Grafana Labs

Slash your cloud network costs with eBFP network monitoring

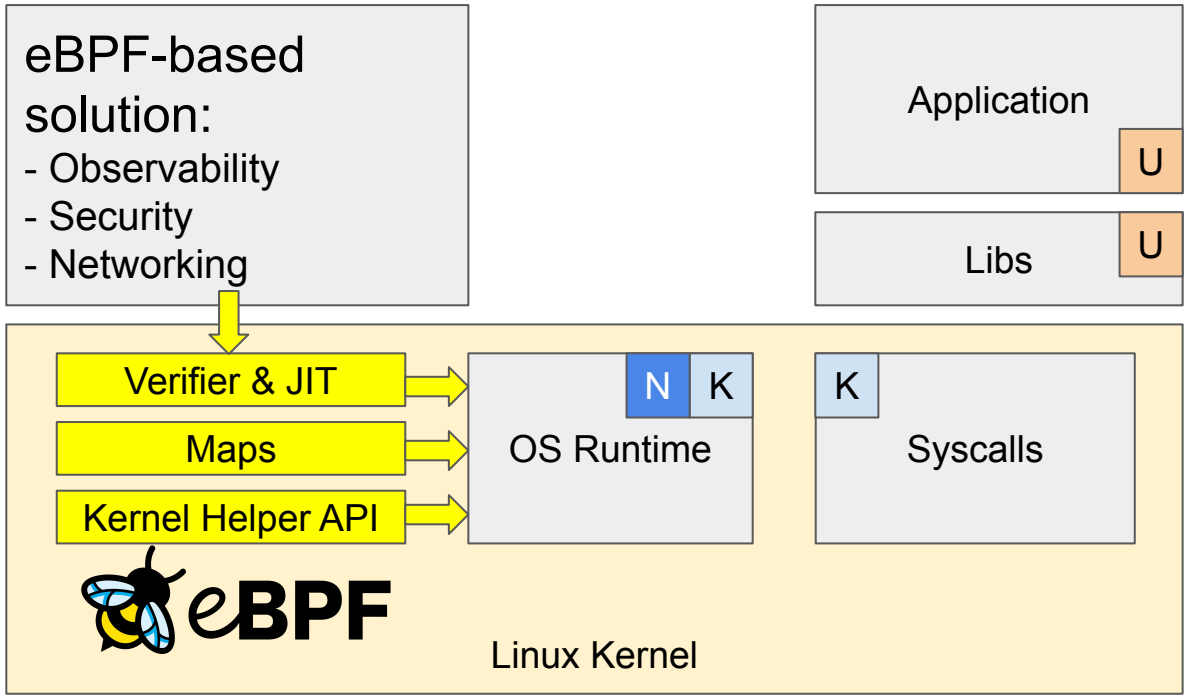
Mario Macias
Staff Software Engineer

Nikola Grcevski
Principal Software Engineer

The problem we are trying to solve

- Cross-availability zone (AZ) data transfers regularly account for at least 25% of the public cloud users' production cost
- Cloud vendor reports lack the granularity of information to tell which pods or workloads are responsible for the elevated cross-zone traffic
- We'd like to improve our bottom line, by cutting down on these costs
- We'd like to do this with as little effort as possible

eBPF at a glance



K KProbes

U UProbes

N Network programs

… etc

eBPF at a glance: observability

- No need to rebuild your code
- No need to redeploy your services
- Native performance (eBPF JIT)
- Safety (eBPF pre-verification)
- eBPF != “magic”
  - Requires API-level knowledge of instrumented targets
  - Requires binary-level knowledge of data
  - eBPF programs are limited in size and functionalities

Grafana Beyla

Grafana's approach to zero-code automatic instrumentation and network monitoring

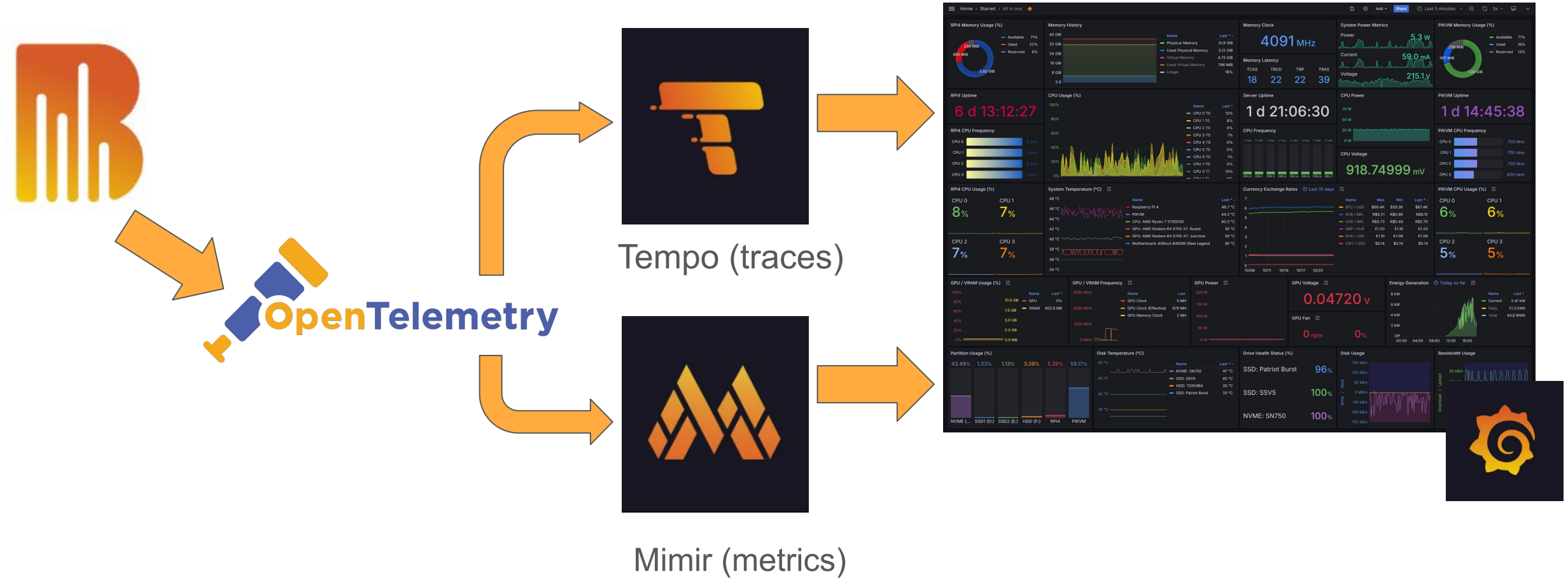
Metrics:

