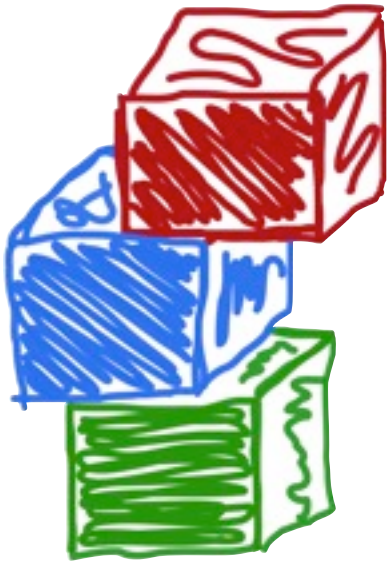


easybuild

building software with ease

kenneth.hoste@ugent.be

HPC UGent

- central contact for HPC at Ghent University
- part of central IT department (DICT)
- member of Flemish supercomputer centre (VSC)
  - collaboration between Flemish university associations

- six Tier2 systems, one Tier1 system
  - #163 in Top500
- team consists of 7 FTEs
- tasks include system administration of HPC infrastructure user support, user training, ...

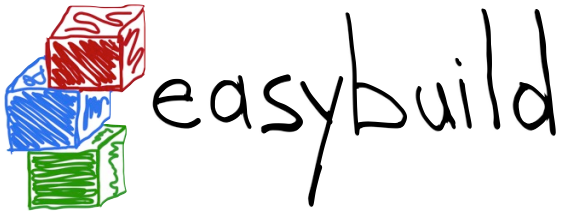
Scientific software

Scientists spend their time and effort in developing (and testing) their code, not in maintaining it.

Build procedures for scientific software are often:

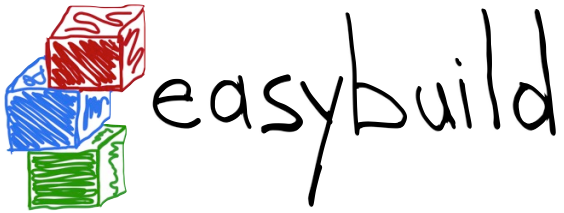
- ***incomplete**:* e.g., no actual *install* step, only build-in-src-dir
- ***non-standard**:* e.g., requiring human interaction
- ***customized**:* custom scripts for configuration, building, ... instead of configure, cmake, make, etc.
- ***hard-coded**:* no configure option for libraries, compiler commands and flags, ...

Very time-consuming for HPC user support teams!



Current tools are lacking

compare with package managers, existing tools



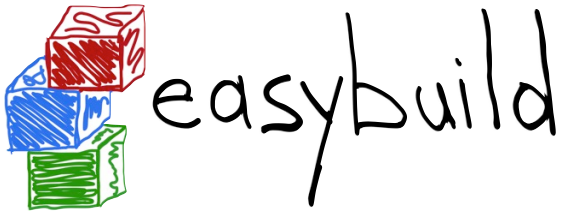
What we need...

flexibility

co-existence of versions

dependency handling

sharing of implementations of install procedures



Building software with ease

EasyBuild is a software build and installation framework written in Python.

Developed in-house for over 3.5 years, available on GitHub (GPLv2) since April 2012, v1.0.0 (stable API) just released.

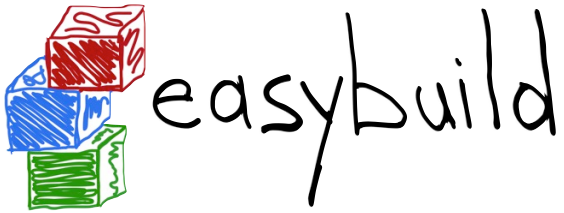
- a robust framework with supporting functionality
  - extracting, patching, executing shell commands, creating module files, ...
- support for GCC and Intel compilers, ATLAS, Intel MKL, ...
- modular support for custom software build/install procedures
  - custom *easyblocks* available for 77 software packages, more being ported
- 338 example easyconfigs for 148 different software packages

Basic usage and dependencies

mention (current) dependencies:

Python 2.4 or higher (no Python 3.x yet)

environment modules



Quick demo for the impatient

```
$ easy_install --user easybuild
$ eb HPL-2.0-goalf-1.1.0-no-OFED.eb --robot
```

