



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

## Developing Embedded Linux Devices Using the Yocto Project™



**Mark Hatle**  
Wind River Systems  
September, 2011

# Agenda

- What is the Yocto Project (YP)?
- How does it work?
- How to get started with building OS, apps, and debugging
- What's Next?
- Q&A

# What is the Yocto Project?

- A collection of embedded projects and tooling
- Supports ARM, PPC, MIPS, x86 (32 & 64 bit)
- Open source project with a strong community
- Key project is the distribution build environment (Poky)
  - Complete Linux OS with package metadata
  - Releases every 6 months with latest (but stable) kernel, toolchain, and package versions
  - Place for Industry to publish BSPs
  - App Dev Tools which allow development against the stack, including Eclipse plug-ins and emulators
  - Full documentation representative of a consistent system

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

# Why is the Yocto Project Valuable?

- Linux is becoming increasingly popular for Embedded
- Non-commercial and commercial embedded Linux has many distributions
- Result is:
  - Developers spend lots of time porting or making build systems
  - Leaves less time/money to develop interesting software features
- The industry needs a common build system and core technology
- Industry leaders have joined together to form the Yocto Project
- The benefit of doing so is:
  - Less time spent on things which don't make money (build system, core Linux components)
  - Linux grows more in embedded
  - More time spent on things which do make money (application development, product development, ...)

# Why Should a Developer Care?

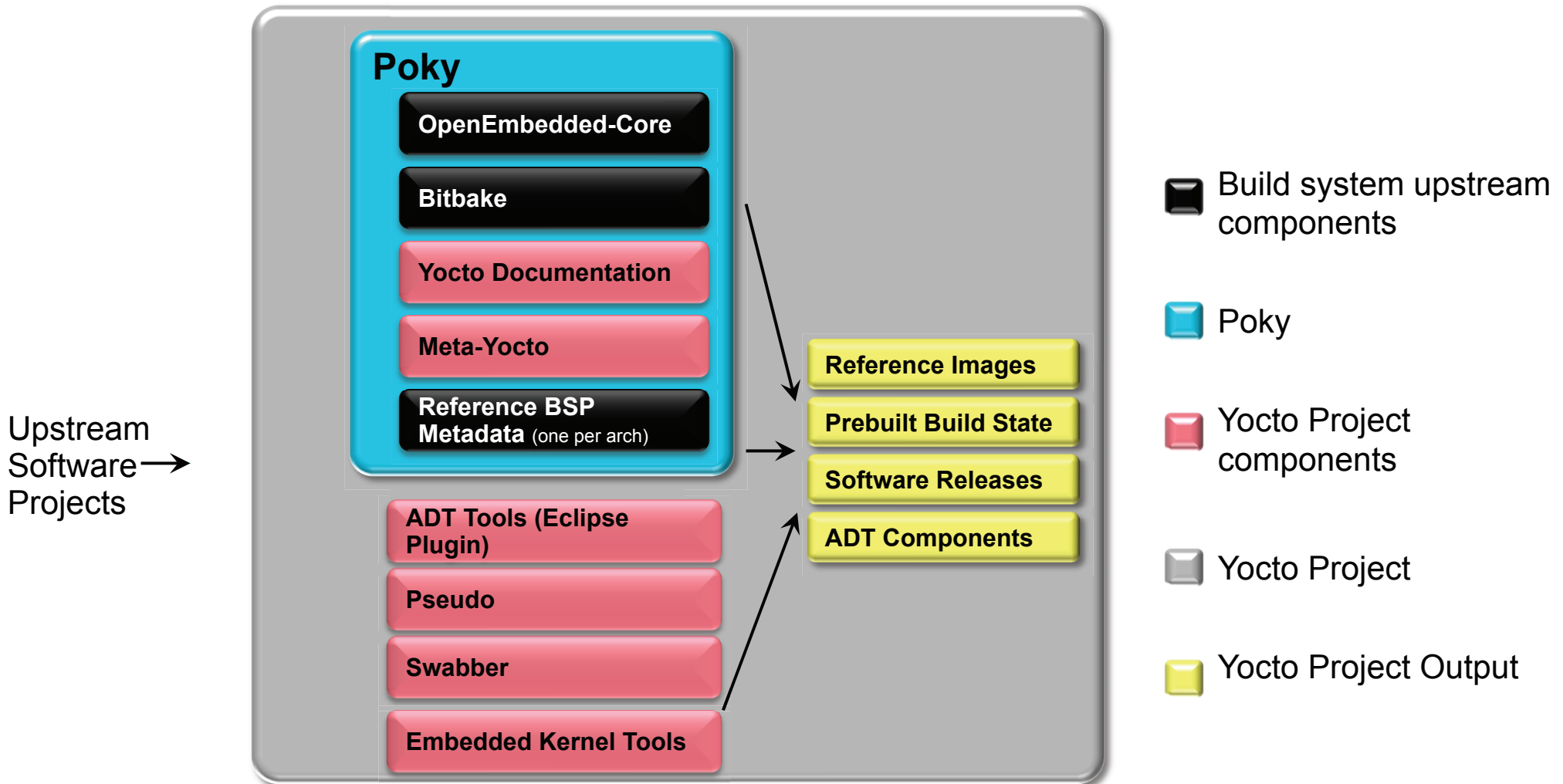
- Build a complete Linux system –from source– in about an hour (about 90 minutes with X).
- Start with a validated collection of software (toolchain, kernel, user space).
- Access to a great collection of app developer tools (performance, debug, power analysis, Eclipse). We distinguish app developers system developers and we support both.
- Advanced kernel development tools.
- Supports all major embedded architectures (x86, x86-64, ARM, PPC, MIPS), just change a line in a config file and rebuild.
- Transitions easily to a commercial embedded Linux (Mentor Graphics, MontaVista, Timesys, Wind River).

# How Does It Work? – Quick Start

1. Go to <http://yoctoproject.org>, click “documentation” and consult the Quick Start guide
2. Set up your Linux system with the right packages (and firewall access, if needed)
3. Click “Download” and download the latest stable release (or check out “bernard” from the git repo)
4. Edit `conf/local.conf` and set `MACHINE`, `BB_NUMBER_THREADS` and `PARALLEL_MAKE`
5. Source `poky-init-build-env` script (or `oe-init-build-env`)
6. Run `$ bitbake poky-image-sato` (or `core-image-sato`)
7. Run `$ poky-qemu qemux86` (or `runqemu qemux86`) (if `MACHINE=qemux86`)

**Note:** File or command names in this presentation are subject to change, several are different now in master.

# Yocto Project = Poky + Upstreams + Tools



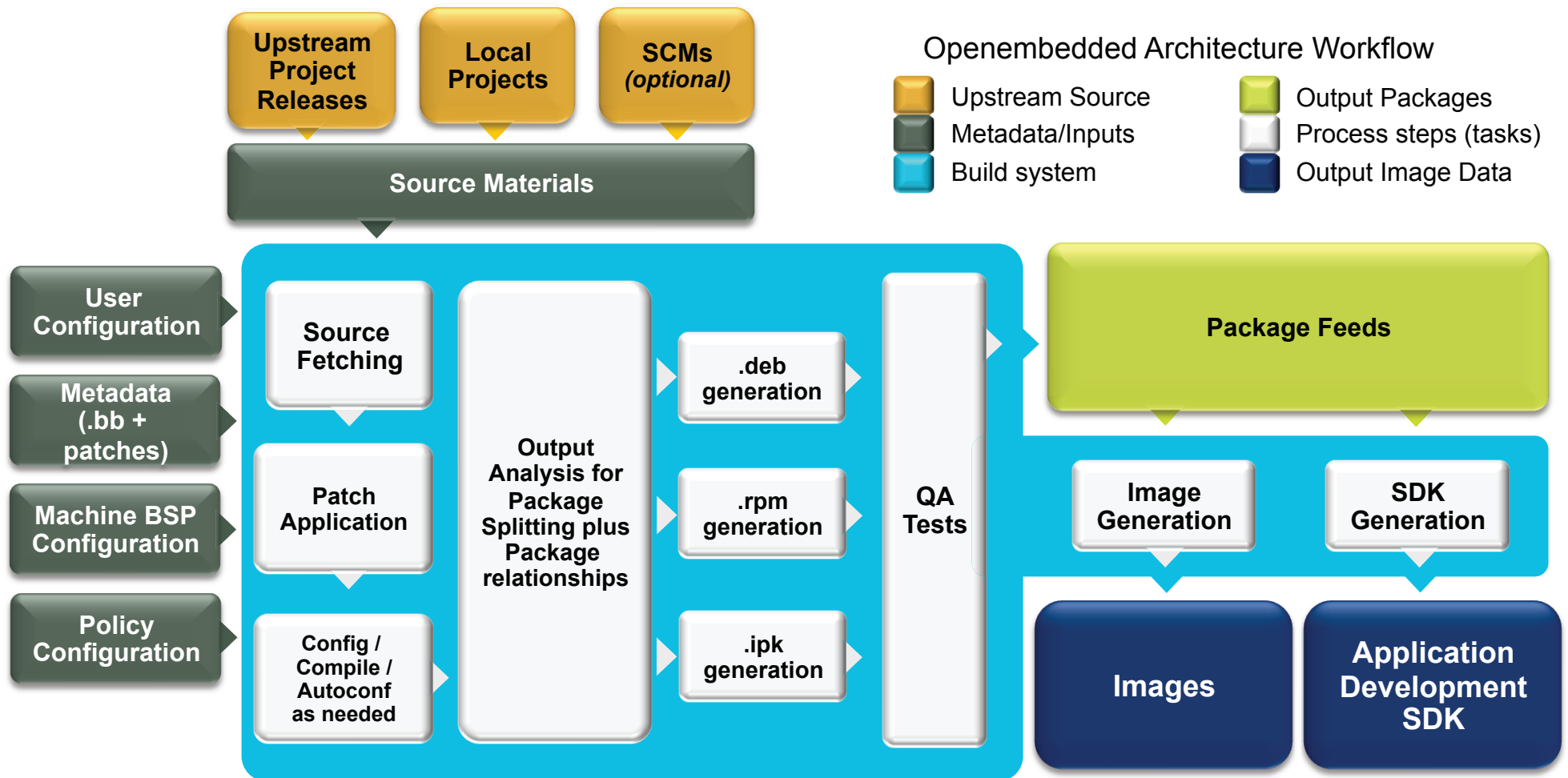
*Yocto Project provides best of upstream for a stable base*

