Building Images with Yocto Project

e-ale

What is the Yocto Project?

- A Linux Foundation project that helps you build your own custom Linux distribution, from source to installable image
- A collection of tools to make building your own custom Linux distribution easier
- “yocto” is the smallest unit prefix in the SI system (10^{-24})

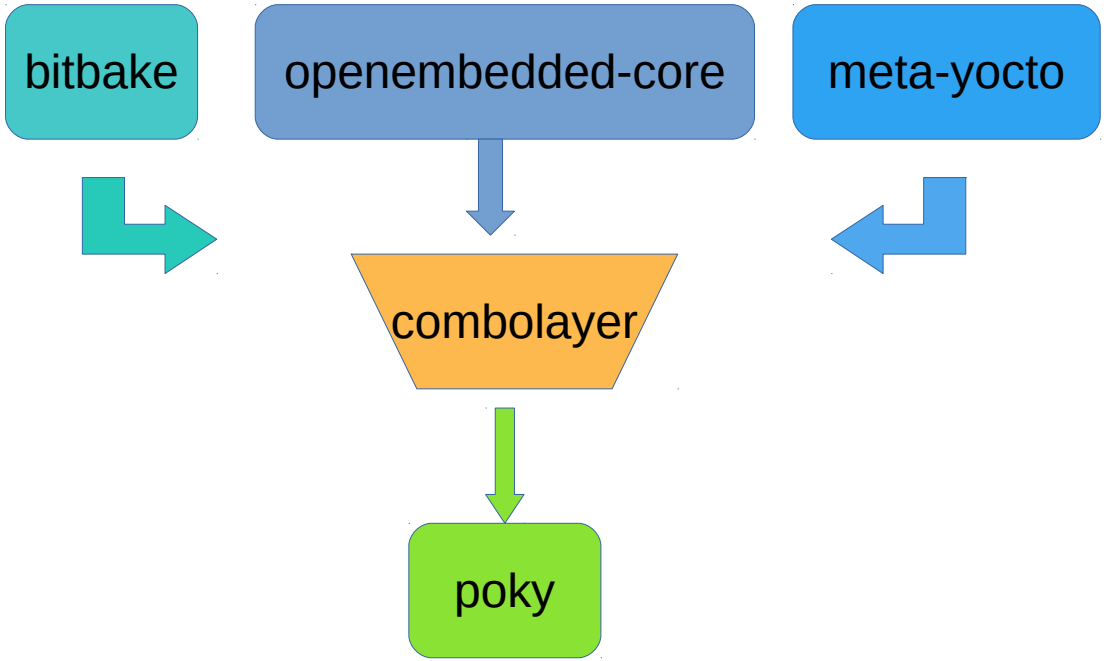
What is Open Embedded?

- A build system, using the “bitbake” tool, to create a Linux distribution
  - Inspired by Gentoo portage system
- A long time ago, in a galaxy far far away, it was what came to be known as Open Embedded Classic
- A not for profit organization which aims to champion embedded Linux

What is poky?

