# Open Source secure software updates for Linux-based IVI systems

October 20, 2016

**Arthur Taylor, CTO**
**ATS Advanced Telematic Systems**

# Background

ATS Advanced Telematic Systems GmbH - Berlin start-up, founded 2013

ATS is AGL and GENIVI member since 2013

Have worked on client- and server-side software solutions for automotive

- Ported AGL to Freescale iMX.6-based automotive platform 2015
- Developed RVI Big Data solution (demo at AGL AMM Feb 2015)
- Developed Open Source SOTA solution 2015 (delivered for GENIVI AMM Oct 2015)
- Developed JASPAR WiFi / Miracast / NFC / Bluetooth handoff PoC in 2015/2016

Strong focus on and experience with Open Source

# This Session - Enabling OTA updates for Linux-based IVI systems

- Why OTA?
- Automotive update requirements
- Approaches for updating automotive Linux platforms
- Available open source tools and software
  - Server-side and client-side
- Status of our work enabling OTA in AGL Reference Platform and GDP
  - We welcome your feedback and collaboration! :)
- Roadmap to enable OTA for GDP builds

GENIVI®

# Background - Why OTA?

Increasing amount of software in vehicles

Increasing use of open source software

- Shared R&D efforts and shared software between manufacturers
- Shared vulnerabilities

Need to respond quickly:

- Deploy critical updates
- Deploy optimizations
- Deploy new features

# Background - Automotive OTA Requirements

- Atomic updates
  - Updates apply fully or not at all - no partial updates
- Revert to previous system on update failure
- Update of bootloader, kernel and configuration data, and filesystems
- Enable specific software features for specific vehicles
- Secure
  - Support for signing of images and verification of images
  - Support for TPM / TEE
  - Maintain confidentiality of software in transit

# Background - Automotive OTA Requirements

- Minimize download / transfer size
  - Fastest possible flashing time for lowest possible cost
- Synchronize with vehicle lifecycle
- Track update campaign success / failure
  - Installation / Error reports
  - Campaign tracking and overview
- Integrate with existing back-end / logistic / warranty systems
- Support diverse in-vehicle architecture
  - Multiple ECUs connected over different busses
  - Possible encrypted communication between ECUs

GENIVI®

# Background - Linux-based IVI Systems

## Automotive Grade Linux

Open Source - AGL Reference Platform

## GENIVI

Proprietary Linux-based [1]

Accenture, ADIT, Aisin, Delphi, Freescale, KPIT, LG, Magneti, Mentor XSe, Neusoft, NVidia, Continental, Pelagicore, QuEST, Renesas, Harman, TCS, Visteon, Wind River

Open Source

Tizen IVI

GENIVI Development Platform

## Open IVI

1. https://www.genivi.org/compliant-products

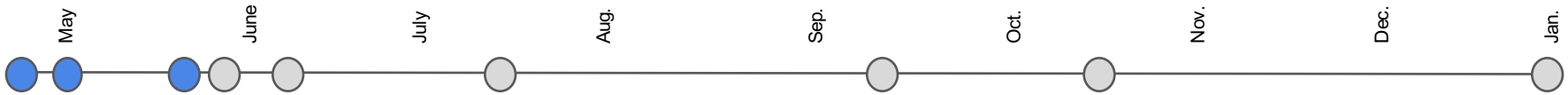
 

# Yocto

AGL RP, GDP, OpenIVI and many embedded projects use Yocto

| | GENIVI / GDP | AGL Reference Platform | Open IVI |
|---|---|---|---|
| **Base Layer** | meta-poky | meta-poky | meta-poky |
| **Arch Layer** | | meta-intel | meta-intel |
| **Middleware Layer** | meta-ivi | meta-ivi-common | |
| **Application Layer** | meta-genivi-demo-platform | meta-agl | meta-oim |

# Research and Planning




07/04 Added OTA Client to GDP

26/04 End-to-end integration of OTA Client with GDP @ GENIVI AMM

26/04 Discussed possible integration options for AGL with iot.bzh

01/05 Commissioned a study from Konsulko to investigate possibilities

SWUpdate, mender.io, resin.io, OSTree, swupd

11/05 Added OTA Client to meta-agl

24/05 Published study to AGL mailing list [1]

1. https://lists.linuxfoundation.org/pipermail/automotive-discussions/2016-May/002061.html

# Update Strategies - Master / Slave full-system

Master device receives update binaries

Master device updates Slave device

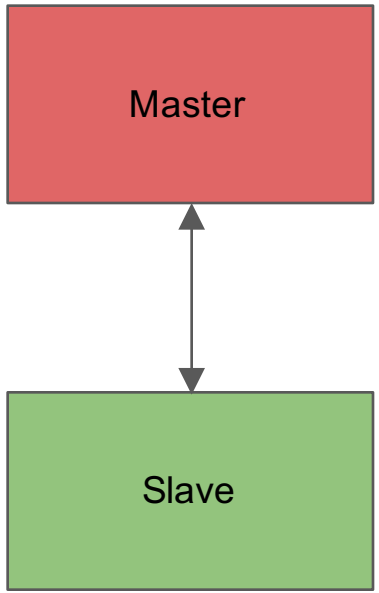
Master device validates Slave health

- Rollback if necessary

Robust, Simple to implement

Used in multi-ECU systems

What to do about the master?

