



LPC 2026 • LIVE PATCHING MICROCONFERENCE

Coexistence of Live Patching and Production Observability

Yi Zhu,
Kris Van Hees,
Song Liu,

Google
Oracle
Meta

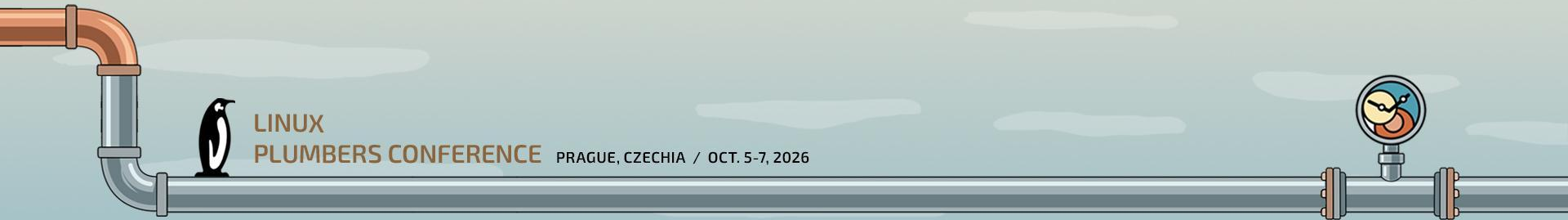


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

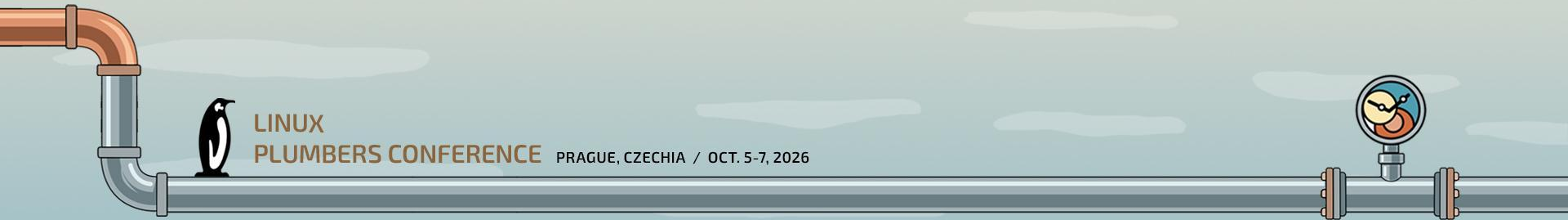
Agenda

- Why does this matter?
- Live patch vs. ftrace users
- Live patch vs. fault injection
- Live patch vs. kprobe with offset
- Live patch & tracepoints



Agenda

- **Why does this matter?**
- Live patch vs. ftrace users
- Live patch vs. fault injection
- Live patch vs. kprobe with offset
- Live patch & tracepoints



Livepatch conflicts with tracing

- In datacenters, various services are tracing kernel functions all the time
- Livepatch could update any ftrace-able kernel functions
- Known issue for years
 - Livepatch is compatible with many ftrace users
 - Livepatch is compatible with BPF trampolines since 6.0
- Still not perfect

Status of livepatch & tracing compatibility #259

Coordination between live patching and tracing #370

A Common Coexistence Model for Live Patching and Production Observability #203



LINUX

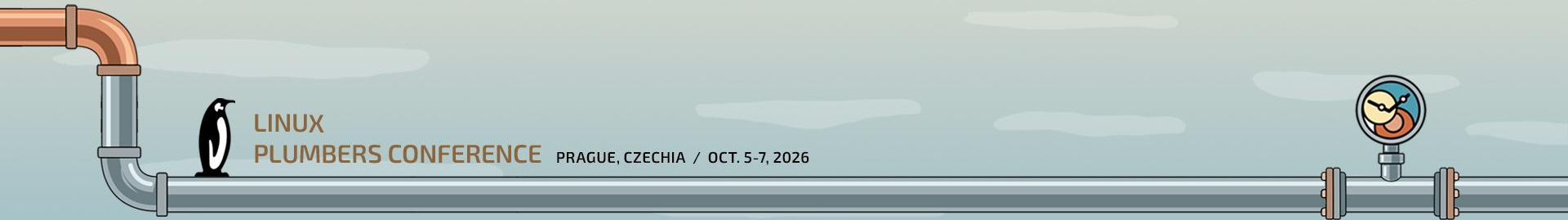
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



