I ❤ the Yocto Project

foss-north // Gothenburg 2018

Introduction

AUTOSAR

yocto PROJECT

Qt

- Gordan Markuš
- Embedded Linux, Luxoft, PELUX, AUTOSAR Adaptive

Questions for the crowd

- How many attendes have built their own Embedded Linux image?
- How many attendes have used the Yocto Project?

```
$ systemctl start yocto-presentation
```

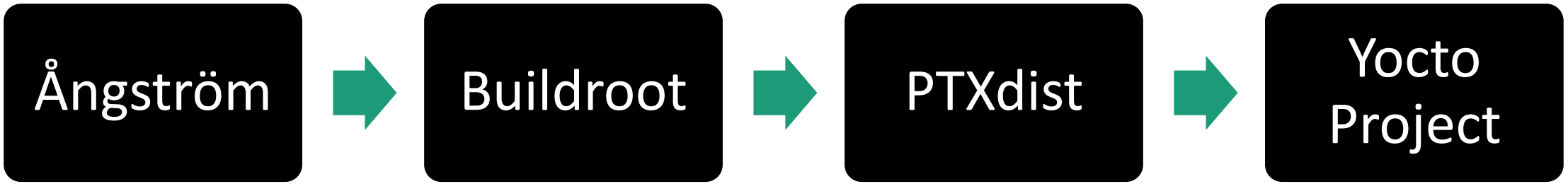
Embedded Linux Build Systems

- Goals
  - Creating custom Embedded Linux images
  - Cross-compiling applications
  - Packaging applications
  - Create integration points for custom software
  - Testing output binaries
  - Checking ecosystem compatibilty

Embedded Linux Build Systems

- Buildroot
- OpenWRT
- PTXdist
- Yocto Project / OpenEmbedded
- …

My Embedded Linux Path



What is the Yocto Project?

„The Yocto Project (YP) is an open source collaboration project that helps developers ***create custom Linux-based systems for embedded products****, regardless of the hardware architecture.”*

Yocto Project

- Core building blocks
  - OpenEmbedded-Core
  - BitBake
  - Poky

OpenEmbedded-Core

- Evolved from OpenEmbedded in collaboration with Yocto
- Compilation of components shared between OE based systems
  - Layers
  - Recipes
  - Classes

Yocto Recipe

- Metadata file
- Describes the component dependencies, build, deploy steps, etc.
- Specific syntax

```
#
#   Copyright (C) 2017 Pelagicore AB
#   SPDX-License-Identifier: MIT
#

SUMMARY = "A template C/C++ library"
DESCRIPTION = "A C/C++ shared library blueprint"
AUTHOR = "Gordan Markuš <gordan.markus@pelagicore.com>"
HOMEPAGE = "https://github.com/Pelagicore/template-library"
LICENSE = "MPL-2.0"
LIC_FILES_CHKSUM = "file://LICENSE;md5=815ca599c9df247a0c7f619bab123dad"
PV = "1.0+git${SRCREV}"
PR = "r0"

SRCREV = "29d6dbd859d725ee6d10686ee5449069412de2c6"
SRC_URI = "git://github.com/Pelagicore/template-library.git;protocol=https;branch=master"

S = "${WORKDIR}/git/"

inherit cmake pkgconfig

PACKAGES = " \
    ${PN}     \
    ${PN}-dbg \
    ${PN}-dev \
"

FILES_${PN} += " \
    ${libdir}/libtemplate-library.so \
"
```

Component information

Source information

Build and deploy instructions

OpenEmbedded-Core

- Example recipes
  - ALSA
  - Busybox
  - cmake
  - autoconf

Yocto Layer

- Collection of recipes that have a common purpose
- Including a layer gives you the possibility to use its components
- `meta-` prefix