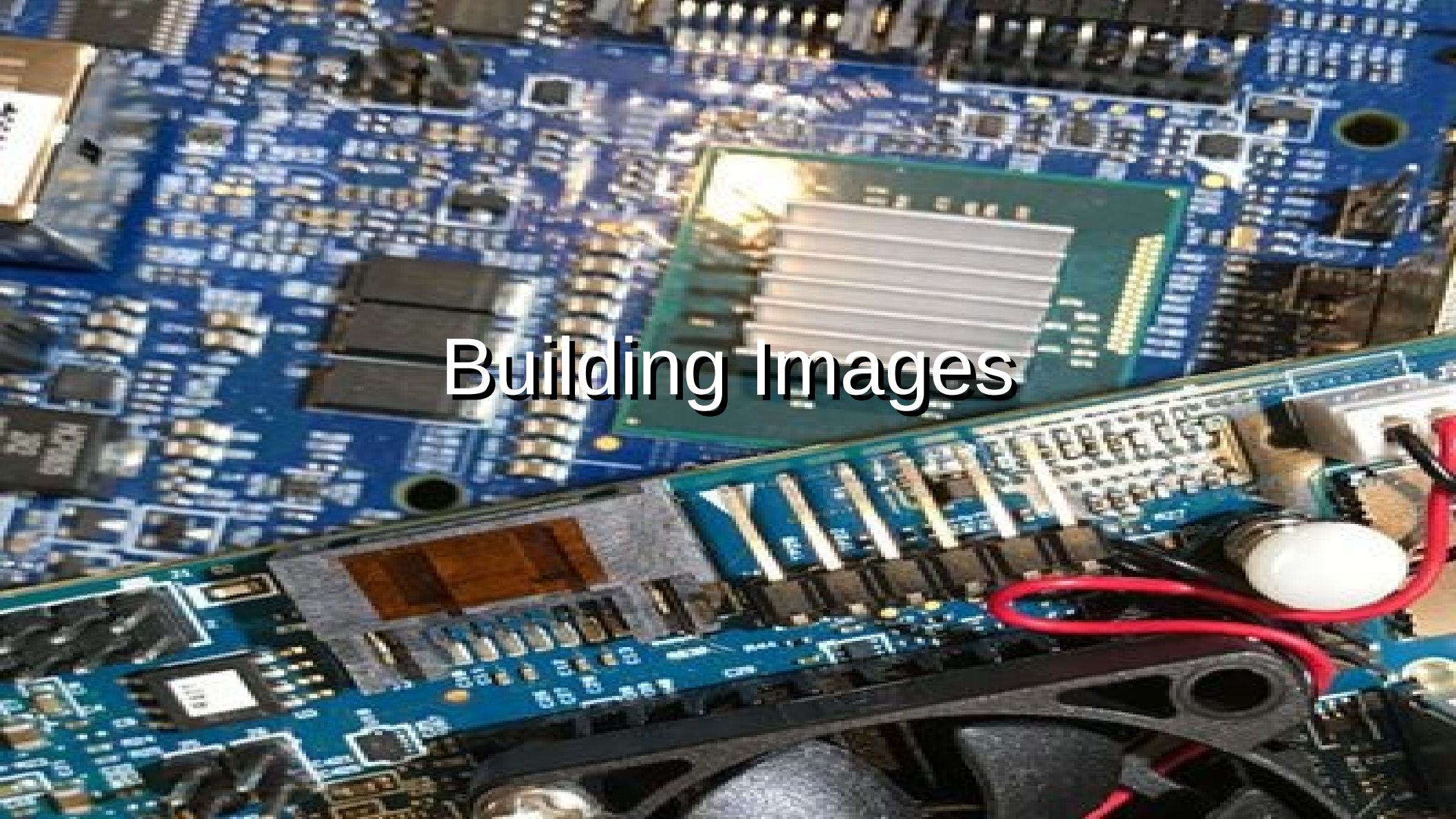
- “poky” is the reference distribution used to make sure the Yocto Project is “all systems normal”
- A conveniently and legally different name for something that sounds the same as a chocolate dipped cookie stick popular in Japan
- Not an equine companion to a green clay-mation character from a popular children’s Saturday morning show that sounds like it might be not so fast

What is poky?



What is poky?

```
$ tree -L 1 poky
poky
├── bitbake
├── documentation
├── LICENSE
├── meta
├── meta-poky
├── meta-selftest
├── meta-skeleton
├── meta-yocto-bsp
├── oe-init-build-env
├── README.hardware
├── README.LSB
├── README.poky
└── scripts
```



Building Images

What do we mean by “image”?

- /boot (and boot loader)
- kernel (and kernel modules)
- rootfs
  - /etc
  - /var
  - /usr

Cloning our tools and metadata

```
git clone bitbake
```

```
git clone openembedded-core
```

OR

```
git clone poky
```

```
git clone meta-foo
```

Setting up our build environment

- Without any options
  - `. ./poky/oe-init-build-env`

```
MACHINE ??= “qemux86”
```

```
TOP_DIR = “./build”
```

Typical Image Building Commands

```
$ bitbake core-image-minimal
```

- minimal bootable image
- small partitions, not for on-target development

```
$ bitbake core-image-base
```

