



# Do we need a Livepatch Developers Guide?

Linux Plumbers Conference 2019

Joe Lawrence  
Senior Software Engineer

# Current livepatch documentation

## Livepatch

1. Motivation
2. Kprobes, Ftrace, Livepatching
3. Consistency Model
4. Livepatch module
5. Livepatch life-cycle
6. Sysfs
7. Limitations

## (Un)patching Callbacks

1. Motivation
2. Callback types
3. How it works
4. Use cases

## Shadow Variables

1. Brief API summary
2. Use cases
3. References

## Livepatch module Elf format

1. Background and motivation
2. Livepatch modinfo field
3. Livepatch relocation sections
4. Livepatch symbols
5. Architecture-specific sections
6. Symbol table and Elf section access

## Atomic Replace & Cumulative Patches

1. Usage
2. Features
3. Limitations

Documentation/livepatch/\*.rst

# Current kpatch author guide

- kpatch vs livepatch vs kGraft
- Patch upgrades
- Data structure changes
  - Change the code which uses the data structure
  - Use a kpatch callback macro
    - Pre-patch return status
    - Callback context
  - Use a shadow variable
- Data semantic changes
- Init code changes
- Header file changes
- Dealing with unexpected changed functions
- Removing references to static local variables
- Code removal
- Other issues - `prink_once()`

# Current kpatch author guide

- <https://github.com/dynup/kpatch/blob/master/doc/patch-author-guide.md>
- Out of date
  - Still references kpatch.ko helper functions (shadow variables, load hooks, etc.)
  - Doesn't reference new upstream features like atomic replace

# Future options

- Nothing, upstream documentation is already great.
- Update kpatch documentation, extract livepatch relevant parts into a new upstream livepatch developer's guide?
- Other ideas:
  - FAQ: use kpatch author guide as an outline, directing readers to appropriate .rst file and section?
  - Collect post-embargoed CVE livepatches with commentary?
  - Create a livepatch blog on <https://people.kernel.org> documenting livepatch battle stories?

**THANK YOU**