# Working with Kernels in the Yocto Project

Tom Zanussi, Intel

Yocto Project DevDay• San Francisco• 19 Feb 2013

# Overview

- Intro
- Traditional Kernel Recipe
- Custom Kernel Recipes
- linux-yocto Kernel Recipe
- Multi-meta Kernel Recipe
- When to Use Which
- Recipe-space vs Repo-space
- Using Local Clones
- Questions

# Intro

- **Everything covered here is in the Kernel Manual:**
  - http://www.yoctoproject.org/docs/latest/kernel-dev/kernel-dev.html
- **Most is also covered by the 'Hands-on Kernel Lab'**
  - https://www.yoctoproject.org/sites/yoctoproject.org/files/elc2013-kernel-lab.pdf

# Intro (cont'd)

- There's no such thing as a 'Yocto kernel'
- `linux-yocto` recipes point to 'Yocto kernel repos'
- 'Yocto kernel repos' are based on upstream kernels
  - kernel.org linux/linux-stable, ltsi-kernel, etc
  - Patches are temporary and expected to go upstream
  - Unless they're BSP-specific
- Yocto simply adds machinery and metadata on top
- It would be more accurate to say 'Yocto meta-kernel'

# Intro (cont'd)

- There are three 'Yocto kernel repos' at any one time:
  - `linux-yocto_3.2`, `linux-yocto_3.4`, and `linux-yocto-dev`
    - `linux-yocto-dev` tracks current Linus master
  - In Yocto 1.4 this will be '3.4 + LTSI', 3.8, and `linux-yocto-dev`
- LTSI kernels merge into `linux-yocto` when released
  - `linux-yocto_3.4` has the LTSI 3.4 release merged in
- Other kernel recipes point to 'non-Yocto kernel repos':
  - `linux-yocto-custom` handles arbitrary git repos
    - Can also point to tarballs (Yocto 1.4)
  - Traditional tarball-based recipes handle only tarballs

# Traditional Kernel Recipe

- Follows the typical oe-core recipe pattern
  - `SRC_URI` points to a kernel tarball
  - Patches added via `SRC_URI`
- Inherits from `kernel.bbclass`
- Uses `defconfig` as the `.config`
- Consider `linux_3.0.18.bb`:

```
DESCRIPTION = "Mainline Linux Kernel"
SECTION = "kernel"
LICENSE = "GPLv2"

LIC_FILES_CHKSUM = "file://COPYING;md5=d7810fab7487fb0aad327b76f1be7cd7"

inherit kernel

SRC_URI = "${KERNELORG_MIRROR}/linux/kernel/v3.0/linux-${PV}.tar.bz2; \
           name=kernel file://defconfig"
SRC_URI += "file://yocto-testmod.patch"
```

# Traditional Kernel Recipe (cont'd)

## Linux 3.18 tarball

### .config (monolithic defconfig)

```
# CONFIG_64BIT is not set
CONFIG_X86_32=y
# CONFIG_X86_64 is not set
CONFIG_X86=y
CONFIG_INSTRUCTION_DECODER=y
CONFIG_OUTPUT_FORMAT="elf32-i386"
CONFIG_GENERIC_CMOS_UPDATE=y
CONFIG_GENERIC_CLOCKEVENTS=y
CONFIG_LOCKDEP_SUPPORT=y
CONFIG_STACKTRACE_SUPPORT=y
CONFIG_MMU=y
CONFIG_NEED_SG_DMA_LENGTH=y
CONFIG_GENERIC_ISA_DMA=y
CONFIG_GENERIC_BUG=y
CONFIG_GENERIC_HWEIGHT=y
CONFIG_ARCH_MAY_HAVE_PC_FDC=y
CONFIG_ARCH_HAS_CPU_RELAX=y
CONFIG_ARCH_HAS_DEFAULT_IDLE=y
CONFIG_ARCH_SUSPEND_POSSIBLE=y
CONFIG_DRM=y
CONFIG_DRM_I915=y
CONFIG_DRM_I915_KMS=y
CONFIG_USB_EHCI_HCD=y
```

+

### Kernel source (linux-3.18.tar.gz)

```
├── arch
├── block
├── COPYING
├── Documentation
├── drivers
│   └── drm
├── fs
├── include
├── init
├── Kbuild
├── Kconfig
├── kernel
├── lib
├── MAINTAINERS
├── Makefile
├── mm
├── net
├── REPORTING-BUGS
├── scripts
├── security
├── sound
├── tools
└── usr
```

