
# Yocto Project® devtool Overview and Hands-On

## Demo Replay

Michael Opdenacker, Root Commit
(with material by David Reyna, Trevor Woerner, Saul Wold, and Paul Eggleton)

Yocto Project Summit 2024.12

root commit

# Michael Opdenacker

- Contributor to OpenEmbedded and the Yocto Project
  - Former documentation maintainer
  - Misc contributions to OE-core, ALSA recipes, and updates to PR server.
- Also offering a new “Yocto Project and OpenEmbedded” training course (4 days, on-site or on-line)
  - At least 75% of practical lab time!
  - https://rootcommit.com/training/yocto/

# devtool – what we are going to do today

- Introduce `devtool`
- Import an external package
  - Create a local recipe
  - Develop, Deploy, and Test
  - Export to a layer
- Upgrade a package and modify it
- Clone an existing package and modify it

# Devtool - Introduction

- Collection of tools for working on *recipes*:
  - **`devtool add`**
  - **`devtool edit-recipe`**
  - **`devtool upgrade`**
  - **`devtool finish`**
  - *etc...*

# devtool

- *...and more!*
  - **`devtool modify`**
  - **`devtool deploy-target`**
  - **`devtool undeploy-target`**
  - **`devtool build`**
  - **`devtool build-image`**
  - *etc...*

# devtool – why it exists

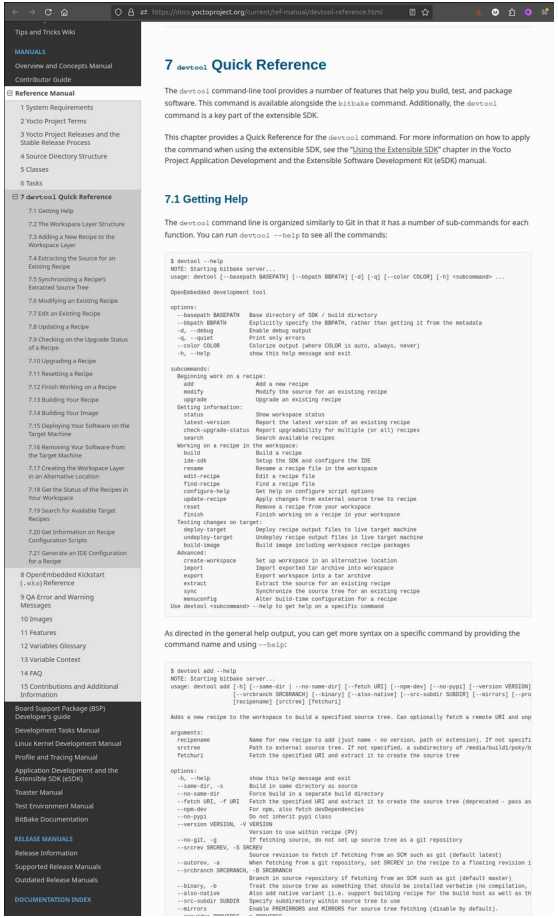
- Our build system is great for repeatable builds from source
- Working with the source itself was hard
  - Tempting to just edit sources under `tmp/work/`…
  - But the workflow is painful after that (forced builds, manual patch generation, lost work…)
- Help newer users add new software
  (within regular builds and within eSDKs)
- Devtool helps with the bitbake
  and target “paperwork”

# devtool – past presentations

- YPS 2023 – An earlier version of this presentation – David Reyna
  Video: https://www.youtube.com/watch?v=TP_9ZbgNcaw
  Slides: https://s.42l.fr/vyr-eJLs
- YPDD 2018 – “Session 3, Devtool 1” - Tim Orling
  https://www.youtube.com/watch?v=C-usM6gFVSY
- YPDD 2018 – “Session 7, Devtool 2” - Tim Orling & Henry Bruce
  https://www.youtube.com/watch?v=UYsqIP_Qt_Q
- ELC 2017: “Using Devtool To Streamline
  Your Yocto Project Workflow” - Tim Orling
  https://www.youtube.com/watch?v=CiD7rB35CRE
- ELC 2017: Yocto Project Extensible SDK: Simplifying
  the Workflow for Application Developers - Henry Bruce
  https://www.youtube.com/watch?v=d3xanDJuXRA

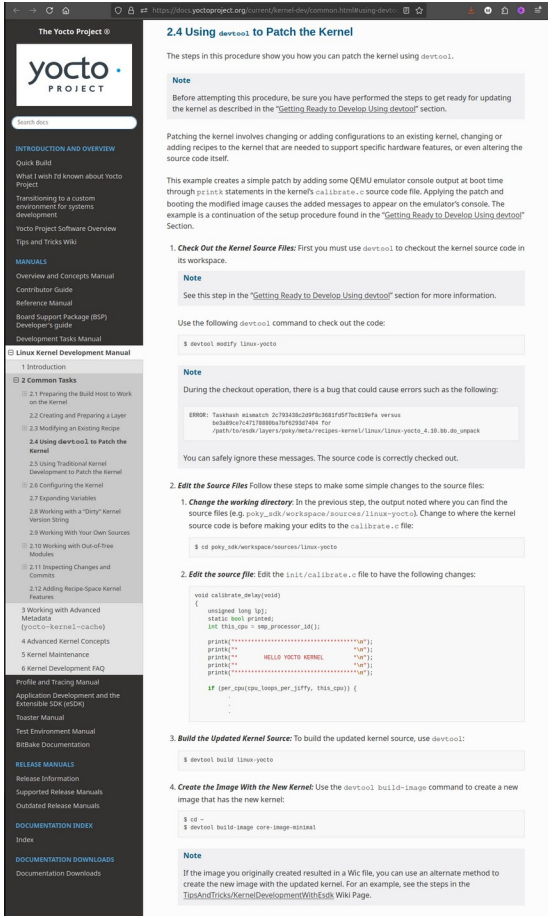
# devtool – official documentation

- Yocto Project Reference Manual
  - Chapter 7 - *devtool* Quick Reference
  - https://docs.yoctoproject.org/current/ref-manual/devtool-reference.html
- Yocto Project Application Development and the Extensible Software Development Kit (eSDK)
  - Chapter 2 - Using the Extensible SDK
  - https://docs.yoctoproject.org/current/sdk-manual/



# devtool – official documentation

- Yocto Project Linux Kernel Development Manual
  - Section 2.4 - Using *devtool* to Patch the Kernel
  - https://docs.yoctoproject.org/current/kernel-dev/common.html#using-devtool-to-patch-the-kernel



## 2.4 Using devtool to Patch the Kernel

The steps in this procedure show you how you can patch the kernel using `devtool`.

