TOOLS FOR

SMARTDEVICELINK

SCOTT SMEREKA

- Software Engineer & Web Lead at Livio
- Background
  - Used to work on Software for Tanks
  - Join Livio January 2012
  - Livio was acquired by Ford in late 2013
  - Working on SDL and internal research projects

TABLE OF CONTENTS

HOW WE HELP YOU DO WHAT YOU DO BEST — BETTER

SLACK

Open chat with the SDL community

ASK US ANYTHING

- Fastest way to find an answer to any problem, just ask!
  - But, please attempt a solution first
- Live chat with over 230 members of the SDL community on our public Slack Channel
  - Register at slack.smartdevicelink.com

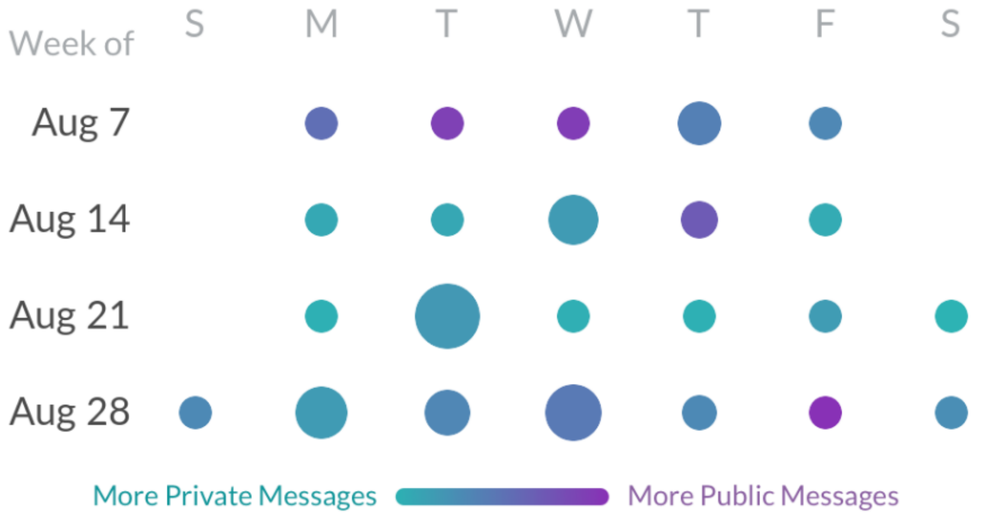
SLACK CHANNELS

- Channels – Separate conversations by topics
  - Quicker answers if using correct channel
  - Frequently Asked Questions
    - Search
    - Slackbot

8/28 - 9/4 SUMMARY

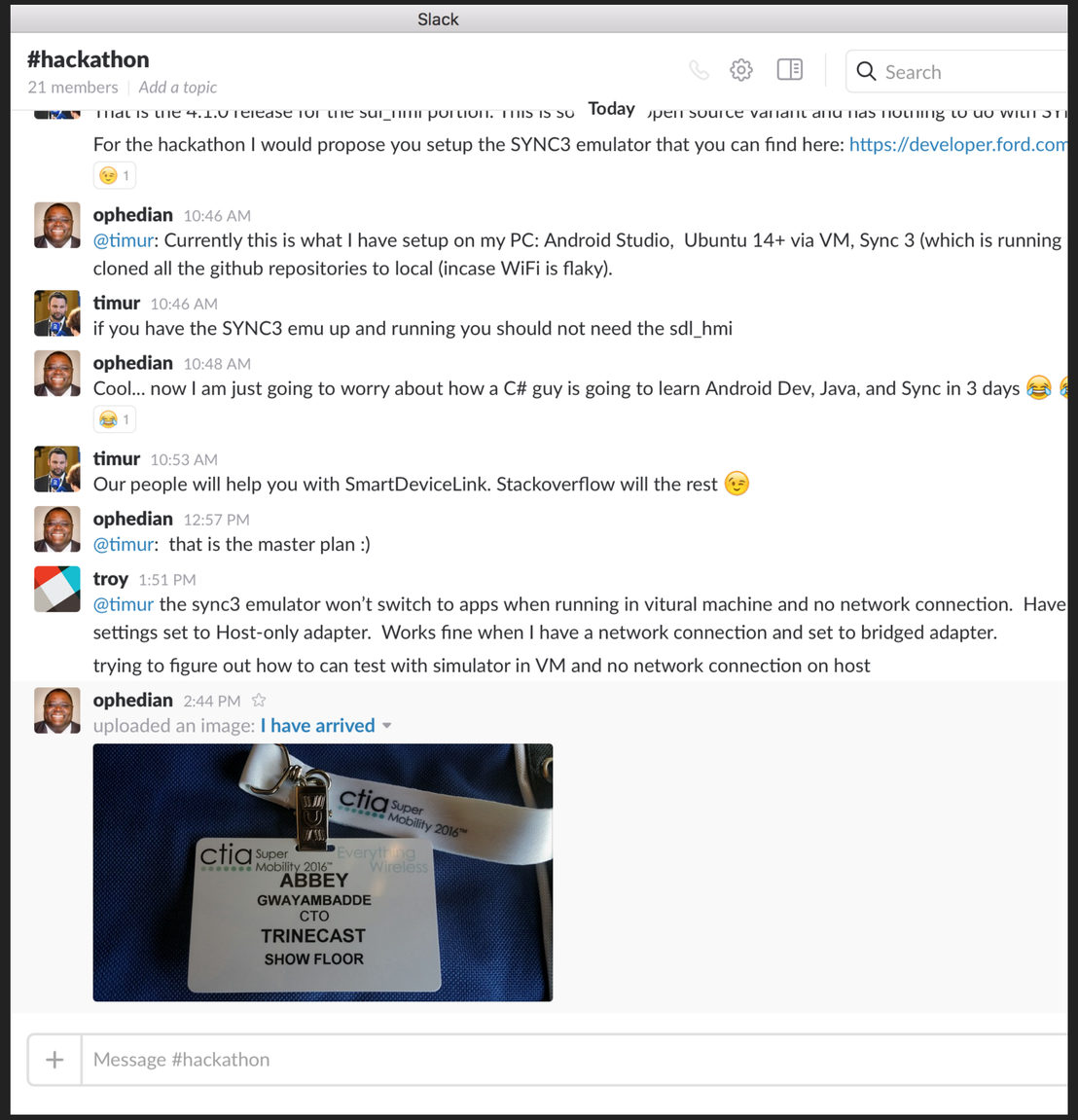
- Developers are actually using slack
- 43% of traffic in public chats
- People tend to talk directly to whoever is helping them in a direct message

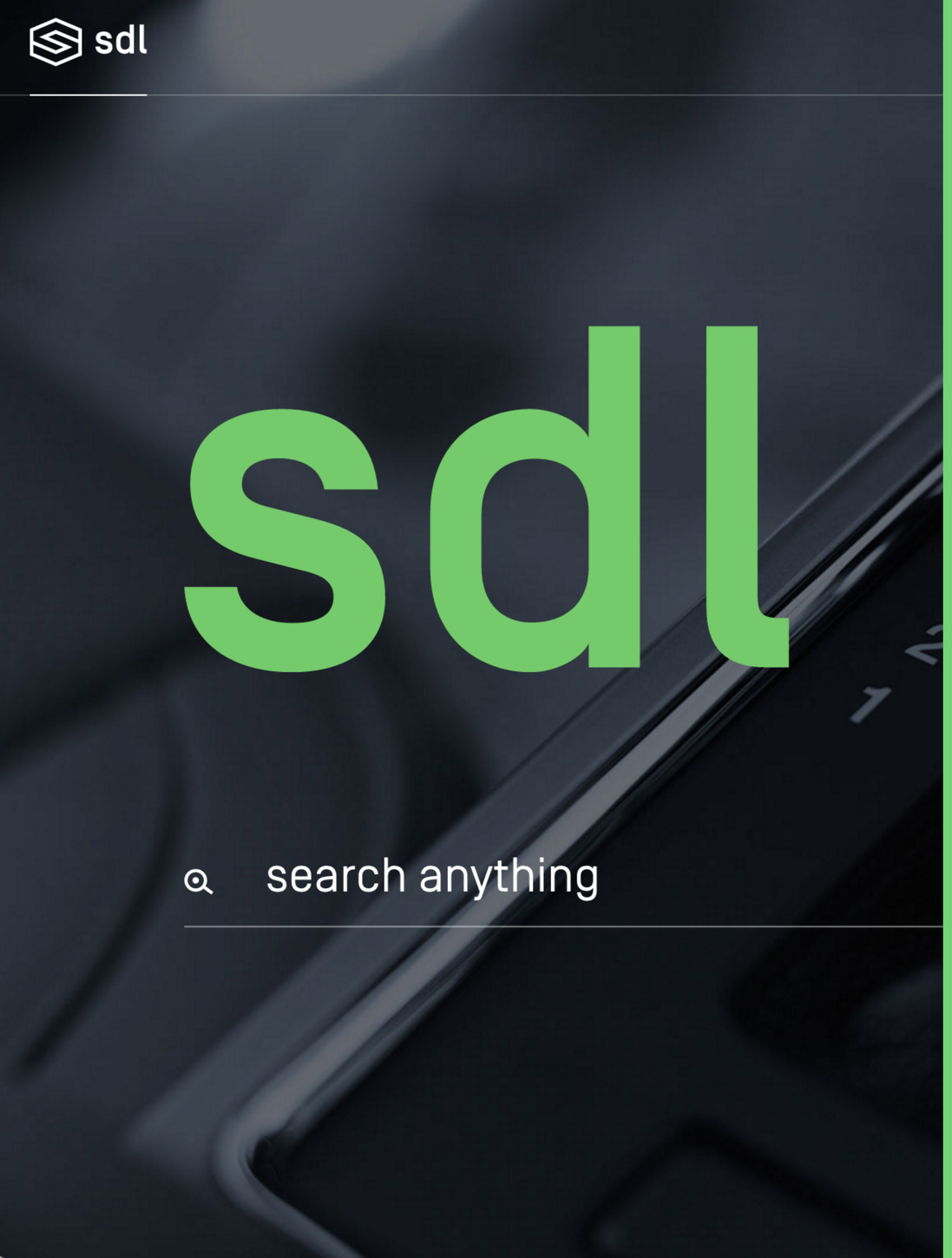
Your team sent a total of **997 messages** last week (that's 461 more than the week before). Of those, **43% were in public channels**, **2% were in private channels** and **55% were direct messages**. Your team also uploaded **12 files** (that's 3 more than the week before).



HACKATHON CHANNEL

- A slack channel is available for tonights #hackathon





DEVELOPER PORTAL

Learn about smartdevicelink.com

OVERVIEW

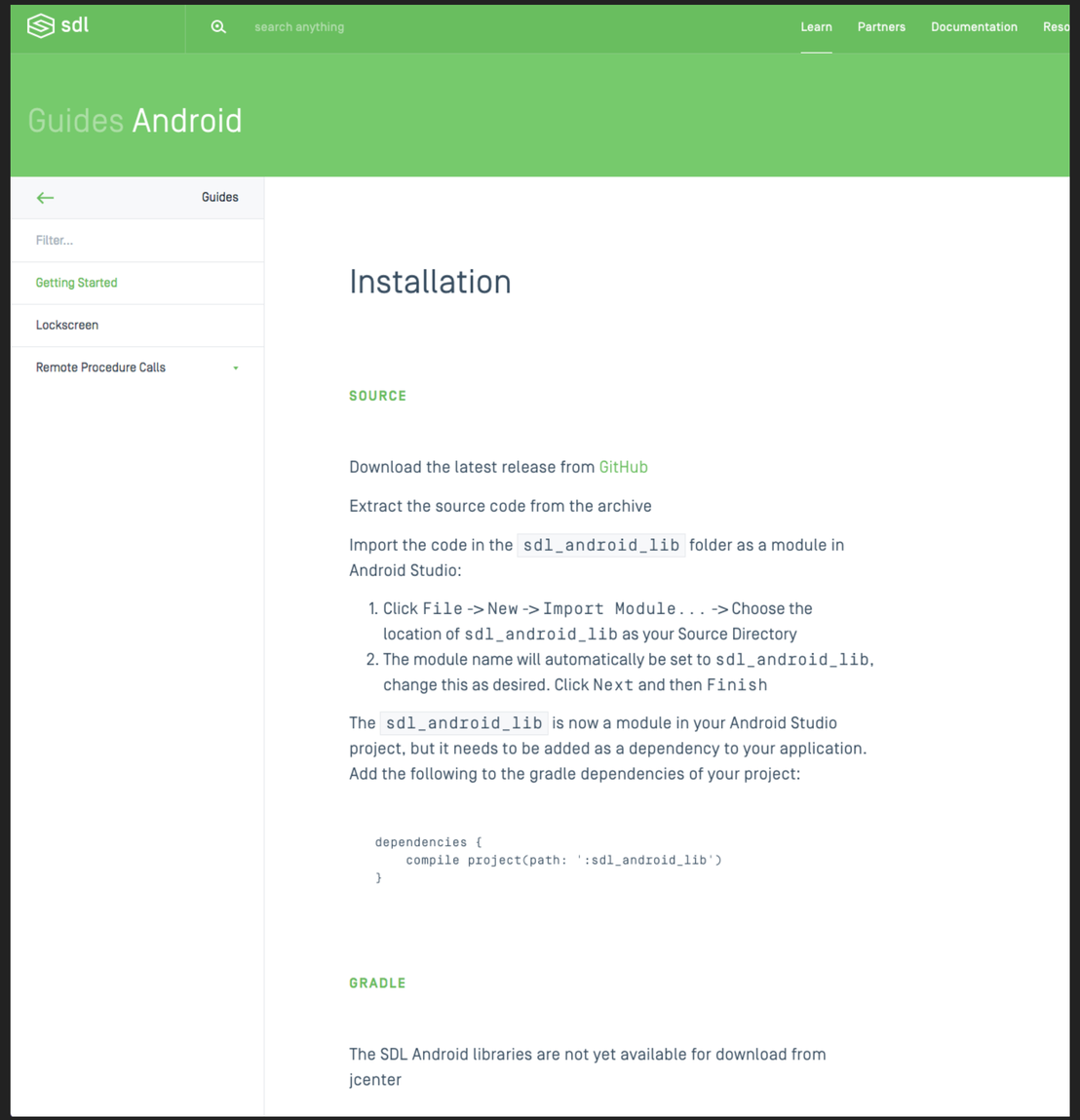
- The first place you should go to find anything related to SDL is SmartDeviceLink.com
- The main goals are to help
  - Developers get their apps into vehicles
  - OEMs implement SDL in their vehicles

WHATS IN THE PORTAL?