- downloads all required sources
- builds and installs *goalf* compiler toolchain

  GCC, OpenMPI, ATLAS, LAPACK, FFTW, BLACS, ScaLAPACK
- builds and installs HPL benchmark (Linpack) with *goalf* toolchain
- default: everything under $HOME/.local/easybuild

EasyBuild terminology

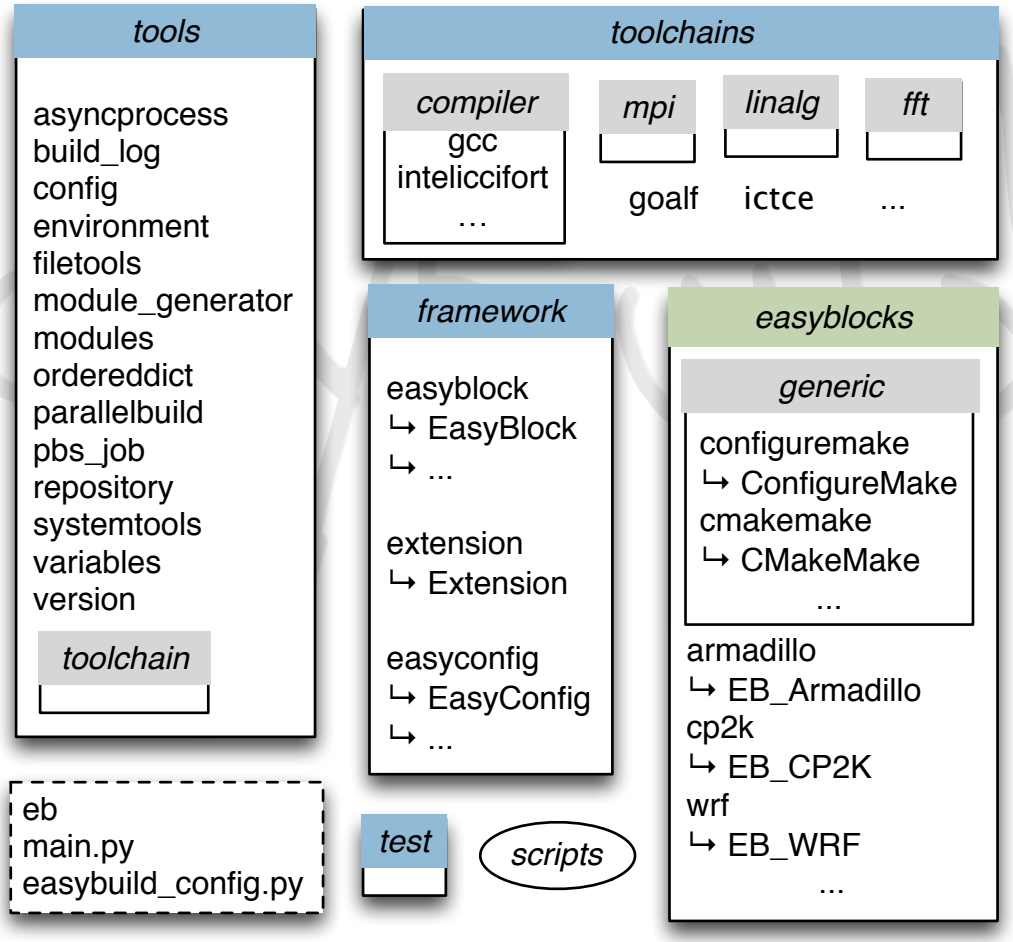
easyblocks

easyconfigs

