Devtool Hands-On Class

Yocto Project Summit 2025

Michael Opdenacker

Root Commit

Dec. 2, 2025

Michael Opdenacker

root commit

Embedded Linux consultant and trainer

- https://rootcommit.com/about/michael-opdenacker/
- Former founder of Bootlin
- New founder of Root Commit
- Offering embedded Linux training courses with a focus on practical activities, interactivity and learning techniques. https://rootcommit.com/training/
- Contributor to Yocto / OE, and maintaining a few recipes (`devtool` helps a lot!)
- Free Software enthusiast and advocate (member of April.org)

April
promouvoir et défendre le logiciel libre

Hands-On — Setup

root commit

- Set up your BitBake environment
- Build your first image

devtool Class Preparation (1)

root commit

To save time, please log into your VM.
First, set up the BitBake environment:

```
$ source bitbake-builds/distro_poky-master/build-devtool/init-build-env
```

Then, let's change the configuration to allow for an empty root password (convenient for debugging):

```
$ bitbake-config-build enable-fragment core/yocto/root-login-with-empty-password
```

Also add these settings to `conf/local.conf` if not set yet:

```
MACHINE = "qemuarm64"
WARN_QA:append = " version-going-backwards"
ERROR_QA:append = " version-going-backwards"
IMAGE_INSTALL:append = " openssh"
IMAGE_FSTYPES:append = " ext4"
```

devtool Class Preparation (1)

root commit

We can now start building an image:

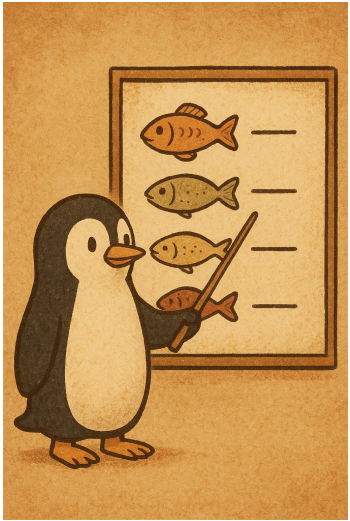
```
$ bitbake core-image-base
```

This should take a few minutes to build,
thanks to a cache for downloads and for built artifacts (*shared state cache*).

Plans For Today

root commit

- Introduction to `devtool`
- Import an external package
  - Create a new recipe from sources on the Internet
  - Fix, deploy and test this recipe
  - Export this recipe to a layer
- Modify a recipe to upgrade a package
- Make changes to an existing recipe



Introduction to `devtool`

devtool — Why it exists

root commit

- The build system is great at generating reproducible builds from sources and metadata
- However, it's not adapted to software development.
  - You could try to directly work with sources under `tmp/work`
  - But this leads to all sorts of complications and the build system may erase your changes doing that.
- devtool is the cornerstone of the BitBake Extensible Software Development Kit (eSDK)
  - It makes it easy to build experimental code,
    taking care of all the tedious recipe management tasks.
  - It can help new developers create new recipes.
  - Even developers without BitBake can build applications for a given product.

`devtool` — What you can do with it

Devtool can be used to

- Create a new recipe from the sources of a component
  - It guesses the recipe name
  - It automatically recognizes the build system for the program
  - It detects the license files
  - It figures out some dependencies
  - The generated recipe needs human review, but a big part of the job is done
- Check whether a recipe has an update upstream
- Propose patches to upgrade a recipe to a newer upstream
- Modify an existing recipe
- Create an IDE configuration for your recipes
- Compile applications that you are developing
- Even without having BitBake (while using an *Extended SDK*)
- Deploy your applications to your target system
- Build an image with recipes under development

devtool subcommands (1)

```
$ devtool --help
NOTE: Starting bitbake server...
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
    modify                Modify the source for an existing recipe
    upgrade               Upgrade an existing recipe
  Getting information:
    status                Show workspace status
    search                Search available recipes
    latest-version        Report the latest version of an existing recipe
    check-upgrade-status  Report upgradability for multiple (or all) recipes
```

