yocto PROJECT

Introduction to the Yocto Project

Mark Hatle & Bruce Ashfield
WIND RIVER

Yocto Project Dev Day | Barcelona, Spain
08-Nov-2012

Agenda

- What is the Yocto Project?
- Who is the Yocto Project?
- Poky (Bitbake/OpenEmbedded-Core)
  - Getting Started
  - Build Configuration
  - Build System Workflow
- QA

[yoc-to]

The smallest unit of measure, equal to one septillionth (10^{-24}).

What is the Yocto Project?

- **Open source project with a strong community**
- **A collection of embedded projects and tooling**
  - Place for Industry to publish BSPs
  - Application Development Tools including Eclipse plug-ins and emulators
- **Key project is the reference distribution build environment (Poky)**
  - Complete Linux OS with package metadata
  - Releases every 6 months with latest (but stable) kernel (LTSI), toolchain, and package versions
  - Full documentation representative of a consistent system

It's not an embedded Linux distribution – it creates a custom one for you

The State of Embedded Linux…

- **DIY/Roll-Your-Own or modified traditional distro:**
  - Long Term Maintenance is difficult
  - Upstream changes are difficult to track
  - Not embedded friendly
  - Licensing issues
  - No commercial embedded support
- **Commercial/Community Embedded Linux:**
  - Too many competing systems
  - Incompatible distributions/build systems
- ***Developers spend lots of time porting or making build systems***
- ***Leaves less time/money to develop interesting software features***

What the Yocto Project Provides

- **The industry needed a common build system and core technology**
  - Bitbake and OpenEmbedded build system
- **The benefit of doing so is:**
  - Designed for the long term
  - Designed for embedded
  - Transparent Upstream changes
  - Vibrant Developer Community
- ***Less time spent on things which don't make money (build system, core Linux components)***
- ***More time spent on things which do make money (app development, product development, …)***

Who is the Yocto Project?

- **Advisory Board and Technical Leadership:**
  - Organized under the Linux Foundation
  - Individual Developers
  - Embedded Hardware Companies
  - Semiconductor Manufacturers
  - Embedded Operating System Vendors
  - OpenEmbedded / LTSI Community

Supporting Organizations

Sidebranch – O.S. Systems – Gumstix
Timesys – Tilera – Ridge Run – Dell
Secret Lab Technologies – Panasonic
NetLogic Microsystems – Mindspeed
LSI Logic – Freescale Semiconductor
Cavium Networks

Member Organizations

See http://www.yoctoproject.org/ecosystem for up-to-date information

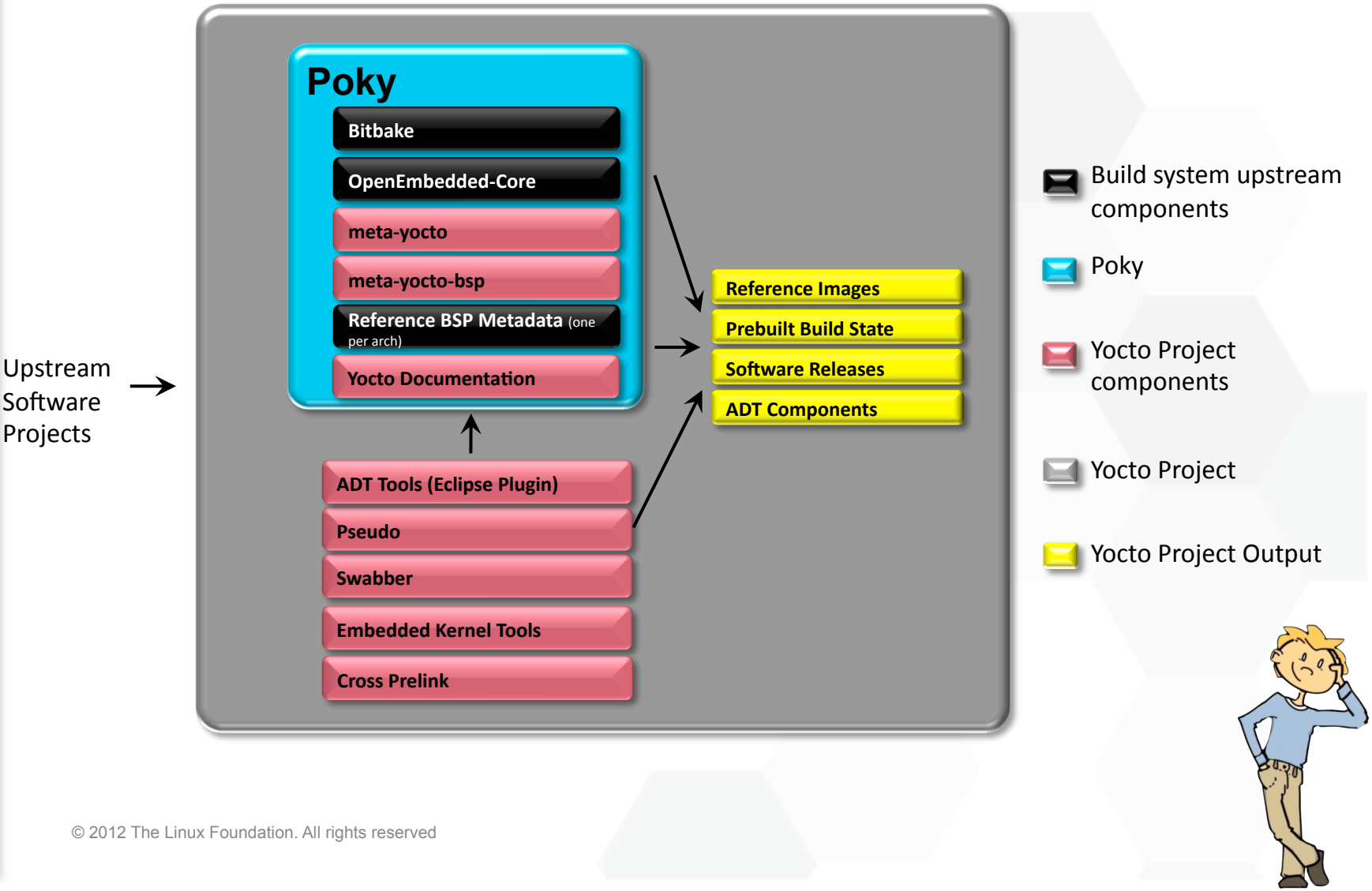
Why Should a Developer Care?

- **Build a complete Linux system –from source– in about an hour (about 90 minutes with X)**
  - Multiple cores (i.e. quad i7)
  - Lots of RAM (i.e. 16 GB of ram or more)
  - Fast disk (RAID, SSD, etc…)
- **Start with a validated collection of software (toolchain, kernel, user space)**
- **We distinguish app developers from system developers and we support both**
- **Access to a great collection of app developer tools (performance, debug, power analysis, Eclipse)**

Why Should a Developer Care? (cont)

- **Supports all major embedded architectures**
  - x86, x86-64, ARM, PPC, MIPS
  - Coming soon, MIPS64 and ARM Arch 64
- **Advanced kernel development tools**
- **Layer model encourages modular development, reuse, and easy customizations**
- **Compatibility program that is used to encourage interoperability and best practices**

Yocto Project Provides Embedded Tools, Best Practices, and Reference Implementation



Why not just use OpenEmbedded?

- **OpenEmbedded is an Open Source project focused on enabling cross-compiled systems.**
  - *Not a reference distribution*
  - *Designed to be the foundation for others*

- **Co-maintains Bitbake, OpenEmbedded-Core with the Yocto Project.**
- **Owns the meta-openembedded components.**

Why not just use OpenEmbedded? (cont)

- **Yocto Project is focused on enabling commercial developers. It does this by helping to improve the quality of OpenEmbedded.**
  - The Yocto Project co-maintains and leverages Bitbake and OpenEmbedded-Core, and extends them by adding COTS BSPs, a reference distribution, documentation, etc.
  - Provides a tested, pre-prepared combination of build system components
  - Yocto Project includes autobuilder sessions
  - QA testings
  - Eclipse Plugins
  - Branding / Compatibility Program
  - …etc…