- beyla_network_flow_bytes (L3-L4)
- beyla_network_inter_zone_bytes (L3-L4)
- Application-level metrics (L5-L7, OTEL spec)
  - HTTP/s, gRPC, Kafka, Redis, SQL…

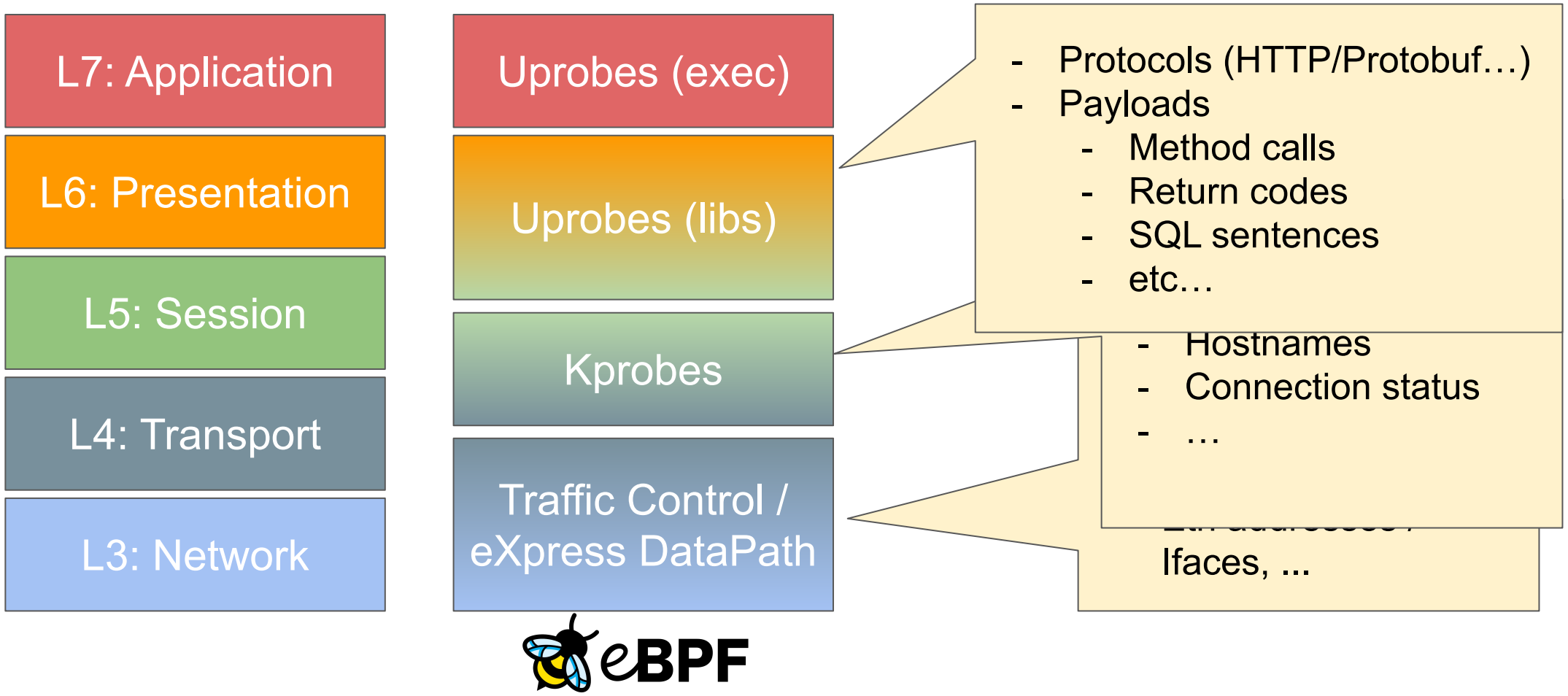
Traces:

- L5-L7 application-level traces (OpenTelemetry)

