KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon CloudNativeCon
THE LINUX FOUNDATION OPEN SOURCE SUMMIT
AI_dev Open Source GenAI & ML Summit
China 2024

KubeSkoop - Deal with the Complexity of Network Issues and Monitoring with eBPF

Yutong Li, Alibaba Cloud
Bingshen Wang, Alibaba Cloud

Agenda

- The complexity of Kubernetes Networking
- Introduction to KubeSkoop
- KubeSkoop Network Diagnosis based on eBPF

Alibaba Cloud Kubernetes Service(ACK)

30+	tens of thousands	10k+
Regions	Clusters	Nodes in single cluster

KubeCon | CloudNativeCon

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev Open Source GenAI & ML Summit

China 2024

The Complexity of Kubernetes Networking

in Concepts

in Implementations

Concepts in Kubernetes

The network concepts of Kubernetes lead to the complexity of networking configuration:

Ingress/Service/NetworkPolicy

- LabelSelector selects unexpected pods.
- Overlapping of multiple NetworkPolicy rules.
- NATed service ports do not match the real pod ports.

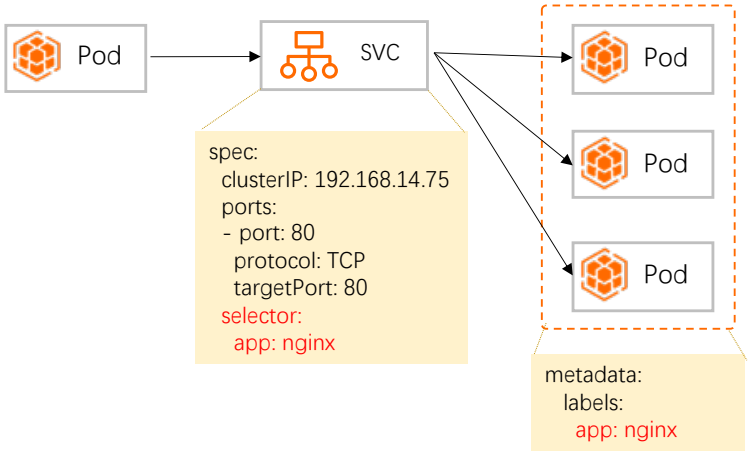
ServiceMesh

- More complicated Layer 7 network strategies

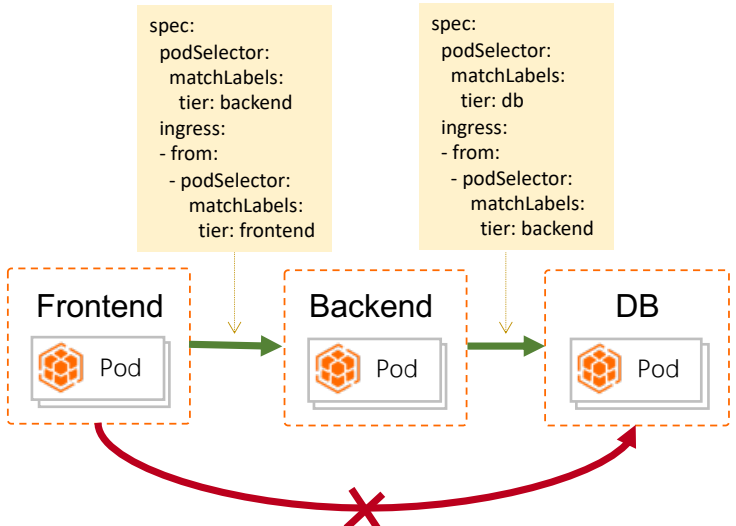
Third-party Networking/Ingress Plugins

- Provide custom networking extensions.

Service Discovery



NetworkPolicy



Implementation of Container Networking

Complexity of data-plane:

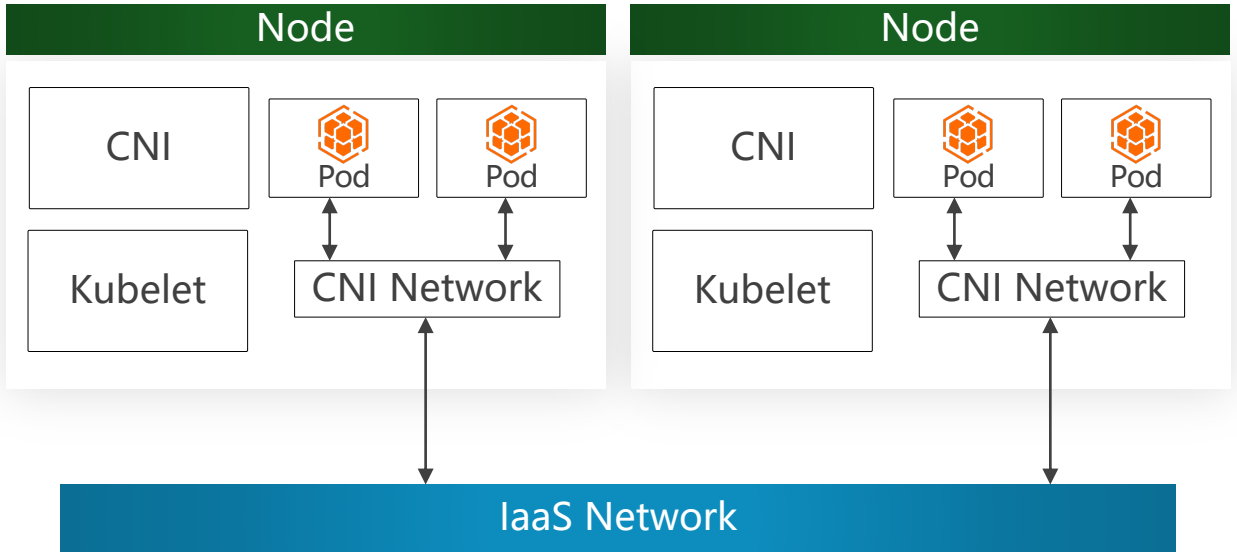
- ServiceMesh/KubeProxy/CNI
- CNI Implementations (Overlay/Underlay…)

Complexity of the network stack

- Long data path, including NIC drivers/ netfilter/route/bridge etc.
- Complicated networking configuration

Complexity of underlay network:

- Different configurations per cloud provider
- Security groups, route tables, etc.



Traditional Network Troubleshooting

1 Long Issue Diagnosis Process:

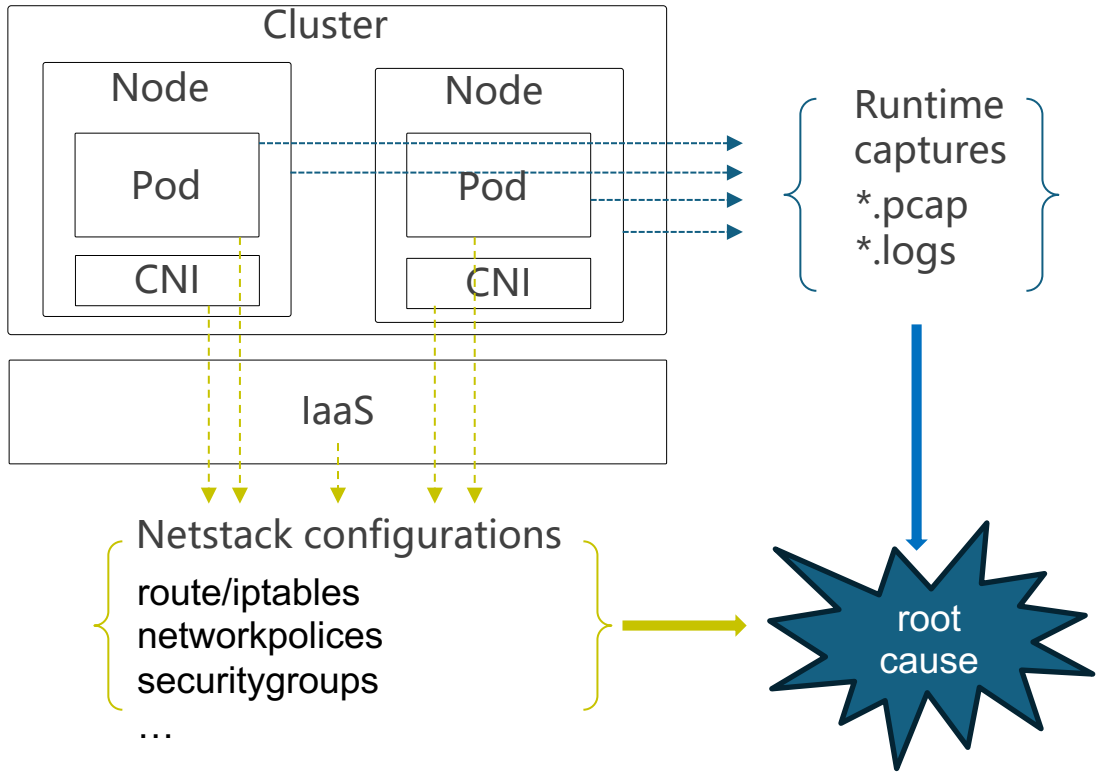
Capture packets -> Compare and analyze ->
Check configurations at packet loss points

