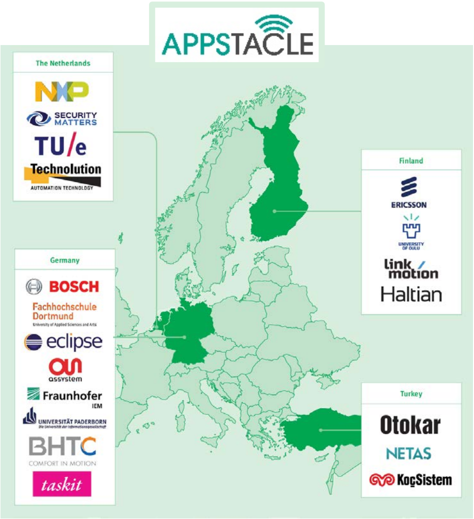
Eclipse Kuksa

Introduction & W3C VISS Implementation

- Intro to the European research project APPSTACLE and the OSS project Eclipse Kuksa
- Some insights to the Kuksa In-Vehicle platform
- The W3C VISS Implementation

APPSTACLE
ITEA 3 Publicly funded project

- **APPSTACLE:** open standard Application Platform for carS and TrAnsportation vehicles
- **Eclipse Kuksa:** the Open Source project hosting all code developed
- **Objective:** Development of an **Open Source** Connected Car Ecosystem
  - Development of an open source automotive IoT Cloud Platform
    - Architectural considerations for the cloud platform
    - Establishment of standardized interfaces to the vehicle
  - Definition and development of Service enablers for car-to-cloud connectivity
    - Network infrastructure considerations
    - Next generation mobile networks
  - Development of an open source in-vehicle platform
    - Safe and secure gateway to the cloud
    - In-vehicle data access mechanism and application platform



Project start	Project end	Project lead	Cluster
01/2017 Germany: 05/2017	12/2019	Ericsson	ITEA 3

Eclipse Kuksa
Kuksa Vision

Our vision of KUKSA

Create a ***cross-vendor*** connected vehicle platform that relies on ***open standards*** and uses ***open source software*** to leverage the potential of a ***large developer community***!

