


EMBEDDED LINUX WITH



Julio Faracco <jcfaracco@gmail.com>

/ME

- Kernel Linux Engineer at  **Red Hat**
- Kernel Team ->
 - Core Kernel ->
 - **Red Hat Kernel Maintainers ->**
 - Real Time RHEL Kernel ->
 - (*) **Automotive Kernel (ARM boards)**

PLAYGROUND



<https://f1tenth.org/build.html>

WHAT IS LINUX?

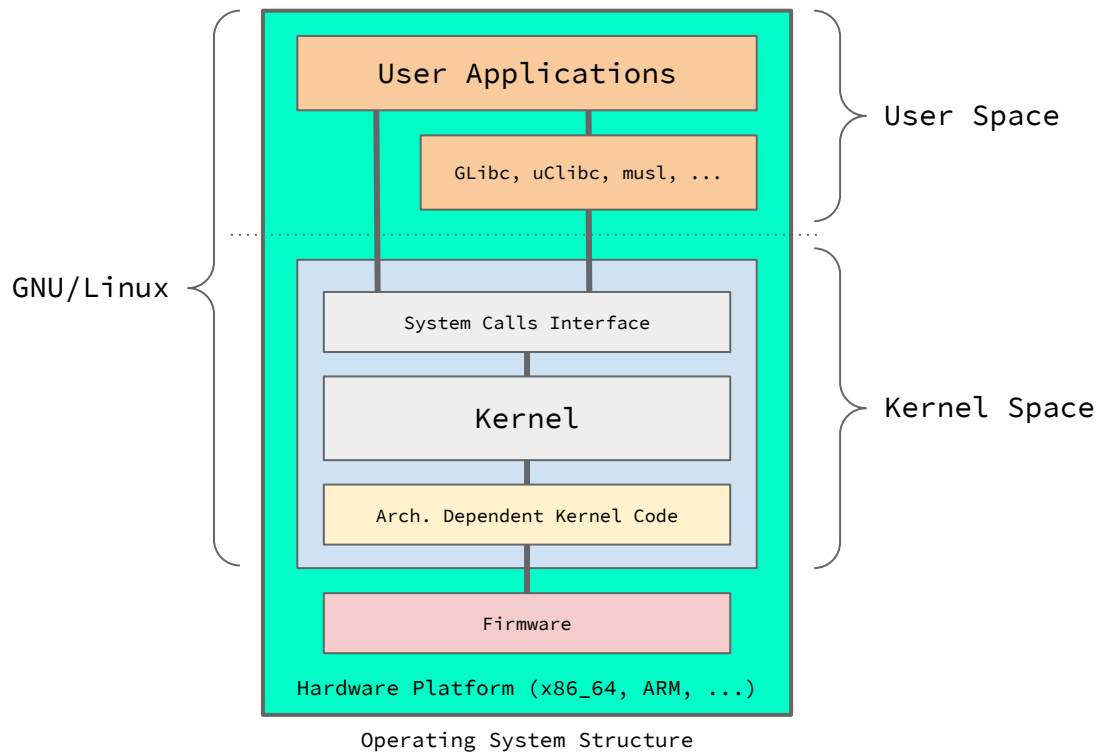


WHAT IS LINUX?

