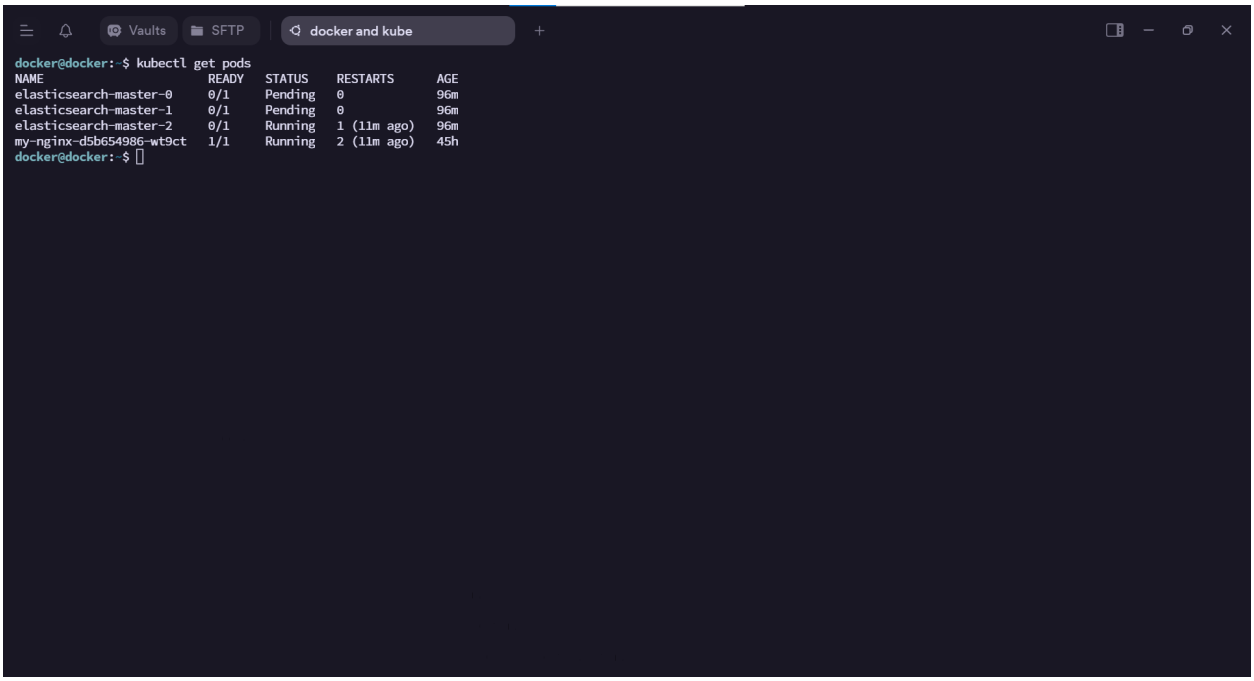
Helm and database Task




```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  annotations:
    esMajorVersion: "8"
    meta.helm.sh/release-name: elasticsearch
    meta.helm.sh/release-namespace: default
  creationTimestamp: "2024-07-14T14:13:20Z"
  generation: 1
  labels:
    app: elasticsearch-master
    app.kubernetes.io/managed-by: Helm
    chart: elasticsearch
    heritage: Helm
    release: elasticsearch
```

```
  name: elasticsearch-master
  namespace: default
  resourceVersion: "9665"
  uid: fa0e60df-4678-4a44-a948-8ff3112dcbb7
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Retain
  podManagementPolicy: Parallel
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: elasticsearch-master
  serviceName: elasticsearch-master-headless
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: elasticsearch-master
        chart: elasticsearch
        release: elasticsearch
      name: elasticsearch-master
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - elasticsearch-master
            topologyKey: kubernetes.io/hostname
      automountServiceAccountToken: true
      containers:
```

```
      - env:
        - name: node.name
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: metadata.name
        - name: cluster.initial_master_nodes
          value: elasticsearch-master-0,elasticsearch-master-
1,elasticsearch-master-2,
        - name: node.roles
          value: master,data,data_content,data_hot,data_warm,
data_cold,ingest,ml,remote_cluster_client,transform,
        - name: discovery.seed_hosts
          value: elasticsearch-master-headless
        - name: cluster.name
          value: elasticsearch
        - name: network.host
          value: 0.0.0.0
        - name: ELASTIC_PASSWORD
          valueFrom:
            secretKeyRef:
              key: password
              name: elasticsearch-master-credentials
        - name: xpack.security.enabled
          value: "true"
        - name: xpack.security.transport.ssl.enabled
          value: "true"
        - name: xpack.security.http.ssl.enabled
          value: "true"
        - name: xpack.security.transport.ssl.verification_mod
e
          value: certificate
        - name: xpack.security.transport.ssl.key
          value: /usr/share/elasticsearch/config/certs/tls.ke
y
