Creating A Truly Open Automotive Distribution with Automotive Grade Linux

Walt Miner - Community Manager, Automotive Grade Linux

wminer@linuxfoundation.org

LINUX FOUNDATION

AUTOMOTIVE GRADE LINUX

Who Is This Guy?

- Linux Foundation / AGL Manager since 2014
- Prior 15 years a mix of Tier 1 Automotive Suppliers and Mobile Devices
  - MontaVista / Mentor Embedded
  - Continental BU Infotainment and Connectivity
  - Motorola Mobile Devices
  - Motorola Telematics
- Defense Aerospace prior

LINUX FOUNDATION

AUTOMOTIVE GRADE LINUX

Goals

- Understand what AGL is and how it compares to other automotive platforms including GENIVI, QNX, Apple, Google
- Generate interest in developer community to participate in AGL

Just a Thought

- Automotive Linux succeeds with
  - Cooperation with Open Source community and traditional automotive software developers
  - Multiple vehicle OEMs and Tier One suppliers providing a robust set of use cases and requirements

Automotive Grade Linux

AGL Introduction and History

Automotive Grade Linux

Collaborating to build the car of the future through rapid innovation

http://AutomotiveLinux.org

AGL Member Companies

Gold

Silver

Bronze

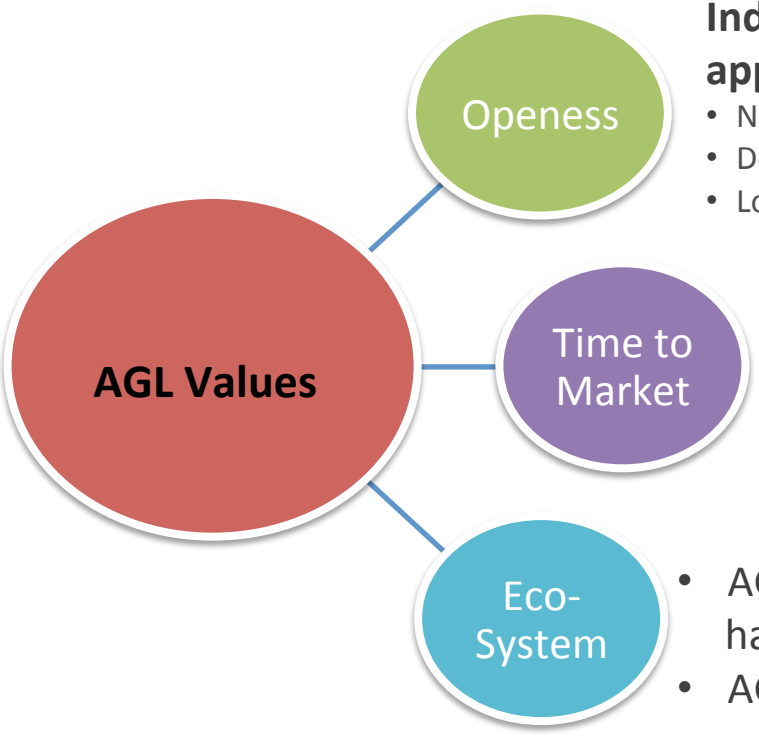
LINUX FOUNDATION

AUTOMOTIVE GRADE LINUX

AGL Charter and Scope

- Open collaboration project to enable rapid innovation in the automotive industry by leveraging Linux and Open Source technologies
- Develop reference distributions for multiple automotive applications
  - In-Vehicle Infotainment (IVI), Cluster, HUD, Telematics, Control Systems
- Develop reference distributions on multiple hardware platforms
- Develop requirements specifications
  - Which leads to implementation and creation of AGL software projects
- Work with upstream projects
  - Contribute AGL specific code to upstream projects where necessary
- Educate the industry in open source collaboration and best practices

Why is AGL important?



Industry needs a standardized <u>open</u> operating system and application framework

- Not under the control of any one company (e.g. Google/Apple)
- Developed collaboratively between many companies
- Lower cost by sharing development work of the common bits

- AGL will **<u>decrease time to market</u>** and can provide functionality similar to the smart phone

- AGL **will <u>enable an ecosystem of "AGL Ready" suppliers</u>** for hardware, software, UI
- AGL will enable a global app developer ecosystem

Comparison of AGL and GENIVI

	AGL	GENIVI
True open source project, open developer community, open to non-members, goal to build developer ecosystem	Yes	No
“Code first” methodology	Yes	No
Single reference distribution (ARM and Intel)	Yes	No
Full IVI prototype (ARM and Intel)	Yes	No
Application Framework and HTML5/Native Apps	Yes	No
Requirements specification	Yes	**Members-only**
Compliance program	No	**Members-only**
Continuous Integration Infrastructure	Yes*	**Members-only**
System Architecture Team with automotive experience	Yes	Yes
Profiles for Instrument Cluster, HUD, Telematics	Yes*	No
Functional Safety	Yes*	No
Automotive specific middleware	Yes	Yes
Linux Foundation backing	Yes	No

AGL is the only organization that is fully open and plans to address all functions in the vehicle.