2 Long Diagnosis Time:

Stress Testing to Reproduce the problem ->
Analyze the problem with a large number of information

3 High Experience Requirements:

Mastery of the Linux protocol stack, CNI implementation, and IaaS layer network configuration.




KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Introduction to KubeSkoop

Introduction to KubeSkoop

Network monitoring & diagnosis suite for Kubernetes

Connectivity Diagnosis

Anomaly Tracing

Flow Logs

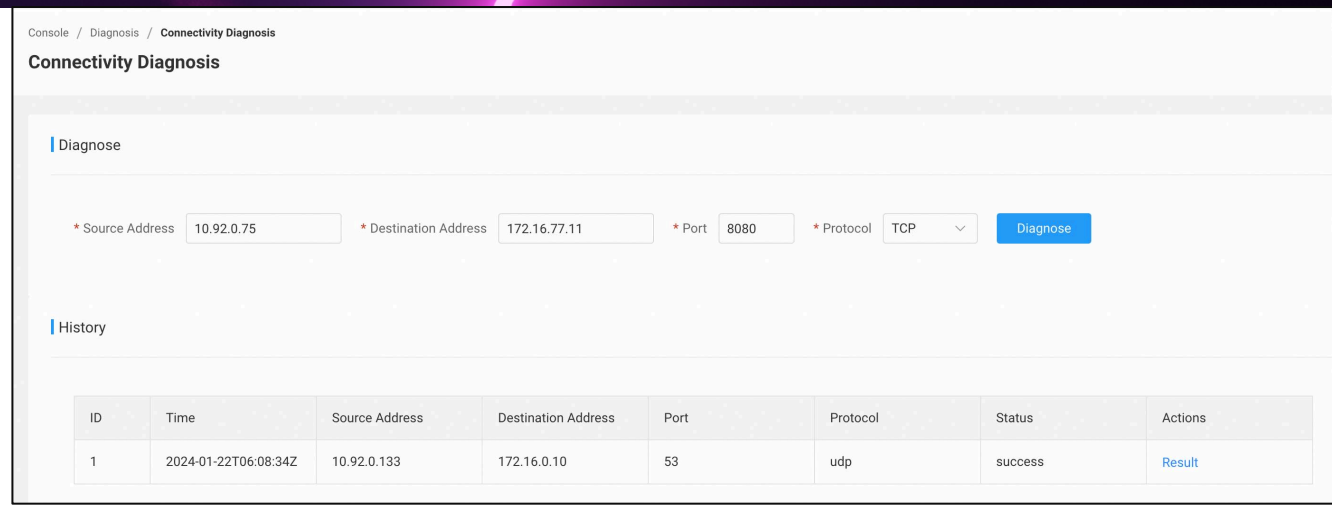
Latency Detection

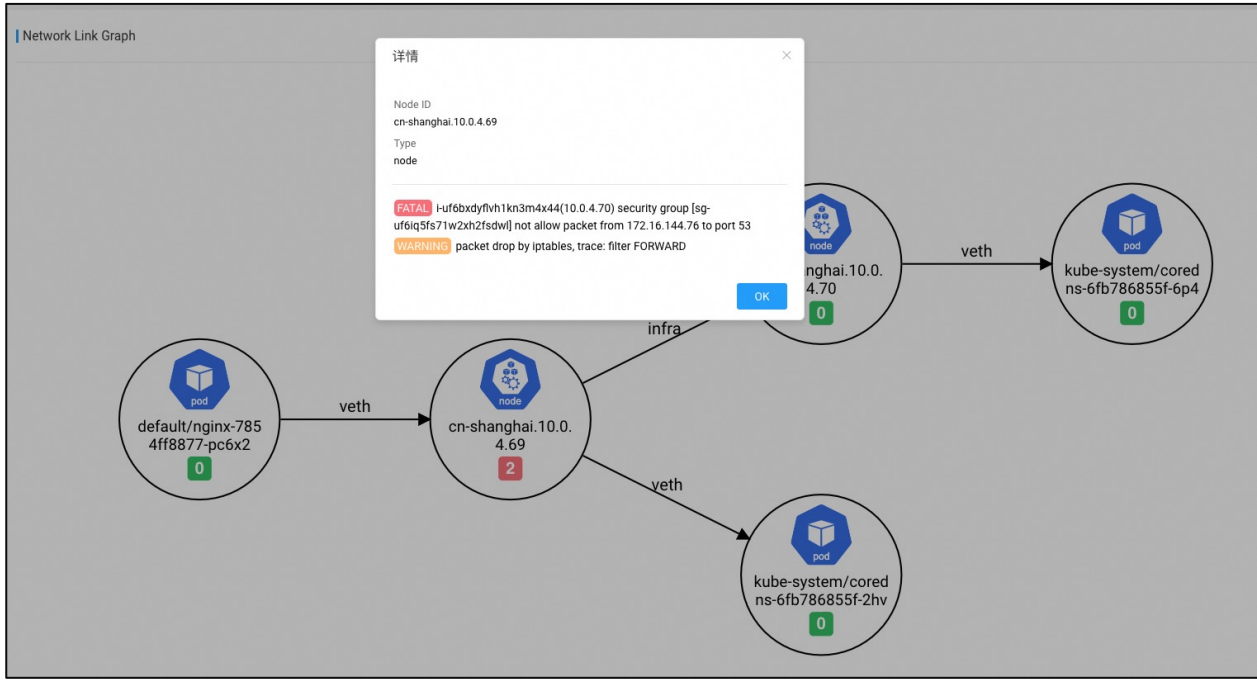
Packet Capturing

https://github.com/alibaba/kubeskoop

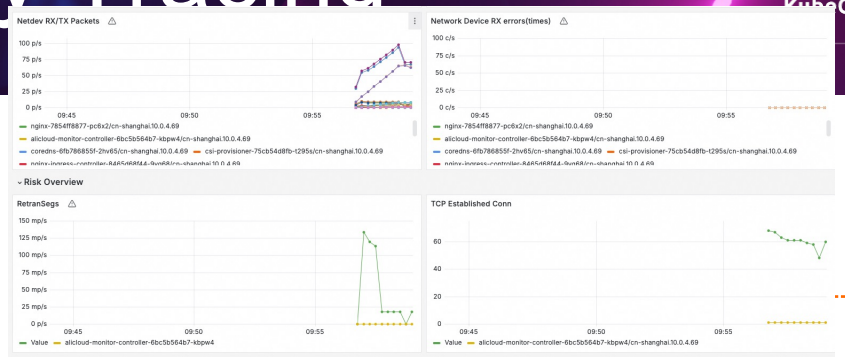
Connectivity Diagnosis

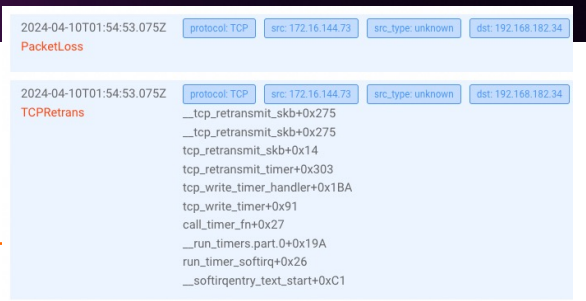
- Specify the source and destination address of the problem. Wait for the diagnostic output.
- Constructs network links and analyzes the problem automatically.
- Including analysis of Kubernetes Service, NetworkPolicy, etc.
- Analysis of kernel network stack and underlying IaaS configuration.
- No need the troubleshooting experience of CNI implementations and complicated network problems.