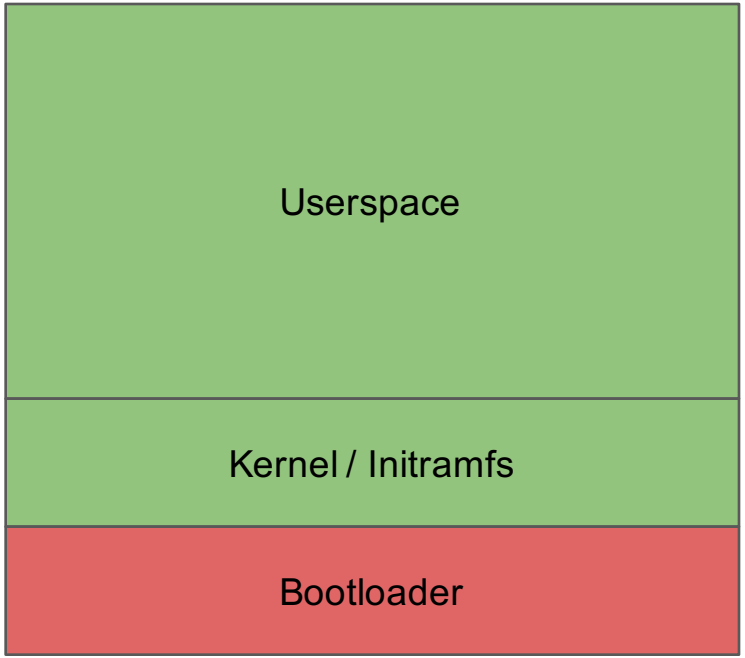


# Update Strategies - Full-system by Bootloader 1/2

Userspace downloads binaries

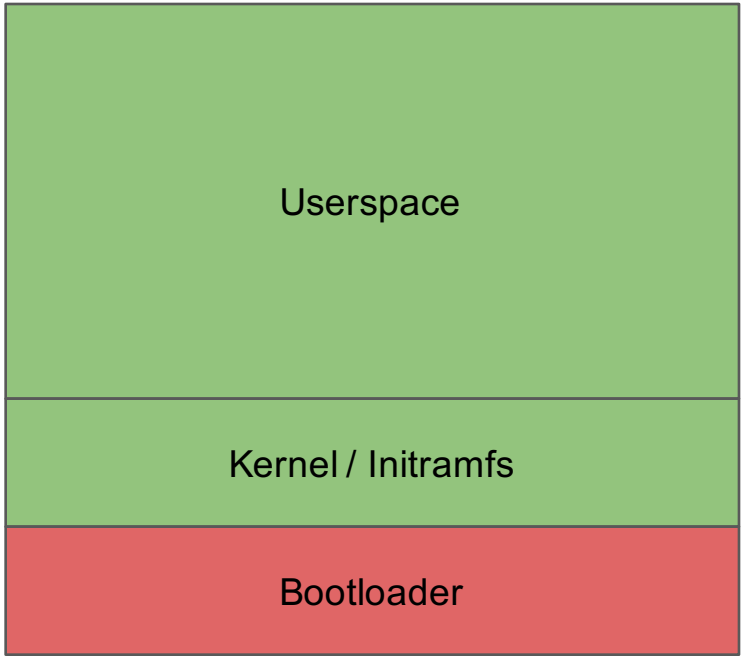
Reboot bootloader into update mode

Bootloader flashes new system



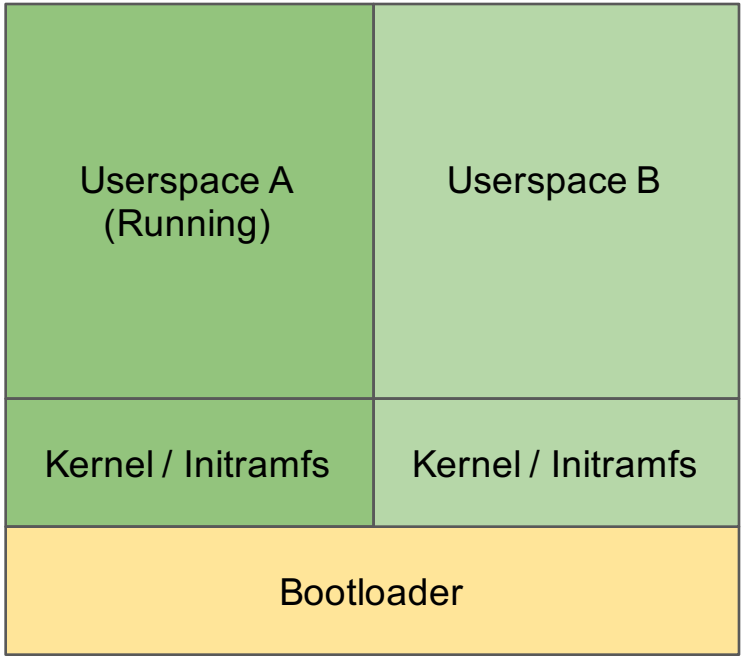
# Update Strategies - Full-system by Bootloader 2/2

- Implemented in **SWUpdate, Android (old)**
- Require temp storage for update files
- Have to hope that updated system boots
- Bootloader may not be capable of rollback
- How to update bootloader?



# Update Strategies - A/B full-system

- Userspace A receives update binaries
- Userspace A flashes Userspace B
- Userspace A validates Userspace B
- Userspace A notifies Bootloader
- Bootloader attempts to boot Userspace B
- Revert to Userspace A on error



# Update Strategies - A/B full-system

Implemented in **SWUpdate, Mender.io, rauc**

Similar approach in **CoreOS, Resin.io**
**Android (new), ChromeOS**

Atomic, with rollback support

Very robust against unbootable updates

Block-level - supports secure boot

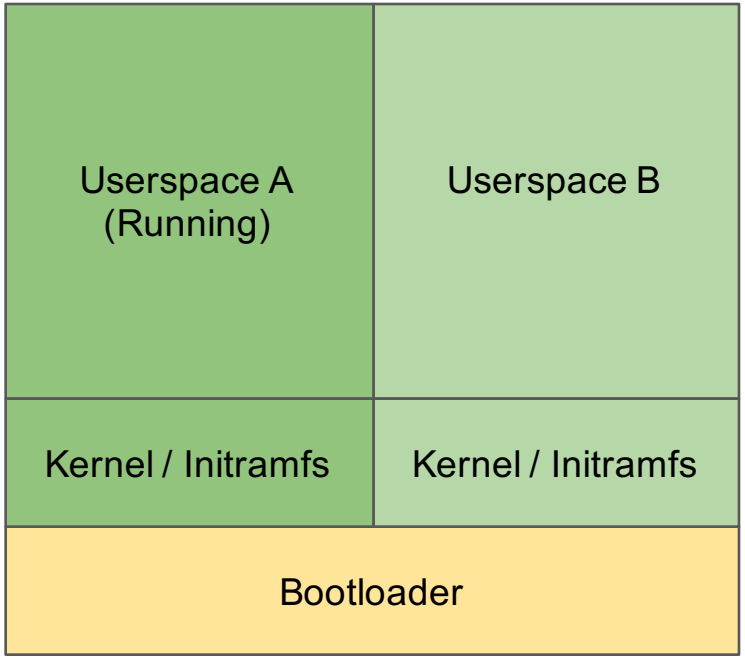
Requires 2 x Storage (eMMC, not RAM)

Bootloader updates must be done 'blind'

(hard to test before reboot)

All user settings on a separate partition

/home easy. /etc on overlayfs?

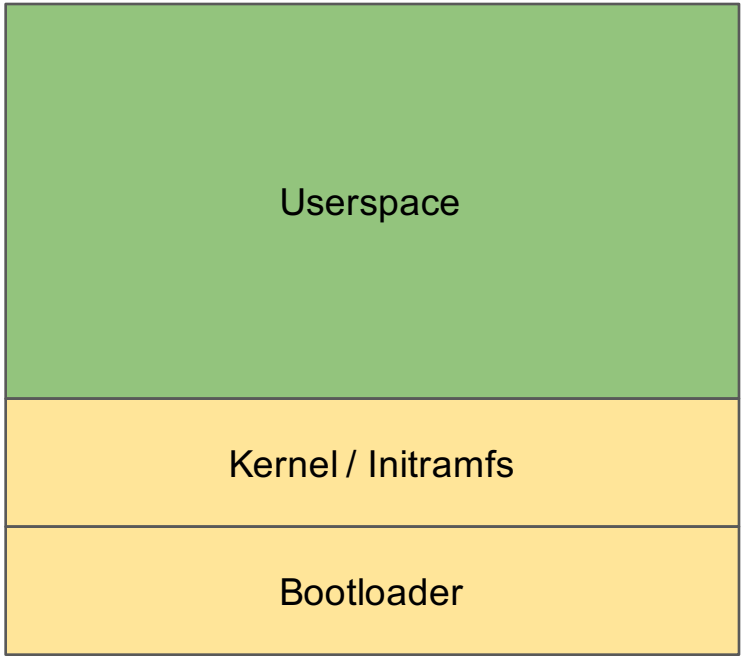



GENIVI®

# Update Strategies - Progressive by Userspace

Download updated binaries

Install them into the running system



# Update Strategies - Progressive by Userspace

Implemented in desktop / server Linux systems

- .deb, .rpm, etc.

