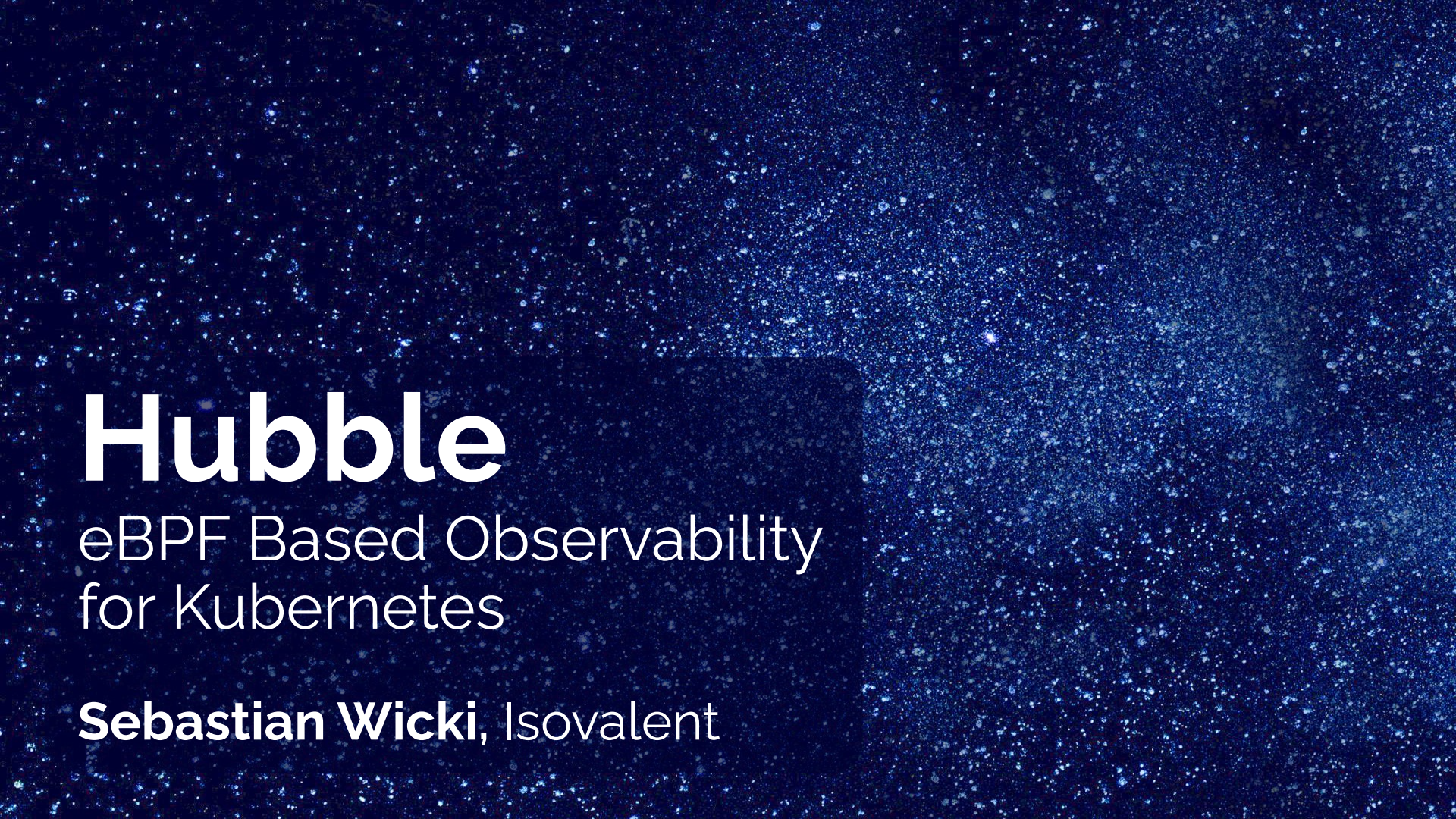

Hubble
eBPF Based Observability
for Kubernetes
Sebastian Wicki, Isovalent

Observability in Kubernetes

Application Observability

- What services does an application depend on?
- What HTTP/gRPC/Kafka calls are being made?
- What is the performance of my application?