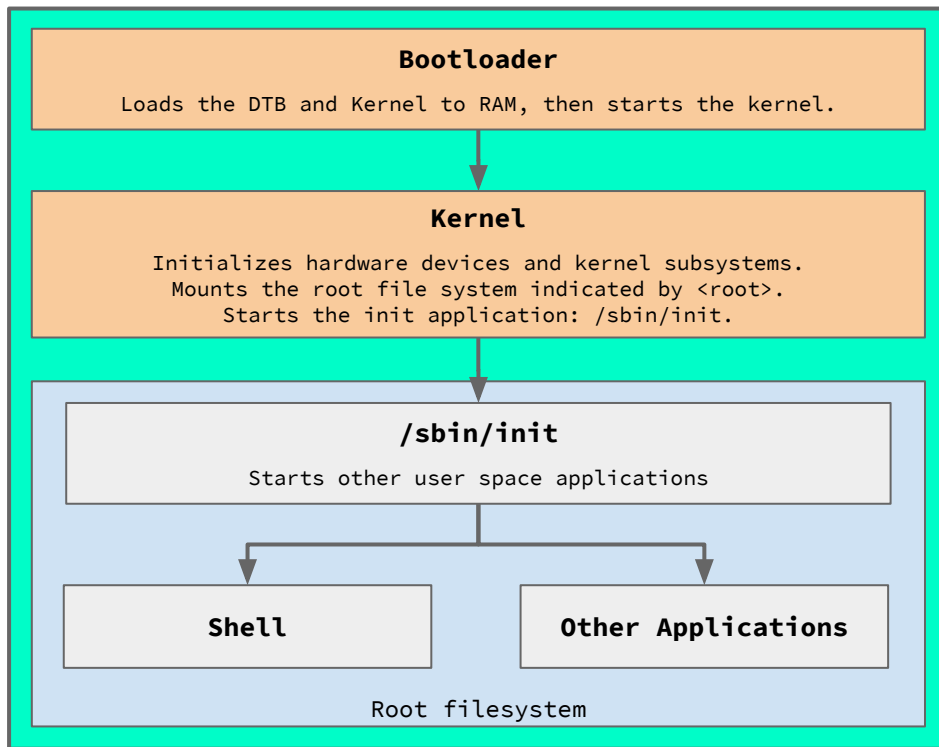
- Linux is one of the most popular (and used) operating system distribution.
- It is open source and “totally” customizable: the same source code runs in super computer and small boards.
- So, what is GNU/Linux? It is a collection of Linux Kernel and other supporting tools like ***glibc***.
- Extra: Linux is a huge project. People and companies usually take over some drivers and trees to maintain by themselves.



WHAT IS LINUX?



WHAT IS LINUX?



Linux Kernel Boot



WHAT IS LINUX?

Linux () arm
Startup finished in 1.170s (kernel) + 37.765s (userspace) = 38.936s



- Why is it important to understand the concepts behind Linux in an embedded world?
- Your Smart TV taking ~39 seconds to start and open your favourite movie app.





WHY SHOULD I CARE?

- Default Ubuntu 21.10 for ARM 64 bits has **~10000 configurations** defined.
- All of them are really required for your project? Should we include more or reduce them?
- What is the solution? Build your own custom kernel?