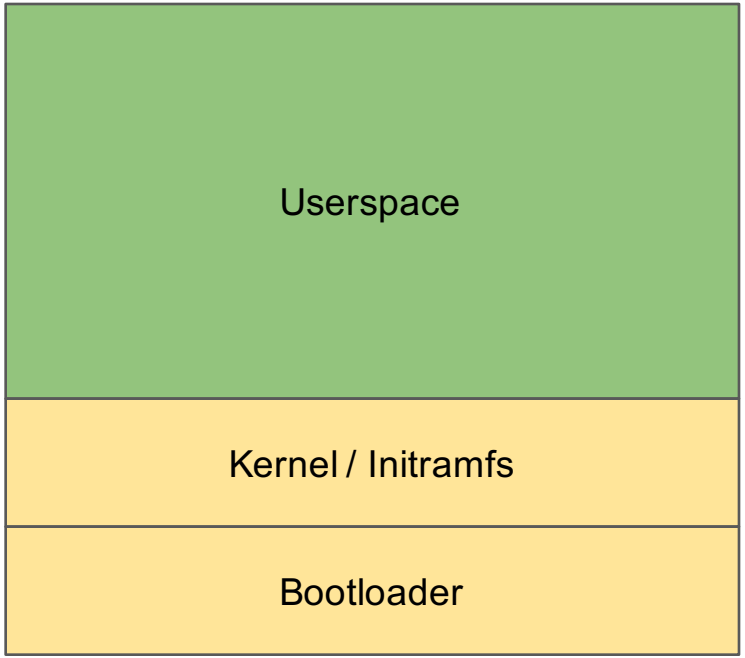
Similar approach in **OSTree, swupd**

- OSTree - Git-like tree of hard-links
- swupd - software "bundles"

Support for file-level updates

Package approaches widely used

- Easy to package and distribute updates
- Update binaries highly portable
- Rollback can be tricky



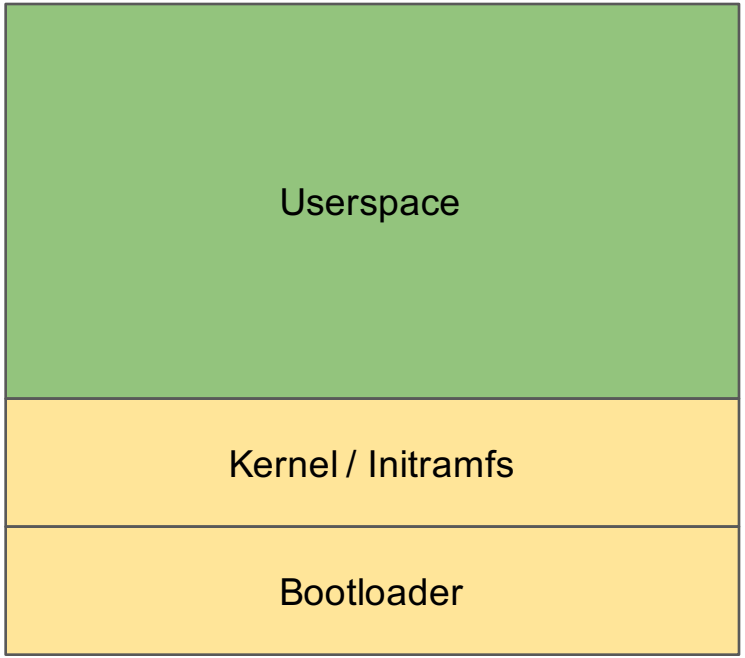

GENIVI®

# Update Strategies - Progressive by Userspace

OSTree approach has good traction

Used in **GnomeContinuous**, **QtOTA, Project Atomic, Flatpak**

Allows atomic updates and rollback




GENIVI®

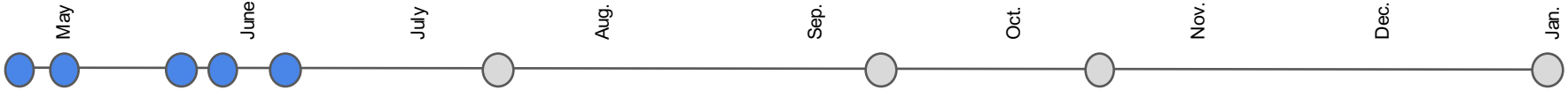
 

# Implementation

30/05 Agreed with AGL team during Vannes F2F on OSTree approach

01/06 Analyse technical risk of implementation

In particular risks with OSTree and TEE




1. https://lists.linuxfoundation.org/pipermail/automotive-discussions/2016-May/002061.html

# OSTree - "git for filesystems"

- Root filesystem-level version control
- "Check-in" filesystem changes after updates
- Sync repositories between devices
- "Check-out" filesystem changes on target device
- Atomic changes on checkout
  - Updates hard-links to files on filesystem
- Easy rollback



# Challenges - OSTree

Needs integration with target OS

Requires modification of bootloader / initramfs for full root-fs integration

What to do about modified files?

- Configuration (/etc)
- App working data (/var)
- User data (/home)



# Challenges - OSTree

## ostree-admin requirements

/sysroot must be an empty folder (where the OS will reside)

/sysroot is bind-mounted at boot-time to /

(Symlink /ostree -> /sysroot/ostree)

Only /var is preserved across updates

/home -> /var/home, /opt -> /var/opt, /srv -> /var/srv

/root -> /var/roothome, /usr/local -> /var/local, /mnt -> /var/mnt

/tmp -> /sysroot/tmp

… plus a bunch of other well-documented stuff[1]

1. http://ostree.readthedocs.io/en/latest/manual/adapting-existing/

# QtOTA Approach

Just introduced in Qt 5.7 for Device Creation[1][2]

Introductory blog posts from Gatis Paeglis[3][4]

Support for Intel/ARM, Grub2 and uboot

Easy to implement on Yocto-based systems:

