Hewlett Packard Enterprise

eBPF

No More Blindspots

How eBPF Transforms Observability

March 26th, 2025

Presented by:
Neil Pearson, Principal Presales Architect

OpsRamp

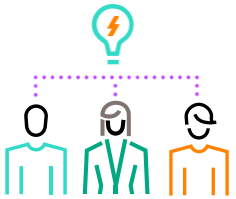
a Hewlett Packard Enterprise company

Agenda

- Intro & Current Observability Approaches
- Challenges Today
- What is eBPF?
- How eBPF Transforms Observability of Systems
- eBPF & OpsRamp Use-Cases
- Installation & Demo of OpsRamp with eBPF Telemetry
- Limitations, Resources, Summary + Q&A

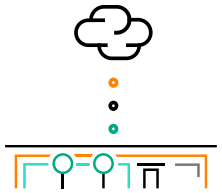
The world we live in today

Explosion of Observable data



Convergence of Ops

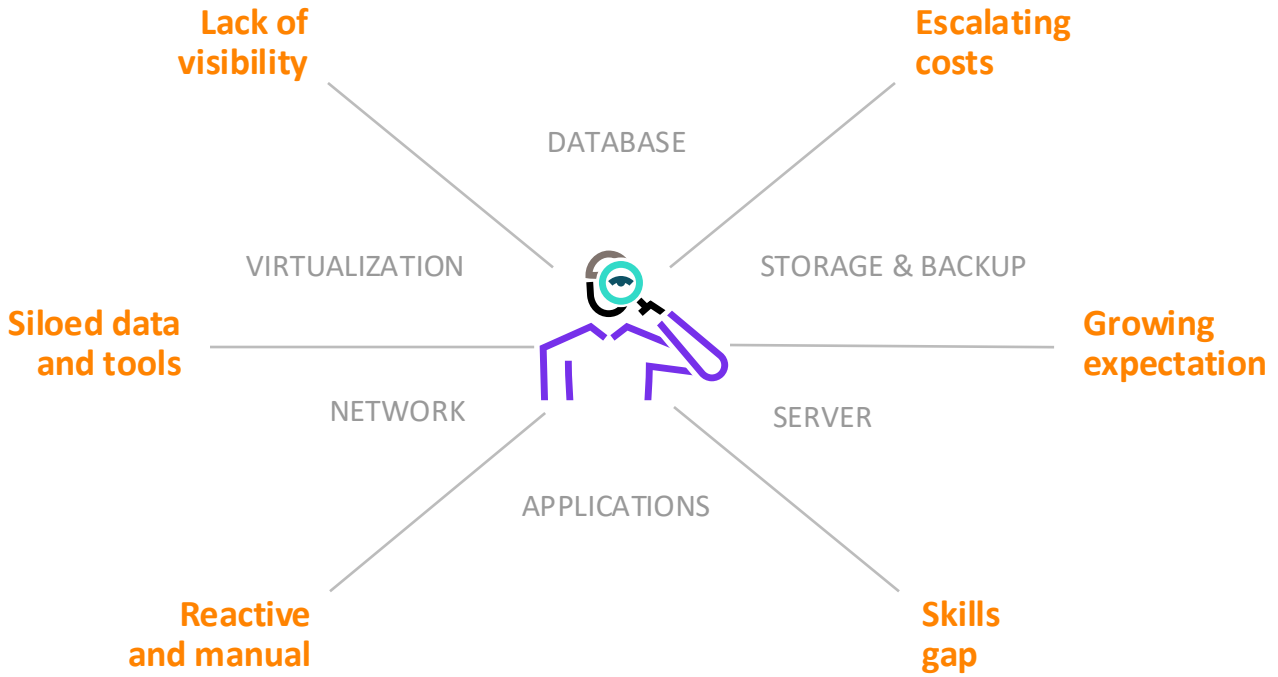
IT Ops / DevOps / SRE



Data center relevance

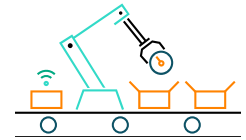
Impact of AI

Pressures on IT & DevOps



ON-PREMISES

MULTI-CLOUD

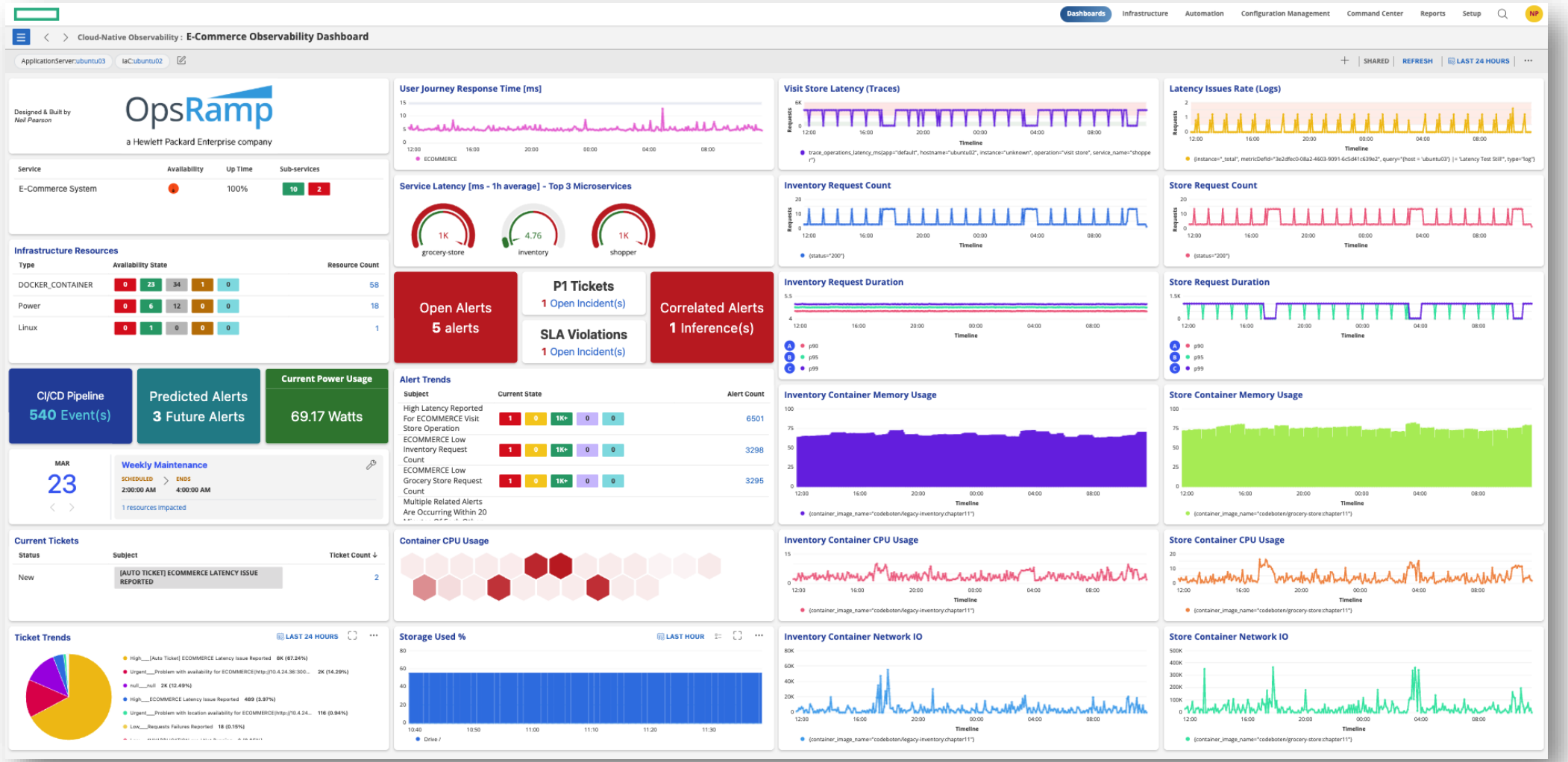


EDGE

Current Observability Approach

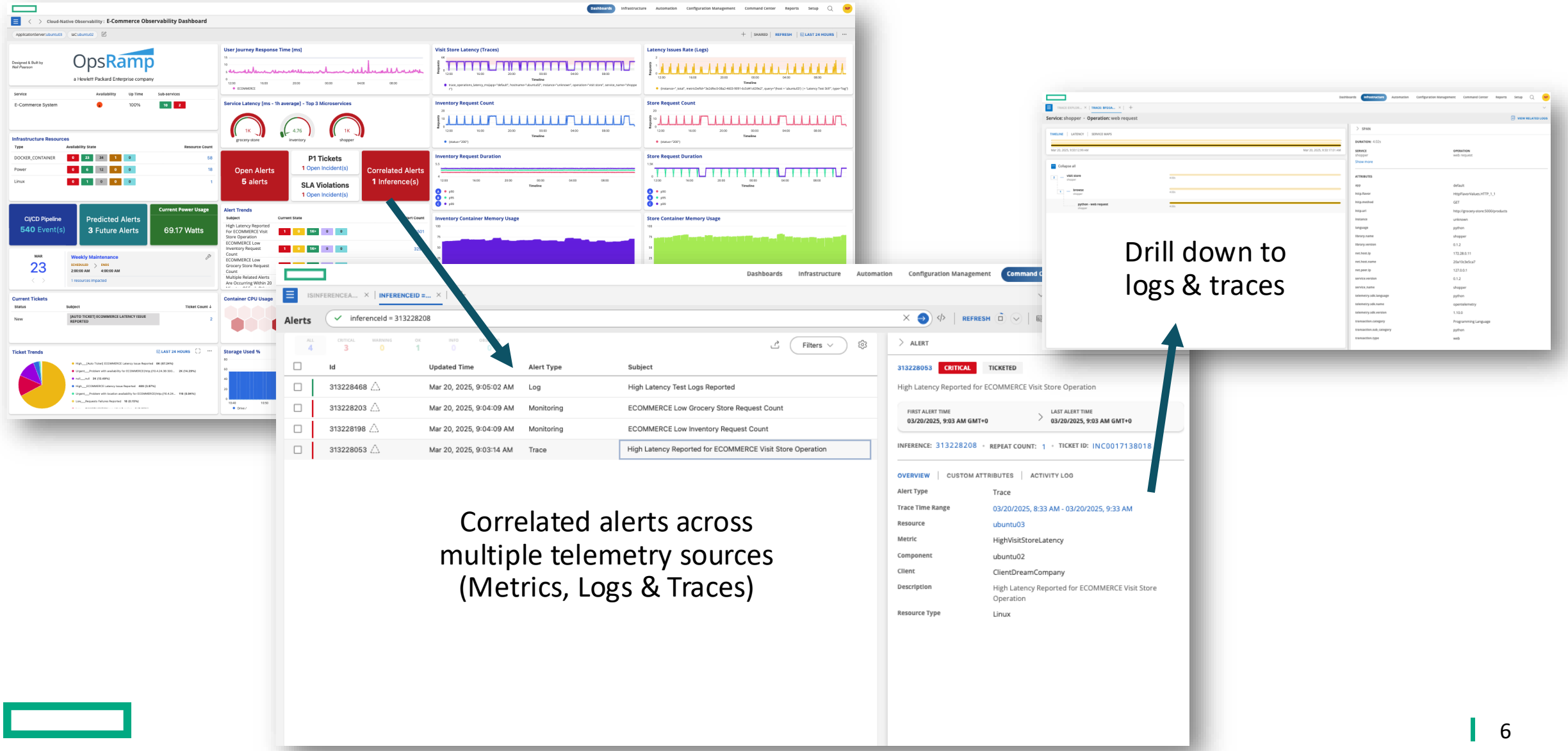
Current Observability Approach

OpsRamp
a Hewlett Packard Enterprise company



https://hpedemoportal.ext.hpe.com/

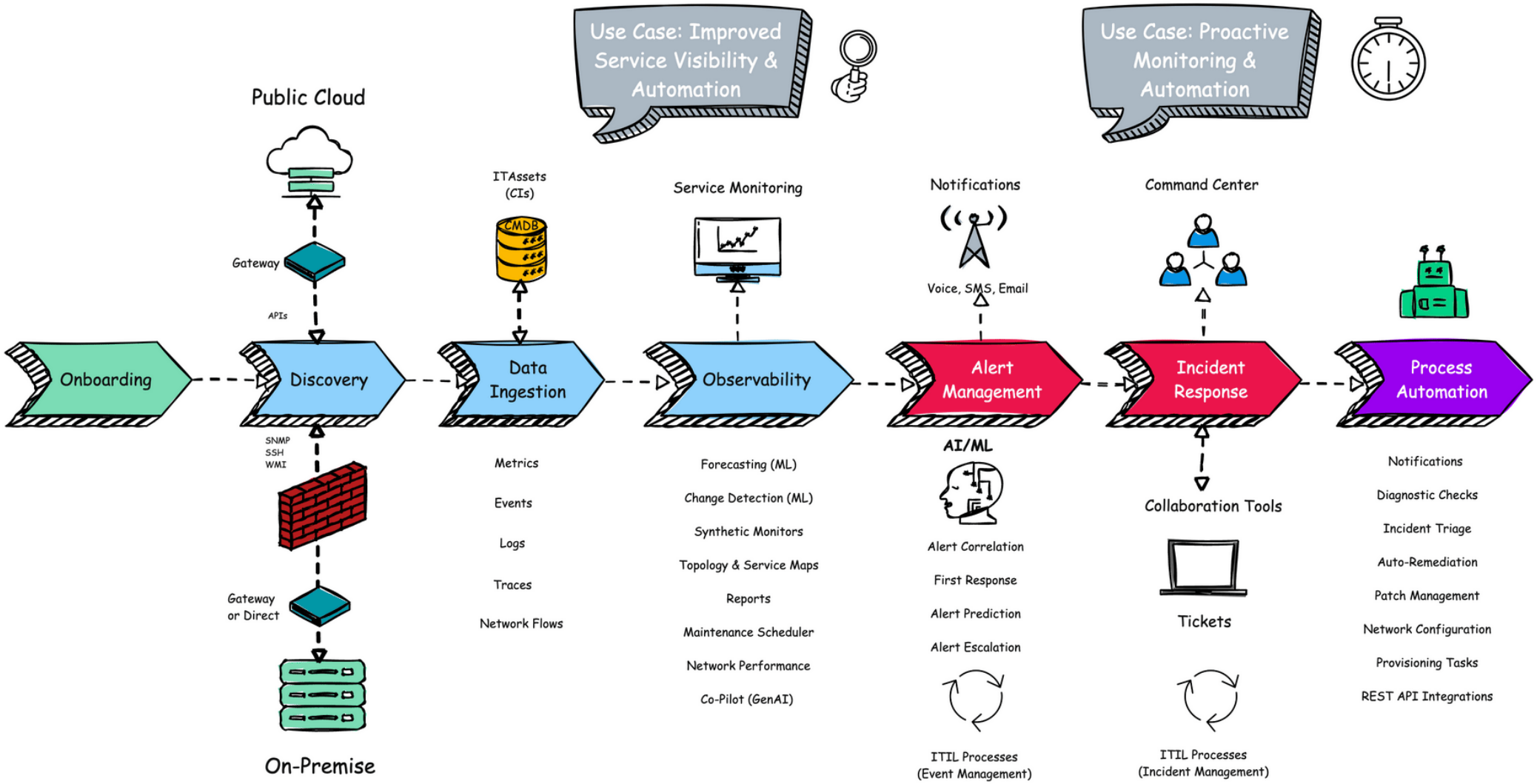
Current Observability Approach



Drill down to logs & traces

Correlated alerts across multiple telemetry sources (Metrics, Logs & Traces)