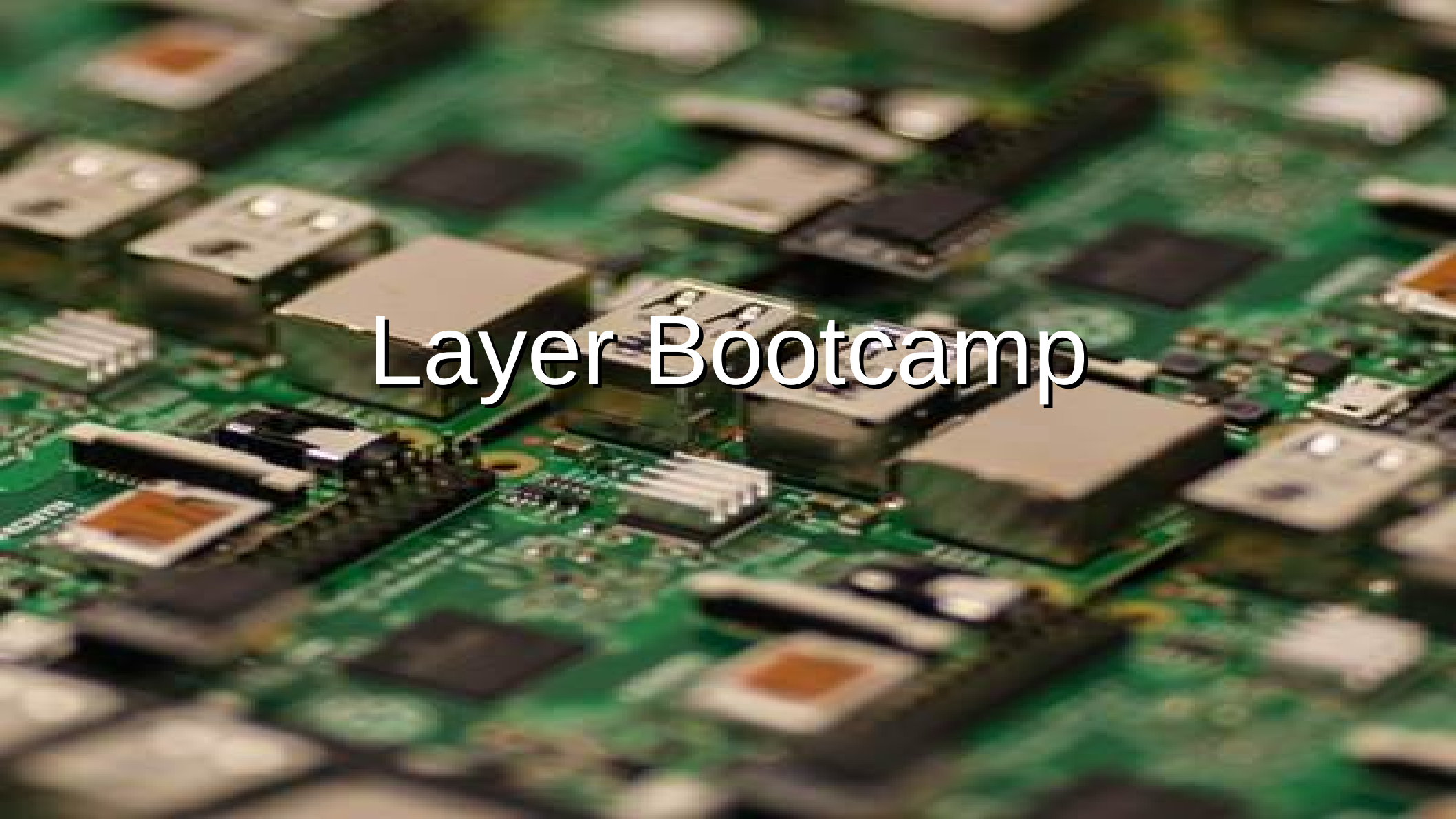
- more typical basic “server” or console image

```
$ bitbake core-image-full-cmdline
```

- more tools (editors)

```
$ bitbake core-image-sato
```

- reference graphical desktop



Layer Bootcamp

Why layers?

- Open Embedded Classic
  - monolithic repository with everything
  - and the kitchen sink
- Needed
  - Flexibility
  - Modularity
  - Distributed

Levels of Abstraction

DISTRO	MACHINE	IMAGE	RECIPE
musl systemd x11	kernel bootloader drivers	console graphics automotive	scripts applications libraries support

Thank you to Stephano Cetola for this content

Layers of Abstraction

DISTRO “Distro layer”	HARDWARE “BSP layer”	SOFTWARE “Application layer”	CONCEPT “Functional layer”
meta-my-distro	*meta-my-arm*	*meta-my-apps*	*meta-my-extra*
X11 or wayland	LCD	demos	manufacturing
systemd or sysvinit	bt / wifi / nfc	customer app	OTA updates
	automotive	support tools	Support tools

Borowed from/Inspired by Stephano Cetola

What is a layer?

- Special organization of metadata and configuration files

```
meta-skeleton
├── conf
│   ├── layer.conf
│   ├── multilib-example2.conf
│   └── multilib-example.conf
├── COPYING.MIT
├── recipes-core
│   └── busybox
├── recipes-kernel
│   ├── hello-mod
│   └── linux
├── recipes-multilib
│   └── images
└── recipes-skeleton
    ├── service
    └── useradd
```

Distro layer

```
meta-e-ale-distro
├── conf
│   ├── bblayers.conf.sample
│   ├── distro
│   │   ├── e-ale.conf
│   │   └── e-ale-tiny.conf
│   ├── layer.conf
│   └── local.conf.sample
├── COPYING.MIT
├── README
└── recipes-core
    └── base-files
        ├── base-files
        │   ├── issue
        │   └── issue.net
        └── base-files_%.bbappend
```

