

# Utilisation de devtool

**Christophe BLAESS**

christophe.blaess@logilin.fr

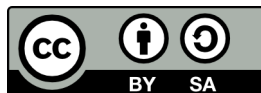
<https://www.blaess.fr/christophe/>



Ingénierie et formations sur Linux et les logiciels libres  
<https://www.logilin.fr>

<b>Principe de Devtool.....</b>	<b>3</b>
<b>Créer ou mettre à jour une recette.....</b>	<b>4</b>
Création de recette.....	4
Mise à jour d'une recette existante.....	5
Travaux pratiques : création d'une recette.....	6
<b>Tester et valider une recette.....</b>	<b>7</b>
Test d'une recette en cours de modification.....	7
Édition du fichier de recettes.....	8
Abandon d'une modification.....	8
Valider une recette.....	9
<b>Créer un patch sur une recette existante.....</b>	<b>11</b>
Rechercher le package qui a installé un fichier sur la cible.....	11
Rechercher la recette d'un package.....	11
Patcher les fichiers source d'une recette.....	11
Travaux pratiques : patch sur une recette existante.....	12

Ce support de formation est distribué sous licence **Creative Commons 4.0**



*(Attribution - Partage dans les mêmes conditions).*

Vous êtes libres de copier et partager ce document, en mentionnant son origine. Si vous l'intégrez dans un contenu plus vaste, ce dernier devra être distribué avec les mêmes droits.

Ce cours a été rédigé en utilisant des logiciels libres sur système d'exploitation Linux :

- *LibreOffice Writer* pour le support et la mise en page
- *Gimp* pour les images bitmap
- *Excalidraw* (en ligne) pour les schémas vectoriels.

Yocto Project avancé

YPA v. 1.2

<https://www.blaess.fr/christophe/>

<https://www.logilin.fr>

# Principe de Devtool

La commande **devtool** (script pour Python3) est livrée avec Poky.

Elle permet de rechercher des recettes, d'en créer, de préparer des patches pour étendre une recette existante, etc.

Dans le répertoire de *build*, devtool crée un layer de travail nommé **`workspace`**, qu'elle ajoute aux layers en cours :

```
$ ls
bitbake-cookerdaemon.log  cache  conf  tmp  workspace

$ bitbake-layers show-layers
[...]
```

meta	/home/training/builds/build-qemu/../../../../layers/poky/meta	5
meta-poky	/home/training/builds/build-qemu/../../../../layers/poky/meta-poky	5
workspace	/home/training/builds/build-qemu/workspace	99

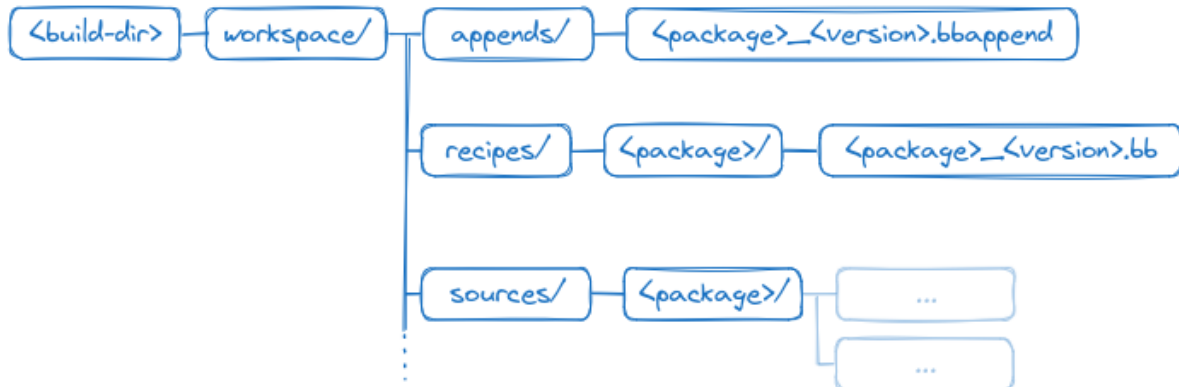
On peut supprimer sans problème le layer **`workspace`**, en pensant à effacer la ligne correspondante de **`conf/bblayers.conf`**.

# Créer ou mettre à jour une recette

## Création de recette

Créer une recette automatiquement en analysant les sources présentes dans un dépôt distant :

```
$ devtool add <package> <URL>
```



## Mise à jour d'une recette existante

Vérifier la dernière version de code disponible :

```
$ devtool latest-version <package>
```

Affiche la version actuellement utilisée et la dernière version disponible. Elle est déterminée en examinant le site indiqué dans la variable `UPSTREAM\_CHECK\_URI` de la recette.

Vérifier la possibilité de faire une mise à jour :

```
$ devtool check-upgrade-status <package>
```

Affiche la version actuelle, la dernière disponible, le nom du mainteneur.

Si la mise à jour n'est pas possible (en raison des dépendances) une explication est ajoutée sur la ligne.