Monitoring and Anomaly Tracing





Container Network Anomaly Monitoring

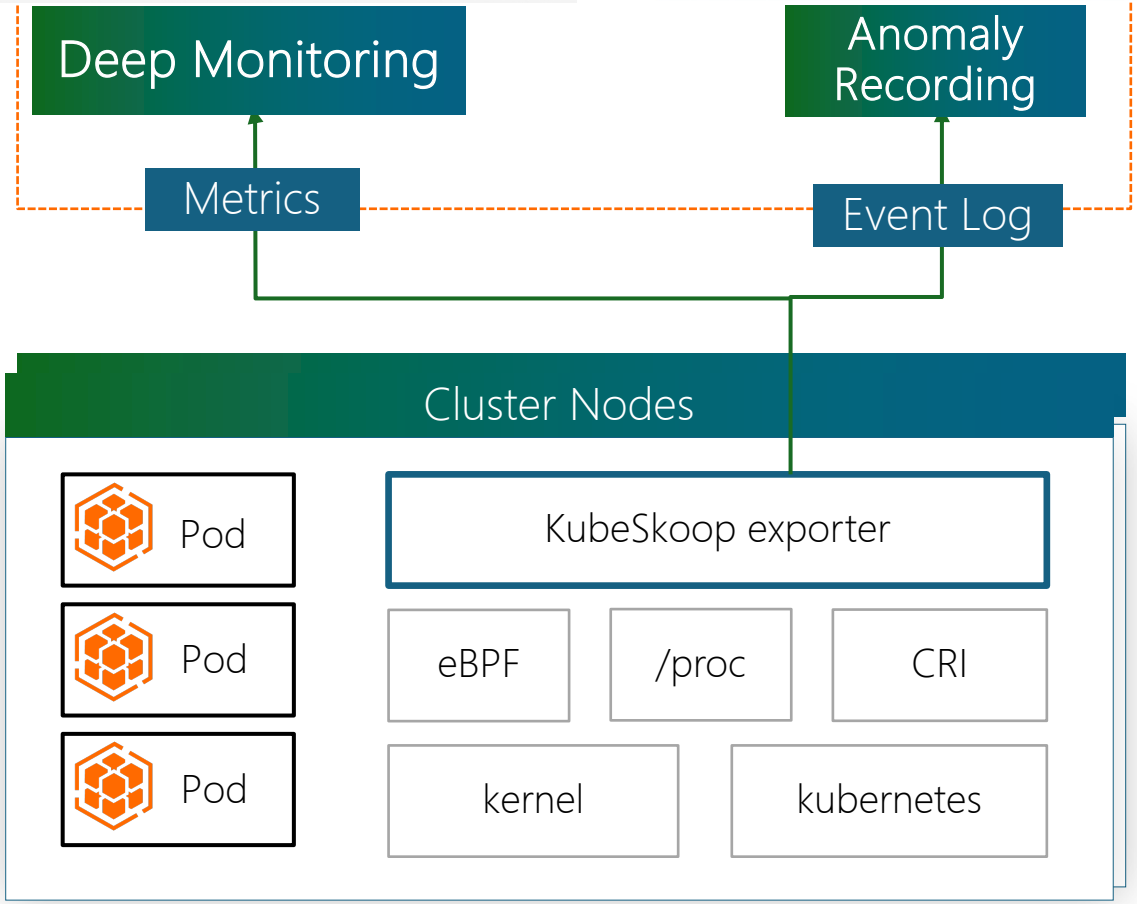
- Cloud-native deployment, work with observability systems like Prometheus
- Pod-level network monitoring
- Lightweight, low-overhead kernel anomaly tracing

Covering Multiple Scenarios

- Occasional packet loss and retransmission
- Flow-level anomaly identification
- Network latency analysis

Exposing Multiple Types of Anomalies

- Metrics for network traffic and relationships
- Event logging of network anomalies



