← All articles DevOps

eBPF Beyond Security: Networking, Observability, and Performance in One Technology

eBPF is not just for security. Learn how Cilium replaces iptables, how Pixie provides zero-instrumentation observability, and how eBPF profiles application performance without overhead.

Y Yash Pritwani 24 March 2026 12 min read read

Linux kernels have always been powerful, but accessing that power meant writing complex kernel modules, dealing with unstable APIs, and risking system crashes during development. eBPF (extended Berkeley Packet Filter) changed all of that. What started as a packet filtering mechanism has evolved into a programmable substrate that runs sandboxed programs directly inside the Linux kernel — without modifying kernel source code or loading risky modules. Today, eBPF is the foundation of some of the most ambitious cloud-native tooling ever built: replacing iptables entirely, enabling zero-instrumentation observability, and unlocking performance profiling that was previously impossible without stopping production systems. This guide digs into all three domains and shows you exactly how to get started.

What eBPF Actually Is (And Why It Matters)

eBPF programs are small pieces of bytecode that run inside a sandboxed virtual machine in the Linux kernel. When you load an eBPF program, the kernel runs it through a verifier — a static analysis pass that proves the program cannot crash the kernel, loop infinitely, or access memory it should not touch. Only after passing verification does the program get compiled to native machine code via a JIT compiler.

This safety guarantee is what makes eBPF revolutionary. Before eBPF, if you wanted to observe kernel behavior or intercept network packets at a fine-grained level, your options were:

- **Kernel modules**: Powerful but dangerous. A bug crashes the entire system.
- **ptrace/strace**: Safe but catastrophically slow (30-50x overhead).
- **Netfilter/iptables**: Flexible but built on 1990s abstractions that do not scale.

eBPF programs attach to hook points throughout the kernel — system calls, network stack events, tracepoints, kprobes, uprobes, and more. When kernel execution hits one of these points, your eBPF

program runs. It can inspect arguments, modify behavior, record data into shared maps, or redirect packets — all at kernel speed.


```
# Check if your kernel supports eBPF (need 4.15+ for most features, 5.8+ for full feature set)
uname -r

# Check what eBPF program types are supported
ls /sys/fs/bpf/

# View currently loaded eBPF programs
bpftool prog list

# View eBPF maps (shared data structures between kernel and userspace)
bpftool map list
```

The data exchange between eBPF programs and userspace happens through **eBPF maps** — key-value stores that both sides can read and write. This is how observability tools aggregate data: the eBPF program in kernel space increments counters or records events into a map, and a userspace daemon reads those maps and exports metrics or traces.

Cilium: Replacing iptables With eBPF Networking

Every Kubernetes cluster depends on networking plumbing — routing packets between pods, enforcing network policies, load balancing services. The traditional approach uses iptables, a rule-matching framework that was never designed for the scale and dynamism of container orchestration.

The problem with iptables in Kubernetes is fundamental: it uses sequential rule matching. With 10 services, this is fine. With 1000 services, the kernel is evaluating thousands of rules for every single packet. At 10,000 services — common in large clusters — iptables becomes a serious performance bottleneck.

Cilium replaces iptables entirely with eBPF programs that hook into the kernel networking stack at XDP (eXpress Data Path) and TC (Traffic Control) levels. Instead of linear rule matching, Cilium uses eBPF hash maps for O(1) lookups regardless of cluster size.

Installing Cilium on Kubernetes

```
# Install Cilium CLI
curl -L --fail --remote-name-all https://github.com/cilium/cilium-cli/releases/latest/downloa
```

```
sudo tar xzvfC cilium-linux-amd64.tar.gz /usr/local/bin
```



```
# Install Cilium on a fresh cluster
cilium install --version 1.15.0 \
  --set kubeProxyReplacement=true \
  --set k8sServiceHost=<YOUR_API_SERVER_IP> \
  --set k8sServicePort=6443

# Verify installation
cilium status --wait

# Run connectivity tests
cilium connectivity test
```

eBPF-Based Network Policies

Cilium extends standard Kubernetes NetworkPolicy with `CiliumNetworkPolicy`, which enables L7 filtering — something iptables cannot do without a sidecar proxy.

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-api-to-backend
  namespace: production
spec:
  endpointSelector:
    matchLabels:
      app: backend
  ingress:
    - fromEndpoints:
        - matchLabels:
            app: api
      toPorts:
        - ports:
            - port: "8080"
              protocol: TCP
          rules:
            http:
              - method: "GET"
                path: "/api/v1/.*"
              - method: "POST"
                path: "/api/v1/data"
```

Hubble: The Observability Layer

Cilium includes Hubble, an eBPF-powered network observability platform.