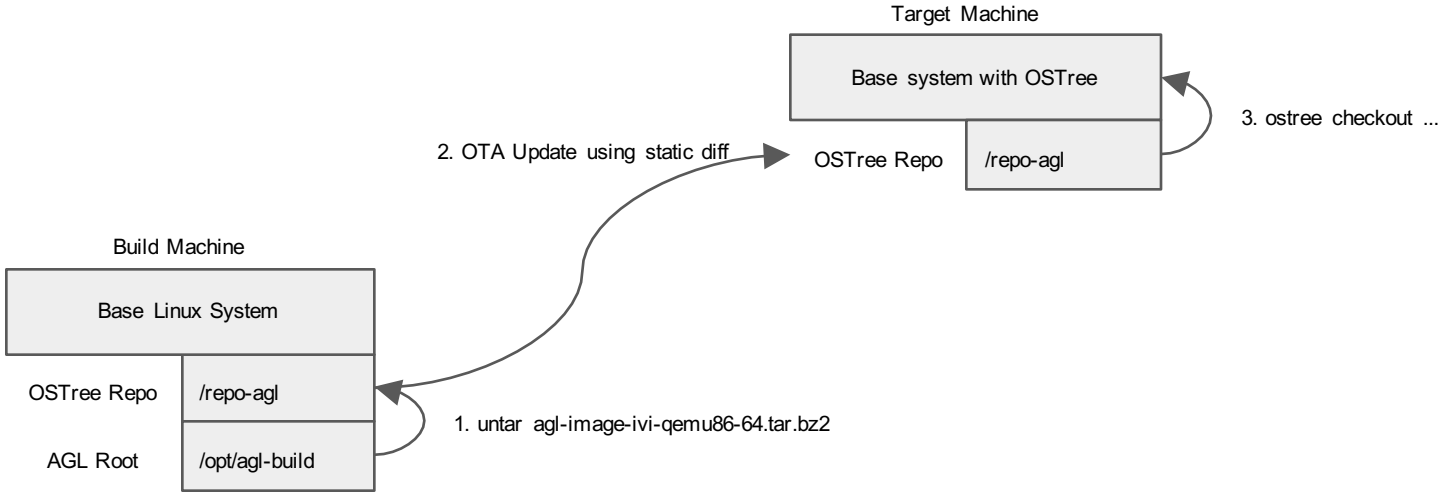
- Generate initramfs (`SDK_INSTALL_DIR/Tools/ota/dracut/generate-initramfs`)
- Generate bootloader (`SDK_INSTALL_DIR/Tools/ota/qt-ostree/ostree-grub-generator`)
- Convert your sysroot into an OTA enabled sysroot (`sudo ./qt-ostree --sysroot-image-path ...`)
- Deploy the generated OTA image to an SD card
- Check that system is OSTree enabled (`ostree admin status`)
- Generate update (`sudo ./qt-ostree --sysroot-image-path …`)
- Deploy update

1. http://code.qt.io/cgit/qt/qtotaupdate.git/
2. http://doc.qt.io/QtOTA/
3. https://blog.qt.io/blog/2016/05/31/over-the-air-updates-part-1-introduction/
4. https://blog.qt.io/blog/author/gapaegli/

# Chosen Approach - ostree-builtin

ostree-builtin - minimal tool to apply changes to hard-link tree

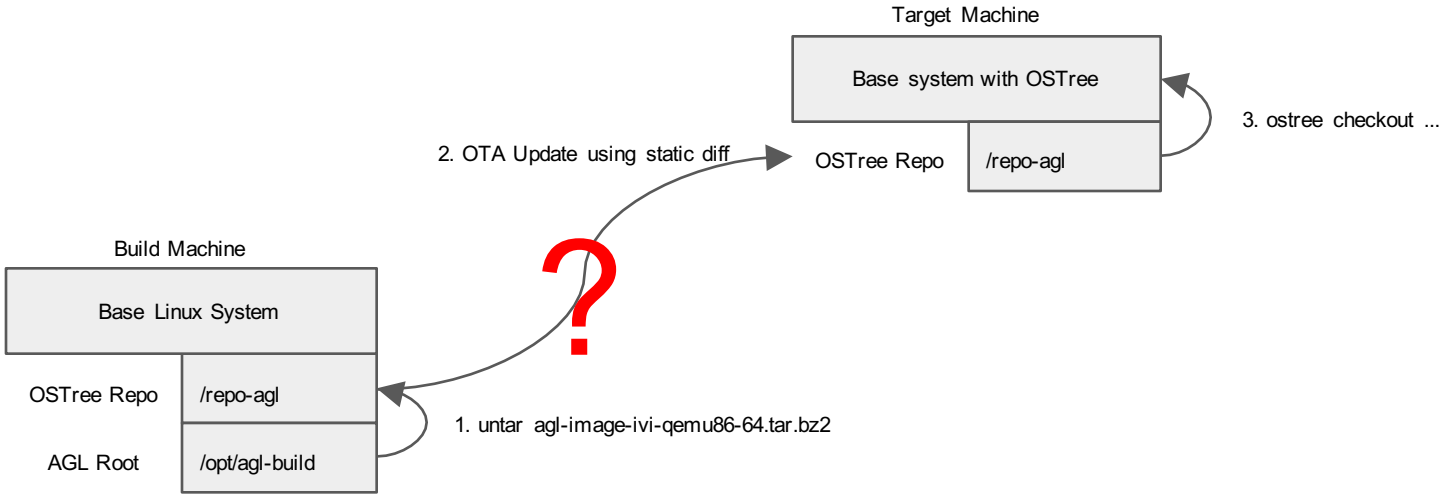
Allows us to create and deploy updates without modifying the OS



# Transmitting the updates

How to get OSTree diffs from build machine to client device?



# Transmitting the updates - GENIVI SOTA Server

Project for fully open source OTA management system

Commissioned by GENIVI / Jaguar Land Rover in 2015

Developed by ATS and delivered as PoC in October 2015

ATS continued development since, maintainers of GENIVI upstream

- Over 50% of ATS‘ resources dedicated to SOTA development for the last year

Soon to be available SaaS from ATS, GENIVI

- Also from AGL pending feedback / interest from members

1. https://genivi.github.io/rvi_sota_server

# Transmitting the updates - GENIVI SOTA Server 1/2

## 100% Open Source OTA Campaign Management system

Easily update individual devices, or create templates for small groups

Manually or programmatically create scalable campaigns for millions of vehicles

## Campaign Monitoring / Reporting

Progressively roll out updates, monitoring installation reports

Block update queues when errors are detected

Export Campaign status to external analytics dashboards and ERP tools

1. https://genivi.github.io/rvi_sota_server

# Transmitting the updates - GENIVI SOTA Server 2/2

- Microservice architecture
  - Horizontally scalable - add support for 1,000's or 1,000,000's rapidly
- Open, fully documented REST APIs[1]
  - Integrate with existing back-end systems
  - Develop custom monitoring and management tools

1. https://genivi.github.io/rvi_sota_server

# Transmitting the updates - GENIVI SOTA Server 1/2

Format-agnostic, secure transmission of binaries

Take binaries from existing back-end systems and deliver them to client

Support for packages (RPM, .deb, etc.), filesystem images (with diff), ECU / firmware

Validate secure, authentic delivery of binaries on client side before installation