devtool subcommands (2)

```
  Working on a recipe in the workspace:
    ide-sdk               Setup the SDK and configure the IDE
    build                 Build a recipe
    rename                Rename a recipe file in the workspace
    edit-recipe           Edit a recipe file
    find-recipe           Find a recipe file
    configure-help        Get help on configure script options
    update-recipe         Apply changes from external source tree to recipe
    reset                 Remove a recipe from your workspace
    finish                Finish working on a recipe in your workspace
  Testing changes on target:
    deploy-target         Deploy recipe output files to live target machine
    undeploy-target       Undeploy recipe output files in live target machine
    build-image           Build image including workspace recipe packages
  Advanced:
    create-workspace      Set up workspace in an alternative location
    export                Export workspace into a tar archive
    extract               Extract the source for an existing recipe
    sync                  Synchronize the source tree for an existing recipe
    import                Import exported tar archive into workspace
    menuconfig            Alter build-time configuration for a recipe
Use devtool <subcommand> --help to get help on a specific command
```

devtool subcommand help

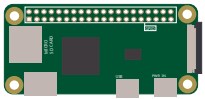
```
devtool check-upgrade-status --help
NOTE: Starting bitbake server...
usage: devtool check-upgrade-status [-h] [--all] [recipe ...]

Prints a table of recipes together with versions currently provided by recipes, and latest upstream versions, when there is a later
version available

arguments:
  recipe      Name of the recipe to report (omit to report upgrade info for all recipes)

options:
  -h, --help  show this help message and exit
  --all, -a   Show all recipes, not just recipes needing upgrade
```

root commit



devtool subcommands

deploy-target
undeploy-target

your layers

finish

modify

upgrade

configure-help build build-image

devtool workspace

find-recipe status

rename reset

add

check-upgrade-status

status

Image credits:
https://openclipart.org/detail/35389/tango-applications-internet
https://openclipart.org/detail/299799/raspberry-pi-zero

The core machinery to work on recipes — used by `devtool`

- Used by `devtool add` to create recipes
- Can also be used:
  - to set variables in a recipe from a script
  - to create and update `bbappend` files

devtool command sample output

```
$ devtool status  # Information about recipes in your workspace
hello: /home/mike/yocto-labs/poky/build/workspace/sources/hello
myman: /home/mike/yocto-labs/poky/build/workspace/sources/myman
```

```
$ devtool search i2c  # Smart search for recipes (name, description, package contents...)
i2c-tools             Set of i2c tools for linux
linux-libc-headers    Sanitized set of kernel headers for the C library's use
linux-mainline
i2cdev                i2c dev tools for Linux
```

```
$ devtool find-recipe i2c-tools  # Find path to recipe
/home/mike/yocto-labs/poky/meta/recipes-devtools/i2c-tools/i2c-tools_4.3.bb
```

Hands-On — QEMU Target Setup

root commit

- Boot the generated image through QEMU
- Setup shortcut for SSH access

Boot the Generated Image Through QEMU

```
$ runqemu slirp nographic ext4
```

Keep this machine running in its terminal.
We will use another terminal for BitBake commands.

Use: `[Ctrl][a],[x]` to quit QEMU when you are done

SSH host definitions

Useful to directly ssh to the target, in particular for `devtool deploy-target` and `devtool undeploy-target`

- ✅ Need an SSH server (like `openssh`) in the target image
- Usage `devtool deploy-target <recipe> <target>`
- `devtool deploy-target` needs the recipe to be built in the workspace
- `devtool undeploy-target` is optional before one more `devtool deploy-target`, but useful to remove no longer wanted files.