= [b]zImage

# Traditional Kernel Recipe (cont'd)

- To add `CONFIG_*` items you modify the `defconfig`
  - Use `bitbake -c menuconfig` to modify the `.config`
  - Copy the `.config` to the recipe's `defconfig`
  - `diff` the before/after `.config` in `WORKDIR` to see changes
- This method provides a 'fallback' for older releases
  - `linux-yocto-custom` can deal with kernel tarballs (Yocto 1.4)
- See 'Hands-on Kernel Lab' lab1 for a worked example

# `linux-yocto-custom` Kernel Recipe

- Provides a wrapper for any git-based Linux kernel
  - The `SRC_URI` points to a git clone and `defconfig`
- Adds Yocto kernel tooling to any kernel
  - Enables 'config fragments'
  - Here's a config fragment named `smp.cfg`:

```
CONFIG_SMP=y
CONFIG_SCHED_SMT=y
```

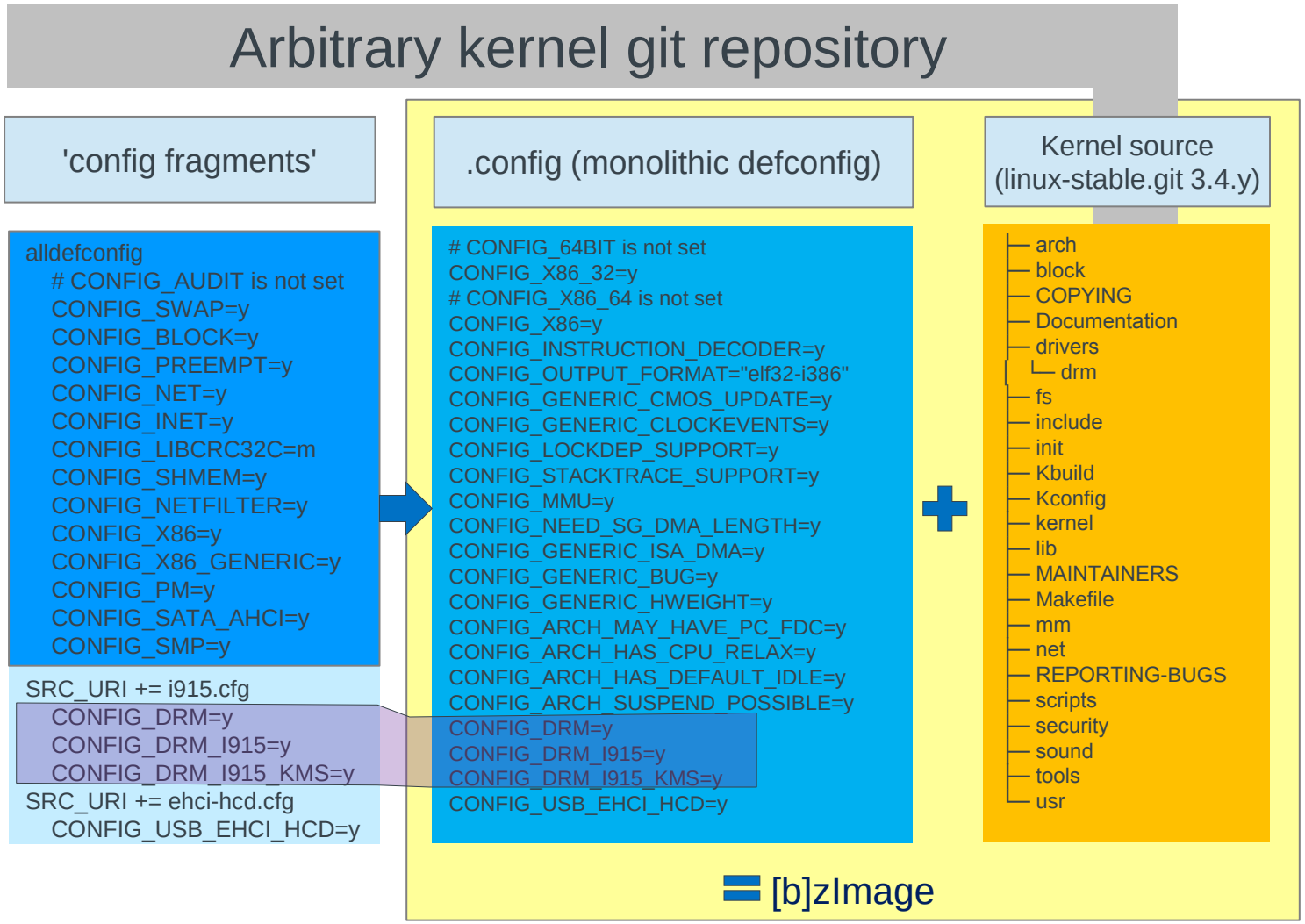
  - To add `CONFIG_*` items add a `.cfg` file to the `SRC_URI`
- Typically just a modified copy of `linux-yocto-custom.bb`
  - Found in `meta-skeleton/recipes-kernel/linux`
- See 'Hands-on Kernel Lab' lab3 for a worked example

