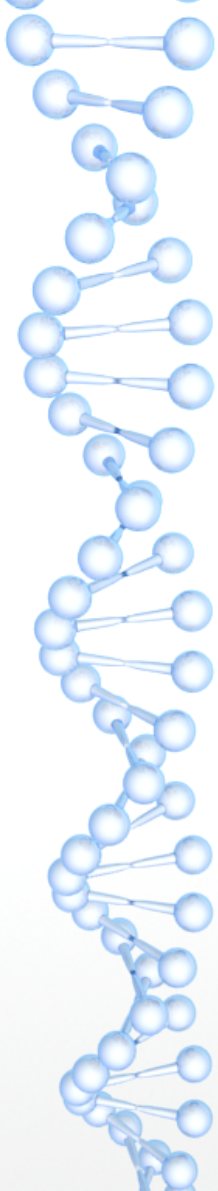


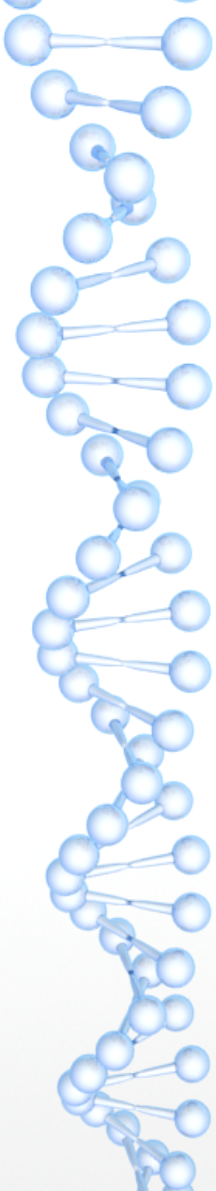
Yocto for PELUX

2018-03-29

The Yocto Project

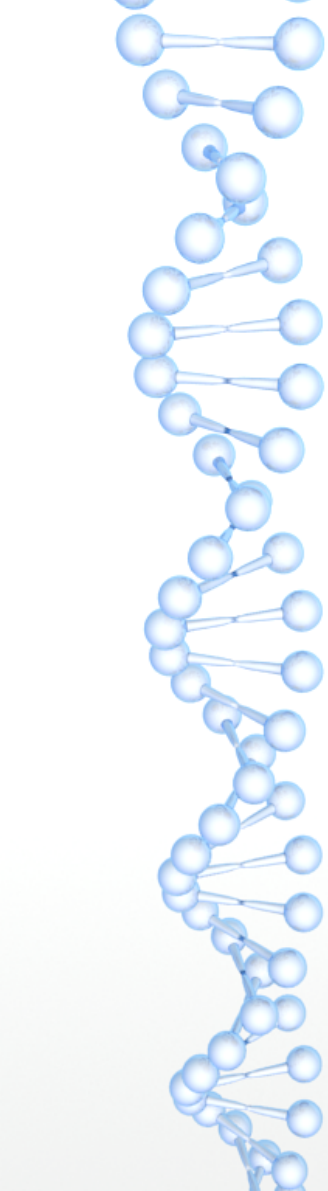
- *“The Yocto Project is an open source collaboration project that helps developers create custom Linux-based systems for embedded products, regardless of the hardware architecture.”*

- *https://www.yoctoproject.org/docs/current/ref-manual/ref-manual.html*



How does it work?