        - name: xpack.security.transport.ssl.certificate
```

```
          value: /usr/share/elasticsearch/config/certs/tls.cr
t
        - name: xpack.security.transport.ssl.certificate_auth
orities
          value: /usr/share/elasticsearch/config/certs/ca.crt
        - name: xpack.security.http.ssl.key
          value: /usr/share/elasticsearch/config/certs/tls.ke
y
        - name: xpack.security.http.ssl.certificate
          value: /usr/share/elasticsearch/config/certs/tls.cr
t
        - name: xpack.security.http.ssl.certificate_authoriti
es
          value: /usr/share/elasticsearch/config/certs/ca.crt
        image: docker.elastic.co/elasticsearch/elasticsearch:
8.5.1
        imagePullPolicy: IfNotPresent
        name: elasticsearch
        ports:
        - containerPort: 9200
          name: http
          protocol: TCP
        - containerPort: 9300
          name: transport
          protocol: TCP
        readinessProbe:
          exec:
            command:
            - bash
            - -c
            - |
              set -e

              # Exit if ELASTIC_PASSWORD in unset
              if [ -z "${ELASTIC_PASSWORD}" ]; then
                echo "ELASTIC_PASSWORD variable is missing, e
```

```
xiting"
                exit 1
              fi

              # If the node is starting up wait for the clust
er to be ready (request params: "wait_for_status=green&timeou
t=1s" )
              # Once it has started only check that the node
itself is responding
              START_FILE=/tmp/.es_start_file

              # Disable nss cache to avoid filling dentry cac
he when calling curl
              # This is required with Elasticsearch Docker us
ing nss < 3.52
              export NSS_SDB_USE_CACHE=no

              http () {
                local path="${1}"
                local args="${2}"
                set -- -XGET -s

                if [ "$args" != "" ]; then
                  set -- "$@" $args
                fi

                set -- "$@" -u "elastic:${ELASTIC_PASSWORD}"

                curl --output /dev/null -k "$@" "https://127.
0.0.1:9200${path}"
              }

              if [ -f "${START_FILE}" ]; then
                echo 'Elasticsearch is already running, lets
check the node is healthy'
                HTTP_CODE=$(http "/" "-w %{http_code}")
```

```
                RC=$?
