

# eBPF NOTES

TJ ROBINSON

November 4, 2025

## 1 What is eBPF?

- eBPF (extended Berkeley Packet Filter)
- Outperforms *IPtables* based solutions

### 1.1 Logs vs. Metrics vs. Observability

- Logs: Detailed, unstructured, aggregated data about individual events.
  - Useful for troubleshooting and forensic analysis.
  - Can be voluminous and harder to analyze at scale.
- Metrics: Aggregated, structured data for to monitor a program or system performance at a specific point in time.
  - Useful for identifying trends and triggering alerts.
  - Typically less detailed but more efficient to store and query.
- Observability: Combines logs, metrics, and traces to provide a comprehensive view of system behavior.
  - Capacity to ask arbitrary questions and receive complex answers about a system's state.

### 1.2 Namespaces & Cgroups

Both are fundamental for containerization and resource management in Linux.

- Namespaces: Isolate system resources for processes (e.g., PID, network, mount).
  - inside a namespace, you experience the operating system like there were no other tasks running on the computer
- Cgroups: Control and limit resource usage (CPU, memory, I/O) for process groups.
  - gives you fine grain control over resource usage like CPU, disk I/O, network, and etc.

Tracepoints are static marks in the kernel code that can be used to inject code to inspect the kernel's execution.

## 2 BPF Program Types

### 2.1 Socket Filter Programs

- Attach to network sockets to filter packets.
- Commonly used for packet filtering and monitoring.
- You can not modify the packets, only accept, drop, forward, or observe them.

### 2.2 Kprobes and Uprobes

- Kprobes: Attach to kernel functions to trace and monitor kernel events.
  - define dynamic breakpoints in the kernel code.
  - defined with the type `BPF_PROG_TYPE_KPROBE`.
  - BPF VM ensures kprobe programs are safe to run.
  - You'll need to decide whether to attach at function entry or exit of the syscall.
    - \* Entry probes: Capture arguments passed to the function.
    - \* Exit probes: Capture return values from the function.
- Uprobes: Attach to user-space functions to trace and monitor application events.

### 2.3 Tracepoint Programs

- Defined by the type `BPF_PROG_TYPE_TRACEPOINT`.
- All system tracepoints are defined in the the `/sys/kernel/debug/tracing/events/` directory.
- Attach to predefined tracepoints in the kernel.
- Used for monitoring specific kernel events.
- More stable than kprobes as tracepoints are less likely to change.

### 2.4 XDP (eXpress Data Path) Programs

- Defined by the type `BPF_PROG_TYPE_XDP`.
- Attach to the earliest point in the network stack.
- Used for high-performance packet processing.
- Ability to modify or drop packets before they reach the kernel networking stack leads to improved performance.