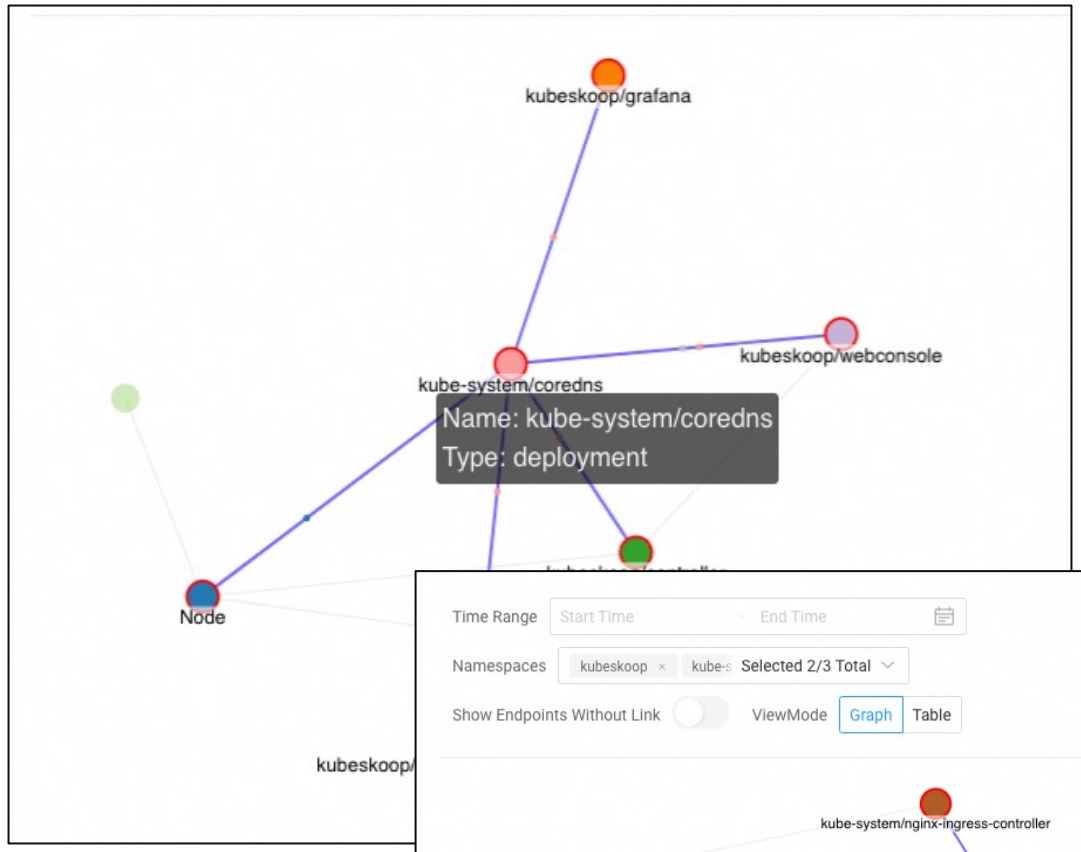
Access Flow

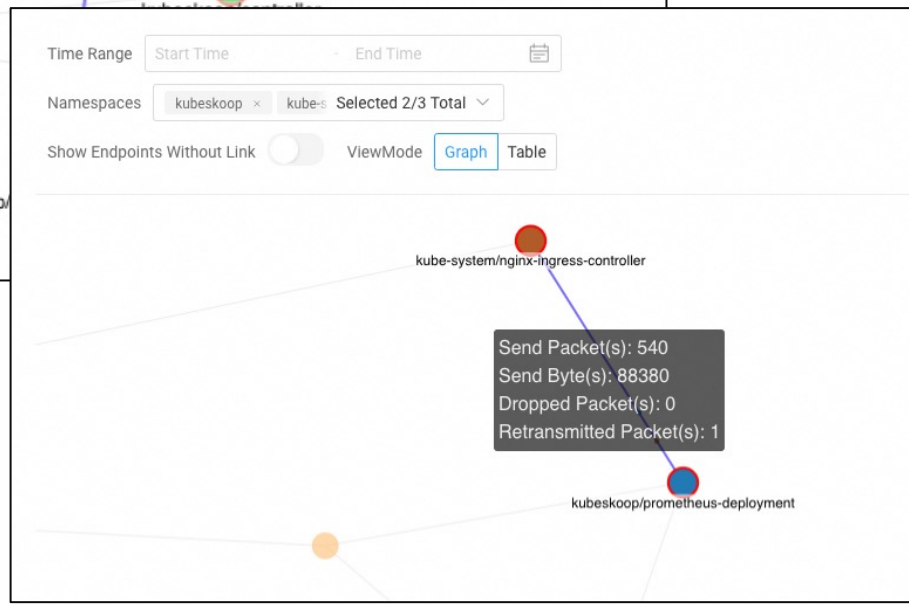
Historical access relationships recording and backtracing

- Recording connections, packets and throughtput
- Analyze network performance and bottleneck of applications
- Troubleshooting historical resource usage of shared services (e.g. CoreDNS)

Low-overhead access relationship capturing

- Access flow capture based on eBPF

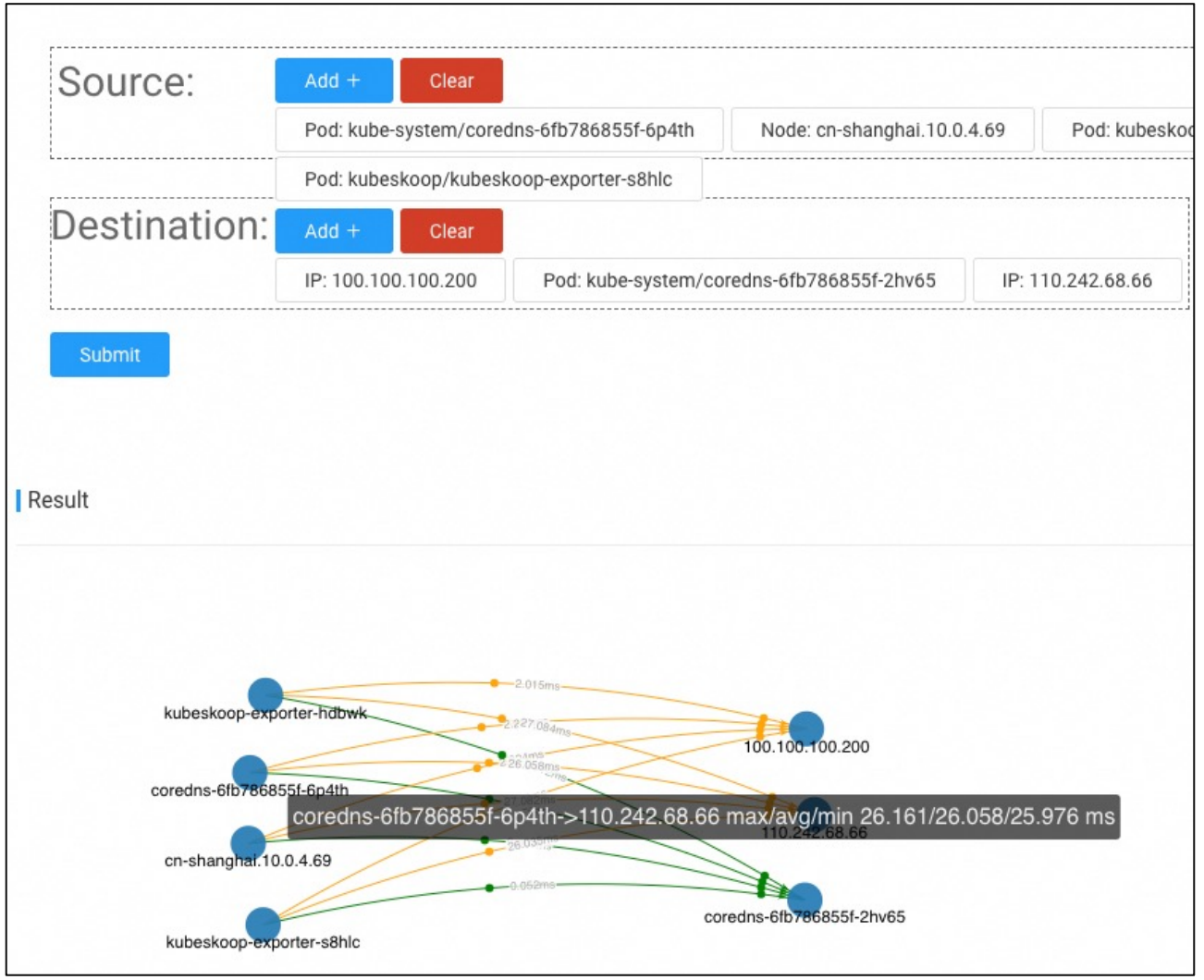




Latency Detection(PingMesh)

Latency Detection in Cluster

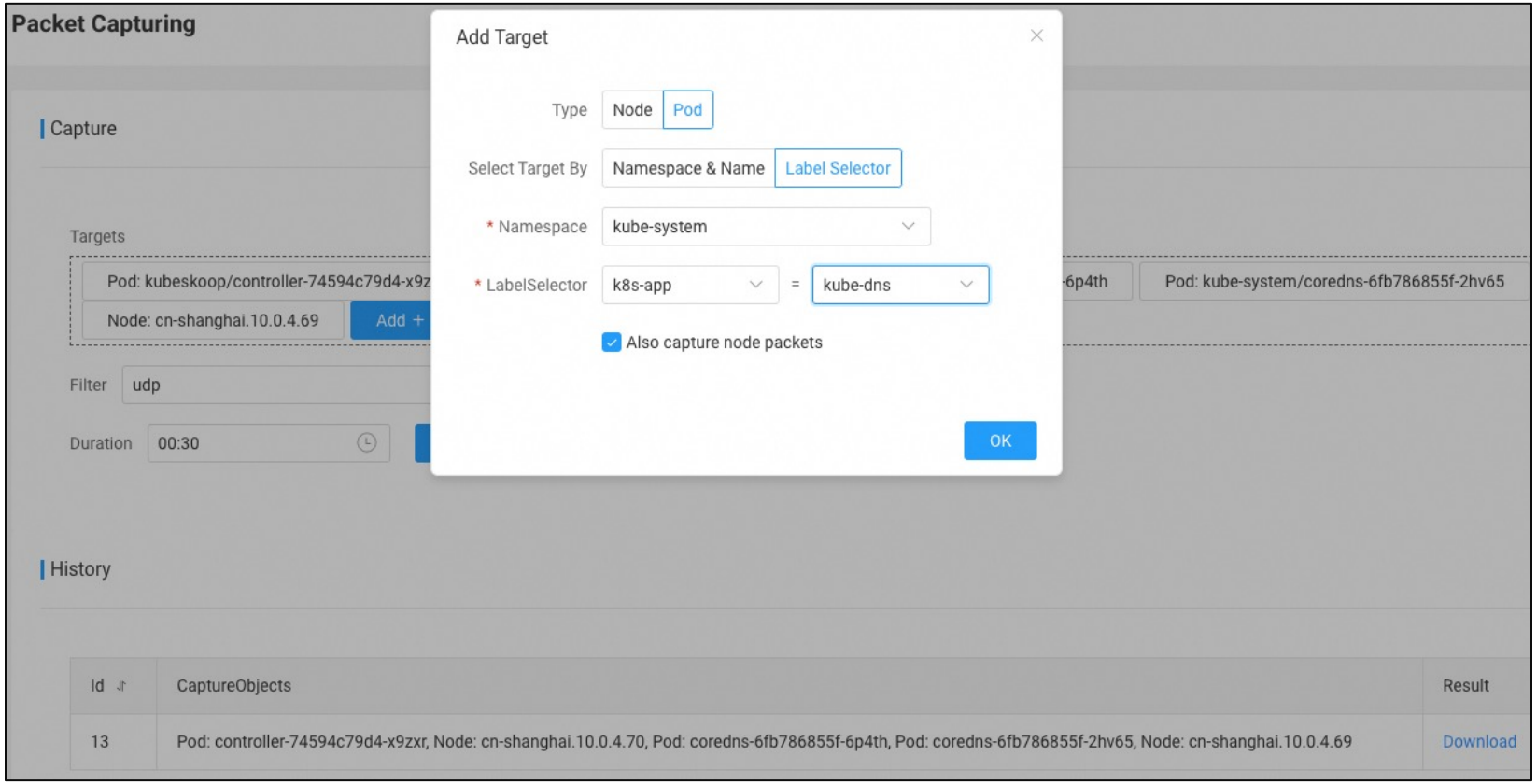
- Generate latency report in one click
- Troubleshooting performance bottlenecks
- Optimize application deployment topology

