Grafana Beyla Sidecar Deployment

Prerequisites

- Kubernetes cluster - kubeadm (1 master + 2 workers) nodes should be ready.

```
pranavpp37@k8s-master:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE     VERSION
k8s-master    Ready    control-plane   7d13h   v1.28.15
k8s-worker1   Ready    <none>          7d13h   v1.28.15
k8s-worker2   Ready    <none>          7d13h   v1.28.15
pranavpp37@k8s-master:~$
```

- pranavpp37/my-java-app:latest Docker image on all nodes

```
pranavpp37@k8s-master:~$ sudo docker images | grep pranavpp37/my-java-app:latest
pranavpp37/my-java-app:latest        aed06dfe0a49        460MB             0B
WARNING: This output is designed for human readability. For machine-readable output, please use --format.
pranavpp37@k8s-master:~$
```

Note: pranavpp37 – username.

- kubectl configured

STEPS

Step – 1: Create Namespace

```
kubectl create namespace monitoring
```

Step-2: **Deploy Beyla Sidecar Application**

1. **beyla-rbac**

```
# 1. RBAC for Beyla

cat > 01-beyla-rbac.yaml << 'EOF'

apiVersion: v1

kind: ServiceAccount

metadata:

  name: beyla

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:
```

```
  name: beyla
rules:
- apiGroups: [""]
  resources: ["pods", "services", "nodes"]
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: beyla
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: beyla
subjects:
- kind: ServiceAccount
  name: beyla
  namespace: default
EOF


kubectl apply -f 01-beyla-rbac.yaml
```

What it does?

Job: Grants Beyla ServiceAccount permissions to discover K8s metadata

Creates:

ServiceAccount: beyla

ClusterRole: beyla (pods/services/nodes read)

ClusterRoleBinding: beyla → default namespace

Flow: RBAC FIRST → Beyla can enrich metrics with pod/deployment labels.

2. Java App + Beyla Sidecar

```
# 2. Java App + Beyla Sidecar
cat > 02-my-java-app.yaml << 'EOF'
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-java-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-java-app
  template:
    metadata:
      labels:
        app: my-java-app
    spec:
      serviceAccountName: beyla
      shareProcessNamespace: true
      containers:
      - name: java-app
        image: pranavpp37/my-java-app:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
      - name: beyla
        image: grafana/beyla:latest
        securityContext:
          privileged: true
          capabilities:
            add: ["SYS_ADMIN", "NET_ADMIN", "BPF"]
```

```
        env:
        - name: BEYLA_OPEN_PORT
          value: "8080"
        - name: BEYLA_SERVICE_NAME
          value: "my-java-app"
        - name: BEYLA_LOG_LEVEL
          value: "info"
        - name: BEYLA_PROMETHEUS_EXPORT
          value: "true"
        - name: BEYLA_PROMETHEUS_PORT
          value: "9999"
        - name: BEYLA_OTEL_TRACES_EXPORT
          value: "true"
        - name: OTEL_EXPORTER_OTLP_ENDPOINT
          value: "http://jaeger.monitoring.svc.cluster.local:4317"
        - name: OTEL_EXPORTER_OTLP_PROTOCOL
          value: "grpc"
        - name: OTEL_SERVICE_NAME
          value: "my-java-app"
        - name: OTEL_TRACES_SAMPLER
          value: "always_on"
        - name: OTEL_TRACES_SAMPLER_ARG
          value: "1"
        - name: BEYLA_OTEL_METRICS_EXPORT
          value: "false"
        - name: BEYLA_EXECUTABLE_NAME
          value: "java"
        ports:
        - containerPort: 9999
          name: metrics
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: my-java-app
spec:
  selector:
    app: my-java-app
  ports:
  - port: 8080
    targetPort: 8080
    name: http
    nodePort: 30081
  - port: 9999
    targetPort: 9999
    name: metrics
    nodePort: 30100
  type: NodePort
EOF


kubectl apply -f 02-my-java-app.yaml
```

What it does?

Job: Deploys your Spring Boot app + Beyla eBPF sidecar

Creates:

Deployment: my-java-app (2 containers)

java-app: pranavpp37/my-java-app:latest (port 8080)

beyla: grafana/beyla:latest (port 9999, privileged)

Service: my-java-app (NodePorts 30081+30100)

Flow: Beyla eBPF attaches to Java process → captures HTTP metrics/traces.

Step-3: Deploying monitoring stack

1. Prometheus

```
# 1. Prometheus (with Beyla scrape config)
cat > prometheus-stack.yaml << 'EOF'
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: monitoring
data:
  prometheus.yml: |
    global:
      scrape_interval: 15s
    scrape_configs:
    - job_name: 'beyla'
      static_configs:
      - targets: ['my-java-app.default.svc.cluster.local:9999']
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
  template:
    metadata:
      labels:
```

```
        app: prometheus
    spec:
      containers:
      - name: prometheus
        image: prom/prometheus:latest
        args:
        - "--config.file=/etc/prometheus/prometheus.yml"
        - "--storage.tsdb.path=/prometheus"
        - "--web.enable-lifecycle"
        ports:
        - containerPort: 9090
        volumeMounts:
        - name: config
          mountPath: /etc/prometheus
        - name: storage
          mountPath: /prometheus
      volumes:
      - name: config
        configMap:
          name: prometheus-config
      - name: storage
        emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: prometheus
  namespace: monitoring
spec:
  selector:
    app: prometheus
  ports:
```

```
  - port: 9090
    targetPort: 9090
  type: NodePort
EOF
kubectl apply -f prometheus-stack.yaml
```

What it does?

