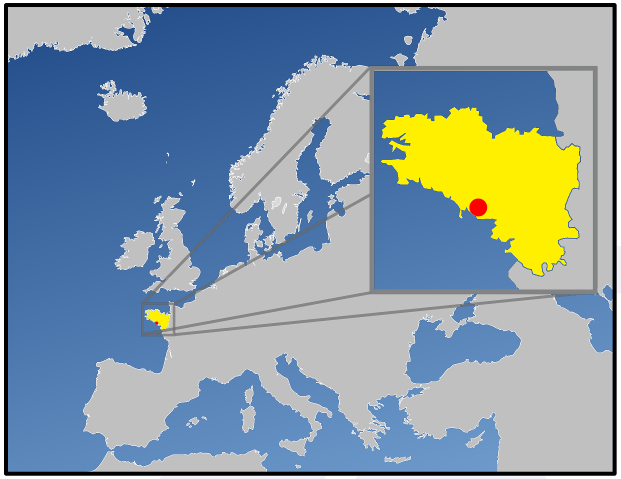
AGL Development Kit

Features and Roadmap

AGL F2F Meeting
Karlsruhe - April 2017

1st technical contributor

- **Application Development**
  - AGL Development Kit
  - Secure Application Framework (life cycle, cybersecurity)
  - Application Binder Framework (APIs exposure & protection)
- **Integration**
  - Yocto recipes
  - Releases automation & Testing (CI)
  - Renesas boards support
  - Security (MAC, Cynara, Systemd, CGroups, Namespaces,...)
- **Low Level Services**
  - Audio Management
  - Connectivity
  - Signaling & Events / CAN
  - SOTA
  - Secure Boot & Trusted Zone
- **Community Support**
  - Documentation (kickstart, developer samples, guides …)
  - White Papers & Conferences (Genivi, AGL, Fosdem, ELC …)
  - Renesas Community support



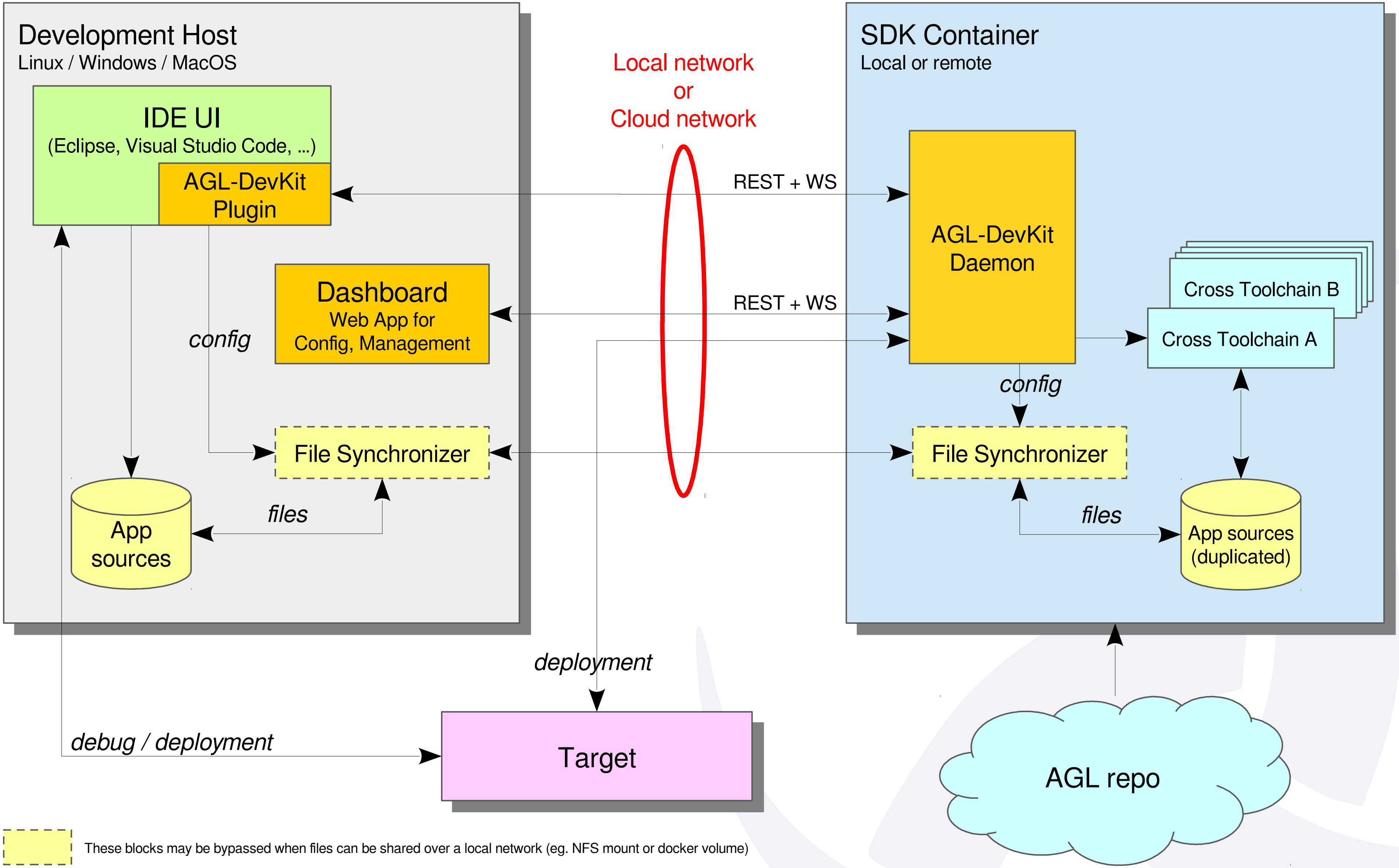
Introducing AGL DevKit

- Dedicated to **Applications Developers**
  → Yocto/bitbake platform builds are not covered
- Cross-platform **build** using AGL SDK toolchain
- Secure **packaging**
  → creation of .wgt files including signatures
- **Deploy** on development boards (or Qemu image)
- Remote **debugging** from IDE
- Easy target **access** (console, SSH, …)
- Developer environment is a **standard IDE**
  → Eclipse, Visual Studio Code, Visual Studio, Netbeans, ...
- **Dashboard Web App** to manage configuration and trigger actions
  → automated build, QA …
- Provide an **AGL DevKit API**
  → for CI workflows or specific environments