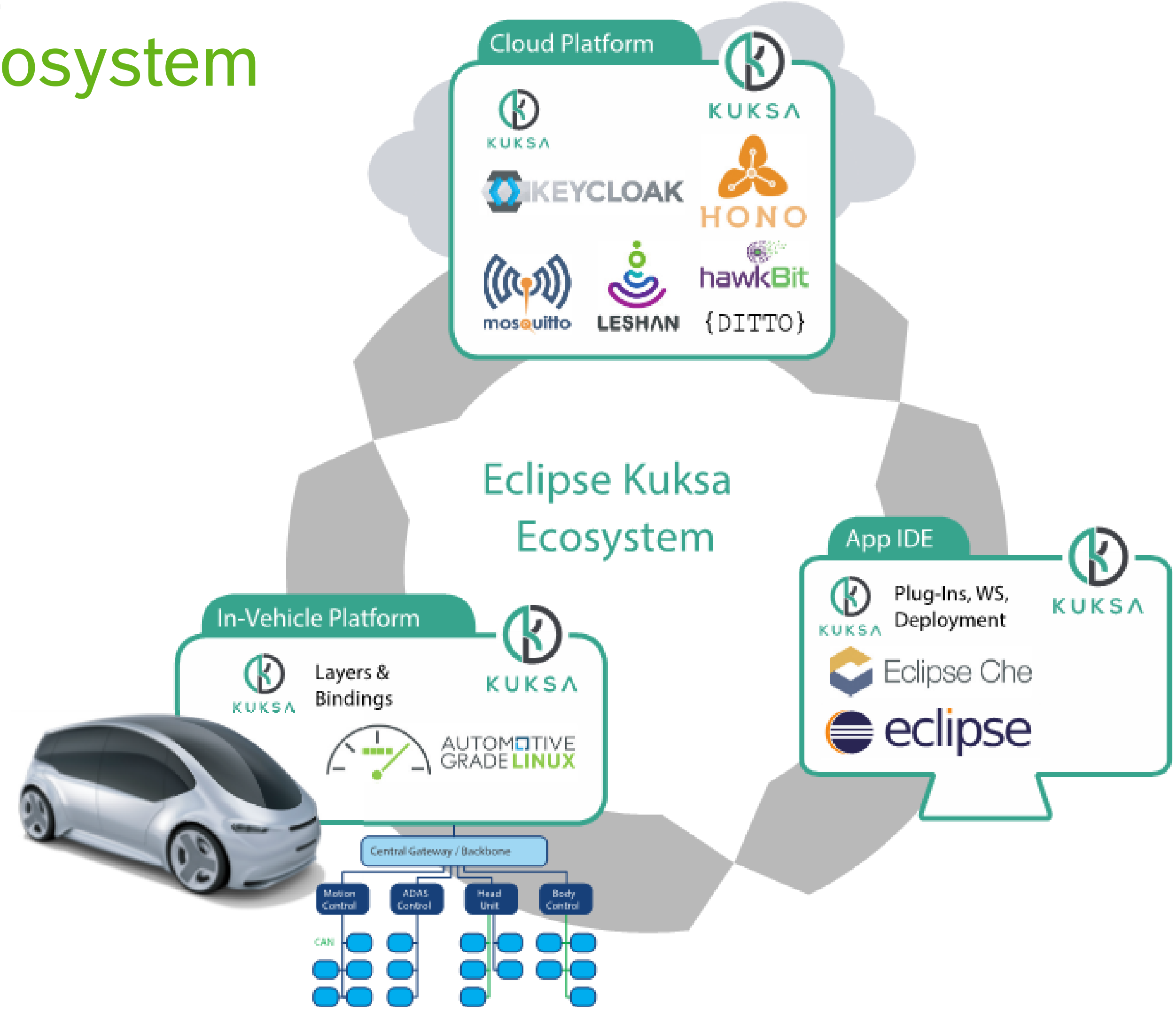
ECLIPSE FOUNDATION

iot eclipse.org

+

Eclipse Kuksa

The Kuksa Ecosystem


Eclipse Kuksa Ecosystem

Cloud Platform

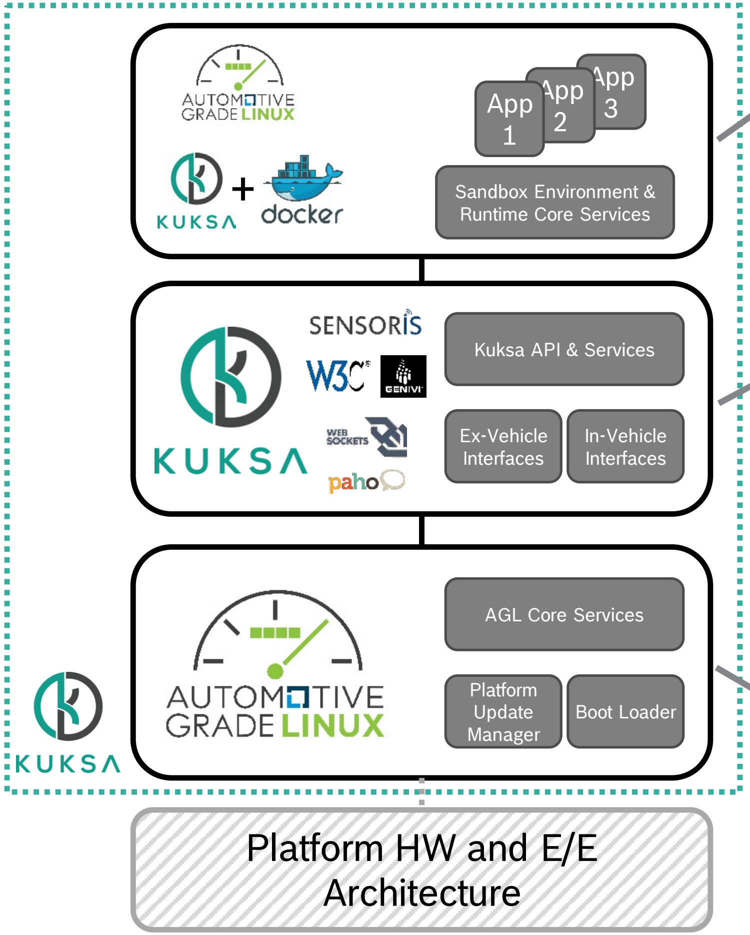
In-Vehicle Platform

Layers & Bindings

App IDE

Plug-Ins, WS, Deployment

Eclipse Kuksa



Application layer:

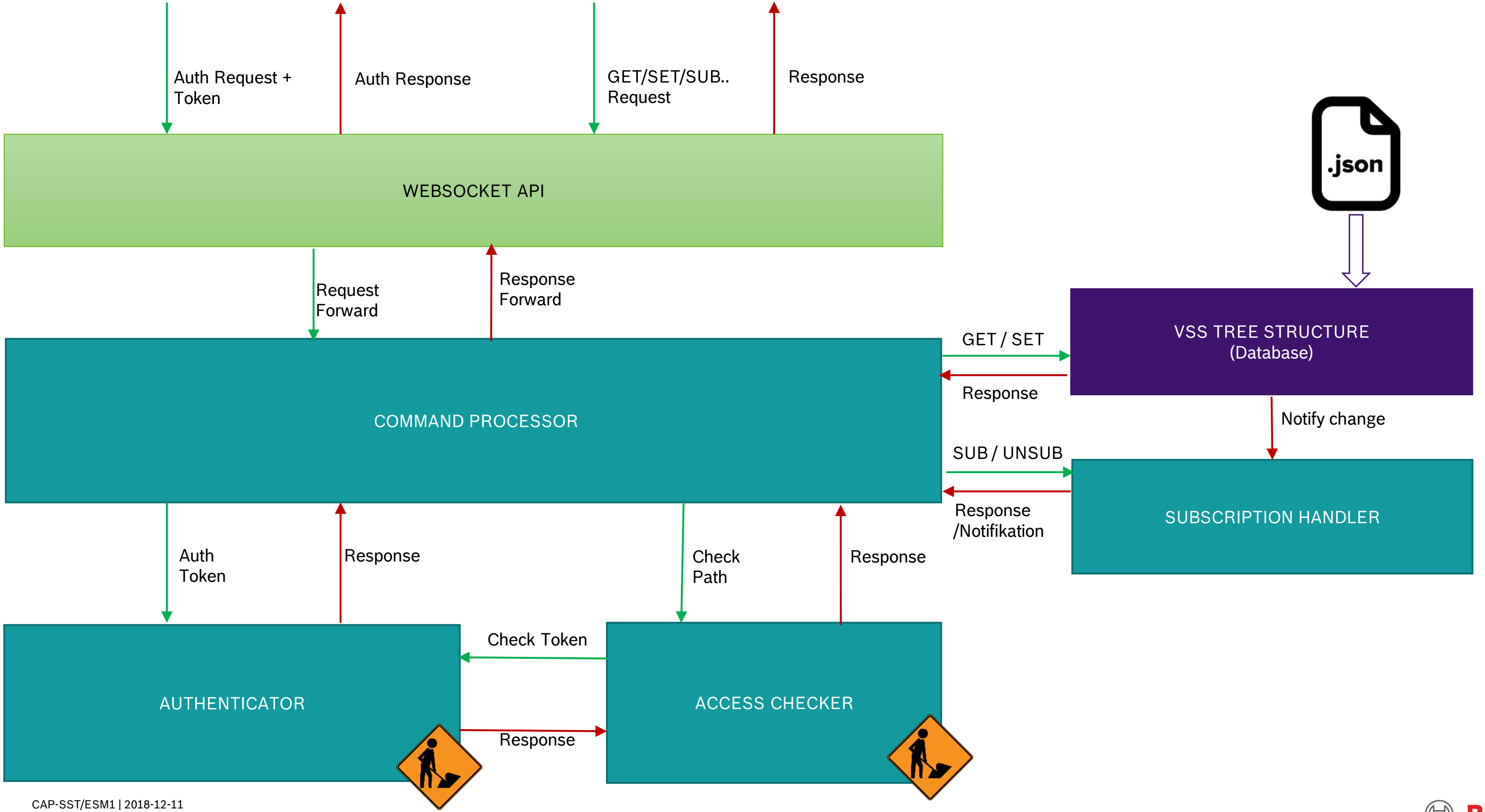
- Runs 3rd party apps on the platform
- Contains a Sandbox Environment & Additional Services

Middleware layer (Yocto layer):

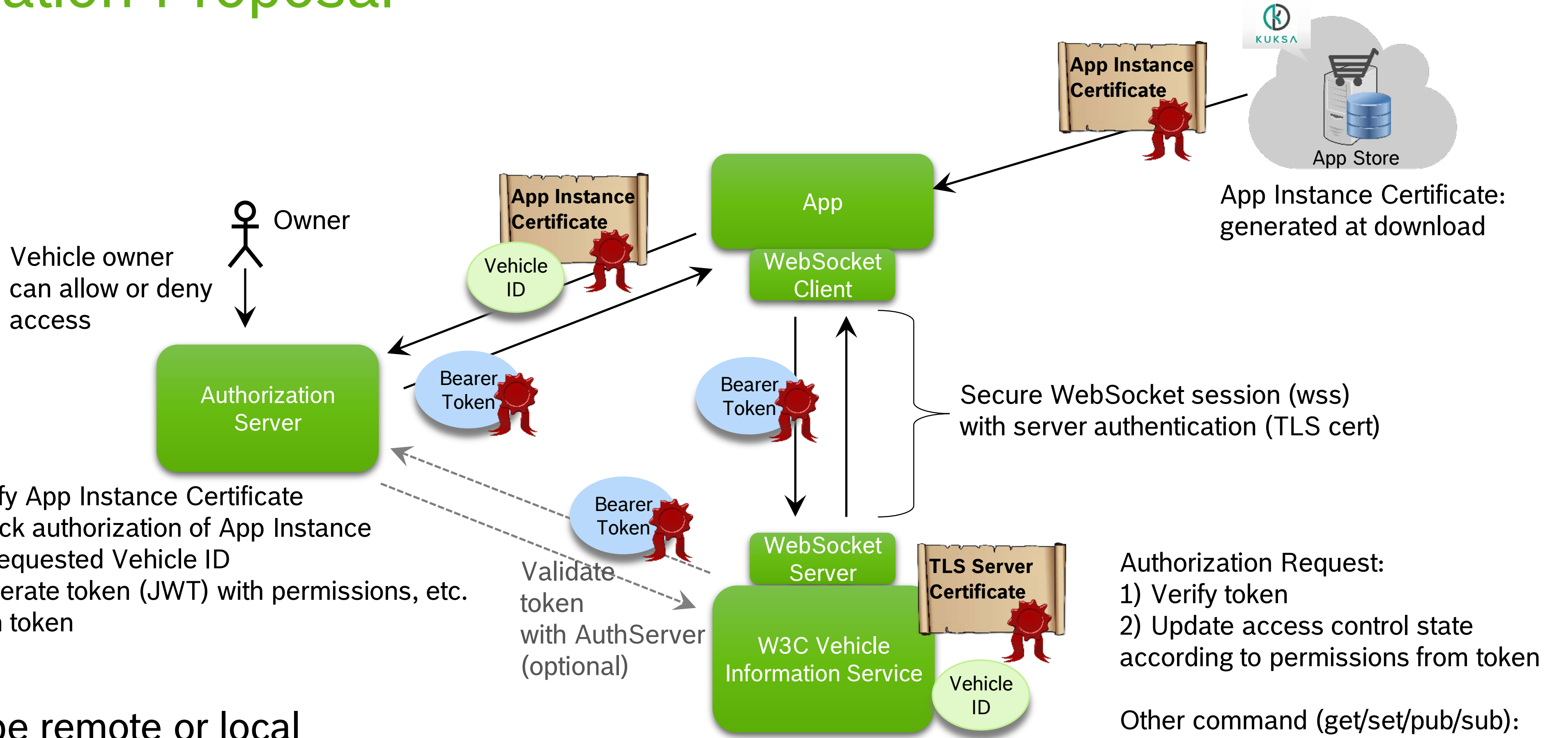
- APIs to abstract the vehicles' E/E architecture (W3C VISS, Sensoris…)
- Communication Services to manage network access and provide data from the vehicle
- Includes communication libs, protocols, security layers,…