Grafana + Beyla in a nutshell

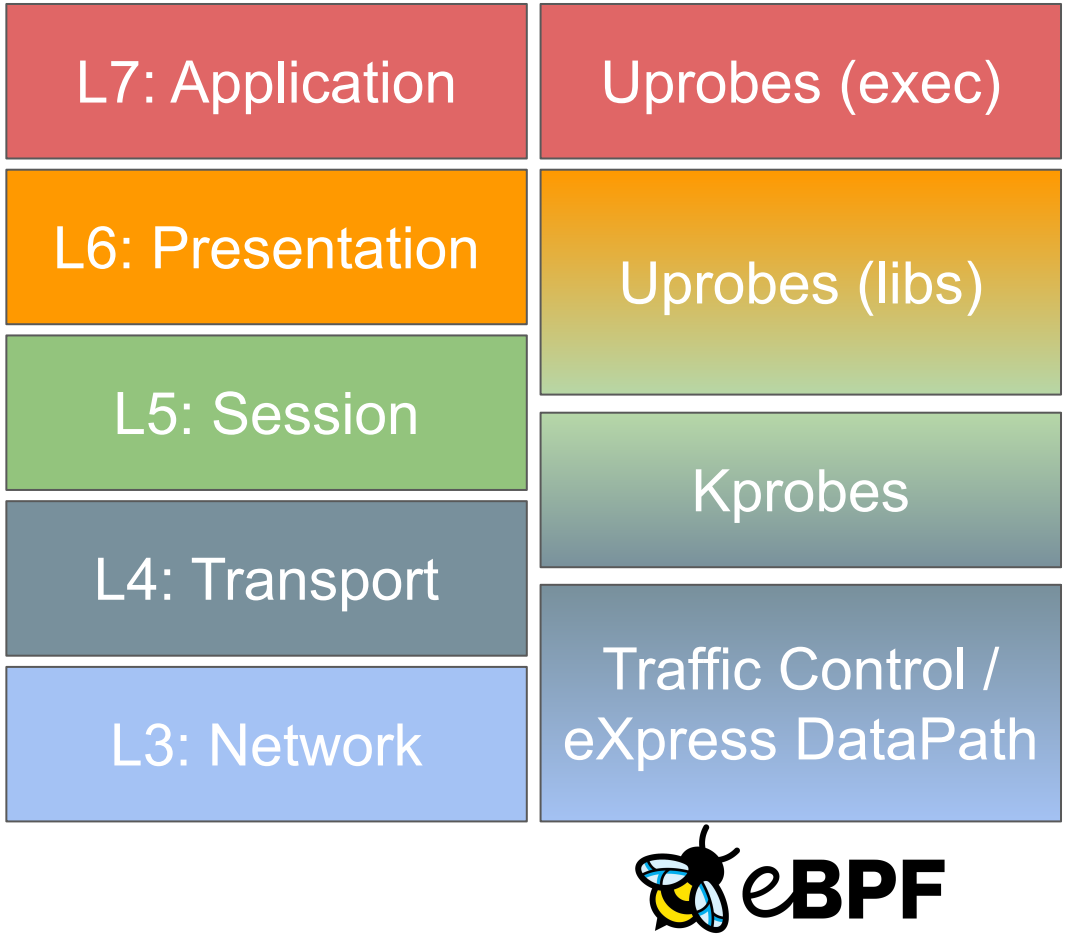


How to instrument your network

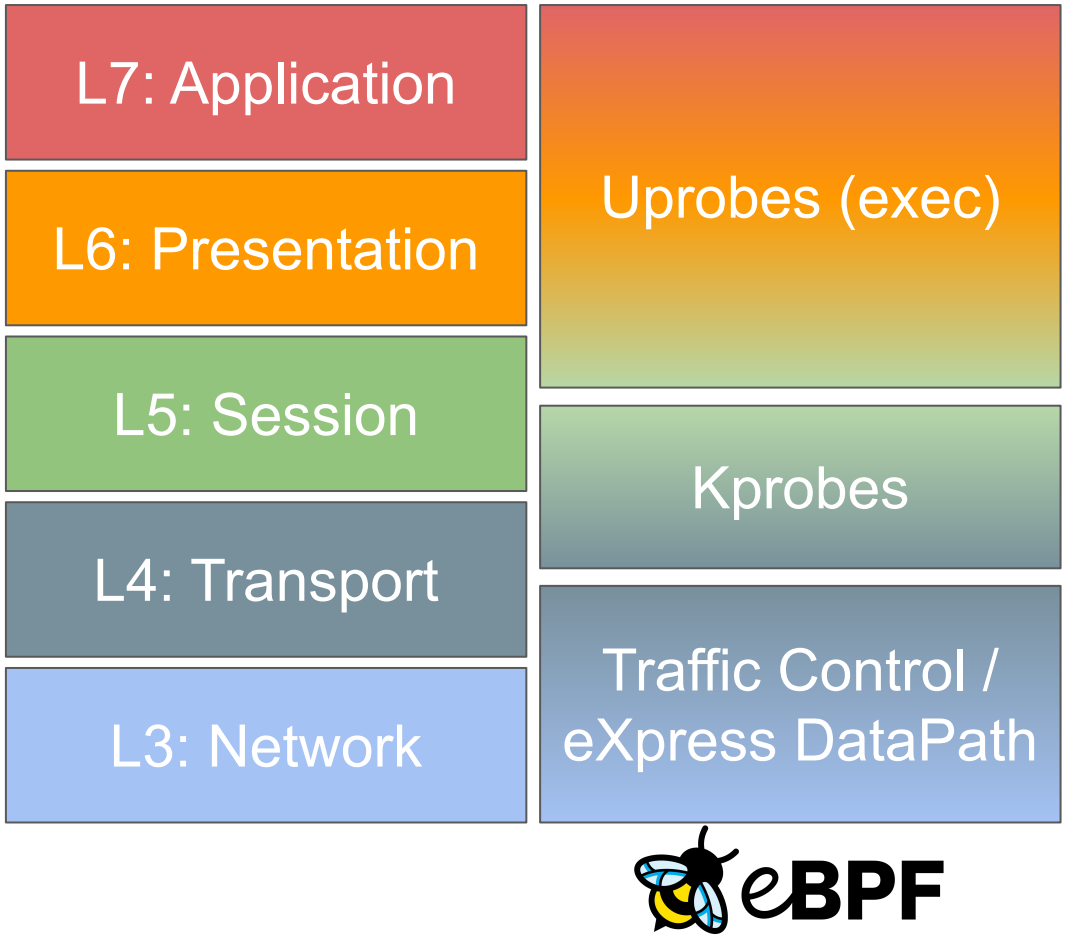


Instrumentation is platform-dependant

C, Rust, Python…



Go, Java…



We have all the puzzle pieces…

All that we can monitor with eBPF (Beyla)