Job: Prometheus scrapes Beyla metrics endpoint

Creates:

ConfigMap: prometheus-config (beyla job: my-java-app:9999)

Deployment: prometheus (scrape every 15s)

Service: prometheus (NodePort)

Flow: Beyla metrics → Prometheus → Grafana queries.

2. Jaeger

```
cat > jaeger-stack.yaml << 'EOF'
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jaeger
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jaeger
  template:
    metadata:
      labels:
        app: jaeger
    spec:
      containers:
```

```
      - name: jaeger
        image: jaegertracing/all-in-one:latest
        ports:
        - containerPort: 16686
        - containerPort: 4317
        env:
        - name: COLLECTOR_OTLP_ENABLED
          value: "true"
---
apiVersion: v1
kind: Service
metadata:
  name: jaeger
  namespace: monitoring
spec:
  selector:
    app: jaeger
  ports:
  - name: ui
    port: 16686
    targetPort: 16686
    nodePort: 31020
  - name: otlp-grpc
    port: 4317
    targetPort: 4317
  type: NodePort
EOF

kubectl apply -f jaeger-stack.yaml
```

What it does?

Job: Receives Beyla OTLP traces (port 4317)

Creates:

Deployment: jaeger (all-in-one)

Service: jaeger (UI:31020, OTLP:4317)

Flow: Beyla traces → Jaeger → UI visualization

3. Grafana Dashboards

```
cat > grafana-stack.yaml << 'EOF'
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grafana
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: grafana
  template:
    metadata:
      labels:
        app: grafana
    spec:
      containers:
      - name: grafana
        image: grafana/grafana:10.2.0
        ports:
        - containerPort: 3000
        env:
```

```
        - name: GF_SECURITY_ADMIN_USER
          value: admin
        - name: GF_SECURITY_ADMIN_PASSWORD
          value: admin123
        volumeMounts:
        - name: grafana-storage
          mountPath: /var/lib/grafana
      volumes:
      - name: grafana-storage
        emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: grafana
  namespace: monitoring
spec:
  selector:
    app: grafana
  ports:
  - port: 3000
    targetPort: 3000
  type: NodePort
EOF


kubectl apply -f grafana-stack.yaml


sleep 60
kubectl get pods -n monitoring
```

What it does?

Job: Visualizes Prometheus metrics (Dashboard 19923)

Creates:

Deployment: grafana (admin/admin123)

Service: grafana (NodePort)

Flow: Prometheus data → Grafana → Live HTTP graphs.

Step-4: Verify

```
# Check all pods

kubectl get pods -A
```

```
pranavpp37@k8s-master:~$ kubectl get pods -A
NAMESPACE     NAME                                 READY   STATUS    RESTARTS       AGE
default       my-java-app-86db499486-5fkp9         2/2     Running   0              15m
kube-system   cilium-9ghwk                         1/1     Running   8 (4d1h ago)   7d13h
kube-system   cilium-lcz7f                         1/1     Running   8 (4d1h ago)   7d13h
kube-system   cilium-operator-89b79bd9f-g57ln      1/1     Running   8 (4d1h ago)   7d13h
kube-system   cilium-psjn4                         1/1     Running   8 (4d1h ago)   7d13h
kube-system   coredns-5dd5756b68-k7wqg             1/1     Running   8 (4d1h ago)   7d13h
kube-system   coredns-5dd5756b68-wqcm2             1/1     Running   8 (4d1h ago)   7d13h
kube-system   etcd-k8s-master                      1/1     Running   8 (4d1h ago)   7d13h
kube-system   kube-apiserver-k8s-master            1/1     Running   8 (4d1h ago)   7d13h
kube-system   kube-controller-manager-k8s-master   1/1     Running   8 (4d1h ago)   7d13h
kube-system   kube-proxy-4rdgp                     1/1     Running   8 (4d1h ago)   7d13h
kube-system   kube-proxy-ssrdr                     1/1     Running   8 (4d1h ago)   7d13h
kube-system   kube-proxy-zlx99                     1/1     Running   8 (4d1h ago)   7d13h
kube-system   kube-scheduler-k8s-master            1/1     Running   8 (4d1h ago)   7d13h
monitoring    grafana-689b99599f-nsvgq             1/1     Running   0              13m
monitoring    jaeger-5b6586d675-7q2j6              1/1     Running   0              13m
monitoring    prometheus-58779ccf44-rb8r4          1/1     Running   0              13m
```