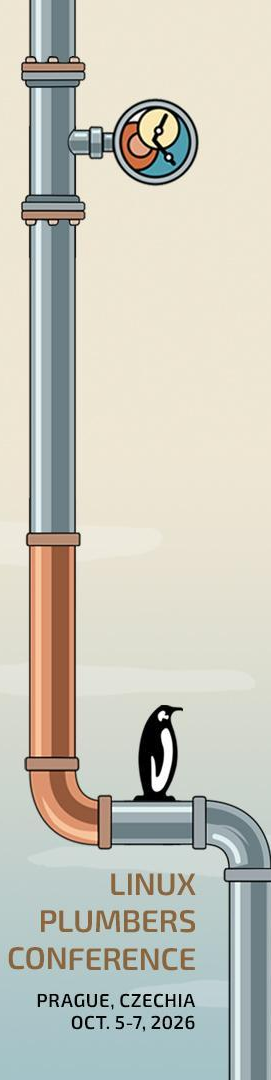
Agenda

- Why does this matter?
- **Live patch vs. ftrace users**
- Live patch vs. fault injection
- Live patch vs. kprobe with offset
- Live patch & tracepoints



How does Livepatch coexist with all Ftrace Users today?

Ftrace Users	Coexistence with Livepatch
Livepatch	OK (klp_ftrace_handler)
kprobes fail_function	registration -EBUSY
bpf_override_return	new handler fails silently
BPF fmod_ret	OK (DIRECT wins)
kprobe+post	registration -EBUSY
BPF fexit	OK
fgraph	OK
kretprobe	OK
fprobe+exit	OK
BPF fentry	OK
fprobe (entry)	OK
kprobe (entry)	OK
function tracer (tracefs)	OK

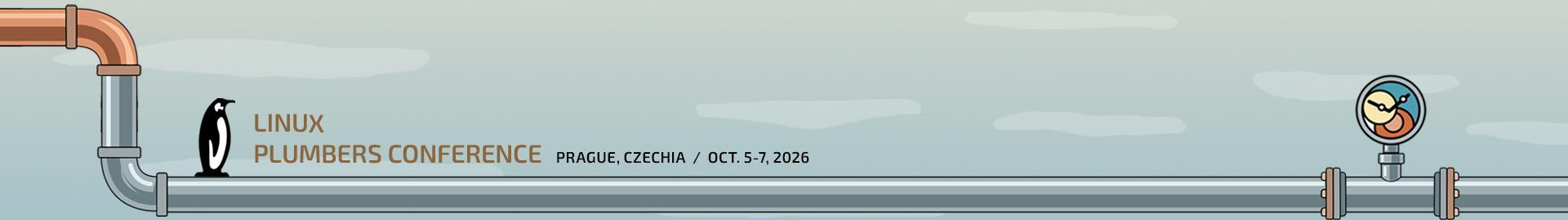


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Agenda

- Why does this matter?
- Live patch vs. ftrace users
- **Live patch vs. fault injection**
- Live patch vs. kprobe with offset
- Live patch & tracepoints

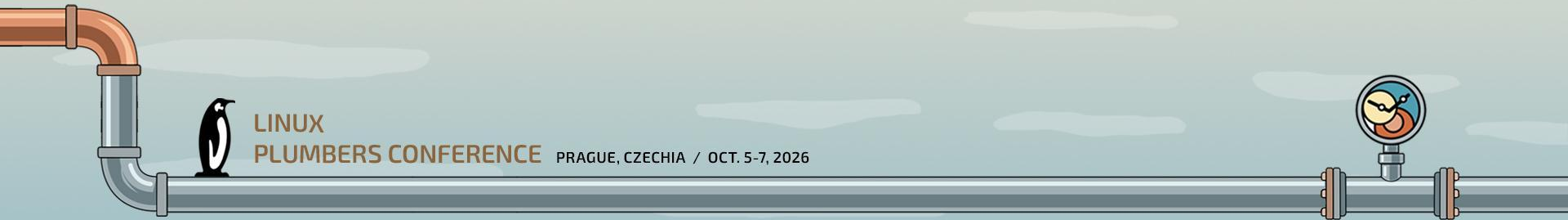


Example: Error Injection with ftrace, kprobes, and BPF

Error injection	kprobes fail_function	bpf_override_return	BPF fmod_ret / LSM
Who decides to fail	should_fail() knobs: probability, interval, times	BPF program SEC("kprobe")	BPF program SEC("fmod_ret")
How is the injector get attached	debugfs → kprobe → ftrace	BPF → kprobe → ftrace	BPF → bpf_trampoline → ftrace
ftrace registration	kprobe_ipmodify_ops	kprobe_ftrace_ops	register_ftrace_direct()
Skips by	regs->ip = just_return_func	regs->ip = just_return_func	trampoline skips orig_func if fmod_ret != 0
Livepatch coexists	new handler -EBUSY on registration	new handler's return value is overridden silently	DIRECT can skip livepatch

