Installing LSST Data Management Software Stack, Winter 2012 Release

Prerequisites

Install these using your distribution's package manager:

OS	**Packages**	Notes
RHEL 6 and derivatives	gcc-c++ gcc-gfortran flex bison libXt-devel ncurses-devel readline-devel libuuid-devel zlib-devel bzip2-devel perl make	Officially supported platform
RHEL 5 and derivatives	which perl make gcc-c++ flex bison libX11-devel readline-devel zlib-devel gcc44-c++ gcc44-gfortran e2fsprogs-devel bzip2-devel libXt-devel	Requires extra step
Ubuntu 10.04	**Minimum:** curl flex bison make perl zlib1g-dev libbz2-dev libreadline-dev libncurses5-dev libxt-dev g++ gfortran uuid-dev **Development:** g++ gfortran git autoconf automake libtool m4 make flex bison libx11-dev libncurses5-dev libreadline5-dev patch libuuid1 uuid-dev latex2html libxaw7-dev graphviz zlib1g-dev libbz2-dev	Known to work
Debian 6 ("Squeeze")	Exact dependencies unknown but likely similar to those for Ubuntu	Known to work (needs >= 8G RAM to compile)
Mac OS X 10.7 (Lion)	?XCode 4.3, Command-line tools (use Download prefs pane in Xcode), ?gfortran for XCode 4.3	Requires the use of clang and special instructions
Mac OS X 10.x, (x<7)		Does not work (compiler too old)

Installing a New Stack

For the impatient

RHEL 6 (or derivatives)

```
sh users:
oot/directory/where/lsst/stack/will/be/installed (e.g., ~/lsst)

t LSST_HOME EUPS_PATH LSST_DEVEL

rt NCORES=$((sysctl -n hw.ncpu || (test -r /proc/cpuinfo && grep processor /proc/cpuinfo | wc -l) || echo 2) 2>
rt MAKEFLAGS="-j $NCORES"
rt SCONSFLAGS="-j $NCORES"

 http://dev.lsstcorp.org/pkgs/std/w12/newinstall.sh -o newinstall.sh
 newinstall.sh

ce loadLSST.sh
 distrib install --nolocks -t Winter2012 lsstactive
```

RHEL 5 (or derivatives)

```
sh users:
oot/directory/where/lsst/stack/will/be/installed (e.g., ~/lsst)

t LSST_HOME EUPS_PATH LSST_DEVEL

rt NCORES=$((sysctl -n hw.ncpu || (test -r /proc/cpuinfo && grep processor /proc/cpuinfo | wc -l) || echo 2) 2>
rt MAKEFLAGS="-j $NCORES"
rt SCONSFLAGS="-j $NCORES"

 http://dev.lsstcorp.org/pkgs/std/w12/newinstall.sh -o newinstall.sh
 newinstall.sh
ce loadLSST.sh

 distrib install -r http://nebel.rc.fas.harvard.edu/mjuric/std/w12 rhel5_gcc44 4.4
p rhel5_gcc44

 distrib install --nolocks -t Winter2012 lsstactive
```

Mac OS X 10.7 (Lion) and Linux with clang

If you're running on OS X, this may be needed:

```
# make a symlink for gfortran:
sudo ln -s /usr/bin/gfortran-4.2 /usr/bin/gfortran
```

```
sh users:
oot/directory/where/lsst/stack/will/be/installed (e.g., ~/lsst)

t LSST_HOME EUPS_PATH LSST_DEVEL

rt CC=clang
rt CXX=clang++

rt NCORES=$((sysctl -n hw.ncpu || (test -r /proc/cpuinfo && grep processor /proc/cpuinfo | wc -l) || echo 2) 2>
rt MAKEFLAGS="-j $NCORES"
rt SCONSFLAGS="-j $NCORES cc=clang"

 -O http://nebel.rc.fas.harvard.edu/mjuric/pkgs/newinstall.sh
 newinstall.sh
ce loadLSST.sh

 distrib install --nolocks -t Winter2012 lsstactive
```

Step by step instructions (for Linux)

Bootstrapping the environment

If you have already installed the software stack, be sure unset your environment variables for that stack. To do this, you can type (after the % prompt) one of the following:

```
% unset LSST_HOME EUPS_PATH      # for bash users
% unsetenv LSST_HOME EUPS_PATH   # for tcsh users
```

Create and change into the directory where LSST DM stack is to be installed (the "LSST home"):

```
% mkdir -p /the/LSST/installation/root && cd /the/LSST/installation/root
```

The home of the LSST installation is your choice. Just don't use the same location as a previous LSST installation unless you move it out of the way first.