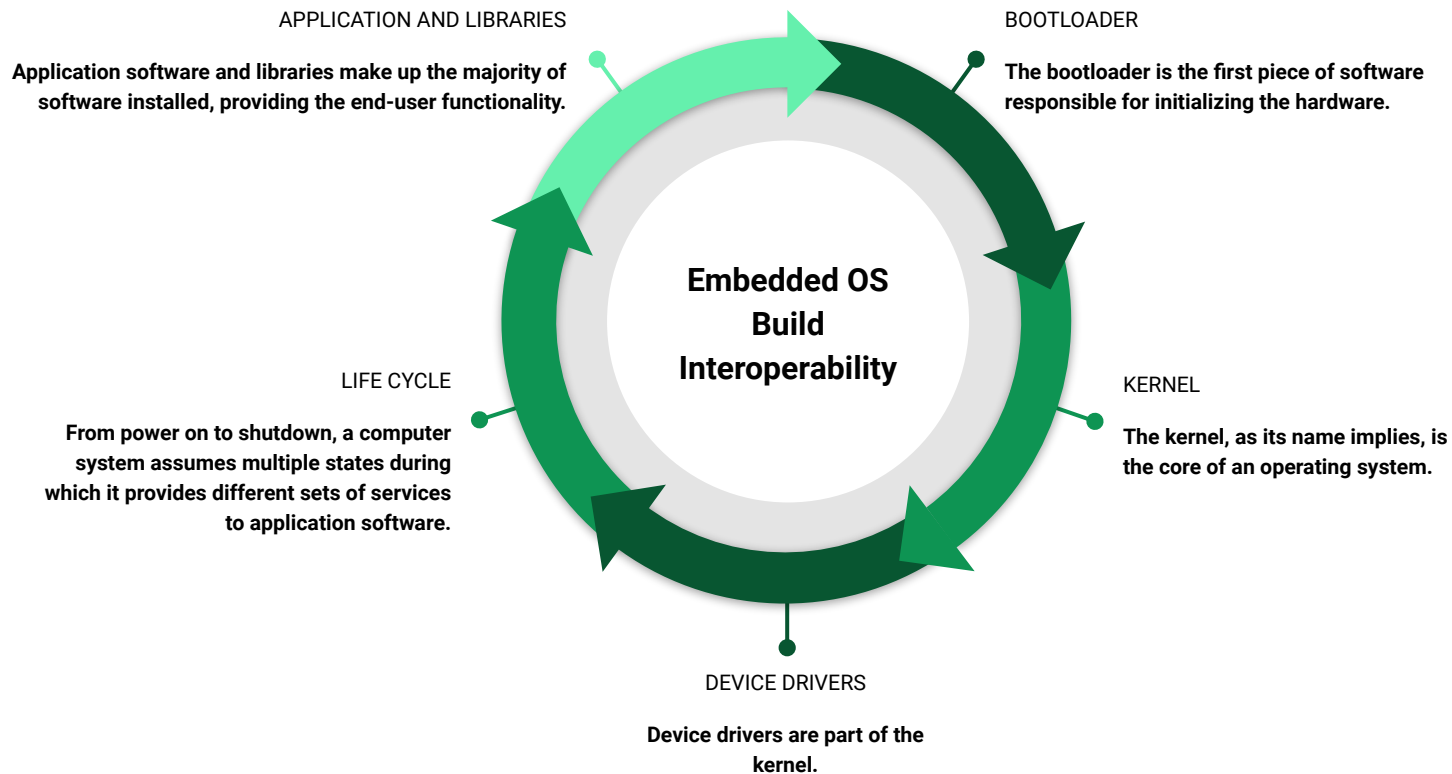


WHY SHOULD I CARE?

- How can I turn my project into a product? Products requires Long Term Support (LTS). How can I handle that?
- What if I need to extend my product to support other platforms... Should I handle that configs again? Probably...
- Have you ever seen a buildroot script to automate a customized Linux for Embedded?
- Build versus toolchain changes. What about applications and libraries.



WHY SHOULD I CARE?



WHAT SHOULD I DO
THEN?



THE YOCTO PROJECT:



It's not an embedded Linux Distribution. It creates a custom one for you, regardless of the hardware architecture¹.

[1] <https://www.yoctoproject.org/>

WHAT IS YOCTO?

- Open Source Project lead by Linux Foundation.
- With a Commercial Product Development focus.
- A whole Ecosystem of collaboration.
 - Industry can work collaboratively to publish their own BSPs (?).
- Yocto uses OpenEmbedded components like bitbake and openembedded-core, taking advantage of recipes, classes and other metadatas.