- Application level data and application level protocols
- Low level network data
  - This is where we'll focus today
  - We'll look into how data enrichment of the network data can lead to meaningful optimisations and savings

Joining pieces for network metrics

Packet events captured by Linux Traffic Control

Time	Src IP	Src Port	Dst IP	Dst Port	…other…	Payload Length
98	…	…	…	…	…	…
100	10.0.0.4	54200	10.0.0.23	80	…	123
101	10.0.0.4	54201	10.0.0.36	3361	…	234
102	10.0.0.33	80	162.168.1.12	50342	…	322
103	10.0.0.4	54200	10.0.0.23	80	…	1234
102	10.0.0.33	80	162.168.1.12	50342	…	101
103	…	…	…	…	…	…

beyla_network_flow_bytes{
  src_ip=”10.0.0.4”, src_port=”54200”,
  dst_ip=”10.0.0.23”, dst_port=”80”
} 1357

L3-L4: network-level metrics

- ✌️ Robust: Based on stable APIs and standard binary representations (TCP, UDP, IP… packets)

- 🤯 Basic information
  - Sorce/Destination endpoints and ports
  - Packet count
  - Bytes sum

Enrichment of the network data
(Kubernetes and beyond)

What eBPF sees

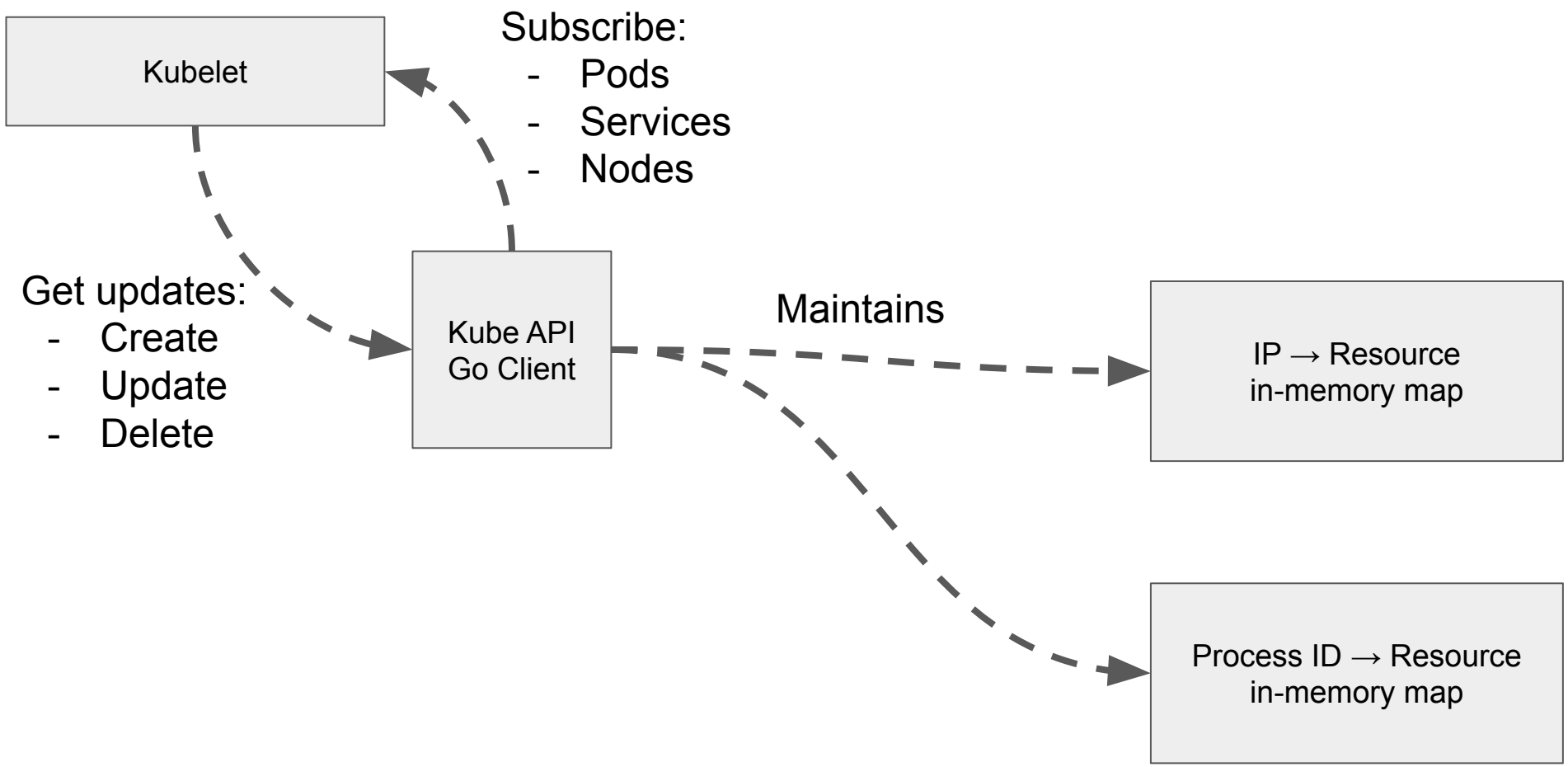
```
beyla_network_flow_bytes{
  src_address=”10.0.0.4”,
  src_port=”54200”,
  dst_address=”10.0.0.23”,
  dst_port=”80”
} 1357
```

What K8s users need

```
beyla_network_flow_bytes{
  src_address=”10.0.0.4”,
  src_port=”54200”,
  dst_address=”10.0.0.23”,
  dst_port=”80”
} 1357
```