- Guides
- Documentation
- Developer Console – New!
- Partner Information – New!
- Downloads / Links to Resources
- Latest News

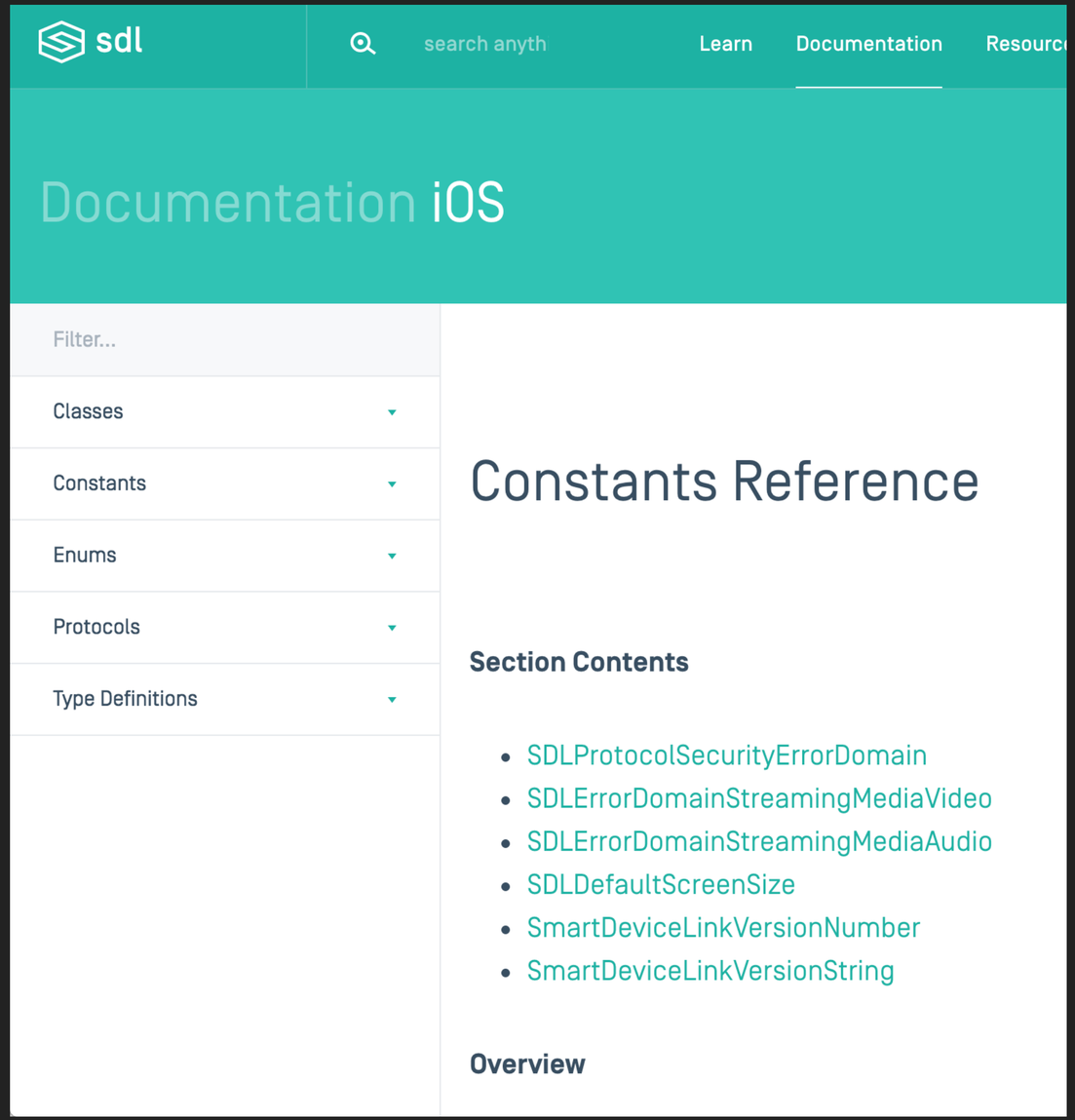
GUIDES

- Guides - Explain how to do common or more complicated tasks
  - e.g. Getting Started, Configuring Core, Video Streaming
- Created based on need of the developers
- Located under Learn on smartdevicelink.com



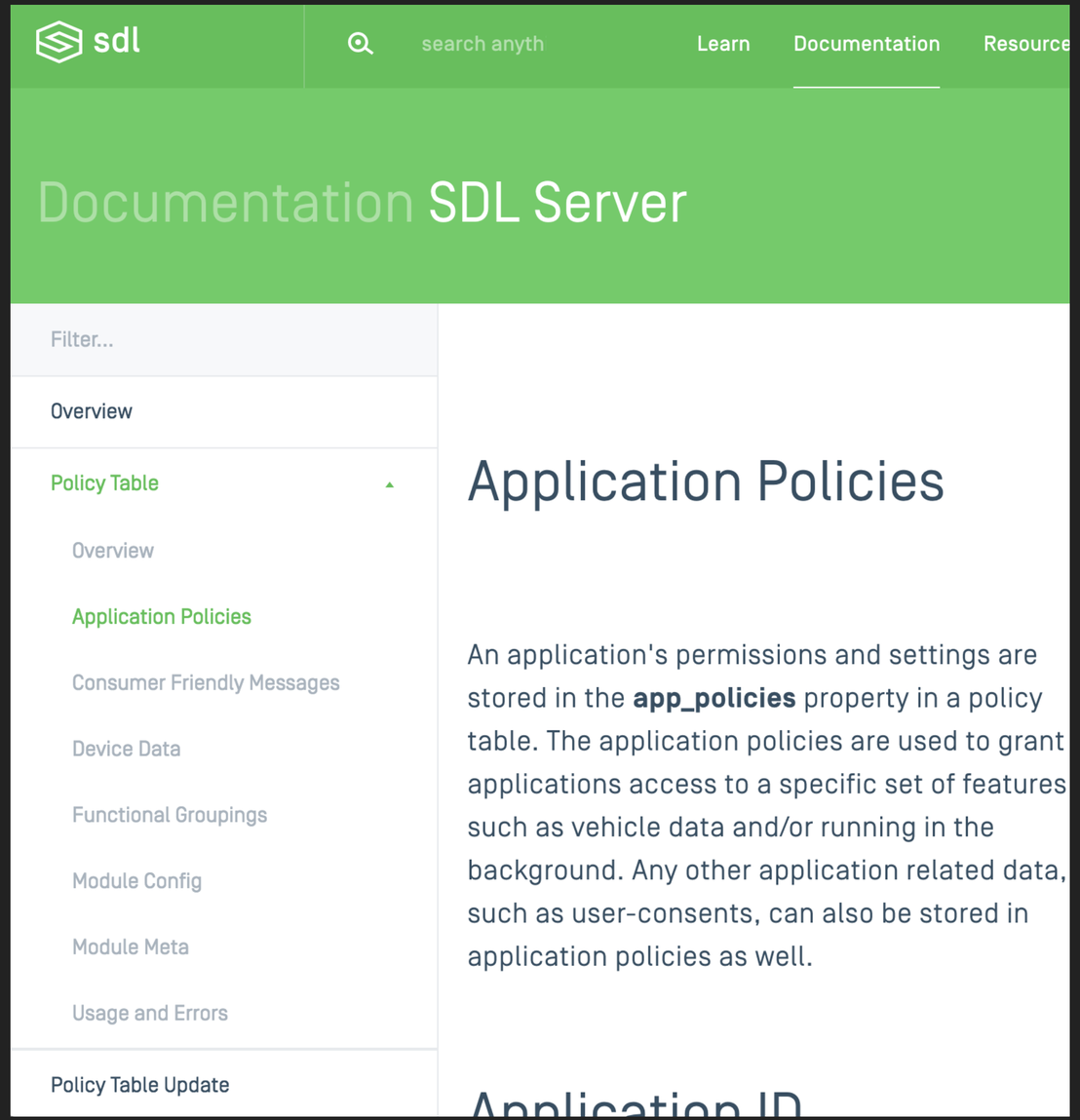
DOCUMENTATION

- Android and iOS
  - Comprehensive documentation for the mobile libraries
  - e.g Classes, Constants, Enums, Protocols, Type Definitions, etc
- HMI
  - Describes how to integrate an OEM's user interface with SDL Core
  - e.g WebSocket Transport, Buttons, Navigation, TTS, VR, etc



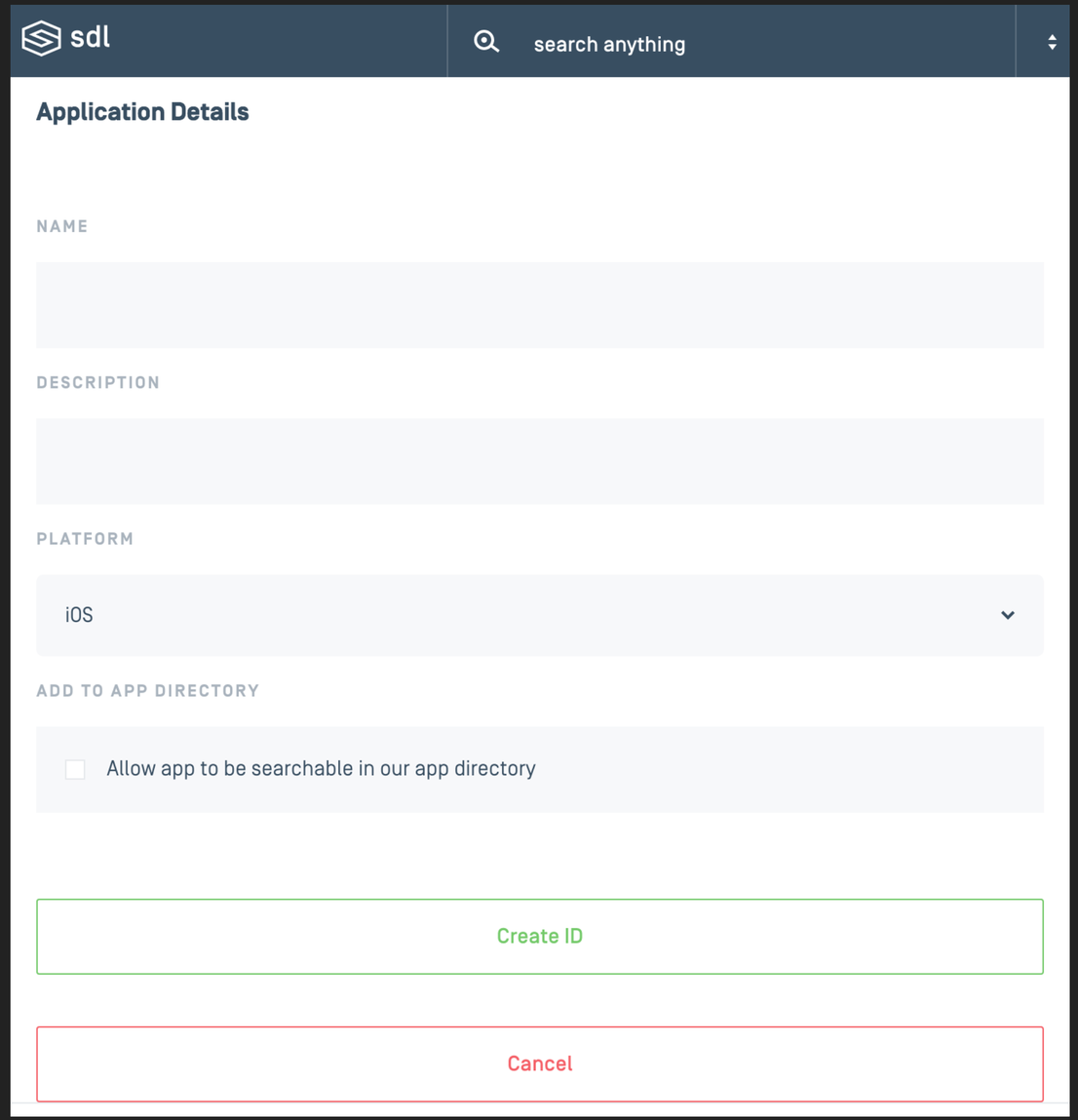
DOCUMENTATION

- SDL Server
  - In depth look at policy tables and how to update policies in a vehicle
  - e.g Policy Table Attributes and Policy Table Updates
- SHAID
  - How to integrate with the software tool used to synchronize application and user data across the OEMs
  - e.g API documentation, Authentication, and Examples