Yocto Project Branding

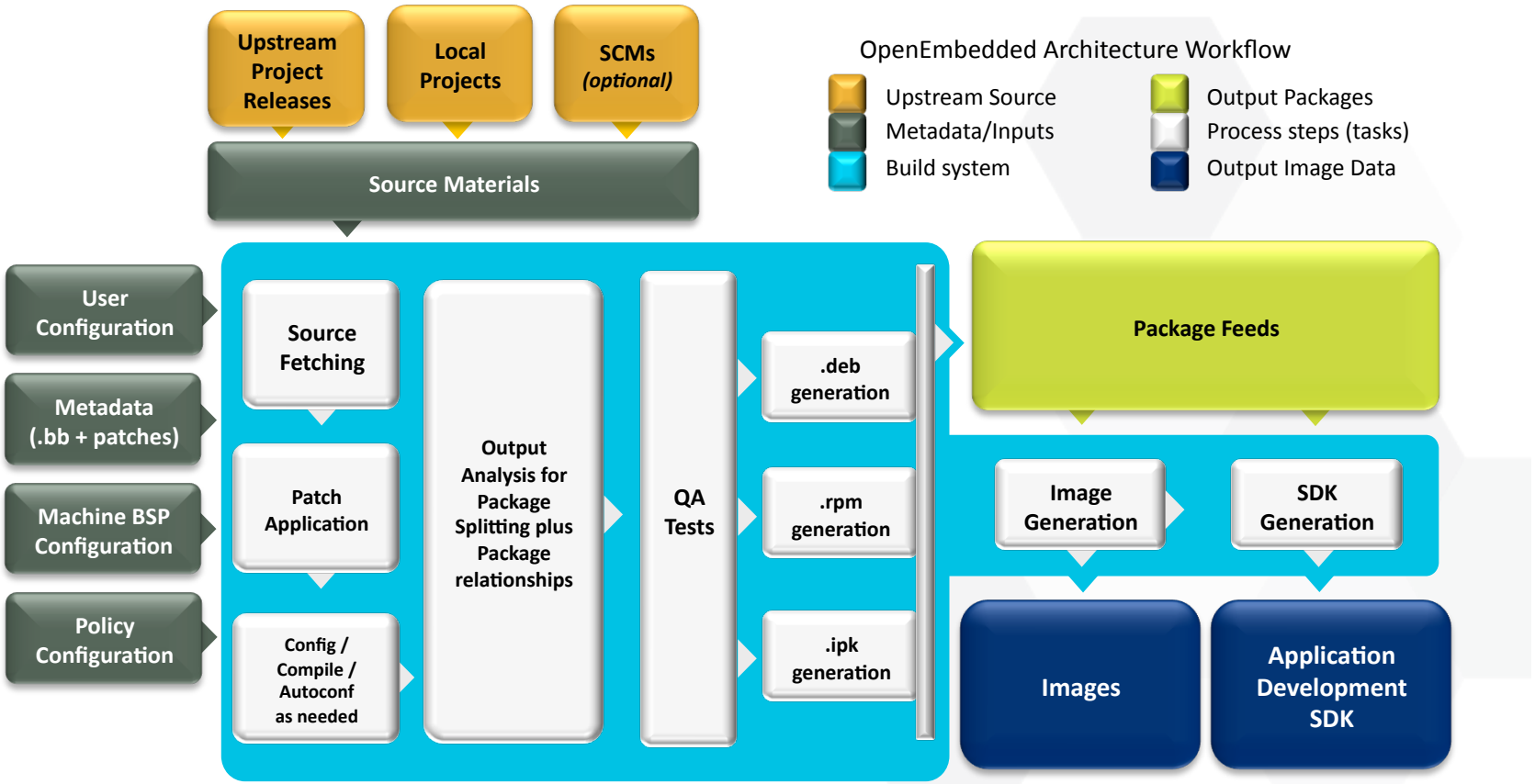
- Reduce fragmentation in the embedded market by encouraging collaborative development of a common set of tools, standards, and practices.
- Ensure that these tools, standards, and practices are architecturally independent, as much as possible.
- Encourage Interoperability and Contributions.

Enough about the Project, what about using it?

Quick Start

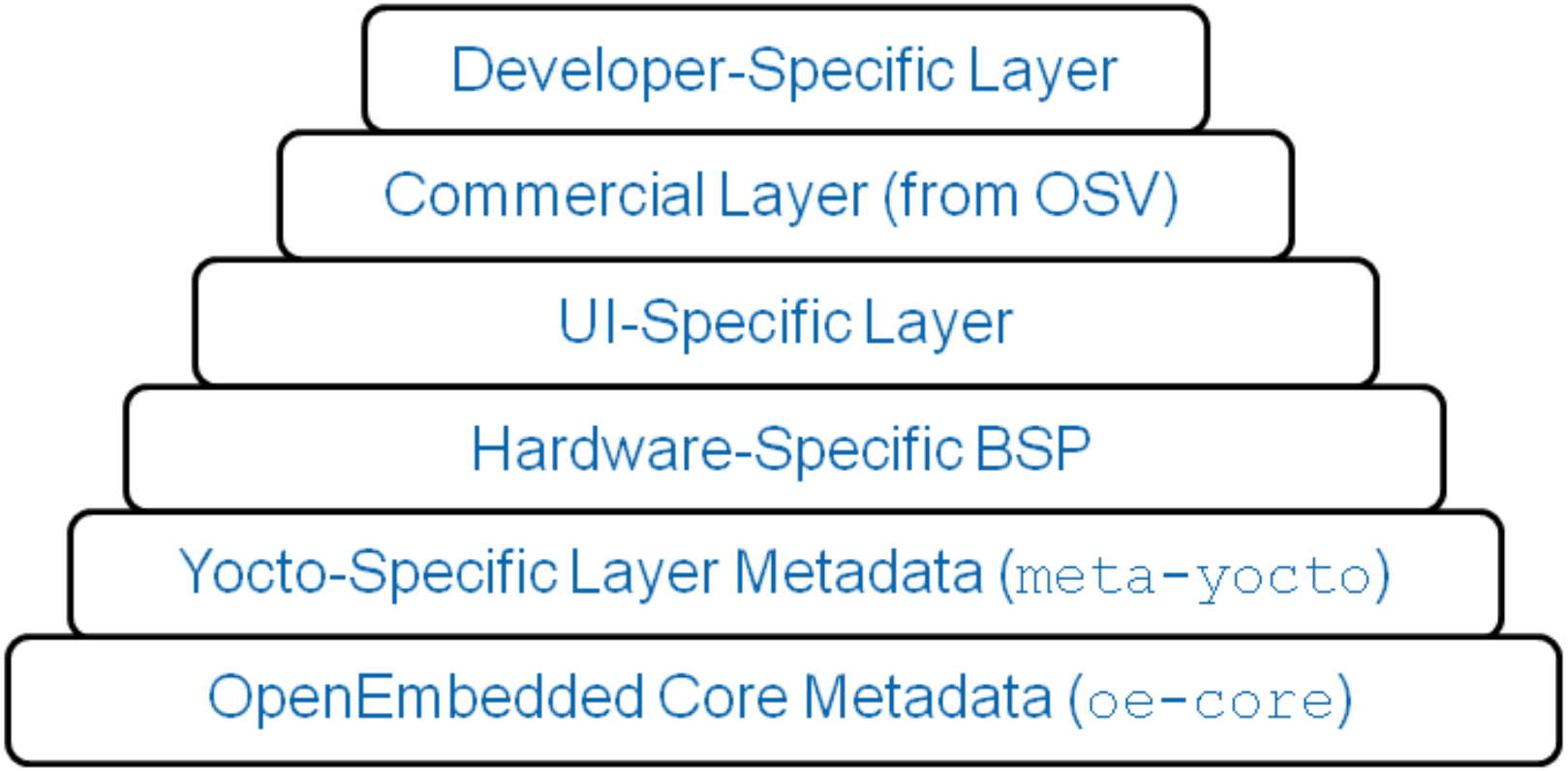
1. Go to http://yoctoproject.org click “documentation” and read the Quick Start guide
2. Set up your Linux build system with the necessary packages (and firewall as needed)
3. Go to http://yoctoproject.org click “downloads” and download the latest stable release (Yocto Project 1.3 “Danny” 8.0.0) – extract the download on your build machine
4. Source `oe-init-build-env` script
5. Edit `conf/local.conf` and set `MACHINE`, `BB_NUMBER_THREADS` and `PARALLEL_MAKE`
6. Run `bitbake core-image-sato`
7. Run `runqemu qemux86 (if MACHINE=qemux86)`

Build System Workflow



Layers

- The build system is composed of layers



Layers (cont)

