# Update on A New Build System for LArSoft

Brett Viren

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

LArSoft Libbers
24 Feb 2015

# Outline

Reminder of Approach

Status

Installation Procedure

Docker

# Principle Instigators

**Patrick Gartung** Application of pure-CMake to LArSoft and `lbnecode`, Jenkins integration, Worch configuration development, Ubuntu and Mac porting. Chasing upstream changes.

**Ben Morgan** Initial pure-CMake package build method and its application to FNALCore and art.

**Brett Viren** Worch and initial `lbne-build` configuration, UPS packaging, release management. `lbne-docker`.

# Overview - Reminder

Goal: allow all LBNE collaborators build and use LArSoft.

Major ingredients:

1. Delete build-time dependency on FNAL s/w infrastructure (UPS, cetbuiltools, cetpkgsupport, UPS products).
2. Add a simple, standard CMake package-level build system to art, LArSoft and related packages.
3. Use Worch for build orchestration and release management

Major features:

- Automated, comprehensive portable build of the experiment's entire software stack.
- Full-stack build-configuration and release management in concise configuration files.
- Policy agnostic approach supports a wide variety of package-level build methods and file system layout.
- Packaging of build output. Currently support **UPS** and **Environment Modules**, others easy to add.

# Plan - Reminder

Detailed written plan exists, just the gist:

- Primary challenge is disentangling the build from FNAL s/w infrastructure.
- Do initial "pure CMake" work in LBNE GitHub to avoid any potential interference with Redmine repository users.
- Port changes back to Redmine in order:
  → `lbnecode` → `larsoft` → `art`
- Initially, keep "pure CMake" code hidden inside a default-off `if`-switch
- Flip switch in Redmine to make "pure CMake" default
  - Interim: we are "chasing" upstream releases in GitHub.
  - A somewhat tedious procedure and requiring ongoing effort better applied to doing other things.

# xLBNx Transition Commentary Pertaining To Builds

An aside and just my view:

- I hope the new xLBNx collaboration will be even stronger in the area of software than LBNE.
  - Potential new contributions from "hard core" s/w types from CERN and other new institutions.
- I expect LArSoft will retain a central role but competing systems should be expected.
  - Alternative frameworks/applications or major changes to LArSoft/art may be proposed.
  - We should welcome and evaluate them and where beneficial look for efficient ways for adopting/adapting or interoperability.
- A portable build system for LArSoft is needed now more than ever.
  - More diverse collaboration means more diverse computing.
  - If a competition develops, lack of a portable build will be a strong factor in any comparison.

Reminder of Approach

Status

Installation Procedure

Docker

# Release Status of first Pure-CMake Release

**FNALCore** "1.13.01" released

- Version in quotes as there are four source packages of different versions aggregated into one via `git-subtree`: `cpp0x`, `cetlib`, `messagefacility`, `fhicl-cpp`
- versions chosen as required by art 1.13.01

**art** version "1.13.01-p0" released. ("p" = "pure" or "patch", counts our downstream releases)

**lbne-build** version 0.7.1 on PyPI released, builds just up to art.

**larsoft** v4 (and lbnecode) will be added soon.

Reminder of Approach

Status

Installation Procedure

Docker

# Try it!

```
$ virtualenv venv
$ source venv/bin/activate
$ pip install lbne-build==0.7.1
$ cp $VIRTUAL_ENV/share/worch/wscripts/lbne/wscript .

$ waf --orch-config=lbne/suite-art.cfg \
      --prefix=install \
        configure
$ waf
```

Reminder of Approach

Status

Installation Procedure

Docker

# What is Docker

- A system for creating and managing **Linux containers**.
  - A Docker *image* is an entire OS minus Linux (the kernel)
  - A running *image* is a *container*.
  - Running a *container* may lead to creating a new *image*.
- Like a **Virtual Machine** but runs on the host's Linux
  - Docker containers are more efficient and require less resource overhead than VMs.
  - Much easier to provision than a VM, scriptable via `Dockerfile`s or programmatic via API.
- Like a **chroot** but with efficient reuse of "layers"
  - Union FS gives a kind of "git for file systems"
  - Idempotent build battery included, each build instruction is a layer, exact replay uses cached results.

Docker allows easy specification and construction of an exact and minimal set of dependencies needed to run some task, such as building/testing LArSoft.

# Why Docker

- LBNE has more platforms than FNAL does.
- LBNE has more platforms that I do.
- I want a "cheap" way to populate Linux-based installations.
- I want a way to precisely characterize these installations.
- Automation and idempotency are keys to efficiency.

Thus is born:

**lbne-docker** = Docker + Worch

`https://github.com/LBNE/lbne-docker/`

# Using lbne-docker

1. Write a brief section of **config file**.
2. Run the `graph-docker.py` script on it.
3. there is no #3.

The configuration file becomes a **single document** which precisely specifies **all** release management configuration information for **all** currently supported releases on **all** currently supported Linux platforms.

Anyone can reproduce any of the release with minimal effort.

Explicitly and precisely captures and tests what is required for native host installation.

