Getting Started with Automotive Grade Linux
( builds, emulator, SDK)

Jan-Simon Möller
October 2022, Tokyo
Welcome, I am

Jan-Simon Möller
AGL Release Manager

reach me via:
jsmoeller@linuxfoundation.org

IRC: 'DL9PF' on libera.chat join #automotive there!
Overview
Goals and Topics

We will learn:

• where to get AGLs source code
• the prerequisites to build AGL
• what machines or boards are supported
• what images do exist
• how to start a build
• what prebuilt images exist
• how to run an image in the emulator
We'll not cover now:

- What 'is' AGL -> See 'AGL Architecture' presentation and docs.automotivelinux.org
- HTML5 and Flutter images are in their own presentations later
- Specifics of machines or target images → we'll present the generic concepts here first
Notes

● We won't have time to do all steps on your PC in parallel during the presentation
● All steps that need to be executed on the commandline are marked below like so:
  
  ```bash
  source aglsetup.sh
  ```

● Find this and more on https://docs.automotivelinux.org/
AGL source code
AGL source code repositories

- AGL hosts a gerrit instance for code review at
  https://gerrit.automotivelinux.org
AGL source code repositories

• A mirror is available at
  https://git.automotivelinux.org

Directory structure:

/AGL/  - layers and infra
/apps/  - application code
/src/   - middleware and platform code
/staging/  - experimental code
For simplicity, we define a shell variable for the top-level folder $HOME/AGL named $AGL_TOP:

```bash
export AGL_TOP=$HOME/AGL

echo 'export AGL_TOP=$HOME/AGL' >> $HOME/.bashrc

mkdir -p $AGL_TOP
```
Clone AGL repositories

We do use the 'repo' tool to construct the folder from multiple git repositories:

```bash
sudo apt-get install curl python-is-python3 git gnupg\language-pack-en wget qemu-system-x86-64
mkdir -p $HOME/bin
export PATH=$HOME/bin:$PATH
echo 'export PATH=$HOME/bin:$PATH' >> $HOME/.bashrc
curl https://storage.googleapis.com/git-repo-downloads/repo \  > $HOME/bin/repo
chmod a+x $HOME/bin/repo
```
Clone AGL repositories II

Next we will create a folder and download the code:

```bash
git config --global user.email "you@example.com"
git config --global user.name "Your Name"

cd $AGL_TOP
mkdir needlefish
cd needlefish
repo init -b needlefish \
-u https://gerrit.automotivelinux.org/gerrit/AGL/AGL-repo
repo sync
```
```bash
export AGL_TOP=$HOME/AGL

echo 'export AGL_TOP=$HOME/AGL' >> $HOME/.bashrc

mkdir -p $AGL_TOP

sudo apt-get install curl python-is-python3 language-pack-en wget qemu-system-x86-64 git gnupg

mkdir -p $HOME/bin

export PATH=$HOME/bin:$PATH

echo 'export PATH=$HOME/bin:$PATH' >> $HOME/.bashrc

curl https://storage.googleapis.com/git-repo-downloads/repo > $HOME/bin/repo

chmod a+x $HOME/bin/repo

git config --global user.email "you@example.com"
git config --global user.name "Your Name"

cd $AGL_TOP

mkdir needlefish

cd needlefish

dir repo init -b needlefish -u https://gerrit.automotivelinux.org/gerrit/AGL/AGL-repo

dir repo sync
```
Prerequisites to build AGL
Build host

• Your build host needs:
  • >= 8GB of RAM
  • >= 8 CPU cores
  • >= 100GB free space
    • better more
    • better a SSD or NVMe
  • Native machine preferred
    → no VM if possible
AGL does build upon the Yocto Project layers and Openembedded tooling (bitbake). Thus refer to the:

- **Supported distributions document**
  - Recommendation: Ubuntu 20.04 or Debian 10.x

```bash
sudo apt install gawk wget git diffstat unzip texinfo gcc
build-essential chrpath socat cpio python3 python3-pip
python3-pexpect xz-utils debianutils iputils-ping python3-git
python3-jinja2 libegl1-mesa libstdc++1.2-dev pylint3 xterm
python3-subunit mesa-common-dev zstd liblz4-tool
```
As much re-use of artifacts as possible …

