Walk-thru- Build Lustre MASTER on RHEL 7.3/CentOS 7.3 from Git

This walk-thru is targeting developers who want to explore the bleeding edge of Lustre. If you are evaluating Lustre for production, you should choose a Lustre Release.

Purpose

Describe the steps you need to build and test a Lustre system (MGS, MDT, MDS, OSS, OST, client) from the **`master`** branch on a x86_64, RHEL/CentOS 7.3 machine.

Prerequisite

- A newly installed RHEL/CentOS 7.3 x86_64 machine connected to the internet.
- EPEL Repository: this is a convenient source for git.
- **NOTE** It is suggested that you have at least 1GB of memory on the machine you are using for the build.
- **NOTE** Verify that SElinux is disabled.

Overview

pre-built RPMs are available

Lustre servers **no longer** require a patched and compiled kernel. If desired, a patched and compiled Lustre server kernel is available from Whamcloud. A separate page is available to walk thru setting up Lustre with these pre-built RPMs. This document is for those who wish to build their Lustre system from source. Note that if you are not modifying the kernel patches on the server, it is possible to use the pre-built Lustre server kernel RPMs, and only build the Lustre code. Note that a patched kernel is **NOT** needed for the Lustre client.

Patches are available in the Git source repository. A test suite is included with the Lustre source. This document walks through the steps of patching the kernel, building Lustre and running a basic test of the complete system.

Procedure

The procedure requires that a OS is setup for development - this includes Lustre sources, kernel source and build tools. Once setup, a new kernel can be patched, compiled, run and tested. Further reading on building a RHEL RPM based kernel is available from, among other sources, the CentOS site.

Provision machine and installing dependencies.

Once RHEL 7.3 is newly installed on `rhel6-master` login as user `root`.

1. Install the kernel development tools:

```
# yum -y groupinstall "Development Tools"
```

Problem with installing 'Development Tools'

If the Development Tools group is not be available for some reason, you may find the following list if individual packages necessary to install.

```
# yum -y install automake xmlto asciidoc elfutils-libelf-devel zlib-devel binutils-devel newt-devel python-devel hmaccalc perl-ExtUtils-Embed rpm-build make gcc redhat-rpm-config patchutils git
```

2. Install additional dependencies:

```
# yum -y install xmlto asciidoc elfutils-libelf-devel zlib-devel binutils-devel newt-devel python-devel hmaccalc perl-ExtUtils-Embed bison elfutils-devel audit-libs-devel
```

3. Install EPEL 7:

```
# rpm -ivh http://download.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-9.noarch.rpm
```

4. Install additional packages:

```
# yum -y install pesign numactl-devel pciutils-devel ncurses-devel libselinux-devel
```

Preparing the Lustre source.

1. Create a user `build` with the home directory `/home/build`:

```
# useradd -m build
```

2. Switch to the user `build` and change to the build `$HOME` directory:

```
# su build
$ cd $HOME
```

3. Get the **`master`** branch from git:

```
$ git clone git://git.whamcloud.com/fs/lustre-release.git
$ cd lustre-release
```

4. Run `sh ./autogen.sh`
5. Resolve any outstanding dependencies until `autogen.sh` completes successfully. Success will look like:

```
$ sh ./autogen.sh
configure.ac:10: installing 'config/config.guess'
configure.ac:10: installing 'config/config.sub'
configure.ac:12: installing 'config/install-sh'
configure.ac:12: installing 'config/missing'
libcfs/libcfs/autoMakefile.am: installing 'config/depcomp'
$
```

Prepare a patched kernel for Lustre

You can have different ways to prepare a patched kernel for Lustre. The easier method is to download built RPM packages from the Releases page. You're going to need the packages starting with '`kernel-`'. After new kernel packages are downloaded, you can skip the following few steps and go to the section 'Installing the Lustre kernel and rebooting'.

If you want a more challenge life, you can patch the kernel by yourself, in that case, please follow the steps below.

Prepare the kernel source

In this walk-thru, the kernel is built using `rpmbuild` - a tool specific to RPM based distributions.