**Note**

Before attempting this procedure, be sure you have performed the steps to get ready for updating the kernel as described in the "Getting Ready to Develop Using devtool" section.

Patching the kernel involves changing or adding configurations to an existing kernel, changing or adding recipes to the kernel that are needed to support specific hardware features, or even altering the source code itself.

This example creates a simple patch by adding some QEMU emulator console output at boot time through `print` statements in the kernel's `calibrate.c` source code file. Applying the patch and booting the modified image causes the added messages to appear on the emulator's console. The example is a continuation of the setup procedure found in the "Getting Ready to Develop Using devtool" Section.

1. ***Check Out the Kernel Source Files:*** First you must use `devtool` to checkout the kernel source code in its workspace.

   **Note**

   See this step in the "Getting Ready to Develop Using devtool" section for more information.

   Use the following `devtool` command to check out the code:

   ```
   $ devtool modify linux-yocto
   ```

   **Note**

   During the checkout operation, there is a bug that could cause errors such as the following:

   ```
   ERROR: Taskhash mismatch 2c793438c2d9f8c3681fd5f7bc819efa versus
          be3a89ce7c47178880ba7bf6293d7404 for
          /path/to/esdk/layers/poky/meta/recipes-kernel/linux/linux-yocto_4.10.bb.do_unpack
   ```

   You can safely ignore these messages. The source code is correctly checked out.

2. ***Edit the Source Files*** Follow these steps to make some simple changes to the source files:

   1. ***Change the working directory***: In the previous step, the output noted where you can find the source files (e.g. `poky_sdk/workspace/sources/linux-yocto`). Change to where the kernel source code is before making your edits to the `calibrate.c` file:

      ```
      $ cd poky_sdk/workspace/sources/linux-yocto
      ```

   2. ***Edit the source file***: Edit the `init/calibrate.c` file to have the following changes:

      ```
      void calibrate_delay(void)
      {
          unsigned long lpj;
          static bool printed;
          int this_cpu = smp_processor_id();

          printk("*********************************\n");
          printk("*                               *\n");
          printk("*        HELLO YOCTO KERNEL     *\n");
          printk("*                               *\n");
          printk("*********************************\n");

          if (per_cpu(cpu_loops_per_jiffy, this_cpu)) {
                  .
                  .
                  .
      ```

3. ***Build the Updated Kernel Source:*** To build the updated kernel source, use `devtool`:

   ```
   $ devtool build linux-yocto
   ```

4. ***Create the Image With the New Kernel:*** Use the `devtool build-image` command to create a new image that has the new kernel:

   ```
   $ cd ~
   $ devtool build-image core-image-minimal
   ```

   **Note**

   If the image you originally created resulted in a Wic file, you can use an alternate method to create the new image with the updated kernel. For an example, see the steps in the TipsAndTricks/KernelDevelopmentWithEsdk Wiki Page.

# devtool – command information

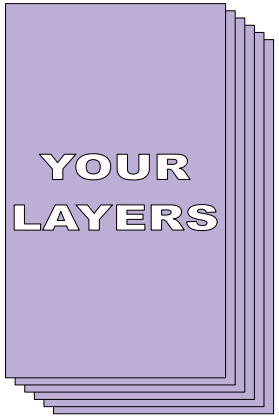
```
$ devtool --help
usage: devtool [--basepath BASEPATH] [--bbpath BBPATH] [-d] [-q]
               [--color COLOR] [-h]
               <subcommand> ...

OpenEmbedded development tool

options:
  --basepath BASEPATH  Base directory of SDK / build directory
  --bbpath BBPATH      Explicitly specify the BBPATH, rather than getting it
                       from the metadata
  -d, --debug          Enable debug output
  -q, --quiet          Print only errors
  --color COLOR        Colorize output (where COLOR is auto, always, never)
  -h, --help           show this help message and exit

subcommands:
  Beginning work on a recipe:
    add                  Add a new recipe

...
```

# devtool – subcommand help

```
$ devtool add --help
usage: devtool add [-h] [--same-dir | --no-same-dir] [--fetch URI]
                   [--fetch-dev] [--version VERSION] [--no-git]
                   [--srcrev SRCREV | --autorev] [--srcbranch SRCBRANCH]
                   [--binary] [--also-native] [--src-subdir SUBDIR]
                   [--mirrors] [--provides PROVIDES]
                   [recipename] [srctree] [fetchuri]

Adds a new recipe to the workspace to build a specified source tree. Can
optionally fetch a remote URI and unpack it to create the source tree.

arguments:
  recipename            Name for new recipe to add (just name - no version,
                        path or extension). If not specified, will attempt to
                        auto-detect it.
  srctree               Path to external source tree. If not specified, a
                        subdirectory of /z/ypdd/2018-10-devtool/my-
                        class/poky/build/workspace/sources will be used.
  fetchuri              Fetch the specified URI and extract it to create the
                        source tree

options:
  -h, --help            show this help message and exit
...
```