Operations Observability

- Is any network communication failing?
- Is it an application or network problem? Which network layer?

Security Observability

- Which connections were blocked due to network policy?
- What services have been accessed from outside the cluster?

eBPF

What is eBPF?

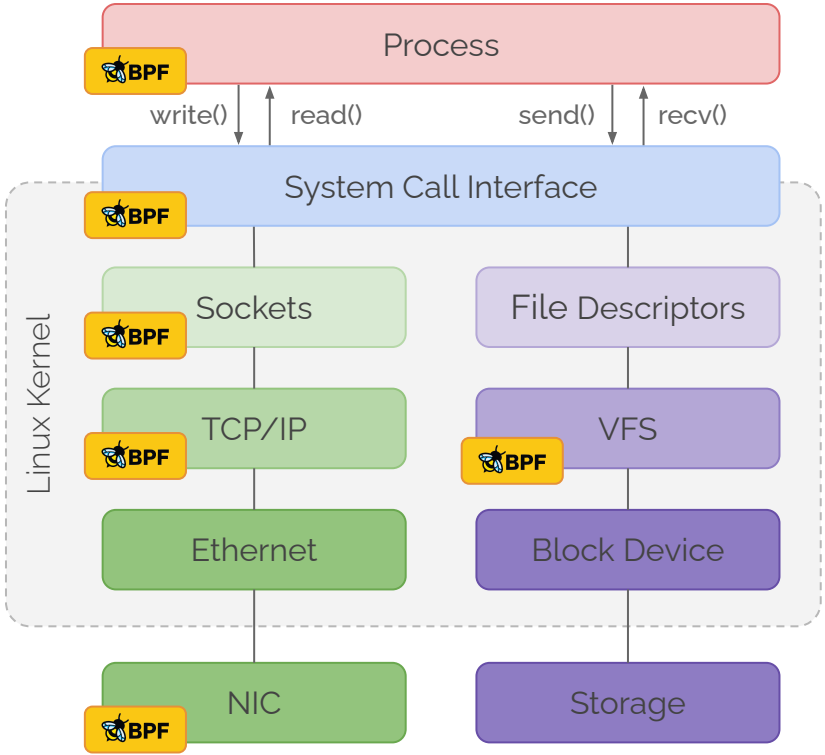
eBPF is a Linux kernel feature that allows to **dynamically** attach small programs to the kernel in a **secure** and **efficient** manner.

```
SEC("action")
int observe_egress(struct __sk_buff *skb) {
  // Extract data for Ethernet and IP Header
  void *data = (void *)(long)skb->data;
  void *data_end = (void *)(long)skb->data_end;
  if (data + sizeof(struct ethhdr) + sizeof(struct iphdr) > data_end)
    return TC_ACT_UNSPEC;

  // Read Ethernet Header
  struct ethhdr *eth = data;
  if (eth->h_proto != __constant_htons(ETH_P_IP))
    return TC_ACT_UNSPEC;

  // Read IPv4 Header
  struct iphdr *ip = (data + sizeof(struct ethhdr));
  trace_printk("Observed IP Packet: %lu -> %lu (%d)\n",
                    ip->saddr, ip->daddr, ip->protocol);

  return TC_ACT_OK;
}
```



Why eBPF?

- **Transparent**
  - Visibility without having to modifying the application
- **Minimal overhead**
  - Dynamically enable visibility where needed
- **Widely Available**
  - In prod at large-scale users
  - Hubble supports Linux 4.9+