BSP layer

```
meta-e-ale-bsp
├── conf
│   ├── layer.conf
│   └── machine
│       ├── include
│       │   ├── ti33x.inc
│       │   └── ti-soc.inc
│       └── pocketbeagle.conf
├── COPYING.MIT
├── README
├── recipes-bsp
│   └── u-boot
│       ├── u-boot
│       │   └── ...
│       └── u-boot_2018.01.bbappend
├── recipes-kernel
│   └── linux
│       ├── linux-pocketbeagle
│       │   ├── ...
│       │   └── defconfig
│       └── linux-pocketbeagle_4.14.bb
└── wic
    └── pocketbeagle-yocto.wks
```

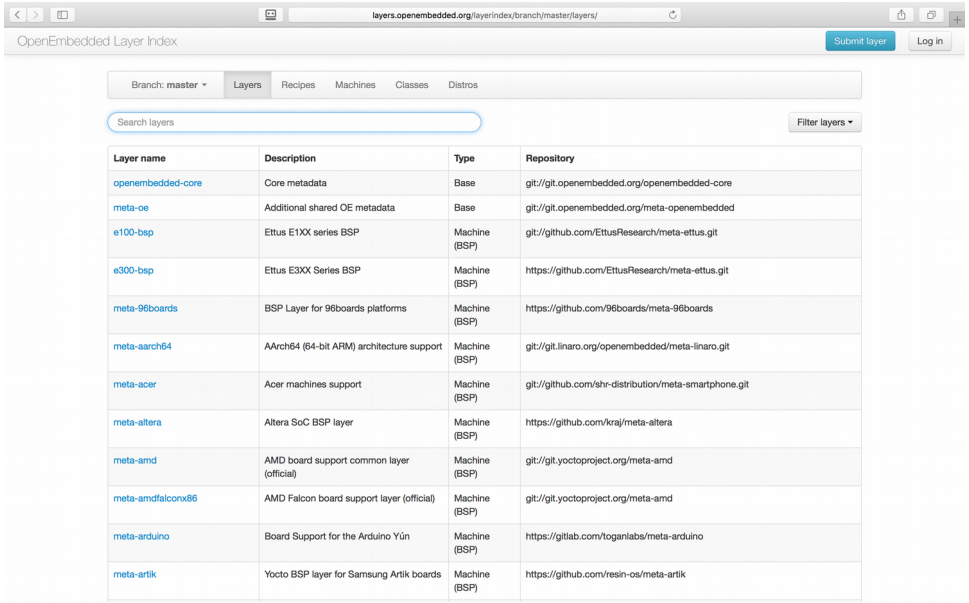
How to create a layer?

```
$ bitbake-layers create-layer meta-foo
```

- Manually
  - Clone/Copy another layer
  - Keep it consistent (Distro vs. BSP vs. Application)

What layers are out there?

- Layer index - http://layers.openembedded.org/
- Searchable:
  - Layers
  - Machines
  - Recipes
  - Classes



Layer name	Description	Type	Repository
openembedded-core	Core metadata	Base	git://git.openembedded.org/openembedded-core
meta-oe	Additional shared OE metadata	Base	git://git.openembedded.org/meta-openembedded
e100-bsp	Ettus E1XX series BSP	Machine (BSP)	git://github.com/EttusResearch/meta-ettus.git
e300-bsp	Ettus E3XX Series BSP	Machine (BSP)	https://github.com/EttusResearch/meta-ettus.git
meta-96boards	BSP Layer for 96boards platforms	Machine (BSP)	https://github.com/96boards/meta-96boards
meta-aarch64	AArch64 (64-bit ARM) architecture support	Machine (BSP)	git://git.linaro.org/openembedded/meta-linaro.git
meta-acer	Acer machines support	Machine (BSP)	git://github.com/shr-distribution/meta-smartphone.git
meta-altera	Altera SoC BSP layer	Machine (BSP)	https://github.com/kraj/meta-altera
meta-amd	AMD board support common layer (official)	Machine (BSP)	git://git.yoctoproject.org/meta-amd
meta-amdfalconx86	AMD Falcon board support layer (official)	Machine (BSP)	git://git.yoctoproject.org/meta-amd
meta-arduino	Board Support for the Arduino Yún	Machine (BSP)	https://gitlab.com/toganlabs/meta-arduino
meta-artik	Yocto BSP layer for Samsung Artik boards	Machine (BSP)	https://github.com/resin-os/meta-artik

Application Developer Workflow Tools

Eclipse IDE plugin

- eclipse-yocto (formerly eclipse-poky)
  - Autotools
  - Cmake
  - Makefile
  - GDB
- Later in 2018 (2.6): Leveraging Containers