Mettre à jour automatiquement une recette :

```
$ devtool upgrade <package>
```

## Travaux pratiques : création d'une recette

→ Créez une recette `hello-auto` pour le projet se trouvant ici :

<https://github.com/logilin/hello-autotools>

→ Quelle est la dernière version de l'utilitaire `nano` ?

→ Pouvez-vous faire une mise à jour ? Si oui, faites-la.

### Solution

```
$ devtool add hello-auto https://github.com/logilin/hello-autotools
[...]
INFO: Recipe /home/training/builds/build-qemu/workspace/recipes/hello-
autotools/hello-auto_git.bb has been automatically created; further editing
may be required to make it fully functional
```

```
$ ls workspace/appends/
hello-auto_git.bbappend
```

```
$ ls workspace/recipes/
hello-auto
```

```
$ ls workspace/sources/
hello-auto
```

```
$ ls workspace/sources/hello-auto/
LICENSE Makefile.am Makefile.in README.md aclocal.m4 autom4te.cache
compile config.h.in configure configure.ac depcomp hello-autotools.c
```

```
$ devtool latest-version nano
[...]
INFO: Current version: 6.2
INFO: Latest version: 7.2
```

```
$ devtool check-upgrade-status nano
[...]
INFO: nano          6.2          7.2          None
```

```
$ devtool upgrade nano
[...]
INFO: Fetching https://nano-editor.org/dist/v7/nano-7.2.tar.xz...
INFO: Upgraded source extracted to /home/training/builds/build-qemu/
workspace/sources/nano
INFO: New recipe is /home/training/builds/build-qemu/workspace/recipes/nano/nano_7.2.bb
```

## Tester et valider une recette

### Test d'une recette en cours de modification

Compiler un simple package en cours d'édition :

```
$ devtool build <package>
```

Faire un build complet en intégrant les recettes en cours d'édition :

```
$ devtool build-image <image-name>
```

### À vous...

Compilez votre image, puis sur la cible vérifiez la version de `nano` et la présence de `hello-autotools`.

### Solution

```
$ devtool build-image my-image  
[...]  
$ runqemu nographic  
[...]  
# hello-autotools  
Hello from qemuarm (built with autotools)
```

## Édition du fichier de recettes

Une recette créée par ``devtool add`` ou modifiée par ``devtool upgrade`` peut être éditée avec :

```
$ devtool edit-recipe <package>
```

L'éditeur utilisé est celui indiquée dans la variable ``EDITOR`` du shell.

## Abandon d'une modification

Une recette en cours de travail peut être abandonnée avec

```
$ devtool reset <package>
```

Il est alors souvent nécessaire de nettoyer le répertoire ``workspace/sources/<package>`` avec un ``rm -r``.



## Valider une recette

On peut enregistrer une recette que l'on vient de créer avec :

```
$ devtool finish <package> <layer>
```

Si on a fait un build, il faut nettoyer le répertoire de sources au préalable avec

```
$ cd workspace/sources/<package>
```

```
$ git stash
```

```
$ git clean -d -f
```