                if [[ ${RC} -ne 0 ]]; then
                  echo "curl --output /dev/null -k -XGET -s -
w '%{http_code}' \${BASIC_AUTH} https://127.0.0.1:9200/ faile
d with RC ${RC}"
                  exit ${RC}
                fi
                # ready if HTTP code 200, 503 is tolerable if
ES version is 6.x
                if [[ ${HTTP_CODE} == "200" ]]; then
                  exit 0
                elif [[ ${HTTP_CODE} == "503" && "8" == "6"
]]; then
                  exit 0
                else
                  echo "curl --output /dev/null -k -XGET -s -
w '%{http_code}' \${BASIC_AUTH} https://127.0.0.1:9200/ faile
d with HTTP code ${HTTP_CODE}"
                  exit 1
                fi

              else
                echo 'Waiting for elasticsearch cluster to be
come ready (request params: "wait_for_status=green&timeout=1
s" )'
                if http "/_cluster/health?wait_for_status=gre
en&timeout=1s" "--fail" ; then
                  touch ${START_FILE}
                  exit 0
                else
                  echo 'Cluster is not yet ready (request par
ams: "wait_for_status=green&timeout=1s" )'
                  exit 1
                fi
              fi
          failureThreshold: 3
```

```
          initialDelaySeconds: 10
          periodSeconds: 10
          successThreshold: 3
          timeoutSeconds: 5
        resources:
          limits:
            cpu: "1"
            memory: 2Gi
          requests:
            cpu: "1"
            memory: 2Gi
        securityContext:
          capabilities:
            drop:
            - ALL
          runAsNonRoot: true
          runAsUser: 1000
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /usr/share/elasticsearch/data
          name: elasticsearch-master
        - mountPath: /usr/share/elasticsearch/config/certs
          name: elasticsearch-certs
          readOnly: true
      dnsPolicy: ClusterFirst
      enableServiceLinks: true
      initContainers:
      - command:
        - sysctl
        - -w
        - vm.max_map_count=262144
        image: docker.elastic.co/elasticsearch/elasticsearch:
8.5.1
        imagePullPolicy: IfNotPresent
        name: configure-sysctl
```

```
        resources: {}
        securityContext:
          privileged: true
          runAsUser: 0
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext:
        fsGroup: 1000
        runAsUser: 1000
      terminationGracePeriodSeconds: 120
      volumes:
      - name: elasticsearch-certs
        secret:
          defaultMode: 420
          secretName: elasticsearch-master-certs
  updateStrategy:
    type: RollingUpdate
  volumeClaimTemplates:
  - apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      creationTimestamp: null
      name: elasticsearch-master
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 30Gi
      volumeMode: Filesystem
    status:
      phase: Pending
status:
  availableReplicas: 0
```

```
  collisionCount: 0
  currentReplicas: 3
  currentRevision: elasticsearch-master-d469cccdf
  observedGeneration: 1
  replicas: 3
  updateRevision: elasticsearch-master-d469cccdf
  updatedReplicas: 3
```

1- Explain how Prometheus work.

```
Answer: Prometheus is widly used an open-source monitoring an
d alerting tool designed
It is known for its reliability and scalability.Its main job
to scap data as metrics from to useres configured endpoints.
Prometheus used time series databses for storing data.
Some main components of prometheus Are :

1- Prometheus Server
2- Alertmanager
3- Exporters
4- Pushgateway
5- PromQL

Prometheus Server: Its the main server components store data
and scraping endpoints.
Alertmanager: Its responsible for handle all alerts generate
by Prometheus
Exporters: Its basically exposes metrics and applications.
Pushgateway: pushes metrics to Prometheus .
PromQL: Its a Query language responsible for  retrieve and ma
nipulate time-series data.
```

2- How do you create custom Prometheus alerts and alerting rules for Kubernetes monitoring? Provide an example alert rule and its configuration.

```
groups:
  - name: example yaml file for task
    rules:
      - alert: HighPodCPUUsage # its the name of alert
        expr: sum(rate(container_cpu_usage_seconds_total[1
m])) by (pod) > 0.5 # its a
        query of PromQl language to measure
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "High CPU usage detected for pod {{ $label
s.pod }}"
          description: "Pod {{ $labels.pod }} is using more t
han 50% CPU for the last 5 minutes."
```

3- What is the Prometheus query you can use in Granfana to properly show usage trend of an application metric that is a counter?

```
rate(metric_name[interval])
For example:
rate(http_requests_total[5m]) # we take the time intervel of
5 mins here

# The rate() function is widly used for most use cases as it
gives you a per-second rate, which is for visualizing trends
over time period .
```