(compiler) toolchain

High-level design

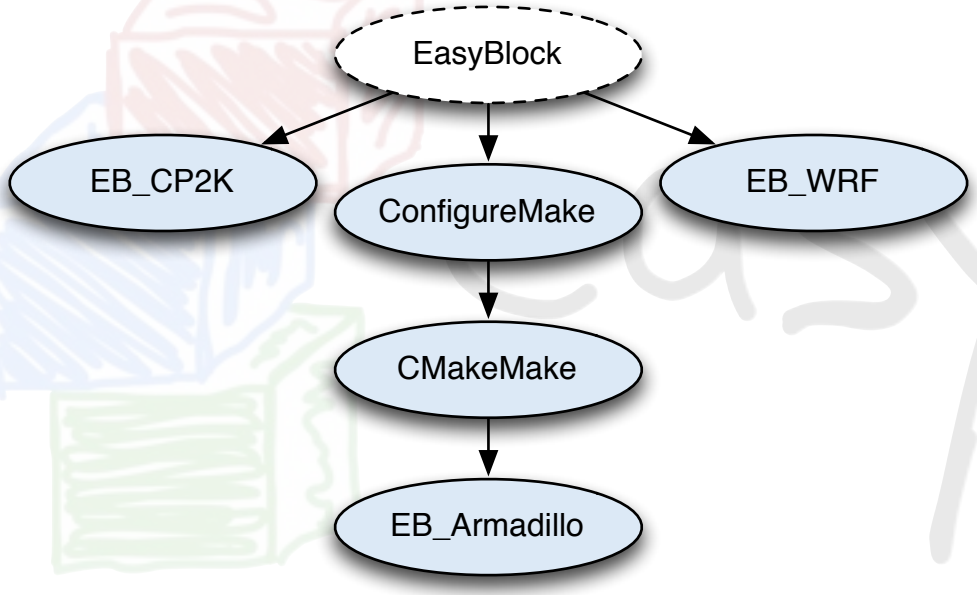
discuss high-level design in packages/modules



Easyblocks

discuss easyblocks

highlight modular design



easyblocks

generic

configuremake
↳ ConfigureMake
cmakemake
↳ CMakeMake
...

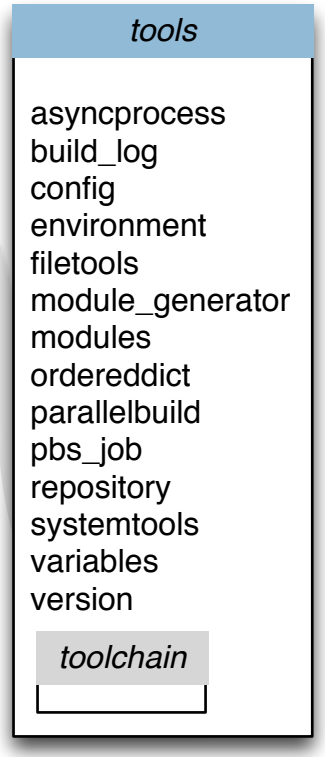
armadillo
↳ EB_Armadillo
cp2k
↳ EB_CP2K
wrf
↳ EB_WRF
...