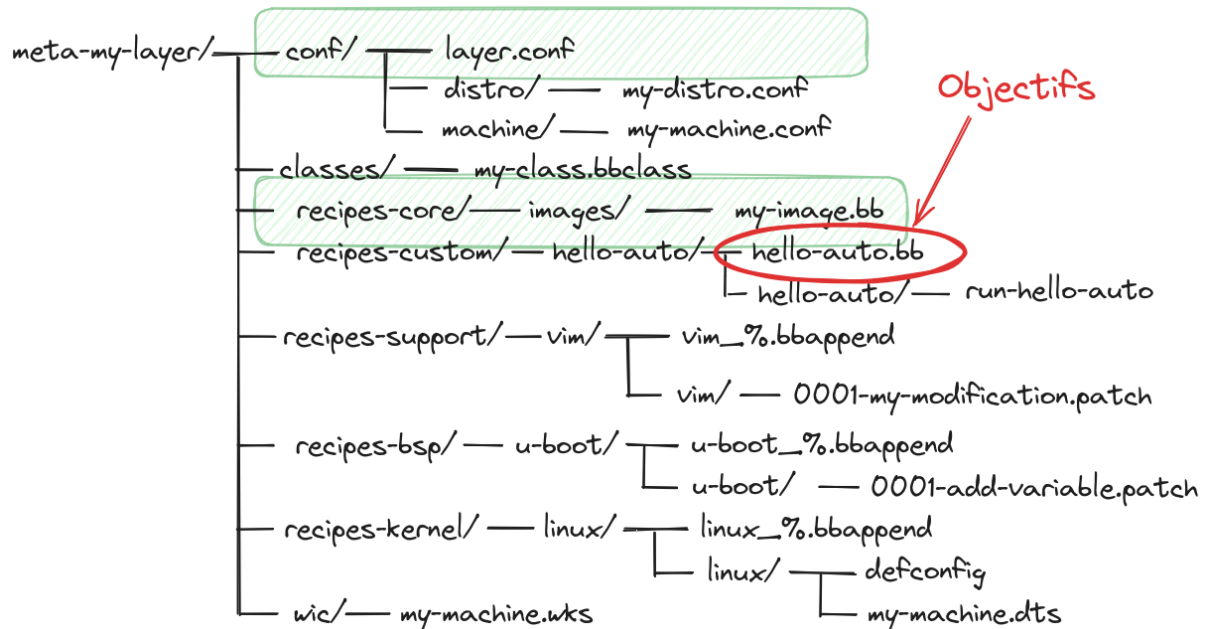
```
$ cd ../../../../
```

Il est conseillé, une fois la validation terminée d'effacer les sources avec :

```
$ rm -rf workspace/sources/<package>
```

## Travaux pratiques : enregistrement d'une nouvelle recette

→ Validez la recette `hello-auto` et enregistrez-la dans votre *layer*.



## Solution

```
$ cd workspace/sources/hello-auto/
$ git clean -d -f
$ git stash
$ devtool finish hello-auto ../../layers/meta-my-layer/
$ rm -rf workspace/sources/hello-auto/
```

## Créer un patch sur une recette existante

### Rechercher le package qui a installé un fichier sur la cible

```
$ devtool search </file/pathname/on/the/target>
```

### Rechercher la recette d'un package

```
$ devtool find-recipe <package-name>
```

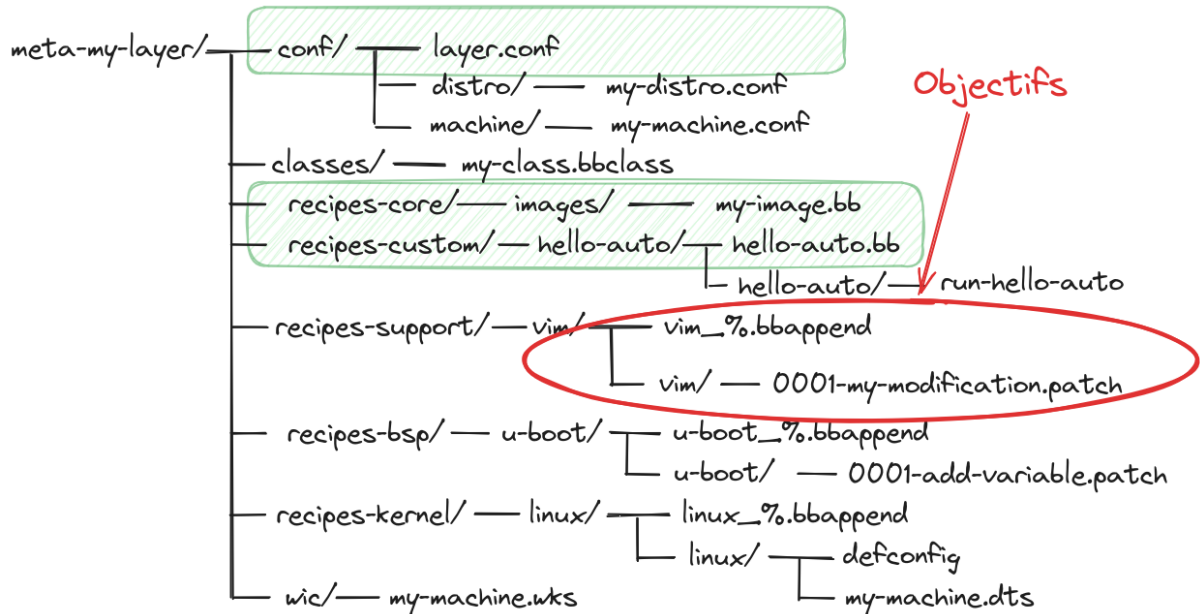
### Patcher les fichiers source d'une recette

```
$ devtool modify <package-name>
```

Éditer les sources dans `workspace/sources/<package>`, *commiter* les modifications et utiliser l'option `--force-patch-refresh` de `devtool finish`.

## Travaux pratiques : patch sur une recette existante

- Utilisez `devtool` pour éditer les sources de `vim`
- Recherchez la chaîne "Vi IMproved" dans le fichier `src/version.c` et ajoutez quelques mots – par exemple "(modified)" – sur la même ligne (5564)



```
$ devtool modify vim
```

```
$ cd workspace/sources/vim/
```

```
$ nano src/version.c
```

```
[...]
    N_("VIM - Vi IMproved - (Modified)",
    "",
    N_("version "),
    N_("by Bram Moolenaar et al."),
[...]
```

```
$ git commit src/version.c -m 'modify welcome message'
```

```
$ cd ../../../../
```

```
$ devtool finish vim ../../layers/meta-my-layer/
```

```
$ rm -rf workspace/sources/vim/
```