** Under development or planned for the future*

Comparing IVI Options

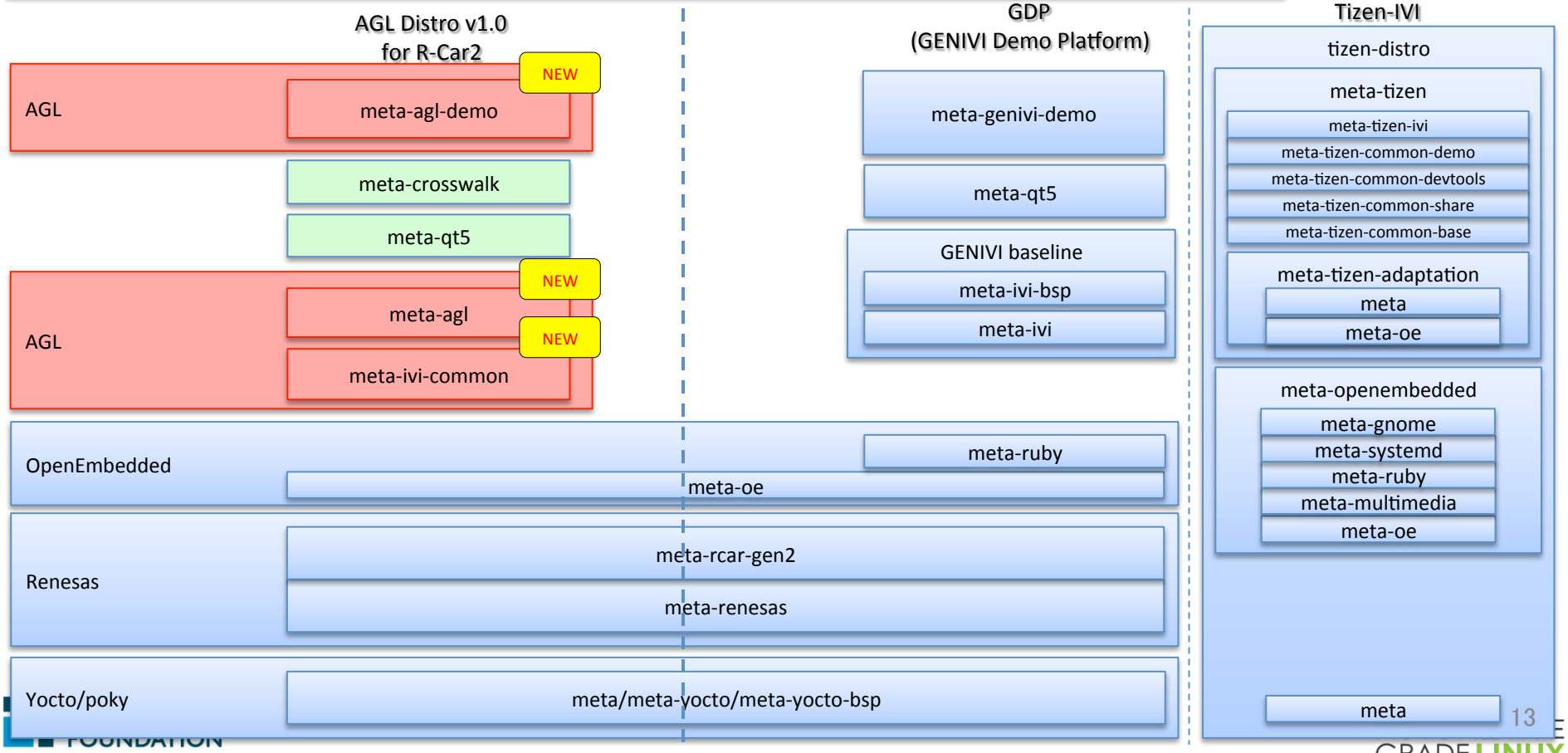
IVI System	Description
AGL	• Complete Linux-based IVI system embedded in the head unit • Open source collaboration between 50+ companies and growing • Plan to support HUD, Cluster, Telematics
QNX	• Incumbent proprietary OS for automotive • Long history in automotive business
Android Auto (Projection)	• It is a projection technology using a display protocol • Must be used with an Android smart phone
Apple CarPlay (Projection)	• It is a projection technology using a display protocol • Must be used with an iPhone/iPad
Android Embedded (Fork)	• OEM takes open source Android, forks it and uses is as an App framework • OEM is fully responsible for porting and long term maintenance • No support from Google
Android Embedded (from Google)	• This does not exist • Google has not made any announcement on whether they will build an embedded version of Android for the head unit

Why an AGL Distribution?

- Used Tizen IVI as distribution previously
- AGL Advisory Board determined that an independent code base was necessary
- AGL Distribution
  - Create the AGL Distribution for the entire industry – Unified Code Base
  - Reduce fragmentation by consolidating the best from AGL, Tizen IVI and GENIVI
  - Complete a full prototype including reference applications for both native Linux and non-native (HTML5) applications
- Future:
  - Address profiles for Instrument Cluster, HUD, Telematics