- bitbake can re-use a folder for downloaded sources and a folder for a binary cache. We prepare to re-use these across builds below:

```bash
echo "# reuse download directories" >> $AGL_TOP/site.conf
echo "DL_DIR = "$AGL_TOP/downloads/"" >> $AGL_TOP/site.conf
echo "SSTATE_DIR = "$AGL_TOP/sstate-cache/"" >> $AGL_TOP/site.conf
```

- To use this configuration fragment, execute in a project folder lateron:

```bash
cd $AGL_TOP/my-qemux86-64-project-folder/
l n -sf $AGL_TOP/site.conf conf/
```
Option: copy content of USB-sticks

USB sticks are provided with a downloads/ and sstate-cache/ folder.
Copy these into $AGL_TOP/downloads and $AGL_TOP/sstate-cache
sudo apt install gawk wget git diffstat unzip texinfo gcc \
    build-essential chrpath socat cpio python3 python3-pip \
    python3-pexpect xz-utils debianutils iputils-ping python3-git \
    python3-jinja2 libegl1-mesa libsd11.2-dev pylint3 xterm \
    python3-subunit mesa-common-dev zstd liblz4-tool

# prepare configuration fragment "site.conf"
echo "# reuse download directories" >> $AGL_TOP/site.conf
echo "DL_DIR = ""$AGL_TOP/downloads/"" >> $AGL_TOP/site.conf
echo "SSTATE_DIR = ""$AGL_TOP/sstate-cache/"" >> $AGL_TOP/site.conf
Supported machines or boards
Supported Boards

- AGL supports these as reference platforms:
  - Renesas R-Car 3
    - h3ulcb, m3ulcb, Kingfisher add-on board
    - AGL Reference Hardware board
  - x86_64 (via qemu686-64 as MACHINE)
  - ARM32 (via qemuarm)
  - AARCH64 (via qemuarm64)
  - Pi4
Supported Boards

- AGL supports these as community supported:
  - TI
    - beaglebone / beaglebone enhanced
    - j721e-evm
  - NXP
    - i.mx6 (via cubox-i)
    - imx8mq-evk
  - qemuriscv64
aglsetup.sh

aglsetup.sh is the setup script to manage all the required settings for boards/layers/agl-features.

TLDR:
aglsetup.sh -h will list all options
aglsetup.sh -m <MACHINE> will select the board
aglsetup.sh -b <myproject> will set the folder
AGL (demo) images
AGL demo images

• There are a number of demo images available:
  • IVI demo:
    • `agl-ivi-demo-platform` (Qt)
      • `agl-ivi-demo-platform-crosssdk`
    • `agl-ivi-demo-platform-flutter`
    • `agl-ivi-demo-platform-html5`
There are a number of demo images available:

- Instrument Cluster demo:
  - `agl-cluster-demo-platform`
  - `agl-cluster-demo-platform-flutter`
  - `(agl-cluster-demo-qtcompositor)`
- Telematics demo:
  - `agl-telematics-demo-platform`
Expert Group images

Instrument Cluster EG:
- agl-cluster-demo-lxc-host

Production-IVI EG:
- agl-image-flutter-runtimedebg
- agl-image-flutter-runtimeprofile
- agl-image-flutter-runtimerelease
- agl-image-boot-boot-basesystem
Generic Images for re-use

- **agl-image-boot**
  - minimal/smallest bootable image
- **agl-image-minimal**
  - minimal console tooling
- **agl-image-weston**
  - image with wayland+weston
- **agl-image-agl-compositor**
  - image with wayland+agl-compositor
How to build an AGL image
Choices

• With all preparations done, it is time to build an example image.
• The choices are:
  • MACHINE = qemux86-64
  • agl-demo-platform-crosssdk

Note:
Images might require certain options to be enabled
This requires the following call to aglsetup.sh:

```
  cd $AGL_TOP
  cd needlefish
  source meta-agl/scripts/aglsetup.sh agl-demo agl-devel
```

Note:

- this uses a default project folder of "./build"
- recommendation is to specify one with "-b"
Recommended option - cache setup:

```bash
ln -sf $AGL_TOP/site.conf conf/
```
agl-demo-platform-crosssdk III

