Contribution ID: **90** Type: **not specified**

# Traceloop and BPF

*Monday, 24 August 2020 07:00 (45 minutes)*

We will present traceloop, a tool for tracing system calls in cgroups or in containers using in-kernel Berkeley Packet Filter (BPF) programs.

Many people use the "strace" tool to synchronously trace system calls using ptrace. Traceloop similarly traces system calls but with low overhead (no context switches) and asynchronously in the background, using BPF and tracing per cgroup. We will show how it is integrated with Kubernetes via Inspektor Gadget.

Traceloop's traces are recorded in perf ring buffers (BPF_MAP_TYPE_PERF_EVENT_ARRAY) configured to be overwritable like a flight recorder. As opposed to "strace", the tracing is permanently enabled on Kubernetes pods but rarely read, only on-demand, for example in case of a crash.

We will present both past limitations with their workarounds, and how new BPF features can improve traceloop. This includes:

- Lack of bpf_get_current_cgroup_id() on Linux < 4.18 and systems not using cgroup-v2. Workaround using the mount namespace id.
- New BPF programs can only be inserted in a PROG_ARRAY map from userspace, making synchronous updates more complicated.
- BPF ringbuffer to replace BPF perf ringbuffer to improve memory usage.

## I agree to abide by the anti-harassment policy

I agree

**Primary authors:** CREQUY, Alban (Kinvolk); LÜKE, Kai (Kinvolk)

**Presenters:** CREQUY, Alban (Kinvolk); LÜKE, Kai (Kinvolk)

**Session Classification:** Networking and BPF Summit

**Track Classification:** Networking & BPF Summit