# `linux-yocto-custom` Kernel Recipe (cont'd)

- Consider this `linux-yocto-custom` recipe:
- It builds branch '`linux-3.4.y`' of `linux-stable.git`

```
inherit kernel
require recipes-kernel/linux/linux-yocto.inc

SRC_URI = "git://git.kernel.org/pub/linux/kernel/git/stable/linux-stable.git; \
          protocol=git;bareclone=1"

SRC_URI += "file://defconfig file://smp.cfg"

KBRANCH = "linux-3.4.y"

LINUX_VERSION ?= "3.4.28"
LINUX_VERSION_EXTENSION ?= "-custom"

SRCREV="${AUTOREV}"

PR = "r0"
PV = "${LINUX_VERSION}+git${SRCPV}"

COMPATIBLE_MACHINE_lab3-qemux86 = "lab3-qemux86"
```

# `linux-yocto-custom` Kernel Recipe (cont'd)



# `linux-yocto` Kernel Recipe

- Points to a 'Yocto kernel repo'
- Adds Yocto kernel tooling and ready-to-use objects:
  - Config fragments
  - Kernel features
    - Named 'config fragments‘ + patches in an `.scc` file
  - Kernel types
    - Collections of objects defining common policy or capabilities
- The `SRC_URI` points to a git clone and 2 branches:
  - 'machine' branch – kernel source for one or more BSPs
    - Base branch plus patches (commits in git)
  - 'meta' branch – orphan branch containing shared 'objects':

# `linux-yocto` Kernel Recipe (cont'd)

- ready-to-use pool of config fragments and kernel features
- BSP definitions (top-level kernel features)
- The combined sum produces the final `.config`

- To add `CONFIG_*` items
  - Add a `.cfg` to the 'meta' branch (preferred)
  - Can also do this via `SRC_URI`, as before
- To add a patch
  - Add it directly to the machine branch (preferred)
  - Or add it to an `.scc` file using the '`patch`' command
  - Or add it via the `SRC_URI`, as before
- See 'Hands-on Kernel Lab' lab2 for a worked example

# `linux-yocto` Kernel Recipe (cont'd)

- Consider this `linux-yocto` recipe:
- It builds branch `KBRANCH` using metadata `KMETA`