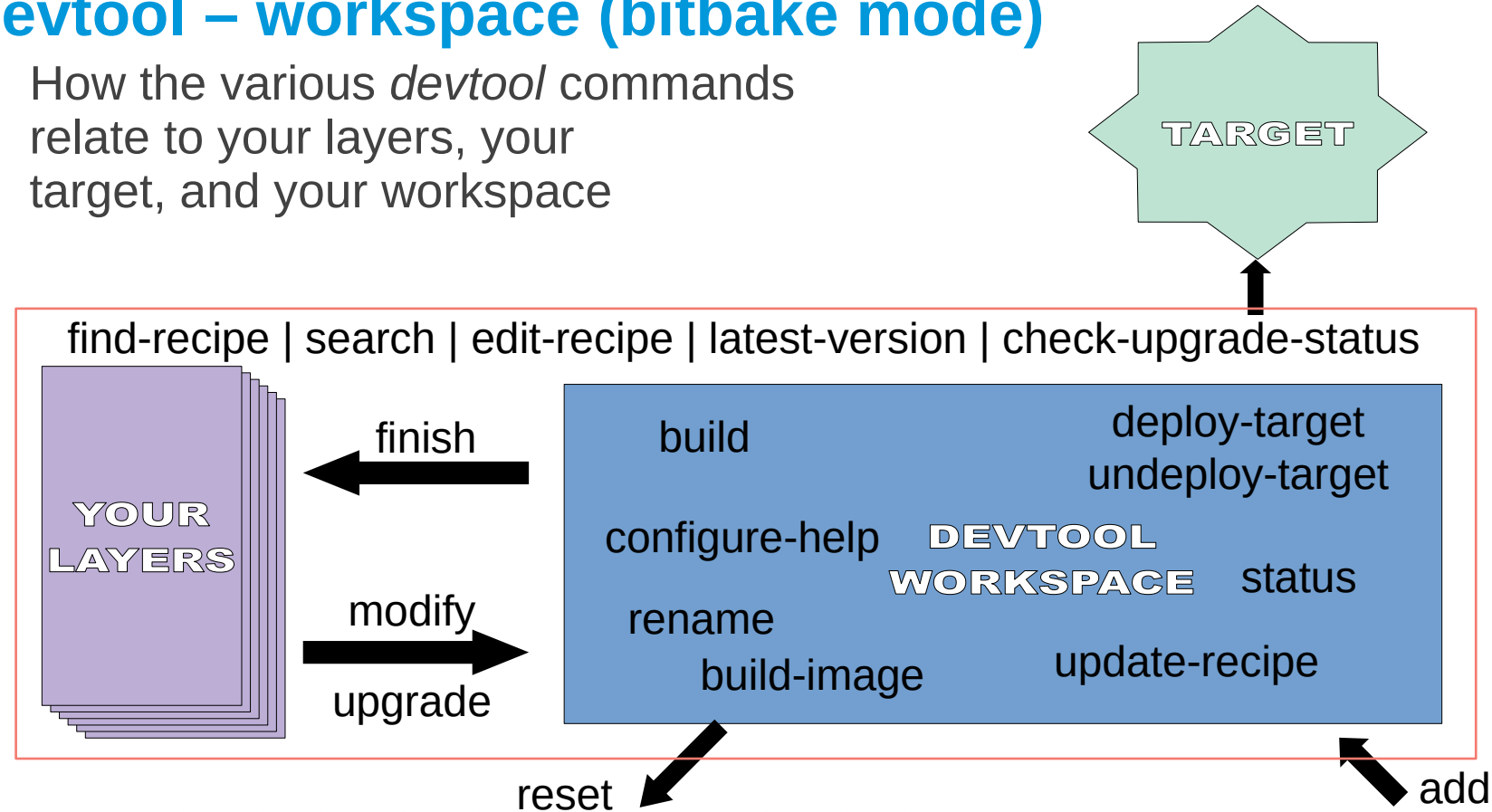
# devtool – workspace

- A separate environment (layer) in which to work on recipes, sources, patches

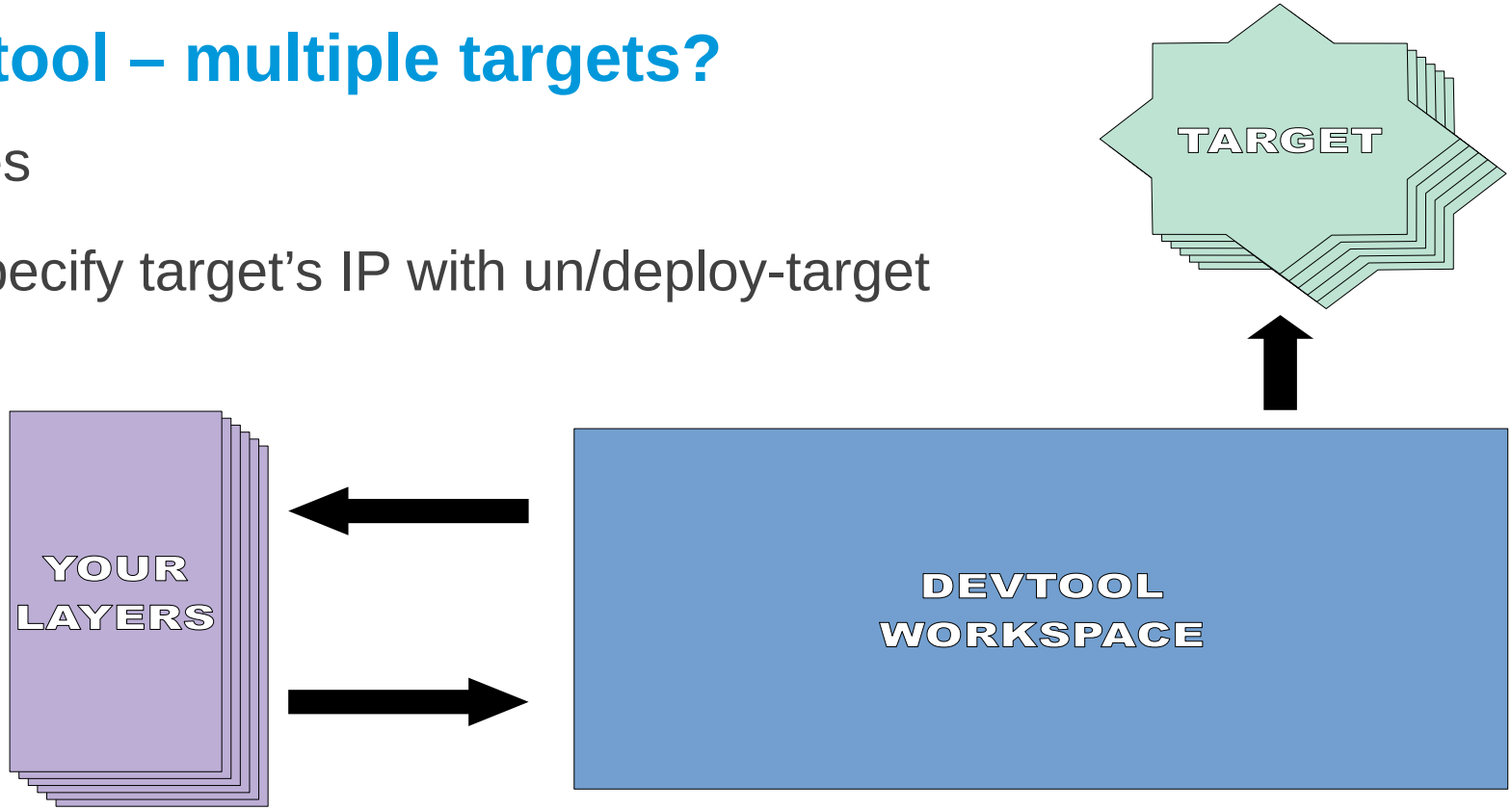

# devtool – workspace (bitbake mode)

- How the various *devtool* commands relate to your layers, your target, and your workspace



# devtool – multiple targets?

- Yes
- Specify target's IP with un/deploy-target



# Sidebar: recipetool

- Extra set of tools for working on recipes
- Contains logic for creating recipes
  - Used by `devtool add`
- Can also create / update “bbappends”, programmatically set variables in recipes, etc.