# lbne-docker Config File

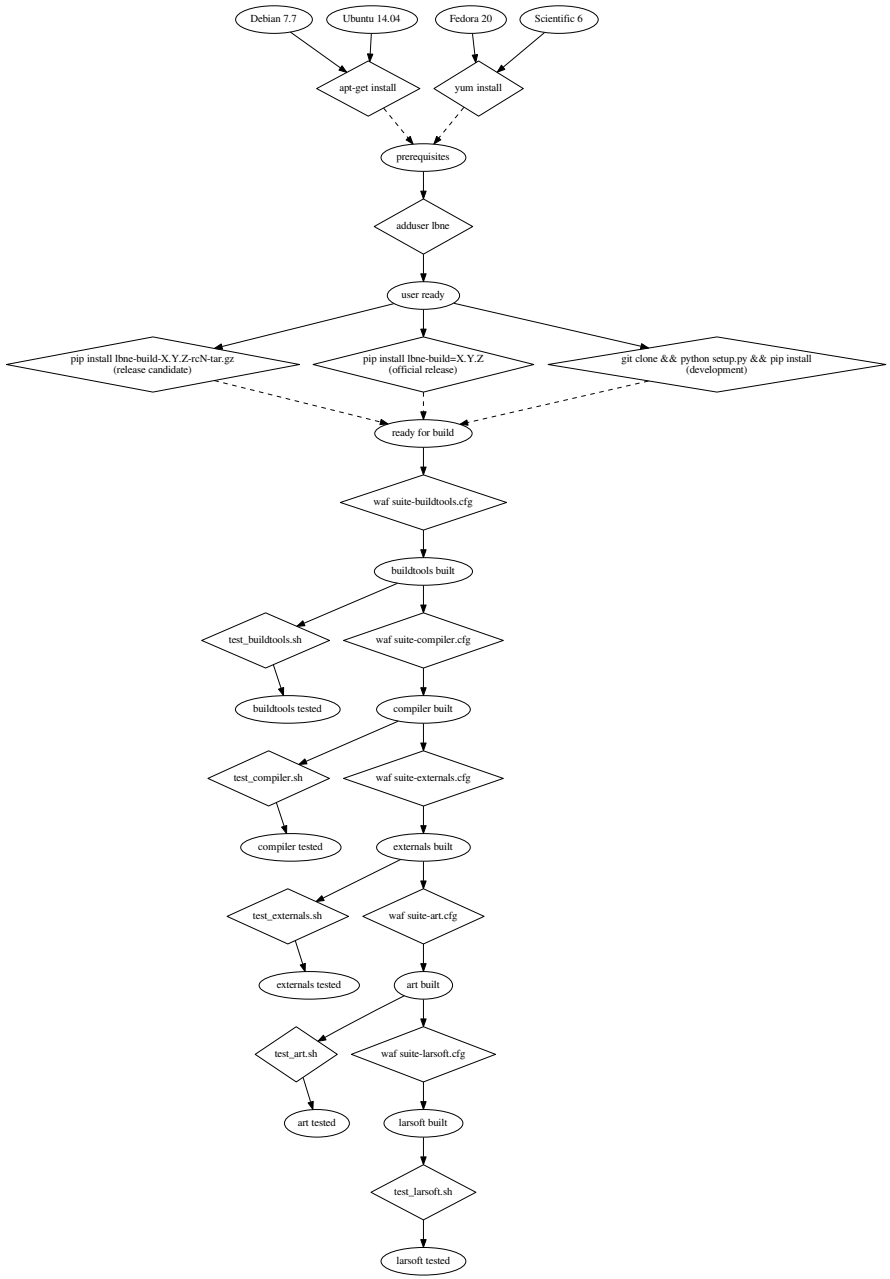
One section per supported platform + LBNE release:

```
[lbne-0.5.2-fedora-21-1]
platform: fedora:21
build: 0.5.2
workflow: prereq, user, release, buildtools compiler,
  externals larsoft
validate: buildtools compiler externals larsoft
packages: wget gettext bzip2 file git subversion
  python-virtualenv tcl-devel
  make gcc-c++ m4 patch automake flex bison bison-devel libtool
  ncurses-devel zlib-devel bzip2-devel
  libX11-devel libXmu-devel libXpm-devel libXft-devel
  mesa-libGL-devel mesa-libGLU-devel
  expat-devel readline-devel krb5-devel
  doxygen-latex gitflow
```

# lbne-docker workflow

1. Starts from specified, minimal Linux distro
2. Install prerequisites
3. Set up `lbne-build` (various ways)
4. Build through the `lbne-build` suites
   - compiler
   - externals
   - art
   - larsoft
   - lbnecode

Each major step has a testing stub which will abort on failure.



# Going further: Docker + Jenkins

I want Jenkins-driven build and test jobs running in Docker containers.

- Vastly wider platform coverage "for free" (no new hardware).
- Precisely characterized tests, reproducible failures.
  - $\rightarrow$ exactly reproducible by experts on their own computer.
- Fast test runs by exploiting idempotency.
- Concise image configuration files well suited to Jenkins management.

Caveats:

- Not all builds/tests must run in Docker containers, (eg, there is no such thing as a Mac OS X Docker container).
- Not advocating Docker images as a means to deliver binaries to end users.