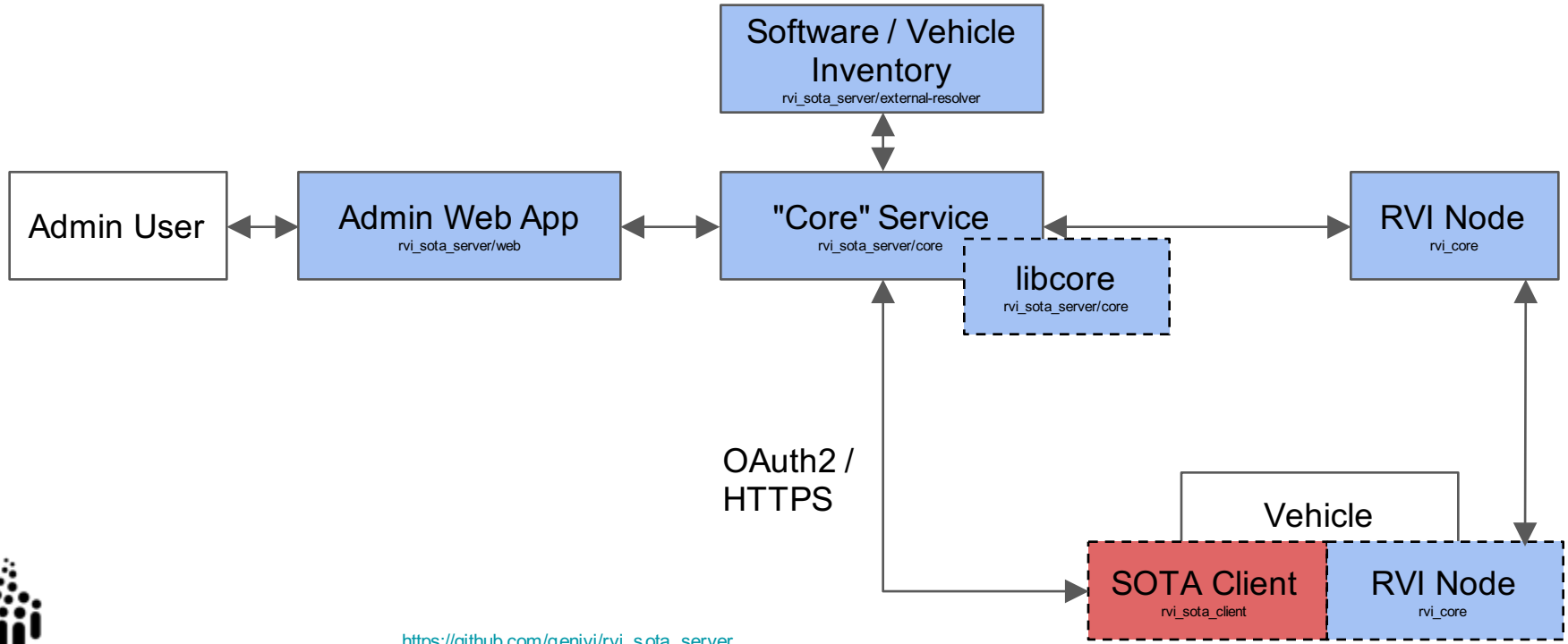


# Transmitting the updates - GENIVI SOTA Server 2/2

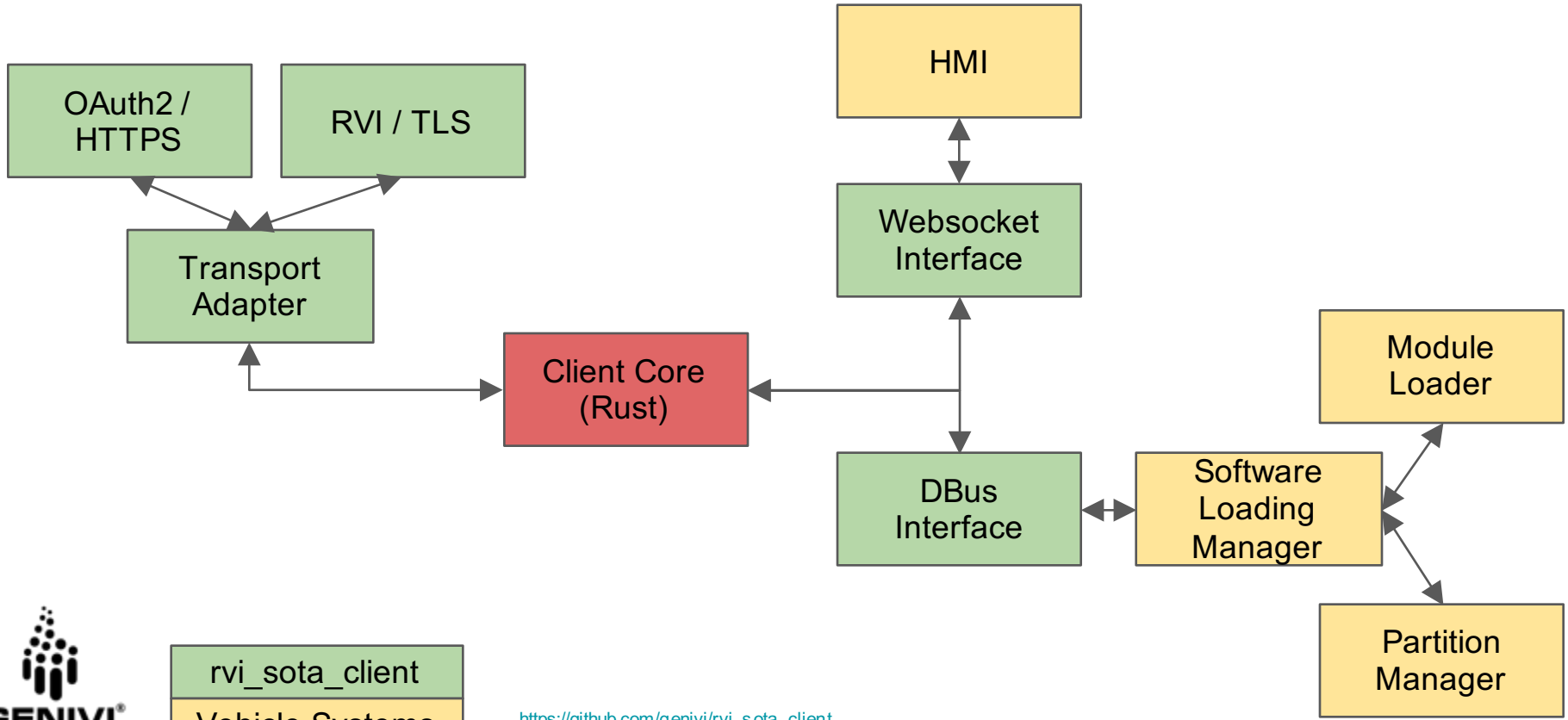
- Best-of-breed security architecture
  - Designed to mitigate all common attacks on software updaters
  - Defense in depth for update authenticity
    - Encrypted transit
    - Binary encryption
    - Binary signing
      - Hierarchical PKI, robust against key compromise
    - Independent metadata signing
      - Secure against mirror-poisoning, dependency injection, rollback attacks