CROPS Docker Containers

- CROss PlatformS
- “Containers Run Other Peoples’ Software”
- Docker containers which support
  - Windows**
  - Mac**
  - Linux

- Based on supported Distributions
  - Fedora, Debian, Ubuntu
- Types of containers
  - yocto-base
  - yocto-builder
  - eoky-container
  - esdk-container

**with Samba container (Linux file system)

CROPS presentation

- ELC 2017
  “Cross Platform Enablement for the Yocto Project with Containers”
  Randy Witt, Intel
- PDF
  - https://elinux.org/images/9/94/2017_ELC_-_Yocto_Project_Containers.pdf
- YouTube
  - https://www.youtube.com/watch?v=JXHLAWveh7Y

SDK (Software Development Kit)

- “Toolchain”
- Create your own

```
$ bitbake image -c populate_sdk
```

- Use Yocto Project releases
  - https://downloads.yoctoproject.org/releases/yocto/yocto-2.4.2/toolchain/

eSDK (Extensible Software Development Kit)

- Create your own

```
$ bitbake image -c populate_sdk_ext
```

- Use Yocto Project releases
  - https://downloads.yoctoproject.org/releases/yocto/yocto-2.4.2/toolchain/
  - Look for files with “ext” in the filename

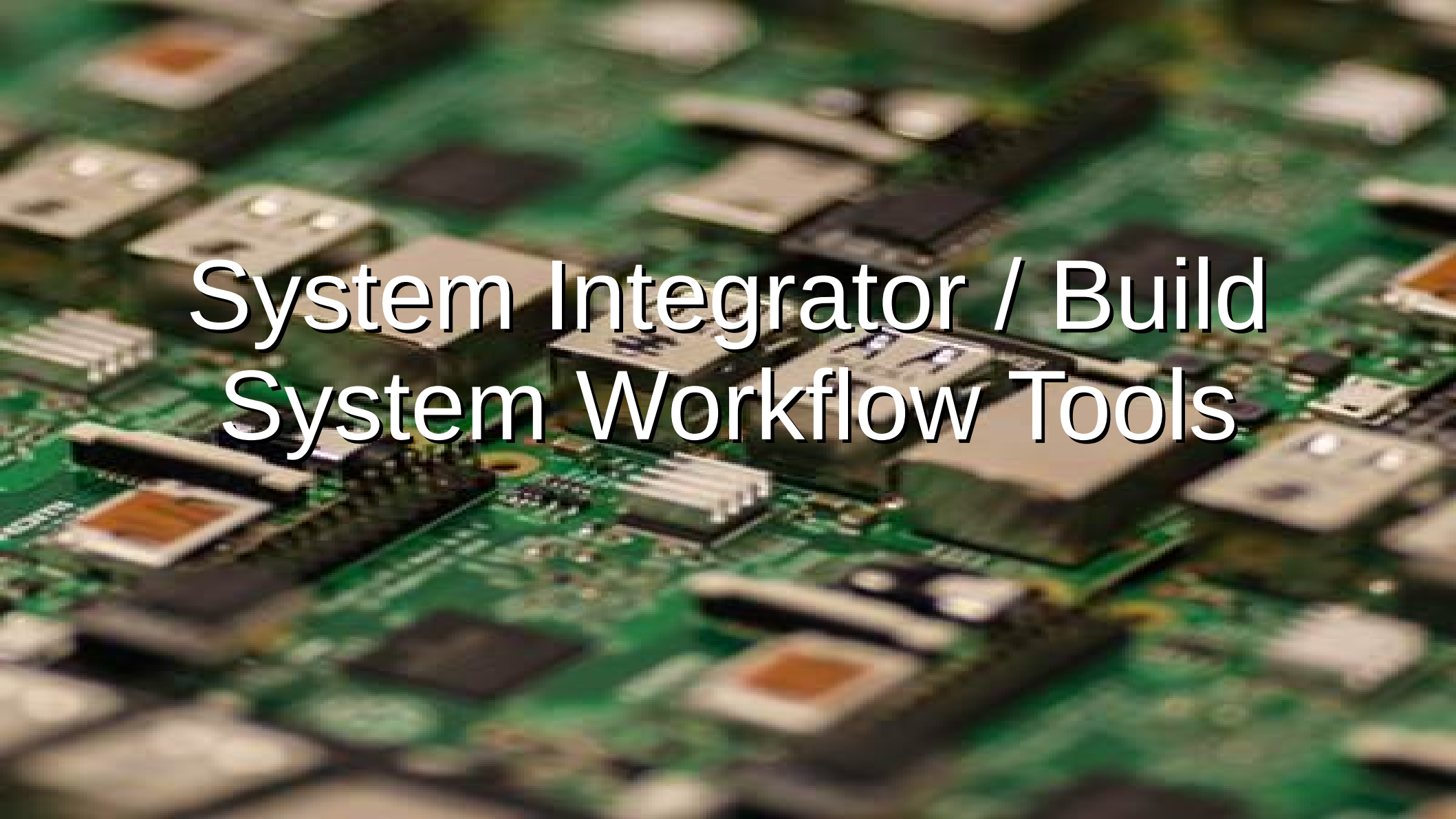
https://www.yoctoproject.org/docs/2.4.2/sdk-manual/sdk-manual.html