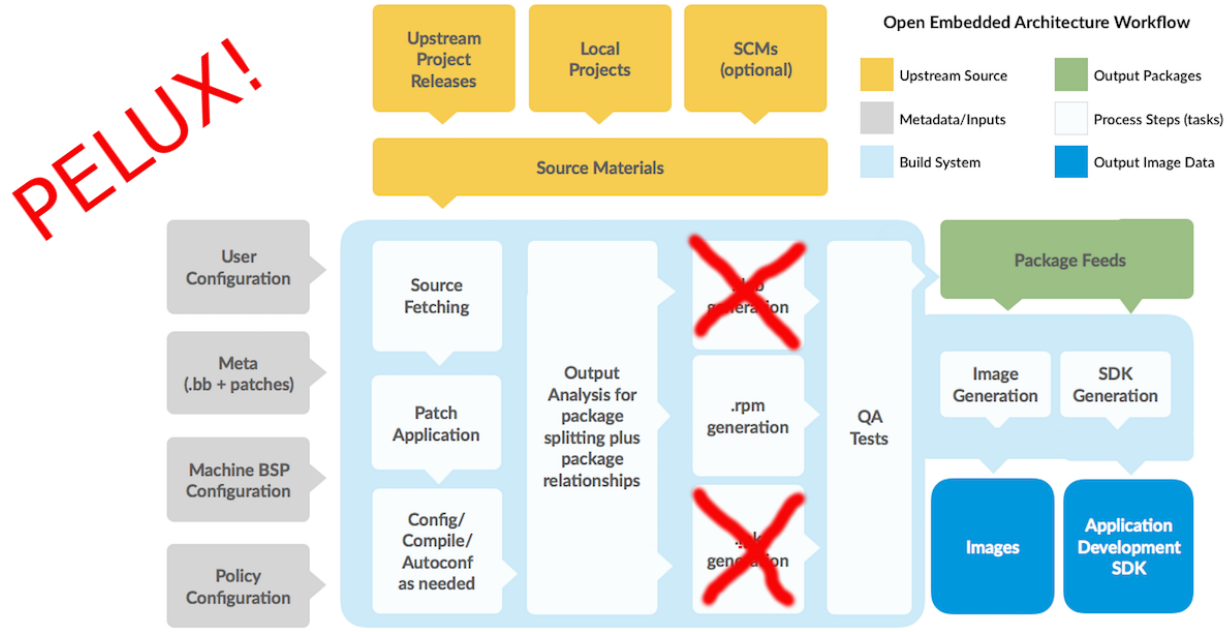
- Separates hardware configuration from software configuration
- A layered approach
- Highly customizable



TL;DR building PELUX

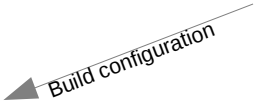
- We build images (core-image-pelux-minimal)
- Images pull in software defined by recipes
- Recipes reside in layers
- Some layers are maintained by us, most are not

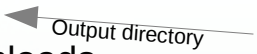
General flow



PELUX build directory structure

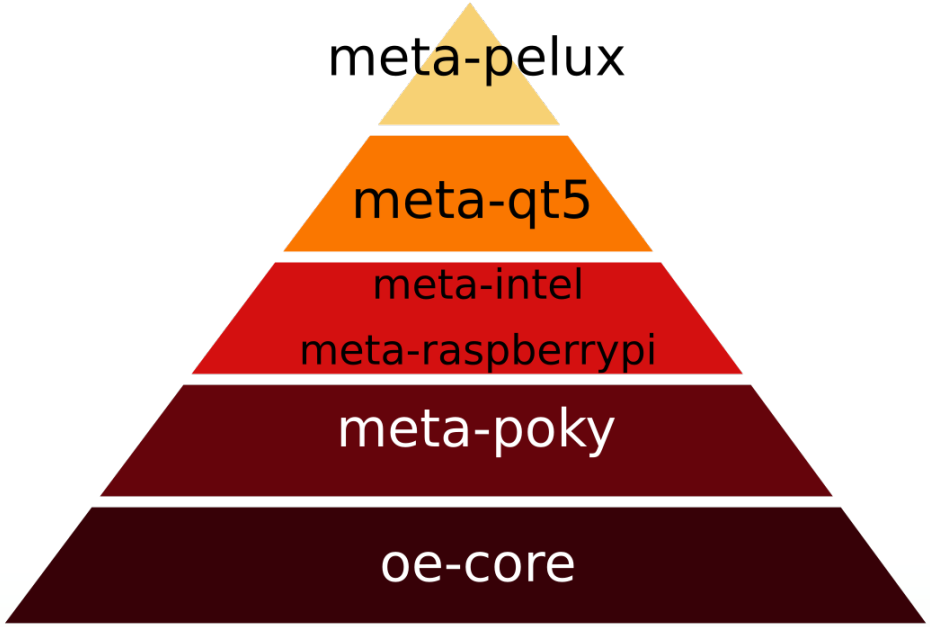
- yocto_pelux
  - Sources
    - meta-openembedded
    - meta-pelux
      - conf
      - layers
      - meta-rpi-extras
      - recipes-core
    - meta-raspberrypi
  - build
    - conf
      - local.conf
      - bblayers.conf
    - tmp
    - downloads