# Hands On

# devtool – setup

```
1$ # Create easy SSH connection for devtool deployments
1$ nano ~/.ssh/config
```

```
Host qemu
          User root
          Hostname localhost
          Port 2222
          StrictHostKeyChecking no
          UserKnownHostsFile /dev/null
```

```
# Create your Yocto Project Installation
1$ git clone -b styhead git://git.yoctoproject.org/poky
1$ cd poky
1$ source ./oe-init-build-env build-devtool
1$ edit conf/local.conf
          MACHINE = "qemuarm64"
          IMAGE_INSTALL:append = " openssh"
          EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
1$ bitbake core-image-base
```

# devtool – setup default layer, default git support

```
1$ bitbake-layers create-layer ../meta-foo
1$ bitbake-layers add-layer ../meta-foo
1$ git config --global user.name "name"
1$ git config --global user.email "name@example.com"
```

- Open a second ssh connection to the build machine for QEMU

```
2$ cd /home/ilab01/yp-summit-dec-24/poky
2$ source oe-init-build-env build-devtool
2$ runqemu slirp nographic serial
```

- Do the exercises in the first connection, work on the target in the second connection

- Login as `root`, no password (thanks to "***debug-tweaks***", feature removed in 5.2 – Walnascar: see https://git.yoctoproject.org/poky/commit/?id=43b8b3fa72d75d8d82a478613a4d9bf4645b5389)
  (Note: CTRL-A,X to quit QEMU when you are done)

# devtool – import the external package “nano”

```
1$ devtool add \
    https://nano-editor.org/dist/v5/nano-5.8.tar.xz
```

- Implicitly creates workspace (if it doesn’t already exist)
- Guesses the recipe name ***nano*** (correctly!)
- Looks at the source and determines it’s an ***autotooled*** project (true! and ***pkgconfig*** and ***gettext***)
- Guesses at DEPENDS (correctly! ***ncurses*** and ***zlib***)
- Creates a “rough” recipe

```
1$ devtool status
1$ devtool find-recipe nano
1$ devtool edit-recipe nano
```

# devtool – building “nano”

- Let’s see if it builds

```
1$ devtool build nano
```

- It builds!

# devtool – what goes in a workspace?

- The things on which you are working:
  - recipes
  - patches
  - sources
  - etc...

```
1$ tree -d workspace
```

- ...except sources can be, optionally, outside the workspace

# devtool – deploy “nano”

- Examine current build manifest, observe no nano package

```
1$ grep nano tmp/deploy/images/qemuarm64/core-image-base-qemuarm64.rootfs.manifest
```

- In the terminal running QEMU, log in and verify there’s no nano

```
2root@qemuarm64# nano
-sh: nano: command not found
```

- Send nano to target *(using SSH’s* qemu *configuration from above)*

```
1$ devtool deploy-target nano qemu   [‘-s’ if connection error]
```

- Observe that nano now runs on the target

# Sidebar – SLIRP versus TUN/TAP

- Yocto Project supports several connection technologies for QEMU

- SLIRP: advantage is no root access required, disadvantages are minimal documentation, requires SSH knowledge, ICMP (e.g. ping) not available by default

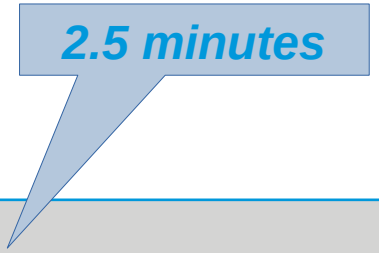
`qemu` is defined in `~/.ssh/config` (see earlier slide)

```
2$ runqemu slirp nographic serial
1$ devtool deploy-target nano qemu
```

- TAP: advantage is simpler setup, disadvantage is that it requires `sudo` access

```
2$ sudo runqemu nographic serial
1$ devtool deploy-target nano root@192.168.7.2
```

# devtool – let's see "nano" run




- Build an entire image

```
1$ devtool build-image core-image-base
...
NOTE: Building image core-image-base with the following additional
packages: nano
...
```

- Examine **tmp/deploy/images/qemuarm64/core-image-base-qemuarm64.rootfs.manifest**
  - Now there is a **nano** package!
- Why not just use **bitbake core-image-base**?
  - **nano** package not automatically added