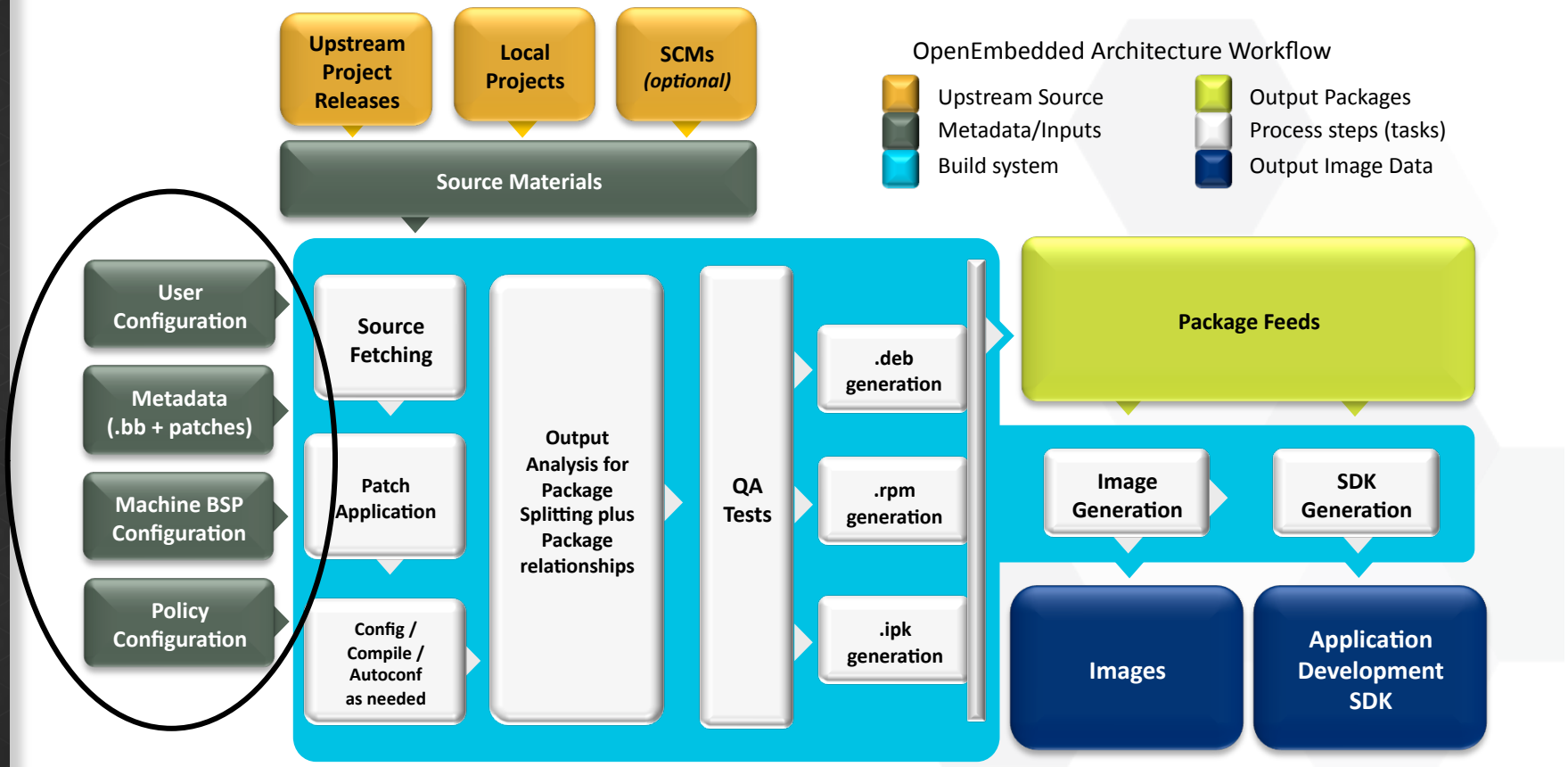
- Layers are containers for the building blocks used to construct the system

LEGO is a trademark of the LEGO Group

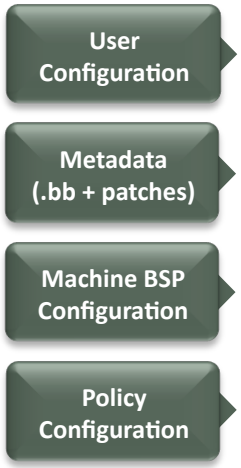
Layers

- **Layers are a way to manage extensions, and customizations to the system**
  - Layers can extend, add, replace or modify recipes
  - Layers can add or replace bbclass files
  - Layers can add or modify configuration settings
  - Layers are added via BBLAYERS variable in `build/conf/bblayers.conf`
- **Best Practice: Layers should be grouped by functionality**
  - Custom Toolchains (compilers, debuggers, profiling tools)
  - Distribution specifications (i.e. meta-yocto)
  - BSP/Machine settings (i.e. meta-yoct-bsps)
  - Functional areas (selinux, networking, etc)
  - Project specific changes

All starts with the Configuration

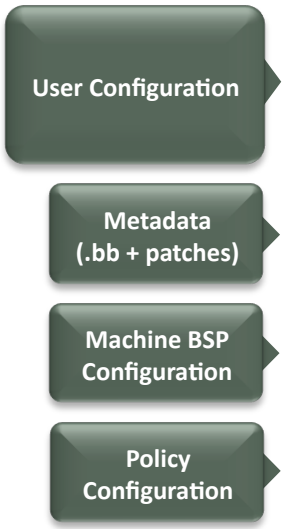


Configuration



- **Configuration files (*.conf) – global build settings**
  - meta/conf/bitbake.conf (defaults)
  - build/conf/bblayers.conf (layers)
  - */conf/layers.conf (one per layer)
  - build/conf/local.conf (local user-defined)
  - meta-yocto/conf/distro/poky.conf (distribution policy)
  - meta-yocto-bsp/conf/machine/beagleboard.conf (BSP)
  - meta/conf/machine/include/tune-cortexa8.inc (CPU)
  - Recipes (metadata)

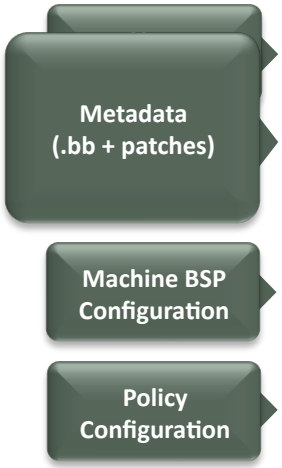
User Configuration



- **build/conf/local.conf is where you override and define what you are building**
  - BB_NUMBER_THREADS and PARALLEL_MAKE
  - MACHINE settings
  - DISTRO settings
  - INCOMPATIBLE_LICENSE = “GPLv3”
  - EXTRA_IMAGE_FEATURES
- **build/conf/bblayers.conf is where you configure with layers to use**
  - Add Yocto Project Compatible layers to the BBLAYERS
  - Default: meta (oe-core), meta-yocto and meta-yocto-bsp

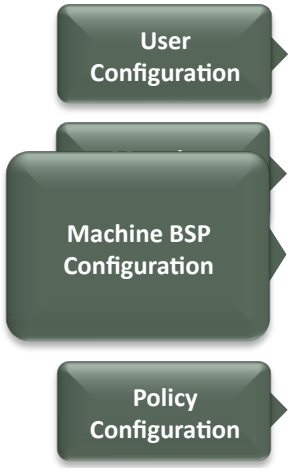
Metadata (Recipes)

- Metadata and patches:
  - Recipes for building packages
  - Recipe:
    - meta/recipes-core/busybox_1.20.2.bb
  - Patches:
    - meta/recipes-core/busybox/busybox-1.20.2
  - Recipes inherit the system configuration and adjust it to describe how to build and package the software
  - Can be extended and enhanced via layers
  - Compatible with OpenEmbedded



Machine (BSP) Configuration

- **Configuration files that describe a machine**
  - Define board specific kernel configuration
  - Formfactor configurations
  - Processor/SOC Tuning files
- **Eg, `meta-yocto-bsp/conf/machine/beagleboard.conf`**