Expected Features

- **Multi-platform** : no dependencies on developer host
  AGL DevKit available for Linux / Windows / MacOS
- **Easy to setup**
  Near-zero install, no admin privileges required for specific configs
- **Application sources remain local**
  Compatibility with existing IT policies (e.g. corporate backup or SCM)
- **SDK Container ubiquity** :
  - Run locally (local subsystem, virtual machine, docker container ...)
  - Run on a local build server
  - Run on the Cloud
- **Leverage specific OS capabilities** where appliable
  (e.g. usage of Windows Subsystem to improve performance)

Architecture

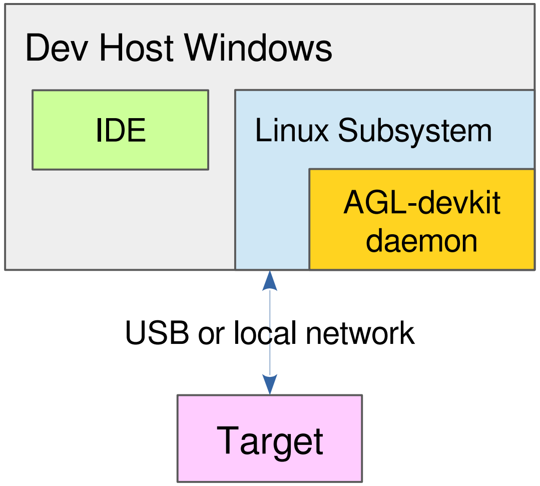


▭ These blocks may be bypassed when files can be shared over a local network (eg. NFS mount or docker volume)

Targeted Use Cases

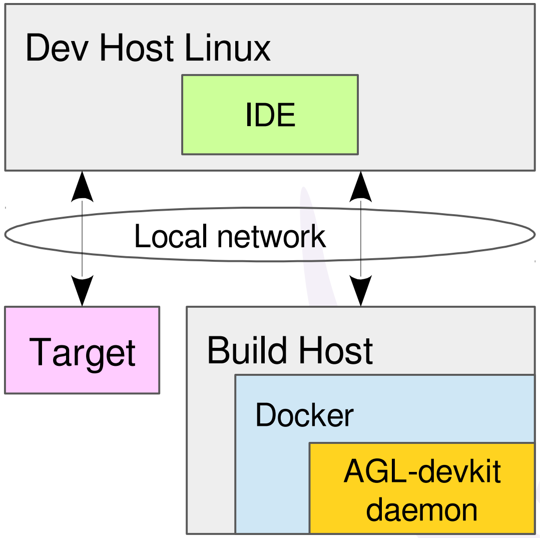
Single Host mode

- Host: Windows
- IDE: Eclipse
- Container: Linux Subsystem
- Sources: shared through native access



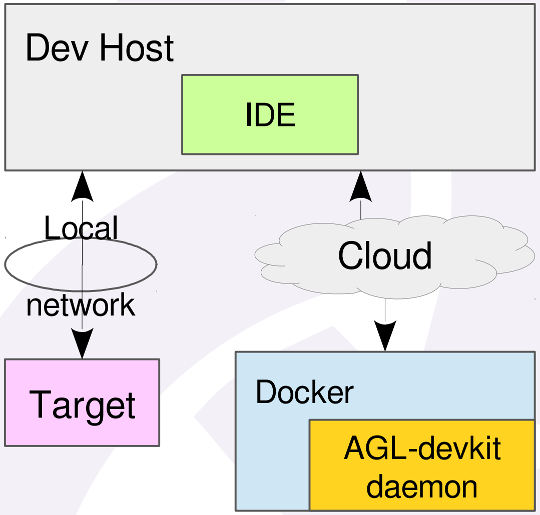
Local Network mode

- Host: Linux
- IDE: Visual Code
- Container: Docker
- Sources: shared through docker volume



Cloud mode

- Host: Linux
- IDE: Eclipse
- Container: Docker running in the Cloud
- Sources: shared through sync tool



Why not reusing CROPS ?

- Core of CROPS project(1)(2) no longer active, replaced by a single Eclipse plugin(3)
- New Eclipse plugin too much focused on Eclipse and Docker
- No RESTful API: based on sockets (doesn’t support corporate networks with firewalls as HTTP does)
- File synchronisation for Cloud configuration not supported

(1): https://github.com/crops/crops
(2): https://www.youtube.com/watch?v=R54vRP0-omw
(3): https://github.com/crops/eclipse-crops

Roadmap

- ALS ’17: Demo of remote build and debug
- AGL AMM Fall ’17: developer preview
- EE/CES ’18: release candidate

Contacts

- Sébastien Douheret

  <sebastien.douheret@iot.bzh>

- Stéphane Desneux

  <stephane.desneux@iot.bzh>

Q&A

Gulf of Morbihan, south of Brittany, France