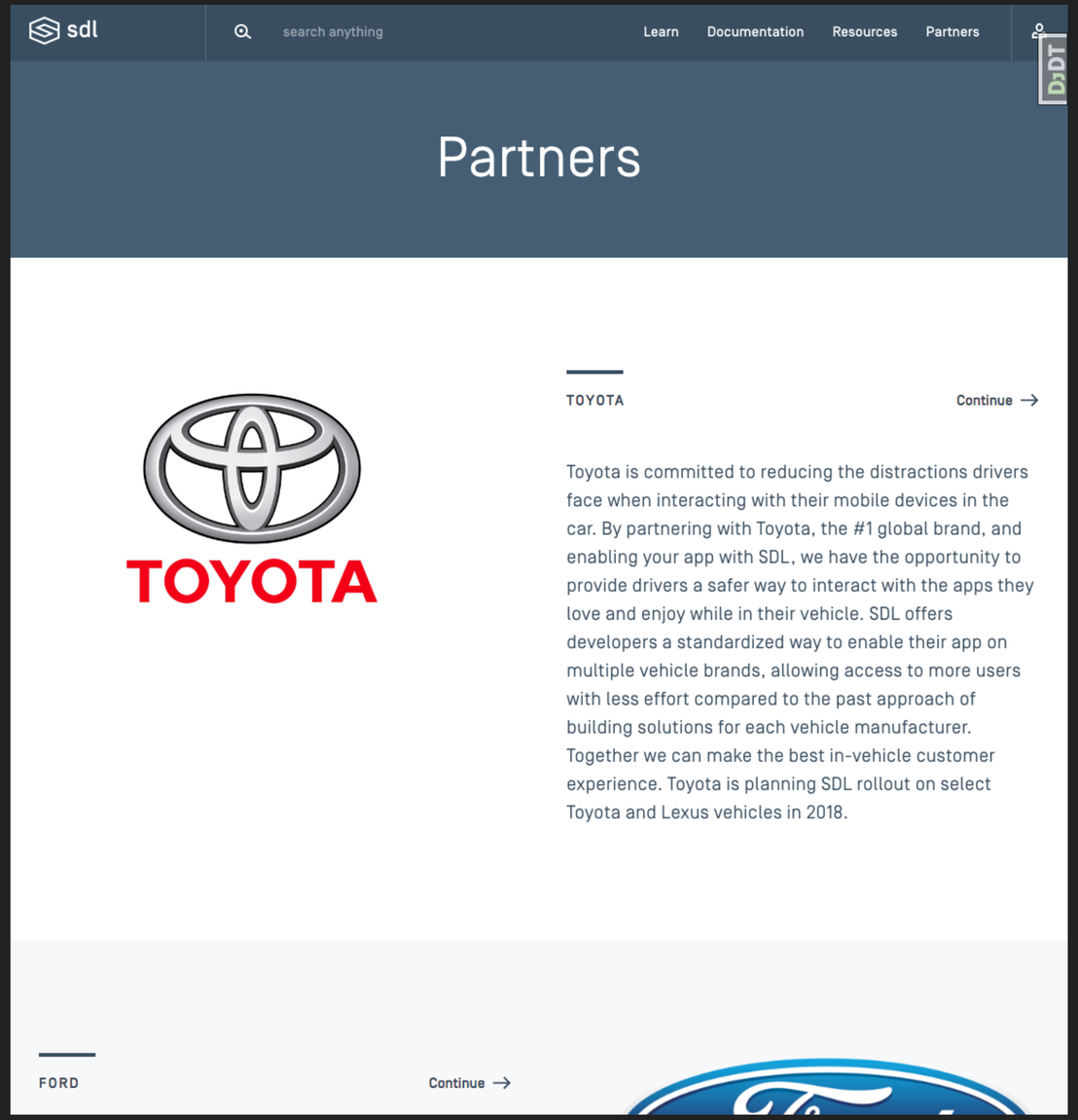
DEVELOPER CONSOLE (NEW)

- Application IDs - Generate IDs that are globally unique across the entire SDL ecosystem
- Profile Settings - Configure your personal developer portal experience. For example
  - Default Code block themes
  - Default Language
  - Email address and user information



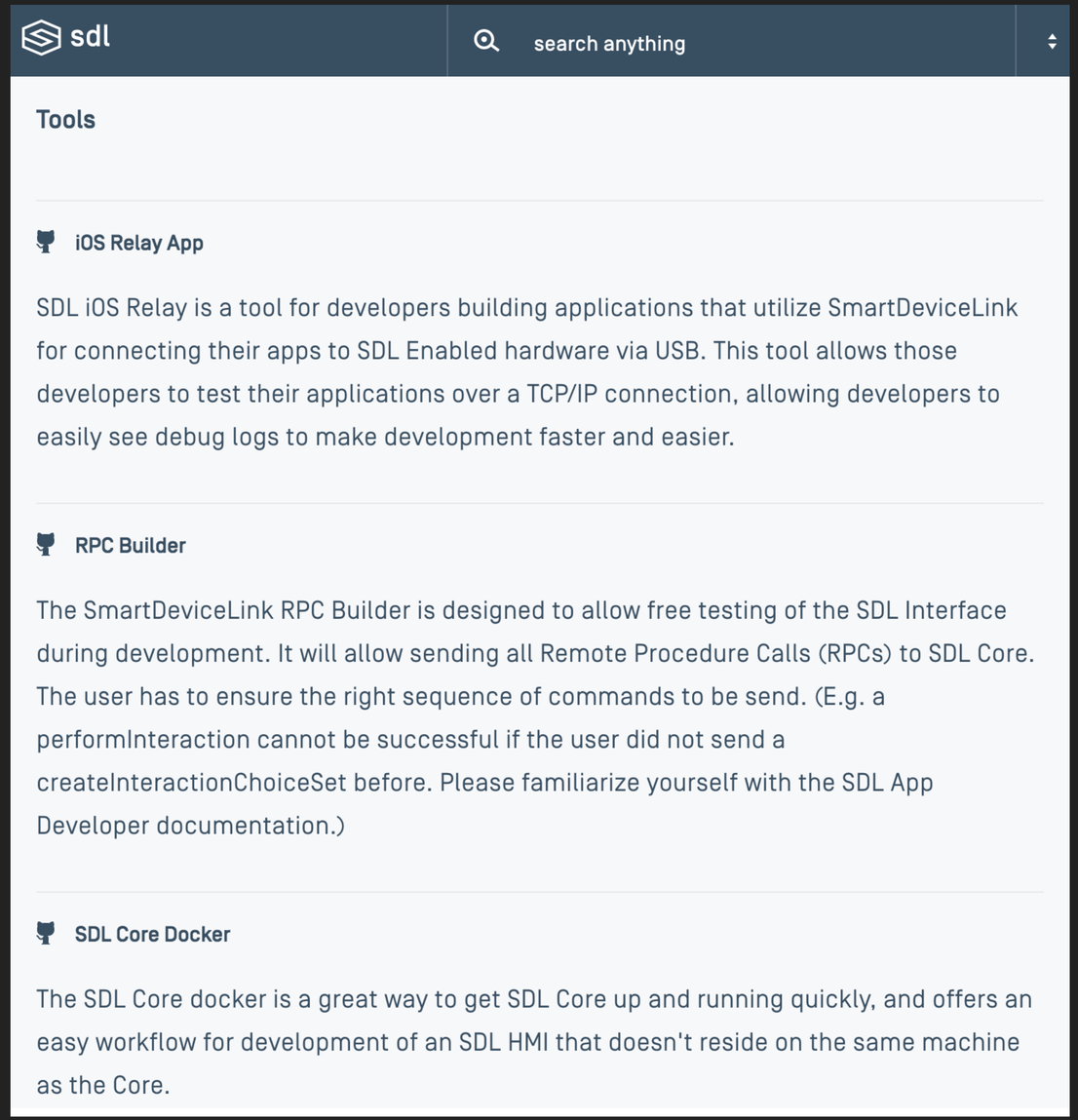
PARTNERS (NEW)

- Each OEM can create and manage their own page
  - Can see what vehicle's SDL is in
  - Helps developers contact the OEM
  - Can use our custom content management system
- Want an OEM page?
  - Email us at support@smartdevicelink.com



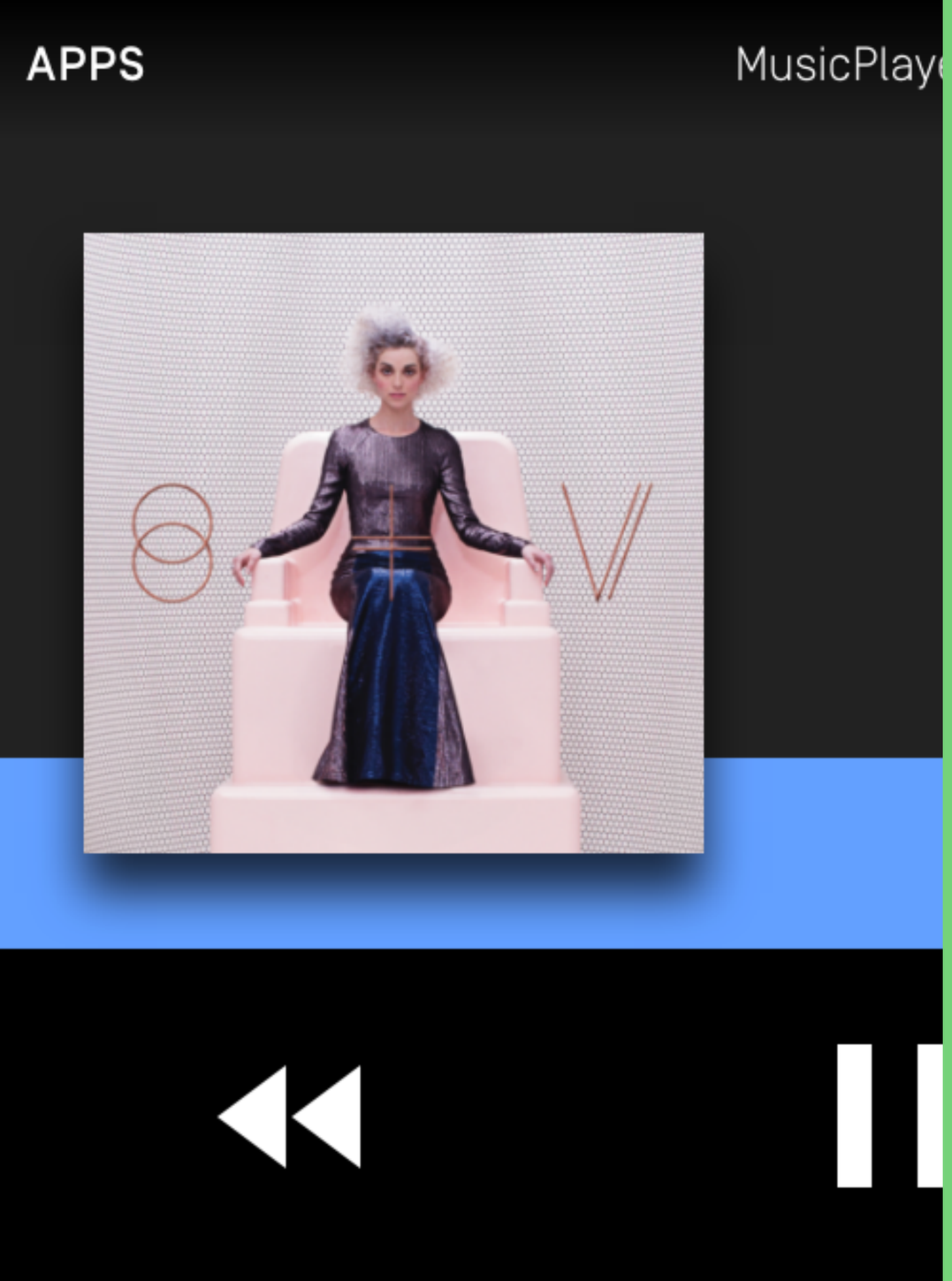
RESOURCES

- Downloads or Links to
  - Example applications and sample code
  - Android and iOS libraries
  - SDL Core docker container
  - Software tools
  - SDL logos and marketing materials



SEARCH

- Easiest way to find anything is to search for it
- Searching is tracked
  - We update documentation to satisfy common search terms

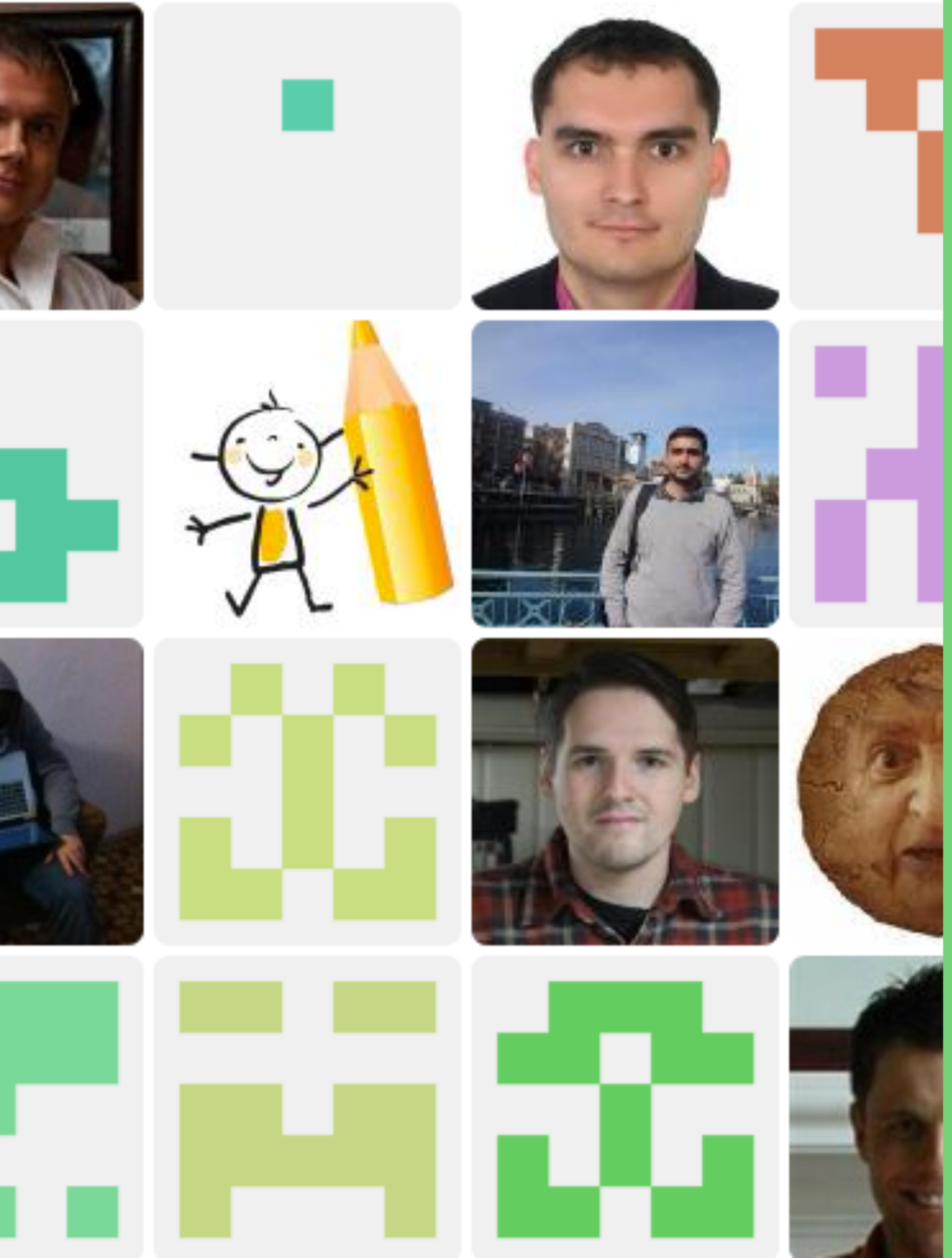


GENERIC HMI

Example SDL user interface

OVERVIEW

- The Generic HMI will become the default user interface for SDL core
  - See Joey's presentation
- The HMI is entirely web based (Javascript & React)
  - Easier for developers to test with
  - Important for other tools like Manticore