```
# Enable Hubble
cilium hubble enable --ui
```

```
# Use Hubble CLI to observe live traffic
hubble observe --pod frontend/web-pod --last 50

# Watch dropped packets in real time
hubble observe --verdict DROPPED --follow
```

Pixie: Zero-Instrumentation Observability

Traditional observability requires you to instrument your code: add trace spans, emit metrics, configure log formats. Pixie takes a fundamentally different approach — it uses eBPF to automatically capture telemetry from running applications without any code changes, SDK installation, or application restarts.

Pixie instruments at the syscall boundary. Because all network I/O, file I/O, and function calls ultimately go through system calls, Pixie can reconstruct HTTP requests, database queries, gRPC calls, and more — all from raw kernel data.

What Pixie Captures Automatically

- Full HTTP/1.x request and response bodies
- gRPC method calls and response codes
- PostgreSQL, MySQL, Redis, Cassandra queries with latency
- DNS lookups
- CPU flame graphs per process

Installing and Using Pixie

```
# Install Pixie CLI
bash -c "$(curl -fsSL withpixie.ai/install.sh)"

# Deploy Pixie to your cluster
px deploy

# Show HTTP error rates per service
px run px/http_data -- --start_time="-5m"

# Show database query latency
```

```
px run px/mysql_data -- --start_time="-15m"
# CPU flamegraph for a specific pod
px run px/cpu_flamegraph -- --start_time="-30m" --namespace=production
```

bpftrace: Production Performance Profiling

bpftrace is a high-level tracing language for eBPF. It lets you write one-liner programs that attach to kernel and userspace probes, making it the go-to tool for performance investigations on live systems.

```
# Trace all file opens by process name
bpftrace -e 'tracepoint:syscalls:sys_enter_openat { printf("%s opened %s\n", comm, str(args->f

# Count syscalls per process (top 10)
bpftrace -e 'tracepoint:raw_syscalls:sys_enter { @[comm] = count(); } END { print(@, 10); }'

# Trace TCP connection latency
bpftrace -e 'kprobe:tcp_connect { @start[tid] = nsecs; } kretprobe:tcp_connect { @ms = hist((n

# Show which processes are doing disk I/O
bpftrace -e 'tracepoint:block:block_rq_complete { @bytes[comm] = sum(args->nr_sector * 512); }
```

Getting Started: A Practical Path

Step 1: Verify Kernel Capabilities

```
uname -r
cat /proc/sys/net/core/bpf_jit_enable
ls /sys/kernel/btf/vmlinux
```

Step 2: Install Core Tools


```
apt-get update
apt-get install -y bpftrace bpfcc-tools linux-headers-$(uname -r) libbpf-dev clang llvm
```

Step 3: Explore With BCC Tools

```
/usr/share/bcc/tools/syscount -i 1
/usr/share/bcc/tools/biolatency -D
/usr/share/bcc/tools/tcpconnect
/usr/share/bcc/tools/filetop
```

Step 4: Set Up Cilium on a Single-Node Cluster

```
curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="--flannel-backend=none --disable-network-poli
export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
cilium install
cilium hubble enable --ui
cilium status --wait
```

The Performance Numbers

- **Service insertion latency**: iptables scales linearly (100ms+ at 10K rules) vs eBPF O(1) sub-millisecond
- **Packets-per-second**: iptables ~1.5M PPS vs XDP eBPF ~25M+ PPS
- **Observability overhead**: strace 50-100x slowdown vs eBPF tracepoint 1-3% overhead

Conclusion

eBPF is a platform — a programmable layer inside the Linux kernel that lets you build radically better versions of existing tools. Cilium demonstrates that iptables can be replaced entirely. Pixie shows that requiring developers to instrument their code is an artificial constraint. bpftrace gives you a probe-anywhere profiler that works on production systems without restarts.

Start with bpftrace one-liners on a development machine. Then deploy Cilium on a test cluster. The picture that emerges — of how your software actually behaves at the kernel level — consistently surprises engineers who have been debugging the same systems for years.

#ebpf #cilium #observability #networking #performance #linux #cloud-native

TE TechSaaS

Need help with devops?

TechSaaS provides expert consulting and managed services for cloud infrastructure, DevOps, and AI/ML operations.

Get a Free Consultation | WhatsApp Us

We Will Build You a Demo Site — For Free

Like it? Pay us. Do not like it? Walk away, zero complaints. You will spend way less than hiring developers or any agency.

47+ companies trusted us | **99.99%** uptime | **< 48hr** response

your@email.com

+91 98765 43210 (optional)

Get My Free Demo

No spam. No contracts. Just a free demo.

Related Articles

Architecture

WebAssembly on the Server: Running Wasm Workloads...

11 min read

DevOps

ArgoCD Beyond the Basics: Multi-Cluster GitOps...

13 min read

DevOps

Version Control at Scale: Git Strategies That Survive...

11 min read

TE TechSaaS

Stay in the Loop

Get product updates, engineering blog posts, and tech insights. No spam, ever.

you@company.com Subscribe

TE TechSaaS

Full-stack product studio delivering AI-powered platforms, SaaS products, and enterprise solutions across HR-Tech, Ed-Te...

PRODUCTS

- Skillety (AI Hiring)
- Entrance (3D Learning)
- OpenClaw (AI Gateway)
- PADC (Cloud Platform)
- View All Products

SERVICES

- AI & ML Solutions
- Full-Stack Development
- Cloud & DevOps
- Product Consulting

COMPANY

- About Us
- Careers
- Blog
- Contact

LEGAL

- Privacy Policy
- Terms of Service
- Cookie Policy
- Shipping Policy
- Refund Policy
- Payment Terms

Privacy Terms Cookies Shipping Refund Payment Terms