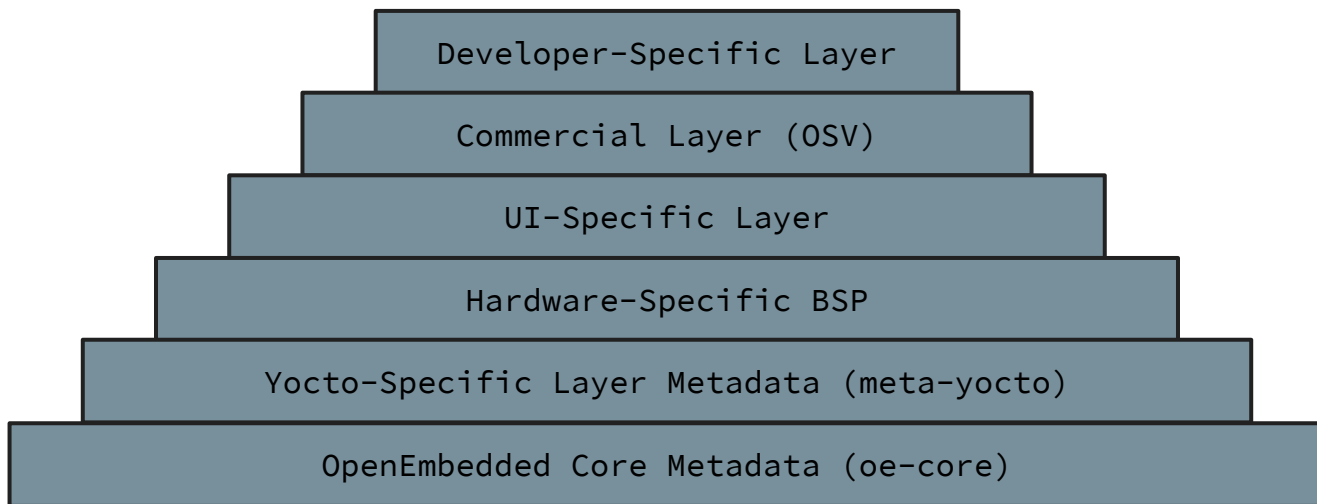
~~WHAT IS YOCTO?~~ (DETAILS)

- Build System composed by:
 - A poky reference system host at:
 - <https://git.yoctoproject.org/git/poky>
 - Bitbake: build engine for embedded:
 - <https://github.com/openembedded/bitbake>
 - OpenEmbedded Core (as previously mentioned):
 - Recipes, classes and other metadatas.
 - Yocto core BSPs and layers.



~~WHAT IS YOCTO?~~ (DETAILS)

- The build system can be composed as layers too:

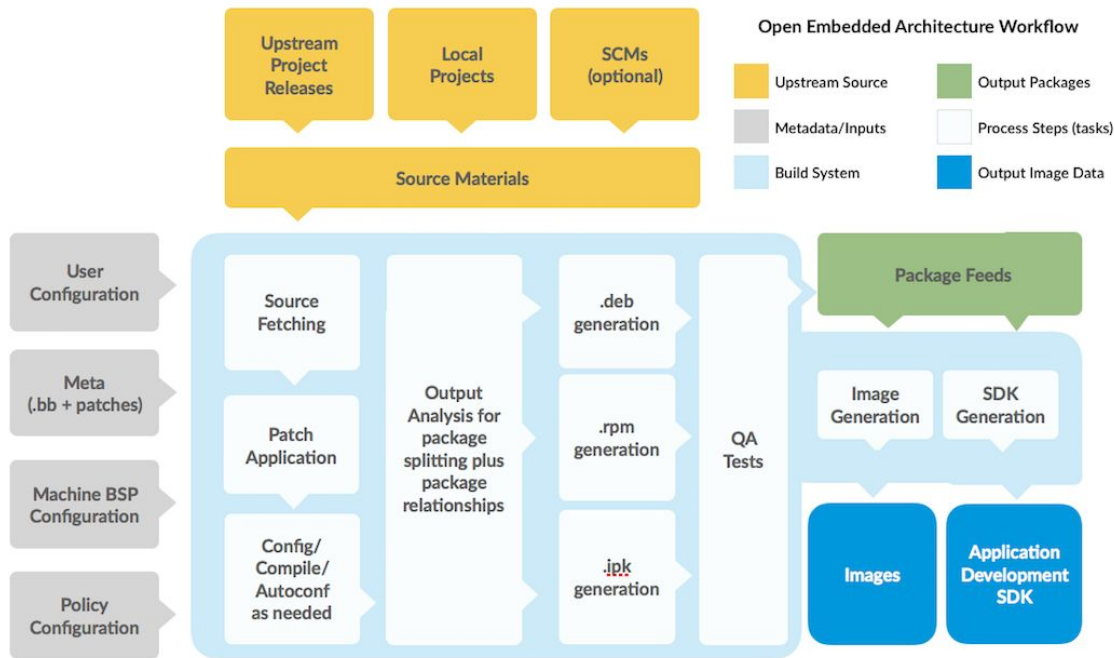


~~WHAT IS YOCTO?~~ (DETAILS)

- Best Practice and Common Behaviors: layers should be grouped by functionality or variations.
 - Custom Toolchains (compilers, debuggers, profiling tools)
 - Distribution specifications (i.e. meta-yocto)
 - BSP/Machine settings (i.e. meta-yocto-bsp)
 - Functional areas (selinux, networking, crypto, etc)
 - Project specific changes



WHAT IS YOCTO? (DETAILS)





~~WHAT IS YOCTO?~~ (DETAILS)

- For example: one kernel base for several flavors/archs.



~~WHAT IS YOCTO?~~ (DETAILS)

- **User Configuration (*.conf):** global definition of variables:
 - **meta/conf/bitbake.conf:** bitbake config variables.
 - **build/conf/bblayers.conf:** layers config variables.
 - ***/conf/layers.conf:** one config per layer.
 - **build/conf/local.conf:** local user-defined variables.
 - **meta-yocto/conf/distro/poky.conf:** yocto policy config variables.
 - **meta-yocto-bsp/conf/machine/beagleboard.conf:** machine-specific variables.

User
Configuration

Meta
(.bb + patches)

Machine BSP
Configuration

Policy
Configuration



~~WHAT IS YOCTO?~~ (DETAILS)

- Metadata and Packages (*.bb + patches):

- Recipe for building packages.
- **meta/recipes-core/busybox/busybox-1.20.2:**
includes patches, also could include extra files to install.
- Recipes inherit the system configuration and adjust it to describe how to build and package the software.
- Recipes can be extended and enhanced through append-files from other layers.
- Yocto Project and OpenEmbedded recipes structures are compatible to each other

User
Configuration

Meta
(.bb + patches)

Machine BSP
Configuration

Policy
Configuration



~~WHAT IS YOCTO?~~ (DETAILS)

- Machine BSP Configuration:

- Layers contain extensions and customizations to base system.
- Can include image customizations, additional recipes, modifying recipes, adding extra configuration:
 - Really just another directory to look for recipes in.
 - Added to the **BBLAYERS** variable in **build/conf/bblayers.conf**.
- BSPs are layers that add machine settings and recipes.
- Machine settings are specified in a layer's **conf/machine/xxx.conf** file(s).
- Machine configuration refers to kernel sources and may influence some userspace software.
- Compatible with OpenEmbedded too.

User
Configuration

Meta
(.bb + patches)

Machine BSP
Configuration

Policy
Configuration



~~WHAT IS YOCTO?~~ (DETAILS)

- Policy Configuration:

- Defines distribution/system wide policies that affect the way individual recipes are built:
 - Set alternative preferred versions of recipes.
 - Enable/disable LIBC functionality (i.e. i18n).
 - Enable/disable features (i.e. pam, selinux).
 - Configure specific package rules.
 - Adjust image deployment settings.
- Enabled via the DISTRO setting.
- Four predefined settings:
 - **poky-bleeding**: Enable a bleeding edge packages.
 - **poky**: Core distribution definition, defines the base.
 - **poky-lsb**: enable items required for LSB support.
 - **poky-tiny**: construct a smaller then normal system.