OS layer:

- Reuse of OE's existing services, layers, HW abstractions, AGL services, etc.


Auth Request + Token
Auth Response
GET/SET/SUB.. Request
Response
.json
WEBSOCKET API
Request Forward
Response Forward
VSS TREE STRUCTURE (Database)
GET / SET
Response
Notify change
COMMAND PROCESSOR
SUB / UNSUB
Response /Notifikation
SUBSCRIPTION HANDLER
Auth Token
Response
Check Path
Response
Check Token
AUTHENTICATOR
ACCESS CHECKER
Response

Eclipse Kuksa
Authorization Proposal



App Instance Certificate: generated at download

Vehicle owner can allow or deny access

Secure WebSocket session (wss) with server authentication (TLS cert)

1) Verify App Instance Certificate
2) Check authorization of App Instance for requested Vehicle ID
3) Generate token (JWT) with permissions, etc.
4) Sign token

Validate token with AuthServer (optional)

Authorization Request:
1) Verify token
2) Update access control state according to permissions from token

Other command (get/set/pub/sub):
1) Check access control state
2) Deny access or execute command

- App could be remote or local

Eclipse Kuksa
Authorization Proposal – Open Questions

- The Security Chapter of the W3C Vehicle Information Service Specification proposes to manage access to data by using security tokens such as OAuth 2.0. However, some questions remain:
  - The exact content / structure of the token

    -> Still not defined in our implementation

    -> Is there any additional information available?
  - Details of the PKI and certificates are not defined

    -> We are currently using an open WebService to create Bearer Token (JWT) with an asymmetric RSA256 algorithm

    -> Is there any further description on the ideas regarding PKIs, certificates and tokens?