# Why not just use OpenEmbedded?

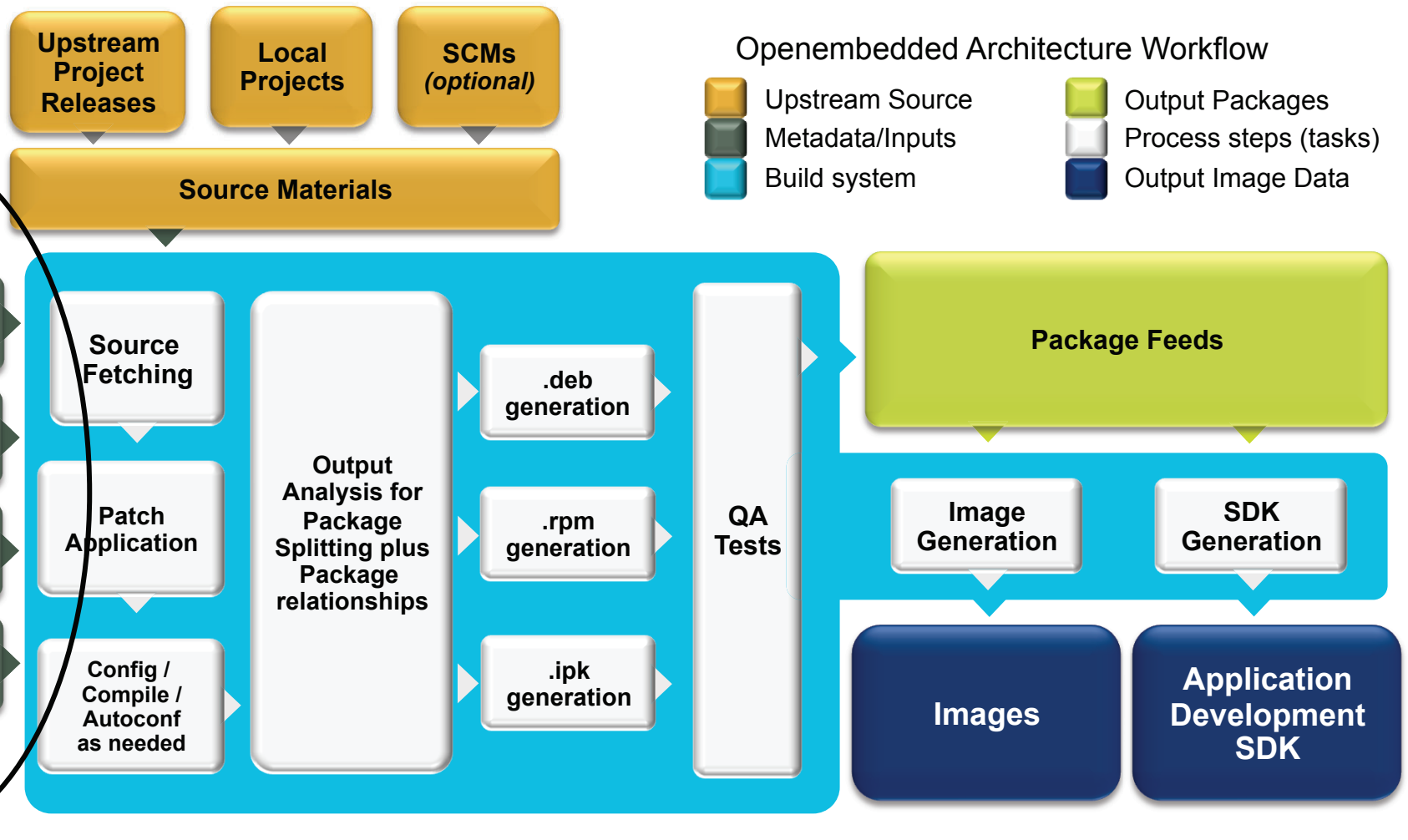
- OpenEmbedded is an Open Source project focused on enabling cross-compiled systems. It contains the build system, and meta data that describe many components of a system.
- Yocto Project is focused on enabling commercial developers. It does this by helping to improve the quality of OpenEmbedded. Leveraging the build system and core components of OpenEmbedded, and extending the support to key COTS boards.
  - Yocto Project includes autobuilder sessions
  - QA testings
  - Eclipse Plugins
  - ...etc...



# Yocto Project\* Workflow



# Configuration



# Configuration

User  
Configuration

Metadata  
(.bb +  
patches)

Machine BSP  
Configuration

Policy  
Configuration

- Configuration (\*.conf) – global definition of variables
  - build/conf/local.conf (local user-defined variables)
  - meta-yocto/conf/distro/poky.conf (Yocto policy config variables)
  - meta/machine/include/tune-mips32.inc (processor specific configuration)
  - meta-yocto/machine/routerstationpro.conf (machine-specific variables)

# User Configuration

## User Configuration

Metadata  
(.bb +  
patches)

Machine BSP  
Configuration

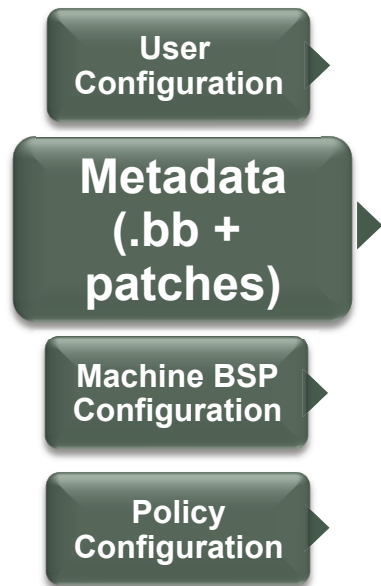
Policy  
Configuration

User configuration:

- conf/local.conf – some things to set:
  - Set BB\_NUMBER\_THREADS and PARALLEL\_MAKE, based on the number of threads in the machine
  - Set MACHINE="foo" for the CPU architecture
  - EXTRA\_IMAGE\_FEATURES adds features (groups of packages)
  - INCOMPATIBLE\_LICENSE = "GPLv3" eliminates packages using this license (for example)

# Metadata

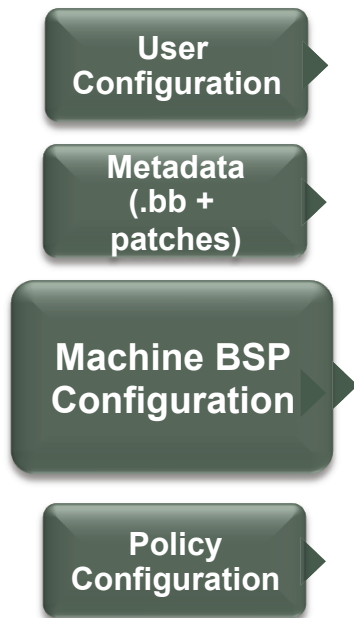
## Metadata and patches:



- Recipes for building packages
- Eg, `meta/recipes-core/coreutils/coreutils_6.9.bb` builds the core utilities (version 6.9) and installs them
- `meta/recipes-core/coreutils/coreutils-6.9/` includes patches, also could include extra files to install
- Can be extended and enhanced via “layers”
- Compatible with OpenEmbedded

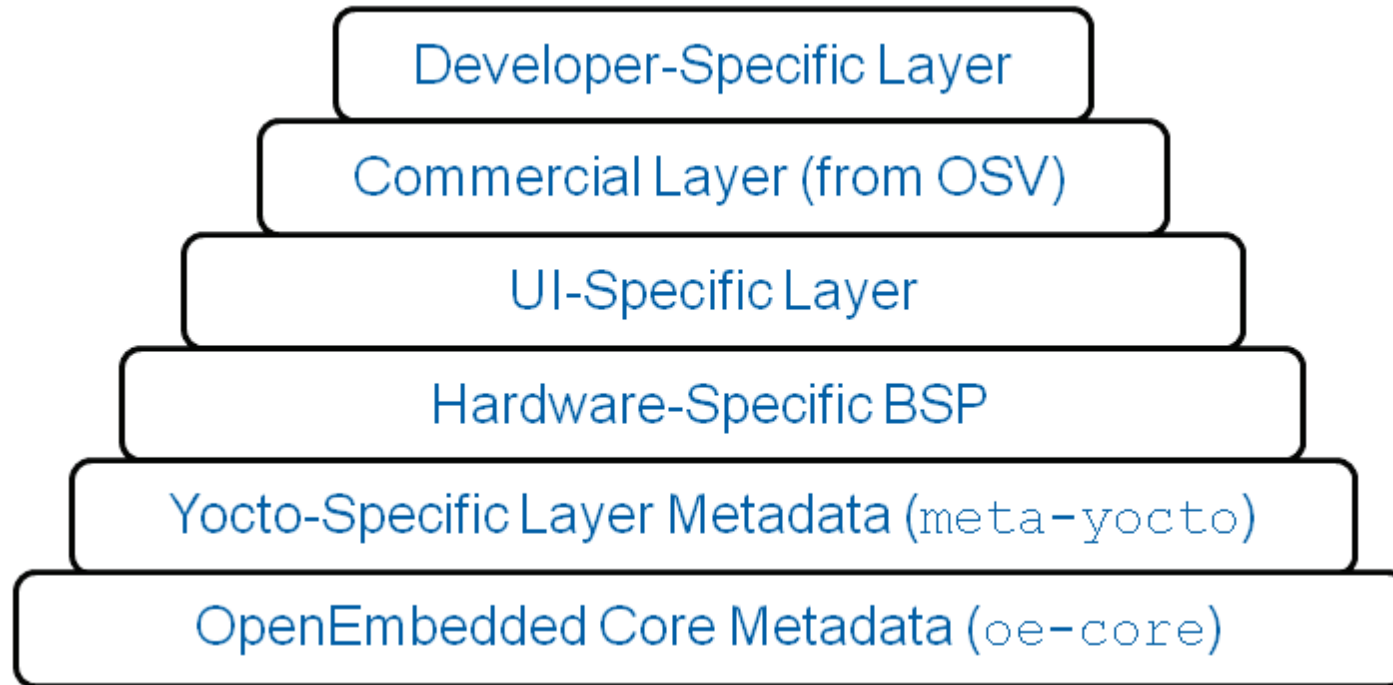
# BSPs

## BSPs and Machine configurations:



- Configuration files to describe a machine
- May include Linux kernel enhancements
- Includes board specific kernel configuration
- Eg, `meta-yocto/machine/routerstationpro.conf` describes the Routerstation Pro features, kernel filenames, etc.
- Should be implemented via “layers”
- Compatible with OpenEmbedded

# Layers



# Board Support “Layers”

- Layers contain extensions and customizations to base system
  - Can include image customisations, additional recipes, modifying recipes, adding extra configuration
    - Really just another directory to look for recipes or recipe extensions
    - Added to the BBLAYERS variable in build/conf/bblayers.conf
- Best Practice: BSPs are layers that add machine settings and recipes
- Machine settings are specified in a layer's conf/machine/xxx.conf file(s)
- Examples:
  - Sandy Bridge + Cougar Point:
    - meta-intel/conf/meta-sugarbay/machine/sugarbay.conf
  - Routerstation Pro (MIPS)
    - yocto/meta/conf/machine/routerstationpro.conf



# Kernel Development

- We try to develop upstream wherever possible
- Two major advances in the Yocto Project:
  - Branching tools: Per-BSP git branches contain machine-specific kernel sources. Tools collect up the relevant tree of branches
  - Kernel features: patches and configuration fragments managed as a functional block
- Results:
  - Can turn on a collection of features for a given BSP
  - Less code duplication
  - Easier to choose a config fragment and patches

# Kernel Tools Details

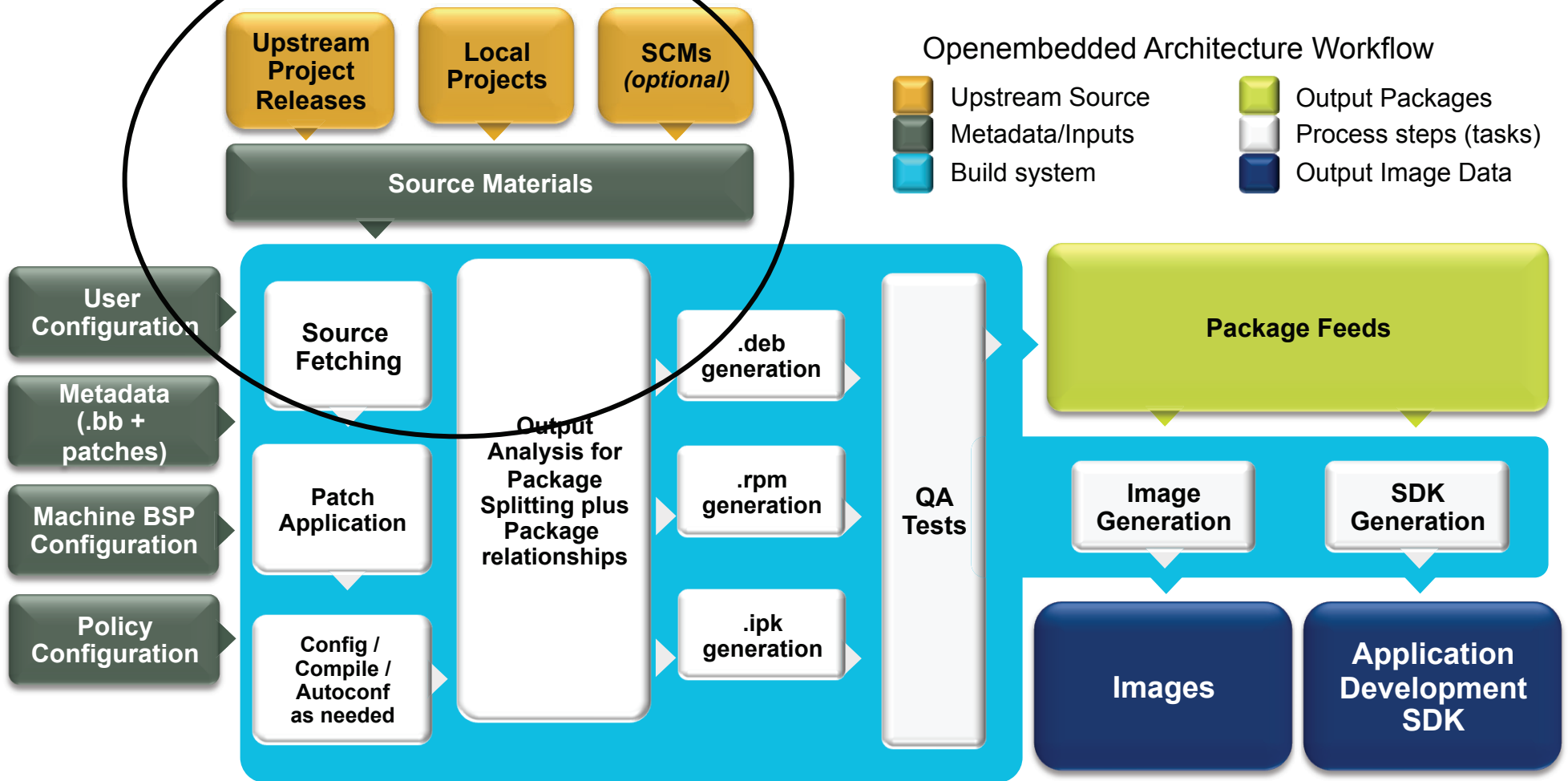
## Components

- Kernel class
  - meta/classes/kernel.bbclass
- Linux-Yocto recipe
  - meta/recipes-kernel/linux/linux-yocto\*bb
- Linux-Yocto git repository
  - <http://git.pokylinux.org/cgi/cgit.cgi/linux-yocto-3.0>