Installing the stack involves downloading and building a number of (sizable) source packages. If you have a multi-core machine, you speed up the builds significantly by allowing the LSST installer to use all cores.

bash users:

```
ad out the number of CPUs to speed up builds via SCONSFLAGS and MAKEFLAGS (works on Linux and Mac)
rt NCORES=$((sysctl -n hw.ncpu || (test -r /proc/cpuinfo && grep processor /proc/cpuinfo | wc -l) || echo 2) 2>
rt  MAKEFLAGS="-j $NCORES"
rt SCONSFLAGS="-j $NCORES"
```

csh users:

```
 the number of CPUs to speed up builds via SCONSFLAGS and MAKEFLAGS (works on Linux and Mac)
RES `bash -c "(sysctl -n hw.ncpu || ( test -r /proc/cpuinfo && grep processor /proc/cpuinfo | wc -l ) || echo 2
KEFLAGS "-j $NCORES"
NSFLAGS "-j $NCORES"
```

Download and run the installation setup script:

```
% curl -O http://dev.lsstcorp.org/pkgs/std/w12/newinstall.sh
% bash newinstall.sh
```

This installs the basic packages required to install other packages. It also sets up the `loadLSST.*` scripts which you should source:

```
% source loadLSST.sh # for bash users
% source loadLSST.sh # for csh users
```

to get LSST tools (e.g., the `eups` command) added to your path.

RHEL 5 specific step

If you're using RHEL 5 (or derivative), your default compiler version is too old too compile LSST DM code. You will need to install the gcc44 RPM packages, and a special package to make it known to EUPS:

```
eups distrib install -r http://nebel.rc.fas.harvard.edu/mjuric/std/w12 rhel5_gcc44 4.4
setup rhel5_gcc44
```

Warning: This package will set environment variables LAPACK=None, ATLAS=None and BLAS=None. These are used by the `numpy` installer to determine whether to look for and build with external LAPACK, blas and/or atlas libraries. If you have compiled either of these (with gcc44 compiler), set these variables to the directories where they reside.

Installing the stack

To install the Winter2012 release of the DM stack, type:

```
% eups distrib install --nolocks lsstactive -t Winter2012
```

Running a Demo

We provide a simple demonstration of using the LSST DM stack to detect sources in a simulated LSST image (a single chip):

```
url -O https://dev.lsstcorp.org/cgit/contrib/demos/lsst_dm_stack_demo.git/snapshot/lsst_dm_stack_demo-Winter201
ar xzf lsst_dm_stack_demo-Winter2012.tar.gz
d lsst_dm_stack_demo-Winter2012

/bin/demo.sh
```

(note: this is a ~210MB download). Look into the `README` file for more information.

On the NCSA lsst* machines this repository is available as /lsst3/lsst_dm_stack_demo-Winter2012

Known Issues

Building the stack takes a long time

The current build system defaults to using a single core when building the stack. This behavior can be overridden using the MAKEFLAGS and SCONSFLAGS environment variables:

```
 ASSUMES YOU ARE IN THE DIRECTORY WHERE YOU WISH TO INSTALL THE STACK


 number of CPUs to speed up builds via SCONSFLAGS and MAKEFLAGS (this works on Linux and Mac)
${NCORES:-$(sysctl -n hw.ncpu 2>/dev/null || (test -r /proc/cpuinfo && grep processor /proc/cpuinfo | wc -l) 2>

asic environment to ~/lsst
v.lsstcorp.org/pkgs/std/w12/newinstall.sh -o newinstall.sh
$NCORES" bash newinstall.sh

er2012 stack
ometry_net and wcslib separately because of -jN bugs (e.g. ticket/1970 and related)
T.sh
$NCORES" SCONSFLAGS="-j $NCORES" eups distrib install --nolocks -t Winter2012 lsstactive
```