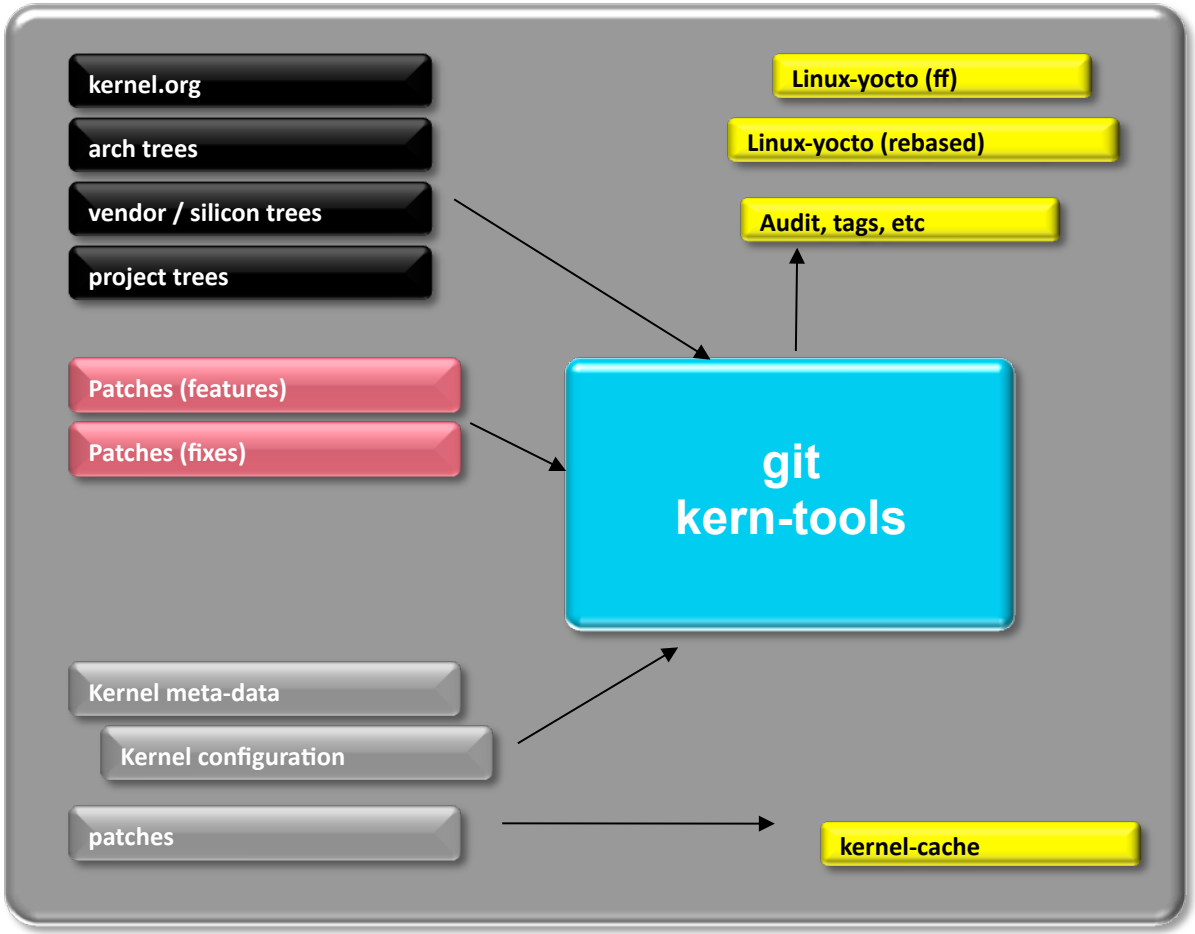


- **Machine configuration refers to kernel sources and may influence some userspace software**
- **Compatible with OpenEmbedded**

Yocto Project Kernel Development

- **We try to develop upstream wherever possible**
- **Two major advances in the Yocto Project:**
  - Kernel Features: patches and configuration fragments managed as functional blocks (supports reuse)
  - Branching Tools: Optional Per-BSP git branches define machine-specific kernel sources. Tools collect the relevant kernel features to produce kernel sources
- **Results:**
  - Can turn on/off a collection of features for a given BSP
  - Reuse – less code duplication
  - Easier to choose a configuration based on features

linux-yocto Construction



Upstream git trees

Yocto project tools

Upstream patches

Yocto Project inputs

Yocto Project Output

Yocto Project Kernel Details

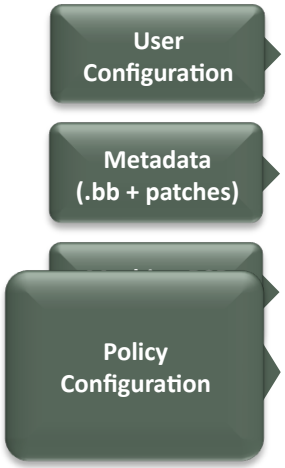
- **Build system infrastructure**
  - kernel.bbclass
  - kernel-arch.bbclass
  - kernel-yocto.bbclass
  - linux-kernel-base.bbclass
- **Linux-Yocto recipes**
  - meta/recipes-kernel/linux/linux-yocto*.bb
- **Linux-Yocto git repository**
  - http://git.yoctoproject.org/git/linux-yocto-*
- **Kernel Versions**
  - 3.0, 3.2, 3.4
  - Dev: 3.7-rc3 (tracks latest kernel.org)

Board Support Layers

- **Best Practices:**
  - Manage BSPs in layers
  - Define machine settings and machine specific recipe changes
  - Base on the Yocto Project kernel version
  - Use the Yocto Project kernel tooling

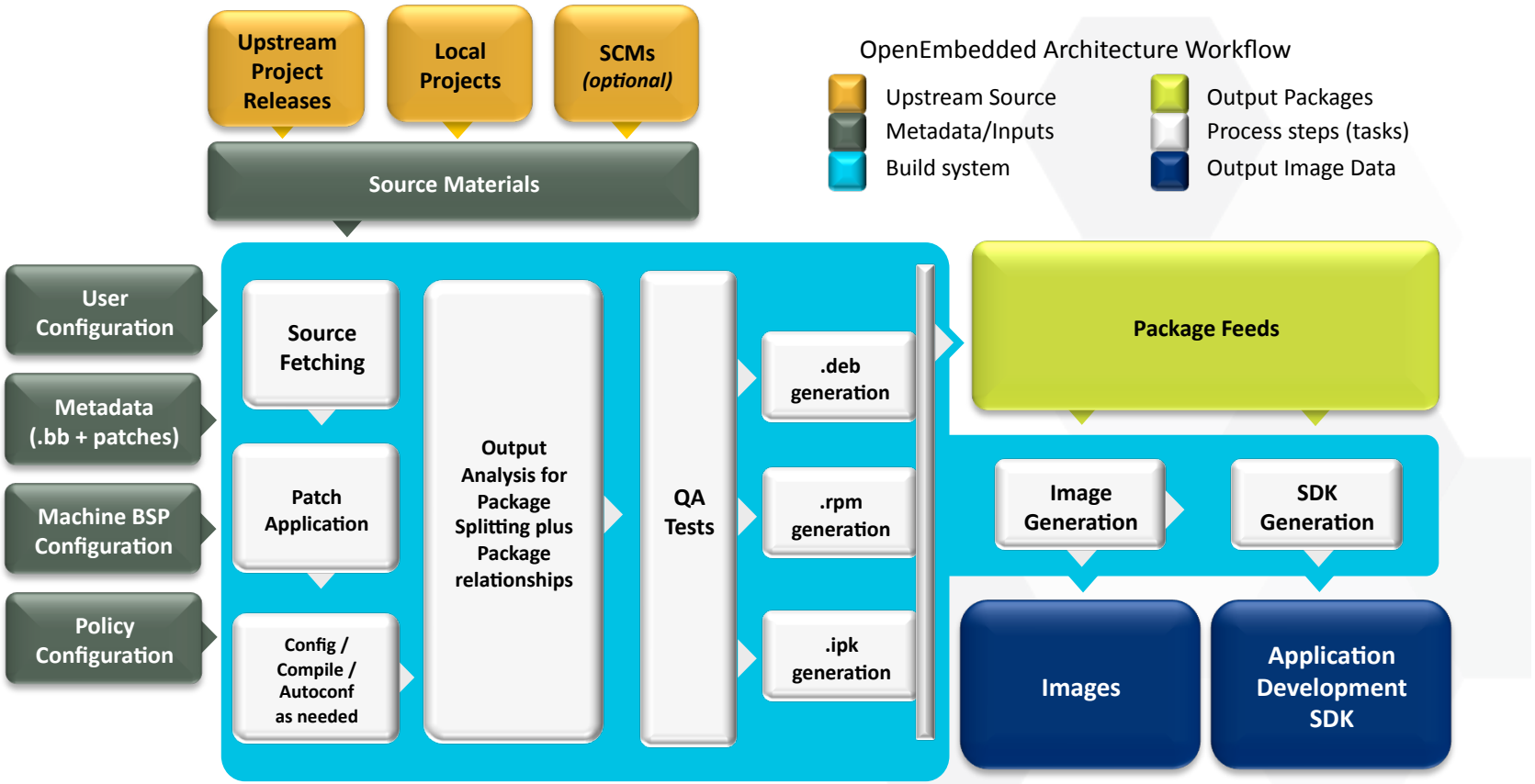
- **A BSP is not required to use the Yocto Project kernel version or kernel tooling**
  - May be get the same cost benefit of scalability and upstream bug fixing

Distribution Policy

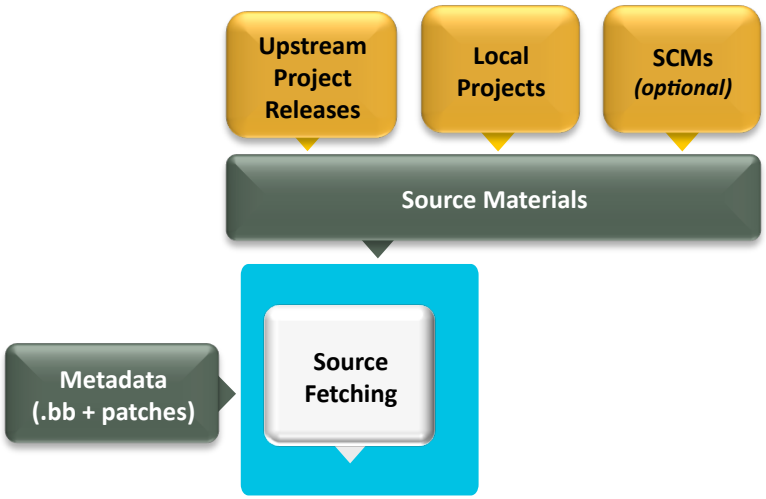


- **Defines distribution/system wide policies that affect the way individual recipes are built**
  - May set alternative preferred versions of recipes
  - May enable/disable LIBC functionality (i.e. i18n)
  - May enable/disable features (i.e. pam, selinux)
  - May configure specific package rules
  - May 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

How does it work? In-depth build process



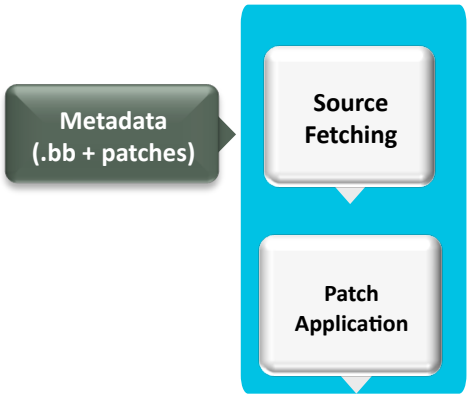
Source Fetching



- Recipes call out the location of all sources, patches and files. These may exist on the internal or be local. (See SRC_URI in the *.bb files)
- Bitbake can get the sources from git, svn, bzr, 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

* Complete list includes: http, ftp, https, git, svn, perforce, mercurial, bzr, cvs, osc, repo, ssh, and svk and the unpacker can cope with tarballs, zip, rar, xz, gz, bz2, and so on.