1. Get the kernel source. First create the directory structure, then get the source from the RPM. Create a `.rpmmacros` file to install the kernel source in our user directory:

```
$ cd $HOME
$ mkdir -p kernel/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
$ cd kernel
$ echo '%_topdir %(echo $HOME)/kernel/rpmbuild' > ~/.rpmmacros
```

2. Install the kernel source:

```
$ rpm -ivh http://vault.centos.org/7.3.1611/updates/Source/SPackages/kernel-3.10.0-514.2.2.el7.src.rpm
```

3. Prepare the source using `rpmbuild`:

```
$ cd ~/kernel/rpmbuild
$ rpmbuild -bp --target=`uname -m` ./SPECS/kernel.spec
```

   This will end with:

```
...
+ make ARCH=x86_64 oldnoconfig
scripts/kconfig/conf --olddefconfig Kconfig
#
# configuration written to .config
#
+ echo '# x86_64'
+ cat .config
+ find . '(' -name '*.orig' -o -name '*~' ')' -exec rm -f '{}' ';'
+ find . -name .gitignore -exec rm -f '{}' ';'
+ cd ..
+ exit 0
```

At this point, we now have kernel source, with all the RHEL/CentOS patches applied, residing in the directory `~/kernel/rpmbuild/BUILD/kernel-3.10.0-514.2.2.el7/linux-3.10.0-514.2.2.el7.x86_64/`

Patch the kernel source with the Lustre code.

1. Gather all the patches from `lustre` tree into a single file:

```
$ cd ~
$ rm -f ~/lustre-kernel-x86_64-lustre.patch
$ cd ~/lustre-release/lustre/kernel_patches/series
$ for patch in $(<"3.10-rhel7.series"); do \
      patch_file="$HOME/lustre-release/lustre/kernel_patches/patches/${patch}"; \
      cat "${patch_file}" >> "$HOME/lustre-kernel-x86_64-lustre.patch"; \
  done
$
```

2. Copy the kernel patch into RPM build tree:

```
# cp ~/lustre-kernel-x86_64-lustre.patch ~/kernel/rpmbuild/SOURCES/patch-3.10.0-lustre.patch
```

3. Edit the kernel spec file `~/kernel/rpmbuild/SPECS/kernel.spec`:

   Find the line with '`find $RPM_BUILD_ROOT/lib/modules/$KernelVer`' and insert following two lines below it

   ```
   cp -a fs/ext3/* $RPM_BUILD_ROOT/lib/modules/$KernelVer/build/fs/ext3
   ```

   ```
   cp -a fs/ext4/* $RPM_BUILD_ROOT/lib/modules/$KernelVer/build/fs/ext4
   ```

   Find the line with '`# empty final patch to facilitate testing of kernel patches`' and insert following two lines below it

   ```
   # adds Lustre patches
   ```

   ```
   Patch99995: patch-%{version}-lustre.patch
   ```

   Find the line with '`ApplyOptionalPatch linux-kernel-test.patch`' and insert following two lines below it

   ```
   # lustre patch
   ```

   ```
   ApplyOptionalPatch patch-%{version}-lustre.patch
   ```

   Find the line with '`%define listnewconfig_fail 1`' and change 1 to 0

   Save and close the spec file.

4. Overwrite the kernel `config` file with `~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-3.10.0-3.10-rhel7-x86_64.config`:

```
echo '# x86_64' > ~/kernel/rpmbuild/SOURCES/kernel-3.10.0-x86_64.config
cat ~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-3.10.0-3.10-rhel7-x86_64.config >> ~
/kernel/rpmbuild/SOURCES/kernel-3.10.0-x86_64.config
```

Build the new kernel as an RPM.

1. Start building the kernel with `rpmbuild`:

```
$ cd ~/kernel/rpmbuild
$ buildid="_lustre" # Note: change to any string that identify your work
$ rpmbuild -ba --with firmware --target x86_64 --with baseonly \
           --define "buildid ${buildid}" \
           ~/kernel/rpmbuild/SPECS/kernel.spec
```

2. A successful build will return:

```
...
...
Wrote: /mnt/home/build/kernel/rpmbuild/SRPMS/kernel-3.10.0-514.2.2.el7_lustre.src.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-headers-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-debuginfo-common-x86_64-3.10.0-514.2.2.
el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/perf-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/perf-debuginfo-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/python-perf-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/python-perf-debuginfo-3.10.0-514.2.2.el7_lustre.
x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-tools-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-tools-libs-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-tools-libs-devel-3.10.0-514.2.2.el7_lustre.
x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-tools-debuginfo-3.10.0-514.2.2.el7_lustre.
x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-devel-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Wrote: /mnt/home/build/kernel/rpmbuild/RPMS/x86_64/kernel-debuginfo-3.10.0-514.2.2.el7_lustre.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.F7X9cL
+ umask 022
+ cd /mnt/home//build/kernel/rpmbuild/BUILD
+ cd kernel-3.10.0-514.2.2.el7
+ rm -rf /mnt/home/build/kernel/rpmbuild/BUILDROOT/kernel-3.10.0-514.2.2.el7_lustre.x86_64
+ exit 0
```

If you receive a request to generate more entropy, you need to trigger some disk I/O or keyboard I/O. In another terminal, you can either type randomly or execute the following command to generate entropy:

```
# grep -Ri 'entropy' /usr
```

At this point, you should have a fresh kernel RPM `~/kernel/rpmbuild/RPMS/x86_64/kernel-[devel-]3.10.0-514.2.2.el7_lustre.x86_64.rpm`.