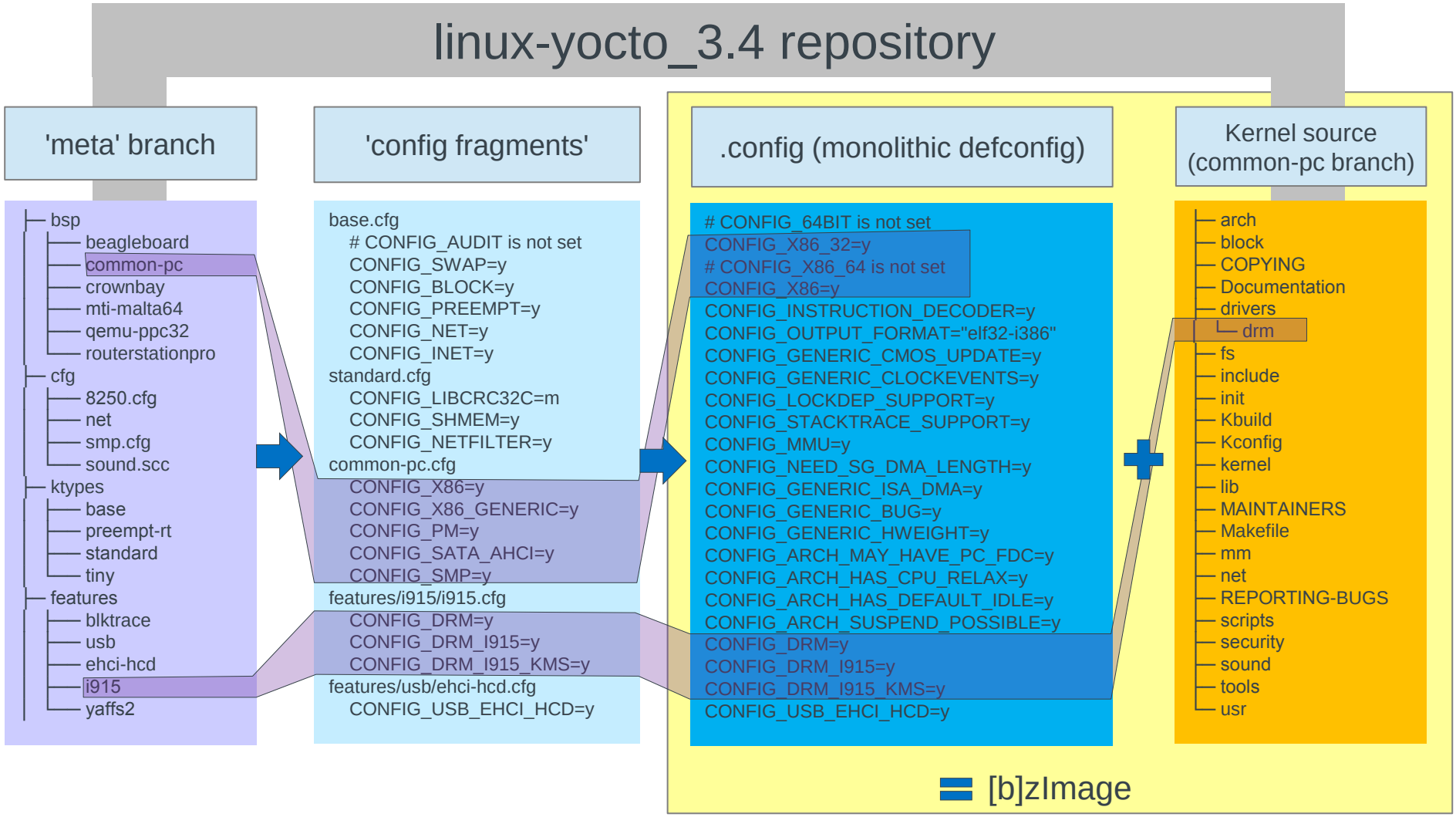
```
require recipes-kernel/linux/linux-yocto.inc

KBRANCH_DEFAULT = "standard/base"
KBRANCH = "${KBRANCH_DEFAULT}"

SRCREV_machine ?= "13809f2cfd9be0ce86bd486e1643f9b90bed6f4f"
SRCREV_meta ?= "f697e099bc76d5df3a307a5bc0cc25021dd6dfe0"

SRC_URI = "git://git.yoctoproject.org/linux-yocto-3.4.git;protocol=git; \
           bareclone=1;branch=${KBRANCH},${KMETA};name=machine,meta"

LINUX_VERSION ?= "3.4.28"

PR = "${INC_PR}.3"
PV = "${LINUX_VERSION}+git${SRCPV}"

KMETA = "meta"

COMPATIBLE_MACHINE = "qemuarm|qemux86|qemuppc|qemumips|qemux86-64"
```

