GRAFANA BEYLA (DAEMONSET)

1. Objective

Set up an observability stack for **Spring Boot applications** running on Kubernetes to:

- Collect **application metrics** (HTTP, gRPC, SQL, Redis, Messaging)
- Collect **process metrics** (CPU, memory, disk I/O, network)
- Expose metrics via **Prometheus**
- Visualize metrics in **Grafana**
- Ensure **high observability** and performance monitoring with minimal code changes

This setup is **non-intrusive**, using a **DaemonSet (Beyla)** to instrument applications automatically.

2. Components

Component	Purpose	Notes
Kubernetes Namespace	Isolate monitoring resources	namespace: beyla
ServiceAccount & RBAC	Grant Beyla permission to read Kubernetes metadata	Required for pod/service discovery
ConfigMap (Beyla)	Configuration file for instrumentation	/config/beyla-config.yml
DaemonSet (Beyla)	Runs the Beyla agent on every node	Instruments Java apps on port 8080
Prometheus	Scrapes metrics from Beyla	Exposed on port 9090
Grafana	Visualizes Prometheus metrics	Exposed on port 3000
Spring Boot Deployment	Application to monitor	Exposed on port 8080

3. Prerequisites

Cluster Requirements

- Kubernetes v1.21+ (supports Lease API for leader election)
- RBAC enabled
- Ability to deploy DaemonSets
- NodePort or Ingress access for Prometheus/Grafana

- Minimum resources per node:
    - 500m CPU, 1GB RAM for monitoring stack (Prometheus + Beyla)

Beyla Requirements

- Privileged mode enabled (SYS_ADMIN, SYS_PTRACE, BPF, etc.)
- Host PID access (hostPID: true) to instrument all processes
- Mount config files via ConfigMap

Proof:

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

```
pranavpp37@k8s-master:~/beyla$ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
k8s-master     Ready    control-plane   12d   v1.29.15
k8s-worker-1   Ready    <none>          12d   v1.29.15
k8s-worker-2   Ready    <none>          12d   v1.29.15
```

- pranavpp37/spring-boot-app: Docker image on all nodes

```
pranavpp37@k8s-master:~/beyla$ sudo docker images -a | grep pranavpp37/spring-boot-app
pranavpp37/spring-boot-app              v1            29ebbb35c055   8 days ago    204MB
pranavpp37@k8s-master:~/beyla$
```

- kubectl configured.

IMPLEMENTATION

Step – 1: Create Namespace

```
kubectl create namespace beyla
```

Step-2: Deploy Beyla RBAC Permissions

```
01-beyla-namespace-rbac.yaml

--

apiVersion: v1

kind: Namespace

metadata:

  name: beyla

---
```

```yaml
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"]
---
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
```