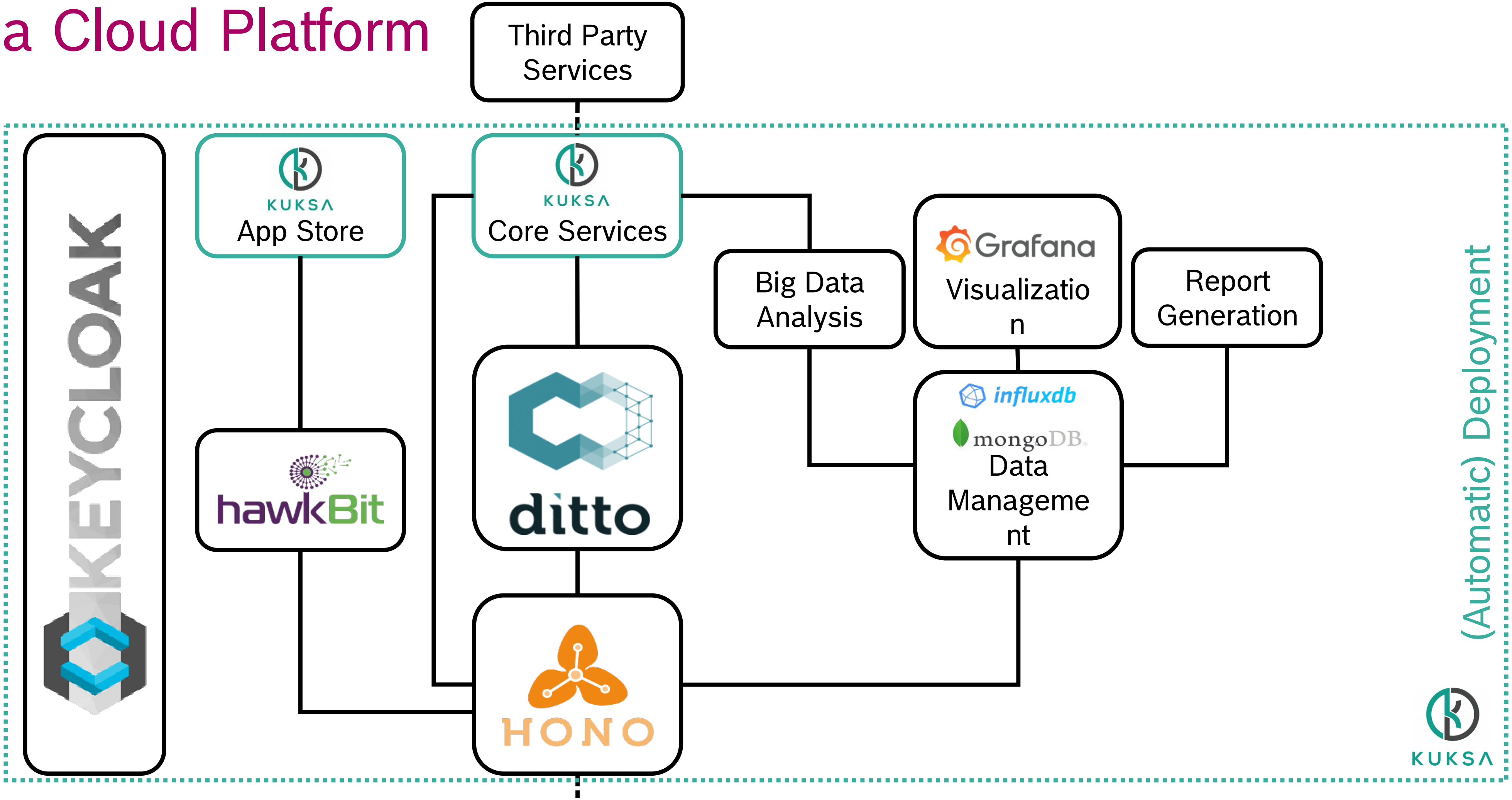
Eclipse Kuksa
Some useful links

- Eclipse Kuksa Website
  - https://www.eclipse.org/kuksa/
- Eclipse Kuksa Codebase
  - Eclipse Kuksa IDE: https://github.com/eclipse/kuksa.ide
  - Eclipse Kuksa Cloud: https://github.com/eclipse/kuksa.cloud
  - Eclipse Kuksa In-Vehicle: https://github.com/eclipse/kuksa.invehicle
  - Eclipse Kuksa Integration: https://github.com/eclipse/kuksa.integration
- Direct Contact
  - Jörg Tessmer Joerg.Tessmer@de.bosch.com
  - Pratheek Rai Pratheek.Rai@de.bosch.com
  - Sebastian Schildt Sebastian.Schildt@de.bosch.com