# devtool – upgrade “nano”

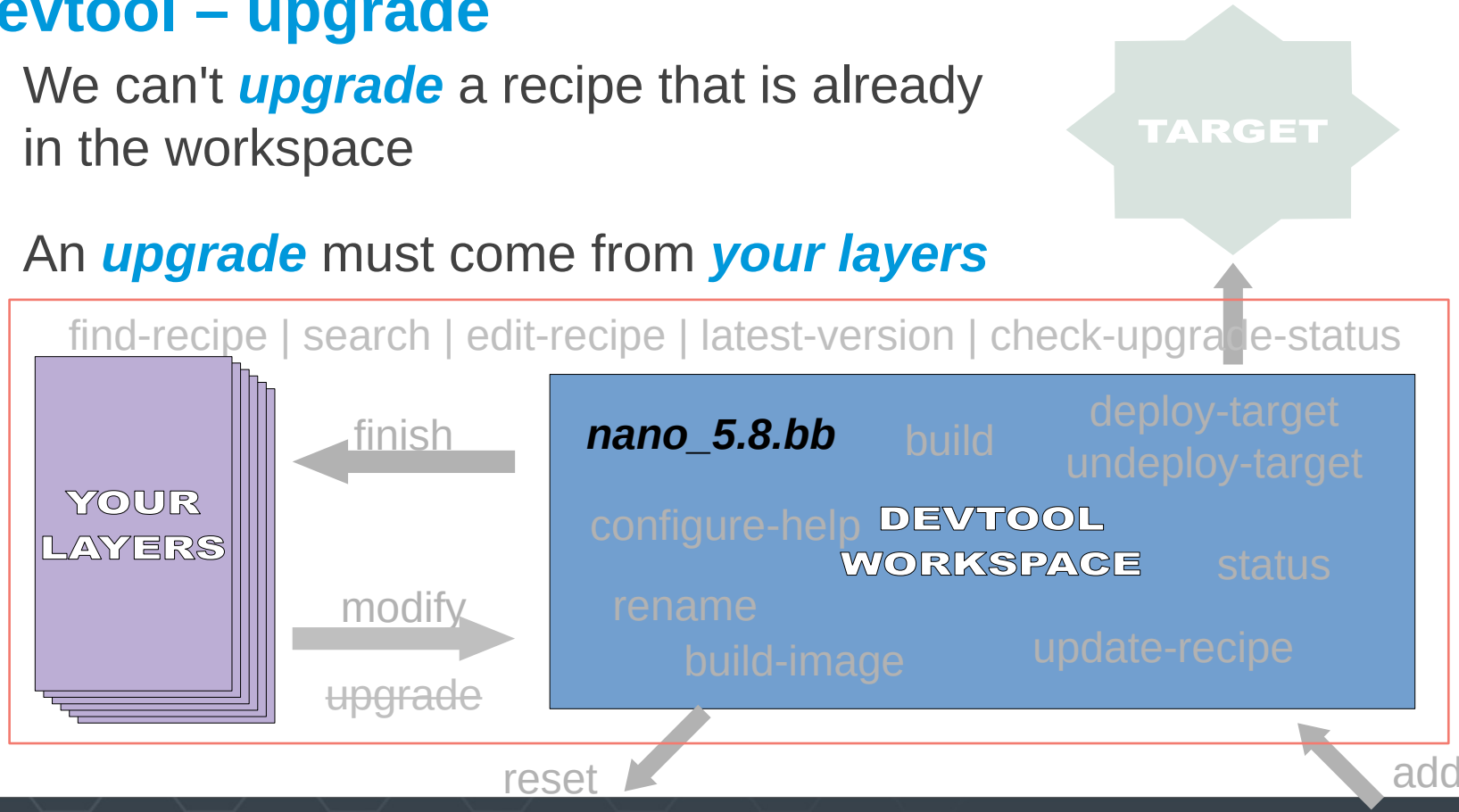
- Try upgrading nano

```
1$ devtool upgrade nano
ERROR: recipe nano is already in your workspace
```

- We need to move the **`nano`** recipe to ***your layers*** before we can **upgrade**
  - Preferably our own (`meta-foo`)
- This is only an issue because **`nano`** is already in the workspace – normally **`devtool upgrade`** is where you start an upgrade for an existing recipe

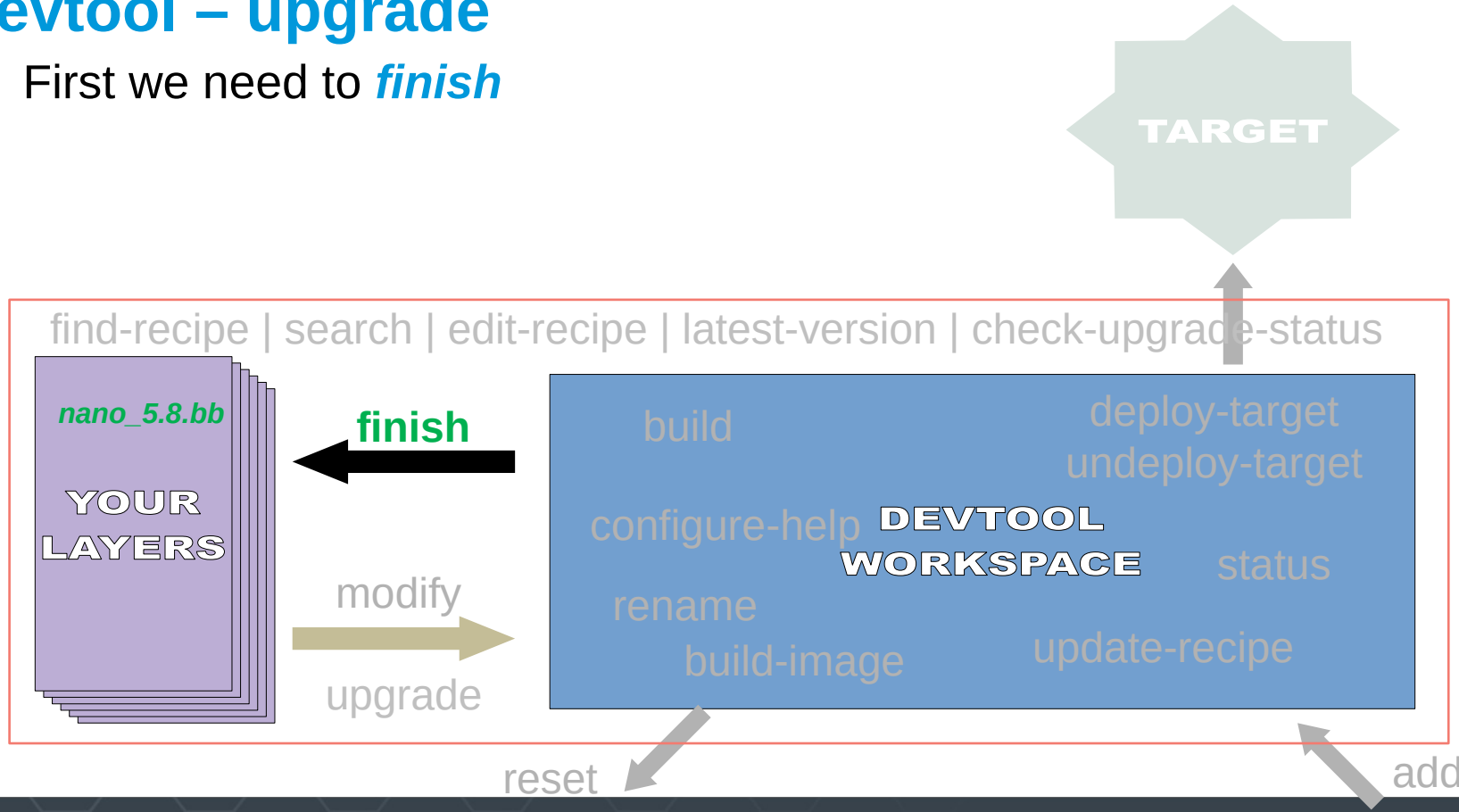
# devtool – upgrade

- We can't ***upgrade*** a recipe that is already in the workspace
- An ***upgrade*** must come from ***your layers***