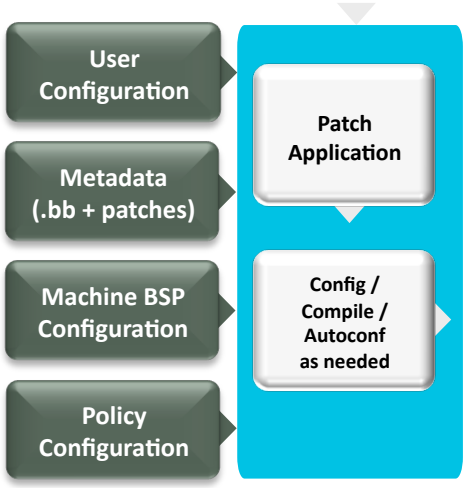
Patching



- 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

Configure / Compile / Install



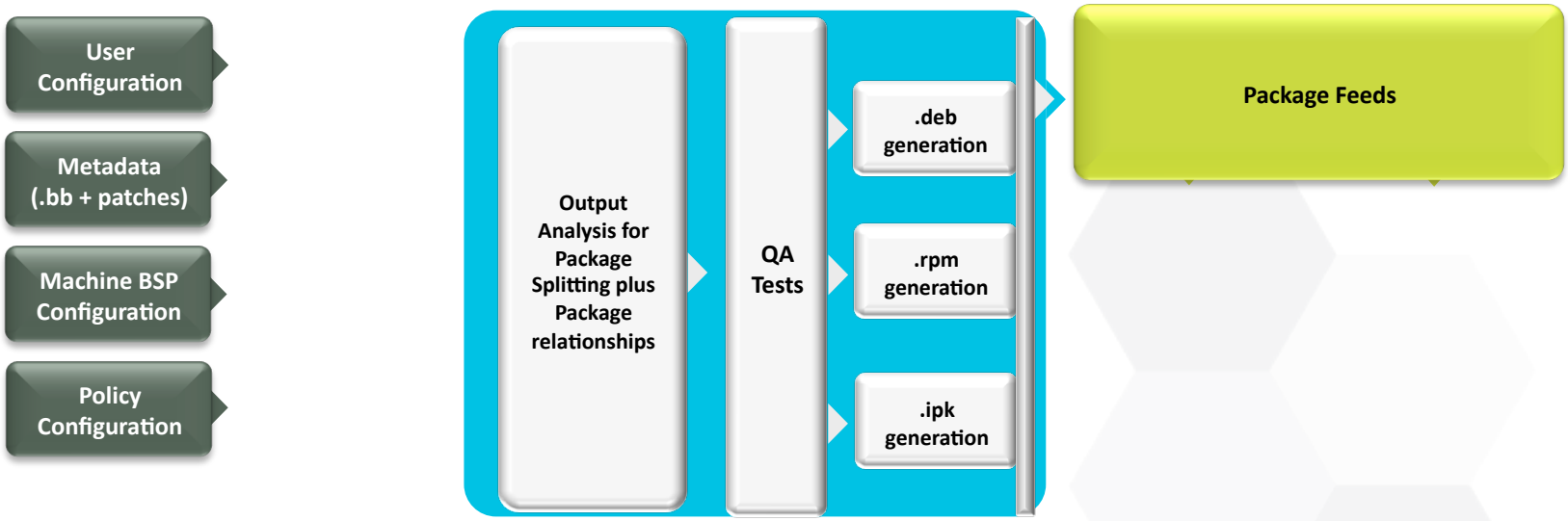
- **Recipe specifies configuration and compilation rules**
  - Various standard build rules are available, such as autotools and gettext
  - Standard ways to specify custom environment flags
  - Install step runs under 'pseudo', allows special files, permissions and owners/ groups to be set
- **Recipe example:**

```
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
```

Output Analysis / Packaging

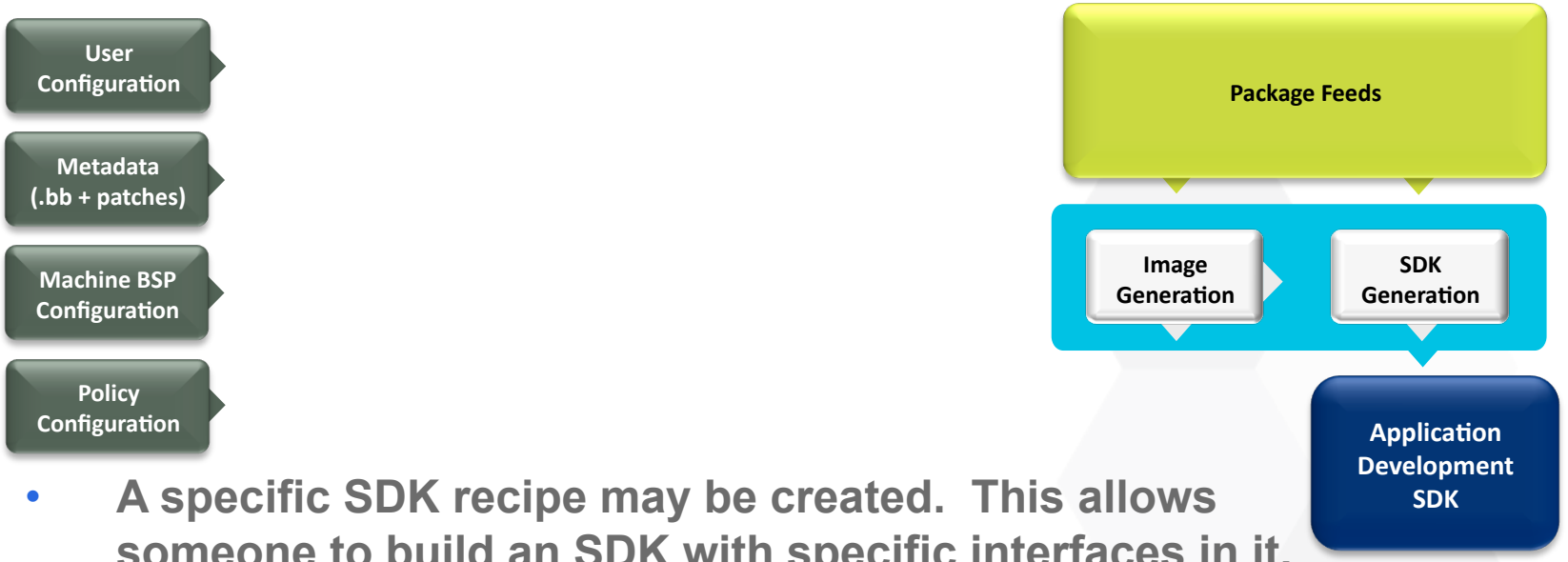


- **Output Analysis:**
  - Categorize generated software (debug, dev, docs, locales)
  - Split runtime and debug information
- **Perform QA tests (sanity checks)**
- **Package Generation:**
  - Support the popular formats, RPM, Debian, and ipk
  - Set preferred format using PACKAGE_CLASSES in local.conf
  - Package files can be manually defined to override automatic settings