# Software Architecture - GENIVI SOTA Server




https://github.com/genivi/rvi_sota_server

# Software Architecture - GENIVI SOTA Client



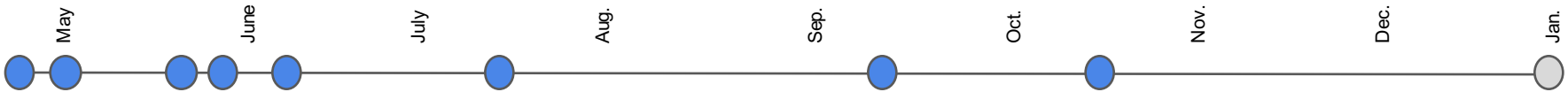

https://github.com/genivi/rvi_sota_client

# Live Demo

13/07 Live demo at ALS, Tokyo

07/09 Live demo at AGL AMM

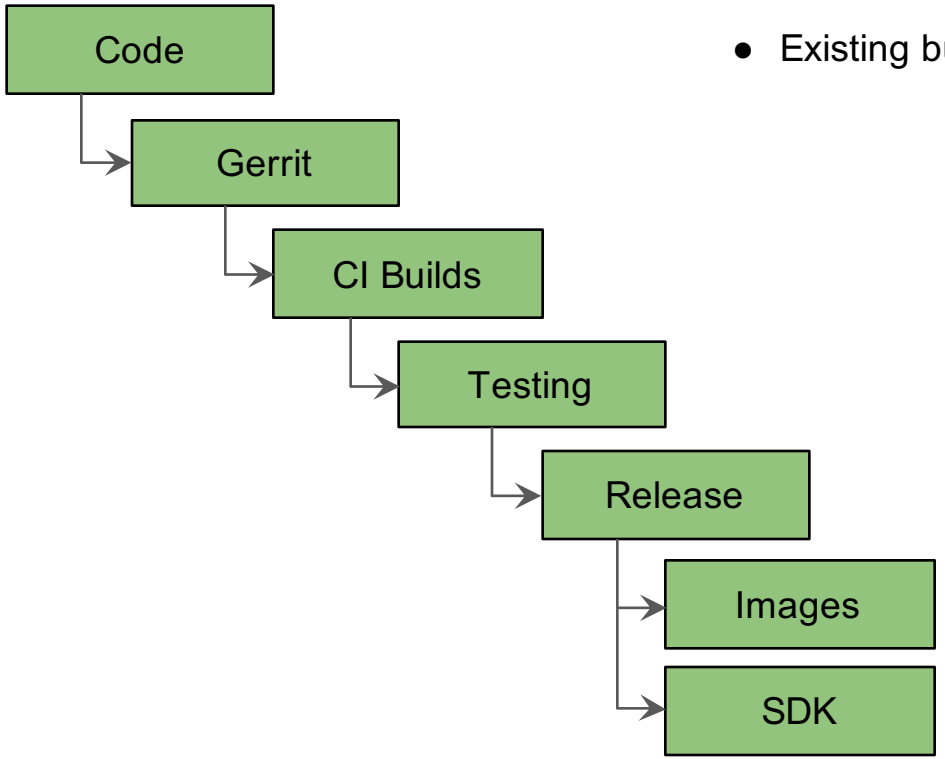
20/10 Live demo today!




GENIVI®

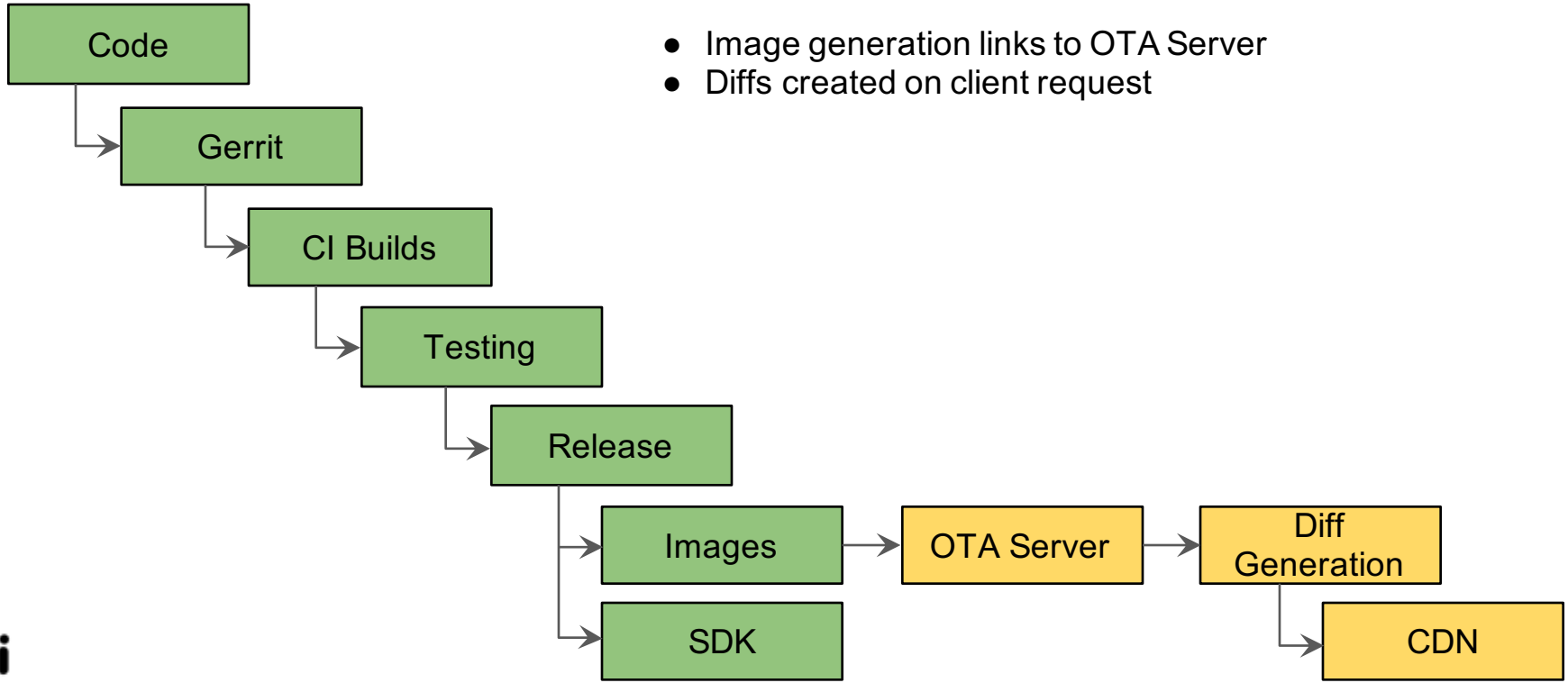
 

# Integration with GDP Buildchain




- Existing buildchain for GDP Images

# Integration with GDP Buildchain




- Image generation links to OTA Server
- Diffs created on client request

# GDP Implementation Plan

## Phase 1

- Launch SOTA Server on GENIVI infrastructure
- Enable multi-user support, provide accounts to GENIVI members / GDP users
- Use existing SOTA Client / SWLM integration to install RPM packages