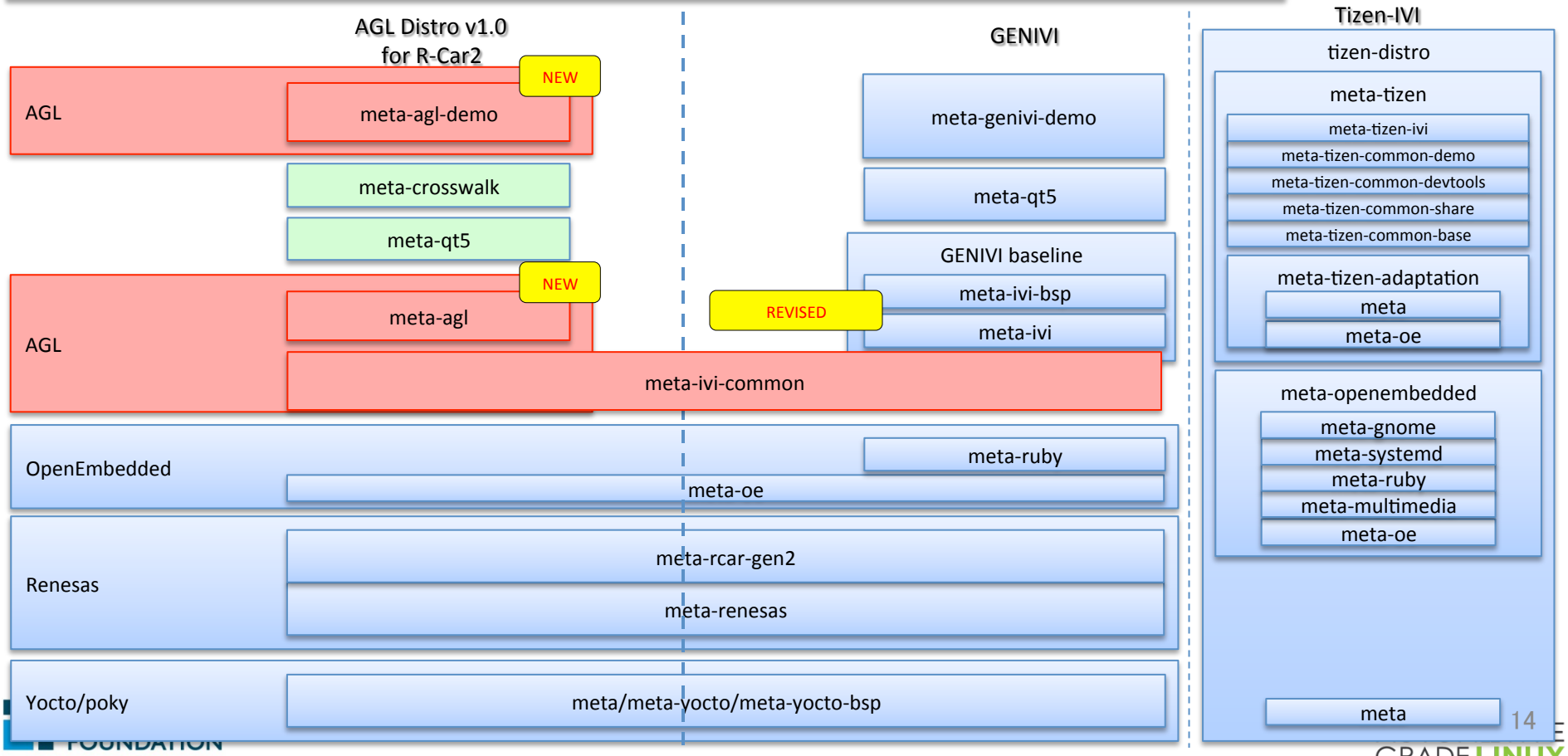
The Layers of AGL Distribution

Suggestion of Phase 1 layers and comparison to current GENIVI and Tizen IVI



The Layers of AGL Distribution

Suggestion of Phase 2 layers and comparison to current GENIVI and Tizen IVI



You demanded a code name!

- Agile Albacore – December 2015

- Brilliant Blowfish – June 2016

- Continue alphabetically from there

Agile Albacore

- Phase 1 - 90% Complete
  - Yocto layers and Bitbake recipes for a minimal AGL Build
    - ✓ QEMU
    - ✓ Renesas Porter
    - ✓ Sample Qt 5 apps
  - Linux Foundation Hosted Infrastructure
    - ✓ Git https://git.automotivelinux.org/
    - ✓ Gerrit https://git.automotivelinux.org/gerrit/#/
    - ✓ Jira https://jira.automotivelinux.org/
    - ✓ Jenkins
  - Preliminary code governance in place

Agile Albacore

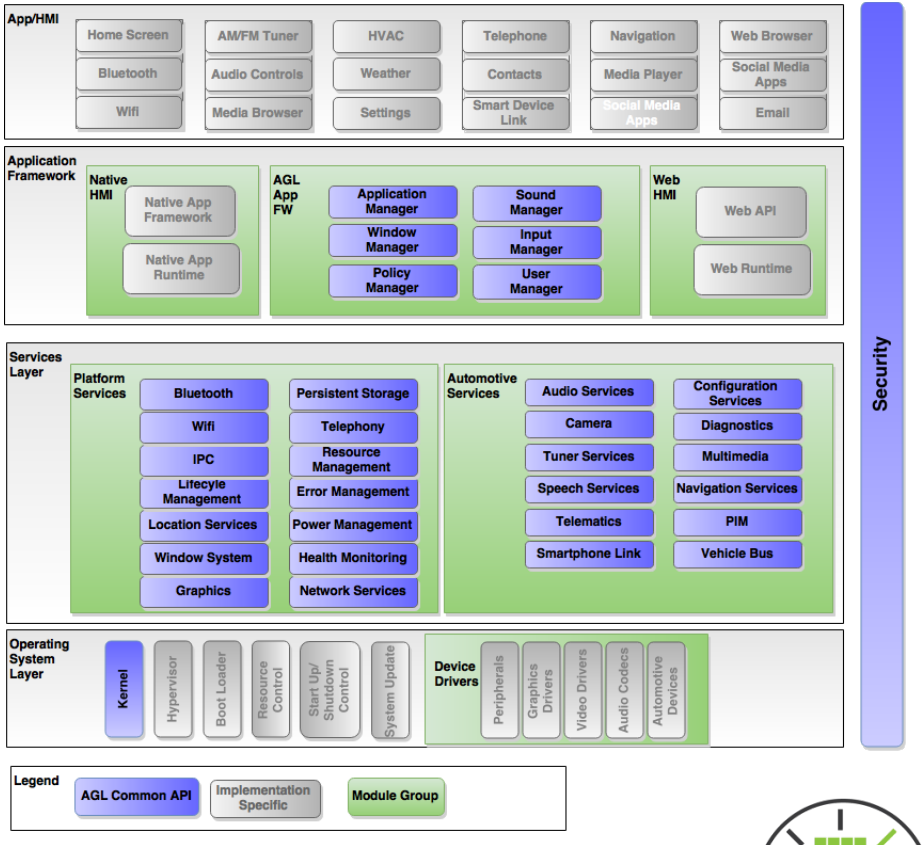
- Phase 2 – Target Completion December 1
  - Complete Build including Demo Apps for CES
  - Complete code governance in place
    - Patch management
    - Branching, Merging, and Tagging Code
    - Keeping in sync with upstream projects
  - Full CI system in place in cooperation with GENIVI
  - Test infrastructure in place and running