Apply: kubectl apply -f 01-beyla-namespace-rbac.yaml

Step-3: Deploy ConfigMap

```
02-beyla-configmap.yaml
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: beyla-config
  namespace: beyla
data:
  beyla-config.yml: |
    # ============================================
    # BEYLA CONFIGURATION - PRODUCTION READY
    # ============================================

    # Enable Kubernetes metadata decoration
    # This adds pod names, namespaces, deployments to metrics
    attributes:
      kubernetes:
        enable: true

    # Configure routes to minimize cardinality
    # This groups similar URLs together (e.g., /user/123 -> /user/{id})
    routes:
      unmatched: heuristic

    # ============================================
    # SERVICE DISCOVERY - CRITICAL SECTION
    # ============================================
    # This tells Beyla which applications to instrument
    discovery:
```

```
  services:
    # Find Java applications listening on port 8080
    # THIS WILL WORK - Your Spring Boot app uses port 8080
    - open_ports: 8080
      name: spring-boot-app

# ============================================
# OPENTELEMETRY METRICS EXPORT
# ============================================
otel_metrics_export:
  # If you have an OTEL collector, set endpoint here
  # Otherwise, leave as-is (uses env var)
  endpoint: ${OTEL_EXPORTER_OTLP_ENDPOINT}
  interval: 30s

  # ============================================
  # FEATURES - WHAT METRICS TO COLLECT
  # ============================================
  features:
    # WILL WORK - HTTP/gRPC RED metrics
    # Collects: request rate, errors, duration
    - application

    # WILL WORK - Process metrics
    # Collects: CPU, memory, disk I/O, network I/O
    - application_process

    # WILL WORK - Span-level metrics
    # Provides detailed per-request telemetry
    - application_span

    # WILL WORK (if you have multiple services)
```

```
    # Shows service-to-service communication
    # For single app, may have limited data
    - application_service_graph

    # WON'T GENERATE METRICS - Your app has no database
    # UNCOMMENT ONLY IF YOU ADD DATABASE TO YOUR APP
    # - sql

    # WON'T GENERATE METRICS - Your app doesn't use Redis
    # UNCOMMENT ONLY IF YOU ADD REDIS TO YOUR APP
    # - redis

    # WON'T GENERATE METRICS - Your app doesn't use MongoDB
    # UNCOMMENT ONLY IF YOU ADD MONGODB TO YOUR APP
    # - mongo

    # WON'T GENERATE METRICS - Your app doesn't use Kafka
    # UNCOMMENT ONLY IF YOU ADD KAFKA TO YOUR APP
    # - kafka

    # WON'T GENERATE METRICS - Your app doesn't use GPU
    # UNCOMMENT ONLY IF YOU HAVE GPU WORKLOADS
    # - gpu

# ==========================================
# PROMETHEUS EXPORT - LOCAL SCRAPING
# ==========================================
prometheus_export:
  port: 9090
  path: /metrics

  # Same features as OTEL export
```

```
  features:
    - application                # HTTP metrics
    - application_process        # Process metrics
    - application_span           # Span metrics
    - application_service_graph  # Service graph
    # - sql                      # Disabled - no database
    # - redis                    # Disabled - no Redis
    # - mongo                    # Disabled - no MongoDB
    # - kafka                    # Disabled - no Kafka


# ============================================
# PROCESS METRICS CONFIGURATION
# ============================================
processes:
  enabled: true
  # These metrics will be collected:
  # process_cpu_time_seconds_total
  # process_cpu_utilization_ratio
  # process_memory_usage_bytes
  # process_memory_virtual_bytes
  # process_disk_io_bytes_total
  # process_network_io_bytes_total


# ============================================
# NETWORK METRICS - CURRENTLY DISABLED
# ============================================
# Network flow metrics are disabled to avoid conflicts
# Enable this ONLY if you want pod-to-pod network flows
# NOTE: Enabling this may put Beyla in network-only mode
# network:
#   enable: true
#
```

```
Apply: kubectl apply -f 02-beyla-configmap.yaml
```

Step-4: Deploy SpringBoot Java Application

```
03-java-app-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-boot-app
  namespace: default
  labels:
    app: spring-boot-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: spring-boot-app
  template:
    metadata:
      labels:
        app: spring-boot-app
    spec:
      containers:
      - name: spring-boot
        image: pranavpp37/spring-boot-app:v1
        imagePullPolicy: IfNotPresent  # Use local image, don't pull from registry
        ports:
        - containerPort: 8080
          name: http
        resources:
          requests:
```

```
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "1Gi"
            cpu: "500m"
---
apiVersion: v1
kind: Service
metadata:
  name: spring-boot-svc
spec:
  type: NodePort
  selector:
    app: spring-boot-app
  ports:
  - port: 8080
    targetPort: 8080


Apply: kubectl apply -f 03-java-app-deployment.yaml
```