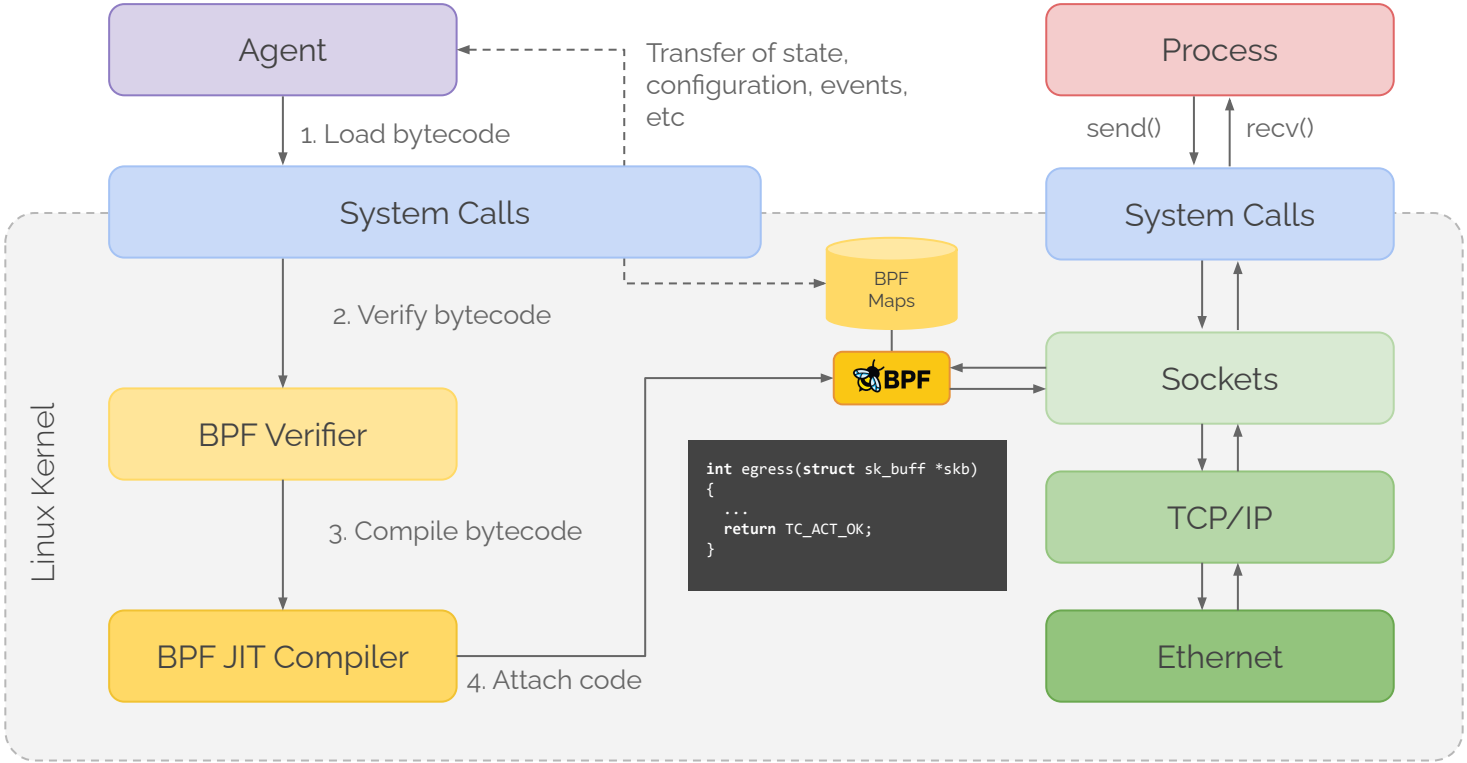
Large-scale users

347 Contributors (Jan 2014 to Jul 2020)

- 588 Daniel Borkmann (Isovalent; maintainer)
- 421 Andrii Nakryiko (Facebook)
- 401 Alexei Starovoitov (Facebook; maintainer)
- 224 Yonghong Song (Facebook)
- 209 Jakub Kicinski (Facebook)
- 183 Martin KaFai Lau (Facebook)
- 179 Stanislav Fomichev (Google)
- 165 John Fastabend (Isovalent)
- 161 Quentin Monnet (Isovalent)
- 130 Jesper Dangaard Brouer (Red Hat)
- 117 Andrey Ignatov (Facebook)
- [...]

