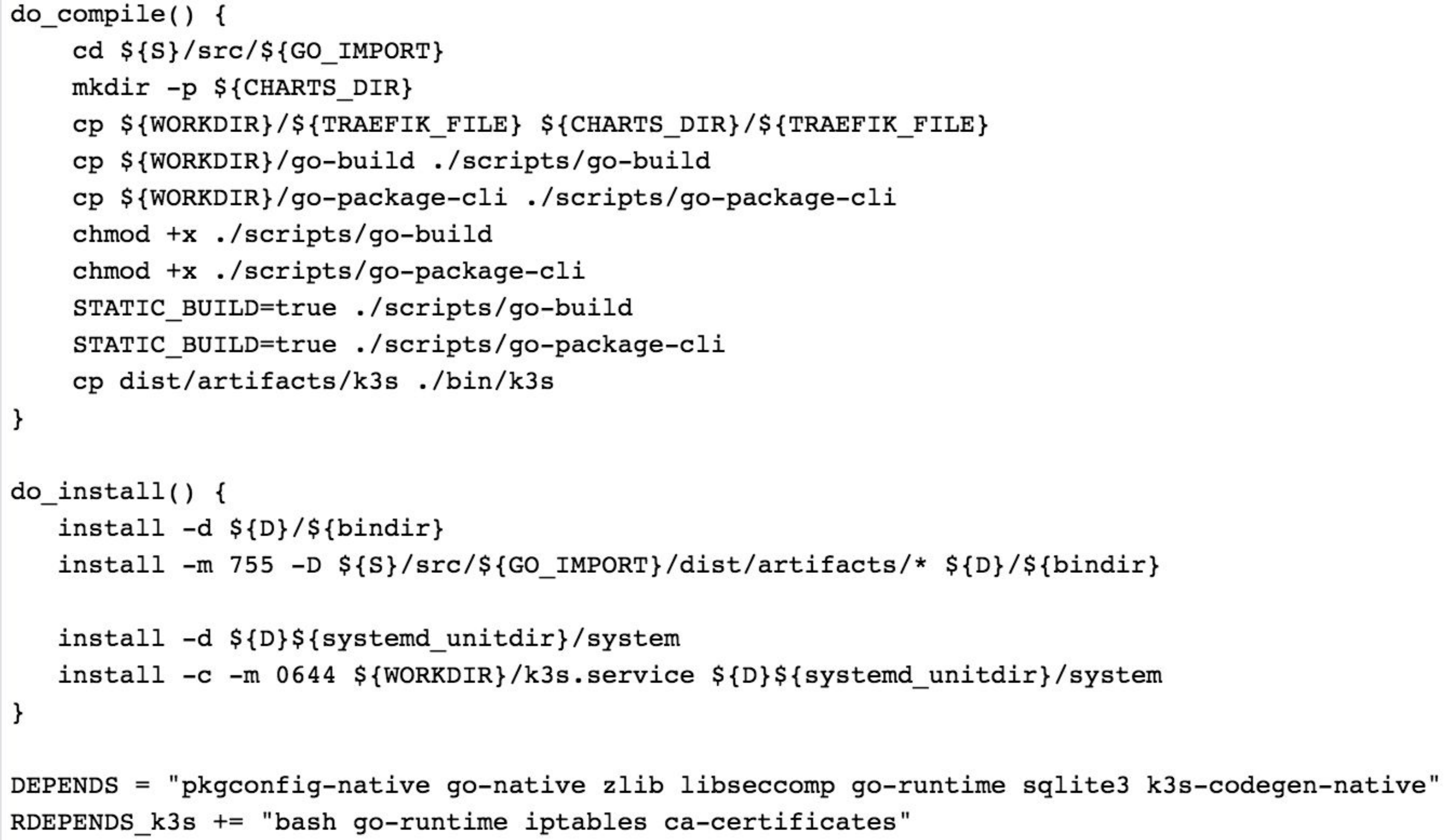
Copyright © 2019 JFrog. All Rights Reserved

Yocto Overview

- Eliminates OS bloat
- Drastically reduces resources required
- BitBake recipes and layers define your build
- Layers for common configurations are provided
- Custom layers to isolate applications or behaviors

Copyright © 2019 JFrog. All Rights Reserved

Yocto Layers



```
do_compile() {
    cd ${S}/src/${GO_IMPORT}
    mkdir -p ${CHARTS_DIR}
    cp ${WORKDIR}/${TRAEFIK_FILE} ${CHARTS_DIR}/${TRAEFIK_FILE}
    cp ${WORKDIR}/go-build ./scripts/go-build
    cp ${WORKDIR}/go-package-cli ./scripts/go-package-cli
    chmod +x ./scripts/go-build
    chmod +x ./scripts/go-package-cli
    STATIC_BUILD=true ./scripts/go-build
    STATIC_BUILD=true ./scripts/go-package-cli
    cp dist/artifacts/k3s ./bin/k3s
}

do_install() {
   install -d ${D}/${bindir}
   install -m 755 -D ${S}/src/${GO_IMPORT}/dist/artifacts/* ${D}/${bindir}

   install -d ${D}${systemd_unitdir}/system
   install -c -m 0644 ${WORKDIR}/k3s.service ${D}${systemd_unitdir}/system
}

DEPENDS = "pkgconfig-native go-native zlib libseccomp go-runtime sqlite3 k3s-codegen-native"
RDEPENDS_k3s += "bash go-runtime iptables ca-certificates"
```

@jfrog | Copyright © 2019 JFrog. All Rights Reserved

Yocto and Artifactory

- After first build, we can make things much faster
- Yocto cache allows for incremental updates
- Build cache can be stored in Artifactory
- Reduces time required to build by up to 50%

The Result - Firmware

- Cuts the total time after first build to 5-10 minutes
- Build is as small as possible
- Updates are signed and secure
- Automatic rollbacks in case of failure

Success!

Copyright © 2019 JFrog. All Rights Reserved

Wrapping Up

Edge and IoT updates are broken

This is a security problem that must be addressed

Modern DevOps tools are here to help

THANKS!

@Dixie3Flatline katc@jfrog.com
jfrog.com/shownotes

JFrog

Copyright © 2019 JFrog. All Rights Reserved