Easyconfig files

briefly discuss easyconfig files

Supporting functionality

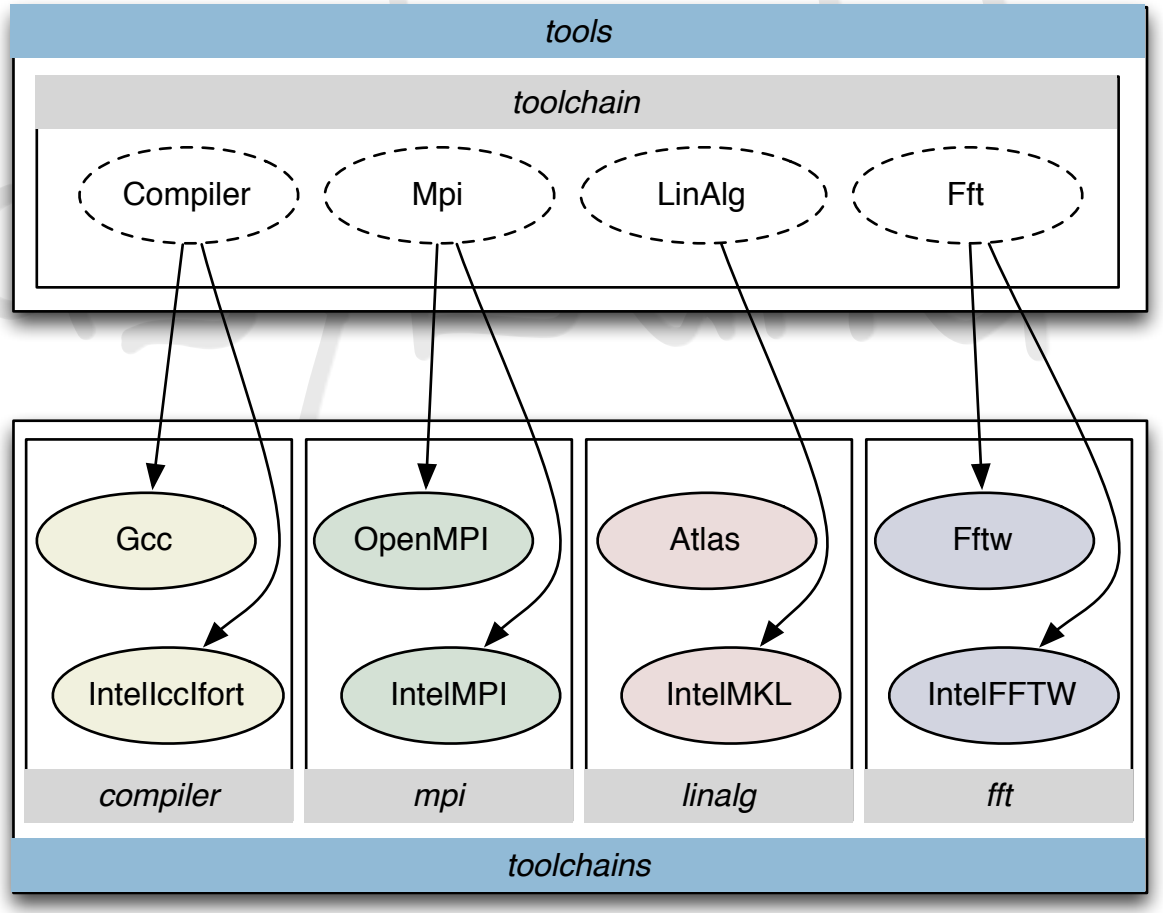
discuss tools package



Compiler toolchains

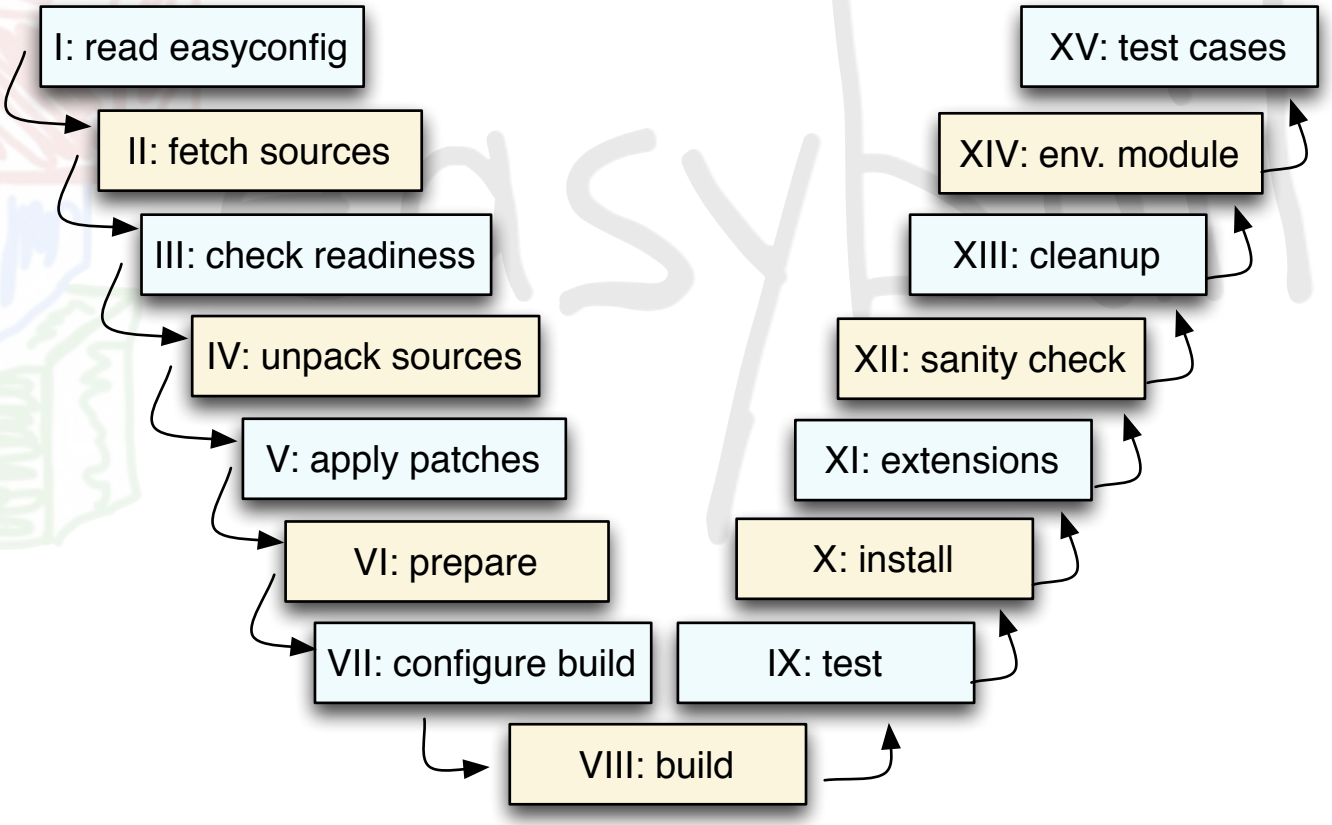
discuss compiler toolchains

highlight modular design



Step-wise install procedure

briefly describe install procedure performed by EasyBuild



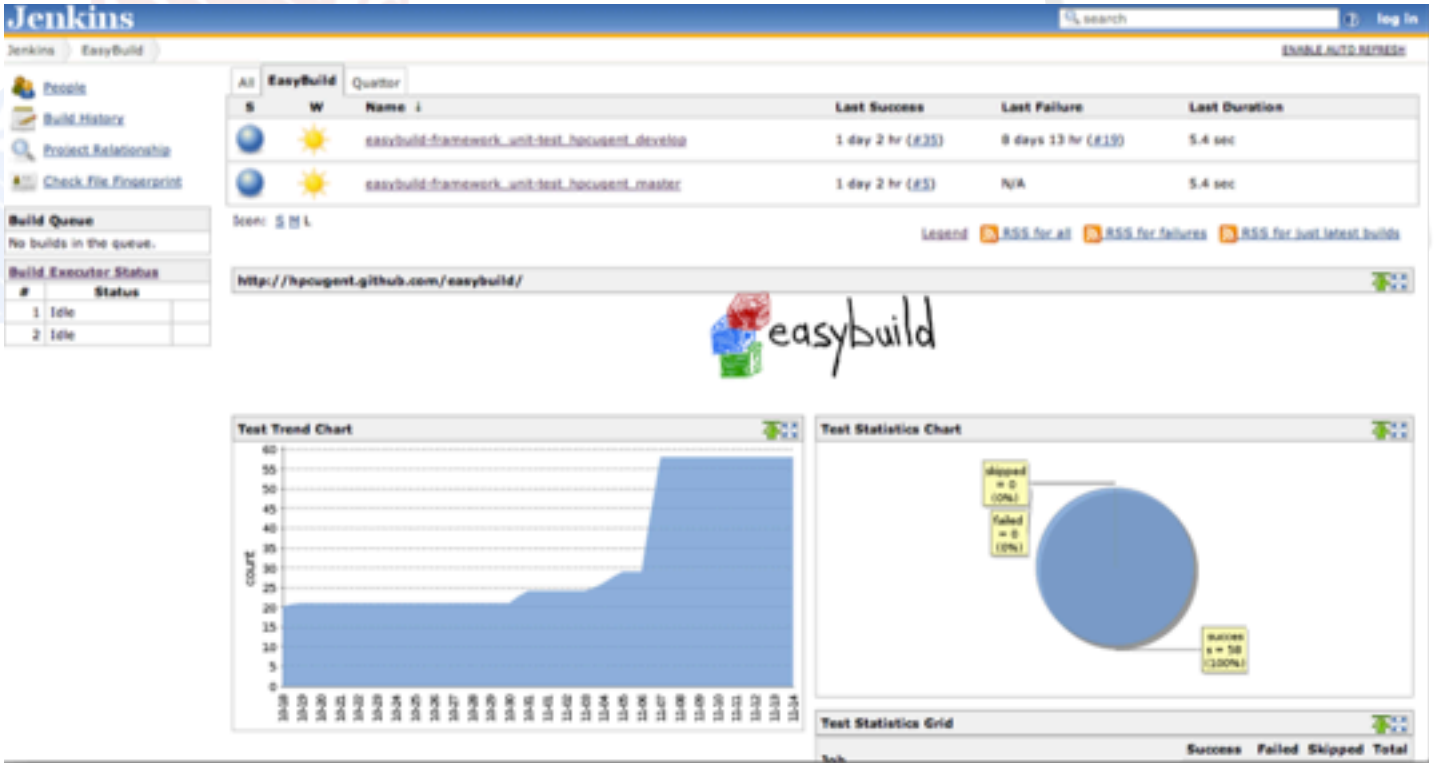
Features

build/install log and easyconfig archiving

Features

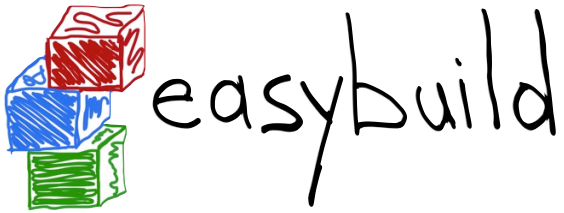
regression and unit testing

mention Jenkins (screenshot)



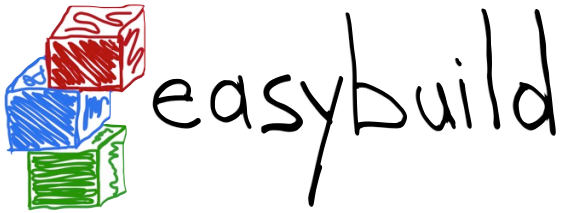
Features

automatic dependency resolution



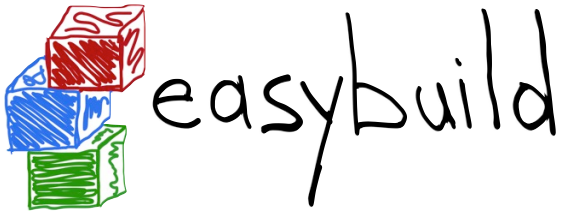
Features

support for interactive installers



Features

building software in parallel



Use case: WRF

explain non-standard install procedure of WRF