See also: eBPF and Kubernetes: Little Helper Minions for Scaling Microservices – Daniel Borkmann

How to use eBPF



Cilium

Pod-to-Pod Network Connectivity (CNI)

Service-based Load-balancing

Security Enforcement (NetworkPolicy)

Encryption

Hubble

Network and Security Observability

Service Dependency Maps

Troubleshooting

Metrics & Monitoring

History of Hubble

November 2019
Hubble Preview

- Metrics and CLI
- Hubble UI Preview
- Built for Cilium 1.6

February 2020
Cilium 1.7

- Hubble UI open-sourced
- Visibility annotations
- Extended metadata available

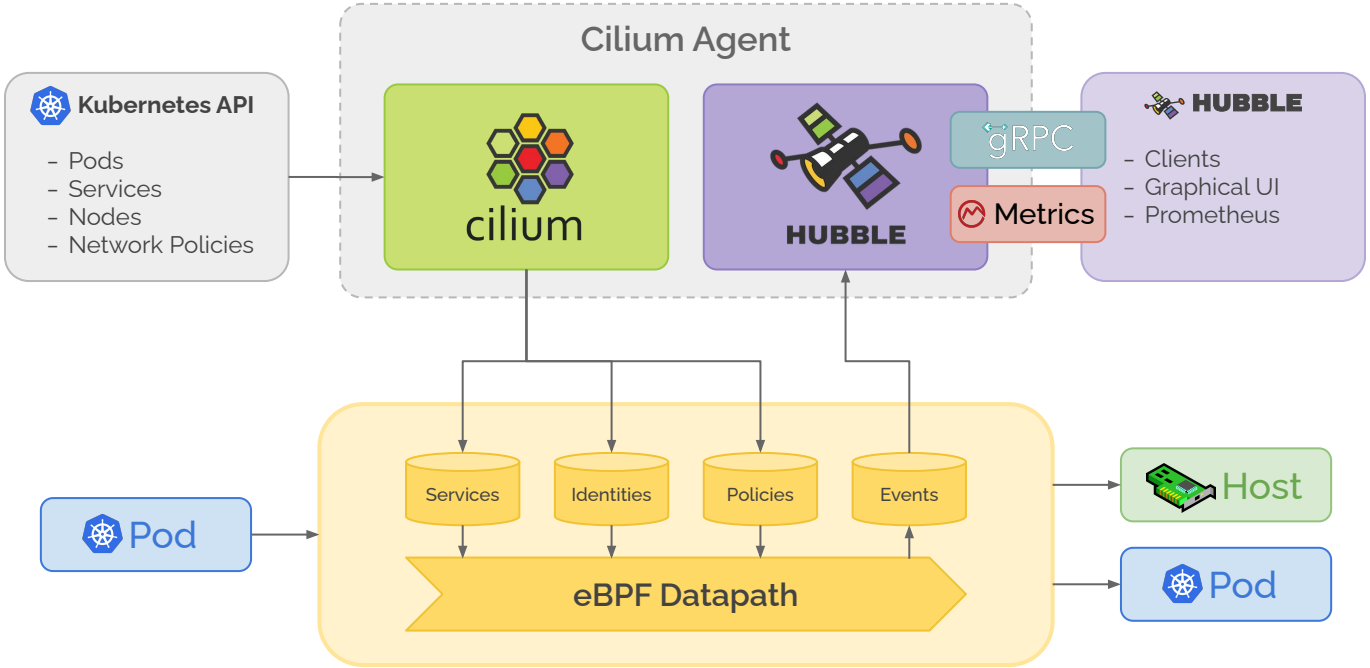
March 2020
Hubble 0.5

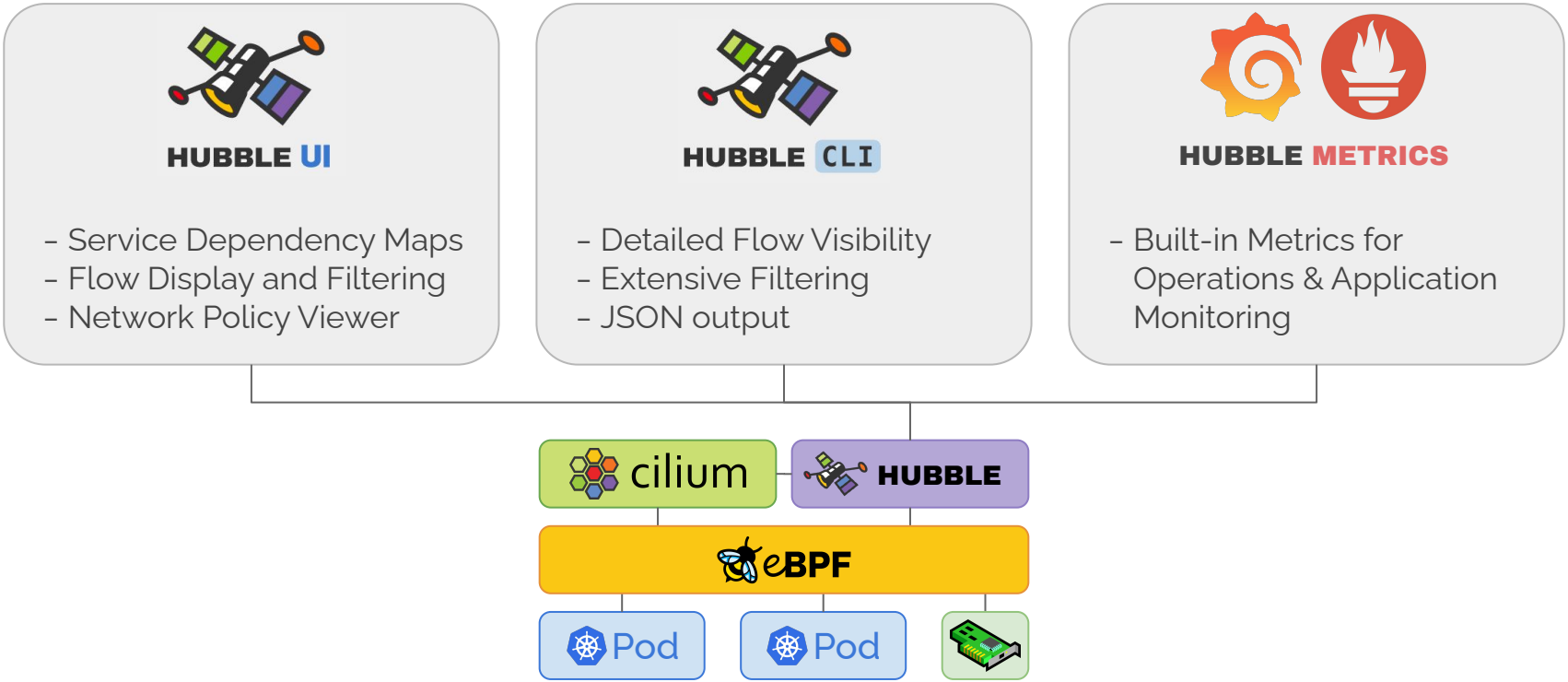
- First Hubble release with API stability
- Last release with DaemonSet for Cilium 1.7

June 2020
Cilium 1.8 + Hubble 0.6

- Cluster-wide visibility with Relay (Beta)
- Hubble API directly embedded in Cilium
- Scalability improvements in UI