Day 2 Operations Workflow



Challenges Today

OpsRamp

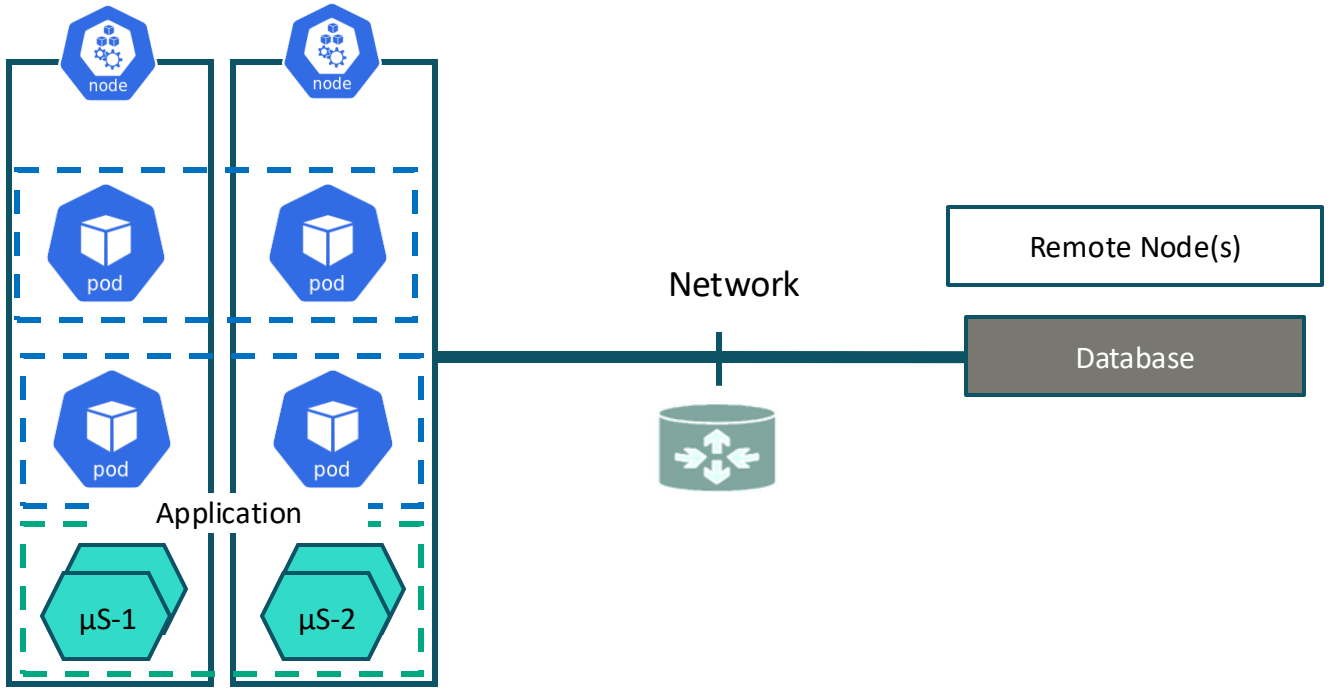
a Hewlett Packard Enterprise company

Challenges of Traditional Observability

Key Points:

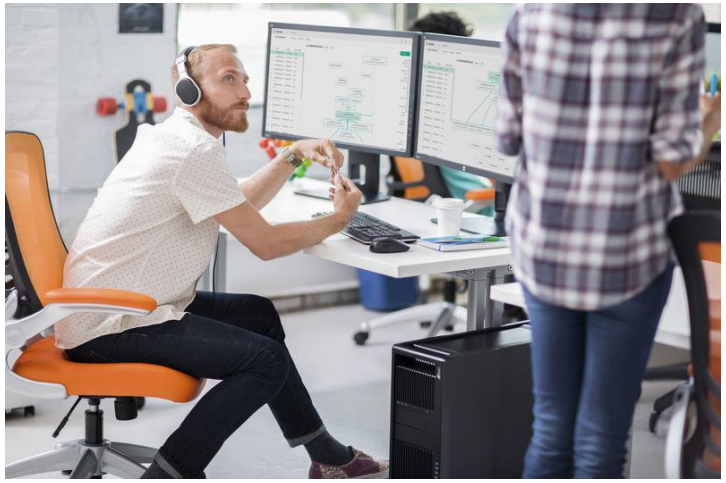
- Metrics, events, logs, and traces are great but provide limited context when troubleshooting unknown problems
- Remote (Agentless) plus Agent-based monitoring can add performance overhead
- Requires modifying applications to instrument code. OTEL does help with this.
- Difficult to gain deep kernel and network visibility (e.g. topology maps)

Challenge: Visibility Gap



Issue?