Source material, explained

Highest prio



Lowest prio

Layers

Distro Layer

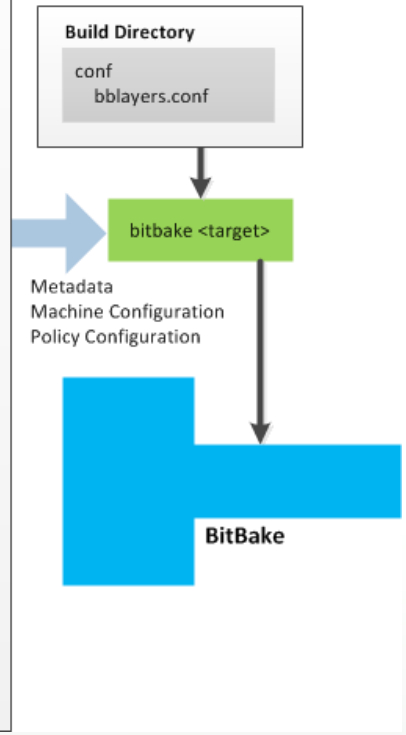
```
COPYING
README
classes
   *.bbclass
conf
   distro
      include
         *.inc
      <distro>.conf
   layer.conf
recipes-*
   <recipe>
      files
         defconfig
         *.h
         init
      <recipe>.bb
   <recipe>
      <recipe>.bbappend
```

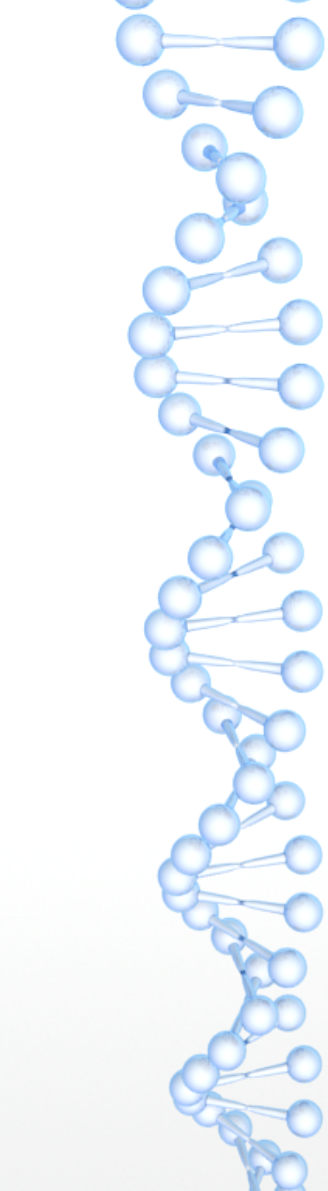
BSP Layer

```
COPYING
README
conf
   machine
      <machine>.conf
   layer.conf
recipes-bsp
   formfactor
      formfactor
         <machine>
            machconfig
      formfactor*.bbappend
recipes-core
   <recipe>
      files
      <recipe>.bbappend
recipes-graphics
   <recipe>
      <recipe>
         <machine>
            *.conf
      <recipe>.bbappend
recipes-kernel
   linux
      files
         <machine>.cfg
         <machine>.scc
      <recipe>.bbappend
```

Software Layer

```
COPYING
README
conf
   layer.conf
recipes-*
   <recipe>
      <recipe>.bb
   <recipe>
      <recipe>.bb
      files
         *.patch
```





Distributions

- High-level configuration for a build
- Sets a distro name and various other settings.
- Sets `DISTRO_FEATURES` variable
  - We'll get back to why this is important