# `linux-yocto` Kernel Recipe (cont'd)

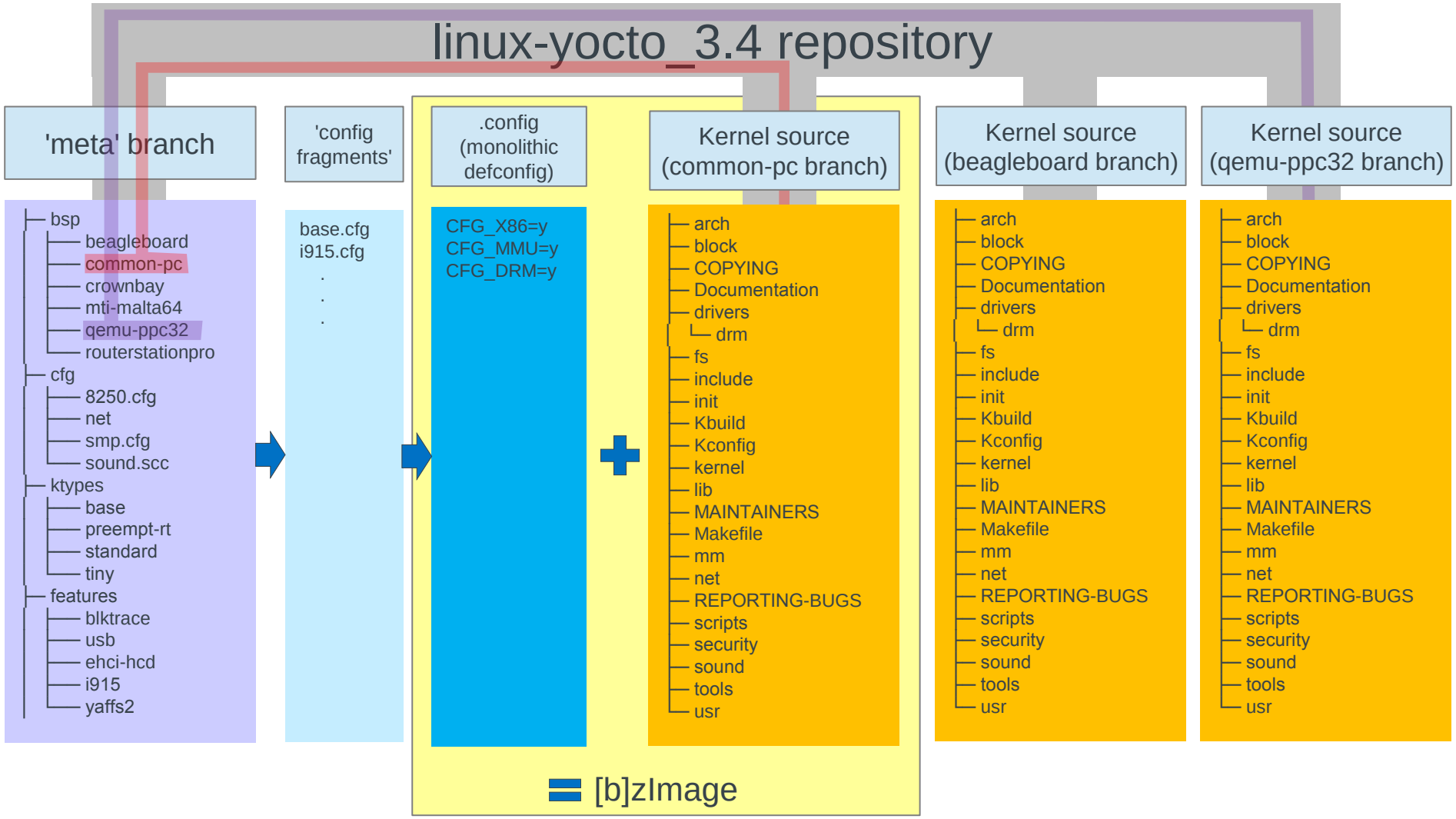
- Normally we modify it via `.bbappends`
- Consider this `linux-yocto_3.4 bbappend`:
  - It overrides the `KBRANCH` making it BSP-specific
  - It adds a 'kernel feature' (`i915`)

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

COMPATIBLE_MACHINE_sugarbay = "sugarbay"

KMACHINE_sugarbay  = "sugarbay"
KBRANCH_sugarbay  = "standard/common-pc-64/sugarbay"

KERNEL_FEATURES_append_sugarbay += " features/i915/i915.scc"

SRCREV_machine_pn-linux-yocto_sugarbay ?= "13809f2cfd9be0ce1643f9b90bed6f4f"
SRCREV_meta_pn-linux-yocto_sugarbay ?= "f697e099bc76d5df3a3c25021dd6dfe0"

