LUG 2026

Whamcloud

LTVM: tiny VMs and LLM tools

LUG ‘26 Devday workshop

Patrick Farrell & Marc Vef

ddn

Overview

Whamcloud

- By the end of this presentation, you will know:
  - What ltvm is
  - How use ltvm to easily cross-build Lustre for different **distros** and **architectures**
    - **Eg, Rocky 9 host, Ubuntu/Rocky 8 VMs**
  - How to create an ltvm VM in ~30 seconds with Lustre mounted
  - How to start/stop/crashdump VMs
  - How a set of CLI tools lets an LLM agent **use** all of this on your behalf — review code, triage CI, analyse vmcores

- Let's start with ltvm

Developer Friction – Testing and Building

- Lustre is a large project, and an old project…
- Building Lustre from source for the first time is … time consuming
- Build process is a wiki page for every distro – no usable build script

Prerequisite

- A newly installed RHEL/Rocky linux 8 x86_64 or aarch64 machine connected to the internet.
- **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.
- This manual uses the Rocky linux 8 distribution available at x86_64 or aarch64.
- It is assumed that Rocky linux 8 was installed as is, and nothing else was installed on it.

Overview

This document walks through the steps of patching the kernel, building Lustre and running a basic test of the complete system.

If you prefer to skip the kernel build steps and get right to building Lustre, you can download and install pre-patched server kernel RPMs from a recent lustre-mas least the `kernel`, `kernel-devel`, `kernel-modules` RPMs to install.

Process

Provision machine and installing dependencies.

Once RHEL/Rocky 8.10 is newly installed on an `x86_64 or aarch64 machine` login as user `root`.

1. Install the kernel development tools:

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

2. Install additional dependencies:

```
yum config-manager --set-enabled powertools
dnf install -y gcc autoconf libtool which make patch diffutils file binutils-devel python38 python3-devel elfutils-devel l
yum -y install audit-libs-devel binutils-devel elfutils-devel kabi-dw ncurses-devel newt-devel numactl-devel openssl-devel
dnf install -y epel-release
dnf install -y ccache pdsh
dnf --enablerepo=ha install resource-agents
```

3. Install tools for kernel RPM build:

```
dnf install -y bpftool dwarves java-devel libbabeltrace-devel libbpf-devel libmnl-devel net-tools rsync
# May only be needed on RHEL9 derivatives:
dnf install -y python3-devel
```

4. Install e2fsprogs packages:

```
git clone "https://review.whamcloud.com/tools/e2fsprogs" e2fsprogs && cd e2fsprogs && git checkout v1.47.3-wc2 && ./config
sudo make install
cd ..
```

Or download and install the following e2fsprogs packages from https://build.whamcloud.com/

Preparing the Lustre source.

Lustre VM Prep Process

Whamcloud

- Create a VM with the distro you want to test…
- Get Lustre source…
- Find the wiki page for your distro…
- Install build packages (list may be outdated)…
- Download kernel source… (Is the wiki current on which version…? Probably not...)
- Install kernel config… (Is the kernel config current with the kernel you downloaded?)
- Patch kernel… Build kernel…
- Fix issues with kernel build…
- Build Lustre… (ldiskf patches don't match? Might have to go back to "Download kernel source")
- Fix issues with Lustre build… (missing packages, etc)
- Lustre built!

Lustre VM Prep Process

Whamcloud

- Install kernel…
- Reboot...
- Install Lustre…
- Build or install e2fsprogs… (Oh, there's a dependency chain issue with the distro e2fsprogs…?)
- Figure out what llmount.sh is…
- Run llmount.sh…
- Fix issues with VM... Is your network config right? Do you have all the necessary packages?
- Run llmount.sh again
- **Success!** You have a Lustre dev VM. If you need another distro or the kernel changes, just repeat the process.

Containerized builds & VMs

Whamcloud

- What if we built the kernel and Lustre in containers?
  - We can build for every distro, every arch, from anywhere.
- What if we had standardized image recipes?
  - Easily generate a VM image with the necessary packages installed **and** configuration in place
- What if we did Lustre dev in mini-vms, created inside a regular VM?
  - VMs can be spun up in seconds without interacting with a hypervisor
- What if we pre-published build artifacts?
  - Container build → Container download
  - Kernel build → Kernel download
  - Image build → Image download
  - But also publish build recipes, so you can build.

ltvm: Containerized builds & ephemeral VMs

Whamcloud

- On a VM **with nested virtualization support**:
- sudo ltvm install
  - Install dependencies, configure qemu and networking…
- ltvm target fetch rocky9
  - Download build container + kernel + image w/Lustre from github releases; ~1.5 GiB
- ltvm create test-vm1
  - Create default VM (rocky9)
- ltvm llmount co1-single
  - Mount with built in Lustre
- **~3 minutes from “install” to “VM running Lustre”**

ltvm: Containerized builds & ephemeral VMs

Whamcloud

- For daily use: Build container also prepped:
  - ltvm build lustre rocky9 ~/lustre-release
  - ltvm deploy-lustre paf-testvm1 --lustre-tree ~/lustre-release
    ^^^ Builds and deploys against that vm
  - Iterate: edit ~/lustre-release → rerun deploy-lustre
  - Multi-node: ltvm cluster create co2 mgs+mds:mds:1 oss:oss:3 (early, not well-verified)

ltvm: Ingredients

Whamcloud

- Lustre source + Kernel source + Container recipe + Image recipe
- Containers
  - Container recipe for every distro + arch combo (x86 + Rocky 9, ARM 64 + Rocky 10, etc)
  - Build container (docker/podman, etc)
  - Build the kernel in the container
  - Build Lustre in the container
- Create base image
  - Dockerfile for base image
  - Additional overlay for variants (eg, mofed)
- Base Image + Kernel + Lustre → ext4 image (bootable by qemu)
  - Can also generate full qcow2 image for other hypervisors