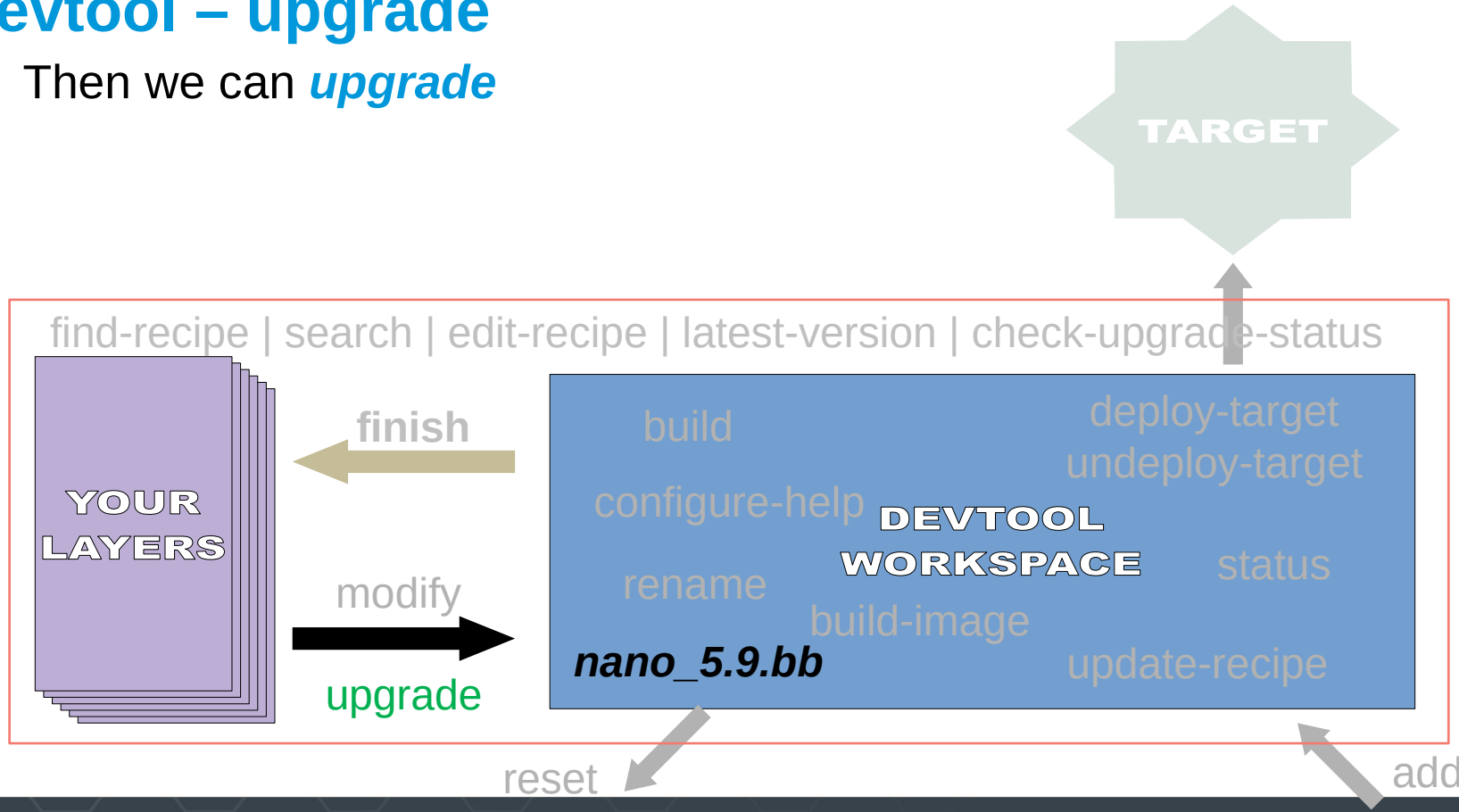
# devtool – upgrade

- First we need to ***finish***



# devtool – upgrade

- Then we can ***upgrade***



# devtool – upgrade

```
1$ devtool finish nano ../meta-foo
ERROR: Source tree is not clean:
...
```

- This error is ***not*** a problem we introduced; it is because of files generated or modified by `devtool build nano`.
- Adding **`-f`** to `devtool finish` is supposed to address this, it doesn't seem to work here (see https://bugzilla.yoctoproject.org/show_bug.cgi?id=15546)
- Here's a workaround, to ignore such file changes in `nano` sources:

```
1$ pushd workspace/sources/nano/
1$ git restore .
1$ popd
```

# devtool – upgrade

- We can now run `devtool finish`:

```
1$ devtool finish -f nano ../meta-foo
INFO: No patches or files need updating
INFO: Moving recipe file to /home/ilab01/yp-summit-dec-24/poky/meta-foo/recipes-
nano/nano
INFO: Preserving source tree in
/home/ilab01/yp-summit-dec-24/poky/build-devtool/workspace/attic/sources/
nano.20241128151430
If you no longer need it then please delete it manually.
It is also possible to reuse it via devtool source tree argument.
```

# devtool – upgrade from upstream

*Yes, lots of output*

```
1$ devtool upgrade nano
...
ERROR: Bitbake Fetcher Error: FetchError('Unable to fetch URL from any source.',
'https://nano-editor.org/dist/v5/nano-8.2.tar.xz')
```

- In some cases like this, devtool can’t figure out how to find and upgrade tarballs (this information is not obvious from the URL). It guessed “8.2” in this case, but that does not exist upstream.

# devtool – upgrade “nano”

- We need to give `devtool` more help

```
$ devtool upgrade -V 5.9 nano
```

- It works!

```
$ devtool build nano
```

- It builds!