Entry in $HOME/.ssh/config

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

Test SSH

root commit

You can now test that SSH to the QEMU target works:

```
$ ssh qemu
```

Define Your Git Identity

root commit

Useful to create git commits and patches

```
$ git config --global user.name "Jules Verne"
$ git config --global user.email "julio@nautilus.net"
```

Create a New Layer

root commit

Useful to store the recipes we create

```
$ source bitbake-builds/distro_poky-master/build-devtool/init-build-env
```

```
$ bitbake-layers create-layer ../layers/meta-foo
$ bitbake-layers add-layer ../layers/meta-foo
```

```
devtool add
```

devtool add — Create new recipe for "nano" package

```
$ devtool add https://www.nano-editor.org/dist/v8/nano-8.6.tar.gz
```

- devtool implicitely creates the workspace if needed

```
INFO: Creating workspace layer in /home/ilab01/bitbake-builds/distro_poky-
master/build-devtool/workspace
```

- It guesses the recipe name: nano (correctly!)
- It looks at the sources and determines it's an *autotooled* project (true!) with pkgconfig and gettext
- It guesses DEPENDS: ncurses, file and zlib (correct!)

devtool add — Create new recipe for "nano" package

root commit

- It creates a draft recipe:

```
INFO: Recipe /home/ilab01/bitbake-builds/distro_poky-master/build-
devtool/workspace/recipes/nano/nano_8.6.bb has been automatically created;
further editing may be required to make it fully functional
```

- You can check the new code:

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

- Let's see if it builds

```
$ devtool build nano
```

- It builds!

```
...
NOTE: nano: compiling from external source tree /home/ilab01/bitbake-
builds/distro_poky-master/build-devtool/workspace/sources/nano
NOTE: Tasks Summary: Attempted 869 tasks of which 861 didn't need to be rerun
and all succeeded.
```

devtool workspace details

- workspace is a layer, with priority 99
  (see conf/layer.conf)
- appends: bbappends for the recipes in the workspace
- recipes: recipes in the workspace
- sources: copies of package sources
  You can directly edit them!
  Each subdirectory is a git repository

~/yocto-labs/poky/build/workspace/sources/myman devtool

~/work/kas/pocketbeagle2/build/workspace tree -L 3

```
.
├── appends
│   └── nano_8.6.bbappend
├── conf
│   └── layer.conf
├── README
├── recipes
│   └── nano
│       └── nano_8.6.bb
└── sources
    └── nano
        ├── ABOUT-NLS
        ├── aclocal.m4
        ├── AUTHORS
        ├── ChangeLog
        ├── compile
        ├── config.guess
        ├── config.h.in
        ├── config.rpath
        ├── config.sub
        ├── configure
        ├── configure.ac
        ├── COPYING
        ├── COPYING.DOC
        ├── depcomp
        ├── doc
        ├── IMPROVEMENTS
        ├── INSTALL
        ├── install-sh
        ├── lib
        ├── m4
        ├── Makefile.am
        ├── Makefile.in
        ├── missing
        ├── NEWS
        ├── po
        ├── README
        ├── src
        ├── syntax
        ├── THANKS
        └── TODO

13 directories, 28 files
```

devtool — deploy nano

- Examine the current build manifest, observe there is no nano package

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

- In the terminal running QEMU, log in and check that there's no nano:

```
root@qemuarm64:~# nano
-sh: nano: not found
```

- Send nano to target (thanks to the SSH configuration defined earlier)

```
$ devtool deploy-target nano qemu
...
INFO: Successfully deployed /home/ilab01/bitbake-builds/distro_poky-master/build-
devtool/tmp/work/cortexa57-oe-linux/nano/8.6/image
```

Testing nano (1)

root commit

- Try to run `nano` on the target

```
root@qemuarm64:~# nano
nano: error while loading shared libraries: libmagic.so.1:
cannot open shared object file: No such file or directory
```

- This is a common issue with `devtool deploy`: it doesn't pull the package dependencies (`RDEPDENDS:nano`), which could be missing on the target.