Time to build the image:

```
bitbake agl-demo-platform-crosssdk
```

This takes a long time. The outcome is in:

```
ls tmp/deploy/images/qemux86-64
```
Re-entering an existing project

Whenever you want to enter an existing project folder, e.g. because you started a new terminal session or rebooted, you need to call this:

```
cd $AGL_TOP/needlefish
cd build
source agl-init-build-env
```
Summary

cd $AGL_TOP

cd needlefish

source meta-agl/scripts/aglsetup.sh agl-demo agl-devel

ln -sf $AGL_TOP/site.conf conf/

bitbake agl-demo-platform-crosssdk

ls tmp/deploy/images/qemux86-64

cd $AGL_TOP/needlefish

cd build

source agl-init-build-env
Prebuilt images and artifacts
Release images

• AGL does provide prebuilt artifacts for releases:
https://download.automotivelinux.org/AGL/release/

E.g.:
• https://download.automotivelinux.org/AGL/release/needlefish/14.0.0/qemux86-64/deploy/images/qemux86-64/agl-demo-platform-crosssdk-qemux86-64.wic.vmdk.xz
• https://download.automotivelinux.org/AGL/release/needlefish/14.0.0/qemux86-64/deploy/sdk/poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-corei7-64-qemux86-64-toolchain-14.0.0.sh
Nightly Snapshots

A nightly build publishes its artifacts to:

https://download.automotivelinux.org/AGL/snapshots/

E.g.:

- https://download.automotivelinux.org/AGL/snapshots/needlefish/latest/qemux86-64/deploy/images/qemux86-64/agl-demo-platform-crosssdk-qemux86-64.wic.vmdk.xz
- https://download.automotivelinux.org/AGL/snapshots/needlefish/latest/qemux86-64/deploy/sdk/poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-corei7-64-qemux86-64-toolchain-14.0.0.sh
Run an image in the emulator
'runqemu'

runqemu is a helper to run the resulting image and part of the bitbake environment:

```bash
cd $AGL_TOP/needlefish
cd build
source agl-init-build-env
runqemu kvm slirp serialstdio [snapshot] [publicvnc]
```
How to use prebuilt artifacts

See:

https://docs.automotivelinux.org/en/needlefish/#0_Getting_Started/1_Quickstart/Using_Ready_Made_Images/
Running a prebuilt image:

### Don't do that at the training (-ETooSlowInternet) - try at home

cd $AGL_TOP
mkdir prebuilt
cd prebuilt
wget -nd -O AGLx86.ext4.xz -c "https://bit.ly/3qjTrel"
xz -d AGLx86.ext4.xz

```bash
qemu-system-x86_64 -device virtio-net-pci,netdev=net0,mac=52:54:00:12:35:02 -netdev user,id=net0,hostfwd=tcp::2222-:22 \
  -drive file=AGLx86.ext4,if=virtio,format=raw -show-cursor -usb -usbdevice tablet \
  -device virtio-rng-pci -vga virtio -machine q35 -cpu kvm64 -cpu host -enable-kvm \
  -m 2048 -serial mon:vc -serial mon:stdio -serial null -kernel bzImage \
  -append 'root=/dev/vda rw console=tty0 mem=2048M ip= dhcp oprofile.timer=1 \
  console=ttyS0,115200n8 verbose fstab=no'
```
Demo

agl-demo-platform image in qemu
Questions ?
Q/A

- Don't hesitate and ask now!
- Slides are available.
- Questions can be sent later-on as well to:
  - https://lists.automotivelinux.org/g/agl-dev-community/messages
- Email: jsmoeller@linuxfoundation.org
- IRC: DL9PF  #automotive on libera.chat
Thank you!

Thanks for joining.
Appendix
The 'traditional' SDK
The SDK installer

• bitbake can output a self-extracting SDK installer
• it is in the folder tmp/deploy/sdk/
• it contains a c/c++ toolchain and the libraries matching the image built
Execute the installer

```bash
cd $AGL_TOP/needlefish
cd build/tmp/deploy/sdk/
./poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-*.sh

# Select target directory for SDK : ~/AGL/agl-sdk
```
Activate the SDK environment

Every time you want to use the SDK environment, you have to source the script:

```
source ~/AGL/agl-sdk/environment-setup-corei7-64-agl-linux
```

To check the compiler is set:

```
echo $CC
```