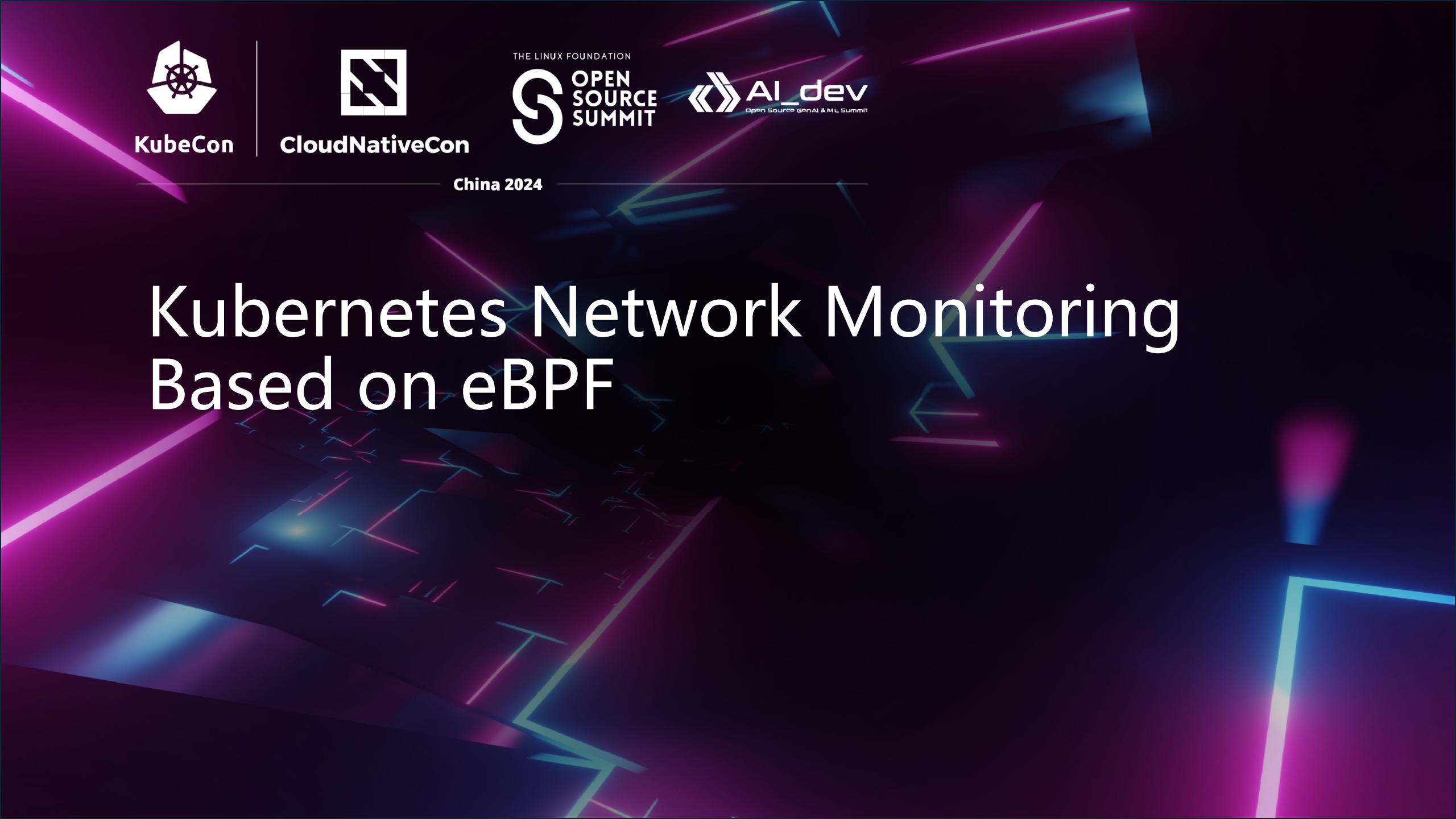


Packet Capturing

Capture packets on multiple nodes simultaneously

- One-click packet capture for all links you need
- Locate packet loss point for unknown reasons

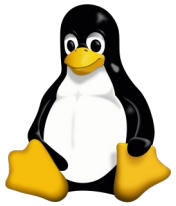



KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Kubernetes Network Monitoring Based on eBPF

Why choose eBPF

Kubernetes uses a lot of network components:

- TCP
- Bridge
- Netfilter
- TC
- …



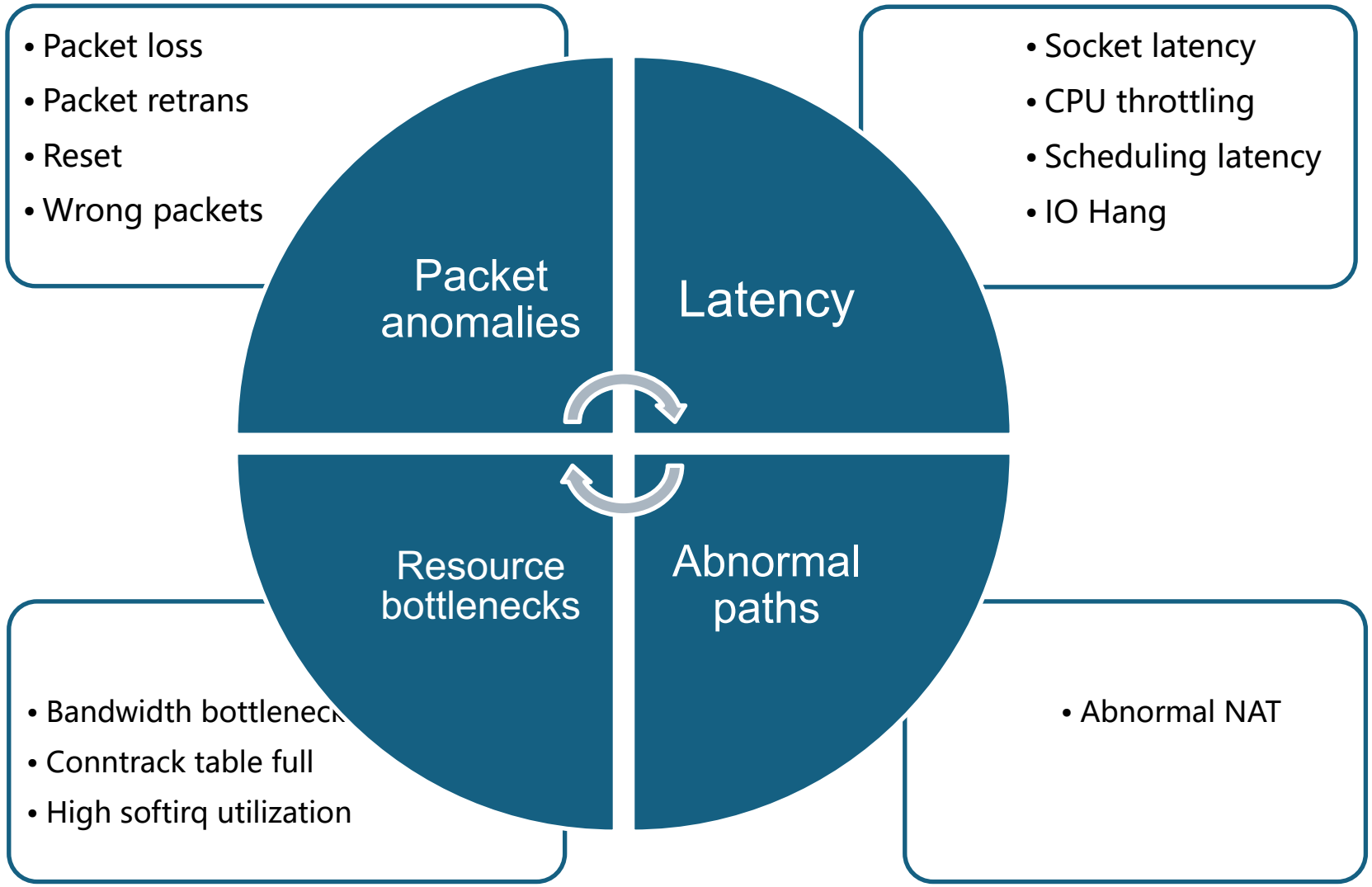
The information exposed by the kernel is limited:

- Lack of netns information
- Missing records for critical links
- Absence of context information
- …

Convenience of eBPF program deployment and distribution:

- Good compatibility
- High security
- Low overhead for kernel-level filtering and statistics
- Creating a reproducible troubleshooting experience.

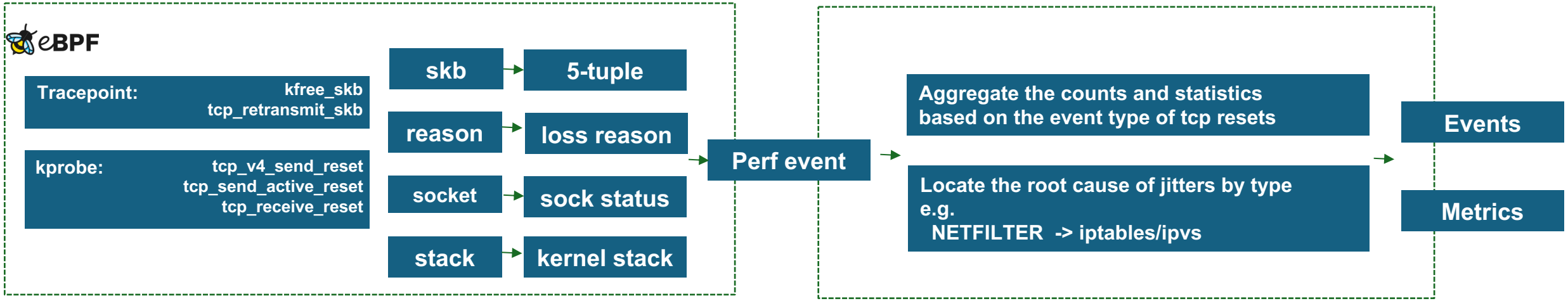
Key Characteristics of Kubernetes Network Issues



eBPF Monitoring for Packet Anomalies

Packet anomalies causing occasional business jitter
- Packet loss & retransmissions
- TCP resets

Using eBPF allows for tracking the call stack, thereby identifying the root cause of packet loss.

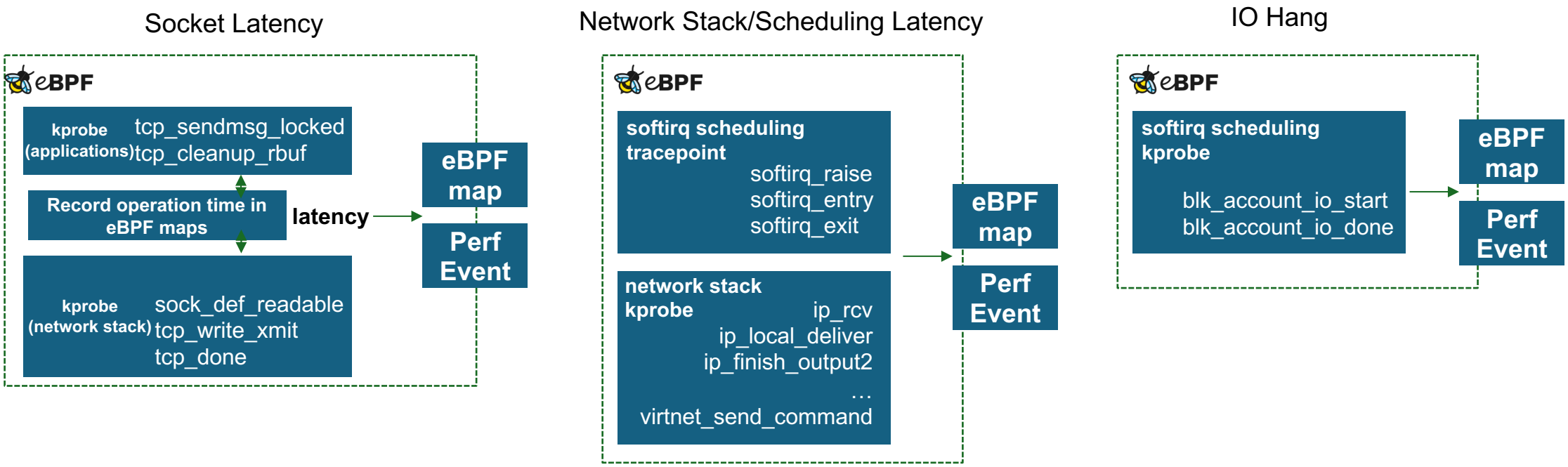


eBPF Monitoring for Network Latency

Network latency issues:

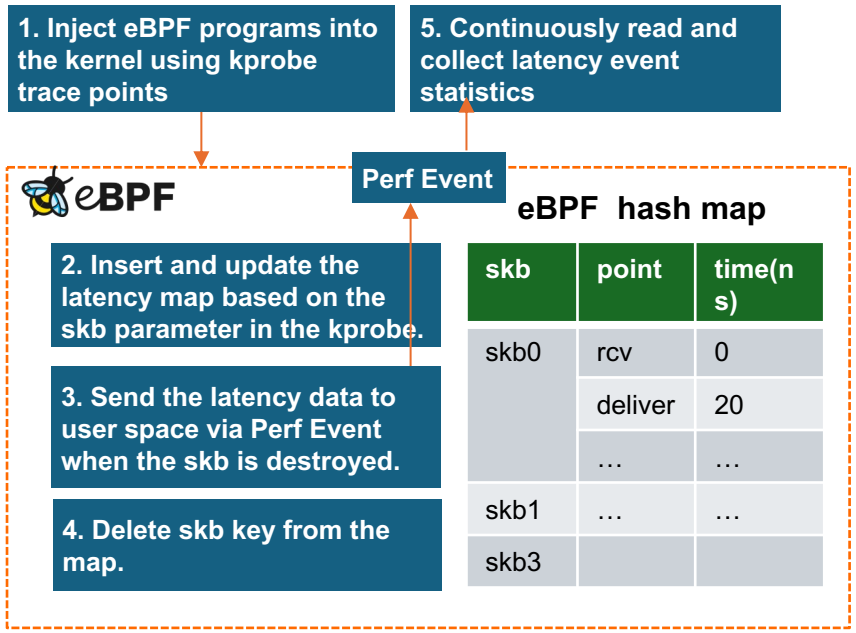
- Socket latency
- Network stack/scheduling latency
- IO hang

eBPF uses skb and pid as keys for tracking, collecting data at various points to identify the root cause of latency.



