Test Stack and Test Drive Update

Ben Morgan
HSF Packaging Working Group

https://github.com/HSF/packaging

Project tidied, all Test Stack stuff under testdrive
New subprojects are welcome!
One subdirectory per package manager

Two bits of documentation:
- Overarching README
- Template README for TestDrives

Packaging Working Group - Test Drive Area

This directory holds documentation and files for test driving various package manager tools to demonstrate the installation and use a HEP test stack in order to evaluate the various identified use cases. We currently have test drives for:

- nix
- portage
- spack

See the READMEs in each directories for more details on the tool and instructions for taking it for a test drive. If you want to create a test drive for a new package manager, please see the information below.
Minimum Test Stack requirements
- Each Test Drive can use newer (but not older!) versions if required
- Can discuss updating this core set as time progresses
HEP Stack Test Drive Program

To enable to WG (and eventually the broader HEP community) to evaluate how well a given package manager solves the identified use cases as well as general considerations such as portability and ease of use, a basic list of tasks is outlined to provide a template for "driving lessons" on:

- Installing the package manager
- Installing the first package
- Installing the HEP Test Stack
- Using installed packages
- Adding new packages
- Developing software against installed packages

To keep things simple some simplifying assumptions are made:

- **Docker** images are used for testing Linux platforms.
  
  This is purely for consistency and reproducibility, and does not suggest that containers will be the only way to use a given package manager! It also helps to enumerate the OS packages that are always needed by the package manager.

- No use of **CVMFS** is assumed yet.
  
  This is to ensure that test drivers get a feel for building from source, installing from binary, writing packages for their own software, and the balance between reuse of OS packages vs “compile the world”.

  It does not imply that CVMFS will not be used later, but users will have to go through the packaging steps to have something to deploy to CVMFS! Smaller experiments may also not have access to CVMFS.

- The test driver may have **sudo** access, but the steps requiring this should be minimized and ideally zero.

Creating a new Test Drive Program

1. Create a new subdirectory under this one named for the package manager you are adding.
2. Copy the **README template** into the new directory and fill in the details noted by the **Authors : add details here...**
Test Driving the Foo Package Manager

**AUTHORS**: Change “Foo” here and elsewhere in the template to the name of the package manager

This document will walk you through preparing and using the Foo (**AUTHORS**: Provide a link to the tool’s home page) package manager to install a basic software stack for HEP.

Foo is ...

**AUTHORS**: Provide a brief description of Foo and what is good about it, defer to the tool’s own docs if preferred!

Base Operating System Install

**AUTHORS**: Assume a base system of macOS High Sierra with Xcode 9, or a Docker image based on centos:centos7 or ubuntu:xenial. In the Docker case, supply (a) Dockerfile(s) for each system, adding any additional system packages required, and (b) an hsf user with sudo privileges that the container will run as. There’s no requirement to build and host the images. Bonus points: Singularity! If macOS needs additional packages, document them below:

Test driving Foo requires either a CentOS7, Ubuntu 16.04LTS, or macOS High Sierra system. For macOS, only the base system plus Xcode 9 from the App Store (**AUTHORS**: add any additional requirements here) is required. For convenience and reproducibility, Docker images are available for Linux, and can be obtained and run as follows:

**AUTHORS**: Add Docker pull/build/run instructions here

Optionally, you may use an existing Linux installation, but you may encounter errors in subsequent steps if it is missing (or has incompatible) packages, or your environment has custom settings.
Next Steps

• Continue filling out READMEs, editing Dockerfiles for each PM

• Can do this via PRs, or add interested parties as collaborators on repo.

• Also: don’t hesitate to use repo to add notes, info under “documentation”, or to propose new packaging related subprojects (e.g. see PR#5 on Relocation!)