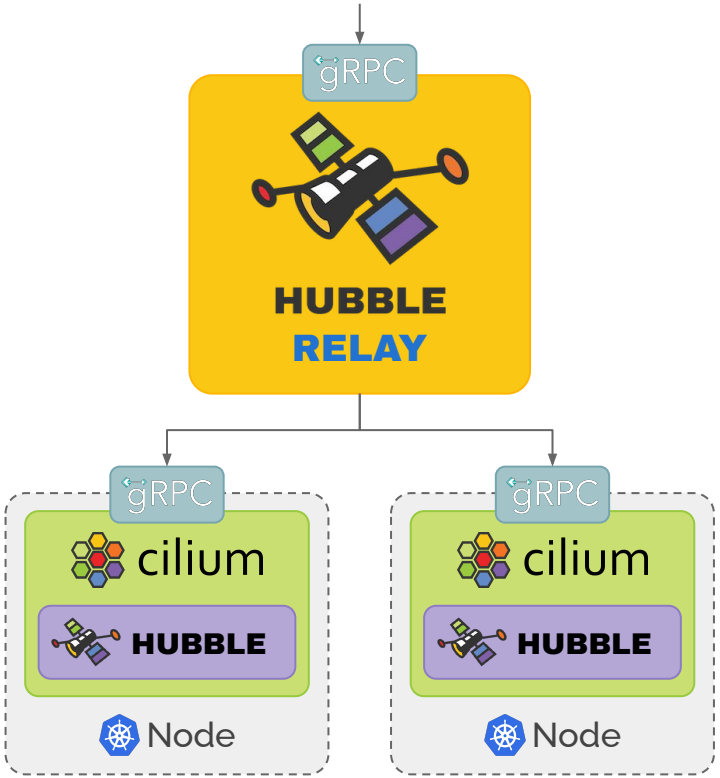
How Hubble uses eBPF




HUBBLE UI
- Service Dependency Maps
- Flow Display and Filtering
- Network Policy Viewer
HUBBLE CLI
- Detailed Flow Visibility
- Extensive Filtering
- JSON output
HUBBLE METRICS
- Built-in Metrics for Operations & Application Monitoring
cilium
HUBBLE
eBPF
Pod
Pod

Hubble API

- **Access to recent flows**
  - 4k events per node by default (configurable)
- **Streams current flows**
  - Matching a predefined filter
- **Cluster-wide visibility**
  - Via Hubble Relay
  - Beta status in Cilium 1.8
- **Accessed by CLI and UI**



Flow Visibility

```
$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
tiefighter                    1/1     Running   0          2m34s
xwing                         1/1     Running   0          2m34s
deathstar-5b7489bc84-crlxh    1/1     Running   0          2m34s
deathstar-5b7489bc84-j7qwq    1/1     Running   0          2m34s

$ hubble observe --follow -l class=xwing
# DNS lookup to coredns
default/xwing:41391 -> kube-system/coredns-66bff467f8-28dgp:53 to-proxy FORWARDED (UDP)
kube-system/coredns-66bff467f8-28dgp:53 -> default/xwing:41391 to-endpoint FORWARDED (UDP)
# ...
# Successful HTTPS request to www.disney.com
default/xwing:37836 -> www.disney.com:443 to-stack FORWARDED (TCP Flags: SYN)
www.disney.com:443 -> default/xwing:37836 to-endpoint FORWARDED (TCP Flags: SYN, ACK)
www.disney.com:443 -> default/xwing:37836 to-endpoint FORWARDED (TCP Flags: ACK, FIN)
default/xwing:37836 -> www.disney.com:443 to-stack FORWARDED (TCP Flags: RST)
# ...
# Blocked HTTP request to deathstar backend
default/xwing:49610 -> default/deathstar:80 Policy denied DROPPED (TCP Flags: SYN)
```

Flow Metadata

- Ethernet headers
- IP & ICMP headers
- UDP/TCP ports, TCP flags
- HTTP, DNS, Kafka, ...

Kubernetes

- Pod names and labels
- Service names
- Worker node names

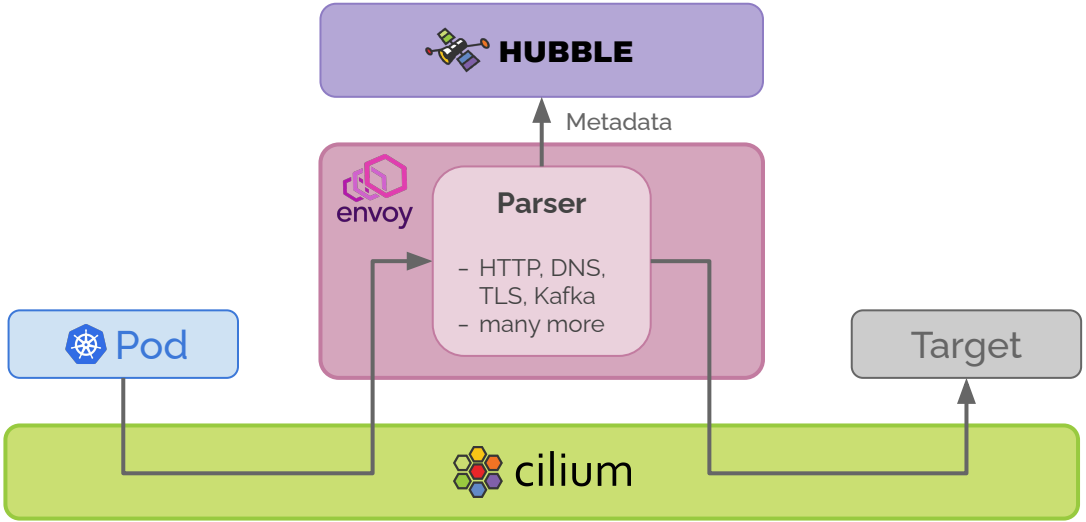
DNS (if available)

