Contents

Quickstart

You can get started right away without understanding anything by reading this short document, but you will lose out on some of the more "advanced" features

So. . . skip the README and get started. Unfortunately this documentation is pretty lacking, but here it goes

First, Consider Using The Development Branch And Submitting Issues

Try the `devel` branch . . . the layout might be less overwhelming to the eye, and less confusing and it eliminates the possibility of file collisions with your project

Starting A Fresh Project

If you are starting with an empty project and nothing yet in source control, try this

Starting From Scratch

Assuming you have *no code whatsoever yet*, get started by just cloning the `pybuild` repository, removing the `.git` directory and then beginning your development. Update `venv/project-requirements.txt` as you go.

`$ $EDITOR venv/project-requirements.txt [add your dependencies into the file] $ make [python2|python3] && source venv/bin/activate $ rm -rf .git $ [start your development and use git as usual]`

Starting From An Empty `git` Repository

First, you should be careful. If your repository already has files in it and there is a name collision with pybuild23 files (i.e. venv/, Makefile, etc.) you could end up with things overwritten. You should really start with a fresh repository, else manually copy pybuild23 into the existing repository

`$ make new REPO=ssh://github.com/yourname/yourproject $ cd ../yourproject $ EDITOR venv/project-requirements.txt [add your dependencies into the file]`

If you want to "install" pybuild23 into an existing project that is already checked out on your local filesystem

`$ cd /your/checkedout/repo $ git clone https://github.com/mzpqnxow/pybuild23 $ cp -r pybuild23/{.gitignore,.styl` `. $ rm -rf pybuild23 $ git add venv Makefile packages etc $ git commit venv Makefile packages etc $ git push`

If you have a project deployed via pybuild23 and you want to save the versions via pip freeze in a dated file

`$ make freeze`