Current discovery & dependency mapping techniques cannot show remote connections



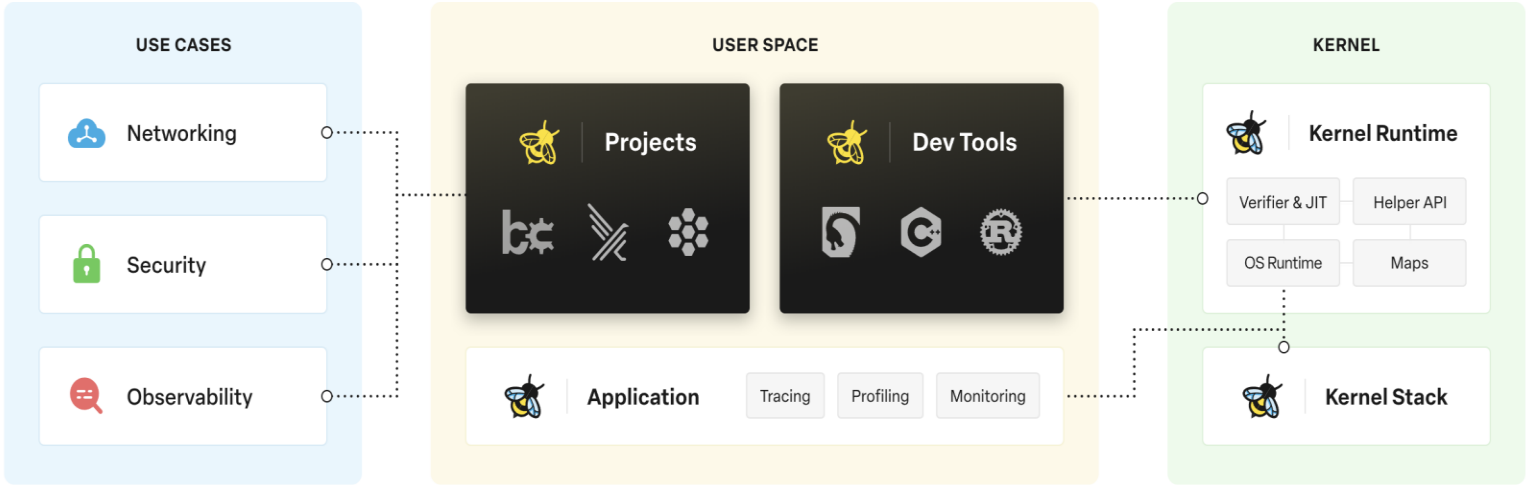
What is eBPF?

OpsRamp

a Hewlett Packard Enterprise company

What is eBPF?

- Extended Berkeley Packet Filter (eBPF)
- A technology that allows running sandboxed programs inside the Linux kernel
- Originally designed for packet filtering, now used for observability, security, and networking
- Runs without changing application code



How eBPF Works & Differences

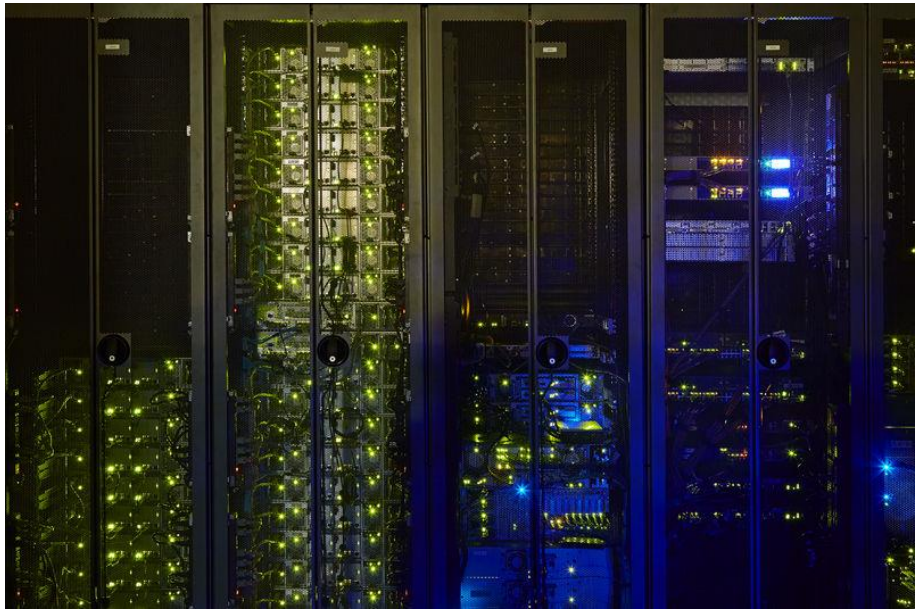
Core Concepts:

- eBPF programs run in response to kernel events
- Safe execution via verification and sandboxing
- Hooks into different kernel subsystems (networking, tracing, security, etc.)
- Low overhead, runs in-kernel

Feature	Traditional Monitoring	eBPF-Based Observability
Requires Agents	Yes plus agentless remote monitoring	No
Kernel-Level Visibility	Limited	Full
Performance Overhead	High (Potentially)	Low
Requires Code Changes	Yes	No
Real-Time Insights	Delayed	Immediate

How eBPF Transforms Observability

- Real-time visibility **without modifying applications**
- **Low-overhead** performance profiling
- **Security monitoring** (syscall tracking, anomaly detection)
- **Network observability** (packet inspection, connection tracking)
- Debugging and tracing application behavior **in production**