- Fully qualified domain names for source and destination

Cilium

- Security identities and endpoints
- Drop reasons
- Policy verdict matches

L7 Visibility

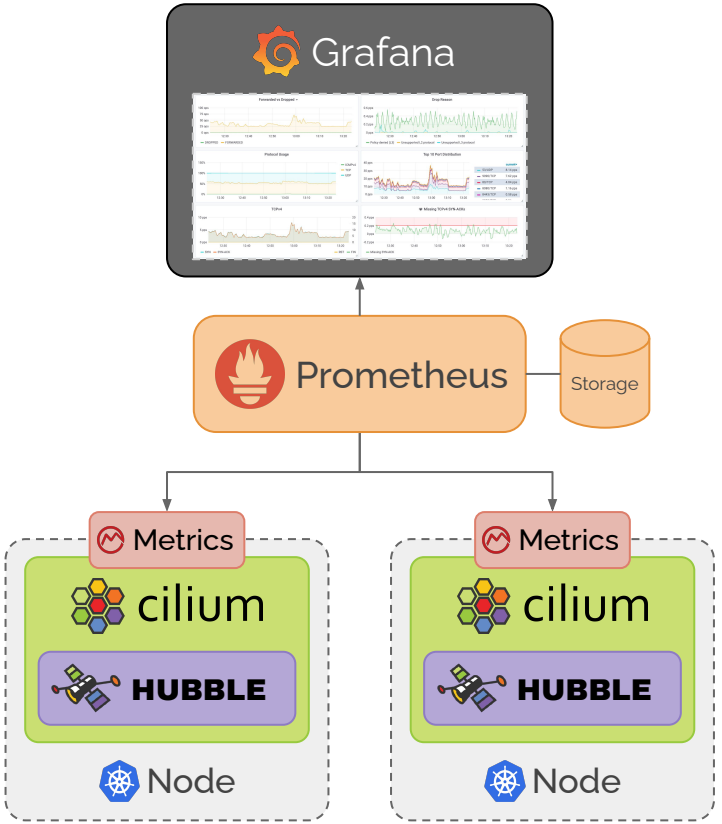



```
$ kubectl annotate pod tiefighter \
   io.cilium.proxy-visibility="<Egress/80/TCP/HTTP>"
pod/tiefighter annotated

$ hubble observe --from-pod tiefighter \
   --http-status 200 --output json | jq '.l7'
{
  "type": "RESPONSE",
  "latency_ns": "4342402",
  "http": {
          "code": 200,
          "method": "POST",
          "url": "http://deathstar/v1/request-landing",
          "protocol": "HTTP/1.1",
          "headers": [
            {
              "key": "User-Agent",
              "value": "curl/7.52.1"
            },
            ...
          ]
  }
}
```

Hubble Metrics

- **Open Metrics**
  - Scraping endpoint for e.g. Prometheus
- **Metrics for Flow Events**
  - e.g. HTTP, DNS, TCP, ICMP
- **Visualization with Grafana**
  - Example dashboards available
- **Customizable & Extendable**



Demo

Thank you!

Try it out:
https://docs.cilium.io/en/v1.8/gettingstarted/hubble/

Contribute:
https://github.com/cilium/hubble
https://github.com/cilium/hubble-ui