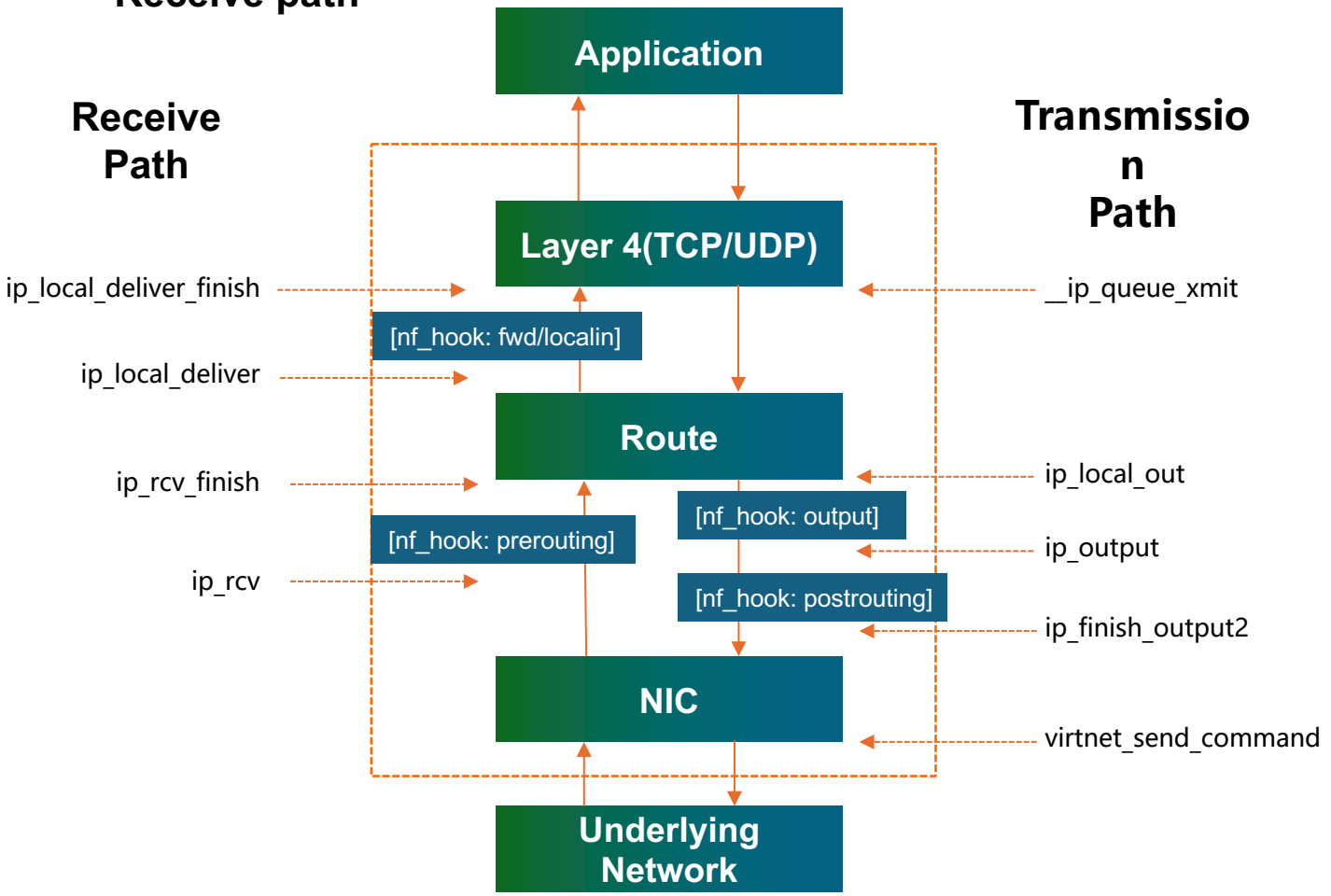
eBPF Monitoring for Network Latency

eBPF Network Stack Latency Detection



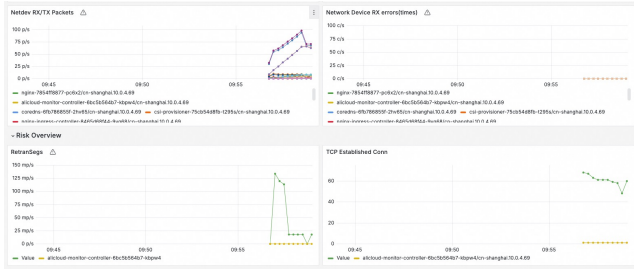
Latency Detection for The Whole Path through the Kernel

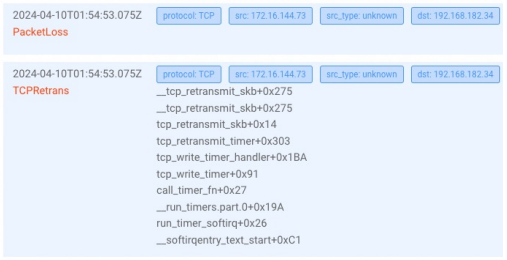
- Transmission path
- Receive path

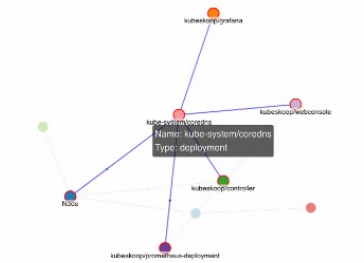


eBPF Metrics Collecting and Processing in KubeSkoop

	Metrics Latency Anomaly count	Events Event log Call stack	Flow 5-tuple Statistics
Collect	Tracepoint kprobe & uprobe	Tracepoint kprobe	TC sockops
Sync	eBPF Map	Perf Event	eBPF Map
Store	Prometheus Time Series	Loki Log Stores	Prometheus Time Series
Visualize	Grafana Dashboard	Event Log Backtrack	Flow Graph







KubeSkoop eBPF Monitoring Case Study

KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

Troubleshooting Root Cause of Business Jitter: Occasional Timeouts When Accessing Ingress NGINX Controller

Backtracking Metrics

Event Logging Events

```
INFO  March  9 21:21:05 VIRTCMDEXCUTE nginx-ingress-
controller-5c7fb5594-jwhnw protocol=TCP
saddr=100.121.88.193 sport=27057 daddr=10.33.0.11
dport=443  stacktrace: virtnet_send_command+0x1
virtnet_set_rx_mode+0x251  __dev_change_flags+0x9c
dev_change_flags+0x21  do_setlink+0x257
__rtnl_newlink+0x600  rtnl_newlink+0x44
rtnetlink_rcv_msg+0x119  netlink_rcv_skb+0x4e
netlink_unicast+0x1d7  netlink_sendmsg+0x240
sock_sendmsg+0x5f  ____sys_sendmsg+0x232
___sys_sendmsg+0x75  __sys_sendmsg+0x49
do_syscall_64+0x30
entry_SYSCALL_64_after_hwframe+0x61
```

Business Anomaly Time Point: Around 21:25

Backtracking with KubeSkoop monitoring:
Increased retransmissions from business Pods.
Sudden spike in Virtnet command latency

Event Loggins reveals the call stack and root cause:
Underlying virtualization execution timeouts leads to soft interrupts being stuck, caused the packet retransmission.