Step-5: Deploy Beyla Daemonset

```
04-beyla-daemonset.yaml
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: beyla
  namespace: beyla
spec:
  selector:
    matchLabels:
```

```
    instrumentation: beyla
template:
  metadata:
    labels:
      instrumentation: beyla
  spec:
    serviceAccountName: beyla
    hostPID: true  # Required to see all processes on the node
    containers:
    - name: beyla
      image: grafana/beyla:latest
      securityContext:
        privileged: true  # Required for eBPF operations
        capabilities:
          add:
            - SYS_ADMIN
            - SYS_PTRACE
            - SYS_RESOURCE
            - NET_ADMIN
            - NET_RAW
            - BPF
            - PERFMON
      env:
      # ============================================
      # CRITICAL: Path to configuration file
      # ============================================
      - name: BEYLA_CONFIG_PATH
        value: /config/beyla-config.yml


      # ============================================
      # Kubernetes metadata
      # ============================================
```

```
        - name: BEYLA_KUBE_METADATA_ENABLE
          value: "true"

        # ============================================
        # REMOVE ANY NETWORK-RELATED ENV VARS
        # These were causing Beyla to run in network-only mode
        # ============================================
        # DO NOT SET: BEYLA_NETWORK_METRICS
        # DO NOT SET: BEYLA_NETWORK_ENABLE

        # ============================================
        # OpenTelemetry endpoint (optional)
        # Only set if you have an OTEL collector
        # ============================================
        # - name: OTEL_EXPORTER_OTLP_ENDPOINT
        #   value: "http://your-otel-collector:4318"

        # ============================================
        # For Grafana Cloud (optional)
        # ============================================
        # - name: OTEL_EXPORTER_OTLP_ENDPOINT
        #   value: "https://otlp-gateway-prod-us-central-0.grafana.net/otlp"
        # - name: OTEL_EXPORTER_OTLP_HEADERS
        #   value: "Authorization=Basic <base64-encoded-user:token>"

        # ============================================
        # CRITICAL: Explicitly enable application metrics
        # ============================================
        - name: BEYLA_OTEL_METRICS_FEATURES
          value:
"application,application_process,application_span,application_service_graph"

        # ============================================
```

```
        # Log level for debugging
        # ==========================================
        - name: BEYLA_LOG_LEVEL
          value: "INFO"

        # ==========================================
        # Service discovery - CRITICAL
        # ==========================================
        - name: BEYLA_OPEN_PORT
          value: "8080"

        volumeMounts:
        - name: config
          mountPath: /config

        # Port for Prometheus metrics endpoint
        ports:
        - containerPort: 9090
          name: prometheus
          protocol: TCP

        resources:
          requests:
            memory: "200Mi"
            cpu: "100m"
          limits:
            memory: "500Mi"
            cpu: "500m"

      volumes:
      - name: config
        configMap:
```

```
          name: beyla-config


      # Tolerate master node taints
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/control-plane
        operator: Exists
---
# Service to expose Prometheus metrics endpoint
apiVersion: v1
kind: Service
metadata:
  name: beyla-metrics
  namespace: beyla
  labels:
    instrumentation: beyla
spec:
  selector:
    instrumentation: beyla
  ports:
  - port: 9090
    targetPort: 9090
    name: prometheus
  type: ClusterIP


Apply: kubectl apply -f 04-beyla-daemonset.yaml
```

Step-6: Deploy Prometheus

```
05-prometheus-setup.yaml
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: beyla
data:
  prometheus.yml: |
    global:
      scrape_interval: 15s
      evaluation_interval: 15s

    scrape_configs:
      # Scrape Beyla metrics using service discovery
      - job_name: 'beyla'
        honor_labels: true
        kubernetes_sd_configs:
          - role: pod
            namespaces:
              names:
                - beyla
        relabel_configs:
          # Only scrape pods with instrumentation=beyla label
          - source_labels: [__meta_kubernetes_pod_label_instrumentation]
            action: keep
            regex: beyla
          # Use pod IP and port 9090
          - source_labels: [__meta_kubernetes_pod_ip]
            action: replace
            target_label: __address__
            replacement: ${1}:9090
```

```
          # Add pod name as label
          - source_labels: [__meta_kubernetes_pod_name]
            action: replace
            target_label: pod
          # Add node name as label
          - source_labels: [__meta_kubernetes_pod_node_name]
            action: replace
            target_label: node

      # Alternative: Direct service endpoint scrape
      - job_name: 'beyla-service'
        honor_labels: true
        static_configs:
          - targets: ['beyla-metrics:9090']
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
  namespace: beyla
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
  template:
    metadata:
      labels:
        app: prometheus
    spec:
      serviceAccountName: prometheus
      containers:
```

```
      - name: prometheus
        image: prom/prometheus:latest
        args:
          - '--config.file=/etc/prometheus/prometheus.yml'
          - '--storage.tsdb.path=/prometheus'
          - '--web.console.libraries=/usr/share/prometheus/console_libraries'
          - '--web.console.templates=/usr/share/prometheus/consoles'
          - '--web.enable-lifecycle'
        ports:
        - containerPort: 9090
          name: web
        volumeMounts:
        - name: config
          mountPath: /etc/prometheus
        - name: storage
          mountPath: /prometheus
        resources:
          requests:
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "2Gi"
            cpu: "1000m"
      volumes:
      - name: config
        configMap:
          name: prometheus-config
      - name: storage
        emptyDir: {}
---
apiVersion: v1
kind: ServiceAccount
```

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

```
    name: prometheus
    namespace: beyla
---
apiVersion: v1
kind: Service
metadata:
  name: prometheus
  namespace: beyla
spec:
  selector:
    app: prometheus
  ports:
  - port: 9090
    targetPort: 9090
    nodePort: 30090
    name: web
  type: NodePort

Apply: kubectl apply -f 05-prometheus-setup.yaml
```