Testing nano (2)

root commit

- Let's build the image again with all the packages in the workspace:

```
$ devtool build-image core-image-base
```

- Restart QEMU, `nano` should be working now.
- Building an image needed to make sure the recipe is complete, as package dependencies should be pulled automatically.
- Why not just use `bitbake core-image-base`?
  - Unless explicitely added to `conf/local.conf` through `IMAGE_INSTALL`, `nano` wouldn't be added to the image.

devtool finish — Publishing nano recipe

root commit

- Let's store the new recipe in a normal layer:

```
$ devtool finish nano ../layers/meta-foo
NOTE: Starting bitbake server...
ERROR: Source tree is not clean:
...
```

- This error is **not** a problem we introduced.
  It is caused by files generated or modified by `devtool build nano`.
- Adding `-f` to `devtool finish` is supposed to address this, but is currently broken
  (see https://bugzilla.yoctoproject.org/show_bug.cgi?id=15546)
- Here's a workaround, to remove such file changes in `nano` sources:

```
pushd workspace/sources/nano/
git restore .
git clean -fdx
popd
```

devtool finish — Working!

root commit

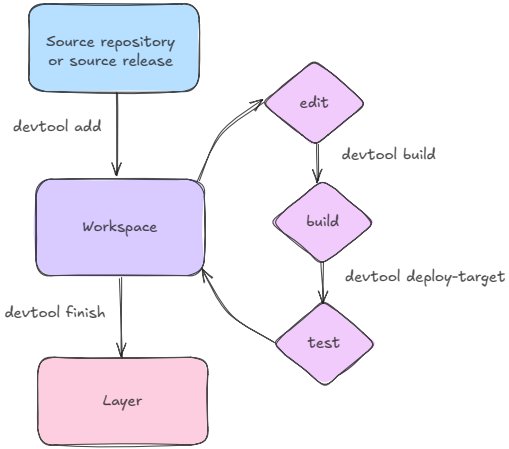
- We can now run devtool finish:

```
$ devtool finish nano ../layers/meta-foo
INFO: No patches or files need updating
INFO: Moving recipe file to /home/ilab01/bitbake-builds/distro_poky-
master/layers/meta-foo/recipes-nano/nano
INFO: Preserving source tree in /home/ilab01/bitbake-builds/distro_poky-
master/build-devtool/workspace/attic/sources/nano.20251126154505
If you no longer need it then please delete it manually.
It is also possible to reuse it via devtool source tree argument.
```

Wrapping up — `devtool add` workflow

root commit

1. Create recipe in workspace from remote or local sources:
   `devtool add https://ftp.gnu.org/gnu/hello/hello-2.12.1.tar.gz`
2. Review and fix the recipe in the workspace
3. Build and test recipe on the target:
   `devtool build hello`
   `devtool deploy-target hello beagleplay`
4. Test `RDEPENDS` by building an entire image:
   `devtool build-image core-image-base`
5. Copy new recipe to layer and clean workspace:
   `devtool finish hello ../../meta-homebrew`
   `devtool finish hello ../../meta-homebrew -f`
   (`-f`: *force*, to ignore irrelevant generated files)



```
devtool upgrade
```

devtool check-upgrade-status

root commit

- Check whether a new release exists upstream:

```
$ devtool check-upgrade-status nano
nano                    8.6             8.7             None
```

- `devtool check-upgrade-status` with no argument can be used for checking for upgrades for all recipes.

devtool upgrade — Upgrade nano (1)

root commit

- Try to upgrade the recipe

```
$ devtool upgrade nano
INFO: Rebasing devtool onto 7689a648b30bb68392afeafe16987d95bfeb2e11
INFO: Upgraded source extracted to /home/ilab01/bitbake-builds/distro_poky-
master/build-devtool/workspace/sources/nano
INFO: New recipe is /home/ilab01/bitbake-builds/distro_poky-master/build-
devtool/workspace/recipes/nano/nano_8.7.bb
```

- Check that the new recipe builds:

```
$ devtool build nano
```

It builds!

devtool upgrade — Upgrade nano (2)

- Check that your image builds:

```
$ devtool build-image core-image-base
```

It builds!