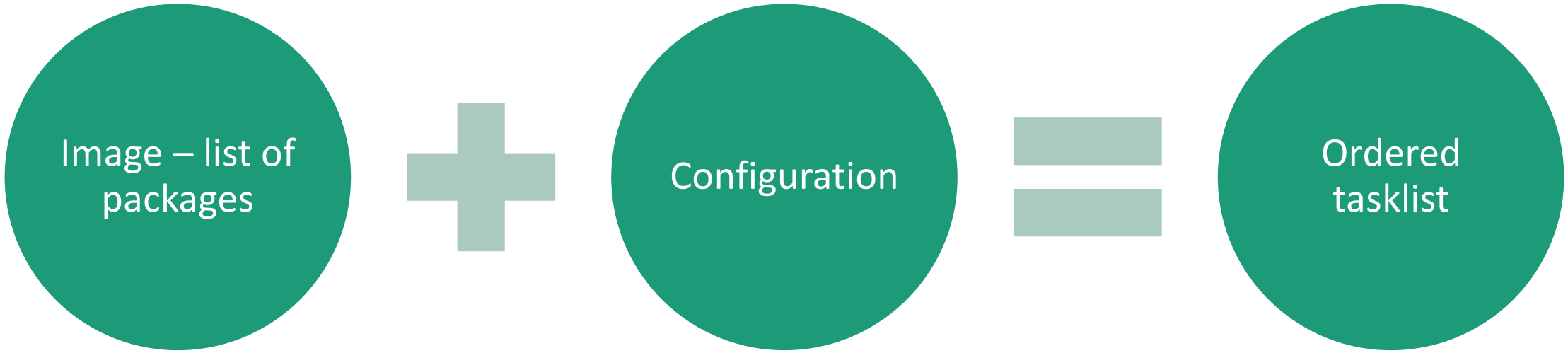
Yocto Layer

- Example layers
  - meta-qt5
  - meta-networking
  - meta-intel
  - meta-project-foo

BitBake

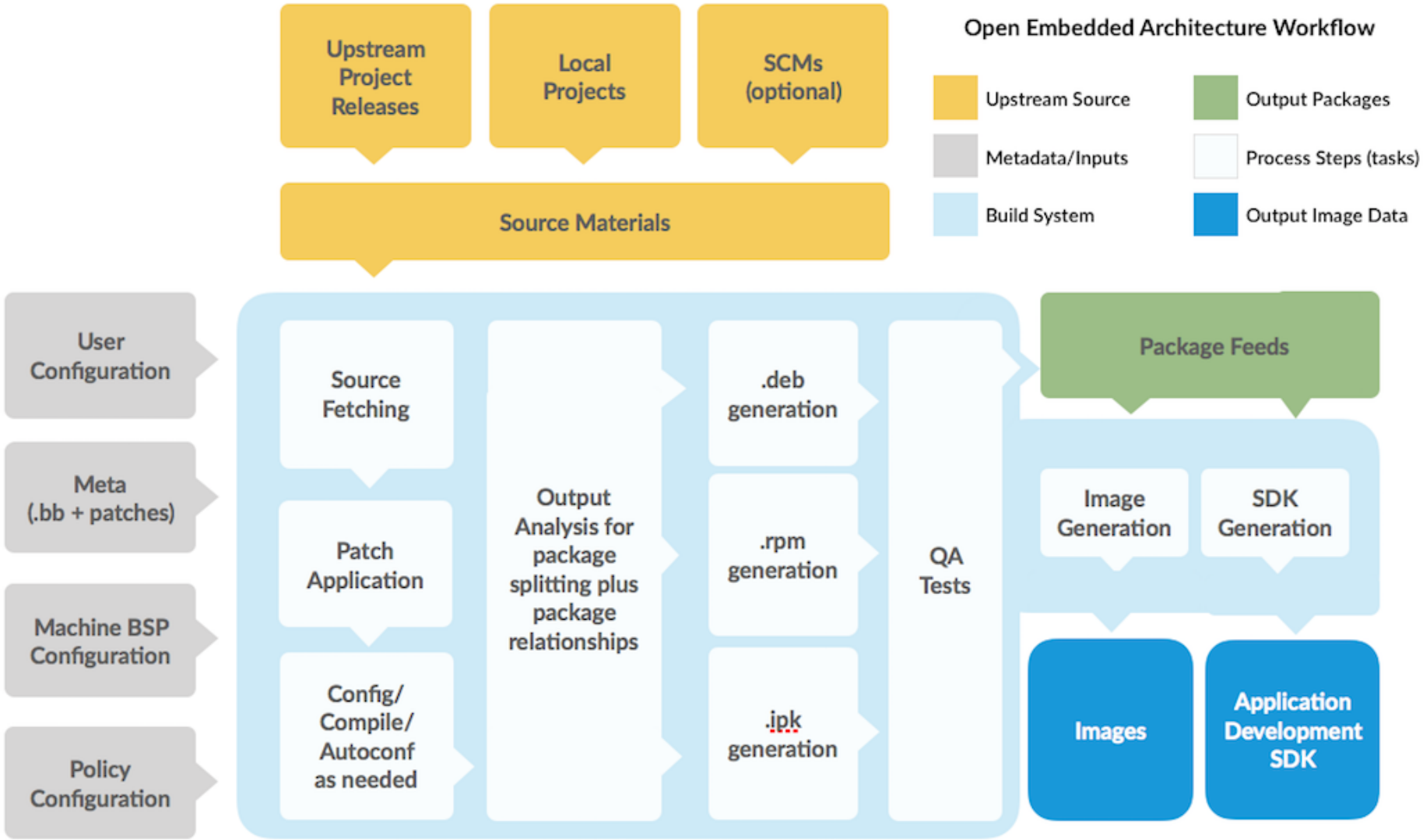
- Build engine
- Parses and interprets the metadata files
- Task scheduler