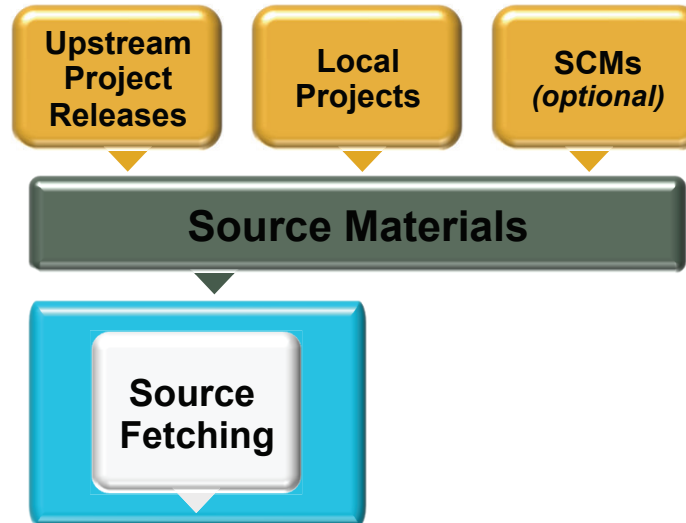
## Kernel Versions

- linux-yocto-stable: 2.6.34 & 2.6.37 & 3.0
- *linux-yocto-dev: 3.1-rc4*

# How Does It Work? More Depth



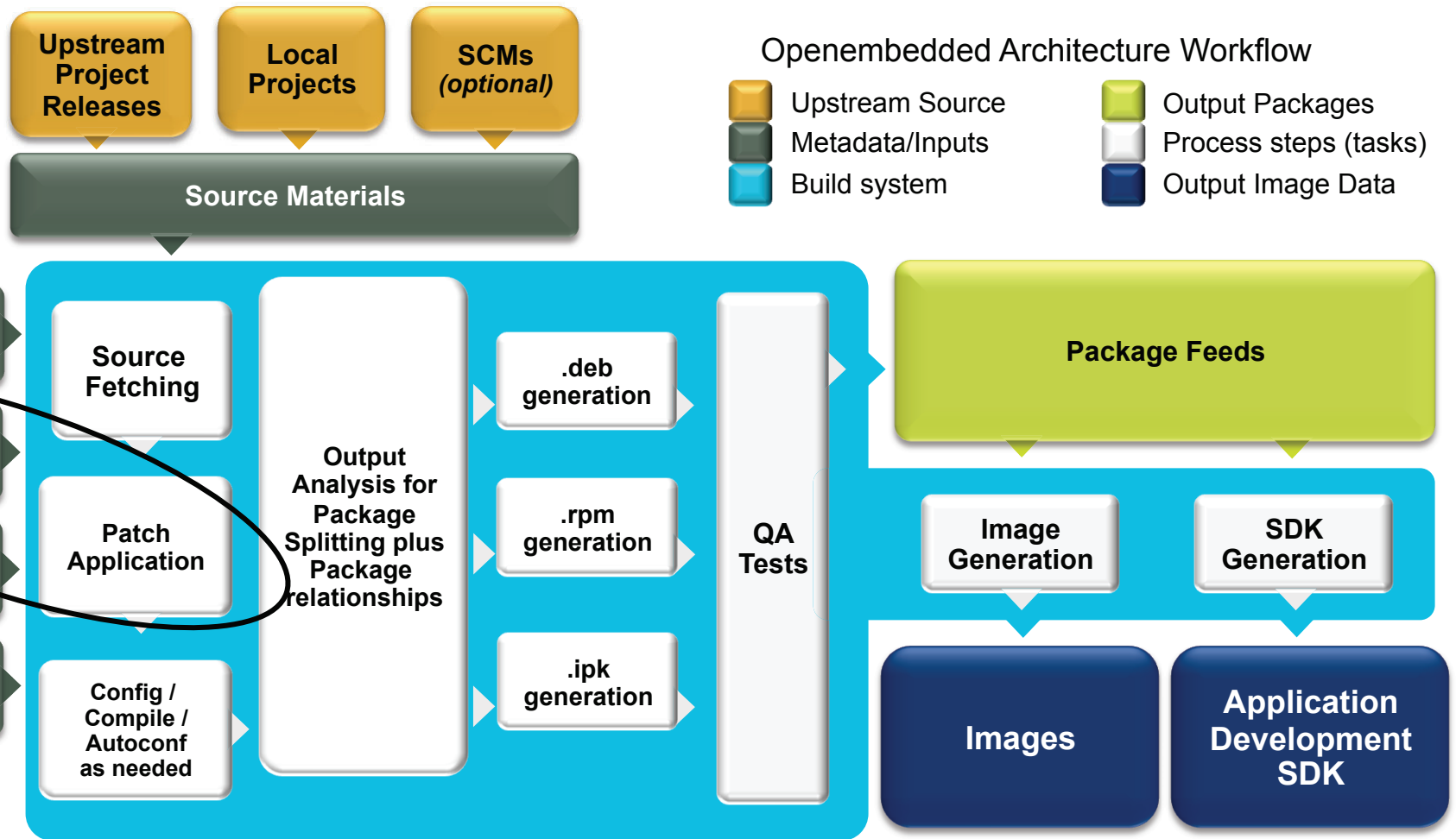
# Source Fetching



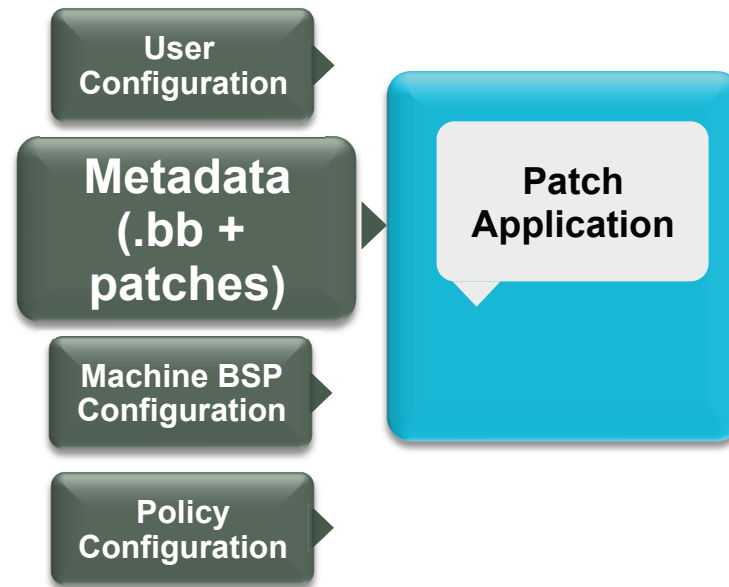
- Recipes call out location of all sources, whether on the internet or local (Look for SRC\_URI in \*.bb files)
- Bitbake can get sources from git, svn, bzd, from tarballs, and many, many more\*
- Versions of packages can be fixed or updated automatically (Add SRCREV\_pn- PN = "\${AUTOREV}" to local.conf)
- Yocto Project sources mirror available as a fallback, if the sources move on the internet