Feature	Standard SDK	Extensible SDK
Toolchain	Yes	Yes*
Debugger	Yes	Yes*
Size	100+ MBytes	1+ Gbytes (or 300+ Mbytes for minimal w/toolchain)
devtool	No	Yes
Build Images	No	Yes
Updateable	No	Yes
Managed Sysroot**	No	Yes
Installed Packages	No***	Yes****
Construction	Packages	Shared State

```
   * Extensible SDK will contain the toolchain and debugger if SDK_EXT_TYPE is "full" or
     SDK_INCLUDE_TOOLCHAIN is "1", which is the default.
  ** Sysroot is managed through use of devtool.  Thus, it is less likely that you will corrupt
     your SDK sysroot when you try to add additional libraries.
 *** Runtime package management can be added to the standard SDK but it is not supported by
     default.
**** You must build and make the shared state available to extensible SDK users for "packages"
     you want to enable users to install.
```

eSDK presentation

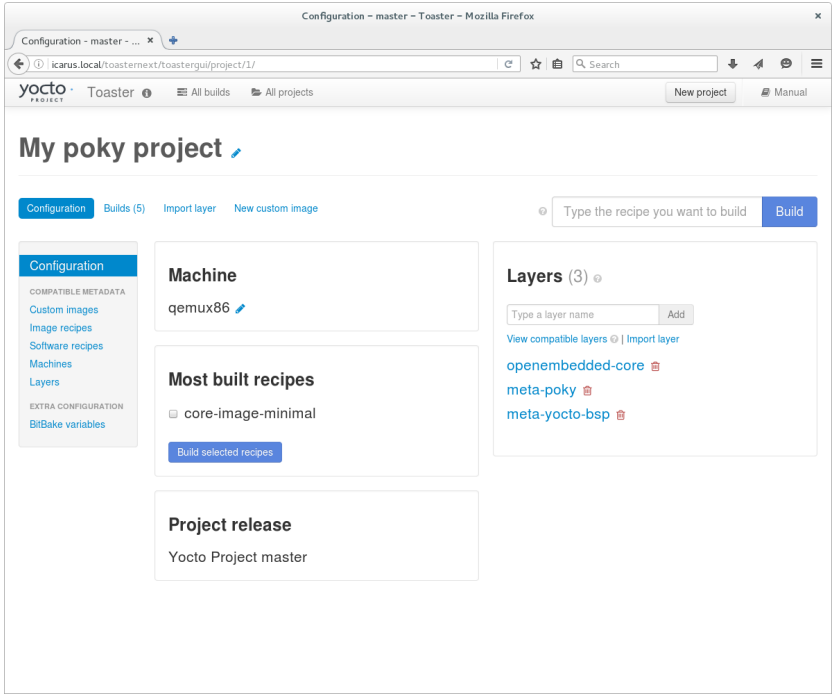
- ELC 2017
  “Yocto Project Extensible SDK:Simplifying Workflow for Application Developers”
  Henry Bruce, Intel
- PDF
  - https://elinux.org/images/7/7a/2017_ELC_Henry_Bruce.pdf
- YouTube
  - https://www.youtube.com/watch?v=d3xanDJuXRA