Step-5: Generate Traffic Once Again and Watch

```
MASTER_IP=$(hostname -I | awk '{print $1}')

APP_PORT=$(kubectl get svc my-java-app -o jsonpath='{.spec.ports[0].nodePort}')

METRICS_PORT=$(kubectl get svc my-java-app -o
jsonpath='{.spec.ports[1].nodePort}')


# Send 50 requests

for i in {1..50}; do

  curl "http://$MASTER_IP:$APP_PORT/hello?test=$i"

  sleep 0.2

done
```

```
# Check updated metrics (counters will increment!)

curl "http://$MASTER_IP:$METRICS_PORT/metrics" | grep
http_server_request_body_size_bytes | tail -5
```

```
# Check updated metrics (counters will increment!)
curl "http://$MASTER_IP:$METRICS_PORT/metrics" | grep http_server_request_body_size_bytes | tail -5
Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring
Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Sp
ring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello fr
om Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hel
lo from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boo
t!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Sprin
g Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from Spring Boot!Hello from
Spring Boot!Hello from Spring Boot!Hello from Spring Boot!  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 36610    0 36610    0     0  11.4M      0 --:--:-- --:--:-- --:--:-- 17.4M
http_server_request_body_size_bytes_bucket{http_request_method="GET",http_response_status_code="200",http_route="/hello",instance="default.my-java-app-86db4
99486-5fkp9.java-app",job="default/my-java-app",k8s_cluster_name="",k8s_container_name="java-app",k8s_cronjob_name="",k8s_daemonset_name="",k8s_deployment_n
ame="my-java-app",k8s_job_name="",k8s_kind="Deployment",k8s_namespace_name="default",k8s_node_name="k8s-worker1",k8s_owner_name="my-java-app",k8s_pod_name="
my-java-app-86db499486-5fkp9",k8s_pod_start_time="2025-12-09 05:42:22 +0000 UTC",k8s_pod_uid="6da22e65-53ed-4366-9eeb-2544bed9130c",k8s_replicaset_name="my-
java-app-86db499486",k8s_statefulset_name="",server_address="my-java-app",server_port="8080",service_name="my-java-app",service_namespace="default",le="4096
"} 145
http_server_request_body_size_bytes_bucket{http_request_method="GET",http_response_status_code="200",http_route="/hello",instance="default.my-java-app-86db4
99486-5fkp9.java-app",job="default/my-java-app",k8s_cluster_name="",k8s_container_name="java-app",k8s_cronjob_name="",k8s_daemonset_name="",k8s_deployment_n
ame="my-java-app",k8s_job_name="",k8s_kind="Deployment",k8s_namespace_name="default",k8s_node_name="k8s-worker1",k8s_owner_name="my-java-app",k8s_pod_name="
my-java-app-86db499486-5fkp9",k8s_pod_start_time="2025-12-09 05:42:22 +0000 UTC",k8s_pod_uid="6da22e65-53ed-4366-9eeb-2544bed9130c",k8s_replicaset_name="my-
java-app-86db499486",k8s_statefulset_name="",server_address="my-java-app",server_port="8080",service_name="my-java-app",service_namespace="default",le="8192
"} 145
http_server_request_body_size_bytes_bucket{http_request_method="GET",http_response_status_code="200",http_route="/hello",instance="default.my-java-app-86db4
99486-5fkp9.java-app",job="default/my-java-app",k8s_cluster_name="",k8s_container_name="java-app",k8s_cronjob_name="",k8s_daemonset_name="",k8s_deployment_n
ame="my-java-app",k8s_job_name="",k8s_kind="Deployment",k8s_namespace_name="default",k8s_node_name="k8s-worker1",k8s_owner_name="my-java-app",k8s_pod_name="
my-java-app-86db499486-5fkp9",k8s_pod_start_time="2025-12-09 05:42:22 +0000 UTC",k8s_pod_uid="6da22e65-53ed-4366-9eeb-2544bed9130c",k8s_replicaset_name="my-
java-app-86db499486",k8s_statefulset_name="",server_address="my-java-app",server_port="8080",service_name="my-java-app",service_namespace="default",le="+Inf
"} 145
http_server_request_body_size_bytes_sum{http_request_method="GET",http_response_status_code="200",http_route="/hello",instance="default.my-java-app-86db49948
6-5fkp9.java-app",job="default/my-java-app",k8s_cluster_name="",k8s_container_name="java-app",k8s_cronjob_name="",k8s_daemonset_name="",k8s_deployment_name
="my-java-app",k8s_job_name="",k8s_kind="Deployment",k8s_namespace_name="default",k8s_node_name="k8s-worker1",k8s_owner_name="my-java-app",k8s_pod_name="my-
java-app-86db499486-5fkp9",k8s_pod_start_time="2025-12-09 05:42:22 +0000 UTC",k8s_pod_uid="6da22e65-53ed-4366-9eeb-2544bed9130c",k8s_replicaset_name="my-java
-app-86db499486",k8s_statefulset_name="",server_address="my-java-app",server_port="8080",service_name="my-java-app",service_namespace="default"} 36544
http_server_request_body_size_bytes_count{http_request_method="GET",http_response_status_code="200",http_route="/hello",instance="default.my-java-app-86db49
9486-5fkp9.java-app",job="default/my-java-app",k8s_cluster_name="",k8s_container_name="java-app",k8s_cronjob_name="",k8s_daemonset_name="",k8s_deployment_na
me="my-java-app",k8s_job_name="",k8s_kind="Deployment",k8s_namespace_name="default",k8s_node_name="k8s-worker1",k8s_owner_name="my-java-app",k8s_pod_name="my
-java-app-86db499486-5fkp9",k8s_pod_start_time="2025-12-09 05:42:22 +0000 UTC",k8s_pod_uid="6da22e65-53ed-4366-9eeb-2544bed9130c",k8s_replicaset_name="my-j
ava-app-86db499486",k8s_statefulset_name="",server_address="my-java-app",server_port="8080",service_name="my-java-app",service_namespace="default"} 145
pranavpp37@k8s-master:~$
```