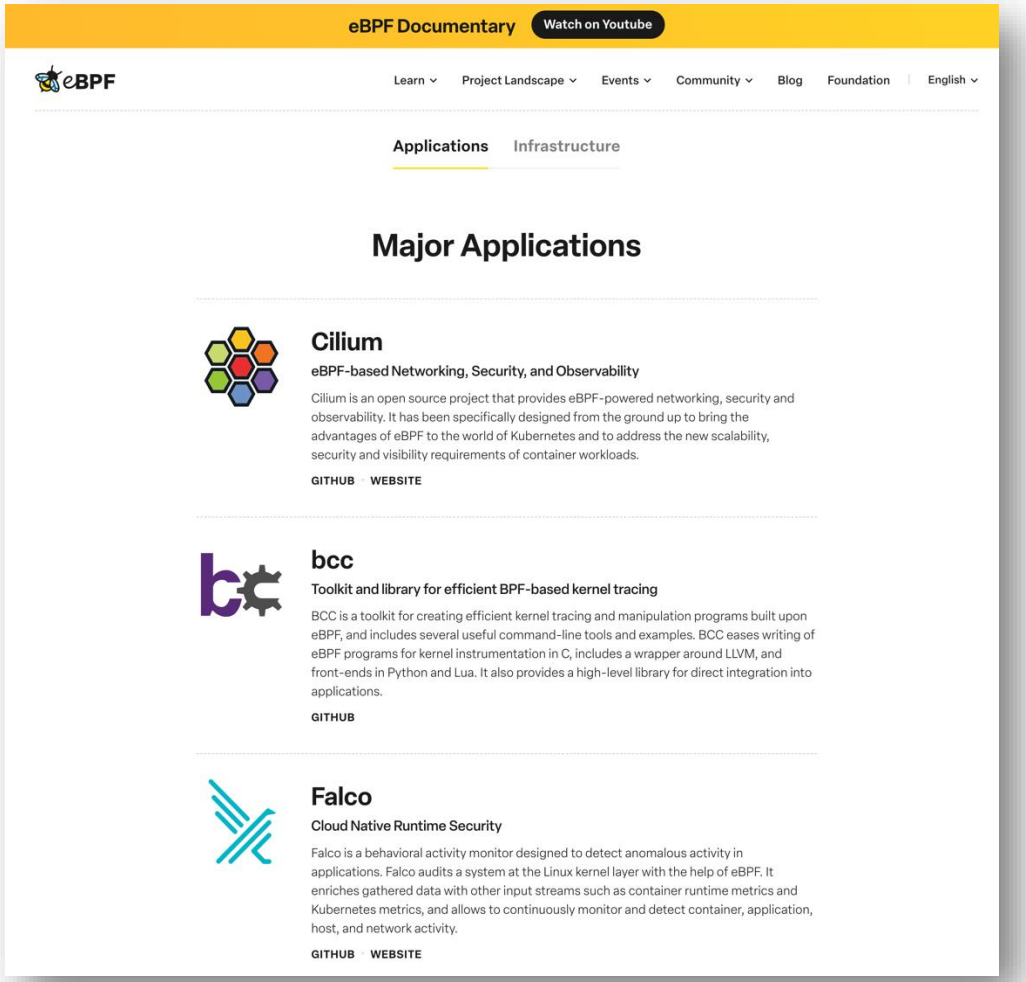


eBPF Tools

Sample Tools Landscape

- **BCC (BPF Compiler Collection):** eBPF tracing toolkit
- **bpftrace:** High-level tracing with one-liners
- **Cilium:** Kubernetes networking & security
- **Falco:** Runtime security monitoring

https://ebpf.io/applications/



eBPF Use-Cases

OpsRamp

a Hewlett Packard Enterprise company

Practical eBPF Use Cases

Performance Monitoring	Network Observability	Security Monitoring
CPU & memory profiling	Capture and analyse network traffic without packet loss	Track system calls for anomaly detection
Identifying slow "syscalls"	Monitor DNS, HTTP, TCP/UDP connections	Detect privilege escalation & malicious activity
Detecting high-latency operations	Identify latency bottlenecks in microservices	**Example**: Preventing unauthorised file access
Example: Profiling a database workload	**Example**: Debugging Kubernetes service-to-service communication	

Real-World Example: Debugging a Slow Database Query

Let’s say a database query is **intermittently slow**, but traditional logs and metrics don’t explain why.

Observability With Current Methods (e.g. OTEL):

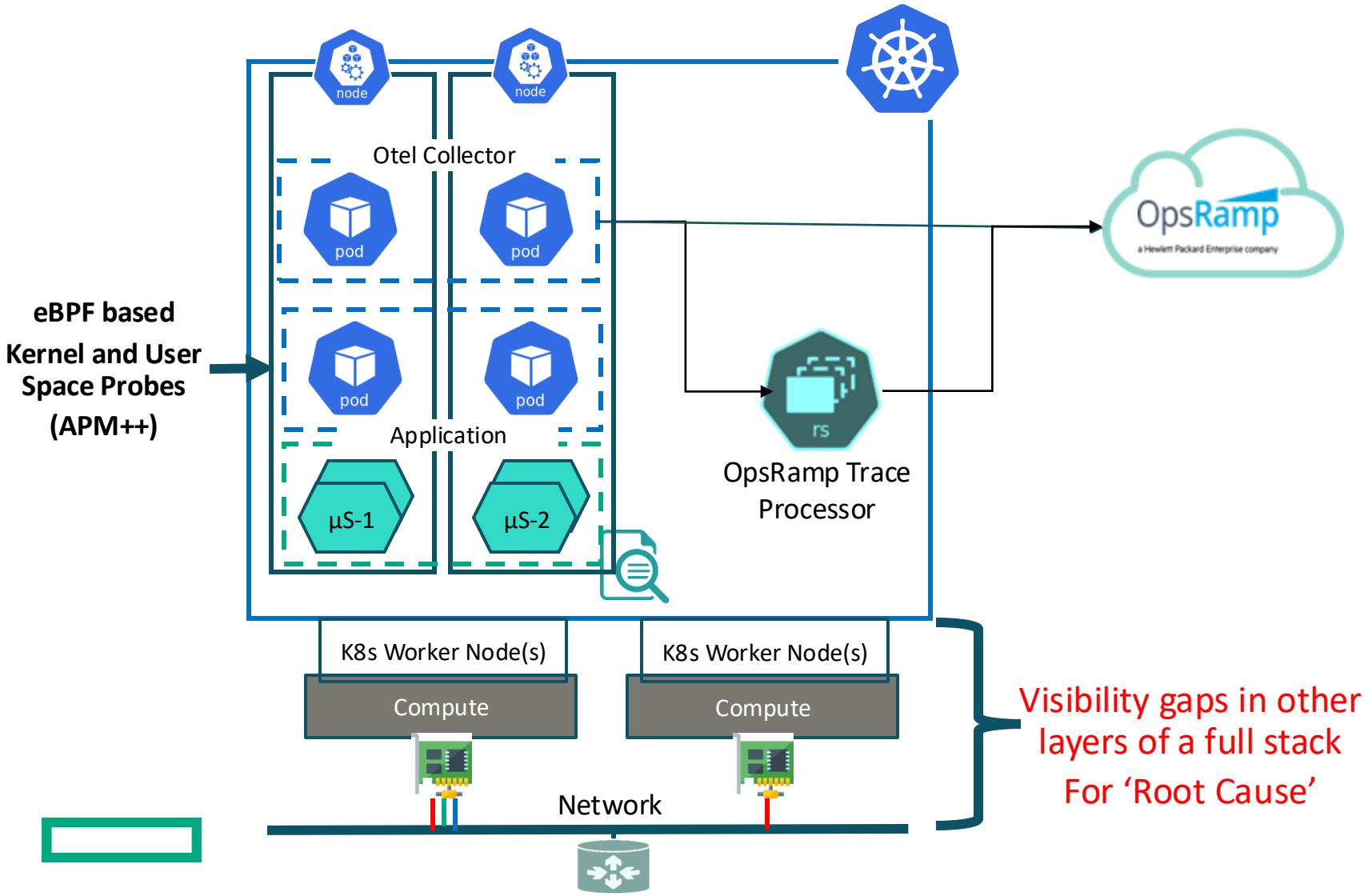
- **Logs:** Query executed at 10:05 AM, completed at 10:10 AM (but why was it slow?)
- **Metrics:** CPU, memory, disk usage look normal.
- **Traces:** Show long query execution time but no deeper insights.