Step-7: Deploy Grafana

```
06-grafana-setup.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grafana
  namespace: beyla
spec:
  replicas: 1
  selector:
```

```
  matchLabels:
    app: grafana
template:
  metadata:
    labels:
      app: grafana
  spec:
    containers:
    - name: grafana
      image: grafana/grafana:latest
      ports:
      - containerPort: 3000
        name: web
      env:
      - name: GF_SECURITY_ADMIN_PASSWORD
        value: "admin"  # Change this!
      - name: GF_INSTALL_PLUGINS
        value: "grafana-piechart-panel"
      volumeMounts:
      - name: storage
        mountPath: /var/lib/grafana
      - name: datasources
        mountPath: /etc/grafana/provisioning/datasources
      resources:
        requests:
          memory: "256Mi"
          cpu: "100m"
        limits:
          memory: "512Mi"
          cpu: "500m"
    volumes:
    - name: storage
```

```
        emptyDir: {}
      - name: datasources
        configMap:
          name: grafana-datasources
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: grafana-datasources
  namespace: beyla
data:
  datasource.yml: |
    apiVersion: 1
    datasources:
    - name: Prometheus
      type: prometheus
      access: proxy
      url: http://prometheus:9090
      isDefault: true
      editable: true
---
apiVersion: v1
kind: Service
metadata:
  name: grafana
  namespace: beyla
spec:
  selector:
    app: grafana
  ports:
  - port: 3000
    targetPort: 3000
```

```
    nodePort: 30000
  type: NodePort


Apply: kubectl apply -f 06-grafana-setup.yaml
```

VERIFICATION

) beyla namespace pods

```
pranavpp37@k8s-master:~/beyla$ kubectl get pods -n beyla -o wide
NAME                          READY   STATUS    RESTARTS       AGE   IP               NODE          NOMINATED NODE   READINESS GATE
S
beyla-cmwm2                   1/1     Running   1 (178m ago)   8d    192.168.230.21   k8s-worker-1  <none>           <none>
beyla-dfgv8                   1/1     Running   1 (178m ago)   8d    192.168.235.204  k8s-master    <none>           <none>
beyla-vrv4f                   1/1     Running   1 (178m ago)   8d    192.168.140.20   k8s-worker-2  <none>           <none>
grafana-dfd5c5c6b-crn8s       1/1     Running   1 (178m ago)   8d    192.168.230.18   k8s-worker-1  <none>           <none>
prometheus-5b5d89b996-g5z6k   1/1     Running   1 (178m ago)   8d    192.168.230.19   k8s-worker-1  <none>           <none>
pranavpp37@k8s-master:~/beyla$
```

) default (java-app pods)

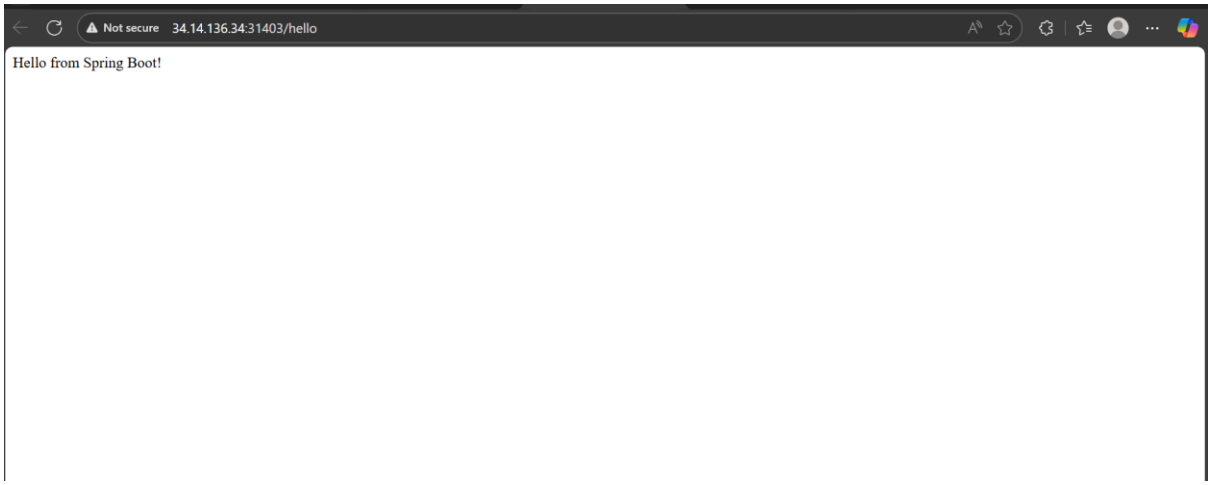
```
spring-boot-app-5b66f9659-ljmvf   1/1     Running   1 (178m ago)   8d    192.168.230.20   k8s-worker-1  <none>           <none>
```