Note:

http_server_request_body_size_bytes_bucket{le="+Inf"} 145 → 145 REQUESTS CAPTURED

http_server_request_body_size_bytes_sum{} 36544 → 36KB total request data

http_server_request_body_size_bytes_count{} 145 → EXACT REQUEST COUNT

Verify once again

```
kubectl logs deployment/my-java-app -c beyla | grep -i
"instrumenting\|java\|HTTP\|tracer"
```

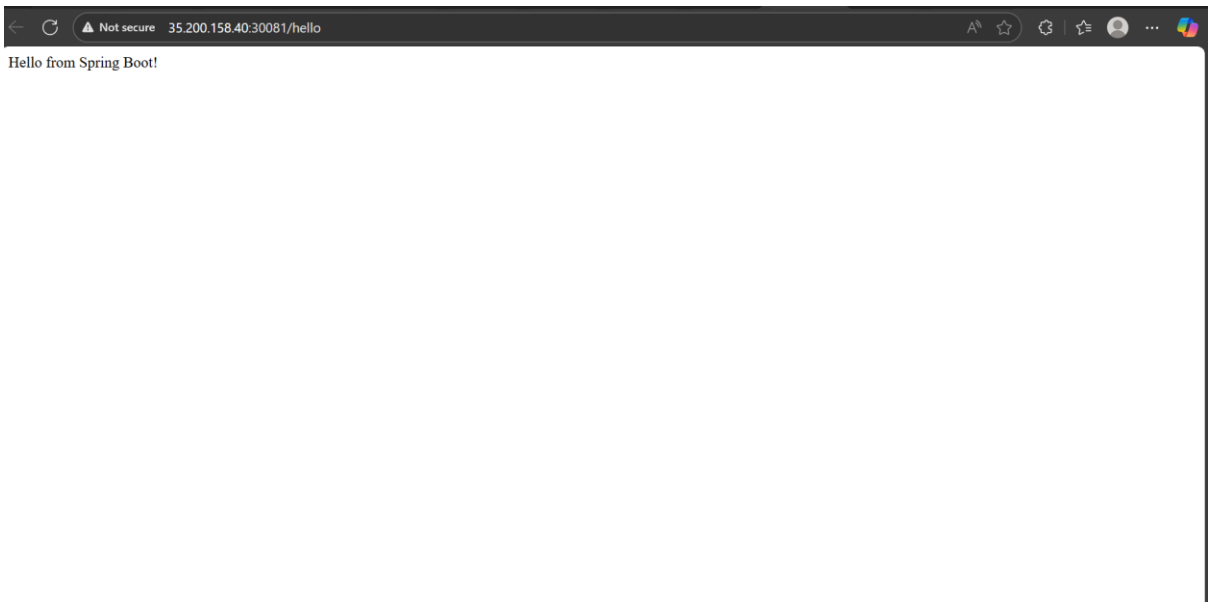
```
pranavpp37@k8s-master:~$ kubectl logs deployment/my-java-app -c beyla | grep -i "instrumenting\|java\|HTTP\|tracer"
SDK 2025/12/09 05:42:27 WARN falling back to IMDSv1: operation error ec2imds: getToken, http response error StatusCode: 405, request to EC2 IMDS failed
time=2025-12-09T05:42:27.276Z level=INFO msg="using hostname" component=traces.ReadDecorator function=instance_ID_hostNamePIDDecorator hostname=my-java-app-
86db499486-5fkp9
time=2025-12-09T05:42:29.979Z level=INFO msg="using hostname" component=traces.ReadDecorator function=instance_ID_hostNamePIDDecorator hostname=my-java-app-
86db499486-5fkp9
time=2025-12-09T05:42:45.545Z level=INFO msg="instrumenting process" component=discover.TraceAttacher cmd=/opt/java/openjdk/bin/java pid=7 ino=1297737 typej
ava service=my-java-app
time=2025-12-09T05:42:47.976Z level=INFO msg="Launching p.Tracer" component=generic.Tracer
time=2025-12-09T05:59:02.979Z level=WARN msg="error flushing evicted metrics provider" component=otel.MetricsReporter service="{Name:my-java-app Namespace:d
efault Instance:default.my-java-app-86db499486-5fkp9.java-app}" error="failed to upload metrics: rpc error: code = Unimplemented desc = unknown service open
telemetry.proto.collector.metrics.v1.MetricsService"
pranavpp37@k8s-master:~$
```