System Integrator / Build System Workflow Tools

Toaster

- Web UI tool to build and customize images



devtool

- Developer tool to help minimize repetitive tasks
- Great self-documentation

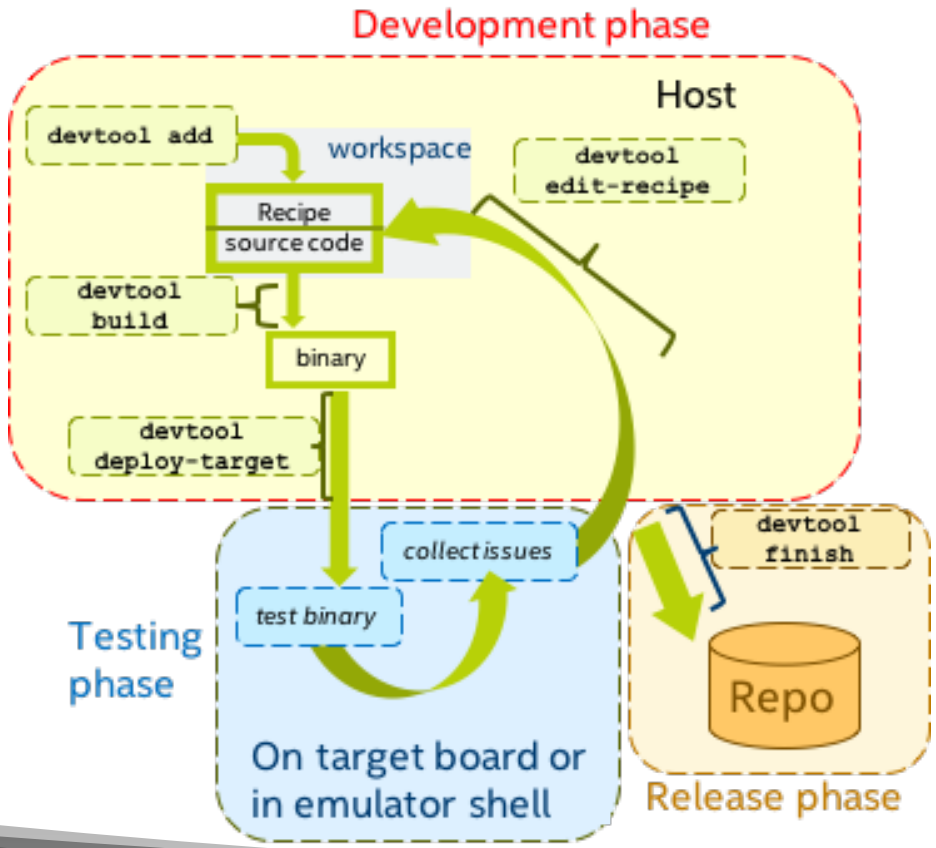
```
$ devtool [cmd] help
```

- Common commands

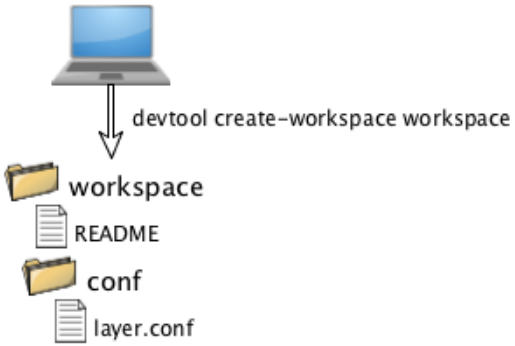
```
$ devtool add my-app <URI to source>
$ devtool modify other-app
$ devtool upgrade existing-recipe
```