Use case: WRF - easyblock (1/2)

show and explain WRF easyblock

```
import fileinput, os, re, sys
import easybuild.tools.environment as env
from easybuild.easyblocks.netcdf import set_netcdf_env_vars
from easybuild.framework.easyblock import EasyBlock
from easybuild.framework.easyconfig import MANDATORY
from easybuild.tools.filetools import patch_perl_script_autoflush, run_cmd, run_cmd_qa
from easybuild.tools.modules import get_software_root
class EB_WRF(EasyBlock):
  def __init__(self, *args, **kwargs):
    super(EB_WRF, self).__init__(*args, **kwargs)
    self.build_in_installdir = True
  @staticmethod
  def extra_options():
    extra_vars = [('buildtype', [None, "Type of build (e.g., dmpar, dm+sm).", MANDATORY])]
    return EasyBlock.extra_options(extra_vars)
  def configure_step(self):
    # prepare to configure
    set_netcdf_env_vars(self.log)
    jasper = get_software_root('JasPer')
    jasperlibdir = os.path.join(jasper, "lib")
    if jasper:
      env.setvar('JASPERINC', os.path.join(jasper, "include"))
      env.setvar('JASPERLIB', jasperlibdir)
    env.setvar('WRFIO_NCD_LARGE_FILE_SUPPORT', '1')
    patch_perl_script_autoflush(os.path.join("arch", "Config_new.pl"))
    known_build_types = ['serial', 'smpar', 'dmpar', 'dm+sm']
    self.parallel_build_types = ["dmpar", "smpar", "dm+sm"]
    bt = self.cfg['buildtype']
    if not bt in known_build_types:
      self.log.error("Unknown build type: '%s' (supported: %s)" % (bt, known_build_types))
    # run configure script
    bt_option = "Linux x86_64 i486 i586 i686, ifort compiler with icc"
    bt_question = "\s*(?P<nr>[0-9]+).\s*%s\s*\(%s\)" % (bt_option, bt)
```