Step-6: Access URLs

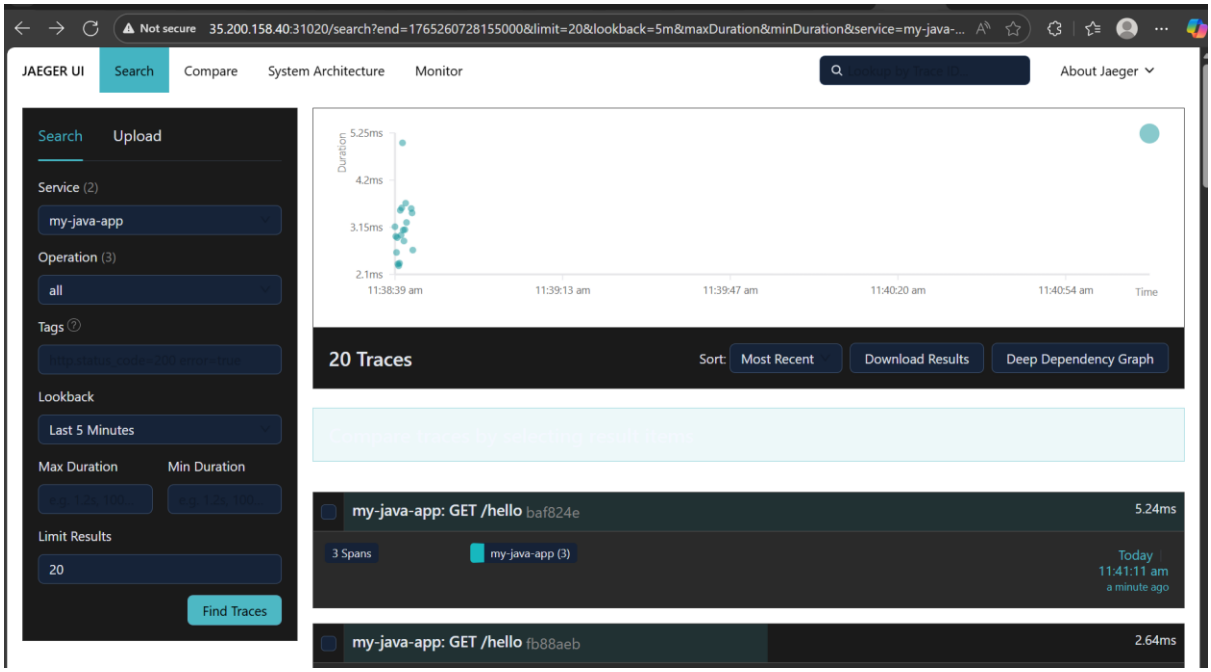
```
MASTER_IP=$(hostname -I | awk '{print $1}')

APP_PORT=$(kubectl get svc my-java-app -o jsonpath='{.spec.ports[0].nodePort}')

METRICS_PORT=$(kubectl get svc my-java-app -o
jsonpath='{.spec.ports[1].nodePort}')

cat << EOF

LIVE BEYLA OBSERVABILITY (COMPLETE)

Java App:           http://$MASTER_IP:$APP_PORT/hello

Beyla Metrics:      http://$MASTER_IP:$METRICS_PORT/metrics

Grafana:            http://$MASTER_IP:$(kubectl get svc -n monitoring grafana -o
jsonpath='{.spec.ports[0].nodePort}') (admin/admin123)

Prometheus:         http://$MASTER_IP:$(kubectl get svc -n monitoring prometheus -
o jsonpath='{.spec.ports[0].nodePort}')

Jaeger Traces:      http://$MASTER_IP:31020

EOF
```

Note: Master IP should be your external IP of instance.