Brilliant Blowfish

- Target Date June 15, 2016
  - Yocto 2.0
  - Additional boards with SOCs from different silicon vendors
  - Telematics and Instrument Cluster Profiles
  - Mature Application Framework
  - Telephony

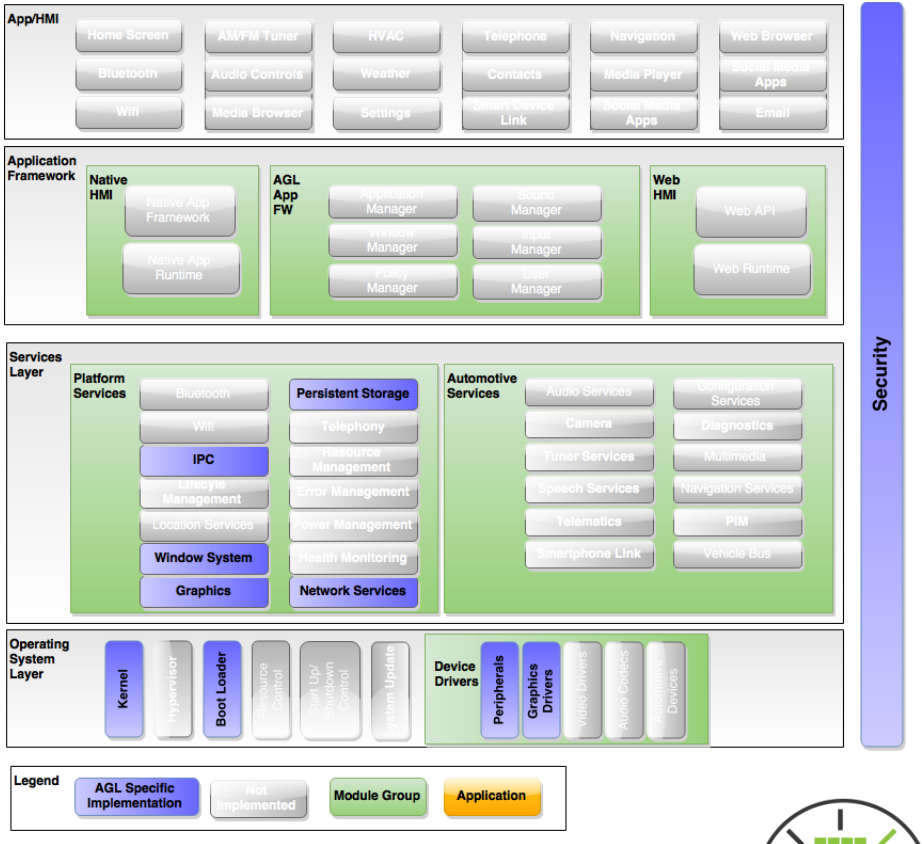
AGL Portability

- *AGL Services API* in the Services Layer *and AGL App Framework* are exposed to App Developers
- Native and Web runtimes are adapted using the *AGL App Framework* and *AGL Services API*



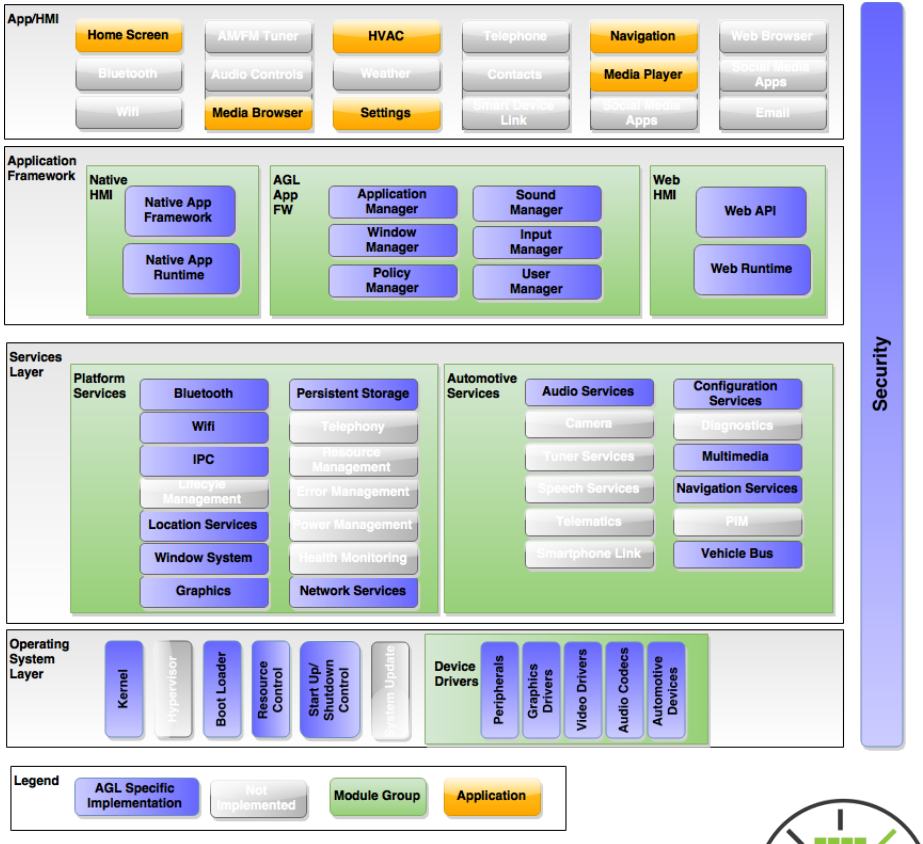
AGL Roadmap – Sep 2015

- Unified Code Base with GENIVI, Tizen, and AGL components.
- Create AGL Distribution using Yocto 1.7 for Renesas board and QEMU
- Minimal AGL build on LF infrastructure