GITHUB

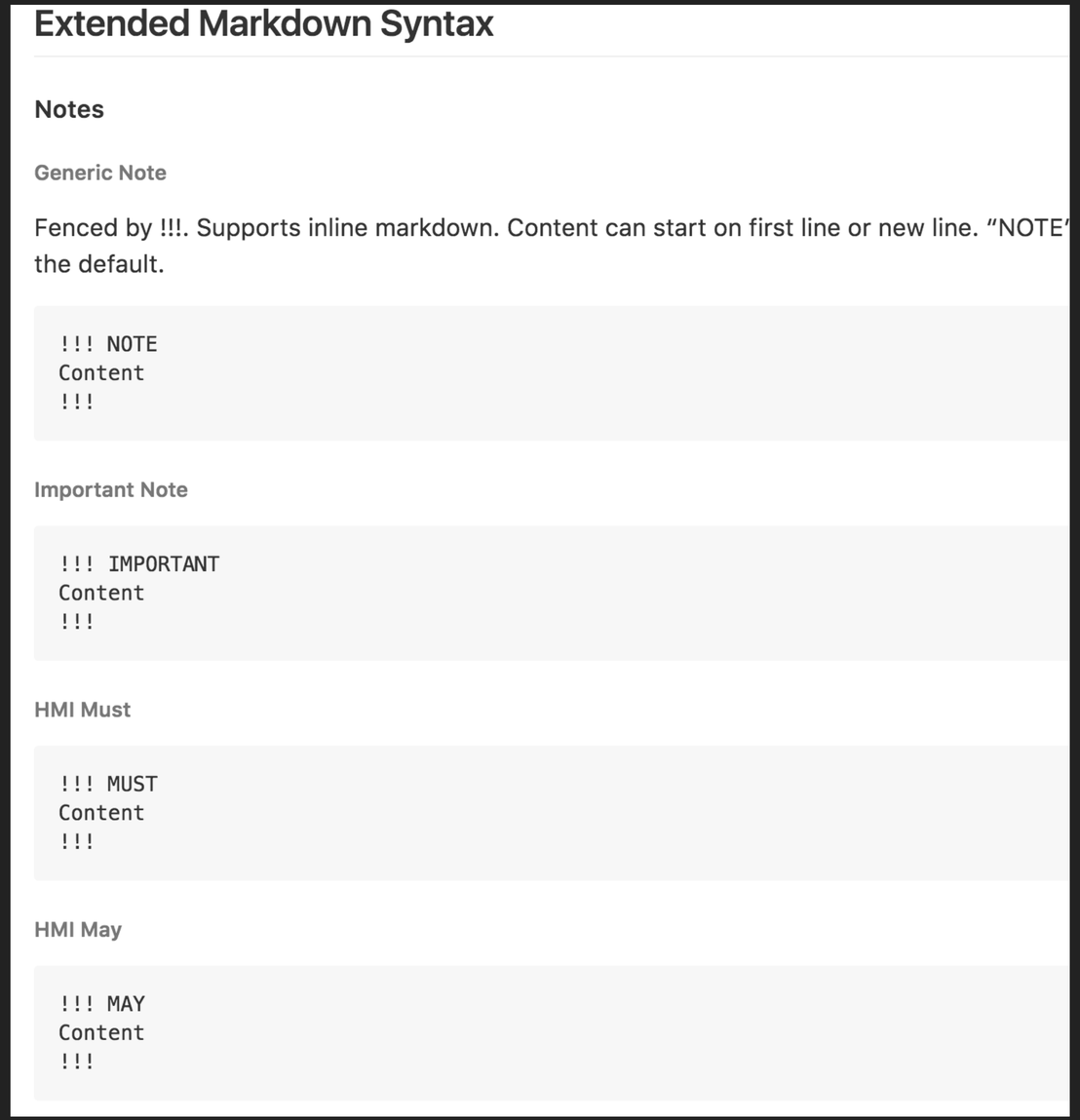
a.k.a. the code mines

OVERVIEW

- All code and documentation is managed on github
- All documentation is written in DocDown
- We follow Gitflow to manage each repository
- Contributions accepted using pull requests
  - New features for Android, iOS, or Core should follow the new evolution process

DOCDOWN

- DocDown is an extension of MarkDown to increase the readability of documentation
  - Custom section headers
  - Configurable structure for multiple pages
  - Relative linking of content
    - Local assets
    - Pages or sections



Extended Markdown Syntax

Notes

Generic Note

Fenced by !!!. Supports inline markdown. Content can start on first line or new line. “NOTE” the default.

```
!!! NOTE
Content
!!!
```

Important Note

```
!!! IMPORTANT
Content
!!!
```

HMI Must

```
!!! MUST
Content
!!!
```

HMI May

```
!!! MAY
Content
!!!
```

CUSTOM HEADER EXAMPLE

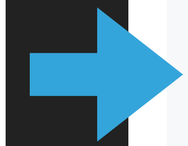
SDL sends `AlertManeuver` together with TTS.Speak. The purpose of this RPC is to notify the embedded navigation system about the next navigation maneuver.

!!! MAY

1. Notify the user by TTS.Speak, which is sent along with `AlertManeuver`.
2. Display the `AlertManeuver` dialog with an alert icon and one of the following:
   - Up to three soft buttons defined within the `softButtons` paremeter.
   - HMI-defined `Close` soft button if a request without `softButtons` is received.
3. Process `SystemContext` behavior for `AlertManeuver` in the same way an `Alert` is handled.

!!!

!!! MUST



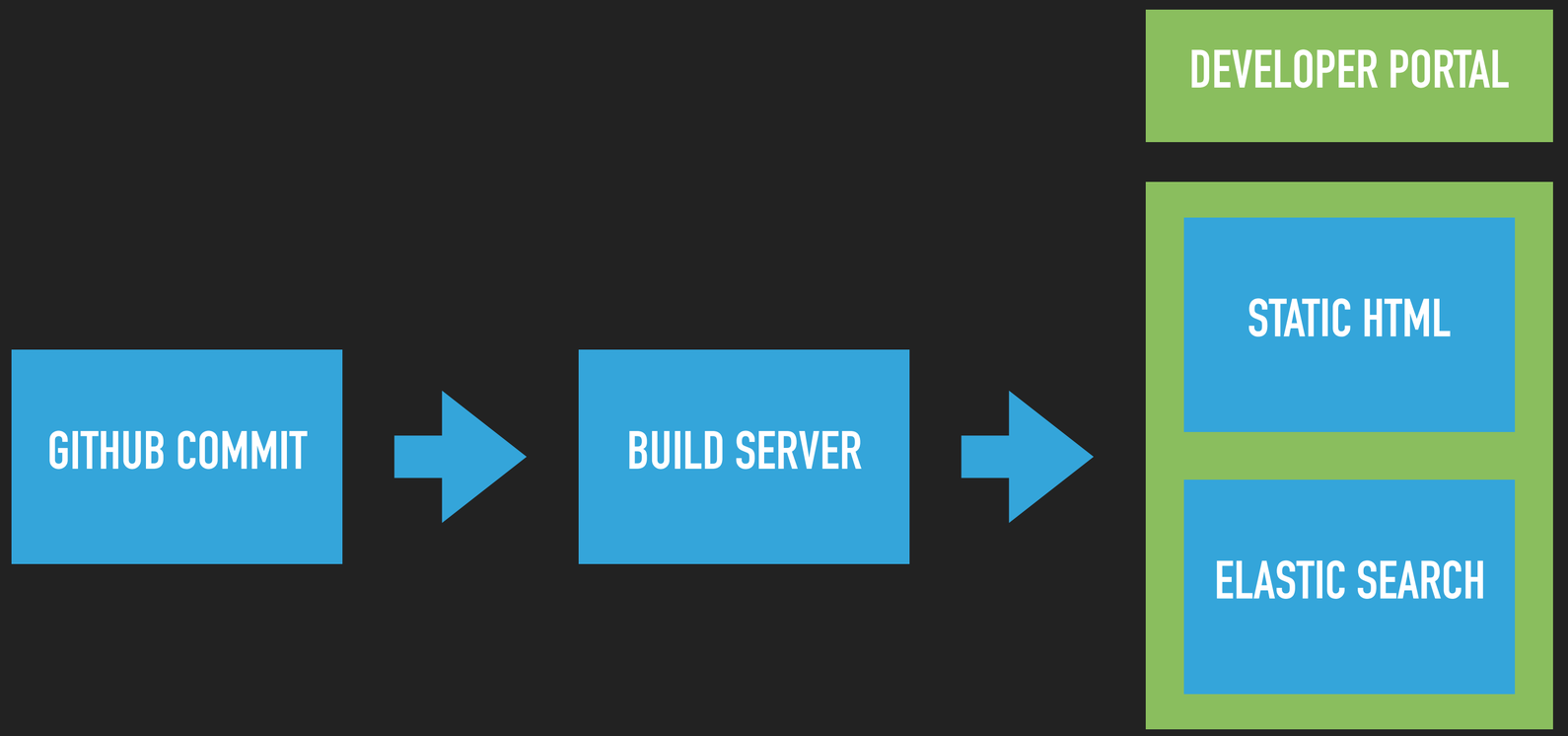
SDL sends `AlertManeuver` together with TTS.Speak. The purpose of this RPC is to notify the embedded navigation system about the next navigation maneuver.

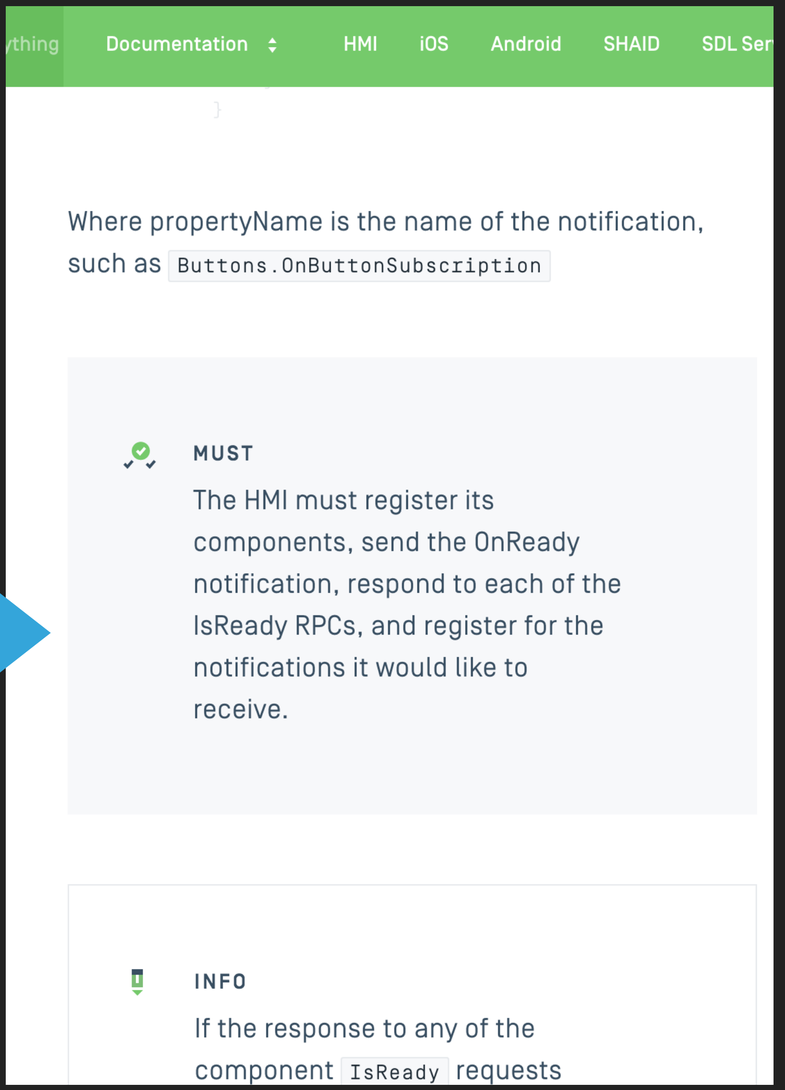
MAY

1. Notify the user by TTS.Speak, which is sent along with `AlertManeuver`.
2. Display the `AlertManeuver` dialog with an alert icon and one of the following:
   - Up to three soft buttons defined within the `softButtons` paremeter.
   - HMI-defined `Close` soft button if a request without `softButtons` is received.
3. Process `SystemContext` behavior for `AlertManeuver` in the same way an `Alert` is handled.