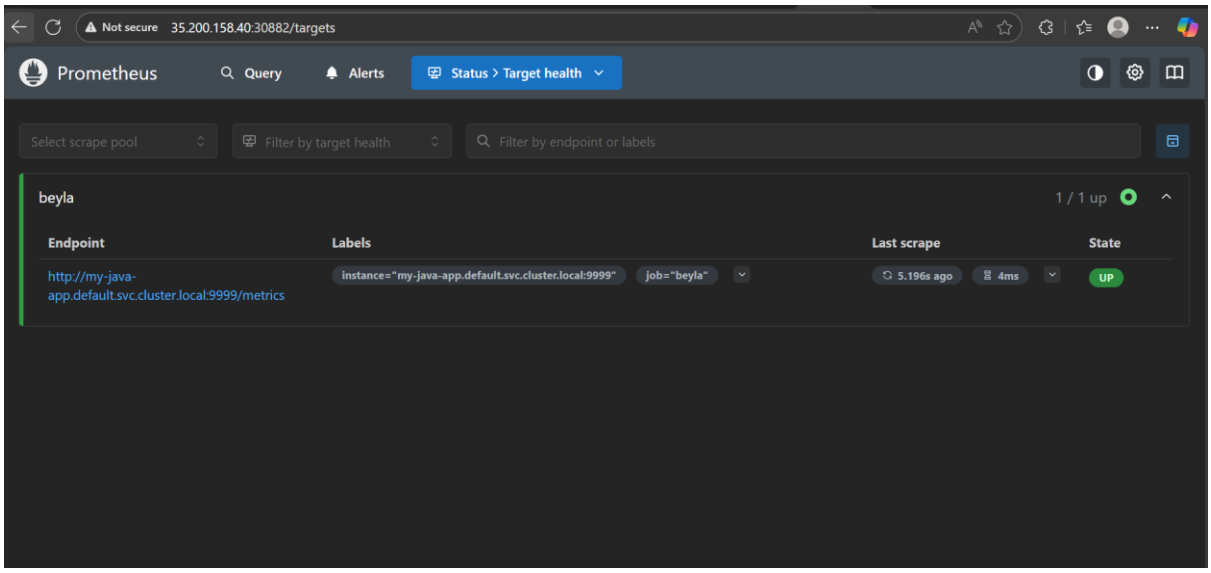
i) Java App:

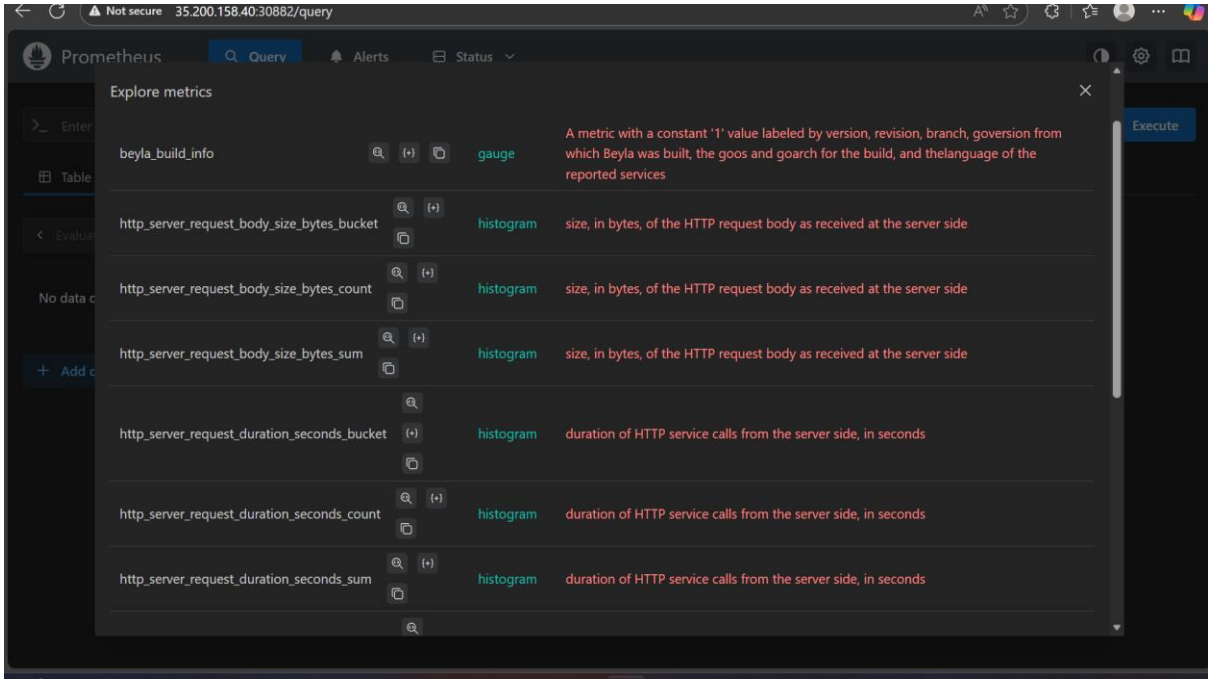


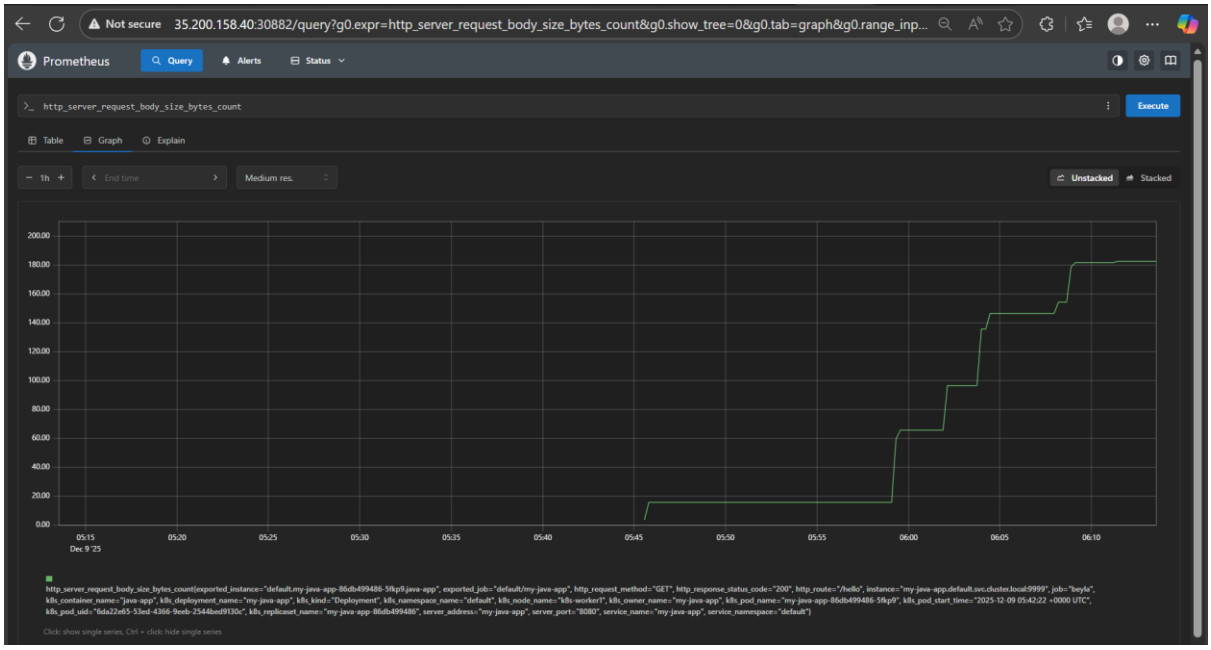
ii) Jaeger



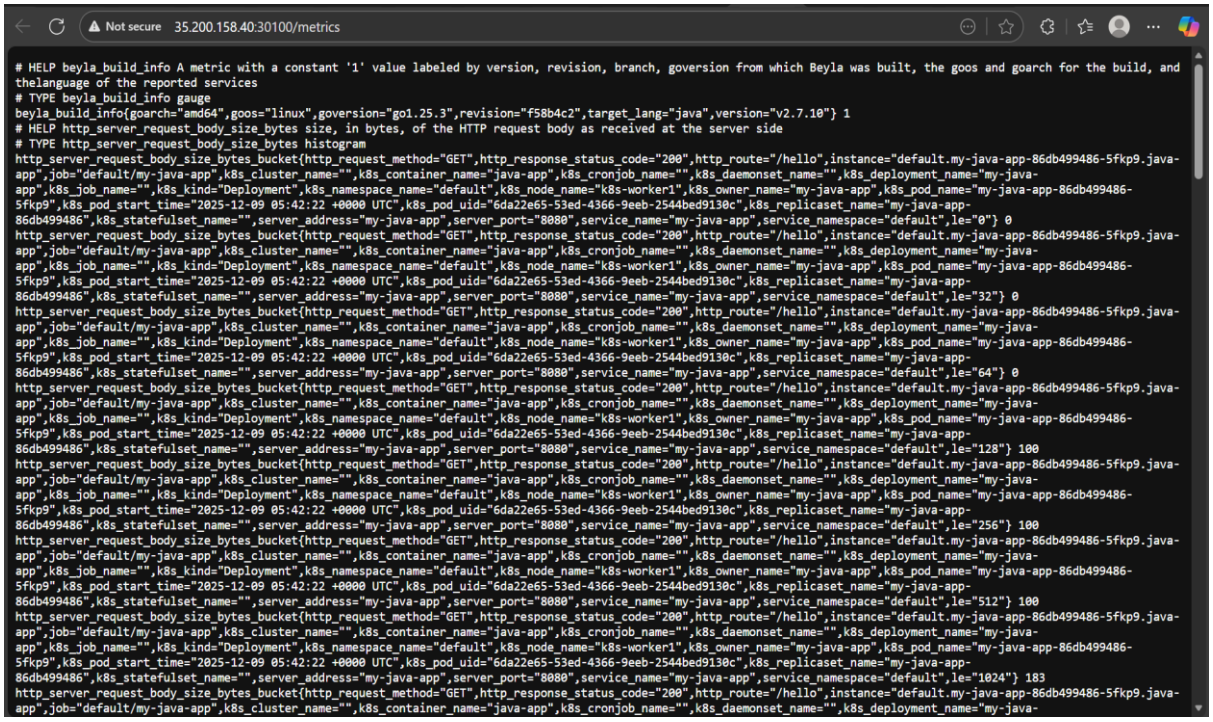
iii) Prometheus




Not secure 35.200.158.40:30882/query
Prometheus
Query
Alerts
Status
Explore metrics
Execute
Table
No data
beyla_build_info gauge A metric with a constant '1' value labeled by version, revision, branch, goversion from which Beyla was built, the goos and goarch for the build, and thelanguage of the reported services
http_server_request_body_size_bytes_bucket histogram size, in bytes, of the HTTP request body as received at the server side
http_server_request_body_size_bytes_count histogram size, in bytes, of the HTTP request body as received at the server side
http_server_request_body_size_bytes_sum histogram size, in bytes, of the HTTP request body as received at the server side
http_server_request_duration_seconds_bucket histogram duration of HTTP service calls from the server side, in seconds
http_server_request_duration_seconds_count histogram duration of HTTP service calls from the server side, in seconds
http_server_request_duration_seconds_sum histogram duration of HTTP service calls from the server side, in seconds