BACKUP

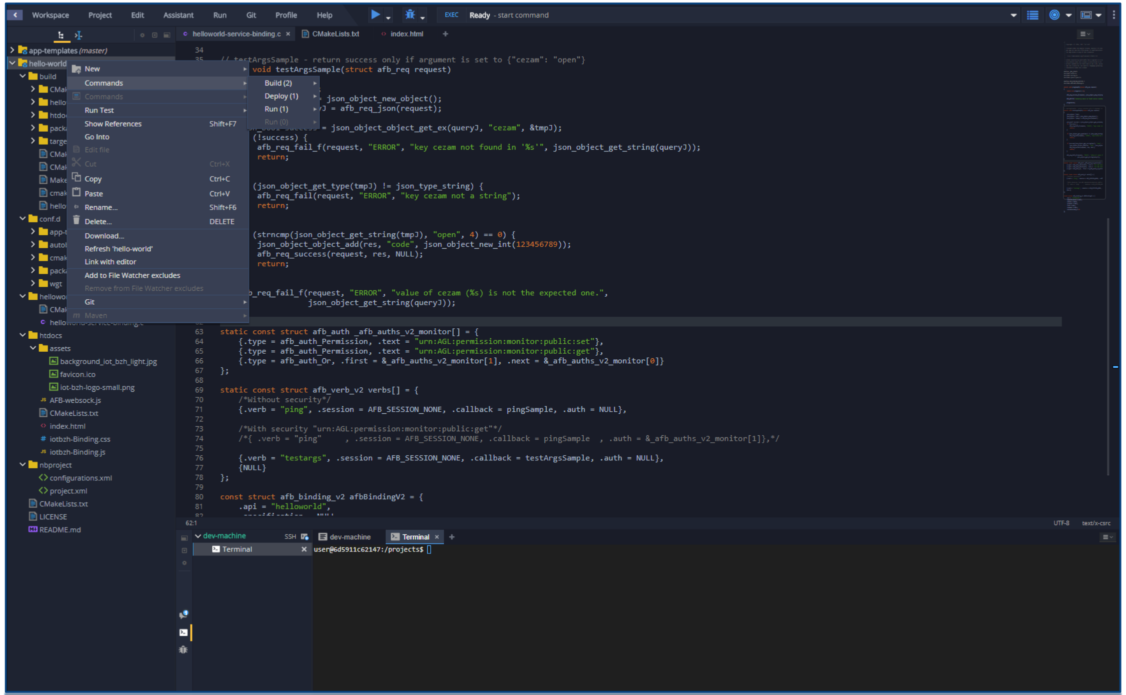
Eclipse Kuksa

Kuksa Cloud Platform



Eclipse Kuksa
Kuksa IDE

- Based on Eclipse Che
- Allows Cloud and In-Vehicle Application development
- Platform independent
- Shared workspaces
- Almost configuration free
- Docker-based: VPN planned to allow remote / network independent cross compilation

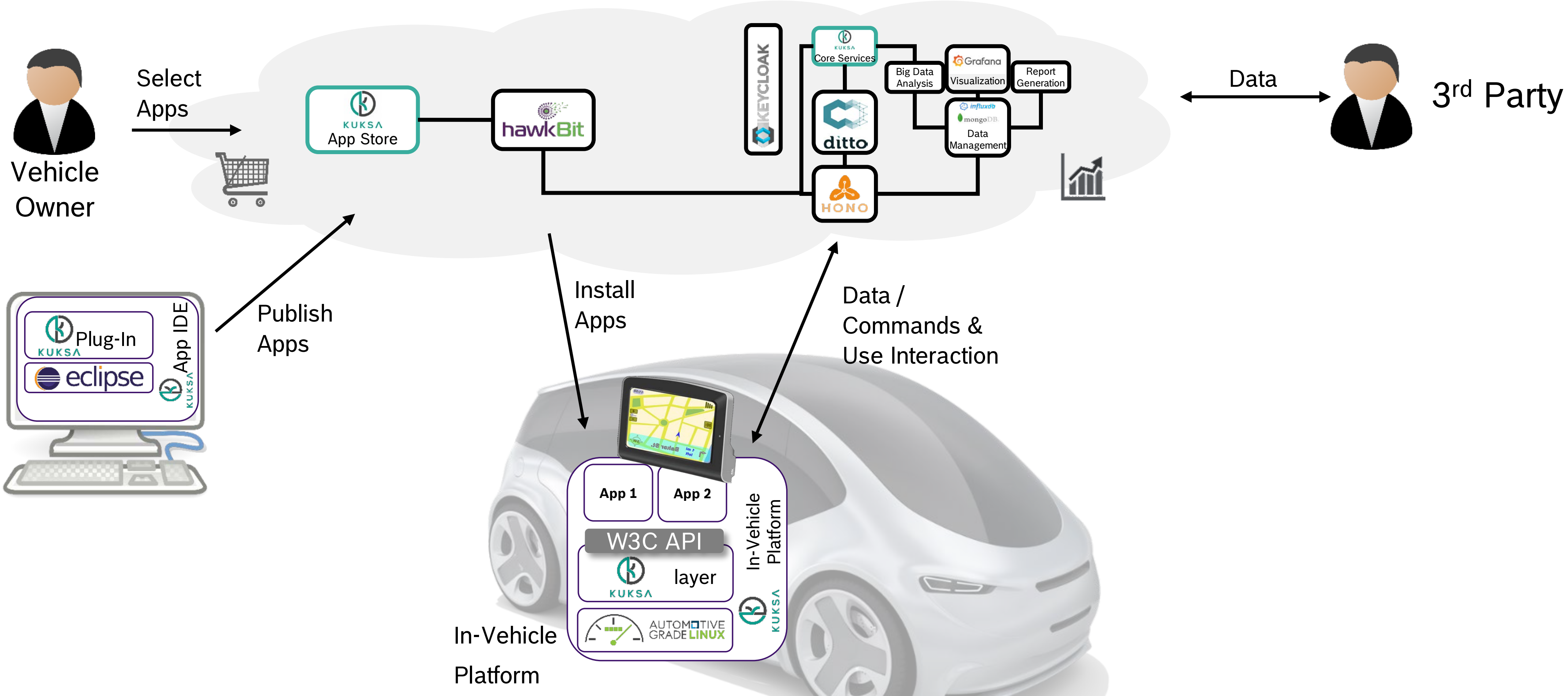


```
// testArgsSample - return success only if argument is set to {"cezam": "open"}
void testArgsSample(struct afb_req request)

= json_object_new_object();
J = afb_req_json(request);

= json_object_object_get_ex(queryJ, "cezam", &tmpJ);
(!success) {
 afb_req_fail_f(request, "ERROR", "key cezam not found in '%s'", json_object_get_string(queryJ));
 return;

(json_object_get_type(tmpJ) != json_type_string) {
 afb_req_fail(request, "ERROR", "key cezam not a string");
 return;

(strncmp(json_object_get_string(tmpJ), "open", 4) == 0) {
 json_object_object_add(res, "code", json_object_new_int(123456789));
 afb_req_success(request, res, NULL);
 return;

_req_fail_f(request, "ERROR", "value of cezam (%s) is not the expected one.",
              json_object_get_string(queryJ));

static const struct afb_auth _afb_auths_v2_monitor[] = {
    {.type = afb_auth_Permission, .text = "urn:AGL:permission:monitor:public:set"},
    {.type = afb_auth_Permission, .text = "urn:AGL:permission:monitor:public:get"},
    {.type = afb_auth_Or, .first = &_afb_auths_v2_monitor[1], .next = &_afb_auths_v2_monitor[0]}
};

static const struct afb_verb_v2 verbs[] = {
    /*Without security*/
    {.verb = "ping", .session = AFB_SESSION_NONE, .callback = pingSample, .auth = NULL},

    /*With security "urn:AGL:permission:monitor:public:get"*/
    /*{ .verb = "ping"     , .session = AFB_SESSION_NONE, .callback = pingSample  , .auth = &_afb_auths_v2_monitor[1]},*/

    {.verb = "testargs", .session = AFB_SESSION_NONE, .callback = testArgsSample, .auth = NULL},
    {NULL}
};

const struct afb_binding_v2 afbBindingV2 = {
    .api = "helloworld",
```

```
user@6d5911c62147:/projects$
```