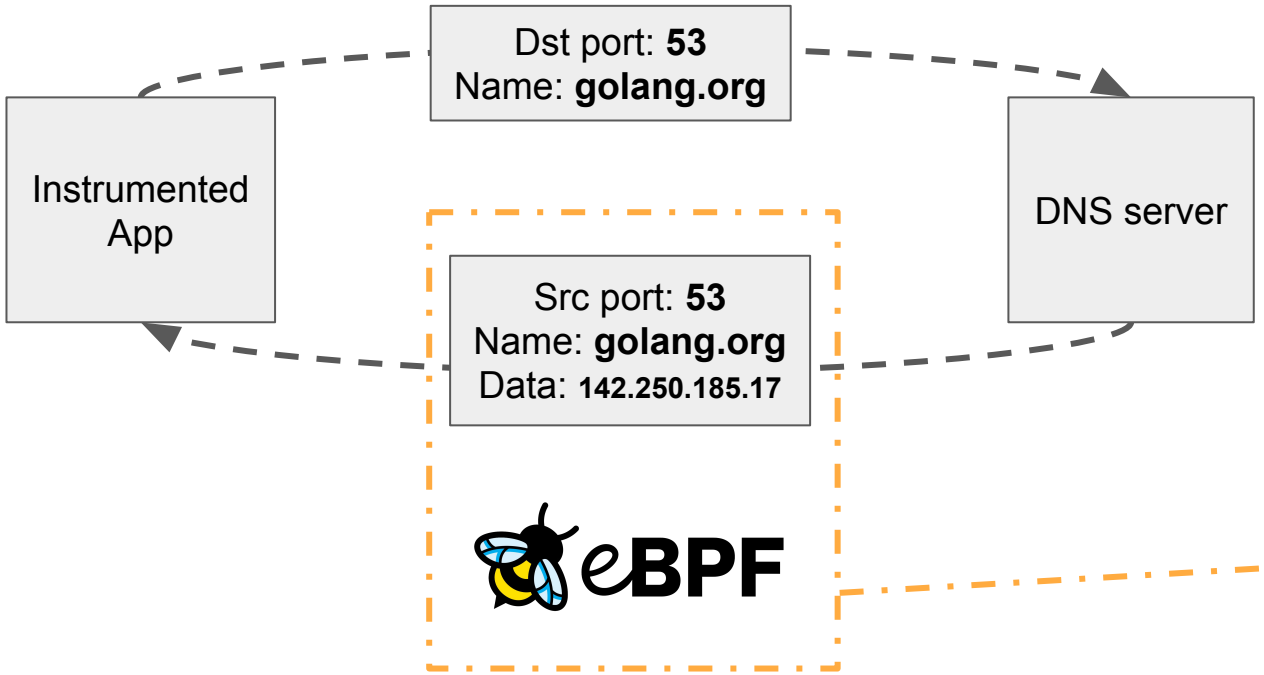
```
beyla_network_flow_bytes{
  k8s_src_owner=”frontend”,
  k8s_src_namespace=”app”
  k8s_dst_owner=”database”
  k8s_src_namespace=”storage”
} 1357
```

Kubernetes Informers to the rescue



Other traffic: Reverse DNS

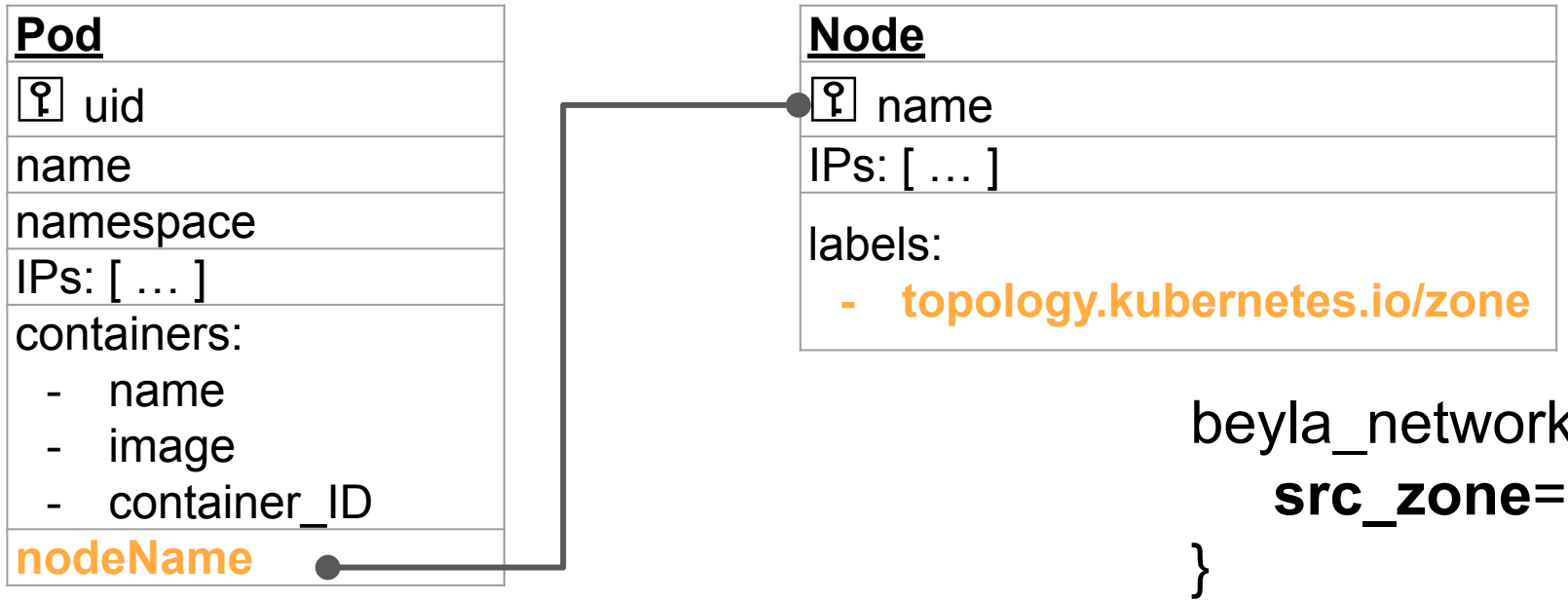
```
$ ping -c 1 golang.org
PING golang.org (142.250.185.17): 56 data bytes
…
$ nslookup 142.250.185.17
…
17.185.250.142.in-addr.arpa    name = mad41s11-in-f17.1e100.net.
```



network_flow_bytes{
…
dst_address=”142.250.185.17”,
dst_name=”golang.org”,
} 1357

