

## Beyla vs Hubble/Cilium

- **Beyla:** Requires additional services like Tempo, OTEL Collector, Prometheus. Becomes memory-intensive. Good for L7 metrics. L3 metrics can be enabled in Beyla configmap and queried in grafana/prometheus(beyla\_network\_flow\_bytes\_total). Can integrate with grafana to display service graphs in grafana dashboards. Does not need to be deployed before any other services, can dynamically add services to the service discovery and remove at any time based on requirements.
- **Hubble/Cilium** - Service graphs are visible in Hubble UI. To view in grafana, we need to purchase Isovalent Enterprise. Good for L7 metrics and dashboards. L3 metrics can be enabled also, but are not easily exportable to grafana or are not as accessible/detailed in Hubble UI. Less memory-intensive, less external configurations. But needs to be started before any other services, monitors all running services in the cluster.

## Cilium and Hubble Setup

Create a GKE cluster and connect to it with these commands (need to use this command with the taint in order to guarantee that pods are only scheduled in the node when Cilium is ready to monitor them):

```
Unset
export NAME="$(whoami)-$RANDOM"
gcloud container clusters create "${NAME}" \
  --node-taints node.cilium.io/agent-not-ready=true:NoExecute \
  --zone us-west2-a \
  --subnetwork default
gcloud container clusters get-credentials "${NAME}" --zone us-west2-a
```

Install the Cilium CLI. <https://docs.cilium.io/en/stable/gettingstarted/k8s-install-default/>  
Then run

```
Unset
cilium install --version 1.15.6
```

to install Cilium.

Run

```
Unset
cilium status --wait
```

Install the Hubble CLI. [https://docs.cilium.io/en/stable/gettingstarted/hubble\\_setup/](https://docs.cilium.io/en/stable/gettingstarted/hubble_setup/)  
Run this command to enable Hubble, Prometheus, Hubble metrics, and Hubble UI.

Unset

```
helm upgrade cilium cilium/cilium --version 1.15.6 \
  --namespace kube-system \
  --reuse-values \
  --set prometheus.enabled=true \
  --set operator.prometheus.enabled=true \
  --set hubble.enabled=true \
  --set
hubble.metrics.enabled="{dns,drop,tcp,flow,port-distribution,icmp,http}" \
  --set hubble.relay.enabled=true \
  --set hubble.ui.enabled=true
```

Allows you to connect to Hubble relay on port 4245 and queries the Hubble relay/flow API on port 4245:

Unset

```
cilium hubble port-forward&
hubble observe
```

Run this command to create a prometheus and grafana deployment in the namespace cilium-monitoring in the cluster. The cilium and hubble grafana dashboards are pre-loaded in the example set-up:

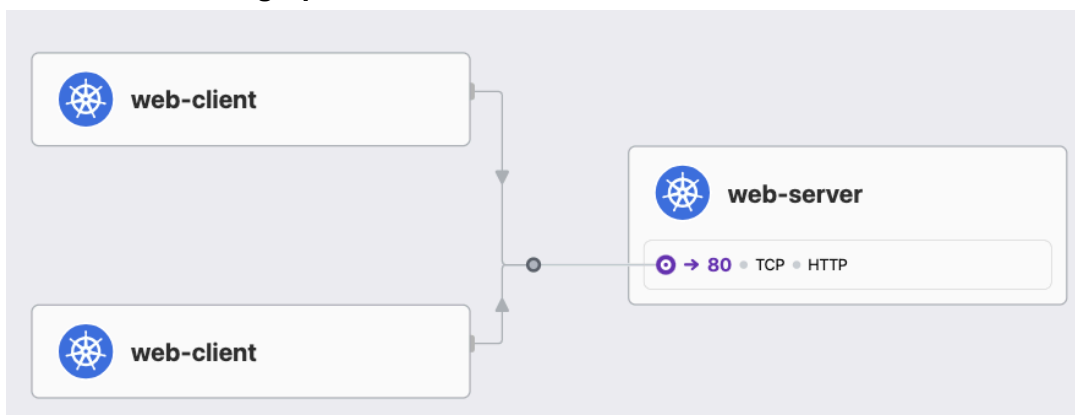
Unset

```
kubectl create -f
examples/kubernetes/addons/prometheus/monitoring-example.yaml
```

## Deploy test server and client in GKE cluster default namespace to generate L7 data flow

default	web-client-2t79b	1/1	Running
default	web-client-87f5659f9-vvhvm	1/1	Running
default	web-server-6446bb766c-p898j	1/1	Running
default	web-server-6446bb766c-w9n65	1/1	Running

Hubble UI service graph of server and clients:



## Hubble L7 visibility:

After creating the test server and client, enable hubble L7 flow visibility using:

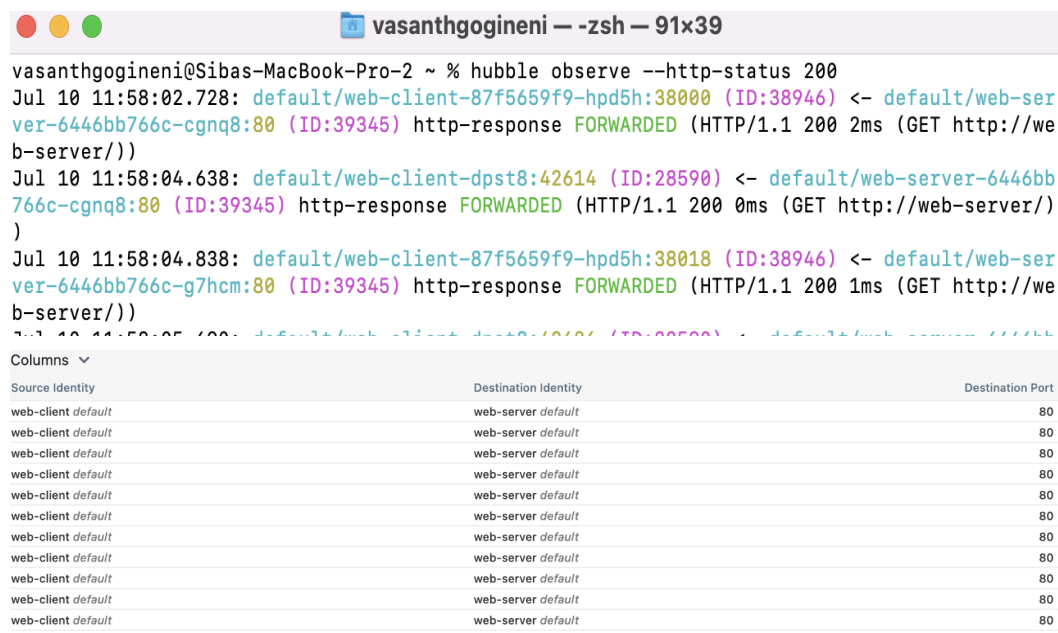
Unset

```
kubectl apply -f
https://docs.isovalent.com/public/http-ingress-visibility.yaml
```

The status codes and HTTP data can be seen in Hubble now and can be queried from Hubble CLI:

Unset

```
Hubble observe --http-status 200
```

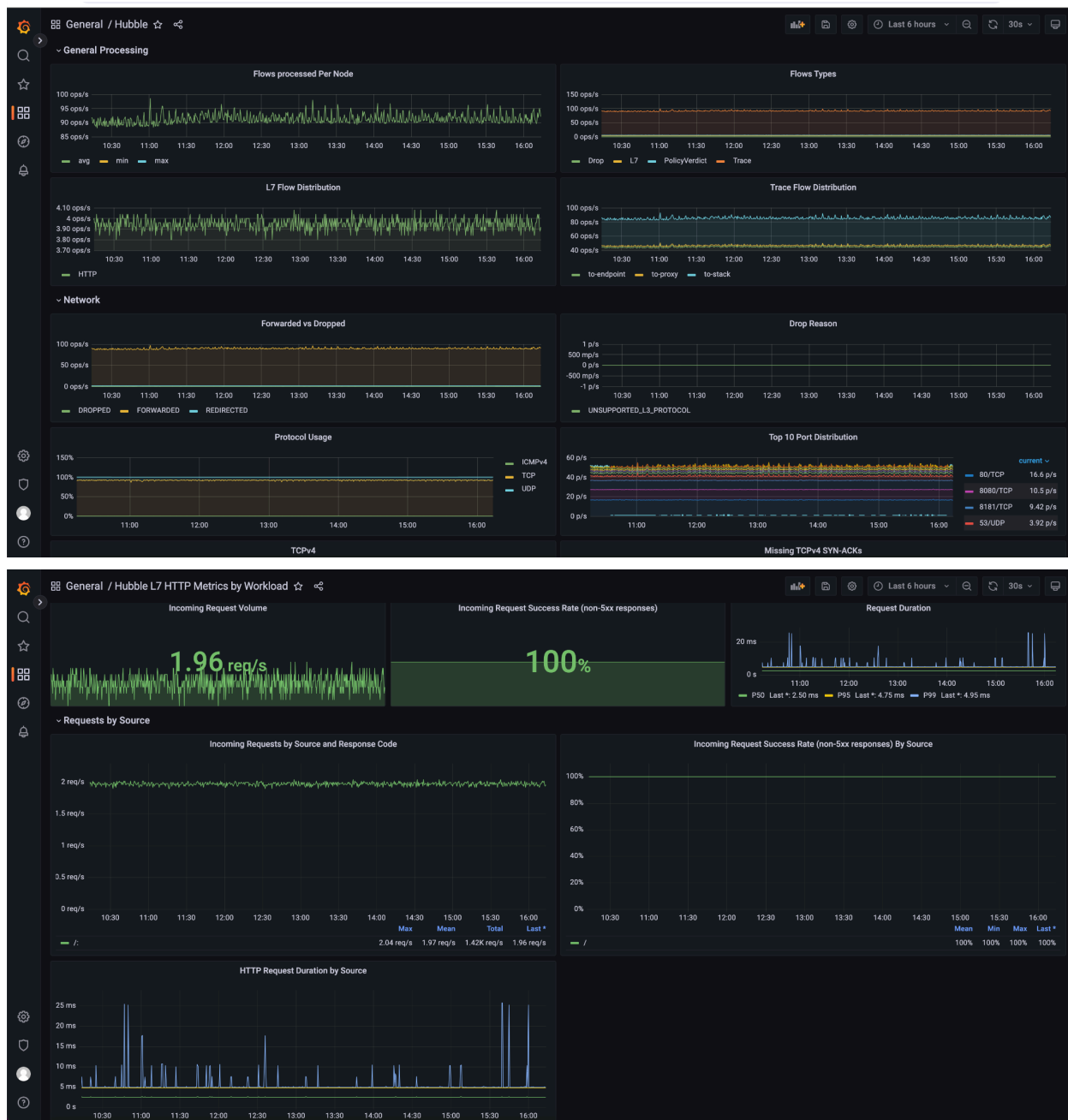


The screenshot shows a terminal window titled "vasanthgogineni — zsh — 91x39". The terminal output displays Hubble CLI results for the command "hubble observe --http-status 200". It shows three log entries with timestamps and details about the flow, including source and destination identities, status codes, and response times. Below the terminal output, there is a table with columns: Source Identity, Destination Identity, Destination Port, and L7 info. The table lists several flows between "web-client default" and "web-server default" on port 80, with L7 info indicating "GET / 0ms".

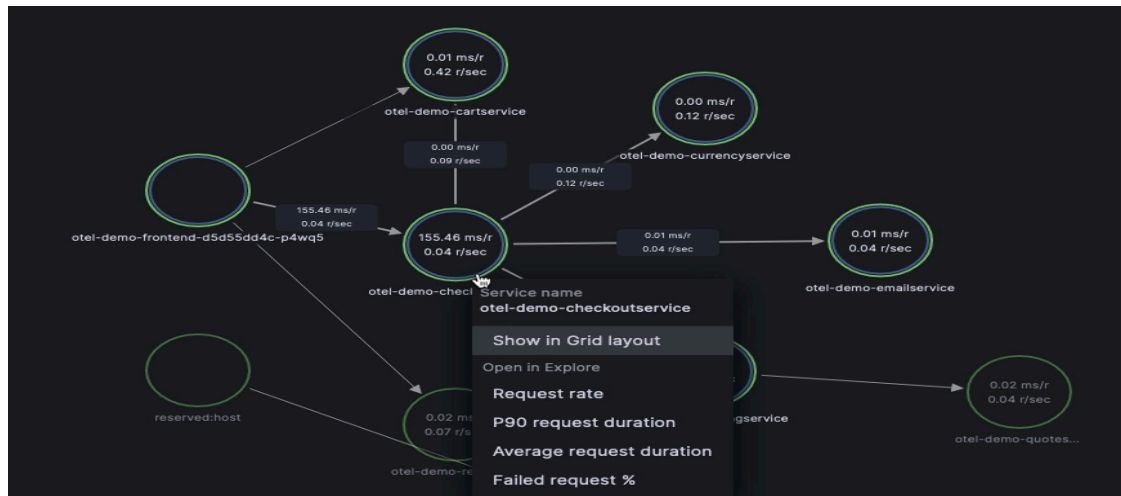
Source Identity	Destination Identity	Destination Port	L7 info
web-client default	web-server default	80	—
web-client default	web-server default	80	—
web-client default	web-server default	80	→ GET / 0ms
web-client default	web-server default	80	—
web-client default	web-server default	80	—
web-client default	web-server default	80	—
web-client default	web-server default	80	—
web-client default	web-server default	80	—
web-client default	web-server default	80	—
web-client default	web-server default	80	→ GET / 0ms

Enabling L7 flow visibility gives the Hubble dashboard and Hubble L7 HTTP metrics dashboard in grafana access to L7 data flow.

# Hubble and Hubble L7 dashboard in Grafana:



## Service Graph:



The service graph needs to be created using a node graph. The datasource should be the Hubble plugin for Grafana. After trying to create the Hubble datasource, I found that the datasource needed Hubble Timescape, which is only in Isovalent Enterprise version or Hubble Enterprise.

**Data Sources / Hubble**  
Type: Hubble

**Settings** | Dashboards

Alerting not supported

Name: Hubble | Default

**Timescape**

**Enterprise feature**  
This feature requires Isovalent Cilium Enterprise. For more information visit <https://isovalent.com/product/>.

**HTTP**

URL:

Allowed cookies:  **Add**

Timeout:

**Auth**

Basic auth: ☒ With Credentials ☐ With CA Cert

TLS Client Auth: ☒ With CA Cert ☐ With Client Cert

Skip TLS Verify: ☐

Forward OAuth Identity: ☐

**Custom HTTP Headers**

+ Add header

**Service Graph**

To allow querying service graph data you have to select a Prometheus instance where the data is stored.

Data source:  **Clear**

**Trace ID search**

To allow linking to traces from Hubble flows you have to select a Tempo instance where traces are stored.

Data source:

**Timescape healthcheck failed**

**Back** **Explore** **Delete** **Save & test**

**Alternative:** Use Node Graph API plugin and deploy a flask app in the cluster to create an API that acts as a datasource for the node graph.