



# Lustre Development

**Eric Barton**

Lustre Group - Lead Engineer



# Strategy

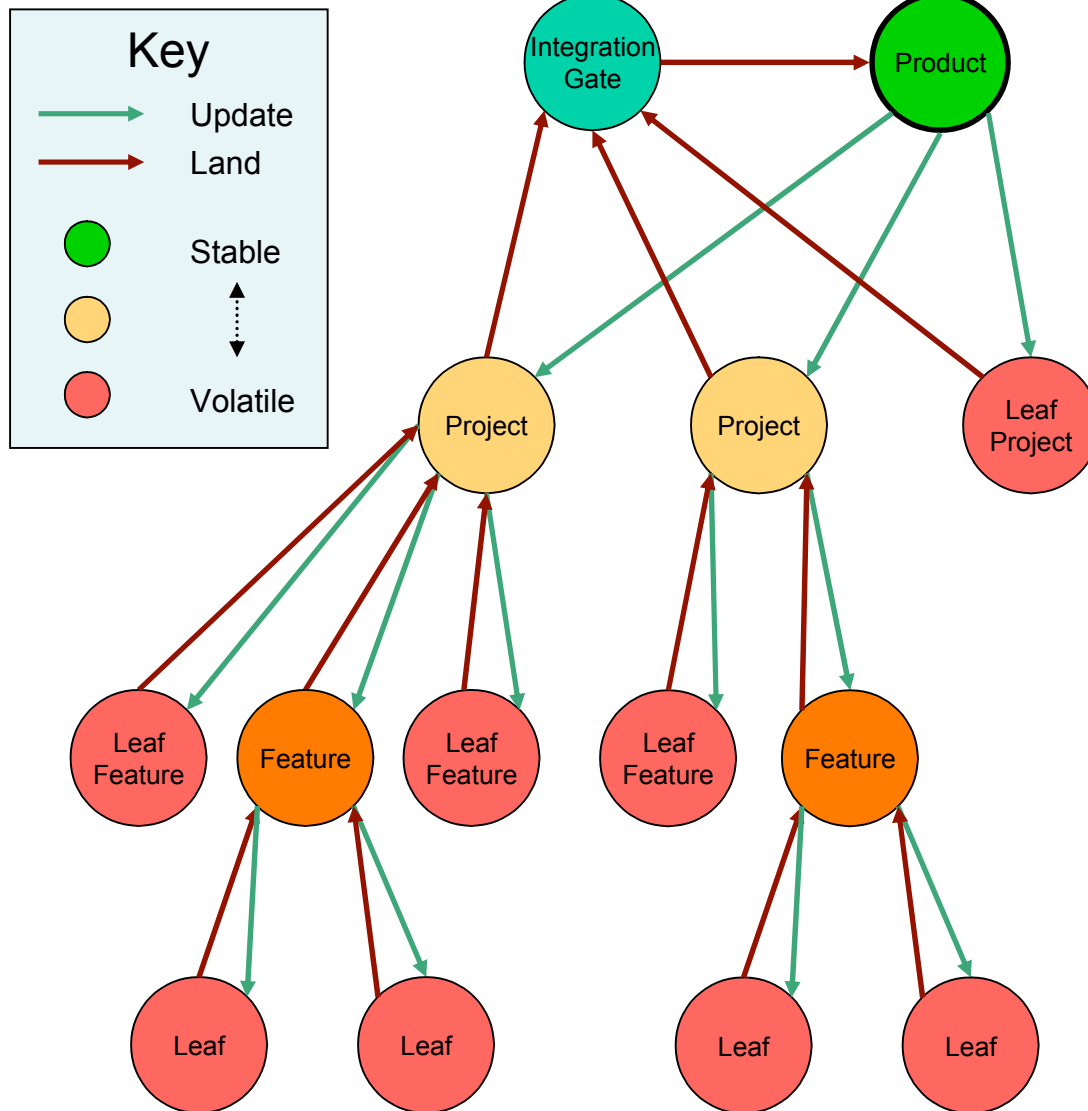
- Believable Roadmap
  - > HEAD to Good 
  - > Interoperability
  - > Userspace servers / ZFS
  - > CMD
- Quality
  - > Improve design and development processes
    - Branch management
    - Design and Coding
  - > Improve our understanding
    - Benchmarking
    - Subsystem map