Recipes

```
inherit qmake5

# Disable parallel make until .pro files properly set dependencies
PARALLEL_MAKE = "-j1"

OE_QMAKE_PATH_HEADERS = "${OE_QMAKE_PATH_QT_HEADERS}"
DEPENDS += "qtbase qtdeclarative"

SRC_URI = "git://github.com/Pelagicore/qmldevinfo;branch=master;protocol=https"
SRCREV = "50a305aa42a8e542cac66b843fdbfaff08d58bf0"

LICENSE = "MPL-2.0"
LIC_FILES_CHKSUM = "file://LICENSE.txt;md5=9741c346eef56131163e13b9db1241b3"

PV = "1.0+git${SRCREV}"
PR = "r1"

S = "${WORKDIR}/git/"
B = "${WORKDIR}/build/"

FILES_${PN} += "/usr/lib/qt5/qml/com/pelagicore/qmldevinfo/*"
FILES_${PN}-dbg += "/usr/lib/qt5/qml/com/pelagicore/qmldevinfo/.debug"

PACKAGES = "${PN}-dbg ${PN}"
```

PACKAGECONFIG

```
SUMMARY = "Canonical libwebsockets.org websocket library"
HOMEPAGE = "https://libwebsockets.org/"

inherit cmake pkgconfig

PACKAGECONFIG ?= "libuv client server http2 ssl"
PACKAGECONFIG[client] = "-DLWS_WITHOUT_CLIENT=OFF,-DLWS_WITHOUT_CLIENT=ON,"
PACKAGECONFIG[http2] = "-DLWS_WITH_HTTP2=ON,-DLWS_WITH_HTTP2=OFF,"
PACKAGECONFIG[ipv6] = "-DLWS_IPV6=ON,-DLWS_IPV6=OFF,"
PACKAGECONFIG[libev] = "-DLWS_WITH_LIBEV=ON,-DLWS_WITH_LIBEV=OFF,libev"
PACKAGECONFIG[libuv] = "-DLWS_WITH_LIBUV=ON,-DLWS_WITH_LIBUV=OFF,libuv"
PACKAGECONFIG[server] = "-DLWS_WITHOUT_SERVER=OFF,-DLWS_WITHOUT_SERVER=ON,"
PACKAGECONFIG[ssl] = "-DLWS_WITH_SSL=ON,-DLWS_WITH_SSL=OFF,openssl"
PACKAGECONFIG[testapps] = "-DLWS_WITHOUT_TESTAPPS=OFF,-DLWS_WITHOUT_TESTAPPS=ON,"

PACKAGES =+ "${PN}-testapps"

FILES_${PN}-dev += "${libdir}/cmake"
FILES_${PN}-testapps += "${datadir}/libwebsockets-test-server/*"
```

Checking DISTRO_FEATURES

```
PACKAGECONFIG_GL ?= "$
{@bb.utils.contains('DISTRO_FEATURES', 'opengl', 'gl', '',
d)}"
```

- If DISTRO_FEATURES contains “opengl”, then add “gl” to PACKAGECONFIG_GL, otherwise add an empty string

- This is a super common pattern
- Note the inline python code!
- One can set REQUIRED_DISTRO_FEATURES for mandatory ones

bbappend

- Append to existing recipes!

  meta-pelux/recipes-graphics/pango/pango_%.bbappend

```
# GObject introspection for pango needs to run some commands on the
# native architecture, and uses qemu for this. For aarch64, these
# commands cause qemu to crash, so we disable introspection.
EXTRA_OECONF_aarch64 += "--disable-introspection"
```