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

# Patching

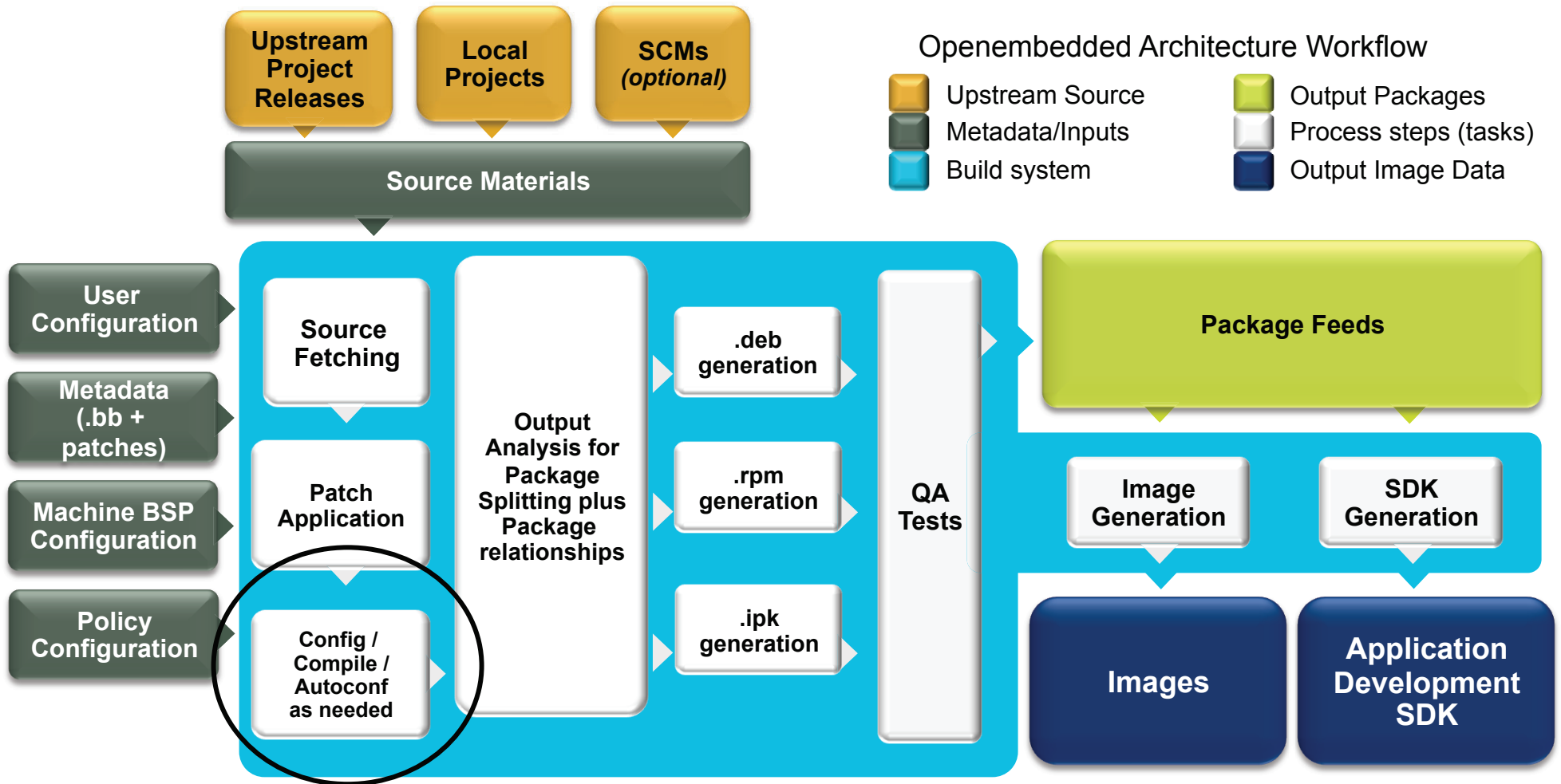


# Patching

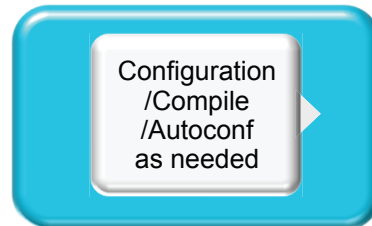


- Once sources are obtained, the patches are applied
- This is a good place to patch the software yourself
- However, we encourage you to contribute development upstream whenever possible (we try to)

# Config / Compile



# Configure/Compile



- **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**

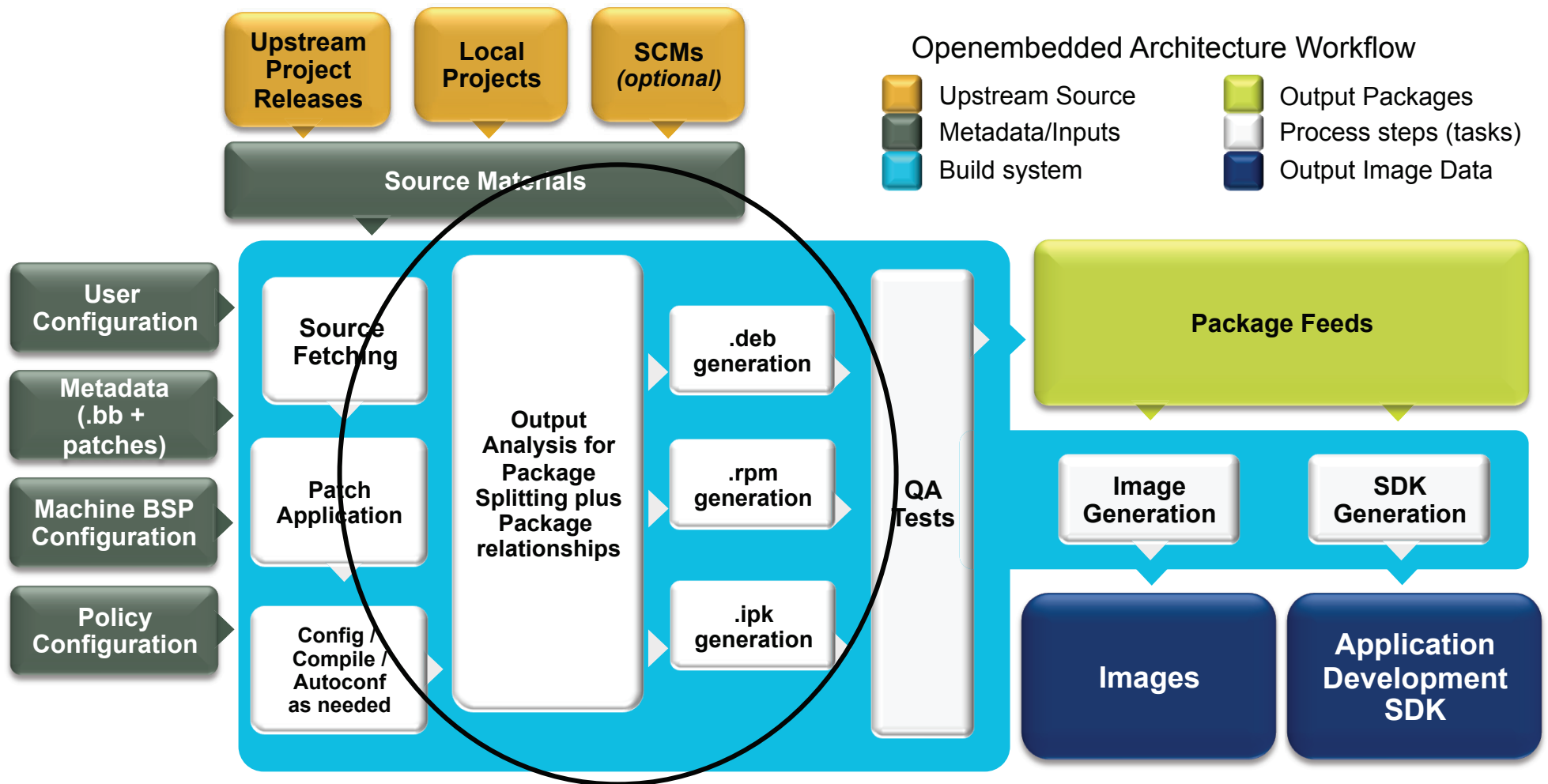
```
CFLAGS_prepend = "-I ${S}/include "
```

- **Install task to set modes, permissions, target directories, done by “pseudo”**

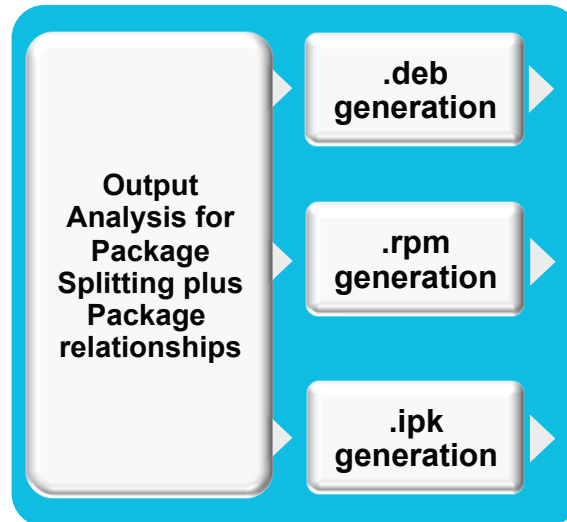
```
do_install () {
    oe_runmake install DESTDIR=${D} SBINDIR=${sbindir} MANDIR=${mandir}
```