- Restart QEMU and check the nano version on the target:

```
root@qemuarm64:~# nano --version
 GNU nano, version 8.7
 (C) 2025 the Free Software Foundation and various contributors
 Compiled options: --enable-utf8
```

devtool finish — Publishing updated nano recipe

root commit

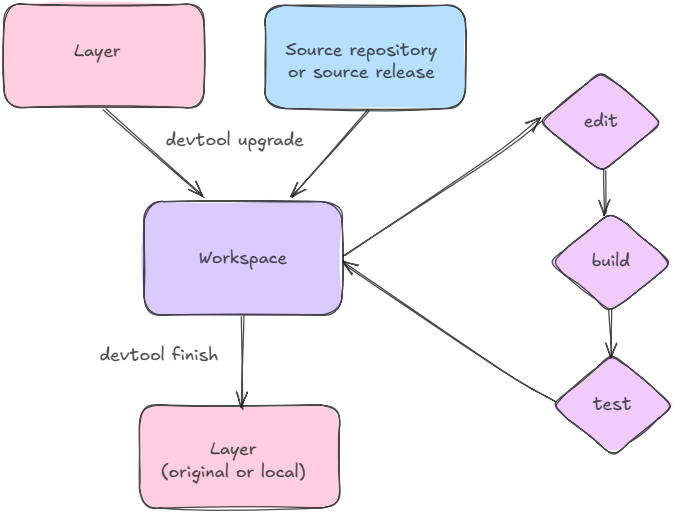
- First, let's clean the sources (same workaround, not needed for all packages)

```
pushd workspace/sources/nano/
git restore .
git clean -fdx
popd
```

- Let's update the recipe in its original layer:

```
$ devtool finish nano ../layers/meta-foo
```

devtool upgrade workflow — Summary



Patching Recipes

Creating Patches With `devtool`

root commit

Use-cases?

- Add or remove functionality
  - Reduce size on target
  - Remove dependencies
- Fix compiling and cross-compiling issues

Let's Add a New Recipe

root commit

```
$ devtool add https://rootcommit.com/pub/conferences/2025/yps/autotool-devtool-
example/v1.0.0.tar.gz
ERROR: Could not auto-determine recipe name, please specify it on the command line
```

```
$ devtool add autotool-devtool-example \
  https://rootcommit.com/pub/conferences/2025/yps/autotool-devtool-
example/v1.0.0.tar.gz
$ devtool build autotool-devtool-example
$ devtool deploy-target autotool-devtool-example qemu
```

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

Modify The Source Code

root commit

Edit the source code:

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

Change from:

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

To:

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

Build, Deploy, Verify

root commit

You can check and build your changes right away,
without committing your changes or making changes to your recipe 😍

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

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

Creating a Patch

Let's store our changes to `meta-foo`

```
$ devtool finish autotool-devtool-example ../layers/meta-foo
NOTE: Starting bitbake server...
ERROR: Source tree is not clean:

 M src/autotool-devtool-example.c
?? cfg/config.guess~
?? cfg/config.h.in~
?? cfg/config.sub~

Ensure you have committed your changes or use -f/--force if you are sure there's nothing that
```

Oops, we have real changes this time. We must commit them:

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

Modified Recipe

meta-foo/recipes-autotool-devtool-example/autotool-devtool-example/autotool-devtool-example_1.0.0.bb

```
SRC_URI = "https://rootcommit.com/pub/conferences/2025/yps/autotool-devtool-
example/v${PV}.tar.gz \
           file://0001-update-salutation.patch \
           "
```

Trying To Upgrade

root commit

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

```
$ devtool upgrade autotool-devtool-example
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 needs help here, as the releases can't be discovered on the website, that's intentional 😉:

```
$ devtool upgrade -V 1.0.1 autotool-devtool-example
ERROR: QA Issue: Missing Upstream-Status in patch
/home/ilab01/bitbake-builds/distro_poky-master/layers/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.

Adding Missing Tag to Patch

root commit

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

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

Trying To Upgrade

```
$ devtool upgrade -V 1.0.1 autotool-devtool-example
INFO: Rebasing devtool onto 13d612f06d0df0c5241797255c3bc69143cec65b
WARNING: Command 'git rebase 13d612f06d0df0c5241797255c3bc69143cec65b' 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/bitbake-builds/distro_poky-master/build-
devtool/workspace/sources/autotool-devtool-example
INFO: New recipe is /home/ilab01/bitbake-builds/distro_poky-master/build-
devtool/workspace/recipes/autotool-devtool-example/autotool-devtool-example_1.0.1.bb
```

Resolving The Conflict (1)

devtool aborted the `git rebase` operation.