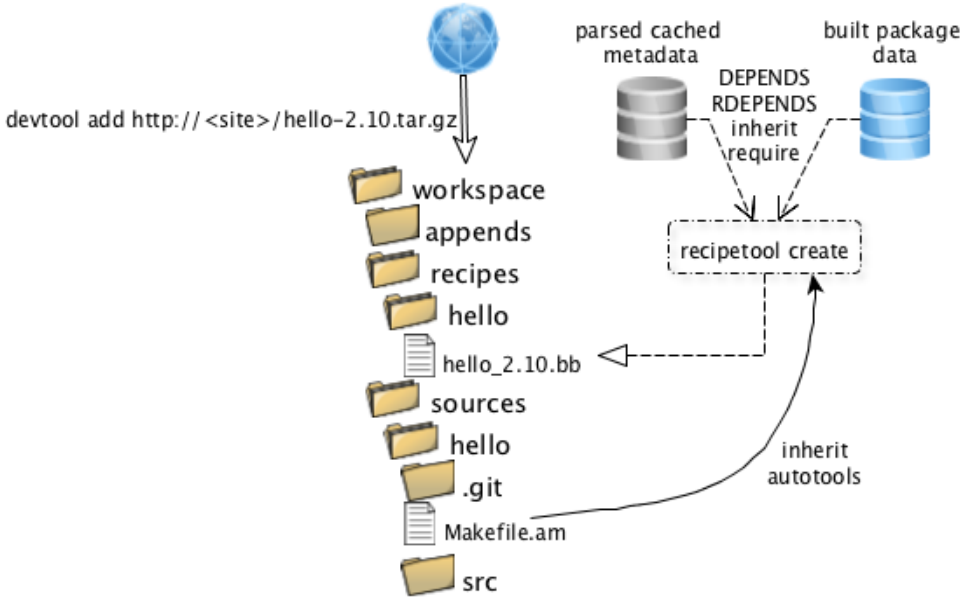
Typical `devtool` workflow

- Add
- Build
- Test
- Edit?
- Commit



Overview of how `devtool` works





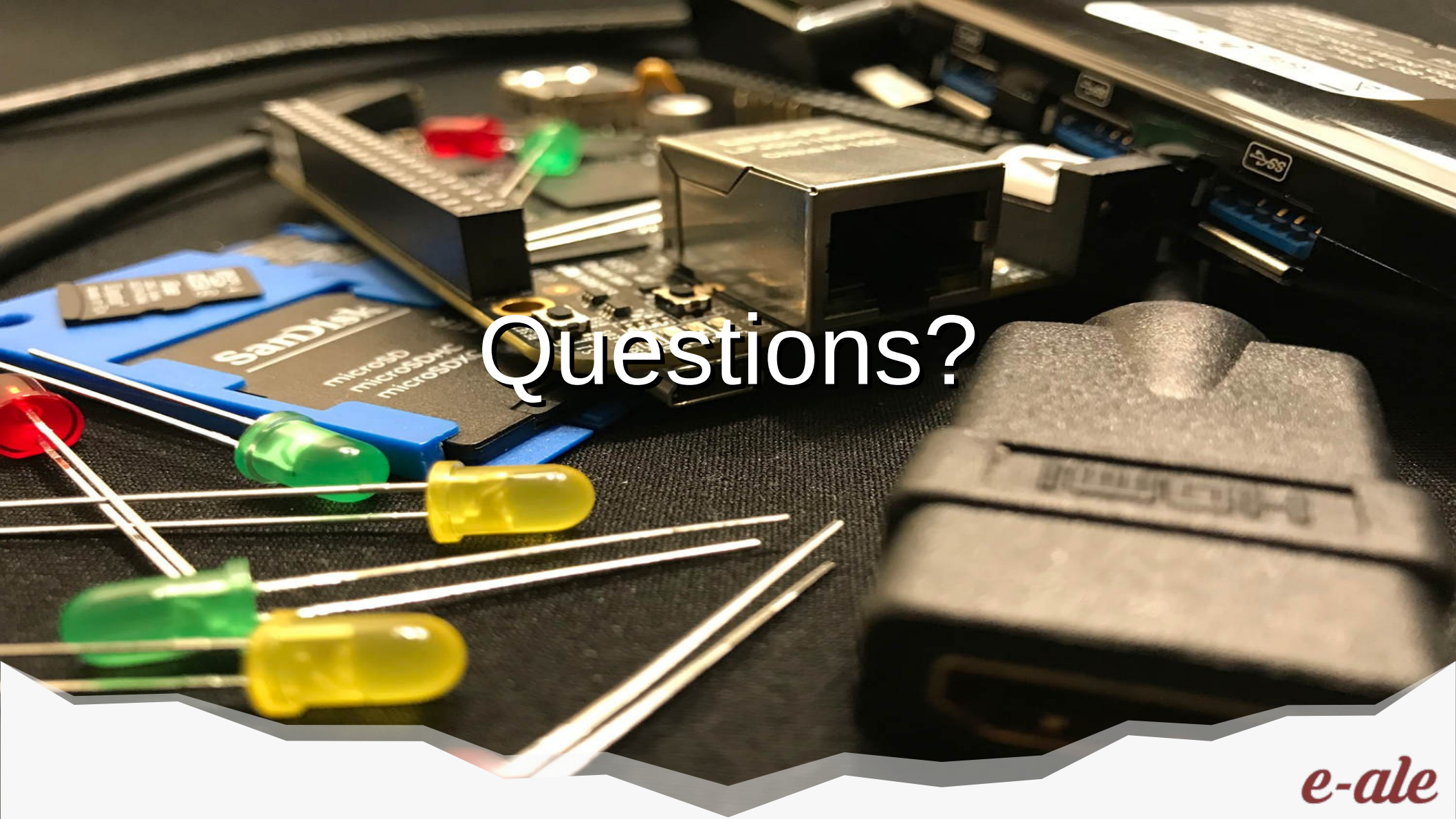
`devtool` presentation

- ELC 2017
  “Using Devtool to Streamline Your Yocto Project Workflow”
  Tim Orling, Intel
- PDF
  - https://elinux.org/images/e/e2/2017_ELC_--_Using_devtool_to_Streamline_your_Yocto_Project_Workflow.pdf
- YouTube
  - https://www.youtube.com/watch?v=CiD7rB35CRE

Auto Upgrade Helper (AUH)

- Script that automatically updates recipes using devtool
- Can build for multiple architectures
  - `qemux86_musl`, `qemux86_64`, `qemuarm`, `qemumips`, etc
- Can run tests (`testimage` and/or `ptests`)

http://git.yoctoproject.org/cgit/cgit.cgi/auto-upgrade-helper/



Questions?

e-ale

Thank you!

e-ale