# Packaging



# Packaging



Once configure/compile/install is completed, packaging commences

The most popular package formats are supported: RPM, Debian, and ipk

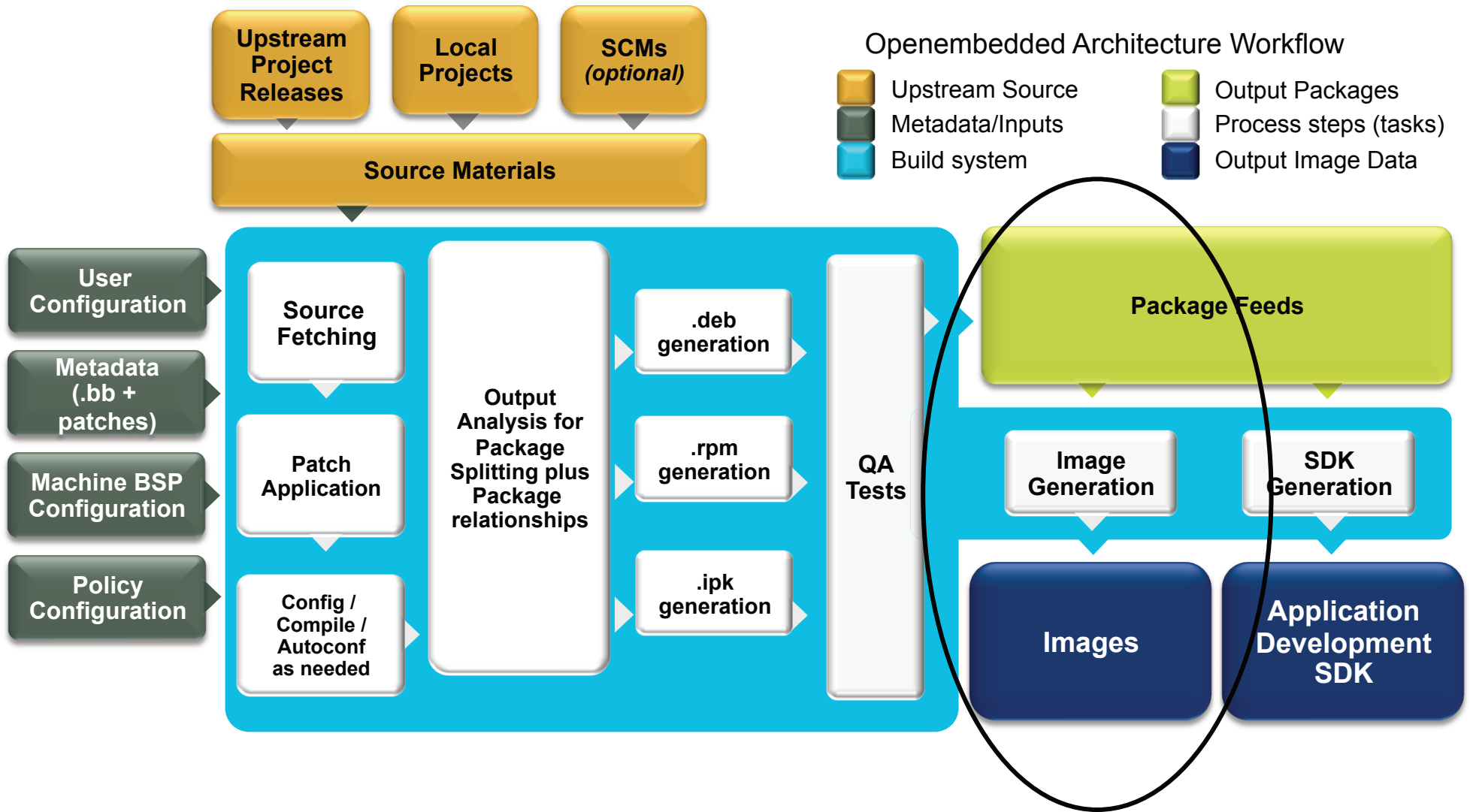
- Set `PACKAGE_CLASSES` in `conf/local.conf`