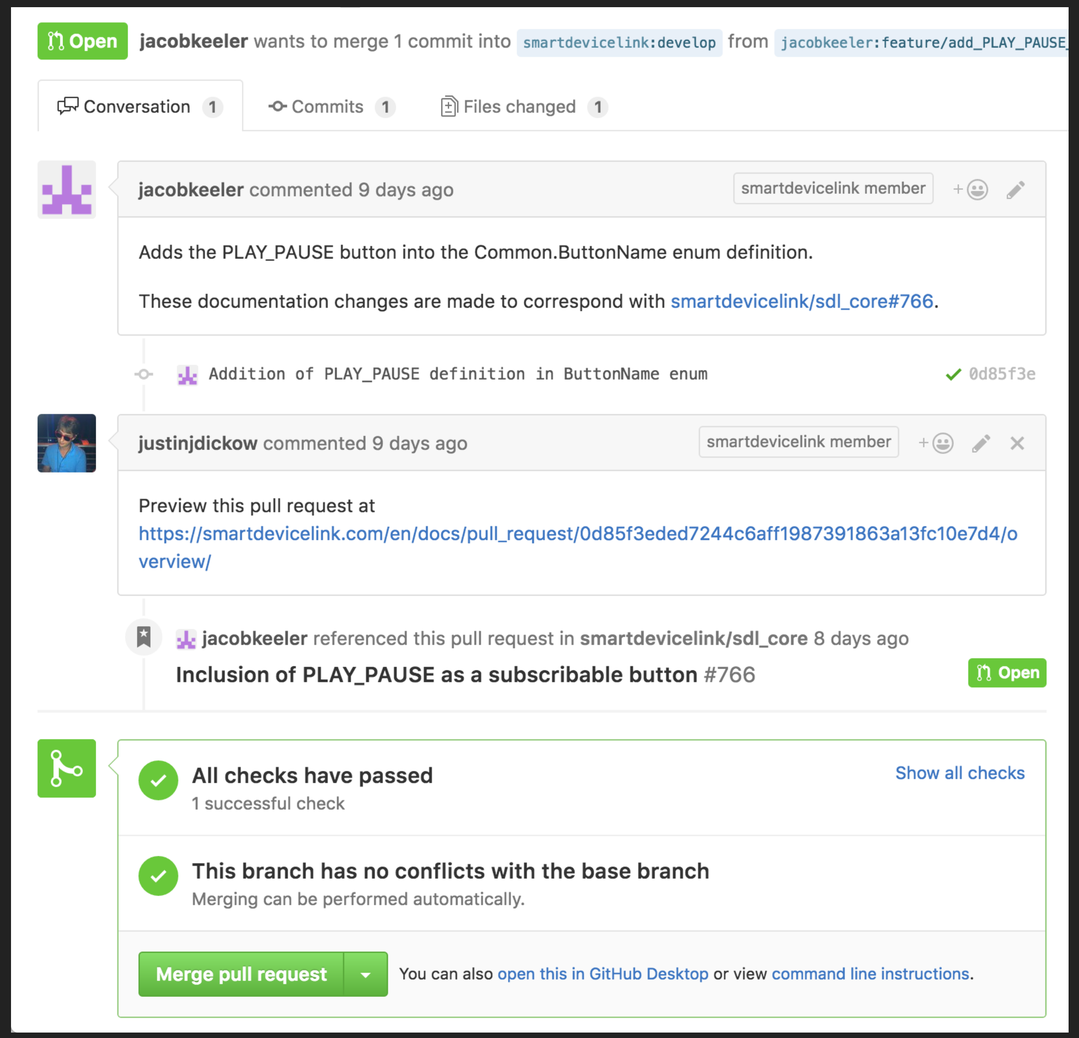
MUST

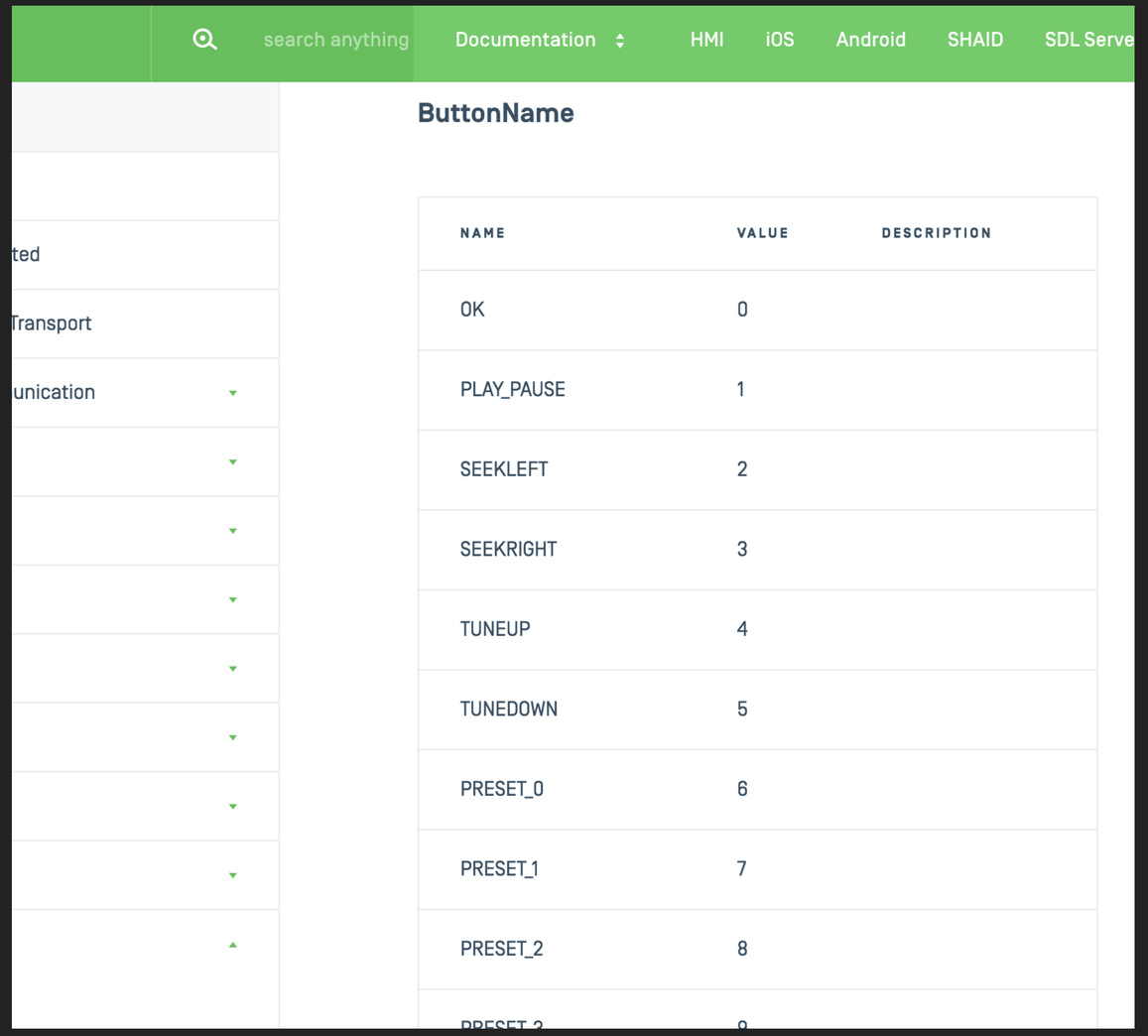
BUILD SERVER





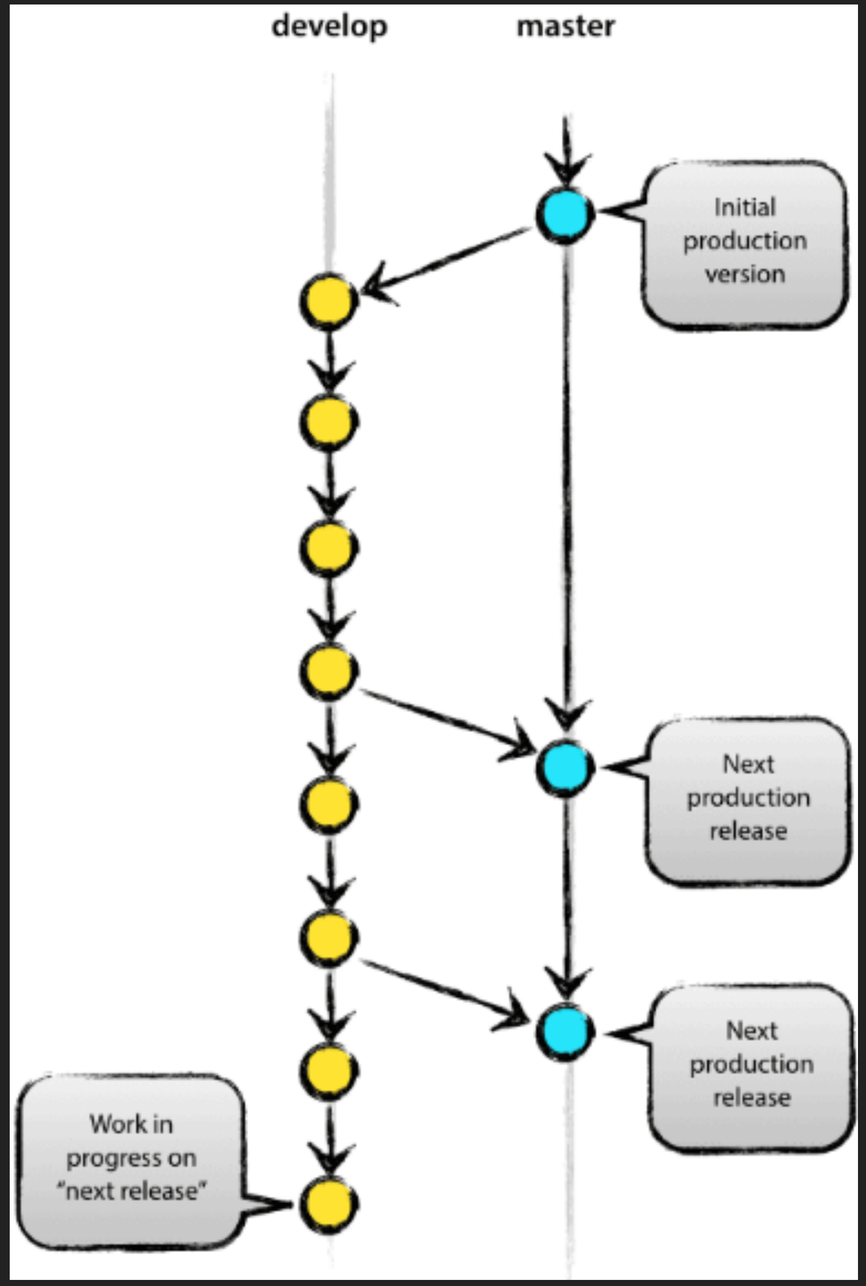
DOCDOWN PULL REQUEST





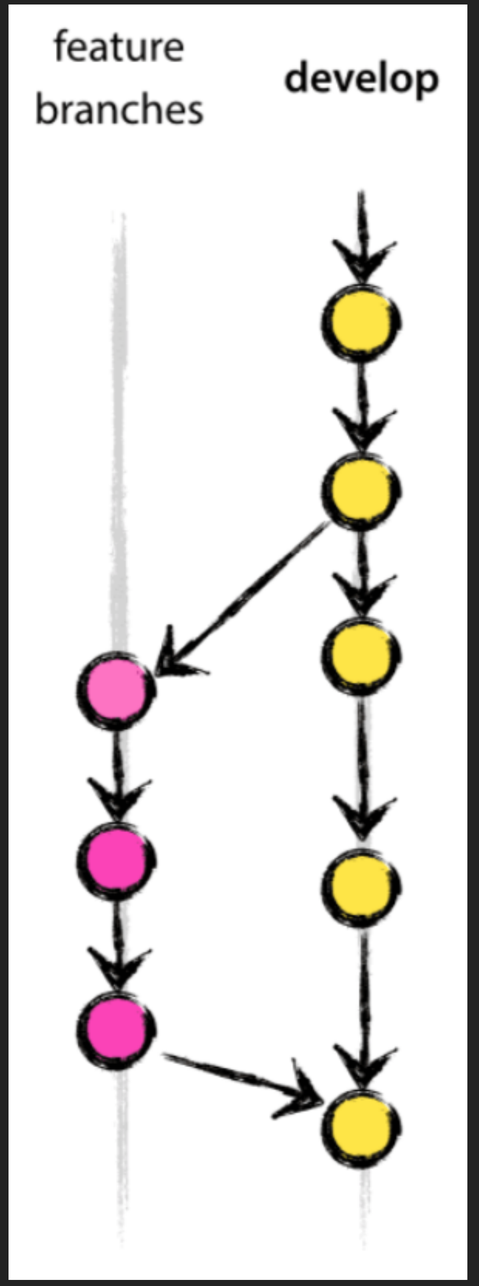
GITFLOW

- A popular development model for managing a project using git
  - Takes advantage of branching and merging
- Two main branches in a repository always exist:
  - Master - Production ready releases
  - Develop - Latest features that are complete and unreleased
- Other temporary branches may exist:
  - Feature, Release, and Hotfix



GITFLOW

- Feature Branch
  - Used to develop new features for future release
  - Branches from and to develop
  - Should only exist while feature is in development