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

Resolving The Conflict (2)

root commit

Let's run the rebase operation again

```
$ 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 b5ed868... 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".
hint: Disable this message with "git config advice.mergeConflict false"
Could not apply b5ed868... update salutation
```

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

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

Resolving The Conflict (3)

root commit

From:

```
<<<<<<< HEAD
 /* a meaningful comment */
 printf("Hello, world!\n");
=======
 printf("Hello, devtool!\n");
>>>>>>> b5ed868 (update salutation)
```

To:

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

Resolving The Conflict (4)

```
$ git add src/autotool-devtool-example.c
$ git rebase --continue
$ popd
```

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

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

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

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

Modified Recipe

root commit

```
$ tree --charset=ascii ../layers/meta-foo
../layers/meta-foo
|-- conf
|   `-- layer.conf
|-- COPYING.MIT
|-- README
|-- recipes-autotool-devtool-example
|   `-- autotool-devtool-example
|       |-- autotool-devtool-example
|       |   `-- 0001-update-salutation.patch
|       `-- autotool-devtool-example_1.0.1.bb
`-- recipes-nano
    `-- nano
        `-- nano_8.7.bb

12 directories, 9 files
```

```
devtool modify
```

Modify An Existing Recipe

`devtool modify`

1. Takes an existing recipe from layers
2. Unpacks sources into the workspace
3. Edit recipe or sources

Same operation as in the previous `devtool add` / `devtool upgrade` workflow.

devtool modify — Example (1)

root commit

```
$ devtool modify bc
INFO: Source tree extracted to /home/ilab01/bitbake-builds/distro_poky-
master/build-devtool/workspace/sources/bc
INFO: Recipe bc now set up to build from /home/ilab01/bitbake-builds/distro_poky-
master/build-devtool/workspace/sources/bc
```

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

```
$ pushd workspace/sources/bc
$ 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  m4  Makefile.am  Makefile.in  missing  NEWS  README
Test  ylwrap
```

devtool modify — Example (2)

root commit

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

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

Commit changes and run devtool finish

```
$ git add bc/main.c
$ git commit -s -m "change help text"
$ popd
$ devtool finish bc ../layers/meta-foo
NOTE: Writing append file /home/ilab01/bitbake-builds/distro_poky-
master/layers/meta-foo/recipes-extended/bc/bc_%.bbappend
NOTE: Copying 0001-change-help-text.patch to /home/ilab01/bitbake-
builds/distro_poky-master/layers/meta-foo/recipes-extended/bc/bc/0001-change-help-
text.patch
INFO: Cleaning sysroot for recipe bc...
```

`devtool finish` realized the `bc` recipe is not in `meta-foo`

- Therefore it created a `bbappend` and placed the patch next to it
- Naturally if we had passed the path to `openembedded-core/meta` it would have modified the original recipe.

devtool modify workflow

1. Import an existing recipe into the workspace

```
$ devtool modify myman
```

2. Modify sources or recipe:
3. Build and test the sources

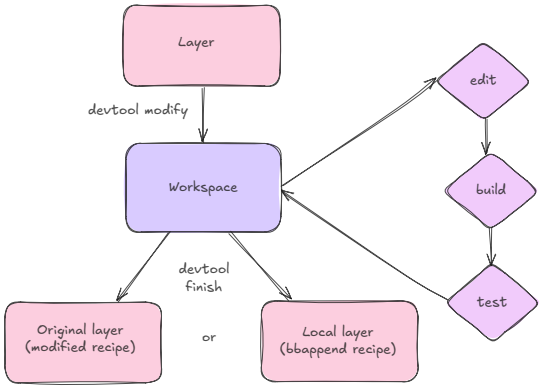
```
$ devtool build myman
$ devtool deploy-target myman beagleplay
```