Use case: WRF - easyblock (2/2)

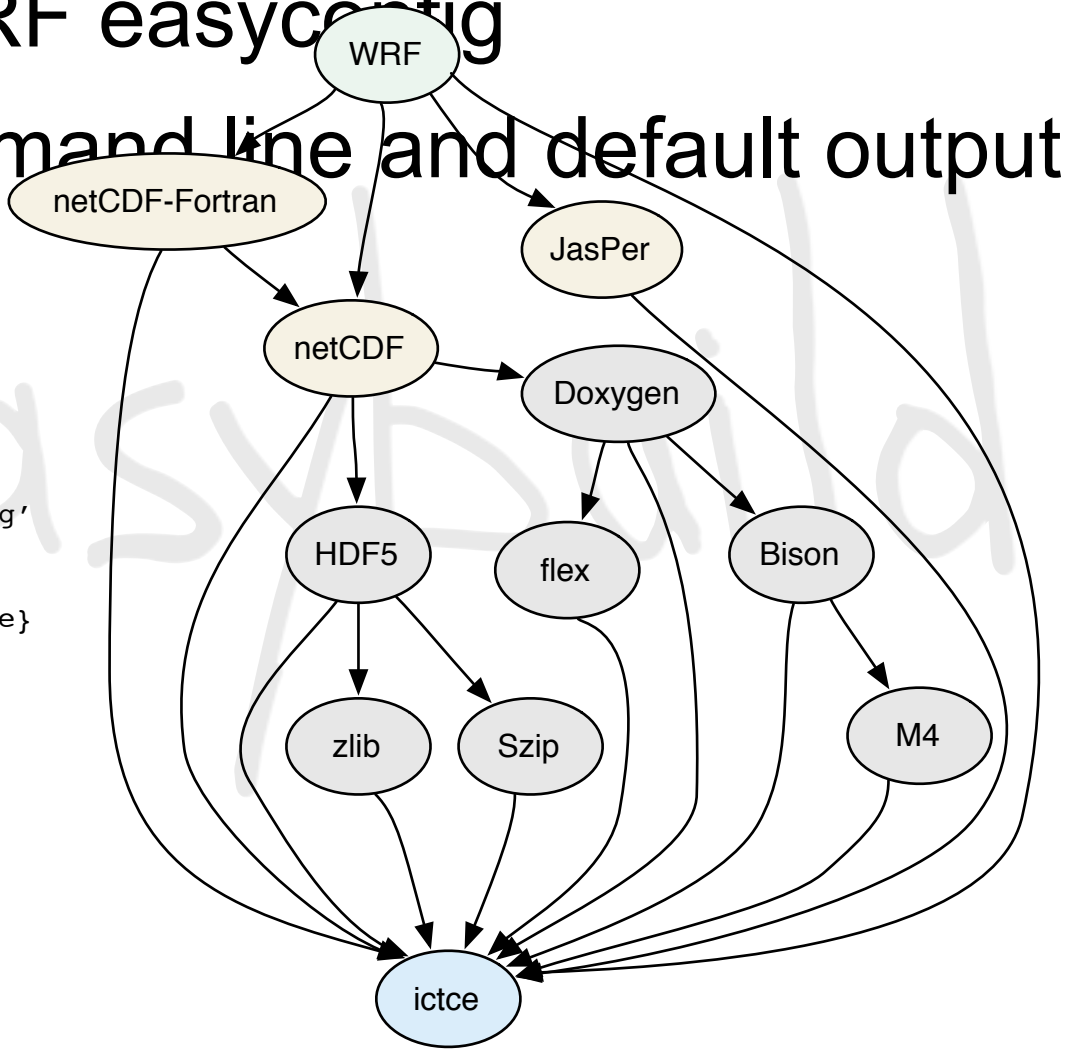
show and explain WRF easyblock

Use case: installing WRF

show and explain WRF easyconfig

show EasyBuild command line and default output

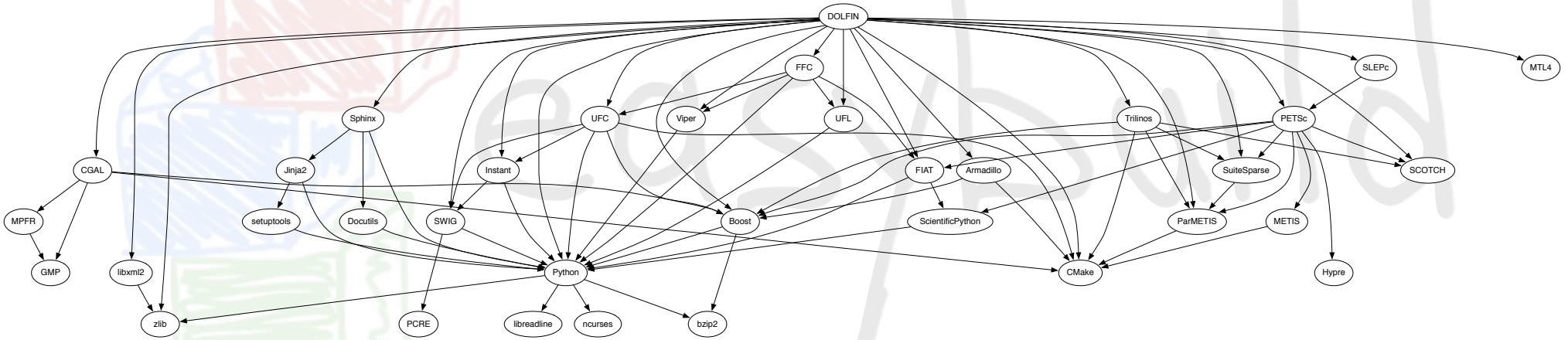
```
name = 'WRF'
version = '3.4'
homepage = 'http://www.wrf-model.org'
description = 'Weather Research and Forecasting'
tcver = '3.2.2.u3'
toolchain = {'name': 'ictce','version': tcver}
toolchainopts = {'opt': False, 'optarch': False}
sources = ['%sV%s.TAR.gz' % (name, version)]
patches = [
    'WRF_parallel_build_fix.patch',
    'WRF-3.4_known_problems.patch',
    'WRF_tests_limit-runtimes.patch',
    'WRF_netCDF-Fortran_separate_path.patch']
dependencies = [('JasPer', '1.900.1'),
                ('netCDF', '4.2'),
                ('netCDF-Fortran', '4.2')]
buildtype = 'dmpar'
```

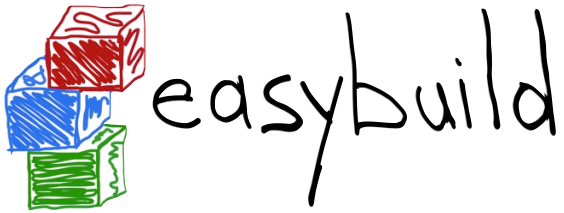