Inter-zone traffic

Traffic between Cloud Availability-Zones is another label in the enriched data




```
beyla_network_flow_bytes {
    src_zone="...", dst_zone="...",
}
beyla_network_inter_zone_bytes {
    src_zone="...", dst_zone="...",
}
```

Variety of attributes to choose from

Attributes of Beyla metrics

For the sake of brevity, the metrics and attributes in this list use the OTEL `dot.notation`. When using the Prometheus exporter, the metrics use `underscore_notation`.

In order to configure which attributes to show or which attributes to hide, check the `attributes->select` section in the configuration documentation.

Expand table

`beyla.network.flow.bytes`	`client.port`	hidden
`beyla.network.flow.bytes`	`direction`	hidden
`beyla.network.flow.bytes`	`dst.address`	hidden
`beyla.network.flow.bytes`	`dst.cidr`	shown if the `cidrs` configuration section exists
`beyla.network.flow.bytes`	`dst.name`	hidden
`beyla.network.flow.bytes`	`dst.port`	hidden
`beyla.network.flow.bytes`	`dst.zone` (only Kubernetes)	hidden
`beyla.network.flow.bytes`	`iface`	hidden
`beyla.network.flow.bytes`	`k8s.cluster.name`	shown if Kubernetes is enabled
`beyla.network.flow.bytes`	`k8s.dst.name`	hidden
`beyla.network.flow.bytes`	`k8s.dst.namespace`	shown if Kubernetes is enabled
`beyla.network.flow.bytes`	`k8s.dst.node.ip`	hidden
`beyla.network.flow.bytes`	`k8s.dst.node.name`	hidden
`beyla.network.flow.bytes`	`k8s.dst.owner.type`	hidden
`beyla.network.flow.bytes`	`k8s.dst.type`	hidden
`beyla.network.flow.bytes`	`k8s.dst.owner.name`	shown if Kubernetes is enabled
`beyla.network.flow.bytes`	`k8s.src.name`	hidden
`beyla.network.flow.bytes`	`k8s.src.namespace`	shown if Kubernetes is enabled
`beyla.network.flow.bytes`	`k8s.src.node.ip`	hidden
`beyla.network.flow.bytes`	`k8s.src.owner.name`	shown if Kubernetes is enabled
`beyla.network.flow.bytes`	`k8s.src.owner.type`	hidden
`beyla.network.flow.bytes`	`k8s.src.type`	hidden
`beyla.network.flow.bytes`	`server.port`	hidden
`beyla.network.flow.bytes`	`src.address`	hidden
`beyla.network.flow.bytes`	`src.cidr`	shown if the `cidrs` configuration section exists
`beyla.network.flow.bytes`	`src.name`	hidden
`beyla.network.flow.bytes`	`src.port`	hidden
`beyla.network.flow.bytes`	`src.zone` (only Kubernetes)	hidden
`beyla.network.flow.bytes`	`transport`	hidden
Traces (SQL, Redis)	`db.query.text`	hidden

https://grafana.com/docs/beyla/next/metrics/

NOTE

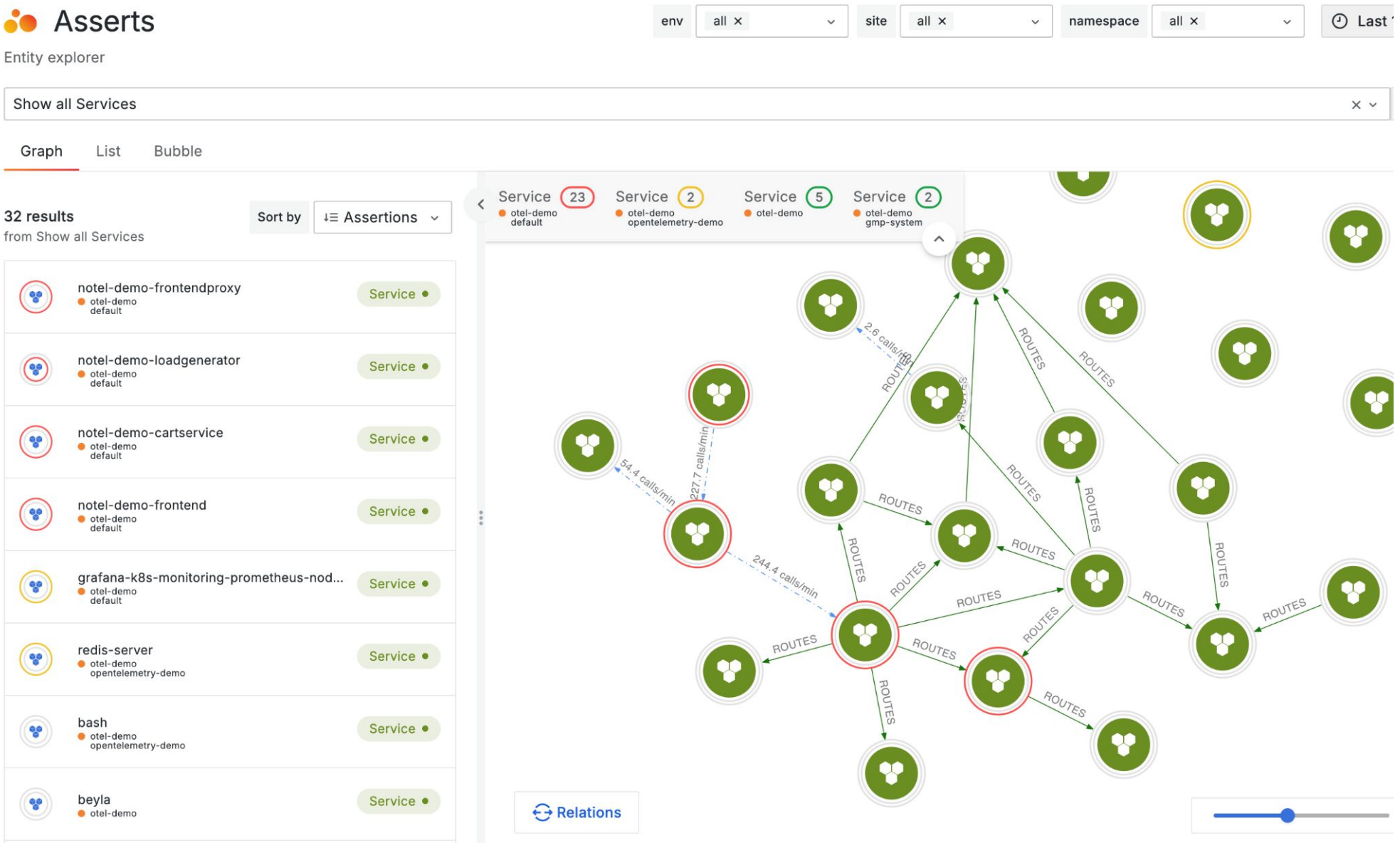
The `beyla.network.inter.zone.bytes` metric supports the same set of attributes as `beyla.network.flow.bytes`, but all of them are hidden by default, except `k8s.cluster.name`, `src.zone` and `dst.zone`.