Not secure 35.200.158.40:30882/query?g0.expr=http_server_request_body_size_bytes_count&g0.show_tree=0&g0.tab=graph&g0.range_inp...
Prometheus
Query
Alerts
Status
http_server_request_body_size_bytes_count
Execute
Table
Graph
Explain
1h
Medium res.
Unstacked
Stacked
200.00
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
05:15
Dec 9 '25
05:20
05:25
05:30
05:35
05:40
05:45
05:50
05:55
06:00
06:05
06:10

iv) Beyla Metrics



v) Grafana

a) Open access url of Grafana.

Username: admin

Password: admin123

b) Choose Datasource – Prometheus

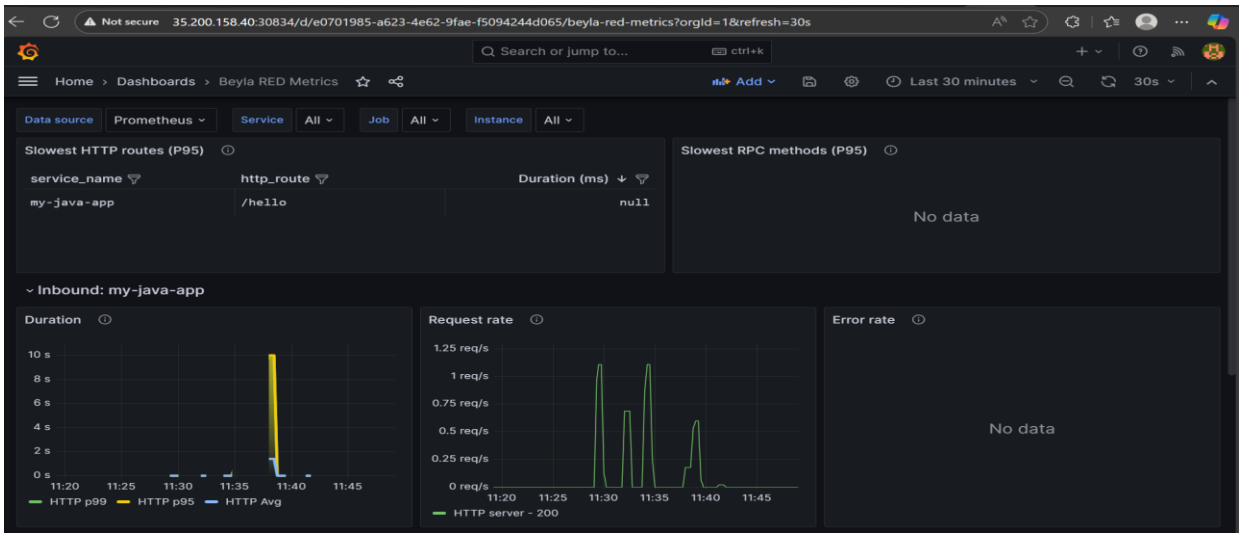
Set url: http://prometheus.monitoring.svc.cluster.local:9090

Save and Test.

c) Choose Dashboard

Import dashboard id: 19923

Choose Prometheus Datasource configured.



OVERALL FLOW

Beyla Observability Data Flow

- **HTTP Traffic → Java App (8080)**
    - Browser/curl hits http://10.160.0.6:30081/hello
    - Spring Boot app receives GET /hello?test=123
    - Returns "Hello from Spring Boot!" (200 OK)
- **eBPF Sidecar Interception (Zero-Code)**
    - Beyla container (port 9999) shares process namespace
    - Attaches to Java PID via eBPF kernel hooks
    - Captures: method=GET, route=/hello, status=200, body_size=256B
    - Example→ Result: http_server_request_body_size_bytes_count=182
- **Beyla Metrics Export (Port 9999)**
    - Exposes /metrics endpoint: http://10.160.0.6:30100/metrics
    - Prometheus scrapes every 15s: my-java-app.default.svc:9999
    - Stores time-series with K8s labels (pod/deployment/node)
- **OTLP Traces Export (Parallel Path)**
    - BEYLA_OTEL_TRACES_EXPORT=true → Jaeger port 4317
    - gRPC spans: GET /hello (50ms) → service=my-java-app
    - Full trace waterfalls with pod metadata
- **Visualization Layer**
    - **Prometheus**: Raw metrics storage + queries
    - **Jaeger**: http://10.160.0.6:31020 → Trace UI
    - **Grafana Dashboard 19923**: Live graphs (e.g., 182 requests)
- **Final Result**
    - Example→182 GET /hello requests visualized
    - P95 latency, RPS, error rate (0%)
    - Zero Java code changes required
    - Auto-scales with pod replicas.