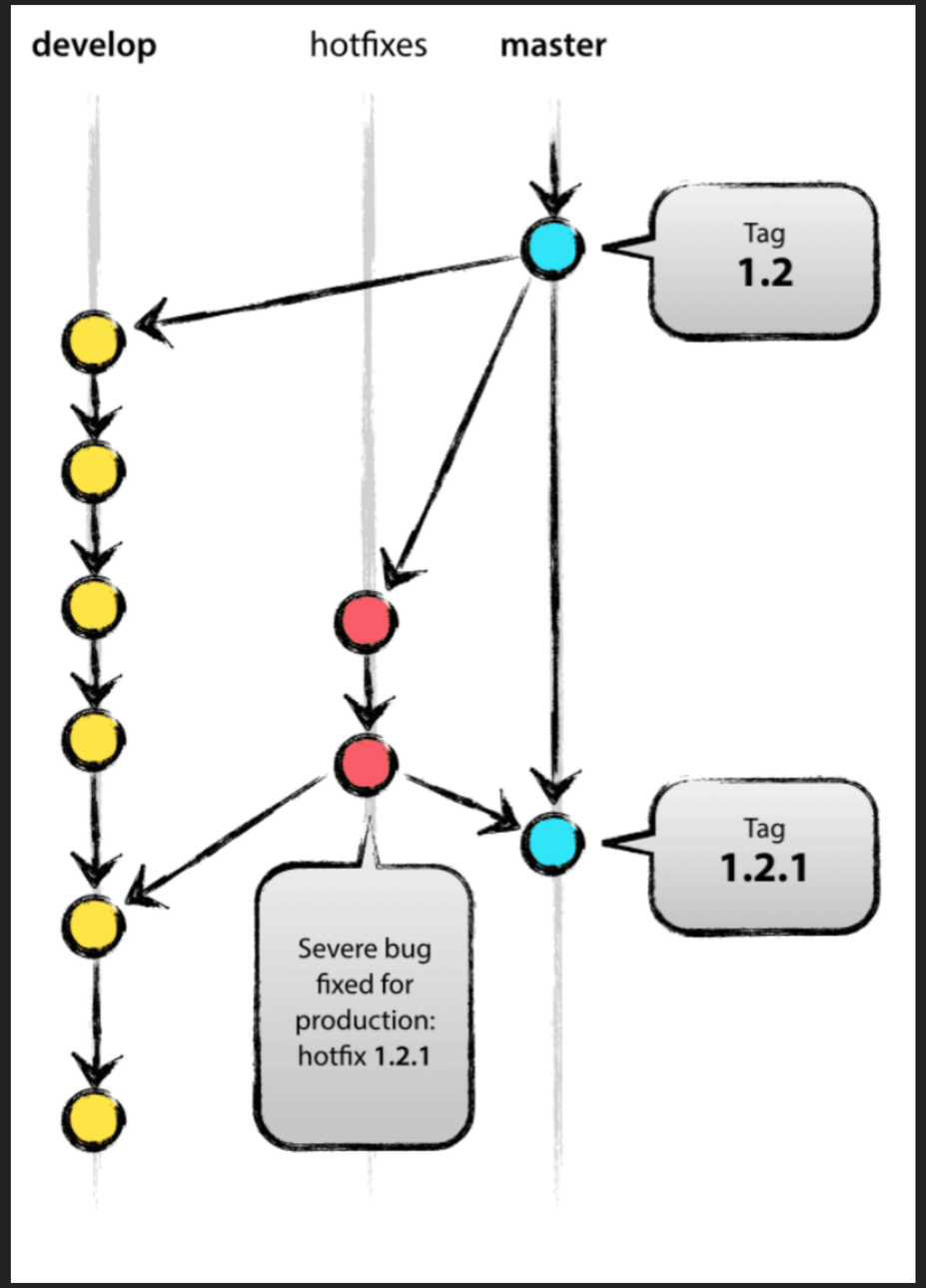


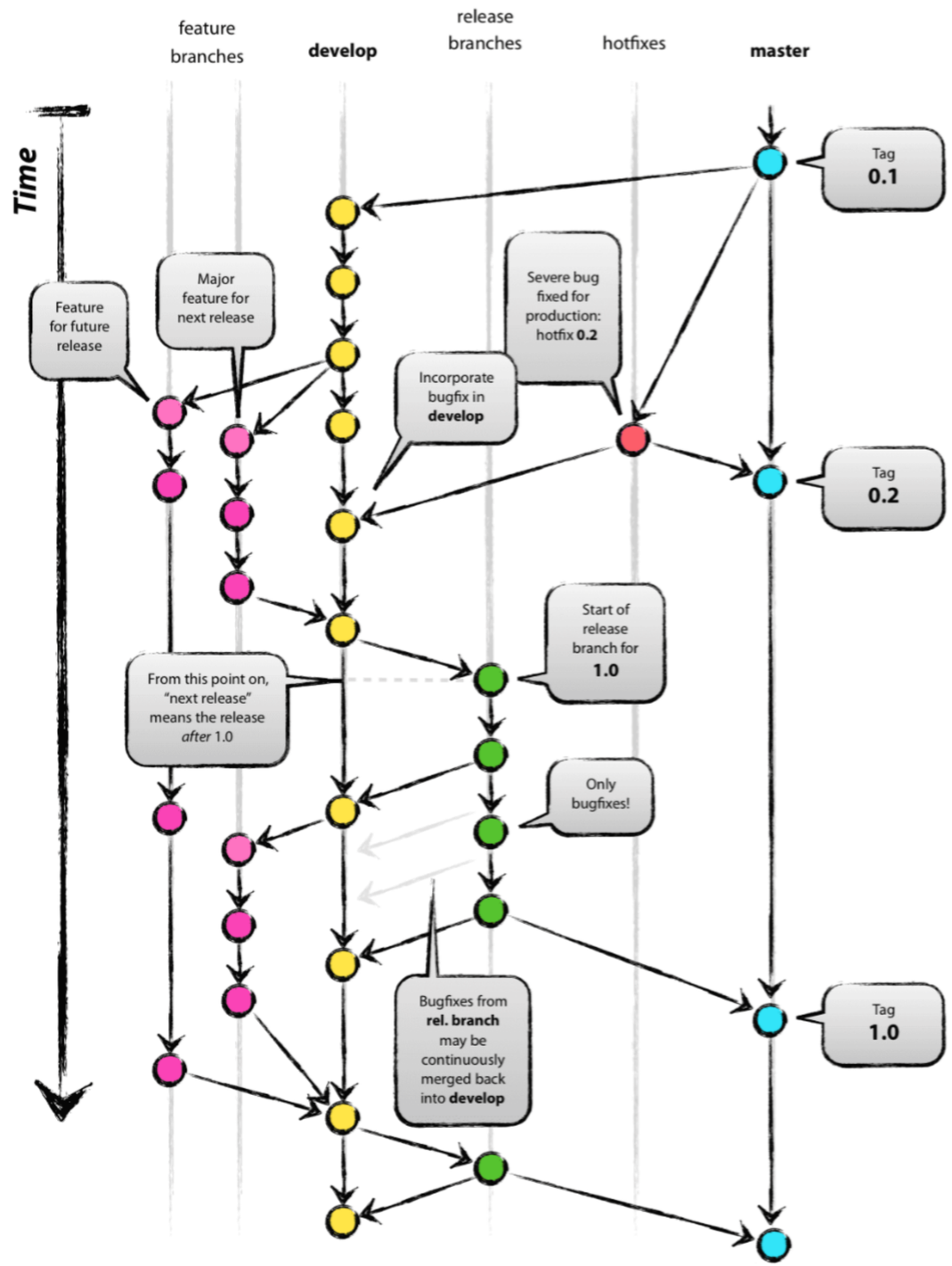
GITFLOW

- Release Branch
  - Created to prepare a new production release
  - Allows last minute dotting of i’s and crossing t’s
  - Branches from develop
  - Merges into master and develop

GITFLOW

- Hotfix Branch
  - Typically created to address an issue in production release
  - Branches from master
  - Merged into master and develop





HOW TO CONTRIBUTE

Using github, gitflow, and the evolution process.

STEP 0

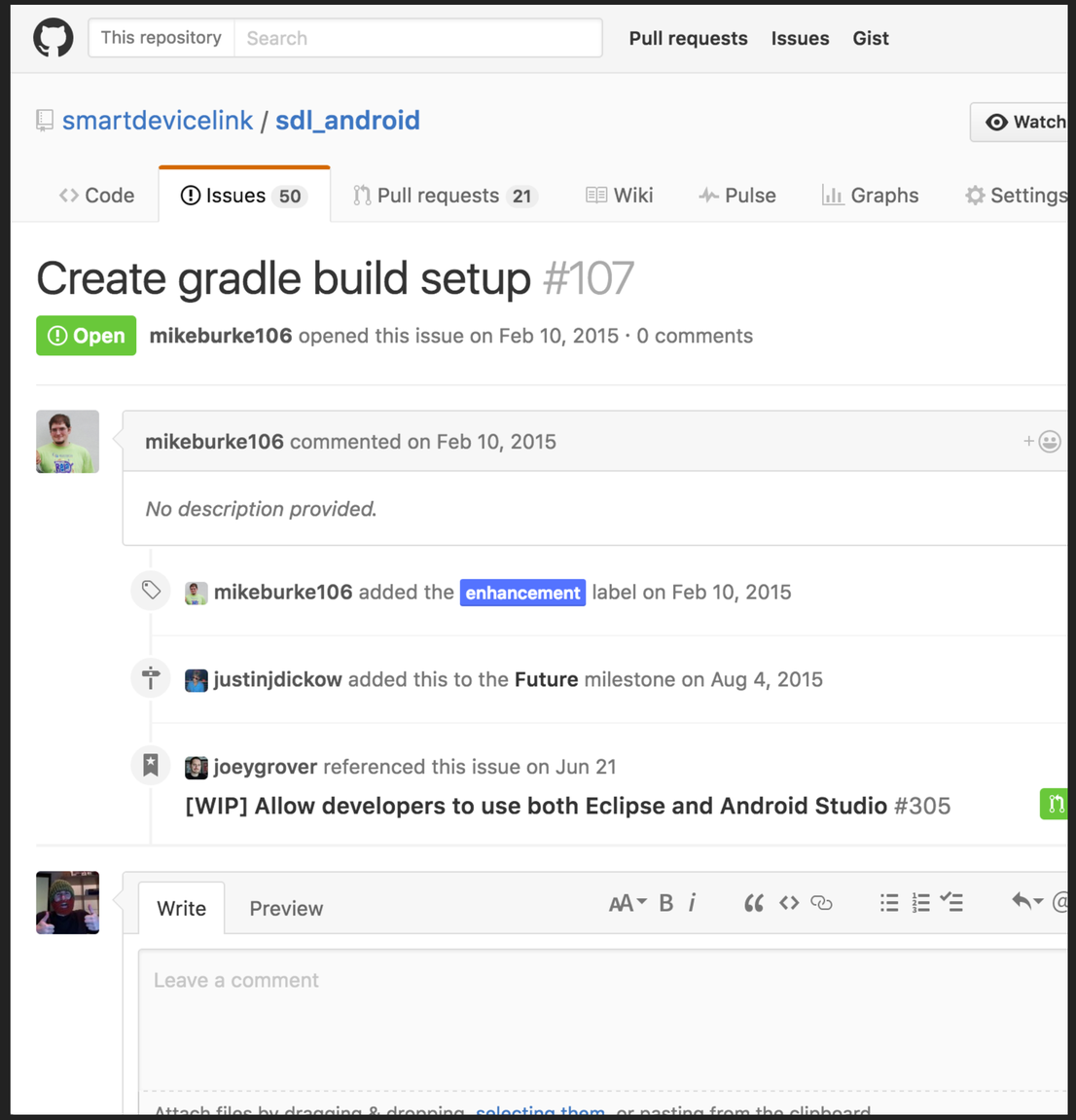
- Sign the Contributor's License Agreement which basically says (IANAL):
  - We have a right to use your contributed code, even if you patent it
  - You can't revoke anyones right to use your contributed code
  - We can't sue you if your contributed code is bad

STEP 1

- If contributing a new feature for Android, iOS, or Core you must follow the evolution process. Otherwise skip this step
  - Submit your proposal to the sdl_evolution repository in a pull request using the provided proposal template
  - The proposal will be scheduled for review where the SDL community can discuss the feature
  - The SDL committee will review the feedback and either approve or deny the new feature

STEP 2

- Go to the project's github repository and search to see if the issue already exists
- If it doesn't create one in the issue tracker

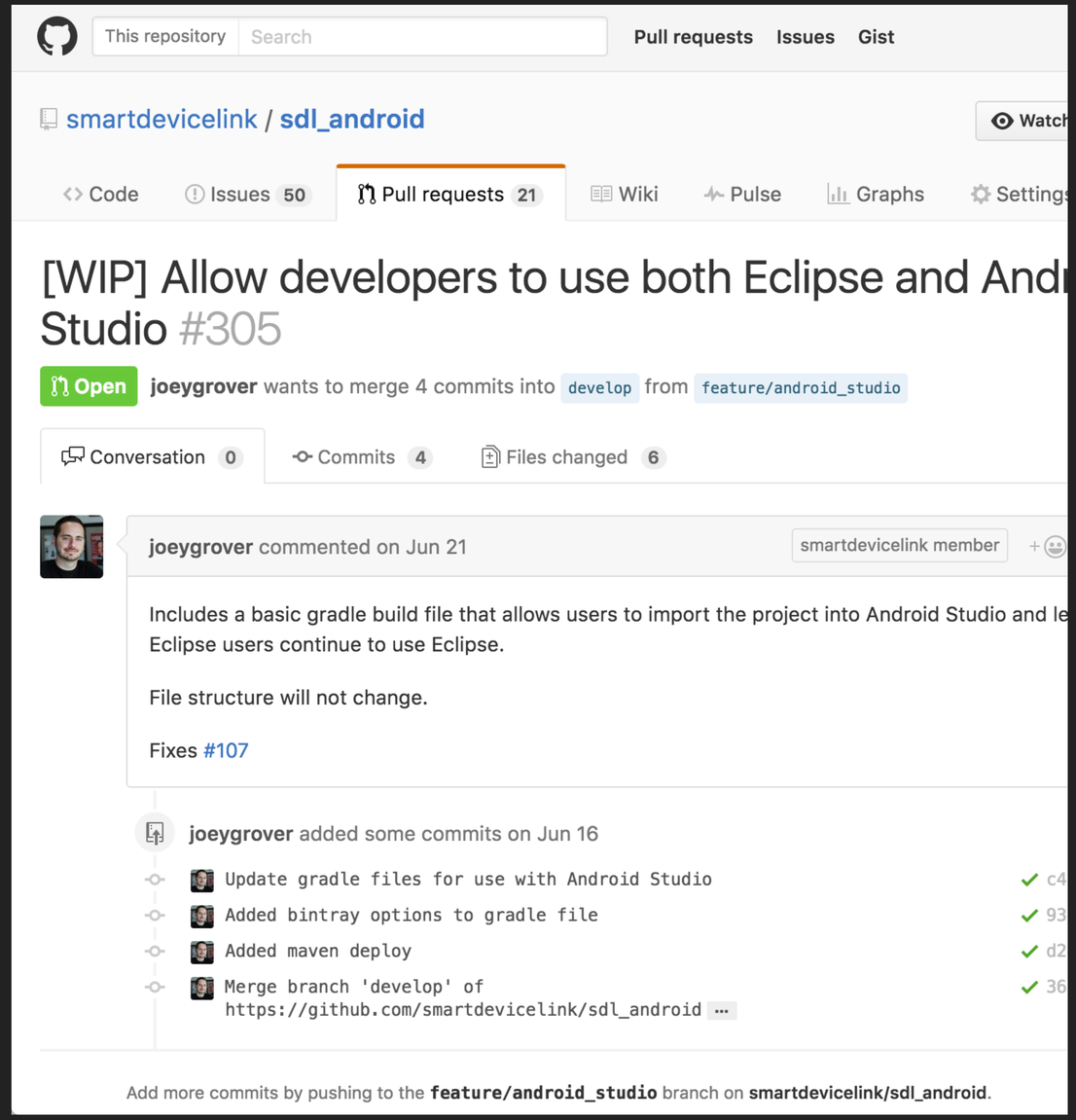


STEP 3

- Create a feature or hotfix branch in the desired repository
  - Features should branch from the develop branch
  - Named issue_<Issue #>_<Feature Name>
    - git checkout -b feature/issue_107_androidStudio develop
  - Hotfixes should branch from the master branch
  - Named issues_<Issue #>
    - git checkout -b hotfix/issue_107 master