Installing the Lustre kernel and rebooting.

1. As **root**, Install the `kernel` and `kernel-devel` packages:

```
# rpm -ivh $PKG_PATH/kernel-3.10.0-514.2.2.el7_lustre.x86_64.rpm $PKG_PATH/kernel-devel-3.10.0-514.2.2.
el7_lustre.x86_64.rpm
```

Depending on how you got your kernel packages, the **PKG_PATH** should be `~build/kernel/rpmbuild/RPMS/x86_64` if you built the packages by yourself, or any other directory where you downloaded the pre-built packages from.

2. Reboot the system.
3. Login system after reboot:

```
# uname -r
3.10.0-514.2.2.el7_lustre.x86_64
```

Now you are running a Lustre patched kernel!

Configure and build Lustre

1. Configure Lustre source:

```
$ cd ~/lustre-release/
$ ./configure
...
...
CC:            gcc
LD:            /bin/ld -m elf_x86_64
CPPFLAGS:      -include /mnt/home/build/lustre-release/undef.h -include /mnt/home/build/lustre-release
/config.h -I/mnt/home/build/lustre-release/libcfs/include -I/mnt/home/build/lustre-release/lnet/include -
I/mnt/home/build/lustre-release/lustre/include
CFLAGS:        -g -O2 -Wall -Werror
EXTRA_KCFLAGS: -include /mnt/home/build/lustre-release/undef.h -include /mnt/home/build/lustre-release
/config.h  -g -I/mnt/home/build/lustre-release/libcfs/include -I/mnt/home/build/lustre-release/lnet
/include -I/mnt/home/build/lustre-release/lustre/include

Type 'make' to build Lustre.
```

2. Make RPMs:

```
$ make rpms
...
...
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/lustre-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/kmod-lustre-2.9.51_35_ge240fb5-1.el7.centos.
x86_64.rpm
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/kmod-lustre-osd-ldiskfs-2.9.51_35_ge240fb5-1.el7.
centos.x86_64.rpm
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/lustre-osd-ldiskfs-mount-2.9.51_35_ge240fb5-1.el7.
centos.x86_64.rpm
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/lustre-tests-2.9.51_35_ge240fb5-1.el7.centos.
x86_64.rpm
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/kmod-lustre-tests-2.9.51_35_ge240fb5-1.el7.centos.
x86_64.rpm
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/lustre-iokit-2.9.51_35_ge240fb5-1.el7.centos.
x86_64.rpm
Wrote: /tmp/rpmbuild-lustre-build-JZiW94sq/RPMS/x86_64/lustre-debuginfo-2.9.51_35_ge240fb5-1.el7.centos.
x86_64.rpm
Executing(%clean): /bin/sh -e /tmp/rpmbuild-lustre-build-JZiW94sq/TMP/rpm-tmp.SxgoFt
+ umask 022
+ cd /tmp/rpmbuild-lustre-build-JZiW94sq/BUILD
+ cd lustre-2.9.51_35_ge240fb5
+ rm -rf /tmp/rpmbuild-lustre-build-JZiW94sq/BUILDROOT/lustre-2.9.51_35_ge240fb5-1.x86_64
+ rm -rf /tmp/rpmbuild-lustre-build-JZiW94sq/TMP/kmp
+ exit 0
Executing(--clean): /bin/sh -e /tmp/rpmbuild-lustre-build-JZiW94sq/TMP/rpm-tmp.vYmwdb
+ umask 022
+ cd /tmp/rpmbuild-lustre-build-JZiW94sq/BUILD
+ rm -rf lustre-2.9.51_35_ge240fb5
+ exit 0
```

3. You should now have build the following, similarly named, RPMs:

```
$ ls *.rpm
kmod-lustre-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
kmod-lustre-osd-ldiskfs-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
kmod-lustre-tests-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
lustre-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
lustre-2.9.51_35_ge240fb5-1.src.rpm
lustre-debuginfo-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
lustre-iokit-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
lustre-osd-ldiskfs-mount-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
lustre-tests-2.9.51_35_ge240fb5-1.el7.centos.x86_64.rpm
```

Installing e2fsprogs

`e2fsprogs` is needed to run the test suite.