Observability With eBPF:

- **Syscall Tracing:** Reveals the query spent **90% of its time waiting on disk I/O**.
- **I/O Latency Monitoring:** Shows the disk latency spiked due to high contention.
- **Kernel Profiling:** Indicates that a background process was competing for disk access.
- **Solution:** eBPF pinpointed the issue to disk I/O contention—something that traditional monitoring **wouldn't have detected** without custom instrumentation.

eBPF based Observability – Architecture and Use Cases

OpsRamp
a Hewlett Packard Enterprise company

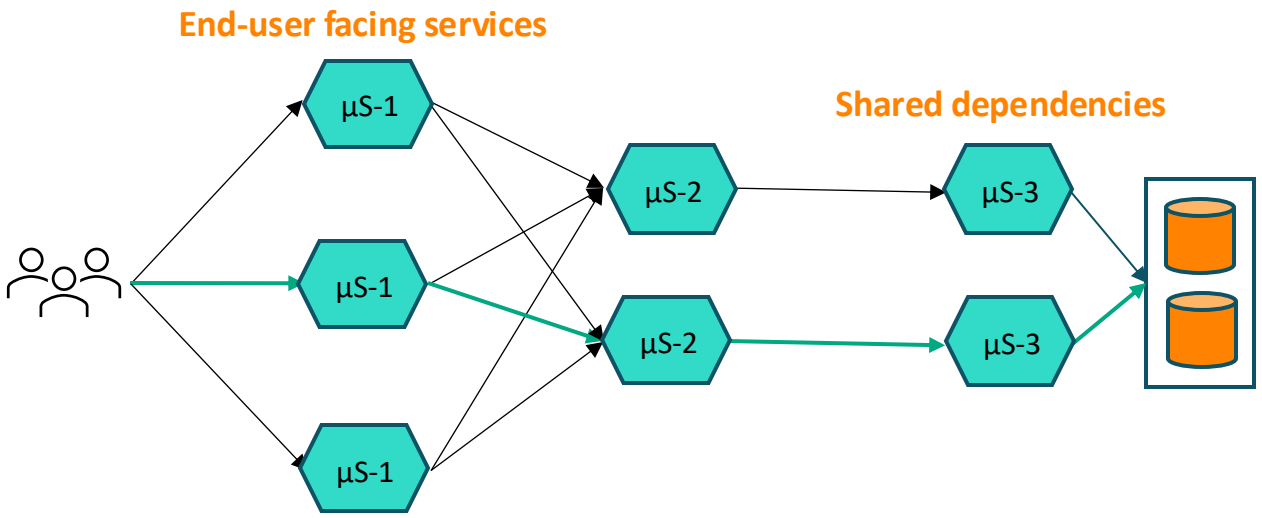


1.1 Workload service map without APM (aka tracing)

1.2 Reasoning on application-level transactions based on network latencies, round-trip times, etc.

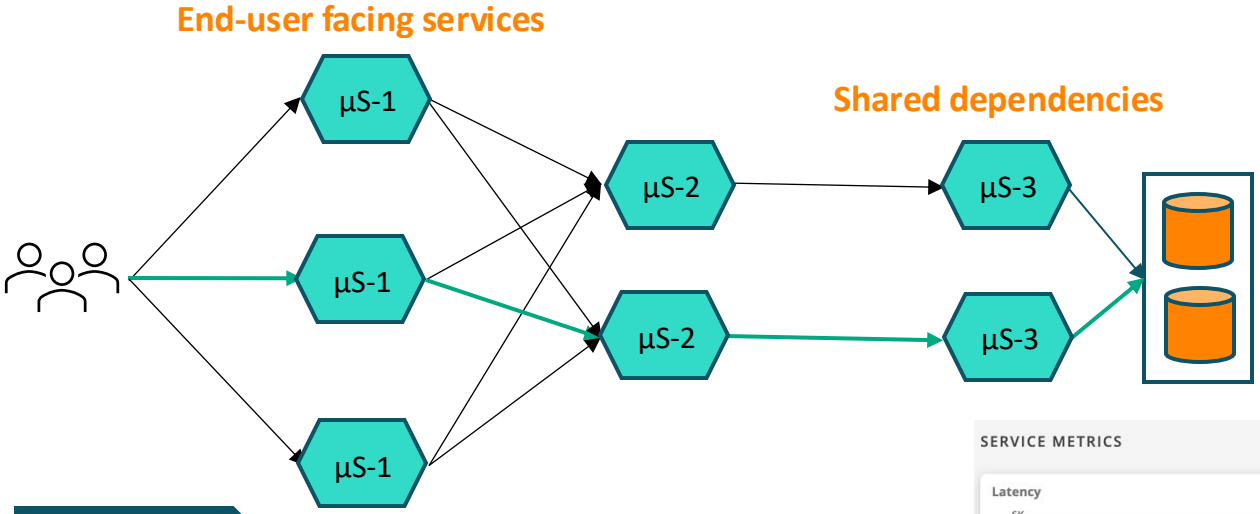
1.3 Per workload and internal and **external** communication visibility and patterns

1.1 Workload service map without APM (aka without tracing)

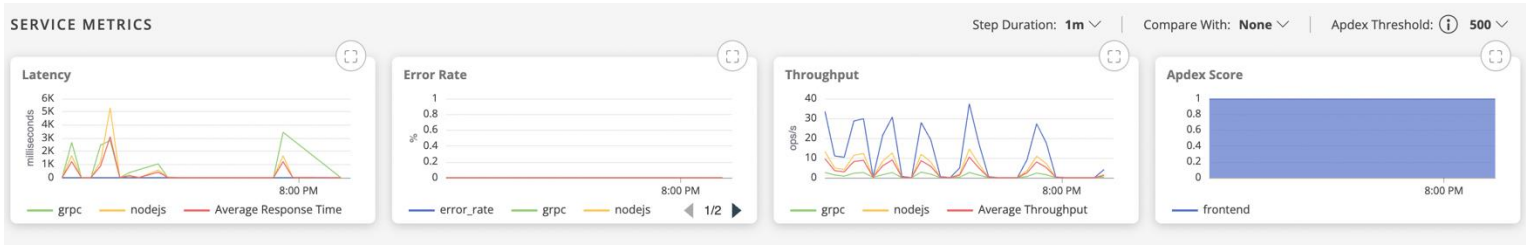


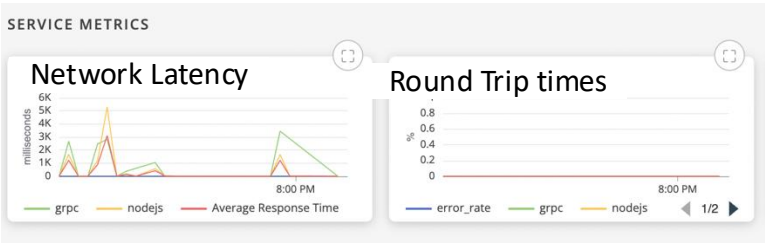
1.2 Reasoning on workload transactions based on network latencies, round-trip times, etc.