Image Generation

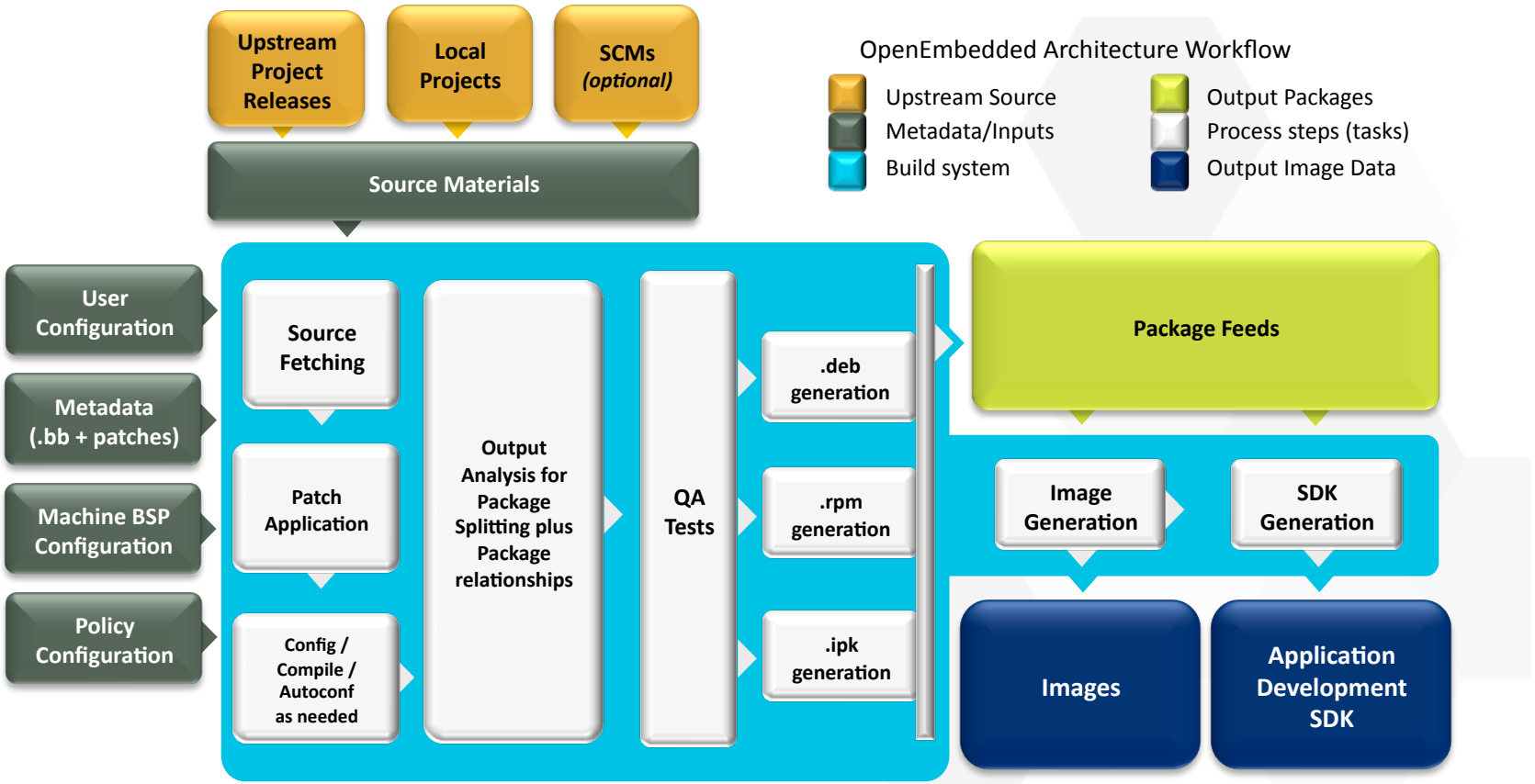
- Images are constructed using the packages built earlier and put into the Package Feeds
- Decisions of what to install on the image is based on the minimum defined set of required components in an image recipe. This minimum set is then expanded based on dependencies to produce a package solution.
- Images may be generated in a variety of formats (tar.bz2, ext2, ext3, jffs, etc…)

SDK Generation



- 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
  - *New in Danny release*
- 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