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"
```

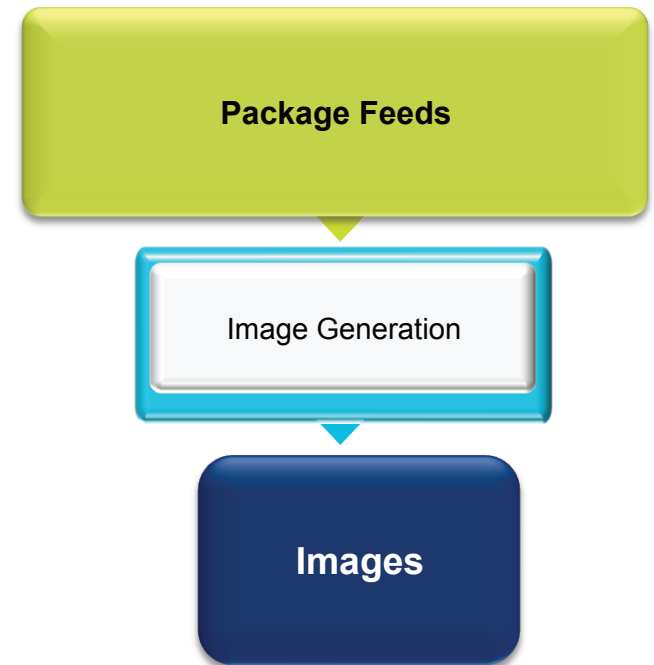
- Automatic splitting into debug, documentation, development and locale data.

# Image Generation



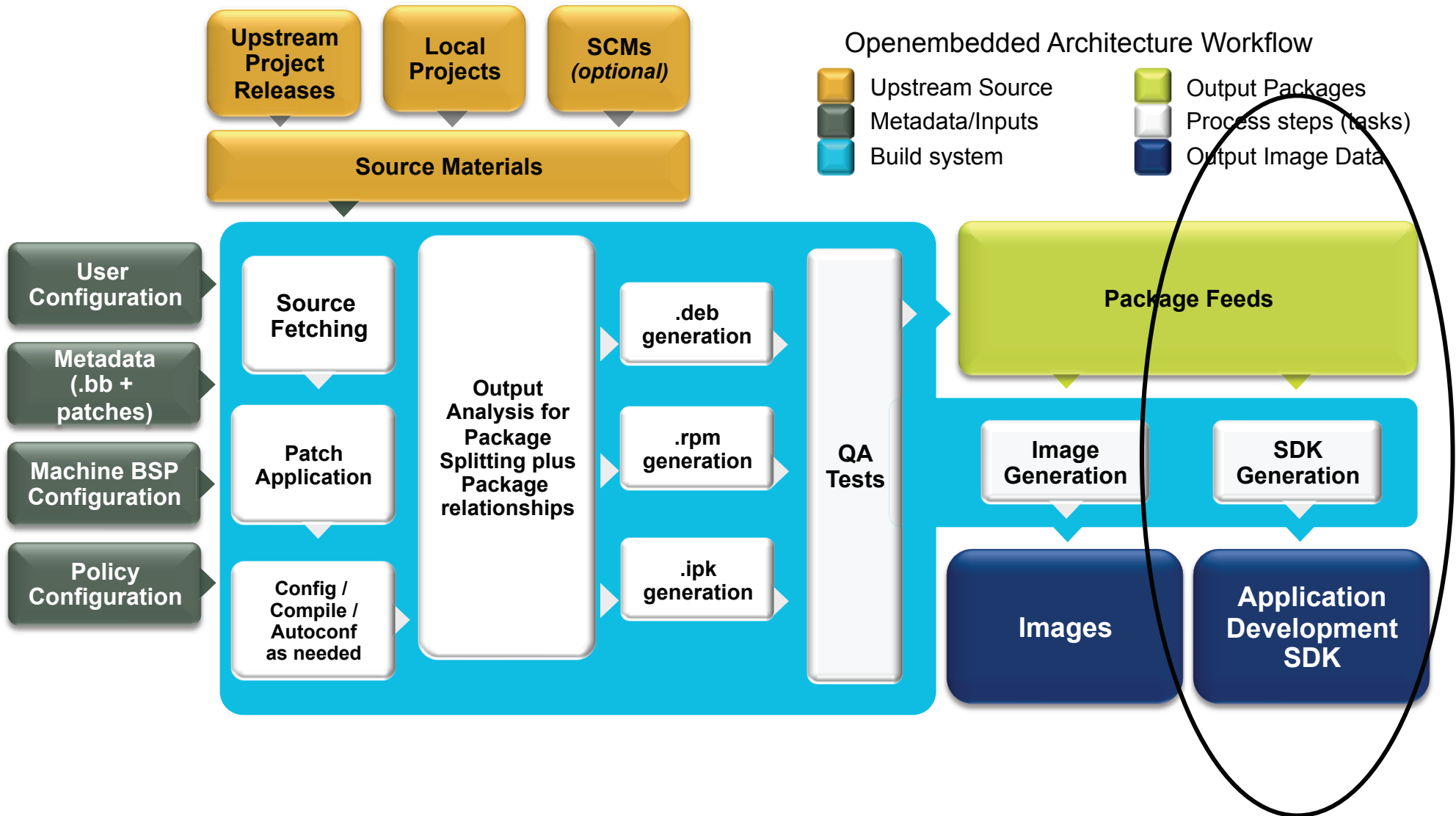
# Image Generation

- Images are constructed using the packages built earlier in the process
- Decision of what to install on the image is based on a defined set of required components and automatically resolved dependent components
- Uses for these images:
  - Live Image to boot a device
  - Root filesystem for QEMU emulator
  - Sysroot for App development



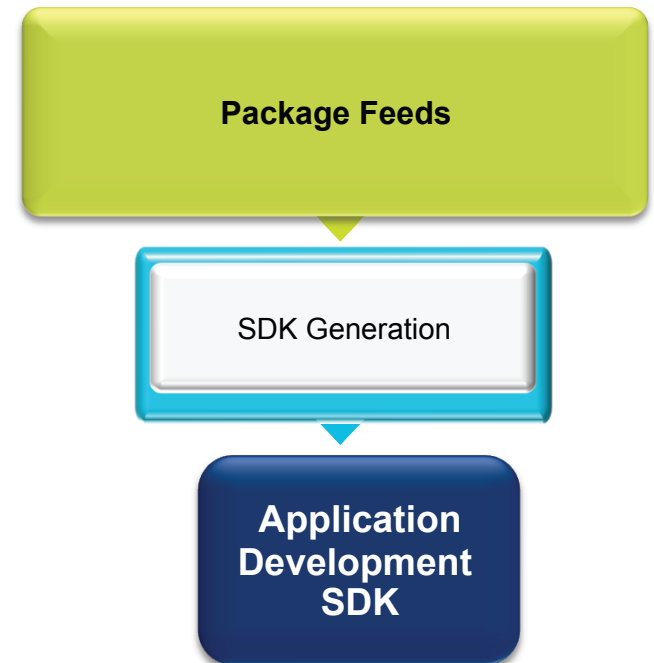
*The Yocto Project allows you customize your embedded Linux OS*

# ADT Generation

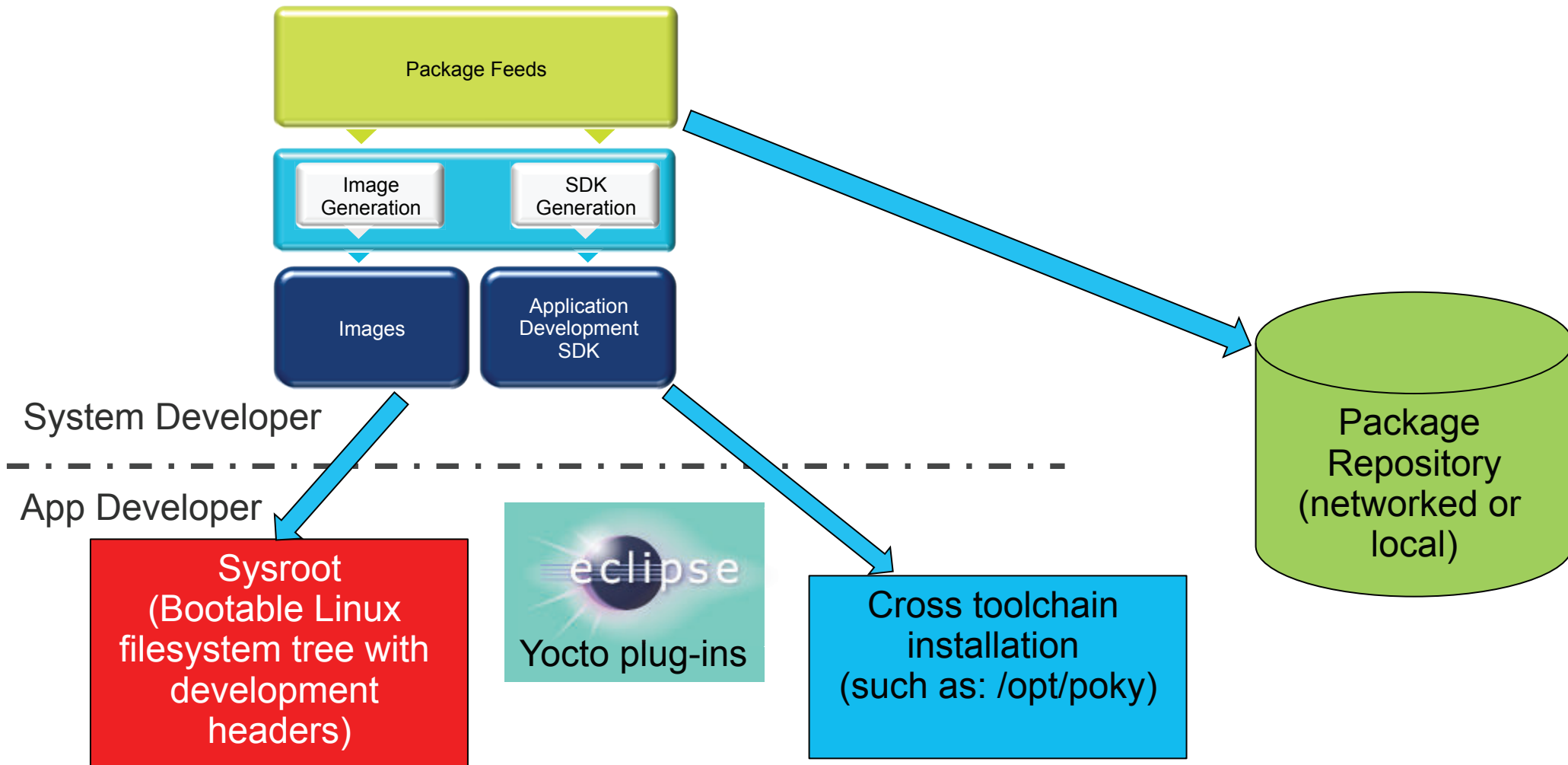


# ADT Generation

- Cross toolchain and installation script generated.
- This can be used to set up an application developer's cross development environment to create apps
- `MACHINE=qemuarm bitbake poky-image-sato-sdk meta-toolchain package-index`
- QEMU built for target architecture emulation

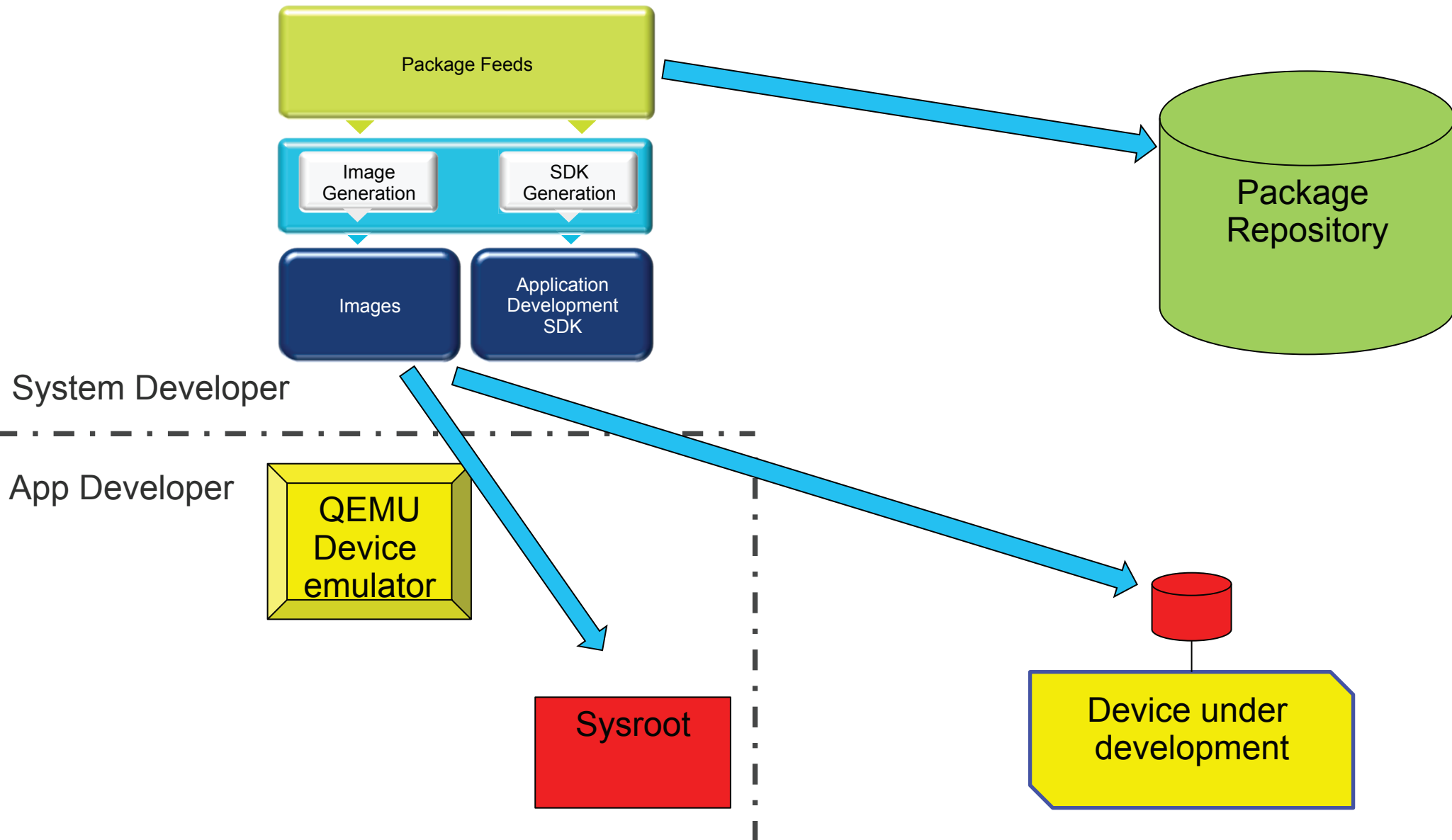


# Setting up the App Developer



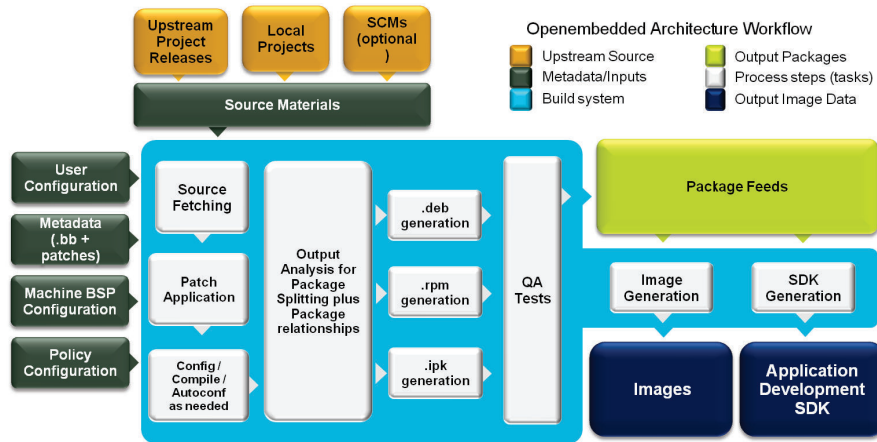
*Yocto Project helps set up the embedded app developer*

# Use NFS/Local Disk, Pkg Manager



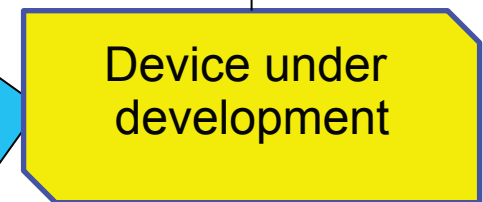
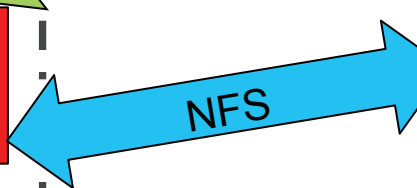
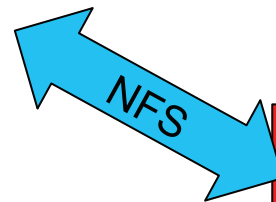
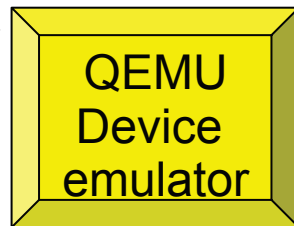


# Use NFS/Local Disk, Pkg Manager

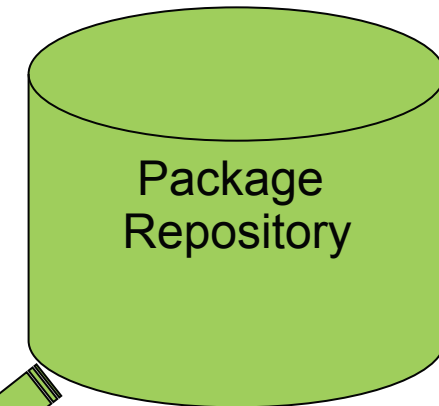
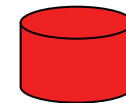


System Developer

App Developer



Package Manager



Both Device and App Development Models Supported

# What's Next?

## The y-HOB – Human Oriented Builder – for v1.1

Machine: atom-pc

Base image: core-image-minimal

Package Collections Packages

Package	Description	License	Group	Included
subversion	subversion version 1.0.15-10	Apache 2.0	console/network	<input type="checkbox"/>
sudo	Provide limited super user privileges to specific users	ISC & UCB	admin	<input checked="" type="checkbox"/>
sysfsutils	Tools for working with sysfs.	GPLv2 & L	base	<input type="checkbox"/>
syslogd	System Log Daemons	GPLv2+ &	base	<input checked="" type="checkbox"/>