# devtool deploy-target - dive in

- Is it okay to re-deploy a second time without cleaning up the first deploy?
  - yes... usually
- On the target

```
2root@qemuarm64# cd /
2root@qemuarm64# ls -a
...
.devtool
...
2root@qemuarm64# cd .devtool
2root@qemuarm64# ls -l
-rw-r--r--    1 root     root          5288 Nov 28 16:42 nano.list
drwxr-xr-x    3 root     root          4096 Nov 28 16:42 nano.preserve
```

# devtool deploy-target - dive in

- **nano.list** is created by devtool, per package, when it deploys to the target
- Examine **poky/scripts/lib/devtool/deploy.py** for all the answers
  - It creates a script that is copied to target
  - Preserves any files that would be clobbered
  - Generates a list of files being deployed, so they can be undeployed
  - Deploying starts by first undeploying (same recipe name)

# devtool deploy-target - dive in

- Undeploy, and check that `nano` is removed from the target, and the plumbing is also removed

```
1$ devtool undeploy-target nano qemu
```

```
2root@qemuarm64# ls -a /.devtool
```

- Remember to finish and cleanup
  (after applying the `git restore` workaround)

```
1$ devtool finish -f nano ../meta-foo
```

# devtool - floating devtool commands

- Some devtool commands don’t care whether the recipe is in the workspace or in the layers

```
1$ devtool status
NOTE: No recipes currently in your workspace - you can use "devtool
modify" to work on an existing recipe or "devtool add" to add a new one

1$ devtool edit-recipe ethtool
(works)

1$ devtool latest-version ethtool
NOTE: Current version: 6.10
NOTE: Latest version: 6.11

1$ devtool find-recipe ethtool
home/ilab01/yp-summit-dec-2024/poky/meta/recipes-extended/ethtool/ethtool_6.10.bb

1$ devtool search ethtool
ethtool             Display or change ethernet card settings
```

# devtool - creating a patch

- Use-case?
  Patches can be needed to
  - Add/remove functionality
    - Reduce size on target
    - Remove dependency/dependencies
  - Allow code to be (cross-)compiled

# devtool - creating a patch

```
1$ devtool add
https://rootcommit.com/pub/conferences/2024/yps/autotool-devtool-example/v1.0.0.tar.gz
ERROR: Could not auto-determine recipe name, please specify it on the command line
```

```
1$ devtool add autotool-devtool-example
https://rootcommit.com/pub/conferences/2024/yps/autotool-devtool-example/v1.0.0.tar.gz
1$ devtool build autotool-devtool-example
1$ devtool deploy-target autotool-devtool-example qemu
```

```
2root@qemuarm64# autotool-devtool-example
Hello, world!
version: 1.0.0
Hello from the library
```

# devtool - creating a patch

- Edit the code

```
1$ pushd workspace/sources/autotool-devtool-example
1$ nano src/autotool-devtool-example.c
```

- change from

```
        printf("Hello, world!\n");
```

- to

```
        printf("Hello, devtool!\n");
```

# devtool - creating a patch

- Build, deploy, verify

```
1$ popd
1$ devtool build autotool-devtool-example
1$ devtool deploy-target autotool-devtool-example qemu
```

```
2root@qemuarm64# autotool-devtool-example
Hello, devtool!
version: 1.0.0
Hello from the library
```

# devtool - creating a patch

*Try “git status”*

- Cleanup (but what about my edit?)

```
$ devtool finish autotool-devtool-example ../meta-foo
ERROR: Source tree is not clean:
 M src/autotool-devtool-example.c
```

- Oops! But it’s ok, it didn’t clobber or lose your work

```
$ pushd workspace/sources/autotool-devtool-example
$ git commit -avs -m “update salutation”
...
$ popd
$ devtool finish autotool-devtool-example ../meta-foo -f
...
INFO: Adding new patch 0001-update-salutation.patch
...
```

# devtool - creating conflict

- Now we'll update to a newer release, but the newer release will conflict with our patch

```
1$ devtool upgrade autotool-devtool-example
Resolving rootcommit.com (rootcommit.com)... 148.135.128.14,
2a02:4780:51:f0be:ff62:d9f2:58f7:dc85
Connecting to rootcommit.com (rootcommit.com)|148.135.128.14|:443...
connected.
HTTP request sent, awaiting response... 403 Forbidden
2024-11-28 18:01:26 ERROR 403: Forbidden.

ERROR: Automatic discovery of latest version/revision failed - you must
provide a version using the --version/-V option, or for recipes that fetch
from an SCM such as git, the --srcrev/-S option.
```

- Devtool can't figure it out, we need to help it

# devtool – trying to upgrade

```
1$ devtool upgrade -V 1.0.1 autotool-devtool-example
...
ERROR: QA Issue: Missing Upstream-Status in patch
/home/ilab01/yp-summit-dec-24/poky/meta-foo/recipes-autotool-devtool-
example/autotool-devtool-example/autotool-devtool-example/0001-update-
salutation.patch
Please add according to
https://docs.yoctoproject.org/contributor-guide/recipe-style-
guide.html#patch-upstream-status . [patch-status]
ERROR: Fatal QA errors were found, failing task.
...
```

- Our patch is missing mandatory `Upstream-Status` information. Let's fix this first.

# Add missing tag to patch

- So, add an `Upstream-Status` tag to the patch file (`../meta-foo/recipes-autotool-devtool-example/autotool-devtool-example/autotool-devtool-example/0001-update-salutation.patch`)

```
Signed-off-by: Jules Verne <jules@verne.net>
Upstream-Status: Inappropriate
```

# devtool - creating conflict

```
1$ devtool upgrade -V 1.0.1 autotool-devtool-example
INFO: Rebasing devtool onto 96788a9efa60046d8ebf166d1cbc8d60c028aa89
WARNING: Command 'git rebase 96788a9efa60046d8ebf166d1cbc8d60c028aa89' failed:
Auto-merging src/autotool-devtool-example.c
CONFLICT (content): Merge conflict in src/autotool-devtool-example.c

You will need to resolve conflicts in order to complete the upgrade.
INFO: Upgraded source extracted to /home/ilab01/yp-summit-dec-24/poky/build-
devtool/workspace/sources/autotool-devtool-example
INFO: New recipe is
/home/ilab01/yp-summit-dec-24/poky/build-devtool/workspace/recipes/autotool-
devtool-example/autotool-devtool-example_1.0.1.bb
```

# devtool - resolving conflict

- `devtool` aborted the `git rebase` operation.

```
1$ pushd workspace/sources/autotool-devtool-example
1$ git status
On branch devtool
nothing to commit, working tree clean
1$ git branch -a
* devtool
  devtool-1.0.1
  devtool-orig
  master
```