Dealing with cardinality (details vs cost)

Choosing what you need is important

```
attributes:
  select:
    beyla_network_flow_bytes:
      include:
        - k8s.src.owner.*
        - k8s.dst.owner.*
        - src.zone
        - dst.zone
    sql_client_duration:
      # report all the possible attributes but db_statement
      include: ["*"]
      exclude: ["db_statement"]
    http_*:
      # report the default attribute set but exclude the Kubernetes Pod information
      exclude: ["k8s.pod.*"]
```

Visualizing: providing meaning to the data



Visualizing: Cross zone traffic reports

```
sum by (cluster, src_zone, dst_zone) (rate(
  beyla_network_inter_zone_bytes_total{src_zone!="",dst_zone!=""
}[$__rate_interval]))/2
```

Four figures per year

Visualizing: Cross zone traffic reports

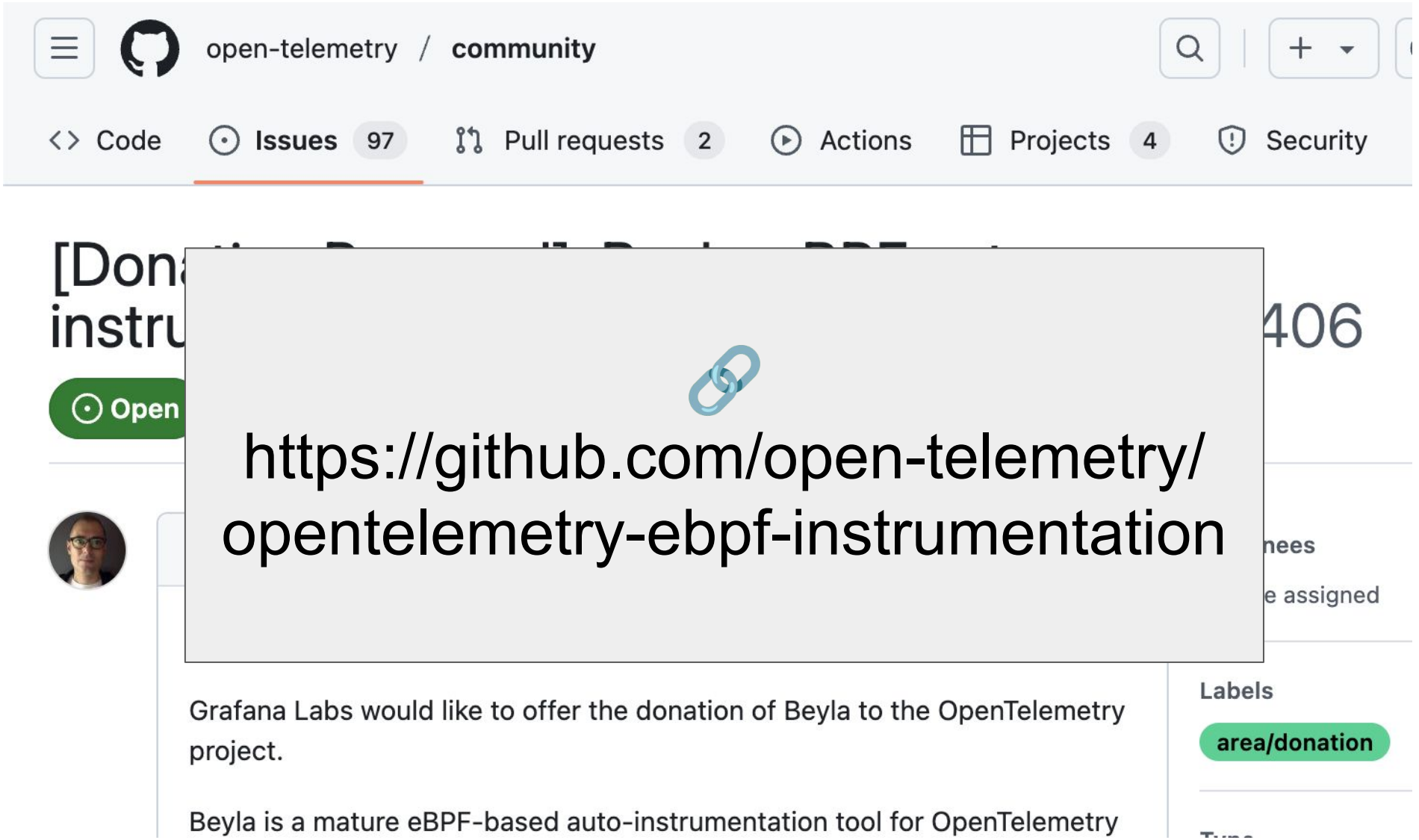
```
topk(3, sum by(k8s_src_owner_name,  k8s_dst_owner_name) (
    beyla_network_flow_bytes_total{
       cluster="dev-us-east-0",
       dst_zone="us-east-2a", src_zone="us-east-2b"}))
```