syslinux	syslinux version 4.03-r1	GPLv2+	base	<input type="checkbox"/>
sysprof	sysprof version 1.1.6+git1+38a6af1f0a45e528fd2842983da71e0f	GPLv2	base	<input type="checkbox"/>

Search packages: sysk

Image contents (37 packages):

Package	Brought in by	Included
task-core-boot	core-image-minimal	<input checked="" type="checkbox"/>
zip	glib-2.0	<input checked="" type="checkbox"/>
core-image-minimal		<input checked="" type="checkbox"/>
sudo		<input checked="" type="checkbox"/>
base-files	task-core-boot	<input checked="" type="checkbox"/>
acl	udev	<input checked="" type="checkbox"/>

Reset

Create Image **34**

# How to Get Started

- **Download the software today**
- Be sure you read the Quick Start to set up your system to use the Yocto Project
- Build, test on QEMU or real hardware, develop apps
- Join the community to get help
  - #yocto on freenode and [yocto@yoctoproject.org](mailto:yocto@yoctoproject.org) (<http://lists.yoctoproject.org/listinfo/yocto>)

*Getting started with the Yocto Project is easy*

# Get Involved

- The Yocto Project is a collaboration of individuals, non-profits, and corporations under the Linux Foundation
- We urge you or your organization to join
- [yoctoproject.org/documentation/getting-started](http://yoctoproject.org/documentation/getting-started) has a number of ways to learn and contribute
  - Contribute code, documentation, fix bugs, provide BSPs
  - Use YP for your embedded projects
  - Work with the community to make YP better

Make an impact – collaboration in its purest sense

# It's Time to Take Action

- It's not an embedded Linux distribution – it creates a custom one for you
- The Yocto Project lets you customize your embedded Linux OS
- It helps set up the embedded app developer
- Both device and app development models supported
- Getting started is easy
- Make an impact – collaboration in its purest sense

# Q & A

Thank You!

# WIND RIVER

# Additional Resources

Yocto Project:

<http://www.yoctoproject.org>

OpenEmbedded:

<http://www.openembedded.org>

Eclipse YouTube presentation:

<http://www.youtube.com/watch?v=3ZIOu-gLsh0>

Dirk Hohndel – LinuxCon 2011:

<http://www.youtube.com/watch?v=BCxCnrcXctk>