# devtool - resolving conflict

- Let's run the `rebase` operation again

```
1$ git rebase devtool-1.0.1
Auto-merging src/autotool-devtool-example.c
CONFLICT (content): Merge conflict in src/autotool-devtool-example.c
error: could not apply 838d113... update salutation
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git
rebase --abort".
Could not apply 838d113... update salutation
```

- So let's follow the instructions and resolve the conflict

```
1$ nano src/autotool-devtool-example.c
```

# devtool - resolving conflict

- From

```
...
<<<<<<< HEAD
        /* a meaningful comment */
        printf("Hello, world!\n");
=======
        printf("Hello, devtool!\n");
>>>>>>> 838d113 (update salutation)
...
```

- To

```
...
        /* a meaningful comment */
        printf("Hello, devtool!\n");
...
```

# devtool - resolving conflict

```
1$ git add src/autotool-devtool-example.c
1$ git rebase --continue
Applying: update salutation
1$ popd
```

- This time, let's inspect recipe updates first with **-N** (dry run)

```
1$ devtool finish autotool-devtool-example ../meta-foo -N
```

- If we're happy with the proposed changes, apply them:

```
1$ devtool finish autotool-devtool-example ../meta-foo
```

# devtool - resolving conflict

```
1$ tree ../meta-foo
../meta-foo/
...
├── recipes-nano
│   └── nano
│       └── nano_5.9.bb
└── recipes-autotool-devtool-example
    └── autotool-devtool-example
        ├── autotool-devtool-example
        │   └── 0001-update-salutation.patch
        └── autotool-devtool-example_1.0.1.bb
```

# devtool – modify an existing recipe

1) Takes an existing recipe from layers
2) Unpacks sources into workspace
3) Edit recipe or sources
4) … (same as **`devtool add`** / **`devtool upgrade`** workflow)

# devtool modify example

```
1$ devtool modify bc
INFO: Source tree extracted to /home/ilab01/yp-summit-dec-24/poky/build-
devtool/workspace/sources/bc
INFO: Recipe bc now set up to build from
/home/ilab01/yp-summit-dec-24/poky/build-devtool/workspace/sources/bc
1$ devtool edit-recipe bc
```

- This gives you a chance to view and edit the recipe. Let's edit the sources too

```
1$ pushd workspace/sources/bc
1$ ls
aclocal.m4  ar-lib  AUTHORS  bc  ChangeLog  compile  config.h.in  configure
configure.ac  COPYING  COPYING.LIB  dc  depcomp  doc  Examples  FAQ  h
INSTALL  install-sh  lib  Makefile.am  Makefile.in  missing  NEWS  README
Test  ylwrap
```

# devtool modify example

- Edit **bc/main.c** and make a trivial change to the help text printed in **usage()** (line 69)

```
1$ nano bc/main.c
```

- Commit changes and run **devtool finish**

```
1$ git add bc/main.c
1$ git commit -s -m “change help text”
1$ popd
1$ devtool finish bc ../meta-foo
NOTE: Writing append file /home/ilab01/yp-summit-dec-24/poky/meta-foo/recipes-extended/bc/bc_%.bbappend
NOTE: Copying 0001-change-help-text.patch to /home/ilab01/yp-summit-dec-24/poky/meta-foo/recipes-extended/bc/bc/0001-change-help-text.patch
INFO: Cleaning sysroot for recipe bc...
...
```

# devtool modify example

- **`devtool finish`** realized the `bc` recipe is not in `meta-foo`
  - Thus it created a `bbappend` and placed the patch next to it
  - Naturally if we had passed the path to **`poky/meta`** it would have modified the original recipe

# Wrap up

# devtool - eSDK Mode

- The *e*SDK includes many improvements over the standard SDK
- Everything the standard SDK can provide, plus all of the functionality we've been looking at which is provided by `devtool`

# devtool – mode commands

- bitbake mode
  - add
  - build
  - build-image
  - configure-help
  - check-upgrade-status
  - ***create-workspace***
  - deploy-target
  - edit-recipe
  - export
  - extract
  - find-recipe
  - finish
  - import
  - latest-version
  - menuconfig
  - modify
  - rename
  - reset
  - search
  - status
  - sync
  - undeploy-target
  - update-recipe
  - upgrade
- eSDK mode
  - add
  - build
  - build-image
  - ***build-sdk***
  - configure-help
  - check-upgrade-status
  - deploy-target
  - edit-recipe
  - export
  - extract
  - find-recipe
  - finish
  - import
  - latest-version
  - menuconfig
  - modify
  - ***package***
  - rename
  - reset
  - ***runqemu***
  - ***sdk-install***
  - ***sdk-update***
  - search
  - status
  - sync
  - undeploy-target
  - update-recipe
  - upgrade

# devtool – mode commands

- Why does eSDK mode get extra features?
  - Because an eSDK doesn't have *bitbake* or *scripts/*
  - *devtool* is the cornerstone of the eSDK

# Future

- `recipetool` enhancements
  (make `devtool add` smarter and support more sources)
- Your idea here :)
- Help is very much welcome!

# Summary

- Try out `devtool` on your own sources / recipes:
  - **`devtool add`** on a source tree / tarball / URL
  - **`devtool modify`** and work on an existing recipe
  - **`devtool upgrade`** existing recipe to a new upstream version
- See documentation links & other presentations (earlier slide)

# Conclusion

- Please send feedback!
  - Yocto Project mailing lists
    - https://www.yoctoproject.org/community/mailing-lists/
  - IRC (#yocto on irc.libera.chat)
  - Email: michael.opdenacker@rootcommit.com
  - License: Creative Commons CC-BY-SA 4.0

To learn more and support the creation of more presentations and videos, you may be interested in my Yocto Project and OpenEmbedded training course:

**https://rootcommit.com/training/yocto**

- Register individually (public session) or in a group
- 4 days (in-person), 24 hours (online – 6 x 4h)
- 75% of practical activities, on BeaglePlay or QEMU
- Really doing the labs in online sessions
- Short lectures only, avoiding exhaustive theory
- Challenging, varied and fun learning techniques
- Public sessions limited to 8 participants
- Lecture recordings for online sessions
- Use the distro of your choice

Image credits: https://openclipart.org/detail/189359/penguins-just-love-openclipart