ENDPOINTS

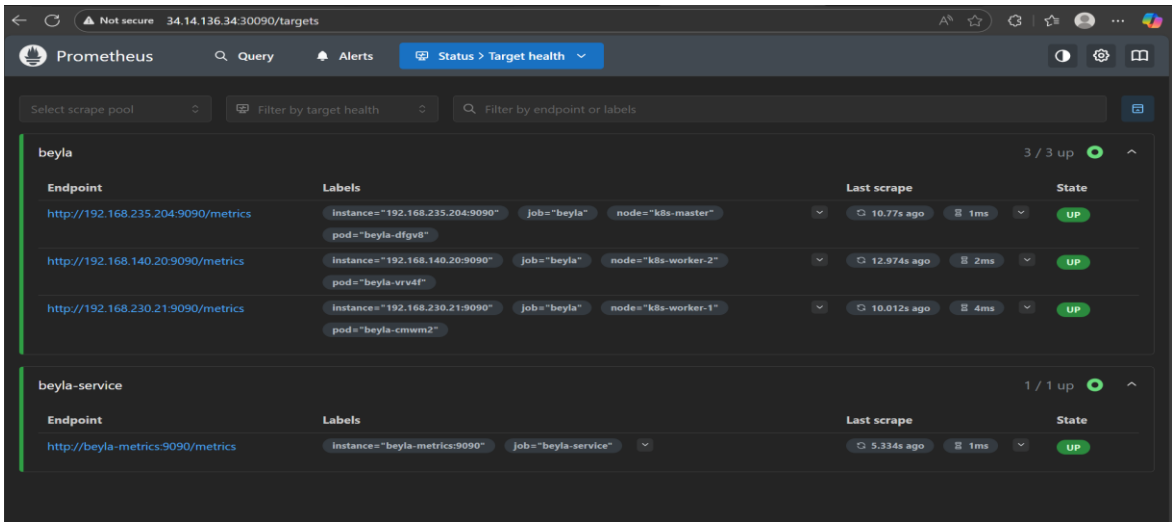
Service / App	Namespace	NodePort / Port	Access URL	Notes
Spring Boot App	default	8080 (ClusterIP)	http://<Node-IP>:8080/hello via port-forward	Use port-forward since NodePort not defined
Prometheus	beyla	30090	http://34.14.136.34:30090	Scrapes Beyla metrics at /metrics
Grafana	beyla	30000	http://34.14.136.34:30000	Default login: admin/admin
Beyla metrics	beyla	9090 (pod)	http://<pod-ip>:9090/metrics	Access via port-forward

