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.

Enabling Service Graphs in Grafana using Beyla, Tempo, and OpenTelemetry

Create a GKE cluster

Unset

```
gcloud container clusters create beyla-test
```

Create beyla namespace

Unset

```
kubectl create namespace beyla
```

Create beyla service account and role binding

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: beyla
 namespace: beyla
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: beyla
rules:
 - apiGroups: [ "apps" ]
   resources: [ "replicasets" ]
   verbs: [ "list", "watch" ]
 - apiGroups: [ "" ]
   resources: [ "pods", "services", "nodes" ]
   verbs: [ "list", "watch" ]
 - apiGroups: [""]
   resources: ["pods", "nodes"]
   verbs: ["get", "list", "watch"]
 - apiGroups: ["apps"]
   resources: ["deployments"]
   verbs: ["get", "list", "watch"]
 - apiGroups: ["monitoring.coreos.com"]
   resources: ["servicemonitors"]
   verbs: ["get", "list", "watch", "create"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: beyla
subjects:
 - kind: ServiceAccount
   name: beyla
   namespace: beyla
roleRef:
 kind: ClusterRole
 name: beyla
 apiGroup: rbac.authorization.k8s.io
```

Deploy Beyla as a DaemonSet

beyla-config and beyla daemonset: (Beyla 1.6.4)

beyla_network_flow_bytes metric can be enabled using the highlighted section

```
apiVersion: v1
kind: ConfigMap
metadata:
 namespace: beyla
 name: beyla-config
data:
 beyla-config.yml: |
   # this is for beyla_network_flow_bytes metric
   network:
     enable: true
     print_flows: true
   attributes:
     kubernetes:
       enable: true
     select:
       beyla_network_flow_bytes:
         include:
           - beyla.ip
           - src.name
           - dst.port
           - k8s.src.owner.name
           - k8s.src.namespace
           - k8s.dst.owner.name
           - k8s.dst.namespace
           - k8s.cluster.name
   # this will provide automatic routes report while minimizing cardinality
   routes:
     unmatched: heuristic
   #service discovery
   discovery:
     services:
       - k8s_deployment_name: "^docs$"
       - open_ports: 8080-8089
       - k8s_deployment_name: "^website$"
       - k8s_namespace: default
       - k8s_namespace: beyla
       - k8s_deployment_name: "^client-service$"
       - k8s_deployment_name: "^hello-service$"
   otel_traces_export:
     reporters_cache_len: 1024
     sampler:
       name: "always_on"
     endpoint: http://opentelemetrycollector.beyla:4317
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
 namespace: beyla
 name: beyla
spec:
 selector:
   matchLabels:
     instrumentation: beyla
 template:
```

```yaml
  metadata:
    labels:
      instrumentation: beyla
  spec:
    serviceAccountName: beyla
    hostPID: true # mandatory!
    hostNetwork: true
    containers:
      - name: beyla
        image: grafana/beyla:1.6.4
        ports:
        - containerPort: 9092
          hostPort: 9092
          name: http-metrics
          protocol: TCP
        imagePullPolicy: IfNotPresent
        securityContext:
          capabilities:
            add:
              - BPF
              - PERFMON
              - NET_ADMIN
              - SYS_RESOURCE
          privileged: true # mandatory!
          readOnlyRootFilesystem: true
        volumeMounts:
          - mountPath: /config
            name: beyla-config
          - mountPath: /var/run/beyla
            name: var-run-beyla
        env:
          - name: BEYLA_CONFIG_PATH
            value: "/config/beyla-config.yml"
          - name: BEYLA_PROMETHEUS_PORT
            value: "9092"
          - name: BEYLA_PRINT_TRACES
            value: 'true'
          - name: OTEL_EXPORTER_OTLP_ENDPOINT
            value: http://opentelemetrycollector.beyla.svc.cluster.local:4317
    volumes:
      - name: beyla-config
        configMap:
          name: beyla-config
      - name: var-run-beyla
        emptyDir: {}
```

Deploy OpenTelemetry Collector

Service:

```
apiVersion: v1
kind: Service
metadata:
 name: opentelemetrycollector
 namespace: beyla
spec:
 ports:
 - name: grpc-otlp
   port: 4317
   protocol: TCP
   targetPort: 4317
 selector:
   app.kubernetes.io/name: opentelemetrycollector
 type: ClusterIP
```

Configmap:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
 name: collector-config
 namespace: beyla
data:
 collector.yaml: |
   receivers:
     otlp:
       protocols:
         grpc:
           endpoint: ${env:MY_POD_IP}:4317
         http:
           endpoint: ${env:MY_POD_IP}:4318
   processors:
     batch:
     memory_limiter:
       # 80% of maximum memory up to 2G
       limit_mib: 1500
       # 25% of limit up to 2G
       spike_limit_mib: 512
       check_interval: 5s
   extensions:
     zpages: {}
   exporters:
     # logging:
     otlp:
       endpoint: "tempo-distributor.tempo-test:4317"
       insecure: true
   service:
     extensions: [zpages]
     pipelines:
       traces/1:
         receivers: [otlp]
         processors: [memory_limiter, batch]
         exporters: [otlp]
       # metrics:
       #   receivers: [otlp]
       #   processors: [batch]
       #   exporters: [logging]
```

Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: opentelemetrycollector
 namespace: beyla
spec:
 replicas: 1
 selector:
   matchLabels:
     app.kubernetes.io/name: opentelemetrycollector
 template:
   metadata:
     labels:
       app.kubernetes.io/name: opentelemetrycollector
   spec:
     containers:
```

```
     - name: otelcol
       args:
       - --config=/conf/collector.yaml
       image: otel/opentelemetry-collector:0.18.0
       volumeMounts:
       - mountPath: /conf
         name: collector-config
     volumes:
     - configMap:
         items:
         - key: collector.yaml
           path: collector.yaml
         name: collector-config
       name: collector-config
```

Deploy Client-service and Hello-service to generate sample traces and metrics

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: hello-service
 namespace: beyla
spec:
 replicas: 1
 selector:
   matchLabels:
     app: hello-service
 template:
   metadata:
     labels:
       app: hello-service
   spec:
     containers:
     - name: hello-service
       image: hashicorp/http-echo:0.2.3
       args:
       - "-text=Hello from Hello Service"
       ports:
       - containerPort: 5678
---
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
 name: client-service
 namespace: beyla
spec:
 replicas: 1
 selector:
   matchLabels:
     app: client-service
 template:
   metadata:
     labels:
       app: client-service
   spec:
     containers:
     - name: client-service
       image: curlimages/curl:7.83.1
       args:
       - "/bin/sh"
       - "-c"
       - "while true; do curl -s hello-service.beyla; sleep 5; done"
---
apiVersion: v1
kind: Service
metadata:
 name: hello-service
 namespace: beyla
spec:
 selector:
   app: hello-service
 ports:
   - protocol: TCP
     port: 80
     targetPort: 5678
---
apiVersion: v1
kind: Service
metadata:
 name: client-service
 namespace: beyla
spec:
 selector:
   app: client-service
```

```
ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

Deploy Prometheus

1. **Create namespace ‘prometheus’**
2. **Prometheus deployment:**

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: prometheus-server
 namespace: prometheus
spec:
 replicas: 1
 selector:
   matchLabels:
     app: prometheus-server
 template:
   metadata:
     labels:
       app: prometheus-server
   spec:
     containers:
       - name: prometheus
         image: prom/prometheus
         args:
           - "--config.file=/etc/prometheus/prometheus.yml"
           - "--web.enable-remote-write-receiver"
         ports:
           - containerPort: 9090
         volumeMounts:
           - name: config-volume
             mountPath: /etc/prometheus
     volumes:
       - name: config-volume
         configMap:
           name: prometheus-server-conf
           defaultMode: 420
```

3. **Prometheus configmap:**

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: prometheus-server-conf
 namespace: prometheus
data:
 prometheus.yml: |
   global:
     scrape_interval: 15s
     evaluation_interval: 15s
   scrape_configs:
     - job_name: 'beyla'
       static_configs:
         - targets: ['beyla.beyla:9092']
```

4. **Prometheus Service:**

```
apiVersion: v1
kind: Service
metadata:
 name: prometheus-service
 namespace: prometheus
spec:
 selector:
   app: prometheus-server
 ports:
   - protocol: TCP
     port: 80
     targetPort: 9090
 type: LoadBalancer
```

Deploy Grafana

1. **Create namespace grafana**
2. **Grafana Deployment:**

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: grafana
 namespace: grafana
spec:
 replicas: 1
 selector:
   matchLabels:
     app: grafana
 template:
   metadata:
     labels:
       app: grafana
   spec:
     containers:
     - name: grafana
       image: grafana/grafana
       ports:
       - containerPort: 3000
       env:
       - name: GF_SECURITY_ADMIN_PASSWORD
         value: "admin"
---
apiVersion: v1
kind: Service
metadata:
 name: grafana
 namespace: grafana
spec:
 type: LoadBalancer
 ports:
   - port: 80
     targetPort: 3000
     protocol: TCP
 selector:
   app: grafana
```