BitBake



Poky

- Yocto reference implementation
- Collection of tooling and configuration used to create a new distro

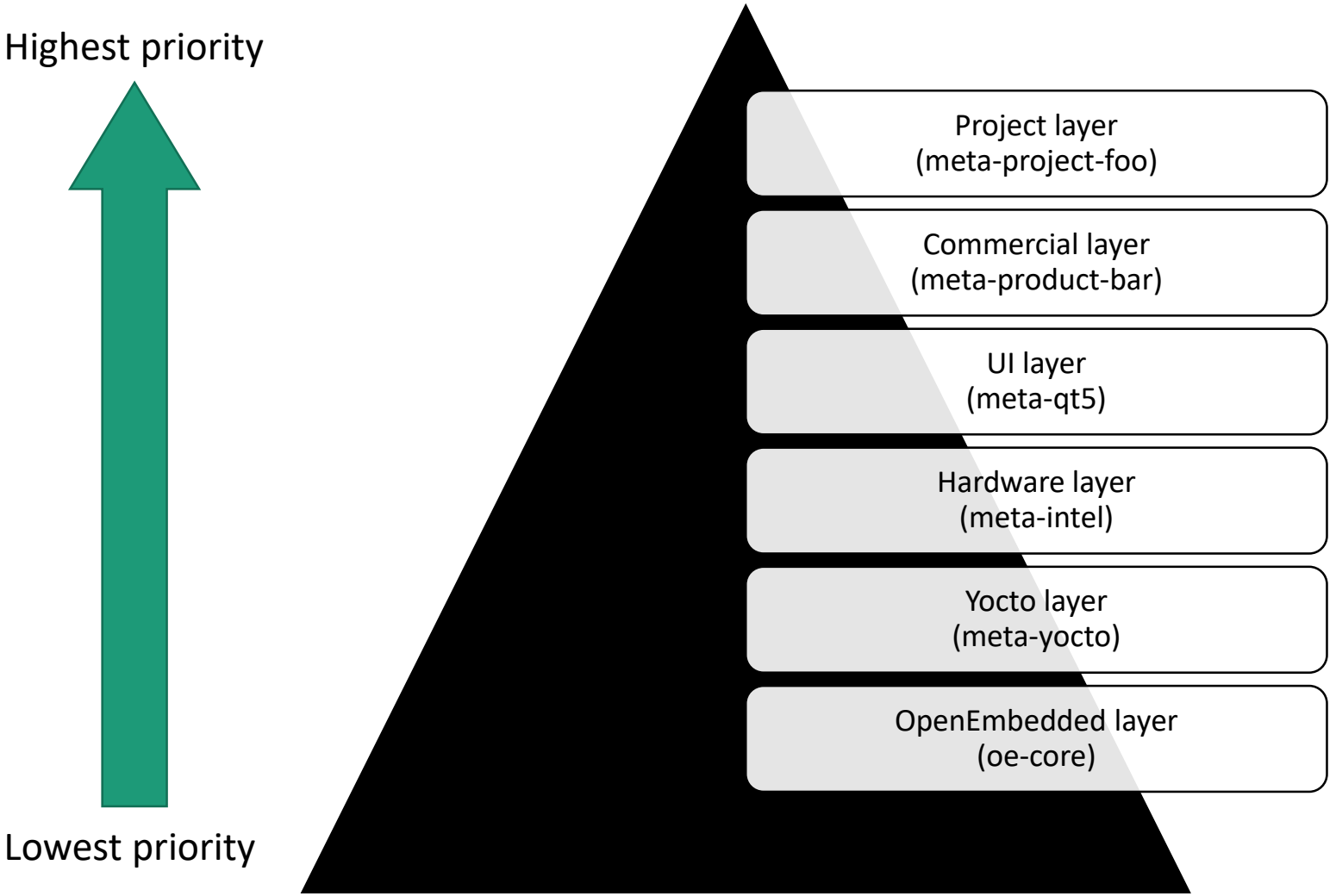

Open Embedded Architecture Workflow
Upstream Source
Output Packages
Metadata/Inputs
Process Steps (tasks)
Build System
Output Image Data
Upstream Project Releases
Local Projects
SCMs (optional)
Source Materials
User Configuration
Meta (.bb + patches)
Machine BSP Configuration
Policy Configuration
Source Fetching
Patch Application
Config/ Compile/ Autoconf as needed
Output Analysis for package splitting plus package relationships
.deb generation
.rpm generation
.ipk generation
QA Tests
Package Feeds
Image Generation
SDK Generation
Images
Application Development SDK

OK, great. But what are the cool parts?

Layered Approach

- Logical separation and aggregation of software components
- Hardware agnostic development
- Maintainability
- Reusability

Layered Approach



Development and Debugging

- Production image vs development/debug image
- Features grouped in `IMAGE_FEATURES`

Development and Debugging

```
48 # EXTRA_IMAGE_FEATURES allows extra packages to be added to the generated images
49 # (Some of these are automatically added to certain image types)
50 # "dbg-pkgs"      - add -dbg packages for all installed packages
51 #                   (adds symbol information for debugging/profiling)
52 # "dev-pkgs"      - add -dev packages for all installed packages
53 #                   (useful if you want to develop against libs in the image)
54 # "tools-sdk"     - add development tools (gcc, make, pkgconfig etc.)
55 # "tools-debug"   - add debugging tools (gdb, strace)
56 # "tools-profile" - add profiling tools (oprofile, exmap, lttng valgrind (x86 only))
57 # "tools-testapps" - add useful testing tools (ts_print, aplay, arecord etc.)
58 # "debug-tweaks"   - make an image for suitable of development
59 #                    e.g. ssh root access has a blank password
60 # There are other application targets too, see meta/classes/poky-image.bbclass
61 # and meta/packages/tasks/task-poky.bb for more details.
62
```

Development and Debugging

- Including dbg packages to the build – NOT FEASIBLE in the long run
- Creating a remote filesystem with debug symbols

```
# conf/local.conf

IMAGE_GEN_DEBUGFS = "1"
IMAGE_FSTYPES_DEBUGFS = "tar.bz2"
```