OpsRamp
a Hewlett Packard Enterprise company

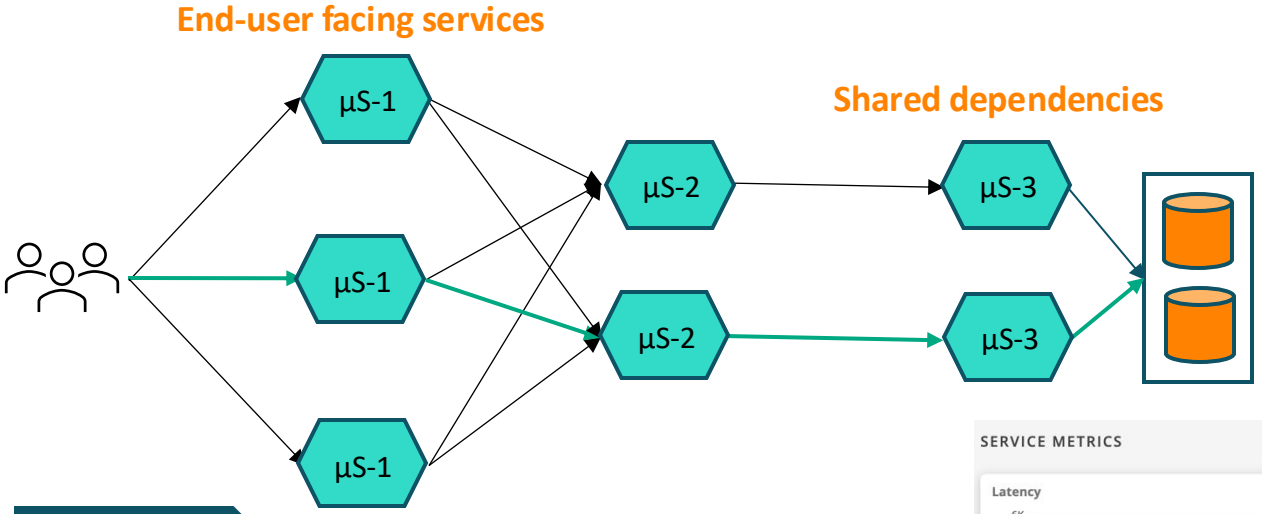


Time Window (10 mins)





1.3 Per Workload internal versus external communications

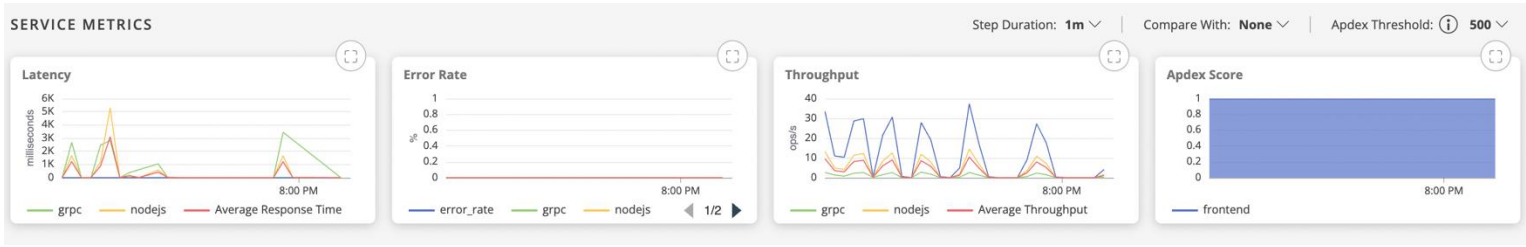


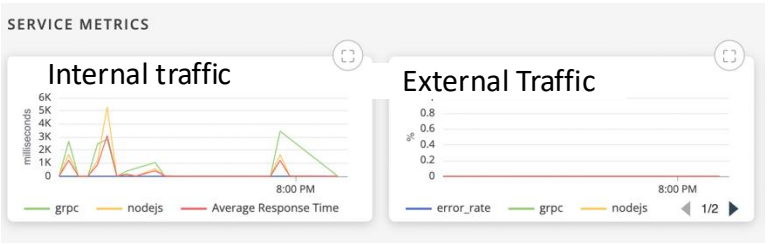
Workload Layer — Latency, Errors, Throughput

Network Layer — Internal, External

Storage Layer — Attach, Network

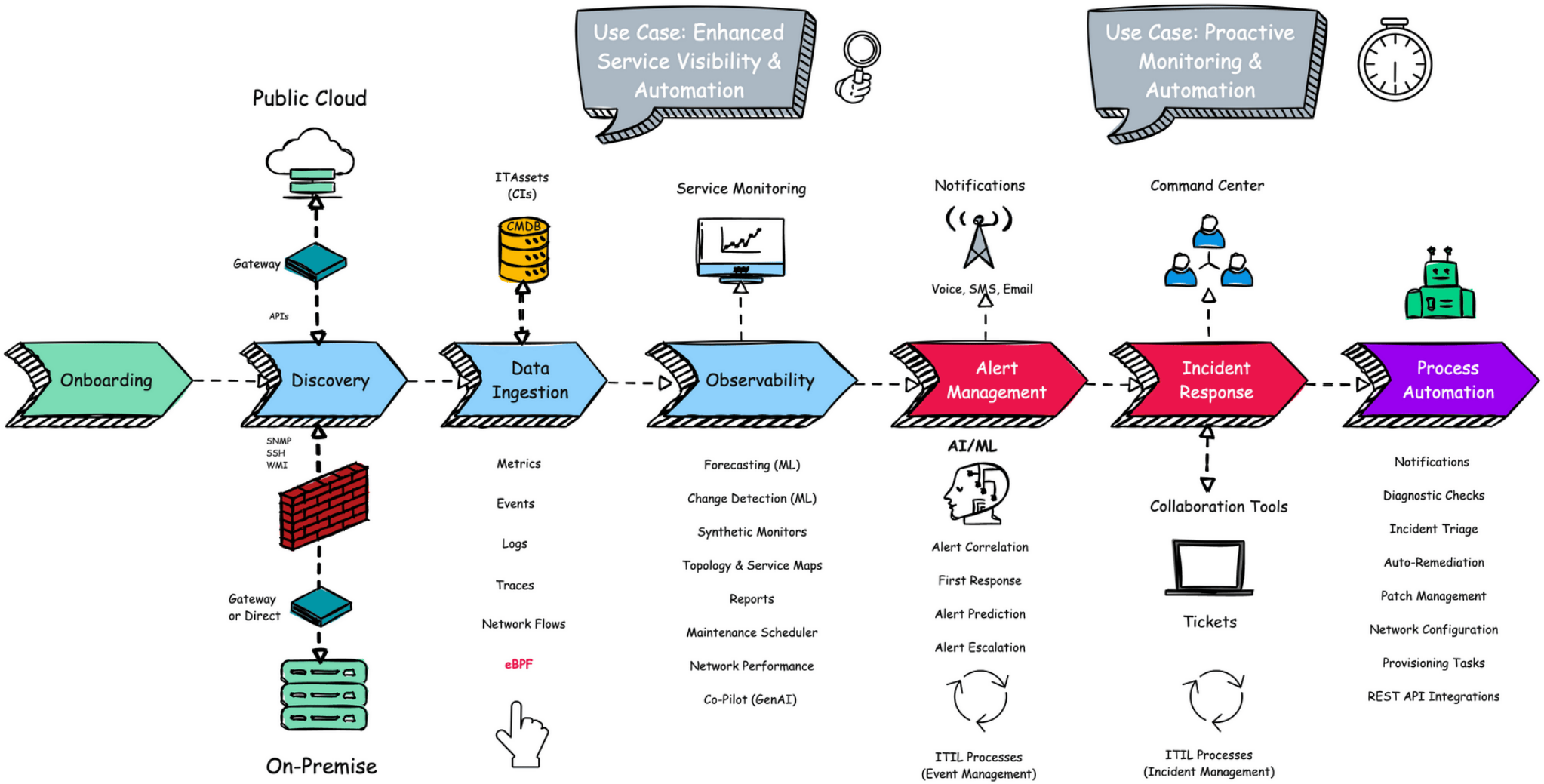
Time Window (10 mins)





Day 2 Operations with eBPF

OpsRamp
a Hewlett Packard Enterprise company



OpsRamp Demo with eBPF Telemetry

OpsRamp

a Hewlett Packard Enterprise company

Getting Started with eBPF Tools

eBPF support is present in Linux kernel 4.4 and later. However, for optimal eBPF functionality, you should be running a more recent version of the kernel (e.g., 5.x or later).