User
Configuration

Meta
(.bb + patches)

Machine BSP
Configuration

Policy
Configuration



~~WHAT IS YOCTO?~~ (DETAILS)

- Source Fetching:

- Recipes call out the location of all sources, patches and files.
- These may exist on the internet or be local. (See **SRC_URI** in the ***.bb** files).
- Bitbake can get the sources from git, svn, bzip, tarballs, and many more*.
- Versions of packages can be fixed or updated automatically (Add **SRCREV_pn-PN = "\${AUTOREV}"** to **local.conf**).
- The Yocto Project mirrors sources to ensure source reliability.
- SCM includes all: http, ftp, https, git, svn, perforce, mercurial, bzip, cvs, osc, repo, ssh, and svn and the unpacker can cope with tarballs, zip, rar, xz, gz, bz2, and so on.



~~WHAT IS YOCTO?~~ (DETAILS)

– Source Unpacking and Patching:

- Once sources are obtained, they are extracted.
- Patches are applied in the order they appear in **SRC_URI**:
 - Quilt is used to apply patches.
- This is where local integration patches are applied.
- We encourage all patch authors to contribute their patches upstream whenever possible
- Patches are documented according to the patch guidelines:
 - http://www.openembedded.org/wiki/Commit_Patch_Message_Guidelines



~~WHAT IS YOCTO?~~ (DETAILS)

- **Configure / Compile / Instal processes:**

- Autoconf can be triggered automatically to ensure latest libtool is used:

```
DESCRIPTION = "GNU Helloworld application"
SECTION = "examples"
LICENSE = "GPLv2+"
LIC_FILES_CHKSUM = "file://COPYING;md5=751419260aa954499f7abaabaa882bbe"
PR = "r0"

SRC_URI = "${GNU_MIRROR}/hello/hello-${PV}.tar.gz"

inherit autotools gettext
```

- CFLAGS can be set:

```
CFLAGS_prepend = "-I ${S}/include "
```

- Install task to set modes, permissions, target directories, done by "pseudo":

```
do_install () {
    oe_runmake install DESTDIR=${D} SBINDIR=${sbindir} MANDIR=${mandir}
}
```

Source
Fetching

Patch
Application

Config/
Compile/
Autoconf
as needed

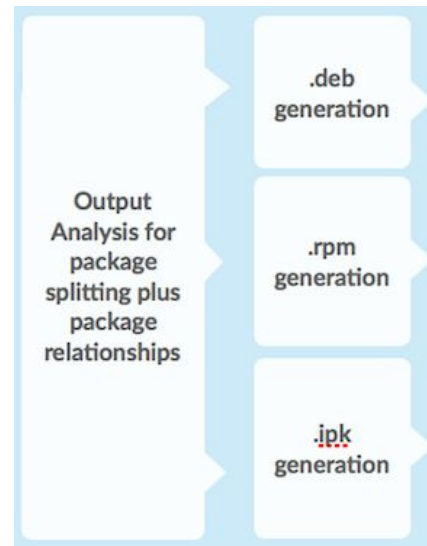


~~WHAT IS YOCTO?~~ (DETAILS)

- Output Analysis and Packaging:

- Categorize generated software (debug, dev, docs, locales).
- Split runtime and debug information.
- Once the previous processes are completed, packaging starts.
- The most popular package formats are supported: RPM, Debian, and ipk:
 - Set **PACKAGE_CLASSES** in **conf/local.conf**.
- You can split into multiple packages using **PACKAGES** and **FILES** in a ***.bb** file:

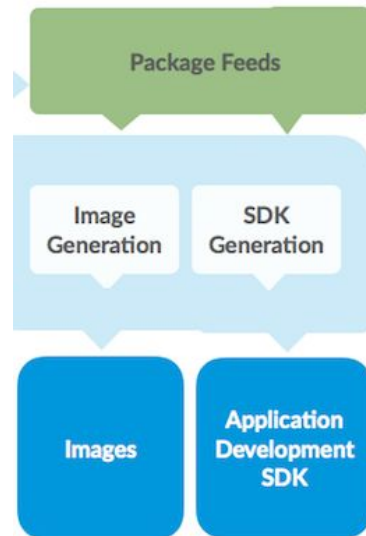
```
PACKAGES += "sxpm cxpm"  
FILES_cxpm = "${bindir}/cxpm"  
FILES_sxpm = "${bindir}/sxpm"
```