Wagner / Tessmer | 2018-12-11

Eclipse Kuksa
Deployment Variants: Integrated



Vehicle Owner

Select Apps

Publish Apps

Install Apps

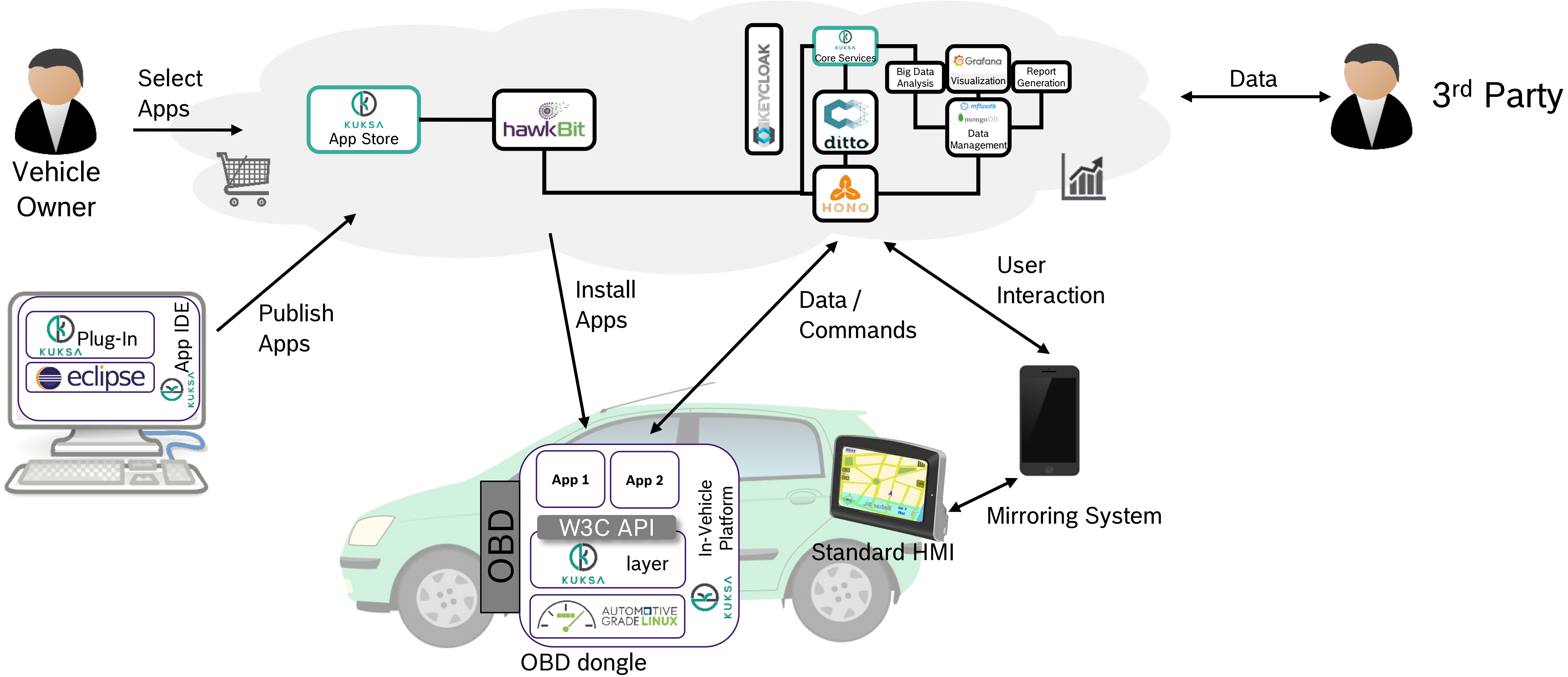
Data / Commands & Use Interaction

Data

3rd Party

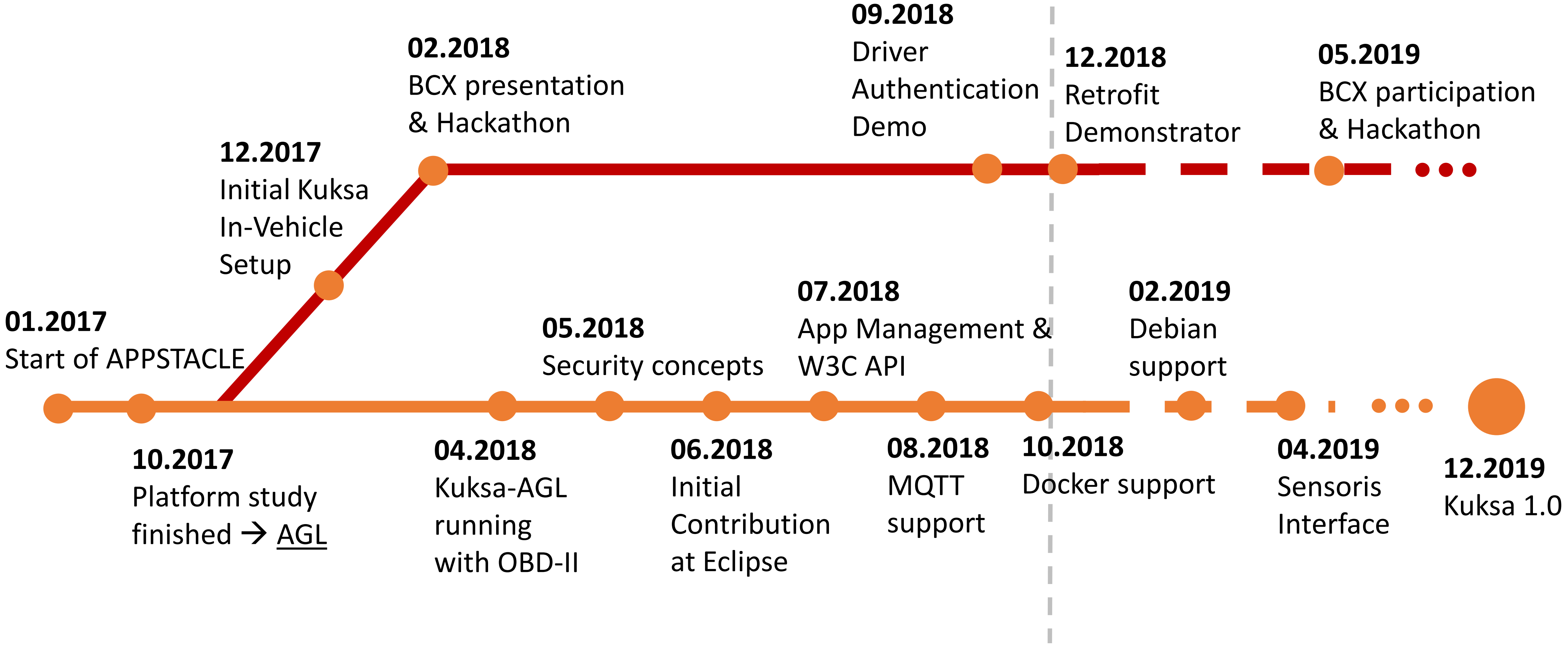
In-Vehicle Platform

Eclipse Kuksa
Deployment Variants: Retrofit



Eclipse Kuksa

Kuksa Roadmap



01.2017
Start of APPSTACLE

10.2017
Platform study finished → AGL

12.2017
Initial Kuksa In-Vehicle Setup

02.2018
BCX presentation & Hackathon

04.2018
Kuksa-AGL running with OBD-II

05.2018
Security concepts

06.2018
Initial Contribution at Eclipse

07.2018
App Management & W3C API

08.2018
MQTT support

09.2018
Driver Authentication Demo

10.2018
Docker support

12.2018
Retrofit Demonstrator

02.2019
Debian support

04.2019
Sensoris Interface

05.2019
BCX participation & Hackathon

12.2019
Kuksa 1.0