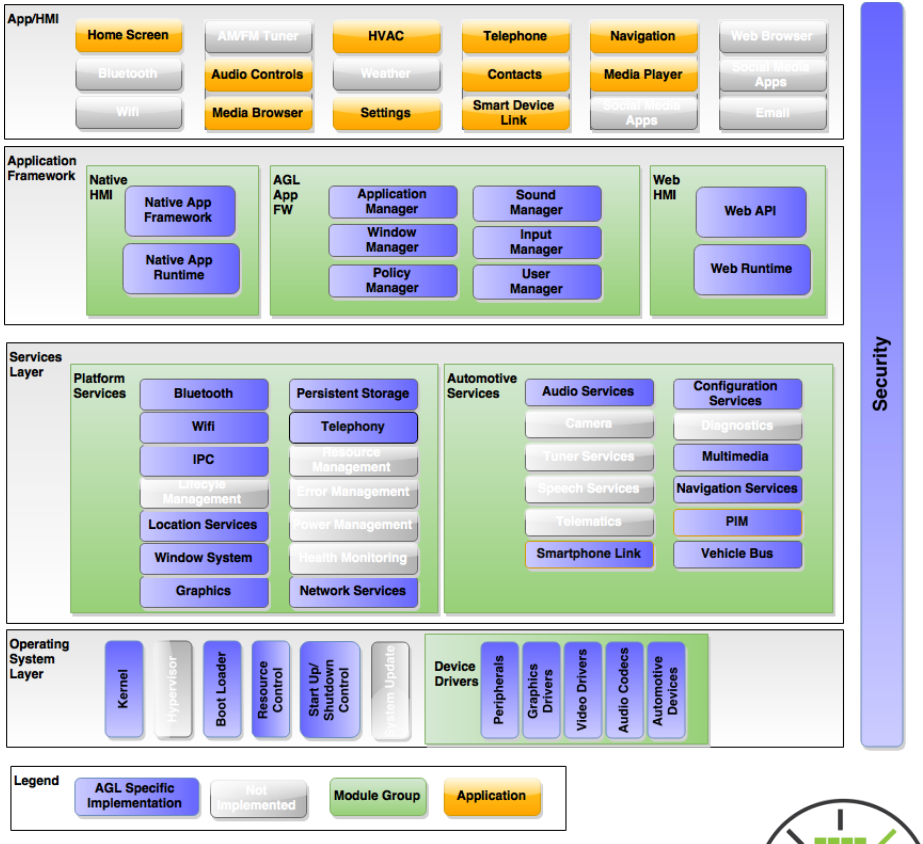
AGL Roadmap 2015

- Unified Code Base with GENIVI, Tizen, and AGL components.
- Create AGL Distribution using Yocto 1.7 for Renesas board and QEMU
- Continuous Integration and Automated Test infrastructure
- Qt5 based reference applications



AGL Roadmap 2016

- New hardware platforms (TBD)
- Yocto 1.9
- Mature Application Framework
- Implement device profiles (IVI, IC, HUD)
- Telephony



Getting Involved

- Most subsystems in need of developers and maintainers particularly user space
- Application developers needed
- Check Jira for open issues and tasks that need to be done
- Mail list
- IRC

Getting Involved

- Single sign-on for AGL sites including Jira, git, gerrit, DOORS NG, and the AGL Wiki
- Register at https://dev.automotivelinux.org/

Getting Involved

Web

http://automotivelinux.org/

Wiki

http://wiki.automotivelinux.org/

Mail Lists

http://automotive.linuxfoundation.org/what-is-automotive-grade-linux/automotive-grade-linux-mailing-lists

IRC

Channel #automotive at freenode.net

Webinars

http://automotive.linuxfoundation.org/webinars

Contribution Process

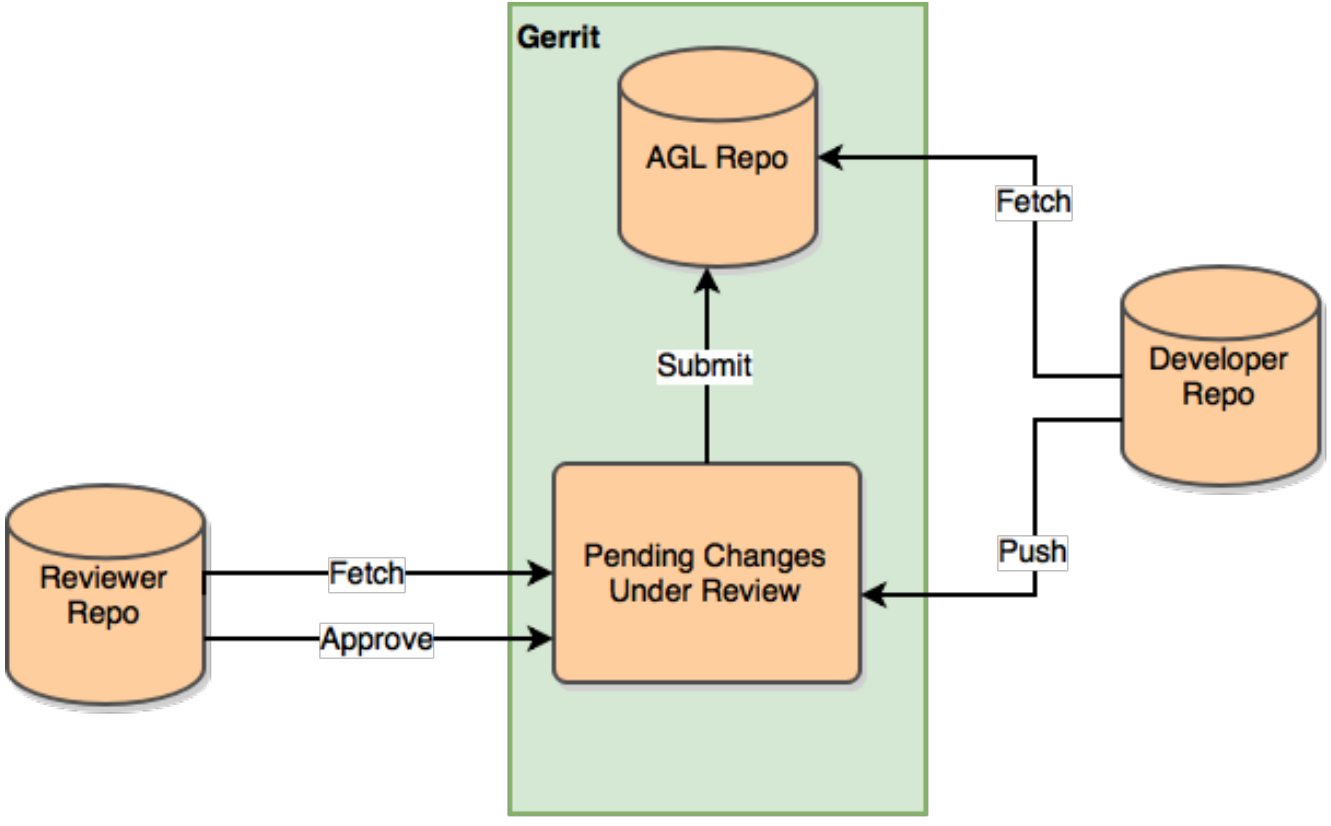
- Code development process is documented
  - https://wiki.automotivelinux.org/agl-distro/contributing
- Process continues to evolve