~~WHAT IS YOCTO?~~ (DETAILS)

- Image Generation branch:

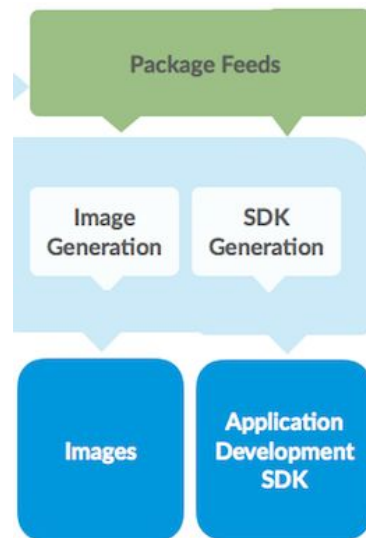
- Images are constructed using the packages built earlier and put into the Package Feeds:
- Decisions of what to install on the image is based on the minimum defined set of required components in an image recipe. This minimum set is then expanded based on dependencies to produce a package solution.
- Images may be generated in a variety of formats (tar.bz2, ext2, ext3, jffs, etc...)
- Uses for these images:
 - Live Image to boot a device
 - Root filesystem for QEMU emulator
 - Sysroot for App development



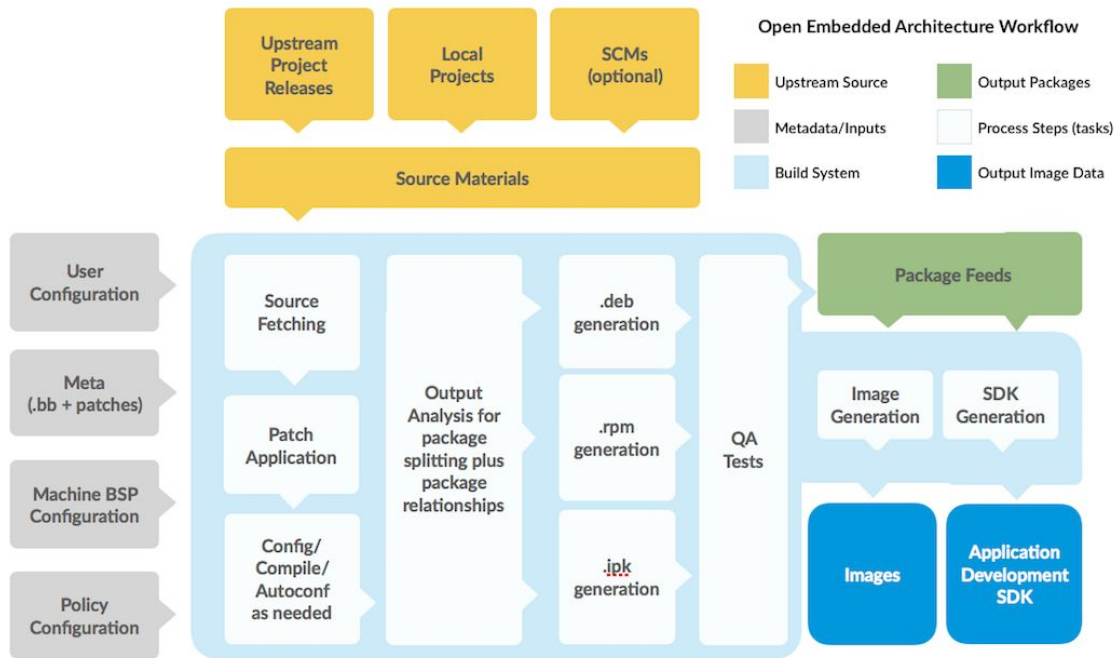
~~WHAT IS YOCTO?~~ (DETAILS)