- Appends are applied according to layer priority

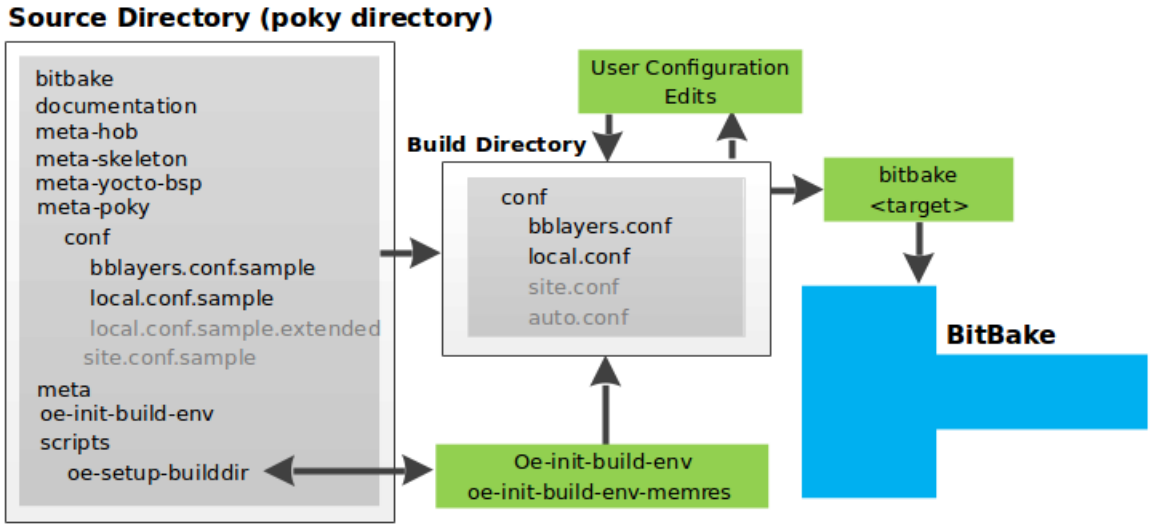
Where does the build happen?

- build/tmp/work/<arch>/<recipe>/<version>/
  - temp/
    - log.do_configure, log.do_compile etc
    - run.do_configure, run.do_compile etc
  - git/
    - If the source is in a git repo
  - build/
    - This is where stuff is compiled

Where is the built software?

- Usually, it is in:
  - tmp/deploy/rpm/<arch>/<recipe>.rpm
- Or you can find it in the build directory
  - tmp/work/<arch>/<recipe>/<version>/deploy-rpms/<arch>/
- Check what goes in what package (recipes create multiple packages)
  - tmp/work/<arch>/<recipe>/<version>/packages-split/

User configuration



Local configuration (local.conf)

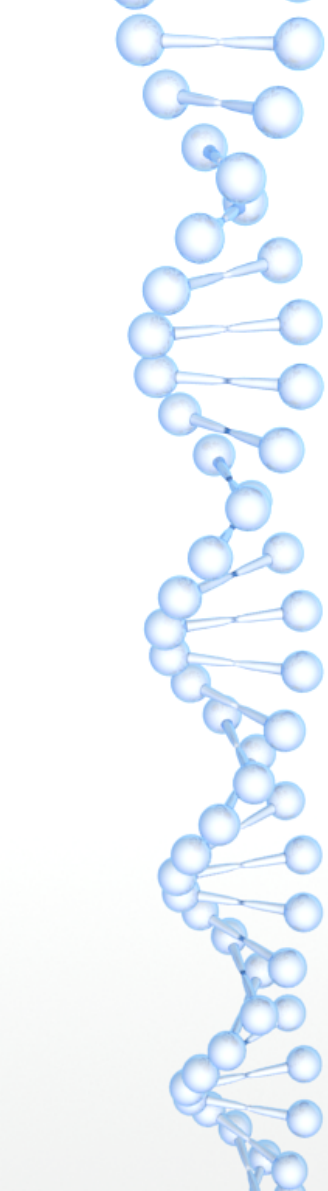
```
CONF_VERSION = "1"
DL_DIR = "${TOPDIR}/downloads"

MACHINE = “intel-corei7-64”
SDKMACHINE = "x86_64"
DISTRO = "pelux"

# Target Static IP address, Override this to configure a static
# ip address for development purposes such as poky ssh and ping test.
STATIC_IP_ADDRESS = ""

BB_NUMBER_THREADS ?= "${@oe.utils.cpu_count()}"
PARALLEL_MAKE ?= "-j ${@oe.utils.cpu_count()}"

PACKAGE_CLASSES ?= "package_rpm"

BB_DANGLINGAPPENDS_WARNONLY = "1"
```