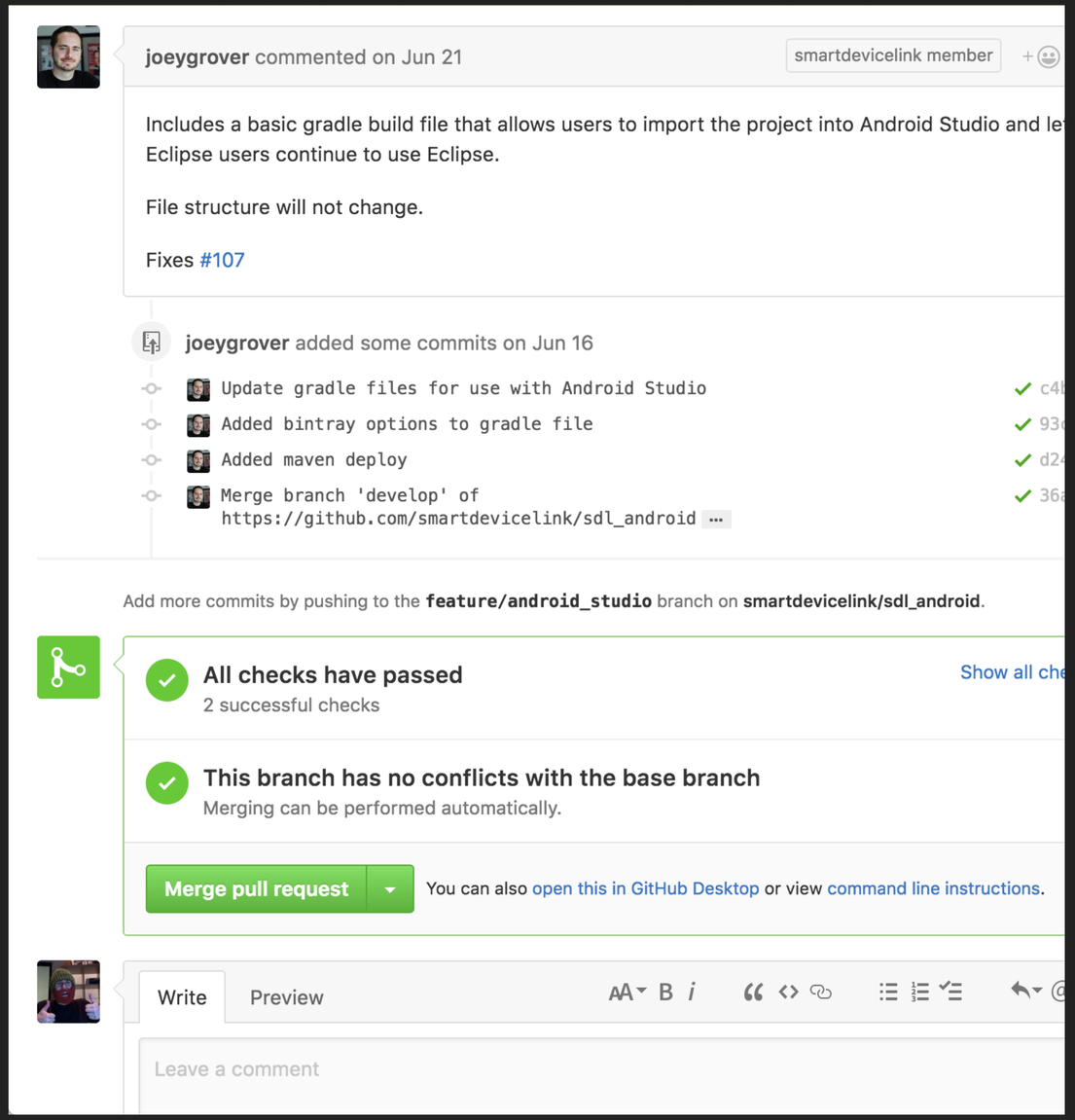
STEP 4

- Create a pull request on github with the tag [WIP]. Include a description of the changes and link any related content
  - Issue(s)
  - Documentation
  - Links to other websites
  - Code snippets



STEP 5

- Implement your contribution remembering to:
  - Commit in logical units
  - Add and update unit tests

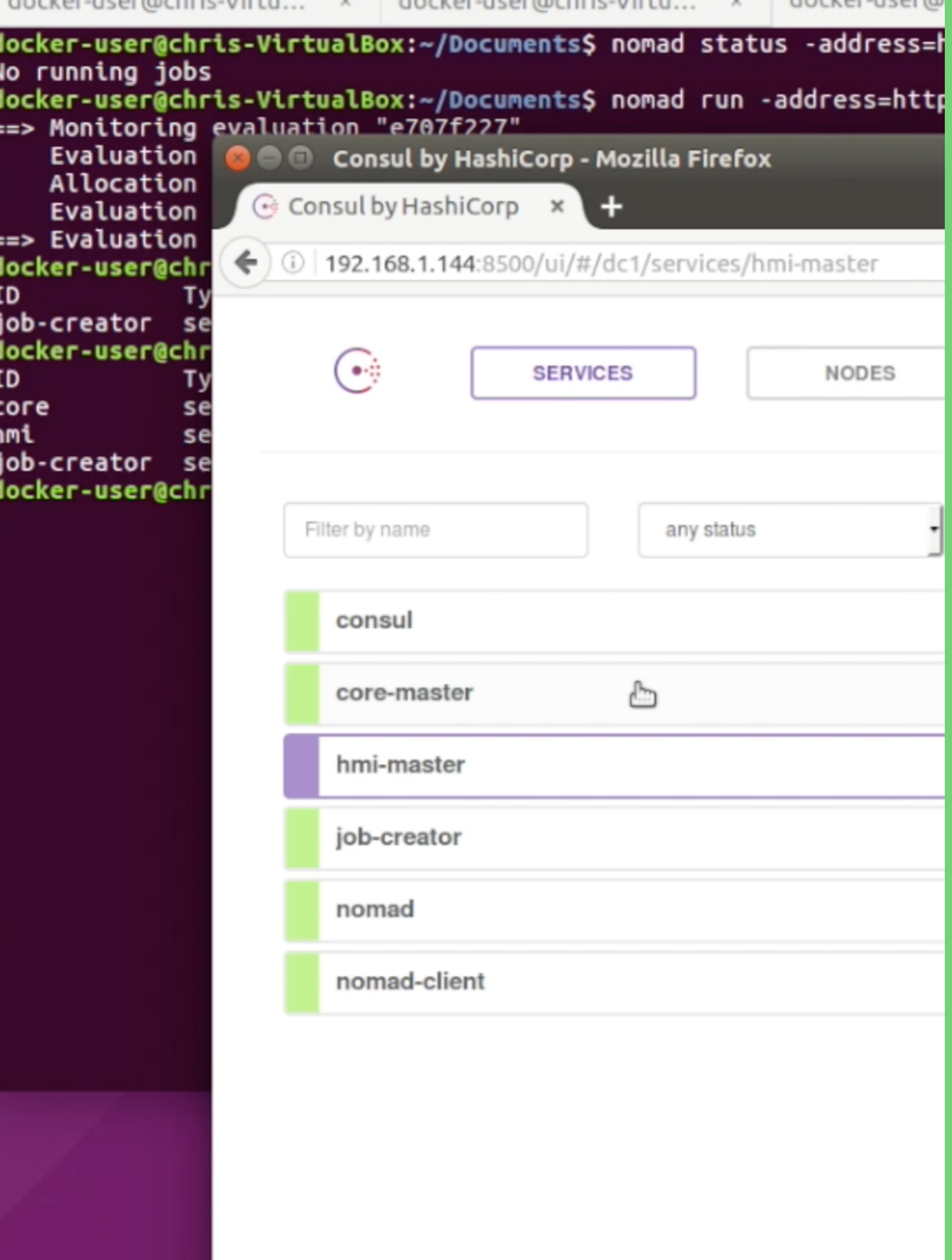


STEP 6

- When you are finished with implementation:
  - Ensure all unit tests pass the continuous integration
  - Remove the [WIP] tag
  - Add a comment mentioning one of the SDL teams
    - e.g. @smartdevicelink/android my feature is ready for review

STEP 7

- The SDL team will review and work with you to get the hotfix or feature merged
  - A feature will be merged into the develop branch for the next release
  - A hotfix will be merged into both develop and master branches



MANTICORE

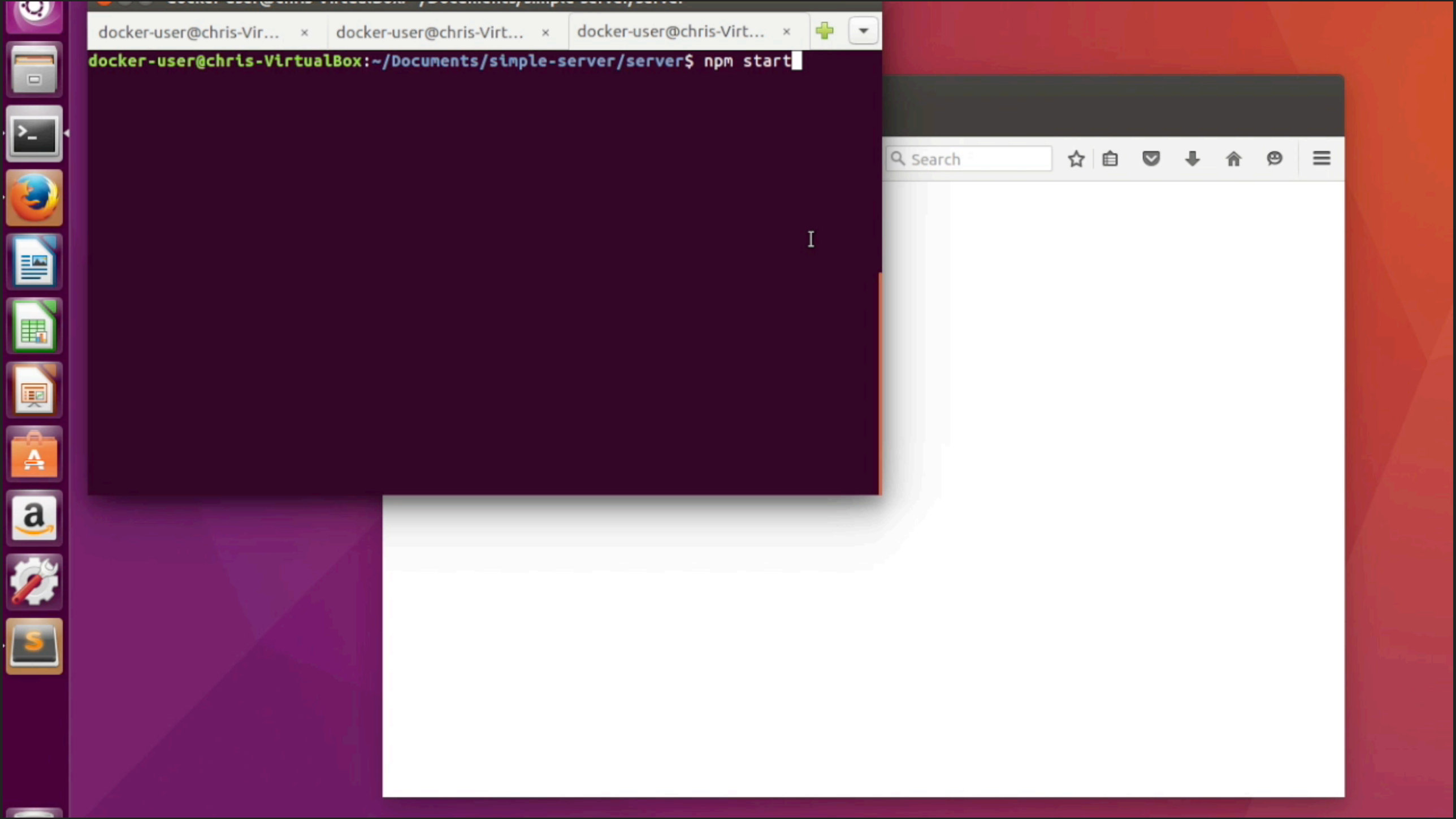
Testing apps just got easier . . .