# Branch Management

Terry Rutledge  
Email: [Terry.Rutledge@Sun.COM](mailto:Terry.Rutledge@Sun.COM)  
Phone: 877-361-1636 / x40289

# Branch Management



- Stable Product Branch
  - > No landings without integration testing
- Develop on Leaf Branches
  - > Frequent commits for backup history and visibility
  - > Developers may share leaf branches (but live in each other's pockets).
- Frequent updates
  - > Up-to-date context for development
  - > Spread the pain
- No landing without testing
  - > Pre-integration tests required before landing on gate
  - > Integration tests required before landing on product

# Branch Management – Naming

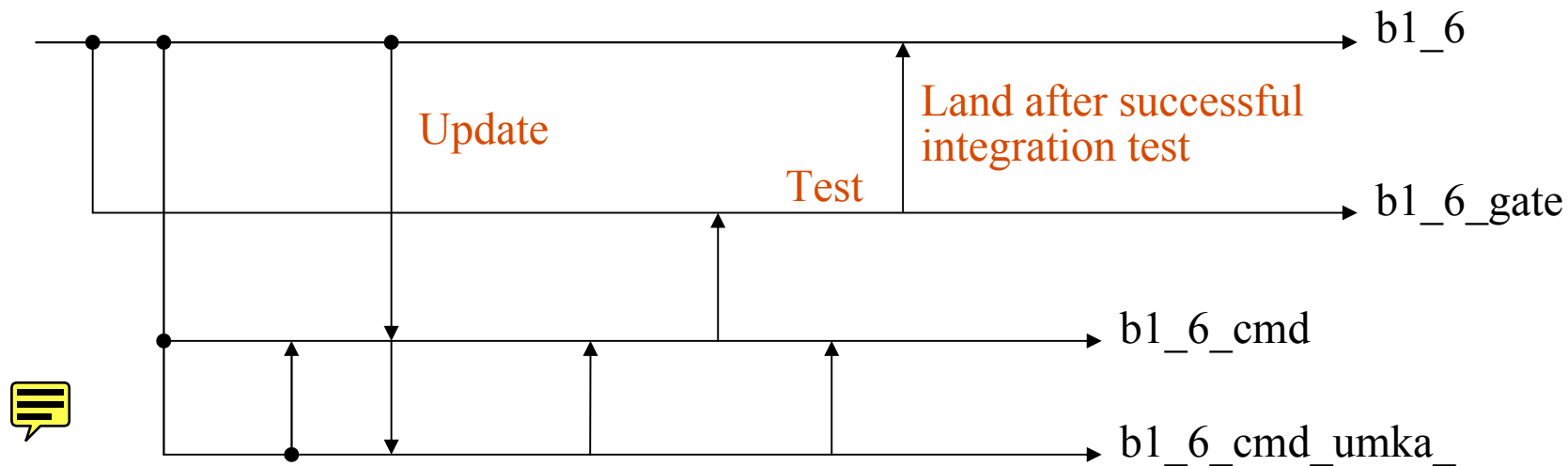
- Branches naming scheme:
  - > *bmajor\_minor*
  - > *bmajor\_minor\_gate*
  - > *bmajor\_minor\_project[\_feature[\_developer]]*
  - > *bmajor\_minor\_bug12345*
  - > *bmajor\_minor\_bugs*
- Examples:
  - > *b1\_6*
  - > *b1\_6\_gate*
  - > *b1\_6\_bugs*
  - > *b1\_6\_cmd*
  - > *b1\_6\_cmd\_umka*
  - > *b1\_6\_bug12345*

# Branch Management – Creation

- Product (including gate & minor bug branches)
  - > Only with permission of PMO
- Direct child of product branch
  - > Only with permission of PMO
  - > Major bugs
  - > Development projects and features
    - Branch plan must be included in project plan
- Indirect children of product branch
  - > Only with permission of parent branch owner
- Wiki branch table



# Branch Management – Landing



- Only gate may land on product branch
  - > Only after successful integration test
- Only direct children of product branch may land on gate
  - > Only with permission of gatekeeper

## Branch Management – Gate Landing Policy

- Code must have been inspected
  - > At least 1 inspector must be a senior engineer
- Code must have been tested
  - > According to feature/project test plan
- Gatekeeper must have granted permission
  - > Request via associated bug
  - > Associated bug must block gate bug
- Minor bugs stay as patches to the ‘bugs’ branch
  - > Commit patch directly to gate when above conditions satisfied

# Branch Management – Integration Testing

- QE runs integration tests
  - > Gatekeeper determines what tests to run
- On Success...
  - > Update all product branch children
- On failure
  - > Diagnose, back out relevant gate landings, retest
  - > ...or...
  - > Back out all gate landings
  - > Require branch/bug owners to retest/resubmit

# Branch Management – What are we working on now?

- Creation of the `_gate` and `_bugs` branches for HEAD and all existing product branches
- Updates to the `land[12].sh` scripts to accommodate the new gate branch
- Letting people know about the new release structure and asking for their assistance in making this transition happen smoothly.
- Getting feedback from you on things which may be unclear or that we could do better.