Layer configuration (bblayers.conf)

```
BBFILES  ?= ""
BBLAYERS ?= "                                          \
   ${BSPDIR}/sources/poky/meta                         \
   ${BSPDIR}/sources/poky/meta-poky                    \
   ${BSPDIR}/sources/poky/meta-yocto-bsp               \
   ${BSPDIR}/sources/meta-openembedded/meta-oe         \
   ${BSPDIR}/sources/meta-openembedded/meta-networking \
   ${BSPDIR}/sources/meta-openembedded/meta-python     \
   ${BSPDIR}/sources/meta-openembedded/meta-multimedia \
   ${BSPDIR}/sources/meta-swupdate                     \
   ${BSPDIR}/sources/meta-ivi/meta-ivi                 \
   ${BSPDIR}/sources/meta-ivi/meta-ivi-bsp             \
   ${BSPDIR}/sources/meta-pelux                        \
   ${BSPDIR}/sources/meta-virtualization               \
   ${BSPDIR}/sources/meta-bistro                       \
   ${BSPDIR}/sources/meta-template                     \
"
```

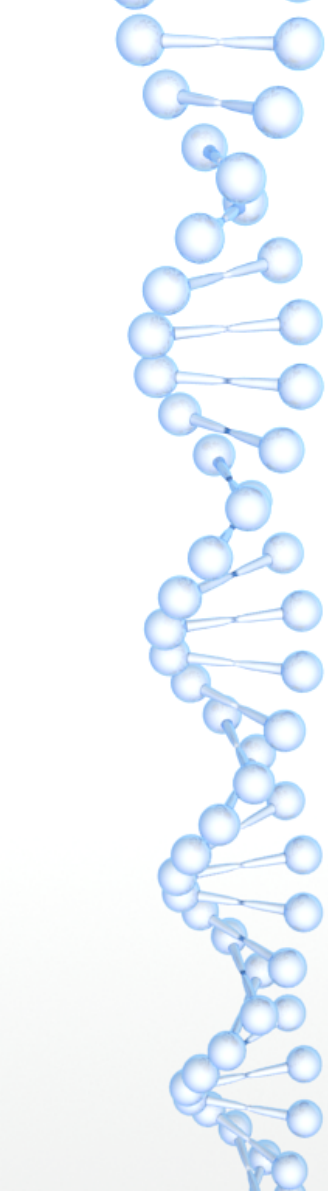
Where should I put my configuration?

- If you want to change a recipe, put the it in the recipe file (.bb) or in an append file (.bbappend).
- If it is a global option
  - distro.conf (if you are in charge of the distro)
  - local.conf (if you are not in charge of the distro)
  - image recipe / append



What happens when I build?

- Parse all your recipes and configurations
- Apply all bbappends
- Build dependency graph
- Build software in order of dependencies
  - Cross-toolchain first
  - Kernel etc next
  - Any other software
  - Image generation



Standard build steps

- Fetch
- Unpack
- Patch
- Configure
- Compile
- Install
- Package
- QA

Detailed description, please read:
https://www.yoctoproject.org/docs/current/ref-manual/ref-manual.html#bitbake-dev-environment

The manifest

- Pelux uses the “repo” tool from the Android project to track revisions of all layers

```
$ repo init -u <url> -b <branch> -m <manifest>
$ repo sync
```

- Practices:
  - Always follow a specific commit, not a branch
  - Sync up on the same yocto release (rocko, pyro etc)
  - Use release branches in the manifest repo as well

Manifest example

http://github.com/Pelagicore/pelux-manifests/

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <remote fetch="git://github.com/" name="github"/>

  <project name="GENIVI/meta-ivi"
           path="sources/meta-ivi"
           remote="github"
           revision="5243d83ac2ef13d117065edae8e4f484e7e4f373"
           upstream="master"/>

  <project name="Pelagicore/meta-bistro"
           path="sources/meta-bistro"
           remote="github"
           revision="b84bd307bb93bcb10f19de04a2b04d26cdce2ea7"
           upstream="master"/>
</manifest>
```