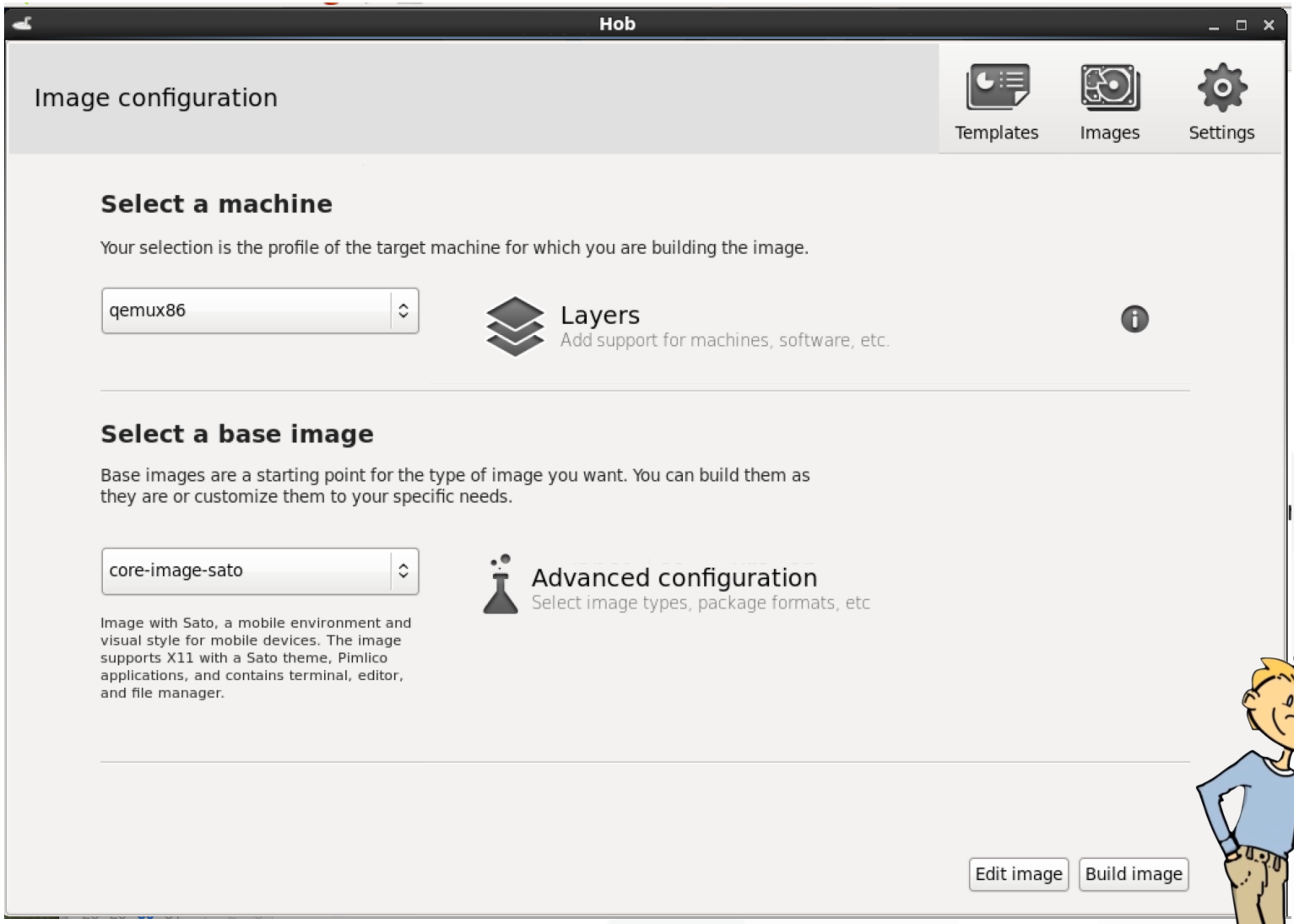
Build System Workflow



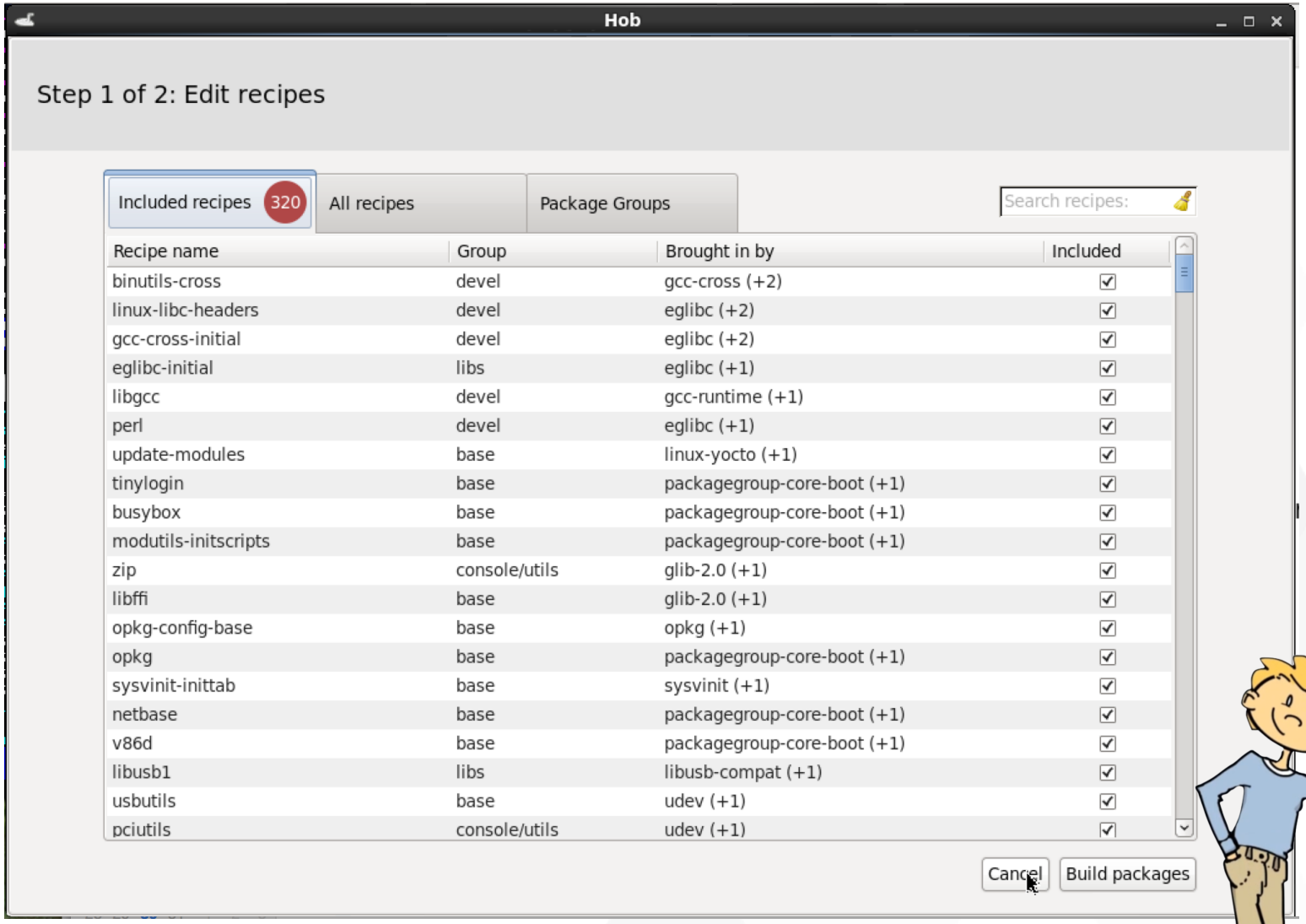
Hob

- **There has to be an easier way than setting various configuration files…**
- **The Hob User Interface is that way.**
- **Hob /häb/**
  - Noun: A flat metal shelf at the side or back of a fireplace, having its surface level with the top of the grate and used esp. for heating pans.
  - A sprite or hobgoblin.