# Process



# Process – Goals

- Estimate as accurately as possible
- Minimize defects
  - > Prevent defects entering the development process
  - > Catch defects as early as possible
  - > Catch as many defects as possible
- Learn and improve
  - > Planners
  - > Designers
  - > Coders
  - > Inspectors

# Process – Ask!

- Ask!
  - > If you think it's pointless
  - > If you think something is missing
  - > If you don't quite see how it fits together
  - > If you think you missed something
  - > If you don't know
  - > If you have any doubt
  - > If you feel unsure

# Process – Ask!

- It takes courage to ask
  - > especially if you're unsure
  - > especially if you think other people know
  - > especially if you feel foolish
- You'll feel much better after you've asked!
  - > You might have found an omission or a defect
  - > You will have learnt something

# Process – Steps

- Process steps up to landing on the gate
  - > Architectural requirements, inspection, plan
  - > HLD, inspection, replan
  - > Coding launch meeting
  - > DLD, inspection, replan
  - > Coding, unit test, inspection
- Everything recorded
  - > All plans
  - > All inspections including defect count
- We get to know “How are we doing?”
  - > Progress
  - > Defects
  - > Estimates



# Process – Inspection

- The output from an inspection is a text file listing the following on the first 6 lines
  - > Total defect count (number first)
  - > Size of thing inspected (lines, number first)
  - > Person who's work was inspected
  - > Inspector
  - > What was inspected
  - > Date
- Numbered list of defects with a short description of each one. Use a blank line between each defect.
- Separate list of suggestions. Use a blank line between each suggestion.



# Process: Inspection

- A single *defect* is any single (or systematic) error or omission that prevents the inspected work meeting its requirements
  - > Architecture: missing requirement
  - > HLD: faulty logic, missing functionality
  - > Code: program error, unacceptable unreadability
- Defects must be fixed
- Suggestions are for guidance
  - > Improve code style
  - > Improve readability
  - > Not absolutely required to be followed, but they're not there for nothing.

# Process – Inspection Example

2 defects

200 LOC

Someone

Eric Barton

{u,k}ptllnd alignment bug CODE

06/11/28

Defects

1. In `ptllnd_rx_buffer_callback()`, when the incoming message is misaligned, first check `ev->mlength` before blindly copying the payload into `rx_space`. If the message is too long you must force a protocol error.

2. In the userspace LND it's wrong simply to assert that the incoming message is aligned - a buggy peer will crash you.

Suggestions

Take care where you handle defect 1 in case returning a protocol error makes the caller drop the connection in interrupt context.

# Process – Architecture

- Every HLD needs a current architectural requirements document with a complete list of high-level requirements
- Check with project management *before* starting an HLD...
  - > Does the architecture document exist?
  - > Is it current?
  - > Has it passed inspection?

# Process – HLD

- HLD writers must be qualified
  - > Senior Engineer
  - > Subsystem owner/deputy
  - > Training exercise – under guidance
- HLD inspectors
  - > Peter Braam & Eric Barton
  - > Mentoring most senior engineers
  - > Develop HLD inspection checklist
- Start from the HLD template
  - > Being revised now
  - > Use HLD inspection checklist
- HLD must include a test plan



# Process – Coding Launch Meeting

- HLD writer + HLD inspector + coder
- Coder must convince HLD writer and inspector s/he understands the architectural requirements and the HLD
- Good preparation + questions essential
- Record time/date of meeting + attendees in bug



# Process – DLD

- Skeleton code in-context
  - > Complete structure definitions
  - > Major procedures
    - Prototypes including parameter lists
    - Definitions in-place
    - Body containing comments + code fragments
    - All non-trivial functionality and state machines described
- Attach patch + CVS tag to bug
  - > Doesn't have to compile
  - > Must apply cleanly



Questions?