1. Download the `e2fsprogs` packages from https://downloads.whamcloud.com/public/e2fsprogs/latest/el7/RPMS/x86_64/ and install `e2fsprogs`, `e2fsprogs-libs`, `libcom_err`, `libss`
2. Or better to use `yum`:

```
# cat <<EOF > /etc/yum.repos.d/e2fsprogs.repo
[e2fsprogs-el7-x86_64]
name=e2fsprogs-el7-x86_64
baseurl=https://downloads.whamcloud.com/public/e2fsprogs/latest/el7/
enabled=1
priority=1
EOF

# yum update e2fsprogs
```

Installing Lustre.

Change to `root` and change directory into `~build/lustre-release/`:

```
# yum localinstall *.x86_64.rpm
```

Disable SELinux (Lustre Servers)

SELinux, which is on by default in RHEL/CentOS, will prevent the format commands for the various Lustre targets from completing. Therefore you must either disable it or adjust the settings. These instructions explain how to disable it.

1. Run `getenforce` to see if SELinux is enabled. It should return '`Enforcing`' or '`Disabled`'.
2. To disable it, edit `/etc/selinux/config` and change the line '`selinux=enforcing`' to '`selinux=disabled`'.
3. Finally, reboot your system.

```
# vi /etc/selinux/config

----
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
# targeted - Only targeted network daemons are protected.
# strict - Full SELinux protection.
SELINUXTYPE=targeted
---
# shutdown -r now
```

Testing

1. run `/usr/lib64/lustre/tests/llmount.sh`:

```
# /usr/lib64/lustre/tests/llmount.sh
Stopping clients: onyx-21vm8.onyx.whamcloud.com /mnt/lustre (opts:)
Stopping clients: onyx-21vm8.onyx.whamcloud.com /mnt/lustre2 (opts:)
Loading modules from /usr/lib64/lustre/tests/..
detected 1 online CPUs by sysfs
libcfs will create CPU partition based on online CPUs
debug=vfstrace rpctrace dlmtrace neterror ha config                  ioctl super lfsck
subsystem_debug=all
gss/krb5 is not supported
Formatting mgs, mds, osts
Format mds1: /tmp/lustre-mdt1
Format ost1: /tmp/lustre-ost1
Format ost2: /tmp/lustre-ost2
Checking servers environments
Checking clients onyx-21vm8.onyx.whamcloud.com environments
Loading modules from /usr/lib64/lustre/tests/..
detected 1 online CPUs by sysfs
libcfs will create CPU partition based on online CPUs
debug=vfstrace rpctrace dlmtrace neterror ha config                  ioctl super lfsck
subsystem_debug=all
gss/krb5 is not supported
Setup mgs, mdt, osts
Starting mds1:   -o loop /tmp/lustre-mdt1 /mnt/lustre-mds1
Commit the device label on /tmp/lustre-mdt1
Started lustre-MDT0000
Starting ost1:   -o loop /tmp/lustre-ost1 /mnt/lustre-ost1
Commit the device label on /tmp/lustre-ost1
Started lustre-OST0000
Starting ost2:   -o loop /tmp/lustre-ost2 /mnt/lustre-ost2
Commit the device label on /tmp/lustre-ost2
Started lustre-OST0001
Starting client: onyx-21vm8.onyx.whamcloudcom:  -o user_xattr,flock onyx-21vm8.onyx.whamcloud.com@tcp:
/lustre /mnt/lustre
UUID                   1K-blocks        Used   Available Use% Mounted on
lustre-MDT0000_UUID       125368        1736      114272   1% /mnt/lustre[MDT:0]
lustre-OST0000_UUID       350360       13492      309396   4% /mnt/lustre[OST:0]
lustre-OST0001_UUID       350360       13492      309396   4% /mnt/lustre[OST:1]

filesystem_summary:       700720       26984      618792   4% /mnt/lustre

Using TIMEOUT=20
seting jobstats to procname_uid
Setting lustre.sys.jobid_var from disable to procname_uid
Waiting 90 secs for update
Updated after 7s: wanted 'procname_uid' got 'procname_uid'
disable quota as required
```

2. You will now have a Lustre filesystem available at `/mnt/lustre`

ENDS~