4. Commit your changes

```
$ pushd workspace/sources/myman
$ git commit -as
$ popd
```

5. Publish your changes to a layer

```
$ devtool finish -f myman ../../meta-homebrew
```



`devtool` eSDK mode

The Extensible SDK — eSDK

root commit

- The eSDK includes additional functionality in addition to the standard SDK
- The additional functionality is available through `devtool`

devtool — Mode Commands

root commit

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

Wrapping Up

devtool — Key Takeaways

- Use `devtool add` to create new recipes
- Very smart logic: produced recipes are almost ready
- Use `devtool modify` and `devtool upgrade` to modify recipes
- The `workspace` directory is a top priority layer
- Very easy to compile applications under development (`devtool build`) and test them on the target (`devtool deploy-target`) without having to commit your changes so that they can be used by a formal recipe.

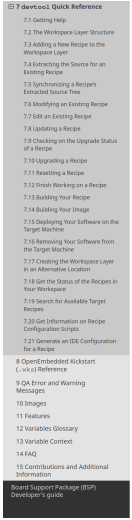
devtool — Official Documentation

Yocto Project Reference Manual

- Chapter 7 — devtool Quick Reference
- https://docs.yoctoproject.org/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/sdk-manual/extensible.html



7 devtool Quick Reference

The devtool command-line tool provides a number of features that help you build, test, and package software. This command is available alongside the bitbake command. Additionally, the devtool command is a key part of the extensible SDK.

This chapter provides a Quick Reference for the devtool command. For more information on how to apply the command when using the extensible SDK, see the "Using the Extensible SDK" chapter in the Yocto Project Application Development and the Extensible Software Development Kit (eSDK) manual.

7.1 Getting Help

The devtool command line is organized similarly to Git in that it has a number of sub-commands for each function. You can run devtool --help to see all the commands:

```
$ devtool --help
NOTE: Starting bitbake server...
usage: devtool [--basepath BASEPATH] [--bbpath BBPATH] [-d] [-q] [--color COLOR] [-h]

OpenEmbedded development tool

options:
  --basepath BASEPATH   Base directory of SDK / build directory
  --bbpath BBPATH       Explicitly specify the BBPATH, rather than getting it from th
  -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
    modify              Modify the source for an existing recipe
    upgrade             Upgrade an existing recipe
  Getting information:
    status              Show workspace status
    latest-version      Report the latest version of an existing recipe
    check-upgrade-status  Report upgradability for multiple (or all) recipes
    search              Search available recipes
  Working on a recipe in the workspace:
    build               Build a recipe
    ide-sdk             Setup the SDK and configure the IDE
    rename              Rename a recipe file in the workspace
    edit-recipe         Edit a recipe file
    find-recipe         Find a recipe file
    configure-help      Get help on configure script options
    update-recipe       Apply changes from external source tree to recipe
    reset               Remove a recipe from your workspace
    finish              Finish working on a recipe in your workspace
  Testing changes on target:
    deploy-target       Deploy recipe output files to live target machine
    undeploy-target     Undeploy recipe output files in live target machine
    build-image         Build image including workspace recipe packages
  Advanced:
    create-workspace    Set up workspace in an alternative location
    import              Import exported tar archive into workspace
    export              Export workspace into a tar archive
```

Questions? Comments?

- mo@rootcommit.com
- https://fosstodon.org/@MichaelOpdenacker
- XMPP: omichael@conversations.im
- Signal: rootcommit.01
- Slides available under the CC-By-SA 4.0 license
  https://rootcommit.com/pub/conferences/2025/yps/devtool-hands-on/
- Sources (LaTeX):
  https://gitlab.com/rootcommit/devtool-hands-on/

Learn by doing

Yocto Project & OpenEmbedded training course

root commit

Online course

- 32 hours (8 x 4 hours)
- 75 % of practical activities
- Do labs by yourself instead of watching demos
- No exhaustive theory
- Challenging, varied and fun learning techniques
- Use your favorite Linux distro
- Limited to 8 participants
- Video recordings of lectures