Use case: installing DOLFIN

show DOLFIN dependency graph

show how to install DOLFIN with EasyBuild

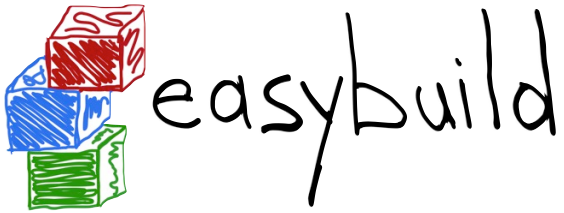




Current status

Developed in-house for over 3.5 years, available on GitHub (GPLv2) since April 2012, v1.0.0 (stable API) just released.

- support for GCC and Intel compilers, ATLAS, Intel MKL, ...
- custom *easyblocks* available for 77 software packages, more being ported
- 338 example easyconfigs for 148 different software packages



Roadmap

say something about future release plans

easybuild

building software with ease

website: *http://hpcugent.github.com/easybuild*

GitHub: *http://github.com/hpcugent/easybuild[-framework|-easyblocks|-easyconfigs]*

PyPi: *http://pypi.python.org/pypi/easybuild[-framework|-easyblocks|-easyconfigs]*

mailing list: *easybuild@lists.ugent.be*

Twitter: *@easy_build*

IRC: *#easybuild @ irc.freenode.net*