Table

Table | Raw

Time	k8s_dst_owner_name	k8s_src_owner_name	Value
2025-05-28 10:07:44.710	[redacted]	[redacted]	135336419
2025-05-28 10:07:44.710	[redacted]	[redacted]	16909359
2025-05-28 10:07:44.710	[redacted]	[redacted]	9060718

OSS Community: a core principle



Grafana Labs

Thank you and keep in touch!

Mario Macias
Staff Software Engineer

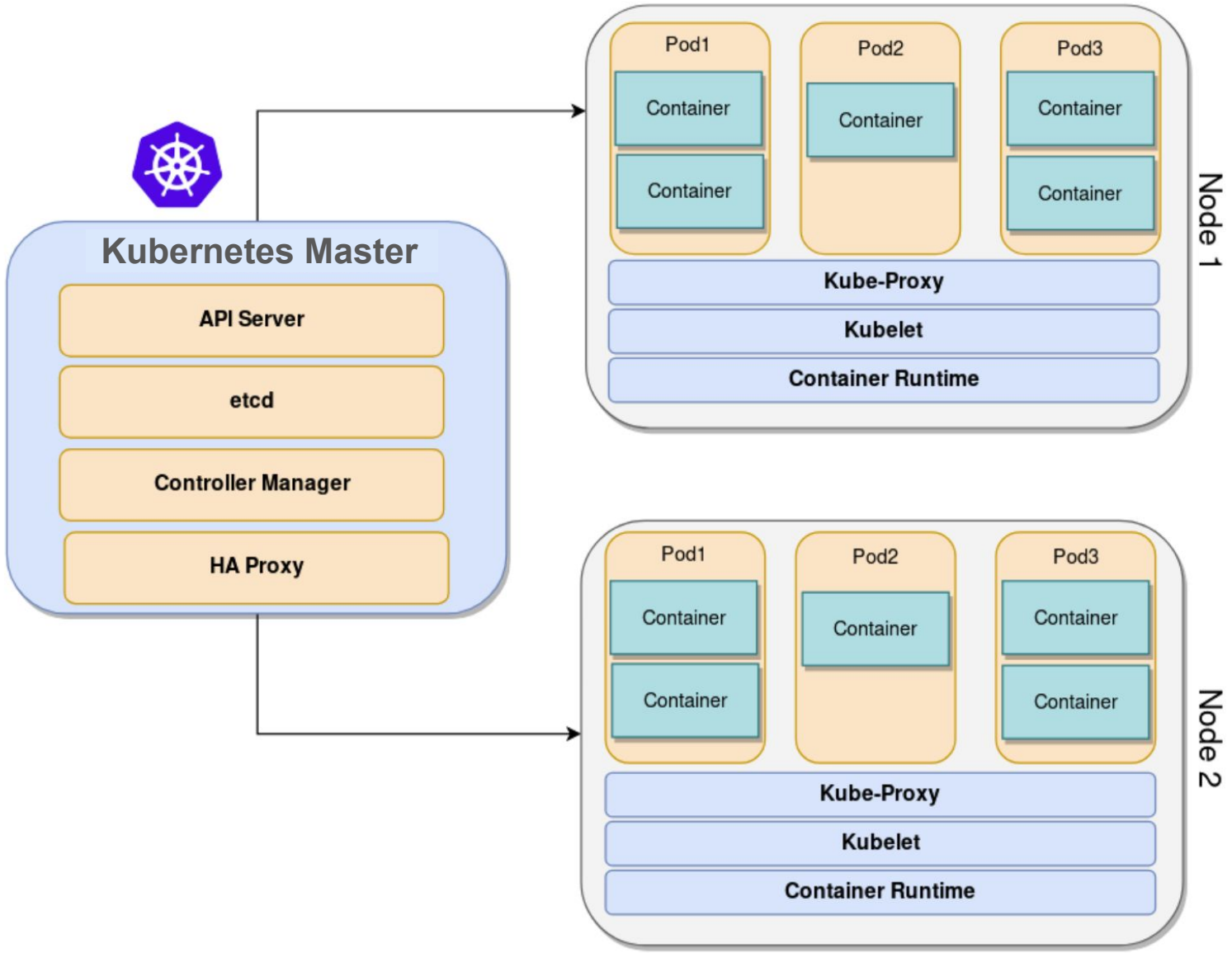
Nikola Grcevski
Principal Software Engineer

- Grafana Beyla
  - https://github.com/grafana/beyla
  - **#beyla** channel in grafana.slack.com

- Opentelemetry eBPF Instrumentation (OBI)
  - https://github.com/open-telemetry/opentelemetry-ebpf-instrumentation
  - **#otel-ebpf-instrumentation** channel in cloud-native.slack.com

Backup slides

Kubernetes (K8s) in a nutshell



source: Hands-On Cloud-Native Applications with Java and Quarkus by Francesco Marchioni