– SDK Generation branch:

- A specific SDK recipe may be created. This allows someone to build an SDK with specific interfaces in it. (i.e. meta-toolchain-gmae).
- SDK may be based on the contents of the image generation.
- SDK contains native applications, cross toolchain and installation scripts
- May be used by the Eclipse Application Developer Tool to enable App Developers.
- May contain a QEMU target emulation to assist app developers.



WHAT IS YOCTO? (DETAILS)





TIME FOR QUICK DEMOS?

Obs: it takes ~5 hours to generate a simple image

WHAT IS TOASTER?

How to start Toaster?

```
$ cd yocto
$ source oe-init-build-env
$ pip3 install --user -r bitbake/toaster-requirements.txt
$ source toaster start
```

```
# Enable access to admin page
```

```
$ ../bitbake/lib/toaster/manage.py createsuperuser
```

```
# Visit https://localhost:8000
```

```
# To stop Toaster (on the same terminal where start command was launched):
```

```
$ source toaster stop
```

FINAL THOUGHTS...

Should I use Yocto?

If you are working on a big project with different changes and many collaborators the answer is probably yes. You have methods to adapt your OS stacks to patch and create a more collaborative environment without pain.

Maintaining an Operating System pipeline requires a lot of expertise of the hardware and software functionalities.

FINAL THOUGHTS...

Other good reasons to use...

Standard formats with layers, recipes and many others: easy to scalate.

Partners and board vendors maintain BSPs by themselves.

Test and build pipeline according best software development practices: a kind of continuous integration.

QUICK START

Yocto Project Website:

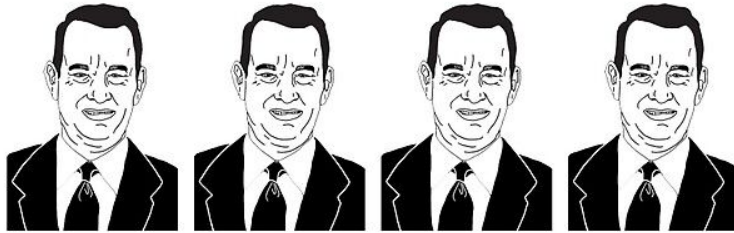
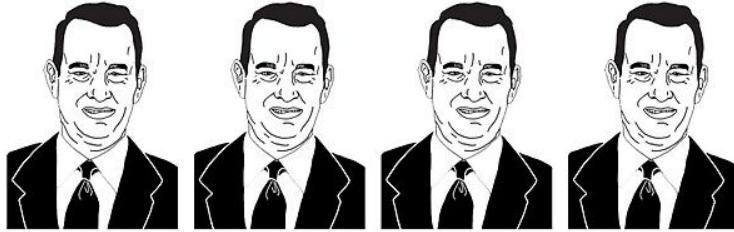
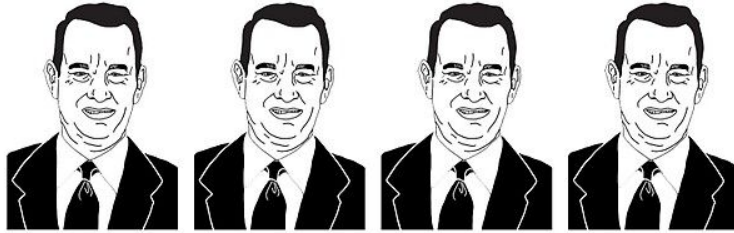
<https://www.yoctoproject.org>

Yocto Project Wiki:

https://wiki.yoctoproject.org/wiki/Main_Page

OpenEmbedded Website/Wiki:

http://www.openembedded.org/wiki/Main_Page



T.HANKS A LOT