EUPS Locking

Some users have encountered difficulties with the more aggressive locking in the new EUPS. If a stack will be used by multiple users, you should at least modify the EUPS lock directory to point to a location writable by all users. This can be accomplished by adding the following to `$LSST_HOME/site/startup.py`:

```
hooks.config.site.lockDirectoryBase = "/tmp"
```

If you still encounter problems, you can disable locking entirely by instead using:

```
hooks.config.site.lockDirectoryBase = None
```

Numpy may fail to build

When you are installing "lsstactive" the installation will sometimes fail on numpy. The workaround is to install numpy, then resume your installation of lsstactive. For example:

- eups distrib install numpy -t Winter2012
- eups distrib install lsstactive -t Winter2012

The underlying problem is that LD_LIBRARY_PATH is set as part of installing lsstactive, and this can confuse the numpy installer.

Developing for the LSST DM Stack

Basics

LSST DM software stacks consists of a number of packages, managed by the EUPS tool. EUPS is similar to ?environment modules that you may have encountered on Beowulf clusters. It allows the user to load and mix and match the desired packages, by manipulating environment variables such as `PATH`, `LD_LIBRARY_PATH`, `PYTHONPATH`, etc. Eups also knows about which packages depend on others; for example loading (or 'setting up', in EUPS speak) the top-level package `pipe_tasks` will automatically load packages on which `pipe_tasks` depends:

```
a ~]$ setup pipe_tasks       # setup package pipe_tasks (thus making available commands such as processCcdLsstS

a ~]$ eups list -s           # see which packages were set up -- most of these have been pulled in as dependenc
           4.8.0.2+1         current Winter2012 setup
net        0.30              current stable setup
           4.8.0.0+1         current beta Winter2012 setup
           1.47.0+5          current beta Winter2012 setup
           3290+1            current beta Winter2012 setup
uared      4.7.3.0+9         current Winter2012 setup
           4.8.1.1+1         current Winter2012 setup
```

EUPS also lets you override default packages with your own versions. Example:

```
[mjuric@moya ~]$ git clone git@git.lsstcorp.org:LSST/DMS/afw.git
Initialized empty Git repository in /home/mjuric/afw/.git/
remote: Counting objects: 33716, done.
remote: Compressing objects: 100% (10542/10542), done.
remote: Total 33716 (delta 22288), reused 29944 (delta 19804)
Receiving objects: 100% (33716/33716), 30.93 MiB | 7.12 MiB/s, done.
Resolving deltas: 100% (22288/22288), done.
[mjuric@moya ~]$ cd afw/
[mjuric@moya afw]$ git checkout Winter2012/Release
Branch Winter2012/Release set up to track remote branch Winter2012/Release from origin.
Switched to a new branch 'Winter2012/Release'

[mjuric@moya afw]$ setup -j -r .

[mjuric@moya afw]$ scons -j 16 opt=3 -s
Setting up environment to build package 'afw'.
Warning: afwdata is not set up; not running the tests!
....
```

The above will make the cloned afw the 'setup'-ed one; that is, other package looking for afw will find your locally built copy, instead of the system one.

Note: There's more in this older (and out of date) document: ?http://dev.lsstcorp.org/GettingStarted.html.

Installing packages to a personal directory

In shared environments, where a system-wide, read-only stack exists, it is useful to have the ability to install one's own packages to a personal directory. This is were `mksandbox` command helps:

```
[mjuric@moya ~]$ mksandbox mystack
```

This will create a subdirectory 'mystack', and create some EUPS-related files in it (for bookkeeping purposes). It needs to be done only once.