MANTICORE

- Automagically creates an instance of SDL core and HMI in the cloud for a developer to use
  - No more virtual machines
  - No more dependency hell
  - Can work with multiple web HMIs
  - A step towards continuous integration and testing

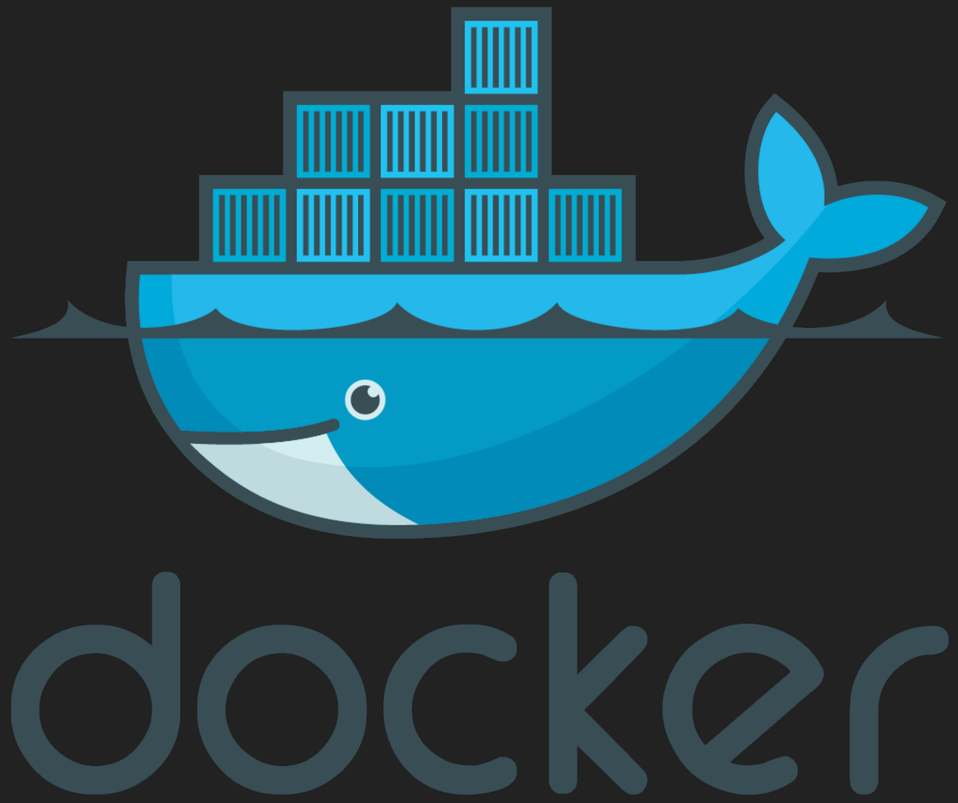
DEMO

- Short video demonstration of Manticore and how it works
  - By Chris Rokita, a Software Engineer at Livio


docker-user@chris-Vir...
docker-user@chris-Virt...
docker-user@chris-Virt...
docker-user@chris-VirtualBox:~/Documents/simple-server/server$ npm start
Search

DOCKER

- SDL Core & HMI docker container available on github & docker hub
- Install Docker, then download and run
  - $ docker run -d -p 12345:12345 -p 8080:8080 -p 8087:8087 -p 3001:3001 --name core smartdevicelink/core:latest
  - Connect app to your computer's IP address and port 12345

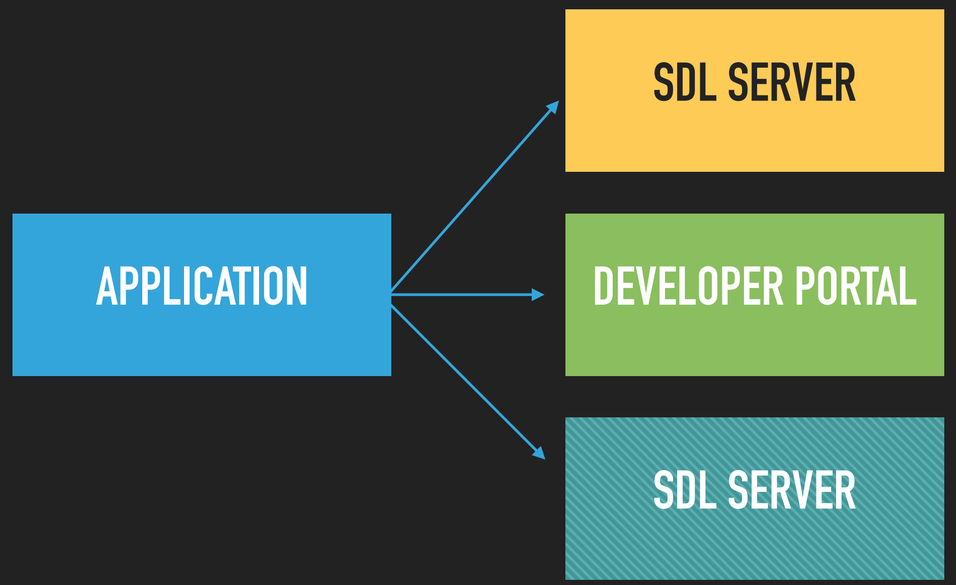


SHAID

Where did all my bits go?

PROBLEM

- Every application must be assigned a single unique application ID.
- This ID is used by SDL Core and SDL Servers to manage that application's permissions.
- However, each OEM must implement their own SDL Server.
  - How do we ensure the Application ID is unique?
  - How do we handle revoking of an ID?
  - How do we handle other application data changing, such as the package name or URL Scheme?

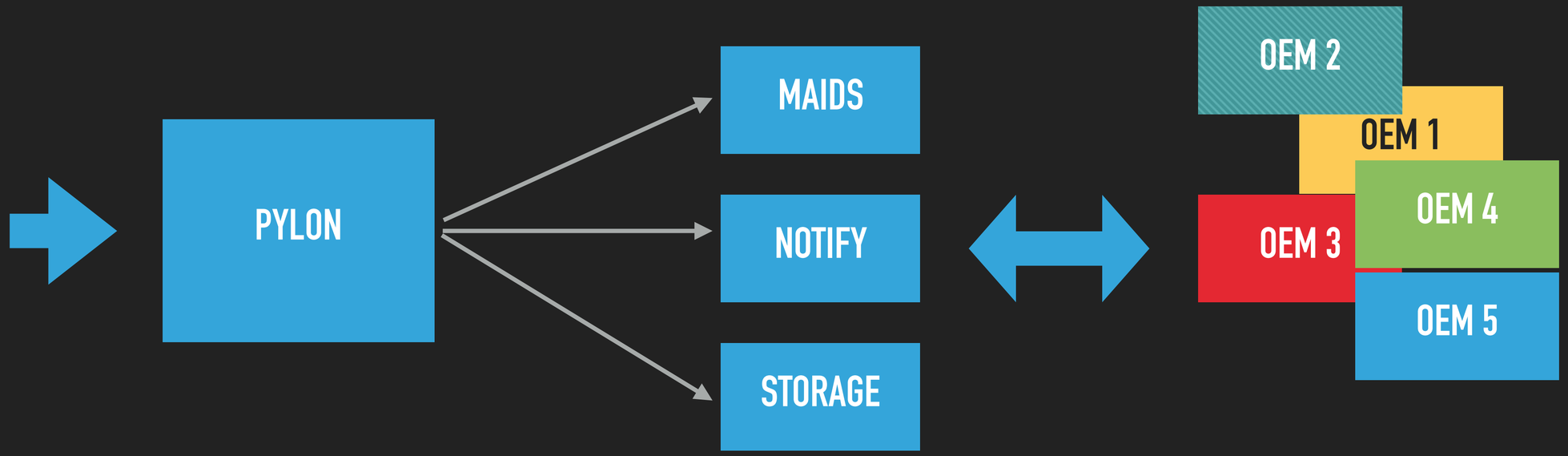


SOLUTION

- A centralized system to
  - Generate unique application IDs
  - Synchronize data
    - e.g. Application IDs, package names, permissions, url schemes
- We call this system the Super Helpful Application ID service (SHAID)

HOW DOES IT WORK

- SHAID is a series of microservices each with a very specific task
- OEM's can integrate with any of the services

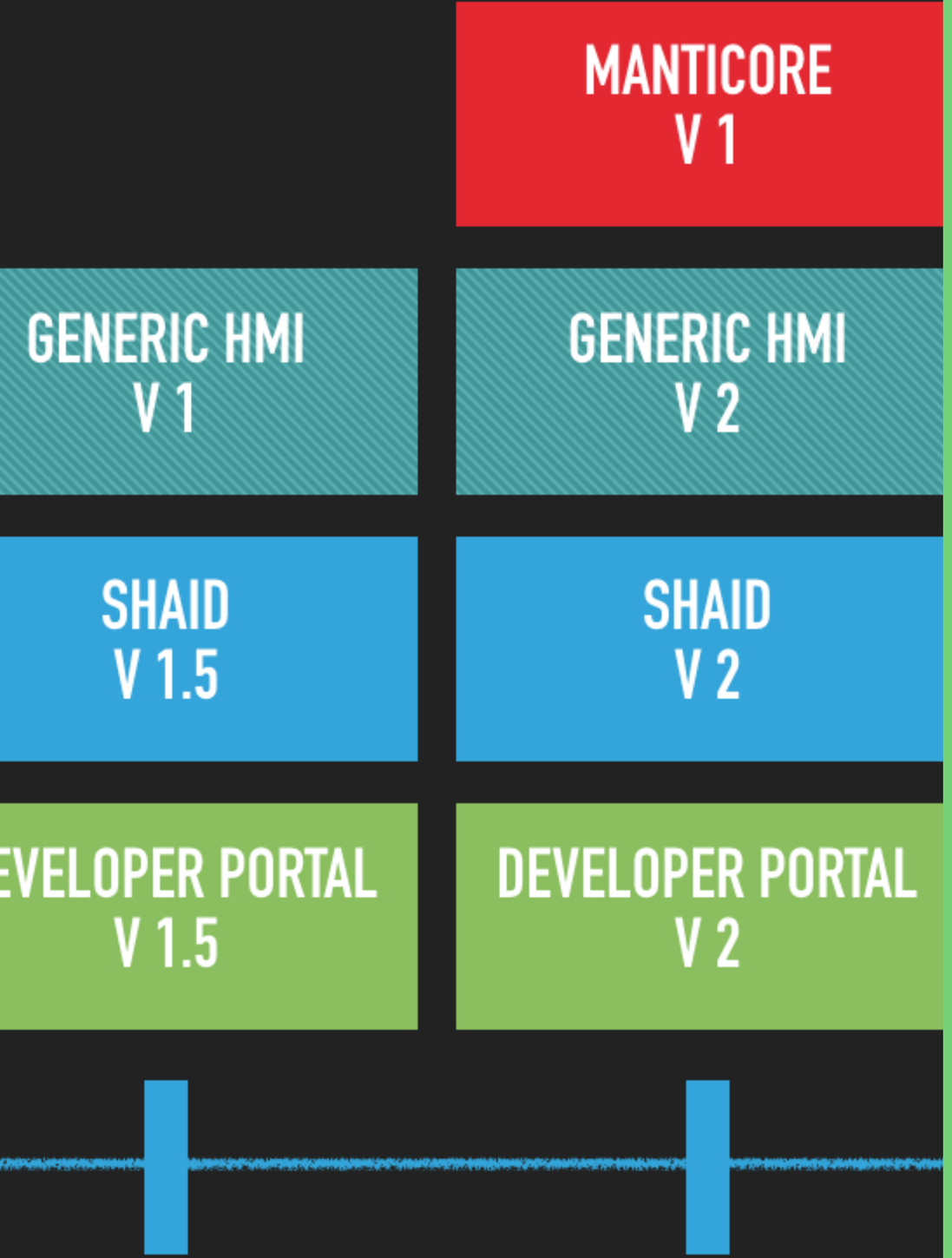


APP DEVELOPERS

- What does this mean for app developers?
  - Keep your information up-to-date on smartdevicelink.com
  - Choose which companies we can share your information
  - Follow up with any notifications you receive

OEMS

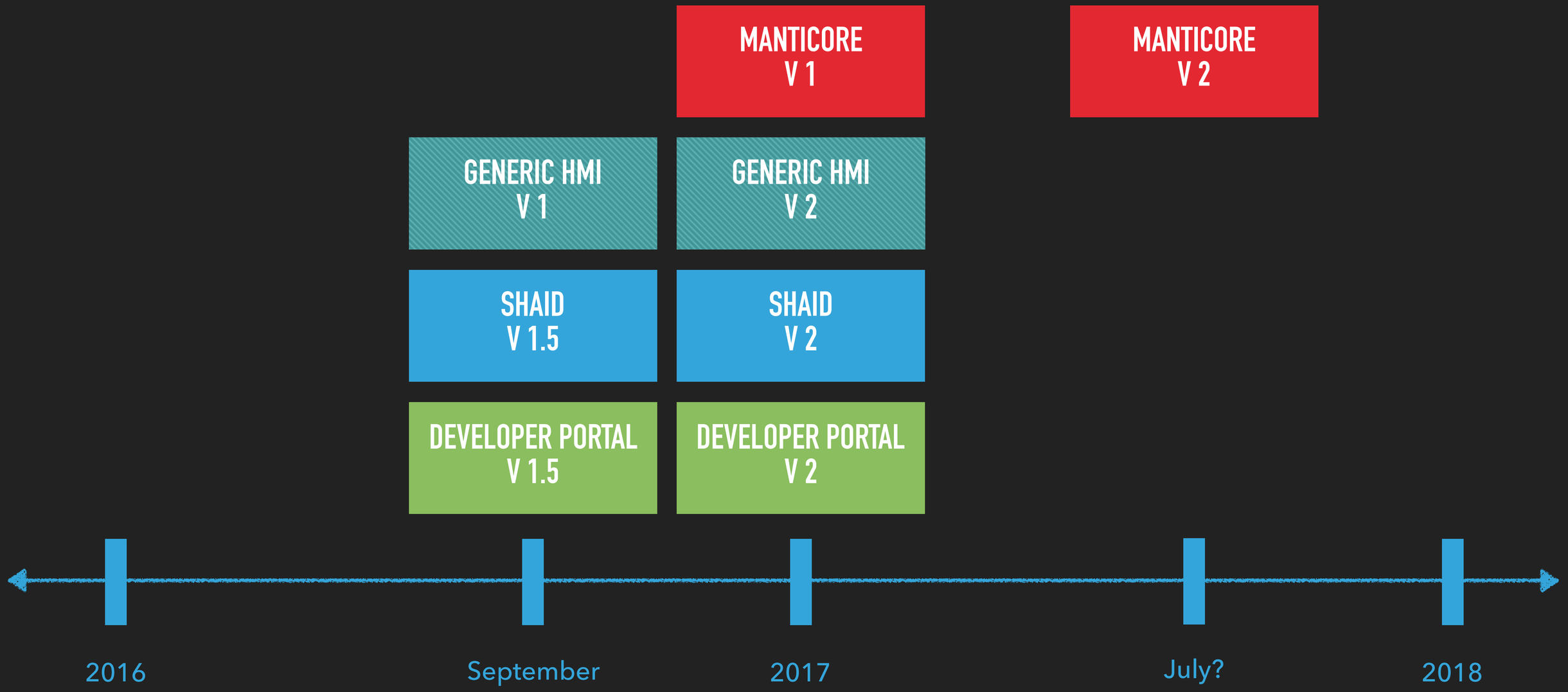
- What does this mean for OEMs?
  - SHAID will generate all application IDs
    - Be able to accept these from the developer or programmatically
  - All other data synchronization is optional
  - To integrate follow the SHAID API documentation on smartdevicelink.com
  - Only application ID generation is currently available until SHAID v2



TIMELINE

So when can I use this stuff?

ESTIMATED TIMELINES



QUESTIONS