LINUX_VERSION = "3.4.28"
```

# linux-yocto Kernel Recipe (cont'd)



# linux-yocto Kernel Recipe (more realistic view)

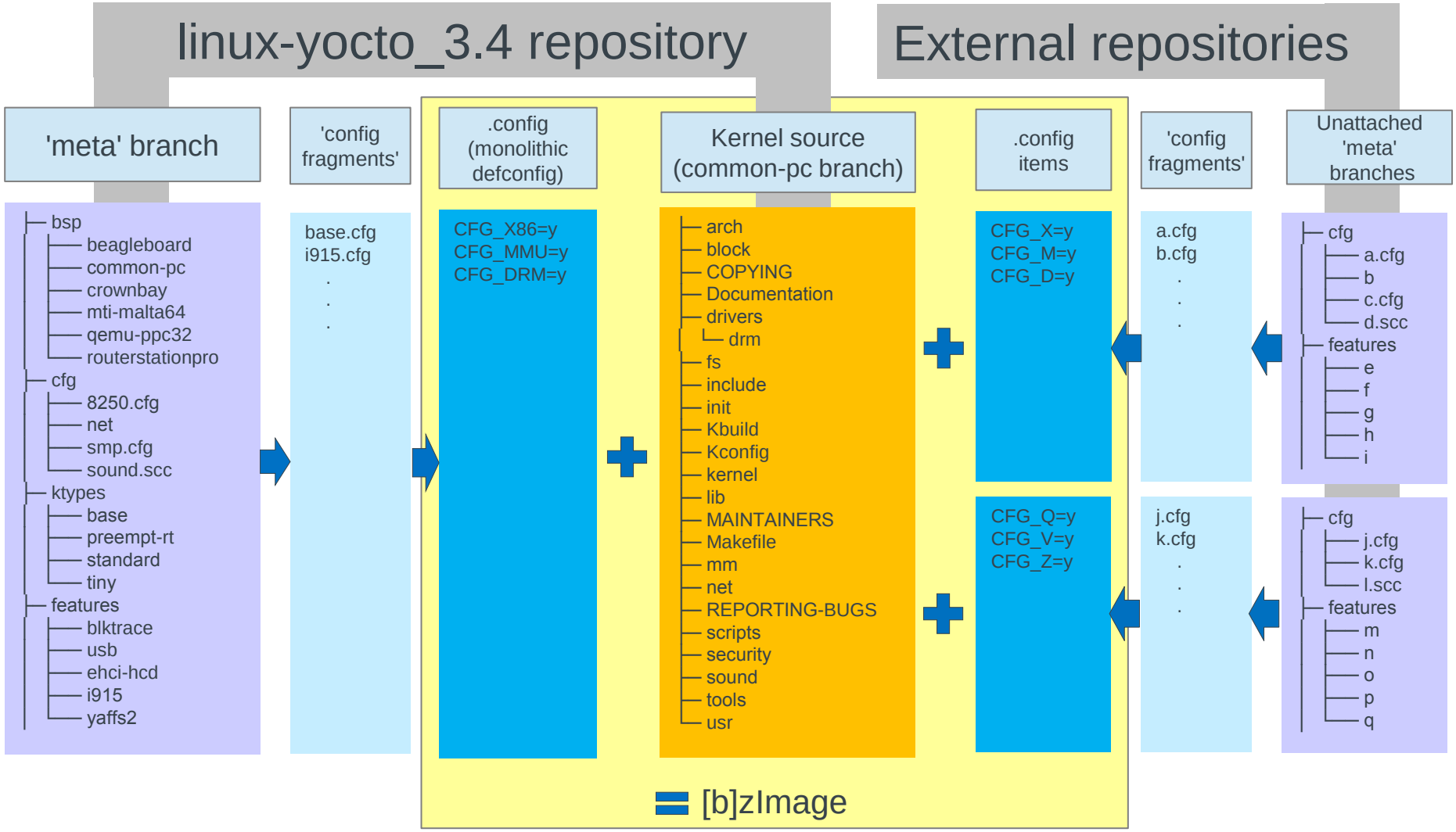


# Multi-meta Kernel Recipe

- In 1.4 a recipe can name multiple 'meta' branches
  - Promotes reuse and sharing of config fragments
- Consider this `linux-yocto_3.4` bbappend:
  - It adds two more 'kernel feature' sources

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

SRC_URI += "git://git.yoctoproject.org/yocto-kernel-cache;protocol=git; \
            branch=master;type=kmeta;name=feat1;destsuffix=kernel-cache/"
SRC_URI += "git://${KSRC_linux_yocto_3_4};protocol=file;branch=meta; \
            name=feat2;type=kmeta;destsuffix=kernel-features-experimental/"

SRCREV_feat1 = "${AUTOREV}"
SRCREV_feat2 = "${AUTOREV}"

KERNEL_FEATURES_append += " features/from_feat1/a_feature1_feature.scc"
KERNEL_FEATURES_append += " features/from_feat2/a_feature2_feature.scc "
```