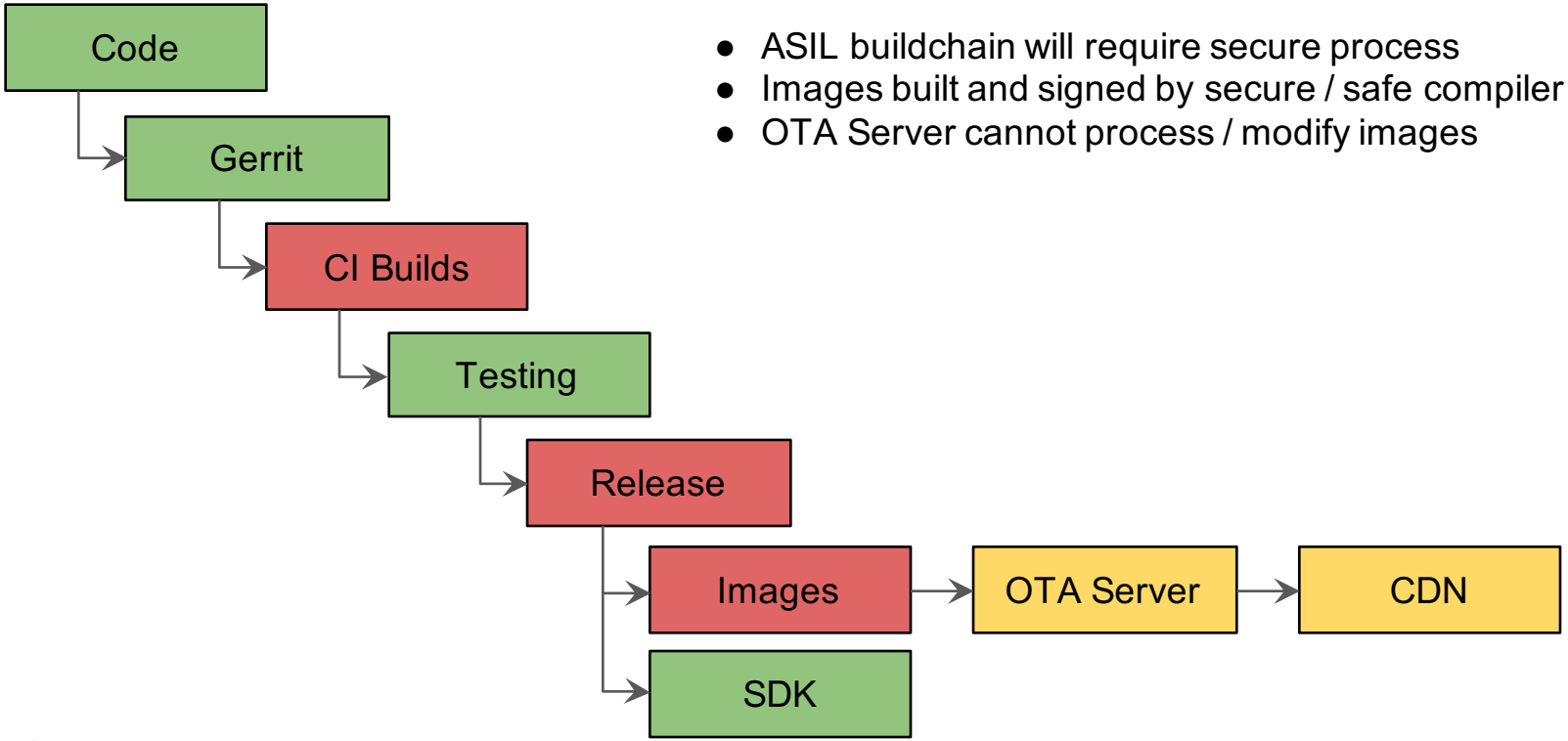
## Phase 2

- Integrate OSTree approach with GDP builds
- Extend SOTA Server to handle OSTree deltas
- Extend SOTA Client to process OSTree metadata / downloads
- Extend SWLM to install OSTree updates

GENIVI®

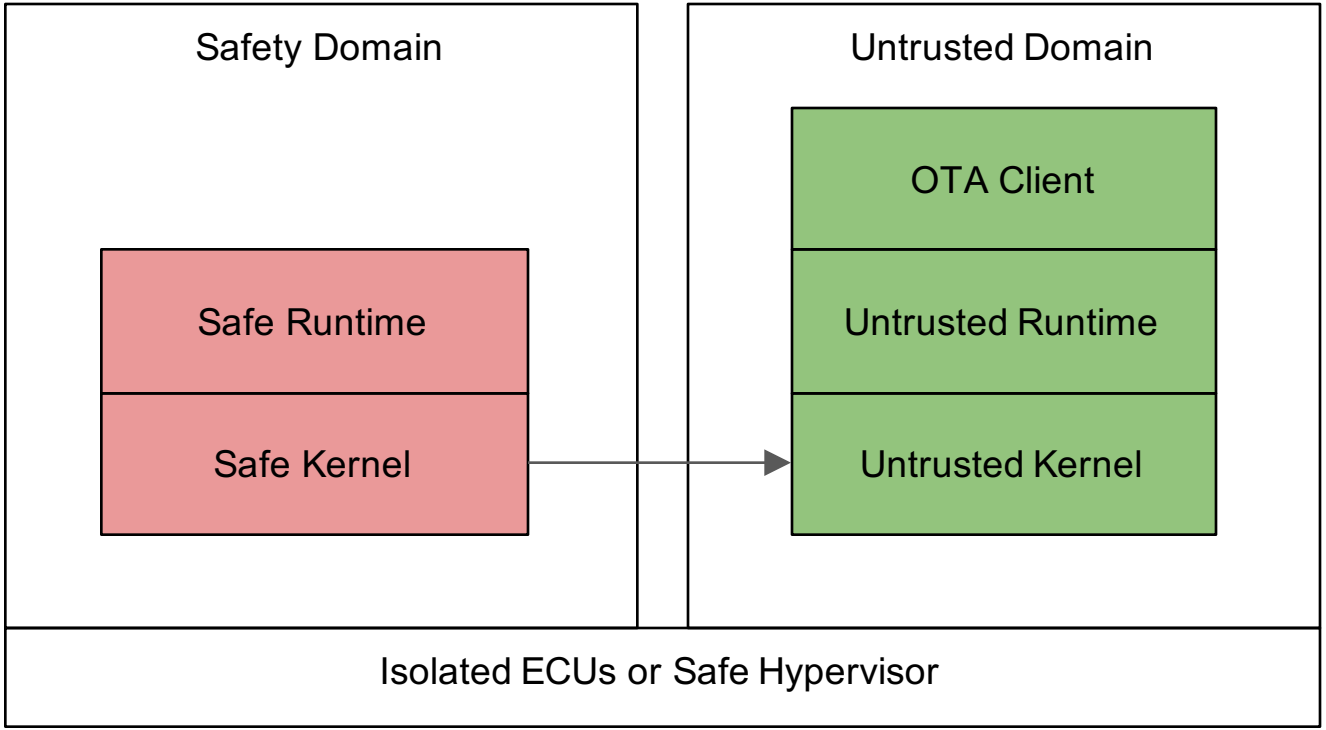
# Security - Integration with ASIL Buildchain