Agenda

- Why does this matter?
- Live patch vs. ftrace users
- Live patch vs. fault injection
- **Live patch vs. kprobe with offset**
- Live patch & tracepoints



Different kinds of kprobes

- kprobe by address
- kprobe by symbol
- kprobe by symbol + offset
- kretprobe

Assumption: CONFIG_KPROBES_ON_FTRACE=y
Otherwise kprobes are independent from ftrace. The only klp coordination that exists in the kernel is based on ftrace.



LINUX

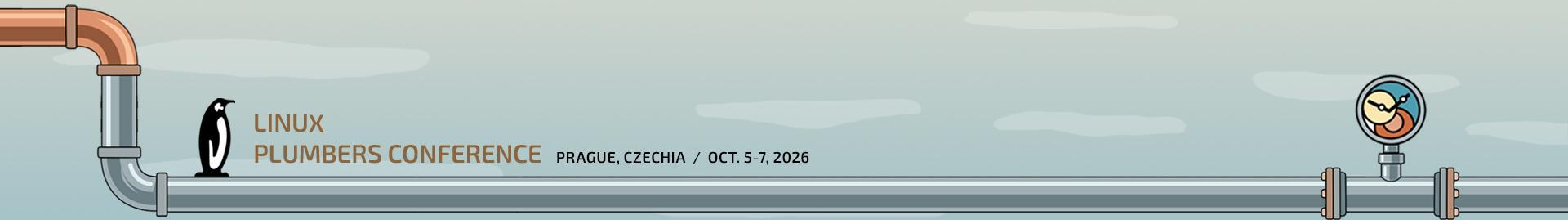
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Different kinds of kprobes

- kprobe by address
 - address == entry point of function
 - address == entry point + offset
- kprobe by symbol (i.e. entry point on function)
- kprobe by symbol + offset
- kretprobe



Different kinds of kprobes

- kprobe by address
 - address == entry point of function → Works
 - address == entry point + offset
- kprobe by symbol (i.e. entry point on function) → Works
- kprobe by symbol + offset
- kretprobe → Works

Note: It is possible to place kprobes on functions that do not have an ftrace entry location - those cannot be livepatched - so not an issue here.



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Different kinds of kprobes

- **kprobe by address**
 - address == entry point of function → Works
 - **address == entry point + offset** → **Unknown to ftrace**
- kprobe by symbol (i.e. entry point on function) → Works
- **kprobe by symbol + offset** → **Unknown to ftrace**
- kretprobe → Works



LINUX

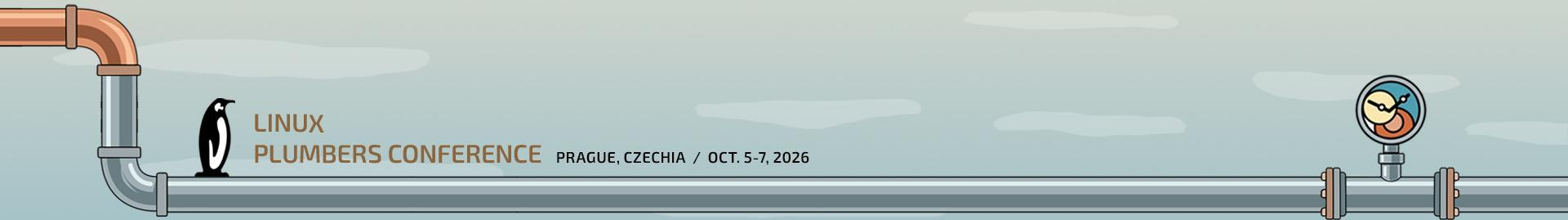
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Why are offset-kprobes a problem?