# Multi-meta Kernel Recipe (cont'd)



# When to Use Which

- **If you have a tarball-based kernel**
  - **Use `linux-yocto-custom` (1.4)**
  - **Use a traditional recipe as a 'fallback' in older releases**
- **If you have a git-based kernel and defconfig**
  - **Use `linux-yocto-custom` for immediate results**
  - **This gives you the ability to add config fragments et al**
- **Consider moving your kernel to `linux-yocto`**
  - **Continually updated with fixes including security**
  - **Direct access to config and feature pool in 'meta'**
  - **You can do it all in 'recipe-space' first**

# linux-yocto Features

```
[trz@empanada kernel-cache]$ ls features
amt          eg20t        inline       kvm          power        tipc
blktrace     emgd         intel-dpdk   latencytop   powertop     uio
bsdjail      ericsson-3g  intel-e1xxxx lttng        pramfs       unionfs
cgroups      ftrace       ipmi         mac80211     profiling    uprobe
ciphers      fuse         irq          namespaces   pvr          uptime
cpuisol      gma500       iwlagn       netfilter    ramconsole   usb
crypto       grsec        iwlwifi      net_sched    revoke       usb-net
dca          hostapd      ixgbe        nfsd         rt           utrace
drm-emgd     hrt          kexec        ocf          scsi         vfat
drm-gma500   hugetlb      kgdb         oprofile     seccomp      xip
drm-psb      i915         kmemcheck    pci          serial       yaffs2
edac         igb          kprobes      pci-iov      systemtap
edf          initramfs    ktest        perf         taskstats
[trz@empanada kernel-cache]$ tree features/i915/
i915
├── i915.cfg
└── i915.scc
[trz@empanada kernel-cache]$ cat features/i915/i915.scc
define KFEATURE_DESCRIPTION "Enable i915 driver"
define KFEATURE_COMPATIBILITY board

kconf hardware i915.cfg
```

# linux-yocto Continuous Updates

| Branch | Commit message | Author | Age |
|---|---|---|---|
| eg20t | kgit: creating baseline state | Bruce Ashfield | 8 months |
| emgd | kgit: creating baseline state | Bruce Ashfield | 8 months |
| emgd-1.10 | kgit: creating baseline state | Bruce Ashfield | 8 months |
| emgd-1.14 | emgd/pvr: get it building with v3.4 kernel | Nitin A Kamble | 7 months |
| emgd-1.16 | emgd: enable building within the kernel sources | Nitin A Kamble | 3 weeks |
| ltsi | usb: renesas_usbhs: gadget: usbhsg_ep_disable() care pipe | Kuninori Morimoto | 4 weeks |
| master | kgit: creating baseline state | Bruce Ashfield | 8 months |
| meta | meta: Remove Cedartrail Machine | Kishore Bodke | 41 hours |
| standard/arm-versatile-926ejs | Merge branch 'standard/base' into standard/arm-versatile-92 | Bruce Ashfield | 11 days |
| standard/base | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/beagleboard | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc-64/base | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc-64/chiefriver | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc-64/crystalforest | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc-64/jasperforest | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc-64/rangeley | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc-64/romley | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc-64/sugarbay | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc/atom-pc | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/common-pc/base | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/crownbay | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/emenlow | Merge branch 'standard/base' into standard/emenlow | Bruce Ashfield | 11 days |
| standard/fishriver | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/fri2 | Merge tag 'v3.4.28' into standard/base | Bruce Ashfield | 11 days |
| standard/fsl-mpc8315e-rdb | Merge branch 'standard/base' into standard/fsl-mpc8315e-rc | Bruce Ashfield | 11 days |
| standard/mti-malta32 | Merge branch 'standard/base' into standard/mti-malta32 | Bruce Ashfield | 11 days |

# Recipe-space vs Repo-Space

- '`yocto-bsp`' creates a new BSP and kernel `bbappend`:

```
[trz@empanada build]$ yocto-bsp create myqemuarm qemu

Which qemu architecture would you like to use? [default: i386]
          1) i386    (32-bit)
          2) x86_64  (64-bit)
          3) ARM     (32-bit)
          4) PowerPC (32-bit)
          5) MIPS    (32-bit)
Would you like to use the default (3.4) kernel? (y/n) [default: y]
Do you need a new machine branch for this BSP (the alternative is to
re-use an existing branch)? [y/n] [default: y]
Getting branches from remote repo git://git.yoctoproject.org/linux-yocto-3.4.git...
Please choose a machine branch to base your new BSP branch on: [default: standard/base]
          1) standard/arm-versatile-926ejs
          2) standard/base
          3) standard/beagleboard
          12) standard/routerstationpro
Would you like SMP support? (y/n) [default: y]
Does your BSP have a touchscreen? (y/n) [default: n]
Does your BSP have a keyboard? (y/n) [default: y]

New qemu BSP created in meta-myqemuarm
```