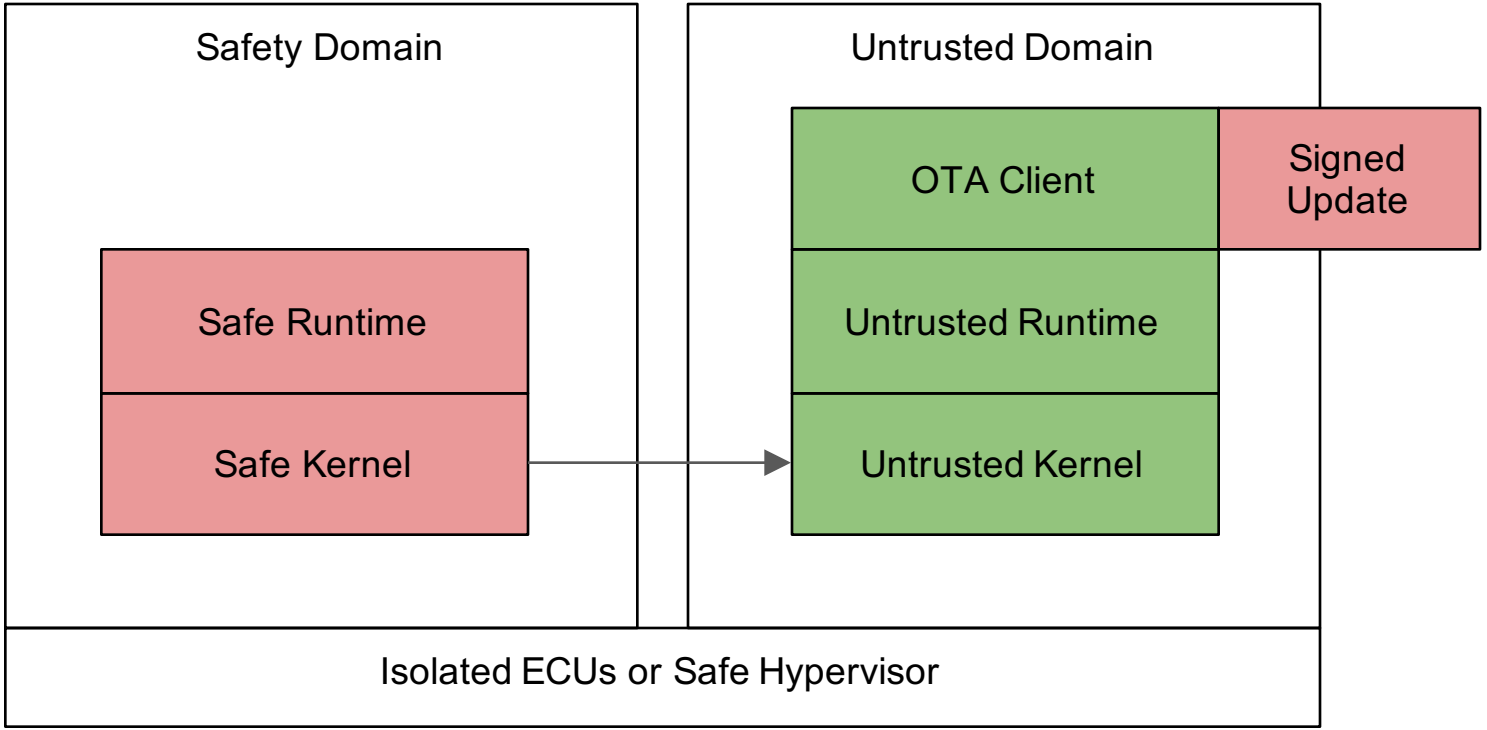

- ASIL buildchain will require secure process
- Images built and signed by secure / safe compiler
- OTA Server cannot process / modify images

GENIVI®

# OTA Updates to ASIL domain

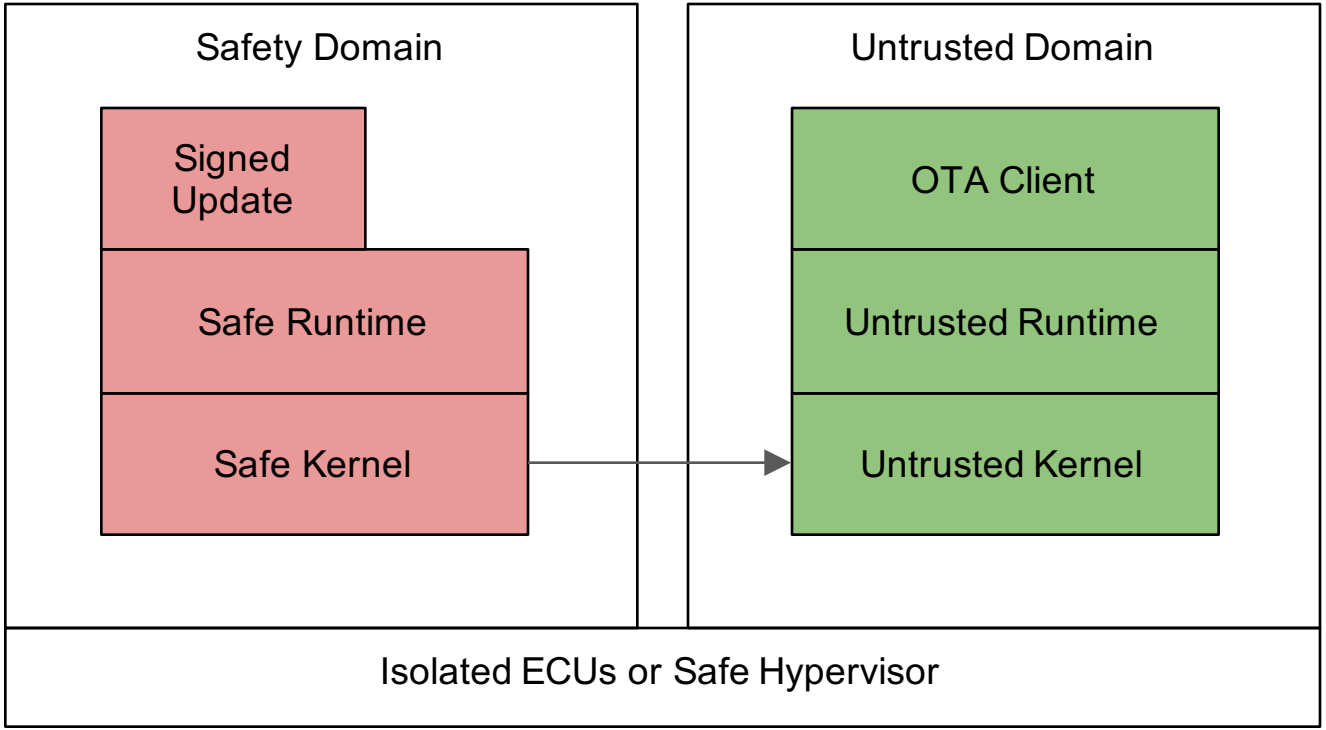


# OTA Updates to ASIL domain




 | 

# OTA Updates to ASIL domain



# Summary

OTA Client is already in AGL distribution (meta-agl-extra)

- OSTree is also available now (meta-agl-extra)
- Will maintain those recipes to keep software available to AGL users

OTA Client is already in GDP distribution

GENIVI SOTA Server is available for download

- Soon to be running SaaS at multiple locations for easy evaluation
- Will continue to maintain and develop SOTA Server



# Next Steps

Integrate into AGL SDK and Buildchain

Integrate into GDP SDK and Buildchain

Implement more automotive platforms - suggest a target!

Extend to include bus attached ECUs - suggest some ECUs!

Share requirements with us!

Build a PoC integration with us!

Next steps are up to you!

# End-to-End Integration

Target CES in January for AGL integration

Working to schedule GDP integration now