- If the kprobe address **is** an ftrace location, it is triggered through the ftrace mechanism.
 - Ftrace knows about the kprobe and can coordinate with klp.
- If the address where the kprobe is placed **is not** an ftrace location, it is implemented as a breakpoint.
 - Ftrace is not involved, so no coordination is possible.
- If the kprobes are unknown to ftrace, klp can livepatch the function and the kprobes will silently stop getting triggered.

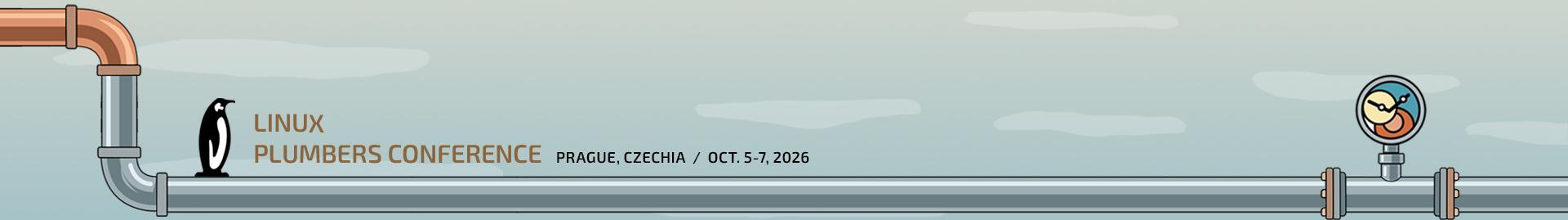


What would we like to happen?

The ideal:

Everything should keep working, just like a function entry kprobe.

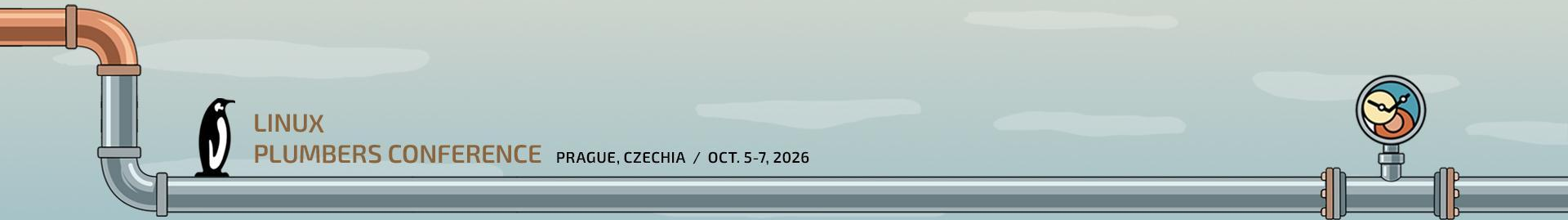
→ Impossible (no way to move kprobes to the new code.)



What would we like to happen?

Alternative 1:

- Livepatch should report -EBUSY when trying to patch a function that has kprobes in it.
- Offset kprobes should get -EBUSY if the address is within a function that has been livepatched.



What would we like to happen?

Alternative 2:

- Livepatching a function that has kprobes in it should trigger a notification that the kprobes have become unreachable.
- Offset kprobes should get -EBUSY if the address is within a function that has been livepatched.
- Removing a livepatch from a function that has kprobes in it should trigger a notification that the kprobes are reachable again.



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Implementing the alternatives

- Both alternatives depend on knowing that there are offset kprobes in the function.
- No need to know specifics about the kprobes.

Alternative 1:

- Action is only needed when adding a livepatch and when registering a kprobe.

Alternative 2:

- Action is only needed when adding a livepatch, removing a livepatch, and when registering a kprobe.



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Implementing the alternatives

Proposal:

- Create an `ftrace_ops` for each function in which any offset kprobes are registered.
- Register entry address of the function as filter ip.
- Keep a count of offset kprobes
 - Use as reference count. At 0, remove the `ftrace_ops` object.
- Use a new `FTRACE_OPS_FL_KPROBES` flag to adjust the existing checks for `FTRACE_OPS_FL_IPMODIFY`:
 - If `IPMODIFY & KPROBES`, allow it.
 - If `IPMODIFY & !KPROBES`, block it.