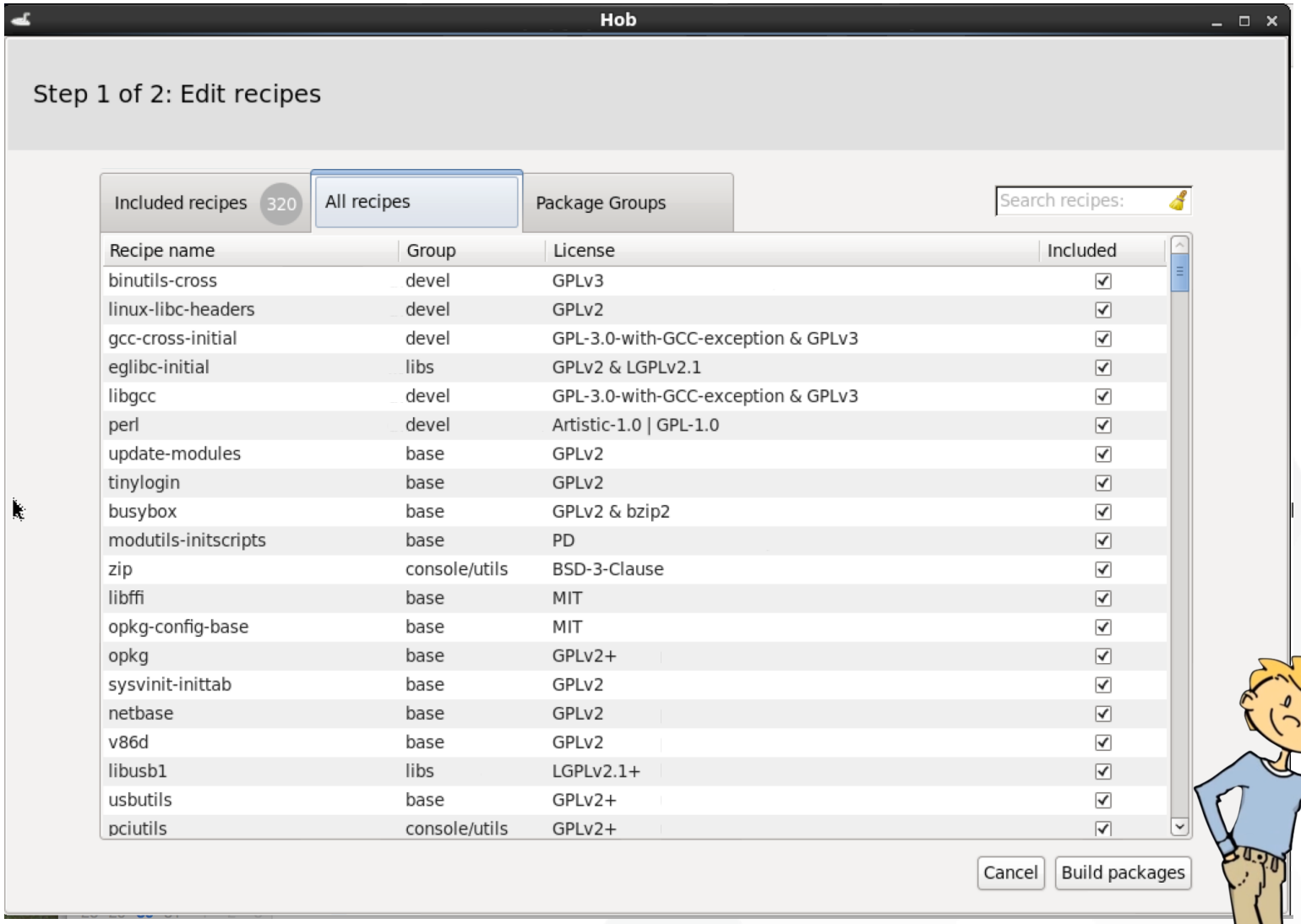
Hob



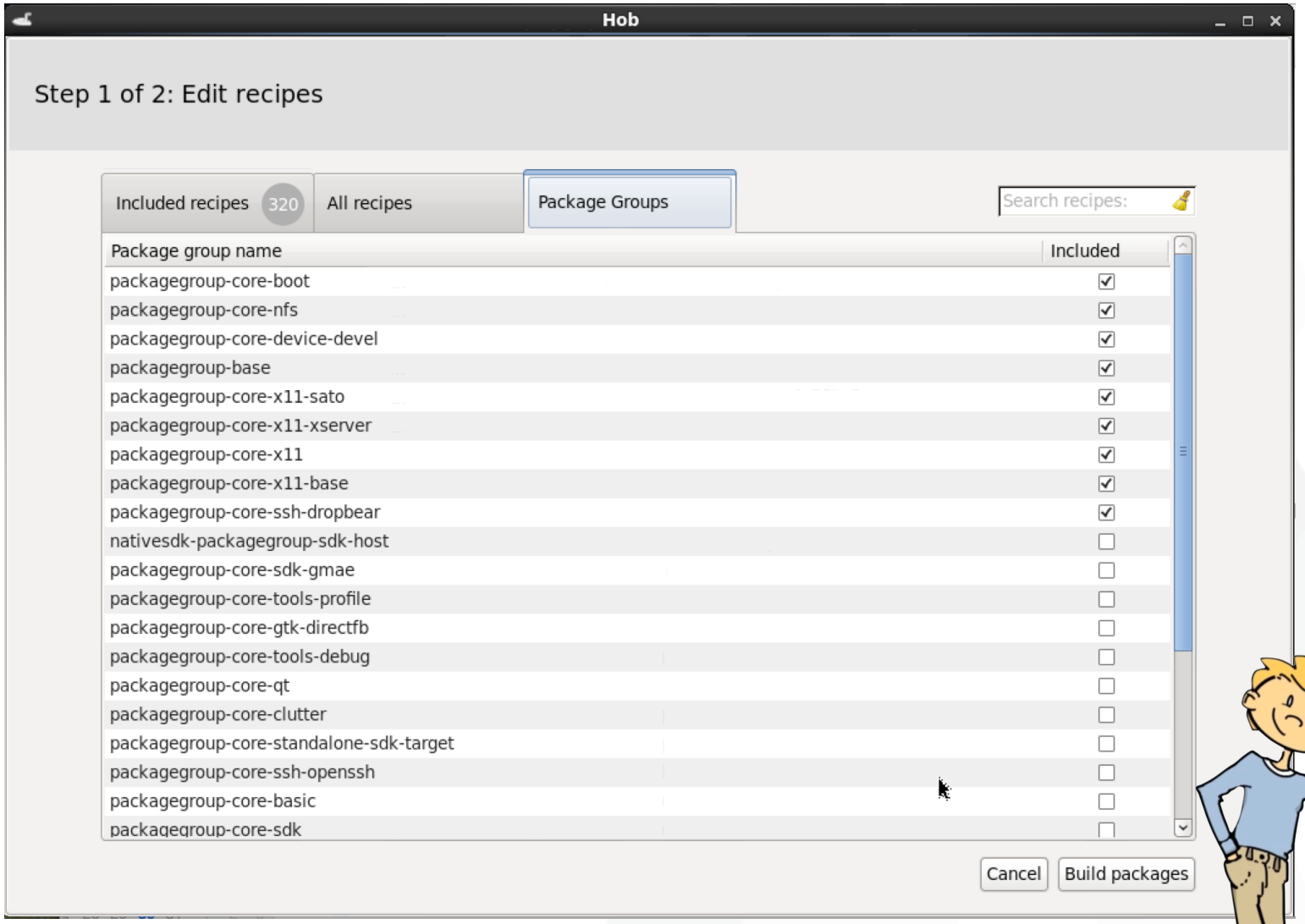
Hob



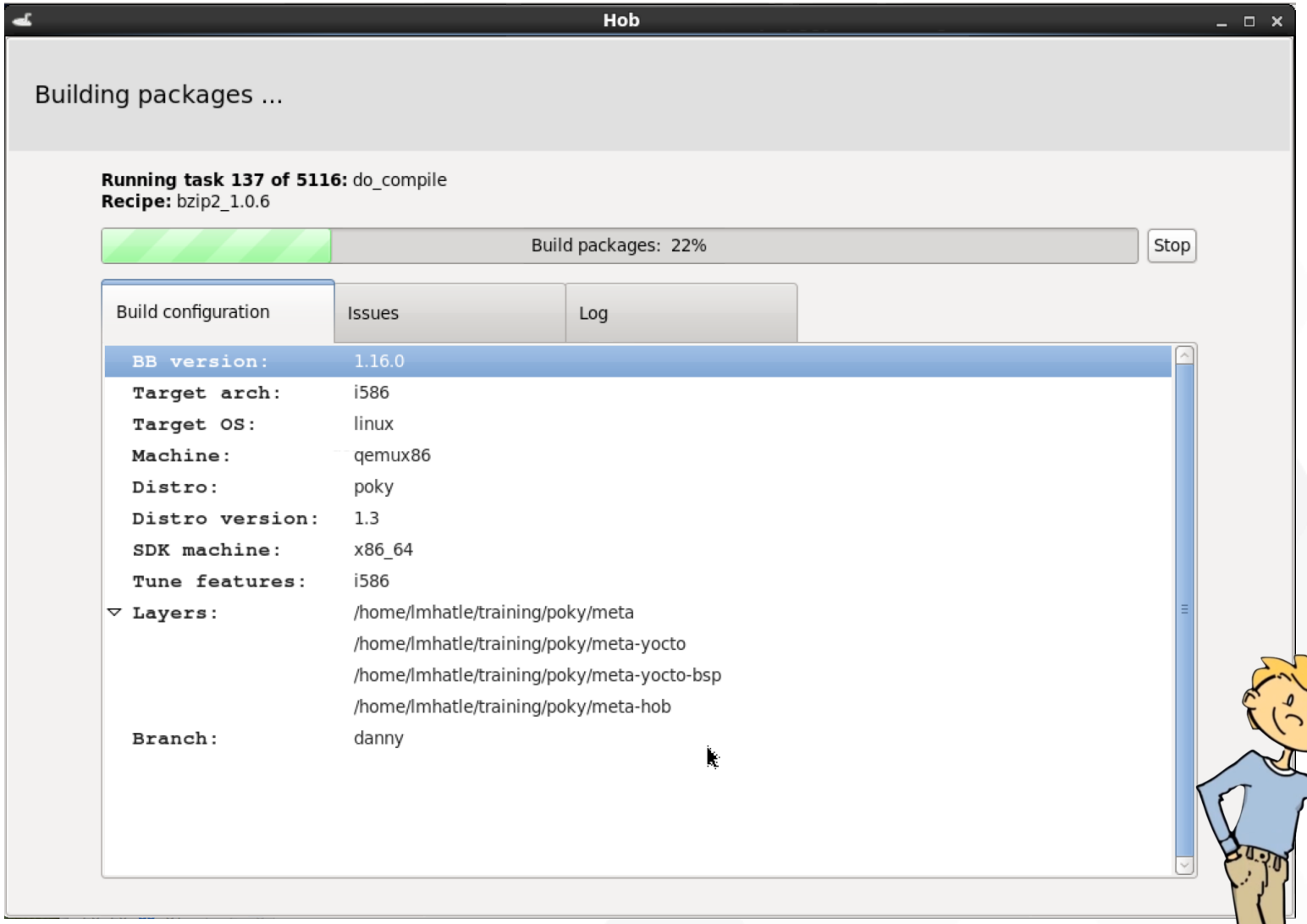
Hob



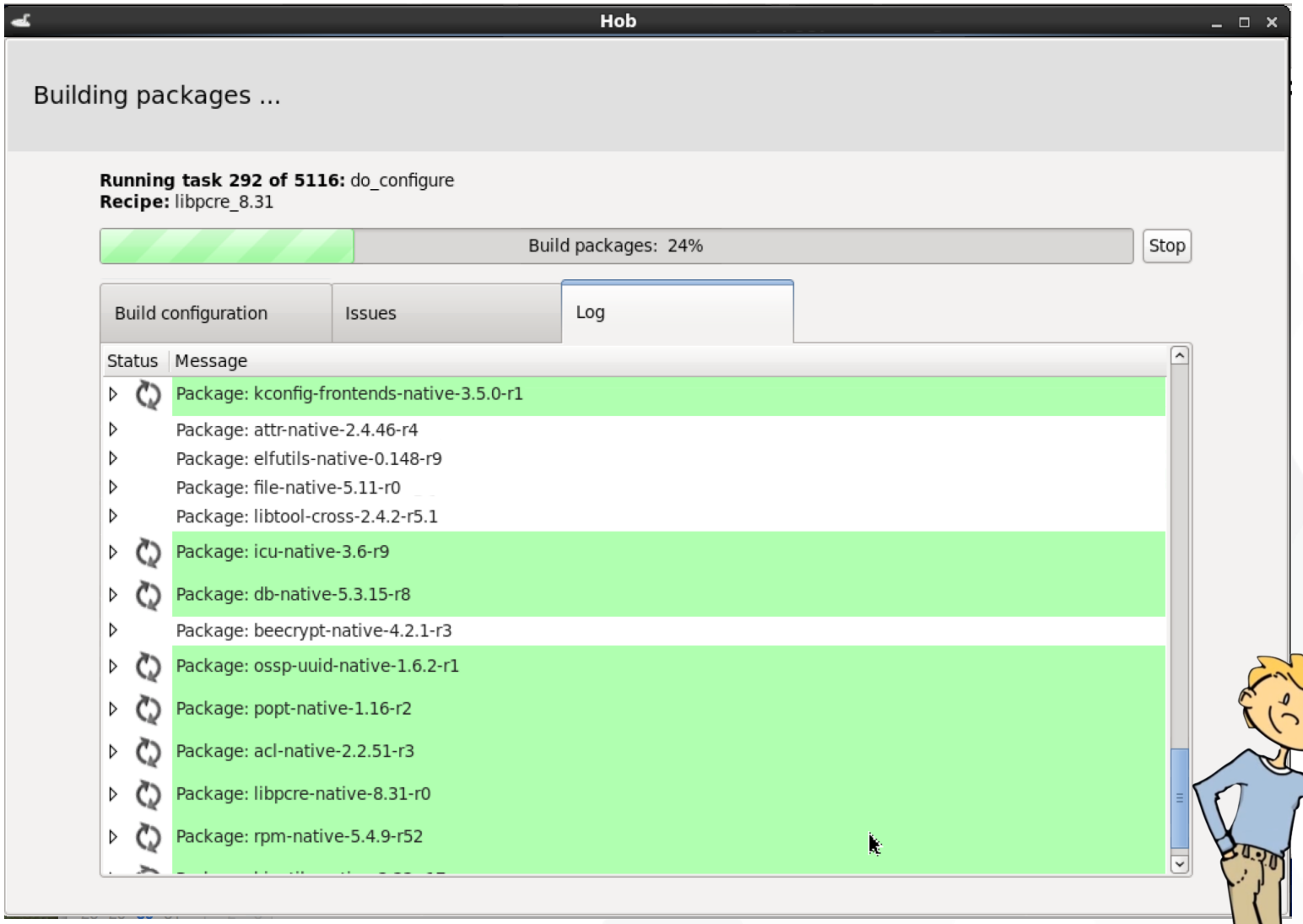
Hob



Hob



Hob



Summary

- **The Yocto Project provides tools, templates and best practices for you to create your embedded Linux OS**
- **The Poky project provides a set of reference distribution components in one place to make it easy to get started**
  - It helps set up the embedded app developer
  - Both device and app development models supported
  - Getting started is easy
- **It's not an embedded Linux distribution – it creates a custom one for you**

Collaboration is the key to success

Spend less time and resources, by an one org, to develop and maintain the commodity parts.

Be able to spend more time and use the resources you already have to create your products and value added components!

Thank You

The Yocto Project is hosted by the Linux Foundation.