Introduction to `devtool` in Yocto

What is `devtool`?

`devtool` is a command-line utility introduced in the **Yocto Project 2.0 (Jethro release)** to simplify the development and modification of recipes and packages within the Yocto build environment. It provides a streamlined workflow for adding new software, modifying existing recipes, and testing builds without making permanent changes to layers until the developer finalizes them.

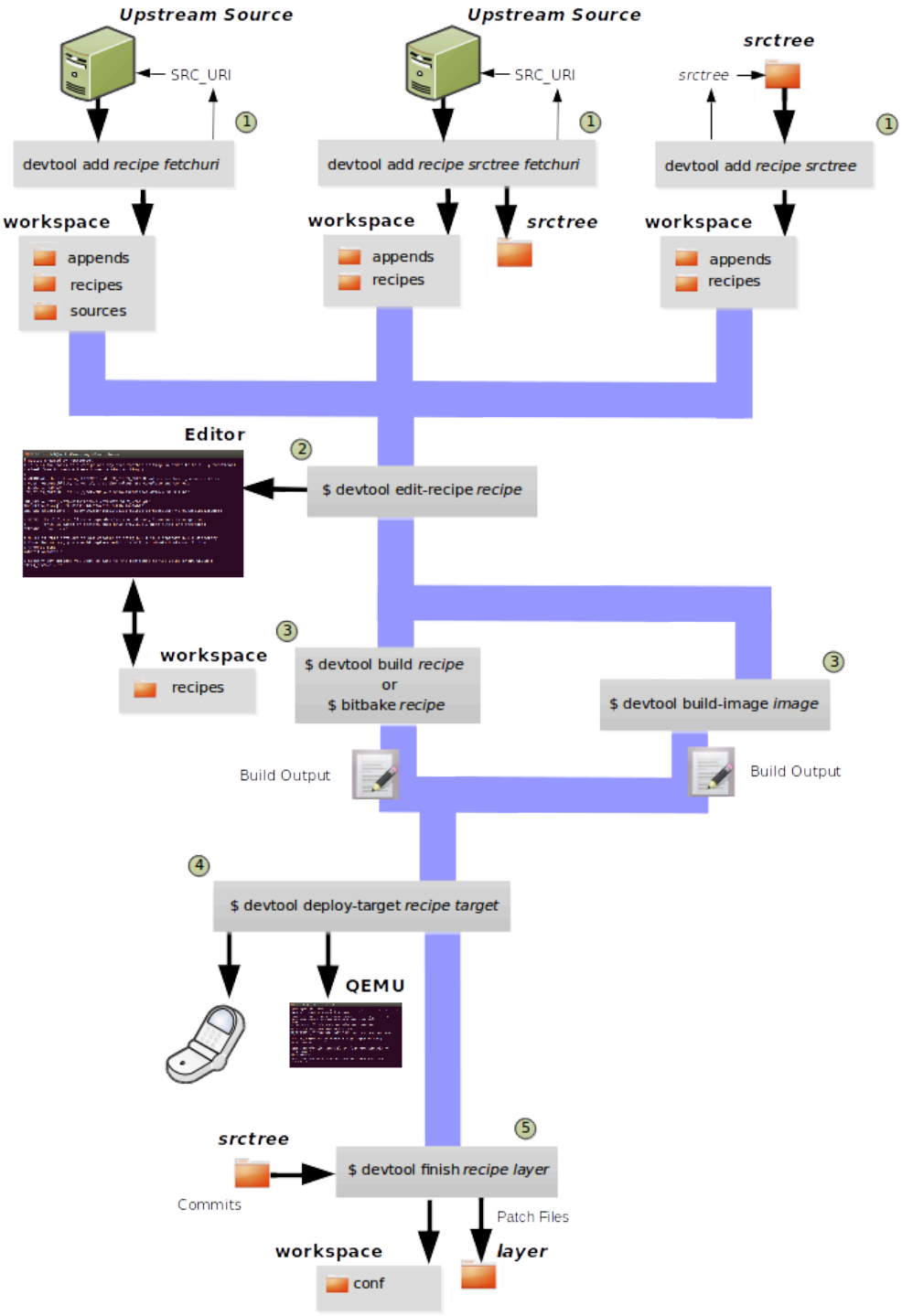
Key Features of `devtool`

- Rapid integration of new software into Yocto.
- Facilitates modifying and testing existing recipes.
- Works seamlessly with both local source code and remote repositories like GitHub.
- Helps developers avoid manually creating `.bb` recipes and debugging dependencies.
- Allows testing on the target device before finalizing the recipe.