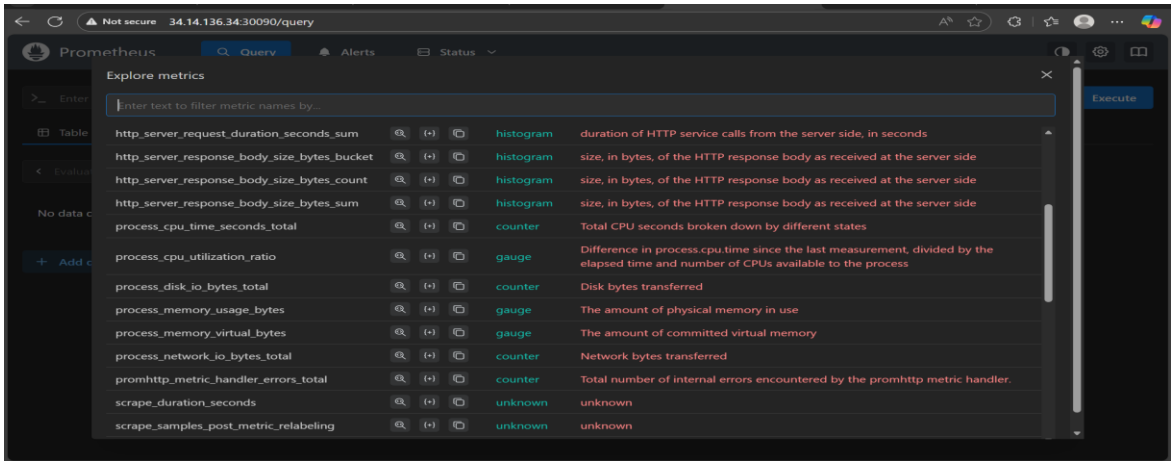
SCREENSHOTS

) **Java App**

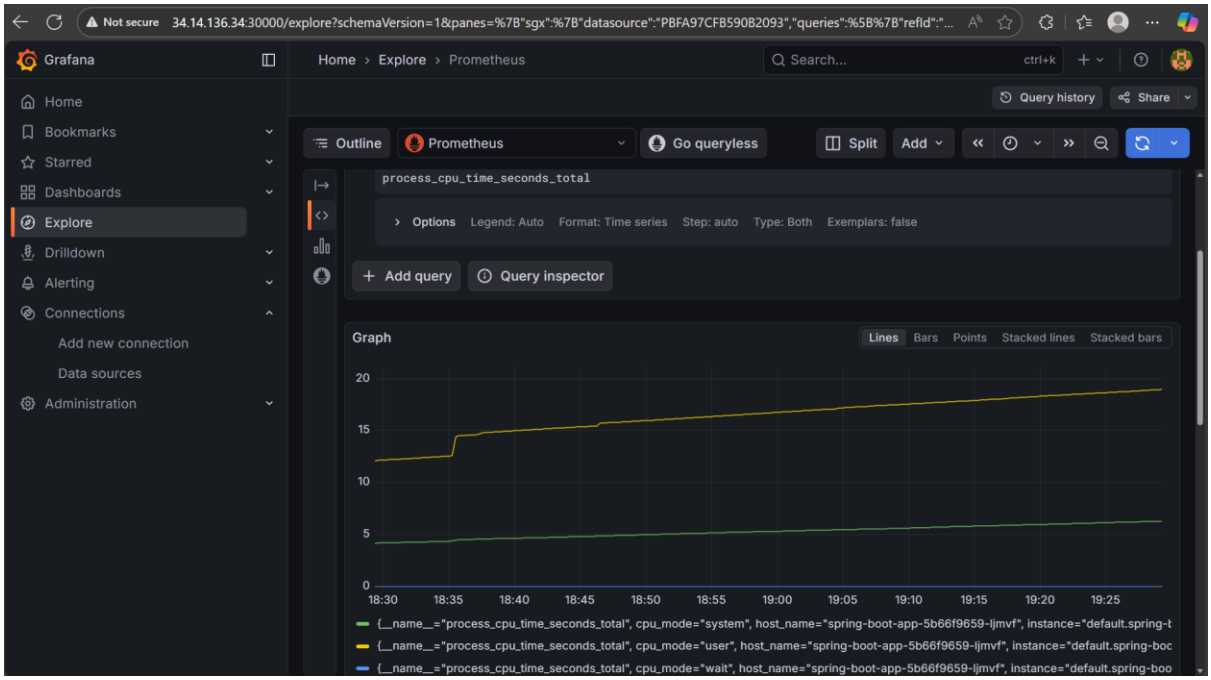


) **Prometheus**





) Grafana


Not secure
34.14.136.34:30000/explore?schemaVersion=1&panes=%7B"sgx":%7B"datasource":"PBFA97CFB590B2093","queries":%5B%7B"refId":"...
Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
Alerting
Connections
Add new connection
Data sources
Administration
Home › Explore › Prometheus
Search...
ctrl+k
Query history
Share
Outline
Prometheus
Go queryless
Split
Add
process_cpu_time_seconds_total
Options
Legend: Auto
Format: Time series
Step: auto
Type: Both
Exemplars: false
Add query
Query inspector
Graph
Lines
Bars
Points
Stacked lines
Stacked bars
20
15
10
5
0
18:30
18:35
18:40
18:45
18:50
18:55
19:00
19:05
19:10
19:15
19:20
19:25
{__name__="process_cpu_time_seconds_total", cpu_mode="system", host_name="spring-boot-app-5b66f9659-ljmvf", instance="default.spring-t
{__name__="process_cpu_time_seconds_total", cpu_mode="user", host_name="spring-boot-app-5b66f9659-ljmvf", instance="default.spring-boc
{__name__="process_cpu_time_seconds_total", cpu_mode="wait", host_name="spring-boot-app-5b66f9659-ljmvf", instance="default.spring-boo