LINUX

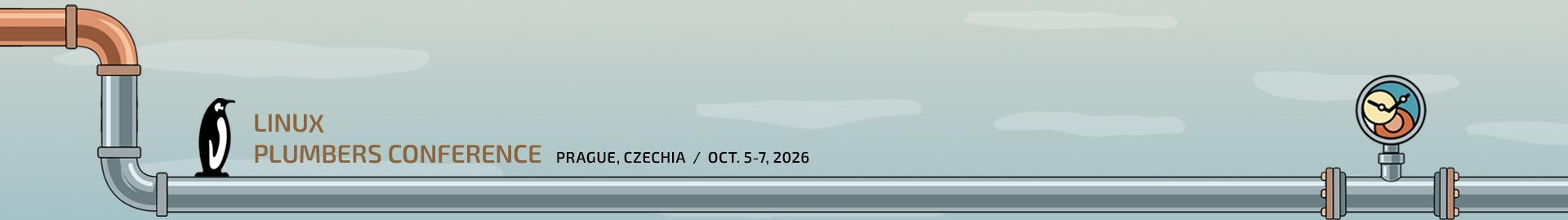
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



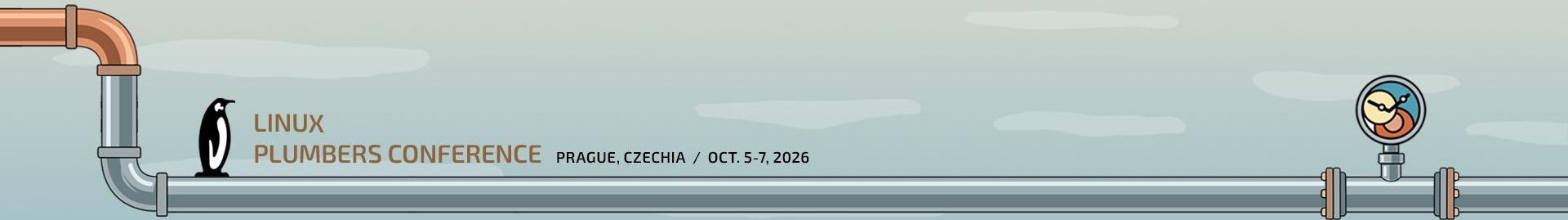
Agenda

- Why does this matter?
- Live patch vs. ftrace users
- Live patch vs. fault injection
- Live patch vs. kprobe with offset
- **Live patch & tracepoints**



Tracepoints in livepatched functions

- Tracepoints defined in vmlinux works as expected
- Tracepoints defined in other modules don't work
 - The toolchain need to remove these tracepoints
- Known issue. Fix?



Vibe coded “solution”

objtool/klp: Support static branches with module keys

objtool: Treat .klp.jump_table sections like __jump_table

livepatch: Register static branches with module keys

jump_label: Add jump_label_add_entries() and jump_label_del_entries()

jump_label: Factor out static_key_add_mod() and static_key_del_mod()

objtool/klp: Fix detection of static keys defined in other files



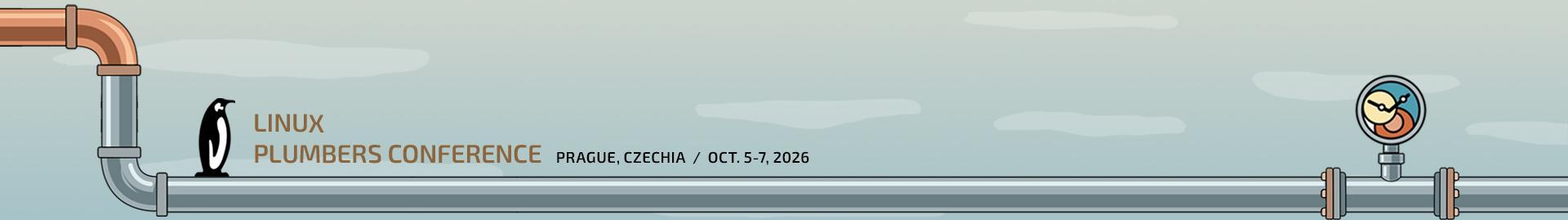
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Submodule KLP idea by Josh

- Each KLP module is multiple submodules
 - One for all changes in vmlinux
 - More submodules for changes in modules
- Could it also solve tracepoints in modules problem?



The background features a light-colored, stylized illustration of a city skyline. On the left, a large, vertical pipe with a blue body and orange-brown flanges extends from the top to the bottom. The city skyline consists of various buildings, including a prominent church with multiple spires and a large arched window. The sky is a pale yellow with soft, horizontal cloud-like shapes. The water in the foreground is a light blue with some darker blue patches.

Thank You! Questions?



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026