To let EUPS know about the directory, do:

```
[mjuric@moya ~]$ export LSST_DEVEL="$PWD/mystack"
[mjuric@moya ~]$ source $LSST_HOME/loadLSST.sh
```

Now you can 'eups distrib install' new packages into your "personal stack".

For more information, see ?here.

Enabling Builds with GPU Acceleration

GPU-aware packages (currently, only `afw`) look for the cuda_toolkit EUPS package to locate the NVIDIA CUDA compilers. This package must be installed explicitly. For example:

```
export CUDA=/usr/local/cuda
eups distrib install -r http://nebel.rc.fas.harvard.edu/mjuric/std/w12 cuda_toolkit 4.1
setup cuda_toolkit 4.1
```

will install and setup cuda_toolkit EUPS support package for CUDA 4.1 toolkit that resides in `/usr/local/cuda`. Note: the cuda_toolkit package will **not** install NVIDIA's compilers -- you already must have those installed. It will only set up the necessary environment variables and symlinks for the other packages to know how to find CUDA.

After setting up `cuda_toolkit`, any future `afw` builds will be GPU-enabled.

Developing on moya.dev.lsstcorp.org

The public stack on moya is located in

```
/home/lsst/Winter2012
```

. As the path suggests, it's the Winter2012 release.

Developing at NCSA

The public stack on the LSST cluster at NCSA is available at

```
/lsst/DC3/stacks/RH6
```

. As the path suggests, it will only work on the machines running Red Hat Enterprise Linux (RHEL) 6.

Building with clang

At NCSA, there's a build of clang in `~juric/clang/3.0`. Add its `bin` subdirectory to PATH to enable it. Otherwise, build clang using these instructions.

Then follow the instructions for [wiki/Installing/Winter2012#MacOSX10.7LionandLinuxwithclang installing on OS X 10.7].

sconsUtils

Documentation for the `sconsUtils` can be found in several locations:

- A tutorial for converting from the old sconsUtils to the new can be found at wiki:Winter2012/sconsUtilsDocumentation.
- A documented template of the expected package can be found at ?devenv/templates.
- The definitive location for sconsUtils reference documentation is in its own Doxygen build; this has been temporarily hosted at ?http://www.astro.princeton.edu/~jbosch/sconsUtils-doxygen until sconsUtils is included with the buildbot Doxygen runs.

NOTE: sconsUtils generates a `version.py` file for every package that carries its version and the versions of the dependencies it was built against. This is imported by the `__init__.py` of each package, causing an import error if it does not yet exist. **This means that even pure-Python packages must now be built with scons before they can be used.**

git

The latest (>= 4.7.1.0) versions of the "lsst" package set the `LSST_GIT` and `LSST_DMS` environment variables to point to `git@git.lsstcorp.org:LSST` and `git@git.lsstcorp.org:LSST/DMS`, so these can be used in largely the same way as the old `LSST_SVN` and `LSST_DMS` variables.

To clone the git repo for an LSST package, do:

```
git clone $LSST_DMS/my_package
```

Installing without the newinstall.sh script and lsst package

These instructions are for those who would like to use an existing EUPS install, and avoid the "lsst" package and its associated scripts (most likely people at Princeton who are developing in both LSST and HSC environments).

- Make sure you have the latest `lssteups` (see the next section).

- Make sure you have `eups >= 1.2.23`.
- Set your `EUPS_PATH` to whatever you like.
- Set your `EUPS_PKGROOT` to include `http://dev.lsstcorp.org/pkgs/std/w12`.
- Remove or otherwise disable old `scons` and `sconsDistrib` packages; the new `scons` is the equivalent of the old `sconsDistrib`, and that can lead to some confusion. Or don't, but come back to this step if you have problems down the road.
- Be aware that things in your `manifest.remap` might break the install (or they might do what you want them to do, but it's another place to look if things go awry).
- Install the new `sconsUtils` (which should install Python, Tcl/Tk?, scons, and Doxygen as dependencies) with `eups distrib install sconsUtils`.
- Realize that you won't get the `LSST_GIT` and `LSST_DMS` environment variables noted above unless you set them yourself.
- Install away!