`devtool` Commands and Their Usage

Command	Description
`devtool add <recipe-name> <source-path or git-repo>`	Creates a new recipe from a local or remote source.
`devtool edit-recipe <recipe-name>`	Opens the recipe file for modification.
`devtool build <recipe-name>`	Compiles the package using BitBake.
`devtool deploy-target <recipe-name> <user@target-ip>`	Deploys the built package to the target device.
`devtool finish <recipe-name> <layer-path>`	Moves the recipe from the workspace to the specified custom layer.
`devtool reset <recipe-name>`	Removes temporary work on the recipe and resets its workspace.

`devtool modify <recipe-name>`	Modifies an existing recipe (including kernel recipes).



Example 1: Adding a Recipe from a Local Source

1. **Set up the Yocto build environment:**

   ```
   source poky/oe-init-build-env
   ```

2. **Add a local project (e.g., a simple C application):**

```
devtool add my-app /path/to/local/source
```

3. **Edit the recipe if necessary:**

```
devtool edit-recipe my-app
```

4. **Build the package:**

```
devtool build my-app
```

5. **Deploy to the target device:**

```
devtool deploy-target my-app user@target-ip
```

6. **Move the recipe to a custom layer once validated:**

```
devtool finish my-app ../meta-mycustomlayer
```

7. **Reset the temporary workspace:**

   ```
   devtool reset my-app
   ```

Example 2: Adding a Recipe from a GitHub Repository

1. **Initialize the build environment:**

   ```
   source poky/oe-init-build-env
   ```

2. **Add a project directly from GitHub:**

   ```
   devtool add my-app
   "git://github.com/user/my-app.git;protocol=https;branch=main"
   ```

3. **Build and test as in the local source example.**

Handling Dependencies and Issues

- If `devtool add` defaults to the `master` branch when the repository has `main`, explicitly specify it as shown in the GitHub example above.
- If the project has multiple dependencies, ensure they are listed under the `DEPENDS` variable in the recipe.

If encountering fetch errors, manually inspect the repository using:
`git ls-remote https://github.com/user/my-app.git`

Using `devtool` for Kernel Recipes

Can We Write Kernel Recipes Using `devtool`?

Yes, but `devtool add` does not directly support kernel recipes. Instead, you must use `devtool modify` to work with the kernel source.

Steps to Work with Kernel Recipes Using `devtool`

1. **Set Up the Yocto Build Environment:**

```
source poky/oe-init-build-env
```

2. **Modify an Existing Kernel Recipe:**

```
devtool modify linux-yocto
```

   This command fetches the kernel source and sets up a working environment under `workspace/sources/linux-yocto`.

3. **Make Changes to the Kernel:** After modifying the kernel (e.g., adding patches, changing configurations), rebuild it:

```
devtool build linux-yocto
```

4. **Deploy to Target for Testing:**

```
devtool deploy-target linux-yocto user@target-ip
```

5. **Finalizing the Kernel Changes:**

```
devtool finish linux-yocto ../meta-mycustomlayer
```

This generates a `.bbappend` file inside your custom layer (`meta-mycustomlayer`).

6. **Reset Temporary Workspace:**

```
devtool reset linux-yocto
```

7. ## Alternative: Creating a New Kernel Recipe Manually

If you need to add a **new kernel recipe**, manually create a recipe in your custom layer (`meta-mycustomlayer/recipes-kernel/linux/`) with a `.bb` file:

```
DESCRIPTION = "Custom Linux Kernel"
LICENSE = "GPL-2.0"
SRC_URI = "git://github.com/torvalds/linux.git;protocol=https;branch=main"
PV = "5.10"
S = "${WORKDIR}/git"

inherit kernel
```

Build it using:

```
bitbake linux-mycustom
```

Best Practices for Using `devtool`

- Always finalize the recipe using `devtool finish` before committing changes.
- Use `devtool reset` to clean up temporary workspaces and avoid clutter.
- Store custom recipes in a dedicated layer (e.g., `meta-mycustomlayer`).
- Prefer `bitbake-layers add-layer` to include custom layers in `bblayers.conf`.
- For debugging, check logs in `tmp/work` or use `bitbake -e` to inspect environment variables.