GIT AND GERRIT

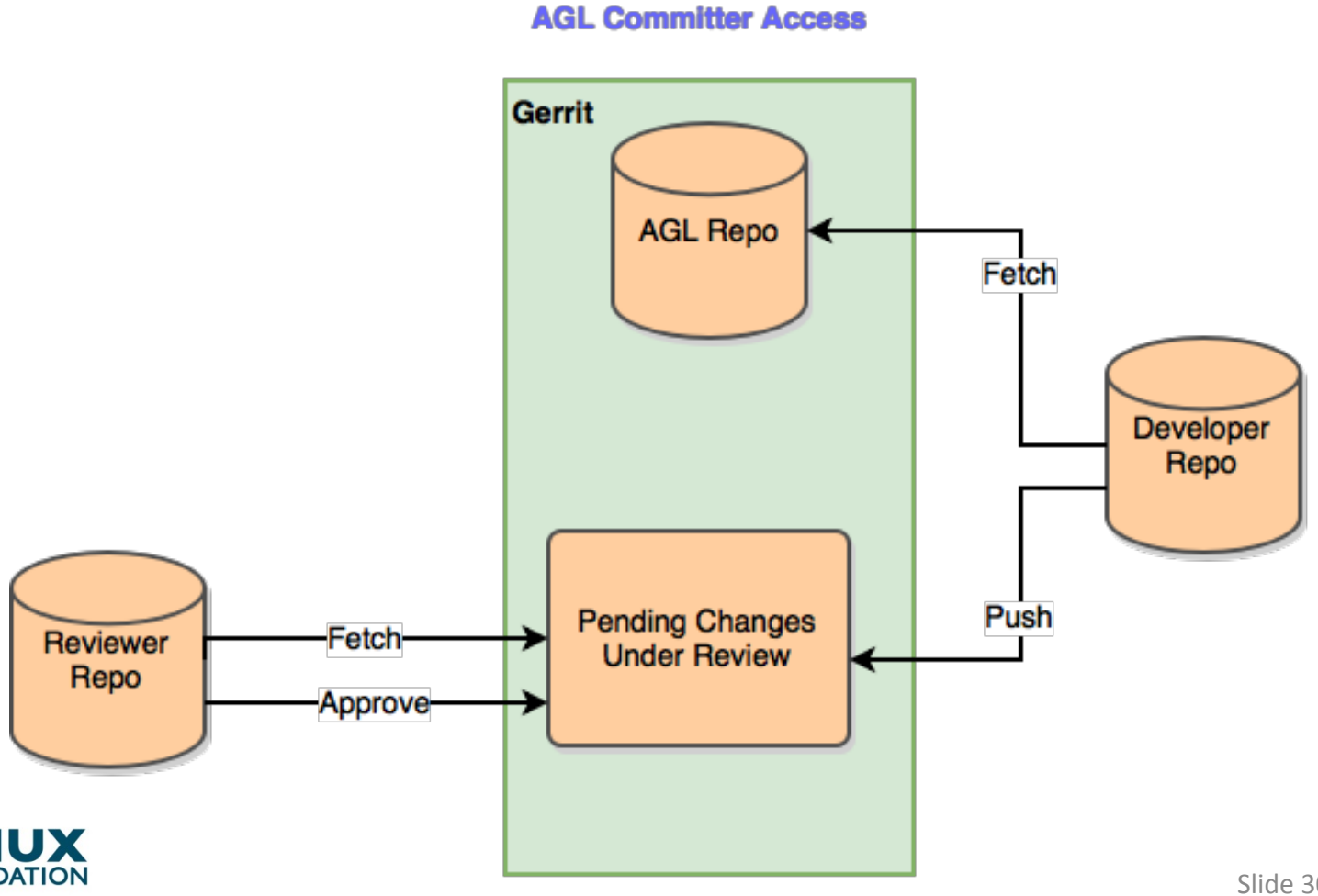
Git and Gerrit

- AGL uses git for version control and gerrit for code reviews
- Code and patch submissions are via gerrit and use the gerrit review and merge process
- These can be found at
  - https://gerrit.automotivelinux.org
  - https://git.automotivelinux.org