Deploy Grafana Tempo

1. **Create namespace 'tempo-test'**
2. **Install tempo distributed**

```
Unset
helm -n tempo-test install tempo grafana/tempo-distributed
```

3. **Edit the helm release Values to enable metrics generator:**

Go to the Helm releases in Lens in the tempo-test namespace. Select the tempo release. Edit the Values file:

Add the defaults section to global_overrides: (highlighted)

```
global_overrides:
 defaults:
   metrics_generator:
     processor:
       service_graphs: null
       span_metrics: null
     processors:
     - service-graphs
     - span-metrics
 per_tenant_override_config: /runtime-config/overrides.yaml
```

Under config: storage: set remote_write: -url: (highlighted in red)
“http://prometheus-service.prometheus/api/v1/write”
Also change metricsGenerator.enabled from false to true. (highlighted)

```
metricsGenerator:
 affinity: |
   podAntiAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
       - labelSelector:
           matchLabels:
             {{- include "tempo.selectorLabels" (dict "ctx" . "component" "metrics-generator") | nindent 10 }}
         topologyKey: kubernetes.io/hostname
     preferredDuringSchedulingIgnoredDuringExecution:
       - weight: 100
         podAffinityTerm:
           labelSelector:
             matchLabels:
               {{- include "tempo.selectorLabels" (dict "ctx" . "component" "metrics-generator") | nindent 12 }}
           topologyKey: topology.kubernetes.io/zone
 annotations: {}
 appProtocol:
   grpc: null
 config:
   metrics_ingestion_time_range_slack: 30s
   processor:
     service_graphs:
       dimensions: []
       histogram_buckets:
       - 0.1
       - 0.2
       - 0.4
```

```
      - 0.8
      - 1.6
      - 3.2
      - 6.4
      - 12.8
      max_items: 10000
      wait: 10s
      workers: 10
    span_metrics:
      dimensions: []
      histogram_buckets:
      - 0.002
      - 0.004
      - 0.008
      - 0.016
      - 0.032
      - 0.064
      - 0.128
      - 0.256
      - 0.512
      - 1.02
      - 2.05
      - 4.1
  registry:
    collection_interval: 15s
    external_labels: {}
    stale_duration: 15m
  storage:
    path: /var/tempo/wal
    remote_write:
    - url: http://prometheus-service.prometheus/api/v1/write
    remote_write_add_org_id_header: true
    remote_write_flush_deadline: 1m
    wal: null
  traces_storage:
    path: /var/tempo/traces
enabled: true
extraArgs: []
extraEnv: []
extraEnvFrom: []
extraVolumeMounts: []
extraVolumes: []
```

```
hostAliases: []
image:
  pullSecrets: []
  registry: null
  repository: null
  tag: null
initContainers: []
kind: Deployment
nodeSelector: {}
persistence:
  annotations: {}
  enabled: false
  size: 10Gi
  storageClass: null
podAnnotations: {}
podLabels: {}
ports:
- name: grpc
  port: 9095
  service: true
- name: http-memberlist
  port: 7946
  service: false
- name: http-metrics
  port: 3100
  service: true
priorityClassName: null
replicas: 1
resources: {}
service:
  annotations: {}
terminationGracePeriodSeconds: 300
tolerations: []
topologySpreadConstraints: |
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        {{- include "tempo.selectorLabels" (dict "ctx" . "component"
"metrics-generator") | nindent 6 }}
walEmptyDir: {}
```

Save and deploy a new release of tempo with the edited Values. Metrics Generator pods should now appear in the tempo-test namespace.

Create prometheus and tempo data sources in Grafana.

Prometheus: http://prometheus-service.prometheus:80
Tempo: http://tempo-query-frontend.tempo-test:3100
Tempo data source additional settings:

- Trace to metrics data source - prometheus
- Additional settings - service graph data source - prometheus

View Service Graphs

Go to the Explore tab in Grafana. Click on the service graph tab in the query.

IF OTEL COLLECTOR OR OTHER PODS KEEP GETTING EVICTED/KEEP RESTARTING DUE TO MEMORY PRESSURE:

- **INCREASE NODE POOL SIZE OF THE GKE CLUSTER.**
- **REMOVE THE BEYLA NAMESPACE WATCH FROM THE BEYLA CONFIGMAP**

Change these settings to the following in the tempo helm chart if there is some ResourceExhausted error in the OTEL collector logs when sending to tempo:

Unset

```
server:

  grpc_server_max_recv_msg_size: 123412341234123

  grpc_server_max_send_msg_size: 123412341234123
```