ltvm: qemu VMs

Whamcloud

- qemu VMs
  - Full control of the VM from your VM – no hypervisor required
  - Create and boot in seconds (16 seconds on GCP, 8 seconds on my desktop)
  - Support kdump (Unlike Amazon Firecracker)
  - Support x86, aarch64 (either directly or via emulation)
  - Support PCIe passthrough (can use SR-IOV + real Infiniband cards (thanks to Cyril Bordage of DDN for verifying this))
  - ltvm vm crash-collect <vm> — vmcore, crash/drgn-ready for analysis

ltvm: Targets

- Can support every kernel and OS Lustre supports (mostly works already)
  - Targets split by base OS
  - “variants” for different configs, eg mofed
  - Too much to go into here…

```
ltvm target list
Local  Remote  Target       Arch     Variants                          Mode              Default?
-------------------------------------------------------------------------------------------------
-      -       rocky10      x86_64   6.12-rhel10.1                     server_ldiskfs
-      -                               base
-      -                               6.12-rhel10.0                                       yes
yes    -                               base
-      -       rocky8       x86_64   4.18-rhel8.10                     server_ldiskfs    yes
yes    -                               base
-      -                               4.18-rhel8.9
-      -                               base
-      -                               4.18-rhel8.8
-      -                               base
```

ltvm: Community development

Whamcloud

- ltvm is brand new – first commits April 2026
- Lustre test configuration is complicated – many corner cases
  - Who other than Oleg can say their dev VMs can run **all** of the test-framework?
- Normally everyone does this for themselves…
- ***Let's share the load***.
- ltvm is open for contributions – check out the code, try something, find it doesn't work → Make a PR on github.
  - Add new OS and kernel targets, fix tests, add features…
- Not owned by any company – Given freely to the community. (BSD licensed)
- Artifacts are currently built and pushed by Patrick – maybe this isn't the best long term plan. TBD.
- Marc will focus on aarch64 builds for Apple Silicon Macs

ltvm: Github

Whamcloud

- ltvm: Free, open source, lives on github
- https://github.com/lustre-tools/lustre-test-vms
- Requirements:
  - Modern Linux VM (host distro agnostic) **with nested virtualization support**
  - On Mac, that means a UTM VM (not Parallels, not VMWare fusion): https://mac.getutm.app/
    - This doesn't work yet but we'll hopefully have that soon
    - In any case, requires M3+, latest Mac OS and latest UTM
  - You can use ltvm without nested VM support
    - But you lose the agent isolation of a VM
- Check out, run ltvm install, ltvm --help, read README.md – or just have your agent use it

- Demo..

LLM code and review tools

Whamcloud

- With LTVM, agents can spin up mini vms to complete various tasks
- LLM tools help agents to further interact with our services and more
  - Gerrit CLI: get patch state, comments, reviews, and interact with them
  - Jira CLI: get, search, comment, get attachments
  - Jenkins: build status, console output, retrigger, abort
  - Janitor: get Janitor test results
  - Maloo: get failures and its logs, find related bugs
  - drgn-tools: drgn-based Lustre vmcore analysis tools
  - Gerrit graph: Generate an interactive graph to visualize patch series including comments and state
- LLM tools are made for LLMs (output JSON)
  - Can use --pretty for humans
  - Needs API keys for each service
- LLM tools and LTVM allow a full workflow for LLMs:
  - Look at Jira ticket, spin up mini vm, reproduce (and possibly propose a fix)

Some examples

Whamcloud

```bash
# Simple lookup
jira --pretty get LU-10911

# With comments — what humans said about the bug
jira --pretty get LU-10911 --comments 5

# Just the comments
jira --pretty comments LU-10911 --limit 5
```

```
# Drilling into a specific session:
# overview: which suites passed/failed
maloo session <session-UUID> | jq
# only the failed suites + subtest errors
maloo failures <session-UUID> | jq
```

```bash
# unresolved comments with code context — what the LLM reads to fix review feedback
gc comments https://review.whamcloud.com/61965 | jq

# Full diff of the latest patchset (can be large — head it for the demo)
gc review https://review.whamcloud.com/61965 | jq | head -80
```

Gerrit Graph visualizer

Whamcloud

- Gerrit patch series are notoriously difficult and time-consuming to manage
- When patches are updated and the series runs out of sync, ordering is random
- To find the current state of patch, you must navigate to each individually
- The Gerrit graph visualizer aims to solve this:
  - It visually presents a patch series
  - See what depends on what and when
  - Node colors help to quickly understand if Jenkins, Maloo, Janitor failed, or a reviewer vetoed
  - Unresolved comments are shown
  - Can navigate to Gerrit, checkout, and cherry pick from the visualizer
  - Include related patches via hashtag and topics
- Processes large graphs (wbc) in about a minute (fetching comments is opt-in)
- Demo…

Closing

Whamcloud

- LTVM and LLM tools are open source and are continuously improved
- https://github.com/lustre-tools/llm_code_and_review_tools/
  - LLM tools and Gerrit graph visualizer (part of Gerrit CLI)
  - BSD-licensed
- https://github.com/lustre-tools/lustre-test-vms
  - BSD-licensed
- Made for and by the community
- Both projects are open for contributions
- Bring your LLM! If you see issues, make a PR on Github

Whamcloud

Thank you!

Marc-André Vef - mvef@whamcloud.com