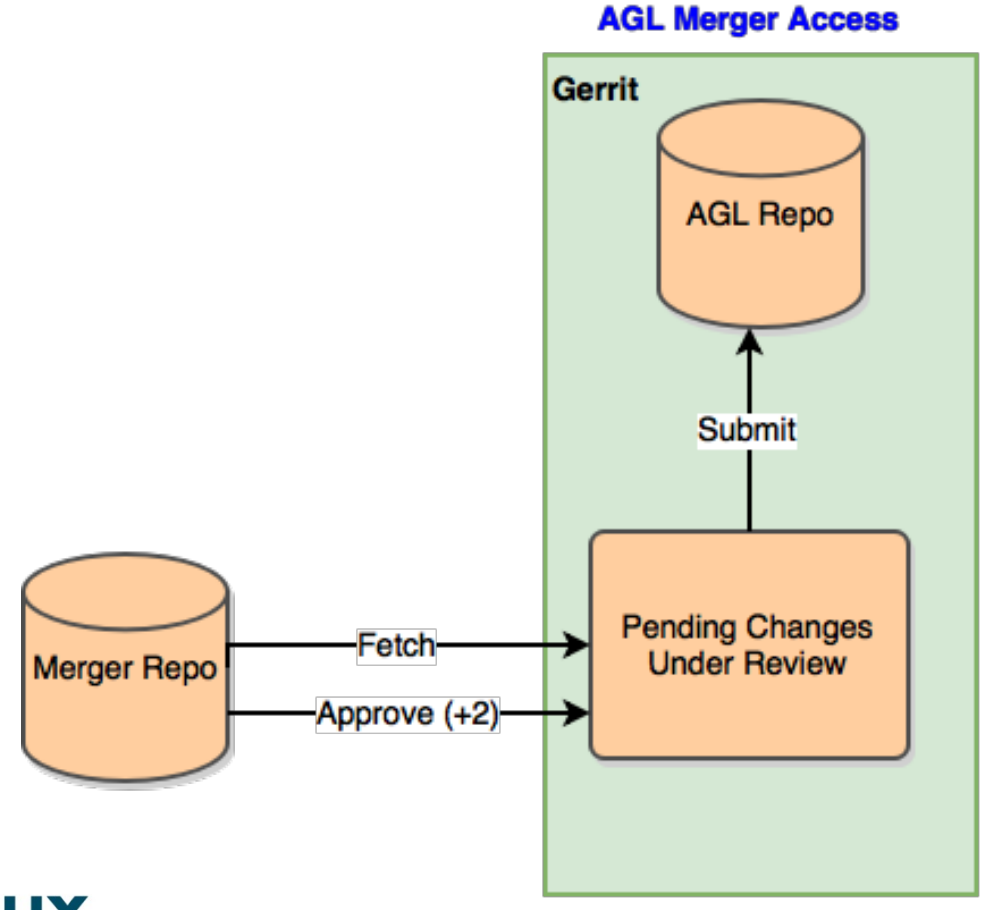
Gerrit Overview



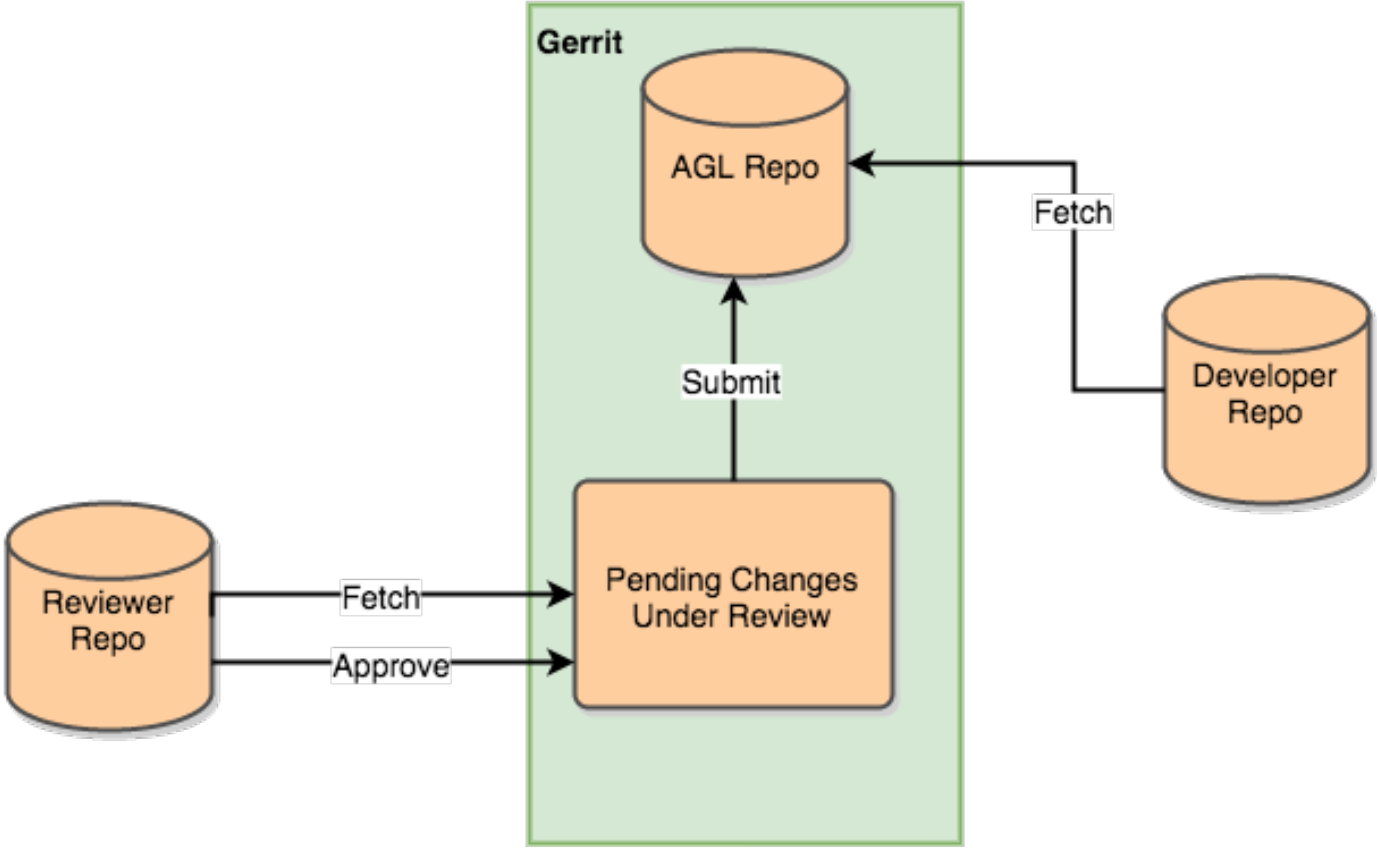
Gerrit Committer



Gerrit Merger



Gerrit Registered Users



Gerrit Code Reviews

- Reviewers complete code review with comments and assign a value.
  - -1: I would prefer that this is not merged as is.
  - 0: I am not making any statement about this change at the moment.
  - +1: Looks good to me, but someone else must approve it.
- Reviewers are assigned by the developer who submitted the branch.
- May be a Committer or a Registered User

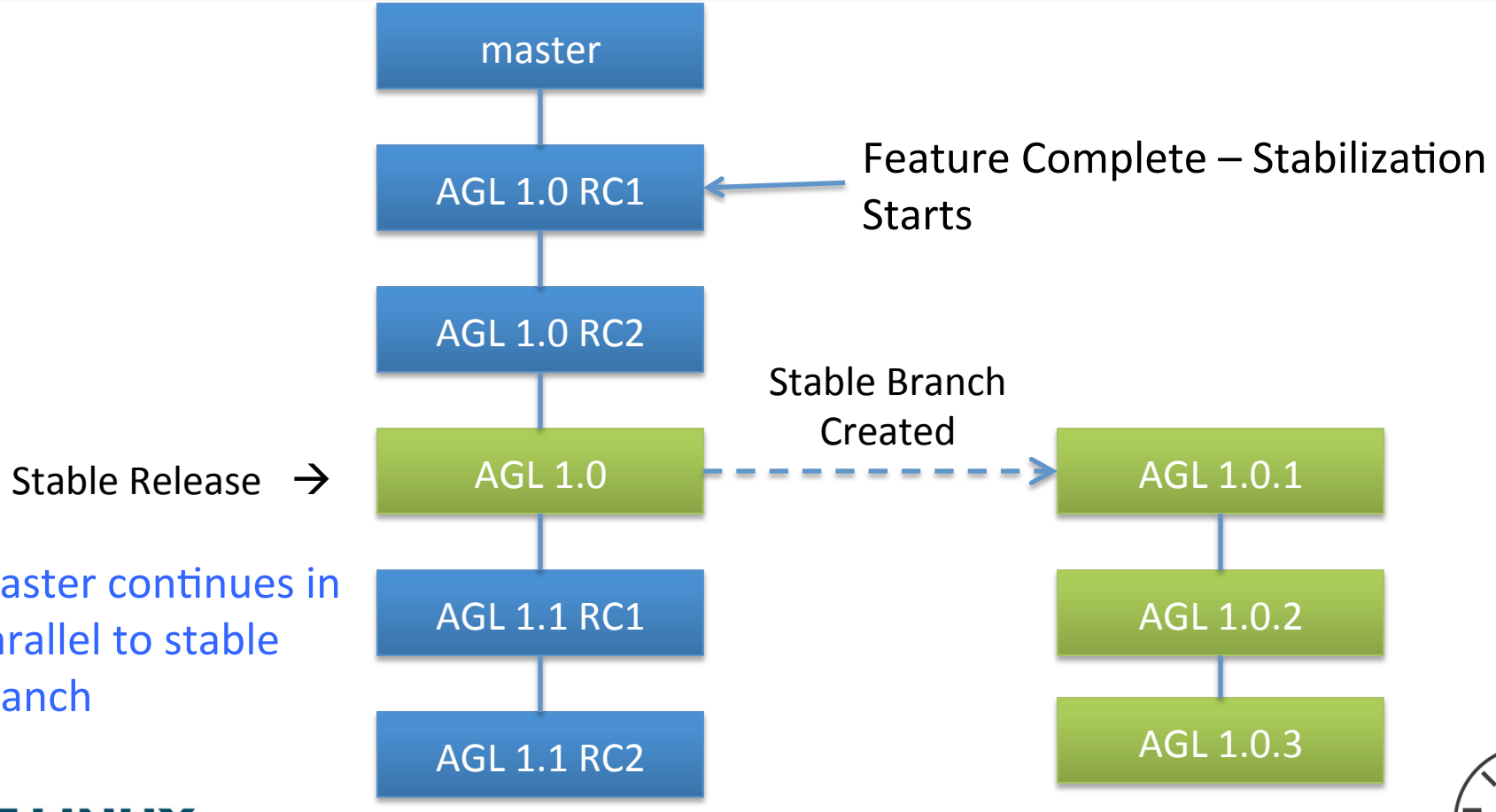
Gerrit Code Merges

- Code merges to the requested branch require an AGL Merger to assign a +2 to the change.
- AGL Mergers can reject the change (-2) or merge the change
- Two or more +1s do not allow the merger to be completed
  - -2: This must not be merged.
  - +2: Looks good to me, approved.
- Generally the AGL Mergers will wait for two +1s before assigning +2 and submitting the change

Branches

- Tentative decision to use Linux style branching where most work is on the mainline.
- Stable releases cause a branch to be created for further maintenance
- For repos where we are downstream, eg meta-renesas, we are working in branch, aim to offer those patches upstream.
- For repos where we are upstream, eg meta-agl, we are currently reviewing patches manually (+2 required) and committing to master

Branches and Tags



Jenkins

- Using Jenkins for Continuous Integration
- Successful build in Jenkins gives +1 to new code in Gerrit
- Tests will be added as part of the +1 criteria as we go forward
- Another session on Wednesday for CI and System Test

Automotive Grade Linux

Q&A

THANK YOU