Challenges of eBPF Program Management

Deployment and Distribution

- Dependency on kernel header file
- Large clang/llvm compilation environment (300MB+)

eBPF Probes Management

- Dynamic insertion and removal of dozens of eBPF programs
- Dynamic features configuration for each program

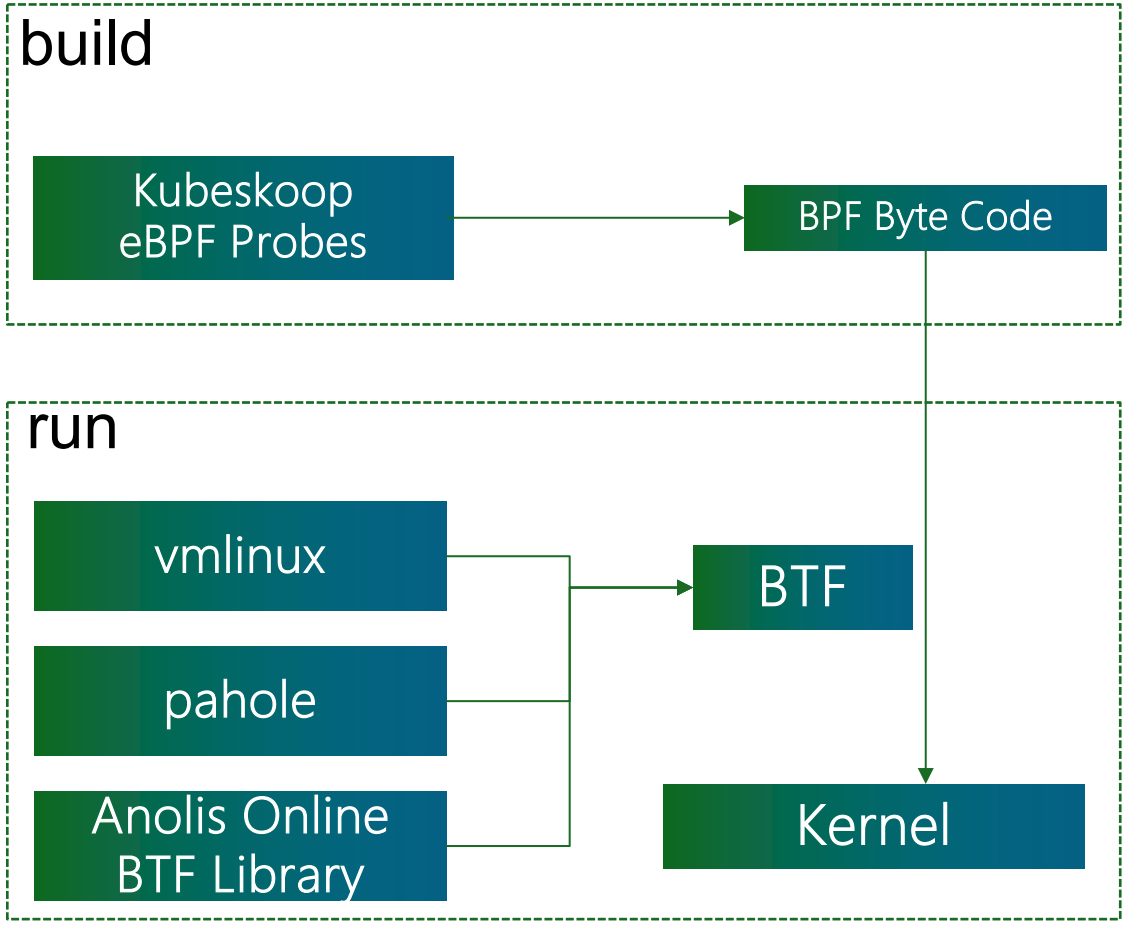
Effective Collection of Large Volumes of Data

- Different data labels across various programs
- Events and monitoring metrics need to be output to multiple storages

KubeSkoop eBPF Program Management

Leveraging eBPF CO-RE

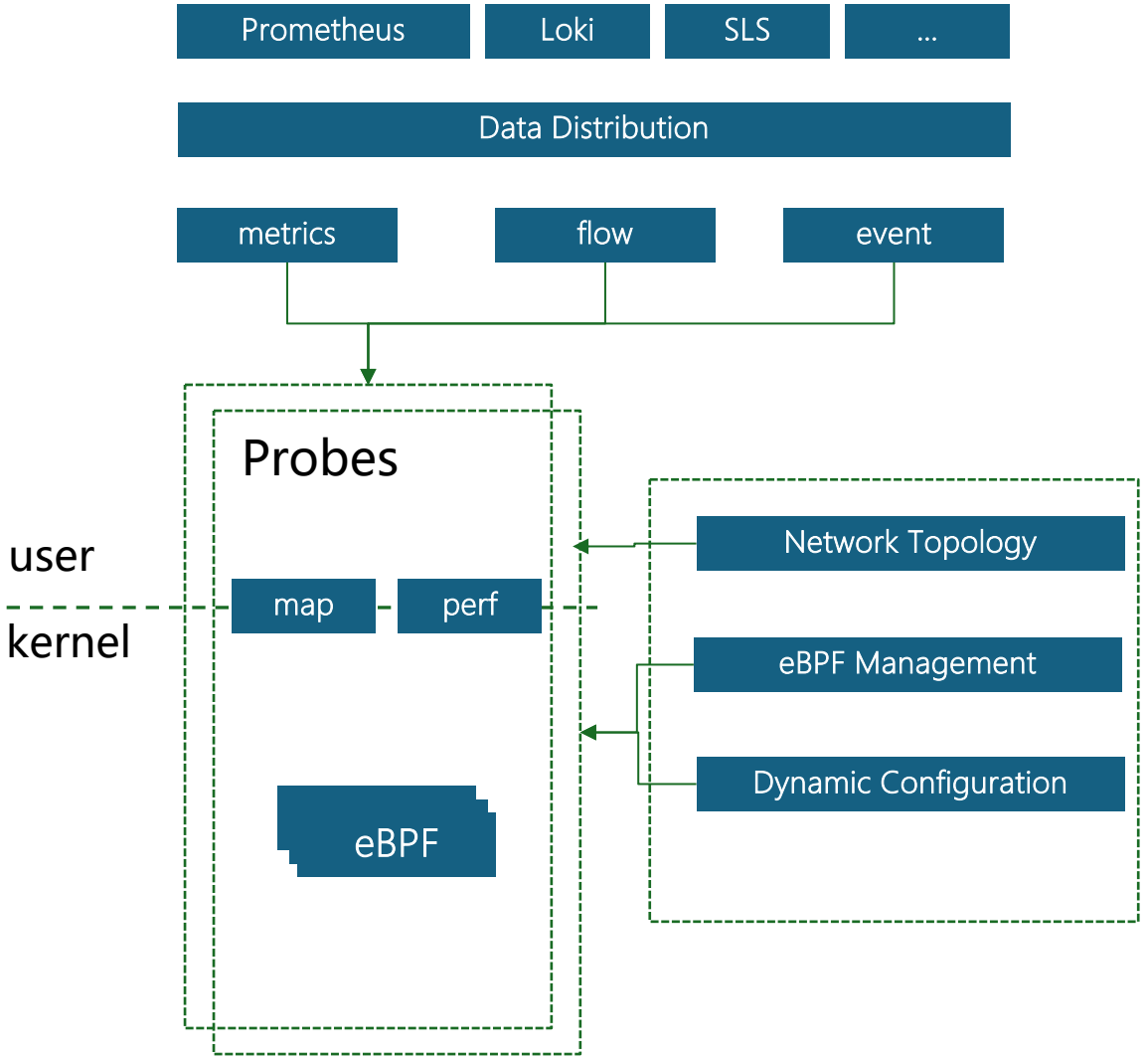
- No need for clang/llvm dynamic compilation
- Reduced OS header file and environment dependencies
- Use btfhack to obtain BTF files from various source
- Distribution size reduced from 300mb to 33mb



KubeSkoop eBPF Program Management

Prober Extensibility Design

- Abstract of Metrics、Event、Flow interface
- Generation of shared network topology information from CRI & netns
- Dynamic plugging and configuration of eBPF programs based on .rodata
- Unified data collection and distribution
- Simplified development efforts for adding new monitoring capabilities



Future Plan

- Support for more cloud providers and network plugins
- Enable display of RTT, window size, and other information in flow diagrams to identify network bottlenecks within the cluster
- Support multiple protocol types for latency detection, allowing simultaneous packet capture and analysis of latency paths
- Allow the extension of probe using bpftrace scripting language to reduce development barriers
- Provide the KubeSkoop Analysis tool to intelligently analyze KubeSkoop metrics and events, lowering the threshold for understanding issues
- Application layer awareness for perception and processing of Layer 7 protocols (e.g., HTTP, Redis, etc.)