# Recipe-space vs Repo-Space (cont'd)

- The `yocto-bsp`-generated kernel is in recipe-space:

```
meta-myqemuarm/recipes-kernel/
└── linux
    ├── files
    │   ├── myqemuarm.cfg
    │   ├── myqemuarm-preempt-rt.scc
    │   ├── myqemuarm.scc
    │   ├── myqemuarm-standard.scc
    └── linux-yocto_3.4.bbappend
```

- The relevant parts of the kernel recipe look like this:

```
KBRANCH_DEFAULT_myqemuarm  = "standard/arm-versatile-926ejs/myqemuarm"
KBRANCH_myqemuarm  = "${KBRANCH_DEFAULT}"

KMACHINE_myqemuarm  = "myqemuarm"

KERNEL_FEATURES_append_myqemuarm += " cfg/smp.scc"

SRC_URI += "file://myqemuarm-standard.scc \
            file://myqemuarm.scc \
            file://myqemuarm.cfg \
            file://myqemuarm-user-config.cfg \
            file://myqemuarm-user-patches.scc"
```

# Recipe-space vs Repo-Space (cont'd)

- Here's the generated `myqemuarm-standard.scc`:

```
define KMACHINE myqemuarm
define KTYPE standard
define KARCH arm

include bsp/arm-versatile-926ejs/arm-versatile-926ejs-standard
branch myqemuarm

include myqemuarm.scc
```

- It's the top-level kernel feature defining this BSP
- It exists only in recipe-space
- The `KBRANCH` doesn't exist in the repo either
- Yet the BSP works with everything in recipe-space

# Recipe-space vs Repo-Space (cont'd)

- Here's the sugarbay BSP in the `linux-yocto_3.4` repo:

```
sugarbay/
├── sugarbay.cfg
├── sugarbay-preempt-rt.scc
├── sugarbay.scc
└── sugarbay-standard.scc
```

- And here's `sugarbay-standard.scc` in the repo:

```
define KMACHINE sugarbay
define KTYPE standard
define KARCH x86_64

include bsp/common-pc-64/common-pc-64-standard.scc
branch sugarbay

include sugarbay.scc
```

- All features, patches, and config can be moved to the repo
  - Central sharing, reuse, and git-based management

# Using Local Clones

- For `linux-yocto-*` kernels you can use a local repo
- This makes for an easier development workflow
- Simply replace the `SRC_URI` in the recipe
- For example in `linux-yocto-custom`, replace:

```
SRC_URI = "git://git.kernel.org/pub/scm/linux/kernel/git/stable/ \
           linux-stable.git;protocol=git;bareclone=1"
```

- with:

```
SRC_URI = "git:///home/myacct/poky-danny-8.0.1/linux-stable-work.git; \
protocol=file;bareclone=1"
```

# Using Local Clones (cont'd)

- The workflow looks like this with a bare local clone:
  - Point recipe to the bare clone, work in the working clone
  - Here's how you create the clone(s):

```
$ git clone --bare git://git.yoctoproject.org/linux-yocto-3.4 linux-yocto-3.4
$ git clone linux-yocto-3.4 linux-yocto-3.4-work
```

  - Basic workflow: edit, commit, push to bare clone, build:

```
$ cd linux-yocto-3.4-work
$ emacs -nw fs/filesystems.c
$ git commit -a -s -m "new commit"
$ git push origin standard/common-pc/base:standard/common-pc/base
$ bitbake -c deploy virtual/kernel
```

- You don't strictly need a bare clone, here's non-bare:

```
$ git clone git://git.yoctoproject.org/linux-yocto-3.4 linux-yocto-3.4
$ emacs -nw fs/filesystems.c
$ git commit -a -s -m "new commit"
$ bitbake -c cleanall virtual/kernel
$ bitbake -c deploy virtual/kernel
```


# Questions?

# Thank you for your participation!

yocto PROJECT | THE LINUX FOUNDATION