Ubuntu Installation

sudo apt update
sudo apt install bpfcc-tools linux-headers-$(uname -r) clang llvm gcc make
sudo apt install bpftrace

sudo bpftrace -e 'tracepoint:syscalls:sys_enter_execve { printf("execve called with path: %s\n", str(args->filename)); }'

```
devops@lon-dc1-app-s1-ul24:~$ sudo bpftrace -e 'tracepoint:syscalls:sys_enter_execve { printf("execve called with path: %s\n", str(args->filename)); }'
Attaching 1 probe...
execve called with path: ./create_files.sh
execve called with path: /usr/bin/mkdir
execve called with path: /usr/bin/touch
execve called with path: /usr/bin/touch
execve called with path: /usr/bin/touch
execve called with path: /usr/bin/touch
execve called with path: /usr/bin/touch
execve called with path: /usr/libexec/tracker-extract-3
```

OpsRamp Demo with eBPF Telemetry

Use-Case: Monitor deleted files

Trace Deleted Files Program

```
// Attach the eBPF program to a tracepoint
tp, err := link.Tracepoint("syscalls",
"sys_enter_unlinkat",
coll.Programs["trace_unlinkat"], nil) if err
!= nil { log.Fatalf("Failed to attach
tracepoint: %v", err) } defer tp.Close()
```

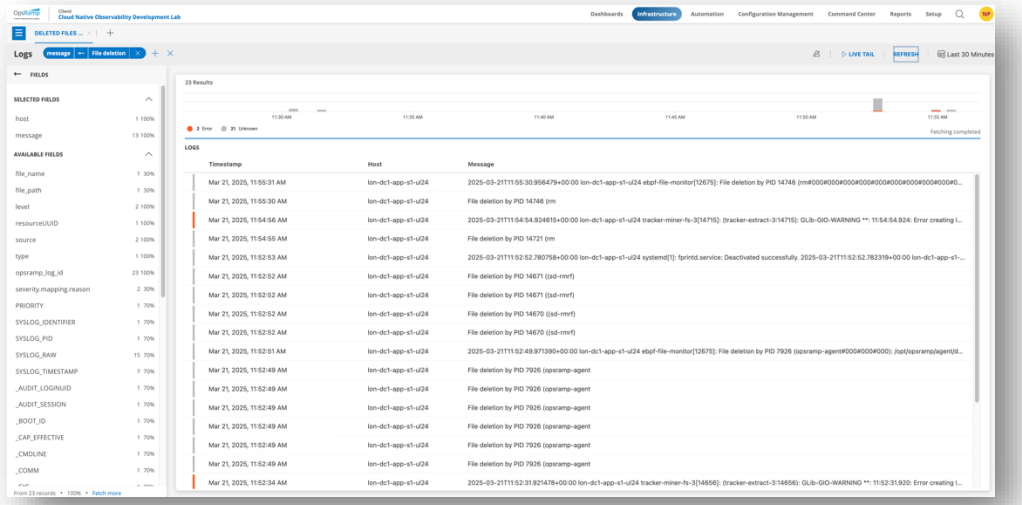
```
# sudo go run ebpf-demo.go
```

Write to Syslog

```
msg := fmt.Sprintf("File deletion by PID
%d (%s): %s", event.Pid, event.Comm,
event.Filename) fmt.Println(msg) // Print
to stdout as well sysLogger.Info(msg) //
Write to syslog
```

Observe & Mange

Metrics | Dashboards | Alert



OpsRamp Log Ingestion

Current Limitations of eBPF & Resources

OpsRamp

a Hewlett Packard Enterprise company

Limitations

- **Complexity:** Requires deep kernel and system knowledge to implement effectively.
- **Compatibility Issues:** Not all Linux distributions support eBPF fully.
- **Security Risks:** Poorly written eBPF programs can introduce vulnerabilities.
- **Debugging Difficulty:** Lack of robust debugging tools for eBPF programs.
- **Resource Consumption:** Though low-overhead, excessive use of eBPF programs can impact system performance.
- **Windows Support:** eBPF is primarily designed for Linux; Windows support is still evolving with limited functionality.
- **Mobile OS Support**: Android currently in development

Resources

Linux kernel documentation: The official Linux kernel documentation provides a great starting point to understand the technicalities and implementation of eBPF in the kernel. The documentation covers topics like BPF type, APIs, and how to use eBPF in different contexts.
https://docs.kernel.org/

BPF Compiler Collection (BCC) Documentation: BCC is a set of tools and libraries for interacting with eBPF. The documentation offers detailed information on how to use BCC to write and load eBPF programs.
https://github.com/iovisor/bcc

"BPF Performance Tools" by Brendan Gregg:
This is one of the most comprehensive books available on eBPF and performance monitoring with eBPF. It covers a wide range of topics, including networking, tracing, and security, and provides many practical examples using tools like bpftrace and bcc.
https://www.oreilly.com/library/view/bpf-performance-tools/9780136588870/

"Linux Observability with BPF" by David Calavera, Lorenzo Fontana
This book focuses on using eBPF for system observability, including detailed explanations of tracing, debugging, and performance monitoring. It also contains practical examples and use cases.
https://www.oreilly.com/library/view/linux-observability-with/9781492050193/

Summary

OpsRamp

a Hewlett Packard Enterprise company

Key Takeaways

- eBPF enables **deep, real-time observability** without overhead
- Eliminates blind spots in performance, security, and networking
- Open-source tools make eBPF accessible for monitoring & troubleshooting
- Day 2 transformation of observability is **kernel-native, event-driven, and zero-instrumentation**

Future of eBPF in Observability

- Expanding adoption in Kubernetes and cloud-native environments (Initial OpsRamp Focus)
- More tools integrating eBPF for security and performance
- Growing ecosystem and standardization efforts
- Unlikely to replace OTEL

Calling all HPE GreenLake Flex Solutions customers (or their CSM / ASM)

- OpsRamp subscription key is included with each HPE GreenLake Flex Solutions
- Getting started:
  - https://support.hpe.com/hpesc/public/docDisplay?docId=a00120892en_us&page=GUID-9EDAAB42-9182-488D-A06F-6E8CB4BFAB60.html
  - https://docs.opsramp.com/guides/getting-started/
- Getting help:
  - hpedev@hpe.com
  - #hpe-greenlake-flex-observability in HPEDEV Slack

OpsRamp
a Hewlett Packard Enterprise company

Thank you

Do you have ideas or suggestions surrounding eBPF integration with OpsRamp?

Contact your local HPE CSM or ASM to arrange a meeting with an Observability specialist.