Development and Debugging

- Connect host GDB to the GDB server on the target
- Point to remote debugfs to find the debug symbols
- Profit

Testing

- Test image concept
- Automated runtime testing
- Virtualized or real hardware target

```
# conf/local.conf

TEST_IMAGE = "1"
```

Testing

- Set of predefined unit tests
  - ping, ssh, dmesg, syslog, etc.
- Easy to add custom tests

```
# conf/local.conf

TEST_SUITES_append = "ping ssh auto"
```

Testing

- Test results are available in the BitBake console
- Easy integration with the continous integration system

License Compliance

- How to maintain compliance with various open source licenses during the product’s lifecycle?

License Compliance

- Recipe variables
  - `LICENSE`
  - `LIC_FILES_CHKSUM`
- What happens if the `LICENSE` file changes between revisions?

```
6   SUMMARY = "A template C/C++ service"
7   DESCRIPTION = "A C/C++ source code repository blueprint"
8   AUTHOR = "Gordan Markuš <gordan.markus@pelagicore.com>"
9   HOMEPAGE = "https://github.com/Pelagicore/template-service"
10  LICENSE = "MPL-2.0"
11  LIC_FILES_CHKSUM = "file://LICENSE;md5=815ca599c9df247a0c7f619bab123dad"
12  PV = "1.0+git${SRCREV}"
13  PR = "r0"
```

License Compliance

- QtWebEngine license

```
SUMMARY = "QtWebEngine combines the power of Chromium and Qt"

# Read http://blog.qt.io/blog/2016/01/13/new-agreement-with-the-kde-free-qt-foundation/
LICENSE = "BSD & ( GPL-3.0 & The-Qt-Company-GPL-Exception-1.0 | The-Qt-Company-Commercial ) & ( LGPL-3.0 | The-Qt-Company-Commercial )"
LIC_FILES_CHKSUM = " \
    file://src/core/browser_context_qt.cpp;md5=b5193b7d68699260f3b40b201365c8d2;beginline=1;endline=38 \
    file://src/3rdparty/chromium/LICENSE;md5=0fca02217a5d49a14dfe2d11837bb34d \
    file://LICENSE.LGPL3;md5=8211fde12cc8a4e2477602f5953f5b71 \
    file://LICENSE.GPLv3;md5=88e2b9117e6be406b5ed6ee4ca99a705 \
    file://LICENSE.GPL3;md5=d32239bcb673463ab874e80d47fae504 \
    file://LICENSE.GPL3-EXCEPT;md5=763d8c535a234d9a3fb682c7ecb6c073 \
    file://LICENSE.GPL2;md5=b234ee4d69f5fce4486a80fdaf4a4263 \
"
```

License Compliance

- Bill of materials
  - Source code
  - License text
  - Modifications

License Compliance

- Blacklisting licenses using the `INCOMPATIBLE_LICENSE` variable
- Manually remove dependencies on or provide alternatives to components that are required

License Compliance

- GPLv3 software is still a „big no-no!” in certain industries ☹
- The goal is to prevent user modification on an embedded device

```
# conf/local.conf

INCOMPATIBLE_LICENSE = „ \
     GPL-3.0  \
     LGPL-3.0 \
     AGPL-3.0 \
"
```

License Compliance

- Exclusion using the meta-gplv2 layer

*„This layer contains a set of recipes corresponding to old, obsolete versions of software that are **GPLv2 licensed where the upstreams have moved to GPLv3 licenses**. These were part of OE-Core until it was realised they are a ticking timebomb with regard to security updates and general maintenance.”*

To summarize

- Yocto is **fun**
- It has **awesome** features to ease your development
- It is hard to learn but it is totally **worth it**
- **Share** your experience with a technology you like

```
$ systemctl stop yocto-presentation
```

Question time!

Contact: gordan.markus@gmail.com